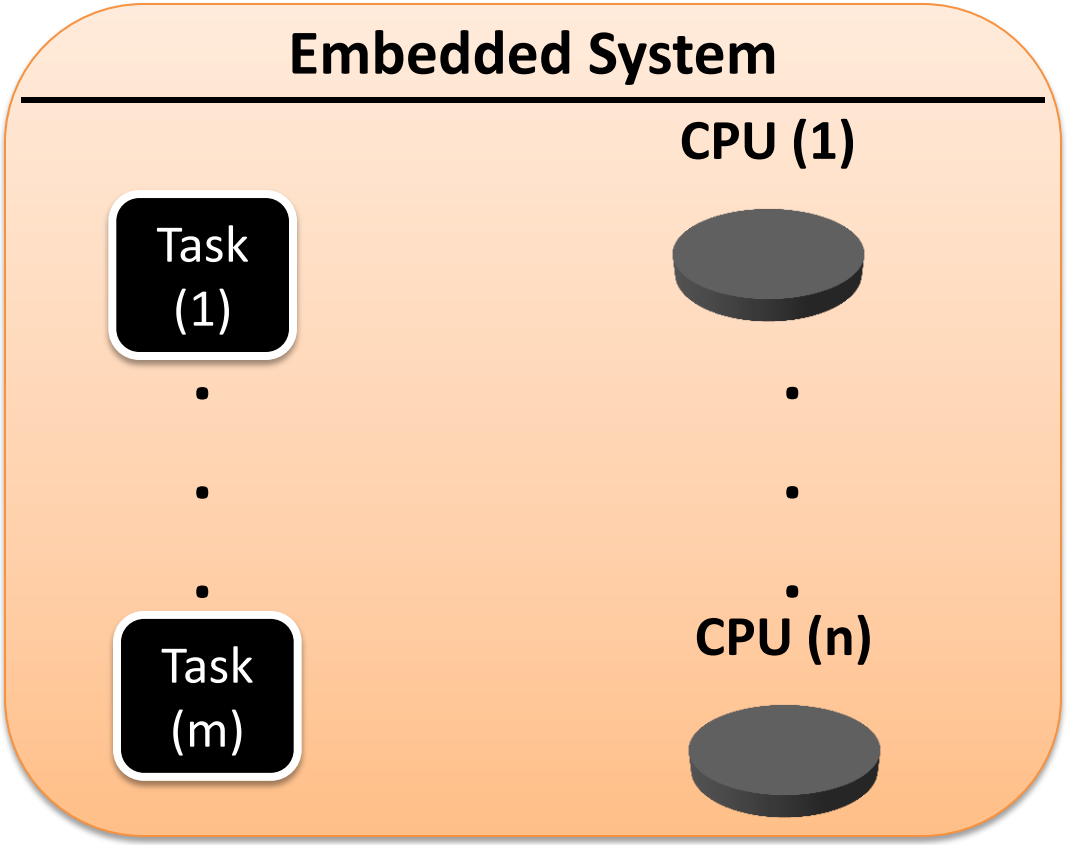
“Implementation of the Multi-Level Adaptive Hierarchical Scheduling Framework”

- *Nima Moghaddami Khalilzad*

July 2013

More applications than processors



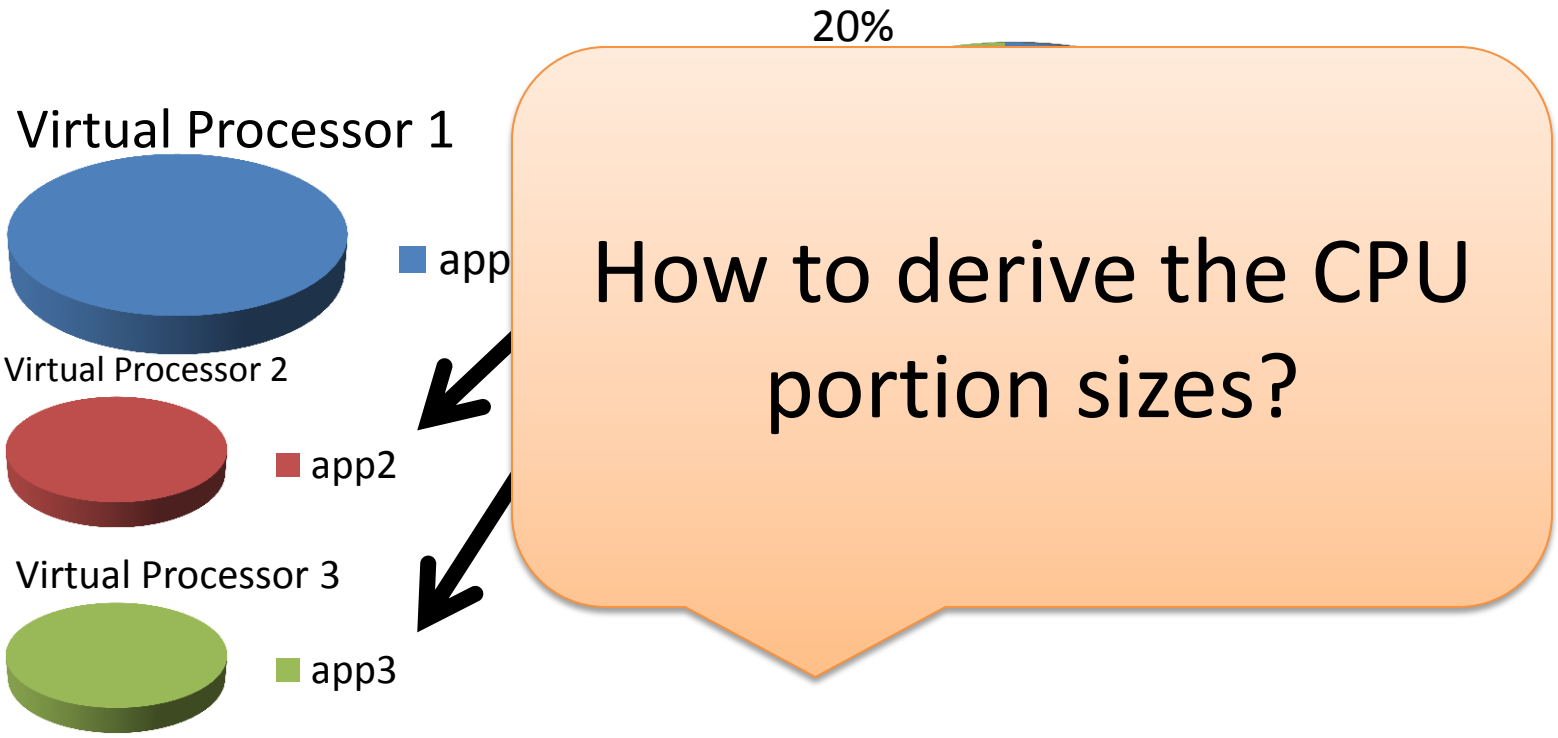
m > n

Sharing CPU

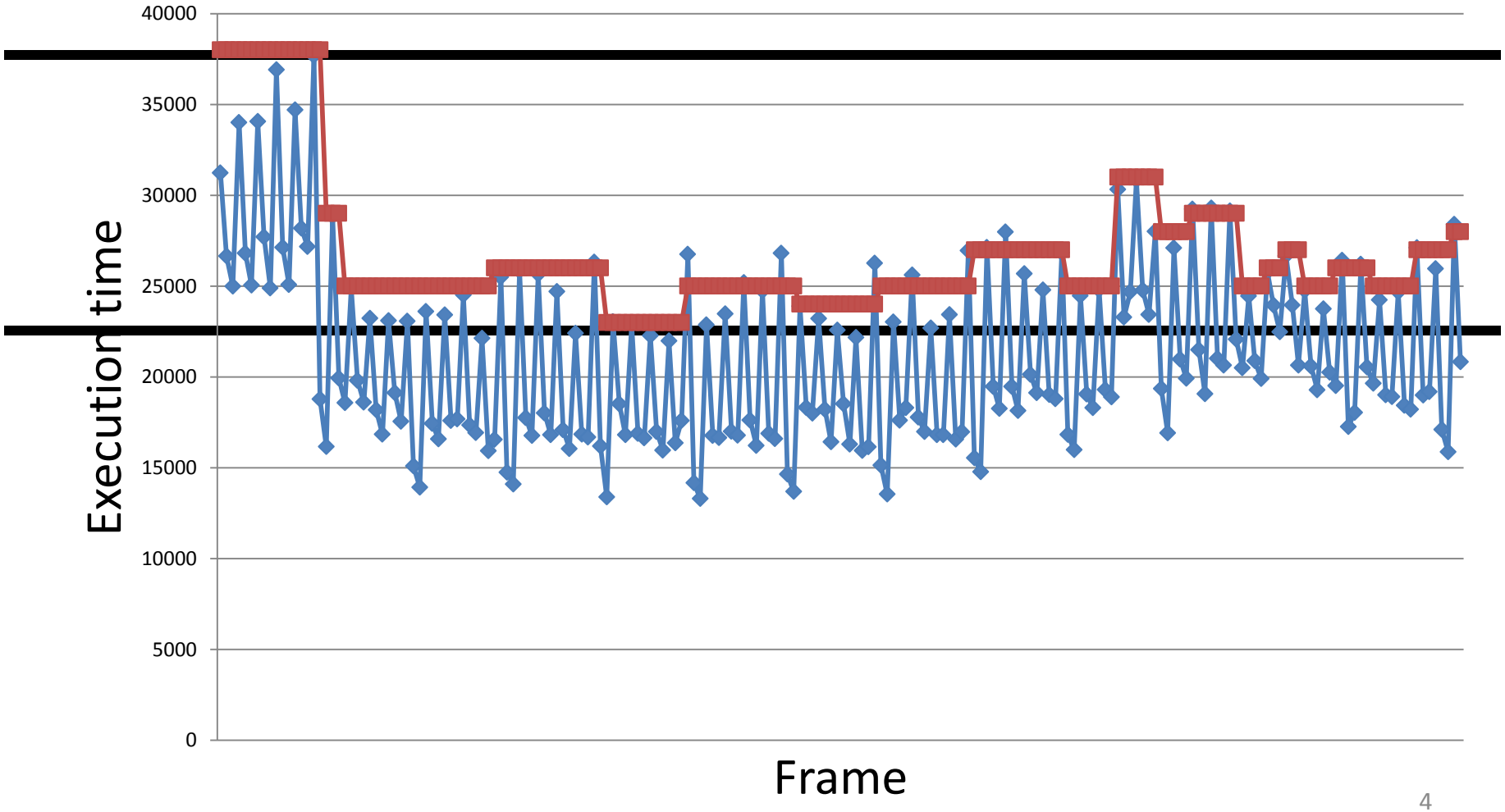
- 800 functions (tasks)
- 70 ECUs

Hierarchical scheduling

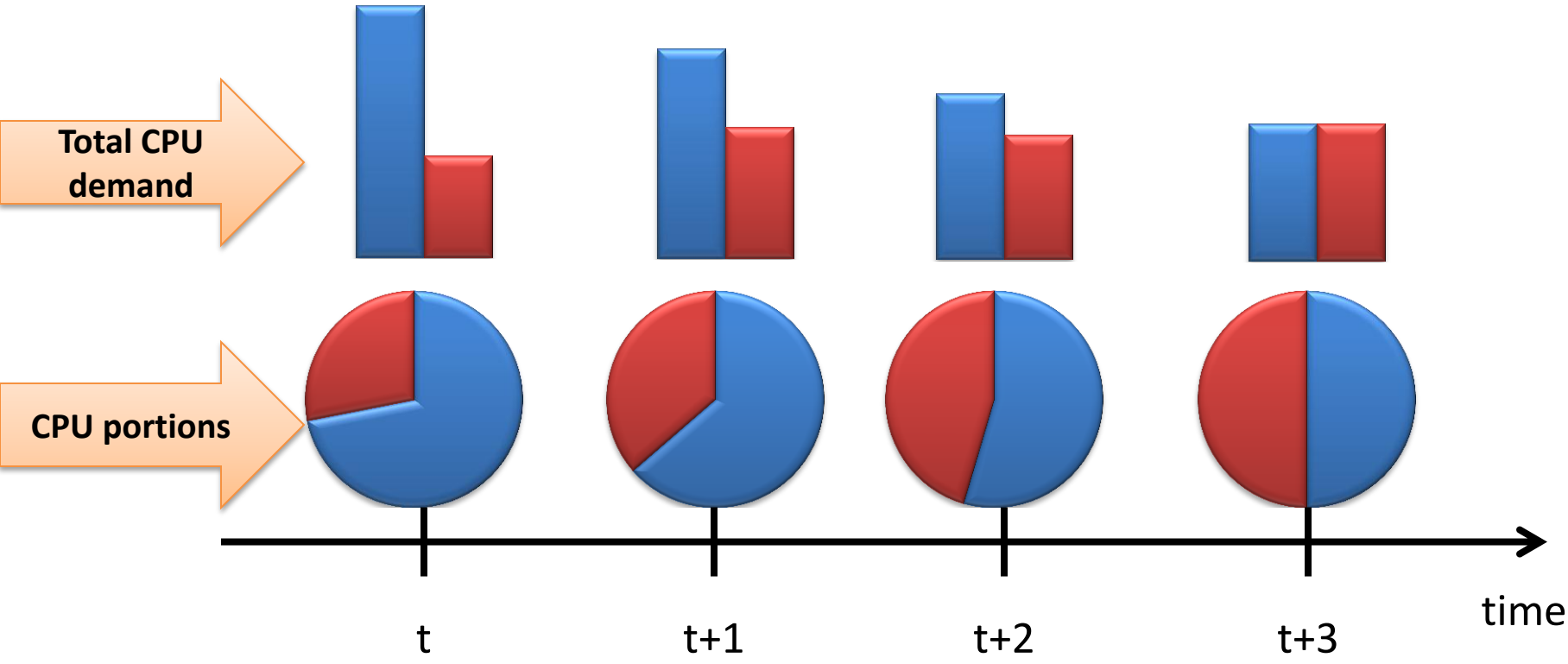
Application point of view:



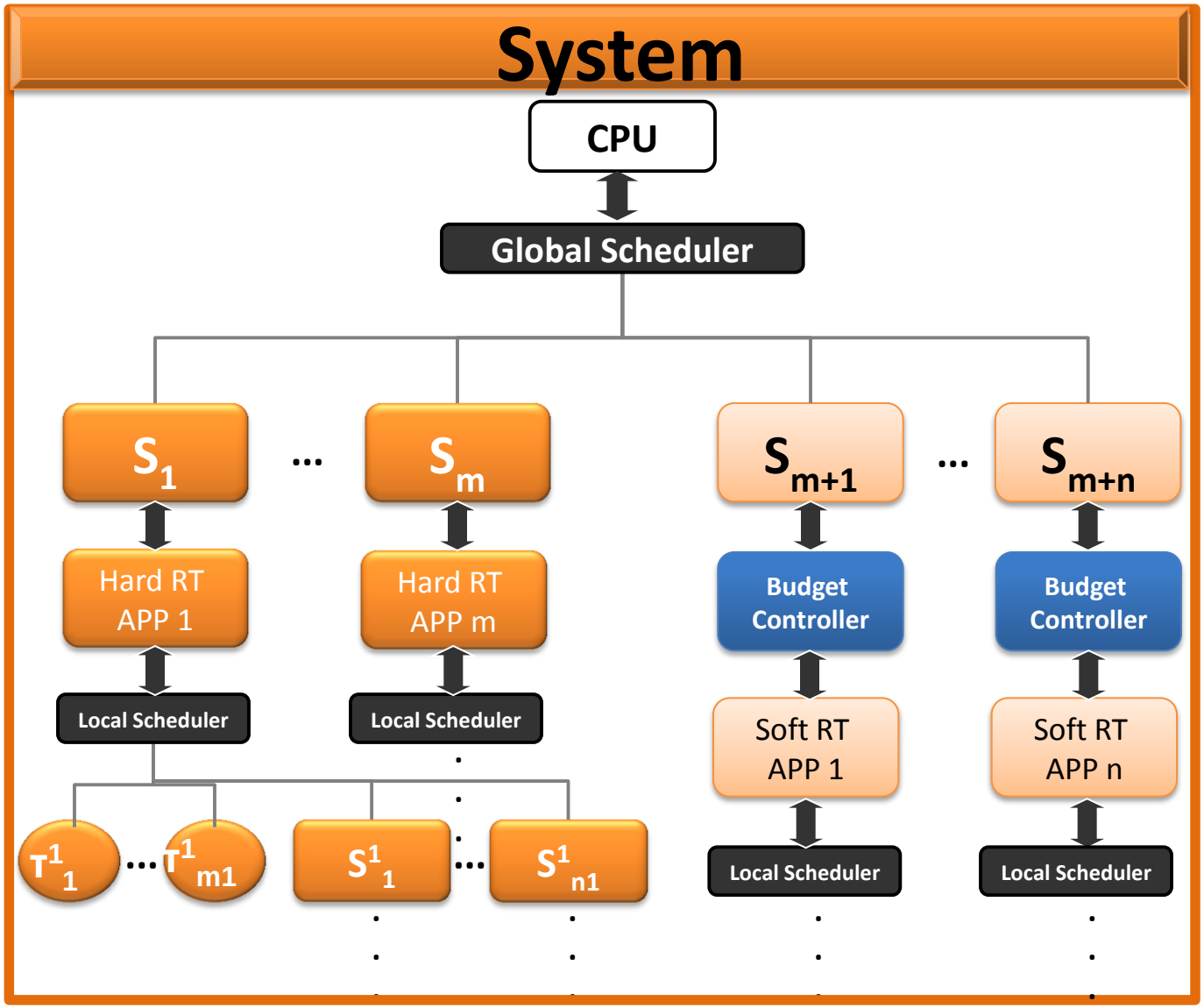
Motivating example: video decoder task



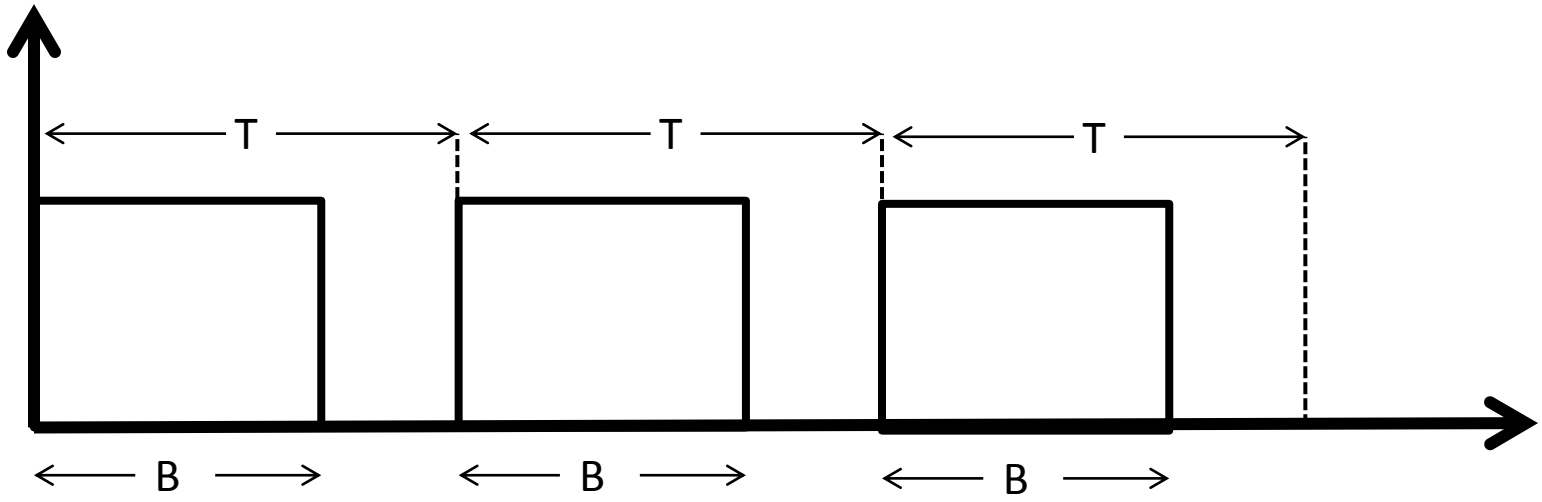
Adaptive hierarchical scheduling framework



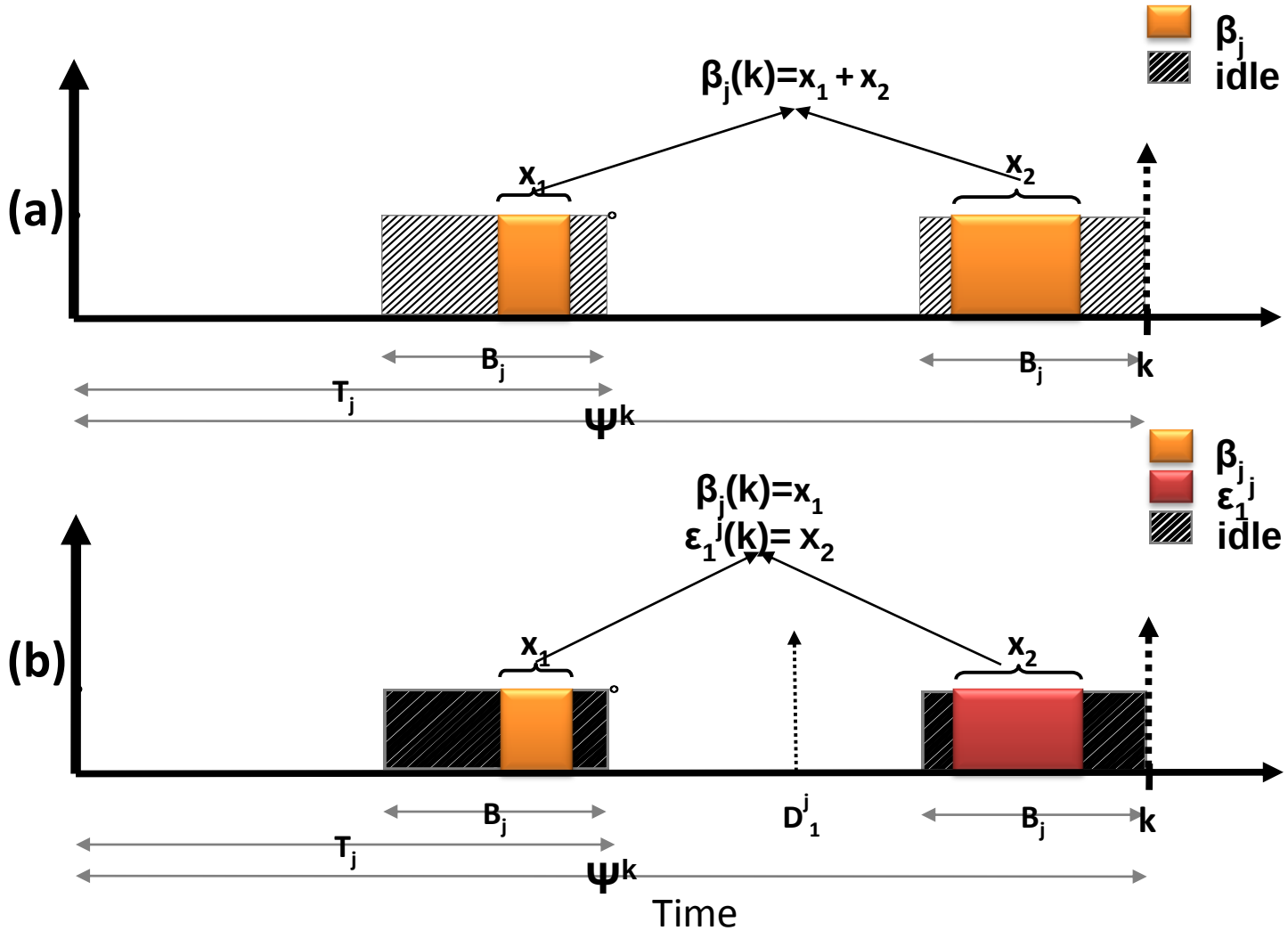
Multi-level AHSF



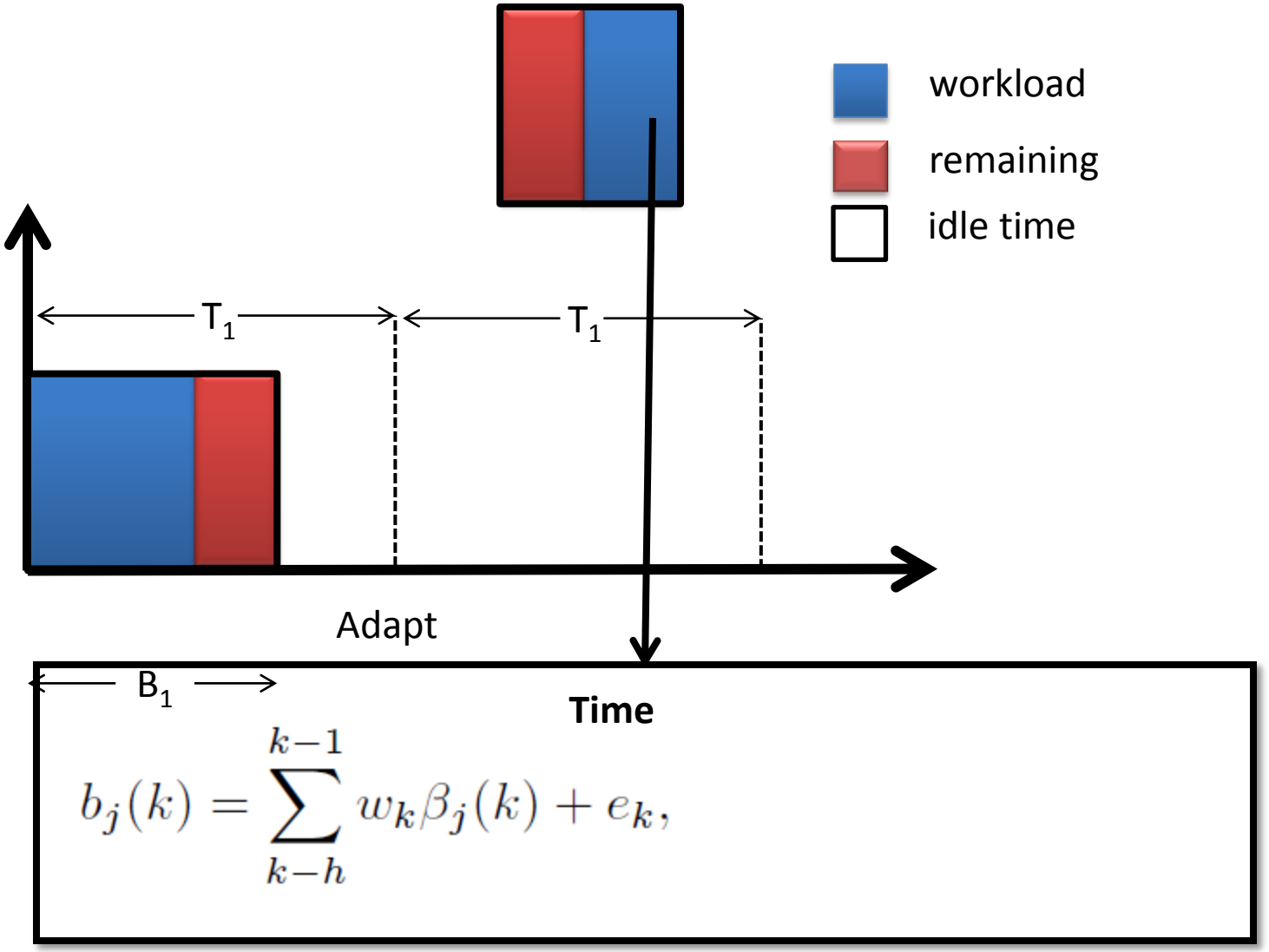
Periodic servers



Controled variables



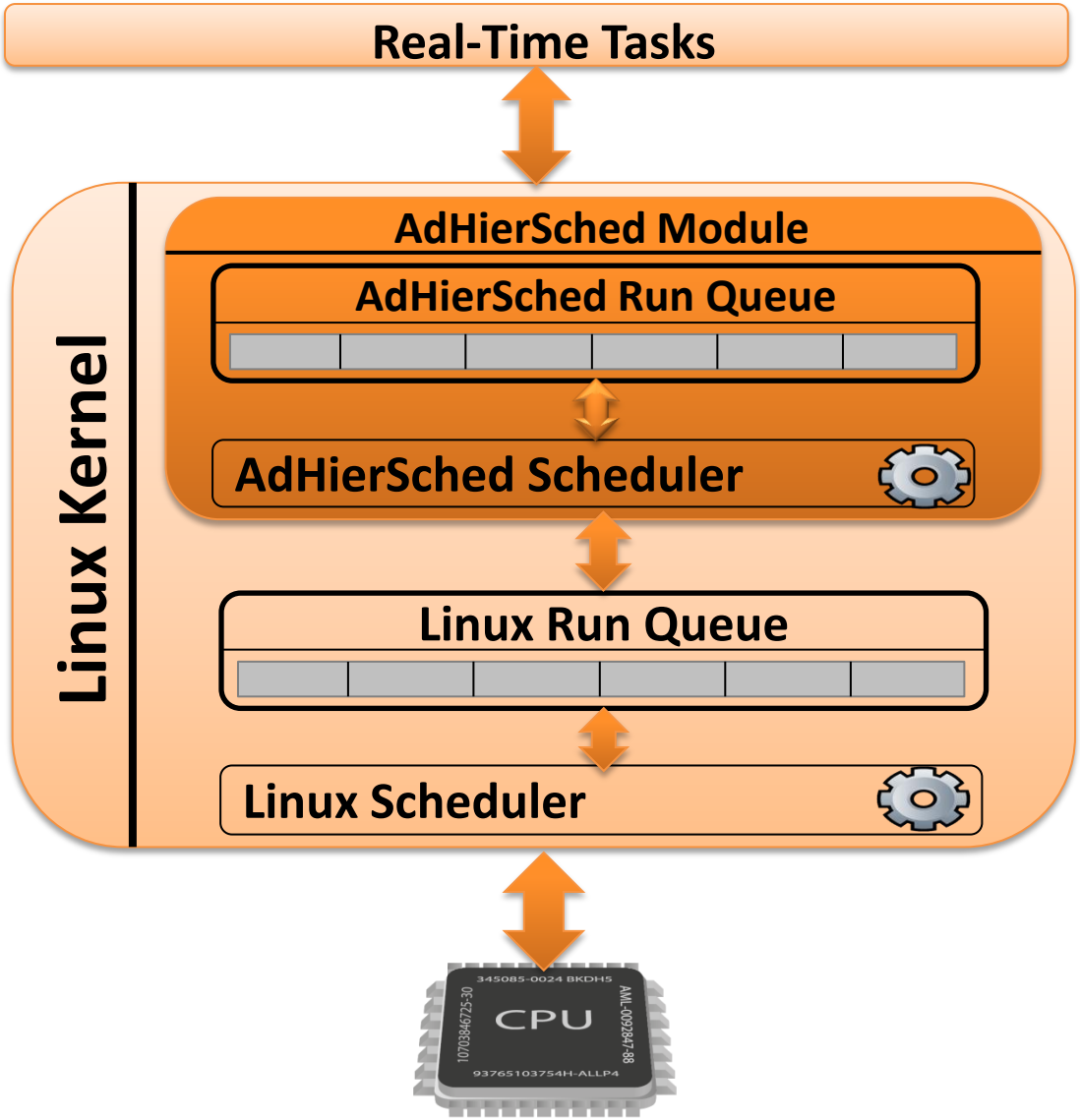
Adaptation model



$$b_j(k) = \sum_{k-h}^{k-1} w_k \beta_j(k) + e_k,$$

AdHierSched

Kernel loadable module



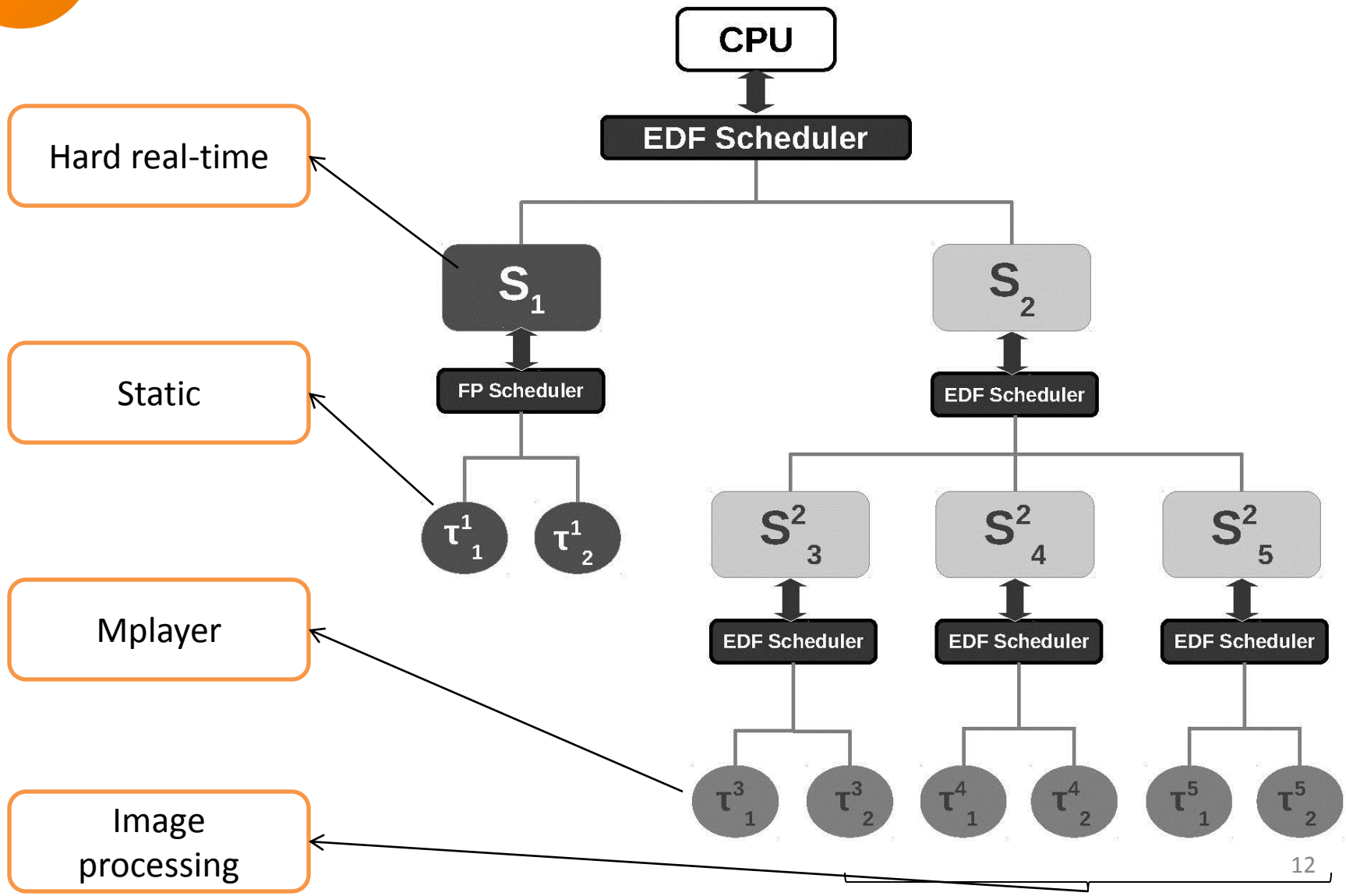
Implementation details

- *Low resolution timers*
- *Task descriptor data structure*
- *API functions (attach, start, detach, job_finish)*

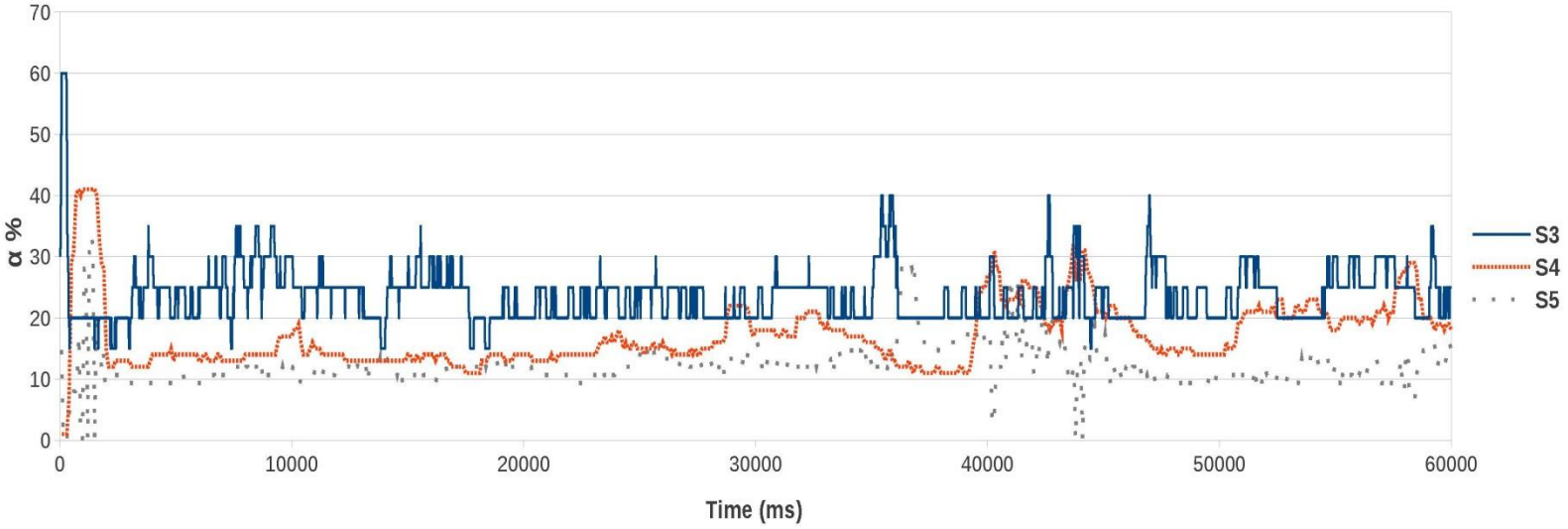
Code Snippet 3: Sample task structure.

```
1: int main(int argc, char* argv[]){
2: task_id = atoi(argv[1]);
3: attach_task_to_mod(task_id);
4: while i < job_no do
5:    /* periodic job */
6:    task_finish_job(task_id);
7: end while
8: detach_task(task_id);
9: return 0; }
```

Case study

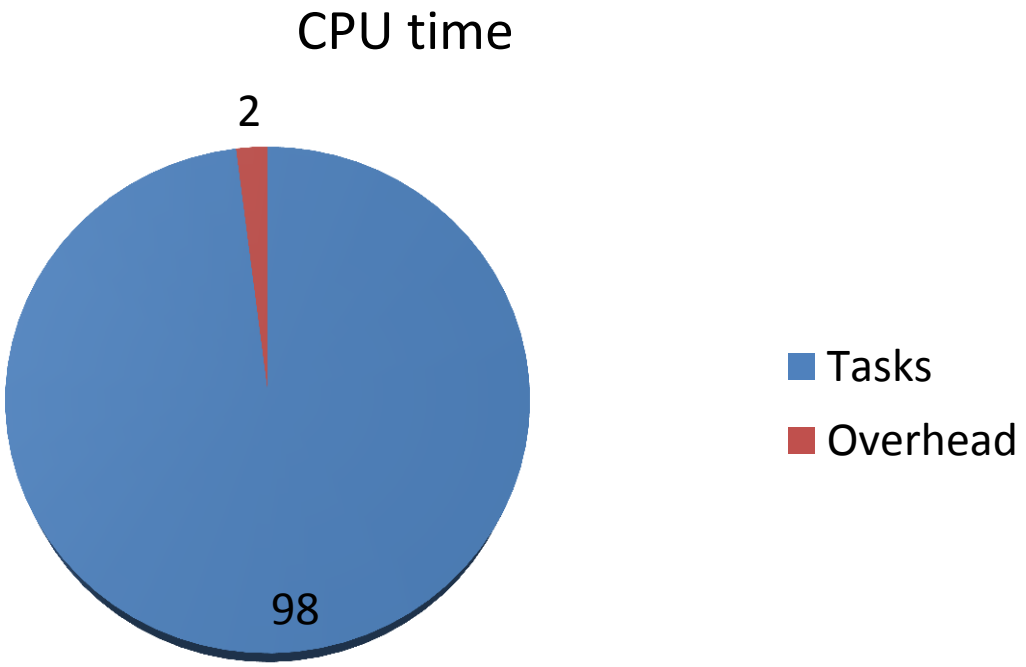

Hard real-time
Static
Mplayer
Image processing
CPU
EDF Scheduler
S_1
S_2
FP Scheduler
EDF Scheduler
τ^1_1
τ^1_2
S^2_3
S^2_4
S^2_5
EDF Scheduler
EDF Scheduler
EDF Scheduler
τ^3_1
τ^3_2
τ^4_1
τ^4_2
τ^5_1
τ^5_2

Budget adaptations

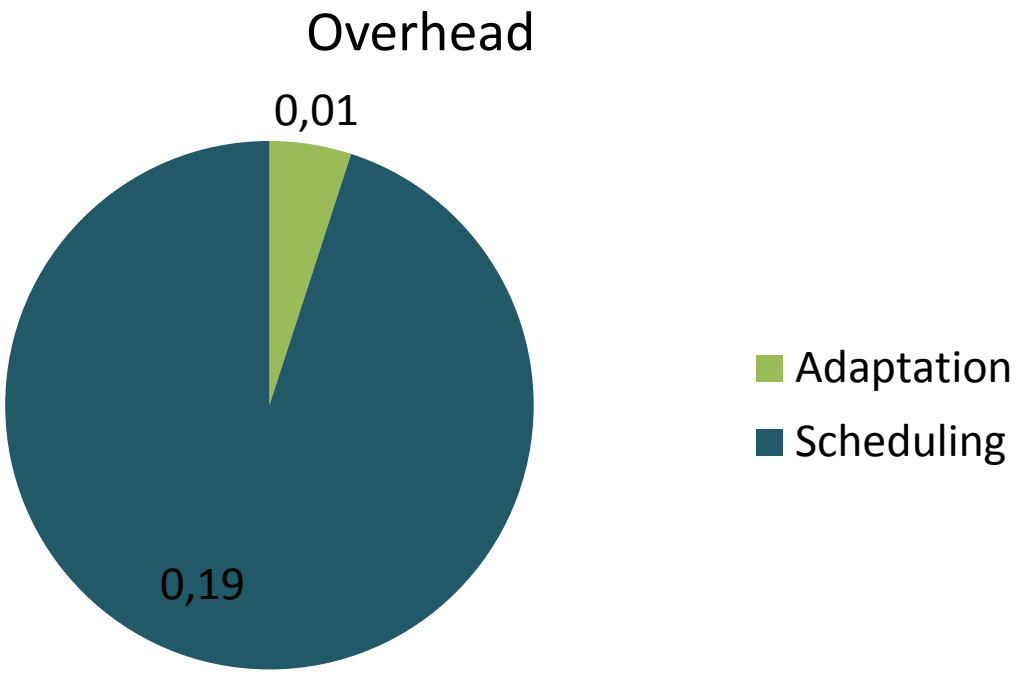


	Mean	Adaptive
S3	3.36 %	1.11 %
S4	27.28 %	6.49 %
S5	2.19 %	4.69 %
total	32.83 %	12.29 %

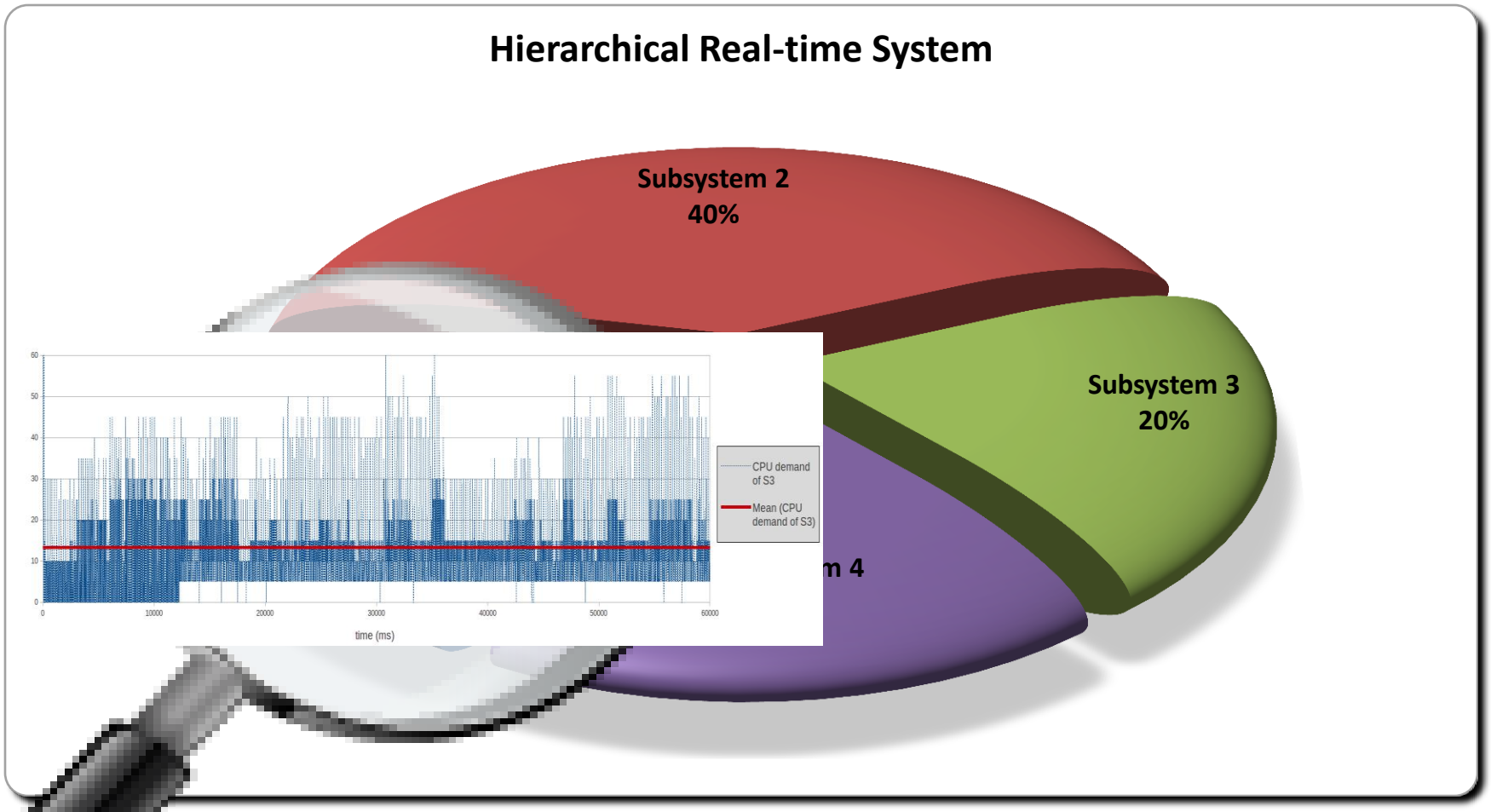
Total overhead



Adaptation overhead

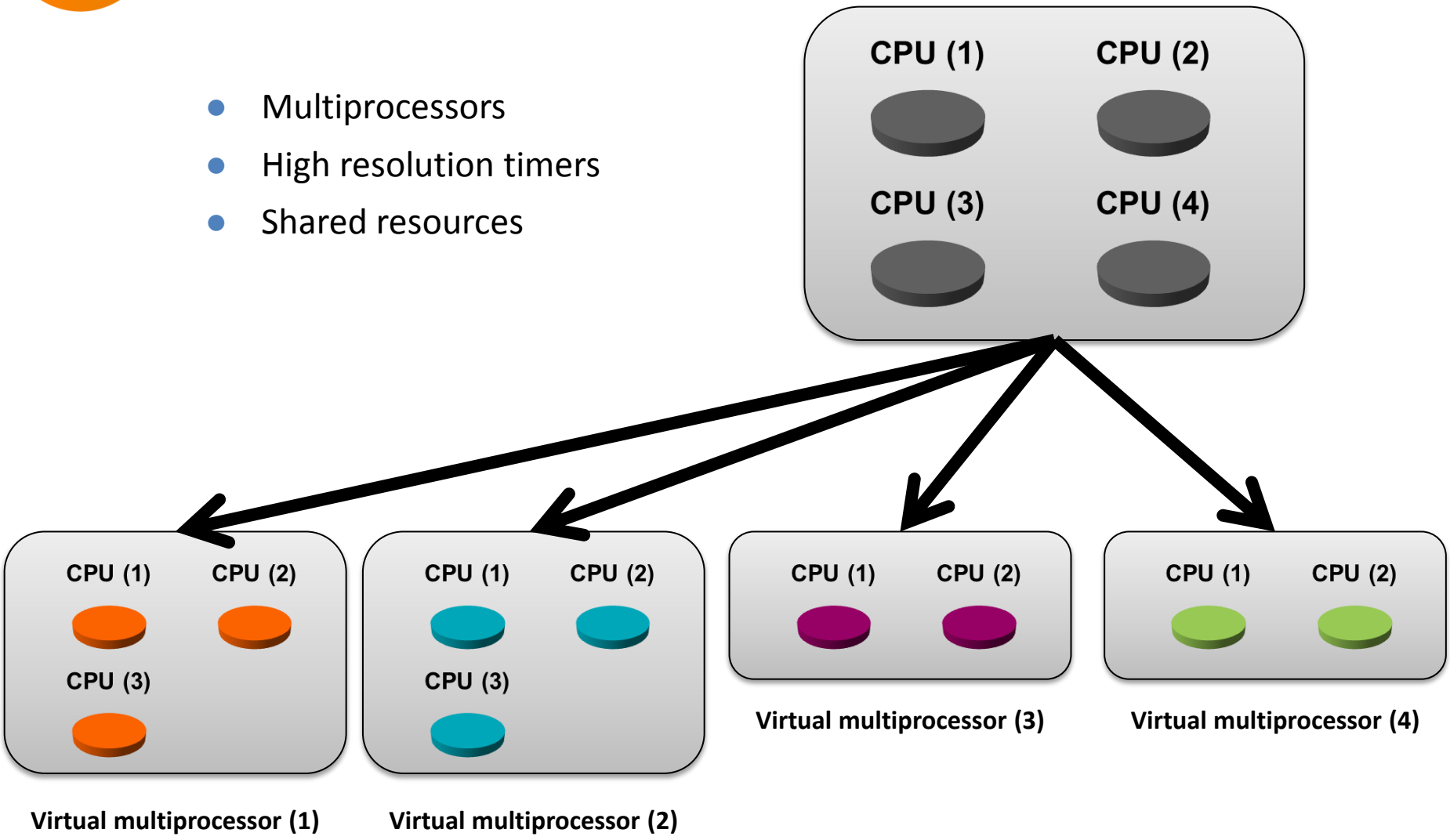


Summary


Hierarchical Real-time System
Subsystem 2
40%
Subsystem 3
20%
m 4
CPU demand of S3
Mean (CPU demand of S3)
time (ms)

Future work

- Multiprocessors
- High resolution timers
- Shared resources



Thank you ☺