Boot from network attached devices using mkosi-initrd

(or why systemd distributions should really start considering mkosi-initrd)

Antonio Álvarez Feijoo

FOSDEM 2025

Contents

About me

- Antonio Álvarez Feijoo
- System Boot and Init @ SUSE
- Working on:
  - initrd related topics (mostly dracut)
  - systemd
- Contact: antonio.feijoo@suse.com

initrd

- Minimal hardware init

- Load kernel and initrd image(s) into memory

- Mount initrd as tmpfs
- exec initrd's /sbin/init

- Mount rootfs (local device, network, ...)
- switch_root
- exec rootfs'/sbin/init

- Init all hardware, filesystems, services, drivers, ...

initrd

initrd generators

- Traditional initrd generators:
  - dracut: openSUSE, Fedora, ...
  - initramfs-tools: Debian, Ubuntu
  - mkinitcpio: Arch
- Bash scripts.
- Discretionally pick files from the host system.
- Add custom logic to the boot process: hooks, breakpoints...

mkosi-initrd

How does it work

- Wrapper on top of mkosi (image builder).
- It creates CPIO (compressed) archives using:
  - Distribution packages.
  - Package managers (apt, dnf, pacman, zypper) and their dependency resolution mechanism.
- It does not pick files from the host system to build the initrd (*ExtraTrees=).
  - (*) By default, it includes the **loaded** kernel modules from the system.
  - (**) It can include other files if configured to do so.
- It does not add custom logic to the boot process (*).
  - (*) Right now includes 2 udev rules (for compatibility, already upstreamed to lvm2/mdadm).

mkosi-initrd

Pros

- It builds initrds out of distribution packages.
  - The source is known beforehand → reproducibility.
  - Less complexity: choose only packages and kernel modules.
  - Distributions can provide predefined configuration sets to ship specific functionality.
- Clear ownership of bugs.
- Any change on a package would automatically apply to the initrd.
- **It shifts the maintenance effort from the initrd generator to the responsible package.**

mkosi-initrd

Cons

- Fragile to packaging errors.
  - But on the other hand, a build error should not reach end users (early detection in openQA).
- initrd generation can be slow.
  - It improves if package manager caches are enabled on the host.
- Larger initrd size.
  - Careful packaging is required… or, RemoveFiles= via conf meanwhile.
  - openSUSE Tumbleweed: basic: ~25 MB → ~50MB, w/NetworkManager: ~40 MB → ~70 MB.
- Limited arch support: x86_64 (UEFI/BIOS), aarch64
  - RFE open for s390x… help wanted!
- mkosi-initrd wrapper only for “hostonly” initrds.
  - “non hostonly” available with mkosi: mkosi --format=cpio --include=mkosi-initrd …

From dracut to mkosi-initrd

How to implement the missing functionality (without adding custom logic)

- Analyze how complex setups currently work: the reference is dracut.
- Cleanup & focus: **dracut-core** (https://github.com/aafeijoo-suse/dracut-core)
  - Removed non-systemd code, other unnecessary code, deprecated code and options...
  - Removed the messy --sysroot option (and the $dracutsysrootdir var present everywhere).
  - Removed the ability to create UEFI executables (ukify is better).
  - ...

```
$ find dracut-ng/modules.d/95iscsi/ -type f -print0 | sort -z | xargs -0 wc -l
    5 modules.d/95iscsi/cleanup-iscsi.sh
  322 modules.d/95iscsi/iscsiroot.sh
  286 modules.d/95iscsi/module-setup.sh
   15 modules.d/95iscsi/mount-lun.sh
  160 modules.d/95iscsi/parse-iscsiroot.sh
  788 total
```

```
$ find dracut-core/modules.d/95iscsi/ -type f -print0 | sort -z | xargs -0 wc -l
  297 modules.d/95iscsi/iscsiroot.sh
  279 modules.d/95iscsi/module-setup.sh
  108 modules.d/95iscsi/parse-iscsiroot.sh
  684 total
```

From dracut to mkosi-initrd

Translate dracut modules into mkosi configuration

- mkosi-initrd configuration under /{etc,usr/lib}/mkosi-initrd/...
- .../mkosi.conf.d/<name>.conf
  - INI configuration files (drop-ins allowed, systemd style).

```
[Content]
Packages=...
KernelModulesInclude=...
ExtraTrees=...
```

- .../mkosi.extra/
  - Custom filesystem tree.
  - Everything here that is not configuration should be upstreamed.

From dracut to mkosi-initrd

Translate dracut modules into mkosi configuration

1. Identify installed binaries, units, libs...
   - Add packages that provide them via Packages= in conf.
2. Identify installed kernel modules
   - KernelModulesInclude= in conf.
3. Identify hooks (inst_hook <hook> <prio> <script>)
   - Rework <script> to be independent of dracut-lib.
   - Service in mkosi.extra ordered like dracut-<hook>.service with ExecStart=<script>
   - Then, upstream these new services into the responsible package!
4. Identify custom sd-units/udev rules/binaries/...
   - If required, mkosi.extra first... then upstream!

From dracut to mkosi-initrd

Alternatives to custom logic added by dracut

1. Breakpoints.
2. Parse the kernel cmdline.
3. Extend the kernel cmdline.
4. Hooks.
5. The initqueue.
6. Restore on shutdown (exitrd).

From dracut to mkosi-initrd

[1/6] dracut internals: breakpoints

- Kernel cmdline option to pause the boot process at a certain point and drop to a shell.
  - rd.break={cmdline|pre-udev|pre-trigger|initqueue|pre-mount|mount|pre-pivot|cleanup}
  - ConditionKernelCommandLine=rd.break=... in custom dracut-* systemd services.
- Basic for debugging complex setups.
- Alternative:
  - Implement this directly in systemd: https://github.com/systemd/systemd/pull/35410

From dracut to mkosi-initrd

[2/6] dracut internals: parse the kernel cmdline

- dracut provides Bash functions: getarg, getargs, getargbool, getargnum...
- Alternative:
  - Packages can provide their own binary helpers to do so (better than scripts, right?).
  - E.g., NetworkManager: nm-initrd-generator.

From dracut to mkosi-initrd

[3/6] dracut internals: extend the kernel cmdline

- It extends /proc/cmdline with /etc/cmdline and /etc/cmdline.d/*.conf.
- At build or at boot.
- Mostly used for auto-detection.
- Alternative:
  - This will not be supported by systemd: https://github.com/systemd/systemd/issues/22935
  - cmdline options do not change often and bootloaders allow to define them statically (e.g. GRUB_CMDLINE_LINUX_DEFAULT).

From dracut to mkosi-initrd

[4/6] dracut internals: hooks

- It allows to inject and run external scripts at certain points in the boot process.
  - cmdline, pre-udev, pre-trigger, initqueue, pre-mount, mount, pre-pivot, cleanup.
- Can be injected at build or at boot.
- How does dracut source these scripts during the boot process: using its custom dracut-* systemd services.
- Alternative:
  - Instead of asking dracut to run a script spawned by its dracut-* services, packages can provide their own systemd services properly ordered.
  - man bootup(7)

From dracut to mkosi-initrd

[5/6] dracut internals: the initqueue

- Wait until all necessary hardware has been discovered.
  - Hardware detection is asynchronous.
- Blocking loop after sysinit.target
  - Hooks: initqueue, initqueue/settled, initqueue/finished, initqueue/timeout, initqueue/online.
  - It loops until udev has settled and all scripts in initqueue/finished return true.
- Alternatives:
  - Order after systemd-udev-settle.service → wrong!
  - Services can subscribe to udev events and react to any new hardware as it is discovered, e.g., ListenNetlink=... (AF_NETLINK)
  - E.g.: systemd-networkd-wait-online@.service → wait for specific network iface
  - Maybe we need to implement some like this in the future?

From dracut to mkosi-initrd

[6/6] dracut internals: restore on shutdown (exitrd)

- exitrd: unpack and load initrd on shutdown to perform custom cleanups.
- 2 special hooks: pre-shutdown, shutdown.
- Alternative:
  - Before shutdown, systemd-shutdown runs all executables in /usr/lib/systemd/system-shutdown/*
  - E.g., mdadm: /usr/lib/systemd/system-shutdown/mdadm.shutdown

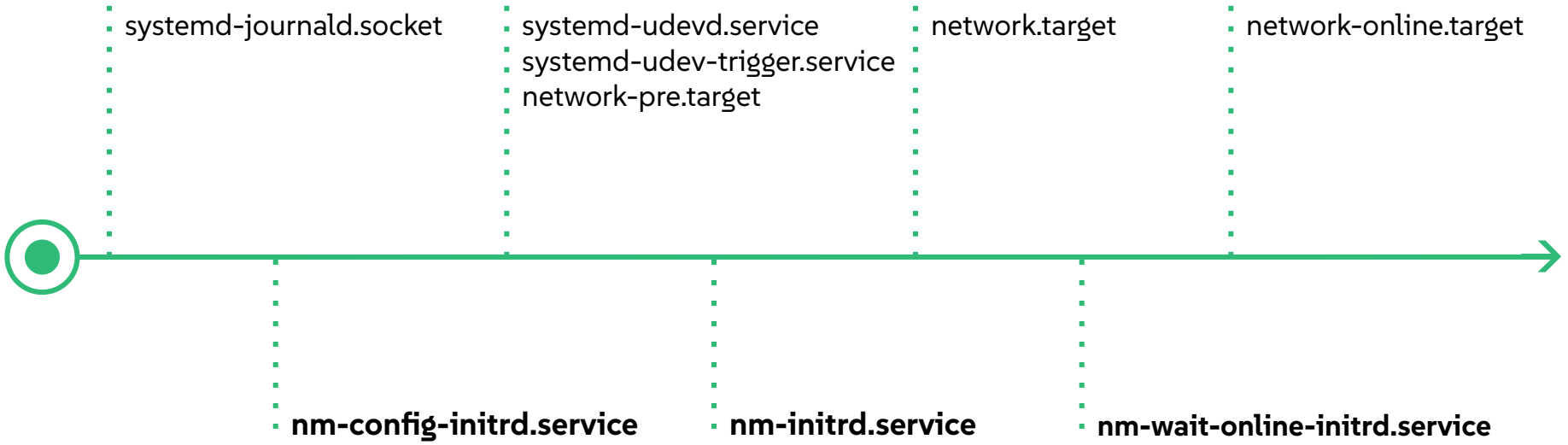
Set up the network using NetworkManager

dracut → modules.d/35network-manager

- Packages: NetworkManager.
- Kernel modules: mkosi-initrd adds currently loaded network modules.
  - Need more? → KernelModulesInclude=net/... in conf.
- Custom conf (/usr/lib/NetworkManager/conf.d): `no-auto-default=*`
- 2 custom systemd services: **nm-initrd.service** + **nm-wait-online-initrd.service**
- inst_hook cmdline nm-config.sh → **nm-config-initrd.service**
  - nm-initrd-generator binary already provided by NM.
  - Wouldn't a systemd generator be better instead of a regular service?
- Reduce scope: avoid handling dhclient, use NM's internal DHCP client.

Set up the network using NetworkManager

How NM systemd services are ordered at boot



Set up the network using NetworkManager

mkosi-initrd + NetworkManager

```
/etc/mkosi-initrd/
├── mkosi.conf.d
│   └── 20-network.conf
└── mkosi.extra
    └── usr
        └── lib
            ├── NetworkManager
            │   └── conf.d
            │       └── nm-initrd.conf
            └── systemd
                ├── system
                │   ├── nm-config-initrd.service
                │   ├── nm-initrd.service
                │   └── nm-wait-online-initrd.service
                └── system-preset
                    └── 20-network.preset
```

Set up the network using NetworkManager

mkosi-initrd + NetworkManager

```
/etc/mkosi-initrd/
    ├── mkosi.conf.d
    │   └── 20-network.conf
    └── mkosi.extra
        └── usr
            └── lib
                ├── NetworkManager
                │   └── conf.d
                │       └── nm-initrd.conf
                └── systemd
                    ├── system
                    │   ├── nm-config-initrd.service
                    │   ├── nm-initrd.service
                    │   └── nm-wait-online-initrd.service
                    └── system-preset
                        └── 20-network.preset
```

```
[Content]
Packages=
        NetworkManager

        # For libnss_dns
        glibc

        # For libnss_mdns4
        nss-mdns

        # Nice to have for debugging
        iproute2
        iputils

# Add modules not loaded when the initrd is built
# KernelModulesInclude=
#       net/…
```

Set up the network using NetworkManager

mkosi-initrd + NetworkManager

```
/etc/mkosi-initrd/
   ├── mkosi.conf.d
   │   └── 20-network.conf
   └── mkosi.extra
       └── usr
           └── lib
               ├── NetworkManager
               │   └── conf.d
               │       └── nm-initrd.conf
               └── systemd
                   ├── system
                   │   ├── nm-config-initrd.service
                   │   ├── nm-initrd.service
                   │   └── nm-wait-online-initrd.service
                   └── system-preset
                       └── 20-network.preset
```

→ **upstream to NetworkManager [1]**

[1] https://gitlab.freedesktop.org/NetworkManager/NetworkManager/-/merge_requests/2089

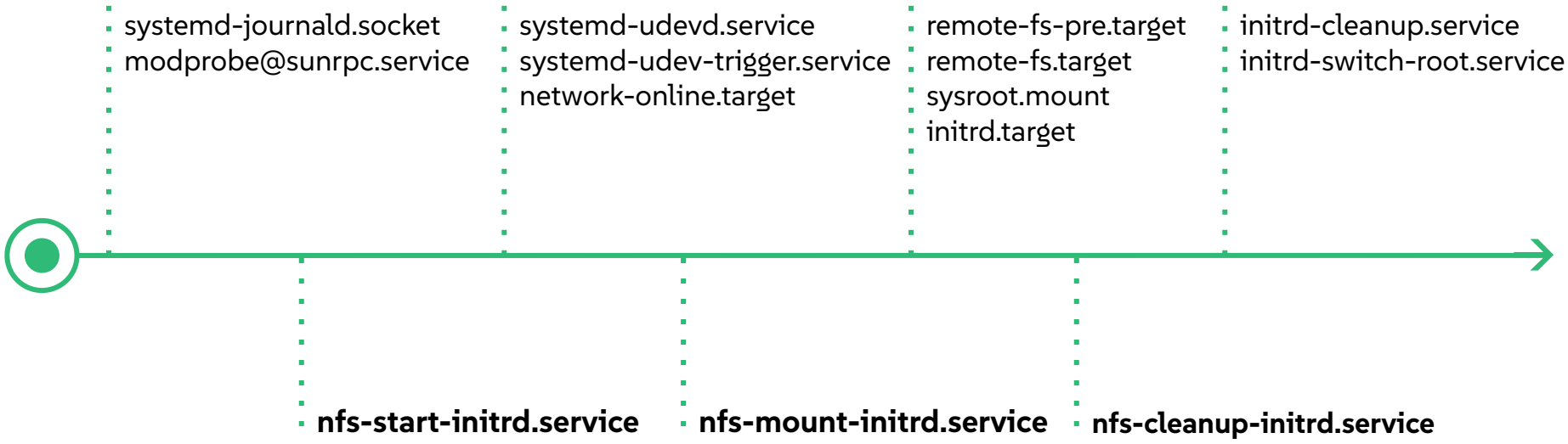
Root filesystem on NFS

dracut → modules.d/95nfs

- Packages: nfs-client, libnfsidmap1, rpcbind...
- Kernel modules: fs/nfs/, nfs_acl.ko, net/sunrpc/
- inst_hook cmdline parse-nfsroot.sh + pre-udev nfs-start-rpc.sh → **nfs-start-initrd.service**
  - Parse kernel cmdline, e.g.: root=nfs:<SERVER_IP>:<SERVER_DIR>,rw ip=dhcp
  - Start RPC.
- nfsroot.sh ($netroot) → **nfs-mount-initrd.service**
  - mount -t "$nfs" -o"$options" "$server:$path" /sysroot
- inst_hook cleanup nfsroot-cleanup.sh → **nfs-cleanup-initrd.service**

Root filesystem on NFS

How new NFS systemd services are ordered at boot



Root filesystem on NFS

mkosi-initrd + NetworkManager + NFS

```
/etc/mkosi-initrd/
├── mkosi.conf.d
│   ├── 20-network.conf
│   └── 30-nfs.conf
└── mkosi.extra
    └── usr
        ├── lib
        │   └── systemd
        │       ├── system
        │       │   ├── nfs-cleanup-initrd.service
        │       │   ├── nfs-mount-initrd.service
        │       │   ├── nfs-start-initrd.service
        │       │   ├── nm-config-initrd.service
        │       │   ├── nm-initrd.service
        │       │   └── nm-wait-online-initrd.service
        │       └── system-preset
        │           ├── 20-network.preset
        │           └── 30-nfs.preset
        └── sbin
            ├── nfs-cleanup.sh
            ├── nfs-mount.sh
            └── nfs-start.sh
```

Root filesystem on NFS

mkosi-initrd + NetworkManager + NFS

```
/etc/mkosi-initrd/
├── mkosi.conf.d
│   ├── 20-network.conf
│   └── 30-nfs.conf
└── mkosi.extra
    └── usr
        ├── lib
        │   └── systemd
        │       ├── system
        │       │   ├── nfs-cleanup-initrd.service
        │       │   ├── nfs-mount-initrd.service
        │       │   ├── nfs-start-initrd.service
        │       │   ├── nm-config-initrd.service
        │       │   ├── nm-initrd.service
        │       │   └── nm-wait-online-initrd.service
        │       └── system-preset
        │           ├── 20-network.preset
        │           └── 30-nfs.preset
        └── sbin
            ├── nfs-cleanup.sh
            ├── nfs-mount.sh
            └── nfs-start.sh
```

```
[Content]
Packages=
        nfs-client
        libnfsidmap1
        rpcbind

        # netconfig
        libtirpc-netconfig
        netcfg

        # libnss_*
        glibc
        libnss_usrfiles2
        nss-mdns

        [...]

KernelModulesInclude=
        /nfs_acl.ko
        fs/nfs/
        net/ipv6/
        net/sunrpc/
```

Root filesystem on NFS

mkosi-initrd + NetworkManager + NFS

```
/etc/mkosi-initrd/
├── mkosi.conf.d
│   ├── 20-network.conf
│   └── 30-nfs.conf
└── mkosi.extra
    └── usr
        ├── lib
        │   └── systemd
        │       ├── system
        │       │   ├── nfs-cleanup-initrd.service
        │       │   ├── nfs-mount-initrd.service
        │       │   ├── nfs-start-initrd.service
        │       │   ├── nm-config-initrd.service
        │       │   ├── nm-initrd.service
        │       │   └── nm-wait-online-initrd.service
        │       └── system-preset
        │           ├── 20-network.preset
        │           └── 30-nfs.preset
        └── sbin
            ├── nfs-cleanup.sh
            ├── nfs-mount.sh
            └── nfs-start.sh
```

→ TODO: upstream to nfs-utils

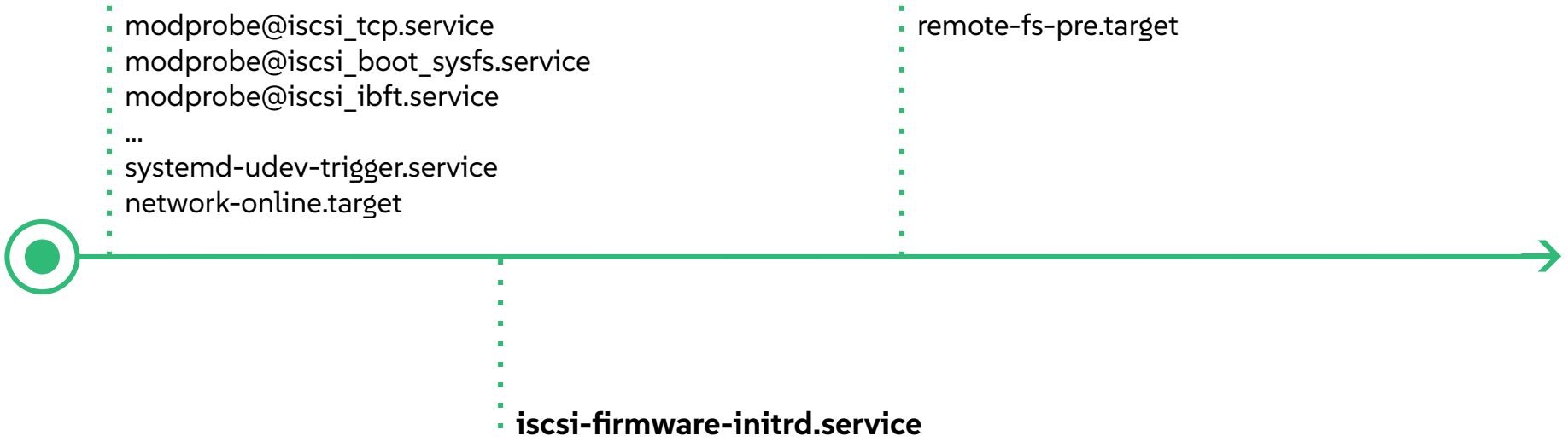
Root filesystem on iSCSI

dracut → modules.d/95iscsi

- Packages: open-iscsi, iscsiuio.
- Kernel modules: drivers/scsi/ tree.
- inst_hook cmdline parse-iscsiroot.sh
- iscsiroot.sh ($netroot)
  - It also parses the kernel cmdline...
- Reduce scope, handle iSCSI params from:
  - firmware (root=? netroot=iscsi rd.iscsi.firmware) → **iscsi-firmware-initrd.service**
  - cmdline (root=? netroot=iscsi:[<servername>]:[<protocol>]:[<port>]:[<LUN>]:<targetname>) → **iscsi-config-initiator-initrd.service** + **iscsi-start-initrd.service**

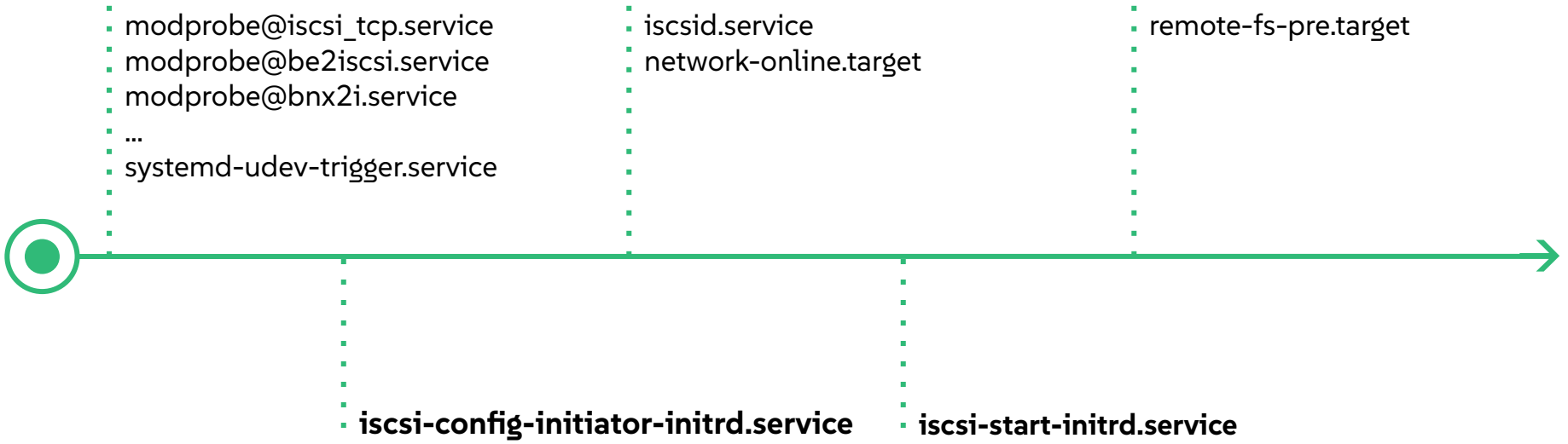
Root filesystem on iSCSI

How new iSCSI systemd services are ordered at boot



Root filesystem on iSCSI

How new iSCSI systemd services are ordered at boot



Root filesystem on iSCSI

mkosi-initrd + NetworkManager + iSCSI

```
/etc/mkosi-initrd/
├── mkosi.conf.d
│   ├── 20-network.conf
│   └── 30-iscsi.conf
└── mkosi.extra
    └── usr
        ├── lib
        │   └── systemd
        │       ├── system
        │       │   ├── iscsi-config-initiator-initrd.service
        │       │   ├── iscsi-firmware-initrd.service
        │       │   ├── iscsi-start-initrd.service
        │       │   ├── nm-config-initrd.service
        │       │   ├── nm-initrd.service
        │       │   └── nm-wait-online-initrd.service
        │       └── system-preset
        │           ├── 20-network.preset
        │           └── 30-iscsi.preset
        └── sbin
            ├── iscsi-config-initiator.sh
            ├── iscsi-firmware.sh
            └── iscsi-start.sh
```

Root filesystem on iSCSI

mkosi-initrd + NetworkManager + iSCSI

```
/etc/mkosi-initrd/
├── mkosi.conf.d
│   ├── 20-network.conf
│   └── 30-iscsi.conf
└── mkosi.extra
    └── usr
        ├── lib
        │   └── systemd
        │       ├── system
        │       │   ├── iscsi-config-initiator-initrd.service
        │       │   ├── iscsi-firmware-initrd.service
        │       │   ├── iscsi-start-initrd.service
        │       │   ├── nm-config-initrd.service
        │       │   ├── nm-initrd.service
        │       │   └── nm-wait-online-initrd.service
        │       └── system-preset
        │           ├── 20-network.preset
        │           └── 30-iscsi.preset
        └── sbin
            ├── iscsi-config-initiator.sh
            ├── iscsi-firmware.sh
            └── iscsi-start.sh
```

```
[Content]
Packages=
        open-iscsi
        iscsiuio

KernelModulesInclude=
        /8021q.ko
        /crc32c.ko
        drivers/scsi/
```

Root filesystem on iSCSI

mkosi-initrd + NetworkManager + iSCSI

```
/etc/mkosi-initrd/
├── mkosi.conf.d
│   ├── 20-network.conf
│   └── 30-iscsi.conf
└── mkosi.extra
    └── usr
        ├── lib
        │   └── systemd
        │       ├── system
        │       │   ├── iscsi-config-initiator-initrd.service
        │       │   ├── iscsi-firmware-initrd.service
        │       │   ├── iscsi-start-initrd.service
        │       │   ├── nm-config-initrd.service
        │       │   ├── nm-initrd.service
        │       │   └── nm-wait-online-initrd.service
        │       └── system-preset
        │           ├── 20-network.preset
        │           └── 30-iscsi.preset
        └── sbin
            ├── iscsi-config-initiator.sh
            ├── iscsi-firmware.sh
            └── iscsi-start.sh
```

→ **TODO: upstream to open-iscsi**

Conclusions

- Not found any blocker yet that can’t be implemented.
- Benefit for classic initrd generators: remove/adapt custom logic to the new upstreamed functionality.
- Benefit for distributions: it leads to improve packaging.
- mkosi-initrd is doing The Right Thing™: being as dumb as possible.
- It's a good alternative for some use cases, but... could it become a full replacement for a traditional initrd generator in the future?

What's next?

- Polish and upstream NFS and iSCSI initrd-specific units.
- More network setups: NVMe-TCP, NBD...
- Multipath
- ...

References

- https://en.opensuse.org/Mkosi-initrd
- https://github.com/systemd/mkosi
- https://archive.fosdem.org/2024/schedule/event/fosdem-2024-2888-mkosi-initrd-building-initrds-out-of-distribution-packages/
- https://cfp.all-systems-go.io/all-systems-go-2024/talk/JTXJR7/

SUSE

Thank you!