Scheduler-driven CPU Frequency Selection
Or, Rewriting Parts of CPUfreq

Michael Turquette <mturquette@baylibre.com>

Concept

The scheduler is the right place to choose a CPU frequency target

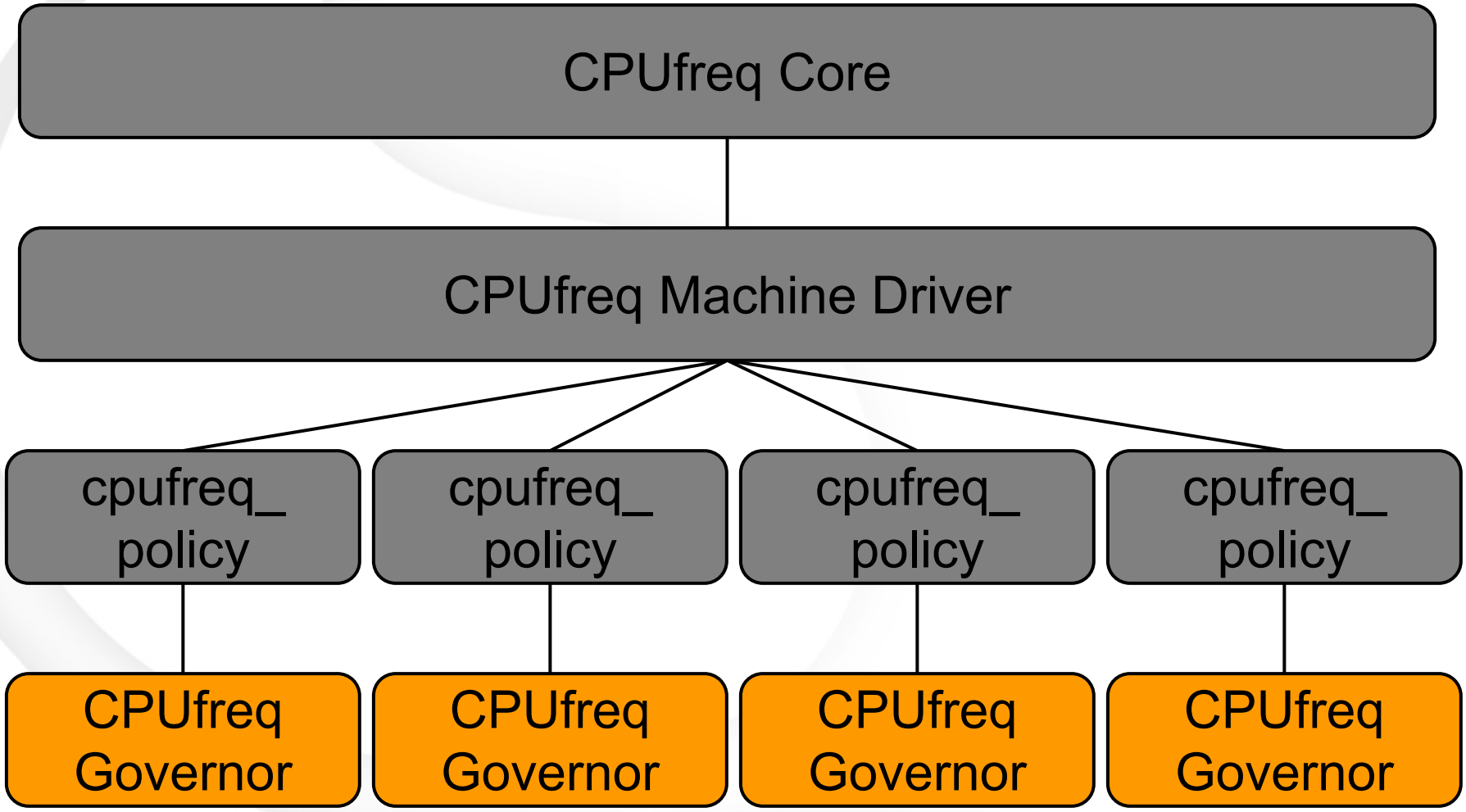
- Has the most relevant information
- Can be coordinated with interrelated decisions
  - Task placement
  - Idle state selection
- Legacy governors will stick around for as long as people want

History

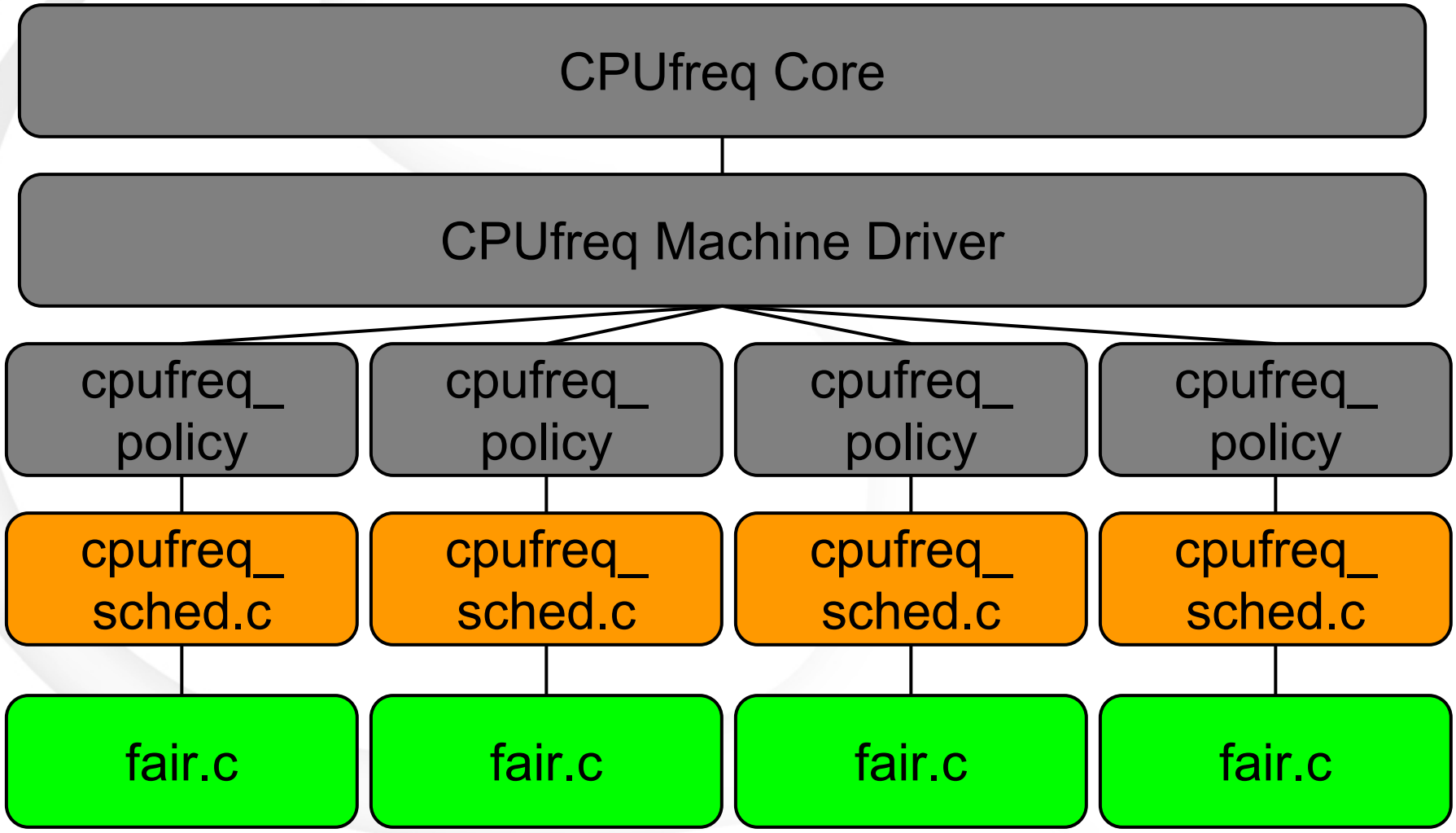
Four RFCs, 2014 to present

- http://marc.info/?l=linux-kernel&m=141395808301422
- http://marc.info/?l=linux-kernel&m=143077749912612
- http://marc.info/?l=linux-kernel&m=143139687308197
- http://marc.info/?l=linux-kernel&m=143536286305669

Legacy CPUfreq architecture



cpufreq_sched.c



cpufreq_sched.c

kernel/sched/cpufreq_sched.c

- CPUfreq governor
- Shim layer
- Implements no CPU frequency selection policy
- Exposes API to fair.c for setting CPU frequency
  - Additional abstraction layer/indirection
- Event-driven, not a poll/loop

Problems

Locking sucks

- http://marc.info/?l=linux-kernel&m=143966846703034

Layering is weird

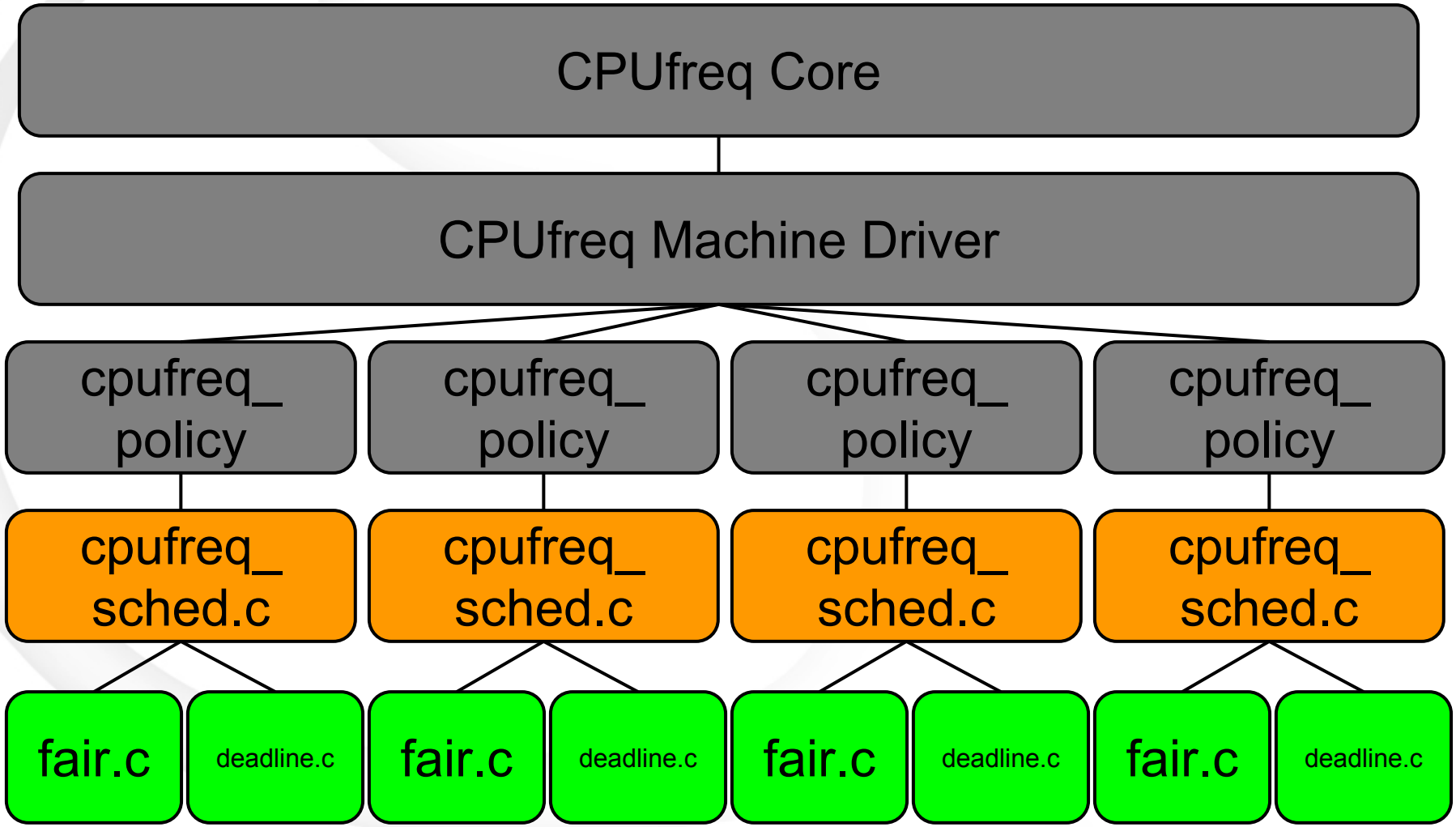
- Governor knows the CPU topology/frequency domain, but fair.c does not
  - Picks frequency based on CPU with max capacity request in the policy
- Better that CFS knows which CPUs belong in the frequency domain and move that code out of cpufreq_sched.c

Problems, 2

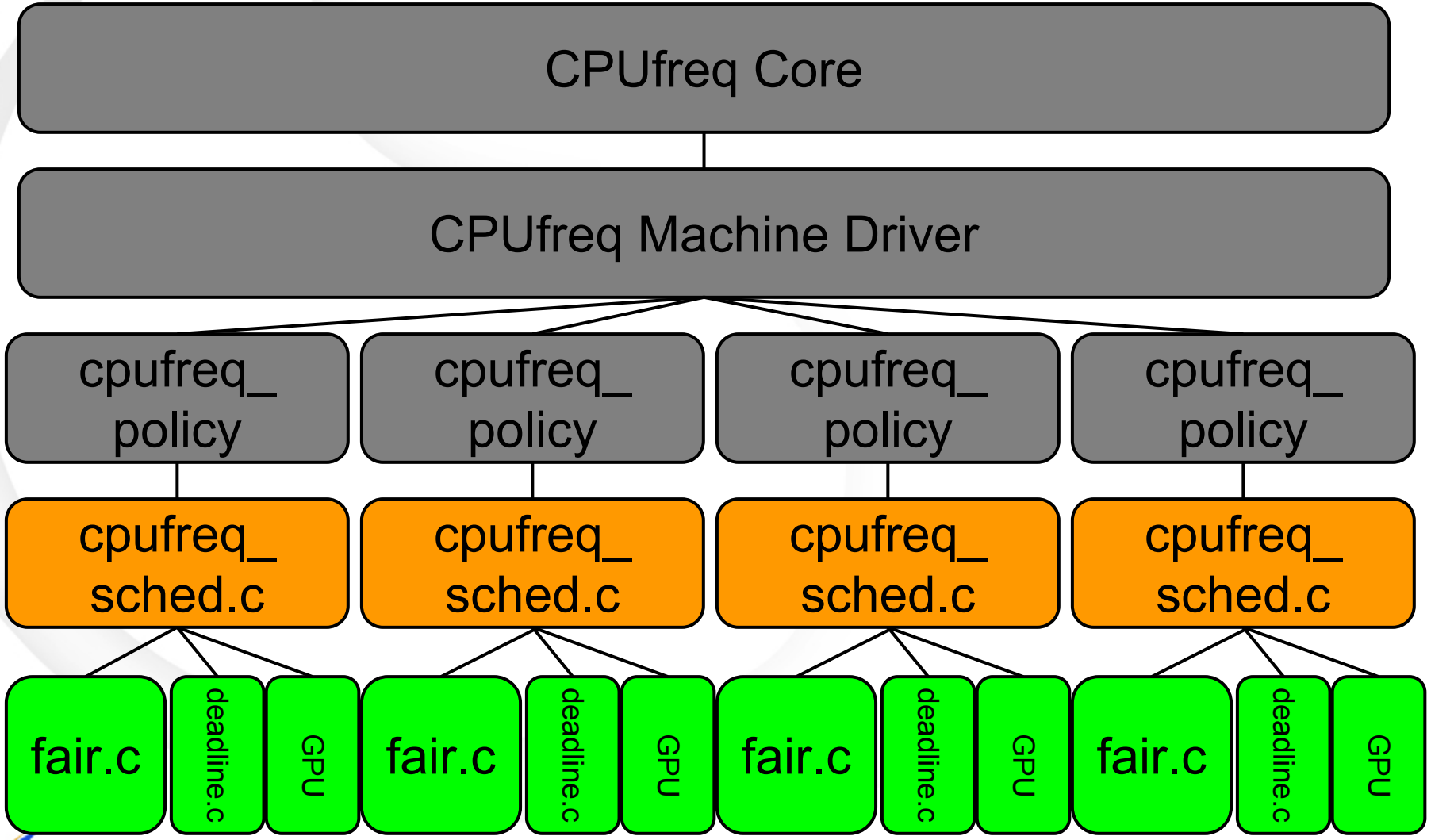
No async interface

- Governor does some weird stuff to schedule a frequency transition from within schedule() context
- See “Locking sucks”

Supporting multiple sched_class



Supporting peripheral devices



cpufreq_sched.c should go away?

- Shim layer used by multiple consumers
  - Probably should be absorbed into CPUfreq core
- struct cpufreq_policy should support multiple simultaneous consumers
  - Opaque cookie/handle provided by the core to each consumer
    - CFS, Deadline, Peripheral devices
- Arbitration and aggregation done by the core

The proposal