fw_devlink

Extracting and using device dependencies from firmware

Saravana Kannan

android

Simplified Boot Sequence - Fully Modular Kernel

Bootloader — **Late initcall sync** — **Init loads modules** — **??**

Bootloader/HW reset turns on devices

For example, a HW reset might turn ON a few power supplies to the SoC, clocks and interconnects (so that it can boot).

Bootloader might turn on display and UART and any clocks and regulators needed for them.

Core kernel init

Only thing done upto this point is arch specific core init (for example, ARM64 Timer, GIC).

Kernel probes devices

Module load ordering doesn't guarantee probe ordering. For example: probe deferrals, one module containing multiple drivers, etc.

Kernel cleans up hardware state

At what point can the kernel safely clean up the hardware state? For example, turn off all regulators not explicitly requested to be kept ON by consumers.

LPC 2019 slides and video

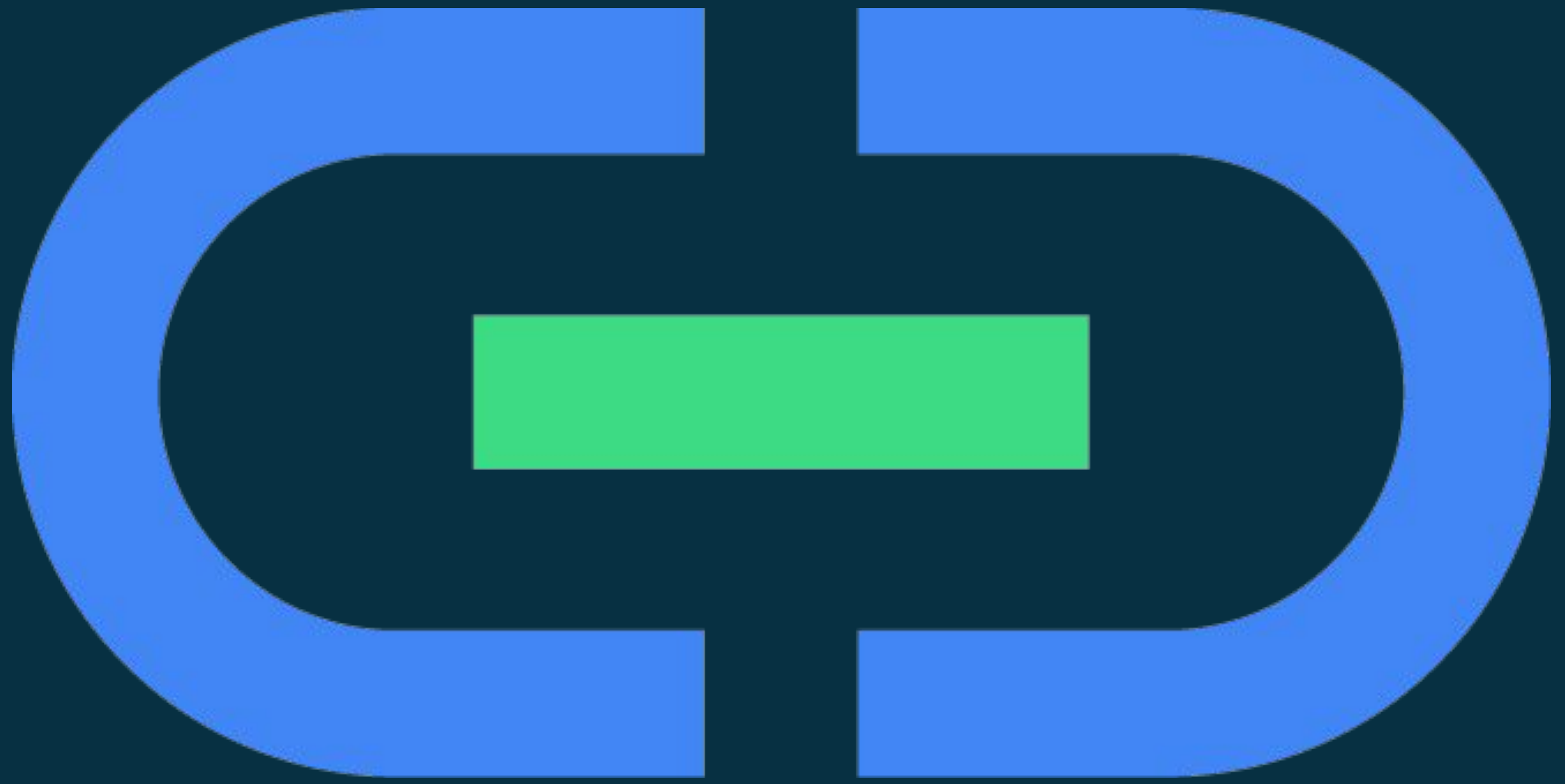
fw_devlink and sync_state

`fw_devlink` - Derive device dependencies from firmware

- Parses firmware to create device links between devices
- Doesn't depend on drivers
- Avoids unnecessary probe deferrals

`sync_state` - Per-device "safe to clean up" callback

- sync_state() callback is called after ALL the consumers of a device have probed
- Uses device links to track probe completion of consumers



android

fw_devlink: upstream status

60+ patches merged upstream since LPC 2019

Added support for several modes:

- off = doesn’t parse firmware
- permissive **[default]** = doesn’t affect probes
- on = enforces probe and suspend/resume ordering
- rpm = on + enforces run time pm ordering

Improved cycle handling.

Significantly faster firmware parsing.

Device link info now exposed in sysfs:

- `/sys/class/devlink`
- `/sys/devices/.../<device>/[supplier|consumer]:*`

android

fw_devlink DT support

Currently supports the following 18 devicetree bindings:

- clocks
- interconnects
- iommus
- iommu-map
- mboxes
- io-channel
- interrupt-parent
- dmas
- power-domains
- hwlocks
- extcon
- interrupts-extended
- nvmem-cells
- phys
- wakeup-parent
- pinctrl-*
- -supply (regulators)
- -gpio and -gpios (gpios)

sync_state(): upstream status

PSCI CPUidle driver support added by Ulf Hansson

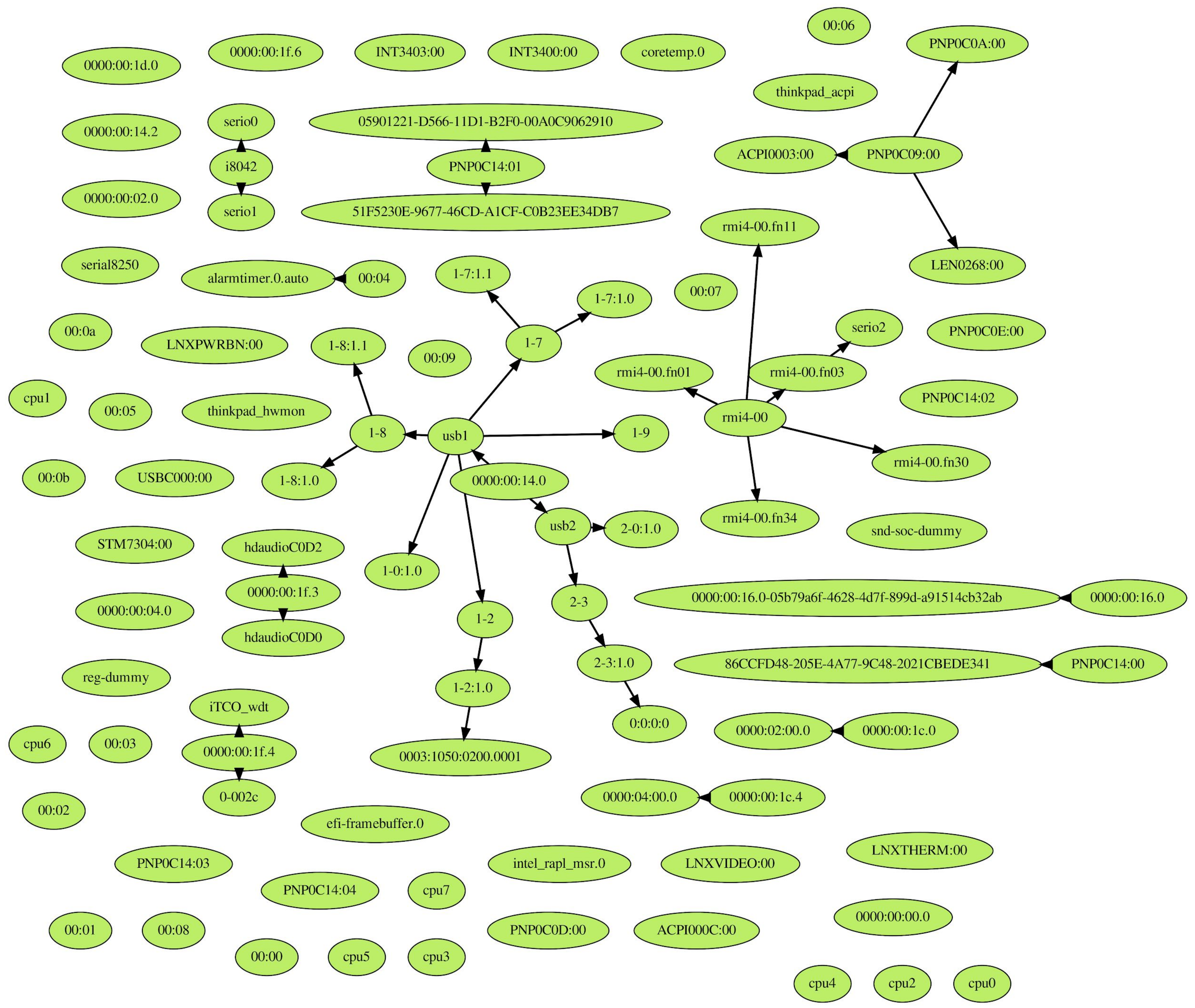
Interconnect framework support in progress by Georgi Djakov

Regulator framework patches under discussion

android

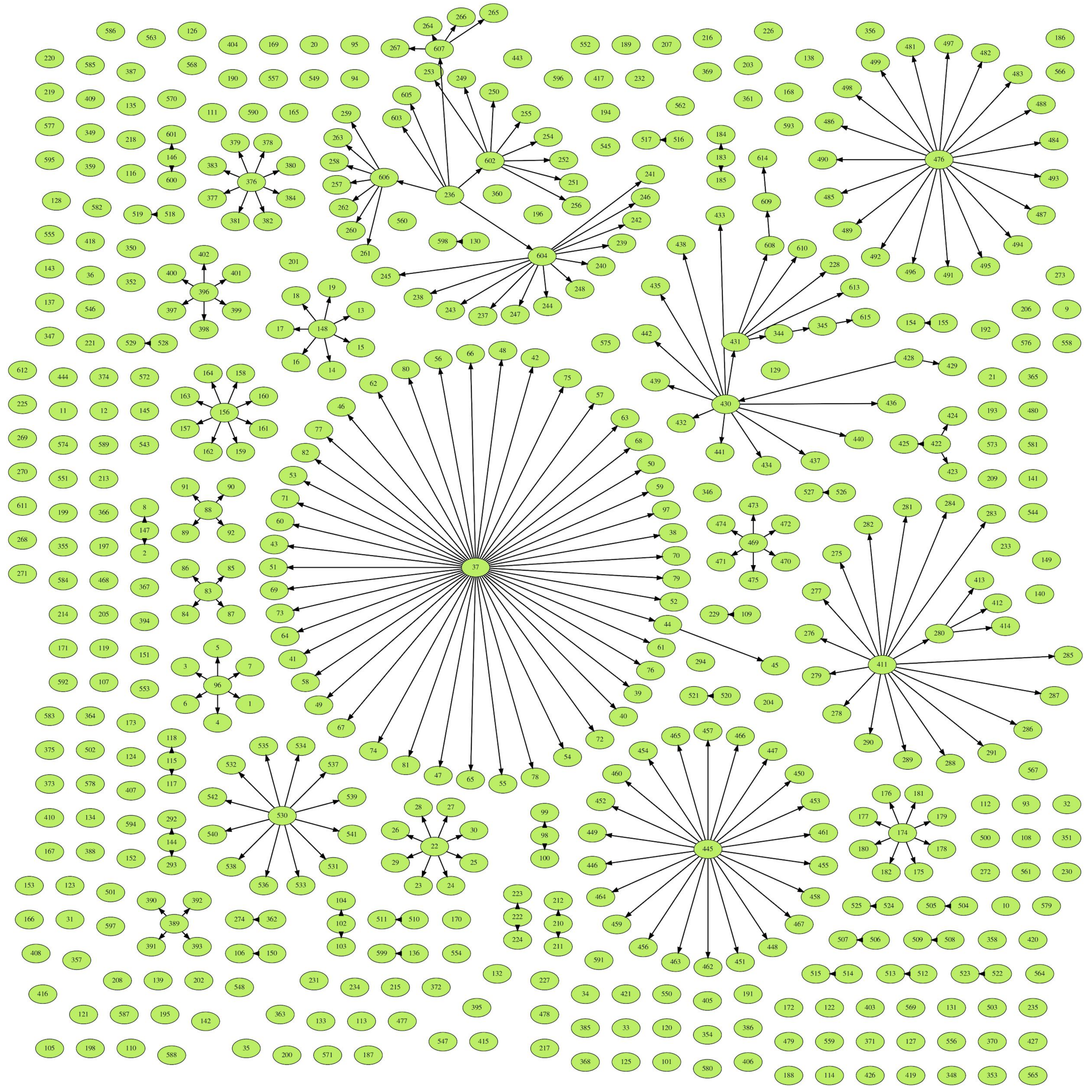
Device dependency graphs!

Laptop (without fw_devlink)



Note: Does NOT include “class” devices because they never bind to drivers

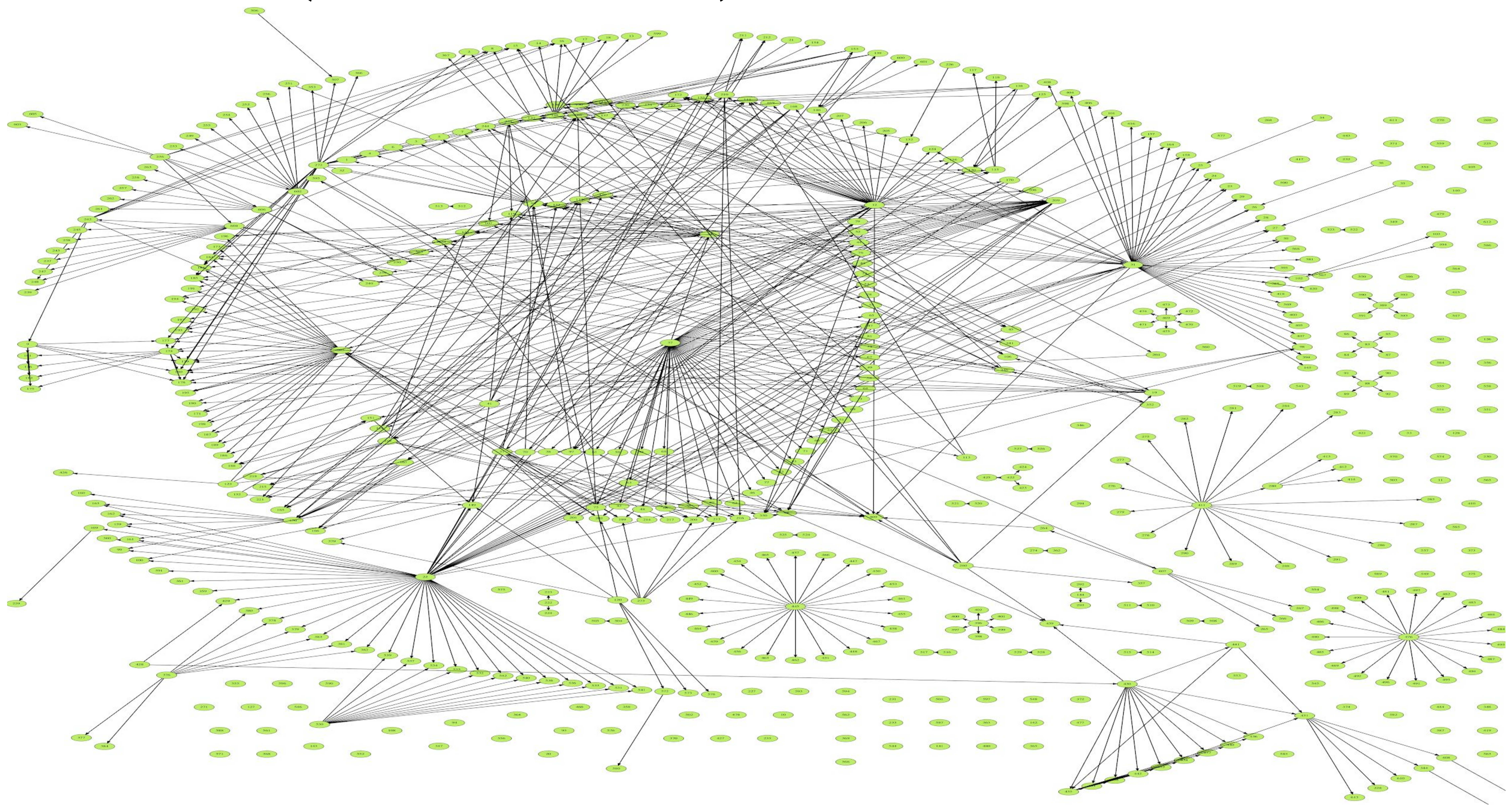
Mobile device (without fw_devlink)



Note: Does NOT include "class" devices because they never bind to drivers

android

Mobile device (with fw_devlink)



Note: Does NOT include "class" devices because they never bind to drivers

Discussion slides

android

Discussion topics

sync_state() tracking granularity

fw_devlink=on by default

android

sync_state() tracking

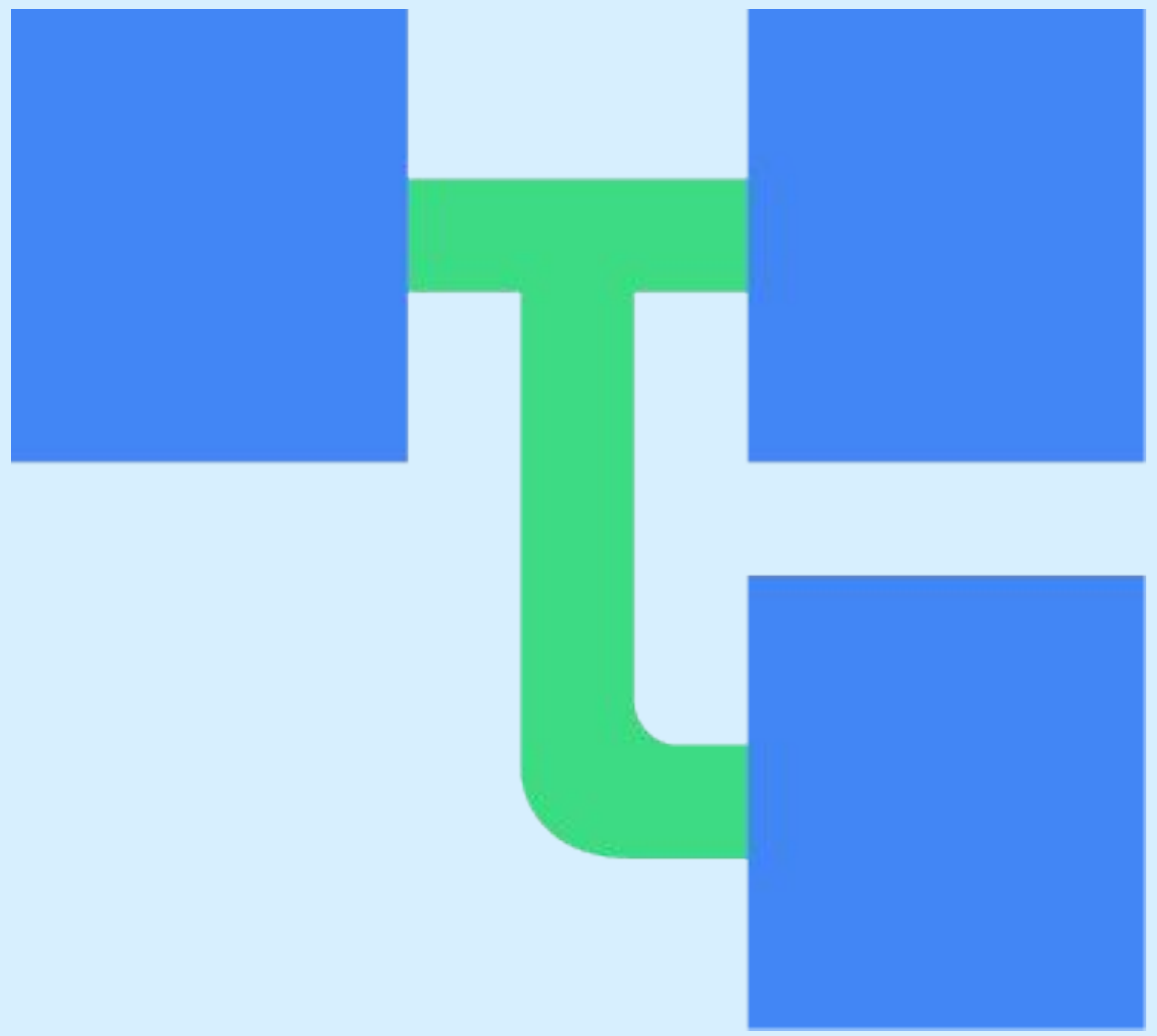
Concern:

If a device (Eg: PMIC) provides multiple resources (Eg: regulators, gpios, etc) and at least one of its consumers doesn’t probe, then all the regulators <u>left on by the bootloader</u>:

- Cannot be cleaned up.
- Some drivers use regulator_get() on these regulators, but expect exclusive access in some systems. They won’t have exclusive access till sync_state() is called.

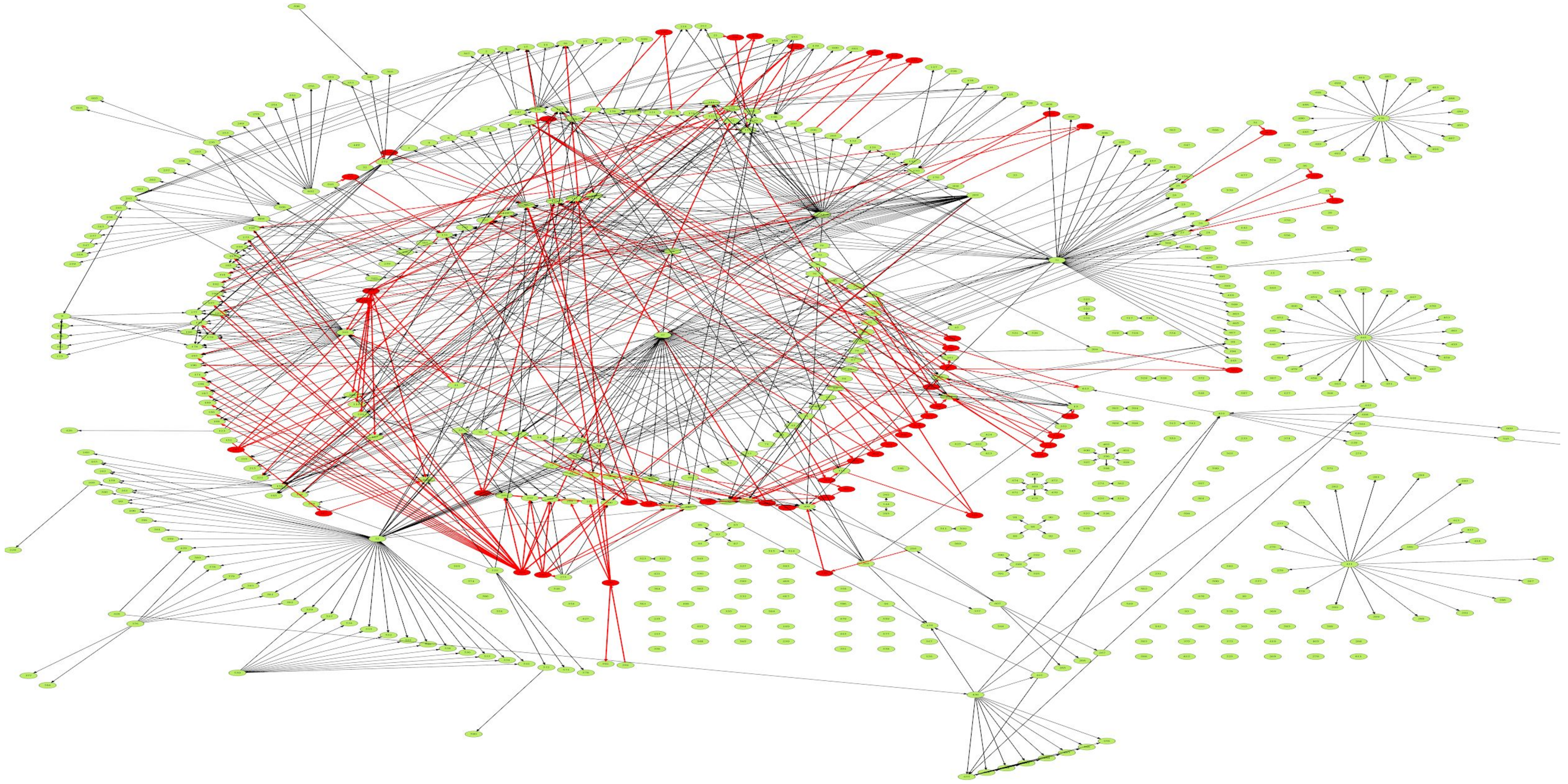
Thoughts:

- Resource level tracking.
- Kernel config/command line parameter to disable keeping resources on (Eg: regulator, clocks, etc) till sync_state().
- Command line timeout after which sync_state() is always called.
- Other options?



android

Mobile device (with fw_devlink + per-regulator links)



Note: Does NOT include “class” devices because they never bind to drivers

android

fw_devlink=on by default

Why?

- Can significantly reduce deferred probing in its current state
- No more _initcall chicken/Makefile ordering
- Getting to topological probe ordering seems feasible.
- Significantly simplifies module load ordering
- In the long run, it could allow simplifying deferred probe handling in drivers.

android

fw_devlink=on by default

Problems

fw_devlink=on can block probing for a few corner cases

- Cycles in DT which can't be broken with logic.
- Devices with "compatible" property with drivers that parse and "probe" without using a struct device.
  - Not talking about early devices like GIC/Timer.

android

fw_devlink=on by default

Thoughts?

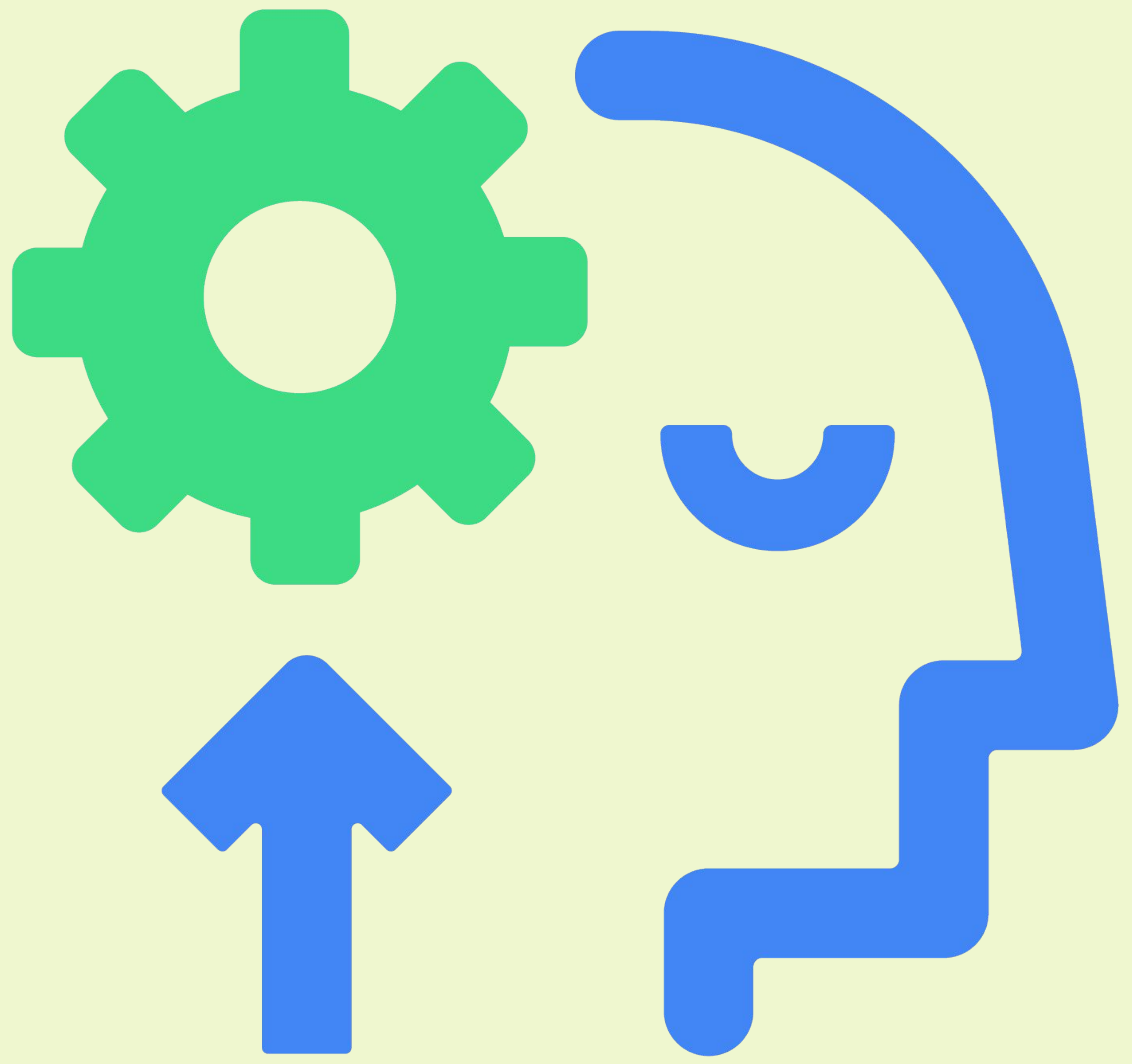
Need to be backward compatible with existing DT.

Therefore:

- Stop blocking probing if !CONFIG_MODULES && deferred probe workqueue is done.
- If CONFIG_MODULES, use timeout command line param set to 30s by default.

Platforms without this corner case/Future platforms can still do full enforcement:

- Disable timeout (timeout=-1).
- Disambiguate cycles in DT.
- Fix drivers to use device driver core.



android

Disambiguate cyclic-dependency in DT

Cycle uncertainty

```
sdhci: sdhci@xxxx{
  compatible = "acme,sdhci";
  ...
  phys = <&emmc_phy>;
}

emmc_phy: phy@yyyy {
  compatible = "acme,emmc-phy";
  ...
  clocks = <&sdhci>;
}
```

No cycle uncertainty

```
sdhci: sdhci@xxxx{
  compatible = "acme,sdhci";
  ...
  phys = <&emmc_phy>;
}

emmc_phy: phy@yyyy {
  compatible = "acme,emmc-phy";
  ...
  init_optional { // Name has no meaning here
    clocks = <&sdhci>;
  }
}
```

android

Thank you!