A Workload-oriented Programming Model for Temporal Isolation with VBS

Ana Sokolova

Department of Computer Sciences
University of Salzburg

joint work with
Silviu Craciunas and Christoph Kirsch

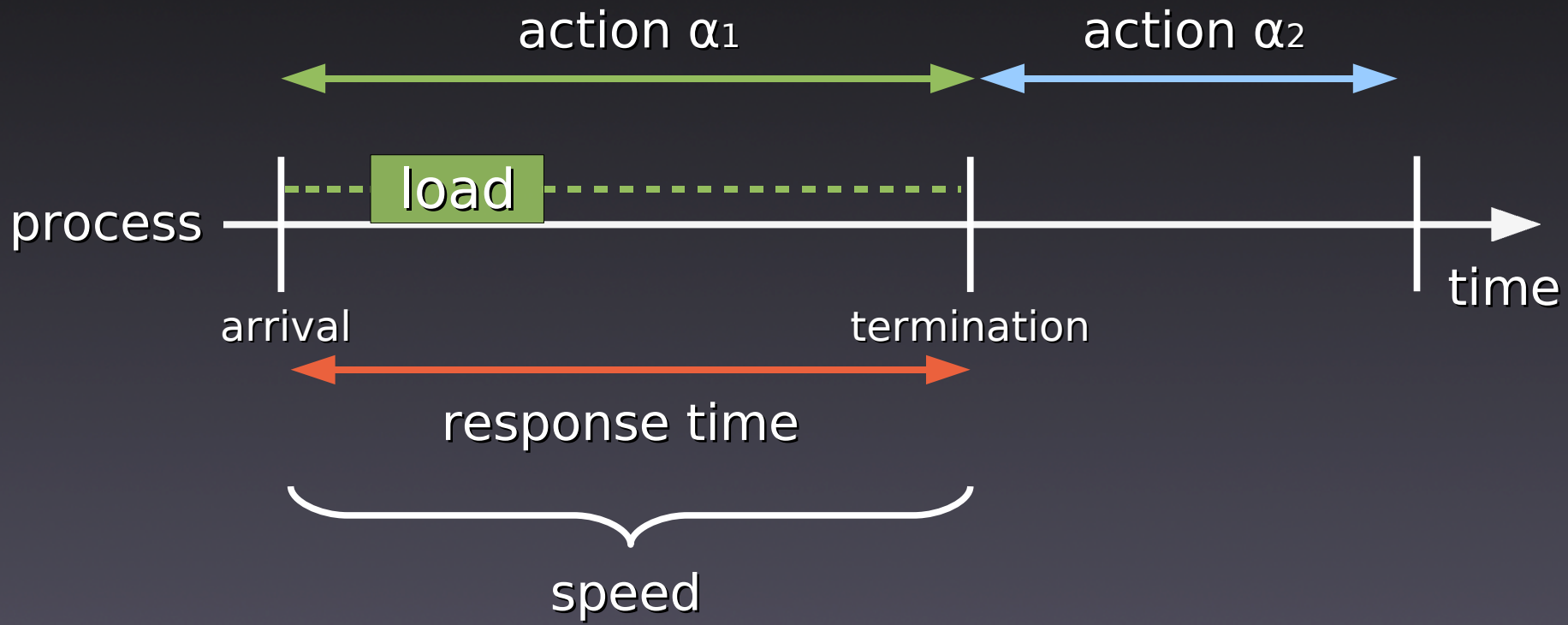
It is about

- Scheduling processes in temporal isolation

> the time it takes to execute a piece of code is bounded, independently of concurrently running code

- Using variable bandwidth servers for predictability
- Server design for performance

Process model



- action is a piece of code
- process is a sequence of actions

Problem

schedule the processes so that each of their actions maintains its response time

Problem

process 1

proces

proces

ocessor

Solvable with variable bandwidth servers (VBS)

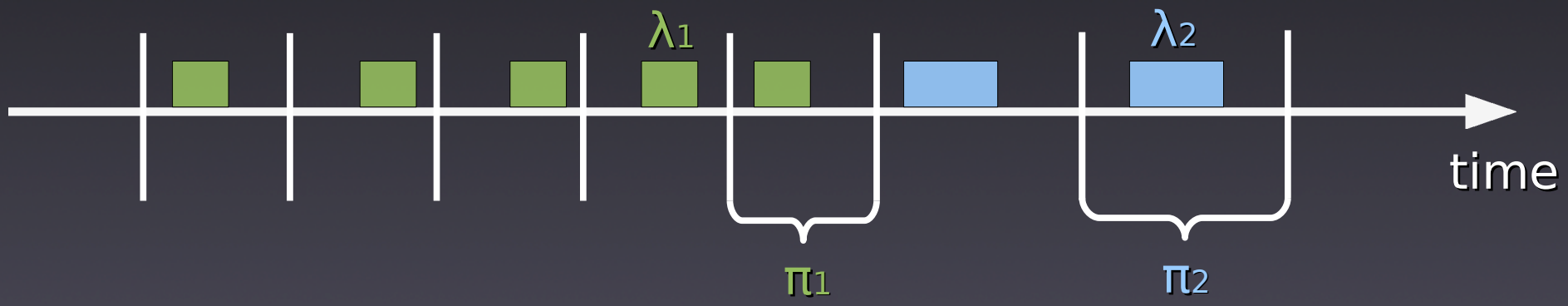
Results:
- a constant-time scheduling algorithm
- a constant-time admission test

schedule the processes so that each of their actions maintains its response time

Resources and VBS

virtual periodic resources

period π limit λ utilization λ/π



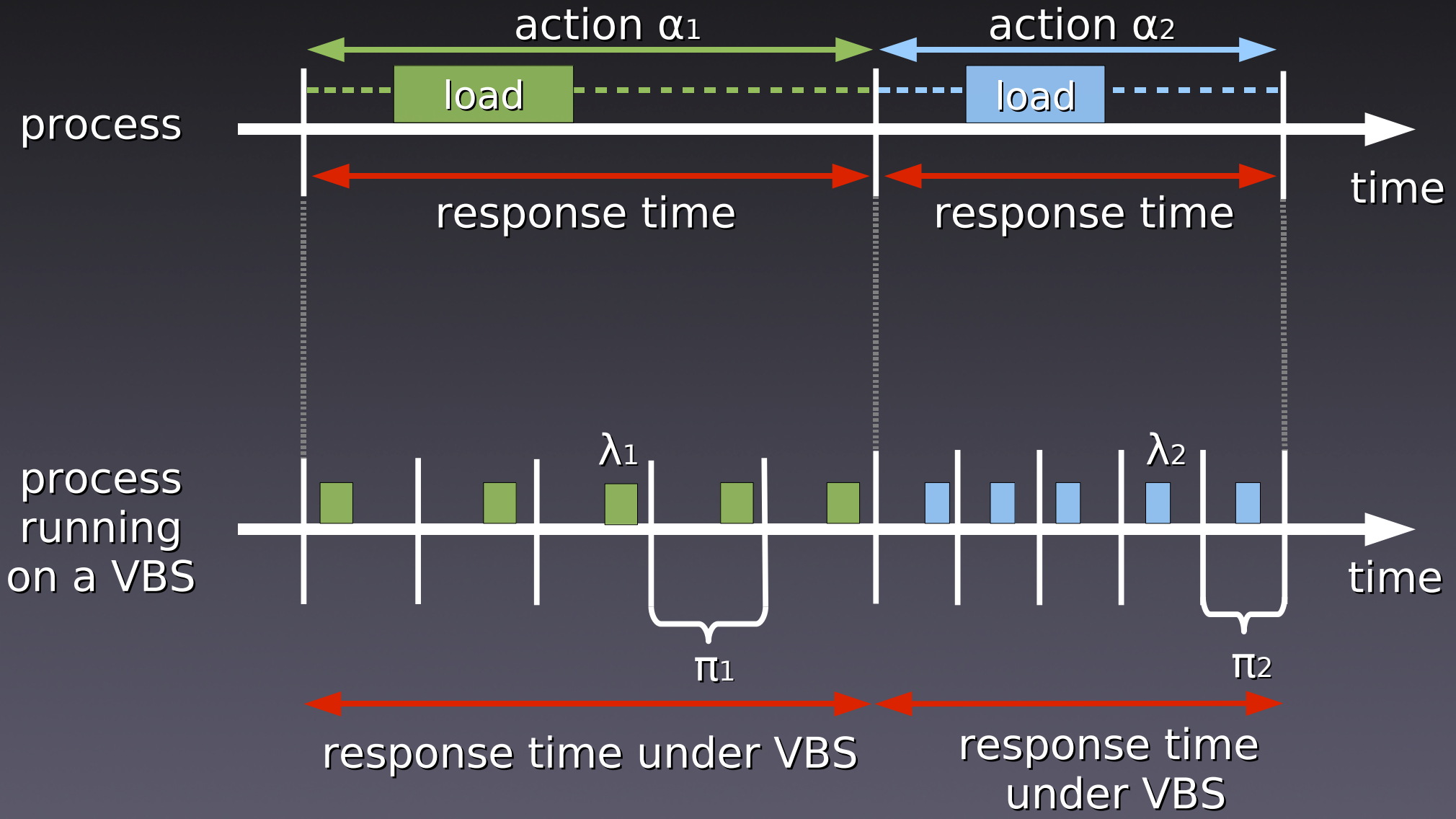
- VBS is determined by a bandwidth cap (u)
- VBS processes dynamically adjust speed (resource)

$$\lambda_1/\pi_1 \leq u \text{ and } \lambda_2/\pi_2 \leq u$$

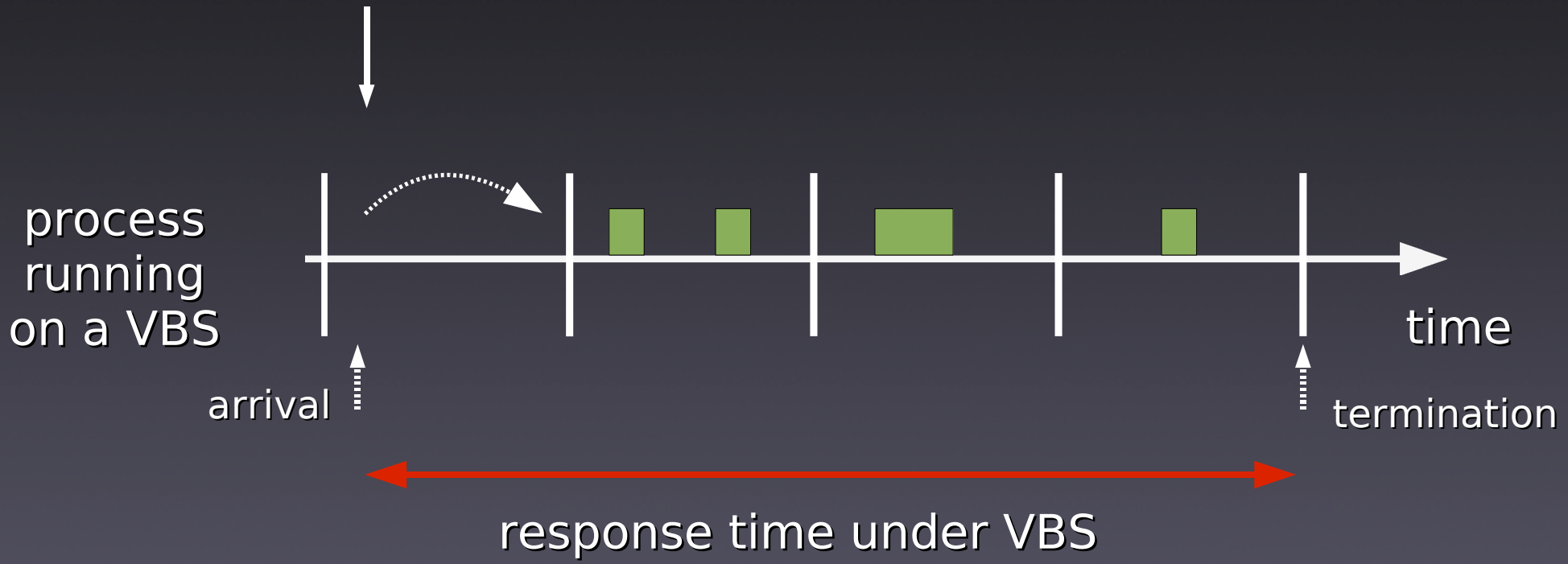
- generalization of constant bandwidth servers (CBS)

[Abeni and Buttazzo 2004]

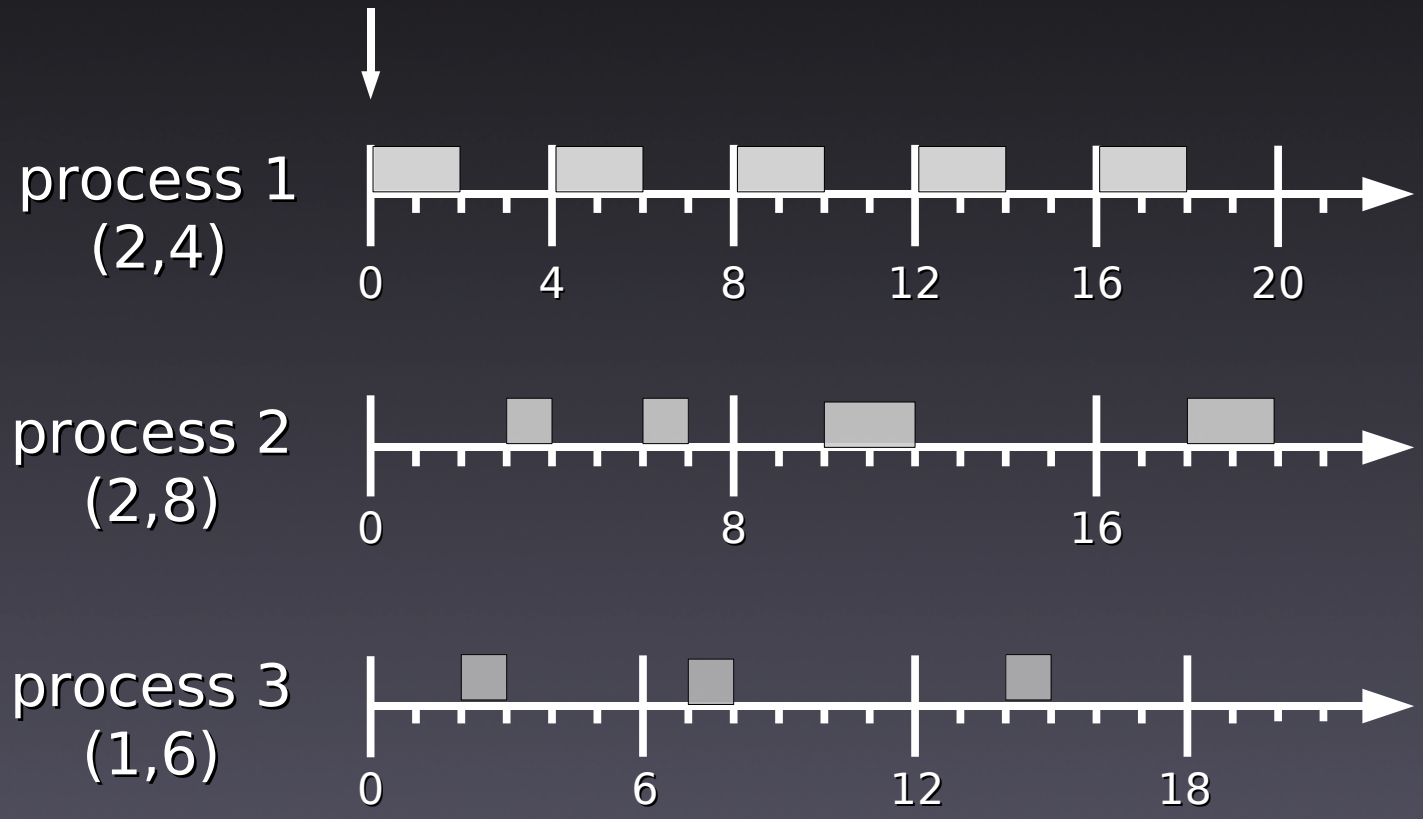
One process on a VBS



VBS


process
running
on a VBS
time
arrival
termination
response time under VBS

VBS



multiple processes are EDF-scheduled

Scheduling result and bounds

Processes $P_1, P_2, \ldots, P_n$ on VBSs $u_1, u_2, \ldots, u_n$, are schedulable

if $\sum u_i \leq 1$

For any action α on a resource (λ, π) we have

upper response time bound
$\lceil load / \lambda \rceil \pi + \pi - 1$

lower response time bound
$\lceil load / \lambda \rceil \pi$

jitter
$\pi - 1$

Implementation

- constant-time scheduling algorithm
- different queue management plugins

	list	array	matrix/tree
time	$O(n^2)$	$O(\log(t)+nlog(t))$	$\Theta(t)$
space	$\Theta(n)$	$\Theta(t+n)$	$O(t^2+n)$

n – number of processes t – number of time instants

Implementation

- constant-time scheduling algorithm
- different queue management plugins

trade-off time and space

	list	array	matrix/tree
time	$O(n^2)$	$O(\log(t)+n\log(t))$	$\Theta(t)$
space	$\Theta(n)$	$\Theta(t+n)$	$O(t^2+n)$

n – number of processes t – number of time instants

Programming model

```
loop {

    int number_of_frames=determine_rate();

    allocate_memory(number_of_frames);
    read_from_network(number_of_frames);

    compress_data(number_of_frames);
    write_to_disk(number_of_frames);
    deallocate_memory(number_of_frames);

} until (done);
```

action 1

action 2

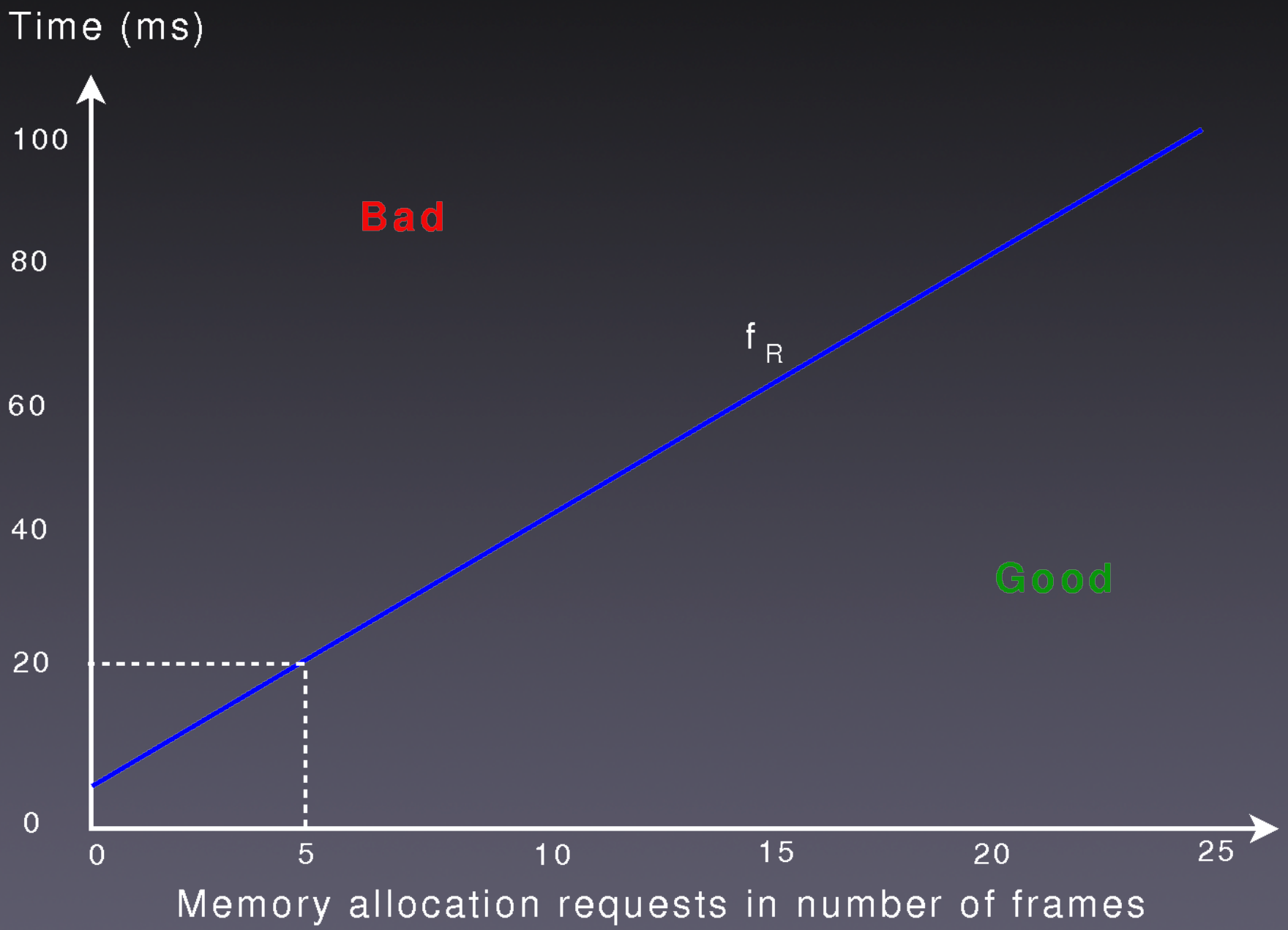
loop period

different throughput and latency requirements for different portions of code

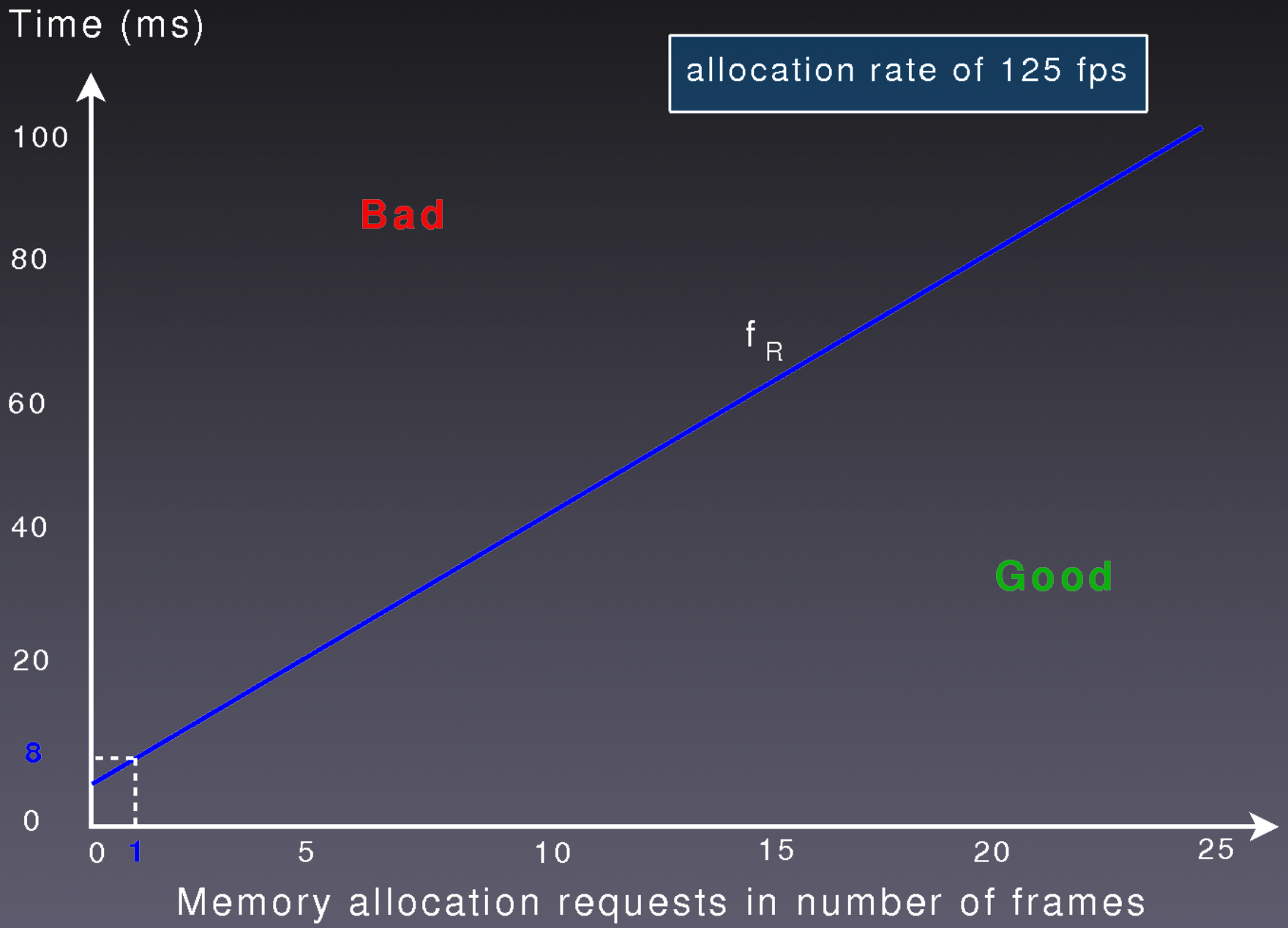
How do we get VBS parameters for an action?

“server design problem”

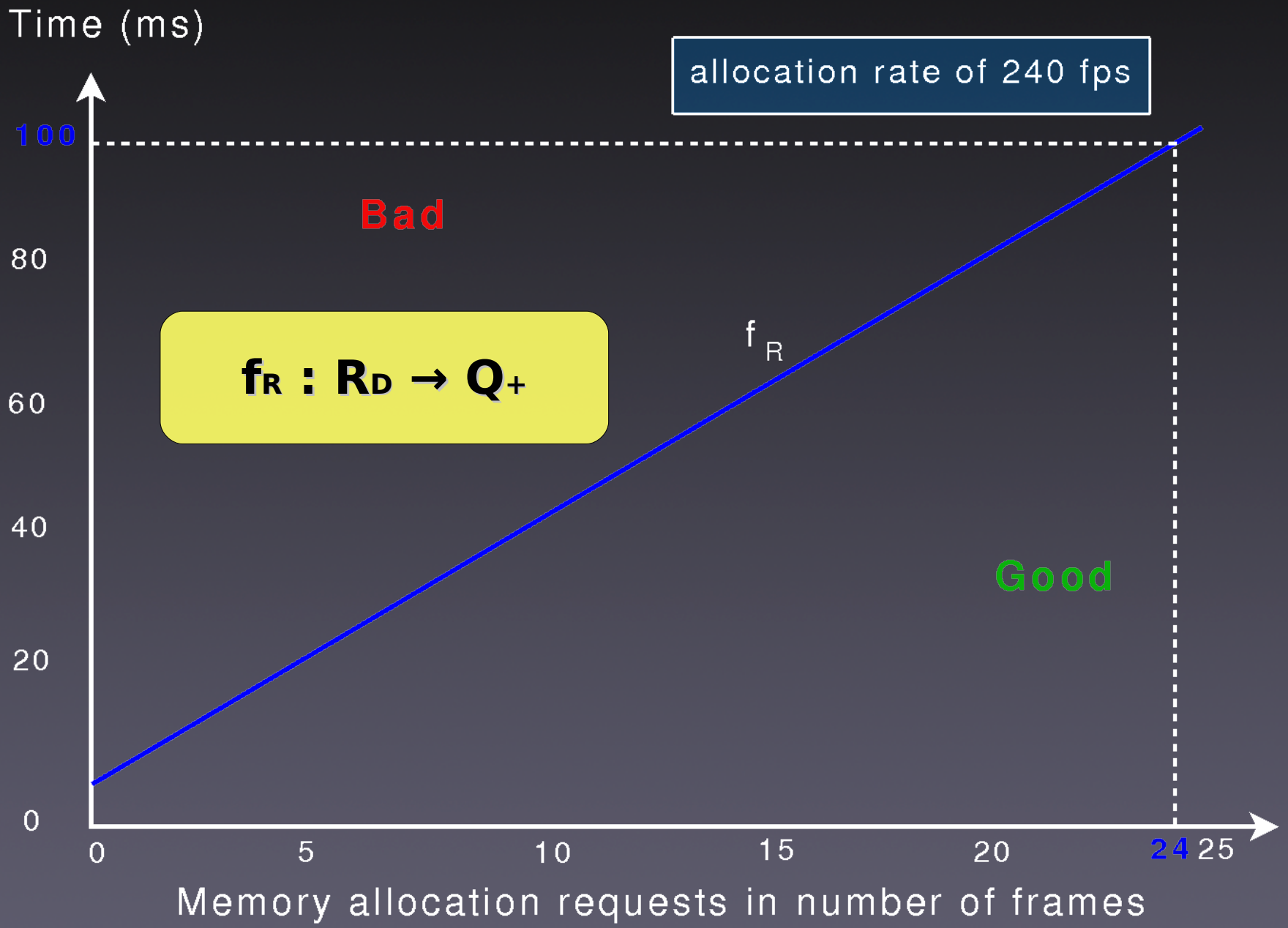
Response-time function



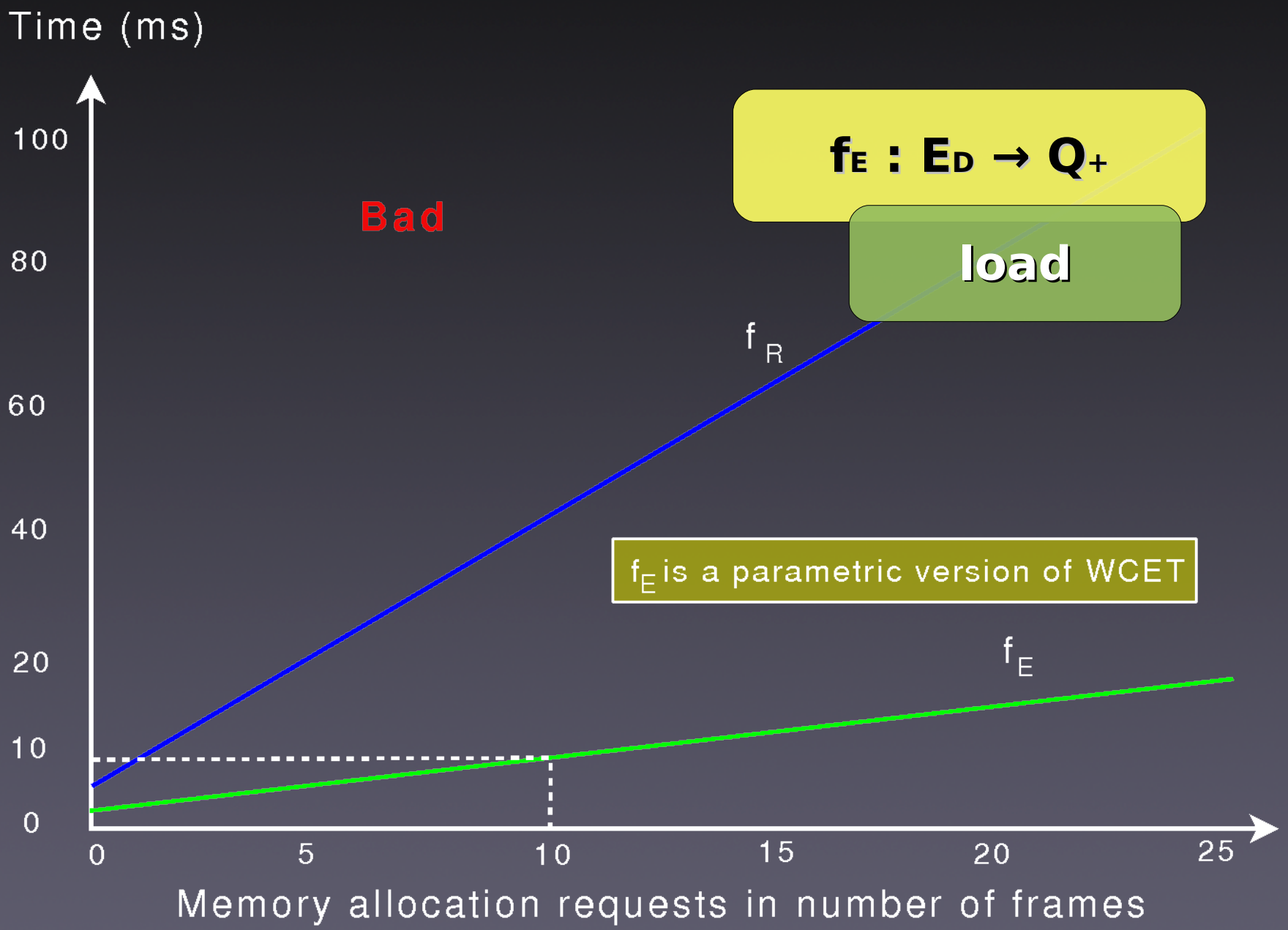
Response-time function



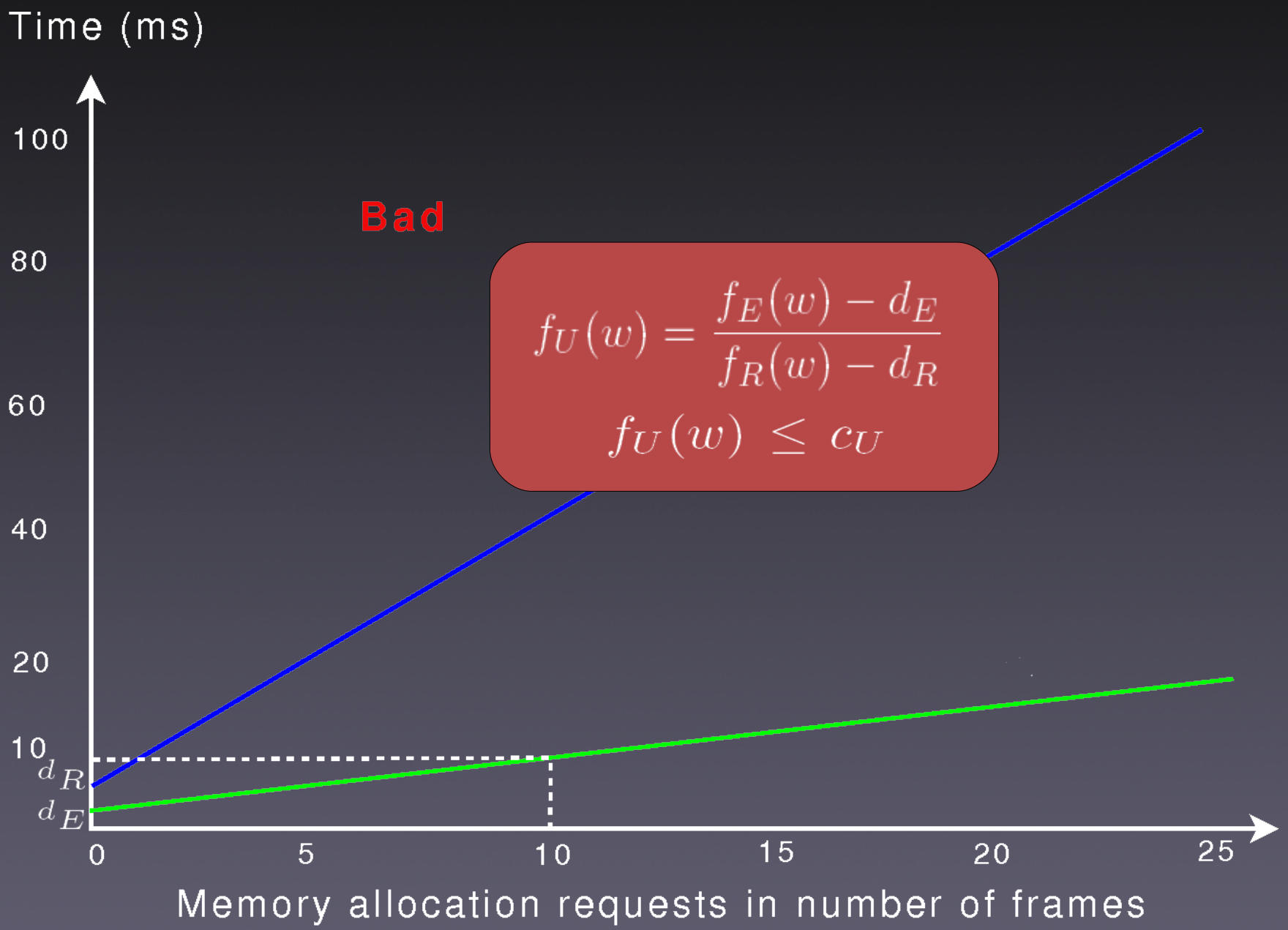
Response-time function



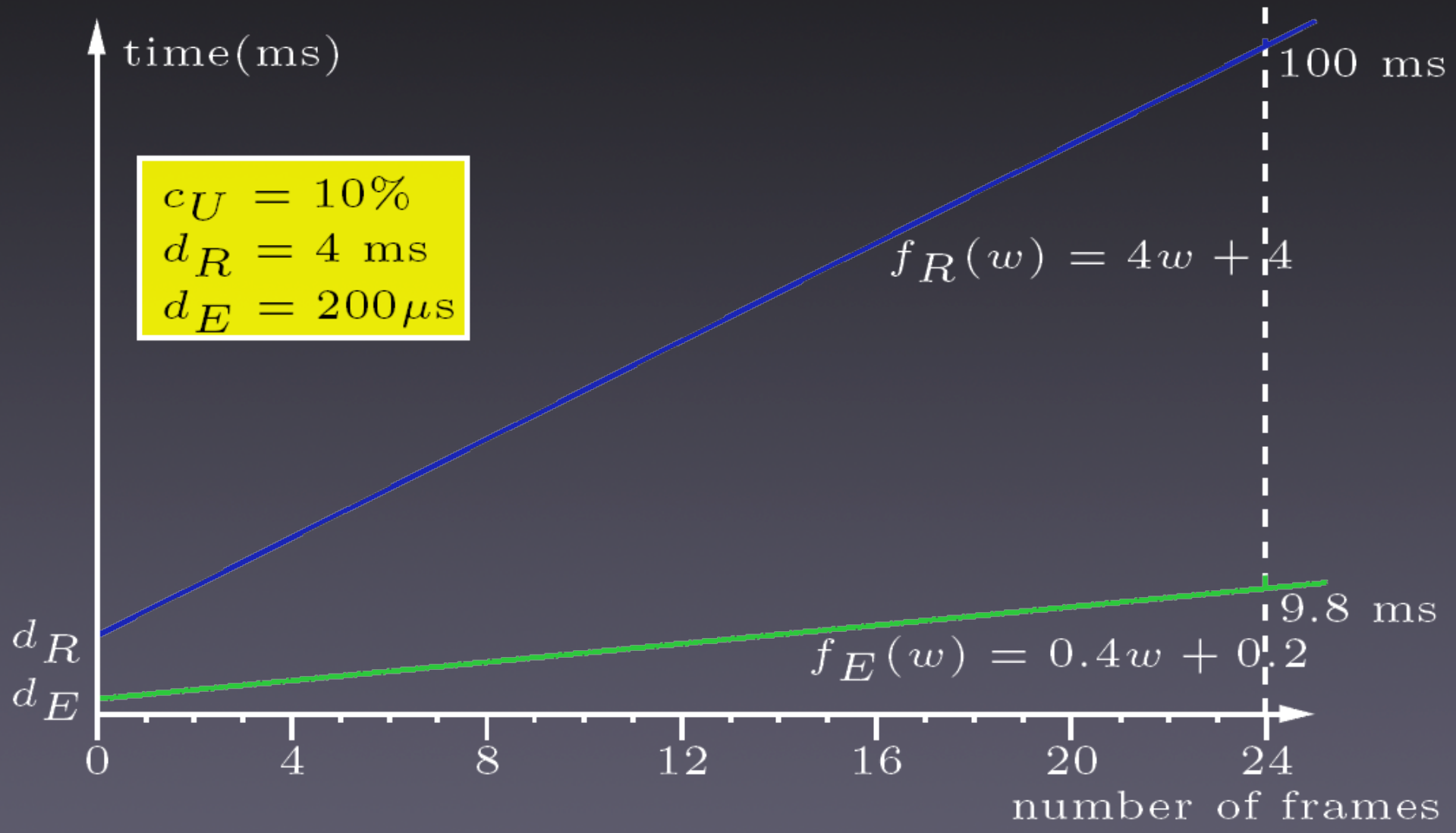
Execution-time function



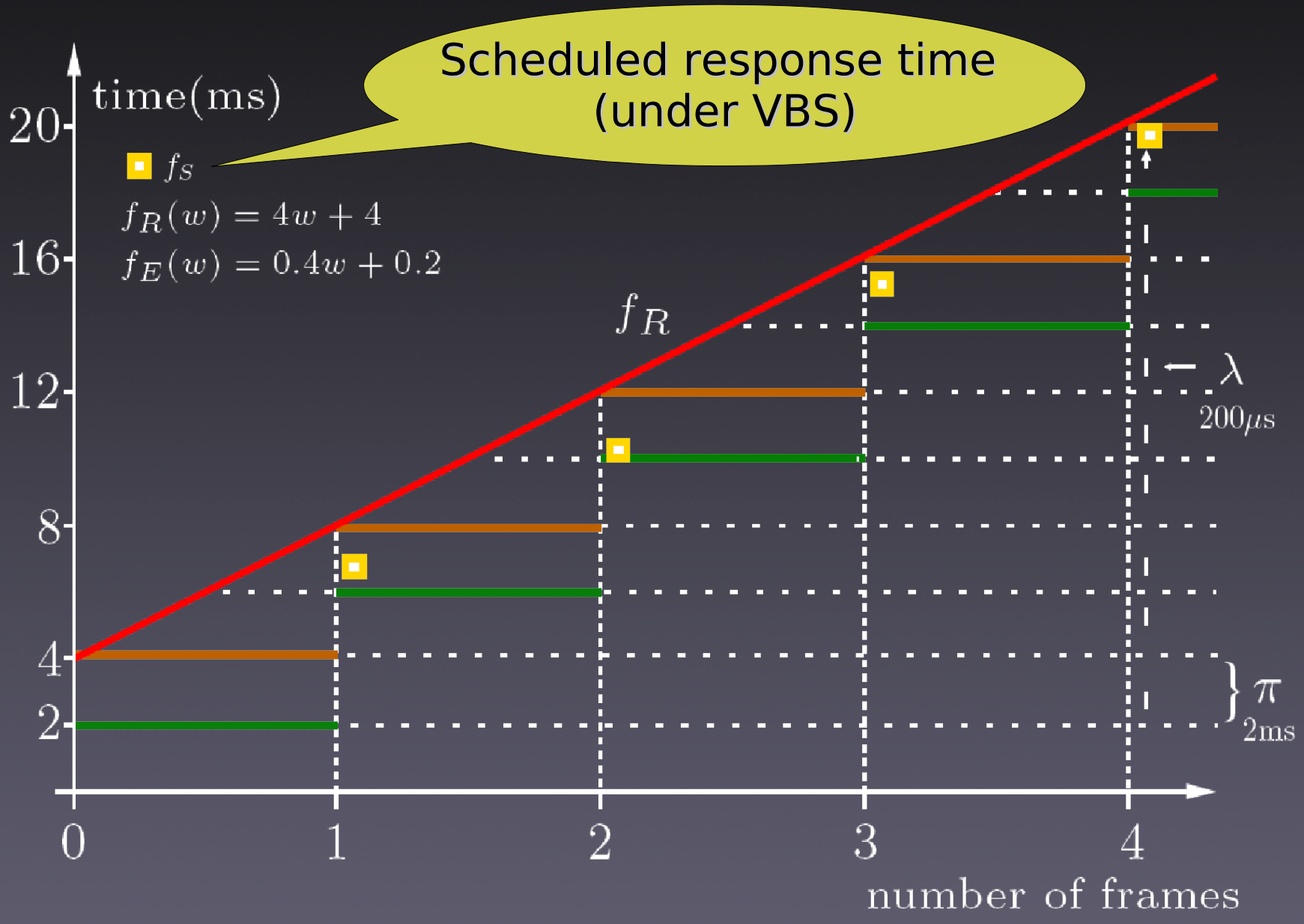
Utilization



Timing of the allocate_memory action



Response-time sampling



Server Design Problem

Finding the right λ,π is difficult.

For $f_S(w) \leq f_R(w)$ one can choose π as follows:

- $0 < \pi < d_R - d_E / c_U$
- π divides d_R evenly
- π divides $f_R(w) - d_R$ evenly or

  λ divides $f_E(w) - d_E$ evenly

$\lambda = \pi * c_U$

Server Design Problem

Smallest π possible:

- f_S approximates f_R best 
- less response-time jitter 
- increased scheduler overhead 

Scheduler overhead accounting:

- utilization accounting 
- response-time accounting 
- combined accounting

Higher-level scheduler:

- small period for the first part of an action
- large period for the remaining part

Conclusion

For scheduling processes in temporal isolation:

- Programming model as a link to VBS
- VBS provide predictability
- Server design for better performance

http://tiptoe.cs.uni-salzburg.at/