INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES

Scuola Superiore
Sant'Anna

Deadline scheduling: can your mobile device last longer?

Juri Lelli, Mario Bambagini, Giuseppe Lipari

Linux Plumbers Conference 2012
San Diego (CA), USA, August 31

TeCIP Insitute, Scuola Superiore Sant'Anna
Area della Ricerca CNR, Via G. Moruzzi 1
56127 Pisa, Italy

Retis
Real-Time Systems Laboratory

Outline

- Classical Real-Time concepts
- An implementation
- Power Aware Deadline Scheduling
- Conclusions & Discussion

Classical Real-Time concepts

What's Real-Time?

- All about “**deadlines**”
- Useful result → **correct output + before its deadline**
- “Go as fast as you can” is not always the best answer
- “**Go as fast as you need to respect deadlines**”

“But not faster! Or you will spend too much energy!”

Hard vs. Soft RT

- **Hard real-time** tasks: a missed deadline may cause **catastrophic consequences.**
- **Soft real-time** tasks: timing constraints are important, but not critical **(QoS degradation)**

Resource Reservation

- Concurrent real-time tasks compete for resources (CPU time)
- **Resource Reservation** mechanism
- A task is allowed to execute for:
  - Q time units *(budget)*
  - in every interval of length P *(reservation period)*
- Allocated bandwidth is U=Q/P (a.k.a. **utilization**)

Temporal Isolation

- nothing new ?
  - CFS bandwidth control
  - RT throttling
- **Reservations**, like containers
- Each reservation has its own **dynamic deadline** (Constant Bandwidth Server)
- Reservations are scheduled by Earliest Deadline First

Temporal Isolation

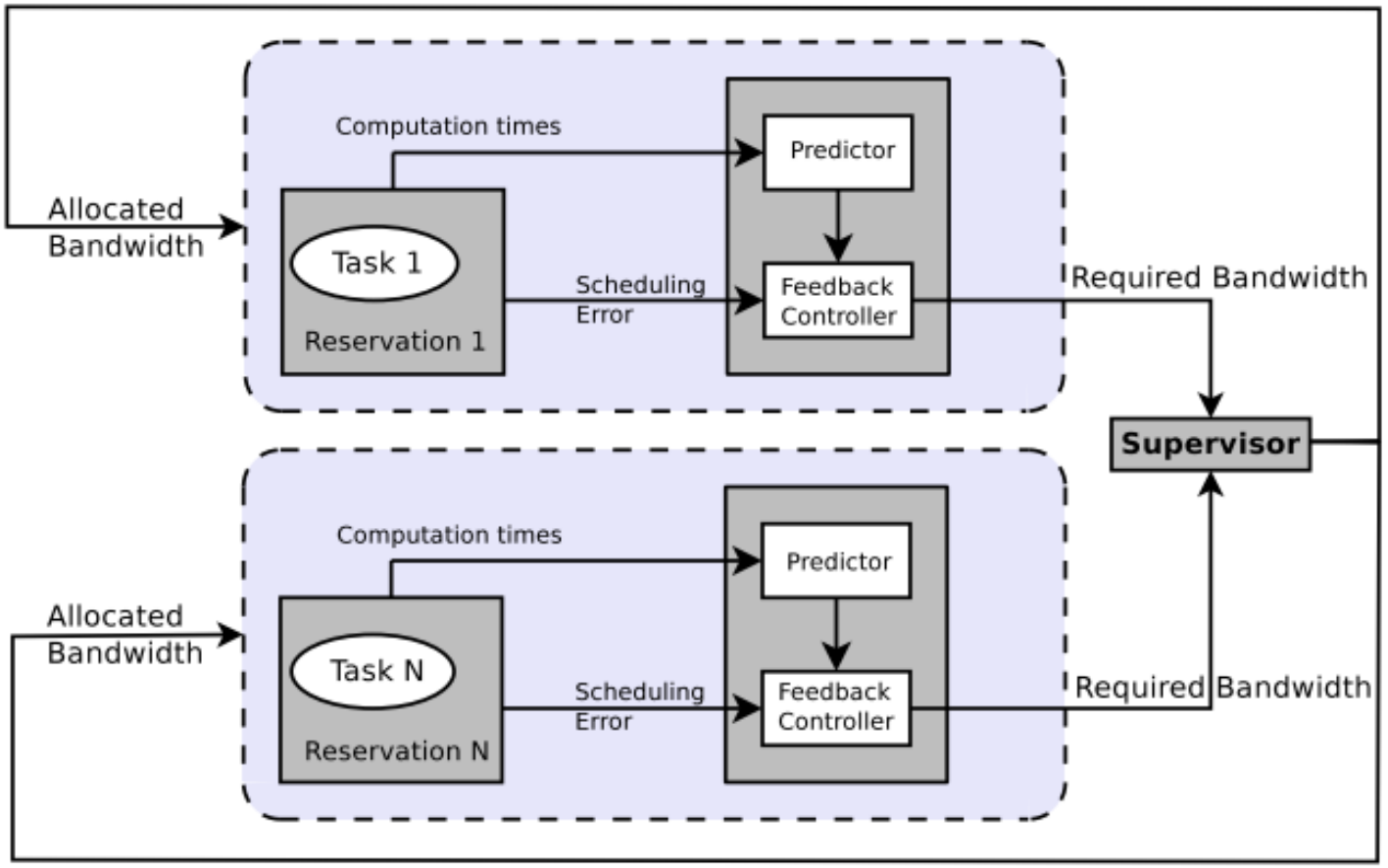
- EDF gives higher priority to more urgent tasks
- CBS slows down (or throttles) misbehaving tasks
- EDF + CBS ensures **temporal isolation**
- Admission control provides timing guarantees

$$\sum U_i \leq M$$

Choosing parameters

- **Period** is easy: P = T or P = NT
- **Budget** assignment is painful :'-(
- Hard real-time → WCET
  - calculated with tools from source code
  - estimated via benchmarks
- Soft real-time
  - measure computation time and adapt budget
  - *adaptive reservations*

Adaptive/Feedback Scheduling



Mobile Data

- Not only Multimedia
  - not strictly periodic
  - no explicit deadline
- how to cope with other data sources (e.g., Web browsing, emails, user interaction, etc.) ?
  - pack similar sources on container reservations
  - best effort (we don't care)

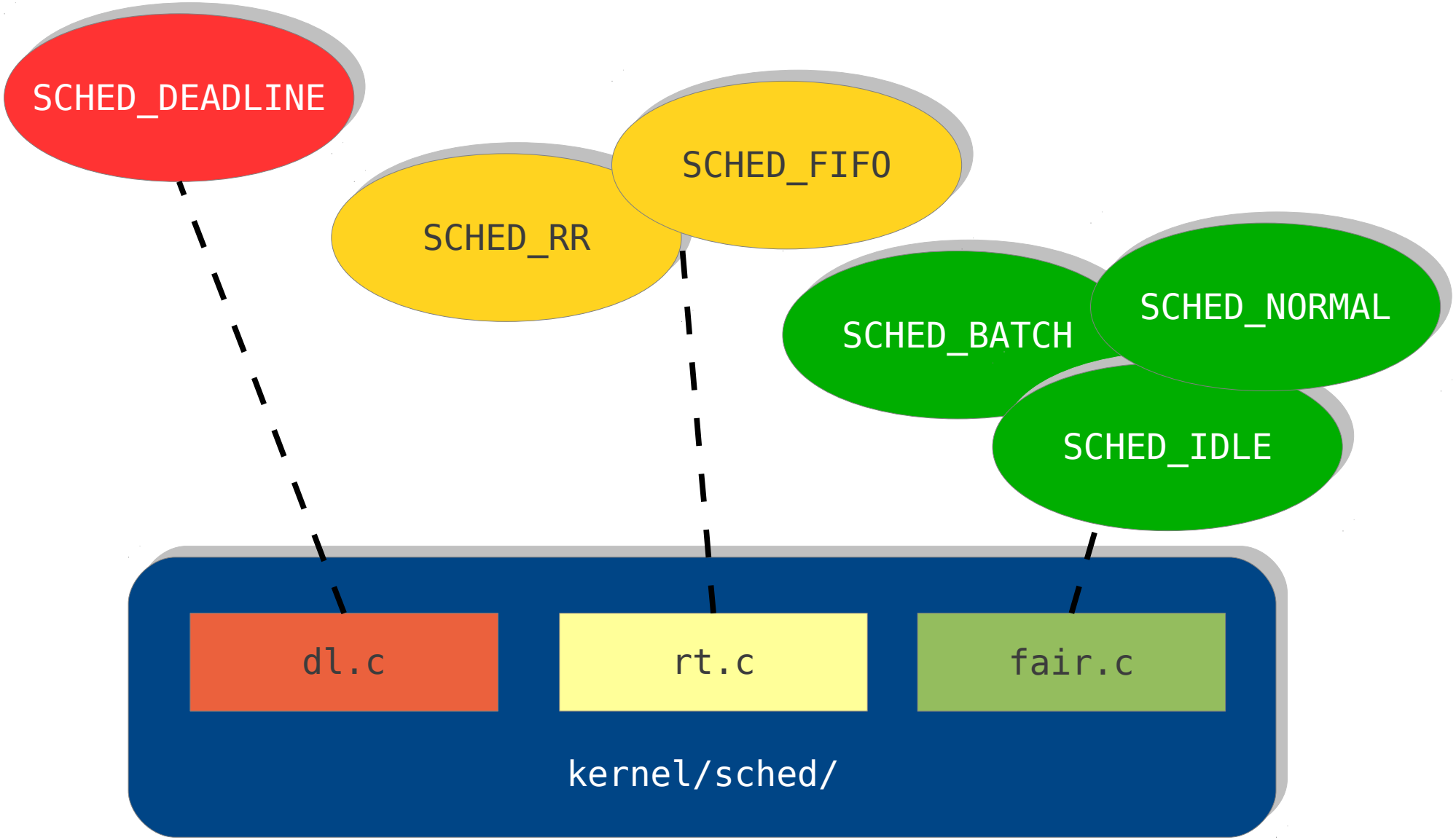
An implementation

What we already have

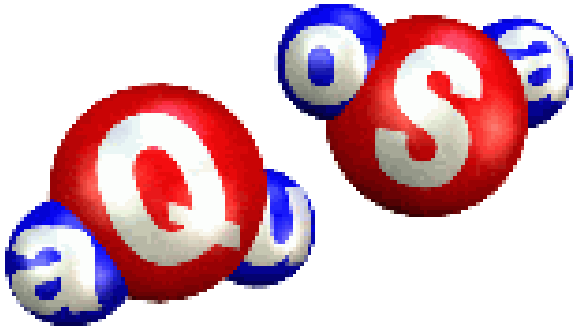
SCHED_DEADLINE

- deadline based real-time task scheduling policy
- bandwidth isolation
- each task runs inside its own reservation
- global/clustered multiprocessor scheduling through dynamic task migration

What we already have



What we already “have”



"Adaptive Quality of Service Architecture"
(for the Linux kernel)

- adaptive resource reservation layer
- dynamically adapting CPU reservation
- QoS aware applications API
- a bit outdated & UP only :-(

Power Aware Deadline Scheduling

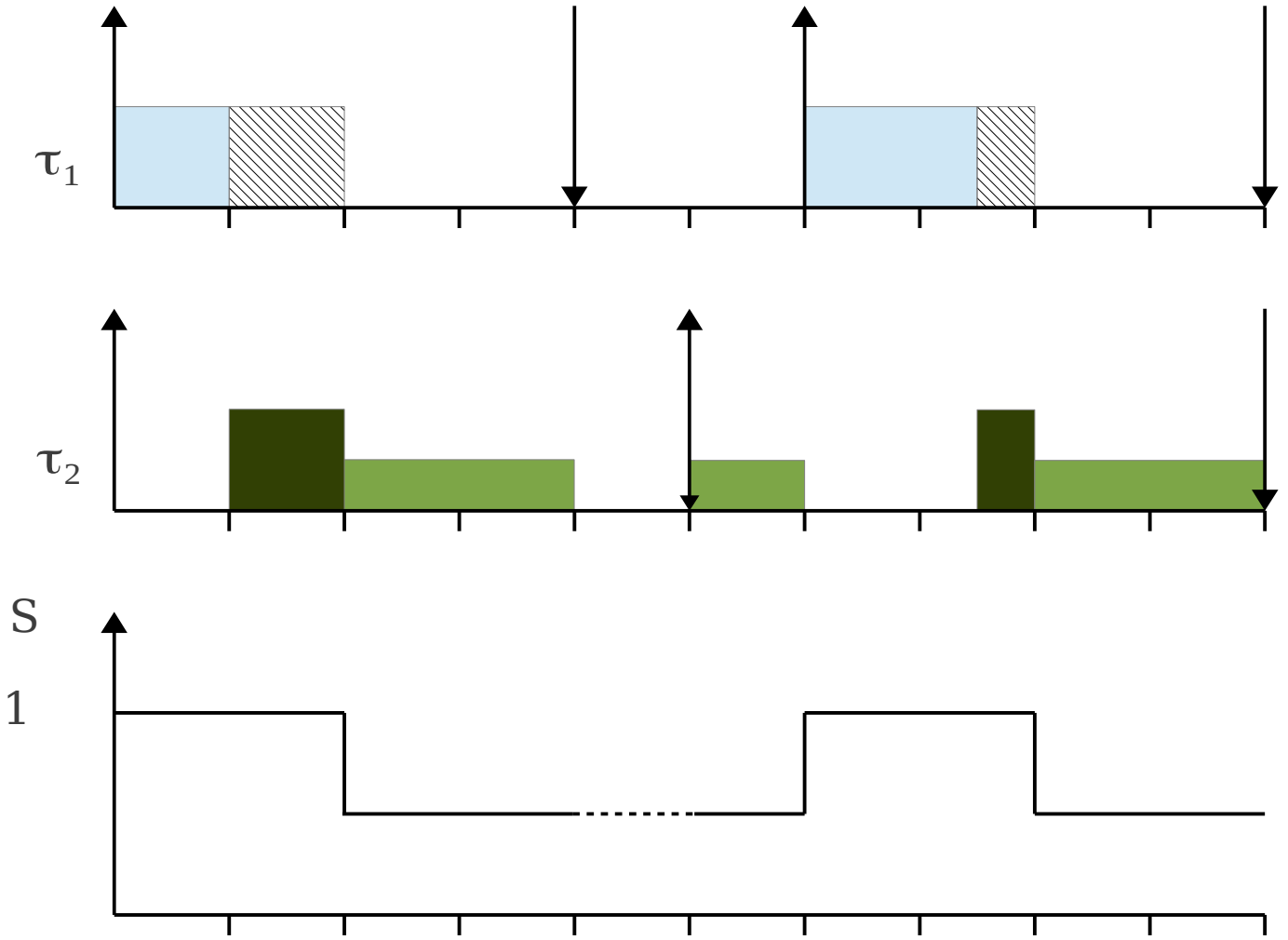
Power aware RT scheduling

- Let's bring power efficiency into the picture...
- Hardware context (simplified)
  - homogeneous / heterogenous
  - single / multiple clock domains
  - discrete operating frequencies
  - transitories to adjust to a new frequency
  - overhead for frequency scaling
  - overhead for switching a core either ON or OFF

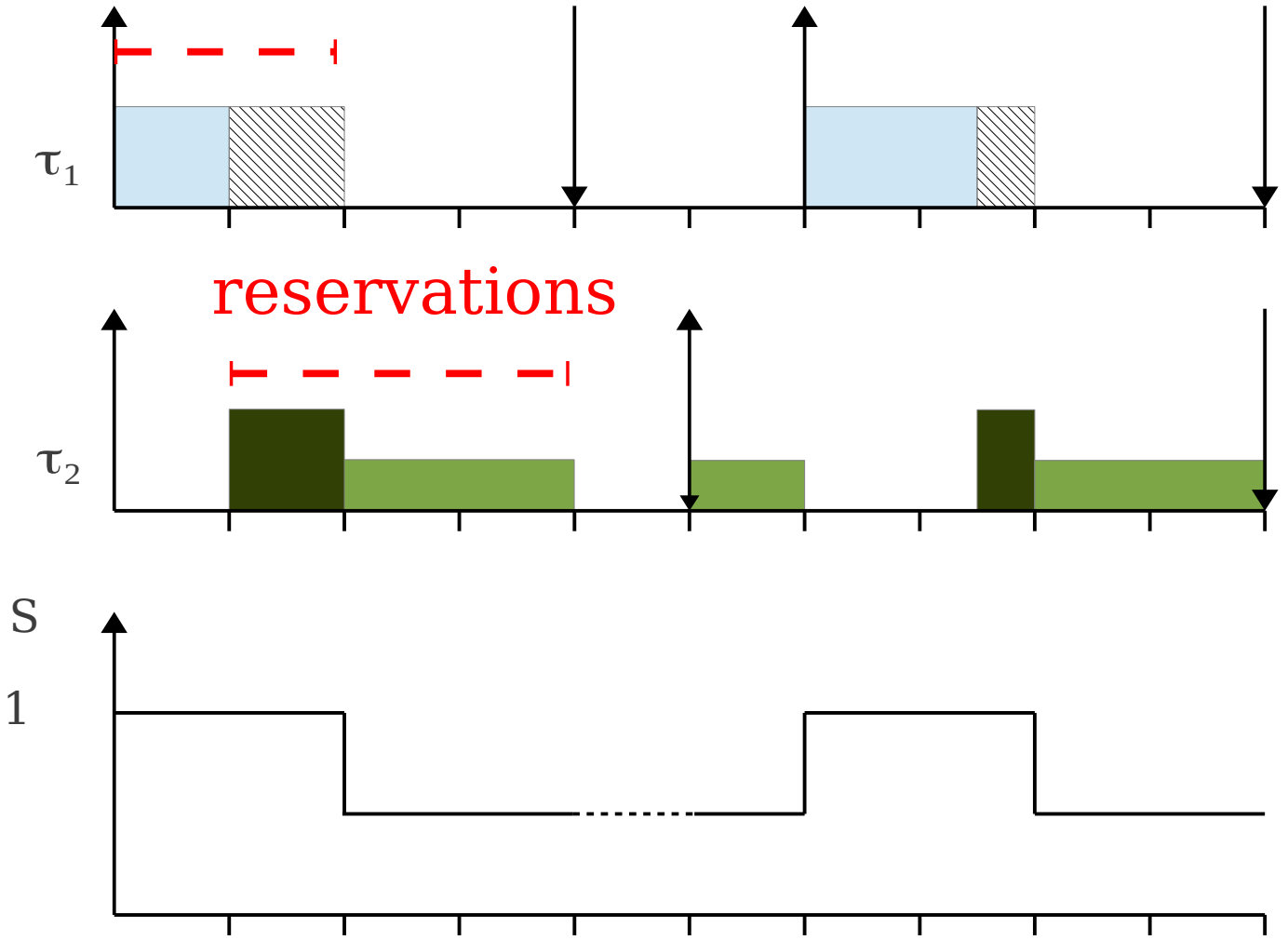
“Ideal” UP example

- Associate a *reservation* to each task
- Track actual CPU utilization (active servers contribute) → U
- Dynamically change CPU speed S to follow U
- Admission Control → don't go below U if you don't want deadline misses
- Power efficiency → slow down as much closer to U as you can to save power

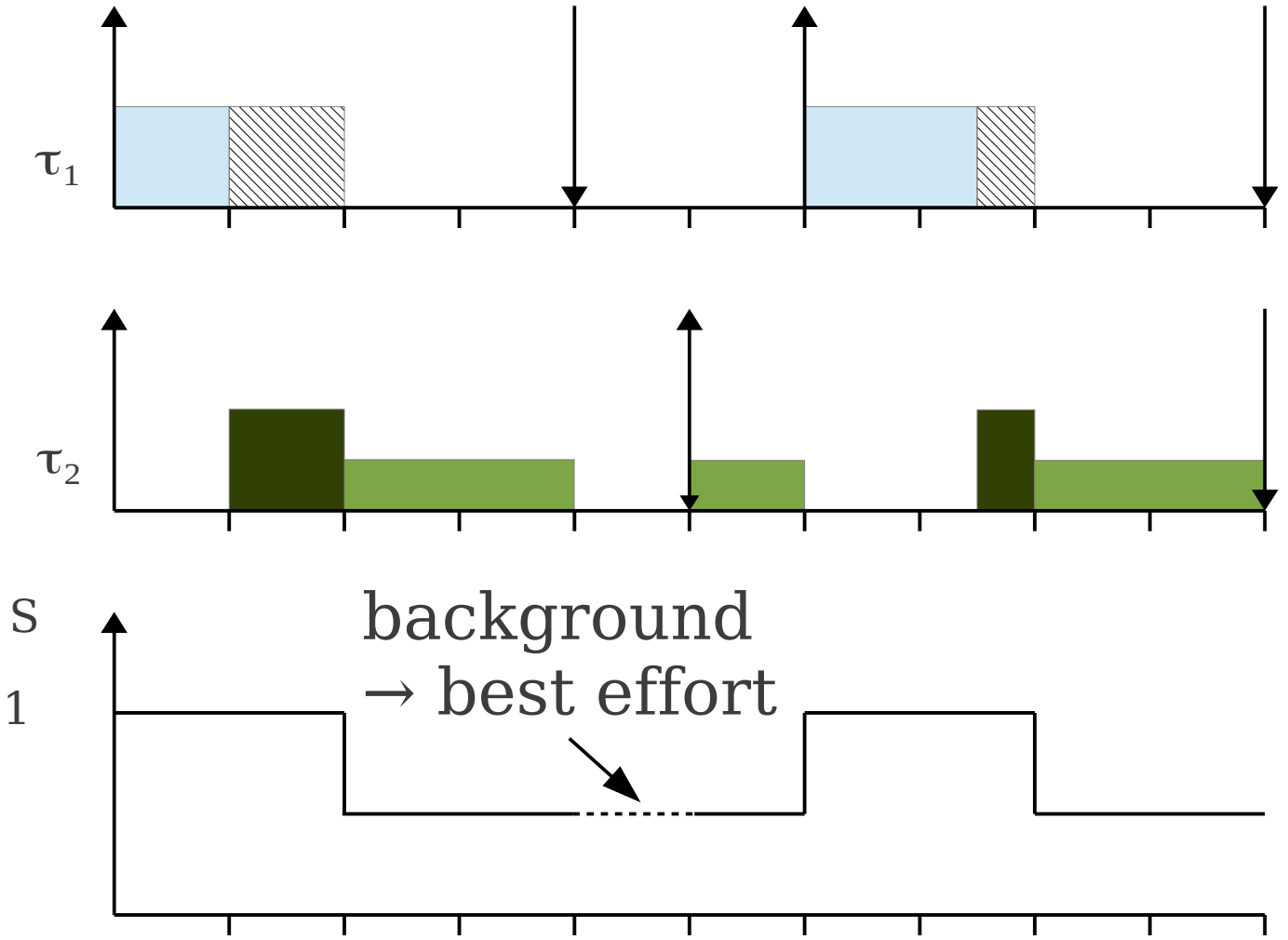
“Ideal” UP example



“Ideal” UP example



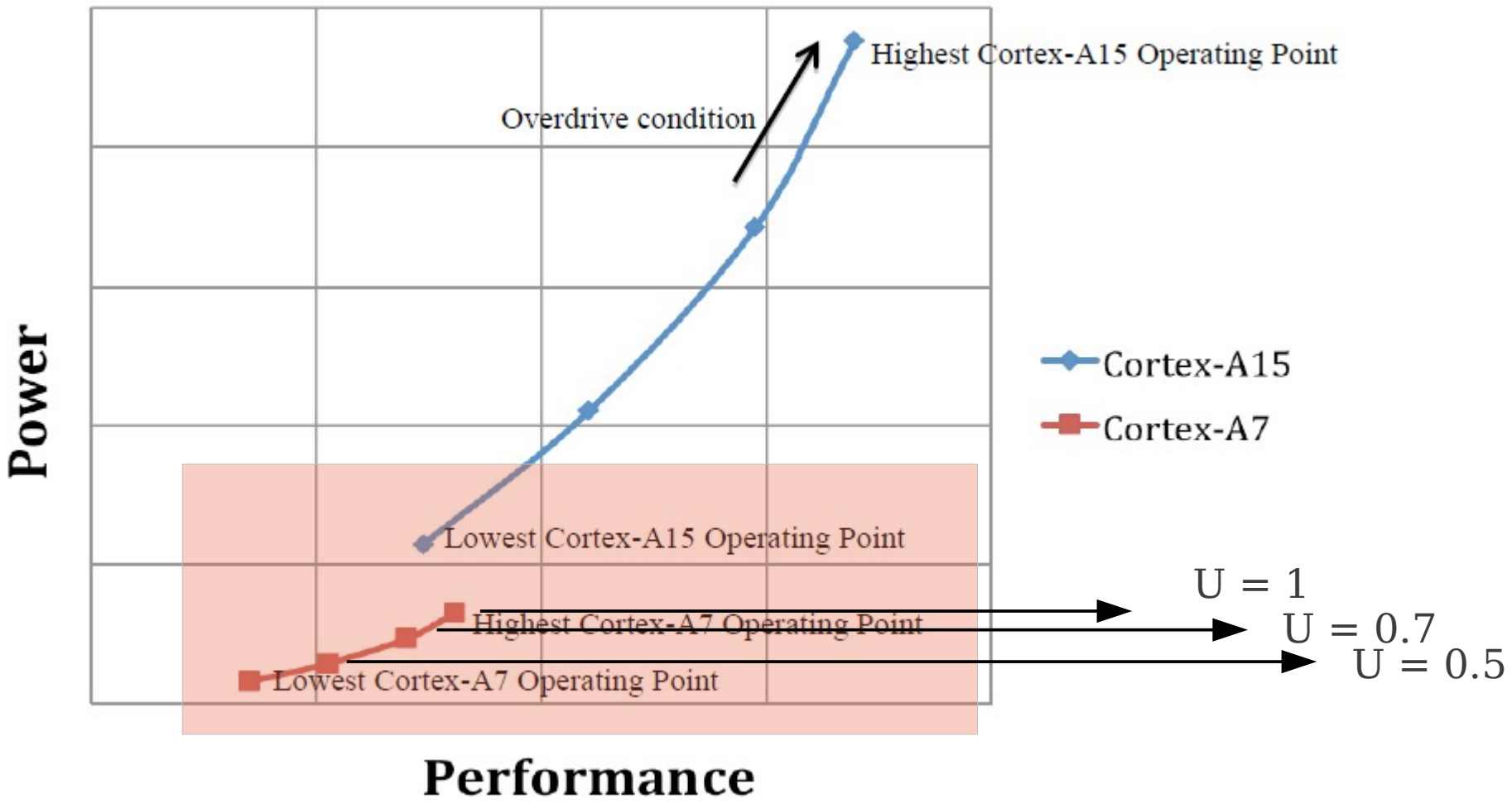
“Ideal” UP example



UP with “thresholds”

- No existing processor can vary its frequency with continuity
- set some “thresholds” on the values of the total system bandwidth
- CPU with N different frequencies
- compute N different bandwidth thresholds
- use them to know when a CPU speed change is needed, for example...

UP with “thresholds”



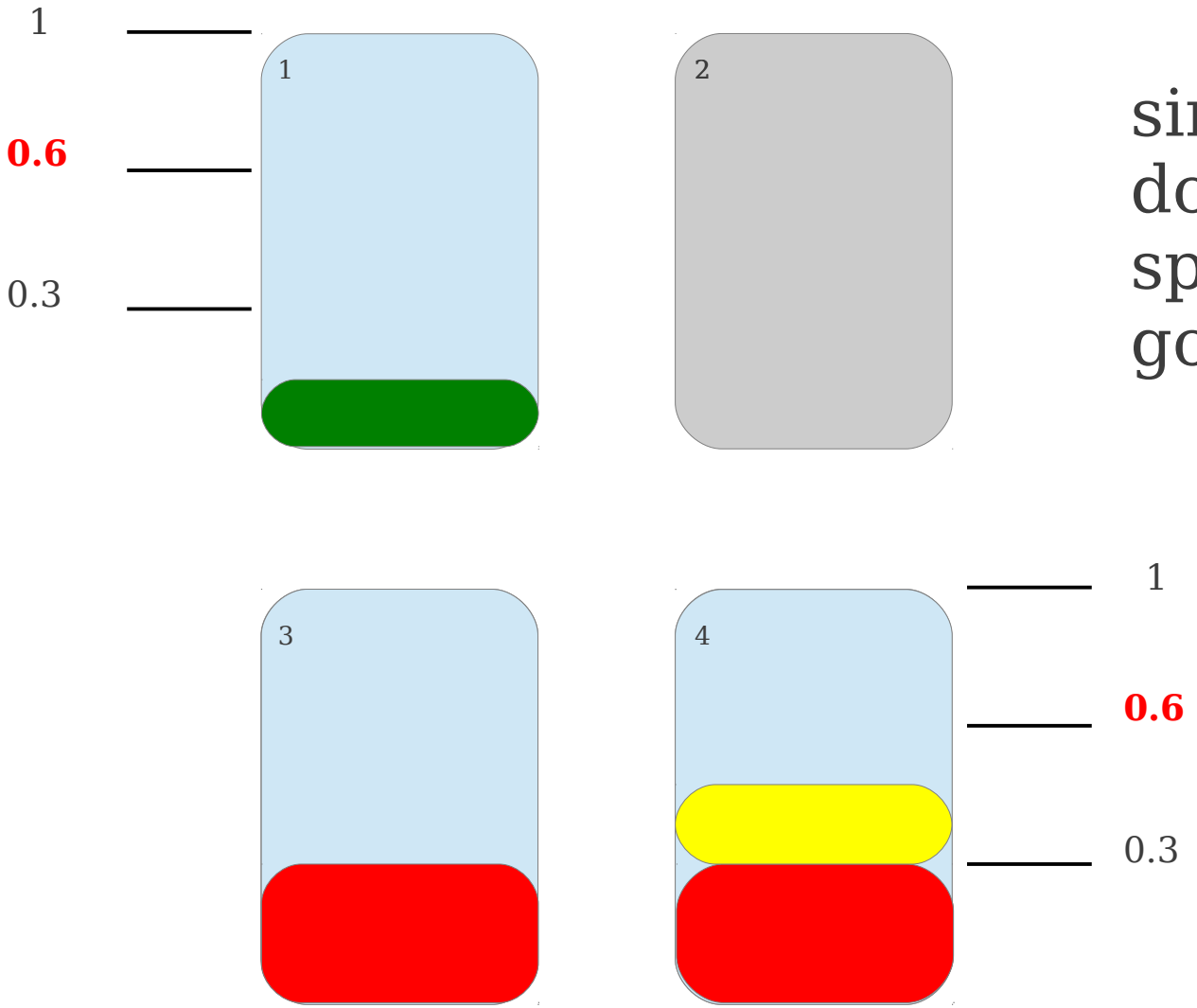
Going SMP

- SCHED_DEADLINE performs active “load-balancing” → Global-EDF
  - on a M-CPUs system the M earliest deadline ready tasks are always running
  - achieved through **push/pull** migrations (as in rt) – **deadline based decisions**

Power aware push/pull

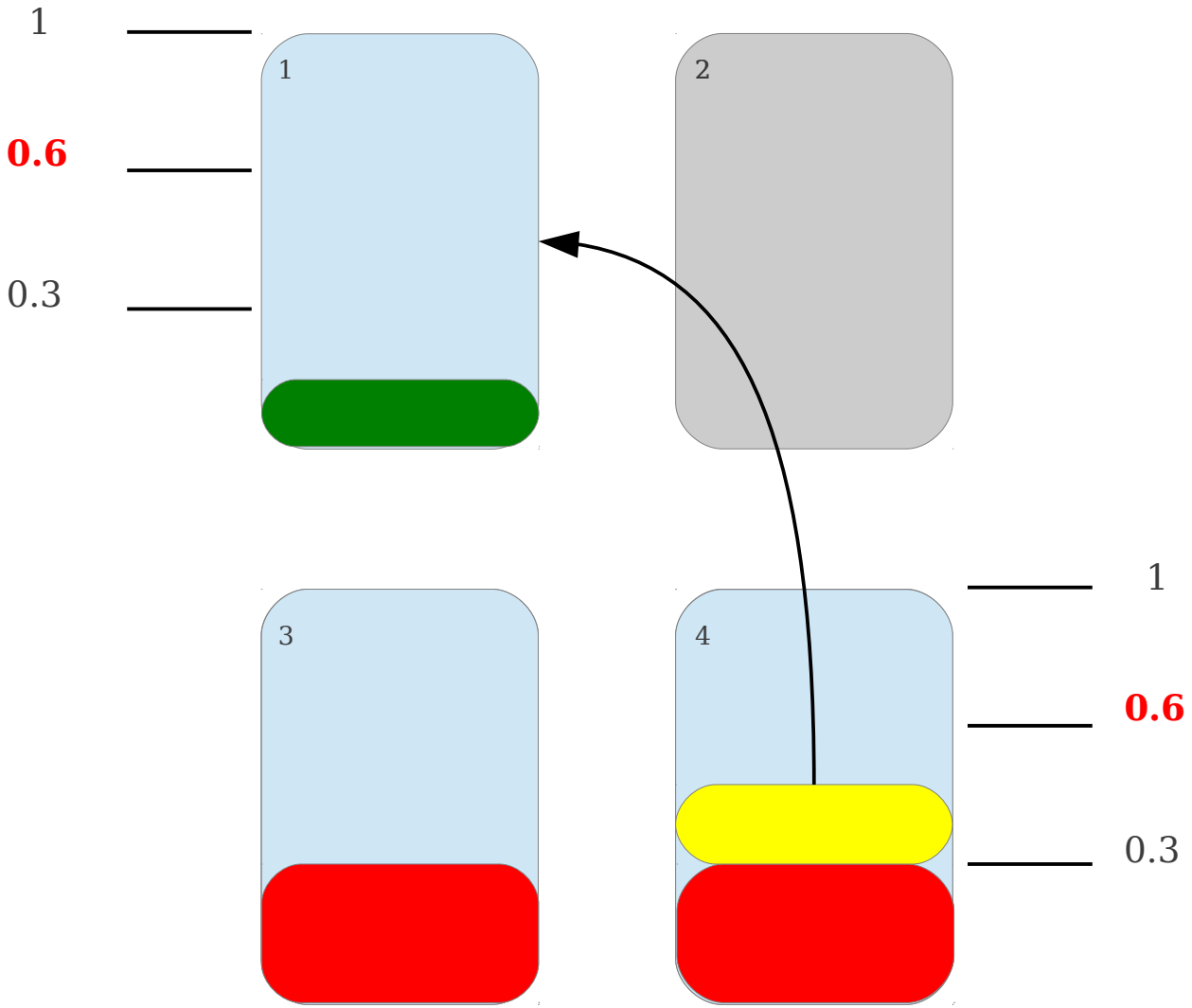
- “hybrid” scheduling
  - **EDF on each CPU**
  - **U driven migrations**
- different policies
  - single clock domain → minimize mean U
  - multiple clock domains → compact tasks to turn off CPUs as long as possible
  - big.LITTLE → compact on little core, move only big tasks on big cores ?

PA push/pull – an example

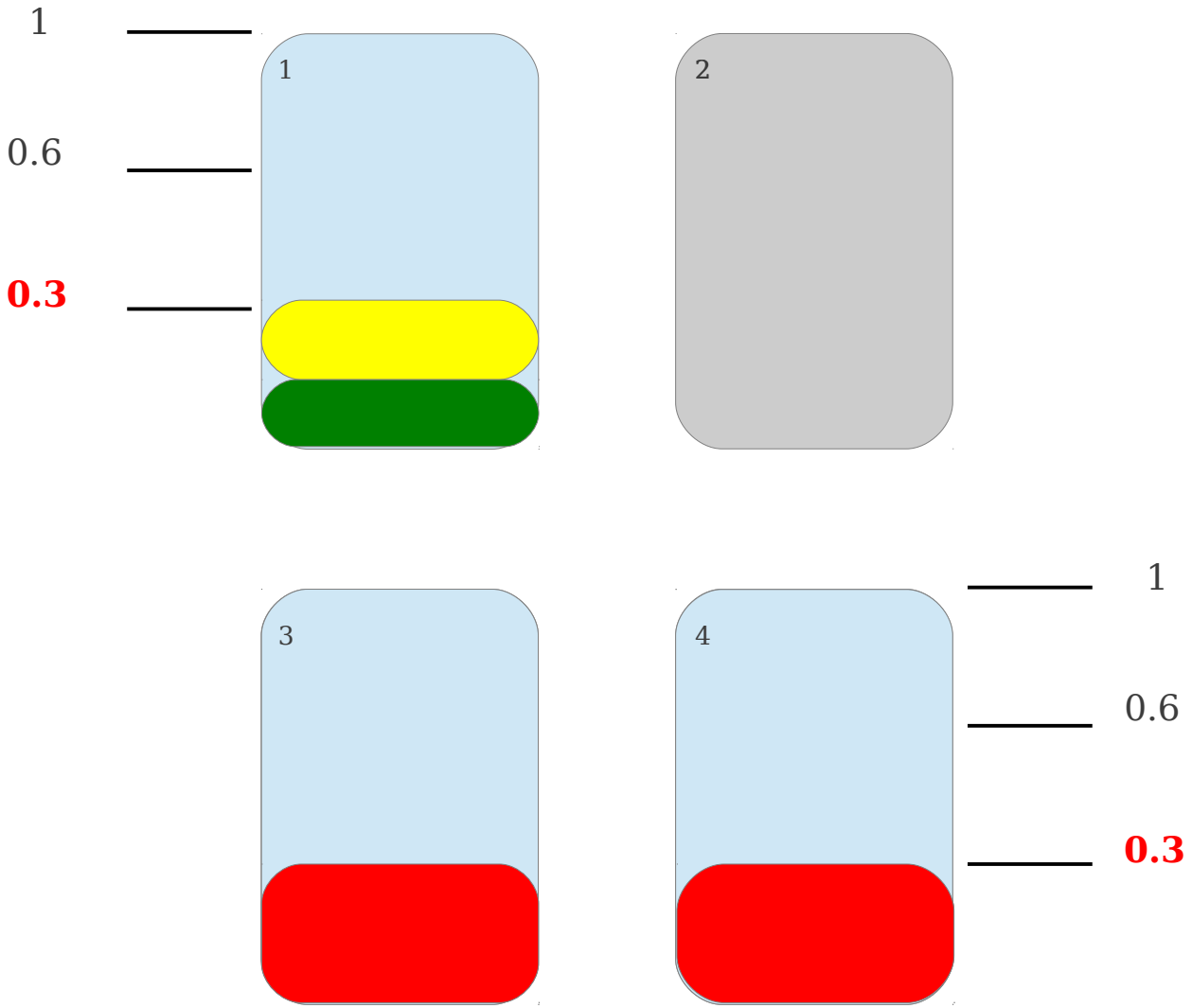


single clock domain → spread load to go slower

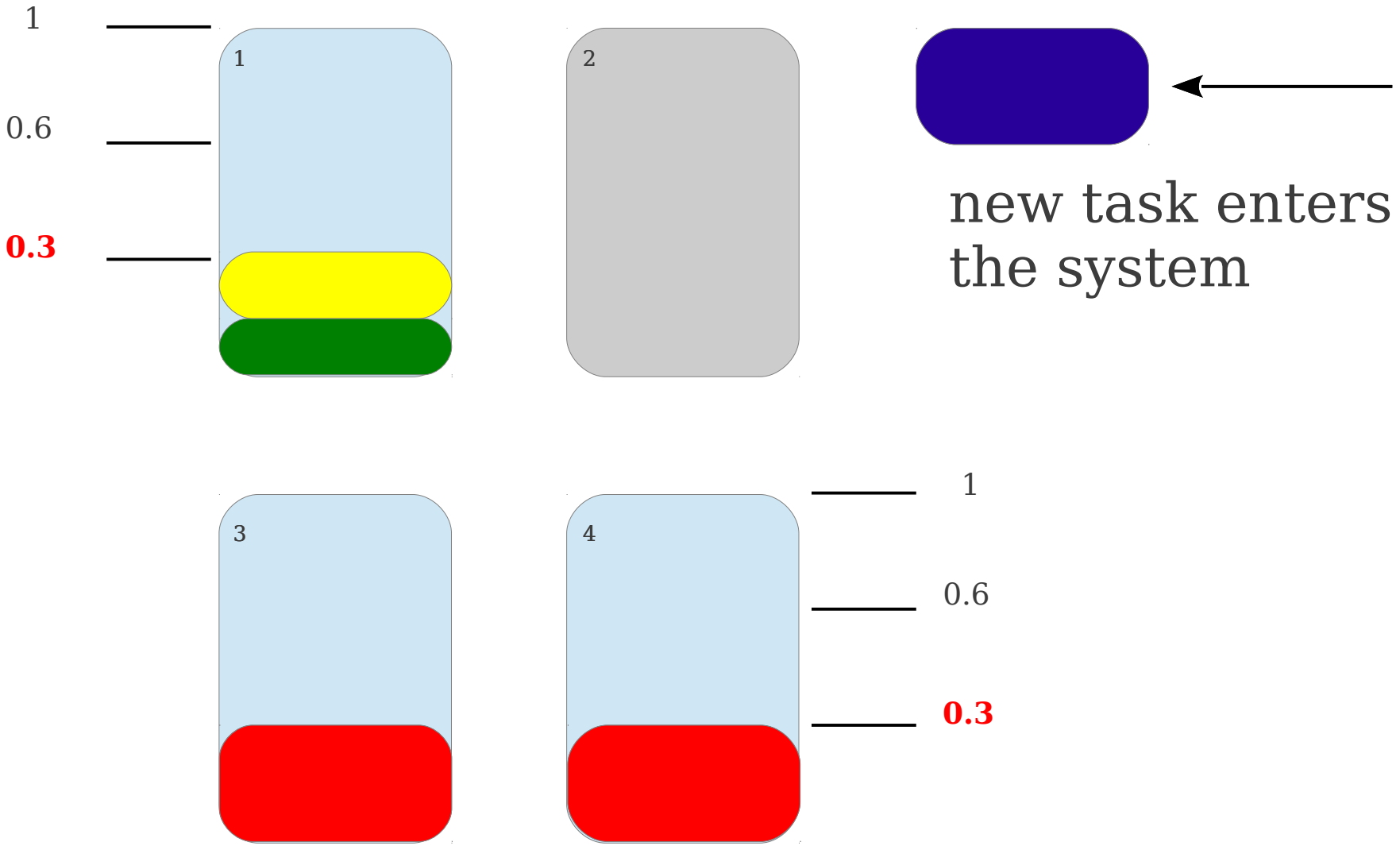
PA push/pull – an example



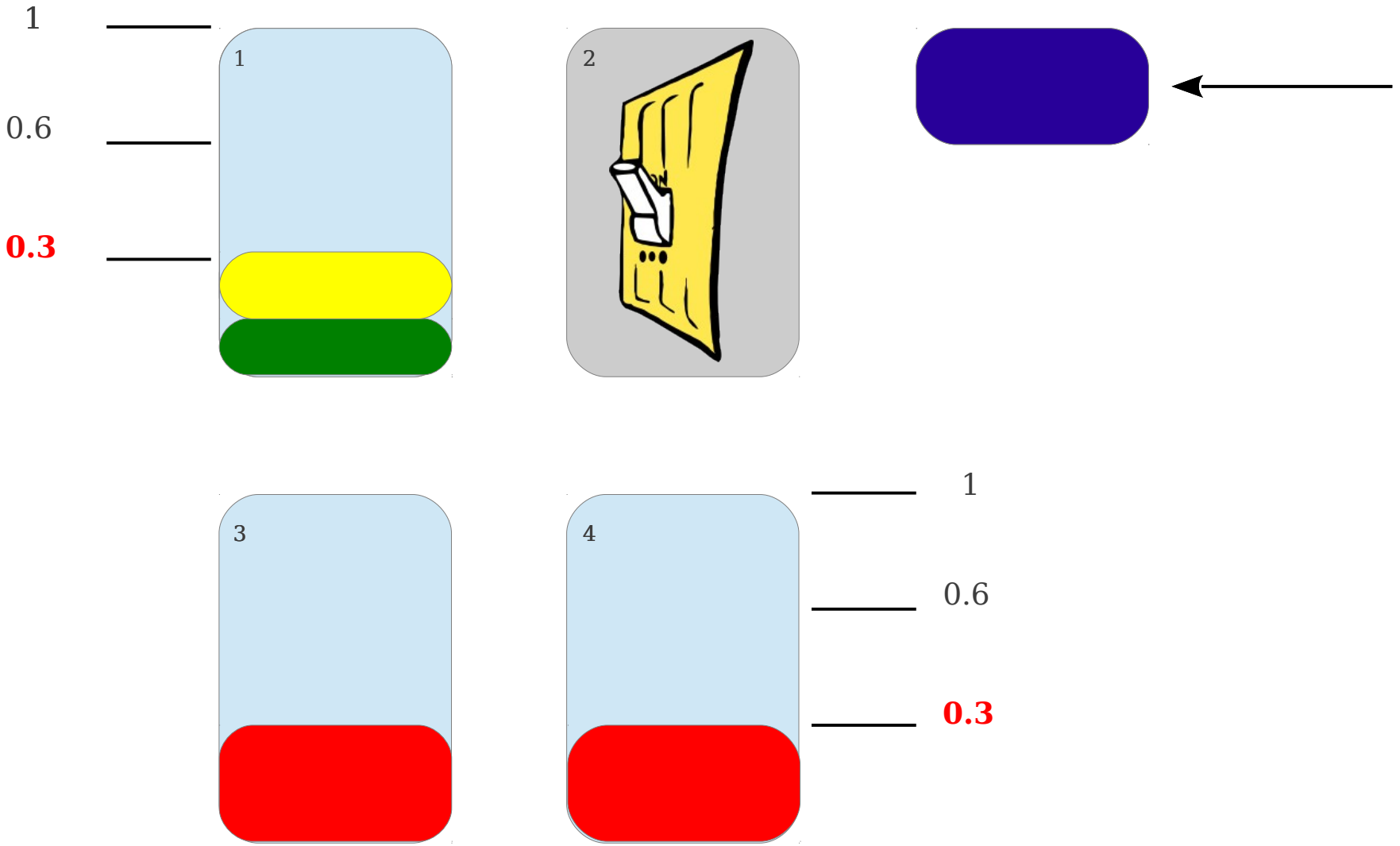
PA push/pull – an example



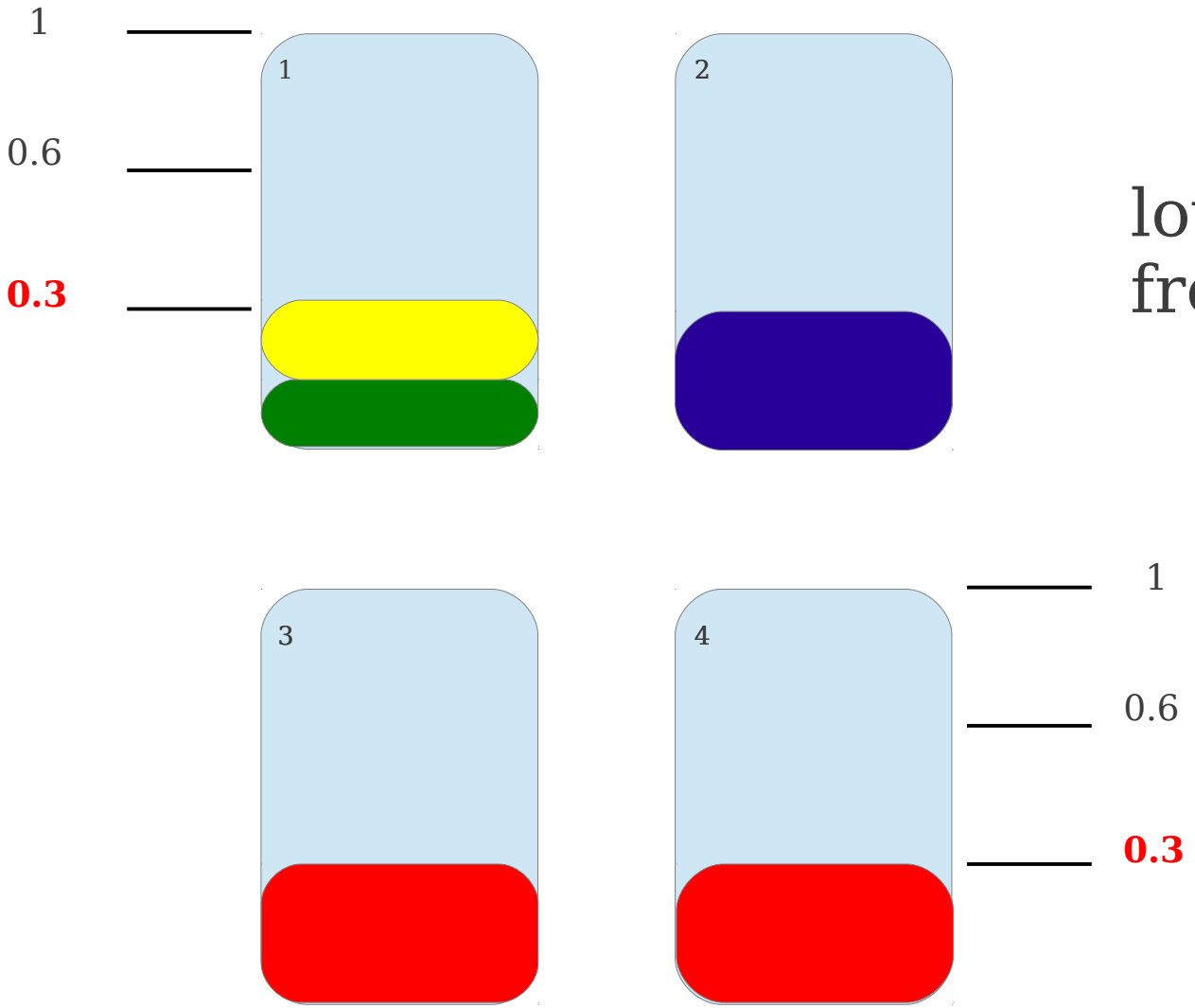
PA push/pull – an example



PA push/pull – an example

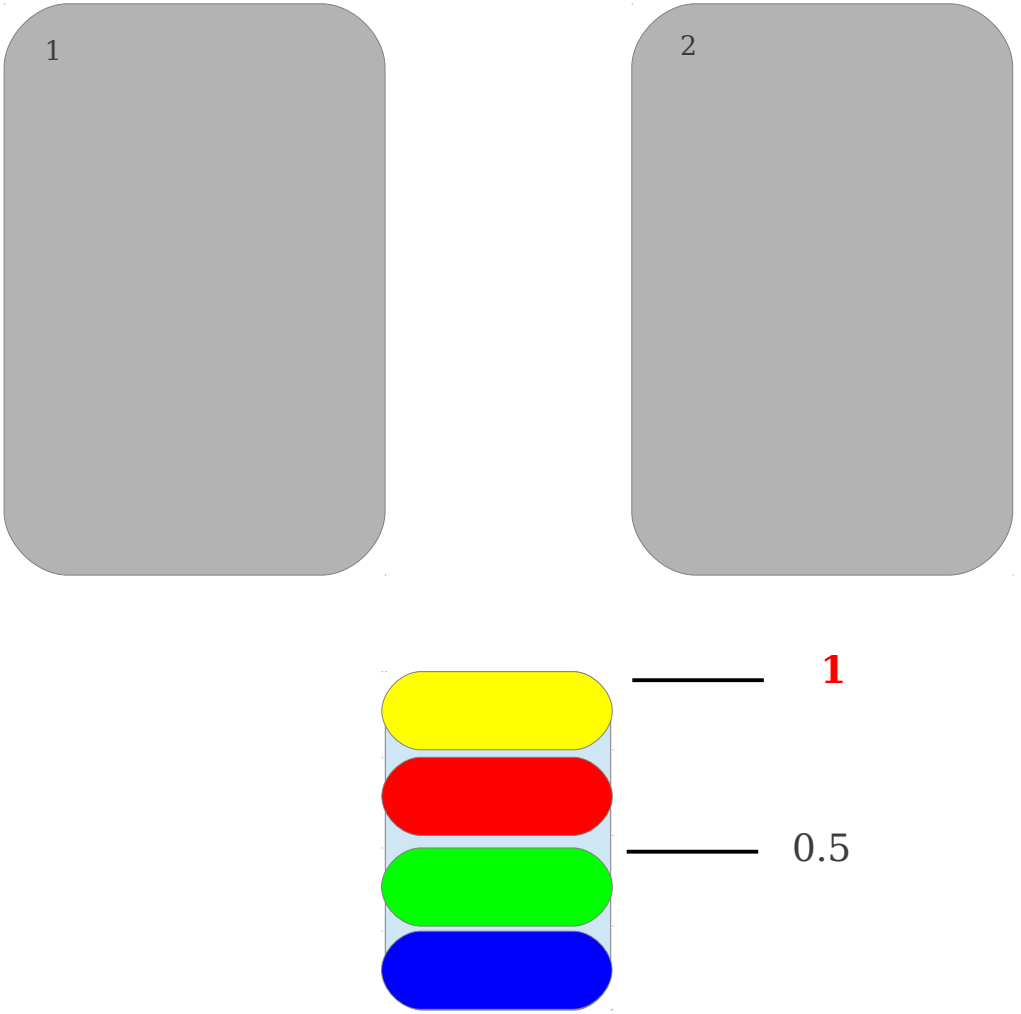


PA push/pull – an example

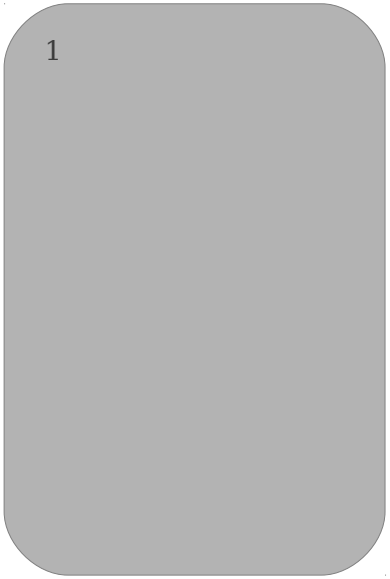


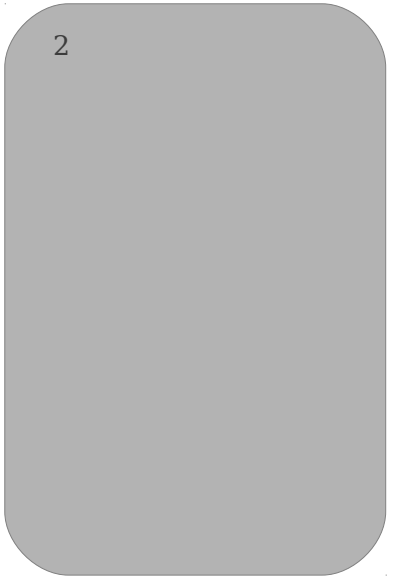
lower mean frequency

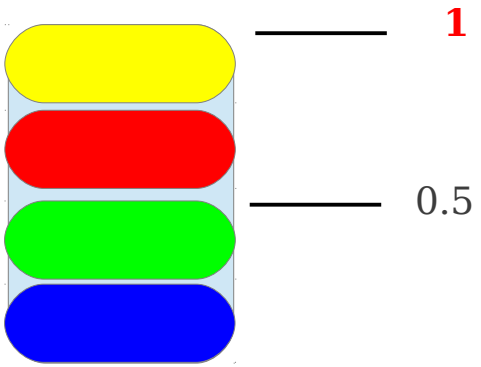
PA push/pull – Tegra like

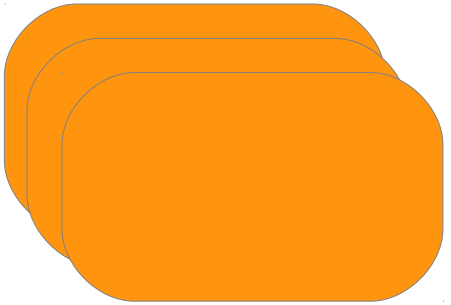


PA push/pull – Tegra like



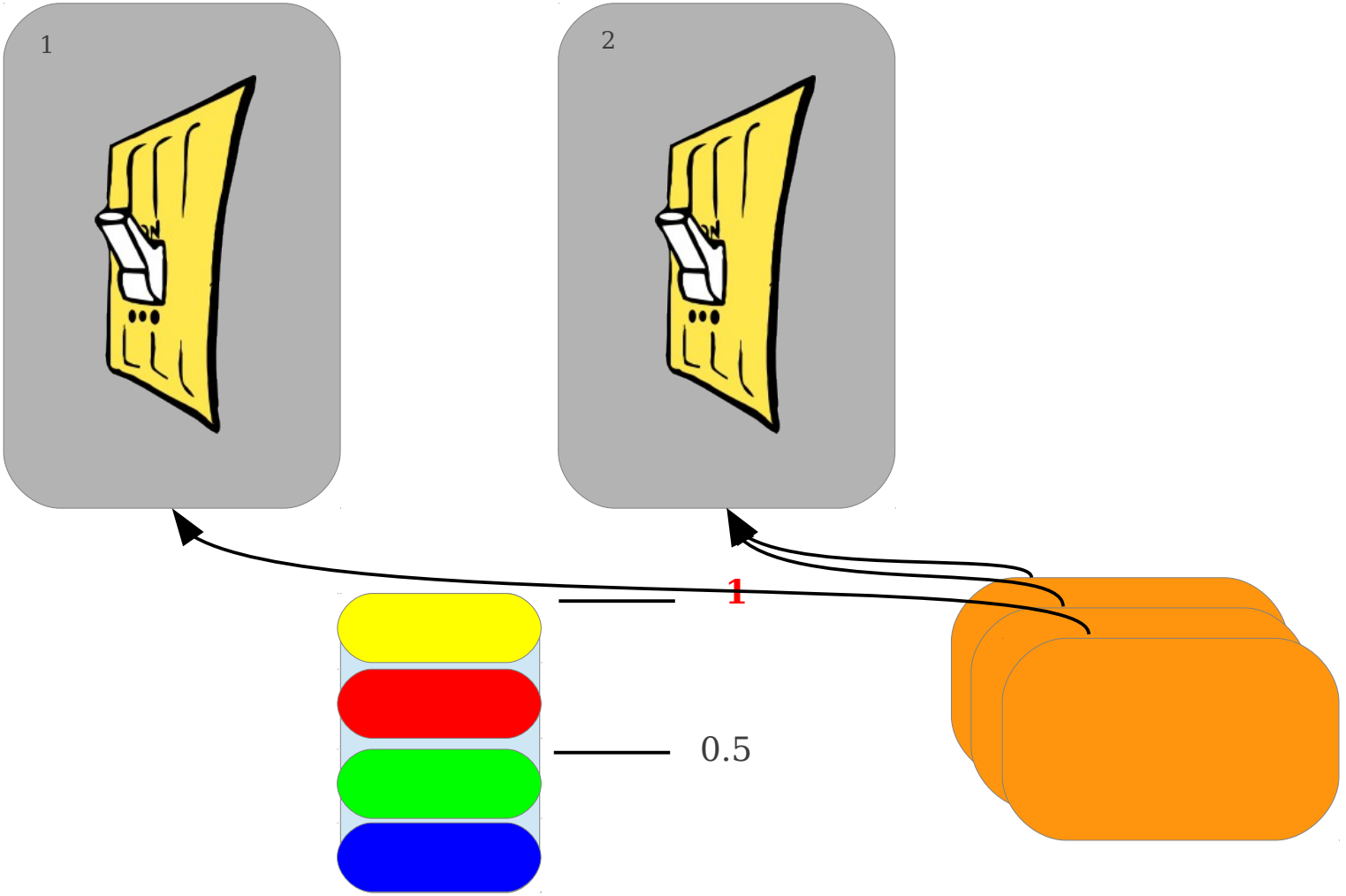




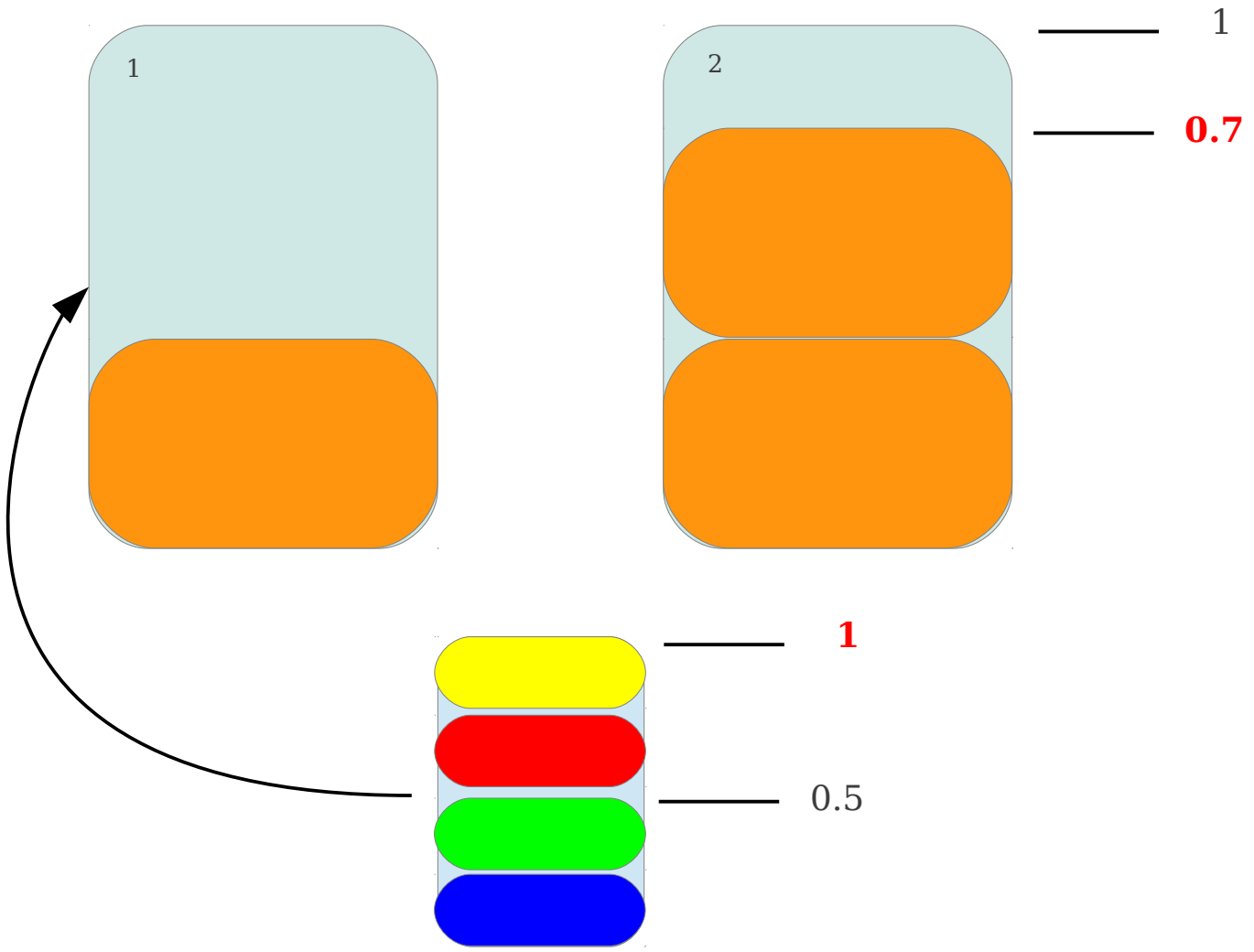


user starts
a console class
video game

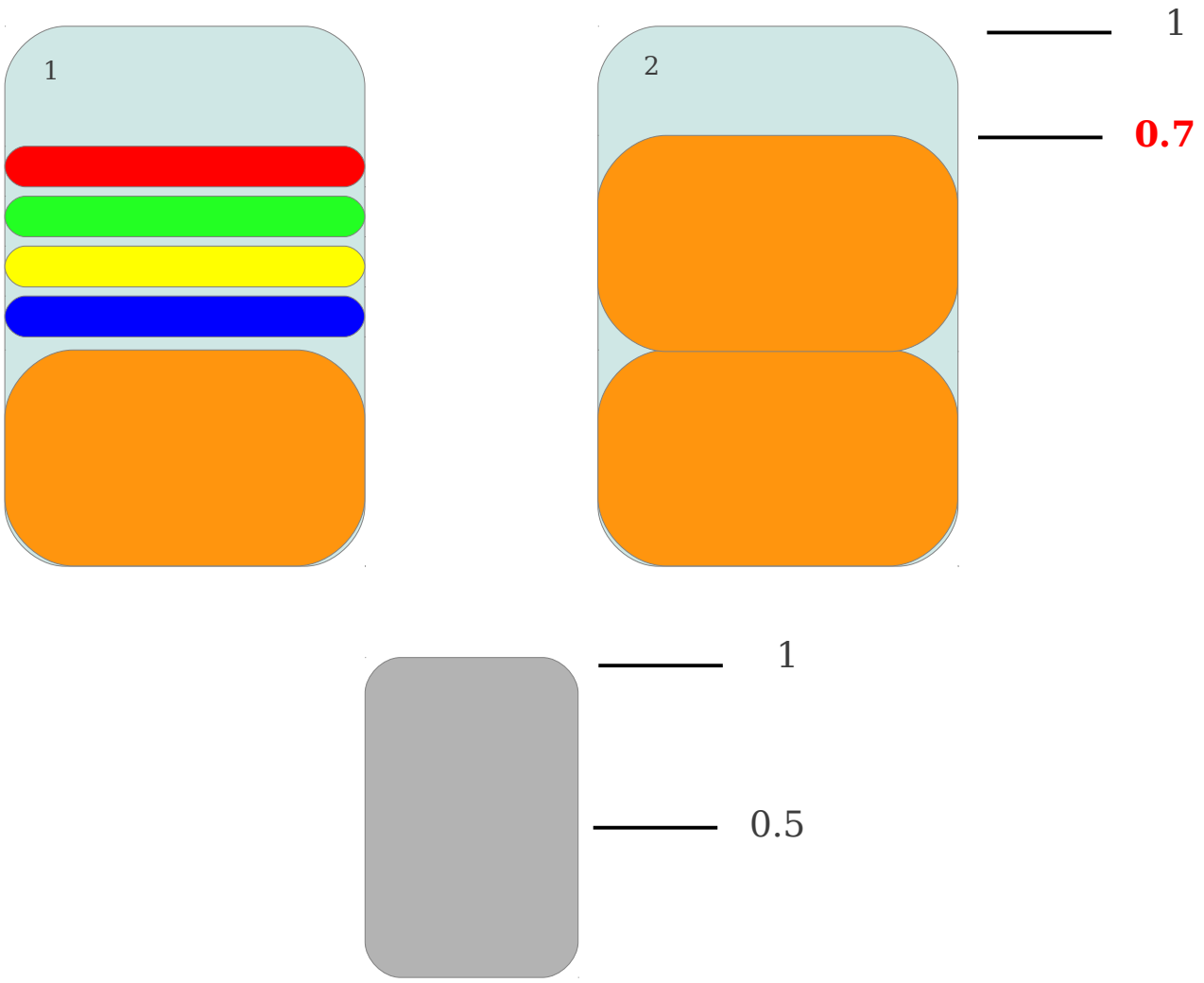
PA push/pull – Tegra like



PA push/pull – Tegra like



PA push/pull – Tegra like



Conclusions & Discussion

Conclusions

- prototype implementation should be straightforward
  - account for runqueue actual utilization
  - modify push/pull mechanism
- integration in a real system (Android ?) could be more difficult
- the plan → start simple and then add bits one ofter the other :-)

Open Questions

- When is it worth to wake up a CPU ?
- Interaction with other components (e.g., thermal management, drivers, etc.) ?
- Test platforms ?
- add here your own concern or ...

Thanks!

Questions ?

juri.lelli@gmail.com

The work leading to this talk has been supported by the European Community's Seventh Framework Programme n.248465 “S(o)OS - Service-oriented Operating Systems” (http://www.soos-project.eu/)