INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

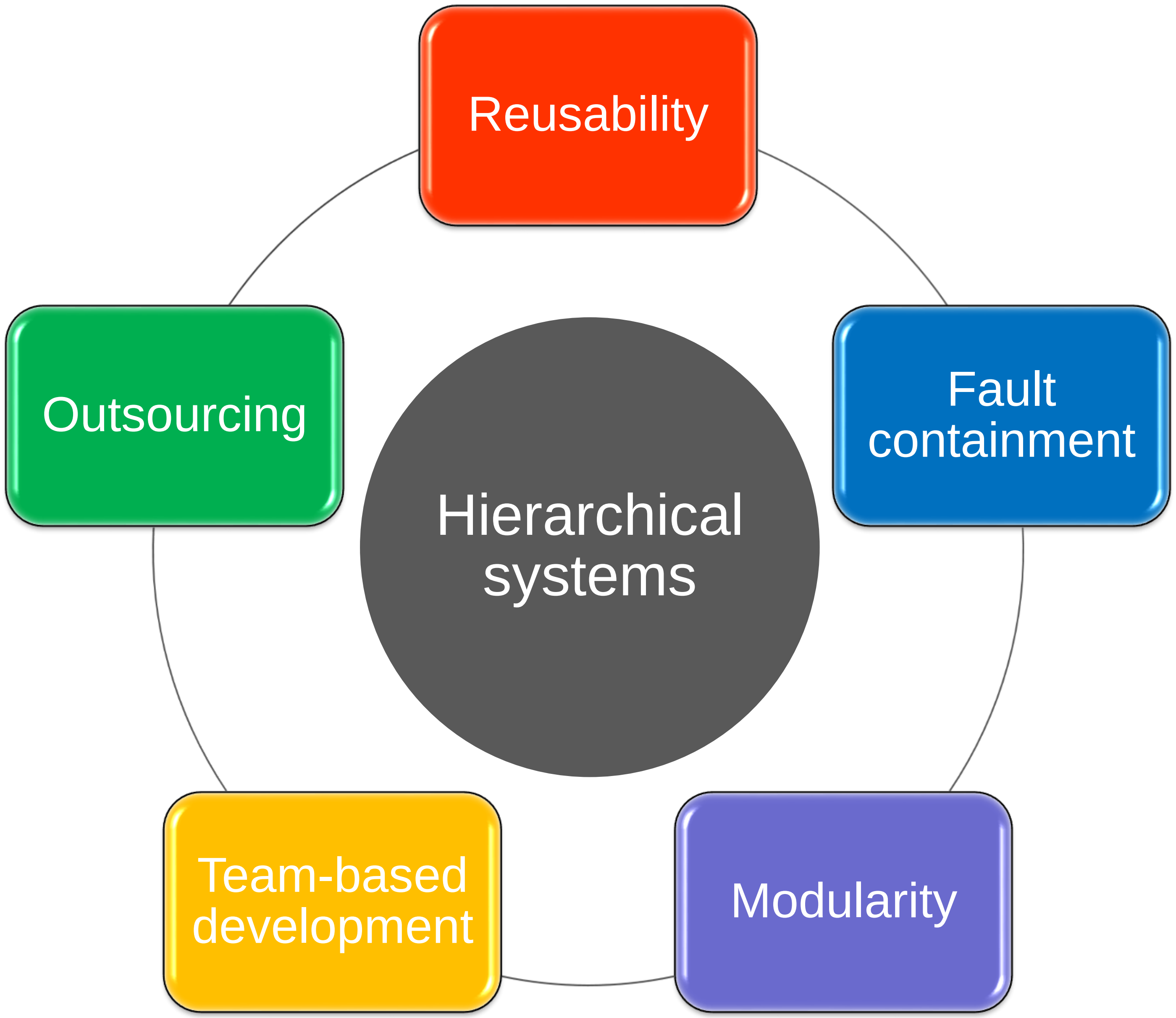
Retis Real-Time Systems Laboratory

Optimal Design for Reservation Servers under Shared Resources

Alessandro Biondi, **Alessandra Melani**, Marko Bertogna, Giorgio Buttazzo

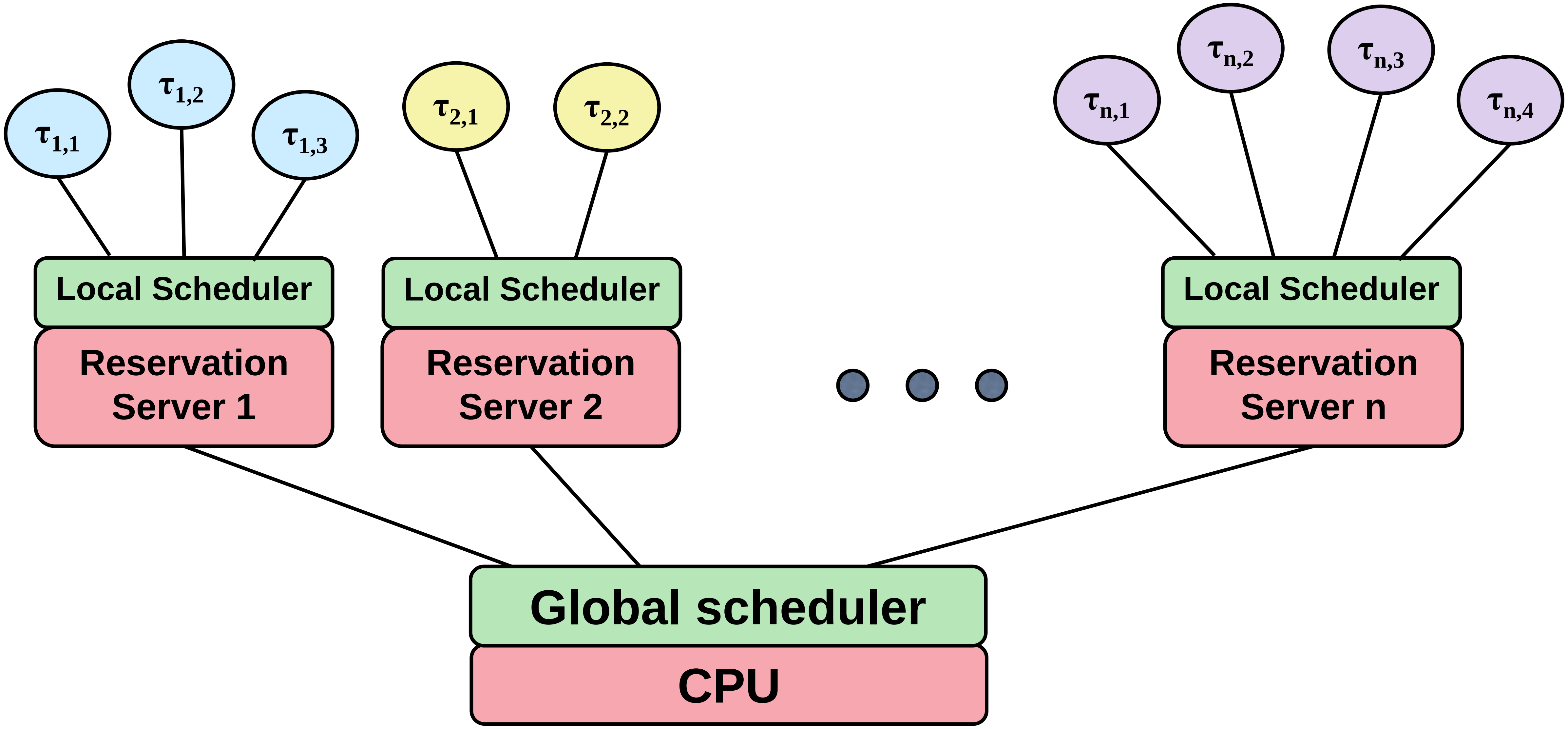
Towards integrated architectures

- The increased SW complexity and HW performance motivate some basic practices for SW development



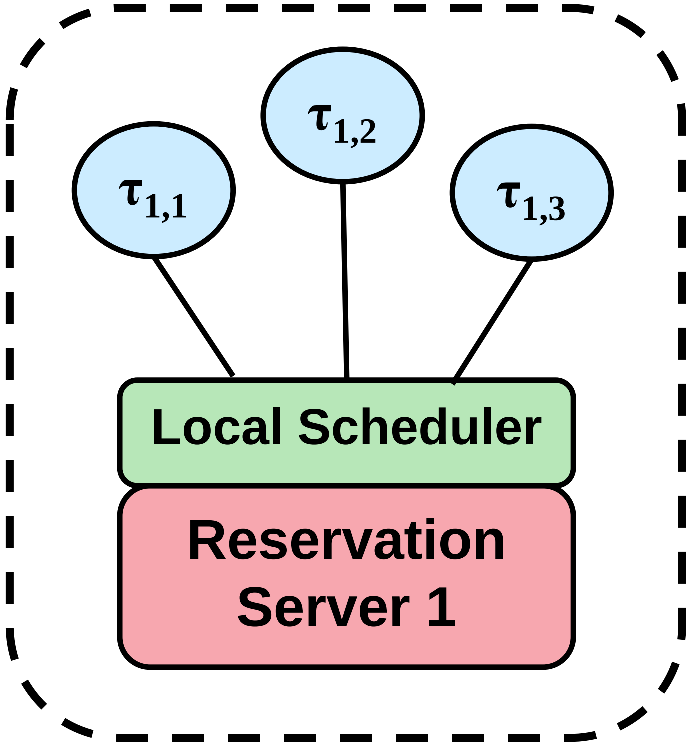
Hierarchical Systems

- Many SW applications can be integrated forming a **Hierarchical System**



Hierarchical Systems

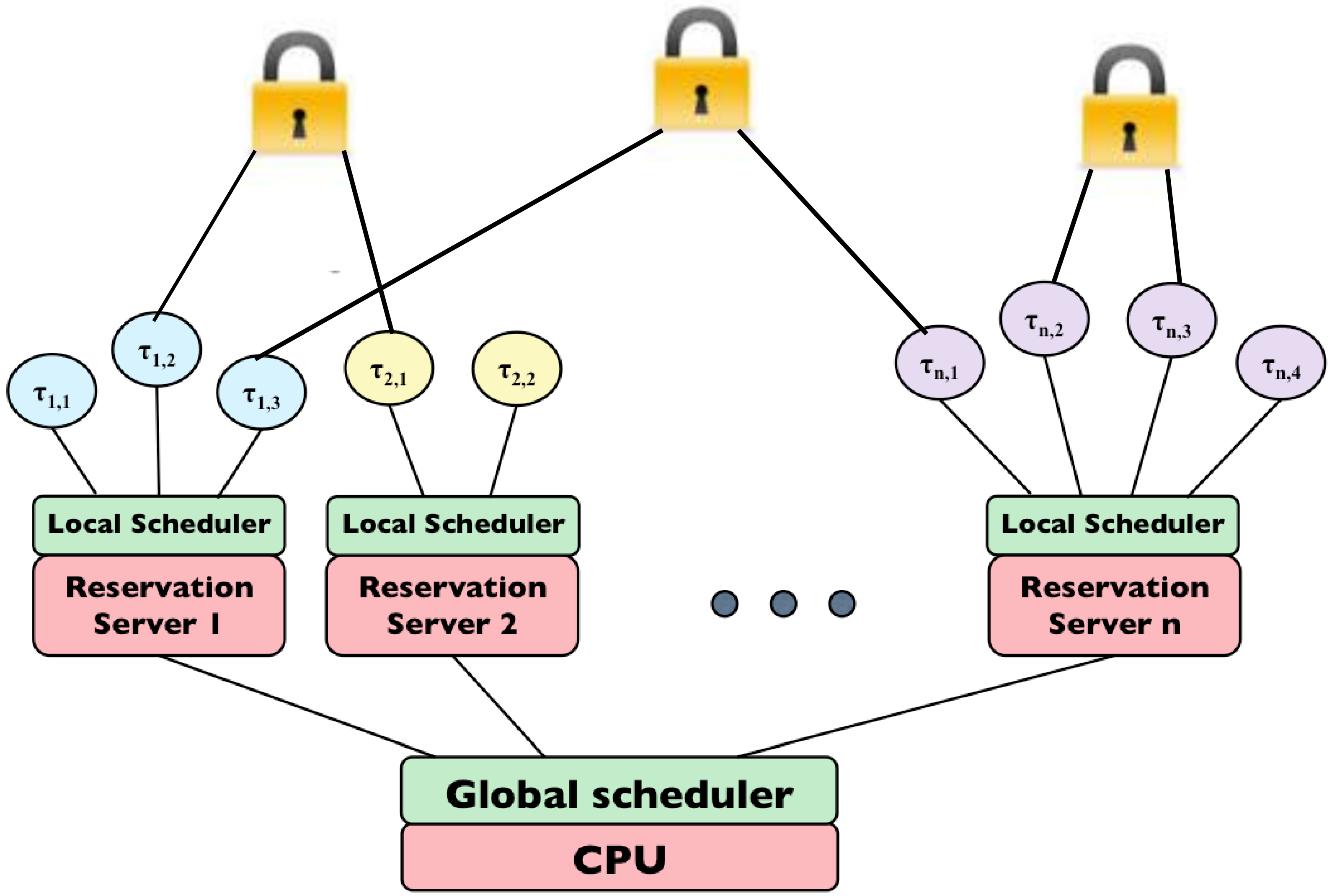
- Each application is scheduled within a **reservation** implemented as a **periodic server**



- Q → server **budget**
- P → server **period**
- $\alpha = \frac{Q}{P}$ → server **bandwidth**
- $\Delta = 2 * (P - Q)$ → server **delay**

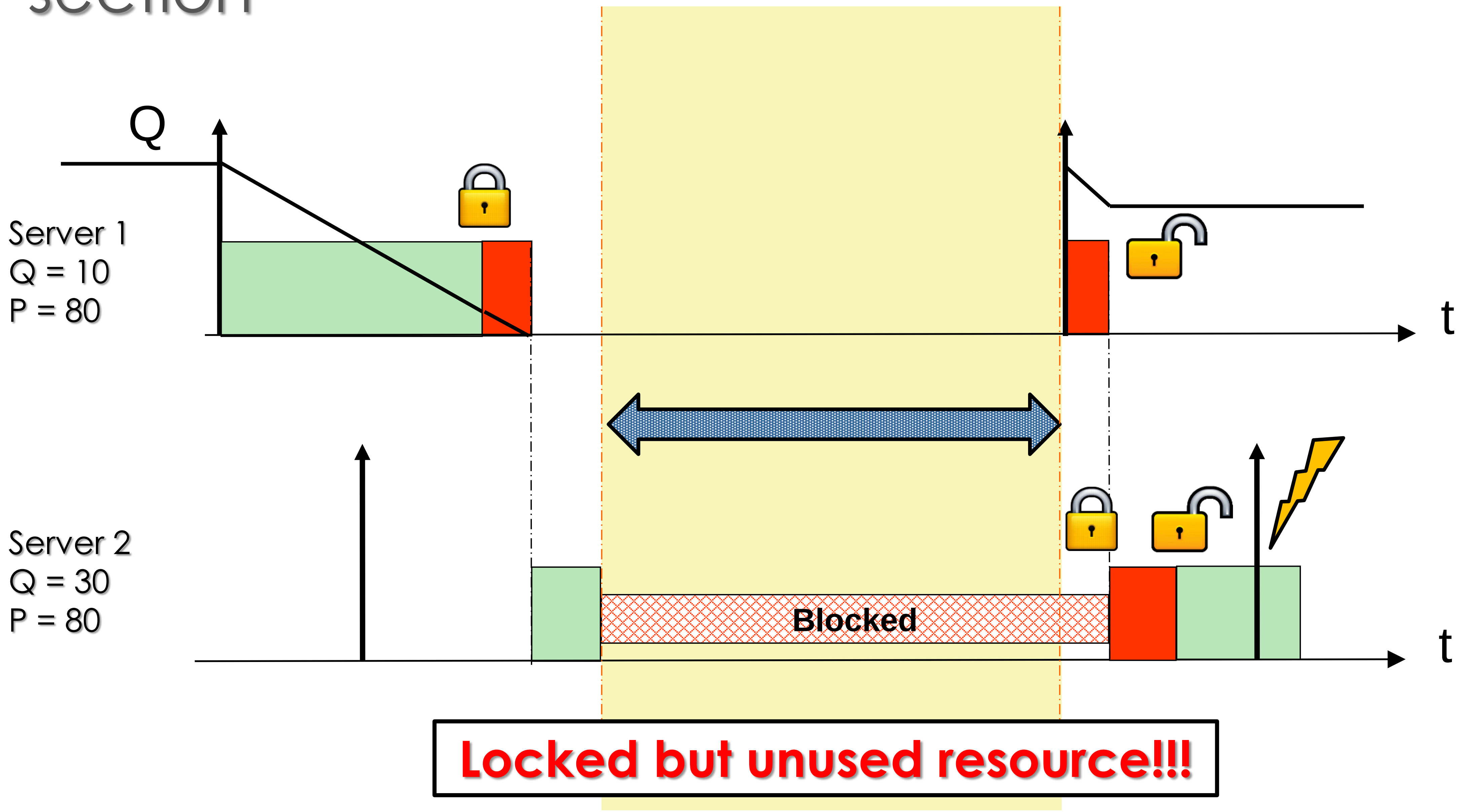
Resource Sharing

- Subsystems are usually <u>not independent</u>: tasks may share **resources** (globally and/or locally)



Resource Sharing

- Problem: budget exhaustion inside a critical section



Existing approaches

- **Reactive** (handle budget exhaustion)
  - Overrun (Ghazalie and Baker, 1995; Davis and Burns, 2006)

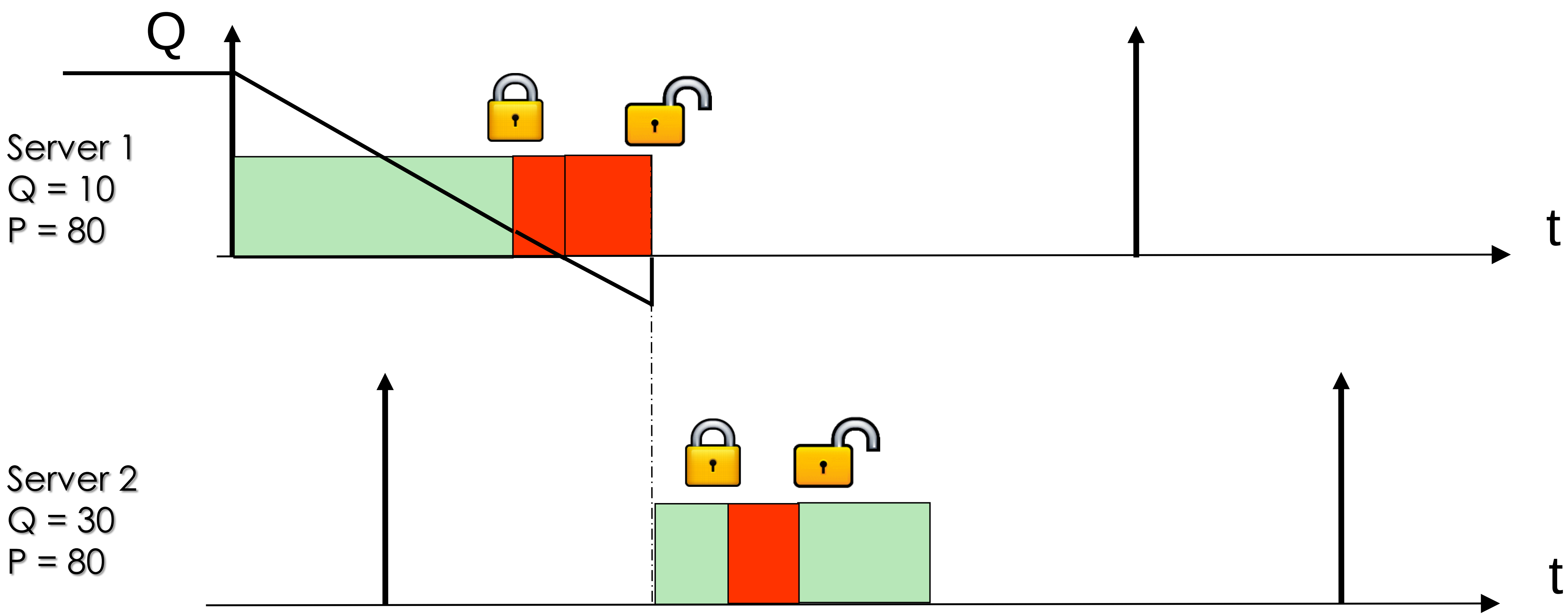
- **Proactive** (prevent budget exhaustion)
  - SIRAP (Behnam et al., 2007)
  - BROE (Bertogna, Fisher and Baruah, 2009)

Require the knowledge on the maximum Resource Holding Time

Maximum amount of budget consumed
from the lock to the release
of a globally shared resource

Overrun

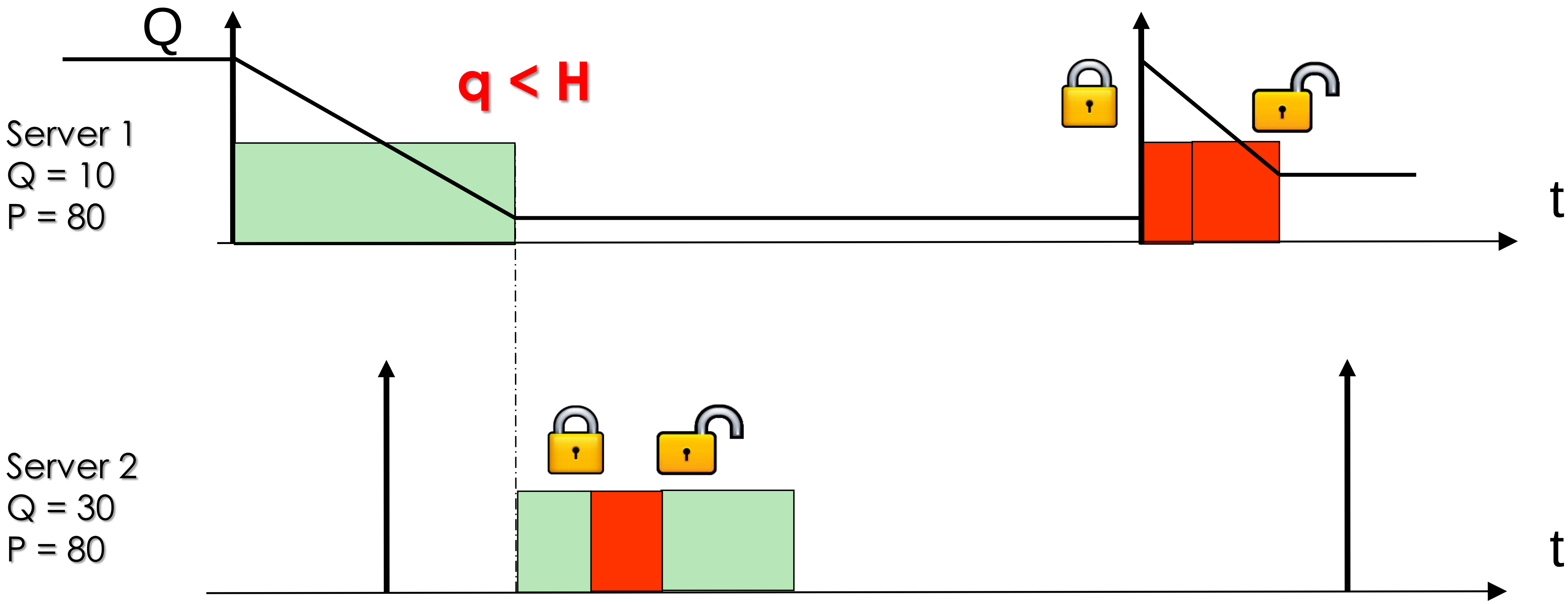
- Consume extra budget until critical section is completed
- With or without payback



Cons: greater bandwidth requirement for the reservation!

SIRAP

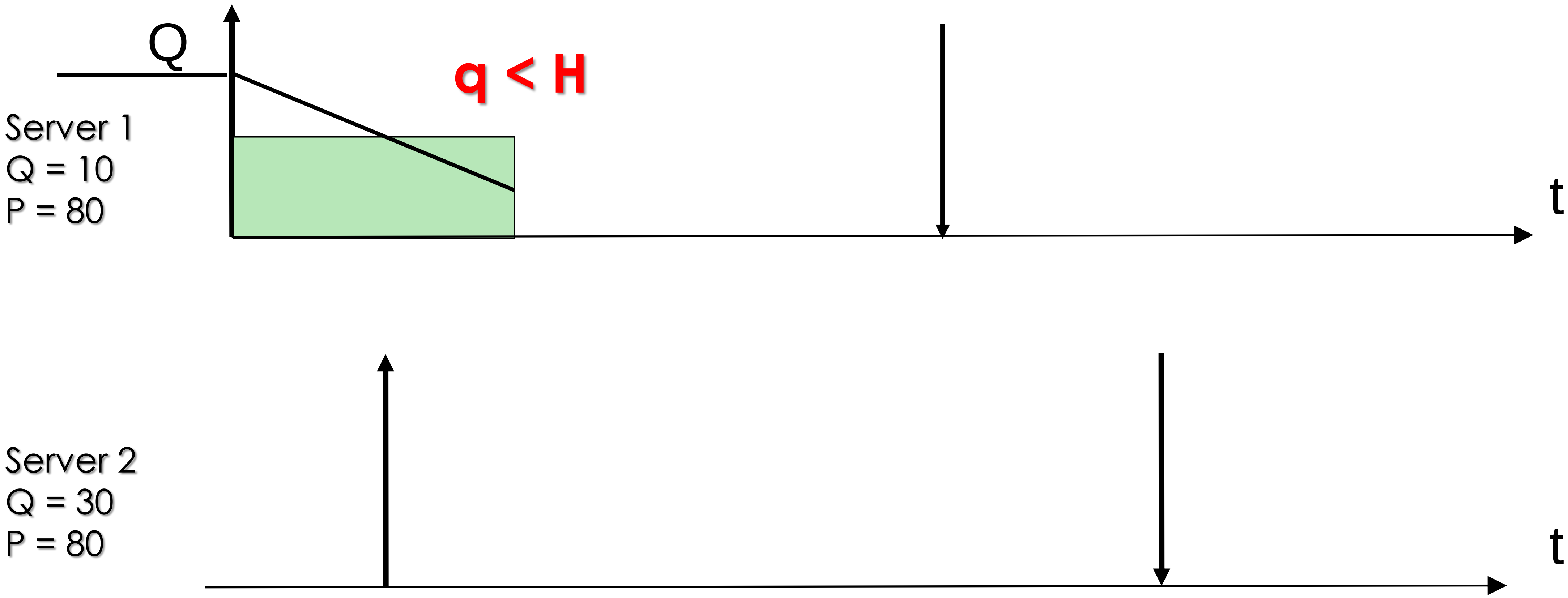
- **Budget check** before each locking operation
  - If the budget is not sufficient to complete the critical section, the access is postponed until next replenishment



Cons: - penalizes response time of the served task!
- remaining budget could be wasted!

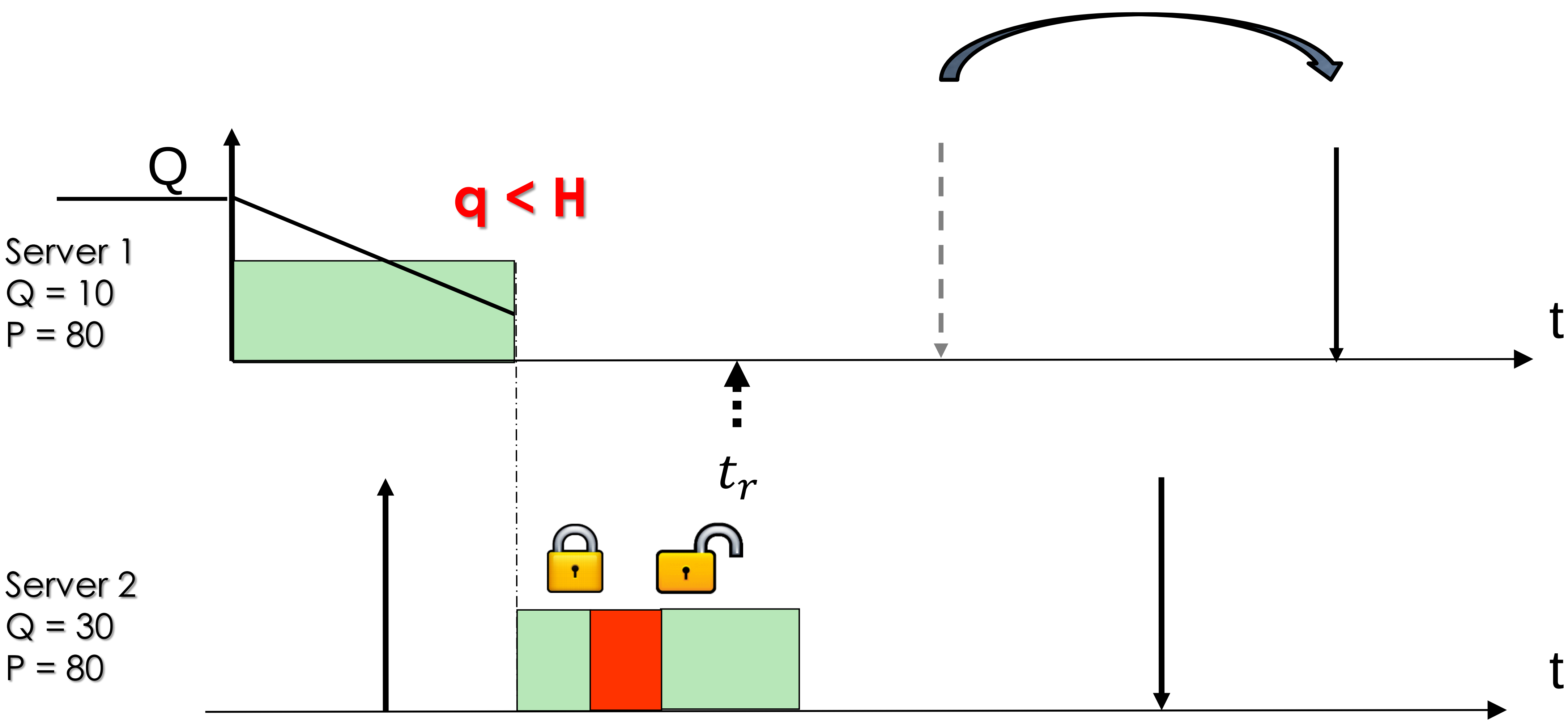
BROE (1)

- **Budget check** before each locking operation
  - If the budget is not sufficient to complete the critical section, a replenishment time is set at the earliest time that preserves bandwidth & delay; the server is reactivated “as soon as possible”
  - Based on EDF scheduling



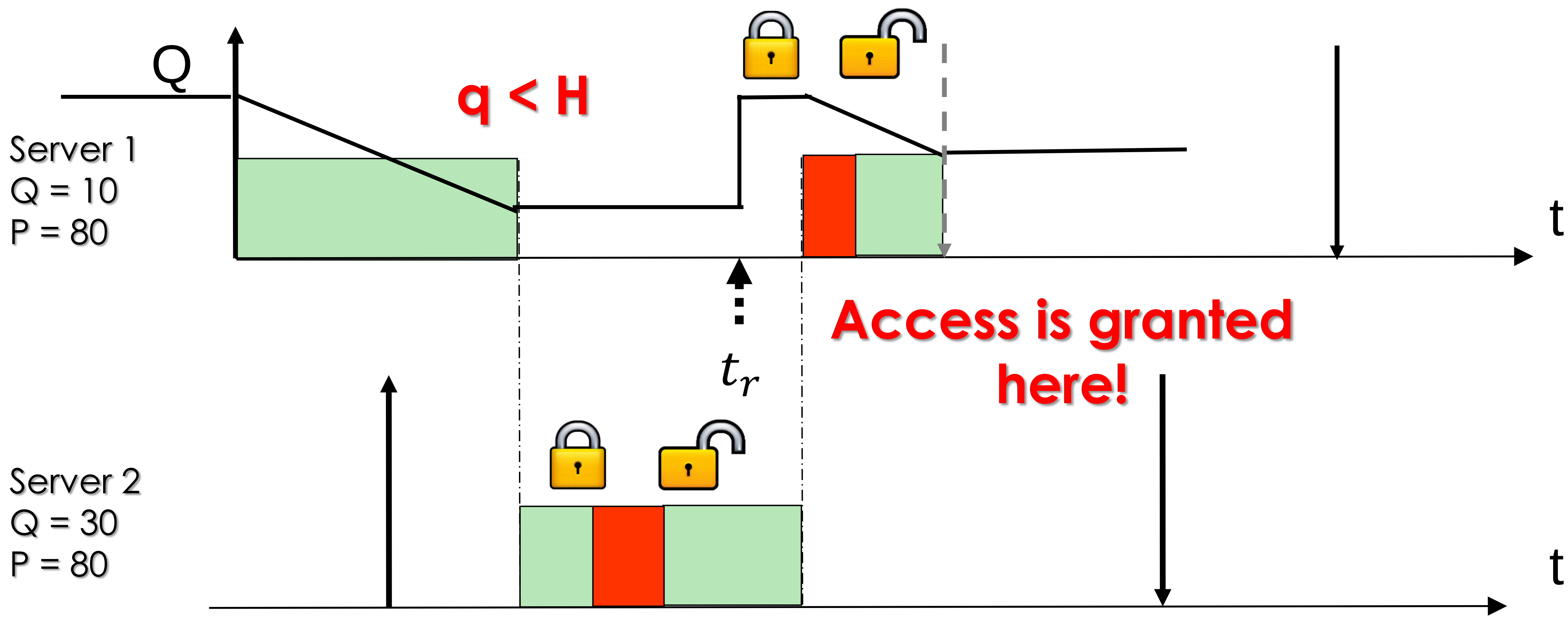
BROE (2)

- **Budget check** before each locking operation
  - If the budget is not sufficient to complete the critical section, a replenishment time is set at the earliest time that preserves bandwidth & delay; the server is reactivated "as soon as possible"
  - Based on EDF scheduling



BROE (3)

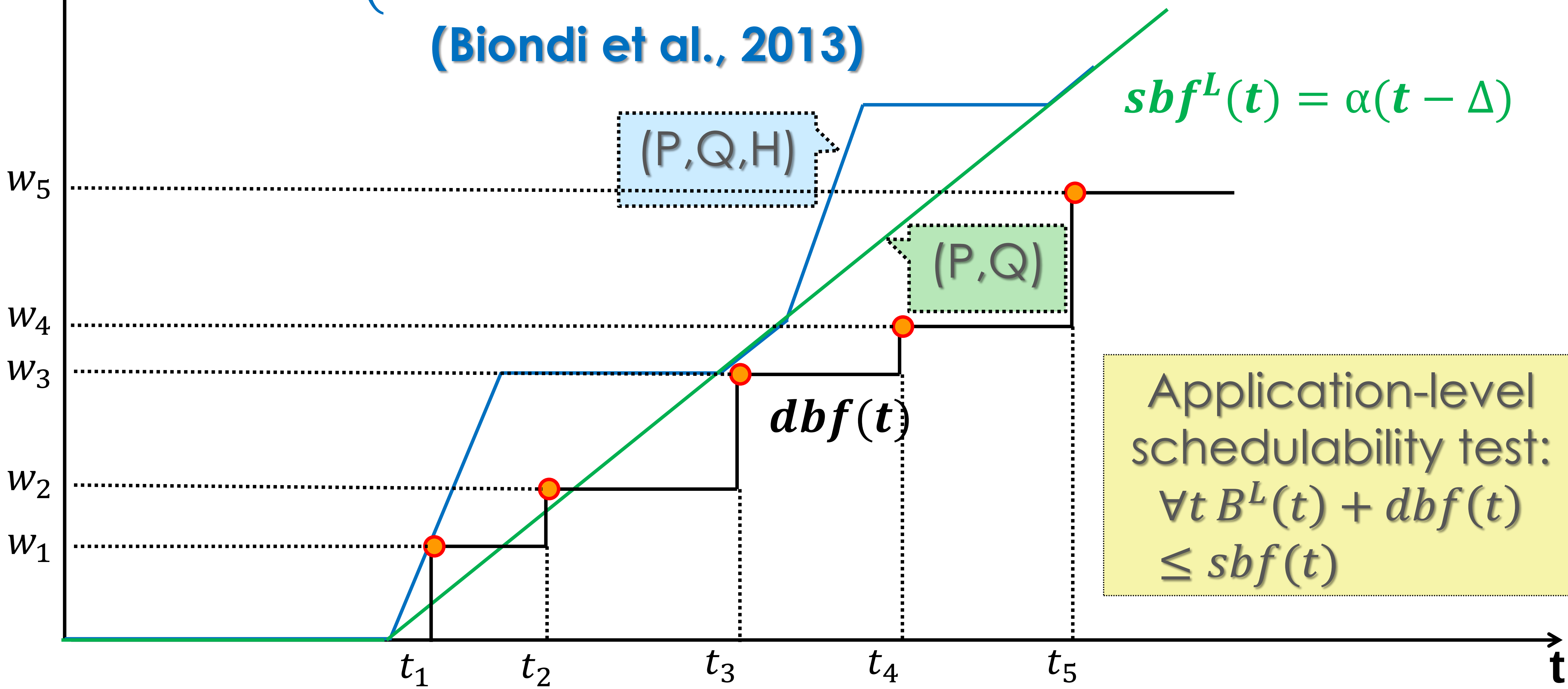
- **Budget check** before each locking operation
  - If the budget is not sufficient to complete the critical section, a replenishment time is set at the earliest time that preserves bandwidth & delay; the server is reactivated "as soon as possible"
  - Based on EDF scheduling



BROE schedulability analysis

$$sbf^B(t) = \begin{cases} t - \Delta - (k-1)(P-Q) & t_A \le t \le t_B \\ kQ - kH & t_B \le t \le t_C \\ \alpha * (t - \Delta) & t_C \le t \le t_D \end{cases}$$

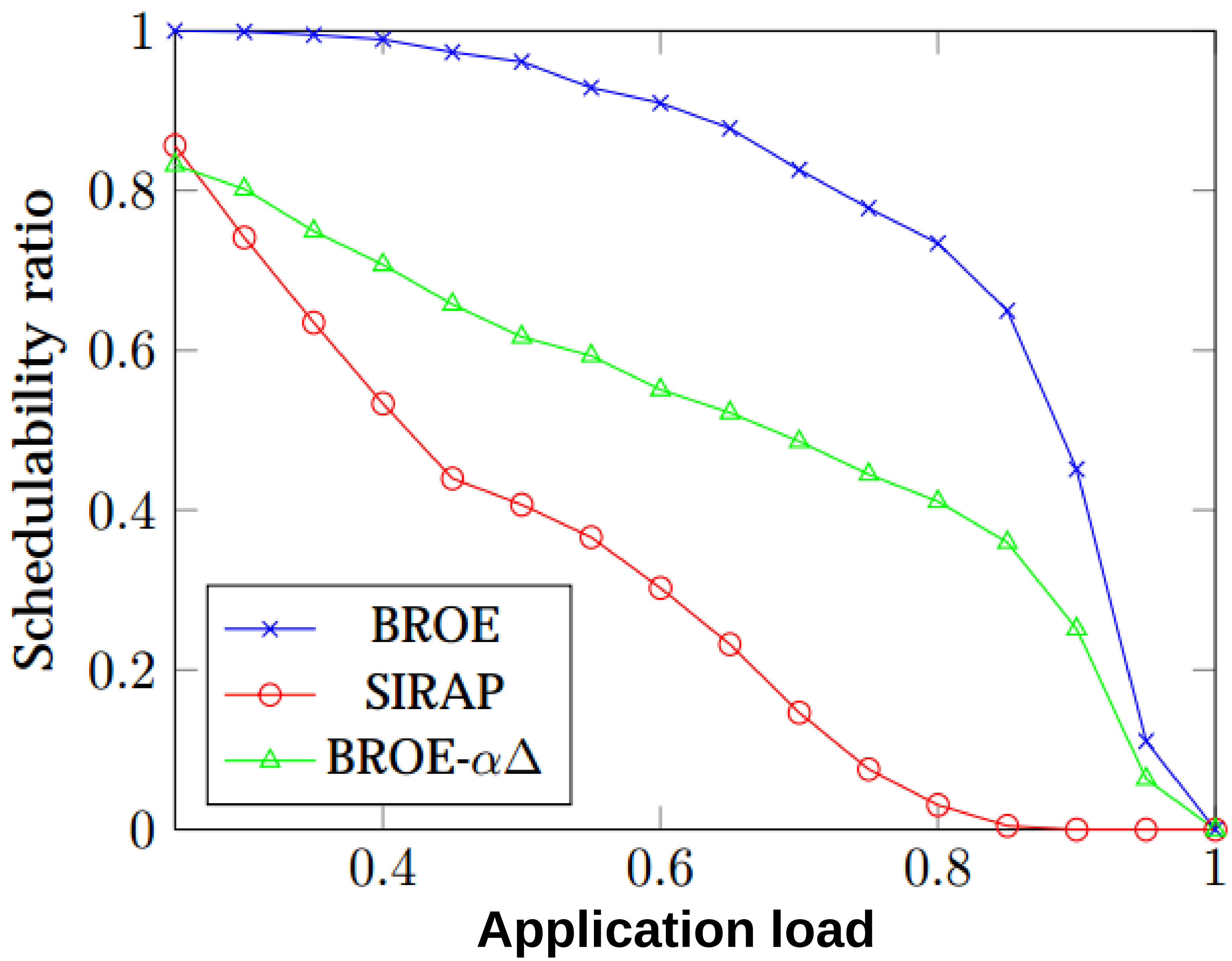
(Biondi et al., 2013)



Application-level schedulability test:
$\forall t \; B^L(t) + dbf(t) \le sbf(t)$

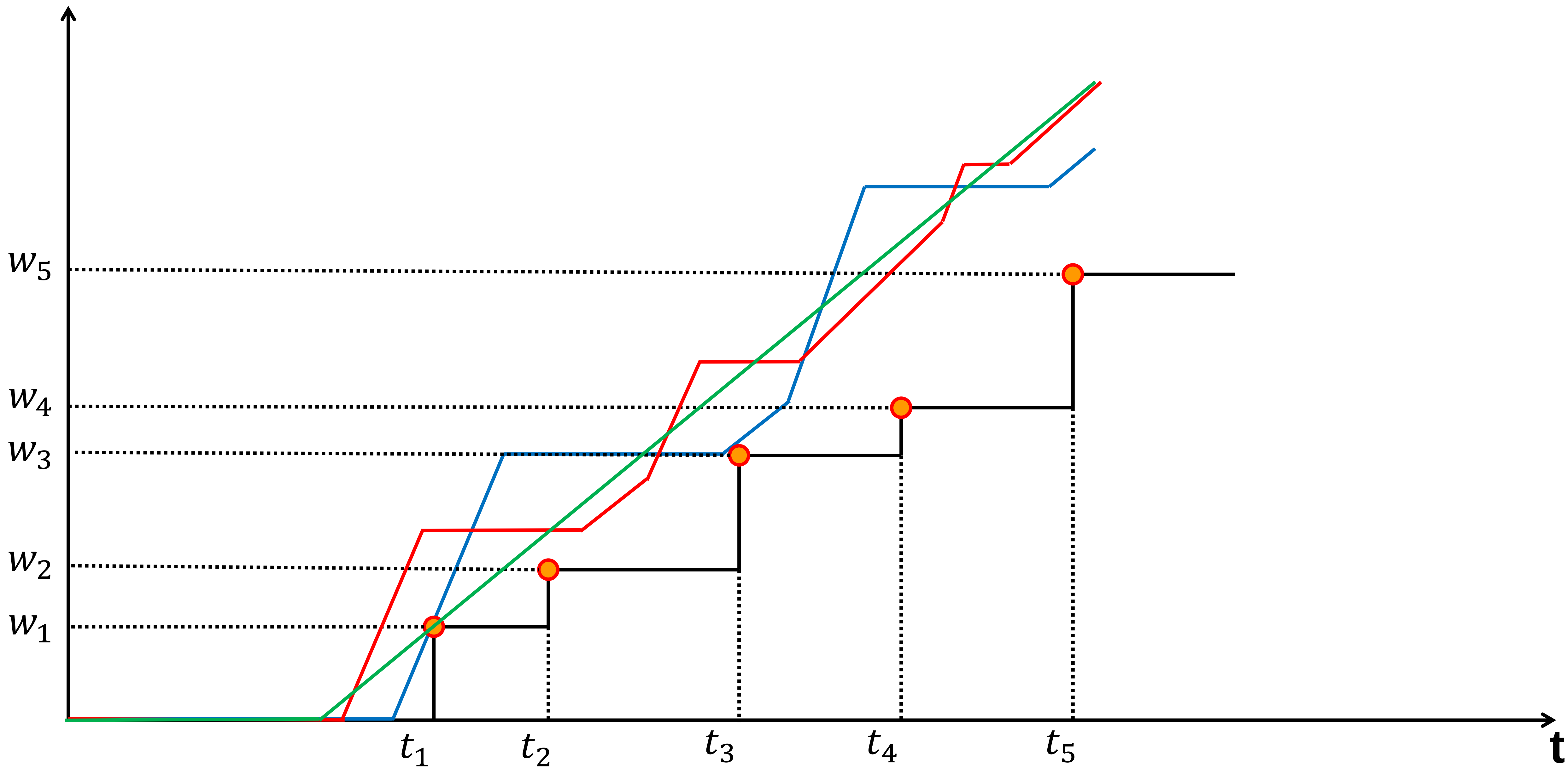
Experimental results

- **EDF** local scheduler, medium critical sections



The problem

Which is the "best" supply bound function that can guarantee a given application?



Our contribution

- We propose a methodology to find the optimal parameters (Q_{opt}, P_{opt}) of a BROE server that minimize its bandwidth requirement α'

$$\alpha' = \frac{\boldsymbol{Q} + \sigma}{\boldsymbol{P}} = \alpha + \frac{\sigma}{\boldsymbol{P}}$$

Context-switch overhead

- Accounting for the context-switch overhead allows discarding trivial solutions

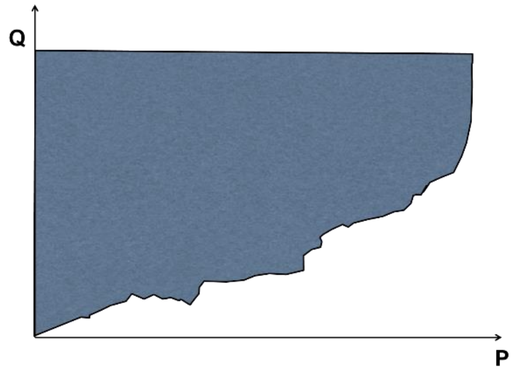
Our model

- *Task model*: each application Γ_k is composed by n_k preemptive tasks $\tau_i = (C_i, D_i, T_i)$, with $D_i \leq T_i$
- *Global scheduler*: BROE (based on Hard-CBS)
- *Local scheduler*: EDF
- *Resource sharing*: SRP-G and SRP-L
- Application interface: $(Q_k, P_k, \boldsymbol{H_k})$
  - $\boldsymbol{H_k}$: maximum Resource Holding Time of application Γ_k

Optimization strategy

- We want to find the **sbf** that gives the minimum bandwidth occupation, satisfying schedulability
- For each point of the dbf:

1. Express schedulability constraint as a function of P and Q



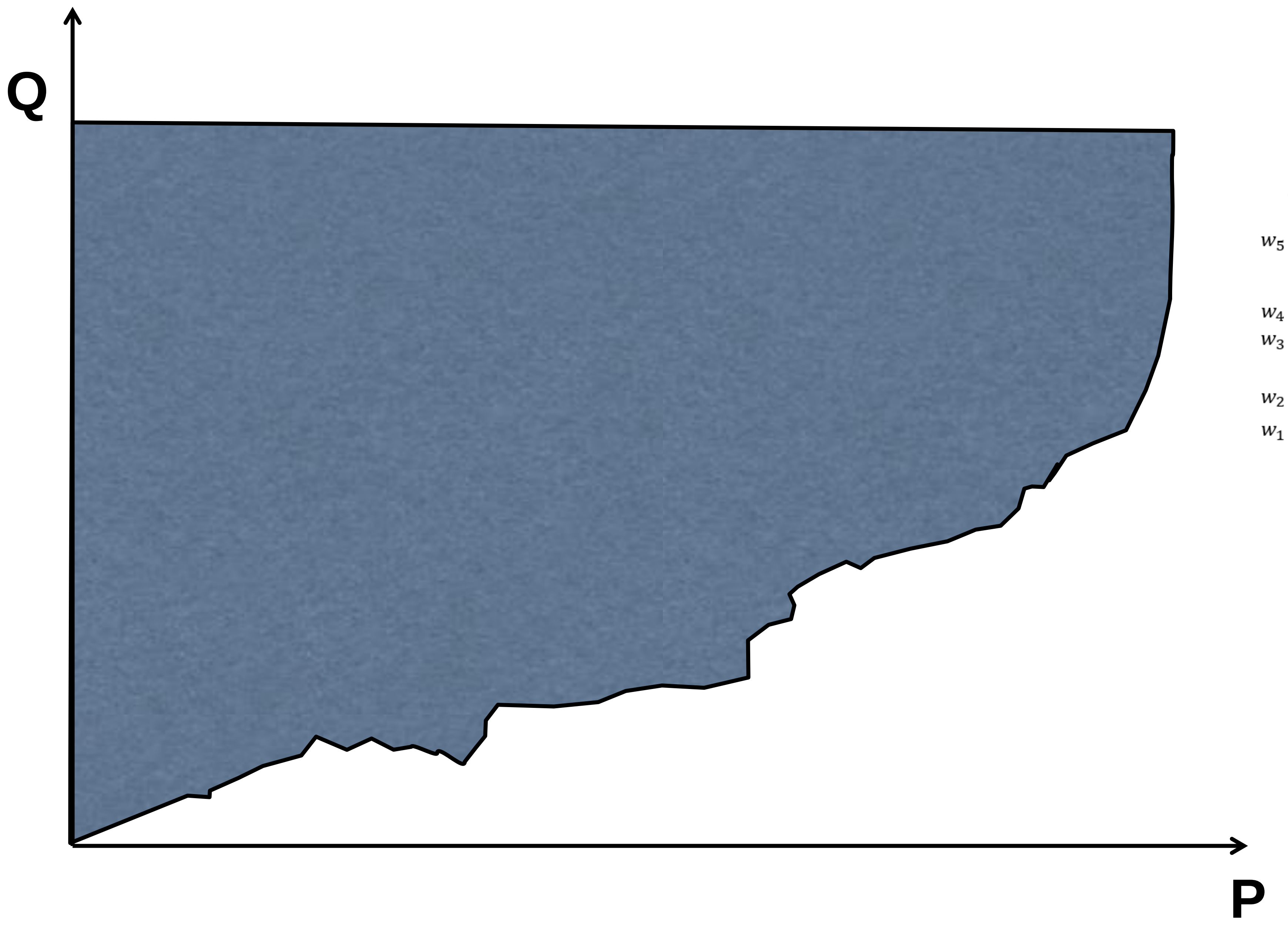
Feasibility region for a single point

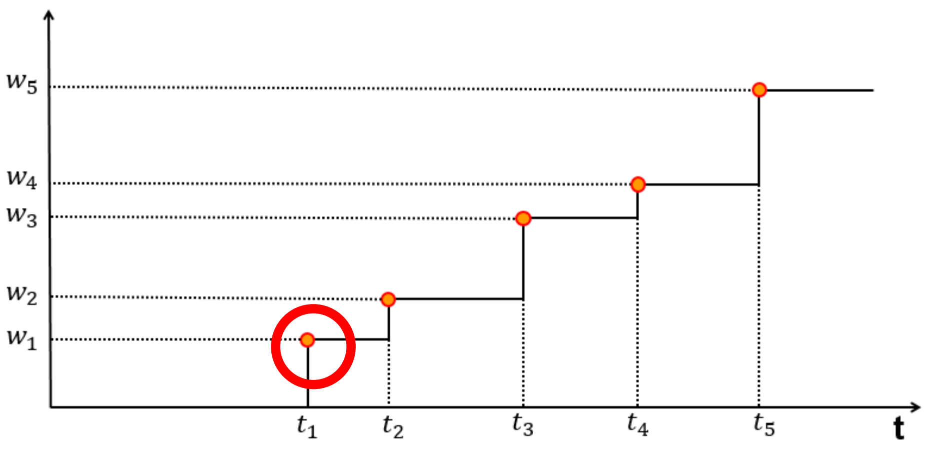
2. Intersect with current feasibility region

$$\mathbb{D} = \bigcap_{d_i \in dSet} \mathbb{D}(w_i, t_i)$$

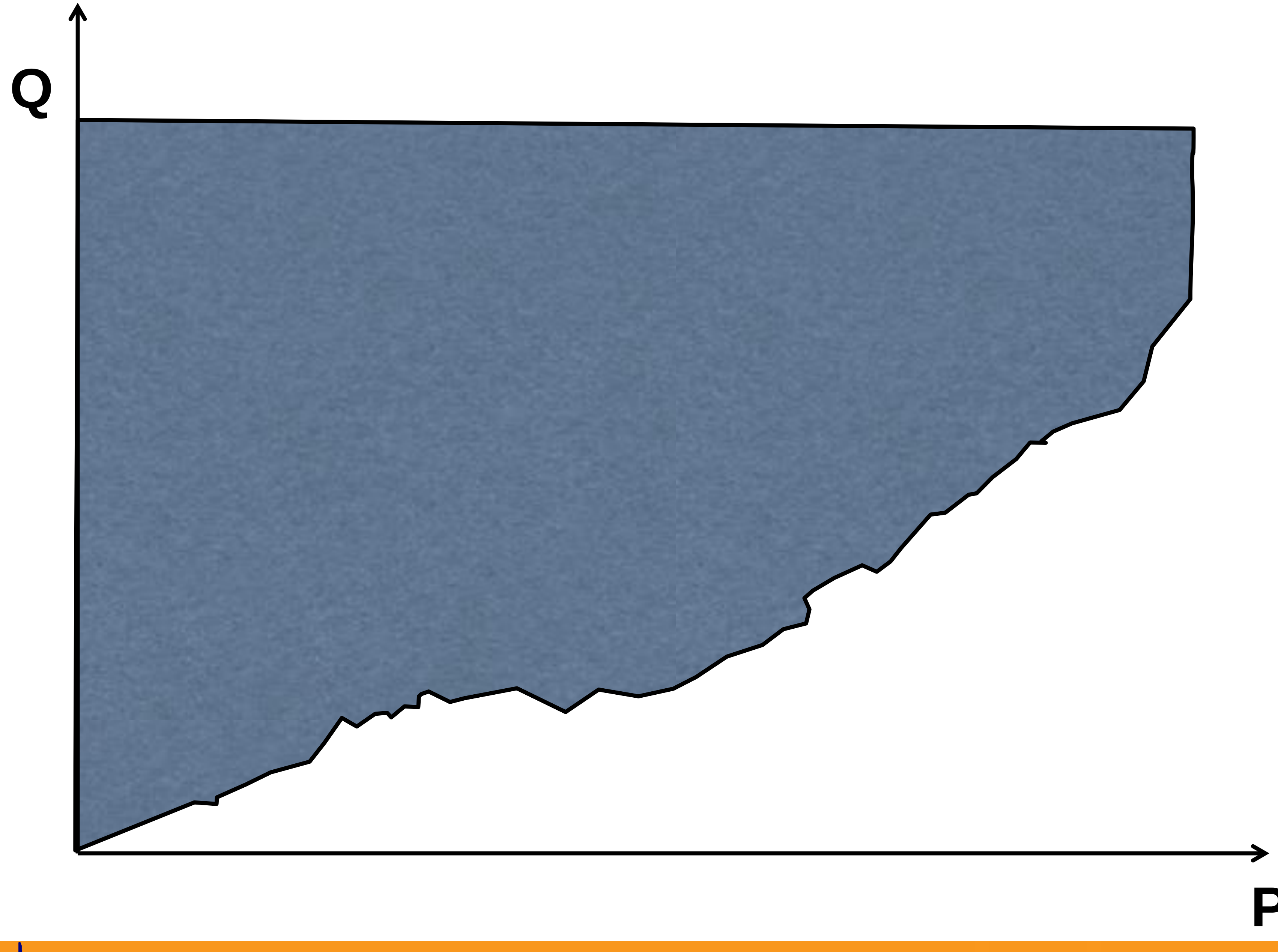
Final feasibility region

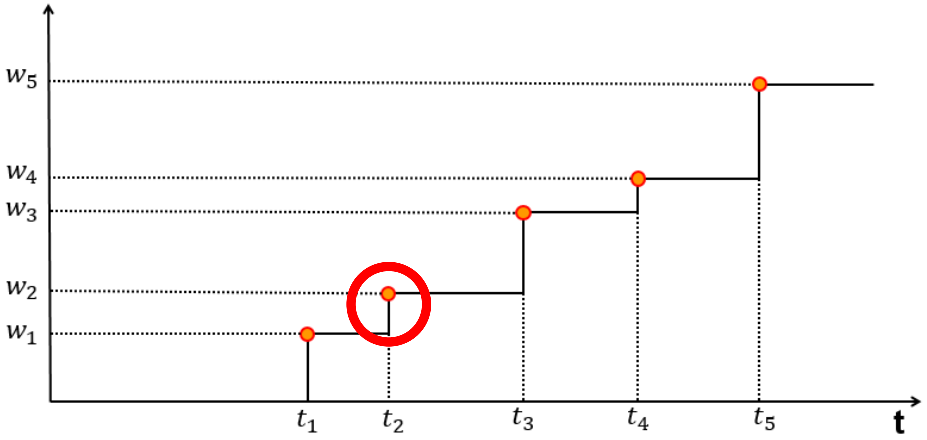
Visual example(1)



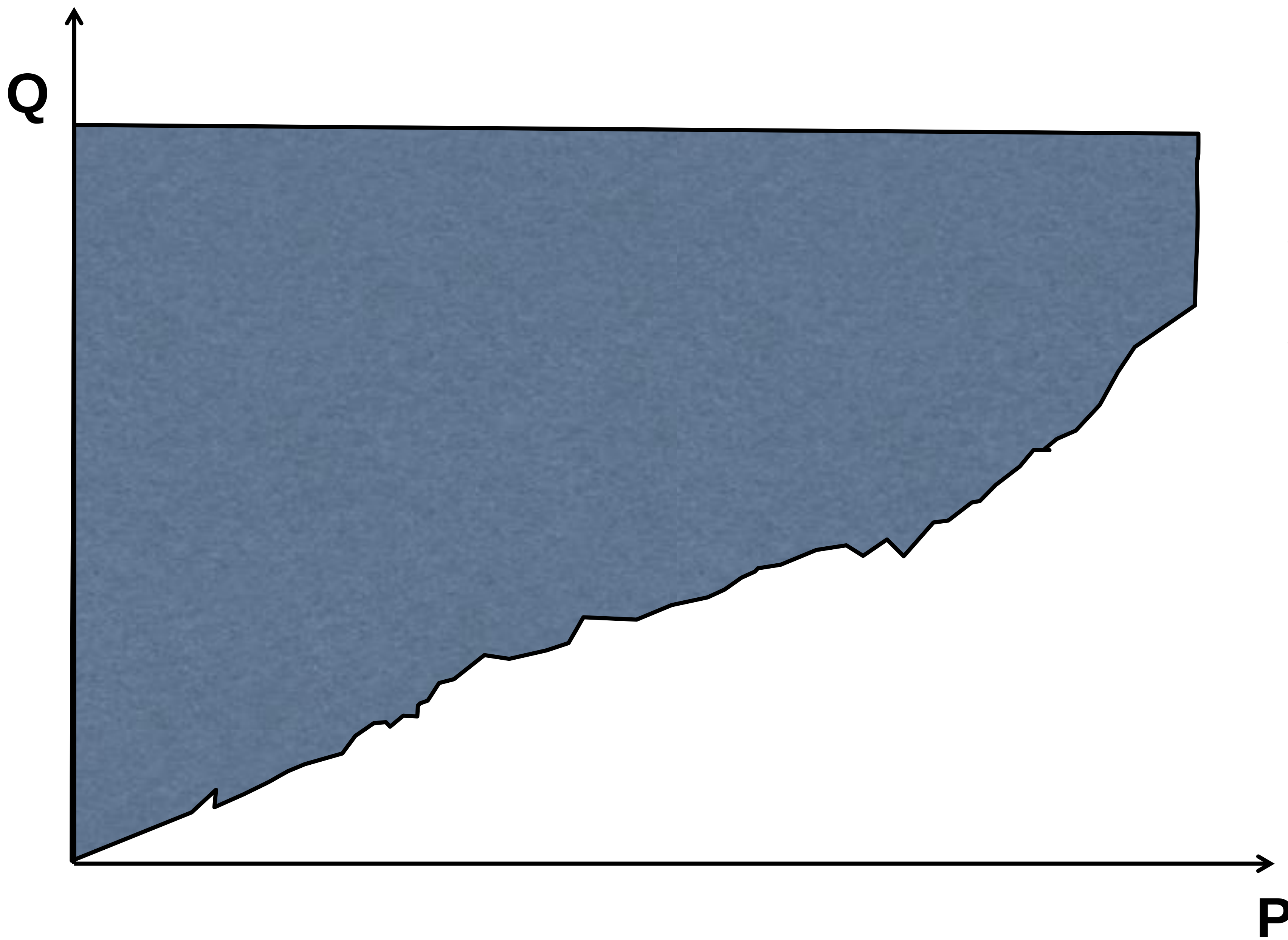


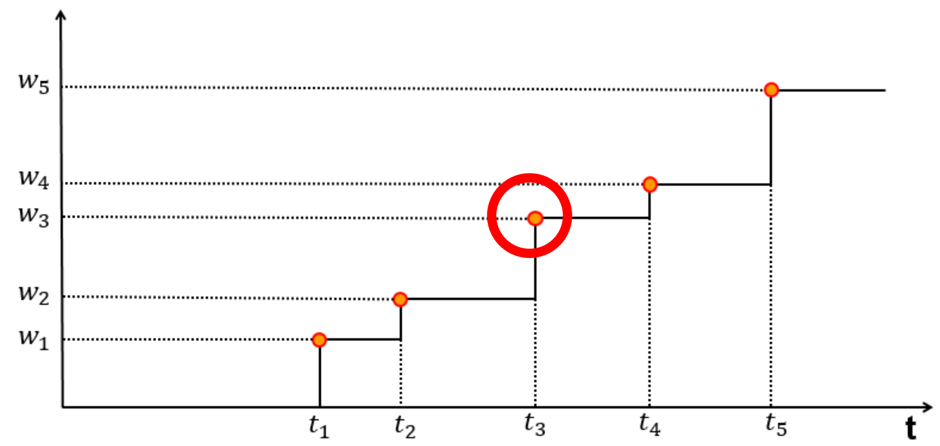
Visual example(2)





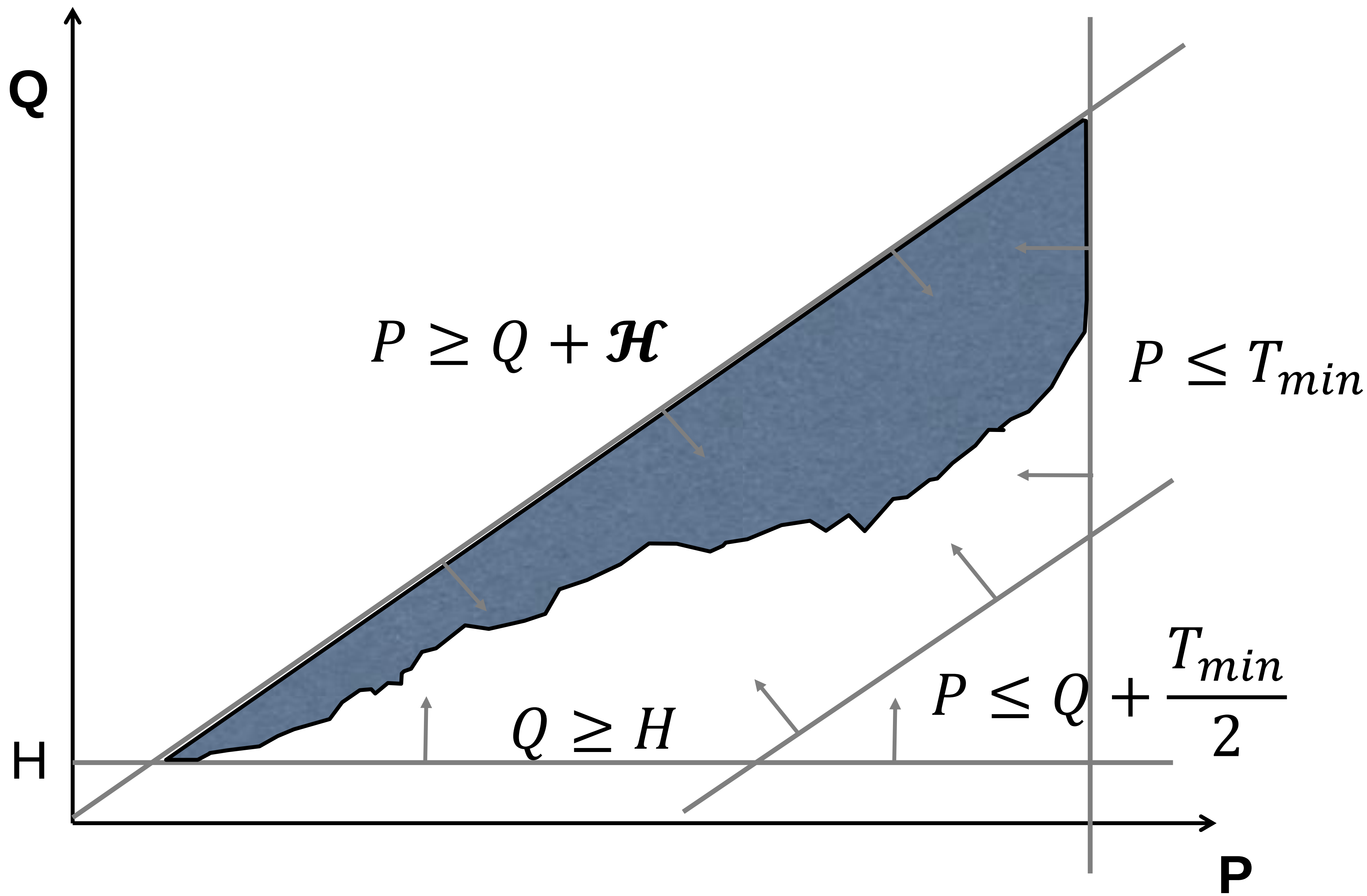
Visual example(3)



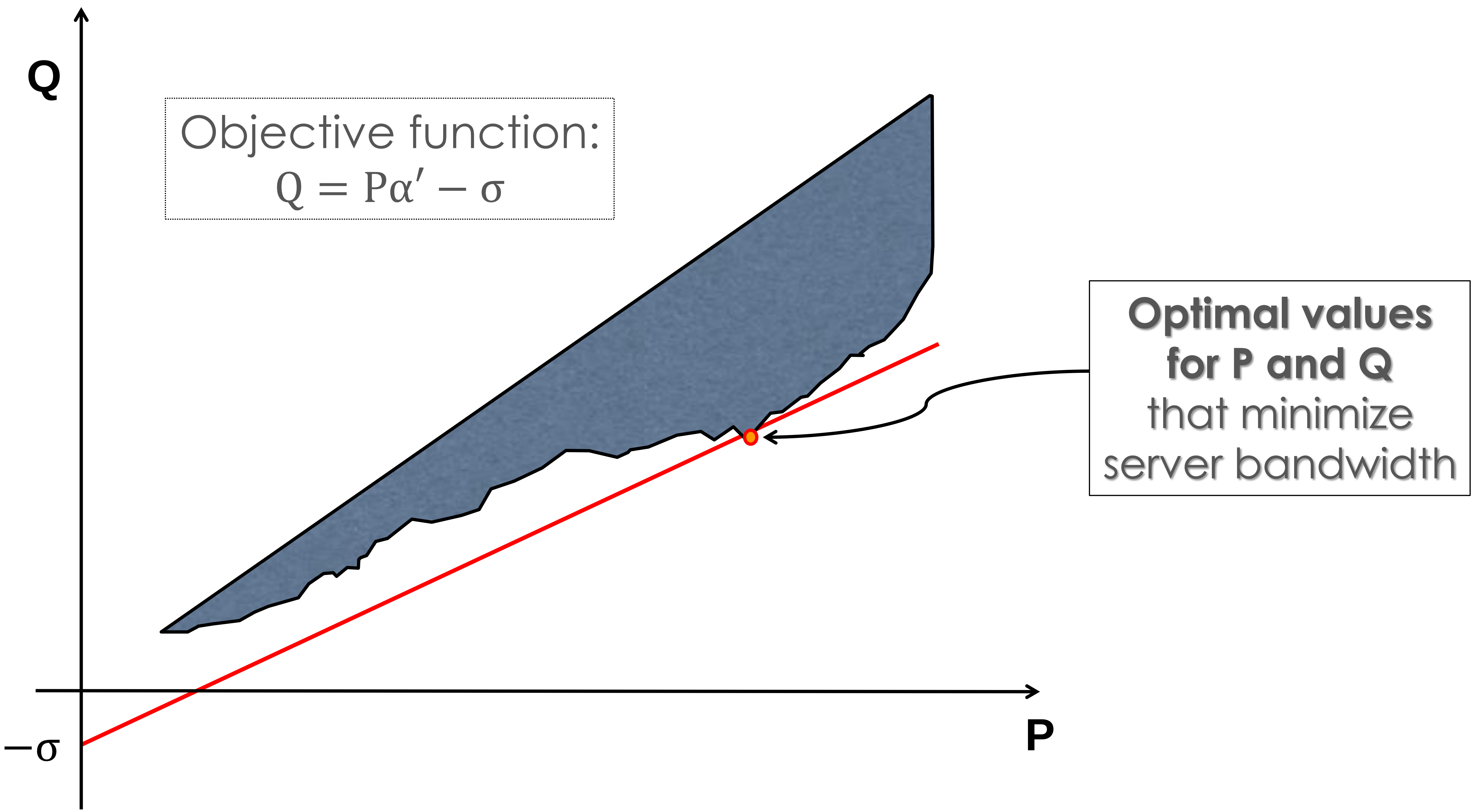


How to refine the search

- Some generic constraints can refine the feasibility region

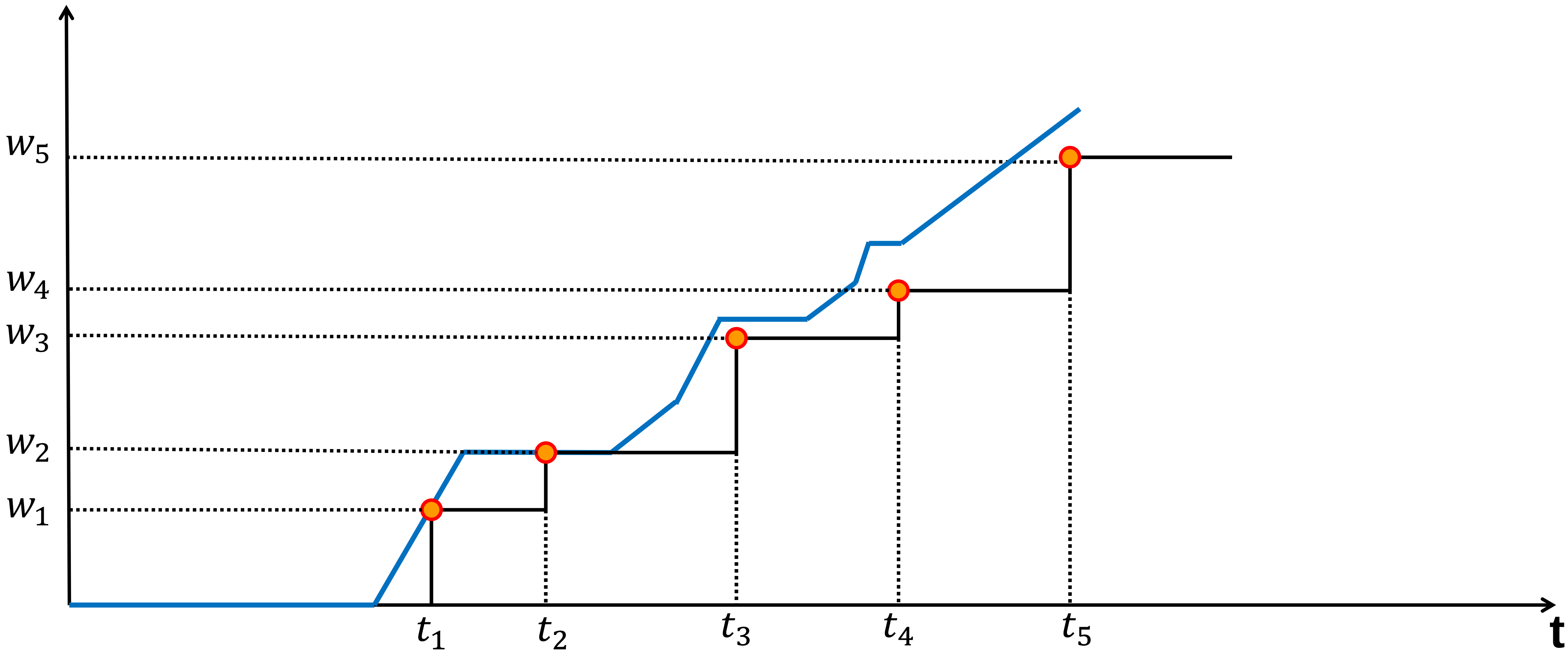


How to find the optimum



Final result

- The optimal parameters (Q_{opt}, P_{opt}) determine the optimal supply bound function for the application



Implementation

- The design algorithm has been implemented as MATLAB code and is publicly available at:

  http://retis.sssup.it/~a.biondi/optBROE

Conclusions

- We have proposed a methodology to design a BROE server with parameters (Q_{opt}, P_{opt}) that minimize its bandwidth, ensuring:
  - **Pseudo-polynomial complexity**
  - A **standard interface** for each application
- Using this approach, BROE dominates all other existing solutions
- As future work, we plan to extend it to local fixed priority scheduling and to multi-core platforms

Thank you!

Alessandra Melani
alessandra.melani@sssup.it