LibX11, libxcb, xerver libwayland, wayland server

Keith Packard: Deverloper of Xorg

Kristian: Deverloper of wayland

https://www.x.org/wiki/ArvindUmrao/

Demo of Xorg and Xwayland

1. Switching from Xorg to Wayland. And what we gain?
2. Hello world of libX11, libwayland and libxcb
3. Demo of xclock and xlogo.
4. Running xclock at xorg/xserver and waylandserver
5. xclock redering with and without HDMI/DP
6. Export DISPLAY=:0
7. To understand wayland we need to understand xorg first

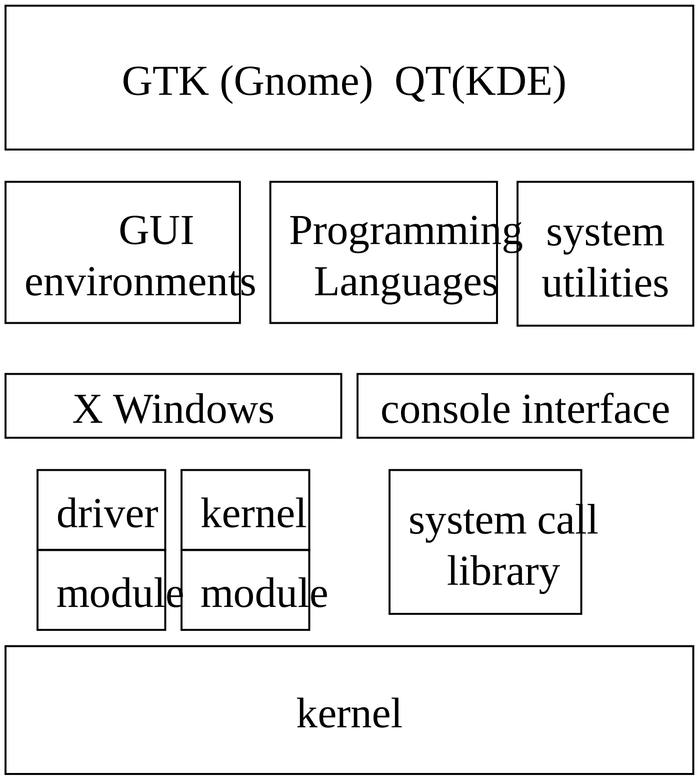
```
gdm          916     914  0 14:14 tty1     00:00:01 /usr/lib/xorg/Xorg vt1 -displayfd 3 -auth
/run/user/125/gdm/Xauthority -background none -noreset -keeptty -verbose 3
```

To enable wayland server vi /etc/gdm3/custom.conf do WaylandEnable=true

```
root@inspiron-5559:/etc/gdm3# systemctl restart gdm3
```

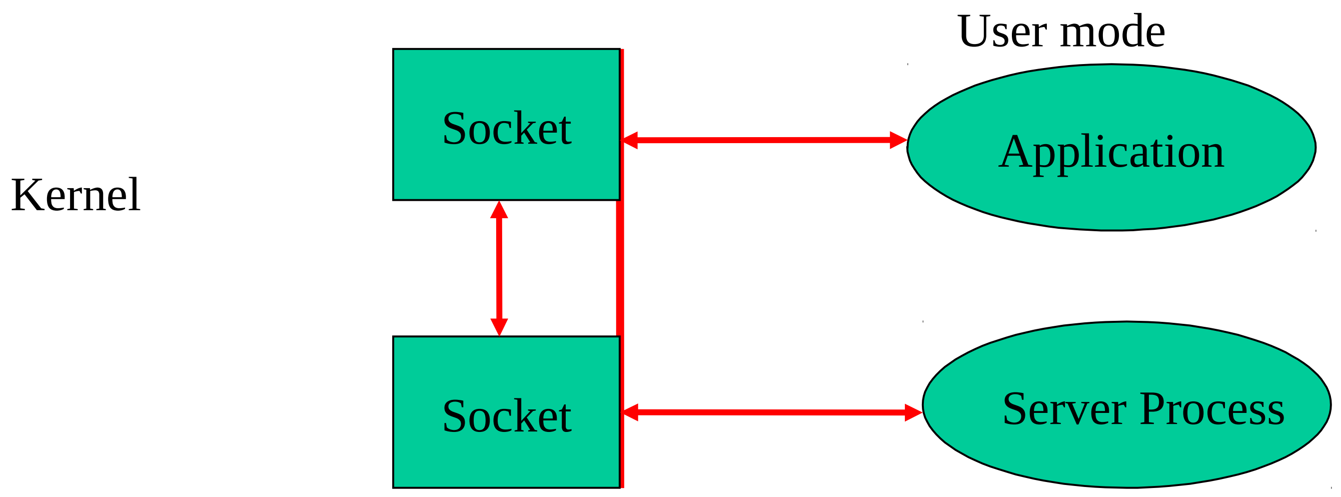
```
gdm         1588    1517  0 14:20 tty1     00:00:00 /usr/bin/Xwayland :1024 -rootless -noreset -
accessx -core -auth /run/user/125/.mutter-Xwaylandauth.NRZLU2 -listen 4 -listen 5 -displayfd
6 -listen 7
```

Basic introduction of X11 and Windowing before understading wayland



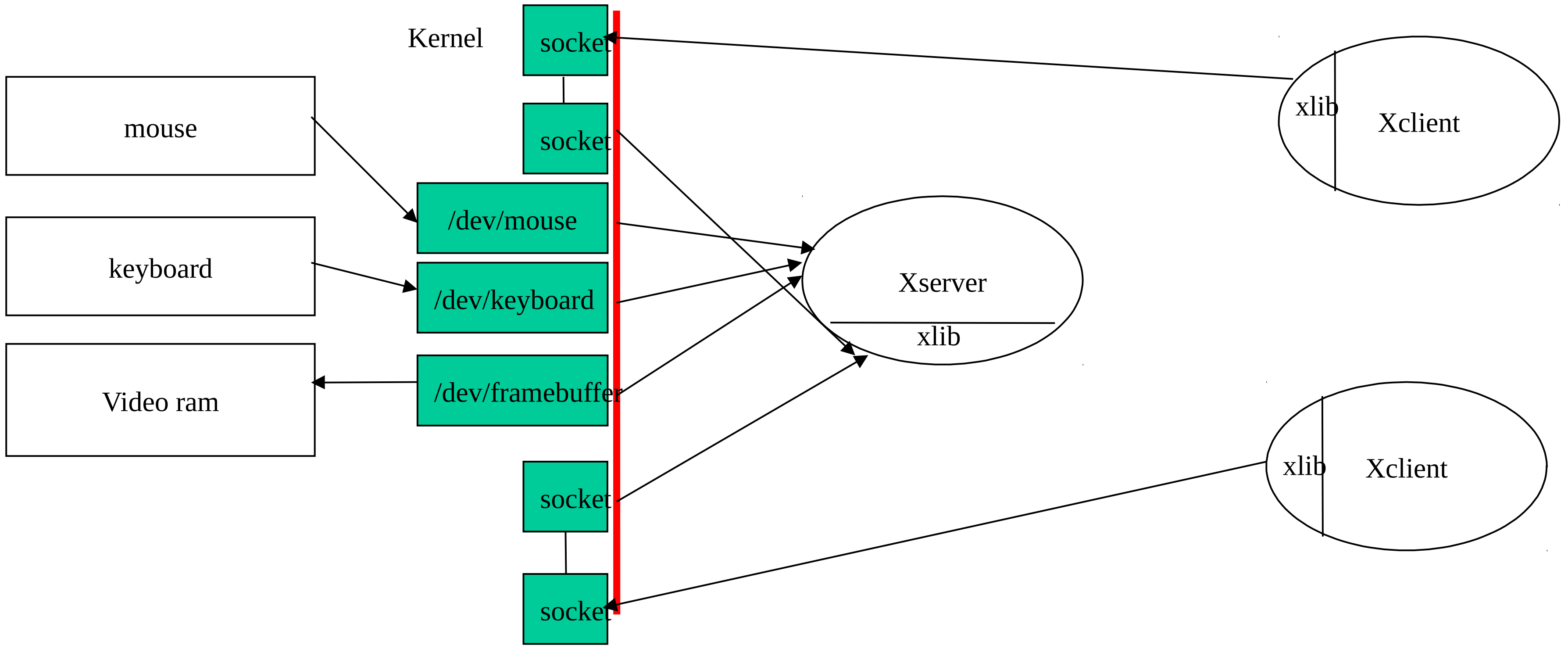
X Windows based applications can display there output either on the local screen or on some other X Window based station in the network.

Basic Interprocess Communication with Sockets



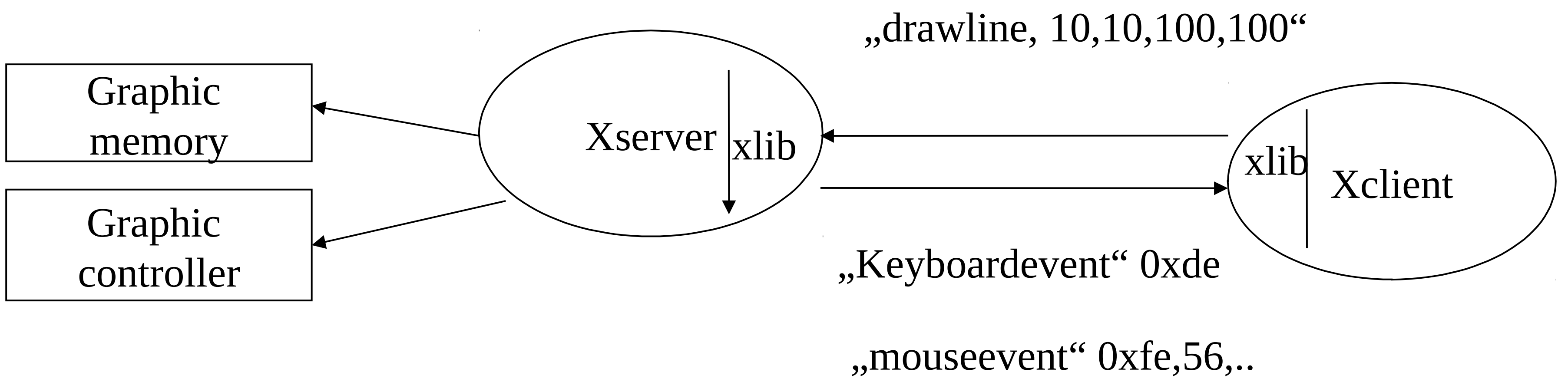
Sockets hostname:port number and protocol. They work across machines and also locally. Typically a server process waits on a server socket for requests from clients. A client opens a socket to the server and sends requests. The connection is bi-directional and either stream or packet oriented (protocol tcp or udp). Lately the DBUS architecture provides an IPC layer on top of sockets for the communication of desktop applications.

Basic X window system: IPC



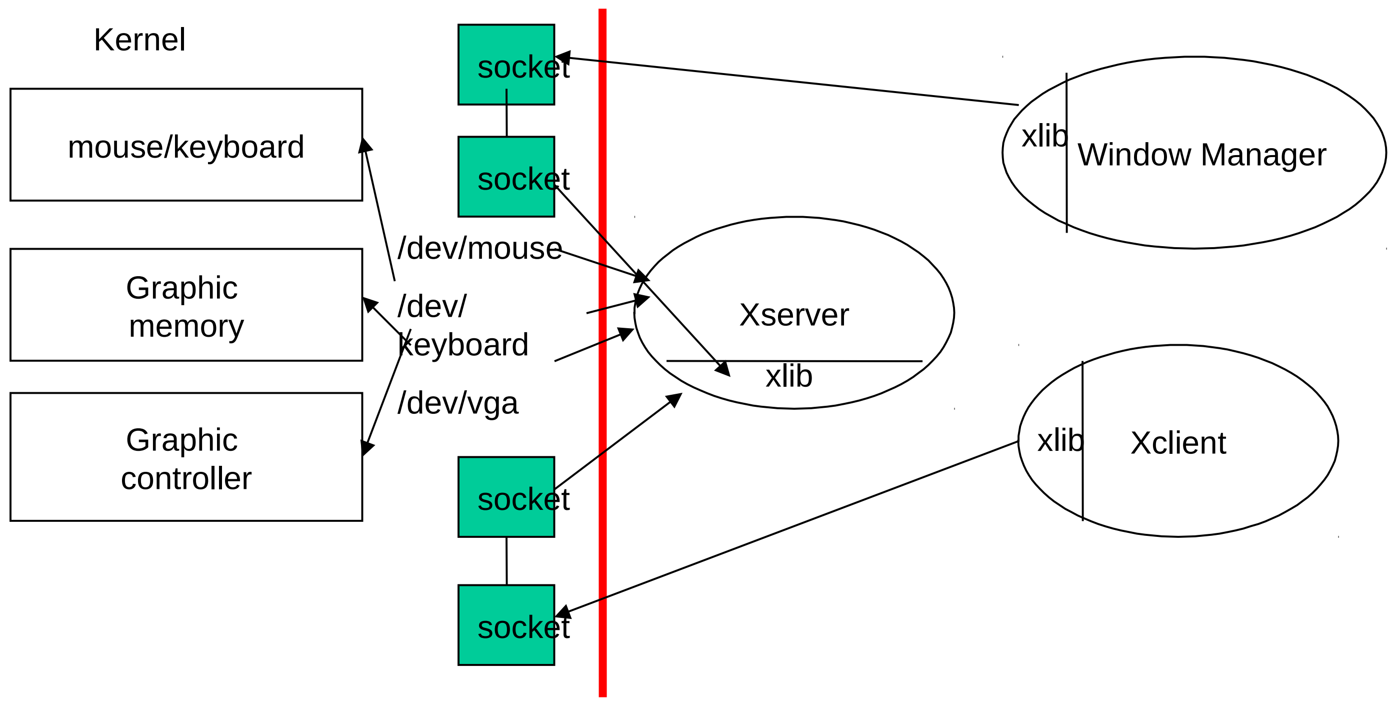
The graphic subsystem resides completely in USER space. Xclients talk via sockets to an Xserver. The X server is the only one who controls the graphic hardware (screen, keyboard, mouse, controller) through the device driver interfaces (/dev/xxx...). Clients know nothing about graphic hardware: The Xserver will render every command either by writing to the video memory or by issuing commands to the intelligent graphic controller. Note: The communication endpoints are network wide: X Windows works across machines.

Basic X window system: API



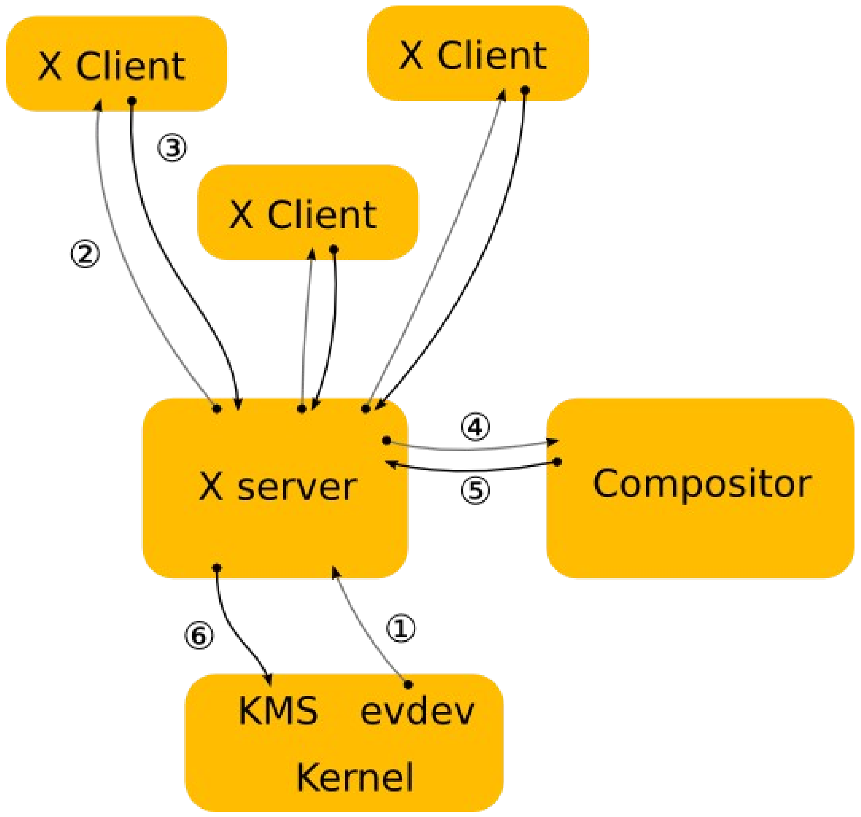
The X Windows protocol is implemented in the X Windows library (xlib). Example: „Draw line, 10, 10, 100, 100, gc) would tell an X Server to draw a line between the coordinates and use the graphical context given for that. Depending on the hardware the XServer either has to translate the command into pixel values (e.g. Bresenham) or command to intelligent graphics hardware. The Xclient receives mouse and keyboard events from the server. This make it clear: the XServer controls the viewing station. The graphics application can run on a machine without any graphics hardware or display.

Basic X Desktops and Window Managers



X Windows does only provide mechanism, no policy, e.g. how a desktop or window manager should look. Separating mechanism and policy allows different policies to be implemented on the same platform. Reality has shown that users do not really appreciate this feature... Who wants a different user interface in every car?? This was one of the frequent cases where a clever technical idea did not meet the users demands.

X Architecture

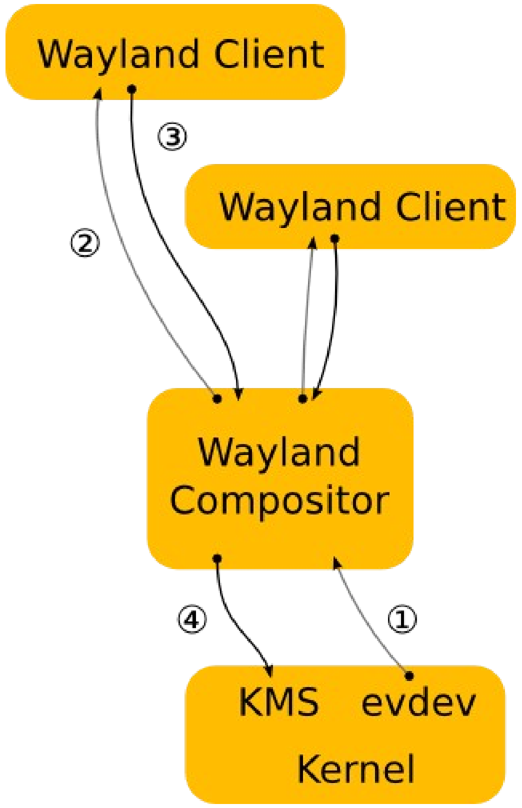


1. The kernel gets an event from an input device and sends it to X.
2. The X server determines which window the event affects and sends it to the clients that have selected for the event in question on that window.
3. The client looks at the event and decides what to do.
4. When the X server receives the rendering request, it sends it to the driver to let it program the hardware to do the rendering.
5. The damage event tells the compositor that something changed in the window and that it has to recomposite the part of the screen where that window is visible.
6. The X server receives the rendering requests from the compositor and either copies the compositor back buffer to the front buffer or does a pageflip.

http://wayland.freedesktop.org/architecture.html

Wayland

Wayland architecture



1. The kernel gets an event and sends it to the compositor. This is similar to the X case.
2. The compositor looks through its scenegraph to determine which window should receive the event.
3. As in the X case, when the client receives the event, it updates the UI in response.
4. The compositor collects damage requests from its clients and then recomposites the screen.

XWayland was written to enable running X11 applications through an X server, optionally rootless, running as a Wayland client. This is similar to the way X applications run in OS X's native graphics environment

Taken from Wikipedia and the Wayland web site (http://wayland.freedesktop.org/architecture.html)

Wayland core protocol

Object-oriented(async) services offered by the compositor are presented as a series of objects living on the same compositor. Each object implements an interface which has a name, a number of methods (called requests) as well as several associated events

define basic building blocks of clinet to interact with compositor

wl_outpu, wl_seat wl_pointer wl_touch
wl_kaybaourd
wl_surface, wl_buffer
drad and drop drop related interfaces

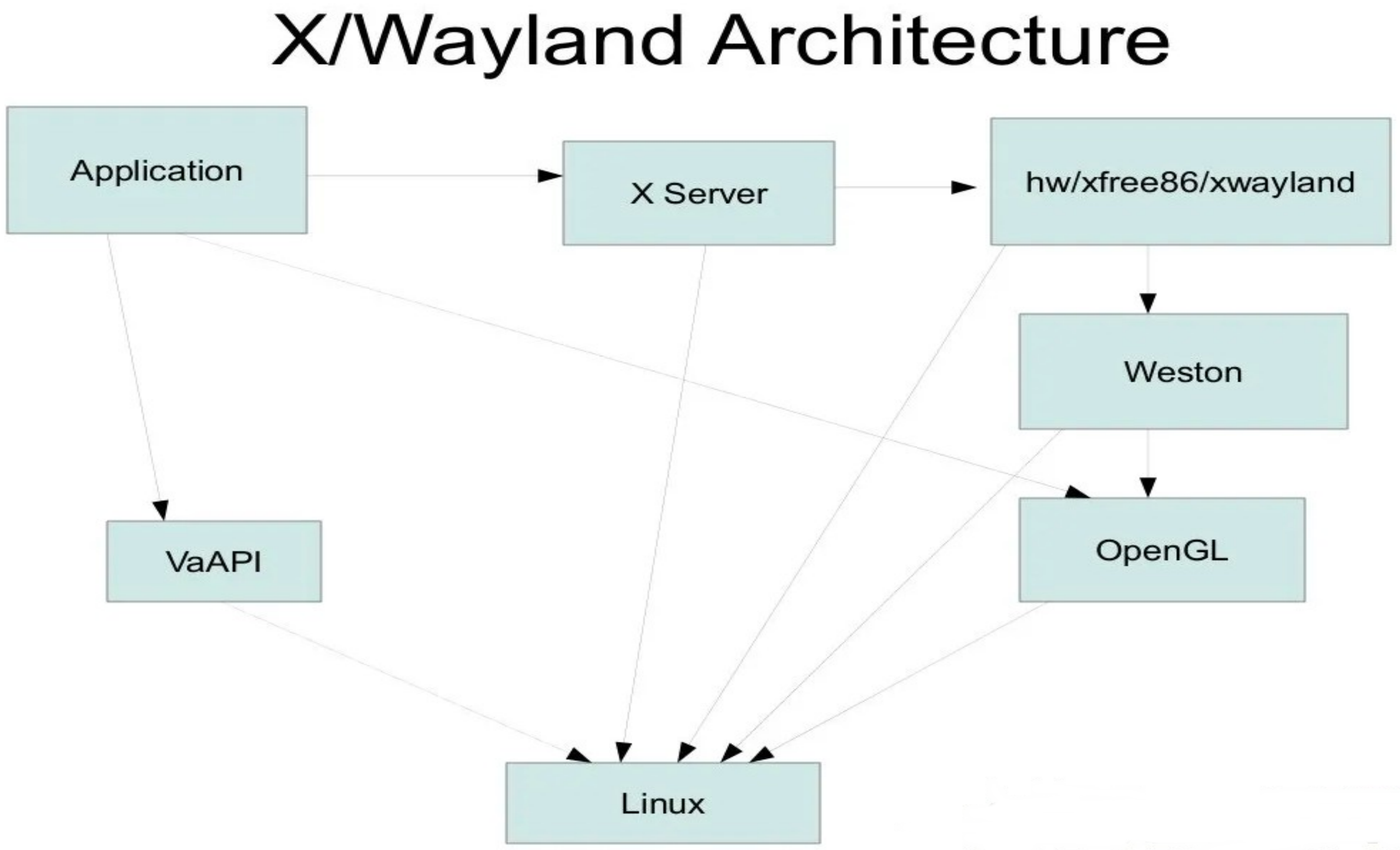
usual flow, client connect to server
server adveristes some objects
client bind some globals and intracting with globals
clint send request to server and getting event from server

Diff x vs wayland

External vs internal compositor

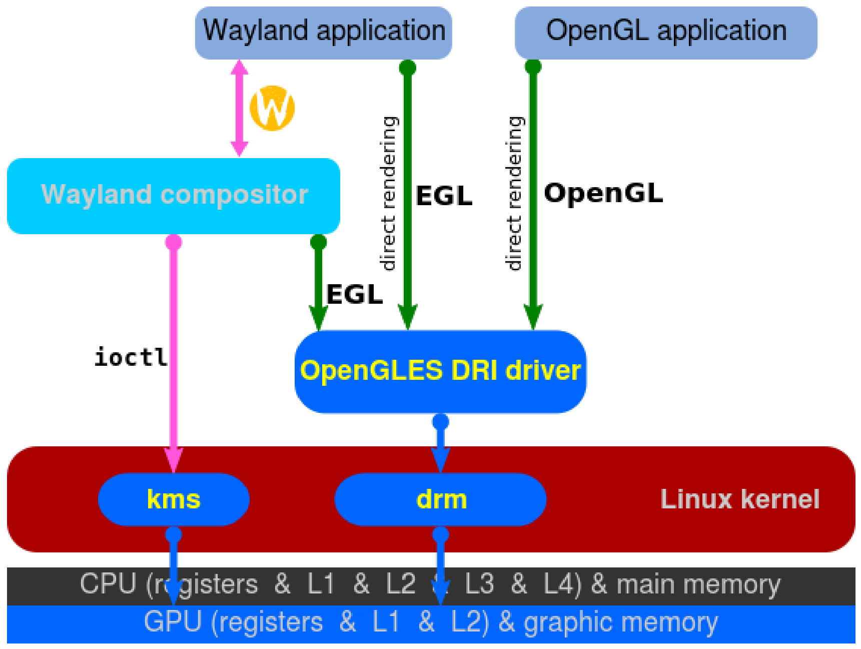
External vs internal windows management

External vs Client-sde decoration



Wayland https://wayland.freedesktop.org/

Wayland uses direct rendering over EGL



Wayland compositors

Display servers that implement the Wayland display server protocol are also called Wayland compositors because they additionally perform the task of a compositing window manager.

- Weston – the reference implementation of a Wayland compositor; Weston implements client-side decoration
- Lipstick – mobile user experience (UX) framework which implements Wayland compositor. It is used in Sailfish OS and Nemo Mobile.
- Enlightenment 0.19 (E19) is expected to have full Wayland support.
- KWin had incomplete Wayland support in April 2013.
- Mutter maintains a separate branch for the integration of Wayland for GNOME 3.9 (in September 2013).
- Clayland is a simple example Wayland compositor using Clutter.

http://en.wikipedia.org/wiki/Wayland_(display_server_protocol)
https://upload.wikimedia.org/wikipedia/commons/a/a7/Wayland_display_server_protocol.svg

Compile and run x11 and wayland test samples

- Dev package requried at ubuntu
- ssh -x
- Dependent libs for xclinet and waylandclient with ldd
- 2D cairo, Pixman, xrender, 3D/2D Mesa, Gallium, DIX, DDX, DRM, DRI2, KMS
- Authorization with xauth and xhost

  Xwaylandauth
- Wayland-scanner
- Font freetype

GUI Toolkits/widget sample code

- Many cross-platform C++ GUI toolkits already exist:
  - Qt
  - GTK+
  - SDL
  - wxWidgets (WxWindows)

https://jan.newmarch.name/Wayland/index.html

```
cc -o damage  damage.c -lwayland-client
g++ gtkwin.c -o base `pkg-config --cflags --libs gtk+-3.0`
```

https://docs.gtk.org/gtk3/getting_started.html

References

https://wayland-book.com/introduction/high-level-design.html

https://wayland.freedesktop.org/faq.html

https://jan.newmarch.name/Wayland/index.html

https://bugaevc.gitbooks.io/writing-wayland-clients/content/black-square/global-objects.html

https://upload.wikimedia.org/wikipedia/commons/a/a7/Wayland_display_server_protocol.svg

https://www.phoronix.com/news/MTA4NDQ

https://wayland.pages.freedesktop.org/weston/toc/libweston/output-management.html

https://blog.lancitou.net/build-and-run-weston-on-ubuntu/

https://jan.newmarch.name/Wayland/ProgrammingClient/

https://www.youtube.com/watch?v=LbDOCJcDRoo

https://www.youtube.com/watch?v=yl6oC7mREFs

Imp. Locations in ubuntu

~/.config/monitors.xml
/etc/gdm3/custom.conf
/etc/X11/xorg.conf
./usr/share/wayland/wayland.xml
./usr/share/doc/libwayland-dev
./usr/share/wayland-session

Jargon

DE Display Engine

DRM (Direct Rendering Manager)
Originally created for 3D GPU management on Linux
Provides common support functions for card-specific drivers as a minimum set of IOCTL operations.
KMS (Kernel Mode Setting)
Component which is solely responsible for the mode setting
It is the device driver for a display controller (IPUv3 for i.MX6)

Framebuffer
Standard object storing information about the content to be displayed
Default implementation available for GEM objects using CMA (Contiguous Memory Allocator)

GEM(Graphics Execution Manager) is a kernel module that provides a library of functions for managing memory buffers
TTM is another option which performs the same as GEM but is more complex and handles on-card memory

Planes
Image layer (cursor or overlay for instance)

CRTC
Configure the appropriate display settings
Scan out frame buffer content to one or more encoders

Encoder
Responsible for converting a frame into the appropriate format

Connector
Represent a display connector (HDMI, DP, VGA, DVI, ...)
Detect display connection/removal
Expose display supported modes