# Running Apache httpd on a mobile phone

Johan Wikman

Principal Research Engineer

Nokia Research Center

johan.wikman@nokia.com

ApacheCon 2007

May 1-4 2007, Amsterdam, The Netherlands



NOKIA
Connecting People

# Topics

- Project's History
- Symbian and S60
- Apache on S60
- Connectivity
- Mobsites

# History

- "Wouldn't it be cool to run a globally accessible web server on a mobile phone?"
- Apache httpd?
  - We wanted a *real* web server.
  - Knew that it would generate interest.
- Wanted to be able to use it in current operator networks.
  - No specific assumptions on the Network.

# Symbian/S60

- Symbian is quite different from pretty much everything else.
  - C++ API.
  - Most system services are provided via asynchronous interfaces.
    - You initiate an operation, after a while you are called back.
  - Threads supported, but primarily you should use a cooperative multitasking framework with so called *Active Objects* and *Scheduler*.
- Microkernelish architecture.
  - Scheduler and memory management in kernel; networking and file systems provided as user-side servers.
  - Device drivers run in kernel as loadable/unloadable plugins.
  - Message-passing framework provided.
  - Straightforward to create your own server.
- S60 is basically a UI and Look&Feel layer on top of Symbian.
  - Contains other functionality as well.





NOKIA
Connecting People

# Libc

- Symbian has for many years had a “standard” C library.
  - Quite limited.
  - Buggy
- First port was implemented on top of it.
  - Porting *Apache Portable Runtime* directly on top of Symbian native constructs would have been prohibitively time-consuming.
- In the spring of 2007, *Open C* was introduced.
  - A significant share of the functions available in a typical “Linux” environment are now available.
  - libc (47%), libpthread (60%), libm (42%), libdl (100%), libz (100%), libcrypt (100%), libcrypto (77%), libglib (77%), libssl (86%)
- Work in progress on taking it into use.





NOKIA
Connecting People

# Apache on Symbian/S60

# Porting httpd to Symbian/S60

- Major issues:
  - Build environment
  - Runtime Structure
  - Exporting Data
  - Importing Modules
  - Security Model

- All of these are either completely different or not supported on Symbian.

# Building for Symbian/S60

# Building for Symbian/S60

- Build environment primarily on Windows.
  - Some experimental build environments on Linux exist as well.
- Emulator but no simulator.
  - You can't run target software on PC, but have to build a separate version.
  - Tricky with peripherals inherent on current smart phones.
  - Rather recently, functioning on-device debugging.
- No cross-compilation environment.
  - You can't run **`configure`**.
- Not a normal build environment.
  - No **`autoconf`**, no **`automake`**, no regular **`makefiles`**.

- So all build files must be created manually.
  - Symbian's **`inf`**-files and **`mmp`**-files (functionality wise the equivalent of makefiles) must be created manually.
  - Header files that typically are generated automatically from **`.in`**-files must be created manually.

- So, quite different from your regular Linux environment.
  - But the situation is not that different from what you have to do on Windows.



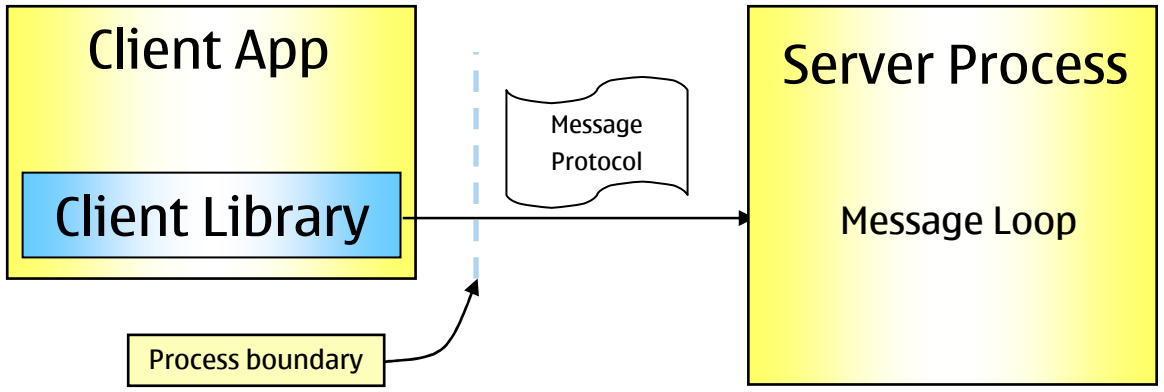


NOKIA
Connecting People

# Runtime Structure

# Runtime Structure

- Anything that should be able to run in the background, independently of a UI, must be turned into a Symbian server.
  - Server process.
  - Client library.
  - Message protocol.
- Client library provides a regular C++ API, using which an application can interact with the server.
  - Synchronously or asynchronously.
- Behind the scenes, the library implements the API by sending messages to the Server.



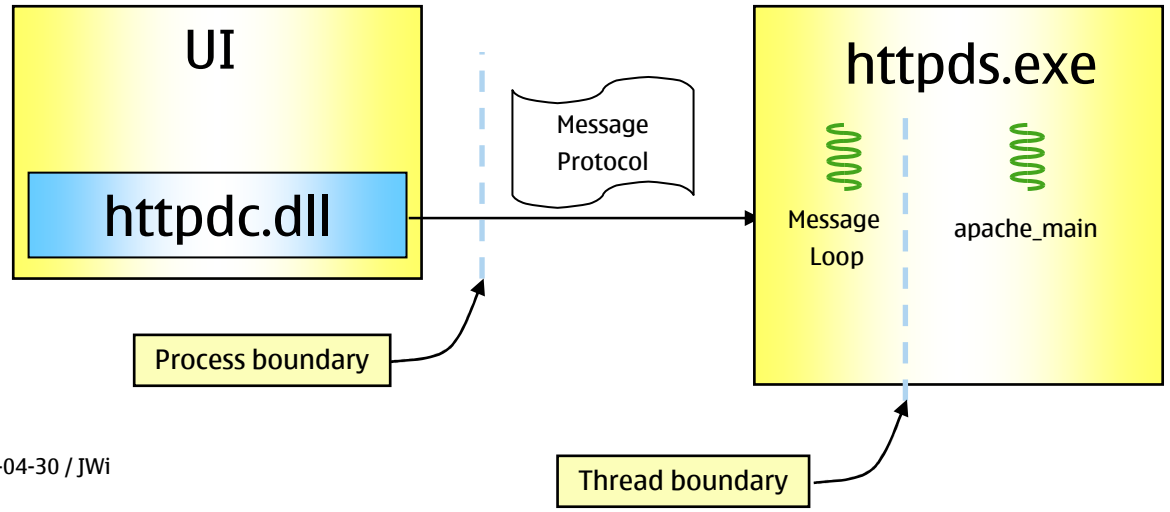



NOKIA
Connecting People

# Httpd as a Symbian Server

- Not trivial to turn Apache httpd directly into a Symbian server.
  - Extensive modifications would be needed.

- Instead, httpd is run in a separate thread inside the Symbian server.
  - From the outside, the process appears as a regular Symbian server.
  - From *httpd*'s point of view, the environment is what it usually is.



NOKIA
Connecting People

# Running

- Some client app creates an instance of the class representing the web-server.

```
RHttpd hHttpd;
hHttpd.Connect();
```

- This starts the server process.
- However, httpd itself is not started before `StartServer()` is called.

```
hHttpd.StartServer();
```

- At that point a secondary thread is created in which httpd's main is called.

```
#if defined(__SYMBIAN32__)
int apache_main(int argc, const char * const argv[])
#else
int main(int argc, const char * const argv[])
#endif
```

# Exporting Data

# Exporting Data

- The actual functionality of Apache httpd is provided by the shared libraries *libhttpd*, *libapr* and *libaprutil*.
- Partly these (and any imported modules) communicate using global variables.

- In operating system versions prior to Symbian 9.1, DLLs could **not** contain static data.
  - No static data => no data to export => not possible to use DLLs.

- Executables could (of course) contain static data.

- Consequently, *libhttpd*, *libapr and libaprutil*, and all modules, were built as static libraries that were linked into one big monolith.
  - Worked, but rather cumbersome to include/exclude a particular module.

# Exporting Data on Symbian 9.1

- From version 9.1 onwards, Symbian supports static data in DLLs.
- However, exporting/importing data *directly* cannot be done.
- But exporting/importing functions has always worked.

- So, by turning all exported/imported data into automatically dereferenced function calls, the limitation can be worked around.

# Declaring Exported Data

- To export a variable, you declare it with a module specific prefix.

```
APU_DECLARE_DATA extern apr_pool_t *apr_hook_global_pool;
```

- On some platforms, the prefix is defined differently depending on whether or not it is the module defining that data that is being compiled.
  - That is, whether the symbol should be exported or imported.

- **`APU_DECLARE_EXPORT`** defines which case it is.
  - Defined => it's the module itself that is being compiled.
  - Undefined => it's some code that uses the symbol that is being compiled.

# Declaring Exported Data on Symbian

```
#if defined(APU_DECLARE_EXPORT)
APU_DECLARE_DATA extern apr_pool_t *apr_hook_global_pool;
#else
#define apr_hook_global_pool (*_apr_hook_global_pool())
#endif

APU_DECLARE(apr_pool_t **) _apr_hook_global_pool();
```

- If it's the module defining the variable that is being compiled, then we just declare the variable.
  - Within the module the variable can be accessed directly.
- Otherwise, we define a macro that expands into a dereferenced function call.

- In both cases we declare a function that returns the address of the variable.

# Defining Exported Data

- The definition looks the same on Symbian as elsewhere.

```
APU_DECLARE_DATA apr_pool_t *apr_hook_global_pool = NULL;
```

- In addition we have to add the function returning the address of the variable.

```
APU_DECLARE(apr_pool_t **) _apr_hook_global_pool()
{
    return &apr_hook_global_pool;
}
```

# Easy to make operating system neutral.

- Declaration

```
APU_DECLARE_EXPORTED_DATA(apr_pool_t *, apr_hook_global_pool);
```

- Definition

```
APU_DEFINE_EXPORTED_DATA(apr_pool_t *, apr_hook_global_pool) = NULL;
```

- In all other cases but Symbian these can expand into their current definition.

# Loading Modules

# Loading modules.

- In general, most modules are not built into httpd, but loaded at runtime according to instructions in *httpd.conf*.
  - First the module is loaded into the process.
  - Then a symbol is looked up by name.

- On Symbian, symbols from a DLL can **not** be looked up by name, but only by ordinal.
  - So, something Symbian specific is needed.

- Requirements
  - No changes to how modules are loaded via httpd.conf.
  - Minimal changes to existing modules.

# Exporting Symbols

- Typically modules export just one or a few **`module`** structures that contain function pointers to non-exported functions.
  - Some modules – e.g. Dav – also exports functions.

- By exporting from each module a function that when given a name can return the address of the corresponding symbol, we can provide a way for looking up symbols by name.

```
apr_status_t ap_lookup_symbol(const char* name, void** symbol);
```

- Further, if this function is *always* exported using the same ordinal, we can make this transparent so that symbols, from httpd's point of view, can be loaded by name just as everywhere else.

- Easy to arrange by using the same exports definition file for all modules.
  - Modules exporting additional functions need a specific definition file.

# Importing Symbols

- A typical entry of `httpd.conf` might be

```
LoadModule access_module modules/mod_access.so
```

- On Symbian, all binaries must be installed to \Sys\Bin.
  - Enforced by the Symbian platform security.
- That is, meaningless to have a path.
  - But for compatibility with `httpd.conf` on other environments, a path is allowed and ignored.
- In Apache's `mod_so.c` and APR's Symbian specific `dso.cpp` knowledge about the arrangement is injected.
  - Function with the specific ordinal is looked up.
  - The function is called with the provided name ("access_module").
- Thereafter everything is just like on every other platform.

# Making it Simple

- Laborious, unwieldy and error prone to do it manually.
- Instead macros are defined in **`ap_config.h`** that makes it quite easy to manually export a symbol from a module.

- Typically only a single line must be added to a module to make it loadable on Symbian.
  - On all other environments it could expand into a NOP.

```
module AP_MODULE_DECLARE_DATA access_module =
...
};

AP_EXPORT_1_SYMBOL(access_module)
```

# AP_EXPORT_1_SYMBOL(access_module)

- Expands into:

```
static const ap_export_entry ap_export_entries[] = {
    { "access_module", &access_module }
};

static const int ap_export_entries_count =
    sizeof(ap_export_entries)/sizeof(ap_export_entries[0]);

EXPORT_C int ap_lookup_symbol(const char* name, void** symbol) {
    return ap_lookup_table_symbol(ap_export_entries,
                                  ap_export_entries_count,
                                  name, symbol);
}
```

- `ap_lookup_table_symbol()` is a helper that looks up the symbol from the passed array of structures.

# Platform Security

# Capabilities

- The security model of Symbian is quite different compared to that of other environments.

- Executables and DLLs have capabilities.
  - Different capabilities allow you to do different things.
  - Different capabilities require different kind of certificates.
    - Least amount of capabilities with a self-signed certificate.
    - Largest amount of capabilities with a vendor-approved certificate.
  - Fixed at built-time, cannot be changed at runtime.
- A DLL must have *at least* as many capabilities as the executable that loads them.
  - If a pre-built httpd is given a lot of capabilities, independent 3rd-party developers must have a certificate that grants as many capabilities.
  - If a pre-built httpd is given few capabilities, then independent 3rd-party developers cannot use more capabilities than that even if their certificate would allow them to.

# Data Caging

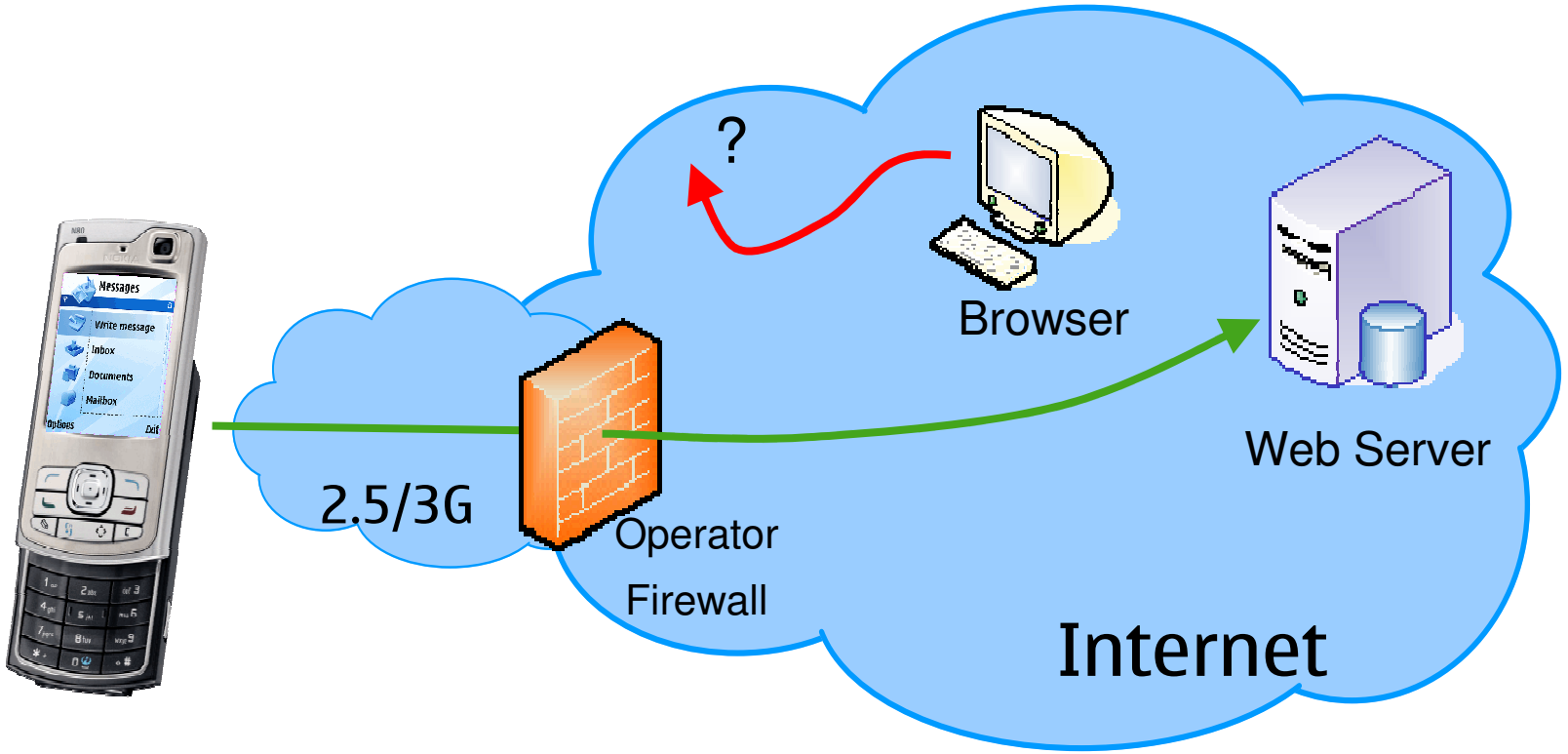
- Every application has a private directory of its own.
  - No other applications can access files in that directory.
- Safe place for all configuration files?
  - `httpd.conf`, `users`, etc.
- In principle yes, but how would you edit them in that case?
  - An editor with lots of capabilities.
  - First mounting the directory using WebDAV, then editing from another computer.

- For the time being, Apache's directories are in the public area where there is no protection of any kind.
  - Tricky issue, because what if some trojan would modify the configuration files and give full access to some perpetrator?
  - Read all your SMSs, see who you call and who calls you, find out your whereabouts and place calls to expensive service numbers.

# Connectivity

# The Basic Problem

- There are NATs/Firewalls between the terminal and the Internet.
- That is, the terminal does not have a name and you can't reach it.

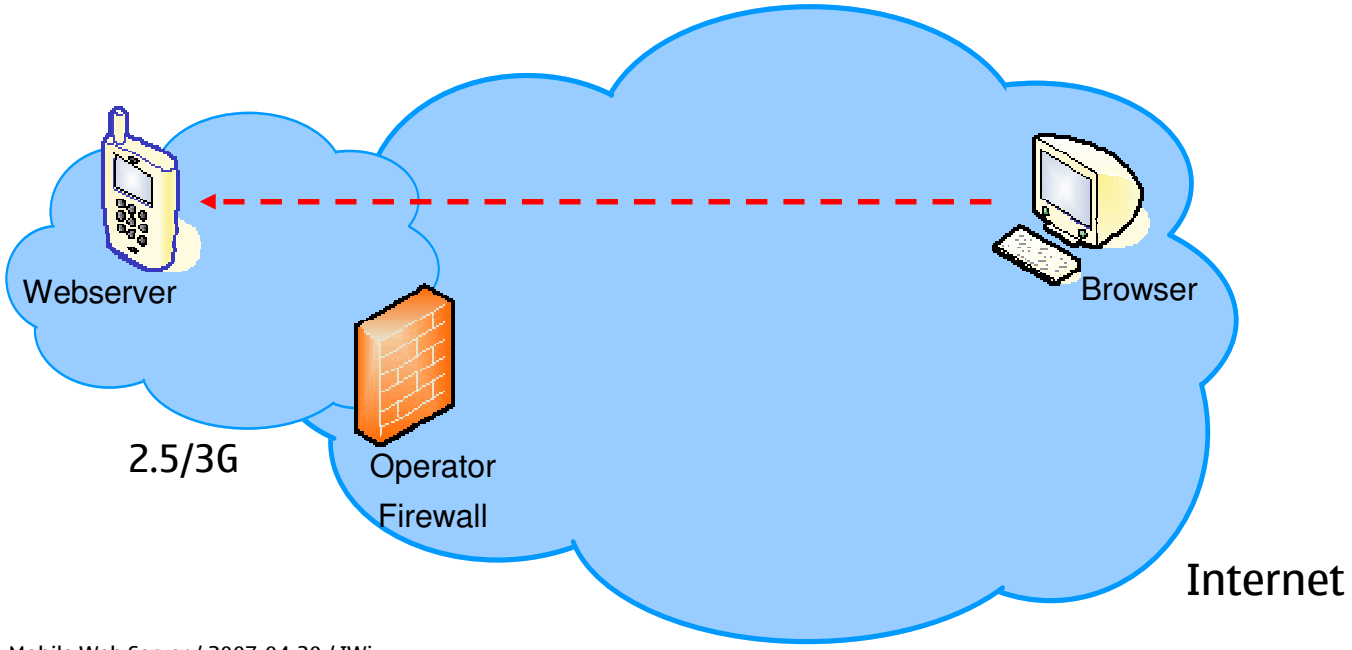


NOKIA
Connecting People

# The Illusion

- In the context of accessing a webserver, the mobile phone appears to have a domain name. That is, the web site on the phone has a URL.
- For the web browser, the web server and the person browsing, it appears as if there would be a direct connection.
- Millions of compatible clients.
- “The mobile phone becomes a full member of the web.”

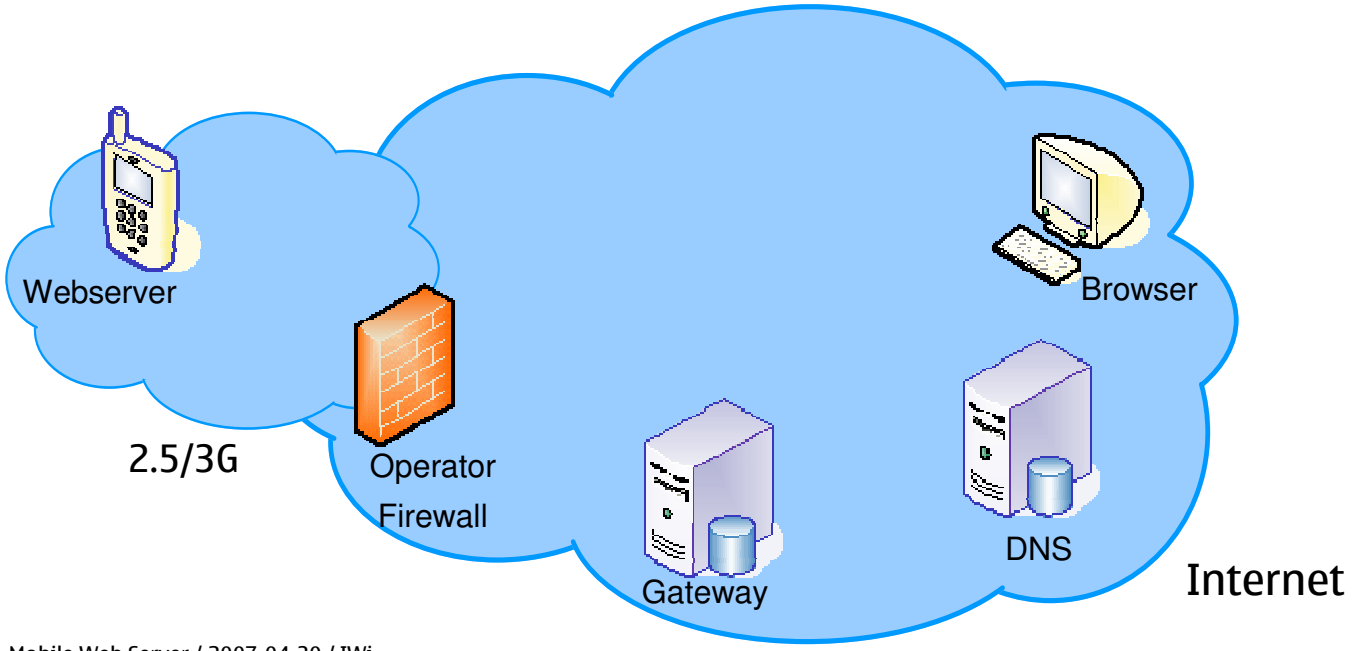





NOKIA
Connecting People

# The Gateway

- Runs on a computer with a static IP on the Internet.
  - In our case 212.213.221.246.
- A DNS mapping `*.at.openlaboratory.net -> 212.213.221.246`.
  - `john.at.openlaboratory.net -> 212.213.221.246`
  - `bob.at.openlaboratory.net -> 212.213.221.246`



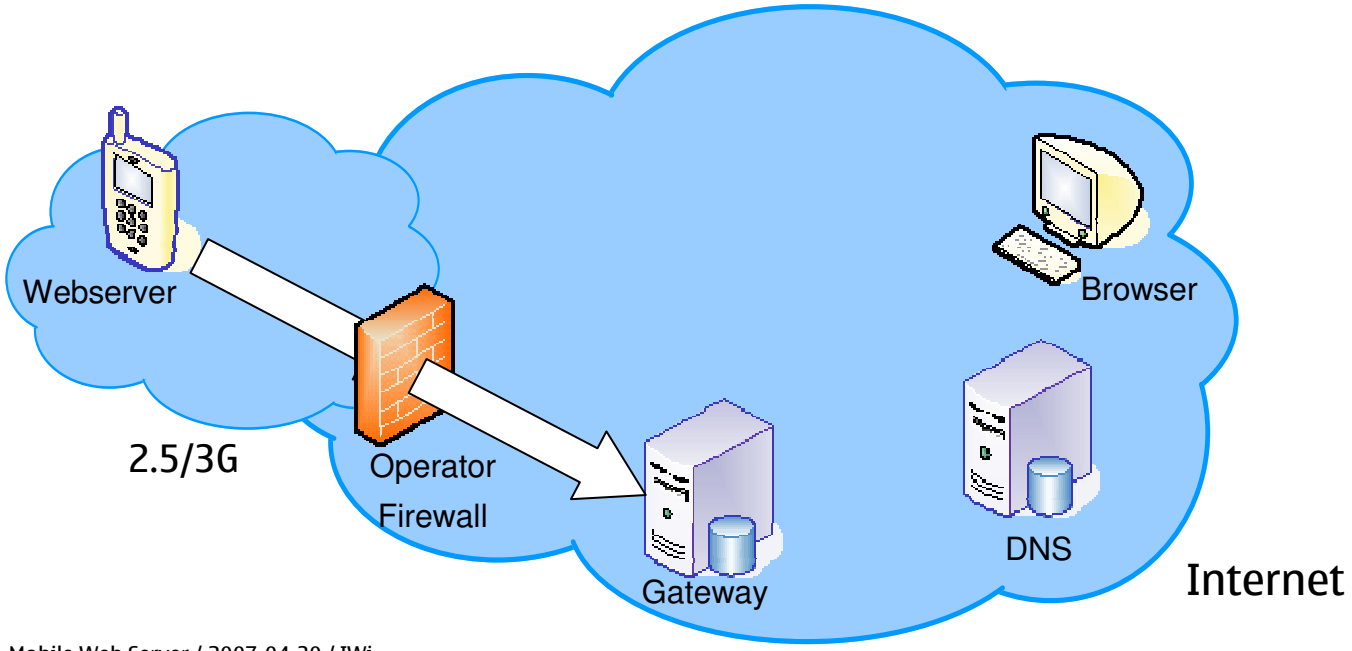



NOKIA
Connecting People

# Going Online

- The terminal connects to the gateway and identifies itself, e.g. at *john*.
  - An account must exist at the gateway.
  - The terminal is now known as http://john.at.openlaboratory.net.

- The connection exists for the entire time the terminal is online.

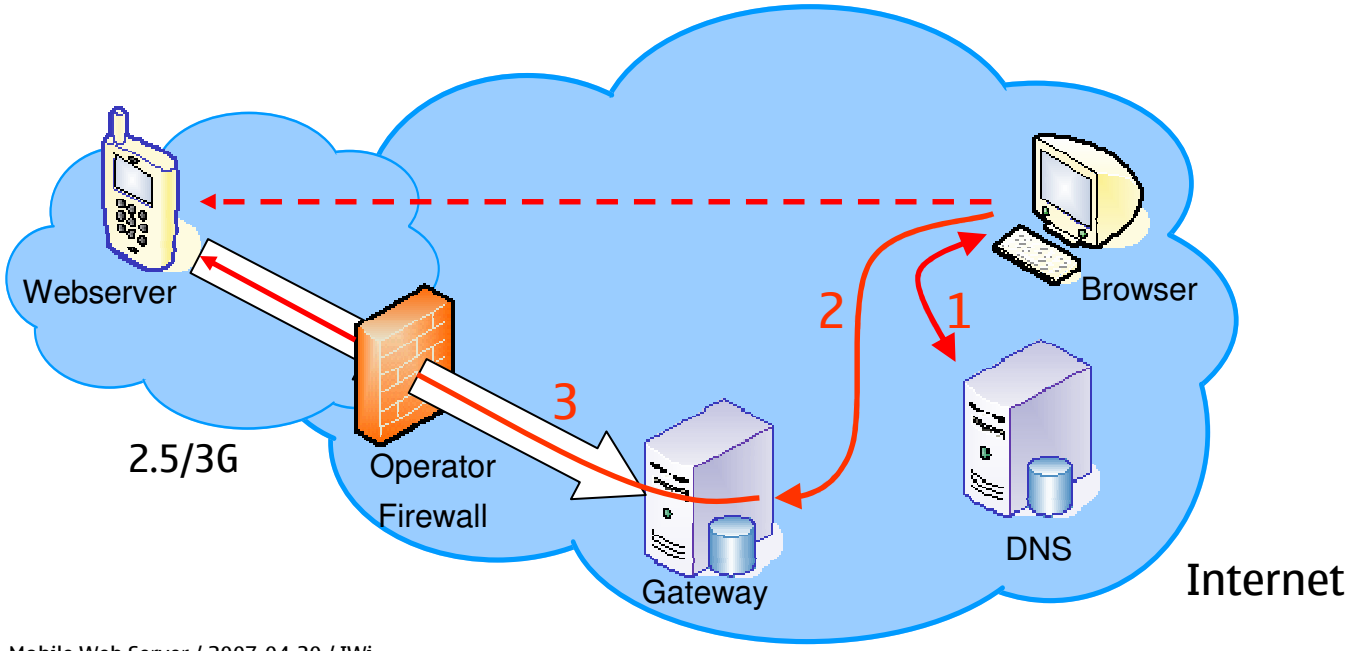





NOKIA
Connecting People

# Browsing

- URLs like http://*.at.openlaboratory.net resolves to the IP address of the gateway computer (1).
- HTTP request delivered to the gateway (2).
- From the HTTP request headers, the gateway can deduce who the request is intended for.
- The gateway then delivers the request of the connection opened by the terminal (3).



NOKIA
Connecting People

# Mobile Websites – Mobsites

# Personal Mobsite


Most recent photo taken.

Search your mobsite

Nice to know.

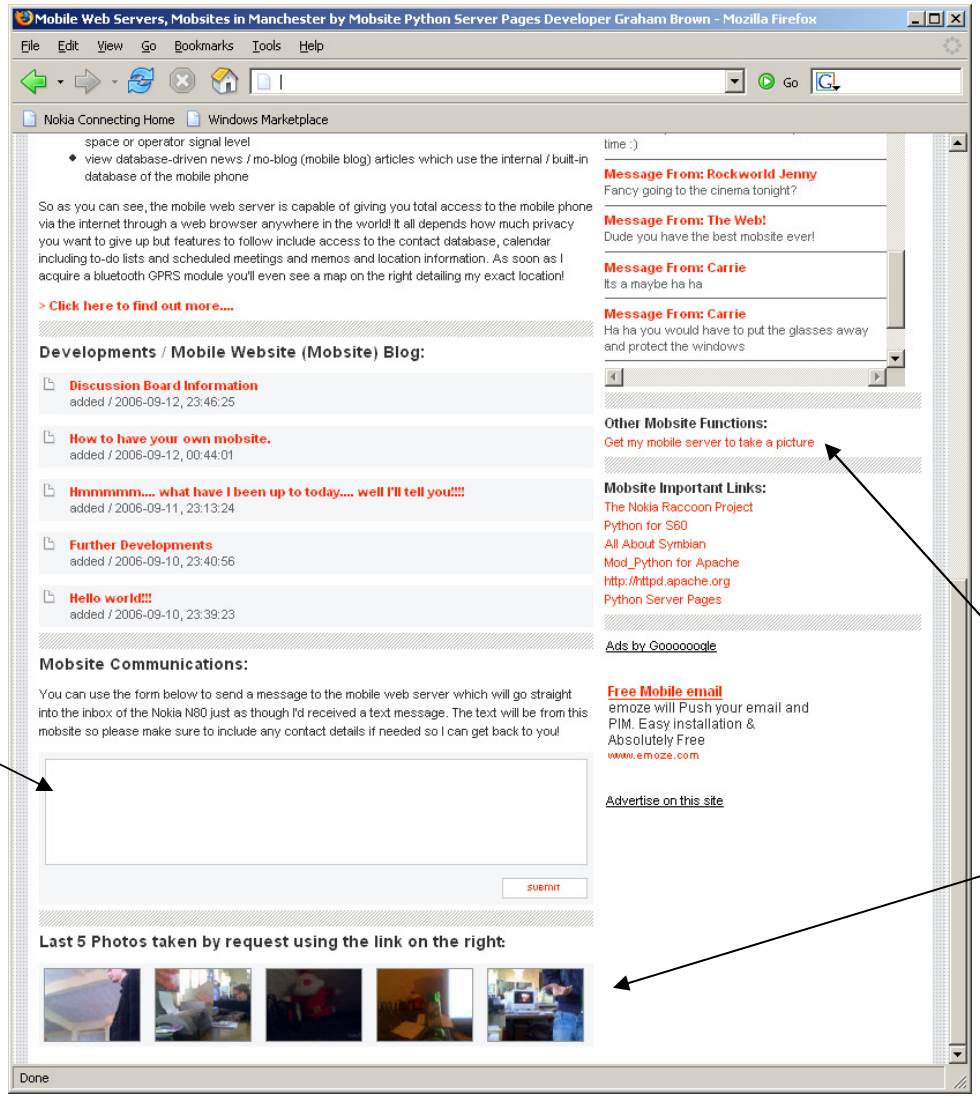
Access to core data:
- Text messages.
- Calendar
- Contacts

By courtesy of
graham@pixel8limited.com



NOKIA
Connecting People

# Personal Mobsite




Integration of the mobile phone and the web.

Web-form for sending messages directly to the inbox of the phone.

Interactively generated content.

By courtesy of
graham@pixel8limited.com

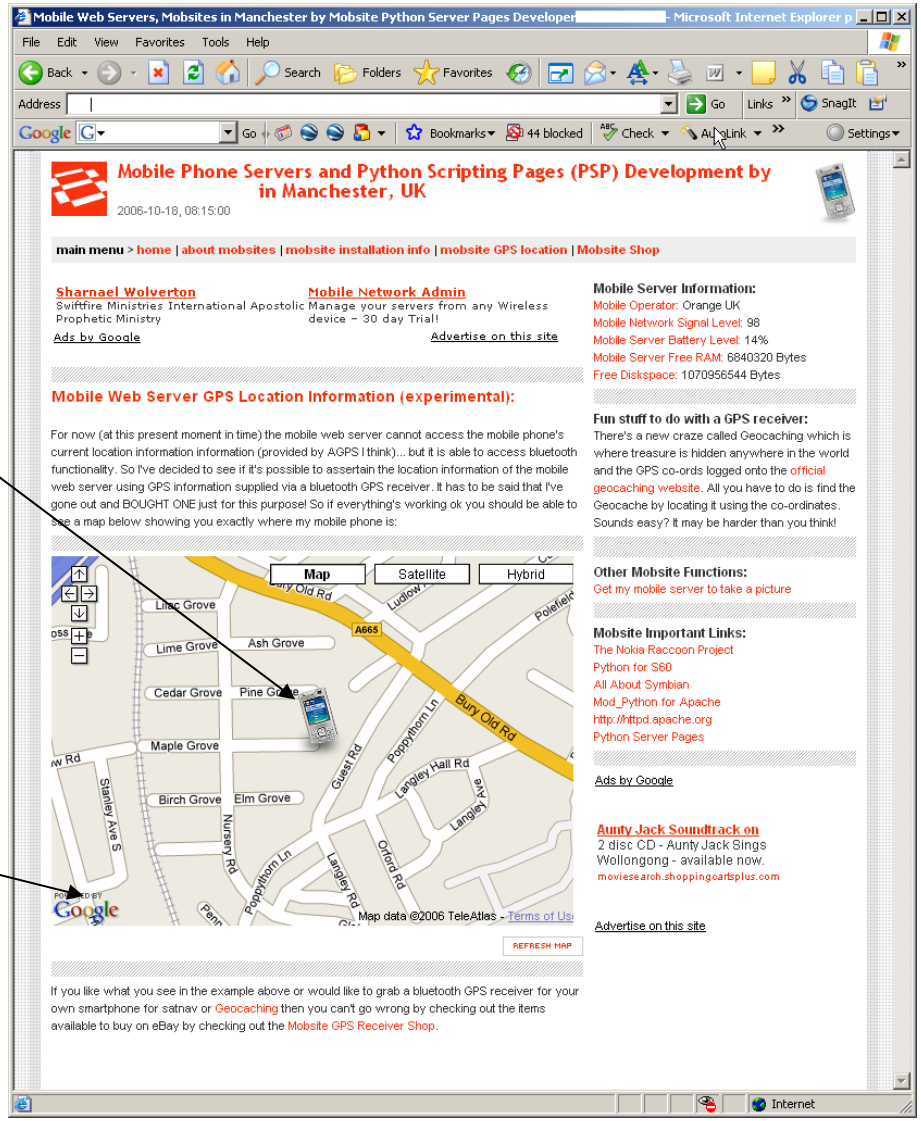


NOKIA
Connecting People

# Show Your Location

Phone knows your location.

Map from Google.




By courtesy of
graham@pixel8limited.com

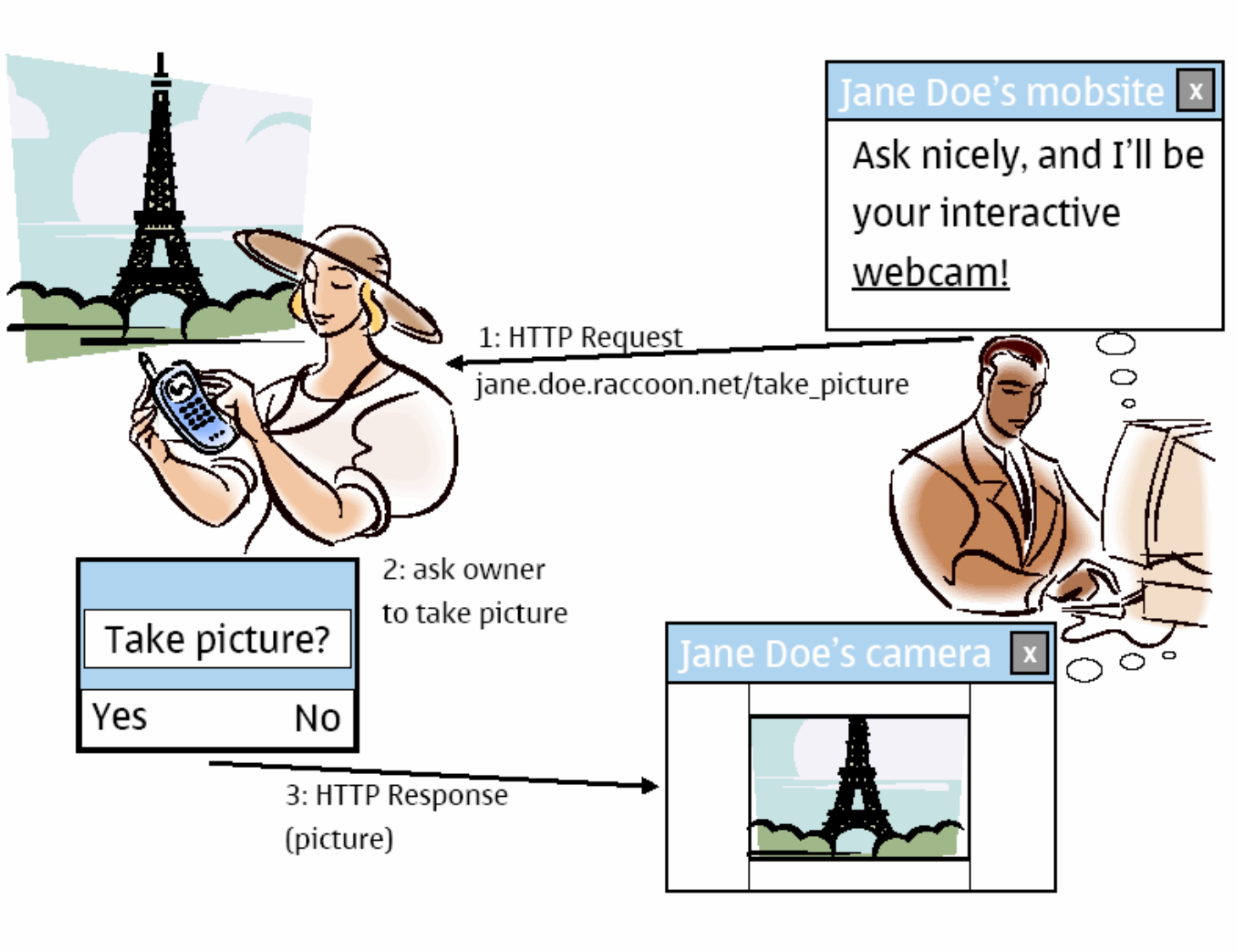


NOKIA
Connecting People

# Interactive Content

- Currently the returned website content depends essentially on the input parameters of the request and sometimes the identity of the one browsing.
  - Often dynamic but there is typically 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 is different.
- The administrator is always nearby and can participate in the content generation.

# Interactive Content - Example







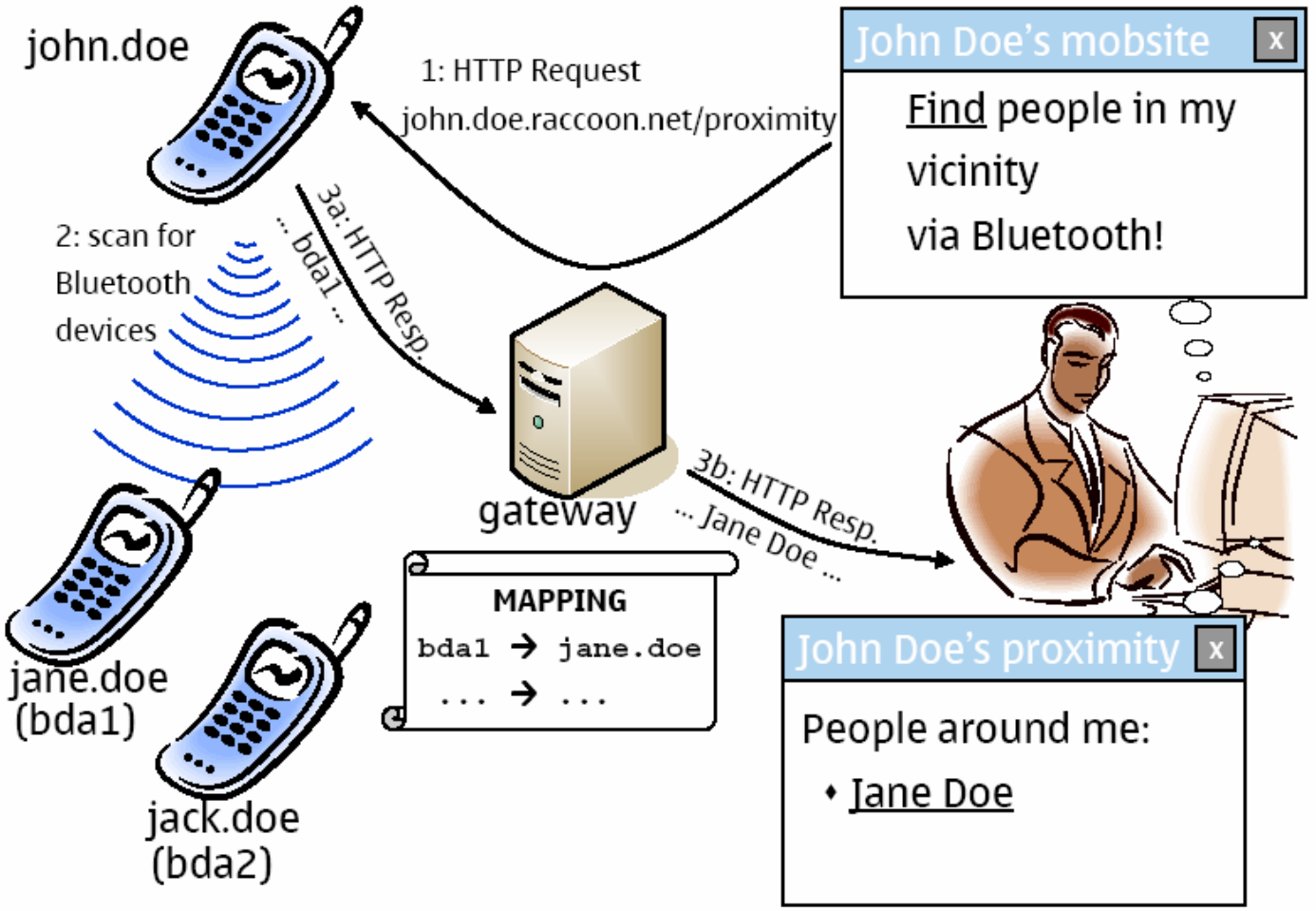

NOKIA
Connecting People

# Context, Location and Time Dependent Content

- In traditional websites the geographical location of the web server lacks meaning since it never changes and it has no impact on the returned content.

- With mobsites this is no longer the case as the returned content may depend on the geographical location and the surrounding context.

# Linking by Proximity – Mobsite Hopping



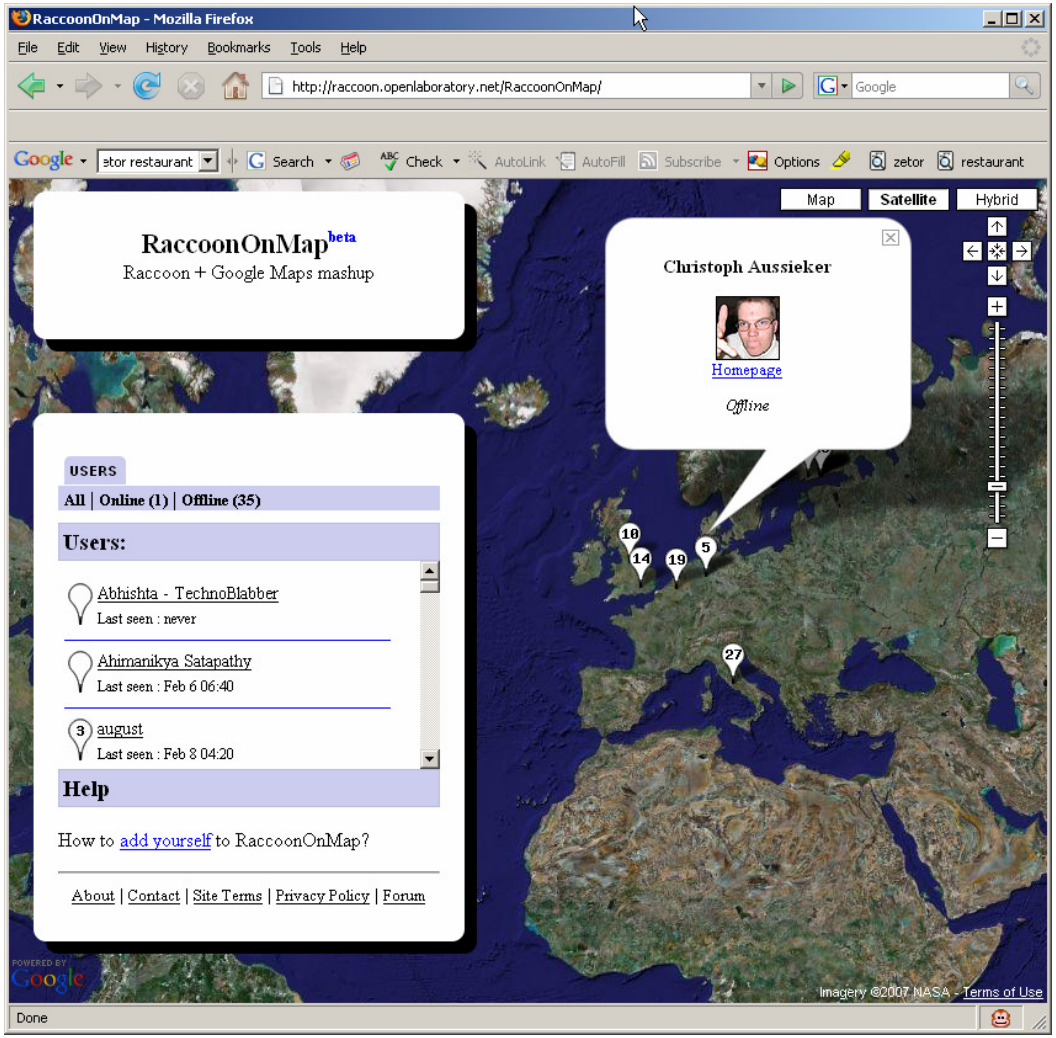

- A new way for linking websites – they are related because they are geographically nearby each other.

NOKIA
Connecting People

# Mashups

- Combine data from mobile with data from other sources.




http://raccoon.openlaboratory.net/RaccoonOnMap



NOKIA Connecting People

# Web UI

- Smartphone UIs pretty user friendly, but still quite constrained when compared with the large display and proper keyboard of a PC.
- But there are PCs everywhere – at home, in the office, in the Internet café.

- If a WEB Interface is created for the core applications of a mobile phone:
  - Whenever you have access to a PC – any PC, not just your own – you could use your phone using the big display and proper keyboard of the PC.
  - You can use any PC as the peripheral of your phone.

- Web UI could become just as important as the native UI.
- Added bonus – you could, for instance, read and answer SMSs when you've forgotten your phone at home.

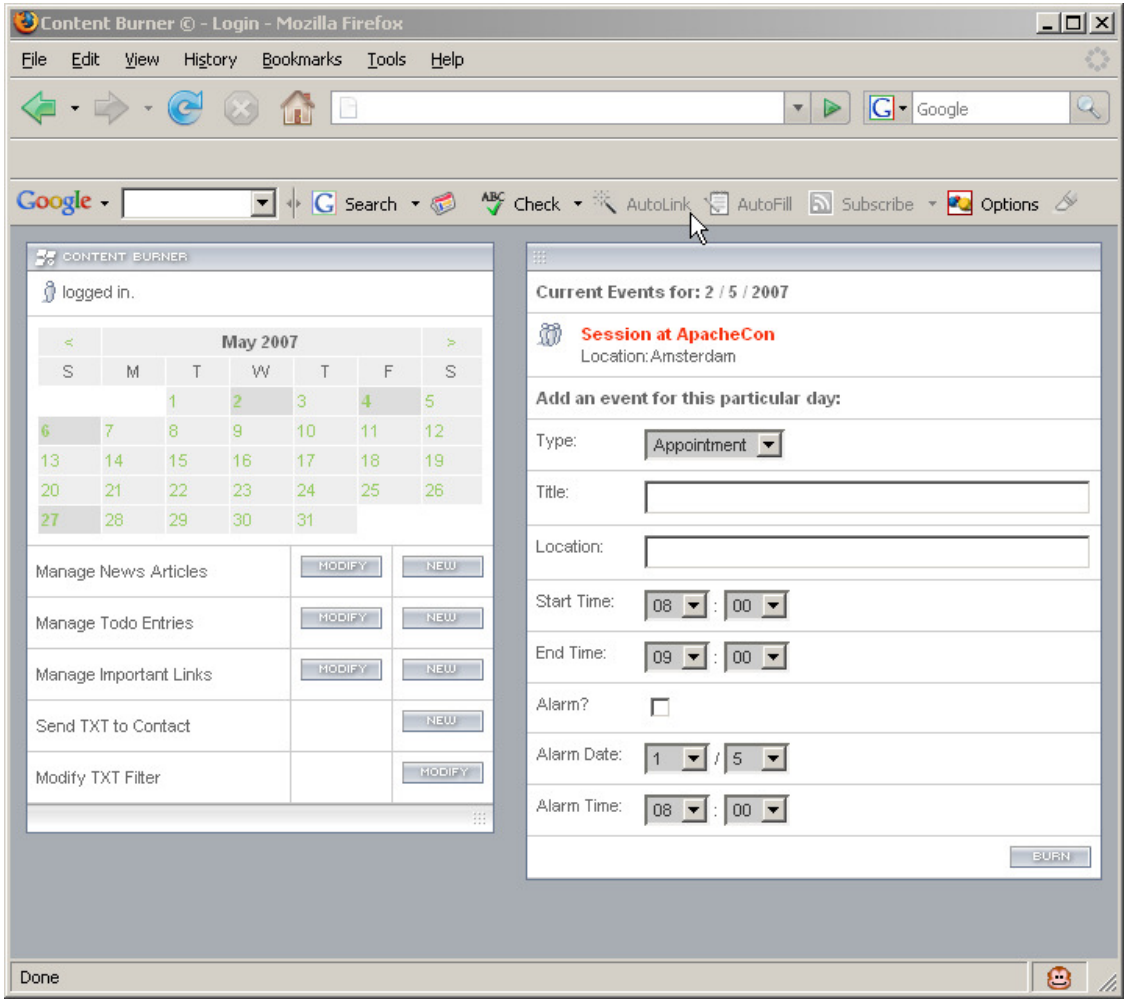




NOKIA
Connecting People

# Content Burner

- Manage your calendar using any browser.




By courtesy of
graham@pixel8limited.com



NOKIA
Connecting People

# Web Services

- Web server implies web services.
- What if there were a web service interface to all core functionality of the mobile phone?

- A calendar is not really useful unless you can make it available to other people.
  - Routine in the corporate world with centralized servers.

- A web service interface to the calendar would make it possible to create a distributed peer-to-peer calendar application without any centralized server.
  - The calendar would suddenly become meaningful in the context of friends or the family.

- Integrate the PC and the mobile phone.

# PC – Phone Integration

- Phone web-server + browser plugin = seamless use of phone from browser.


After plugin installation, all phone-numbers become clickable.

Call, send an SMS or add to contacts without ever leaving your browser.

http://research.nokia.com/research/projects/contacts-browser-plugin/index.html



NOKIA Connecting People

# New Messaging Concepts

- Currently the means for communicating with the phone are basically defined by standards, phone manufacturers and operators,
  - Call, send SMS, send MMS.
- or require custom clients.

- With generic HTTP access it is possible to create new messaging concepts without a need for a-priori standardization or support from the operators.
  - All the client needs is a standard web browser.

- Interesting from a cost perspective as well.
  - If you already have 2.5/3G access – for instance, for browsing the web – functionality similar to that of SMS can be provided essentially for free.

## So, mobsites are just like websites + some more?

## Not quite.

# Cost of Data Transfer

- Two types – monetary and battery.
  - Monetary goes eventually away by itself or does it?
    - Flat rates are becoming more common, but with servers, hmm.
  - Battery cost is not going anywhere in the immediate future.

- Common to both is that the moment traffic enter the wireless media, there is cost, even if the access is later rejected at the mobile web server.

- Using WLAN solves the monetary cost problem, but makes the battery cost problem worse.
  - Except if your mobile happens to be plugged in.

# Access Control

- Regular access-control mechanism can be applied as such, right?

- No
  - The mere delivery of a request to the mobsite causes cost.
  - Access control must be controlled before the request enters the wireless media.

- The reason for access control changes.
  - Perhaps you don't care who can see your pages, but you still might not want just anybody to be able to drain your battery.

- Further, the content delivered by a mobsite may be context dependent.
  - The access control may have to be context dependent as well.

# Security and Privacy

- Phone users are normal people – you can't expect them to become website administrators.
  - Manage accounts and "answer support" calls when some has forgotten his password.
  - You don't want to have a unique login/password at every mobsite you visit.
- It must not be possible to accidentally share personal data.
  - But easy if you want to.
- The identity of someone calling or SMSing you is provided by the system.
  - You should be able to provide similar level of trust when the access comes over the web.
- Some single-sign-on mechanism is needed.

# Availability

- With ordinary websites, administrators go to great lengths in order to ensure the website stays up.
- With mobsites, the assumption can no longer be that the website is always available.
  - People turn off their phones.
  - You run out of battery.

- Difficult to deal with if mobile phones were directly addressable, but easy when you have an intelligent gateway in between.
  - In fact, an intermediate gateway provides an answer to many questions.

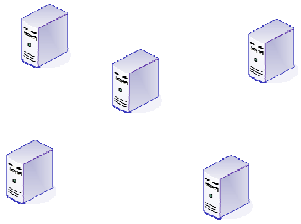
# Litmus Test

Wouldn't it be better to simply upload the stuff to a regular website and share it from there?

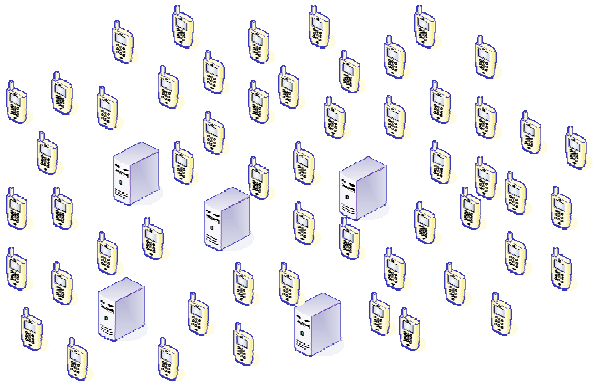
- Without a good answer perhaps you should not proceed...
  - Then again, in some contexts a good answer might be – *I don't have one and I don't trust any providers.*

# What if...

Currently some 50+ million websites.



Tomorrow an additional 500 million mobsites?



- Most websites will reside on mobile and personal devices.
  - Does not imply that all meaningful content would do so.
- The bandwidth becomes an issue – there're limited amounts of it.
- The cost becomes an issue – you pay for all access, if not in monetary terms then in terms of battery consumption.
- The website context becomes an issue – there's lots of it and it changes constantly.
- The timing becomes an issue – there is content that did not exist 5 minutes ago and will not exist 5 minutes from now.
- The interactivity becomes an issue – content can be generated interactively and does not exist before it is asked for.
- The character of searching changes – you can't index content that does not yet exist.



NOKIA
Connecting People

# The Software

- Everything – terminal and gateway software – open sourced under the Apache 2 license.

- http://wiki.opensource.nokia.com/projects/Mobile_Web_Server
- http://opensource.nokia.com/mobile_web_server
- http://research.nokia.com/mobile_web_server
- http://sourceforge.com/projects/raccoon

- We hope that people will play around with the concept.

# Thank You

- Questions?

- email: johan.wikman@nokia.com