redhat.

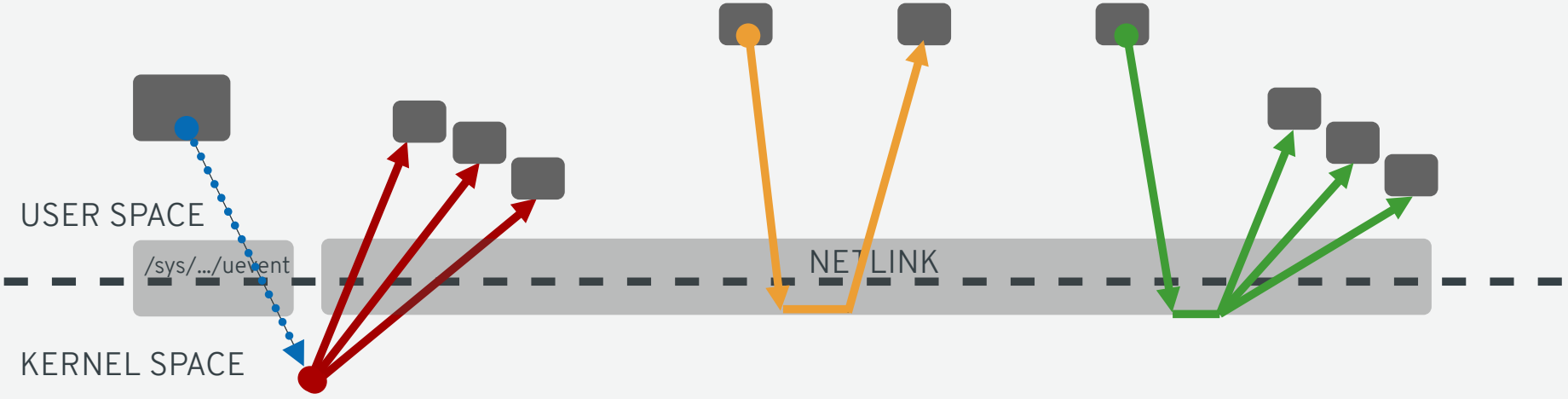
INTRODUCING

STORAGE INSTANTIATION DAEMON

PETER RAJNOHA <PRAJNOHA@REDHAT.COM>

DEVCONF.CZ, JANUARY 27 2019, BRNO

UEVENTS
OVERVIEW



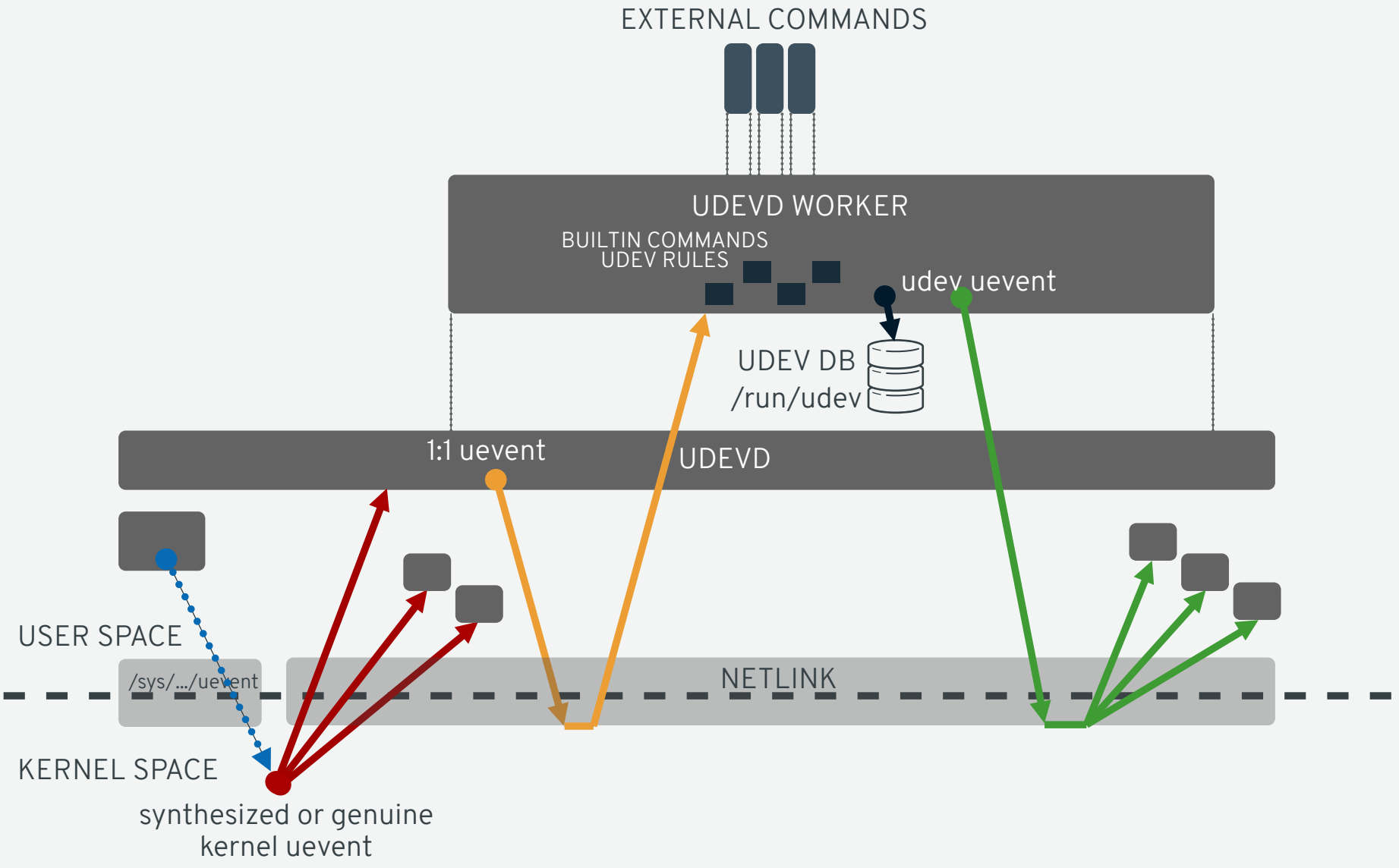
UEVENTS

- **uevents are event notifications that userspace can monitor**
- **both kernel and userspace can cause uevents to get generated**
  - ***kernel multicast*** uevents
    - genuine
    - synthesized (writing to */sys/.../uevent* file)
  - ***userspace multicast*** uevents
  - ***userspace unicast*** uevents
- **uevent environment in KEY=VALUE text format**
  - ACTION, DEVPATH, SUBSYSTEM, SEQNUM
  - more variables added by driver core, subsystems, drivers...
- **8 uevent action types:**
  - **ADD, CHANGE, REMOVE, MOVE**
  - ONLINE, OFFLINE, BIND, UNBIND
- **all uevents sent through netlink socket**

UDEV

- **udev daemon in userspace to support dynamic device management**
- **monitoring netlink socket for uevents (*kernel* uevent type)**
- **processing udev rules**
  - key=value matching/writing
  - sysfs property matching/writing
  - sysctl parameter matching/writing
  - tag matching/creation
  - executing builtin or external commands, collecting output
  - **setting device node permissions**
  - **creating symlinks to device nodes**
- **storing records in udev database**
  - records per device
  - subset of key=value environment sent with uevent
  - key=value pairs added by rules
- **regenerarating uevents including key=value pairs resulted from udev rule processing (*udev* uevent type)**
- **others able to monitor *kernel* and/or *udev* uevents**

UEVENTS + UDEV



STORAGE SPECIFICS

- **the ideal: one single-level device usable after *ADD* uevent**
- **the reality: device usable after further actions**
  - initialization sequence
  - multistep activation scheme
  - grouping
  - layering
- **devices may contain signatures/metadata/external configuration that define the next layer in the stack**
  - *blkid* scan for the majority
  - *multipath -c* to detect multipath components
  - detached header location for LUKS encrypted devices
  - further additional scans by various subsystems

PROBLEMS WITH UDEV
WHILE HANDLING STORAGE DEVICES

- **overloaded uevent** action type - just a *CHANGE* for lots of notifications
- **restricted** udev rule language
- calling **external commands** to make (even simple) decisions
- all **rules and keys are global**, any rule can overwrite values for various keys
- **accessing udev database** from udev rules is clunky and error-prone
- problems with **identification** of current state
- no direct support for **grouping**
- no standard on marking device as **ready/usable, public, private, temporarily private**
- **amount of work** done within udevd context may not be appropriate
- udevd worker process **timeout** causes the process to get killed without further fallback
- scheduling separate work requires complex **synchronization** scheme

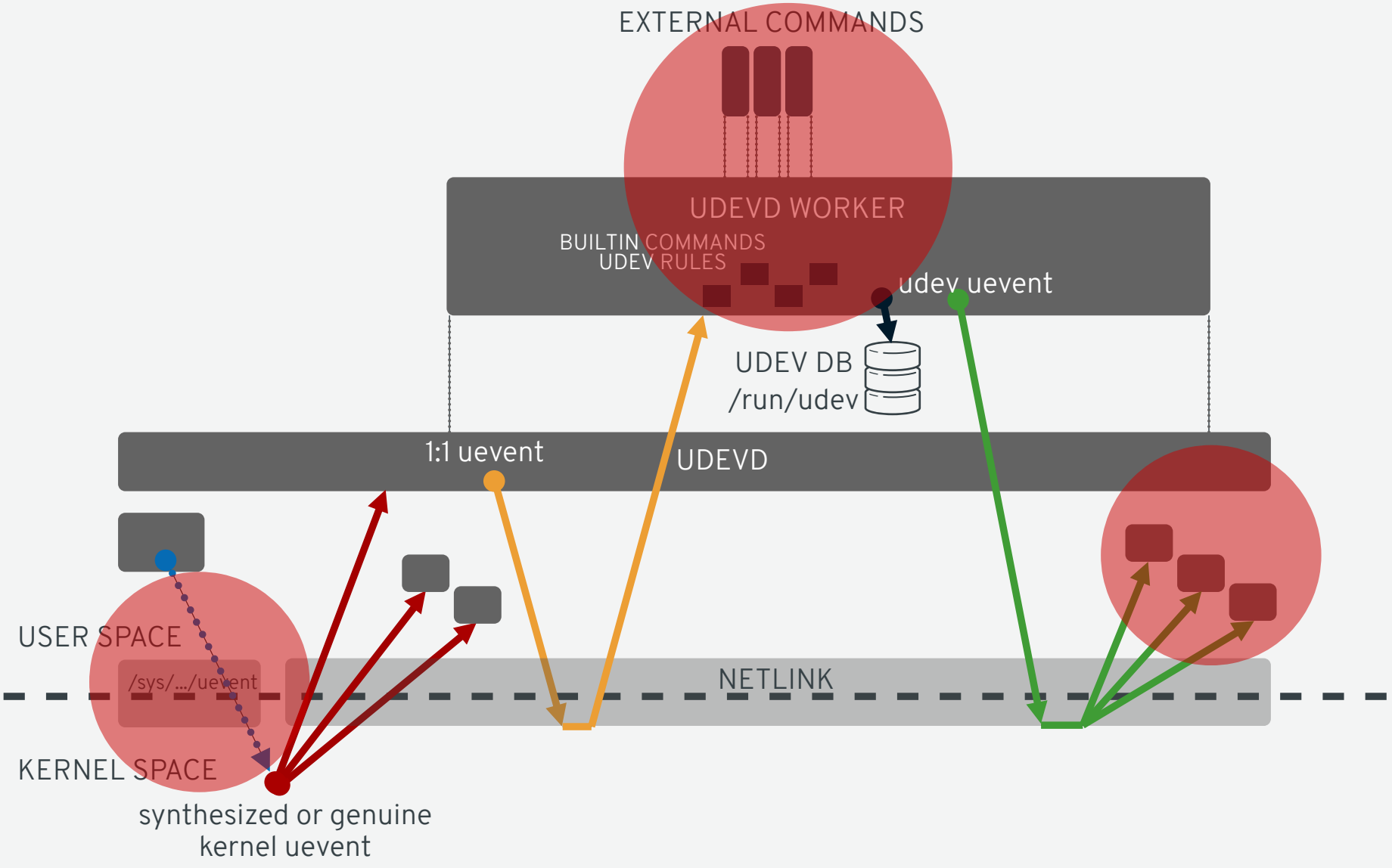
UDEV IS NOT PRIMARILY DESIGNED FOR THIS!

IT'S DESIGNED TO HANDLE NODES AND SYMLINKS IN */DEV* AND THEIR PERMISSIONS

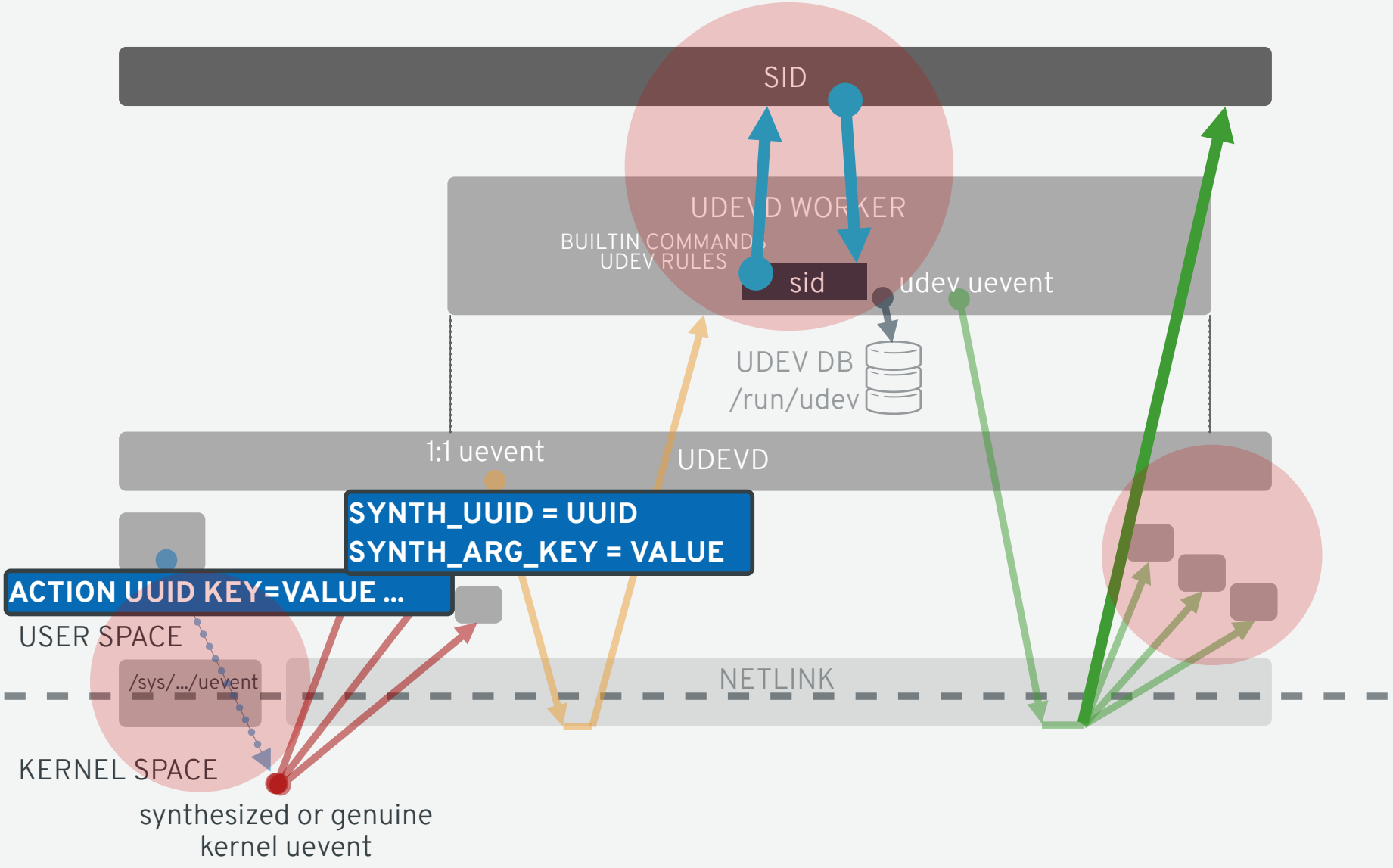
WHICH IT DOES JUST FINE

WE NEED A BIT DIFFERENT APPROACH HERE FOR OUR NEEDS!

CHANGES


EXTERNAL COMMANDS
UDEVD WORKER
BUILTIN COMMANDS
UDEV RULES
udev uevent
UDEV DB
/run/udev
1:1 uevent
UDEVD
USER SPACE
/sys/.../uevent
NETLINK
KERNEL SPACE
synthesized or genuine
kernel uevent

CHANGES

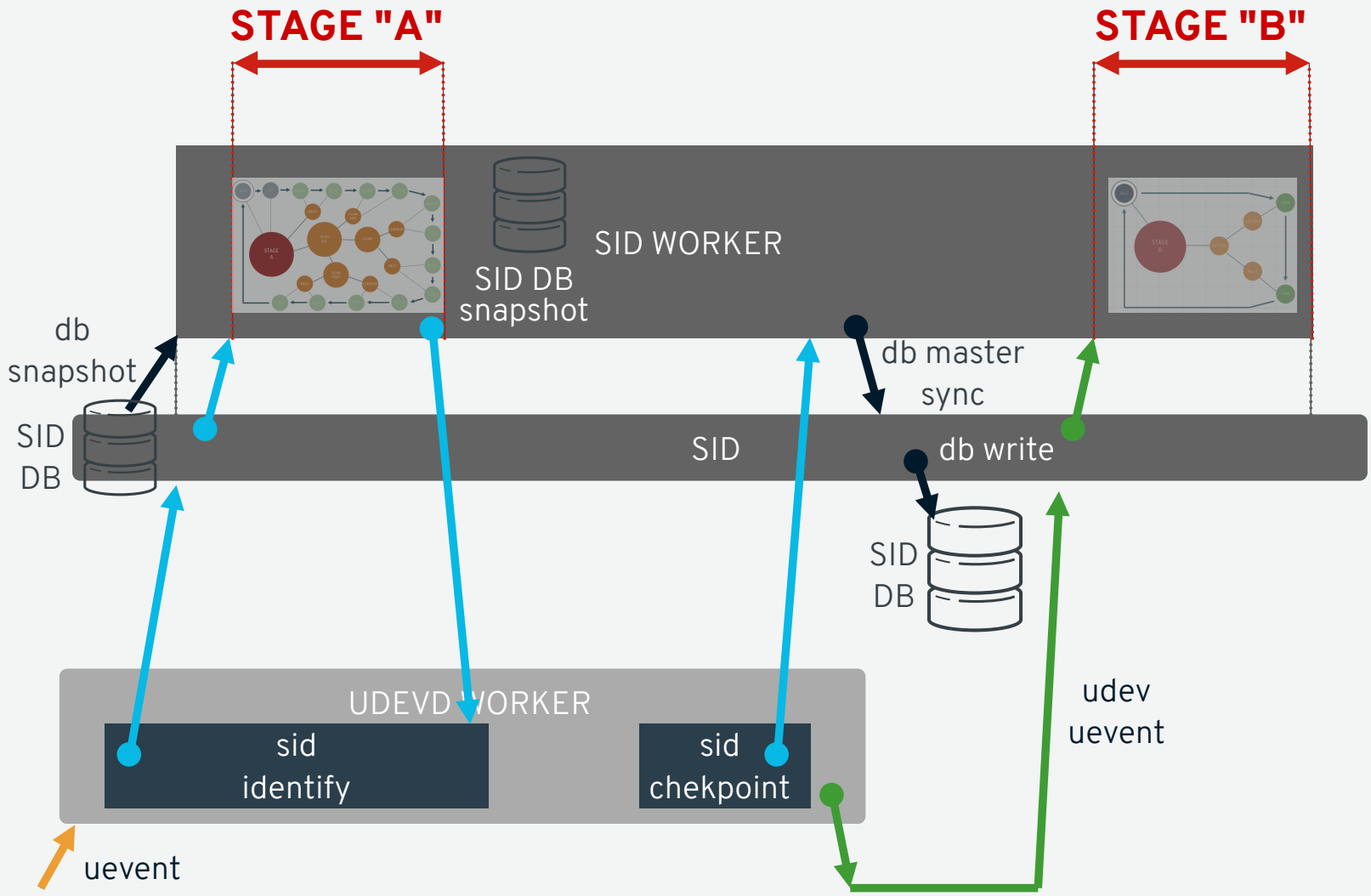

SID
UDEV WORKER
BUILTIN COMMANDS
UDEV RULES
sid
udev uevent
UDEV DB
/run/udev
1:1 uevent
UDEVD
SYNTH_UUID = UUID
SYNTH_ARG_KEY = VALUE
ACTION UUID KEY=VALUE ...
USER SPACE
/sys/.../uevent
NETLINK
KERNEL SPACE
synthesized or genuine
kernel uevent

STORAGE INSTANTIATION
DAEMON AND COMPONENTS

- ***sid* daemon**
  - layered on top of udev
  - executes storage-specific uevent handling and processing
  - keeps its own database
- **udev builtin command**
  - bridge between udev and SID with subcommands:
    - ***sid active***
      - returns *active*, *inactive*, *incompatible*
    - ***sid identify***
      - relays uevent with environment to SID
      - requests execution of identification and related routines
      - returns KEY=VALUE results for use in udev rules or to store in udev db
    - ***sid checkpoint*** *<checkpoint_name> [<key> ...]*
    - ***sid version***
- **library interface**
  - access SID's information store
  - subscribe to SID notifications
- ***sidctl* command line interface**
  - control and access SID and its information store

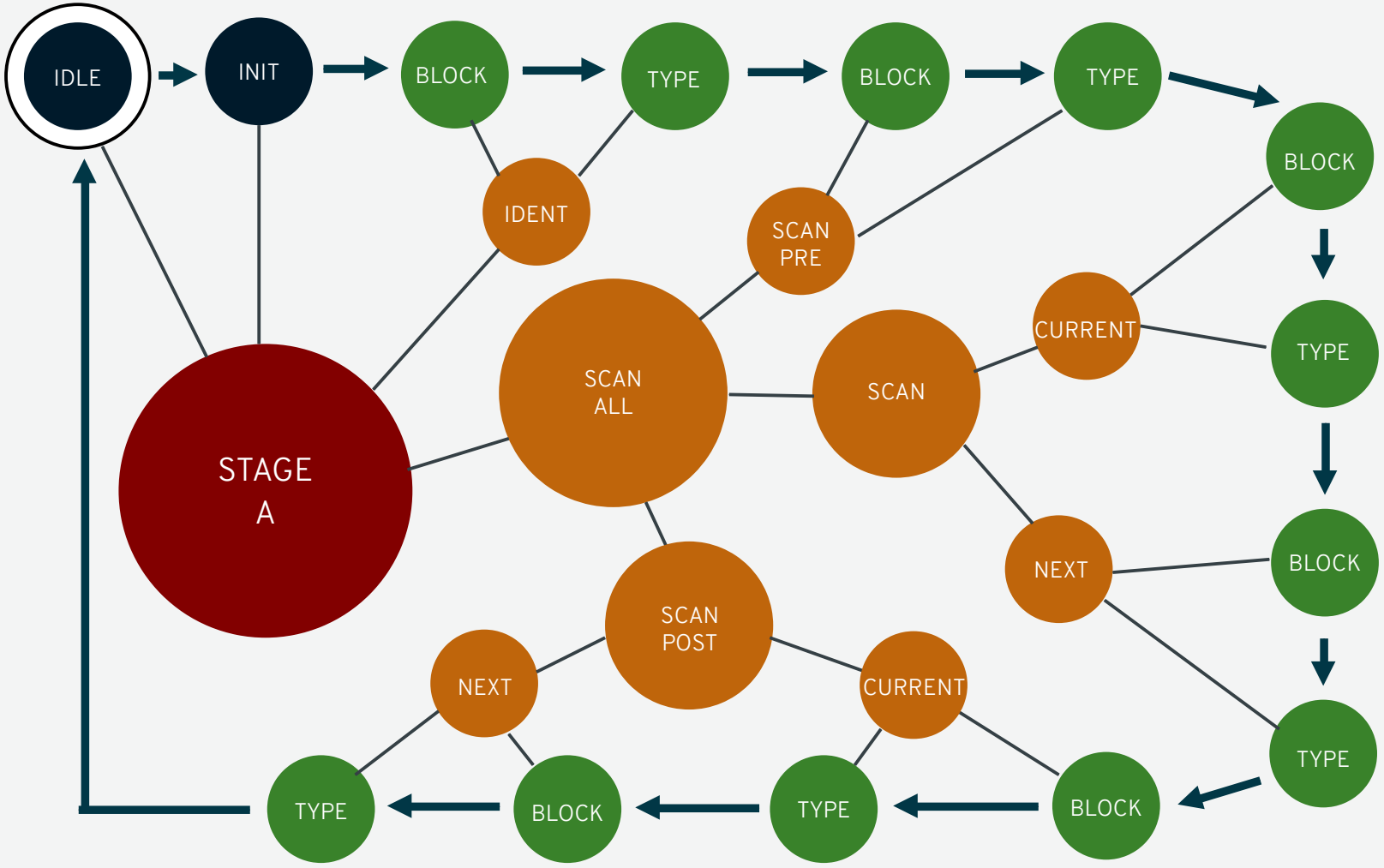
STORAGE INSTANTIATION

DAEMON IDENTIFY STAGES



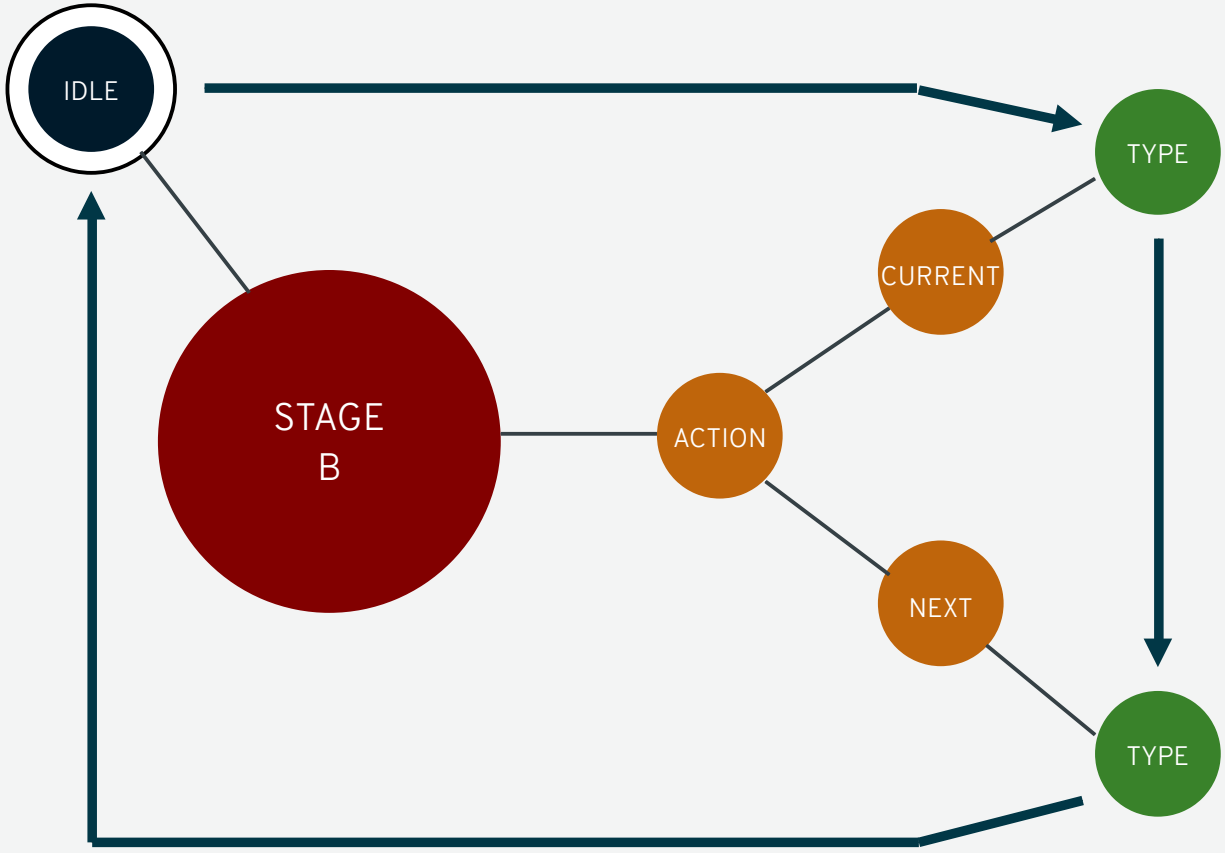
SID DAEMON

IDENTIFY - STAGE "A"



SID DAEMON

IDENTIFY - STAGE "B



SID DAEMON

DATABASE

- **key-value (KV) database with various backends**
- **value types**
  - **simple**
  - **vector**
- **snapshot separation**
- **delta synchronization of vector values**
- **separate key namespaces**
  - **KV_NS_UDEV** (import/export from/to udev)
  - **KV_NS_GLOBAL** (visible globally)
  - **KV_NS_MODULE** (visible only in specific module)
  - **KV_NS_DEVICE** (visible only when processing specific device)
- **per-module protection flags**
  - **KV_PROTECTED** (originating module can read-write, others read-only)
  - **KV_PRIVATE** (originating module can read-write, others unable to access)
  - **KV_RESERVED** (originating module reserves, others can't take over)
- **persistence**
  - **KV_PERSISTENT** (persist record for next use)

QUESTIONS ?

github: https://github.com/prajnoha/sid

freenode: prajnoha on #lvm

email: prajnoha@redhat.com

THANK YOU!

redhat®