

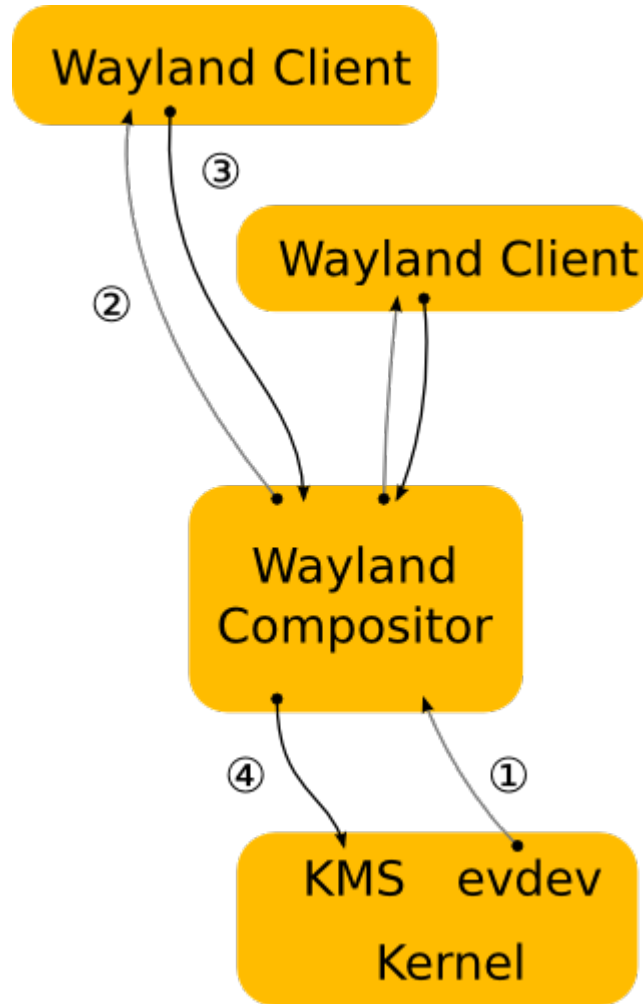
Separating the Wayland Compositor and Window Manager

By Isaac Freund

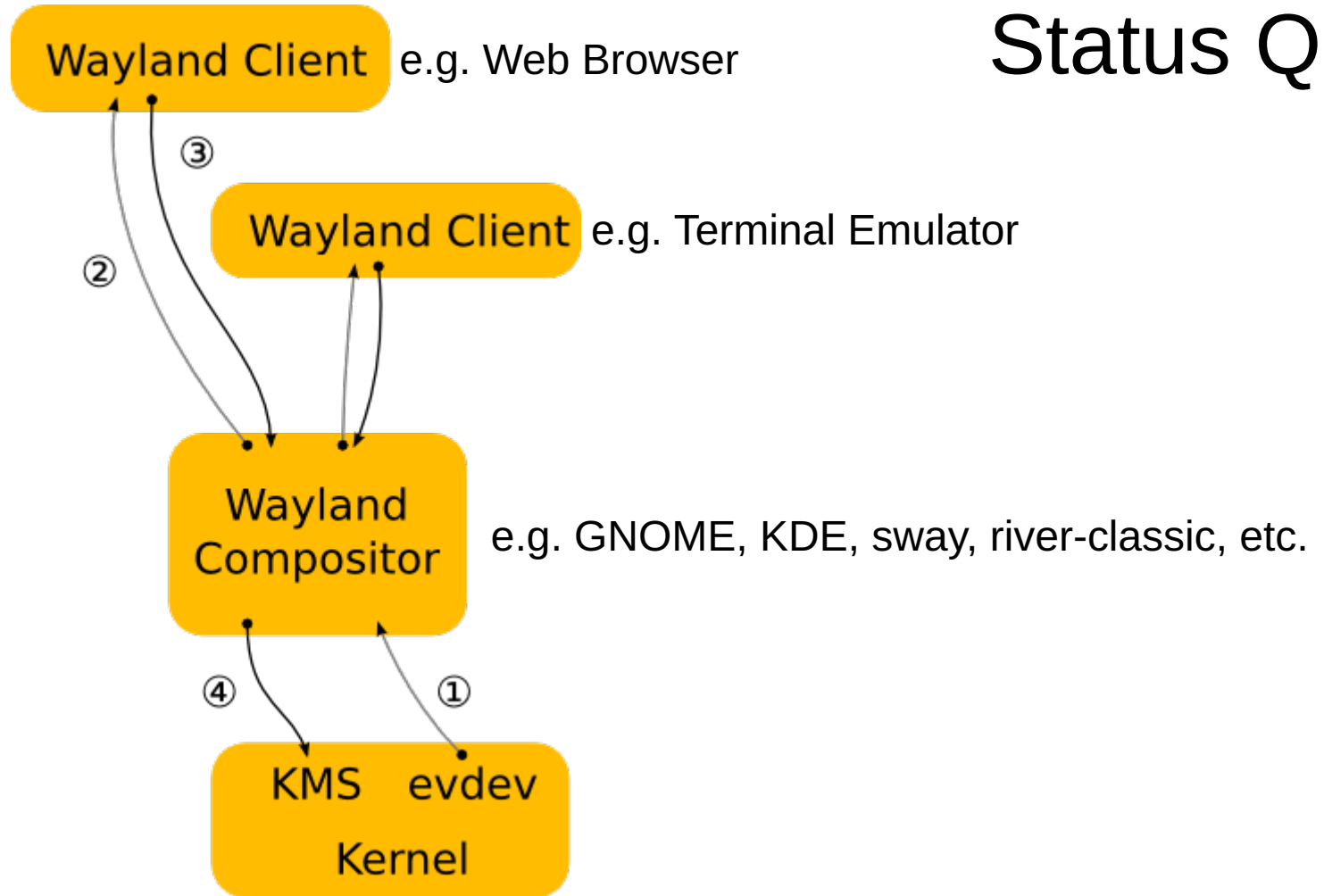
Who am I?

- Author of the river Wayland compositor
- wlroots developer
- Zig core team member

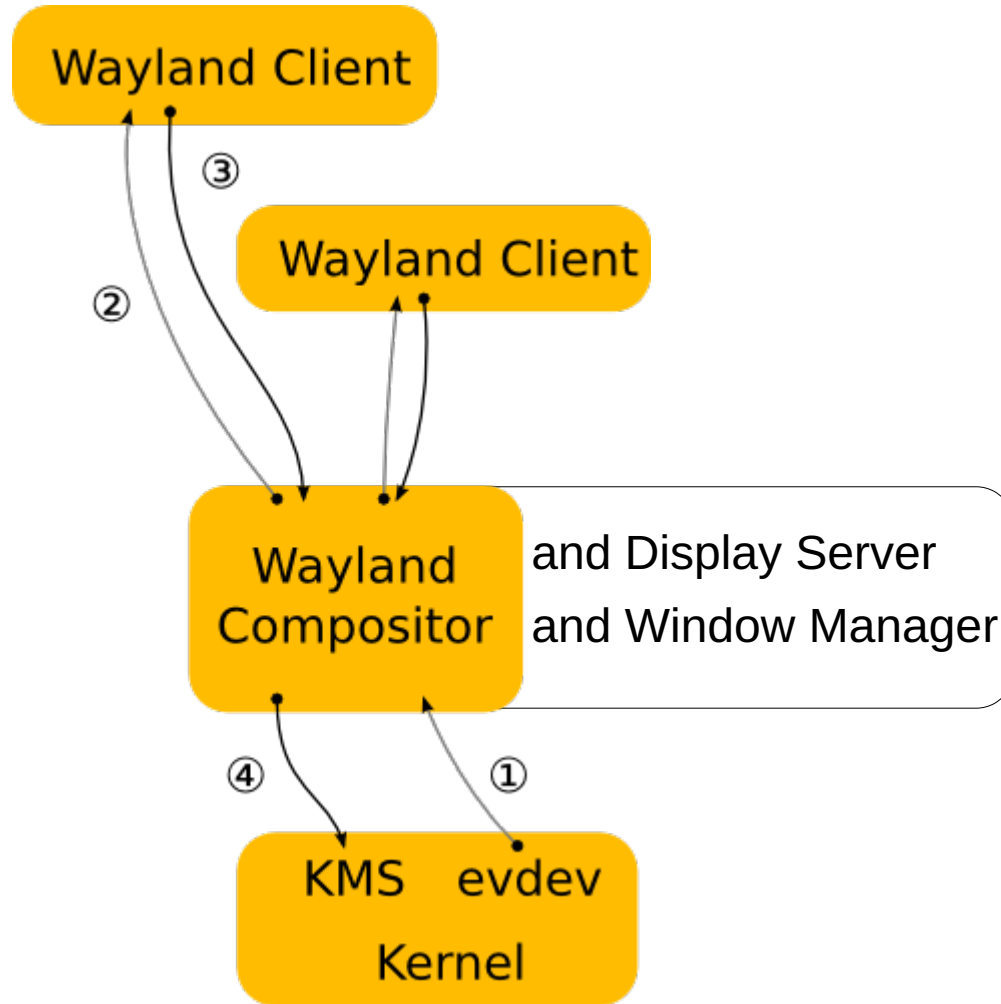
Status Quo



Status Quo



Status Quo



Display server:

- Route input events from kernel to windows
- Provide kernel with buffers to display

Display server:

- Route input events from kernel to windows
- Provide kernel with buffers to display

Compositor

- Combine many buffers from windows into a single buffer to be displayed

Display server:

- Route input events from kernel to windows
- Provide kernel with buffers to display

Compositor

- Combine many buffers from windows into a single buffer to be displayed

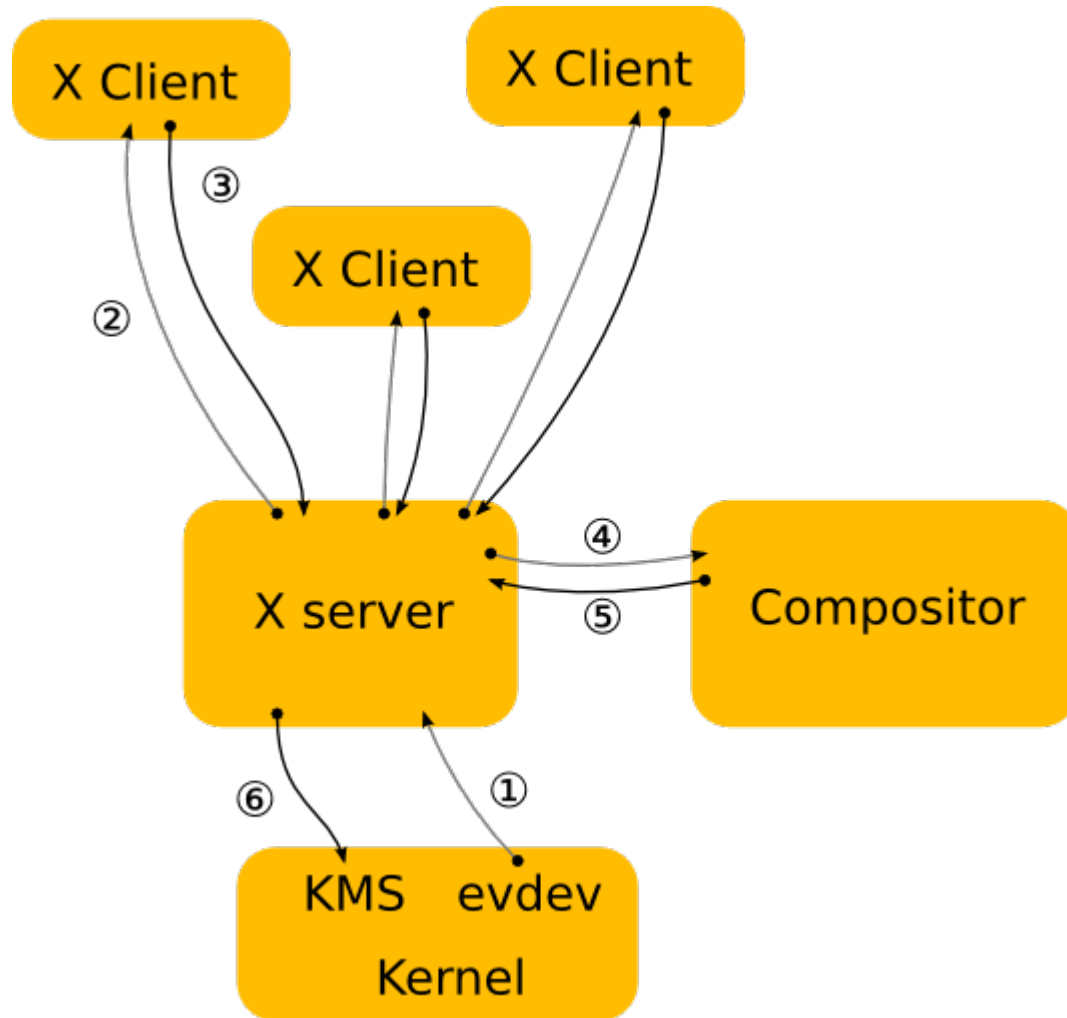
Window manager

- Arrange windows, define keybindings, etc.

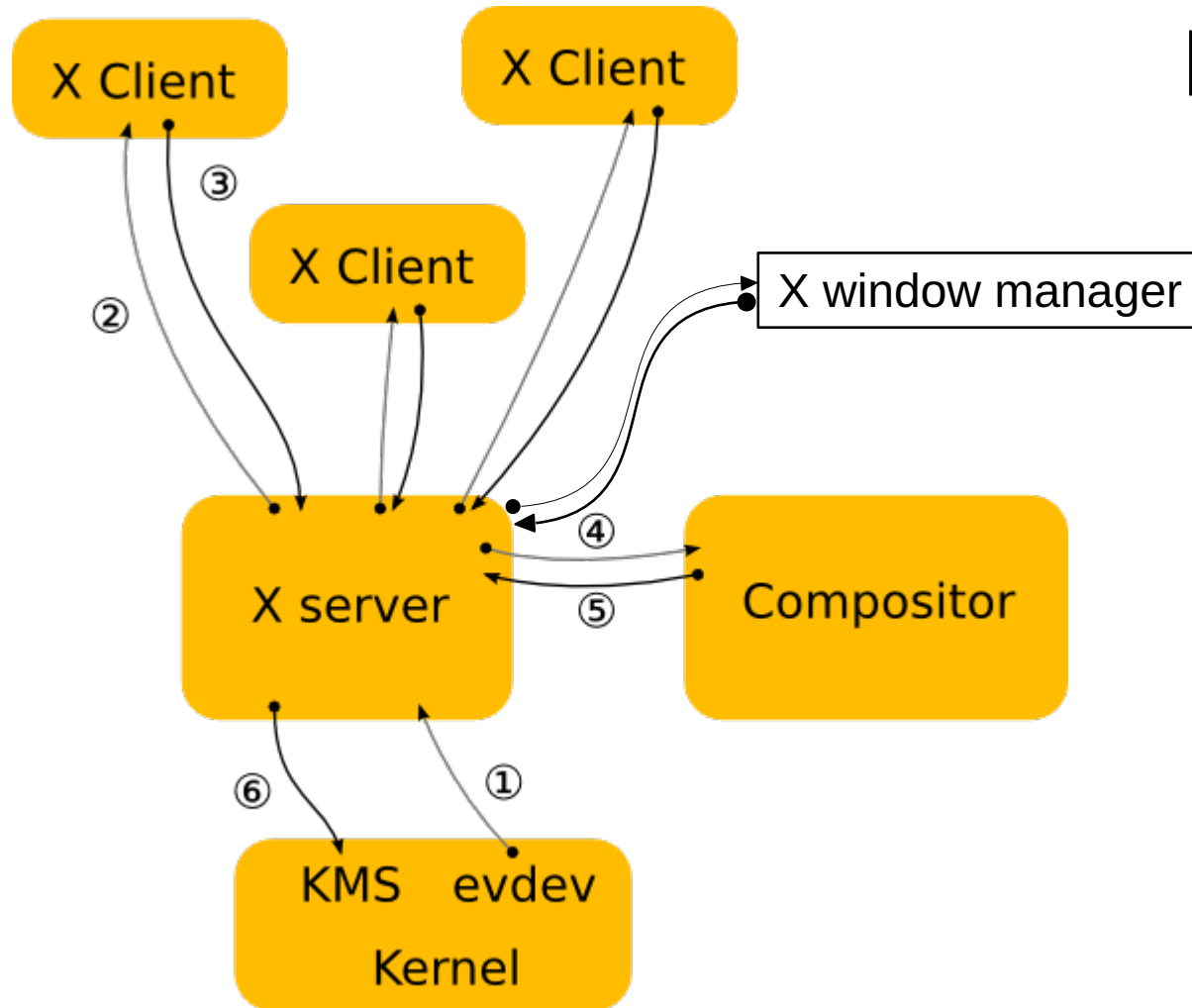
Why did Wayland combine the display server,
window manager and compositor?

Actually the goal of Wayland was just to combine the display server and compositor.

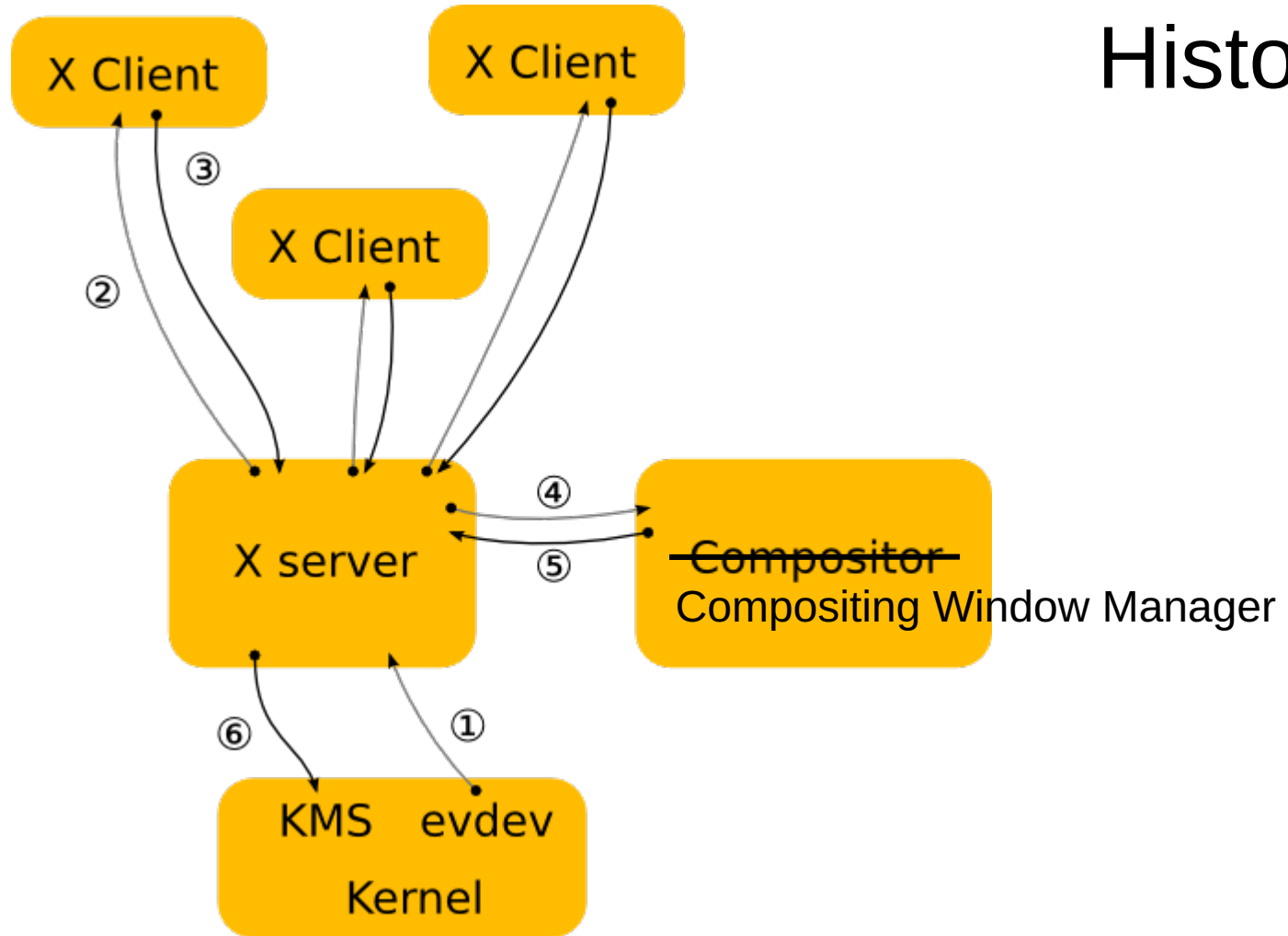
Historical



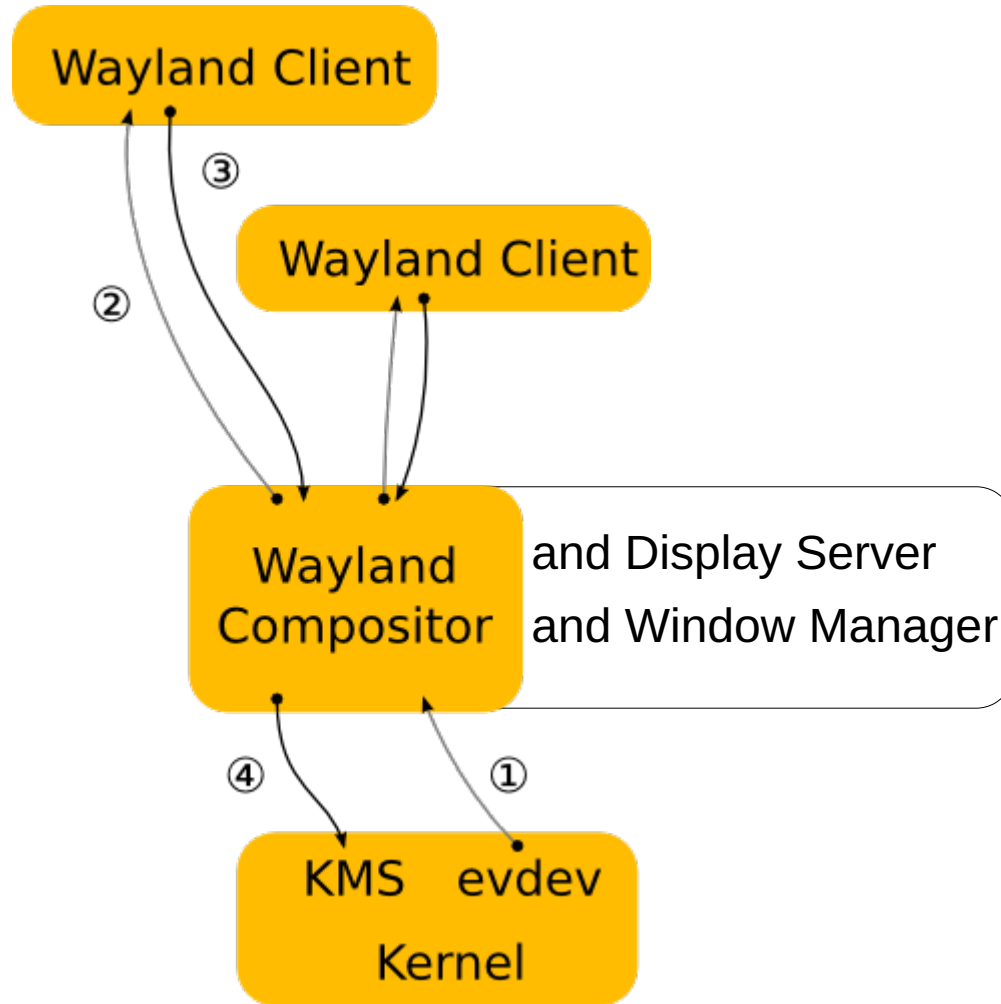
Historical



Historical



Status Quo



X11

- Embarrassing flickering/ flashing/tearing
- Extra round trip every frame
- Mechanism over Policy

Wayland Status Quo

- “Every frame is perfect”
- No extra round trips
- Policy over Mechanism

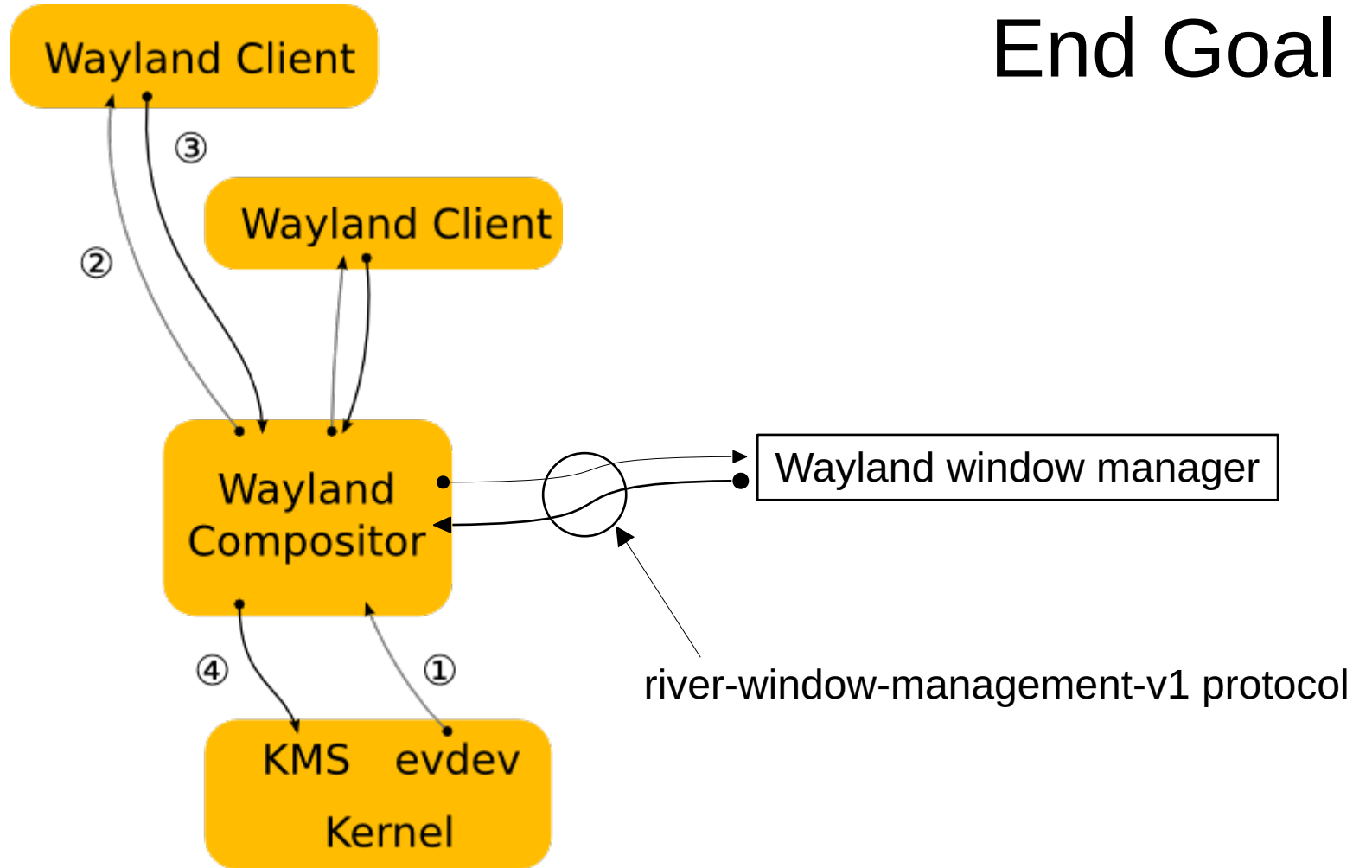
X11

- Embarrassing flickering/ flashing/tearing
- Extra round trip every frame
- Mechanism over Policy

Wayland Status Quo

- “Every frame is perfect”
- No extra round trips
- Policy over Mechanism
- Writing a window manager means writing a display server and compositor as well

End Goal



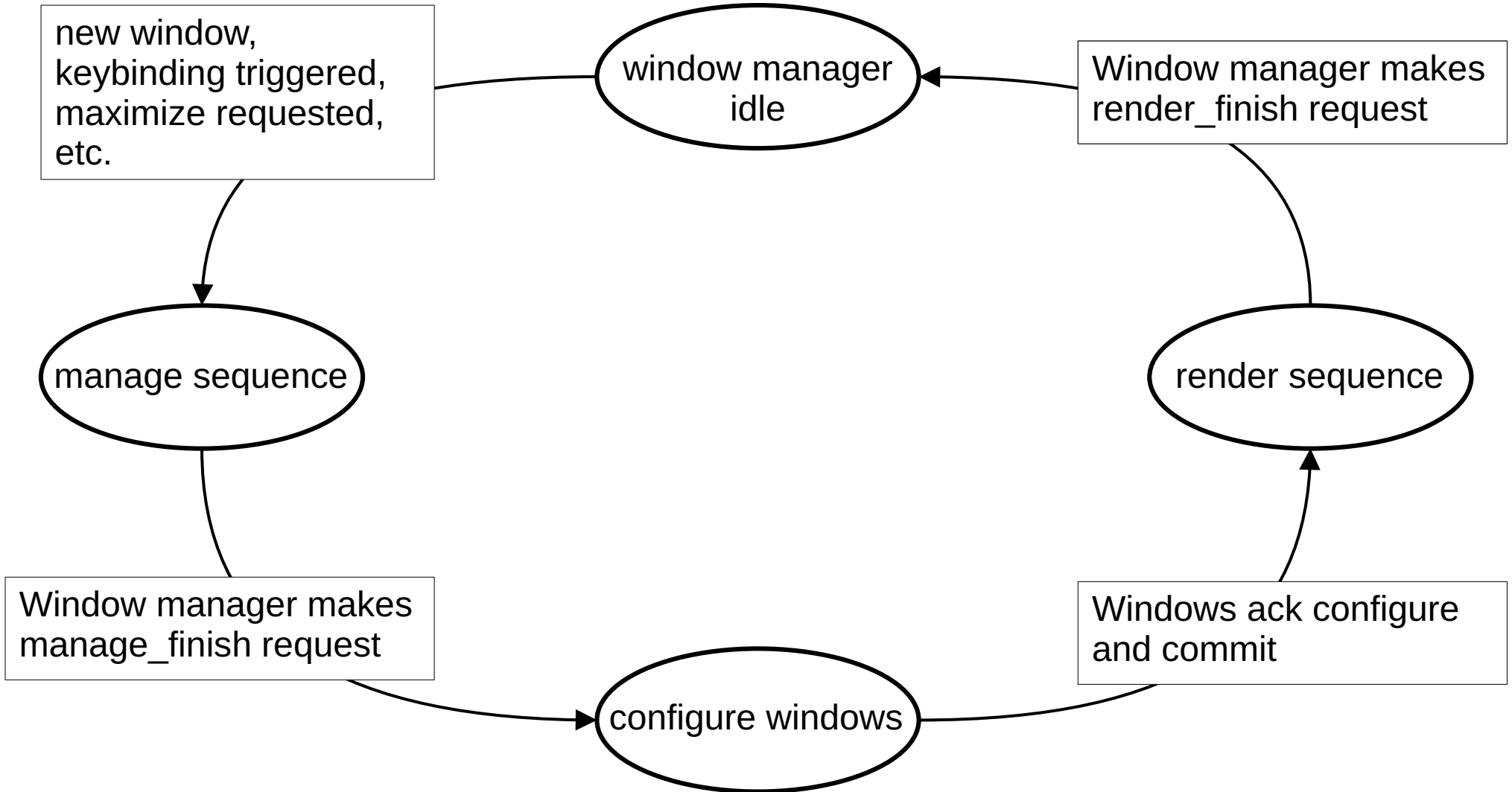
river-window-management-v1

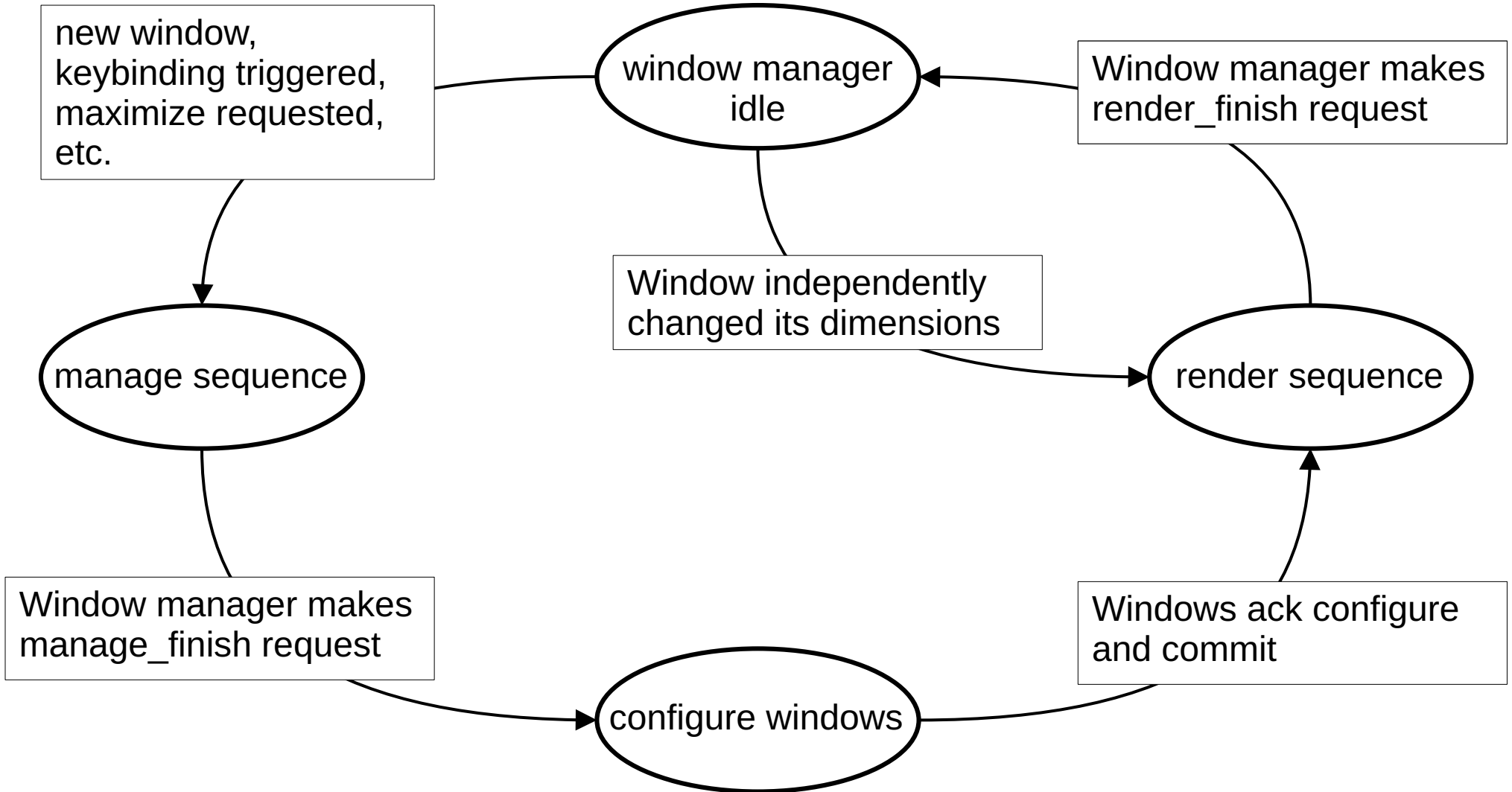
Design constraints (don't be like X11):

- Every frame must be perfect
- Minimally impact latency
- No round trip every frame/input event

river-window-management-v1

Window Management State	Rendering State
<ul style="list-style-type: none">• Window dimensions• Fullscreen• Keyboard Focus• Keybindings• etc.	<ul style="list-style-type: none">• Window position• Rendering order• Server side decorations• Window cropping• etc.





Why?

- Make writing Wayland WMs significantly easier
- Allow writing WMs in high-level languages without destroying latency
- Restart and switch between WMs without losing Wayland session
- Promote ecosystem diversity

Why not?

- Non-traditional environments (VR, etc.)
- Crazy effects (Wobbly windows, etc.)
- Wish to own entire stack
- Global complexity?

Why not?

- Non-traditional environments (VR, etc.)
- Crazy effects (Wobbly windows, etc.)
- Wish to own entire Stack
- Global complexity?
 - In practice, I think both compositor and window manager code is clearer

Current Status

- river-window-management-v1 is implemented on river's main branch, I'm using it right now
- At least 9 independent WMs were written in the 1.5 months since the protocol landed
- The protocol is stable, "We do not break WMs"
- Expect a river 0.4.0 release soon

Links

- Website: <https://isaacfreund.com>
- Mastodon: <https://hachyderm.io/@ifreund>
- Source code: <https://codeberg.org/river/river>
- IRC: `#river` on `irc.libera.chat`
- Zulip (new): <https://river-compositor.zulipchat.com>