ChatGPT

Plattformübergreifende UI-Streaming-Tools

Für das grafische Streaming der Cursor-IDE (eine VSCode-basierte Desktop-App) kommen verschiedene Ansätze in Frage, die Web-basiertes Anzeigen des IDE-Fensters ermöglichen, während Eingaben über Automatisierung (CDP/Playwright) erfolgen. Auffällig sind vor allem Lösungen, die nicht auf klassischem VNC basieren – stattdessen verwenden sie moderne Techniken wie WebRTC oder das RDP-Protokoll.

- **Xpra (Xpra.org)**: Ein „Screen für X11“, der Anwendung-Fenster oder ganze Desktops über das Netzwerk vorhält [1]. Xpra unterstützt Linux (X11-Server) und kann laut Projektseite auch Bildschirme von Windows- oder macOS-Systemen weiterreichen [1]. Es bietet einen HTML5-Client mit hardwarebeschleunigter Wiedergabe [2]. Xpra lässt sich headless betreiben (z.B. mit Xvfb oder einem „headless“ Backend) und per CLI steuern. **Vorteile:** gutes Bandbreiten-Management, Wiederverbinden, Multi-User, Clipboard/Sound-Weiterleitung. **Nachteile:** Performance kann bei 3D- oder Videografik limitiert sein (Scraping über X11 kann Performance-Einbußen bringen) [3]. Wayland-Unterstützung ist nicht nativ, üblicherweise wird hier XWayland eingesetzt.

- **Sunshine/Moonlight (LizardByte Sunshine)**: Ein selbst-gehosteter Game-Stream-Server (Moonlight-kompatibel) mit sehr niedrigem Input-Latenz [4]. Sunshine läuft auf Linux, Windows und macOS [5] und nutzt Hardware-Encoder (Nvidia NVENC, Intel QuickSync, AMD VCE) für H.264/H.265 Streaming [5]. Es kann den X11- oder Wayland-Desktop (über KMS) erfassen. **Vorteile:** extrem leistungsfähig bei Video/3D (GPU-beschleunigt), niedrige Latenz, Streaming-Codec (H.264/5) sehr effizient. **Nachteile:** erfordert moderne GPU/CPU (s. Systemanforderungen [5]), primär für Spiele gedacht, nutzt Moonlight-Clients (kein nativer HTML5-Client). (Theoretisch könnte man einen WebRTC-Client bauen, aber von Haus aus richtet sich Sunshine auf native Clients.)

- **WebRTC-basierte Streamer (z.B. *webrtc-streamer*, *Neko*)**: Diese Open-Source-Projekte fangen den Desktop oder ein einzelnes Fenster ab und streamen es als WebRTC-Videostream. *webrtc-streamer* (C++-basiert) kann etwa über URLs wie `screen://` oder `window://` den gesamten Bildschirm oder ein einzelnes Fenster erfassen und per WebRTC übertragen [6]. Es lässt sich per Docker oder CLI-Server starten [7]. Auch das *Neko*-Projekt ist ein Docker-Container, der einen kompletten Linux-Desktop (z.B. XFCE) via WebRTC streamt [8] [9]. Dabei kann *Neko* mehrere Nutzer bedienen und jede App oder ganzen Desktop isoliert ausführen [9]. **Vorteile:** Reines WebRTC-Streaming ohne VNC/RDP, leichtgewichtig in Containern einsetzbar, Headless-Betrieb möglich, direkt im Browser darstellbar. **Nachteile:** Relativ neue Projekte (möglicherweise weniger ausgereift), Eingabe-Übertragung ist primär für kollaborative Nutzung gedacht (man könnte sie abschalten und nur ein „View-Only“-Streaming nutzen). Stand jetzt sendet *webrtc-streamer* nur Videodaten (kein Maus/Keyboard zurück), was aber gewünscht ist.

- **RDP-basierte Lösungen**: Windows bietet von Haus aus einen RDP-Server (Terminal Services) für Desktop-Streaming. Unter Linux kann **xrdp** installiert werden – ein Open-Source RDP-Server, der Linux-Desktops per RDP freigibt [10]. Für X11 benötigt xrdp oft ein Xorg-Modul (xorgxrdp), um Sitzungen zu verwalten [10]. GNOME-Wayland-Desktops bringen ebenfalls eingebaute Fernsteuerung via RDP mit (bei aktiviertem Remote-Desktop über RDP) [11]. **Vorteile:** RDP ist sehr

performant und weit verbreitet; Clients für Windows/Linux/iOS existieren. **Nachteile:** Webfrontends (z.B. Apache Guacamole) sind nötig, um RDP im Browser anzuzeigen; Einrichtung kann komplex sein. Auf Windows-Seite läuft RDP-Server meistens als Dienst (auch headless möglich). Unter Wayland benötigt man meist eine aktive Desktop-Sitzung (GNOME/Wayland-Remote-Desktop), was Container-Betrieb schwieriger macht.

Nachfolgende Tabelle fasst die wichtigsten Lösungen zusammen:

Lösung	Plattform(en)	Headless-Start möglich	WebRTC	Vor- / Nachteile
Xpra [1]	Linux (X11), *Windows*	Ja (CLI)	Nein	+ Portabel, vollwertiges „Detach“ (persistente Sessions); • Eingabe via Clients (aber hier kann Autom. übernehmen); – Performance-Drossel bei glLastools.
Sunshine (Moonlight) [4] [5]	Linux (X11/ Wayland), Windows, macOS	Ja (Service)	Ja (eigener Codec über GPU)	+ Sehr geringe Latenz, H.264/5-Streaming (GPU-encode) [5] ; – Nur GPU-beschleunigt, primär Game-Streaming-Protokoll (Moonlight-Client, kein Browser-Client).
webrtc-streamer [6]	Linux, Windows, macOS	Ja (CLI/ Docker)	Ja	+ Reines WebRTC-Streaming per `screen://` oder `window://` [6] ; leichtgewichtig, Docker-Image verfügbar; – Noch experimentell, konzentriert sich nur aufs Streaming (Eingabe fehlt, aber wird hier über Automatisierung gelöst).
Neko (WebRTC-Container) [8] [9]	Linux (Docker)	Ja (Docker)	Ja	+ Komplett-Desktop in Container, Multi-User-WebRTC [9] ; + Beliebige Linux-App lauffähig (z.B. IDE) [9] ; – Projektstatus experimentell, weniger dokumentiert.
xRDP / RDP [10] [11]	Linux (X11) via xrdp; Windows (builtin); GNOME/ Wayland (RDP)	Ja (Service)	Nein	+ Eingebaut (Windows) bzw. ausgereift (xrdp auf Linux) [10] , sehr performant; – Kein nativer HTML5-Client (muss über Guacamole o.ä. ins Web gebracht werden); GNOME-Wayland-Remote erfordert User-Session.

Lösung	Plattform(en)	Headless-Start möglich	WebRTC	Vor- / Nachteile
Container-Lösungen (z.B. Kasm)	Linux (Docker)	Ja (Docker)	Ja (WebRTC)	+ Komplett-Plattform für Browser-Streaming, inkl. Webinterface; – Sehr umfangreich, kommerziell, meist überdimensioniert.
Apache Guacamole	Linux-Server, Windows RDP/VNC	Ja (Docker/etc.)	Nein	+ Universaler HTML5-Remotezugriff (RDP/VNC); – Set-up sehr komplex, nutzt intern RDP/VNC (also indirekt) und kann VNC-Performance-Probleme haben.

Fazit: Für leichtgewichtiges, plattformübergreifendes Streaming empfiehlt sich eine WebRTC-Lösung wie *webrtc-streamer* oder ein Xpra-Server. Xpra ist sehr stabil für X11-Anwendungen, *webrtc-streamer* und ähnliche Tools ermöglichen direktes Browser-Streaming ohne VNC. Sunshine liefert maximale Performance über GPU-Capture, erfordert aber spezielle Clients. RDP (xrdp/GNOME) funktioniert ebenfalls, bindet jedoch meist an ein Benutzer-Login und benötigt gegebenenfalls einen Web-Proxy (z.B. Guacamole). Die Automatisierungs-Eingabe per CDP/Playwright würde unabhängig vom Stream erfolgen – etwa indem man im Container VSCode/Cursor steuert, während nur das Bild in den Browser gestreamt wird.

Quellen: Anbieter- und Projektdokumentationen sowie Community-Berichte zu Xpra [1], Sunshine [4] [5], WebRTC-Streamer [6], Neko [8] [9], xRDP [10] und GNOME-Remote-Desktop [11].

---

[1] [2] xpra home page
https://xpra.org/index.html

[3] X.Org Server & XWayland Hit By Four More Security Issues - Phoronix Forums
https://www.phoronix.com/forums/forum/linux-graphics-x-org-drivers/x-org-drm/1454853-x-org-server-xwayland-hit-by-four-more-security-issues/page13

[4] [5] GitHub - LizardByte/Sunshine: Self-hosted game stream host for Moonlight.
https://github.com/LizardByte/Sunshine

[6] [7] GitHub - mpromonet/webrtc-streamer: WebRTC streamer for V4L2 capture devices, RTSP sources and Screen Capture
https://github.com/mpromonet/webrtc-streamer

[8] [9] GitHub - m1k1o/neko: A self hosted virtual browser that runs in docker and uses WebRTC.
https://github.com/m1k1o/neko

[10] GitHub - neutrinolabs/xrdp: xrdp: an open source RDP server
https://github.com/neutrinolabs/xrdp

[11] How do I setup a remote access solution in Wayland? - #4 by vgaetera - Fedora Discussion
https://discussion.fedoraproject.org/t/how-do-i-setup-a-remote-access-solution-in-wayland/121232/4