Chromium Touch Pipeline

Making Life Difficult Since M18

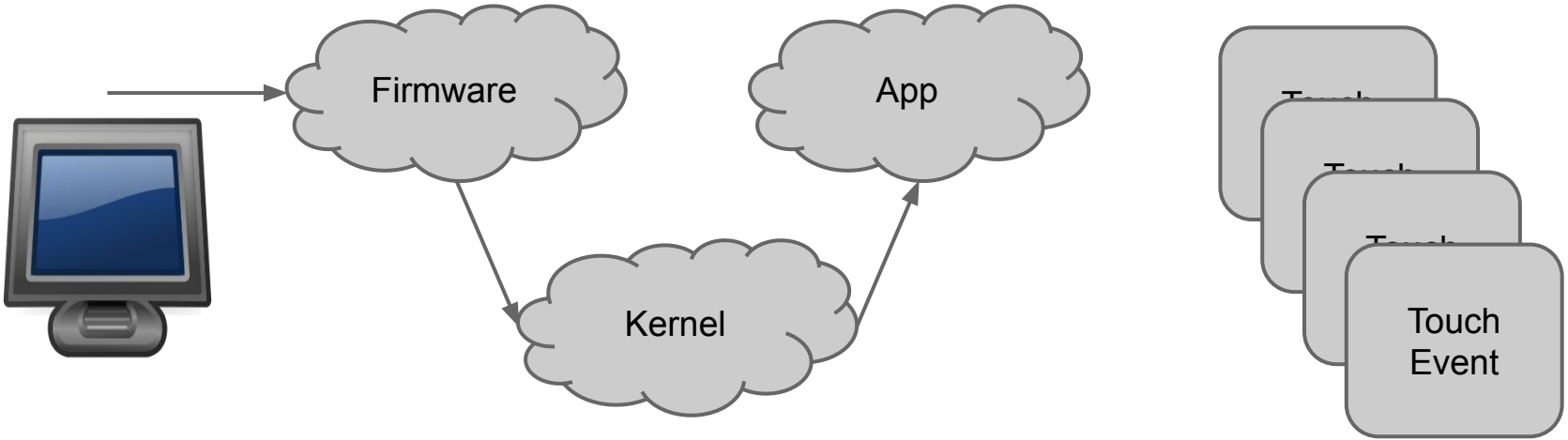
Overview

- Unified, Eager Gesture Recognition
- Input Trace
- Browser Pipeline
- Renderer Pipeline

Random Meme*

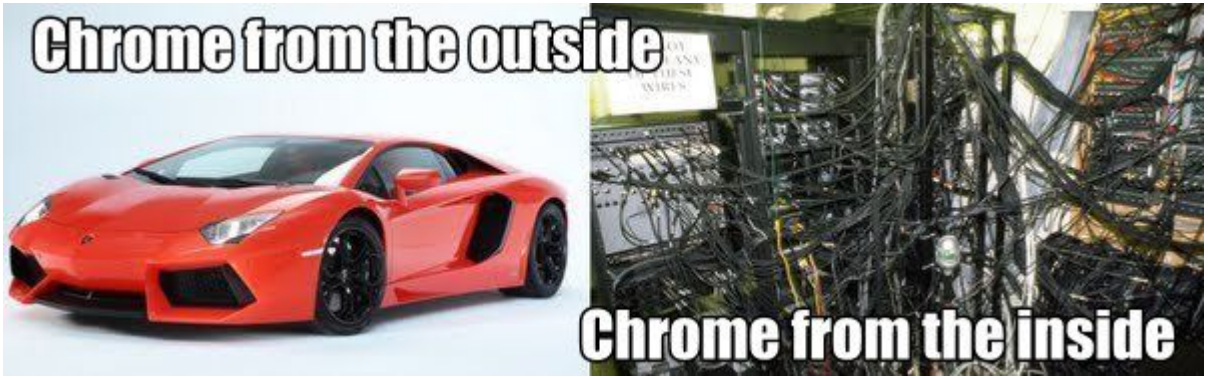
*All of the memes in this presentation have been shamelessly stolen from elsewhere

Touch Origins



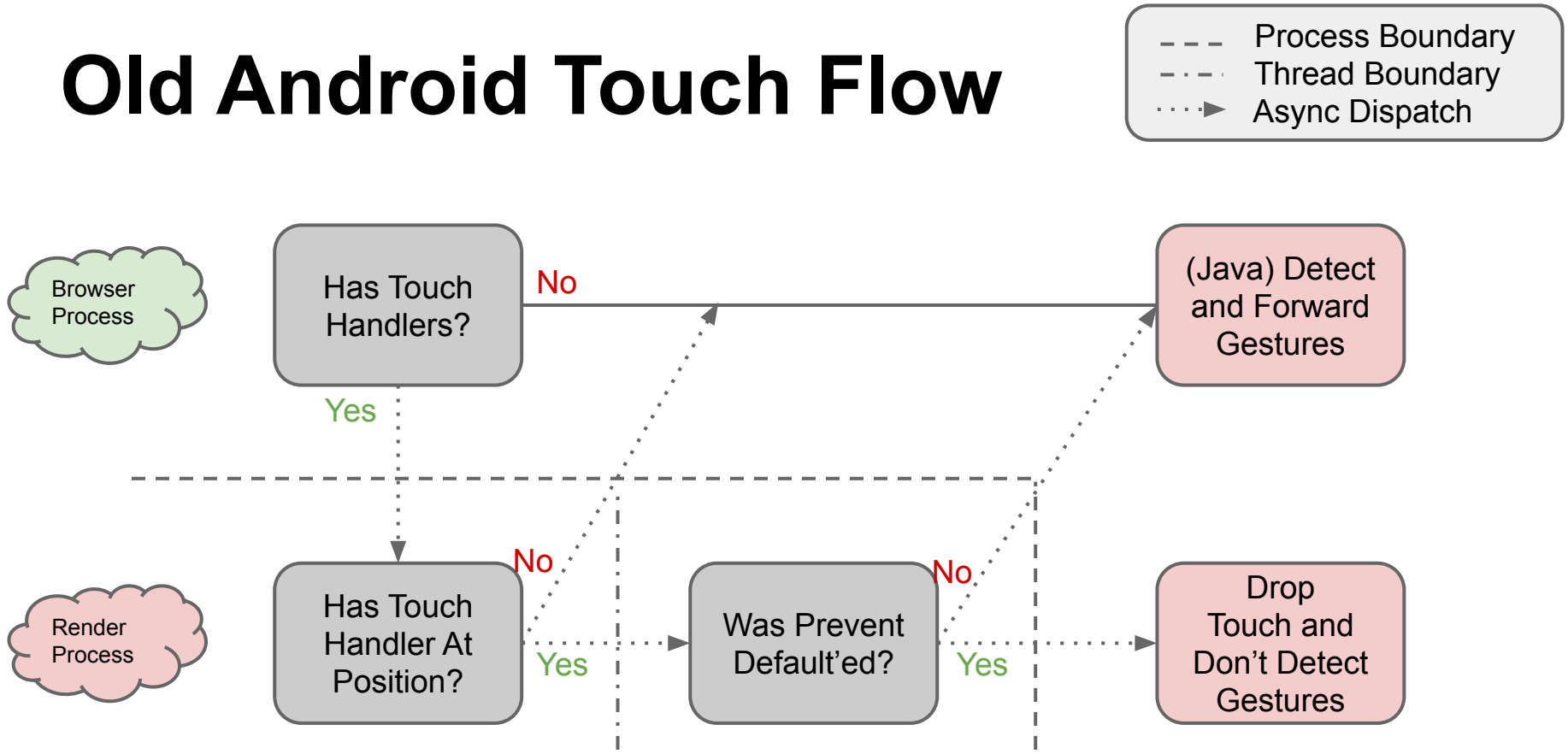
Touch Goals

- Deliver to the appropriate consumer
  - Content? Gesture Detector? Other?
- Deliver as fast as possible
  - Highly asynchonrous, many (potential) bottlenecks



Once upon a time...

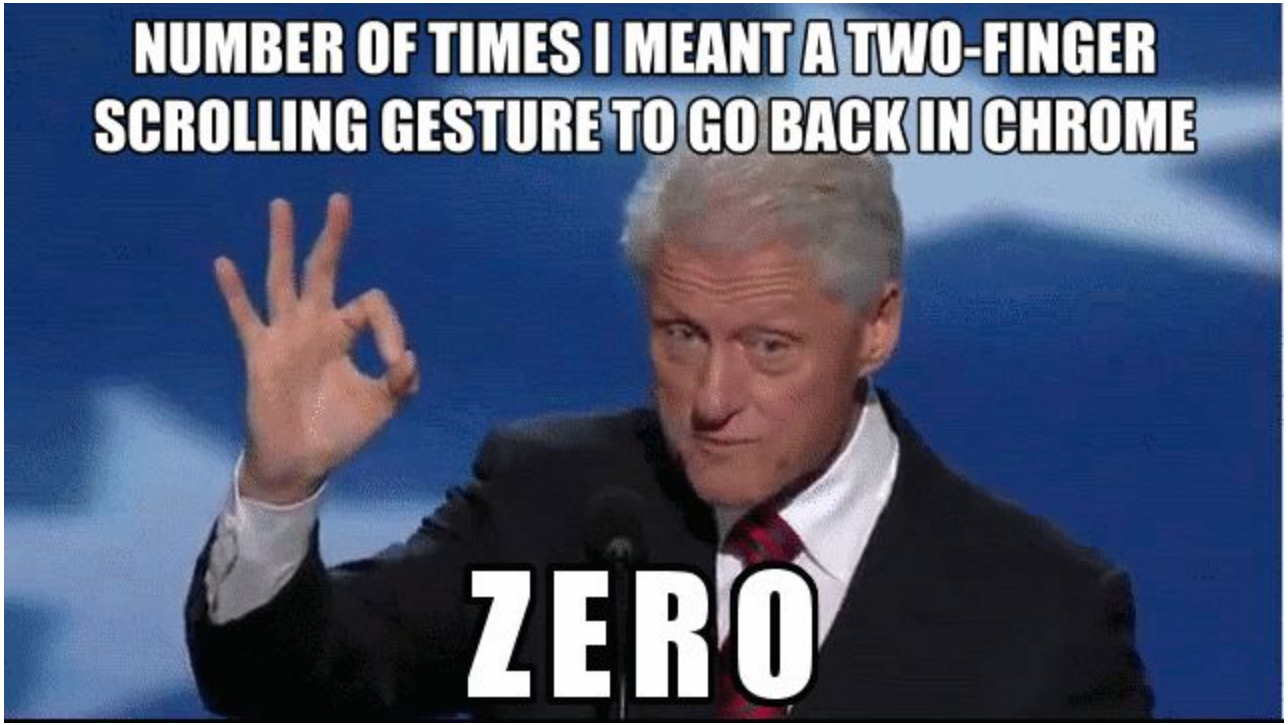
Old Android Touch Flow



Old Android Touch Flow - Problems

- Delayed response
  - Timeout-based gestures may fire before we receive a response (ShowPress/LongPress)
- Partial touch streams
  - Gesture detectors not written to process partial touch streams, yielding unpredictable gesture output and even crashes
- Slow
  - Java gesture detection overhead (20-30+ JNI calls)

Random Meme


NUMBER OF TIMES I MEANT A TWO-FINGER SCROLLING GESTURE TO GO BACK IN CHROME
ZERO

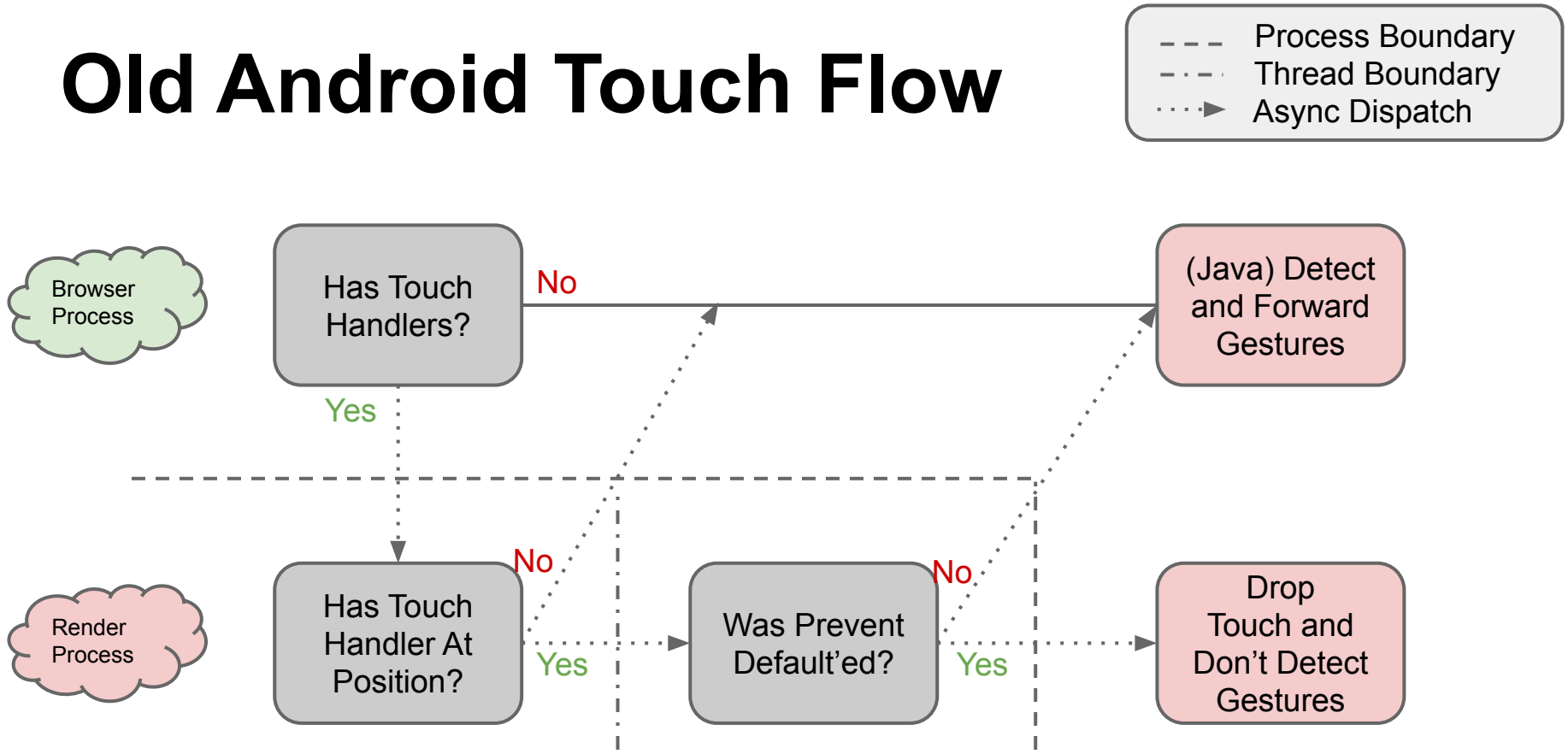
“Eager” Gesture Recognition (GR)

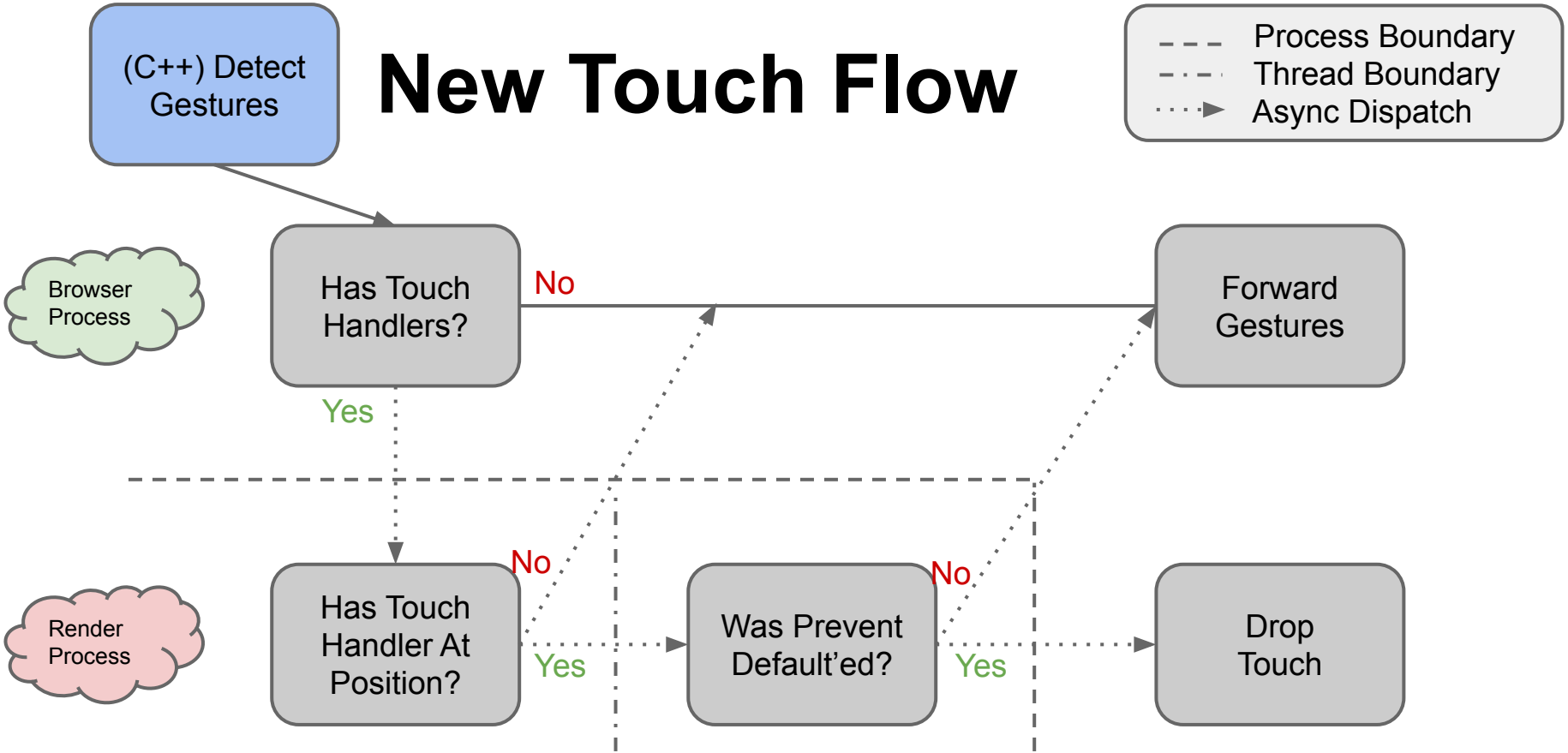
- Feed touch events to the the gesture detector immediately upon receipt
  - Generated gestures stored in buffer
- Intelligently dispatch queued gestures based upon touch response
  - Gesture even streams can be “repaired” as needed when a touch stream is partially consumed by the renderer

Unified Gesture Recognition (GR)

- Port Java-based Android gesture detection library to C++
  - Abstract all Android-specific assumptions/types
  - Allow platform-agnostic configuration of relevant constants, e.g., touch-slop region, timeout duration.
- Expose library to all platforms
  - Minimal dependencies
  - Can host gesture detection anywhere in the pipeline

Old Android Touch Flow

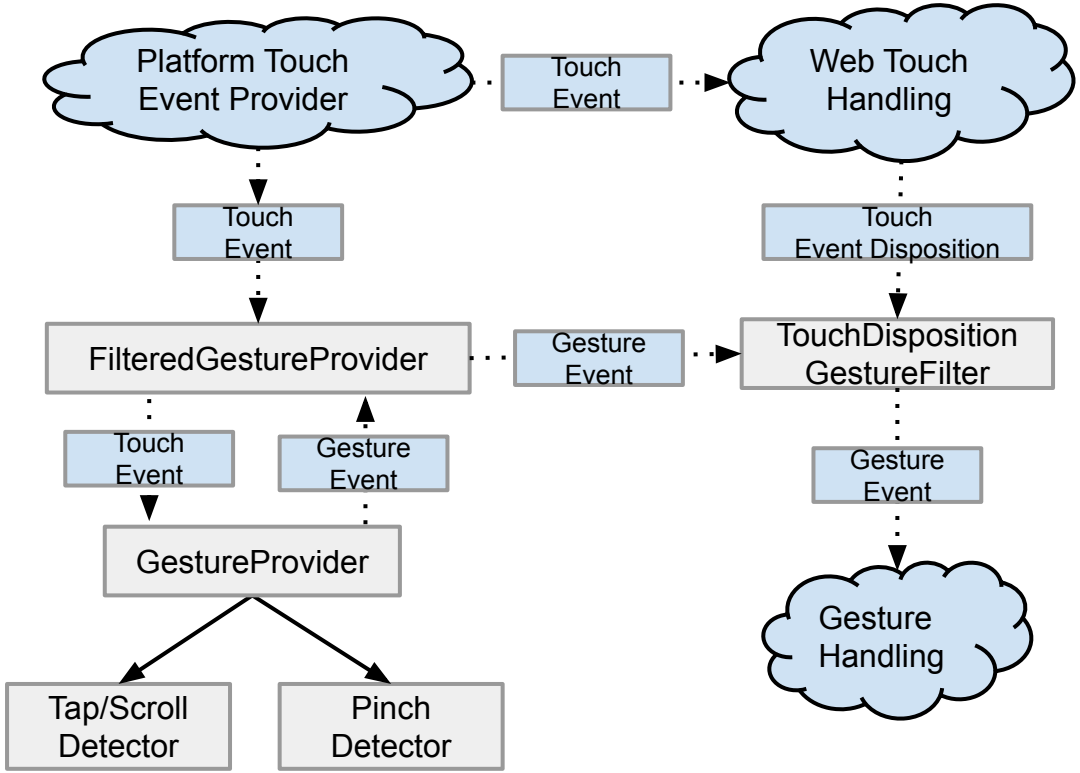



(C++) Detect Gestures
New Touch Flow
Process Boundary
Thread Boundary
Async Dispatch
Browser Process
Has Touch Handlers?
No
Forward Gestures
Yes
Render Process
Has Touch Handler At Position?
No
Yes
Was Prevent Default'ed?
No
Yes
Drop Touch

Trace!

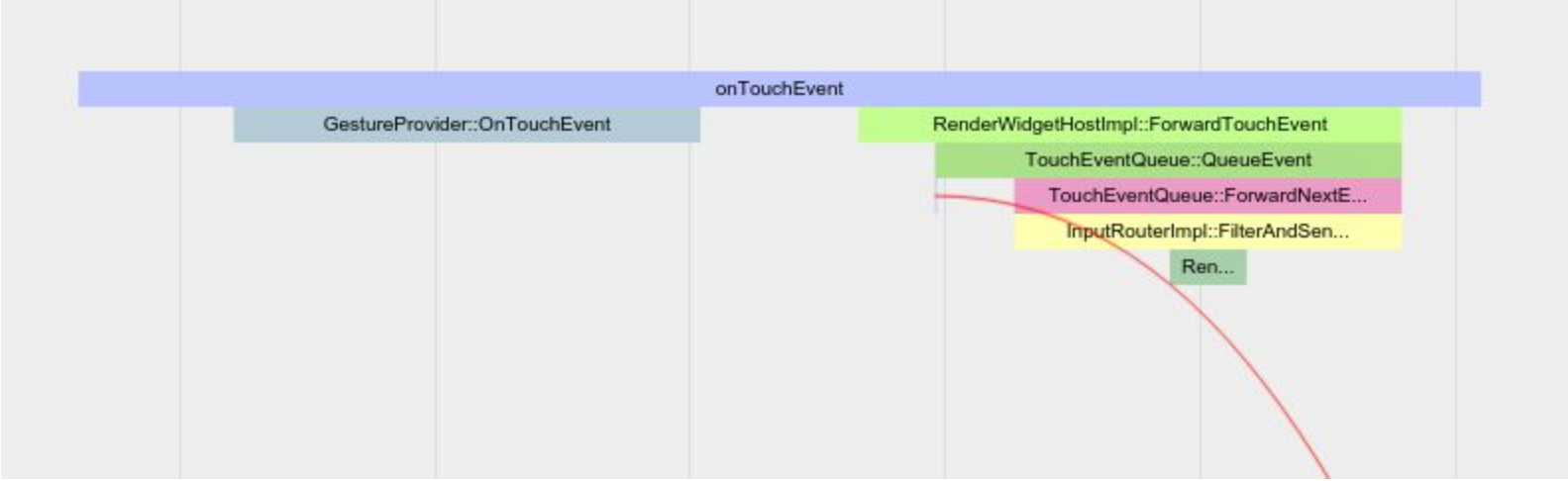
NYTimes scroll sequence

Browser Touch Pipeline Overview



Questions?

Browser Touch Pipeline



GestureProvider

- Wrapper for GestureDetector and ScaleGestureDetector
  - Direct ports from Android
- Synthesizes concrete gesture events
  - Android gesture detectors are callback-based, need to package callbacks in a way that Chrome understands

TouchDispositionGestureFilter

- Stores gesture events created by the GestureProvider in discrete packets
- Forwards/Drops the packeted gestures depending on the touch handling disposition
- “Repairs” the gesture event stream as appropriate when gestures are filtered
  - e.g., inserts TapCancel or ScrollEnd as necessary.

TouchEventQueue

- Staging area for touch event forwarding
  - At most 1 (cancelable) touch event in-flight
- Coalesces adjacent, compatible touchmoves
- Filters events when appropriate:
  - Touchmoves within the slop region
  - Touch pointer has no consumer
- Handles touch ack timeout
  - Only for Android on desktop sites

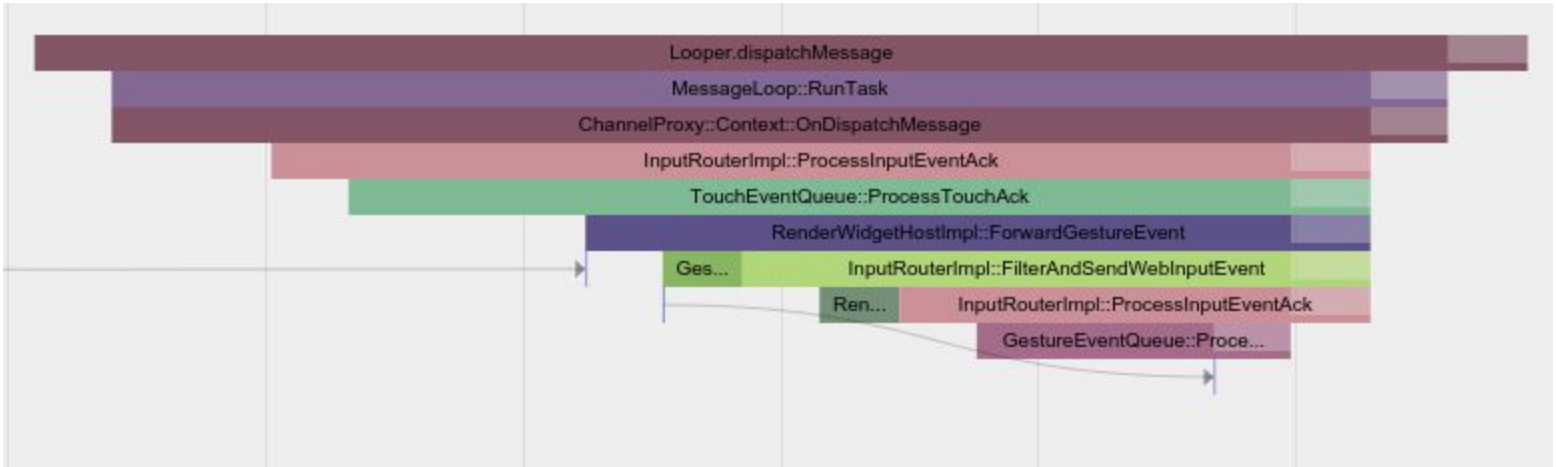
Aura vs Android

- Android:
  - GestureProvider attached to the view
    - Native UI handles touches above and separately from the content event stream
  - Platform buffers touchmoves and aligns dispatch with vsync
- Aura:
  - ?

Random Meme


There are no performance problems in Aura
Any jank is due to either debug build or CamStudio

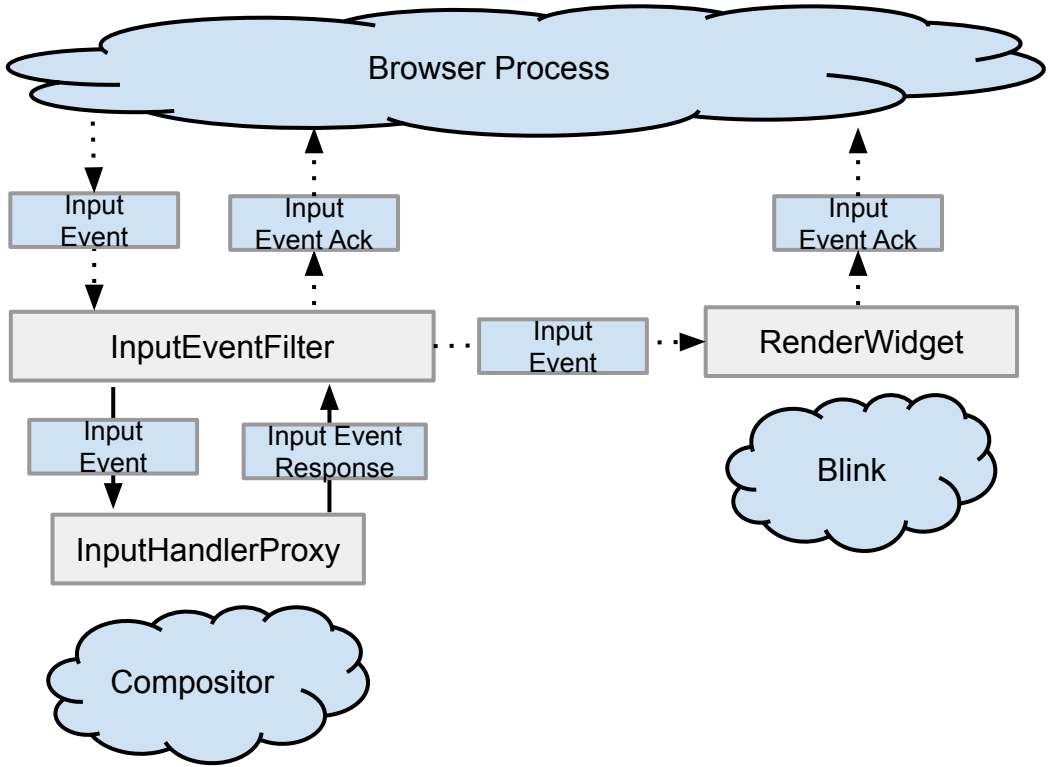
Browser Gesture Pipeline



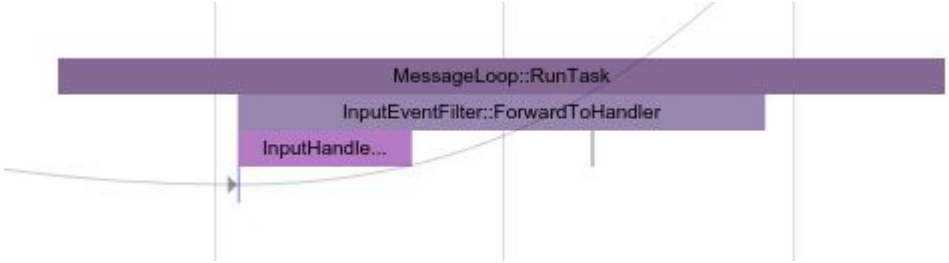
GestureEventQueue

- Staging area for gesture event forwarding
  - At most 1 (ack-respecting) gesture event in-flight
    - Exception is ScrollUpdate + Pinch Update
  - Some gestures “one-way” without requiring an ack
    - TapDown, TapCancel, ScrollEnd, etc...
- Coalesces adjacent, compatible scroll and pinch updates
- Filters events when appropriate:
  - Redundant GestureFlingCancel

Renderer Input Pipeline Overview



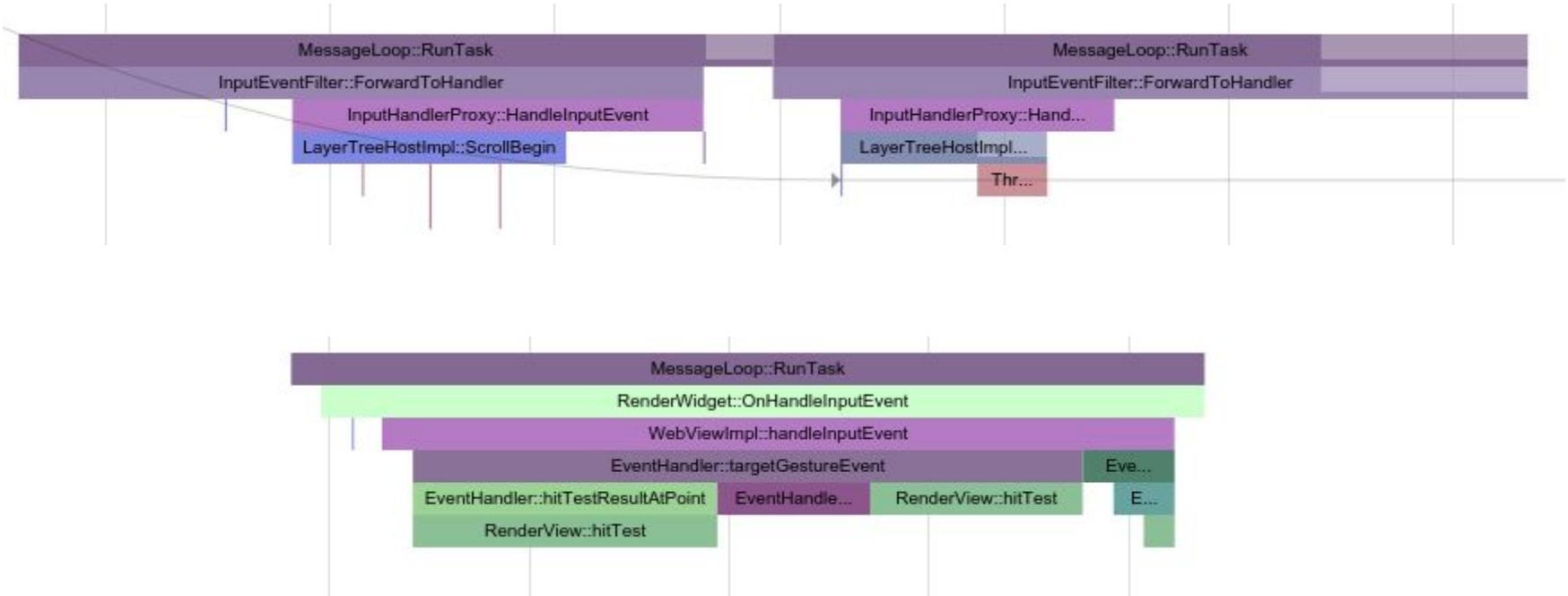
Renderer Touch Pipeline



Random Meme


yo dawg I herd you like JavaScript
so I put JavaScript onScroll so
you can jank while you scroll

Renderer Gesture Pipeline



InputEventFilter

- Render process IPC filter for intercepting input
- Posts *all* input events to the compositor thread
  - Maintains (loose) event ordering
- Touch/Gesture/Wheel/Mouse/Key events offered to the InputHandlerProxy
  - Forwarded to the main thread if unhandled
  - Otherwise returns ack to browser (if necessary)

InputHandlerProxy

- Gateway to the compositor
- Compositor can:
  - Indicate presence of touch event handler
  - Handle (most) scroll events
  - Handle (all) pinch events
- Manages compositor-handled flings
  - Fling boosting, animation, etc…

RenderWidget

- Delegates events unhandled by the compositor to Blink (via WebViewImpl)
  - RenderWidget::HandleInputEvent is a monster, somebody please slay this beast
- Reports ack to the browser (if necessary)

Depressing Numbers

- Roundtrip costs (excluding frame generation):
  - Initial touchstart: 2 process hops + 4-5 thread hops
    - Touch: ui → io ⇒ io → impl (→ main) → io ⇒ io → ui
  - Initial touchmove scrolling response: 4 process hops + 8-10 thread hops
    - Touch: ui → io ⇒ io → impl (→ main) → io ⇒ io → ui
    - Scroll: ui → io ⇒ io → impl (→ main) → io ⇒ io → ui
  - Scrolling: 2 process hops + 4-5 thread hops
    - Scroll: ui → io ⇒ io → impl (→ main) → io ⇒ io → ui

Open Questions

- Can we bypass the IO thread?
- Can we handle touch events off the main thread?
- Can we bypass the impl thread?

Questions?