Upgrade without Bricking

Arnout Vandecappelle

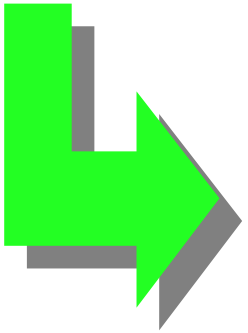
http://mind.be/content/Presentation_Upgrade-without-Bricking.pdf
or .odp

You never know where your product will be used

High-precision GNSS receiver

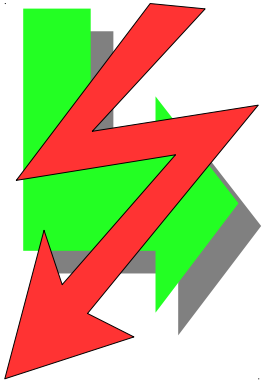
You never know where your product will be used

What if you install new firmware on remote systems?



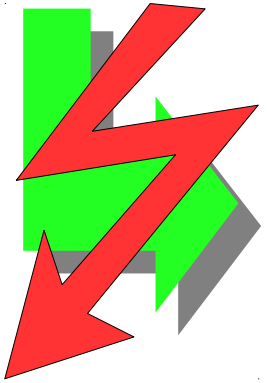
What if you install new firmware on remote systems?

Murphy's Law



What if you install new firmware on remote systems?

Murphy's Law



Upgrade without Bricking

Arnout Vandecappelle

http://mind.be/content/Presentation_Upgrade-without-Bricking.pdf
or .odp

Overview

1. Failure mechanisms
   - Power failure
   - Bad firmware
   - Communication errors
2. Boot loader upgrade
3. Package-based upgrade

Overview

1 Failure mechanisms
- Power failure
- Bad firmware
- Communication errors

2 Boot loader upgrade

3 Package-based upgrade

Power failure

Power fails *during* upgrade

⇒ new firmware only partially written

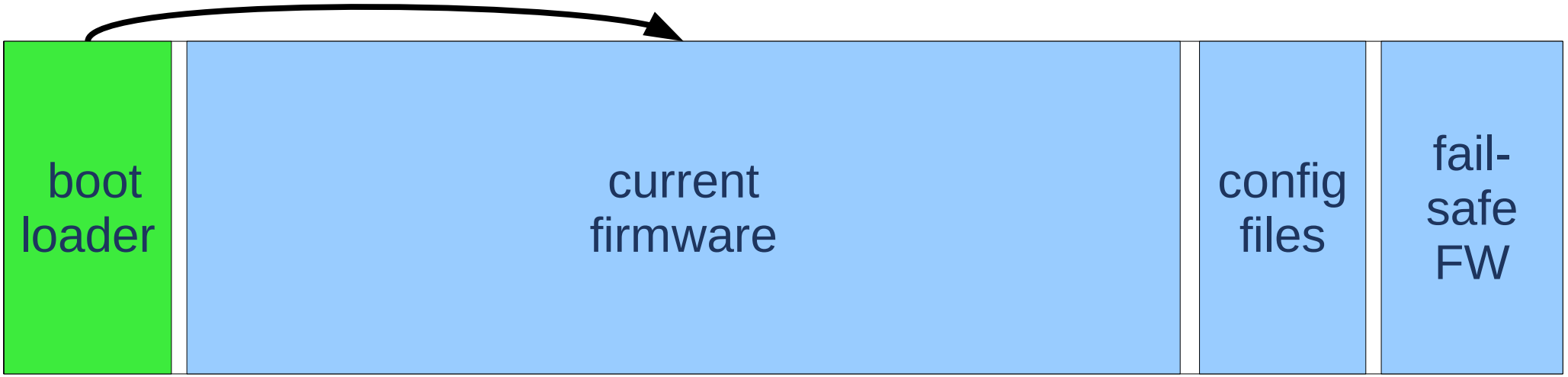
Solutions:

- Add *fail-safe* firmware
- *Detect* failed power
- *Atomic* update of firmware images

- Use journalling filesystem for writable data

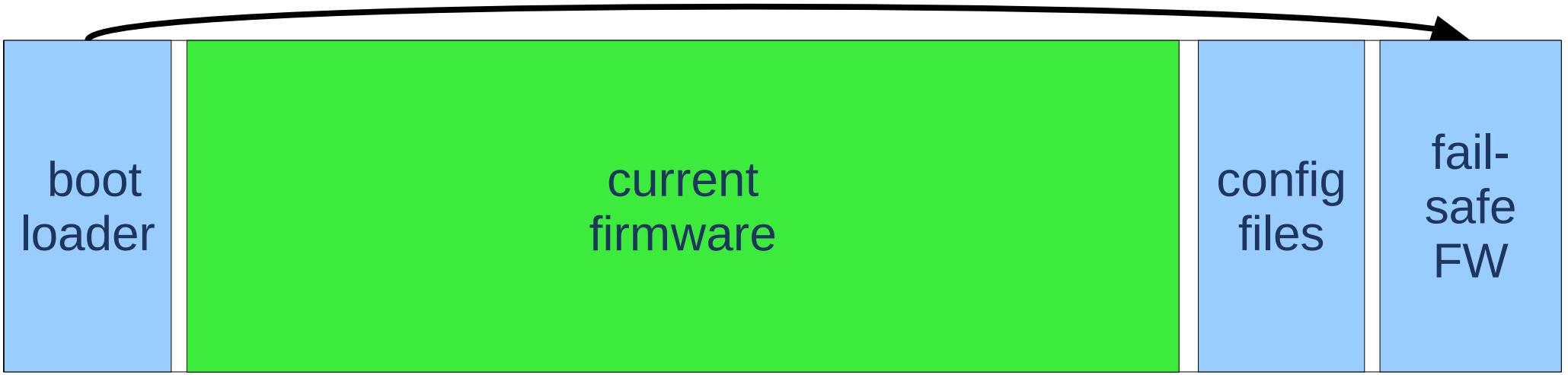
Detecting power failure: Switch to fail-safe firmware

1. Boot current firmware

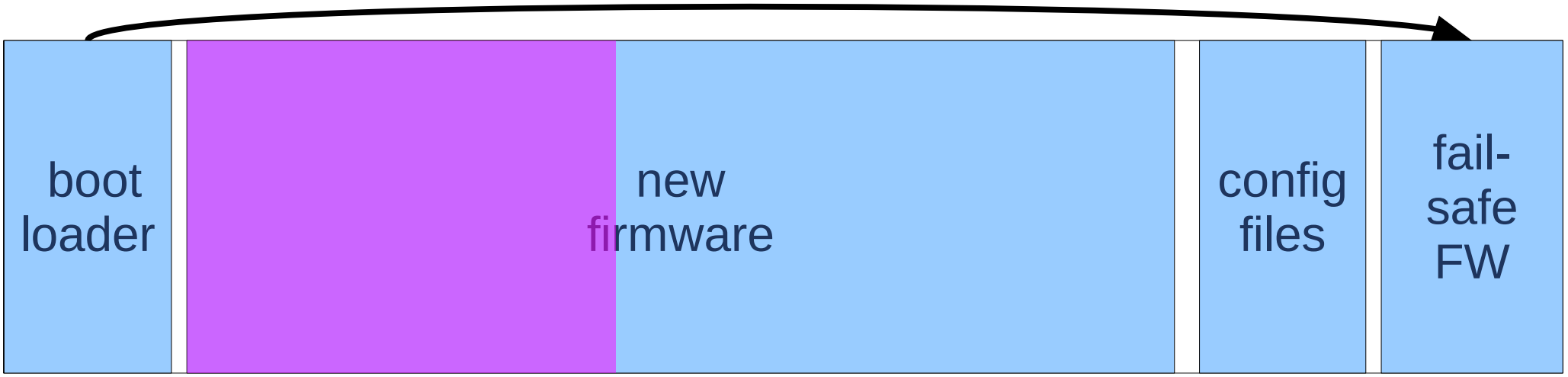


Detecting power failure: Switch to fail-safe firmware

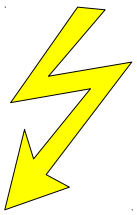
2. Switch to fail-safe



Detecting power failure: Switch to fail-safe firmware

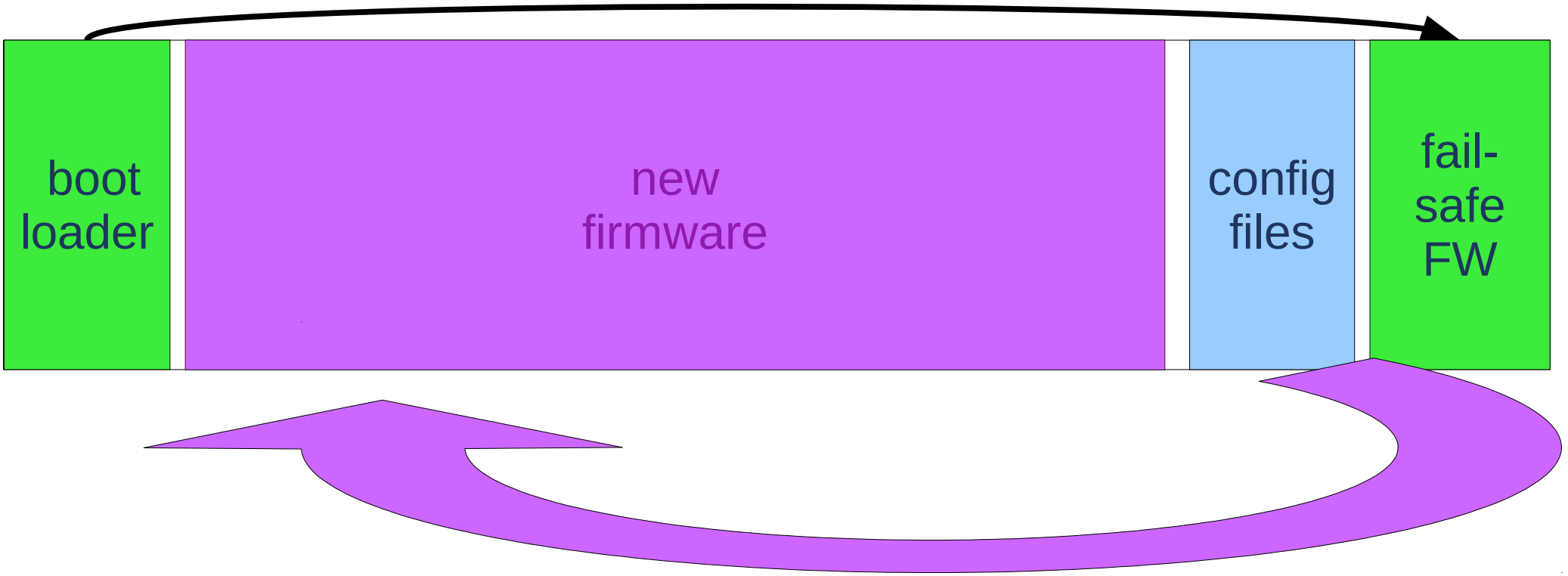


3. Overwrite firmware



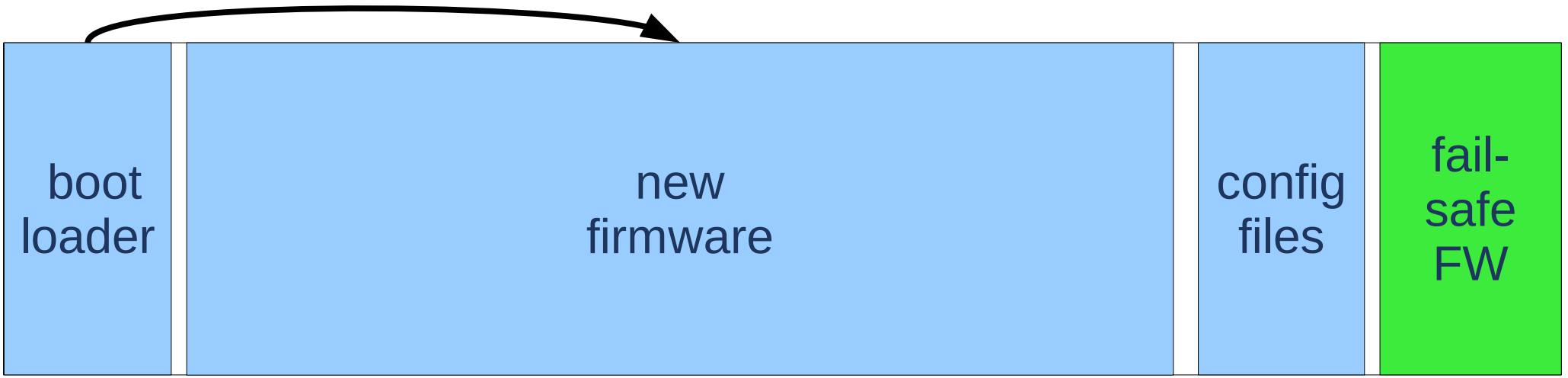
Detecting power failure: Switch to fail-safe firmware

4. Fail-safe restarts upgrade



Detecting power failure: Switch to fail-safe firmware

5. back to new firmware



Can bootloader switch to fail-safe ***atomically***?

- Grub, extlinux
  - Overwrite a file
  - ⇒ Make sure overwrite is atomic, using rename(2)
  - ⇒ Relies on atomicity of underlying filesystem implementation
    e.g. ext4: mount with `barrier=1`
- U-Boot
  - Overwrite environment
  - ⇒ Catastrophic if power fails during environment write

Atomic switching through boot-loader's fallback

All boot loaders have a fallback mechanism

Destroy normal boot before starting the upgrade

Put the normal boot in a separate boot-loader script,
so it can be destroyed independently

Create boot script atomically

gupies project collects upgrade tools

Generic UPgrade Infrastructure for Embedded Systems

https://gitorious.org/gupies

- Generate boot scripts (atomically)
- Boot loader config to use boot script and failsafe
- Upgrade skeleton to write image and boot script

Overview

1 Failure mechanisms

- Power failure
- Bad firmware
- Communication errors

2 Boot loader upgrade

3 Package-based upgrade

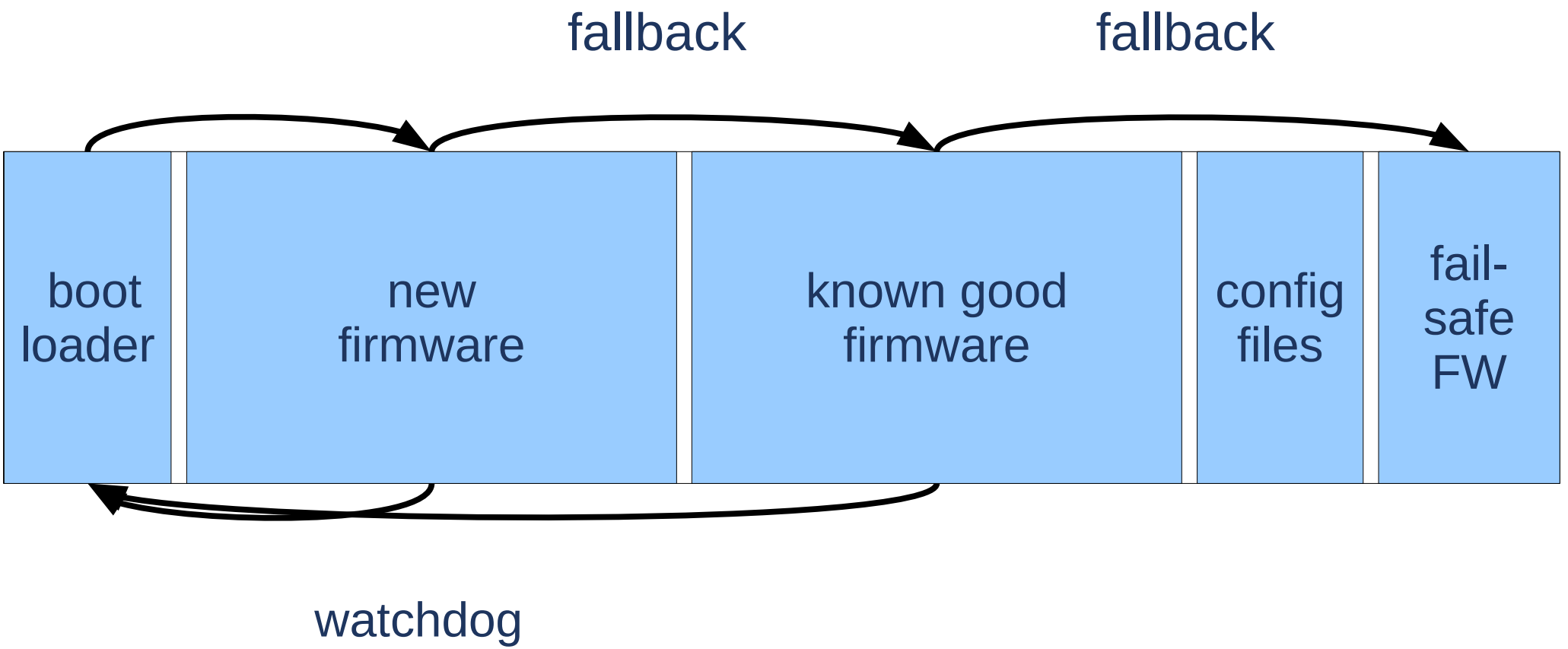
Bad firmware

New firmware fails on some devices

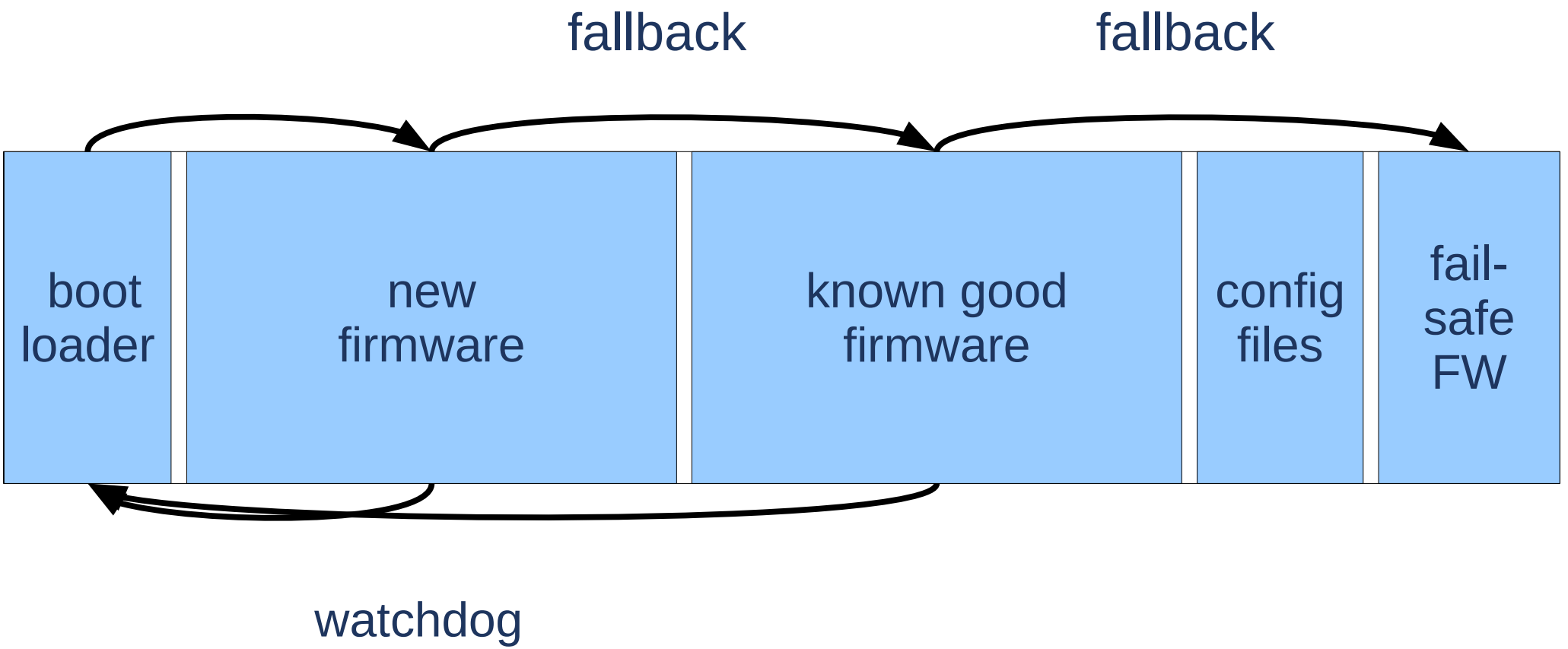
Solutions:

- Fall back on previous (known good) firmware
- Fail-safe firmware that can do upgrades
- Upgrade script included in upgrade image
- Watchdog reboot +
  boot fail-safe after bad boot

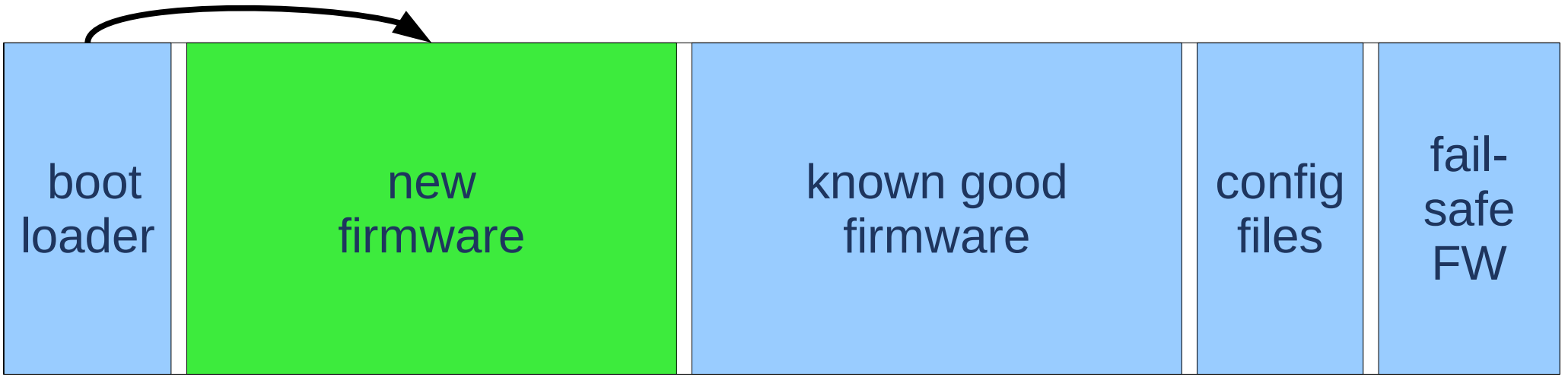
Typical flash layout with known good and fail-safe firmware



Typical flash layout with known good and fail-safe firmware



Boot procedure with watchdog



Boot procedure with watchdog

Reboot when watchdog timer expires
Reset watchdog if firmware runs well
Force reboot if firmware does not run well

gupies project has infrastructure for known-good

- Keep track of valid firmware images
  - taking into account multiple components (kernel, rootfs)
- Clean up old components

Overview

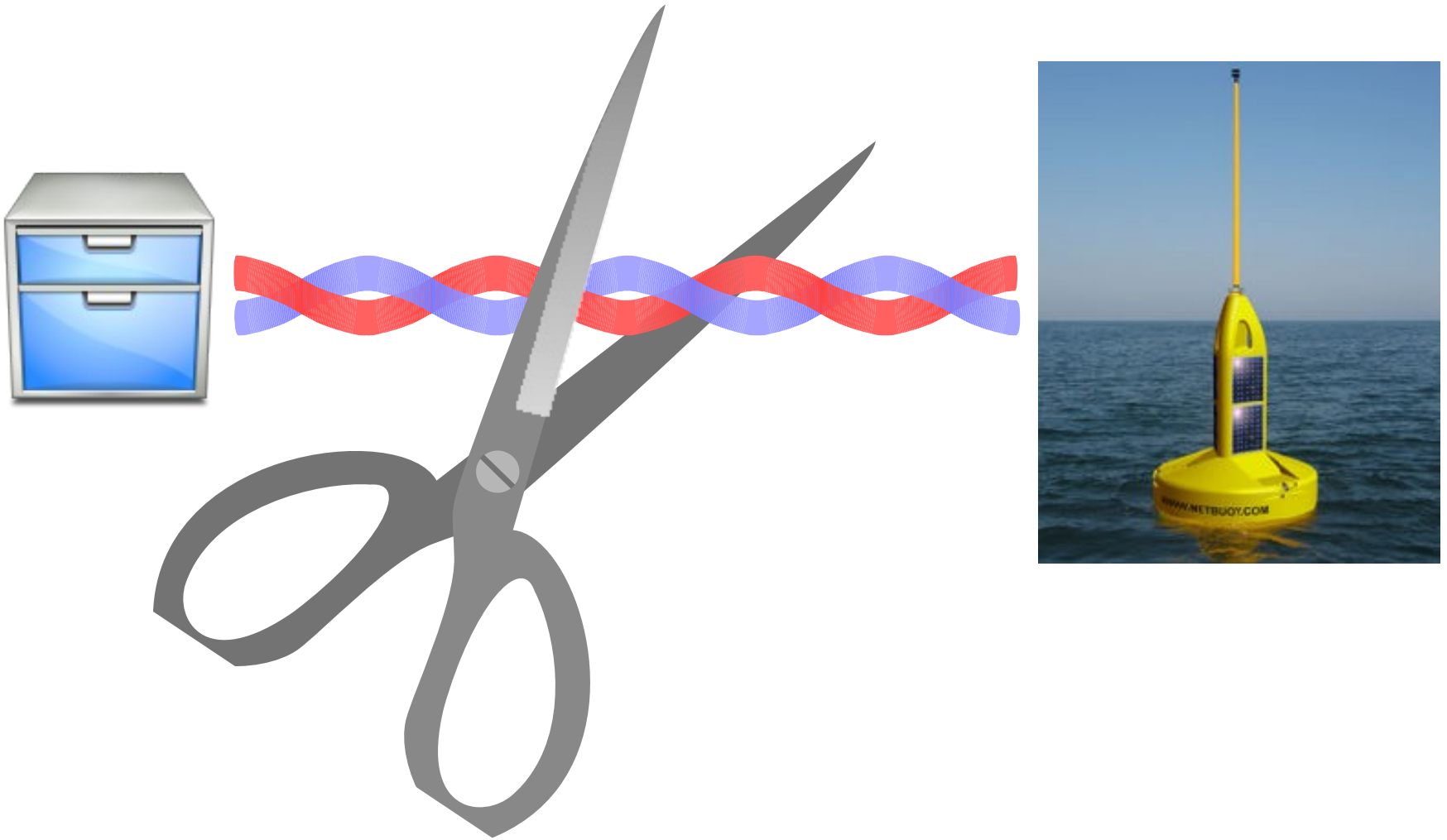
1 Failure mechanisms

- Power failure
- Bad firmware
- Communication errors

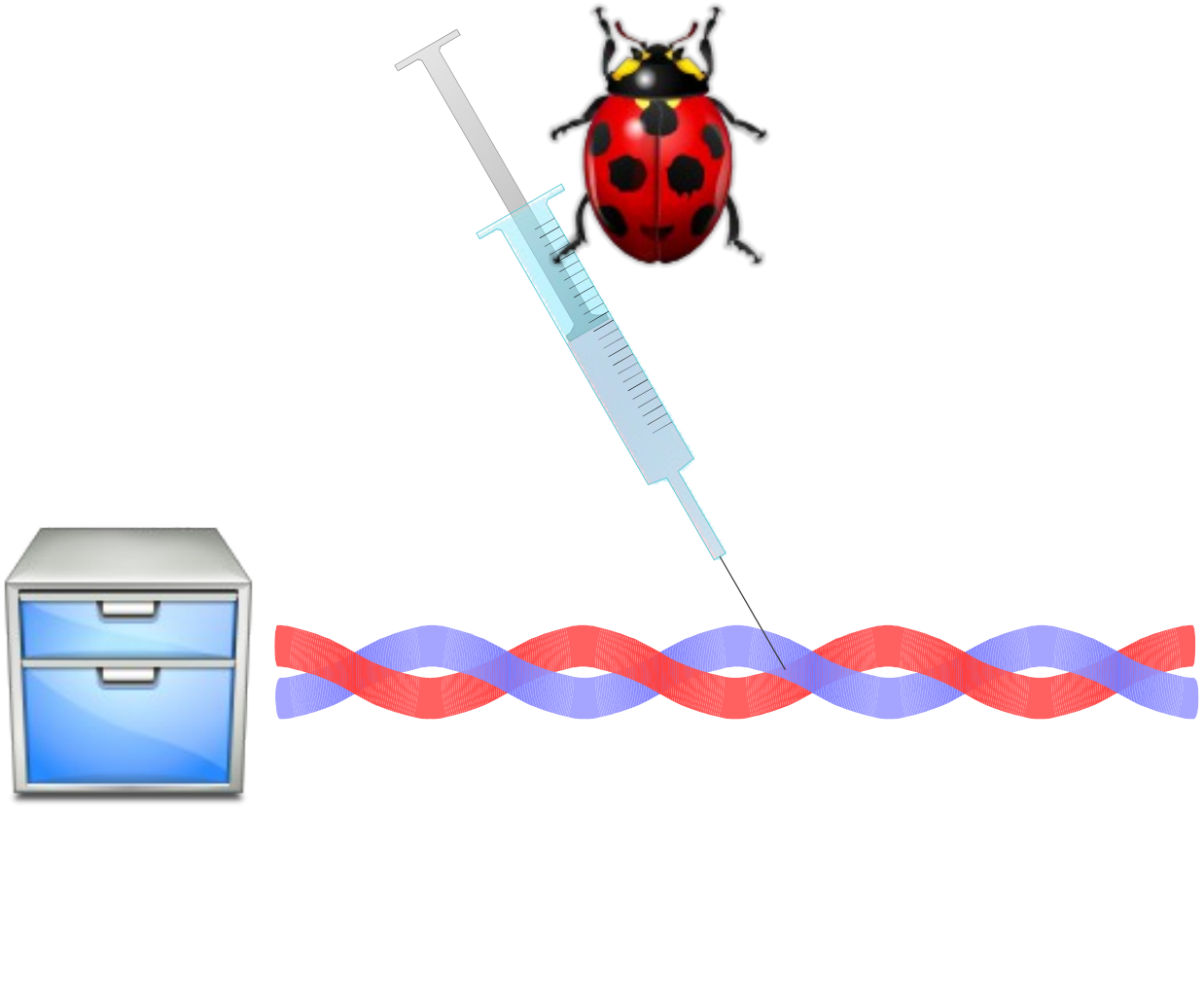
2 Boot loader upgrade

3 Package-based upgrade

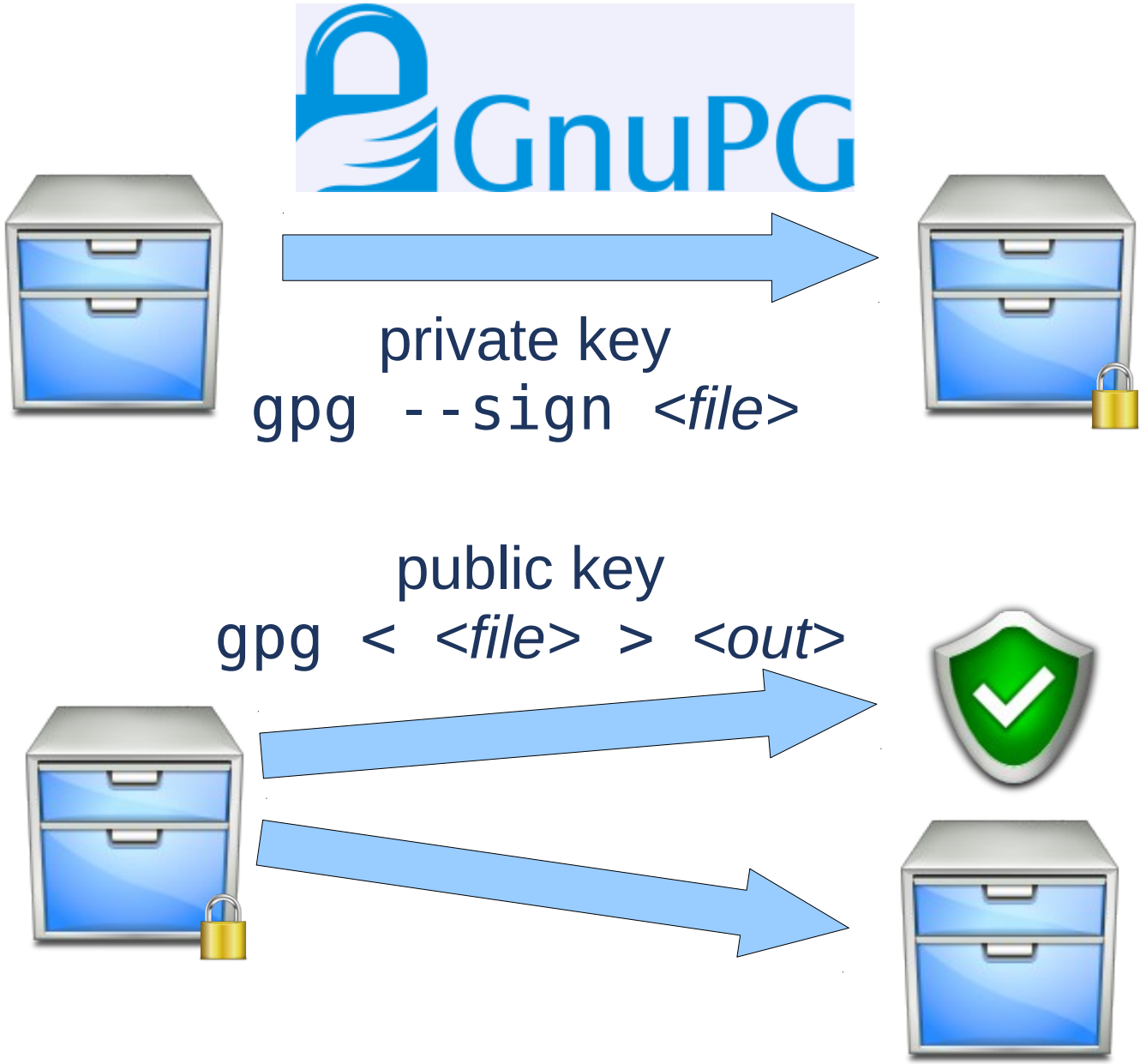
Communication failures: Incomplete upgrade file



Communication failures: False upgrade file injection



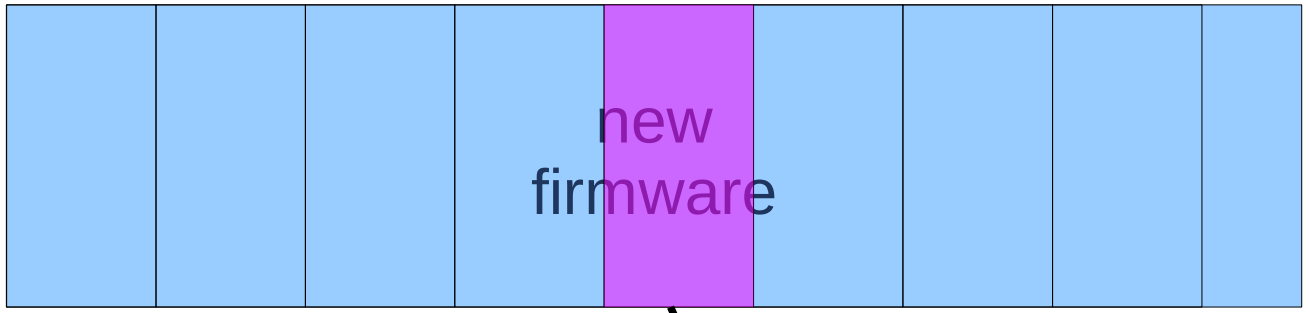
Solution for communication failures: verify data before writing



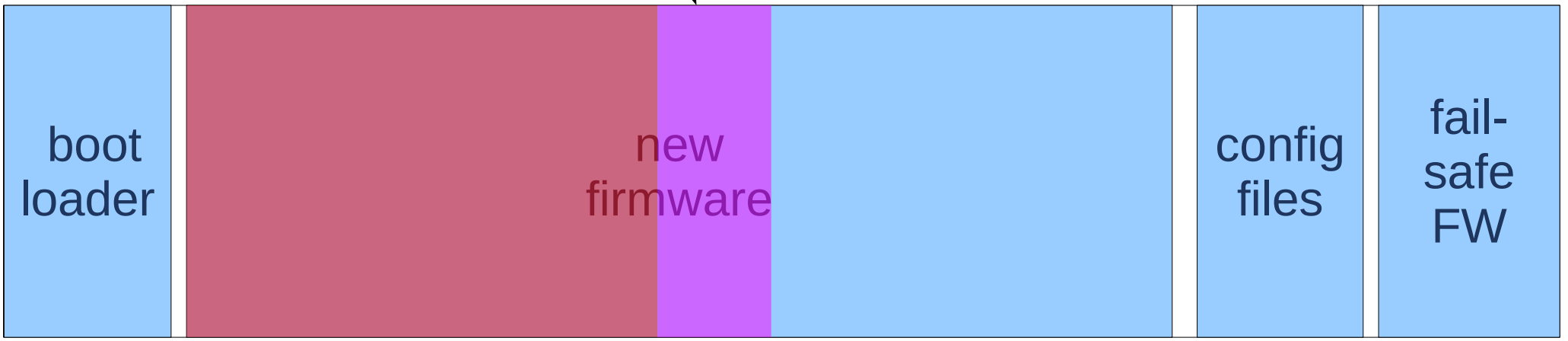
Take care with signed upgrade files

- Make it possible to install new public keys
  - Signer key may expire
  - Give third parties possibility to create upgrades
  - Avoid tivoization
- Make it possible to install revocations
  - Signer key may be stolen
- Make new keys and revocations accessible to fail-safe

Split in chunks for streaming upgrade



- No need to keep entire upgrade in memory
- Early abort in case of failure



gupies has infrastructure for chunked upgrade streams

- GUP format
- Code to generate/parse GUP
- Verify GUP with GPG
- Create GUP script from fragments

GUP format makes streaming possible

```
# SU2 HEADER START #
totalsize=70568
...
# PAYLOAD 0000 #
...
# HASHES 0000 #
bde893df2da0...
# PAYLOAD 0001 #
...
# SU2 HEADER END #
-----BEGIN PGP SIGNATURE-----
...
-----END PGP SIGNATURE-----
#! /bin/sh
...
```

Overview

1 Failure mechanisms

- Power failure
- Bad firmware
- Communication errors

2 Boot loader upgrade

3 Package-based upgrade

Upgrade of boot loader is never safe

If boot loader is broken

No recovery is possible

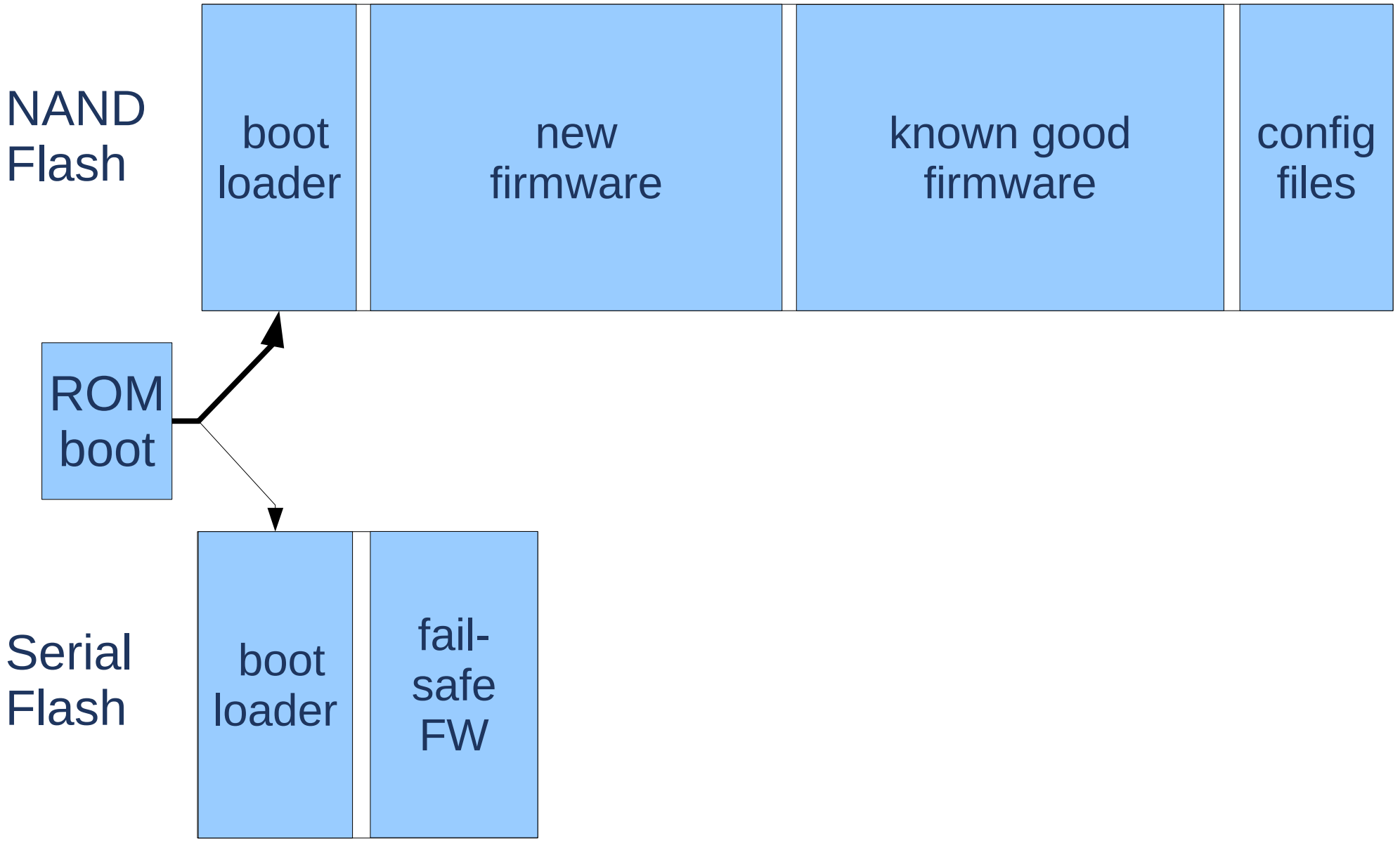
⇒ don't put bugs in the boot loader

⇒ don't put features in the boot loader

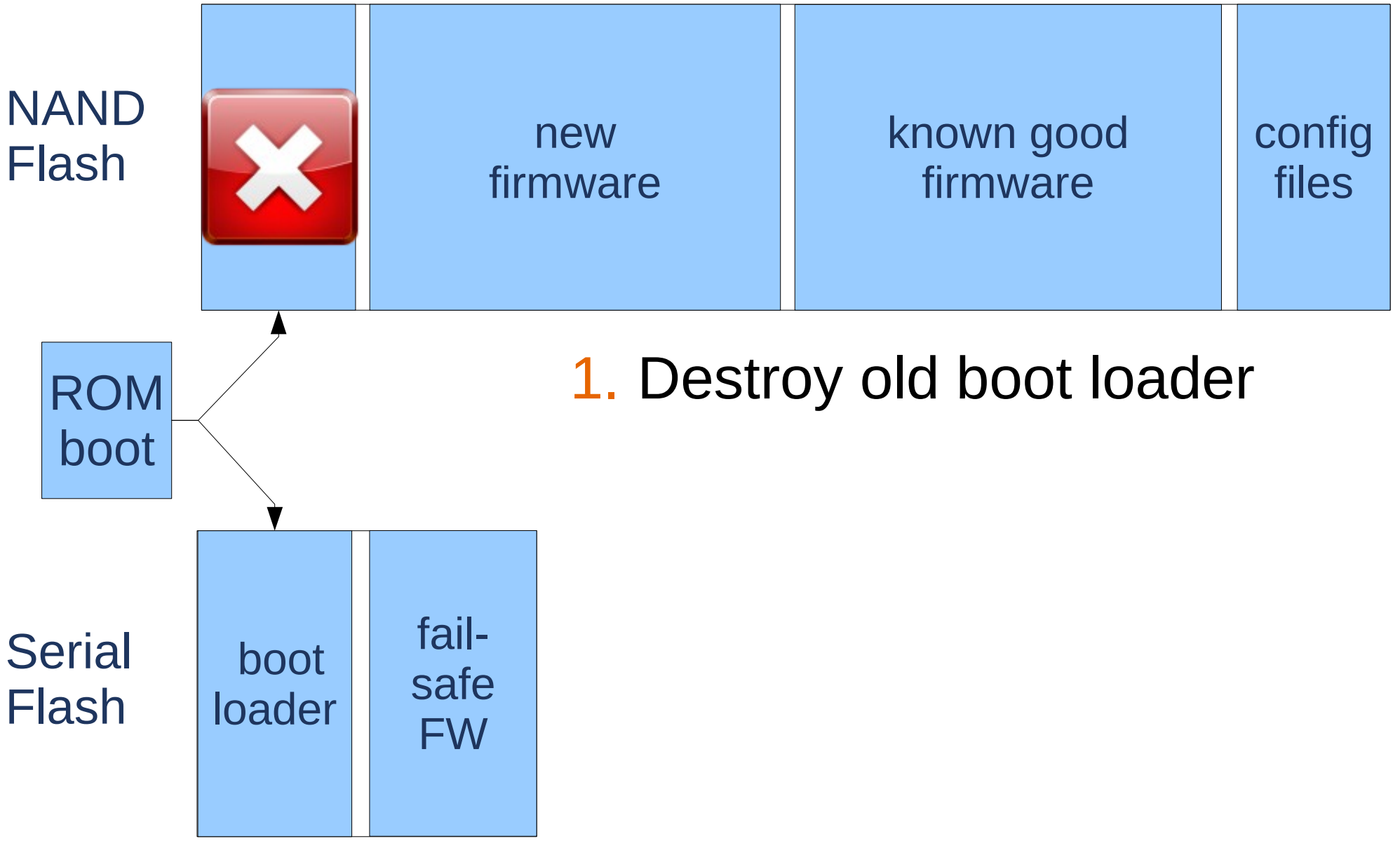
Don't put features in boot loader

- Fancy boot loader is nice in development
- Don't rely on it in deployment
  except for fallback
- Put upgrade intelligence
  in upgrade file itself

Upgrade of boot loader with backup media

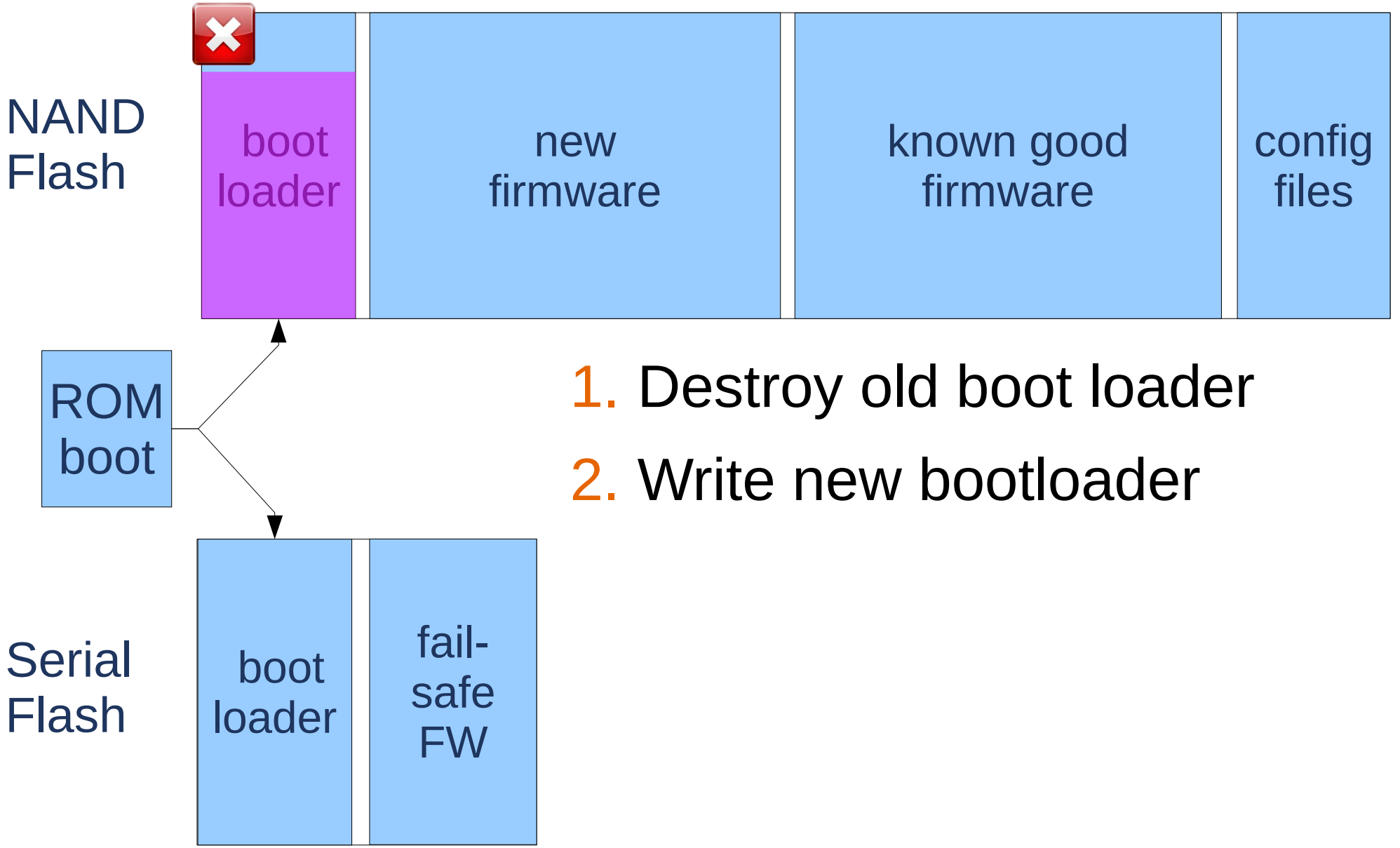


Upgrade of boot loader with backup media



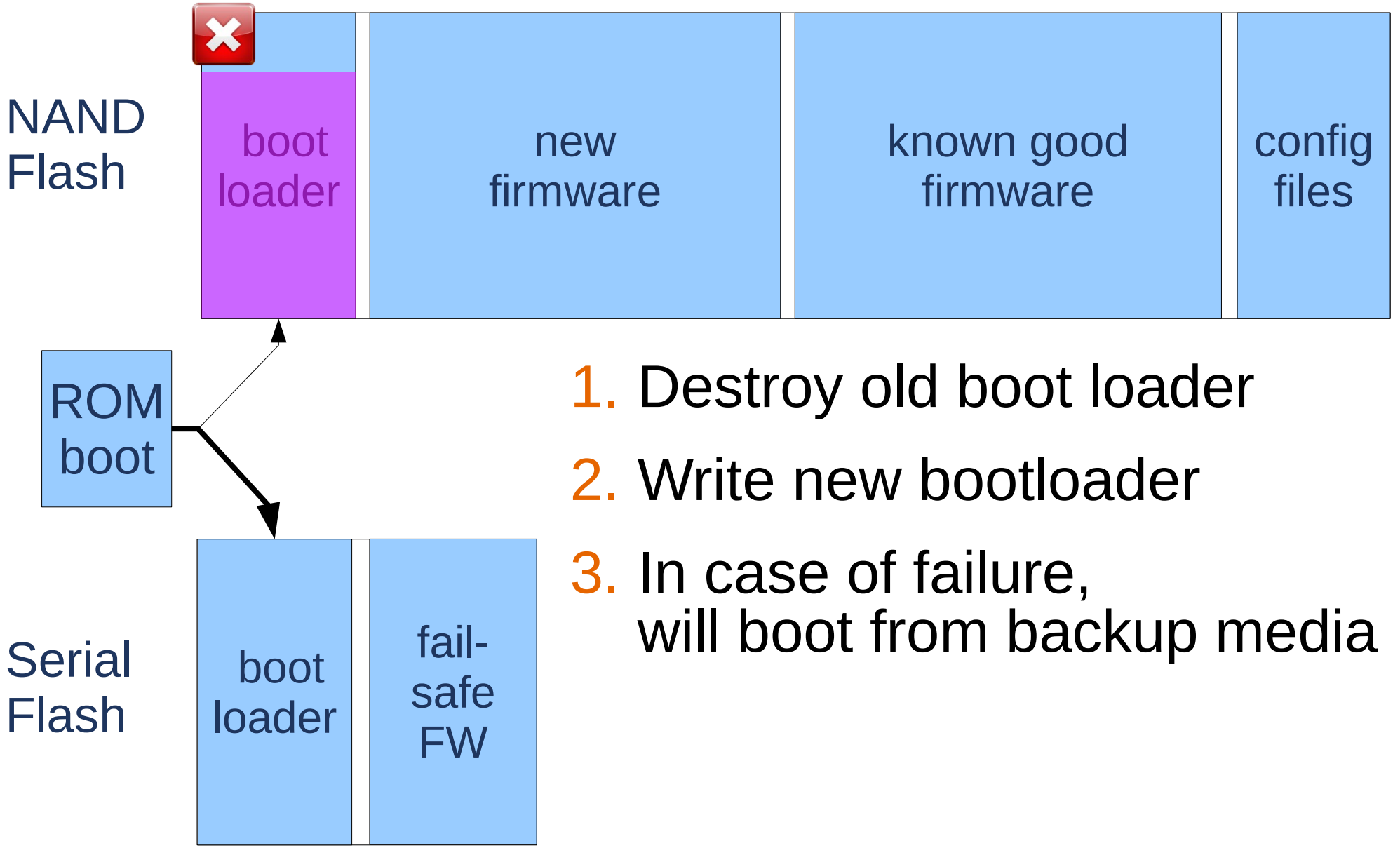
1. Destroy old boot loader

Upgrade of boot loader with backup media



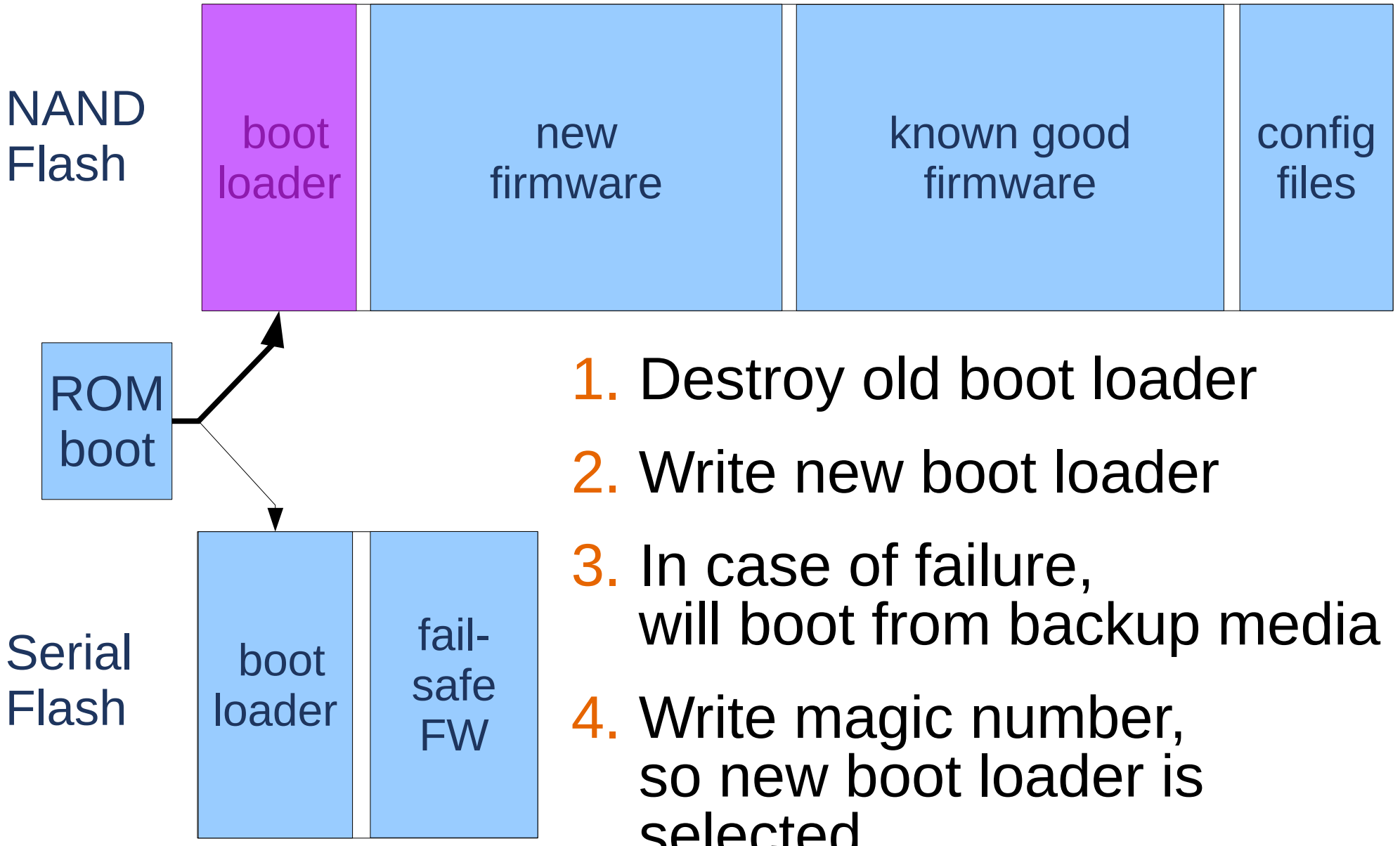
1. Destroy old boot loader
2. Write new bootloader

Upgrade of boot loader with backup media



1. Destroy old boot loader
2. Write new bootloader
3. In case of failure,
   will boot from backup media

Upgrade of boot loader with backup media



1. Destroy old boot loader
2. Write new boot loader
3. In case of failure, will boot from backup media
4. Write magic number, so new boot loader is selected

Overview

1 Failure mechanisms

- Power failure
- Bad firmware
- Communication errors

2 Boot loader upgrade

3 Package-based upgrade

Packaged-based upgrades are not ideal for embedded systems

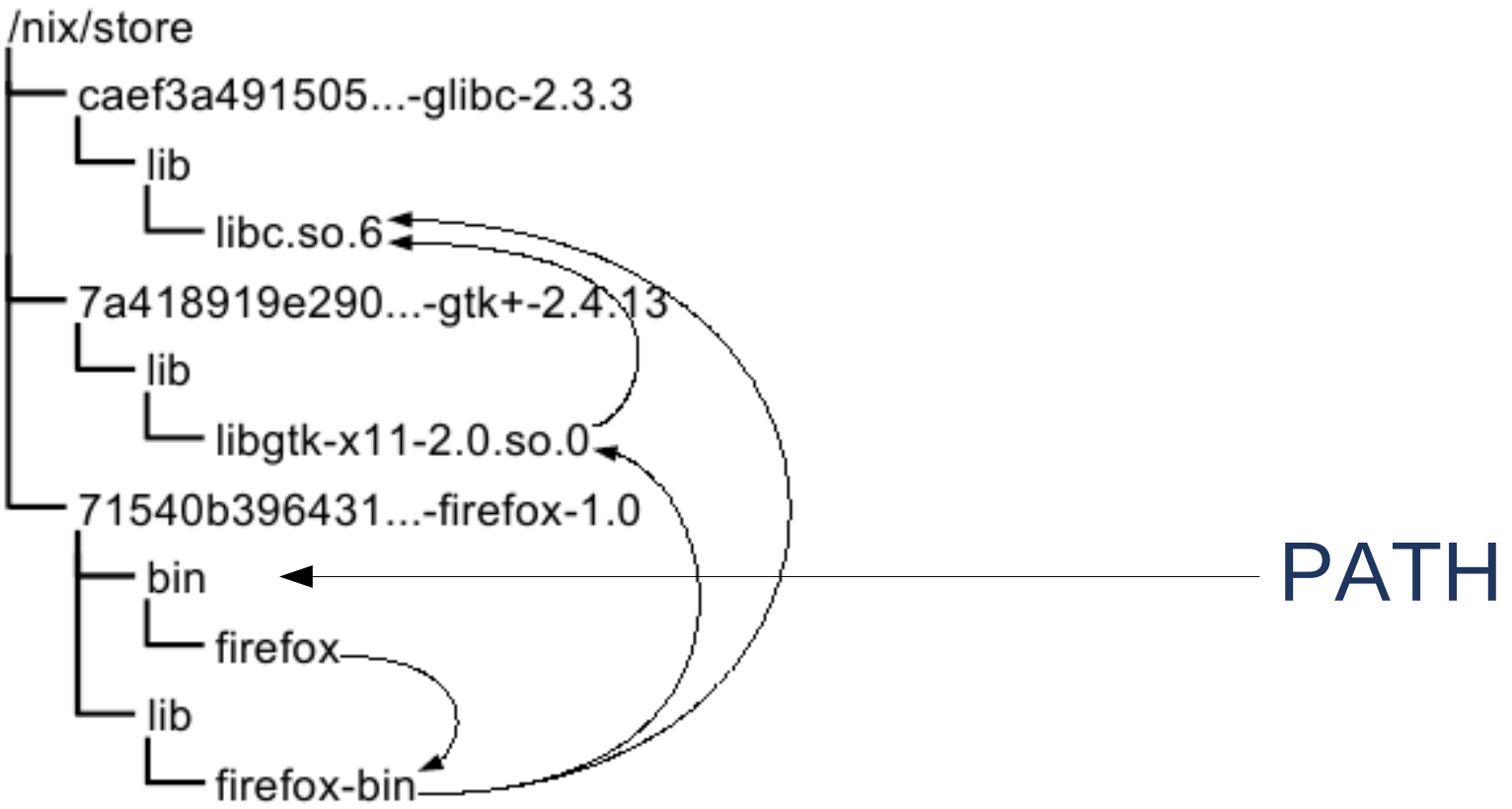
Use a package manager (ipkg, opkg, dpkg, rpm) and upgrade individual packages

Advantage: smaller upgrade files

Disadvantages:

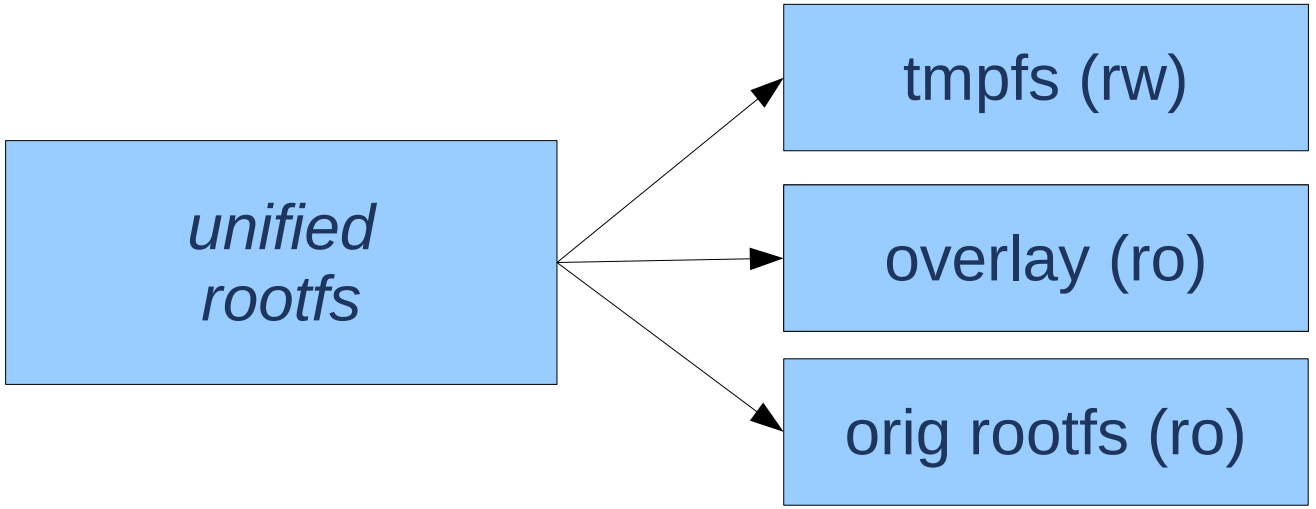
- Difficult to predict what is installed exactly
  ⇒ don't rely on version numbers,
  but use manifest with exact package versions
- More places where something can go wrong (Murphy)
- No package manager is truly atomic
  closest: http://nixos.org

Nix package manager is largely atomic



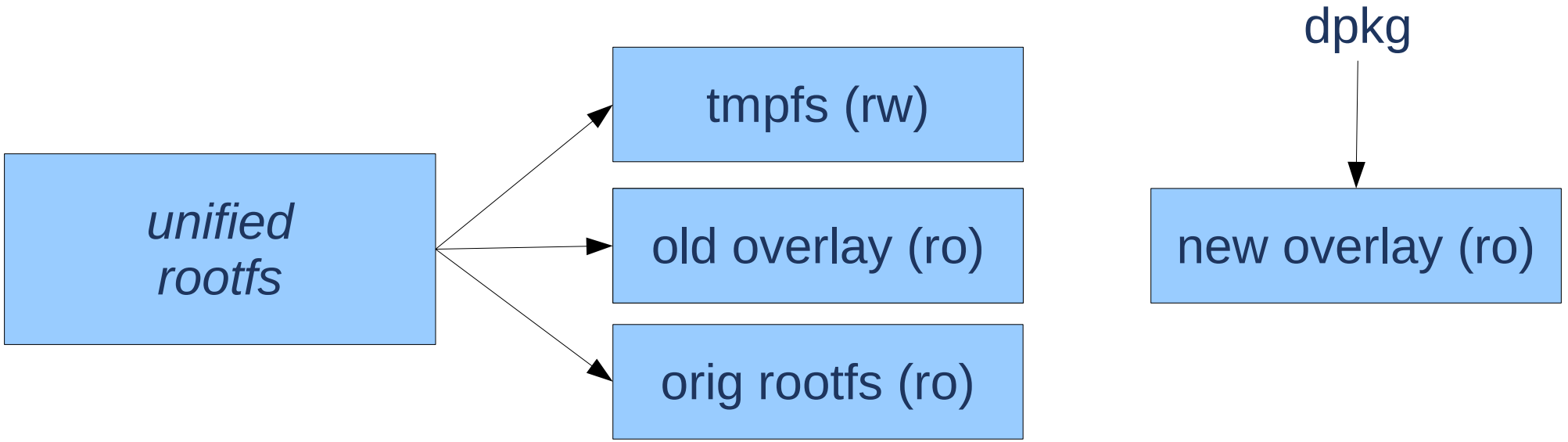
Eelco Dolstra. Efficient Upgrading in a Purely Functional Component Deployment Model. In George Heineman et al. (Ed.), Eighth International SIGSOFT Symposium on Component-based Software Engineering (CBSE 2005), volume 3489 of Lecture Notes in Computer Science, pages 219–234, St. Louis, Missouri, USA. Springer-Verlag, May 2005.

Atomic upgrade is possible with union mount



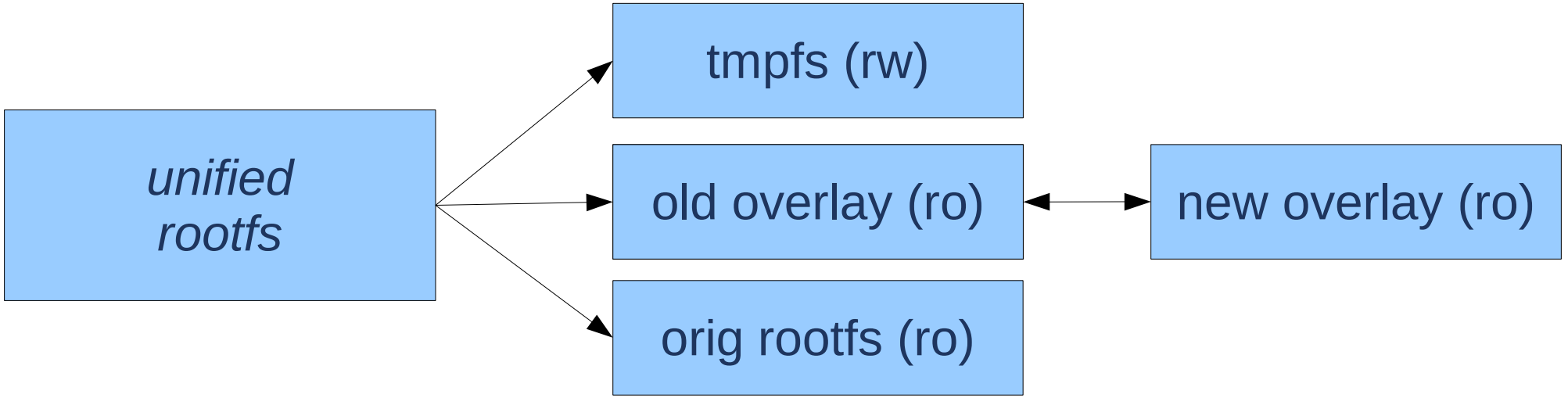
- aufs
- unionfs-fuse
- union mount

Atomic upgrade is possible with union mount



- aufs
- unionfs-fuse
- union mount

Atomic upgrade is possible with union mount



- aufs
- unionfs-fuse
- union mount

Gupies provides infrastructure for union mount

- Script to union-mount at boot time
- Components passed through /proc/cmdline
- Boot script sets kernel args appropriately

Conclusions

- Take into account different failure mechanisms:
  bad firmware, power failure, communication failure, flash corruption
- Put as much as possible in upgrade file
- No single ideal upgrade mechanism exists
  Some things really depend on the hardware
- gupies project collects upgrade infrastructure

Take your time to get the upgrade system right!

- Take into account different failure mechanisms:
  bad firmware, power failure, communication failure, flash corruption
- Put as much as possible in upgrade file
- No single ideal upgrade mechanism exists
  Some things really depend on the hardware
- gupies project collects upgrade infrastructure

http://mind.be/content/Presentation_Upgrade-without-Bricking.pdf or .odp

www.mind.be

www.essensium.com

Essensium NV
Mind - Embedded Software Division
Gaston Geenslaan 9, B-3001 Leuven
Tel : +32 16-28 65 00
Fax : +32 16-28 65 01
email : info@essensium.com