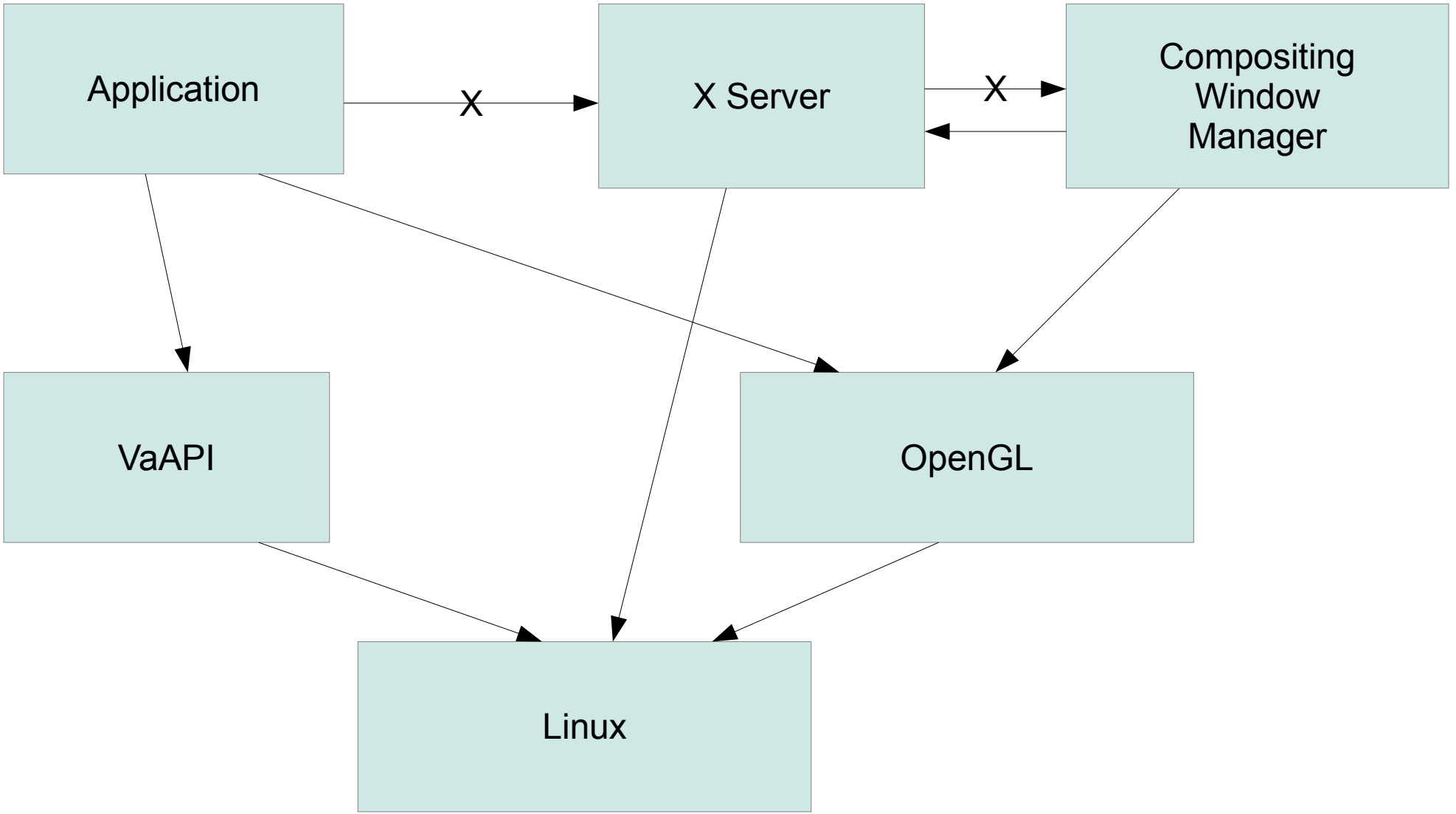
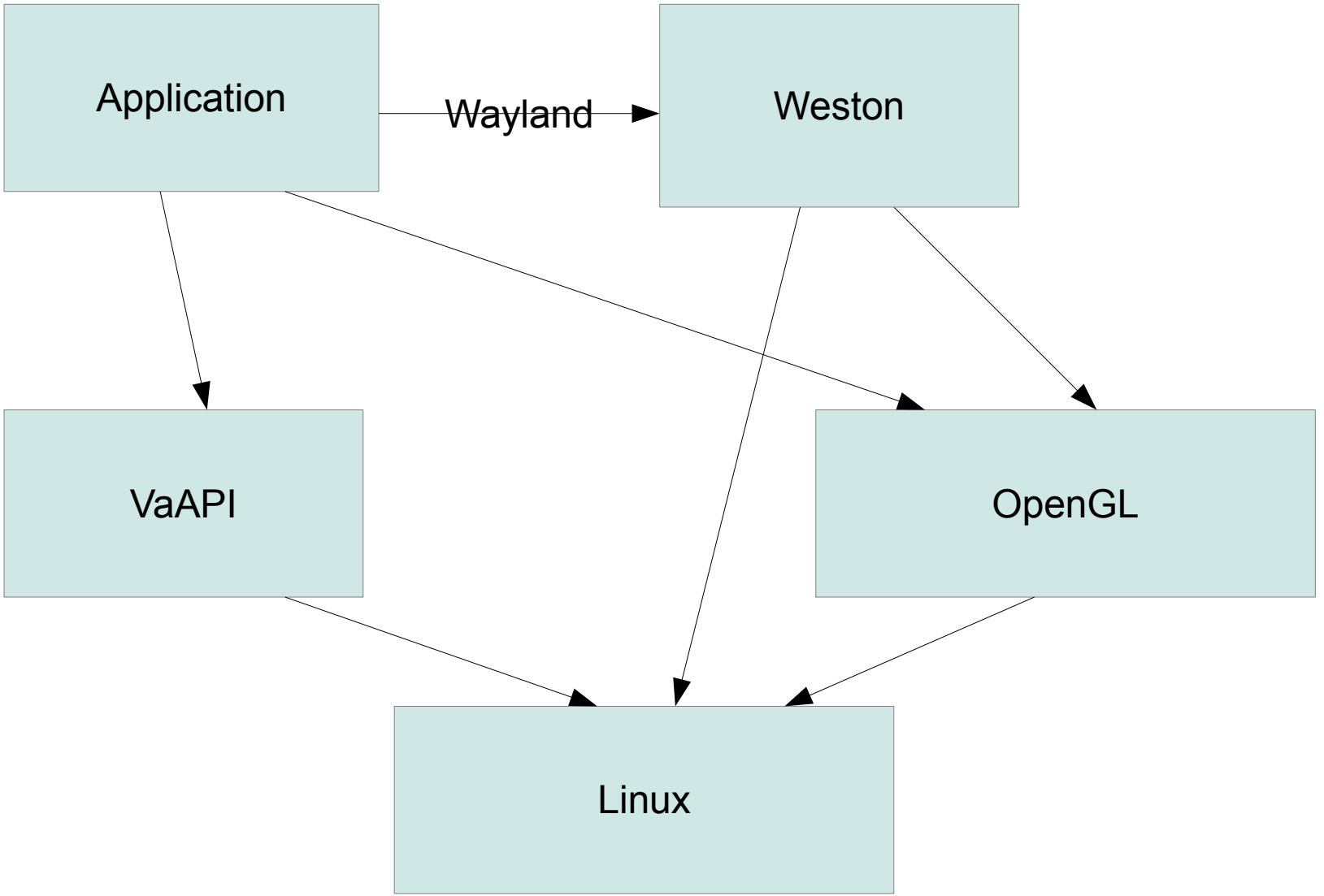
X and Wayland

Keith Packard
Open Source Technology Center
Intel
keithp@keithp.com

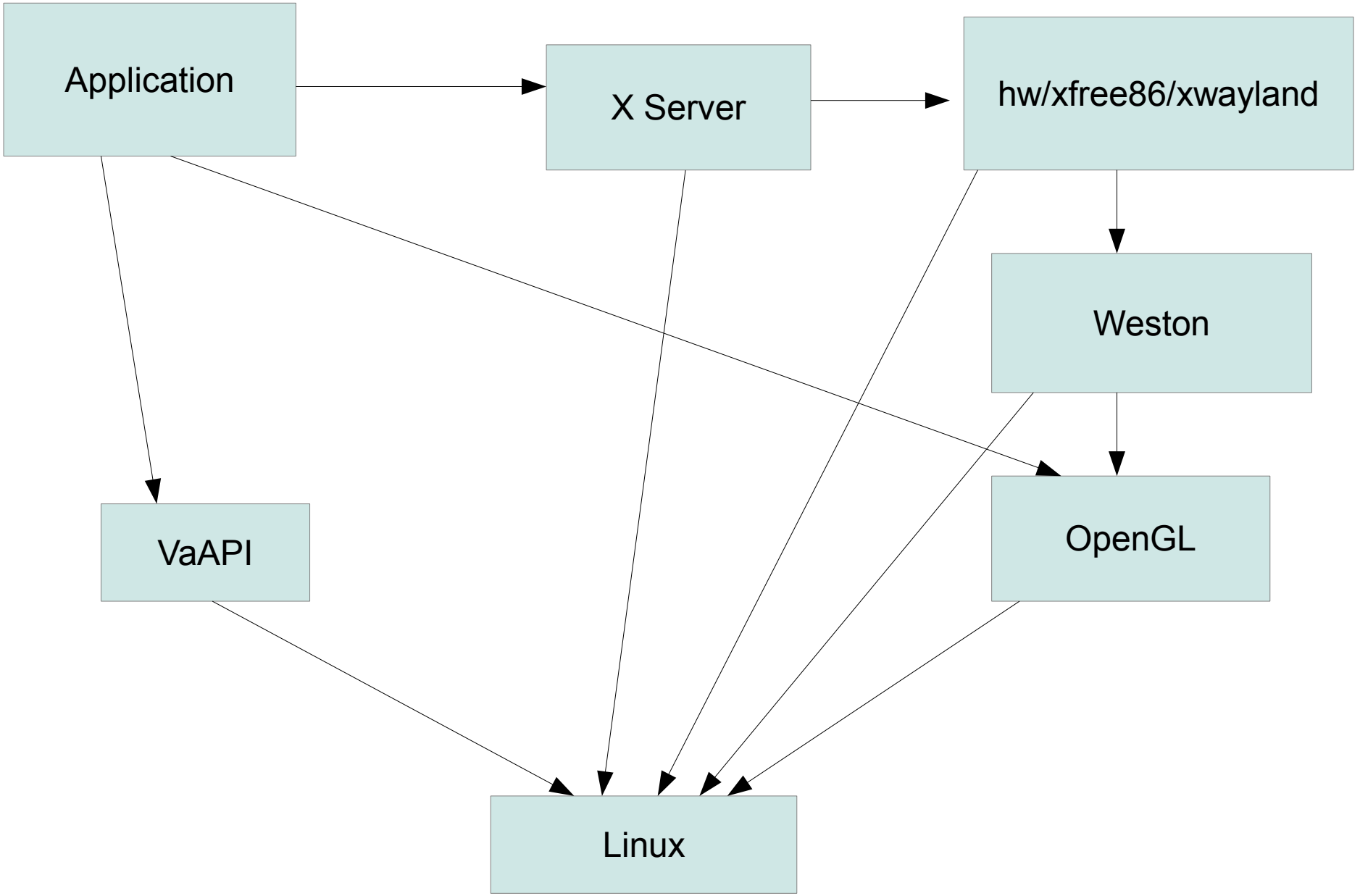
Classic X Architecture



Wayland Architecture



X/Wayland Architecture



Differences between X and Wayland

- External vs Internal compositor
- External vs Internal window management
- External vs Client-side decorations

Shared Output Components

- Direct Rendering APIs
  - OpenGL/OpenGLES
  - VaAPI/VDPAU
- Kernel Drivers
  - Mode setting
  - Execution Management
- X Drawing Code
  - Unhook mode setting
  - Including AIGLX

X Performance Implications

- Drawing
  - Uses identical code paths
  - Direct rendering (Media, OpenGL) are unchanged
  - Indirect X/AIGLX rendering share existing X drivers
- Buffer Swap
  - Goes through Wayland
  - Instead of X compositing manager
    - Which ends up going back to X...
  - ➔ Net reduction in context switches

X/Wayland Window Management

- Wayland-specific window manager
- Translates between ICCCM/EWMH and Wayland
- Provides decorations

X server changes

- Automatically redirect top-level windows
- Disable input device detection
- Create virtual keyboard/mouse devices

X video driver changes

- Disable native mode setting
- Get mode information from Wayland
- (ugly hacks for window move/ resize)
  - To be replaced with X/Wayland WM
- Don't acquire DRM master

Wayland Window Management

- Client-side decorations
  - Integrates decorations into applications
  - Provides uniform L&F
- Server-accelerated manipulation
  - Start window move
- Hacks for dealing with stuck applications

Window Management

- External Wayland-specific X WM
- X windows need decorations
  - Painted by WM
- ICCCM/EWMH
  - Mostly managed by WM
  - Some bits mirrored to Wayland

Cut&Paste, Drag&Drop

- Wayland provides good support
  - MIME-labeled objects
  - Client ↔ client direct transfer
- Work with those
  - Proxy within X/Wayland WM

Starting X server

- Could start at session init time
  - Extra lag for startup
  - Extra memory overhead
- Could start manually
  - Much like 'run in terminal' option
- Instead, start automatically
  - Weston listens on X socket
  - Waits for incoming X connection
  - Launches X, passing client and listen fds

Remaining issues

- Wayland input still in flux
  - Keyboard support
  - Touch-screens, touch pads
- Remote Wayland applications
  - Not a key requirement
  - Still nice to have

Remote Wayland Applications

- Not fundamentally different from local Wayland apps:
  - Apps create new window image
  - Deliver image to window server
- The difference is minor:
  - Networks don't have shared memory

A Wayland Proxy?

- Takes image from local client
- Packs up the data
  - Can compress
  - Need not be lossless?
- Delivers over the wire
  - Either directly to the Wayland server
  - Or perhaps to a separate server proxy which unpacks the data and acts as a local Wayland? client

Remote Wayland Performance?

- Could accelerate with local GPU?
- Eliminate most round trips
  - Which are the bulk of the X network slowdown
- Use lossy compression?
  - Limit network utilization
  - Reliably hit frame rates
- Lots more questions to answer here.

Wayland Keyboard Support

- Currently looks a lot like X
  - Keycodes and Keysyms
- But input methods will be hard
  - No easy client ↔ client communication
  - X has custom Xlib-based input protocol
- Move key → text translation to server.
- What does this mean for X/Wayland?

What's working today?

- xwayland X server backend
  - redirects windows
  - automatically launched
- Synthetic keyboard/mouse
  - Uses current Wayland keyboard mechanisms
- xf86-video-intel wayland support
  - Bypass mode setting
  - Shares 2D acceleration code

What still needs doing

- External X/Wayland window manager
- Cut&Paste/Drag&Drop
- Xinput 2.2
  - Scroll wheels
  - Touch screens
  - Touch pads