

Contents

List of figures	i
List of listings	ii
Introduction	1
1 Backgrounds and Motivations	2
2 X Window System (X11)	6
2.1 Introduction	6
2.2 History	6
2.3 X Window Protocol	7
2.3.1 Introduction	7
2.3.2 Protocol overview	8
2.3.3 Identifiers of resources	10
2.3.4 Atoms	10
2.3.5 Windows	12
2.3.6 Pixmap	14
2.3.7 Events	14
2.3.8 Keyboard and pointer	15
2.3.9 Extensions	17
2.4 X protocol client libraries	18
2.4.1 Xlib	18
2.4.1.1 Introduction	18
2.4.1.2 Data types and functions	18
2.4.1.3 Pros	19
2.4.1.4 Cons	19
2.4.1.5 Example	20
2.4.2 XCB	20
2.4.2.1 Introduction	20

2.4.2.2	Data types and functions	21
2.4.2.3	xcb-util library	22
2.4.2.4	Pros	22
2.4.2.5	Cons	23
2.4.2.6	Example	23
2.4.3	Xlib/XCB round-trip performance comparison	23
3	Implementation	24
3.1	Introduction	24
3.2	Encountered issues	24
3.3	About porting	24
3.4	Existing design	26
3.5	Testing	26
3.6	Debugging tools	26
3.6.1	Xephyr	26
3.6.2	GDB	27
3.7	Remaining issues	28
3.8	Patches	28
3.8.1	Awesome	28
3.8.2	xcb-util library	29
	Conclusion	30
A	Additional code listings	i
A.1	Xlib code of example program #1	i
A.2	XCB code of example program #1	iii
A.3	Performance comparison between Xlib and XCB	vi
B	Patches	ix
B.1	Awesome patches	ix
B.1.1	Replace Pango by Xft to handle client-side fonts	ix
B.1.2	Fix incorrect Xlib function call	xviii
B.1.3	Fix <code>PATH_MAX</code> issue	xix
B.2	xcb-util patches	xxiv
B.2.1	Add non-blocking events loop	xxiv
B.2.2	Fix invalid data returned by ICCCM hints functions	xxv
B.2.3	Add non-blocking events loop	xxix

List of figures

1.1	<i>Screenshot of Awesome</i>	3
1.2	<i>Screenshot of Awesome status bar made of widgets</i>	4
2.1	<i>Two clients using the same display at the same time with different protocols</i>	9
2.2	<i>Interactions between a client and the server</i>	11
2.3	<i>Generated KeyPress event (source: Wikipedia)</i>	14
2.4	<i>Default X keyboard layout</i>	16
3.1	<i>Debugging Awesome using Xephyr (top window) and GDB (bottom window)</i>	27

List of listings

1.1	<i>Show the third tag of the first screen using command line interface</i>	3
1.2	<i>Shell script which displays the current system uptime in the status bar</i>	4
2.1	<i>Windows hierarchy of urxvt X terminal</i>	12
2.2	<i>Window properties defined by urxvt X terminal</i>	13
2.3	<i>Example of generated KeyPress and Expose events when creating a window and typing ls</i>	15
2.4	<i>example program #1 output</i>	20
2.5	<i>Output of program given in sectionA.3</i>	23
3.1	<i>Porting code from Xlib to XCB #1</i>	25
3.2	<i>Porting code from Xlib to XCB #2</i>	25
A.1	<i>Xlib code of example program #1</i>	i
A.2	<i>XCB code of example program #1</i>	iii
A.3	<i>Performance comparison between Xlib and XCB</i>	vi
B.1	<i>[PATCH] Replace Pango by xft to handle client-side fonts #1</i>	ix
B.2	<i>[PATCH] Replace Pango by xft to handle client-side fonts #2</i>	xvi
B.3	<i>[PATCH] Fix incorrect Xlib XGetTransientForHint () call</i>	xviii
B.4	<i>[PATCH] Get rid of PATH_MAX #1</i>	xix
B.5	<i>[PATCH] Get rid of PATH_MAX #2</i>	xxii
B.6	<i>[PATCH] Add non-blocking events loop</i>	xxiv
B.7	<i>[PATCH] Fix invalid data returned by ICCCM hints functions</i>	xxv
B.8	<i>[PATCH] Add non-blocking events loop</i>	xxix

Introduction

Chapter 1

Backgrounds and Motivations

Backgrounds

Awesome¹ is a fast and lightweight Window Manager² initiated by Julien Danjou. It is in fact based on a fork from DWN³ as its authors set terms on its development like that *its source code is intended to never exceed 2000 physical lines of code* and it is *customized through editing its source code*.

Although the project source code was publicly released in September, it has grown quickly to the version 2.2 released on 23th March. The current development version is 2.3.

Awesome supports multihead (e.g. multiple screens) and can be easily customized through a configuration file. It groups windows by *tags*, which may be compared to desktops existing on other window manager or desktop manager, but actually differs in that a window can be *tagged* with one or multiple tags, this allows to display windows from different tags to be displayed on the current one for instance. Each tag has its own layout defining how the windows are managed on the fly, thus the user can adjust the layout depending on the task he wants to accomplish. Windows and tags management can be performed only with the keyboard, ensuring that no mouse is really needed. The following layouts are currently available:

- **floating**: the windows can be resized and moved freely on the current tag like traditional windows managers do. Dialog windows are always displayed as floating independently of the layout set.
- **maximized**: the windows are resized to fill fully the screen.
- **tiled**: this layout is one of the main specificities of Awesome. The windows are managed in a master area, which contains windows the user wants to focus his attention on, and a stacking area where lies all other windows. It assures that no space is wasted on the screen because there is no gaps nor overlaps between

¹<http://awesome.naquadah.org>

²See glossary

³<http://www.suckless.org/wiki/dwm>

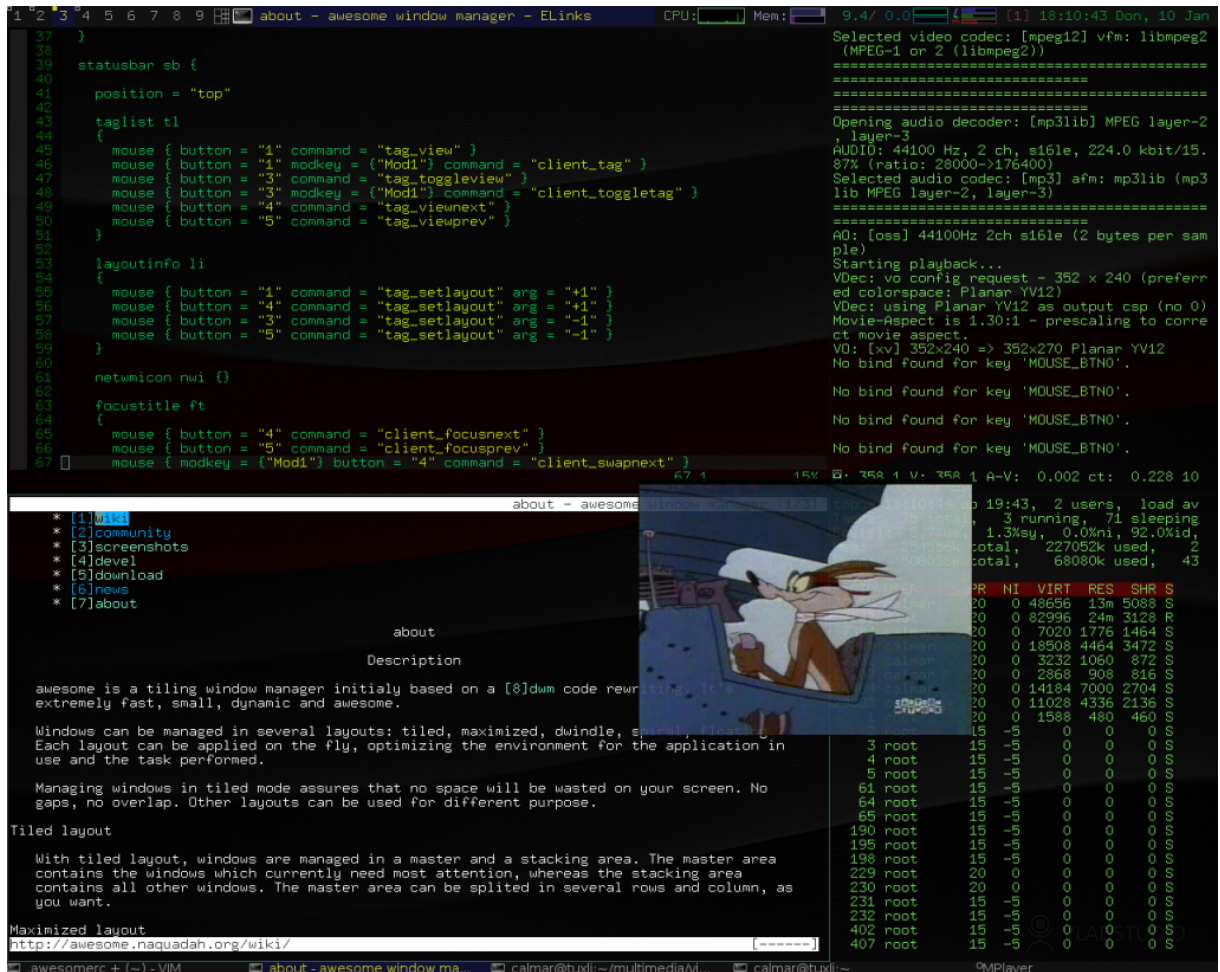


Figure 1.1: Screenshot of Awesome

windows in a given tag. The master area is by default on the right frame, but Awesome also defined `tileleft`, `tilebottom` and `tiletop` layout depending on the master area position. This layout also allows to specify the number of rows and columns of the master area. The figure 1.1 shows windows managed thanks to the default `tile` layout with only one row and column set on the master area.

- **spiral**: the windows are getting smaller towards the right bottom corner of the screen.
- **dwindle**: the windows are getting smaller in a fibonacci spiral.

A third-party program, `awesome-client`, permits to communicate with Awesome process through a socket thanks to the command line interface (or CLI) described in a reference document. Awesome main process receives the command sent by the user or a script and then call the appropriate functions. For instance the user may type the following command to view the third tag of the first screen using command line interface as shown on listing 1.1.

```
$ echo 0 tag_view 3 | awesome-client
```

Listing 1.1: Show the third tag of the first screen using command line interface

Awesome also includes the notion of widgets which allows to display various information in a status bar (figure 1.2) in a flexible way. Indeed, an user can defined his own widget or use the following pre-defined widget:

- **textbox;**
- **tasklist;**
- **iconbox;**
- **progressbar;**
- **graph;**
- **layoutinfo;**
- **taglist;**
- **focusicon;**



Figure 1.2: Screenshot of Awesome status bar made of widgets

The figure 1.2 shows usage of the following widgets (from left to right): *taglist*, *layoutinfo*, *iconbox*, *tasklist*, *textbox*, *graph*, *progressbar*... The user gets the wanted informations using a third party program which calls `awesome-client` in order to populate the informations to the widgets or directly communicate with the socket.

For instance, if the user wants to display the system uptime in the status bar, he firstly has to add a *textbox* widget in the configuration file. Secondly, it pipes the output of `uptime` Unix command (an example shell script is provided by the listing 1.2) to `awesome-client` responsible of updating the previously declared widget by communicating with `awesome` process. The listing shows a *shell* script which displays into a widget named *uptime* the current system uptime.

```
1 #!/bin/sh
3 while true; do
4     # Exit when awesome-client returns an error, for instance when
5     # awesome is not running anymore...
6     ( echo "0_widget_tell_uptime_$(uptime)" | awesome-client > /dev/
7       null 2>&1 ) || exit 1
8 done
```

Listing 1.2: Shell script which displays the current system uptime in the status bar

Motivations

Before switching to Awesome Window Manager last september, I was using Ion3 pre-released versions on DEBIAN GNU/LINUX . However, in May 2007, the lead developer of Ion3 project, previously licensed under LGPL free software license, decided to add a non-free clause to the existing license making it non-free. Basically, packages provided by GNU/LINUX distributions can not *be given names that can not be associated with the "Ion" project, or be qualified as "Ion soup", and still be considerable as customised versions of this software*⁴. Consequently, after looking for a decent replacement of a Window Manager allowing keyboard management of windows like Ion3 does, I decided to give a try to a quite recent and promising Window Manager: Awesome. Then, I moved to Awesome because it suits particularly my needs.

I have always been interested in understanding X Window System concepts and applies them by participating in Ion3 development but I was a bit reluctant to learn Lua, based programming language of Ion3. As Awesome is entirely written in C programming language, when I switched to Awesome, I submitted some short patches fixing various bugs I had. Then, I decided to focus my year project on being familiar with Awesome code and adding features afterwards. In addition, Awesome development moves quite fast thanks to its main and original developer, Julien Danjou who happily applies patches sent to him, ensuring that my work is useful. These are why I thought this project could have been the ideal opportunity to add features and also to be involved in its development.

Julien Danjou suggested some ideas of features I could work on. Porting Awesome from Xlib to more recent XCB X client library specially caught my attention among other features I could have implemented. Actually, XCB provides a cleaner and more consistent API than Xlib as well as improving the performance.

⁴See <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=422527#10> for further details about this issue

Chapter 2

X Window System (X11)

2.1 Introduction

The X Window System (commonly X11 or X) is a standard and complete network-transparent graphical windowing system based on a client/server model built on top of an operating system kernel.

X provides the basic toolkit and protocol used for building GUI environments and toolkits. X is referred as basic because it allows to draw and move window on one or more screens, and interact with input/output devices used in a graphical context (like mouse and keyboard for example), but does not provide the user interface. Actually, X was designed to provide low-level mechanisms for managing the graphic display.

Unlike traditional client/server applications, the server runs on the local machines and owns the input devices, allowing the remote or local client programs to draw or display fonts on the display by sending requests to the it. Usually, there is client, known as a window manager, which interact with the X server in order to manage the windows on a screen.

2.2 History

The X Window System is based on an early window system, namely W WINDOW SYSTEM, developed for the V OPERATING SYSTEM. It is originated from the lack of existing alternatives of an platform-independant and license-available display system and kept only some concepts and the name from W. It was originally developed at MIT in 1984 by Jim Gettys and Robert W. Scheifler and renamed to X when asynchronous protocol replaced the synchronous protocol from W.

After several releases, X11 was released in 1987 along with the current protocol used nowadays. A meeting with several vendors pointed out that the development should be handled by a third neutral party following commercial and educational interests instead of the MIT in order to avoid forks of the project. From this idea originated a non-profit vendor group some months later, the *MIT X Consortium* led by Robert W. Scheifler.

In 1992, X11, originated from the first port of X Window Server for IBM PC compatibles, was donated to the *MIT X Consortium*. X11 evolved over time from a port for IBM PC compat-

ibles to the leading and most popular implementation of X server.

In 1996, the *X Consortium* was dissolved and passed the stewardship of X in 1997 to the *Open Group* which supervised the release of X11R6.4. Then the OPEN GROUP formed X.Org and released X11R6.5.1 but the innovations was taking place only in X11. That's why the latter project was encouraged to join X.Org because some hardware companies were also interested in using this project with GNU/LINUX and by its popularity.

In 2004, many dissensions within the X11 led to the release of X11 4.4 under a more restricted license which didn't suit project relying on it because it was viewed as not compatible with GNU GENERAL PUBLIC LICENSE (GPL). In the meantime, various people from X.Org and FREEDESKTOP.ORG founded the X.Org foundation led to radical changes in X development because the X.Org was led by vendor organization whereas the X.Org foundation is led by software developers. In April 2004, this foundation released X11R6.7 based on X11 4.4RC2 (before the license modification) and X11R6.6.

Nowadays, the most popular X implementation is the one from the X.Org foundation based on X11R7.X version which is a modular release of previous monolithic version. It is available on GNU/LINUX and many Unix-like operating systems.

2.3 X Window Protocol

2.3.1 Introduction

As explained in section 2.1, the X Window System is based on a client/server model which, at a first glance, may appear backward compared to the traditional client/server model.

Indeed, the server provides a display server to graphical applications, therefore it manages the output devices like screens and input devices, like keyboard and mouse for example.

All other applications are considered as clients because they use the service provided by the X Window Server. Usually, there is at least one client started by user that has authority for the layout of windows on the display, namely a WINDOW MANAGER (also known as WM). It usually handles the following kind of operations common to the end-user nowadays:

- move windows around the screens;
- change window size;
- switch to another workspace;
- provide window decorations like titlebar;

A given client and the server communicate asynchronously via the X Window Protocol. It specifies that *it is important to keep in mind that the protocol is intended to provide mechanism, not policy*, thus this protocol does not specify:

- Inter-clients interactions to transfer data between windows (selections, cut and paste and drag and drop are some common examples intended by end-user when using a graphical

interface) but also between the window manager and the windows it manages. These interactions are however described in specifications like ICCCM¹, EWMH² and NetWM³.

- Common widgets like buttons, menus, textbox... but toolkits can be built on top of this protocol in order to provide these features.

These interactions are defined in separate specifications like ICCCM, EWMH and NetWM.

2.3.2 Protocol overview

The X Window Protocol provides network transparency, which means that the server and a client can both run on the same host or on different ones. It allows to secure communications by tunneling the connection over an encrypted channel without including encryption within the protocol, thus make the protocol specification simpler. For instance, it is possible for clients to communicate with the server over a SSH tunnel.

X requires a reliable and bidirectional stream protocol like DECNET (VMS operating system) or TCP/IP, although the usage of the former is deprecated nowadays because it provides a far less address space and does not have the advanced features provided by the latter protocol suite.

When running both of server and clients on the same host, TCP/IP leads to a huge overhead mainly caused by packet processing, that's why X can also be used over Unix domain protocol. This protocol is based on inter-process communication (IPC) and thus allows efficient and bidirectional communication between the server and the clients. The figure 2.1 shows a possible situation where two client applications share the same display over different protocols. Although this figure only shows a situation with only one X server process, a host may start several X server at the same time from different terminals (TTY in GNU/LINUX), this way it allows several displays to be used.

Let is assumed that n is the display number starting from 0, when a X server is started, it creates a TCP end-point by opening a port (by default $6000 + n$) or can also create an Unix domain socket (generally located in `/tmp/.X11-unix/Xn`) or both depending on the server settings. A client can now connect on this X server by the following steps:

1. The client sends the first packet in order to initiate the connection to the server and must send the following datas:

- **Byte order** (endianess).
- **Protocol version.**
- **Authorization mechanism.**

This is not described in the X Window Protocol and it is ignored if not supported.

2. The server may returned back to the client the following status:

- Failure.

¹See glossary

²See glossary

³See glossary

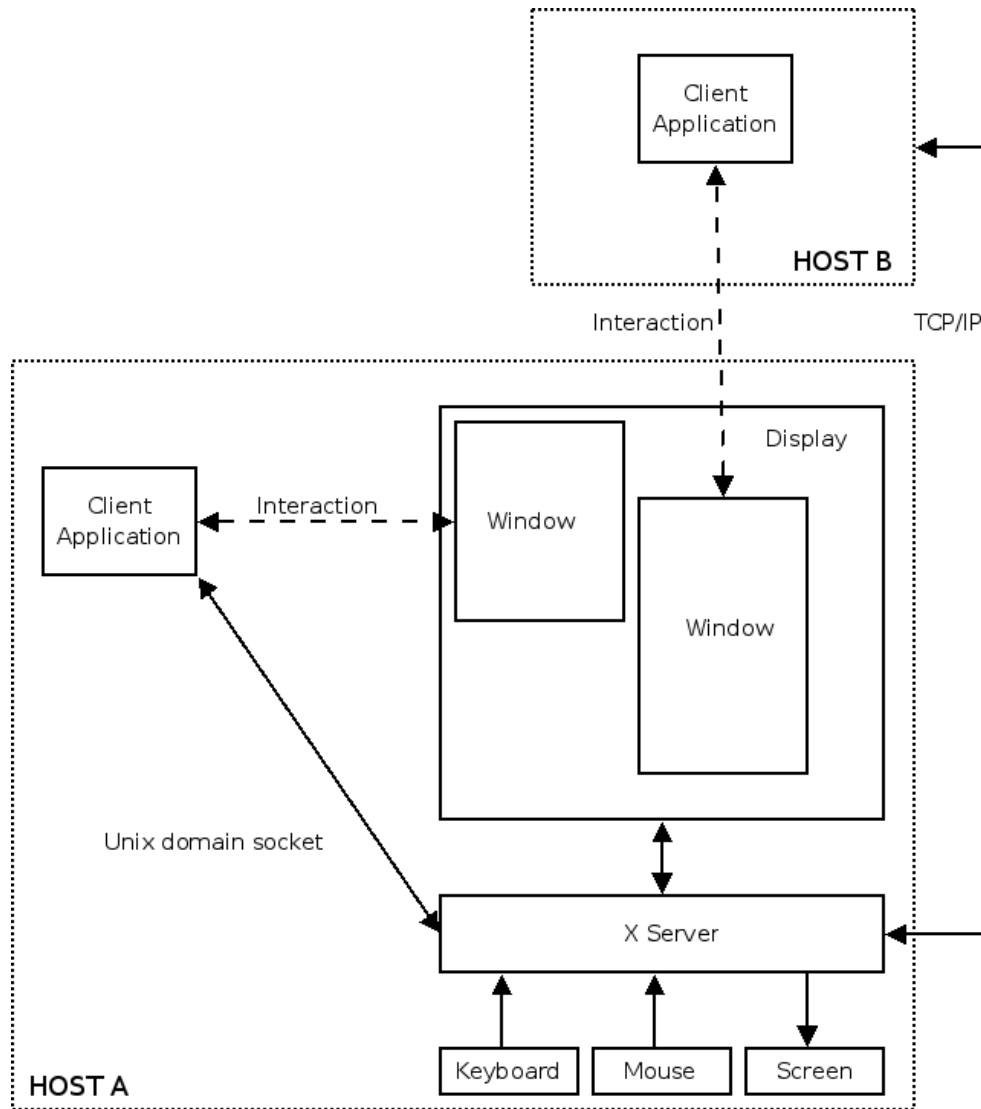


Figure 2.1: *Two clients using the same display at the same time with different protocols*

- Success.
- Authenticate.

In this case, a reason is returned depending on authorization protocol in use and further authentication negotiation is required before accepting or refusing the connection.

3. If the returned status is `Success` and the connection has been successfully established, then the server sends back to the client informations needed for interactions between them (these informations will be described later).

Once the client receives these informations, several types of messages might be exchanged:

- **Request** (varying length).

The client sends a request to the server identified by a sequence number starting from 0.

- **Reply** (varying length).

The server may send a reply (**not all request generates a reply**) which contains the least significant 16 bits of the request sequence number.

- **Error** (32 bytes long).

It includes a 8 bit error code and, like reply, contains the least significant 16 bits of the failed request.

- **Event** (32 bytes long).

It includes a 8 bit type code (further details about events is given in the next section).

A client and a server may exchange a lot of data depending on the client application design. Figure 2.2 shows an example of interactions between the server and the clients when performing no authentication and considering that the connection status is `Success`. After establishing the connection, the client sends to the X server queue a request to create a window which does not generate a reply, then asks for attributes and geometry of the previously created window. All these requests may generate errors which might not be sent immediately because events are queued.

2.3.3 Identifiers of resources

Data like windows, pixmaps and graphic context⁴ are resources stored server-side and destroyed by default when the server is reset. When the client requests to create a new resource, the following steps are executed:

1. the client asks the current identifier (XID) range to the server and commonly picks the next sequential number;
2. the server allocates the resource and associates it to the given XID;
3. the client can now perform operations on this resource (like drawing) by specifying this identifier;

This mechanism avoids copying of the resource between the server and the client which requested the resource allocation and also enables resource sharing among clients on the same X server, consequently resulting in a more efficient use of the network.

The figure 2.1 shows that the root window created during X server startup has been associated to identifier `0x69` and `urxvt` window is associated to identifier `0x2600009`. Another client on the same X server could draw on the window by giving its identifier.

2.3.4 Atoms

An Atom is a unique 32 bits integer identifier assigned by the server (as opposed to a resource identifier explained in section 2.3.3) used often in inter-clients communications. As the identifier has a fixed and short length, it is consequently faster to process on a network. The string is stored in the server and referenced as Atom name.

⁴See glossary

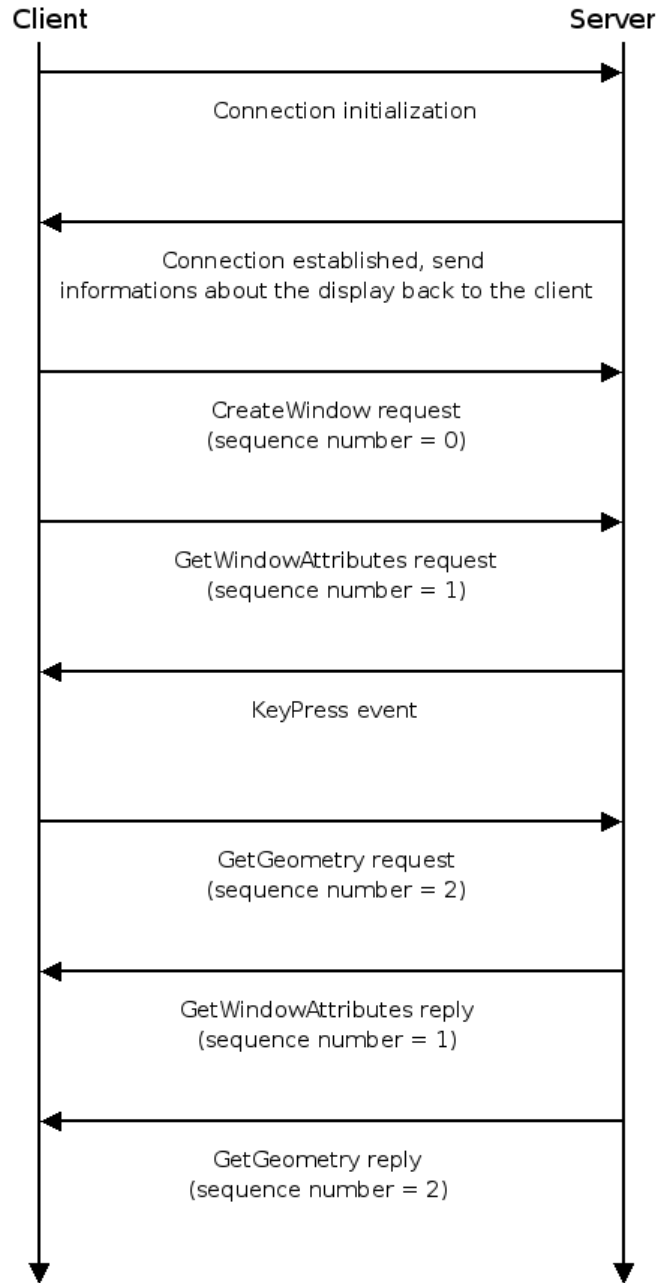


Figure 2.2: *Interactions between a client and the server*

A client can request allocation of an Atom by sending the intended name in a `InternAtom` request, the server replies by sending its identifier. It is automatically created if it does not already exist. The opposite operation is achieved by sending `GetAtomName` request.

Contrary to the default behavior of a resource identifier, an Atom is kept even when the server is reset. In order to reduce network usage, most common Atoms needed by applications are already allocated in the server (like `WM_CLASS` and `WM_NAME`).

2.3.5 Windows

In the X Window System, the windows are hierarchically ordered as a tree where the **root window** is the parent of all other windows (or more precisely *subwindows*). This *special* window is generally as large as the physical screen and is situated behind all its children.

When the client sends a request to create a window (namely `CreateWindow` in the X Window Protocol) or more precisely a subwindow of an existing window, it needs to specify (beside other informations described in later sections of this chapter) the following:

- identifier of its parent;

A top-level window (e.g. a child of the root window) is created by setting the parent window to the identifier of the root window. But as other screen informations, the identifiers of the root window is sent by the server when the connection has been successfully established⁵.

- class of the window;

- `InputOnly`;

Specify that the window can receive events but can not be used as a source or destination for graphics requests.

- `InputOutput`;

Specify that the window can receive events and can be used for drawing.

A client can request window destruction by sending `DestroyWindow` request.

The listing 2.1 shows an example of output when running `xwininfo -tree` command which allows to display the tree based on the window pointed with the mouse. Thus, it indicates that `urxvt` creates only one top-level window (width: 694 pixels, height: 1026 pixels, absolute *x* coordinate is at pixel 703, absolute *y* coordinate is at pixel 21).

```
1 xwininfo: Window id: 0x2600007 "arnau@maggie: ~"
3   Root window id: 0x69 (the root window) (has no name)
4   Parent window id: 0x69 (the root window) (has no name)
5     1 child:
      0x2600009 (has no name): () 694x1026+2+2 +703+21
```

Listing 2.1: *Windows hierarchy of urxvt X terminal*

A window possesses attributes (border, cursor, accepted events...) stored server-side. They can be fetched and set respectively by sending `GetWindowAttributes` and `ChangeWindowAttributes` request. Window decorations are accomplished by the Window Manager, not the client itself.

A window also possesses properties allowing for example to inform about the behavior it desires to the Window Manager (figure 2.2 show some common properties). A property is stored server-side as an Atom and thus characterized by its name, type and value and can be fetched

⁵Second step shown in figure 2.1

or set by sending respectively `GetWindowProperty` and `ChangeWindowProperty`. By design, the X Window Protocol does not describe these properties, but however provides the necessary request and reply format to manage them. For instance, the NetWM specifies that the window title is stored in `_NET_WM_NAME` (an Atom) property (UTF-8 string).

```

WM_STATE(WM_STATE) :
2         window state: Normal
          icon window: 0x0
4  _AWESOME_PROPERTIES (STRING) = "100000000"
  _NET_WM_PID (CARDINAL) = 12313
6  WM_PROTOCOLS (ATOM): protocols WM_DELETE_WINDOW, _NET_WM_PING
  WM_LOCALE_NAME (STRING) = "fr_FR.UTF-8"
8  WM_CLASS (STRING) = "urxvt", "URxvt"
  WM_HINTS (WM_HINTS) :
10         Client accepts input or input focus: True
          Initial state is Normal State.
12         window id # of group leader: 0x2600007
  WM_NORMAL_HINTS (WM_SIZE_HINTS) :
14         program specified minimum size: 11 by 18
          program specified resize increment: 7 by 14
16         program specified base size: 4 by 4
          window gravity: NorthWest
18  WM_CLIENT_MACHINE (STRING) = "maggie"
  WM_COMMAND (STRING) = { "urxvt" }
20  _NET_WM_ICON_NAME (UTF8_STRING) = 0x61, 0x72, 0x6e, 0x61, 0x75, 0x40, 0
          x6d, 0x61, 0x67, 0x67, 0x69, 0x65, 0x3a, 0x20, 0x7e
  WM_ICON_NAME (STRING) = "arnau@maggie: ~"
22  _NET_WM_NAME (UTF8_STRING) = 0x61, 0x72, 0x6e, 0x61, 0x75, 0x40, 0x6d,
          0x61, 0x67, 0x67, 0x69, 0x65, 0x3a, 0x20, 0x7e
  WM_NAME (STRING) = "arnau@maggie: ~"

```

Listing 2.2: *Window properties defined by urxvt X terminal*

It is also worth noting that a window is considered as a **drawable** like Pixmap explained later and the window content is not guaranteed to be preserved. The X server provides two methods to make sure that the window content is preserved:

- using *backing-store*⁶ configurable by sending `ConfigureWindow` request;
- by generating an `Expose` event notifying that the window content has to be drawn again;

The second method is the most widely used because the window content is generally managed by the Window Manager. Indeed, most X server implementations drops backing-store if memory becomes limited (backing-store may consume a lot of memory) but also because some X server implementation doesn't provide backing-store at all.

⁶See glossary

2.3.6 Pixmaps

A Pixmap is just a two dimensional array of bits used for drawing and is therefore considered as a drawable (like a window). A Pixmap is an off-screen resource which can be partially or completely transferred to a window and vice-versa, thus allowing *double-buffering*⁷. It is often used as a picture buffer or a background pattern. In the X Window Protocol, a client requests allocation of a Pixmap using `CreatePixmap` and freeing by `FreePixmap`.

2.3.7 Events

According to the X Window Protocol specification, clients are informed of informations by means of events and can be generated from input/output devices owned by the X server, or also as side effects of clients requests (for instance by `SendEvent` request which allows a client to send a request to another client). As described in section 2.3.2, events is one of the four specific messages exchanged between the server and the clients (see figure 2.2 for an example of interactions involving a `KeyPress` event described below). Each event message usually possesses its own format.

An event is sent relatively to a window. In order to receive events, any clients has to explicitly stated what events it is interested in for a given window (figure 2.3 provides an example of `KeyPress` event). This can be achieved by specifying a *mask* when creating the window by sending `CreateWindow` request or at anytime thanks to `ChangeWindowAttributes` request.

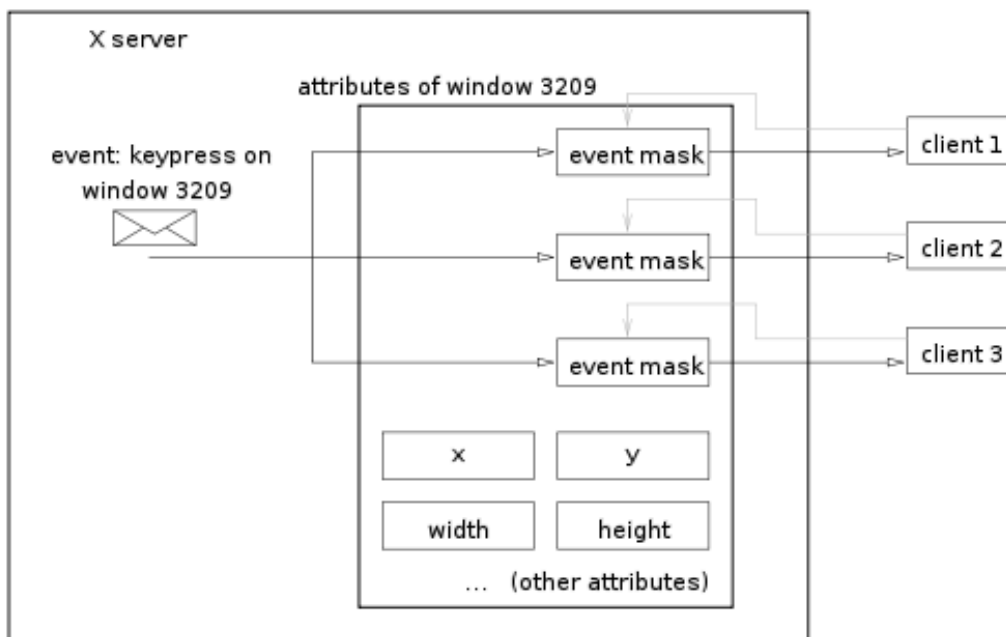


Figure 2.3: *Generated KeyPress event (source: Wikipedia)*

For instance, a client may set `EventMask` to `KeyPress` and `Expose` allowing a window

⁷See glossary

to received these events from the server when a key a button has been pressed or when the window content is invalid meaning that the window has to be redrawn (as explained in section 2.3.5). The listing 2.3 shows events sent when a client application creates a new window and then presses `ls` in the situation described above.

```
1 Expose event, serial 14, synthetic NO, window 0x2200007,
   (0,0), width 1398, height 2, count 3
3
5 KeyPress event, serial 14, synthetic NO, window 0x2200007,
   root 0x69, subw 0x2200009, time 3337013, (212,386), root:(213,405)
   ,
7   state 0x0, keycode 46 (keysym 0x6c, l), same_screen YES,
   XLookupString gives 1 bytes: (6c) "l"
   XmbLookupString gives 1 bytes: (6c) "l"
9   XFilterEvent returns: False
11 KeyPress event, serial 17, synthetic NO, window 0x2200007,
   root 0x69, subw 0x2200009, time 3337132, (212,386), root:(213,405)
   ,
13   state 0x0, keycode 39 (keysym 0x73, s), same_screen YES,
   XLookupString gives 1 bytes: (73) "s"
15   XmbLookupString gives 1 bytes: (73) "s"
   XFilterEvent returns: False
```

Listing 2.3: *Example of generated `KeyPress` and `Expose` events when creating a window and typing `ls`*

In the instance where no clients has stated its interest in a specific event, the server may also send the event to ancestor windows if it is not specified in the *do-not-propagate-mask*.

2.3.8 Keyboard and pointer

Each physical or logical keyboard key is associated with an unique key code (known as `KeyCode` in the X Window Protocol specification). It may be associated to at least one character or special key, names a `KeySym`, selected thanks to special keys called modifiers. In most common case, the following modifiers are available on the end-user keyboard (figure 2.4 shows the default keyboard layout for PCs):

- Shift;
- Control;
- Meta;
- Compose;
- CapsLock;
- NumLock;



Figure 2.4: *Default X keyboard layout*

The association table between `KeyCodes` and `KeySyms` are maintained server-side making it available and modifiable to all clients. It allows the end-user to use a different keyboard layout than the default shown in figure 2.4.

Concerning pointer buttons, the X server also uses a modifier mapping which may only be permuted if the user desired it.

Within a window, when the key state changes or the pointer moves, the following self-explanatory events messages may be generated by the server:

- keyboard related events:
 - `KeyPress`;
 - `KeyRelease`;
- pointer related events:
 - `ButtonPress`;
 - `ButtonRelease`;
 - `MotionNotify`;

All of these event messages shares the same format whose fields are the root window, source window and coordinates available for both. Depending on the source of the event, it also contains a *detail* field holding the key code or the button and a *state* field storing the current keyboard modifiers or mouse buttons. The server does not convert a `KeyCodes` associated with modifiers `KeySyms` in a `KeyCodes` or vice-versa, it is actually performed by the client.

For a given window, a client can grab or ungrab (e.g. consequently all events will respectively be sent or not to this client) the following:

- a key (`((Un)GrabKey request)`);
- the pointer (`((Un)GrabPointer request)`);
- the keyboard (`((Un)GrabKeyboard request)`);
- a button (`((Un)GrabButton request)`);

2.3.9 Extensions

The X server is intended to be kept simple, so defining new request, events or errors packet formats can be achieved through extensions to the protocol and then added as a separate library from the server. As many extensions exist, the list of extensions given in the following table is not exhaustive, it aims to describe only the extensions needed for later explanations:

XRender	provide several rendering operations and <i>alpha blending</i> ⁸
RandR	allow desktop resizing and display rotating on the fly
Xinerama	allow to split the desktop accross multiple monitors
Xkb	enhances keyboard handling and control

2.4 X protocol client libraries

A client program can interact with the X server using a X client library implementing the X Window Protocol, like Xlib and XCB. Generally, when designing a typical graphical application, most programmers rely on a library on top of a X protocol client library like GTK or QT. Indeed, they expect the library to provide a widgets⁹ toolkit, like labels, buttons or menus for example, without writing this code by hand nor understanding a quite complex communication protocol like X Window Protocol. However, writing a Window Manager involves X Window Protocol access for controlling placement and appearance of windows within a screen.

2.4.1 Xlib

2.4.1.1 Introduction

The Xlib is a protocol X client library in the started in 1985 and currently supported by any operating systems relying on the X Window System, mostly Unix-like operating systems. This library is a free software (under the terms of the MIT license) based on a monolithic architecture providing ICCCM-compliant standard properties and programming interfaces. Currently, almost every widget toolkits are built on top of this library to communicate with the X server.

2.4.1.2 Data types and functions

As most resources are maintained by the X server and accessible for clients through identifiers (described in section 2.3.3), the Xlib provides types for these identifiers, which are actually just platform-independent 32-bits integers. Beside these primitive types, the data types and functions in the Xlib library can be roughly grouped as:

Operations on the connection An opaque `Display` structure represents a connection to the X server initialized when calling `XOpenDisplay()` and destroyed by `XCloseDisplay()`. This private structure holds notably the following datas:

- a file handle to the network socket used for communication between the server and the client. This file can be accessed at any time through `ConnectionNumber()` function.
- client-side datas as described in next paragraph.
- Display informations like the number of screens available on the X server, the identifier of the root window used to create top-level windows for example (as described in section 2.3.5).

Clients-side datas and operations In Xlib, each event generated from the server corresponds to a structure containing its specific informations (for instance `XKeyEvent` representing both `KeyPress` and `KeyRelease` events as described in the X Window Protocol). In addition, it specifies an `XEvent` union holding an event structure specified by the `type` field.

⁹See glossary

It is worth noticing that the Xlib does not usually send requests immediately but store them client-side in a specific buffer, commonly called the *output buffer*. These stored requests are usually executed asynchronously on the X server when any function returning a value or waiting for input is called (as a side-effect the buffer is flushed completely). The client may state its interests in protocol errors (generated from a request by the server) by setting an appropriate handler corresponding to the error type. The error is sent as soon as the X server detects reports protocol errors.

Xlib provides specific inspection and management operations on the event queue it maintains client-side:

- `XFlush()`: flushes the request output buffer.
- `XSync()`: flushes the request output buffer and waits until the server has processed all the sent requests.
- `XNextEvent()`: copies the first event in the queue to a specified `XEvent` structure and then removes it. Otherwise, if the event queue is empty, it flushes the request output buffer and block until an event is received.
- `XPending()`: returns the number of events in the events queue.
- `XCheckMaskEvent()`: searches the queue for the first event which matches the specified *event-mask*. If an event matches, then it is copied to the `XEvent` structure given as a parameter before removing it from the queue and returning `True`. Otherwise, it returns `False`.
- `XMaskEvent()`: searches the queue for any events which matches the specified *event-mask*. It returns a list of matching events if any, otherwise it flushes the request output buffer and blocks until one is received.

Requests to the server These include requests for operations and informations as defined in the X Window Protocol. In addition, the Xlib also provides facilities functions around most common requests (like for instance `XSetWindowBorder()` which actually performed a `ChangeWindowAttributes`). It also provides convenient structures for each reply sent by the server.

When a request sent by the client requires a reply to be generated from the server, the client is *blocked* by the Xlib until a reply is generated by the server, even if the client does not need the reply immediately, thus slowing down considerably the client for most requests it sends.

2.4.1.3 Pros

- well-established;
- well-documented;

2.4.1.4 Cons

- monolithic architecture resulting in big library size;
- complex and ugly code;

- inconsistent API;
- requests requiring a reply are synchronous;

2.4.1.5 Example

The listing A.1 (section A.1) shows commented code of a Xlib program displaying on the terminal the output given in listing 2.4 when doing:

1. the window is shown (generate an `Expose` event);
2. a pointer button is pressed;
3. the pointer is moved to another location on the window while keeping the button pressed;
4. the previously pressed button is released;
5. Switch to another virtual desktop;
6. Switch back to the created window virtual desktop;
7. the pointer is pressed and released immediately;

```

Current default screen size: 1400x1050, associated root window: 105
2 Expose event received!
  Button pressed at [61, 132] (window relative coordinates)
4  Button released at [42, 88] (window relative coordinates)
  Expose event received!
6  Button pressed at [42, 88] (window relative coordinates)
  Button released at [42, 88] (window relative coordinates)

```

Listing 2.4: *example program #1 output*

2.4.2 XCB

2.4.2.1 Introduction

XCB (acronym of X C BINDING) is a C-language binding¹⁰ for the X Window System started in 2001 by Bart Massey. The core and extension X Window Protocol are described as XML files generated in via XSLT¹¹. *It is a replacement for Xlib featuring a small footprint, latency hiding, direct access to the protocol, improved threading support and extensibility* (quoted from XCB description). Like Xlib, it is a free software licensed under the terms of the MIT license. Barton Massey and Robert Bauer proved key portions of XCB formally correct using Z notation. Unlike Xlib, XCB is based on a modular architecture.

¹⁰See glossary

¹¹See glossary

2.4.2.2 Data types and functions

Like Xlib, XCB provides types for resources except that XCB relies on fixed-size integers data type standardized by ISO C99 (commonly available in `stdint.h` C library header). Instead of defining macros, XCB makes extensive usage of `enums`, thus avoiding to define specific structures uselessly. For instance, Xlib defines a specific type `XSetWindowAttributes` (listing A.1 in section A.1), whereas XCB relies on a pointer of `uint32_t` and `enum` to achieve exactly the same goal without increasing the API.

Like Xlib, XCB operations can be roughly grouped as:

Operations on the connection An opaque `xcb_connection_t` structure represents a connection to the X server initialized when calling `xcb_connect()` and destroyed by `xcb_disconnect()`. Like the whole XCB API, the number of members in this private structure has been dramatically reduced, the most important members are:

- a file handle to the network socket like Xlib but is accessed through `xcb_get_file_descriptor()` function.
- client-side datas as described in next paragraph.
- a pointer to a `xcb_screen_t` structure holding screen informations. Stepping through the different screens can be achieved by iterators.

Clients-side datas and operations XCB maintains an queue holding requests sent by the client and another queue holding both events and errors sent by the server. The queues are managed thanks to the following functions:

- `xcb_flush()`: flushes the request queue, thus causing all requests to be send to the server.
- `xcb_aux_sync()`: similar to `xcb_flush()` but also waits until the server finishes processing all the requests.
- `xcb_wait_for_event()`: blocks until an event is queued by the X server, and once it becomes available it is immediately dequeued. If an error occurs, this function returns `NULL`. It is considered as an equivalent to Xlib functions `XNextEvent()` and `XMaskEvent()`.
- `xcb_poll_for_event()`: returns the first event in the queue, directly dequeued, or `NULL` if there is no event. It is an equivalent to Xlib functions `XCheckMaskEvent()` and `XMaskEvent()`.

Both `xcb_wait_for_event()` and `xcb_poll_for_event()` return a pointer to a `xcb_generic_event_t` structure allocated in the heap which has to be freed afterwards. This structure may be afterwards casted to the proper event type (like `xcb_key_press_event_t` for example) according to the `response_type` field, because all events types are based on the declaration of this generic event structure. Unlike Xlib functions, XCB doesn't provide a way to inspect and manage the event queue. Indeed, once an event or an error is dequeued, it can't be requeued afterwards, meaning that the client can not step through items in the queue (this kind of operations add much more complexity and are not really needed, especially within a low-level library like XCB).

Requests to the server When a request is sent with XCB, it returns a *cookie* which is an identifier used to get the reply anytime. The type of cookie is `xcb_void_cookie_t` for requests that do not generate a reply from the server, whereas for other requests, each one has its own cookie type, for instance `xcb_get_window_attributes_t` is returned by `xcb_get_window_attributes()` (GetWindowAttributes request in the X Window Protocol). It is preferable to send the request and then ask for the reply as later as possible by the clients ensuring it has already been processed by the X server. Therefore, This mechanism makes requests and replies completely asynchronous.

XCB provides two modes, namely *checked* and *unchecked*, defining where the request is stored for further processing by the client. If a request generates a protocol error in the former mode, then the error goes into the event queue and may be processed in the event loop, whereas the error is held aside in the latter mode until the client asks for the answer. An error is just a structure names `xcb_generic_error_t` where the `response_type` field equals to 0.

2.4.2.3 xcb-util library

XCB intends to be a lower-level library than Xlib, as such it does not provide facilities like ICCCM functions or handler facilities. In order to avoid for programmers to write themselves this kind of code, some XCB programmers started the xcb-util library based as well on a modular which provides the following *module* (only the one used in Awesome XCB port are described below):

- **xcb-atom**: includes standard core X atom constants (notably `WM_NAME`, `WM_CLASS`...) and atoms caching.
- **xcb-aux**: convenient access to connection setup and some core requests. It especially contains `xcb_aux_sync()` described in section 2.4.2.2 and `xcb_aux_get_screen()` allows to get a pointer on a structure holding various informations about a XCB connection and a screen number (for instance the root window identifier associated to the screen number).
- **xcb-event**: callback X events handling like Xlib does.
- **xcb-icccm**: both clients and Window Manager helpers for ICCCM.
- **xcb-keysym**: standard X key constants and conversion to and from keycodes.

2.4.2.4 Pros

- modular architecture resulting in smaller library size;
- making multithreading easier and reliable;
- consistent API;
- completely asynchronous;

2.4.2.5 Cons

- not well documented;
- extensions missing (Xkb ...);

The cons points quoted above result from youthfulness of the XCB project. In addition, most functions are easily understandable by reading Xlib documentation which is quite complete or the X Window Protocol specification.

2.4.2.6 Example

The code given on listing A.2 (section A.2) is just the XCB equivalent to program explained in section 2.4.1.5.

2.4.3 Xlib/XCB round-trip performance comparison

The listing A.3 given in section A.3 shows an example program comparing Xlib and XCB performances about requests and replies (also known as a round-trip). It sends 500 atoms and displays the following output:

```
1 => XCB wrong usage (synchronous)
   Duration: 0.038608s
3
5 => XCB bad usage (asynchronous)
   Duration: 0.002691s
   Ratio    : 14.35
7
9 => Xlib traditional usage
   Duration : 0.043487s
   Ratio    : 16.16
```

Listing 2.5: *Output of program given in sectionA.3*

This output clearly shows a dramatic speedup of requests and replies asynchronous XCB method compared to Xlib traditional synchronous method of sending a request and blocks until the reply is ready. In this case, XCB is about 16 times quicker than Xlib. This difference about requests and replies time processing becomes more and more obvious as the number of Atom requests sent following this method is important. XCB also provides a slight speedup compared to Xlib even when both of them use the same method.

Chapter 3

Implementation

3.1 Introduction

Awesome is maintained thanks to Git¹, a revision control system (also known as a RCS). Basically, this kind of tool allows to work collaboratively on the same source code by providing features like *historic of changes made to the source* among many other interesting features for developers. Since I have begun to work on this project, I maintain my changes thanks to Git². It allows me to stay synchronized with Julien Danjou work by making *merges* between his code and mine periodically.

Before beginning the port, I had to read a lot of documentation because I was pretty new about X programming, like the X Window Protocol reference, Xlib documentation and XCB tutorial. I had only worked with GTK toolkit, which is much more high-level than XCB is. Then, I wrote some small programs to become familiar with both XCB and Xlib API.

3.2 Encountered issues

During the porting effort of Awesome from Xlib to XCB, I encountered several problems, especially about documentation. In fact, XCB is not well-documented making port harder. The XCB tutorial is one of the only resource apart of the list of available functions. Hopefully, I was able to get help on #xcb IRC channel.

3.3 About porting

I worked during several months without actually seeing anything because the port from Xlib to XCB API has to be done completely before testing it. I follow these steps to port whole Awesome code to XCB:

1. modification of the build system (`Makefile.am` files and `configure.ac`);

¹<http://git.or.cz/>

²available on: <http://git.naquadah.org/?p=arnau/awesome.git;a=summary>

2. replace all Xlib types and fonction prototypes, sometimes involving running `grep` through Xlib code to look for undefined values in XCB;
3. port one source file at the same time, once it is finished, build the associated object file ensuring that there is no warning from the compiler;

Functions based on requests defined explicitly in the X Window Protocol are quite straightforward to port from Xlib to XCB, for instance:

```

1  /* Xlib code */
2  XDestroyWindow ((*sw)->display , (*sw)->>window);
3  XFreePixmap ((*sw)->display , (*sw)->drawable);
4
5  /* XCB equivalent */
6  xcb_destroy_window ((*sw)->connection , (*sw)->>window);
7  xcb_free_pixmap ((*sw)->connection , (*sw)->drawable);

```

Listing 3.1: *Porting code from Xlib to XCB #1*

However, porting a call from Xlib to XCB usually involves running `grep` in Xlib source code and then read it to implement properly the code with XCB (figure 3.2 shows a port of Xlib `XMapRaised()`). This step is usually needed to ensure that the ported call has been written properly in order to avoid spending many hours at the end...

```

1  /* Xlib code */
2  int
3  XMapRaised (
4      register Display *dpy ,
5      Window w)
6  {
7      register xConfigureWindowReq *req;
8      register xResourceReq *req2;
9      unsigned long val = Above;          /* needed for macro */
10
11     LockDisplay (dpy);
12     GetReqExtra (ConfigureWindow , 4, req);
13     req->>window = w;
14     req->mask = CWStackMode;
15     OneDataCard32 (dpy , NEXTPTR (req , xConfigureWindowReq) , val);
16     GetResReq (MapWindow , w, req2);
17     UnlockDisplay (dpy);
18     SyncHandle ();
19     return 1;
20 }
21
22 /* XCB equivalent code */
23 void
24 xutil_map_raised (xcb_connection_t *conn , xcb_window_t win)
25 {
26     const uint32_t map_raised_val = XCB_STACK_MODE_ABOVE;

```

```

27     xcb_configure_window (conn , win , XCB_CONFIG_WINDOW_STACK_MODE,
29                             &map_raised_val);

31     xcb_map_window (conn , win);
}

```

Listing 3.2: *Porting code from Xlib to XCB #2*

3.4 Existing design

Awesome relies on `Xinerama` X extension to provide `multihead` support and also on `Randr` extension allowing to resize a screen and rotate a display on the fly.

In addition of these X extension, Awesome is also based on Cairo, a *2D* graphics library with support for multiple output, and most notably a Xlib backend and an experimental XCB backend. The drawing operations are actually all performed by using Cairo.

Font handling is performed with Xft and renders using Cairo until Awesome 2.2. From current development version, text and font are renders with Pango library (forms the core of text and font handling for GTK+-2.X) helps by a Cairo backend for Pango³.

More informations about Awesome sources files and dependencies between them is available in the `doc` directory on the provided CD-ROM.

3.5 Testing

I am currently using the XCB version I have ported from Xlib since one week and a half and didn't notice any bugs. In addition, at first startup of Awesome, I have even noticed that it is now more responsive with XCB library than Xlib, for instance switching between tags seem quicker. However, it is not really possible to measure quantitatively the performance improvement made by switching to XCB.

3.6 Debugging tools

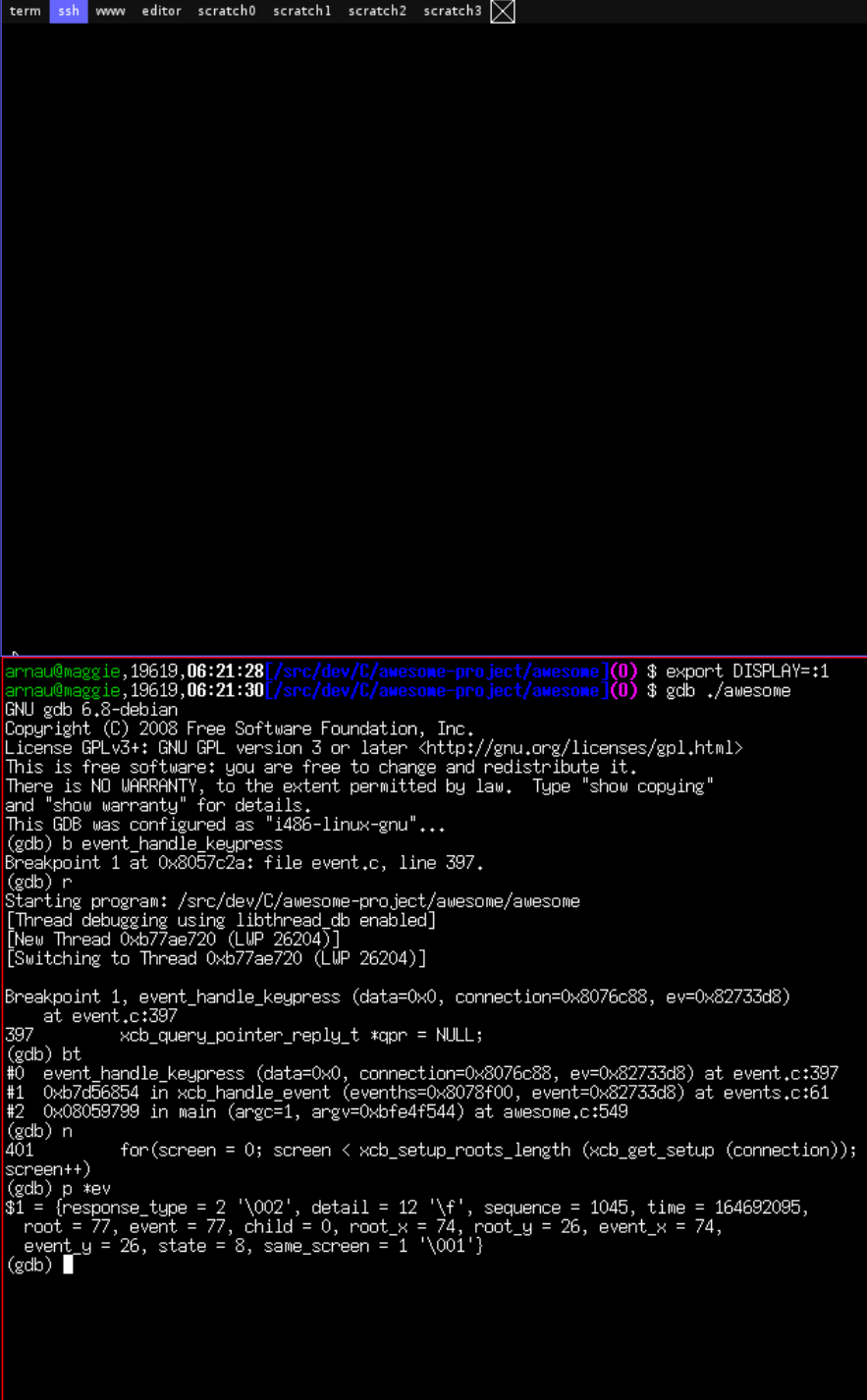
3.6.1 Xephyr

Xephyr is a nested X server, meaning that it can run inside another X server. This is particularly useful when trying to fix bugs in Awesome code because it allows to debug it directly from the main X Window Server.

³See section 3.8.1 for further informations

3.6.2 GDB

GDB is a free software command line debugger, allowing for instance to set *breakpoint* on specific instructions and also display values of variables in memory. The figure 3.1 shows debugging of Awesome using Xephyr and GDB.



```
term ssh www editor scratch0 scratch1 scratch2 scratch3 ✕
arnau@maggie,19619,06:21:28[/src/dev/C/awesome-project/awesome](0) $ export DISPLAY=:1
arnau@maggie,19619,06:21:30[/src/dev/C/awesome-project/awesome](0) $ gdb ./awesome
GNU gdb 6.8-debian
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...
(gdb) b event_handle_keypress
Breakpoint 1 at 0x8057c2a: file event.c, line 397.
(gdb) r
Starting program: /src/dev/C/awesome-project/awesome/awesome
[Thread debugging using libthread_db enabled]
[New Thread 0xb77ae720 (LWP 26204)]
[Switching to Thread 0xb77ae720 (LWP 26204)]

Breakpoint 1, event_handle_keypress (data=0x0, connection=0x8076c88, ev=0x82733d8)
at event.c:397
397     xcb_query_pointer_reply_t *qpr = NULL;
(gdb) bt
#0  event_handle_keypress (data=0x0, connection=0x8076c88, ev=0x82733d8) at event.c:397
#1  0xb7d56854 in xcb_handle_event (events=0x8078f00, event=0x82733d8) at events.c:61
#2  0x08059799 in main (argc=1, argv=0xbfe4f544) at awesome.c:549
(gdb) n
401     for(screen = 0; screen < xcb_setup_roots_length (xcb_get_setup (connection));
screen++)
(gdb) p *ev
$1 = {response_type = 2 '\002', detail = 12 '\f', sequence = 1045, time = 164692095,
      root = 77, event = 77, child = 0, root_x = 74, root_y = 26, event_x = 74,
      event_y = 26, state = 8, same_screen = 1 '\001'}
(gdb) █
```

Figure 3.1: Debugging Awesome using Xephyr (top window) and GDB (bottom window)

3.7 Remaining issues

Xkb X extension support in XCB has begun some months, but at the moment, it is not finished at all yet (in other words, the XML file describing the extension is not still not complete). That is why the XCB port of Awesome misses this feature. I have already begun working on this issue by reading some documentations about Xkb internals, but unfortunately I didn't have enough time yet to begin hacking on Xkb support in XCB.

According to Julien Danjou, the porting effort to XCB is already completed at 95% but the code is still not in a releasable shape because it needs to be optimized. Indeed, the code has to be reorganized by keeping in mind that once a request is sent, the reply must be asked to the server as later as possible leaving time to the server to process the request and thus taking fully-advantage of the asynchronous model proposed by XCB. This reorganization may be achieved, depending on the part of the code, by executing as much code as possible between a request and a reply or by splitting function in two parts: one would send the request while the other one would treat the reply.

3.8 Patches

When porting Awesome from Xlib to XCB, I discovered some bugs or made some improvements in Awesome and xcb-util library. These patches may not be related directly to the Awesome port effort but contributes to improve existing code.

3.8.1 Awesome

Use Pango instead of Xft for handling client-side fonts

Awesome was using Xft library to handle client-side fonts until Awesome version 2.2. However, this library has not been ported yet to XCB and is not the preferable way. Indeed, Awesome is using Cairo which could have been used to handle fonts, but it would have meant to write Pango-like code. In addition, Pango provides code, namely `pangocairo`, to interact with cairo data structures. This patch is available in section B.1.1.

Fix incorrect Xlib function call

When reading Awesome and Xlib codes, I found a bug caused by inconsistency in Xlib between `XGetTransientForHint()` prototype and actual returned value. Indeed, this function returns a `Status` where `Success` equals to 0 whereas the function code returns a boolean (0 in case of error and 1 otherwise). This patch is available in section B.1.2.

PATH_MAX issue

According to POSIX specification, an operating system **may** define a macro called `PATH_MAX` specifying the maximum size of a path name. This macro permits to store a path name in a string allocate statically (e.g. on the stack). However, some operating systems, like GNU/Hurd, does not have such paths size restriction, thus does not defined this macro and compilation fails.

I made two patches which get rid of `PATH_MAX` in favor of heap memory allocations. These patches are available in section B.1.3.

3.8.2 xcb-util library

Fix invalid returned data

`icccm/icccm.c` source file provides a `xcb_get_wm_size_hints()` function. It actually gets the `WM_SIZE_HINTS` client program property defined in ICCCM specification. This property allows the client program to specify to the Window Manager further informations about its geometry, like its minimum and maximum sizes for example. This XCB function used to fill the `xcb_size_hints_t` given as a parameter.

It also contains a `xcb_wm_hints_get_input()` function which gets the `WM_HINTS` ICCCM property, allowing to change window state of the client application top-level window or the window urgency hints requiring the timely response of the user. This function returns a pointer on a `xcb_wm_hints_t` structure newly allocated.

Both functions actually send `XGetProperty` request and gets the returned value according to ICCCM specification. Unfortunately, the value data from `GetProperty` reply is incorrectly copied into structures described above.

I sent a patch to the XCB mailing list which fixes this issue and also make this API consistent, this patch should be applied soon. Indeed, both functions now returns a pointer on hints structures newly allocated and freed using specific destructors. This patch is available in section B.2.2.

Add a non blocking event loop

`event/events.c` source file included a `xcb_wait_for_event_loop()` function. This function only calls `xcb_wait_for_event()` (blocking way to receive events) as a `while` loop condition. Its body calls the handler associated to the received event.

However, this source file does not provide a non-blocking way of event handling based on `xcb_poll_for_event()`. The patch available in section B.2.3 fixes this issue and has already been applied in `xcb-util` official git repository.

Add missing `XLookupString` XCB equivalent

The Xlib function `XLookupString()` translates a key event to a `KeySym` and a string. For instance, if the user presses `Tab`, it would be translated to a `^`string. However, even if `keysyms/keysym.c` includes a function to translate a key event to a `KeySym`, there is no function to get a string.

I haven't written a patch against `xcb-util` yet although this code has already been implemented in `awesome-menu.c` source file. This code is based on Xlib `XLookupString()` debugging, because otherwise I couldn't figure out how it works by just reading the Xlib code. I will prepare a patch implementing this feature in `xcb-util` soon though.

Conclusion

Appendix A

Additional code listings

A.1 Xlib code of example program #1

```
/* Compilation: gcc -lX11 xwindow-xlib-example.c -o xwindow-xlib-  
example */  
2 #include <stdio.h>  
#include <stdlib.h>  
4 #include <error.h>  
  
6 #include <X11/Xlib.h>  
  
8 static int  
xlib_error_handler(Display *dpy, XErrorEvent *err)  
10 {  
    if(err->error_code == BadAlloc || err->error_code ==  
        BadImplementation)  
12        /* The server is screwed: display the error and exit */  
        error(EXIT_FAILURE, 0,  
14            "[ERROR]:_fatal_error:_request_code:_%d,_error_code:_%d",  
                err->request_code, err->error_code);  
16 }  
  
18 int  
main(void)  
20 {  
    /* Open a connection to the X server on the display specified by the  
22     DISPLAY environment variable as the parameter equals to NULL */  
    Display *dpy = XOpenDisplay(NULL);  
24  
    /* Check whether the connection has been successful */  
26    if(dpy == NULL)  
        error(EXIT_FAILURE, 0, "[ERROR]:_cannot_open_display");  
28  
    /* Get the default screen number */  
30    int screen_nbr = DefaultScreen(dpy);  
32  
    /* Get the root window */
```

```

Window root_win = DefaultRootWindow(dpy);
34
/* Display various informations about the display */
36 printf("Current_default_screen_size:_%dx%d, associated_root_window:_%d\n",
        DisplayWidth(dpy, screen_nbr), DisplayHeight(dpy, screen_nbr)
38         ,
        root_win);

40 /* Set the default error handler */
XSetErrorHandler(xlib_error_handler);

42
/* Prepare the window attributes given later to XCreateWindow(), set
44 the background pixel color to white and the event mask stating
interests in 'Expose', 'ButtonPress' and 'ButtonRelease'
46 events */
XSetWindowAttributes wa;
48 wa.background_pixel = WhitePixel(dpy, screen_nbr);
wa.event_mask = ExposureMask | ButtonPressMask | ButtonReleaseMask;
50
/* Create a new InputOutput top-level window whose size is 150x200
52 at origin [10, 50] with a border width equals to 0 */
Window win = XCreateWindow(dpy, root_win, 10, 50, 150, 200, 0,
54         CopyFromParent, InputOutput,
        CopyFromParent,
        CWEventMask | CWBackPixel, &wa);

56
/* Map the window on the screen (e.g. display it!) */
58 XMapWindow(dpy, win);

60 Bool button_release_count = 0;
XEvent ev;

62
/* Main event loop displaying the coordinates of the pointer only in
64 the event window */
while(button_release_count != 2)
66     {
        /* XNextEvent() flushes the request output buffer */
68         XNextEvent(dpy, &ev);
        switch(ev.type)
70         {
            case Expose:
72             printf("Expose_event_received!\n");
            break;

74
            case ButtonPress:
76             {
                XButtonPressedEvent *e = (XButtonPressedEvent *) &ev;
78                 printf("Button_pressed_at_[%d, %d]_(window_relative_
                coordinates)\n",
                    e->x, e->y);
80             }

82             break;

84             case ButtonRelease:

```

```

86     {
        XButtonReleasedEvent *e = (XButtonReleasedEvent *) &ev;
        printf("Button_released_at_[%d, %d]_(window_relative_
            coordinates)\n",
88             e->x, e->y);

90         /* Exiting when a button has been released two times */
        ++button_release_count;
92     }

94     break;
    }
96 }

98 /* Close the connection to the X server leading to destruction of
    all associated windows and the call of XSync() in order to get
100 any pending errors */
    XCloseDisplay(dpy);
102
    return EXIT_SUCCESS;
104 }

```

Listing A.1: Xlib code of example program #1

A.2 XCB code of example program #1

```

/* Compilation: gcc 'pkg-config --cflags --libs xcb xcb-aux xcb-event
    ' \
2   *           xwindow-xcb-example.c -o xwindow-xcb-example
    */
4 #include <stdio.h>
    #include <stdlib.h>
6 #include <error.h>
    #include <stdbool.h>
8
    #include <xcb/xcb.h>
10 #include <xcb/xcb_aux.h>
    #include <xcb/xcb_event.h>
12
    /* Not defined in XCB */
14 #define BadAlloc 11
    #define BadImplementation 17
16
    /* Number of errors */
18 #define ERRORS_NBR 256

20 static int
    xcb_error_handler(void *data, xcb_connection_t *conn,
        xcb_generic_error_t *err)
22 {

```

```

24  /* Get the real request code which just failed */
    int real_request_code = *((uint8_t *) err + 10);

26  if(err->error_code == BadAlloc || err->error_code ==
    BadImplementation)
    /* The server is screwed: display the error and exit */
28  error(EXIT_FAILURE, 0,
    "[ERROR]:_fatal_error:_request_code:_%d,_error_code:_%d",
30  real_request_code , err->error_code);
}

32
int
34 main(void)
{
36  /* Open a connection to the X server on the display specified by the
    DISPLAY environment variable as the parameter equals to NULL */
38  int default_screen;
    xcb_connection_t *conn = xcb_connect(NULL, &default_screen);
40
    /* Check whether the connection has been successful */
42  if(xcb_connection_has_error(conn))
    error(EXIT_FAILURE, 0, "[ERROR]:_cannot_open_display");
44
    /* Get the screen informations */
46  xcb_screen_t *screen = xcb_aux_get_screen(conn, default_screen);

48  /* Display various informations about the display */
    printf("Current_default_screen_size:_%dx%d,_associated_root_window:_%d\n",
50  screen->width_in_pixels , screen->height_in_pixels ,
    screen->root);
52
    /* Allocate events handler */
54  xcb_event_handlers_t *evenths = xcb_alloc_event_handlers(conn);

56  /* Set the default error handler for all types of errors */
    unsigned int err_num;
58  for(err_num = 0; err_num < ERRORS_NBR; err_num++)
    xcb_set_error_handler(evenths , err_num , xcb_error_handler , NULL);
60

    /* Prepare the window attributes given later to xcb_create_window(),
    set the background pixel color to white and the event mask
    stating interests in 'Expose', 'ButtonPress' and 'ButtonRelease'
    events */
62  const uint32_t create_win_vals [] = {
64  screen->white_pixel ,
    XCB_EVENT_MASK_BUTTON_PRESS |
68  XCB_EVENT_MASK_BUTTON_RELEASE | XCB_EVENT_MASK_EXPOSURE
    };
70

    /* Create a new InputOutput top-level window whose size is 150x200
    at origin [10, 50] with a border width equals to 0, but firstly ,
    allocate a window resource on the server.... */
72  xcb_window_t win = xcb_generate_id(conn);
74  xcb_create_window(conn ,
76  XCB_WINDOW_CLASS_COPY_FROM_PARENT,
```

```

78         win, screen->root, 10, 50, 150, 200, 0,
          XCB_WINDOW_CLASS_INPUT_OUTPUT,
          XCB_WINDOW_CLASS_COPY_FROM_PARENT,
80         XCB_CW_BACK_PIXEL | XCB_CW_EVENT_MASK,
          create_win_vals);
82
      /* Map the window on the screen (e.g. display it!) */
84      xcb_map_window(conn, win);
86
      bool button_release_count = 0;
      xcb_generic_event_t *ev;
88
      /* Main event loop displaying the coordinates of the pointer only in
90       the event window */
      while((ev = xcb_poll_for_event(conn)) && button_release_count != 2)
92      {
          switch((ev->response_type & 0x7f))
94          {
              case XCB_EXPOSE:
96                  printf("Expose_event_received!\n");
                  break;
98
              case XCB_BUTTON_PRESS:
100             {
                  xcb_button_press_event_t *e = (xcb_button_press_event_t *)
                      ev;
102                  printf("Button_pressed_at_[%d, %d]_(window_relative_
                      coordinates)\n",
                          e->event_x, e->event_y);
104             }
106             break;
108
              case XCB_BUTTON_RELEASE:
110             {
                  xcb_button_release_event_t *e = (
                      xcb_button_release_event_t *) ev;
                  printf("Button_released_at_[%d, %d]_(window_relative_
                      coordinates)\n",
112                          e->event_x, e->event_y);
114
                  /* Exiting when a button has been released two times */
                  ++button_release_count;
116             }
118             break;
          }
120
          free(ev);
122      }
124
      /* Close the connection to the X server leading to destruction of
          all associated windows */
126      xcb_disconnect(conn);
128
      xcb_free_event_handlers(evenths);

```

```

130     return EXIT_SUCCESS;
    }

```

Listing A.2: XCB code of example program #1

A.3 Performance comparison between Xlib and XCB

```

1  /*
2  *  Compilation: gcc -std=c99 'pkg-config --libs --cflags xcb' -lX11 \
3  *                xwindow-xlib-xcb-performances.c -o xwindow-xlib-xcb-
4  *                performances
5  *  This program is based on: http://xcb.freedesktop.org/tutorial/
6  */
7
8  /* asprintf() */
9  #define _GNU_SOURCE
10
11 #include <stdlib.h>
12 #include <stdio.h>
13 #include <string.h>
14 #include <sys/time.h>
15
16 #include <xcb/xcb.h>
17
18 #include <X11/Xlib.h>
19
20 static double
21 get_time(void)
22 {
23     struct timeval timev;
24     gettimeofday(&timev, NULL);
25
26     return (double) timev.tv_sec + (((double)timev.tv_usec) / 1000000);
27 }
28
29 int
30 main(void)
31 {
32     unsigned int i;
33     double start, end, diff;
34
35     /* Number of InterAtom X requests sent for performance
36        measurement */
37     const int count = 500;
38
39     /* Init names */
40     char *names[count];
41     for(i = 0; i < count; ++i)

```



```

    asprintf(names + i, "NAME%d", i);
43
xcb_connection_t *conn = xcb_connect(NULL, NULL);
44
45 /* Beginning of XCB synchronous method (wrong usage!) */
46 xcb_atom_t atoms[count];
47 start = get_time();
48
49 for(i = 0; i < count; ++i)
50     atoms[i] = xcb_intern_atom_reply(conn,
51                                     xcb_intern_atom(conn, 0,
52                                                         strlen(names[i]),
53                                                         names[i]),
54                                                         NULL)->atom;
55
56 end = get_time();
57
58 printf("=>XCB_wrong_usage_(synchronous)\n");
59 printf("Duration:_%fs\n\n", end - start);
60 diff = end - start;
61 /* End of XCB synchronous method */
62
63 /* Beginning of XCB asynchronous method (good usage!) */
64 xcb_intern_atom_cookie_t atom_cookies[count];
65 xcb_atom_t atoms_xcb[count];
66 start = get_time();
67
68 /* Send the request */
69 for(i = 0; i < count; ++i)
70     atom_cookies[i] = xcb_intern_atom(conn, 0, strlen(names[i]), names
71                                     [i]);
72
73 /* Now get the replies */
74 for(i = 0; i < count; ++i)
75     {
76         xcb_intern_atom_reply_t *r = xcb_intern_atom_reply(conn,
77                                                         atom_cookies[
78                                                         i],
79                                                         0);
80
81         if(r)
82             atoms_xcb[i] = r->atom;
83
84         free(r);
85     }
86
87 end = get_time ();
88
89 printf("=>XCB_bad_usage_(asynchronous)\n");
90 printf("Duration:_%fs\n", end - start);
91 printf("Ratio:_%%.2f\n\n", diff / (end - start));
92 diff = end - start;
93 /* End of XCB asynchronous method */
94
95 /* Close XCB connection to the X server */

```

```

95  xcb_disconnect(conn);
97  /* Xlib method
99     Note: one could argue that the library provides XinternAtoms()
101     function which sends the atoms asynchronously, but
103     asynchronous method is quite complicated to use in Xlib, in
105     addition, this is not the purpose of this program */
107  Display *dpy = XOpenDisplay(NULL);
109  Atom atoms_xlib[count];
111  start = get_time();
113  for (i = 0; i < count; ++i)
115     atoms_xlib[i] = XInternAtom(dpy, names[i], 0);
117  end = get_time();
119  printf("=>_Xlib_traditional_usage\n");
121  printf("Duration_:_%fs\n", end - start);
123  printf("Ratio____:_%2f\n", (end - start) / diff);
125  /* End of Xlib method */
127  /* Free variables */
129  for (i = 0; i < count; ++i)
131     free(names[i]);
133  /* Close the Xlib connection to the X server */
135  XCloseDisplay(dpy);
137  return EXIT_SUCCESS;
139 }

```

Listing A.3: Performance comparison between Xlib and XCB

Appendix B

Patches

B.1 Awesome patches

B.1.1 Replace Pango by Xft to handle client-side fonts

```
commit 542a944361e10e1aa4976e10814f0aa51818a21e
2 Author: Arnaud Fontaine <arnau@debian.org>
Date: Mon Mar 17 11:38:49 2008 +0000
4
    Use Pango for fonts instead of Xft (which hasn't been ported yet
    to
6    XCB) to measure text.
8 diff --git a/Makefile.am b/Makefile.am
index 941dd83..55ccfc2 100644
10 --- a/Makefile.am
+++ b/Makefile.am
12 @@ -107,7 +107,7 @@ AWESOME_CFLAGS = -std=gnu99 -pipe \
    -Wunused -Winit-self -Wpointer-arith -Wredundant-
    decls \
14    -Wmissing-prototypes -Wmissing-format-attribute -
    Wmissing-noreturn
    endif
16 -AM_CPPFLAGS = $(XFT_CFLAGS) $(X_CFLAGS) $(CAIRO_CFLAGS) $(
    CONFUSE_CFLAGS) $(XRANDR_CFLAGS) $(XINERAMA_CFLAGS) $(
    AWESOME_CFLAGS)
+AM_CPPFLAGS = $(X_CFLAGS) $(PANGOCAIRO_CFLAGS) $(CONFUSE_CFLAGS) $(
    XRANDR_CFLAGS) $(XINERAMA_CFLAGS) $(AWESOME_CFLAGS)
18
    bin_PROGRAMS += awesome
20    awesome_SOURCES = \
@@ -140,7 +140,7 @@ awesome_SOURCES = \
22    ewmh.c ewmh.h
    awesome_SOURCES += $(LAYOUTS)
24    awesome_SOURCES += $(WIDGETS)
-awesome_LDADD = $(XFT_LIBS) $(X_LIBS) $(CAIRO_LIBS) $(CONFUSE_LIBS) $(
    XRANDR_LIBS) $(XINERAMA_LIBS)
26 +awesome_LDADD = $(X_LIBS) $(PANGOCAIRO_LIBS) $(CONFUSE_LIBS) $(
    XRANDR_LIBS) $(XINERAMA_LIBS)
```

```

28 bin_PROGRAMS += awesome-client
   awesome_client_SOURCES = \
30 @@ -159,7 +159,7 @@ awesome_message_SOURCES = \
       common/xscreen.h common/xscreen.c \
32       awesome-message.c

34 -awesome_message_LDADD = $(XFT_LIBS) $(X_LIBS) $(CAIRO_LIBS) $(
       CONFUSE_LIBS) $(XINERAMA_LIBS)
   +awesome_message_LDADD = $(X_LIBS) $(PANGOCAIRO_LIBS) $(CONFUSE_LIBS)
       $(XINERAMA_LIBS)

36 bin_PROGRAMS += awesome-menu
38 awesome_menu_SOURCES = \
@@ -171,7 +171,7 @@ awesome_menu_SOURCES = \
40       common/xutil.h common/xutil.c \
       awesome-menu.c

42 -awesome_menu_LDADD = $(XFT_LIBS) $(X_LIBS) $(CAIRO_LIBS) $(
       CONFUSE_LIBS) $(XINERAMA_LIBS)
44 +awesome_menu_LDADD = $(X_LIBS) $(PANGOCAIRO_LIBS) $(CONFUSE_LIBS) $(
       XINERAMA_LIBS)

46 if HAVE_XMLTO
   if HAVE_ASCIIDOC
48 diff --git a/README b/README
   index b50e97c..0871992 100644
50 --- a/README
   +++ b/README
52 @@ -5,7 +5,7 @@ awesome is an extremely fast, small, and dynamic
       window manager for X.
       Requirements
54 -----
       In order to build awesome itself, you need header files and libs of:
56 - - Xlib, Xinerama, Xrandr, Xft
   + - Xlib, Xinerama, Xrandr, Pango
58   - libconfuse >= 2.6
     - cairo

60 diff --git a/awesomerc.5.txt b/awesomerc.5.txt
62 index b03c5bd..9c0dcda 100644
   --- a/awesomerc.5.txt
   +++ b/awesomerc.5.txt
64 @@ -524,7 +524,7 @@ Note: when there is no whitespace, quotes are
       optional.
66 <boolean>         -> "true" or "false"
   <color>          -> #ff9933 (hexadecimal color notation: #red green
       blue)
68 <float>           -> 0.3, 0.8 (often values between 0 and 1 are useful)
   <font>           -> Xft font: mono-10, fixed-12, sans-8, ...
70 +<font>          -> Pango font: sans 10, sans italic 10, fixed 12, ...
   <identifier>    -> foobar (choose a name/string)
72 <image>          -> "/home/awesome/pics/icon.png" (path to image)
   <integer>       -> 1, 10, -3 (positive numbers are required mostly)
74 diff --git a/awesomerc.in b/awesomerc.in
   index bfa2d0f..34136e0 100644

```

```

76 --- a/awesomerc.in
+++ b/awesomerc.in
78 @@ -4,7 +4,7 @@ screen 0
    {
80         normal
            {
82 -             font = "vera-10"
+             font = "vera 10"
84             fg = "#eeeeee"
             bg = "#111111"
86             border = "#6666ff"
diff --git a/common/draw.c b/common/draw.c
88 index 275b140..b8a3d4b 100644
--- a/common/draw.c
90 +++ b/common/draw.c
@@ -20,7 +20,6 @@
92     */
94     #include <cairo.h>
-#include <cairo-ft.h>
96     #include <cairo-xlib.h>
98     #include <langinfo.h>
@@ -99,6 +98,7 @@ draw_context_new(Display *disp, int phys_screen, int
    width, int height, Drawable
100     d->drawable = dw;
    d->surface = cairo_xlib_surface_create(disp, dw, d->visual, width
        , height);
102     d->cr = cairo_create(d->surface);
+    d->layout = pango_cairo_create_layout(d->cr);
104
    return d;
106 };
@@ -109,11 +109,70 @@ draw_context_new(Display *disp, int phys_screen,
    int width, int height, Drawable
108     void
    draw_context_delete(DrawCtx *ctx)
110     {
+    g_object_unref(ctx->layout);
112     cairo_surface_destroy(ctx->surface);
    cairo_destroy(ctx->cr);
114     p_delete(&ctx);
    }
116
+/** Create a new Pango font
118 + * \param disp Display ref
+ * \param fontname Pango fontname (e.g. [FAMILY-LIST] [STYLE-OPTIONS]
+ [SIZE])
120 + */
+font_t *
122 +draw_font_new(Display *disp, char *fontname)
+{
124 +    cairo_surface_t *surface;
+    cairo_t *cr;
126 +    PangoLayout *layout;
+    font_t *font = p_new(font_t, 1);

```

```

128 +   PangoContext *context;
129 +   PangoFontMetrics *font_metrics;
130 +
131 +   /* Create a dummy cairo surface, cairo context and pango layout
in
132 +   * order to get font informations */
133 +   surface = cairo_xlib_surface_create (disp,
134 +                                       DefaultScreen (disp),
135 +                                       DefaultVisual (disp),
136 +                                       DefaultScreen (disp)),
137 +                                       DisplayWidth (disp,
138 +                                       DefaultScreen (disp)),
139 +                                       DisplayHeight (disp,
140 +                                       DefaultScreen (disp)));
141 +
142 +   cr = cairo_create (surface);
143 +   layout = pango_cairo_create_layout (cr);
144 +
145 +   /* Get the font description used to set text on a PangoLayout */
146 +   font->desc = pango_font_description_from_string (fontname);
147 +   pango_layout_set_font_description (layout, font->desc);
148 +
149 +   /* Get height */
150 +   pango_layout_get_pixel_size (layout, NULL, &font->height);
151 +
152 +   /* Get ascent and descent */
153 +   context = pango_layout_get_context (layout);
154 +   font_metrics = pango_context_get_metrics (context, font->desc,
155 +   NULL);
156 +
157 +   /* Values in PangoFontMetrics are given in Pango units */
158 +   font->ascent = PANGO_PIXELS (pango_font_metrics_get_ascent (
159 +   font_metrics));
160 +   font->descent = PANGO_PIXELS (pango_font_metrics_get_descent (
161 +   font_metrics));
162 +
163 +   pango_font_metrics_unref (font_metrics);
164 +   g_object_unref (layout);
165 +   cairo_destroy (cr);
166 +   cairo_surface_destroy (surface);
167 +
168 +   return font;
169 +}
170 +
171 +/** Delete a font
172 + * \param font font_t to delete
173 + */
174 +void
175 +draw_font_free (font_t *font)
176 +{
177 +   pango_font_description_free (font->desc);
178 +   p_delete (&font);
179 +}
180 +
181 +/** Draw text into a draw context
182 + * \param ctx DrawCtx to draw to

```

```

178 * \param area area to draw to
@@ -136,7 +195,6 @@ draw_text(DrawCtx *ctx ,
    int nw = 0, x, y;
180    ssize_t len, olen;
    char *buf = NULL, *utf8 = NULL;
182 -    cairo_font_face_t *font_face;

184    draw_rectangle(ctx, area, True, style.bg);

186 @@ -175,12 +233,11 @@ draw_text(DrawCtx *ctx ,
    buf[len - 3] = '.';
188 }

190 -    font_face = cairo_ft_font_face_create_for_pattern(style.font->
pattern);
-    cairo_set_font_face(ctx->cr, font_face);
192 -    cairo_set_font_size(ctx->cr, style.font->height);
+    pango_layout_set_text(ctx->layout, text, -1);
194 +    pango_layout_set_font_description(ctx->layout, style.font->desc);

196    x = area.x + padding;
-    y = area.y + style.font->ascent + (ctx->height - style.font->
height) / 2;
198 +    y = area.y + (ctx->height - style.font->height) / 2;

200    switch(aligned)
    {
202 @@ -201,7 +258,8 @@ draw_text(DrawCtx *ctx ,
                                style.shadow.green / 65535.0,
204                                style.shadow.blue / 65535.0);
        cairo_move_to(ctx->cr, x + style.shadow_offset, y + style.
shadow_offset);
206 -    cairo_show_text(ctx->cr, buf);
+    pango_cairo_update_layout(ctx->cr, ctx->layout);
208 +    pango_cairo_show_layout(ctx->cr, ctx->layout);
    }

210    cairo_set_source_rgb(ctx->cr,
212 @@ -209,9 +267,8 @@ draw_text(DrawCtx *ctx ,
                                style.fg.green / 65535.0,
214                                style.fg.blue / 65535.0);
        cairo_move_to(ctx->cr, x, y);
216 -    cairo_show_text(ctx->cr, buf);
-
218 -    cairo_font_face_destroy(font_face);
+    pango_cairo_update_layout(ctx->cr, ctx->layout);
220 +    pango_cairo_show_layout(ctx->cr, ctx->layout);

222    p_delete(&buf);
    }
224 @@ -597,12 +654,12 @@ draw_rotate(DrawCtx *ctx, int phys_screen,
    double angle, int tx, int ty)
    * \return text width
226    */
    unsigned short
228 -draw_textwidth(Display *disp, XftFont *font, char *text)

```

```

+draw_textwidth(Display *disp, font_t *font, char *text)
230 {
    cairo_surface_t *surface;
232     cairo_t *cr;
    -     cairo_font_face_t *font_face;
234     -     cairo_text_extents_t te;
    +     PangoLayout *layout;
236     +     PangoRectangle ext;

    if (!a_strlen(text))
        return 0;
240 @@ -612,15 +669,15 @@ draw_textwidth(Display *disp, XftFont *font,
    char *text)
                                                DisplayWidth(disp,
                                                    DefaultScreen(disp)),
242                                                DisplayHeight(disp,
                                                    DefaultScreen(disp));

    cr = cairo_create(surface);
244     -     font_face = cairo_ft_font_face_create_for_pattern(font->pattern);
    -     cairo_set_font_face(cr, font_face);
246     -     cairo_set_font_size(cr, font->height);
    -     cairo_text_extents(cr, text, &te);
248     +     layout = pango_cairo_create_layout(cr);
    +     pango_layout_set_text(layout, text, -1);
250     +     pango_layout_set_font_description(layout, font->desc);
    +     pango_layout_get_pixel_extents(layout, NULL, &ext);
252     +     g_object_unref(layout);
    cairo_destroy(cr);
254     cairo_surface_destroy(surface);
    -     cairo_font_face_destroy(font_face);
256
    -     return MAX(te.x_advance, te.width);
258     +     return ext.width;
    }

260     /** Transform a string to a Alignment type.
262 @@ -683,7 +740,7 @@ draw_style_init(Display *disp, int phys_screen,
    cfg_t *cfg,
        return;

264
    if((buf = cfg_getstr(cfg, "font")))
266     -     c->font = XftFontOpenName(disp, phys_screen, buf);
    +     c->font = draw_font_new(disp, buf);
268

    draw_color_new(disp, phys_screen,
270                    cfg_getstr(cfg, "fg"), &c->fg);
diff --git a/common/draw.h b/common/draw.h
272 index d86fea5..2be2cad 100644
--- a/common/draw.h
274 +++ b/common/draw.h
@@ -27,7 +27,7 @@
276     #include <confuse.h>

278     #include <X11/Xlib.h>
    -#include <X11/Xft/Xft.h>
280     +#include <pango/pangocairo.h>

```



```

282 #include "common/util.h"
    #include "common/list.h"
284 @@ -83,6 +83,14 @@ area_get_intersect_area(area_t a, area_t b)

286 typedef struct
    {
288 +   PangoFontDescription *desc;
    +   int height;
290 +   int ascent;
    +   int descent;
292 +} font_t;
    +
294 +typedef struct
    +{
296     /** Foreground color */
    XColor fg;
298     /** Background color */
    @@ -94,7 +102,7 @@ typedef struct
300     /** Shadow offset */
    int shadow_offset;
302     /** Font */
    -   XftFont *font;
304 +   font_t *font;
    } style_t;
306
    typedef struct
308 @@ -108,11 +116,14 @@ typedef struct
    int depth;
310     cairo_t *cr;
    cairo_surface_t *surface;
312 +   PangoLayout *layout;
    } DrawCtx;
314
    DrawCtx *draw_context_new(Display *, int, int, int, Drawable);
316 void draw_context_delete(DrawCtx *);

318 +font_t *draw_font_new(Display *disp, char *fontname);
    +void draw_font_free(font_t *);
320 void draw_text(DrawCtx *, area_t, Alignment, int, char *, style_t);
    void draw_rectangle(DrawCtx *, area_t, Bool, XColor);
322 void draw_rectangle_gradient(DrawCtx *, area_t, Bool, area_t, XColor
    *, XColor *, XColor *);
    @@ -125,7 +136,7 @@ void draw_image(DrawCtx *, int, int, int, const
    char *);
324 void draw_image_from_argb_data(DrawCtx *, int, int, int, int, int,
    unsigned char *);
    area_t draw_get_image_size(const char *filename);
326 Drawable draw_rotate(DrawCtx *, int, double, int, int);
    -unsigned short draw_textwidth(Display *, XftFont *, char *);
328 +unsigned short draw_textwidth(Display *, font_t *, char *);
    Alignment draw_get_align(const char *);
330 Bool draw_color_new(Display *, int, const char *, XColor *);
    void draw_style_init(Display *, int, cfg_t *, style_t *, style_t *);
332 diff --git a/configure.ac b/configure.ac
    index 06e6006..94ea525 100644

```

```

334 --- a/configure.ac
+++ b/configure.ac
336 @@ -106,12 +106,10 @@ AC_DEFINE_UNQUOTED([AWESOME_COMPILE_BY], ["
    Saw_whoami"], [build user])

338 # Checks for libraries.
    AC_PATH_XTRA
340 -PKG_CHECK_MODULES([CAIRO], [cairo],,
    - [AC_MSG_ERROR([awesome requires cairo.])])
342 +PKG_CHECK_MODULES([PANGOCAIRO], [pangocairo],,
    + [AC_MSG_ERROR([awesome requires pangocairo.])])
344 PKG_CHECK_MODULES([CONFUSE], [libconfuse >= 2.6],,
    [AC_MSG_ERROR([awesome requires libconfuse >= 2.6.]])
346 -PKG_CHECK_MODULES([XFT], [xft],,
    - [AC_MSG_ERROR([awesome requires xft.])])
348 PKG_CHECK_MODULES([XINERAMA], [xinerama],,
    [AC_MSG_ERROR([awesome requires Xinerama.])])
350 PKG_CHECK_MODULES([XRANDR], [xrandr],,
diff --git a/widgets/textbox.c b/widgets/textbox.c
352 index 5a85a55..87652cf 100644
--- a/widgets/textbox.c
354 +++ b/widgets/textbox.c
@@ -65,7 +65,7 @@ static widget_tell_status_t
356 textbox_tell(Widget *widget, char *property, char *command)
    {
358     Data *d = widget->data;
    - XftFont *newfont;
360 + font_t *newfont;

362     if(!a_strcmp(property, "text"))
    {
364 @@ -88,11 +88,10 @@ textbox_tell(Widget *widget, char *property, char
    *command)
        return WIDGET_ERROR_FORMAT_COLOR;
366     else if(!a_strcmp(property, "font"))
    {
368 -         if((newfont = XftFontOpenName(globalconf.display,
    -                                     get_phys_screen(widget->
    statusbar->screen), command)))
370 +         if((newfont = draw_font_new(globalconf.display, command)))
    {
372             if(d->style.font != globalconf.screens[widget->statusbar
    ->screen].styles.normal.font)
    - XftFontClose(globalconf.display, d->style.font);
374 + draw_font_free(d->style.font);
    + d->style.font = newfont;
376     }
    else

```

Listing B.1: *[PATCH] Replace Pango by xft to handle client-side fonts #1*

```

1 commit f75f16c3251575da9bc30e9826118872ec0d73cf
   Author: Arnaud Fontaine <arnau@debian.org>
3 Date: Mon Mar 17 16:55:38 2008 +0000

5 Don't get ascent/descent informations about a font because it's

```

```

    not
    useful at the moment (commented out).
7
diff --git a/common/draw.c b/common/draw.c
9 index b8a3d4b..e19ba94 100644
--- a/common/draw.c
11 +++ b/common/draw.c
@@ -126,8 +126,6 @@ draw_font_new(Display *disp, char *fontname)
13     cairo_t *cr;
    PangoLayout *layout;
15     font_t *font = p_new(font_t, 1);
    PangoContext *context;
17     PangoFontMetrics *font_metrics;

19     /* Create a dummy cairo surface, cairo context and pango layout
    in
    * order to get font informations */
21 @@ -147,6 +145,12 @@ draw_font_new(Display *disp, char *fontname)
    /* Get height */
23     pango_layout_get_pixel_size(layout, NULL, &font->height);

25 + /* At the moment, we don't need ascent/descent but maybe it could
    + * be useful in the future... */
27 + #if 0
    + PangoContext *context;
29 + PangoFontMetrics *font_metrics;
    +

31     /* Get ascent and descent */
    context = pango_layout_get_context(layout);
33     font_metrics = pango_context_get_metrics(context, font->desc,
    NULL);
@@ -156,6 +160,8 @@ draw_font_new(Display *disp, char *fontname)
35     font->descent = PANGO_PIXELS(pango_font_metrics_get_descent(
    font_metrics));

37     pango_font_metrics_unref(font_metrics);
    + #endif
39 +
    g_object_unref(layout);
41     cairo_destroy(cr);
    cairo_surface_destroy(surface);
43 diff --git a/common/draw.h b/common/draw.h
    index 2be2cad..18811a7 100644
45 --- a/common/draw.h
    +++ b/common/draw.h
47 @@ -85,8 +85,6 @@ typedef struct
    {
49     PangoFontDescription *desc;
    int height;
51     int ascent;
    int descent;
53 } font_t;

55 typedef struct

```

Listing B.2: [PATCH] Replace Pango by xft to handle client-side fonts #2

B.1.2 Fix incorrect Xlib function call

```
1 commit 7a2b851a03047377c9eb4932ec1c785ad49a14c5
   Author: Arnaud Fontaine <arnaud@andesi.org>
3   Date:   Mon Jan 7 18:57:25 2008 +0100

5       fix XGetTransientForHint() call

7       Signed-off-by: Julien Danjou <julien@danjou.info>

9   diff --git a/client.c b/client.c
   index a31f774..54670da 100644
11  --- a/client.c
   +++ b/client.c
13  @@ -307,7 +307,7 @@ client_manage(Window w, XWindowAttributes *wa, int
       screen)
   {
15     Client *c, *t = NULL;
       Window trans;
17     - Status rettrans;
   + Bool rettrans;
19     XWindowChanges wc;
       Area area, darea;
21     Tag *tag;
   @@ -388,8 +388,10 @@ client_manage(Window w, XWindowAttributes *wa,
       int screen)
23     /* grab buttons */
       window_grabbuttons(phys_screen, c->win, False, True);
25
   - /* check for transient and set tags like its parent */
27     - if((rettrans = XGetTransientForHint(globalconf.display, w, &trans)
       ) == Success)
   + /* check for transient and set tags like its parent,
29     + * XGetTransientForHint returns 1 on success
   + */
31     + if((rettrans = XGetTransientForHint(globalconf.display, w, &trans)
       ))
       && (t = get_client_bywin(globalconf.clients, trans)))
33         for(tag = globalconf.screens[c->screen].tags; tag; tag = tag
           ->next)
           if(is_client_tagged(t, tag))
35   @@ -397,7 +399,7 @@ client_manage(Window w, XWindowAttributes *wa, int
       screen)

37     /* should be floating if transient or fixed */
       if(!c->isfloating)
39     -     c->isfloating = (rettrans == Success) || c->isfixed;
   +     c->isfloating = rettrans || c->isfixed;
41
       /* save new props */
43     client_saveprops(c);
```

Listing B.3: [PATCH] Fix incorrect Xlib XGetTransientForHint() call

B.1.3 Fix PATH_MAX issue

```
1 commit fc9e31ff62a176d4722fbf40ff21e7dd2d53da73
   Author: Arnaud Fontaine <arnaud@andesi.org>
3   Date: Thu Mar 13 15:11:59 2008 +0100

5   get rid of PATH_MAX

7   I replaced stack memory allocations with PATH_MAX by heap
   memory
   allocations on post-2.2 branch because PATH_MAX isn't necessary
   defined
9   according to POSIX specification. For instance GNU/Hurd doesn't
   have
   PATH size restriction, thus doesn't defined PATH_MAX and
   compilation
11  will fail.

13  Signed-off-by: Julien Danjou <julien@danjou.info>

15  diff --git a/awesome-menu.c b/awesome-menu.c
   index 9cbaeff..d79490c 100644
17  --- a/awesome-menu.c
   +++ b/awesome-menu.c
19  @@ -19,7 +19,9 @@
   *
21  */

23  +/* getline(), asprintf() */
   #define _GNU_SOURCE
25  +
   #include <getopt.h>
27
   #include <signal.h>
29  @@ -28,6 +30,7 @@
   #include <dirent.h>
31  #include <pwd.h>
   #include <sys/types.h>
33  +#include <string.h>

35  #include <confuse.h>

37  @@ -222,7 +225,7 @@ get_last_word(char *text)
   static Bool
39  item_list_fill_file(const char *directory)
   {
41  -   char cwd[PATH_MAX], *home, *user, *filename;
   +   char *cwd, *home, *user, *filename;
43  +   const char *file;
   +   DIR *dir;
45  +   struct dirent *dirinfo;
   @@ -234,14 +237,13 @@ item_list_fill_file(const char *directory)
   item_list_wipe(&globalconf.items);

47
   if(!directory)
49  -   a_strcpy(cwd, sizeof(cwd), "./");
```

```

51 +     cwd = a_strdup("./");
    else if(a_strlen(directory) > 1 && directory[0] == '~')
53 {
    if(directory[1] == '/')
55 {
    if((home = getenv("HOME")))
    a_strcpy(cwd, sizeof(cwd), home);
57 a_strcat(cwd, sizeof(cwd), directory + 1);
    home = getenv("HOME");
59 +     asprintf(&cwd, "%s%s", (home ? home : ""), directory + 1)
    +     ;
61     }
    else
63 {
@@ -252,8 +254,7 @@ item_list_fill_file(const char *directory)
65     a_strncpy(user, len, directory + 1, (file - directory) -
        1);
    if((passwd = getpwnam(user))
67 {
    a_strcpy(cwd, sizeof(cwd), passwd->pw_dir);
    a_strcat(cwd, sizeof(cwd), file);
69 +     asprintf(&cwd, "%s%s", passwd->pw_dir, file);
71     p_delete(&user);
    }
73     else
@@ -264,10 +265,13 @@ item_list_fill_file(const char *directory)
75     }
    }
77     else
    a_strcpy(cwd, sizeof(cwd), directory);
79 +     cwd = a_strdup(directory);

    if(!(dir = opendir(cwd)))
81 +     {
83 +         p_delete(&cwd);
            return False;
85 +     }

    while((dirinfo = readdir(dir))
87 {
@@ -296,6 +300,7 @@ item_list_fill_file(const char *directory)
89 {
91     closedir(dir);
93 +     p_delete(&cwd);

    return True;
95 }
97 @@ -568,25 +573,31 @@ handle_kpress(XKeyEvent *e)
    static Bool
99 item_list_fill_stdin(void)
    {
101 -     char buf[PATH_MAX];
    +     char *buf = NULL;
103 +     size_t len = 0;
    +     ssize_t line_len;

```

```

105 +
    item_t *newitem;
107 Bool has_entry = False;

109     item_list_init(&globalconf.items);

111 -     if(fgets(buf, sizeof(buf), stdin))
+     if((line_len = getline(&buf, &len, stdin)) != -1)
113         has_entry = True;

115     if(has_entry)
        do
117         {
-         buf[a_strlen(buf) - 1] = '\0';
119 +         buf[line_len - 1] = '\0';
            newitem = p_new(item_t, 1);
121            newitem->data = a_strdup(buf);
            newitem->match = True;
123            item_list_append(&globalconf.items, newitem);
        }

125 -     while(fgets(buf, sizeof(buf), stdin));
+     while((line_len = getline(&buf, &len, stdin)) != -1);
127 +
+     if(buf)
129 +         p_delete(&buf);

131     return has_entry;
    }
133 diff --git a/uicb.c b/uicb.c
index 0a94a55..63fadbf 100644
135 --- a/uicb.c
+++ b/uicb.c
137 @@ -23,7 +23,11 @@
    * @defgroup ui_callback User Interface Callbacks
139     */

141 +/* strdup() */
+#define _GNU_SOURCE
143 +
+ #include <sys/wait.h>
145 + #include <string.h>

147 #include "awesome.h"
+ #include "tag.h"
149 @@ -45,10 +49,11 @@ extern AwesomeConf globalconf;
    * \ingroup ui_callback
151     */
    void
153 -uicb_exec(int screen __attribute__((unused)), char *arg)
+uicb_exec(int screen __attribute__((unused)), char *cmd)
155     {
        Client *c;
157 -        char path[PATH_MAX];
+        int args_pos;
159 +        char *args, *path;

```

```

161     /* remap all clients since some WM won't handle them otherwise */
162     for(c = globalconf.clients; c; c = c->next)
163 @@ -59,8 +64,21 @@ uicb_exec(int screen __attribute__((unused)), char
    *arg)
    if(globalconf.display)
165         close(ConnectionNumber(globalconf.display));

167 -     sscanf(arg, "%s", path);
168 -     execlp(path, arg, NULL);
169 +     /* Ignore the leading spaces if any */
170 +     for(args_pos = 0;
171 +         args_pos < strlen(cmd) && cmd[args_pos] == ' ';
172 +         ++args_pos);
173 +
174 +     /* Get the beginning of the arguments */
175 +     if((args = strchr(cmd + args_pos, ' ')) == NULL)
176 +     {
177 +         warn("Invalid command %s\n", cmd);
178 +         return;
179 +     }
180 +
181 +     path = strdup(cmd + args_pos, args - (cmd + args_pos));
182 +     execlp(path, cmd, NULL);
183 +     p_delete(&path);
184     }
185
186 /** Spawn another process

```

Listing B.4: [PATCH] Get rid of `PATH_MAX` #1

```

commit 77dfdd29286fd5c6bf8cbc5e918863a00378c9b7
2 Author: Arnaud Fontaine <arnaud@andesi.org>
Date: Tue Mar 18 17:25:57 2008 +0100
4
6 Remove PATH_MAX usage from awesome-menu
8
10 Signed-off-by: Julien Danjou <julien@danjou.info>
12
14 diff --git a/awesome-menu.c b/awesome-menu.c
index cc20528..92f3ab3 100644
--- a/awesome-menu.c
12 +++ b/awesome-menu.c
@@ -22,6 +22,8 @@
14     /* getline(), asprintf() */
    #define _GNU_SOURCE
16
18     #define CHUNK_SIZE 4096
18 +
20     #include <getopt.h>
20
22     #include <signal.h>
22 @@ -93,7 +95,9 @@ typedef struct
    /** Numlock mask */
24     unsigned int numlockmask;
    /** The text */
26 -     char text[PATH_MAX];

```



```

+   char *text;
28 +   /** The text length */
+   size_t text_size;
30   /** Item list */
+   item_t *items;
32   /** Selected item */
@@ -453,6 +457,7 @@ handle_kpress(XKeyEvent *e)
34   KeySym ksym;
+   int num;
36   ssize_t len;
+   size_t text_dst_len;
38
+   len = a_strlen(globalconf.text);
40   num = XLookupString(e, buf, sizeof(buf), &ksym, 0);
@@ -516,6 +521,14 @@ handle_kpress(XKeyEvent *e)
42       if(buf[0] != '/' || globalconf.text[len - 1] != '/')
+       {
44           buf[num] = '\0';
+
+       +
46 +           /* Reallocate text string if needed to hold
+       +           * concatenation of text and buf */
48 +           if((text_dst_len = (a_strlen(globalconf.text) + num -
+       +           1)) > globalconf.text_size)
+       +       {
50 +           globalconf.text_size += ((int) (text_dst_len /
+       globalconf.text_size)) * CHUNK_SIZE;
+       +           p_realloc(&globalconf.text, globalconf.text_size)
+       ;
52 +       }
+       a_strncat(globalconf.text, sizeof(globalconf.text),
+       buf, num);
54     }
+     compute_match(get_last_word(globalconf.text));
56 @@ -677,6 +690,12 @@ main(int argc, char **argv)
+     if(!item_list_fill_stdin())
58         item_list_fill_file(NULL);
60 +     /* Allocate a default size for the text on the heap instead of
+     +     * using stack allocation with PATH_MAX (may not be defined
62 +     +     * according to POSIX). This string size may be increased if
+     +     * needed */
64 +     globalconf.text = p_new(char, CHUNK_SIZE);
+     globalconf.text_size = CHUNK_SIZE;
66     compute_match(NULL);
68     for(opt = 1000; opt; opt--)
@@ -727,6 +746,7 @@ main(int argc, char **argv)
70     }
72
+     p_delete(&globalconf.text);
74     draw_context_delete(globalconf.ctx);
+     simplewindow_delete(globalconf.sw);
76     XCloseDisplay disp);

```

Listing B.5: [PATCH] Get rid of `PATH_MAX` #2

B.2 xcb-util patches

B.2.1 Add non-blocking events loop

```
commit 66728b2d13f3eee94230038c8f8eb236c20b9525
2 Author: Arnaud Fontaine <arnaud@andesi.org>
Date: Mon Jan 28 14:26:05 2008 -0800
4
6 Add xcb_poll_for_event_loop and rename xcb_event_loop to match.
8
10 I'm currently porting Awesome[0] from Xlib to XCB as a school
    project for
12 my bachelor. I discussed with Vincent about adding a non-blocking
    xcb_event_loop on IRC because I had to write one for Awesome and
    wondered
14 if this kind of function could be added to xcb-util.
16
18 [0] http://awesome.naquadah.org
20
22 Signed-off-by: Jamey Sharp <jamey@minilop.net>
24
26 diff --git a/event/events.c b/event/events.c
index 2900bd3..354346a 100644
28 --- a/event/events.c
+++ b/event/events.c
30 @@ -62,7 +62,7 @@ static int handle_event(xcb_event_handlers_t *
    evenths, xcb_generic_event_t *event
32     return 0;
34 }
36
38 -void xcb_event_loop(xcb_event_handlers_t *evenths)
39 +void xcb_wait_for_event_loop(xcb_event_handlers_t *evenths)
40 {
42     xcb_generic_event_t *event;
44     while((event = xcb_wait_for_event(evenths->c)))
46 @@ -72,6 +72,16 @@ void xcb_event_loop(xcb_event_handlers_t *evenths)
    }
48 }
50
52 +void xcb_poll_for_event_loop(xcb_event_handlers_t *evenths)
53 +{
54 +    xcb_generic_event_t *event;
56 +    while ((event = xcb_poll_for_event(evenths->c)))
58 +    {
59 +        handle_event(evenths, event);
60 +        free(event);
62 +    }
64 +}
66
68 +static void set_handler(xcb_generic_event_handler_t handler, void *
    data, xcb_event_handler_t *place)
70 {
72     xcb_event_handler_t eventh = { handler, data };
74 diff --git a/event/xcb_event.h b/event/xcb_event.h
index deb7ba8..9122893 100644
```

```

48 — a/event/xcb_event.h
+++ b/event/xcb_event.h
50 @@ -14,7 +14,8 @@ xcb_event_handlers_t *xcb_alloc_event_handlers(
    xcb_connection_t *c);
    void xcb_free_event_handlers(xcb_event_handlers_t *evenths);
52 xcb_connection_t *xcb_get_xcb_connection(xcb_event_handlers_t *
    evenths);

54 -void xcb_event_loop(xcb_event_handlers_t *evenths);
+void xcb_wait_for_event_loop(xcb_event_handlers_t *evenths);
56 +void xcb_poll_for_event_loop(xcb_event_handlers_t *evenths);

58 typedef int (*xcb_generic_event_handler_t)(void *data,
    xcb_connection_t *c, xcb_generic_event_t *event);
typedef int (*xcb_generic_error_handler_t)(void *data,
    xcb_connection_t *c, xcb_generic_error_t *error);
60 diff —git a/wm/xcbwmm-test.c b/wm/xcbwmm-test.c
index 4d8be5d..c928d7a 100644
62 — a/wm/xcbwmm-test.c
+++ b/wm/xcbwmm-test.c
64 @@ -205,7 +205,7 @@ int main(int argc, char **argv)

66     if(TEST_THREADS)
68     {
-         pthread_create(&event_thread, 0, (void (*)(void *))
xcb_event_loop, evenths);
+         pthread_create(&event_thread, 0, (void (*)(void *))
xcb_wait_for_event_loop, evenths);
70     }

72     root = xcb_aux_get_screen(c, screen_nbr)->root;
@@ -223,7 +223,7 @@ int main(int argc, char **argv)
74     if(TEST_THREADS)
         pthread_join(event_thread, 0);
76     else
-         xcb_event_loop(evenths);
78 +         xcb_wait_for_event_loop(evenths);

80     exit(0);
    /*NOTREACHED*/

```

Listing B.6: [PATCH] Add non-blocking events loop

B.2.2 Fix invalid data returned by ICCCM hints functions

```

1 commit e80aa98cc302e69025493631ffe03fc3f0516c8e
  Author: Arnaud Fontaine <arnau@debian.org>
3 Date: Sun Mar 25 23:16:04 2008 +0100

5     Fix invalid value returned by xcb_get_wm_hints() and
  xcb_get_size_hints() and also make this API consistent. Both now
7     returns a pointer on hints structures newly allocated and freed
  via specific destructors.
9

```

```

diff --git a/icccm/icccm.c b/icccm/icccm.c
11 index cf22a25..aa347a1 100644
--- a/icccm/icccm.c
13 +++ b/icccm/icccm.c
@@ -515,45 +515,46 @@ xcb_set_wm_size_hints (xcb_connection_t *c,
15     xcb_change_property(c, XCB_PROP_MODE_REPLACE, window, property
        , WM_SIZE_HINTS, 32, sizeof(*hints) / 4, hints);
17     }
18
19 -int
20 +xcb_size_hints_t *
21     xcb_get_wm_size_hints (xcb_connection_t *c,
22                           xcb_window_t      window,
23                           xcb_atom_t         property,
24                           xcb_size_hints_t *hints,
25                           long               *supplied)
26     {
27         xcb_get_property_cookie_t cookie;
28         xcb_get_property_reply_t *rep;
29 +         xcb_get_property_reply_t *rep;
30 +         xcb_size_hints_t         *hints = NULL;
31 +         long                     length;
32
33         cookie = xcb_get_property (c, 0, window,
34                                   property, WM_SIZE_HINTS,
35                                   0L, 18); /* NumPropSizeElements = 18
36                                             (ICCCM version 1) */
37         rep = xcb_get_property_reply (c, cookie, 0);
38         if (!rep)
39             return 0;
40 +         return NULL;
41
42         length = xcb_get_property_value_length (rep);
43         if ((rep->type == WM_SIZE_HINTS) &&
44             ((rep->format == 8) ||
45              (rep->format == 16) ||
46              (rep->format == 32)) &&
47             (rep->value_len >= 15)) /* OldNumPropSizeElements = 15 (
48 pre-ICCCM) */
49 +         (length >= 15)) /* OldNumPropSizeElements = 15 (pre-ICCCM)
50 */
51         {
52             char *prop;
53             long length;
54             hints = xcb_alloc_size_hints ();
55             if (!hints)
56             {
57                 free (rep);
58                 return NULL;
59             }
60
61             length = xcb_get_property_value_length (rep);
62             /* FIXME: in GetProp.c of xcl, one move the memory.
63              * Should we do that too ? */
64             prop = (char *) malloc (sizeof(char)*length);
65             memcpy(prop, xcb_get_property_value (rep), length);

```

```

-         prop[length] = '\0';
63 -         hints = (xcb_size_hints_t *)strdup (prop);
+         memcpy (hints, (xcb_size_hints_t *)
xcb_get_property_value (rep),
65 +         length * rep->format >> 3);

67         *supplied = (USPosition | USSize |
PPosition | PSize |
69 PMinSize | PMaxSize |
PResizeInc | PAspect);
71 -         if (rep->value_len >= 18) /* NumPropSizeElements = 18
(ICCCM version 1) */
+         if (length >= 18) /* NumPropSizeElements = 18 (ICCCM
version 1) */
73             *supplied |= (PBaseSize | PWinGravity);
else
75 {
@@ -562,16 +563,11 @@ xcb_get_wm_size_hints (xcb_connection_t *c,
77             hints->win_gravity = 0;
}
79             hints->flags &= (*supplied); /* get rid of unwanted
bits */

-         free (rep);
81 -
-         return 1;
83 -     }

85     hints = NULL;
87     free (rep);

89     return 0;
+     return hints;
91 }

93 /* WM_NORMAL_HINTS */
@@ -592,13 +588,12 @@ xcb_set_wm_normal_hints (xcb_connection_t *c,
95     xcb_set_wm_size_hints(c, window, WM_NORMAL_HINTS, hints);
}

97 -int
99 +xcb_size_hints_t *
xcb_get_wm_normal_hints (xcb_connection_t *c,
101     xcb_window_t window,
-     xcb_size_hints_t *hints,
103     long *supplied)
{
105 -     return (xcb_get_wm_size_hints (c, window, WM_NORMAL_HINTS,
hints, supplied));
+     return (xcb_get_wm_size_hints (c, window, WM_NORMAL_HINTS,
supplied));
107 }

109 /* WM_HINTS */
@@ -638,6 +633,12 @@ xcb_alloc_wm_hints ()
111     return calloc(1, sizeof(xcb_wm_hints_t));

```

```

}
113
+void
115 +xcb_free_wm_hints(xcb_wm_hints_t *hints)
+{
117 +     free(hints);
+}
119 +
uint8_t
121 xcb_wm_hints_get_input(xcb_wm_hints_t *hints)
{
123 @@ -826,7 +827,6 @@ xcb_get_wm_hints (xcb_connection_t *c,
xcb_get_property_cookie_t cookie;
125 xcb_get_property_reply_t *rep;
xcb_wm_hints_t *hints;
127 -     char *prop;
long length;
129
cookie = xcb_get_property (c, 0, window,
131 @@ -836,25 +836,24 @@ xcb_get_wm_hints (xcb_connection_t *c,
if (!rep)
133     return NULL;

+     length = xcb_get_property_value_length (rep);
if ((rep->type != WM_HINTS) ||
137 -     (rep->value_len < (XCB_NUM_WM_HINTS_ELEMENTS - 1)) ||
+     (length < (XCB_NUM_WM_HINTS_ELEMENTS - 1)) ||
139     (rep->format != 32))
{
141     free (rep);
return NULL;
143 }
-     hints = (xcb_wm_hints_t *)calloc (1, (unsigned)sizeof (
xcb_wm_hints_t));
145 +     hints = xcb_alloc_wm_hints ();
if (!hints)
147 {
149     free (rep);
return NULL;
}

151 -     length = xcb_get_property_value_length (rep);
-     prop = (char *) xcb_get_property_value (rep);
-     prop[length] = '\0';
155 -     hints = (xcb_wm_hints_t *)strdup (prop);
-     if (rep->value_len < XCB_NUM_WM_HINTS_ELEMENTS)
157 +     memcpy(hints, (xcb_size_hints_t *) xcb_get_property_value (rep
),
+     length * rep->format >> 3);
159 +     if (length < XCB_NUM_WM_HINTS_ELEMENTS)
hints->window_group = XCB_NONE;

161
return hints;
163 diff --git a/icccm/xcb_icccm.h b/icccm/xcb_icccm.h
index eafc71b..d7b42a9 100644
165 --- a/icccm/xcb_icccm.h

```

```

+++ b/icccm/xcb_icccm.h
167 @@ -190,10 +190,9 @@ void xcb_set_wm_size_hints (
        xcb_connection_t *c,
                                xcb_atom_t
                                property,
169                                xcb_size_hints_t *hints
                                );
171 -int xcb_get_wm_size_hints (xcb_connection_t *c,
+xcb_size_hints_t *xcb_get_wm_size_hints (xcb_connection_t *c,
173                                xcb_window_t
                                window,
                                xcb_atom_t
                                property,
175 -                                xcb_size_hints_t *hints
                                ,
                                long
                                *
                                supplied);
177
/* WM_NORMAL_HINTS */
179 @@ -206,10 +205,9 @@ void xcb_set_wm_normal_hints (xcb_connection_t *c
        ,
                                xcb_window_t window,
181                                xcb_size_hints_t *hints);
183 -int xcb_get_wm_normal_hints (xcb_connection_t *c,
-                                xcb_window_t window,
185 -                                xcb_size_hints_t *hints ,
-                                long *supplied);
187 +xcb_size_hints_t *xcb_get_wm_normal_hints (xcb_connection_t *c,
+                                xcb_window_t window,
189 +                                long *supplied)
        ;
191
/* WM_HINTS */
193 @@ -223,6 +221,7 @@ typedef enum {
        } xcb_wm_state_t;
195
xcb_wm_hints_t *xcb_alloc_wm_hints ();
197 +void xcb_free_wm_hints (xcb_wm_hints_t *hints);
199
uint8_t xcb_wm_hints_get_input (xcb_wm_hints_t *hints);
xcb_pixmap_t xcb_wm_hints_get_icon_pixmap (xcb_wm_hints_t *hints);

```

Listing B.7: *[PATCH] Fix invalid data returned by ICCCM hints functions*

B.2.3 Add non-blocking events loop

```

commit 66728b2d13f3eee94230038c8f8eb236c20b9525
2 Author: Arnaud Fontaine <arnaud@andesi.org>
Date: Mon Jan 28 14:26:05 2008 -0800
4
Add xcb_poll_for_event_loop and rename xcb_event_loop to match.

```

```

6      I'm currently porting Awesome[0] from Xlib to XCB as a school
      project for
8      my bachelor. I discussed with Vincent about adding a non-blocking
      xcb_event_loop on IRC because I had to write one for Awesome and
      wondered
10     if this kind of function could be added to xcb-util.

12     [0] http://awesome.naquadah.org

14     Signed-off-by: Jamey Sharp <jamey@minilop.net>

16     diff --git a/event/events.c b/event/events.c
      index 2900bd3..354346a 100644
18     --- a/event/events.c
      +++ b/event/events.c
20     @@ -62,7 +62,7 @@ static int handle_event(xcb_event_handlers_t *
      evenths, xcb_generic_event_t *event
      return 0;
22     }

24     -void xcb_event_loop(xcb_event_handlers_t *evenths)
      +void xcb_wait_for_event_loop(xcb_event_handlers_t *evenths)
26     {
      xcb_generic_event_t *event;
28     while((event = xcb_wait_for_event(evenths->c)))
      @@ -72,6 +72,16 @@ void xcb_event_loop(xcb_event_handlers_t *evenths)
30     }

32     }

34     +void xcb_poll_for_event_loop(xcb_event_handlers_t *evenths)
      +{
      +    xcb_generic_event_t *event;
36     +    while ((event = xcb_poll_for_event(evenths->c)))
      +    {
38     +        handle_event(evenths, event);
      +        free(event);
40     +    }
      +}

42     +static void set_handler(xcb_generic_event_handler_t handler, void *
      data, xcb_event_handler_t *place)
44     {
      xcb_event_handler_t eventh = { handler, data };
46     diff --git a/event/xcb_event.h b/event/xcb_event.h
      index deb7ba8..9122893 100644
48     --- a/event/xcb_event.h
      +++ b/event/xcb_event.h
50     @@ -14,7 +14,8 @@ xcb_event_handlers_t *xcb_alloc_event_handlers(
      xcb_connection_t *c);
      void xcb_free_event_handlers(xcb_event_handlers_t *evenths);
52     xcb_connection_t *xcb_get_xcb_connection(xcb_event_handlers_t *
      evenths);

54     -void xcb_event_loop(xcb_event_handlers_t *evenths);
      +void xcb_wait_for_event_loop(xcb_event_handlers_t *evenths);

```



```

56 +void xcb_poll_for_event_loop(xcb_event_handlers_t *evenths);
58     typedef int (*xcb_generic_event_handler_t)(void *data,
        xcb_connection_t *c, xcb_generic_event_t *event);
        typedef int (*xcb_generic_error_handler_t)(void *data,
        xcb_connection_t *c, xcb_generic_error_t *error);
60 diff --git a/wm/xcbwmm-test.c b/wm/xcbwmm-test.c
    index 4d8be5d..c928d7a 100644
62 --- a/wm/xcbwmm-test.c
    +++ b/wm/xcbwmm-test.c
64 @@ -205,7 +205,7 @@ int main(int argc, char **argv)
        if(TEST_THREADS)
        {
66         -           pthread_create(&event_thread, 0, (void *(*)(void *))
            xcb_event_loop, evenths);
        +           pthread_create(&event_thread, 0, (void *(*)(void *))
            xcb_wait_for_event_loop, evenths);
70         }
72         root = xcb_aux_get_screen(c, screen_nbr)->root;
    @@ -223,7 +223,7 @@ int main(int argc, char **argv)
74         if(TEST_THREADS)
            pthread_join(event_thread, 0);
76         else
        -           xcb_event_loop(evenths);
78         +           xcb_wait_for_event_loop(evenths);
80         exit(0);
        /*NOTREACHED*/

```

Listing B.8: [PATCH] Add non-blocking events loop