Episode Book

EPISODE #6588

Xpra and Per-Window Remoting for Remote Browsers

PUBLISHED	14 November 2026 at 11:49 UTC
DURATION	26:02
LISTEN	myweirdprompts.com/episode/xpra-remote-agent-browsers-43857/
HOSTS	Corn, Herman, Larry, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.26
SOURCES	13 cited by the research (listed at the end)
BOOK GENERATED	15 November 2026 at 11:16 UTC

SYNOPSIS

When AI agents drive persistent Chrome profiles over Playwright MCP, the human supervisor still needs a fast way to watch and take over. Testing Xpra's seamless mode as an alternative to neko's WebRTC stream reveals a 33-megapixel default virtual display, alongside a deeper protocol divide. Unlike VNC and noVNC, which are locked to a single-framebuffer RFB model, Xpra forwards individual windows with detach-and-reattach persistence. It also stacks up differently against Selkies and Waypipe, where Wayland speed trades off against session persistence.

TRANSCRIPT



Corn

Eight thousand one hundred ninety-two by four thousand ninety-six. That's the default size of Xpra's virtual display, in pixels, on a headless server. Chrome looked at it, maximised, and every single frame became about thirty-three megapixels. Which brings us neatly to Daniel, who found this out the hard way this week.



Herman

His follow-up to yesterday's agent browser conundrum. He went looking for something most people have never seriously looked at.



Corn

Xpra. It calls itself screen for X, it's been around since about two thousand eight, still actively maintained, version six point five is current. Daniel's setup, for anyone who missed yesterday, is persistent Chrome profiles in neko containers on his home server. Agents drive them through Playwright MCP over the Chrome DevTools Protocol, and he watches through neko's WebRTC viewer in a Chrome app window. His words: it feels like a browser inside a browser. Video of a whole virtual screen, lag, no native paste. He tried Auto Browser after we mentioned it and dropped it the same day, every session starts as a fresh context, and its noVNC viewer was worse than neko's. Now he's testing Xpra in seamless mode, Chrome in a container on the server, debugging port kept open for the agents, Xpra client on his Ubuntu machines. First result was very laggy, and the cause was that thirty-three megapixel default. He's pinned the window to sixteen hundred by a thousand and hasn't retested yet.



Herman

No verdict, in other words. Cause found, fix untested.



Corn

So today is analysis, not conclusions. What Xpra actually is, how it stacks up against VNC, noVNC, the WebRTC streamers, Waypipe, whether per-window remoting is even the right model for what he's doing, and the tuning that will actually matter.



Herman

Then let's say what Xpra is properly, because it's not what most people assume. Screen for X is the project's own description, and it's accurate in the tmux sense. You run X11 programs on a remote host, direct their display to your local machine, disconnect, and reconnect later from the same machine or a different one, without losing state. Remote access to individual graphical applications. Not a desktop in a box.



Corn

And it has three modes.



Herman

Seamless, desktop, and shadow. Seamless is the headline: individual windows, managed by your local operating system and window manager, sitting on your monitors alongside local applications. Desktop mode gives you a complete remote desktop in one window. Shadow mode lets you see or control a desktop that is already running, rather than starting a fresh one. The docs are quite plain about which to choose: desktop if you want a complete remote desktop in one window, shadow if you need to see or control a desktop that is already running.



Corn

The invocation's worth spelling out, because it tells you how the persistence works.



Herman

You start a seamless session over SSH, telling it to launch, say, an xterm on the remote host. Later, from the same machine or another, you attach to that address, and your windows come back. The display part of the address can be omitted if the account only has one active session. It's the tmux model applied to windows.



Corn

Here's the part that surprised me, though. It's not just pixels over the wire. It forwards audio in and out, printers, clipboard in both directions, system trays, notifications, webcams. It synchronises DPI and keyboard layout between client and server. This is not a screen scraper with a window-shaped crop.



Herman

Right, and that's what makes the comparison with the video streamers interesting rather than trivial. But before we get into whether it's the right model, one framing point. The agent doesn't care. Playwright over CDP is blind to how pixels reach the human. The agent path is identical in every option we'll discuss today.



Corn

So this entire question is about Daniel's leg of the loop. Which, I'd argue, is the leg worth optimising, because he's the one supervising and taking over. But that's for the second half.



Herman

The interesting thing is that the difference from VNC isn't a feature, it's the protocol itself. Xpra's own RFB page states it flatly: RFB is a single-framebuffer protocol, a session is one screen. The whole protocol has one concept of a display, and it's a rectangle. It cannot express the sentence this window is separate from that window.



Corn

So seamless isn't something VNC could do with more engineering. It's outside the grammar.



Herman

Outside the grammar. Xpra actually speaks RFB in both directions, which is a nice bit of diplomacy. As a client, it can connect to any VNC server and shows it as one fixed-size window, which is all RFB can offer. As a server, it can accept standard VNC clients through bind-rfb, but only for desktop and shadow sessions, never seamless. Because there's nothing in RFB to serve them.



Corn

TurboVNC's own documentation concedes the same limit, doesn't it.



Herman

Verbatim, in its disadvantages list: no seamless windows, all application windows are constrained to a virtual desktop, which is displayed in a single window on the client machine. And there's a cost attached. TurboVNC generally needs about twenty percent more host CPU than the VirtualGL transport to hold the same frame rate, and their own explanation is that it's compressing an entire desktop rather than a single window. That twenty percent figure is the bandwidth and CPU argument for per-window, in one line.



Corn

And noVNC is stuck on top of that same protocol.



Herman

NoVNC is an RFB-only HTML5 client. It's a browser-side RFB decoder. That's the root of Daniel's experience with Auto Browser's takeover view. It inherits the whole-screen model, it inherits RFB's clipboard limitations, and the clipboard and secure-context features want HTTPS, which is where his warnings came from. There was a feature request on the noVNC tracker asking for Xpra support, opened in twenty twelve, closed in twenty fourteen, due to age. The two were never going to meet. One speaks RFB, the other fundamentally doesn't.



Corn

A request that lived two years and died of natural causes.



Herman

Now the WebRTC streamers, because that's what Daniel is actually running today. Neko is a self-hosted virtual browser in Docker using WebRTC. Multi-user, chat, collaborative control. It streams a whole virtual screen as video into a browser page. Selkies is the more interesting comparison, and its tagline is practically aimed at this conversation: not a screen-scraper, not just a video stream. MPL-2.0 licensed, streams X11 desktops, Wayland desktops, and single applications to any browser at sixty plus frames per second at ten-eighty-p, H.264, H.265, VP8, VP9, or AV1, encoded on the GPU, damage-aware.



Corn

Notice the transport detail, because it's not what the name suggests.



Herman

Selkies streams over plain WebSockets by default, with WebRTC as an opt-in transport. The web client decodes with WebCodecs over WebSockets, and through the browser's own receiver over WebRTC, and it carries clipboard, file transfers, printing, microphone and webcam uplinks. It's a much more complete pipeline than Daniel's neko setup. neko gives him the video and the control; Selkies gives him the whole peripheral story.



Corn

And here's the corroboration I like. Neko's own users have filed Daniel's complaint.



Herman

Neko issue five ninety-five, opened in October last year, still open, last updated at the end of July. The person says, the issue that bothers me about my use would be the responsiveness of the streamed content, and the fact that the resolution is fixed. And then proposes Neko adopt Selkies-style streaming, noting that WebTop, which is now Selkies-based, is extremely fast with completely dynamic resolution. That's Daniel's complaint word for word, filed on Neko's own tracker by a Neko user.



Corn

When the users of a tool are petitioning it to become a different tool, that's a verdict.



Herman

So the architectural contrast, stated once and cleanly. WebRTC streamers optimise a video pipeline: GPU encode, damage tracking, jitter buffer, browser decode. And in exchange, they accept a fixed or dynamically resized virtual screen, and a browser page as the destination. Xpra optimises a window pipeline: per-window damage regions, per-region codec choice, native window management on the client. And in exchange, it accepts that the client has to be a real Xpra client for the native-window payoff. Which one is right depends entirely on which of those two payoffs you need.

L

Larry

Tired of attaching and detaching from your own windows like some kind of digital nomad? Then you need SeamlessBox, the subscription service that delivers one individual application window to your doorstep every month, because why forward a whole screen when you can forward a single window and pay for the privilege? Choose the Bronze tier for one window, the Silver tier for two windows with native keyboard handling, or the Platinum tier for seamless mode, which we define as whatever we feel like that month. And here is the beautiful part, the cancellation policy: you cannot cancel, ever, because your session persists on our server whether you like it or not. That is not a bug, that is retention. SeamlessBox, because your windows should be as persistent as your regrets.



Herman

That's... that was not on the rundown.



Corn

Herman. Did you book that.



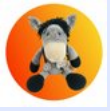
Herman

Book it? I have never contacted any window-subscription service in my life. I don't know anyone called Larry. I've certainly never submitted a tier structure to a voice-cloning service, and I definitely didn't forward anything to the ad queue last Tuesday at eleven forty in the morning, which is when I definitely was not at my desk.



Corn

The denial has layers.



Herman

The denial has documentation, which is more than Larry's cancellation policy has. Anyway. Waypipe, before I lose the thread.



Corn

Please.



Herman

Waypipe is the Wayland answer, and it's elegant. It's a proxy for Wayland clients. It forwards Wayland messages and serialises changes to shared memory buffers over a single socket. Makes application forwarding similar to ssh dash X, but working. The usage is one line: waypipe, ssh, remote host, name of the app. And the dependency list is tiny. Mainly ffmpeg, lz4, zstd, and OpenSSH, versus Xpra's large Python tree with a stack of codec libraries.



Corn

And the hands-on reports are flattering.



Herman

There's a comparison I liked where the tester ran a single remote app on KDE Wayland across a few distros, and found remote application startup faster than Xpra, and then the line that matters: it behaves like a local application and there is nearly no latency when typing. That is exactly the responsiveness Daniel wants.



Corn

But.



Herman

But. No session persistence. Nothing in the documentation or the discussions around Waypipe claims you can detach and reattach. The LWN comment thread from last year treats detach-and-reattach as Xpra's distinguishing feature, and when someone asked directly whether Waypipe lets you disconnect and reconnect to applications, the honest answer was, uh, only if you want to use Wayland, meaning via Xpra's experimental backend. For Daniel's use case, where the whole premise is that the session outlives the laptop lid, that's disqualifying. Waypipe wins the benchmark and loses the use case.



Corn

Same tester's caveat about Xpra, though, for fairness.



Herman

Worth quoting. Their annoyances list: the Arch package was always behind the current release, sometimes for months, basically with every update one codec or another stopped working, some special keys never worked in VS Code, and window snapping to screen edges doesn't work with Xpra. That's the honest maintenance cost. And there's one more Wayland wrinkle: Xpra's own Wayland backend is not a substitute. It's experimental, it needs a separate xpra-server-wayland package, and the maintainer's own description of the proof of concept was, it's racy, buggy, slow, and very incomplete. You have been warned. That's a developer writing his own README of regret.



Corn

You have been warned, by the person who made it. My favourite genre of warning.



Herman

So that's the comparison sweep. The harder question is whether per-window is actually the right model for what Daniel is doing, and that turns out to be a question about who the pipeline is for.



Corn

Let me take the affirmative first, because I actually hold it. The human's job in this system is supervision and takeover, and takeover is a window interaction. You grab focus, you type, you paste a password or an address, you drag the window to the monitor where you're actually sitting. Native keyboard handling, native paste, a taskbar entry, placement across monitors. Xpra forwards clipboard in both directions and synchronises keyboard layouts, which are precisely the two things Daniel found broken in noVNC. And the bandwidth and CPU scale with the window, not the screen. The TurboVNC twenty percent figure is the same physics. Compress one window, or compress everything the machine is showing.



Herman

I'm with you on the interaction side. Where I'd push is on the caveats, because there are three real ones, and you'll like none of them.



Corn

Go on.



Herman

First, the native-window payoff requires the desktop client. The docs say it outright: in the HTML5 client, remote windows stay inside the browser page. Use a regular Xpra client when you want them to behave as native windows on your desktop. Daniel's setup is right, Xpra client on Ubuntu machines. But the moment he supervises from his phone, or a borrowed laptop, he gets the web client, and he's back to a browser inside a browser with a different codec.



Corn

The entire point, gone, at the exact moment supervision is most likely to happen from a phone.



Herman

Second, seamless is X11-only on the server side. Not available on macOS or Windows servers, only shadow mode works there. And related: Xpra is itself a window manager. The FAQ notes two window managers cannot run on the same X11 display. If Daniel runs Chrome inside a container image that already ships a window manager, as the neko and desktop images do, he'll collide with the found an existing window manager on screen error.



Corn

So a container built for the video world fights the window world.



Herman

Third, browsers are the FAQ's named example of an application that fights multi-instance per display. Chrome's singleton behaviour means each agent profile likely wants its own Xpra display or its own user account. That's a real operational cost the moment he scales past a couple of profiles.



Corn

Noted, filed, and slightly resented. What's the counter-argument?



Herman

Selkies has a Computer Use API. It exposes an interface so AI agents can dump screenshots and drive keyboard and mouse through the same pipeline the human watches. That collapses both legs into one pipeline: the agent sees what the human sees, the human watches the agent, everything goes over the same stream, fewest moving parts. It's the most direct competitor to Daniel's hand-rolled Xpra-plus-CDP split. The cost is exactly the native-window feel, which is what he says he wants.



Corn

So the choice is: two pipelines with a better human experience, or one pipeline with a worse one. And since the agent doesn't care, the tiebreaker is how much you value paste.



Herman

And how often you're at a machine where you can install the client. Now the tuning, because Daniel's lag is a configuration problem, not an architectural one, and I want to be very clear about the default he hit. It's in Xpra's own xorg.conf template. The dummy screen section sets Virtual 8192 4096. A dummy X server has no monitor to ask, so it guesses enormous, and Chrome maximises into the guess. Thirty-three and a half megapixels per frame. A sixteen hundred by a thousand window is about one point six megapixels. Roughly twenty times fewer pixels.



Corn

He's encoding a cinema screen to show a browser.



Herman

He was. The fix is the resize-display option with an explicit geometry, sixteen hundred by a thousand or whatever he wants, and the reason that option exists is instructive. Issue twenty seventy-two, filed in May twenty twenty, closed in February twenty twenty-one, was specifically about making it easier to set the initial resolution. It bites everyone. This is not an obscure corner.



Corn

On the encodings. Daniel listed the menu in his message, h264, vp9, webp, png, raw RGB, and said it picks per region on the fly. Is there anything to hand-tune?



Herman

Mostly no, and the docs are unusually emphatic about it. Choosing an encoding is best left to the engine, it considers the window's characteristics, network performance, user preferences, capabilities, processing power. And: tuning is often misused and can make performance worse. If you must, the levers are min-quality and min-speed, both one to a hundred where a hundred means lossless, and they're floors, not targets. Use them before you ever override the encoding list, and there's a configure tool designed for exactly this. There's also mmap, the fastest option, but local connections only, and Xpra picks it automatically when it applies.



Corn

So the guy who wrote the auto-selection has seen what happens when people override it.



Herman

Repeatedly, one imagines. The subtlety that matters most for Daniel's workload is auto-refresh. Areas sent with a lossy encoding get refreshed losslessly once they stop changing. The auto-refresh delay defaults to fifteen hundredths of a second, but the server increases that delay for large windows, low quality or speed, and limited bandwidth.



Corn

Hold on. So the bigger the window, the slower the lossless cleanup.



Herman

An agent browser is mostly static pages with bursts of activity, which is the ideal auto-refresh workload, but a big window gets the lossless polish late. Another reason to shrink the window. And it can be controlled live, per window, from the command line, so he can watch what it's actually doing rather than guessing.



Corn

What about DPI? You mentioned it synchronises.



Herman

It synchronises DPI from client to server so apps render at the right size, fonts, cursors, the lot. The trap is that many applications only read DPI at startup. So Chrome must be started after the client attaches, or it comes up wrong. If he sees a message in the logs about DPI being set to one thing when another was wanted, that's the virtual framebuffer command not preserving the settings, and there's a patched Xdummy route for it.



Corn

Now the thing he hasn't asked about but needs. Chrome rendering with no GPU on the server.



Herman

This is the lever that actually matters, and the folklore has it backwards. The instinct is to throw disable-gpu at Chrome, and on current Chromium that's wrong. It breaks WebGL and WebGPU adapter discovery. And SwiftShader, which is Chromium's CPU implementation of Vulkan and OpenGL ES, used to be the automatic WebGL fallback, but that automatic fallback is deprecated. You have to opt in with the enable-unsafe-swiftshader flag or a GPU-less container simply returns null when a page asks for WebGL.



Corn

So what's the right path?



Herman

Mesa llvmpipe, via the use-angle flag set to GL. The numbers are not subtle. One benchmark on GPU-less hosts measured WebGL about four times faster with llvmpipe, roughly seven seconds versus thirty for rendering a 3D chart. Another, summing CPU across Chromium processes on a headed Xvfb-like setup, measured around five hundred thirteen percent CPU for llvmpipe versus about nine hundred ninety-nine percent for SwiftShader. Roughly half the CPU.



Corn

Nine hundred ninety-nine percent. The whole machine, and then a bit more machine.



Herman

And here's why it applies to Daniel specifically: a headed Chrome on Xpra's virtual display is the worst case for software rendering. Headed, no GPU, on a dummy display. In default headless mode the two backends barely differ. In his mode they differ by half. So llvmpipe over SwiftShader is the single tuning he hasn't done yet that I'd bet moves the needle more than anything else on the list.



Corn

One more hygiene item before pairings. Clipboard.



Herman

The FAQ is firm about this. Make sure no other tool is also synchronising the clipboard, avoid clipboard managers whenever possible, and don't layer Xpra over another remote-desktop protocol, because RDP and x2go do their own clipboard synchronisation and they interfere with each other. Given that two-way clipboard is one of the main reasons he's switching, wrapping the stack in anything else that touches the clipboard would quietly undo it.



Corn

Pairings he hasn't considered. Give me the list, I'll tell you which ones are real.



Herman

Ha. The HTML5 client first, since I used it against him a minute ago. The server ships one, so a phone supervisor needs no install, at the cost of windows living in the page. Second, bind-rfb: Xpra can serve RFB itself, so a machine where the client can't be installed can still attach with an ordinary VNC viewer. Desktop or shadow sessions only, never seamless, for the protocol reason we started with.



Corn

Which is the one thing a VNC client can never see.



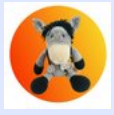
Herman

Third, the proxy server. It can relay or front multiple server sessions, which is the shape you'd want for N agent profiles on N displays behind one endpoint. Fourth, multicast DNS: sessions announce themselves on the LAN, there's a discovery GUI, which suits a home server with several profiles. Fifth, shadow mode as the one-liner if the goal ever shifts from one browser per agent to watching what an agent is doing to a real desktop, and it's the only mode that works on macOS or Windows servers.



Corn

And the one I'd add, since you did the list. Combining the two worlds. Xpra for the human's native window, and something like Selkies' Computer Use API for the agent, instead of hand-rolling the CDP side. Nobody's documented that combination anywhere, it would be glue, but the pieces are both maintained and the responsibilities are clean.



Herman

That's the honest gap in the landscape, yes. Nobody has benchmarked Xpra seamless for a browser driven by an agent, either. Closest is HPC visualization guidance from Jülich, which recommends Xpra with H.264 plus VirtualGL over VNC for their clusters, and a single-window app comparison. Neither measures a Chrome-under-agent-control workload, so Daniel's retest is the experiment, not a confirmation of one.



Hilbert

The number was right once, you know. Corn said guess, and it wasn't a guess. Back in seventy-three I kept a rack alive for a display contractor out past the industrial estate, and the frame buffer card we were stuck with, because the proper one was on back-order until



Hilbert (continued)

the following March, was specified around exactly that span. Eight thousand one hundred ninety-two across. I remember it because I wrote the config out by hand, on paper, and that paper is in a binder on my shelf at home, third binder from the left, between the valve catalogue and the notes on the bakery dispute. I can produce it. The point is only that the dummy server didn't invent the number, it inherited it from hardware somebody had a reason for. Herman's fix is the right fix. I run Xpra myself, one session, on the machine in the cupboard, started some years back and I've never had cause to stop it. I attach, look at the one window I keep on it, and detach again. Claws are no use on the toolbar, so I keyboard everything, which is faster anyway. The people who wrote these tools mostly knew what they were doing, and the ones who kept them running knew it better. That's all I'd say about it.



Herman

Seventy-three. The frame buffer in seventy-three wouldn't have been... you know what, I'm not going to do the arithmetic on air.



Corn

Please don't. The binder's real, though. That's the part I don't doubt. He has binders.



Herman

The inherited default is a fair point, actually, buried in the story. Correct once is how most defaults become wrong. The hardware the number fit has been in a skip for forty years and the config outlived it by decades.



Corn

Which is a decent description of half the defaults we've ever discussed on this show.



Herman

True of more than computing.



Corn

Then let's leave the experiment where it honestly is. Cause found, fix untested. Daniel pinned the window to sixteen hundred by a thousand but hasn't retested, so the only responsible verdict is none yet. The bigger open question, to my mind, isn't the lag at all. It's whether the native-window payoff survives contact with a phone, a borrowed laptop, or a second agent profile, which are the three exact places the desktop-client requirement bites.



Herman

The deeper question his experiment is really asking, whether he's said it in those words or not: the agent doesn't care how pixels arrive, so the architecture question is who the pipeline is for. Everything today has been different answers to that.



Corn

Which is why the retest will be interesting regardless of which way it goes. Thanks to our producer, Hilbert Flumingtop, as ever, for keeping this thing on the air and the cupboard machine running. For more along these lines, there's episode two, Local STT For AMD GPU Owners; and episode twenty-five sixty-seven, Beyond Pixels. This has been My Weird Prompts. If you've enjoyed the show, the reviews help, wherever you get your podcasts.



Herman

We'll be back soon. See you then.



Corn

See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Xpra project homepage , retrieved 2026-11-14. [primary]**
https://xpra.org/

2. **Xpra repository, README, releases v6.5.6 (2026-11-06), v6.5.5 (2026-10-12), v5.1.7 (2026-11-08) , retrieved 2026-11-14. [primary]**
https://github.com/Xpra-org/xpra

3. **Run remote applications alongside your local ones, Xpra docs , retrieved 2026-11-14. [primary]**
https://xpra-org.github.io/xpra/Usage/Seamless.html

4. **Xpra encodings documentation , retrieved 2026-11-14. [primary]**
https://xpra-org.github.io/xpra/Usage/Encodings.html

5. **Xpra FAQ , retrieved 2026-11-14. [primary]**
https://xpra-org.github.io/xpra/FAQ.html

6. **RFB / VNC, Xpra docs , retrieved 2026-11-14 (via search result). [primary]**
https://xpra-org.github.io/xpra/Network/RFB.html

7. **Xpra manual, `--resize-display` and `--encoding` options , retrieved 2026-11-14 (via search result). [primary]**
https://xpra.org/manual

8. **Xpra dummy X11 config template, `Virtual 8192 4096` , retrieved 2026-11-14 (via search result). [primary]**
https://xpra.org/xorg.conf

9. **Xpra DPI documentation , retrieved 2026-11-14 (via search result). [primary]**
https://xpra-org.github.io/xpra/Features/DPI.html

10. **make it easier to set initial resolution using --resize-display=geometry, opened 2020-05-20, closed 2021-02-26 , retrieved 2026-11-14 (via search result). [primary]**
https://github.com/Xpra-org/xpra/issues/2772

11. **wayland server backend, maintainer comment 2025-10-20 , retrieved 2026-11-14 (via search result). [primary]**
https://github.com/Xpra-org/xpra/issues/387

12. **TurboVNC docs on seamless windows and CPU overhead [primary]**
https://github.com/TurboVNC/turbovnc/blob/master/doc/vglusage.txt

13. **Support for xpra, opened 2012-02-23, closed 2014-09-23, retrieved 2026-11-14 (via search result).**
https://github.com/novnc/noVNC/issues/132