Contents

File transfer

How files move on and off these machines without the cloud: **LocalSend** for phone ↔ computer, and **rsync over SSH** for computer ↔ computer. Everything is local-network only, opened *temporarily* on untrusted networks (and auto-closed), and installs from a single recipe.

Install

```
just file-transfer          # tools + firewall + ~/Downloads/Transfers + docs
just file-transfer-harden   # receiving machines only: inbound SSH key-only +
LLMNR off
```

`file-transfer` is idempotent and machine-agnostic — it opens `53317` on whatever firewall zone holds the active interfaces, and renders a per-host `~/Downloads/Transfers/README.md`. The helper scripts and the niri menu entries arrive with `just stow`.

!!! warning "Why `file-transfer-harden` is a separate step" It disables SSH **password** login (key-only). Run it only on machines that will *receive* rsync, and make sure you have a key installed — or are sitting at the machine — first. `sshd` is left disabled; `ssh-here` starts it on demand.

LocalSend — phone ↔ computer

LocalSend discovers devices over UDP `53317` and transfers over TCP `53317`.

- **Home network:** permanently allowed on the `home` zone — your phone just finds the machine.
- **Untrusted network:** open it for the session from the niri power menu (**Mod+Escape**):

Action	Menu entry	Command
Open + launch	📤 LocalSend share (temp)	`localsend-here on`
Close now	🔒 LocalSend close	`localsend-here off`

The opening is runtime-only with a **30-minute timeout**, so it reverts on its own (and a reboot wipes it).

rsync over SSH — computer ↔ computer

Direction decides what you need:

- **This machine → remote** (you push out): nothing to set up — `rsync -av ~/stuff/ user@remote:/dest/`.
- **Remote → this machine** (something connects in): inbound SSH is key-only and every transfer key is locked to `~/Downloads/Transfers`.

Authorize a sender's key (one-time)

```
add-rsync-key 'ssh-ed25519 AAAA… you@laptop'
```

Option	Effect
(default)	upload-only (`-wo -no-del`) into `~/Downloads/Transfers`
`--ro`	download-only
`--dir NAME`	lock to `~/NAME` instead

The key is installed with a **forced command**:

```
restrict,command="/usr/bin/rrsync -wo -no-del Downloads/Transfers" ssh-ed25519 …
```

`restrict` strips PTY and forwarding; `rrsync` (see `man rrsync`) refuses anything but rsync and jails it to that folder. So the key can do nothing but drop files there — no shell, no other paths — even if it leaks.

Receive a transfer

1. Open: menu 🔑 **SSH/rsync in (temp)** or `ssh-here on` — starts `sshd`, opens `22` for 30 minutes.
2. Sender: `rsync -av ./files/ <host>:` (the path is relative to the inbox folder).
3. Close: menu 🔒 **SSH/rsync close** or `ssh-here off`.

How the temporary opening works

`localsend-here` and `ssh-here` share one pattern:

1. Find the zone of the default-route interface — the network you're *actually* on.
2. Add the port to that zone **at runtime only**, with `--timeout=30m`.
3. Never `--permanent`, so it's gone on reboot/reload and can't silently persist.

That's why a conference-Wi-Fi opening is safe: scoped to the live network, and self-closing.

Security model

- Inbound SSH is **public-key only**, no root — `/etc/ssh/sshd_config.d/10-hardening.conf`.
- `sshd` is **disabled**; started on demand by `ssh-here`, stopped on `off`.
- Transfer keys are rsync-only, upload-only, no-delete, single-folder.
- LLMNR is disabled (`file-transfer-harden`) — SMB discovery (mDNS + NetBIOS) is unaffected.
- Every temporary firewall hole is runtime-only with a 30-minute expiry.

Where it lives

Thing	Path
Helper scripts	`~/.local/bin/{localsend-here, ssh-here, add-rsync-key}`
System setup	`~/.local/bin/{setup-file-transfer, harden-file-transfer-ssh}`
Menu entries	niri power menu (`menu-power`)
Doc templates	`~/.local/share/file-transfer/*.md` → rendered into `~/Downloads/Transfers/`
Packages	`file-transfer` category in `setup/packages.yaml`
Recipes	`just file-transfer`, `just file-transfer-harden`