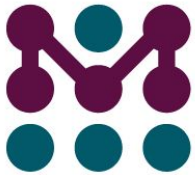
Birds of a Feather Session - OSS Vancouver 2018

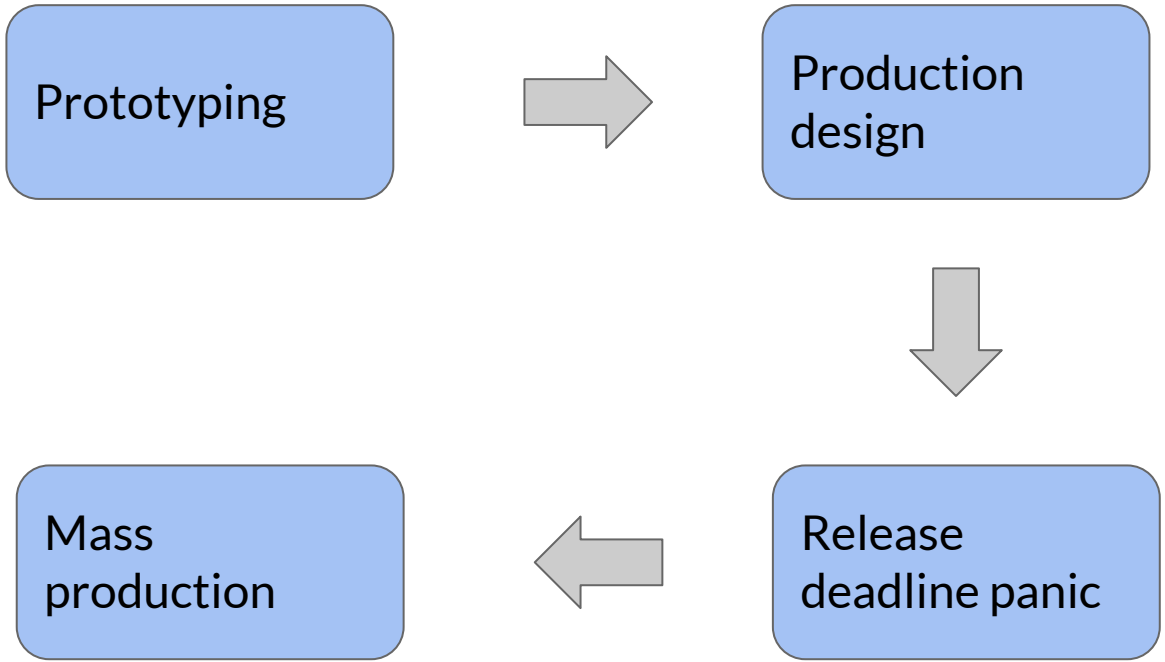


MENDER.io

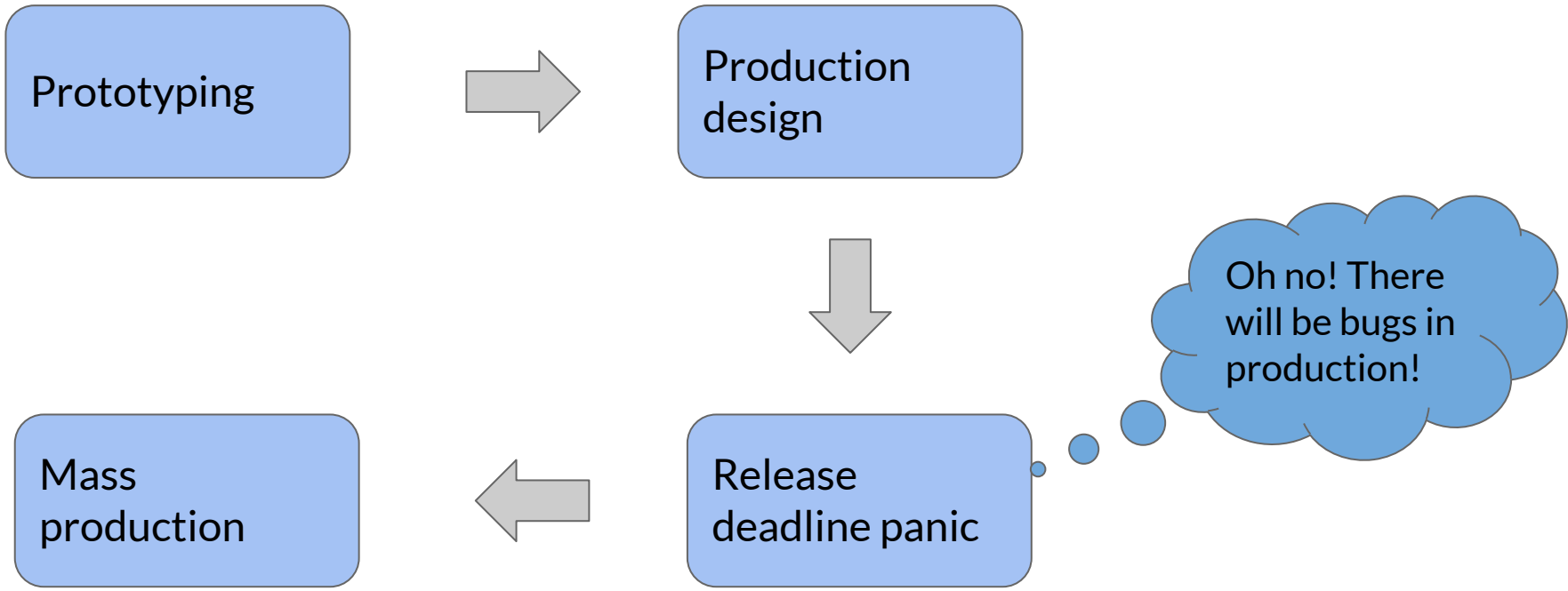
Deploy Software Updates for Linux Devices

Eystein Stenberg, Mender.io

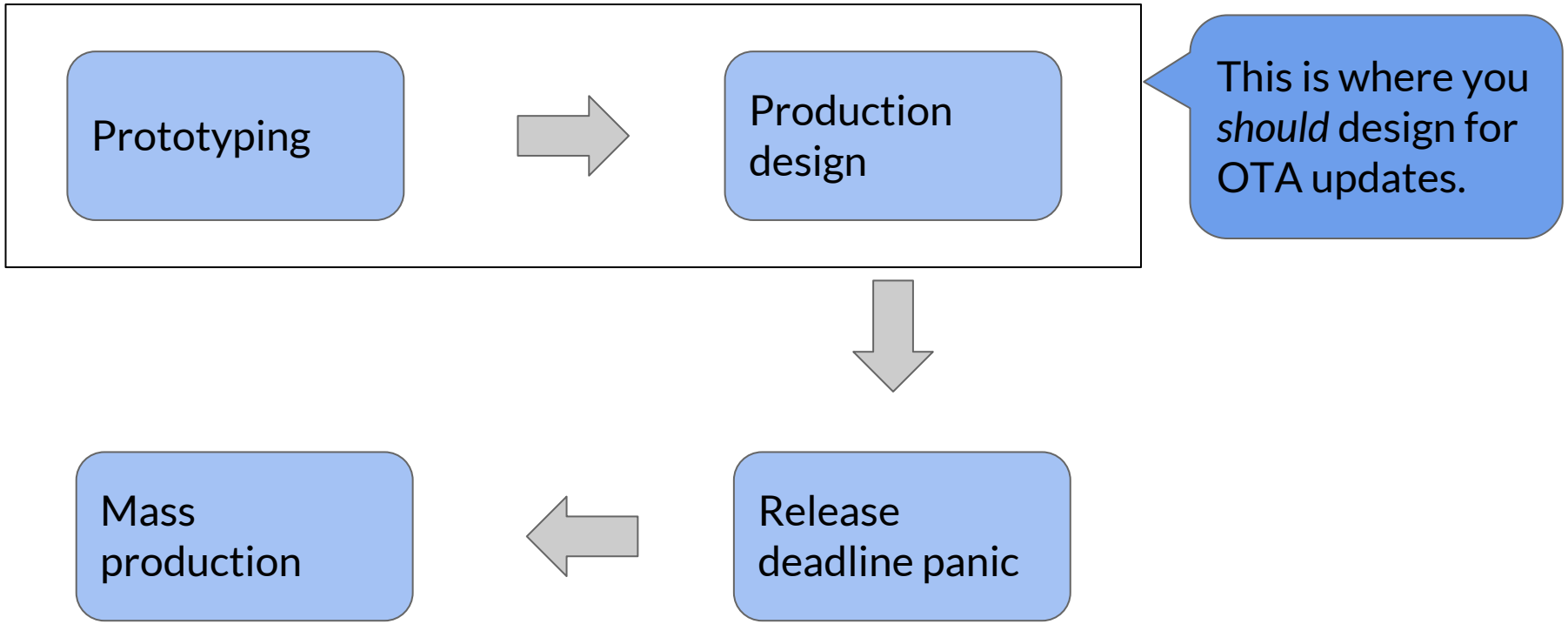
Typical product development process



Updater is too often an afterthought



Updater is too often an afterthought



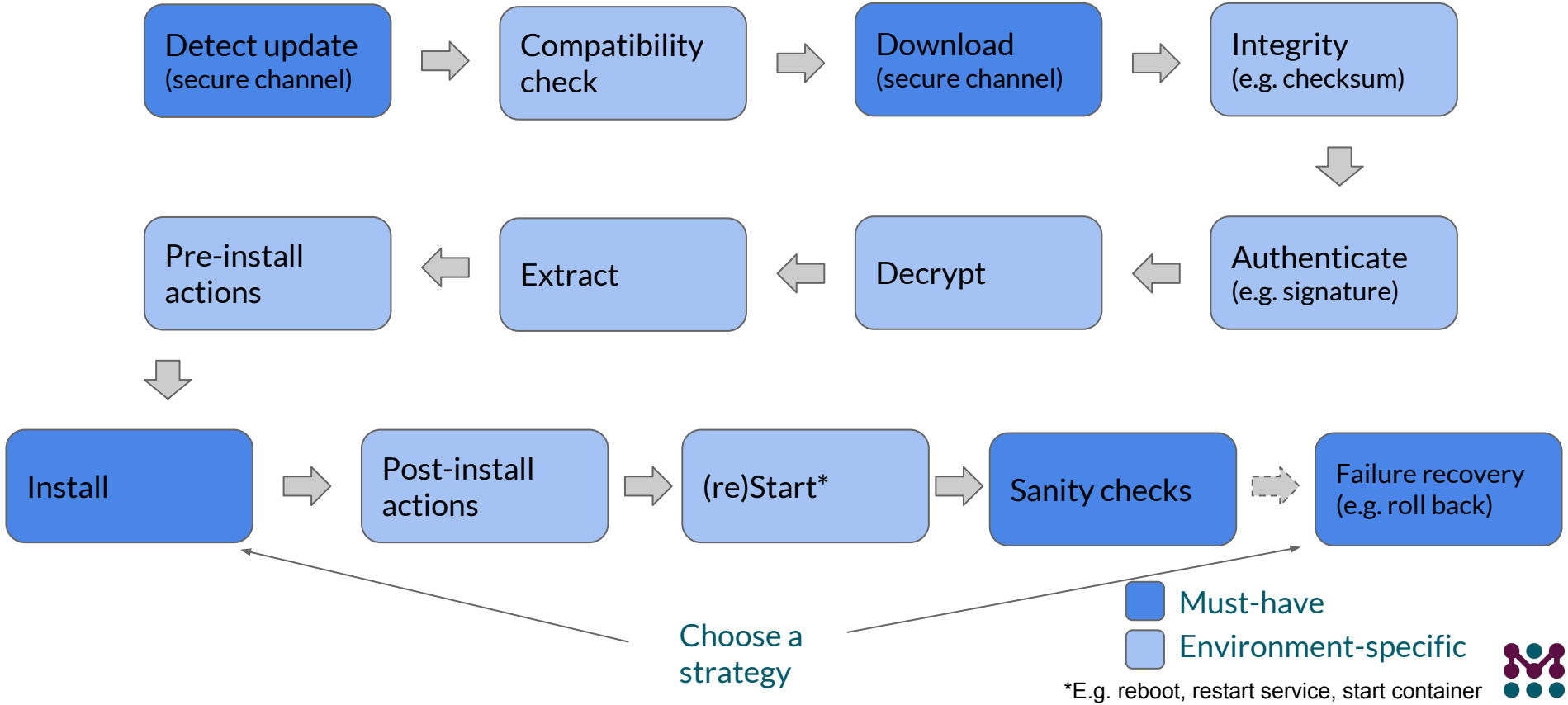
The embedded environment

- Remote
  - Expensive to reach physically
- Long expected lifetime
  - 5 - 10 years
- Unreliable power
  - Battery
  - Suddenly unplugged
- Unreliable network
  - Intermittent connectivity
  - Low bandwidth
  - Insecure

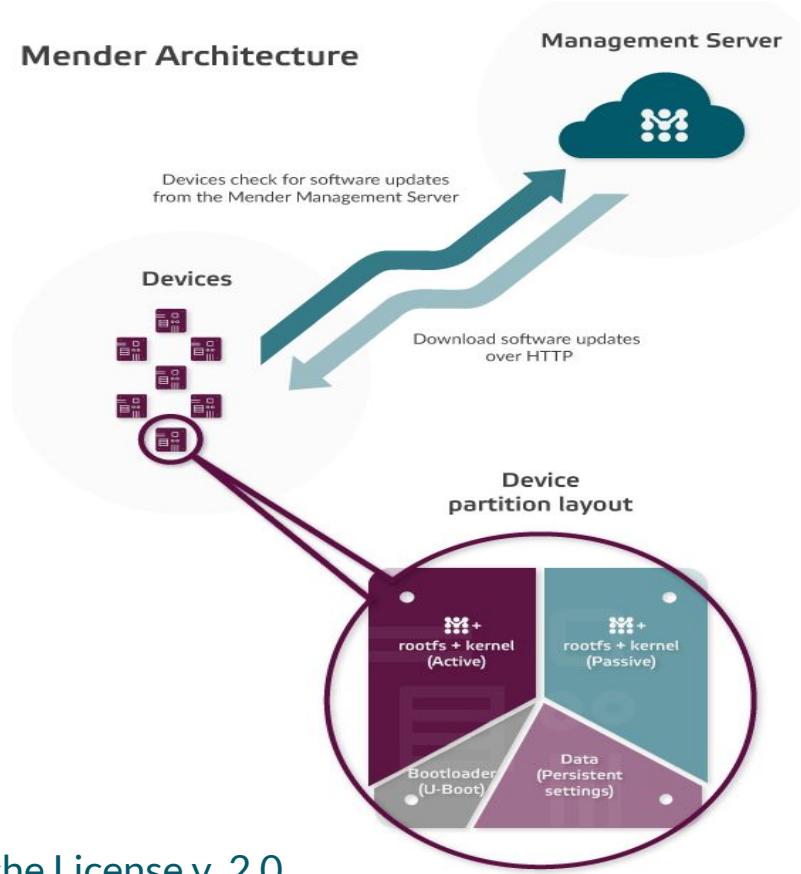
Key criteria for embedded updates

1. Robust and secure
2. Integrates with existing environments
3. Easy to get started
4. Bandwidth consumption
5. Downtime during update
6. Update server enabling mass updates

Generic embedded updater workflow



Mender provides integrated client and update server



- Client-server model
  - Mender provides both
  - Easy integration: No need to “glue” several projects
  - Server can integrate with 3rd party clients through its REST API
- Dual A/B rootfs partition layout
  - Atomic deployments
  - Deploy to inactive partition
  - Robust update process
- Supports updating
  - Kernel, device tree
  - Applications

Mender demo!

Embedded system updaters need board integrations

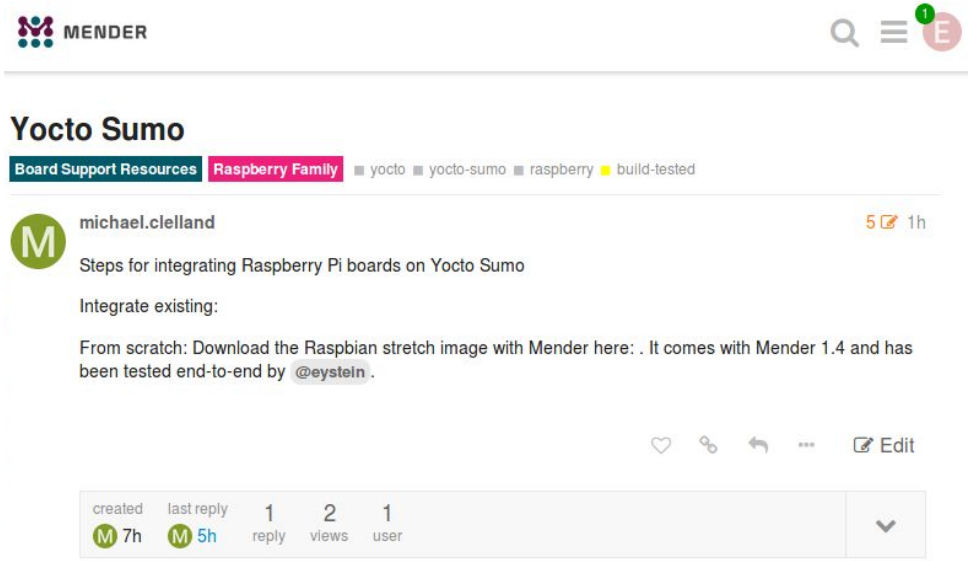
- Atomic system updates (like Mender) need integration to the boot process
  - Write to inactive root fs partition, then flip
- This means interaction with the boot loader & boot process
  - Highly custom for embedded, e.g. proprietary boot code for drivers, many vendor-forks of U-Boot
- Board market highly fragmented
  - No single Single Board Computer / System on Module vendor has more than 5% market share*
  - 80% of product companies manufacture their own boards*
- How we make Mender easily available to everyone in this environment?

* 2017 Embedded Markets Study, Aspencore

Approaches for addressing fragmentation

- Community!
  - I integrated my board, and want to share my code & lessons!
- Product
  - Create run-time abstractions

Community project for device integrations (WIP)



- Make the world's largest repository of OTA-enabled devices
- A page / area for each device (family), e.g. Raspberry Pi family
- Community can edit (wiki-style) and contribute code/scripts

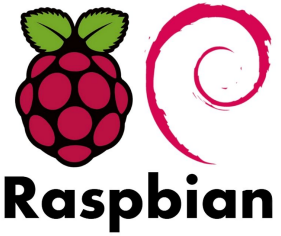
Product: abstractions released for Yocto

- Mender 1.5 (June)
  - UEFI (x86)

- Mender 1.6 (September)
  - BIOS (x86)
  - UEFI emulation (“higher end” ARM w/ U-Boot)
  - Automatic integration patch generation (“lower end” ARM w/ U-Boot)

Product: Support for binary OS integration (WIP)

- mender-conversion-tools
  - https://github.com/mendersoftware/mender-conversion-tools
- Input: Standard OS disk image (.img)
- Output: Image repartitioned & Mender support added
- Enables easy support for binary distributions
  - Debian, Ubuntu, Raspbian
- Last step in Buildroot integration
  - https://patchwork.ozlabs.org/patch/908627

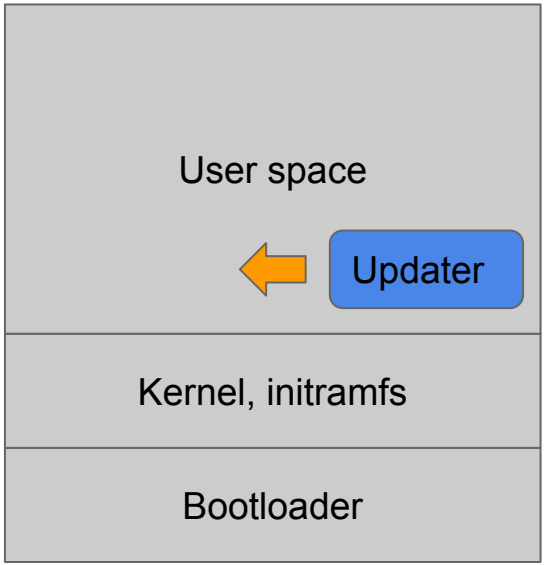


Feedback? What is missing for you?

- Is simplifying device integration worthwhile? How?
- Other product-related items?
- Areas for community & contributions?

Appendix

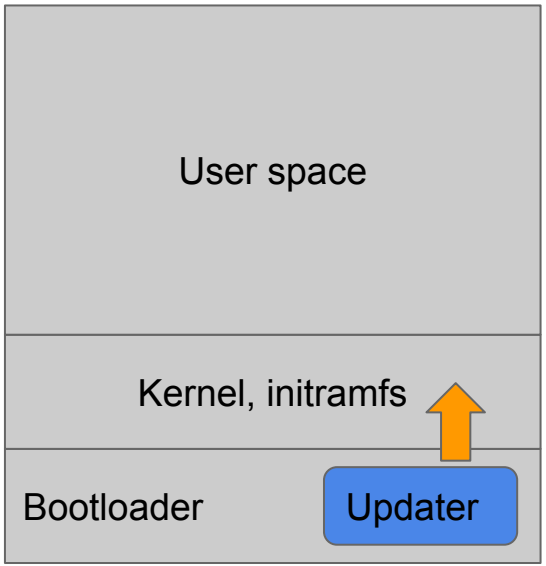
Installer strategy 1: run-time installation



- Updater deploys to running environment
  - Package managers (ipkg, rpm, deb...)
  - OSTree
  - Many homegrown (tar.gz)

- Robustness is hard
  - Atomicity: Hard or impossible
  - Consistency (dev=test): Hard
- Integrates well
  - May already have packages
  - Some userspace tools
- Low bandwidth use (<1mb)
- Short downtime (seconds)

Installer strategy 2: boot to maintenance mode

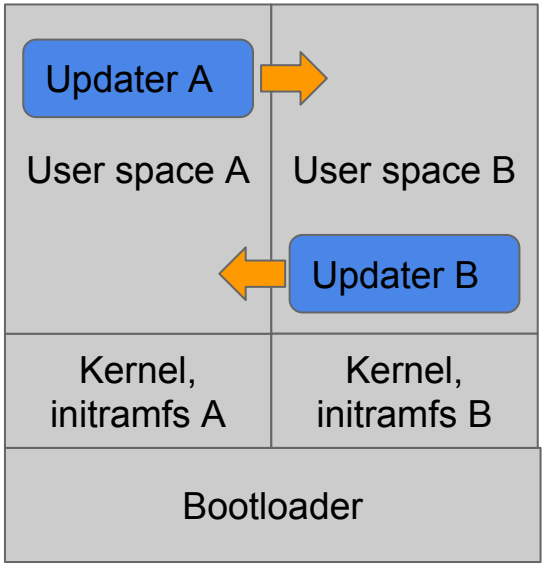


- Updater deploys “up the stack” while running in bootloader
  - Used in older Androids (before ‘N’)
  - “Rescue environment” common in embedded

- Robustness is hard
  - Not atomic (can get partial update)
  - Consistent on success (image)
- Integrates fairly well
  - Bootloader features & intelligence
- High bandwidth use*
  - Whole image
- Long downtime
  - Whole image install
  - 2 reboots

*Can mitigate: compressed/delta

Installer strategy 3: dual A/B rootfs layout

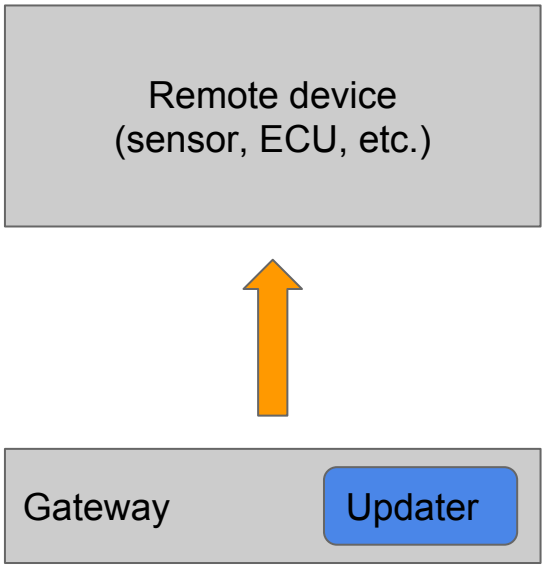


- Updater deploys to inactive partition, then reboots into it
  - Used in newer Androids ('N' and later)
  - Common in "mid/high-end" embedded

- Very robust
  - Fully atomic and consistent
- Integrates fairly well
  - OS, kernel, apps unchanged
  - Needs bootloader "flip" support
  - Partition layout, requires 2x rootfs storage
- High bandwidth use*
  - Whole image
- Fairly short downtime (minute)
  - 1 reboot

*Can mitigate: compressed/delta

Installer strategy 4: proxy



- Updater deploys to remote system
  - Used on smaller devices (sensors, ECUs, etc.), such as in Smart Home or Automotive
  - Requires intelligent gateway to manage

- Slightly different scenario
  - Smaller devices (no client)
  - Complements the others

- Suited for closeby installations only, not internet
  - Robustness (e.g. connection/power loss)
  - Security