Elbonia: Technical Reference

Andreas Ehn Mattias Engblom Oscar Göthberg
Johan Hesselberg Magnus Hult Nicklas Ilebrand
Andreas Mattsson
http://www.nada.kth.se/projects/proj03/elbonia/

7th May 2003

Contents

1 Introduction **1**

2 Handling News Feed **3**
2.1 Plugin Architecture 3
2.2 TTNITF Plugin 3

3 Elbonia Transfer Protocol (ETP) **5**
3.1 The Request 5
3.2 The Response 5

4 Image Format **7**
4.1 Color Palettes 8
4.2 Compressed Images 8

5 The Elbonia Server **9**
5.1 Installation 9
5.1.1 SonyEricsson T300 Compatibility Mode 9
5.2 Starting the Server 10
5.2.1 Command line arguments 10
5.3 How the server works 11
5.4 Security issues 11

6 The Elbonia Client **13**
6.1 About the Client 13
6.2 Building 13
6.3 Installation 13

Chapter 1

Introduction

Elbonia is a system for producing news pages and displaying them on mobile devices. It takes care of everything from getting news, storing them, providing a server and a client to view the pages.

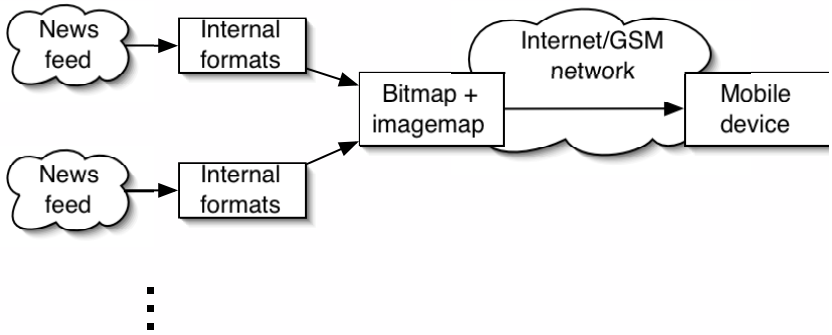

Figure 1.1: Overall system arhictechture

Chapter 2

Handling News Feed

The Elbonia server is built to handle different kinds of news feeds. A flexible plugin architecture allows for getting news feed of different formats and over several channels.

2.1 Plugin Architecture

A plugin may handle its connection to a news feed in any way it finds suitable. When a news item is to be delivered to the Elbonia server, the plugin simply gives the news item to `elbonia-report-news` on standard input.

2.2 TTNITF Plugin

TTNITF[3] is a news format specified by Tidningarnas Telegrambyrå (TT). Getting the news feed from TT is accomplished by means of polling a web server every now and then. FIXME: Only a subset of TTNITF is supported.

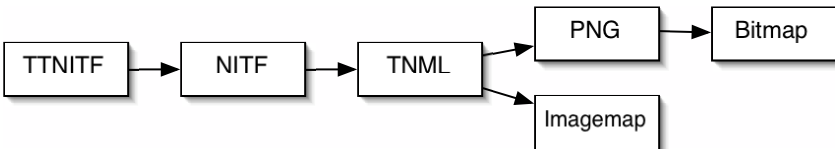

Figure 2.1: Flow of the TTNITF plugin

Chapter 3

Elbonia Transfer Protocol (ETP)

This protocol is inspired by, and very similar to, the Hypertext Transfer Protcol[1]. It defines the way clients and servers communicate with each other.

The protocol is built on top of TCP/IP. Thus, the receiver is guaranteed to get the data in the same order and exactly like the sender sent it.

Unlike HTTP, ETP is not a connectionless protocol. When a client connects to a server, it doesn't disconnect until it is finished talking to that server. The choice not to make ETP connectionless was made because it seems that GSM providers don't allow their customers to connect all the time, and besides, it takes quite some time to initate a WAP connection, both over GSM and GPRS.

3.1 The Request

The client asks the server for resources by issuing a request. A request looks like

`method SP path SP ETP/1.0 (CRLF Resolution: SP width SP height)? CRLF CRLF`

Note that lines end with both carriage return and line feed. Servers should also allow line feed only as line separator.

The only method defined so far is `GET`.

3.2 The Response

A server must answer each request with a response. If the request was in some way errenous, an error response is sent.

A response looks like

`ETP/1.0 SP status-code SP reason-phrase CRLF Content-Length: SP content-length CRLF CRLF (content)?`

where `status-code` and `reason-phrase` are taken from [1], except "501 HTTP Version not supported", which rather means "505 ETP Version not supported".

If `status-code` is 200 (OK), `content-length` is set to the size of the resource in bytes (possibly zero) and `content` is exactly `content-length` bytes of data. Otherwise, `content-length` should be set to zero and no content should be sent.

Chapter 4

Image Format

The news items that are sent from server to client are encoded using the following format. All numbers are sent using network byte order (big endian, that is).

Description	Size/bytes
Number of links	4
List of links	
Palette offset	1
Color format	1
Center along X axis	2
Center along Y axis	2
Image width	2
Image height	2

Table 4.1: Image header format.

Each link in the list of links is laid out as follows:

Description	Size/bytes
Left	4
Top	4
Right	4
Bottom	4
Absolute path (null-terminated string)	varies

Table 4.2: Format of link structure in image header.

Palette offset, center along X axis and center along Y axis should all be set to 0. Color format is one of the ones given in the table below.

Description	Value	Valid image widths
2 colors indexed	3	$8n$, $n > 0$
4 colors indexed	4	$4n$, $n > 0$
16 colors indexed	5	$2n$, $n > 0$
256 colors RGB 332	6	any

Table 4.3: Possible values of color format in image header.

The image that is sent directly after this header is exactly height times width pixels. The color format for the image is the one specified by the member "color format".

Because of the fact that most screen widths on mobile phones aren't powers of two, the image is often sent with 256 colors (either indexed or RGB332).

4.1 Color Palettes

The color palette consists of as many entries as the color format of the image header says it does. If the format is 256 colors RGB 332, the color palette should be exactly 0 bytes long (since the image isn't palettized).

A color palette entry looks like this:

Description	Size/bytes
Red	1
Green	1
Blue	1
Unused (should be 0)	1

Table 4.4: The color palette entry.

The entries are interpreted in order, so that the first entry is palette entry 0, the second is 1 and so forth. If only one entry of the palette is black (RGB 0 0 0) it *must* be entry 0.

4.2 Compressed Images

TODO. We'd like to write something about the Mophun compressed bitmap format here, but unfortunately, we don't know anything about it.

Chapter 5

The Elbonia Server

The Elbonia server listens for connections from clients and delivers content to them once connected. Connections are kept during sessions, unlike for example the Hypertext Transfer Protocol[1]. The server communicates with clients using the Elbonia Transfer Protocol, 3.

5.1 Installation

Installing the server is quite straight-forward. Your system needs to have a C compiler and a version of `make` installed.

First, unpack the distribution archive, and change directory to the distribution directory:

```
    $ gunzip elbonia-x.x.tar.gz
$ tar xf elbonia-x.x.tar
$ cd elbonia-x.x
```

To install, simply follow the instructions in the file `INSTALL`. It might be a good idea to read `README` first.

5.1.1 SonyEricsson T300 Compatibility Mode

The integration between Mophun and the TCP/IP stack on the SonyEricsson T300 (and allegedly on T310 and T610 too) is seriously broken. Thus, no large amounts of data can be transmitted over TCP/IP without considerable care, and even then the system works only partially.

First, the phone can't handle too large packets. If the memory allocation routine does not get the requested memory, the phone shuts off doing a core dump on the serial port.

Secondly, small packets too fast isn't good either. The phone probably has a fixed number of buffers for storing incoming packets, so that sometimes, data is lost and sometimes the transmission just dies.

Therefore, Elbonia provides a so called SonyEricsson T300 Compatibility Mode. When activated, the server does several things before and when sending data to connected clients (T300 or not, the server can't tell):

1. It disables Nagle's algorithm. Nagle's algorithm is used for collecting data to send for some small amount of time (a few hundred milliseconds) and

then send it all together to reduce header overhead. With this disabled, sent packages probably gets a bit smaller.

2. When sending, it sends 200 bytes at a time.
3. After sending 200 bytes, it waits for 0.3 seconds before sending the next 200 bytes.

We've found that this seems to work fairly well. However, with the T300 being able to receive GPRS data in three timeslots and the swedish GSM provider Comviq's bandwidth of 13.4 kb/s, only about 17% of the available bandwidth is used. Also, the transfer works about half of the times tried.

You can read more about how to enable the compatibility mode in the file `INSTALL` in the server distribution directory.

5.2 Starting the Server

When the installation is finished, the Elbonia executable binary should be located where you told `configure` to put it (or `/usr/local/elbonia/` by default).

To start the server, simply type

```
$ elbonia
```

You should now see something like

```
Elbonia Server x.x (SonyEricsson T300 compatibility mode) started.
Server root directory is '.'.
Listening for connections on port 4711.
```

The server is up and running and serves clients that connects to port 4711 on your host.

5.2.1 Command line arguments

If you type `elbonia --help`, you will get a list of command line options. Available options are listed below with a brief description.

–no-daemon, -d

Usually, Elbonia runs as a daemon process. That is, when it starts, it creates a copy of itself and then exits. That way, the process is neither run as a foreground or a background job. To start Elbonia as a normal program, give this option.

–help, -h

Show a list of available command line options and then exit.

–no-reverse-lookup, -l

Tell the server not to perform reverse lookup on remote hostnames. Normally when a client connects, a log message like

```
Connection:  host.name.org:35185, socket 4
```

is printed to the server log. If reverse lookup is disabled, the message is instead something like

```
Connection:  123.234.123.234:35185, socket 4
```

–port=PORT, -p PORT

Tell the server to listen for incoming connections on port `PORT`.

–server-root=ROOT, -r ROOT

Set the server root to `ROOT`. The server root is the directory referred to by a request like

```
    GET / ETP/1.0
Resolution:  101 80
```

–version, -v

Print Elbonia version information and exit.

5.3 How the server works

When the server starts, a socket that listens for connections on a port (4711 by default) is opened.

Each time a client connects, a new server process is spawned. The process accepts the connection and starts processing requests from the client. When the client disconnects, the process dies.

Future versions might use a pool of child processes (`fork()` is expensive) or a multi-threaded arhcitecture instead.

5.4 Security issues

Two big security holes allowing any user to read any file from the server computer (that the user running the server may read) are known.

First, all paths are taken to be relative to the server root and this is achieved by removing one leading slash if present. A malicious client could send this request:

```
GET //home/user/secret-file ETP/1.0
```

which would get the file `/home/user/secret-file` from the server computer.

The second security hole can be exploited if the server root contains any directories. To prevent clients from reading files outside the server root, requests for paths starting with `..` are responded to by "403 Forbidden". If the server root contains a directory called `dir`, a malicious user could send this request:

```
GET dir/../../secret-file ETP/1.0
```

which would get the file `secret-file` from the directory just above the server root.

Both security holes could be avoided by running Elbonia in a `chroot`ed environment. The program will then think that the server root really is the root of the file system. To `chroot` Elbonia , start it with something like

```
$ chroot /home/joe/public_html/news elbonia
```

Chapter 6

The Elbonia Client

6.1 About the Client

The Elbonia client uses the Elbonia Transfer Protocol 3 to request news resources from an Elbonia Server.

The client is written for Mophun[2], a platform independent gaming platform written by Synergenix.

When a user follows a link, an ETP request is issued and the server replies by sending the resource or, if the request was erroneous, an error. If a news resource is sent, it is displayed on the client.

6.2 Building

Since the Elbonia program suite is GPL, source code for all parts is of course released. Digital Rights Management measures prevent users without a mocert (Mophun Certificate) to compile and run their own programs on real phones, but you may use the simulator provided by Synergenix in the Mophun Software Developer's Kit[2] to test the program.

First, download (and, if applicable, build) the Mophun SDK[2]. It's currently available for Windows and Linux (limited support).

Next, unpack the Elbonia client distribution archive and run `make` to compile it. Currently, the address and port to which the client connects is built in at compile time. Make sure to define SERVER_ADDRESS to a string of the IP address you wish to connect to, and SERVER_PORT to the port to which to connect.

6.3 Installation

The Elbonia client can be downloaded to a phone in several ways. It can either be downloaded directly over WAP or transmitted over the infrared port or cable. No installation is required. Refer to the User's Guide for more information of the installation.

Bibliography

[1] R. et. al. Fielding. Hypertext Transfer Protocol – HTTP 1.1. World Wide Web, `http://www.ietf.org/rfc/rfc2616.txt`, 1999.

[2] Synergenix. The Mophun API Reference Guide. World Wide Web, `http://www.mophun.com/download_check.php?id=101`, 2002.

[3] Tidningarnas Telegrambyrå. TTNITF version 3.2. World Wide Web, `http://www.tt.se/tekspec/ttnitfv3_2.doc`, 2002.