



**TECHNISCHE
UNIVERSITÄT
DRESDEN**

Department of Computer Science Institute of System Architecture, Operating Systems Group

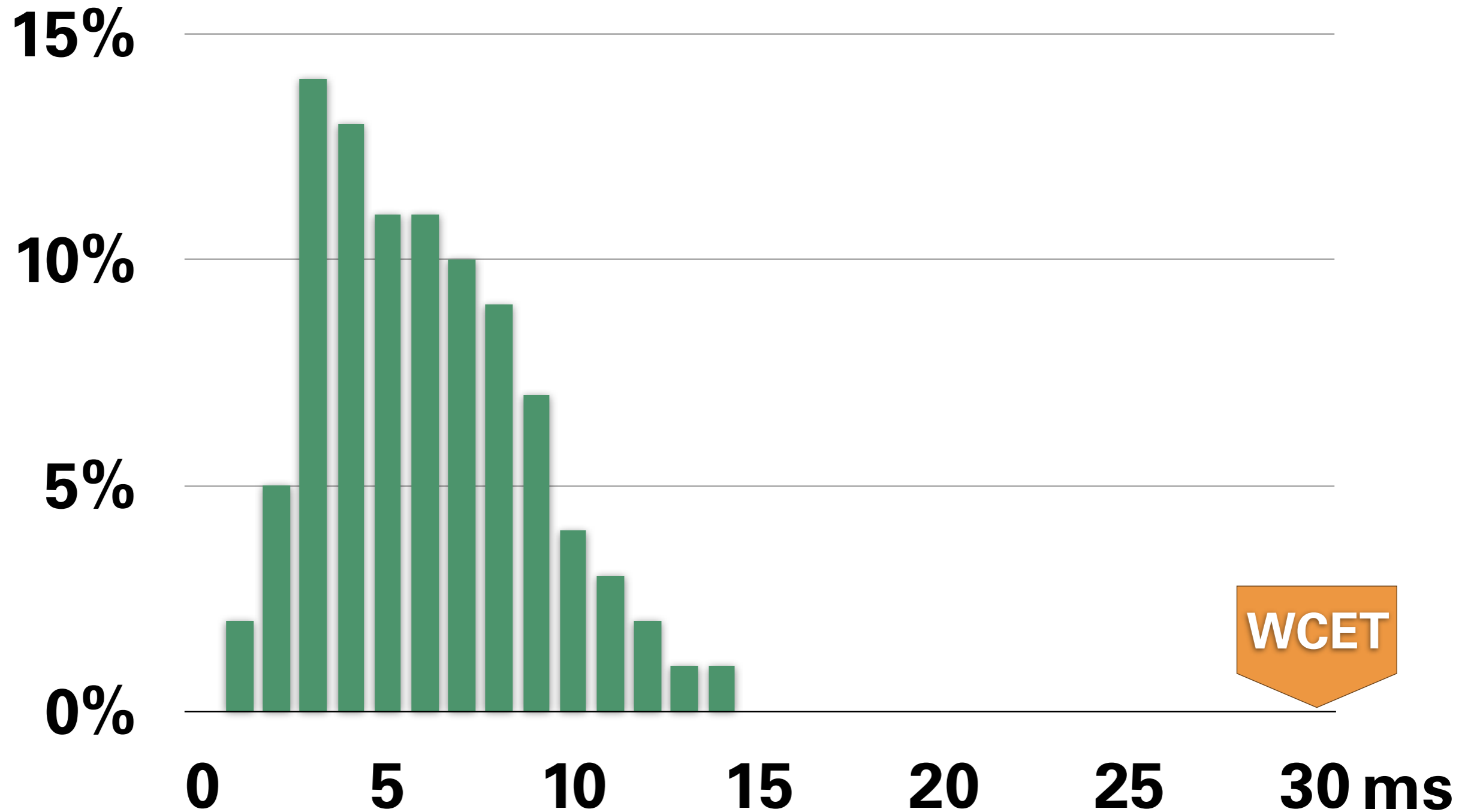
PROBABILISTIC SCHEDULING

MICHAEL ROITZSCH

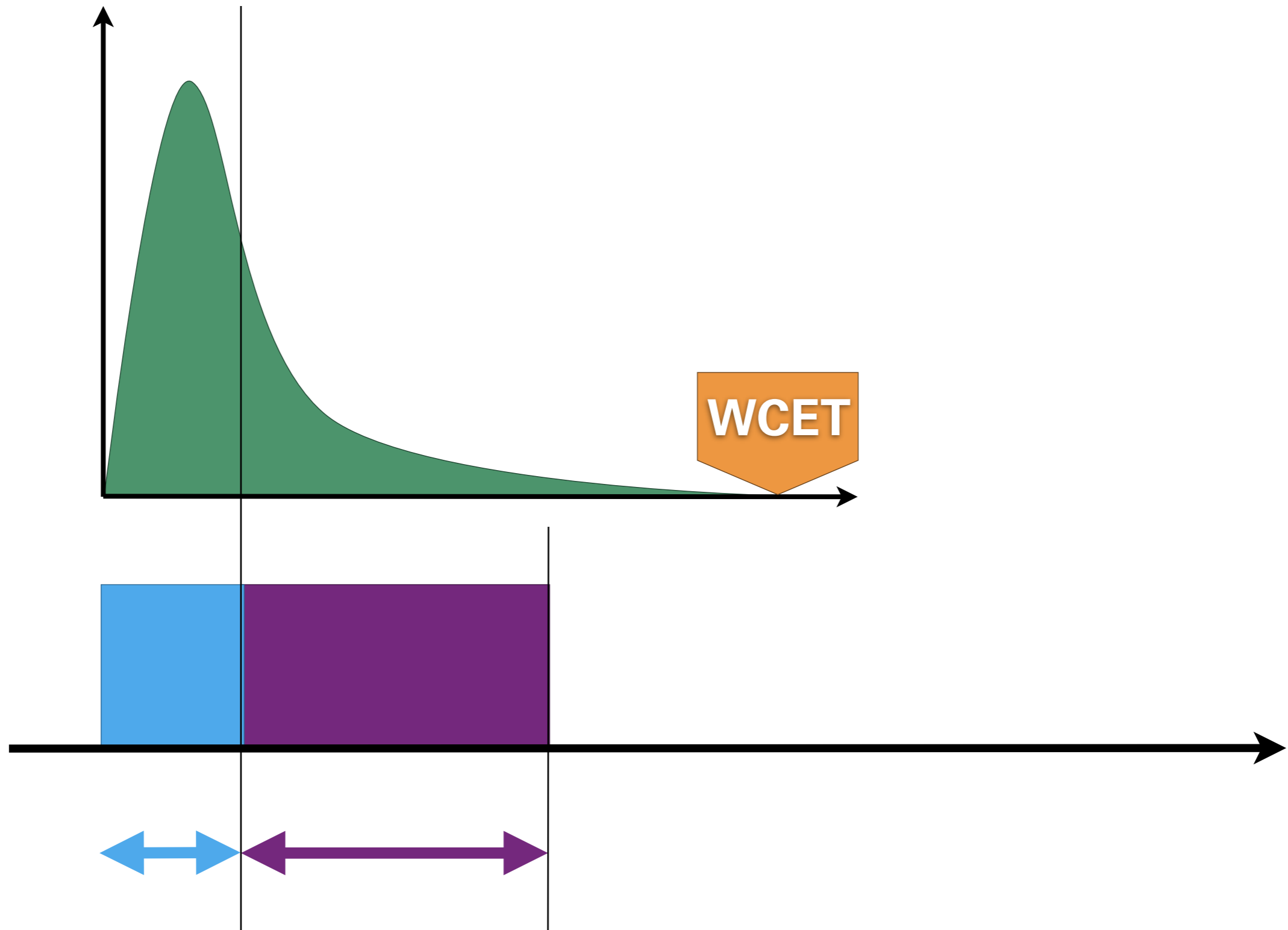
DESKTOP REAL-TIME

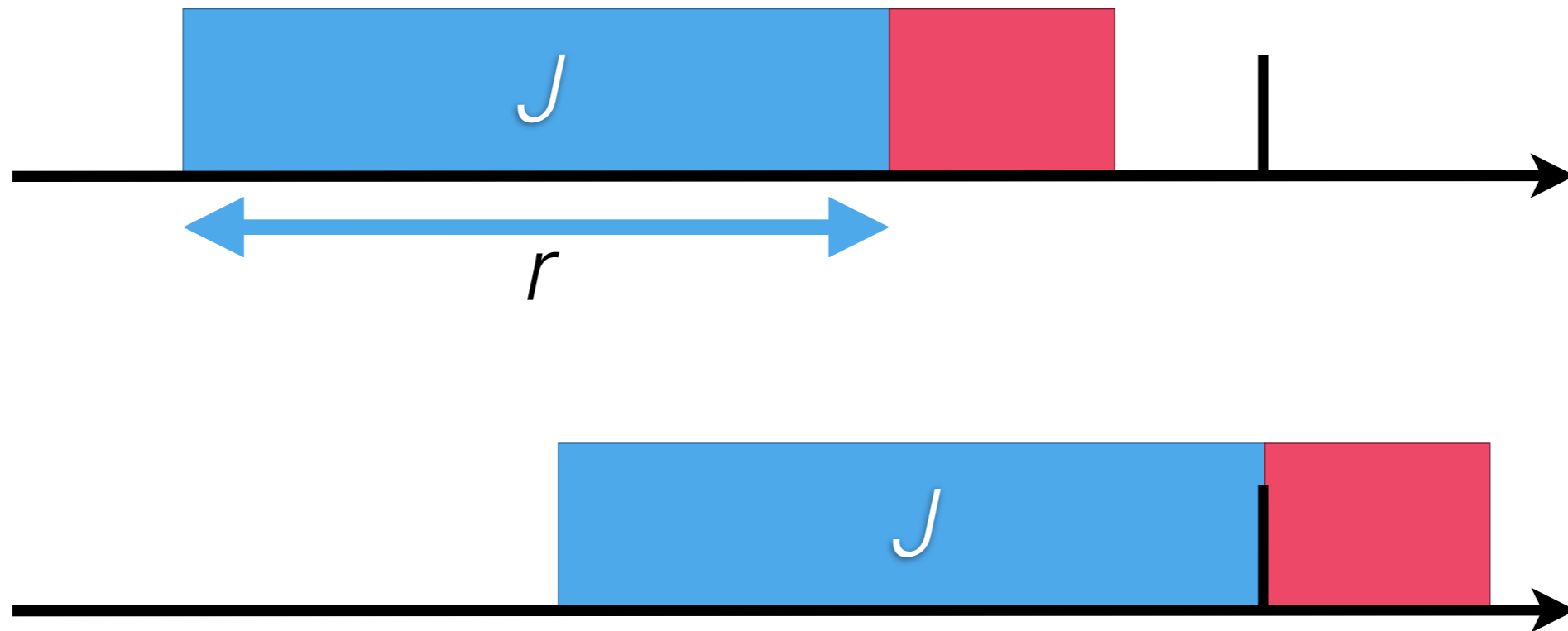
- worst case execution time (WCET) largely exceeds average case
- offering guarantees for the worst case will waste lots of resources
- missing some deadlines can be tolerated with the firm and soft real-time scheme

- desktop real-time
- there are no hard real-time applications on desktops
- there is a lot of firm and soft real-time
 - low-latency audio processing
 - smooth video playback
 - desktop effects
 - user interface responsiveness



- guarantees even slightly below 100% of WCET can dramatically reduce resource allocation
- **slack reclaiming:** unused reservations will be used by others at runtime
- use probabilistic planning to model the actual execution
- quality q : fraction of deadlines to be met





$\mathbf{P}(J \text{ does not run longer than } r \wedge$
 $J \text{ is completed until its relative deadline}) \geq q$

$$r'_i = \min(r \in \mathbb{R} \mid \frac{1}{m_i} \sum_{k=1}^{m_i} \mathbf{P}(X_i + k \cdot Y_i \leq r) \geq q_i)$$

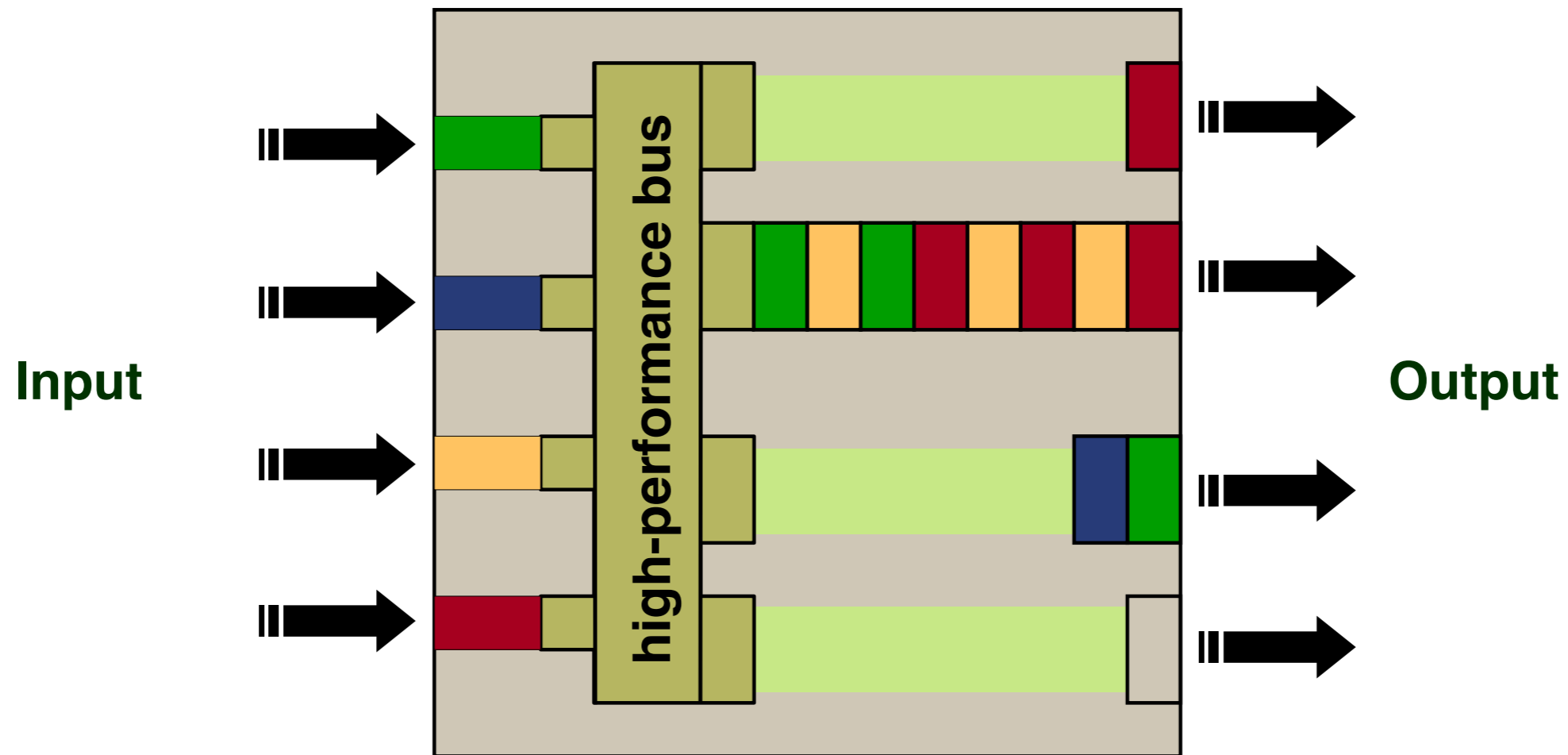
$$r_i = \max(r'_i, w_i) \quad i = 1, \dots, n$$

- to fully understand this: see QRMS paper
- good for microkernel: reservation can be calculated by a userland service
- kernel only needs to support static priorities

- often research only deals with generic management concepts we just discussed
- drilling down is required for usable systems
- coming up next:
specific resources in DROPS (aka TUD:OS)
- for each resource we...
 - outline the real-time guarantee
 - sketch an idea for reservation

NETWORK

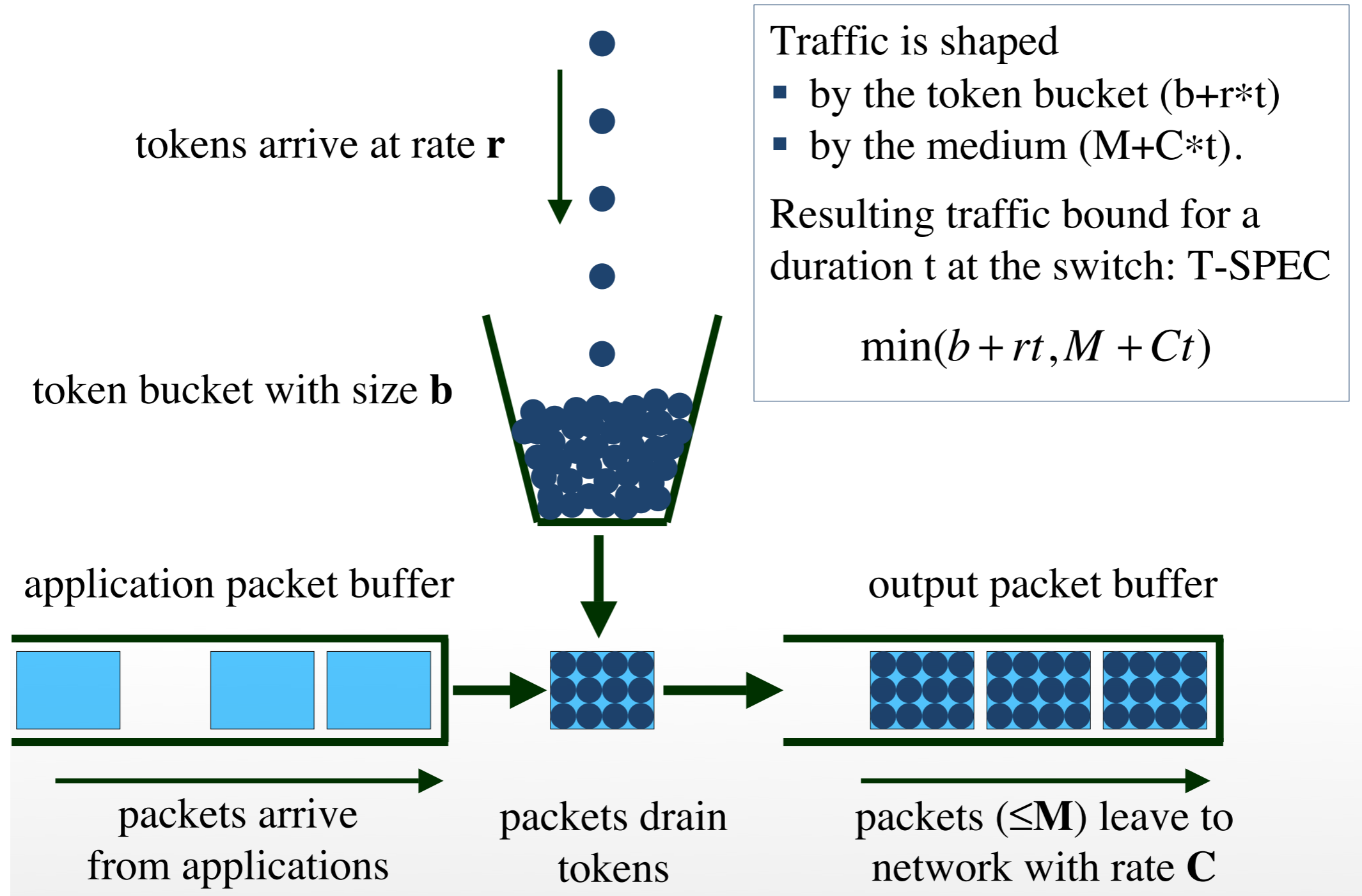
- guaranteed timely communication service
 - lower bound for **bandwidth**
 - upper bound for **latency** and **jitter**
- networks in embedded systems
 - field busses
 - collapsed network stacks
 - bus topology, single broadcast domain
 - example: CAN bus



- switches use buffers on output ports
- delay bound depends on traffic to the output
- if queues get too long, frames are dropped

- traffic on output ports depends on inbound traffic
- inbound traffic depends on the computers sending to the switch
- **shaping** the traffic sent by computers helps
 - bounds incoming traffic at the switch
 - bounds the queue length in the switch
 - prevents dropped packets
- use network calculus for shaping parameters

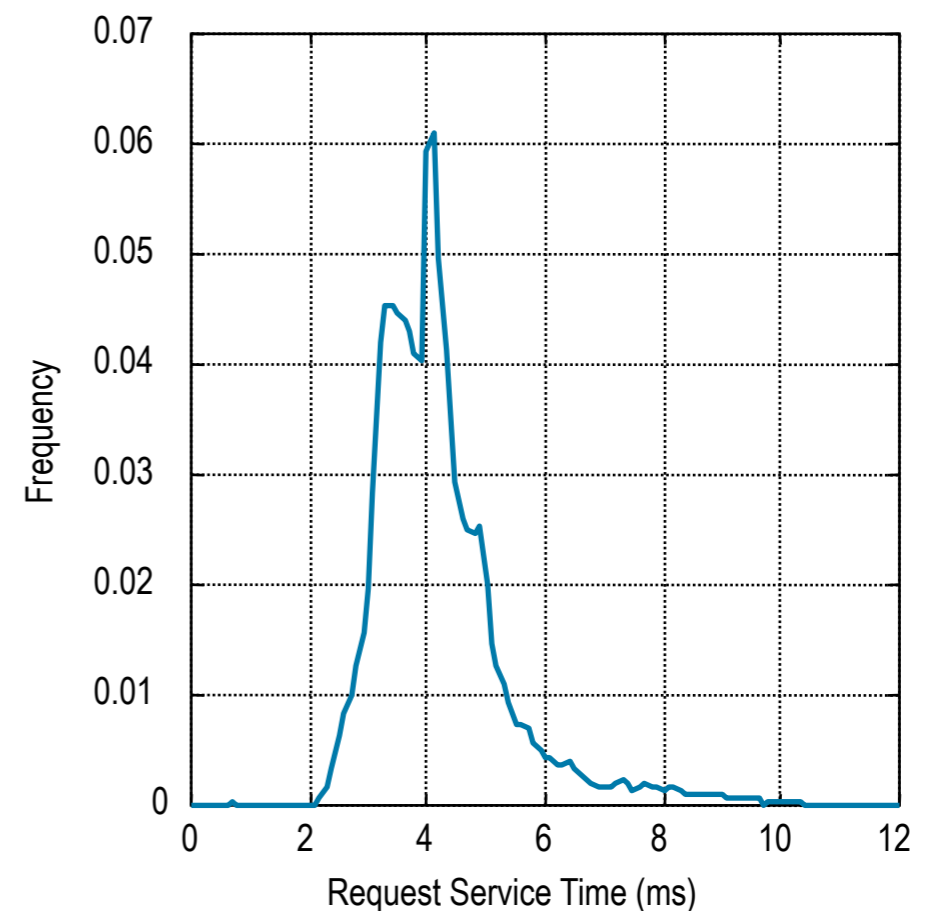
TOKEN BUCKET



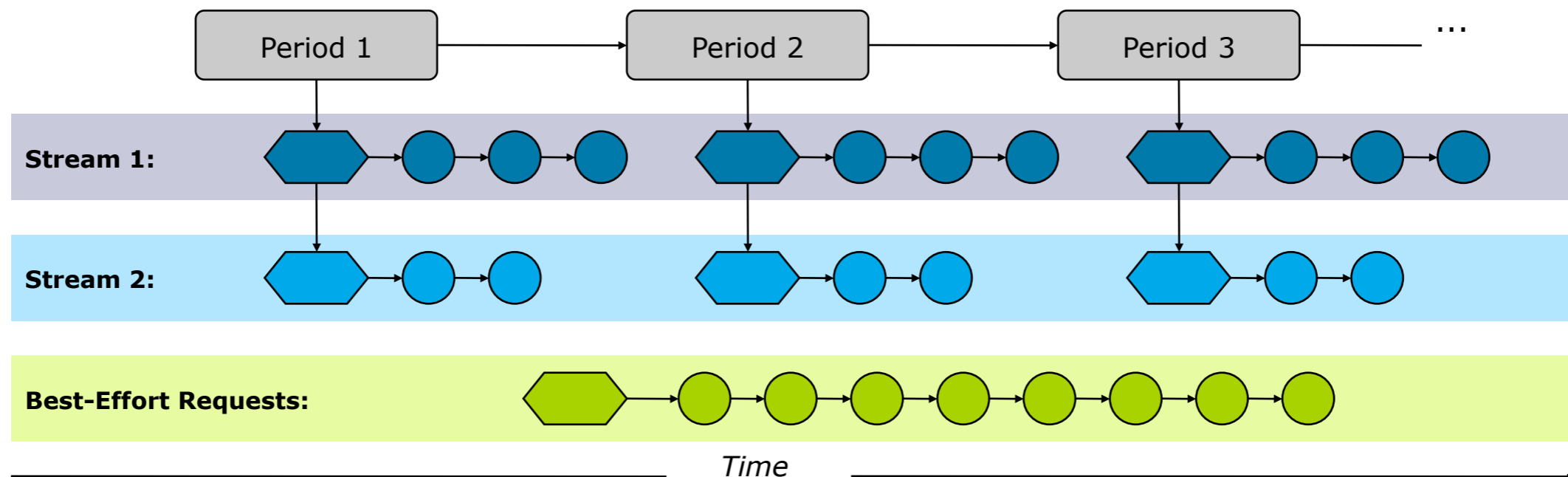
- switch is a shared medium
- all nodes must **cooperate** for this to work
- worst-case delays $\leq 1\text{ms}$
- network utilization $> 90\%$
- no node synchronization required
- predictable packet transmission on off-the-shelf switched ethernet
- **hard real-time** capable

HARD DISK

- guaranteed **bandwidth** of data streams read from / written to disk
- execution times of disk requests vary
 - disk head position
 - rotational delay
- poor ratio between worst case and average case
 - average: 4ms
 - worst: 30ms



- quality-based **probabilistic** scheduling
- map disk bandwidth to the periodic execution of disk requests
- constant number per period
- fixed request size



- quality parameter: fraction of requests processed on time
- admission control calculates **reservation time** for each stream
- disk scheduler enforces reservation
 - requests are only executed as long as the reservation is not depleted
 - problem: disk requests **cannot be aborted**, admission math must deal with this

- scheduler picks requests according to remaining reservation and quality
- not good for disk utilization
- existing non-real-time disk schedulers are much better
 - elevator
 - SATF: shortest access time first

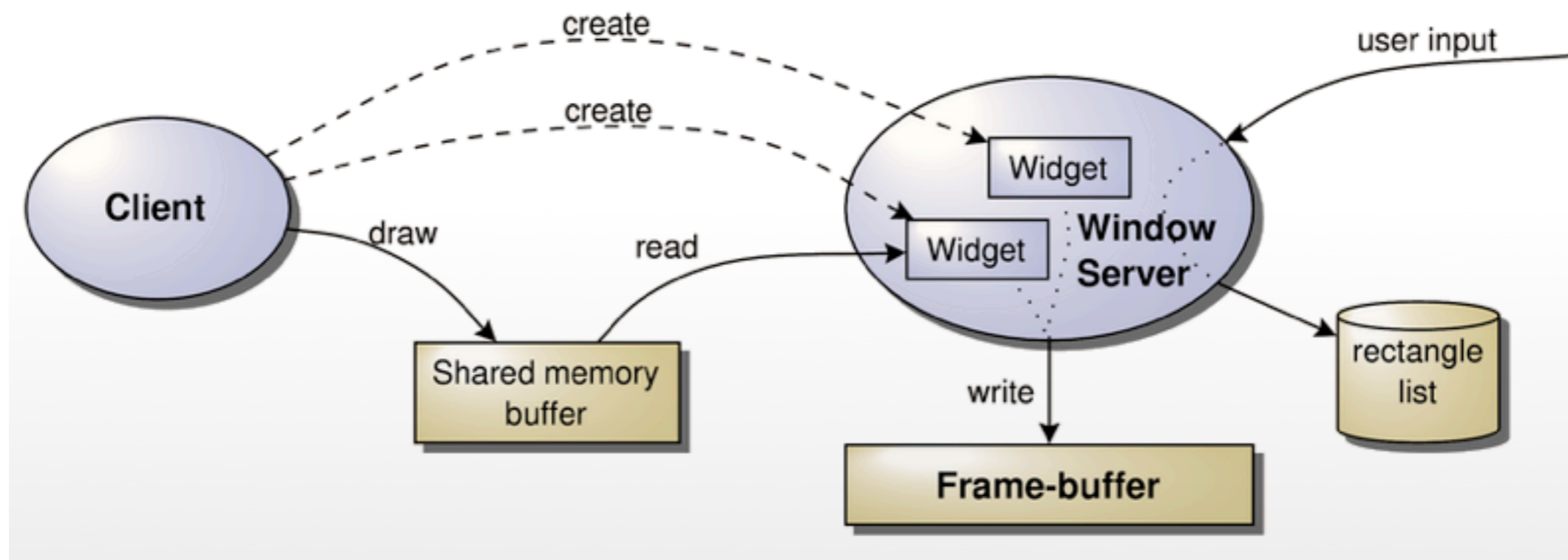
- solution: two level scheduling using **Dynamic Active Subset**
- first level selects set of disk requests
 - that can be executed in any order
 - while still meeting all guarantees
- this set is then handed to the second level scheduler
 - can execute disk requests in any order
 - any non-real-time scheduler works

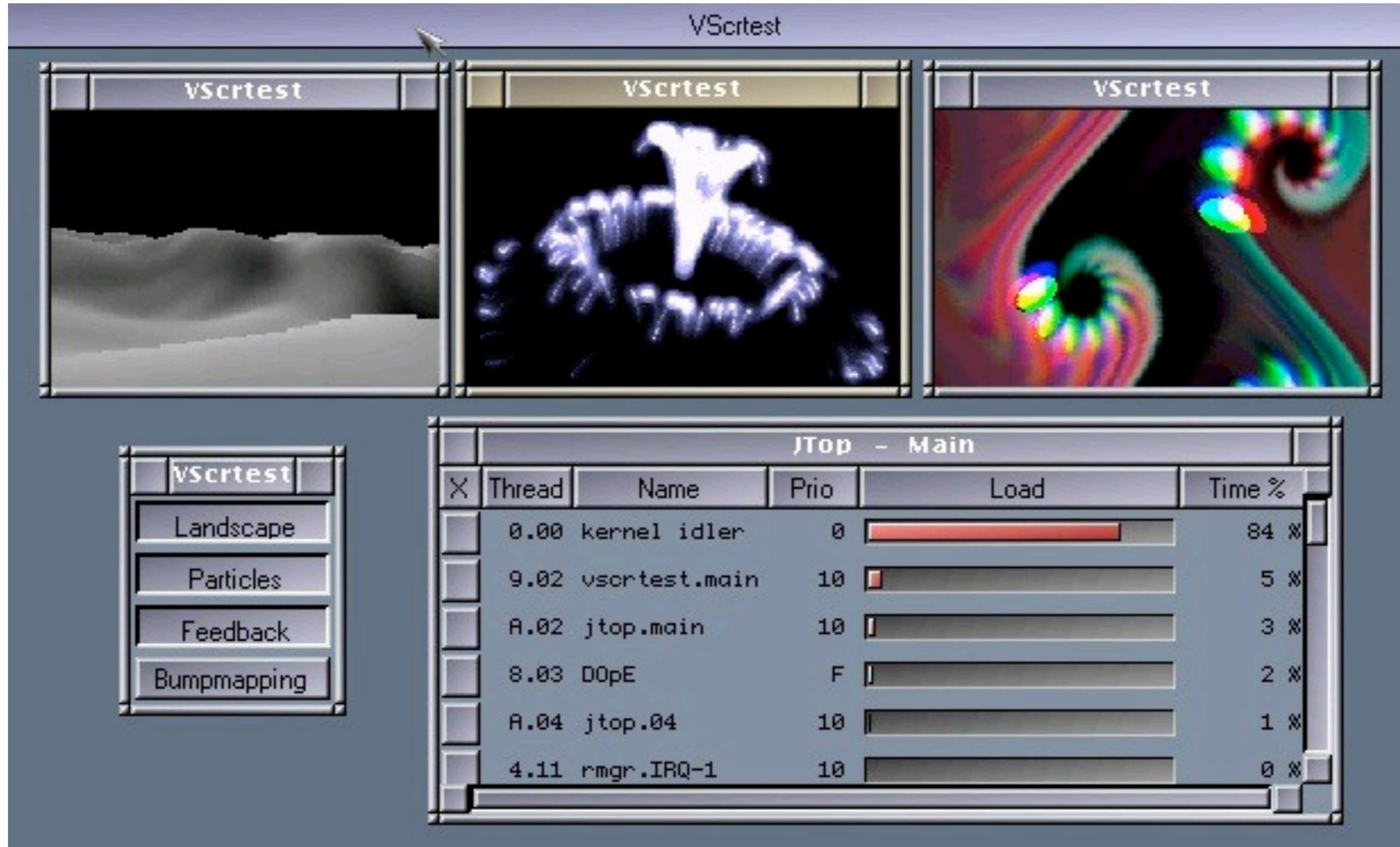
GRAPHICS

- guaranteed **update rates** of GUI elements
 - video output, animations
 - periodic jobs
 - known frame rate and drawing time
- support non-real-time applications at the same time
 - unpredictable
 - minimize latency for responsiveness

- traditional GUIs implement GUI elements („widgets“, „controls“) outside the display system
 - as a library in the application
- window system has no global view on objects involved in a redraw
 - cannot predict effects of redraw operations
 - no guarantees

- DOpE (**D**esktop **O**perating **E**nvironment) implements widgets in the window server
- shared memory buffers for transfer
- no client interaction for redraw operations





The screenshot displays a graphical user interface for a software application named "VScrtest". At the top, there are three separate rendering windows, each titled "VScrtest". The leftmost window shows a grayscale landscape with a horizon line. The middle window displays a complex, glowing blue and white fractal-like structure. The rightmost window features a vibrant, multi-colored fractal pattern with swirling lines in red, green, and blue.

Below the rendering windows is a control panel with a tabbed interface. The active tab is labeled "VScrtest" and contains four buttons: "Landscape", "Particles", "Feedback", and "Bumpmapping".

To the right of the control panel is a task manager window titled "JTop - Main". It displays a table of system threads with columns for Thread ID, Name, Priority, Load, and Time %.

X	Thread	Name	Prio	Load	Time %
	0.00	kernel idler	0	<div style="width: 84%; background-color: red;"></div>	84 %
	9.02	vscrttest.main	10	<div style="width: 5%; background-color: red;"></div>	5 %
	A.02	jtop.main	10	<div style="width: 3%; background-color: red;"></div>	3 %
	8.03	D0pE	F	<div style="width: 2%; background-color: red;"></div>	2 %
	A.04	jtop.04	10	<div style="width: 1%; background-color: red;"></div>	1 %
	4.11	rmgr.IRQ-1	10	<div style="width: 0%; background-color: red;"></div>	0 %

- processing time for redraw correlates with pixels to be carried over the bus
- DOpE reserves fixed CPU shares
- reservation is used to locally schedule redraw operations
- periodic scheduling of real-time redraws
- remaining time used for non-real-time drawing

- complex non-real-time redraws can be split
- outstanding redraws can be merged
 - maximum queue length for outstanding redraws is bounded by the screen pixels
 - **bounded latency** for all graphical output
 - even for non-real-time applications
- **guaranteed response time** to user input

- bus-bandwidth-scheduling only sufficient for software drawing
- today: compositing window managers
- GPU is becoming an essential co-processor
 - needs to be scheduled (like a CPU?)
 - access must be governed
- current hardware not well suited
 - no paging in graphics memory (no MMU!)

COMQUAD

- **Components with Quantitative Properties and Adaptivity**
- combine component technology and resource management
- consider resource consumption of components
- make real-time guarantees accessible to the common software engineer

- probabilistic scheduling
- real-time views for specific devices
 - network
 - disk
 - graphics