TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

PROBABILISTIC SCHEDULING

MICHAEL ROITZSCH

DESKTOP REAL-TIME

PROBLEM

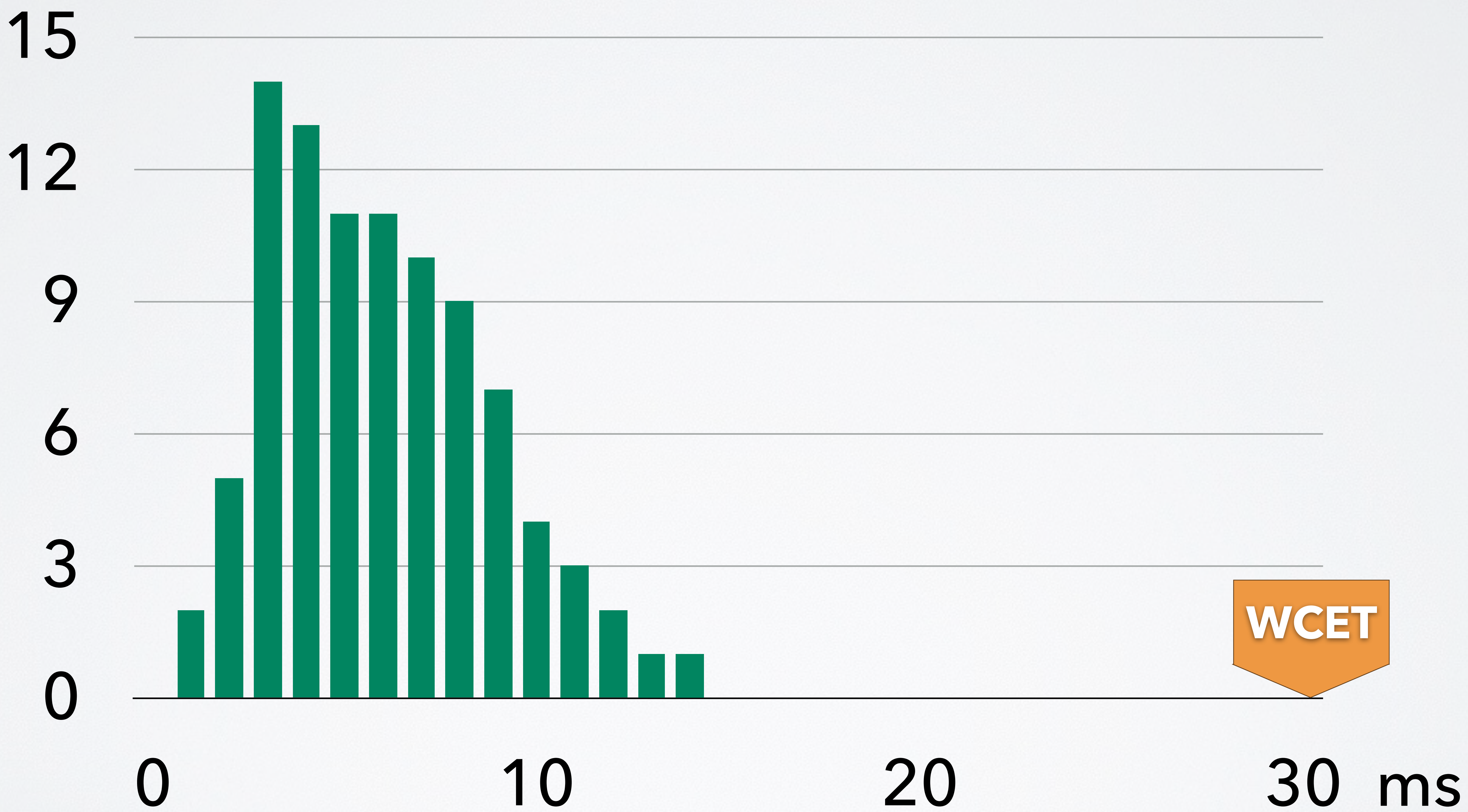
- worst case execution time (WCET) largely exceeds average case
- offering guarantees for the worst case will waste lots of resources
- missing some deadlines can be tolerated with the firm and soft real-time scheme

MOTIVATION

- 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

TECHNISCHE UNIVERSITÄT DRESDEN

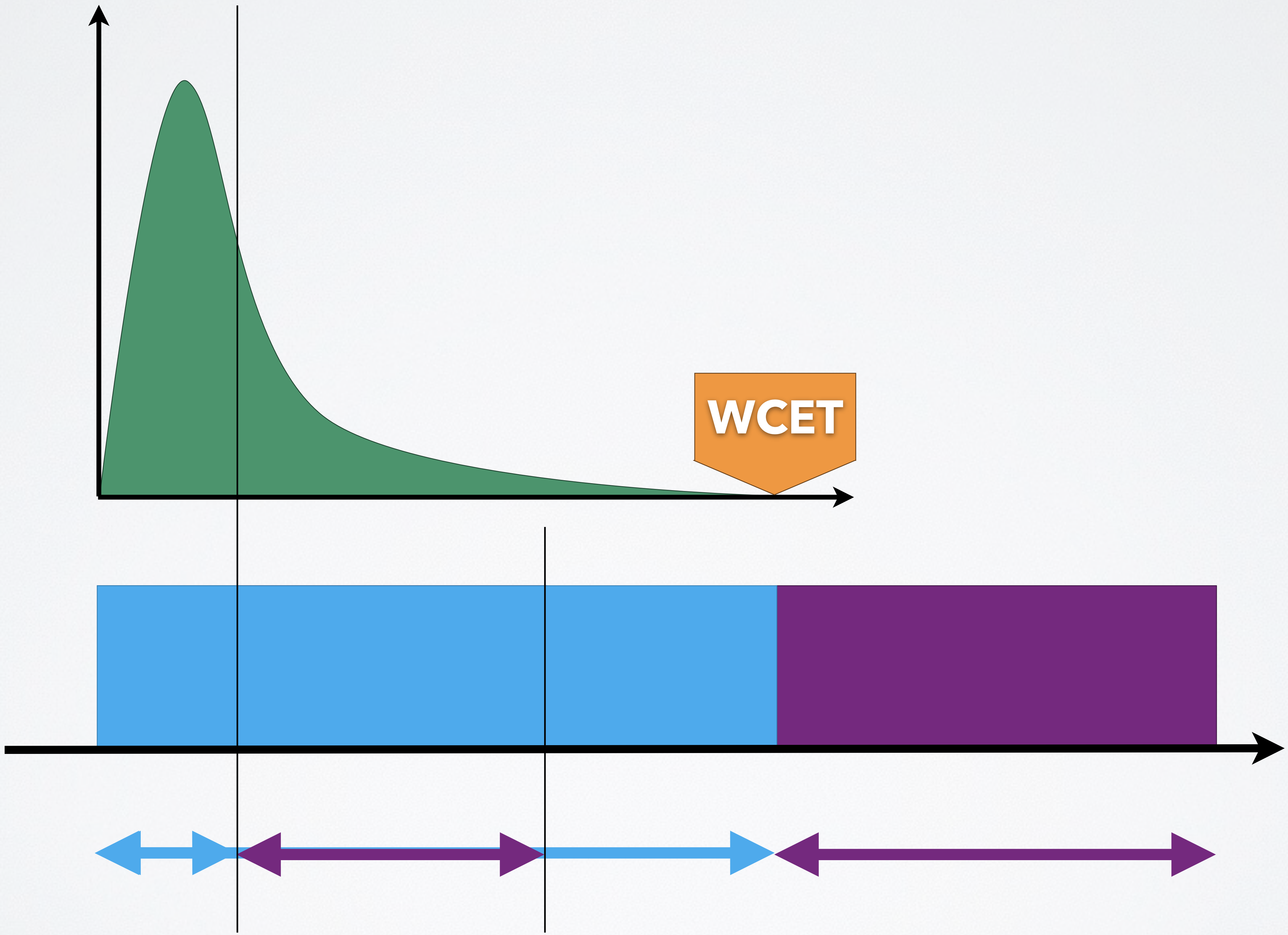
H.264 DECODING



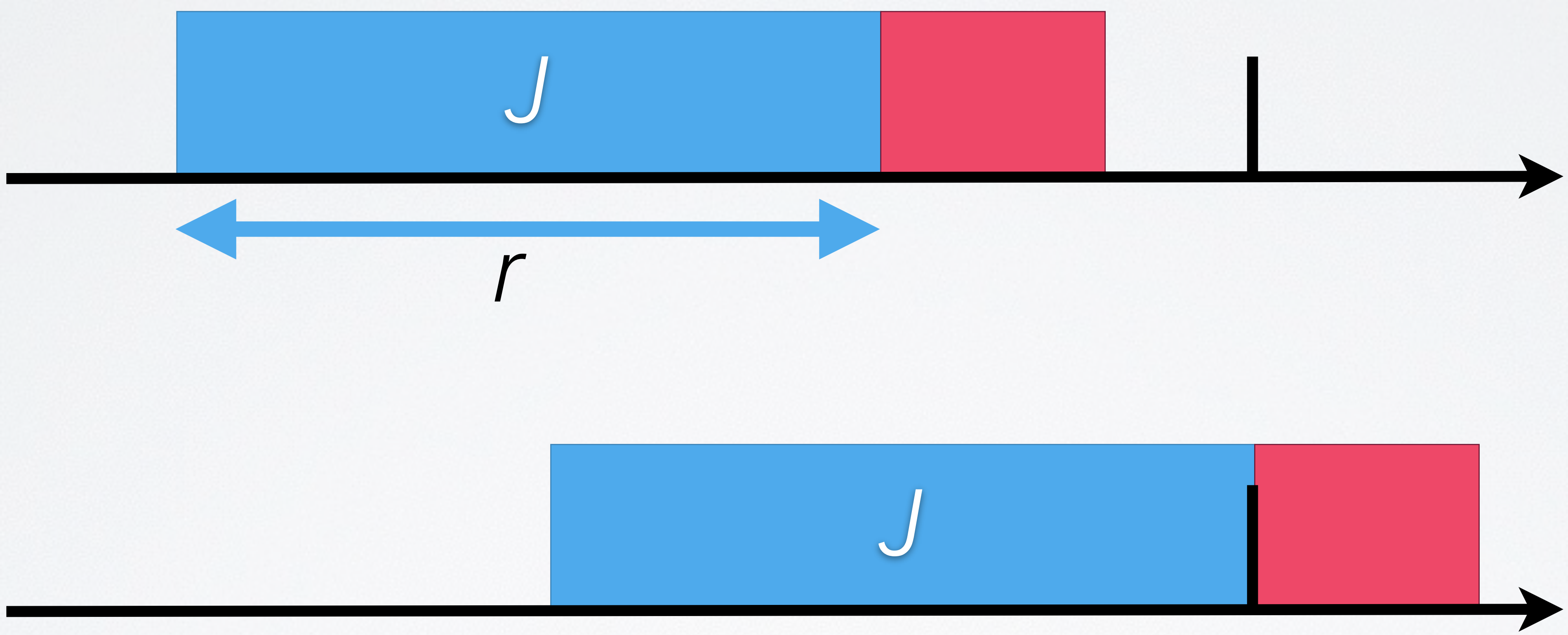
KEY IDEA

- 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

KEY IDEA



RESERVATION



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

RESERVATION

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

$$r_i = \max(r_i', w_i) \quad i = 1, \ldots, n$$

- 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

BREAKPOINT

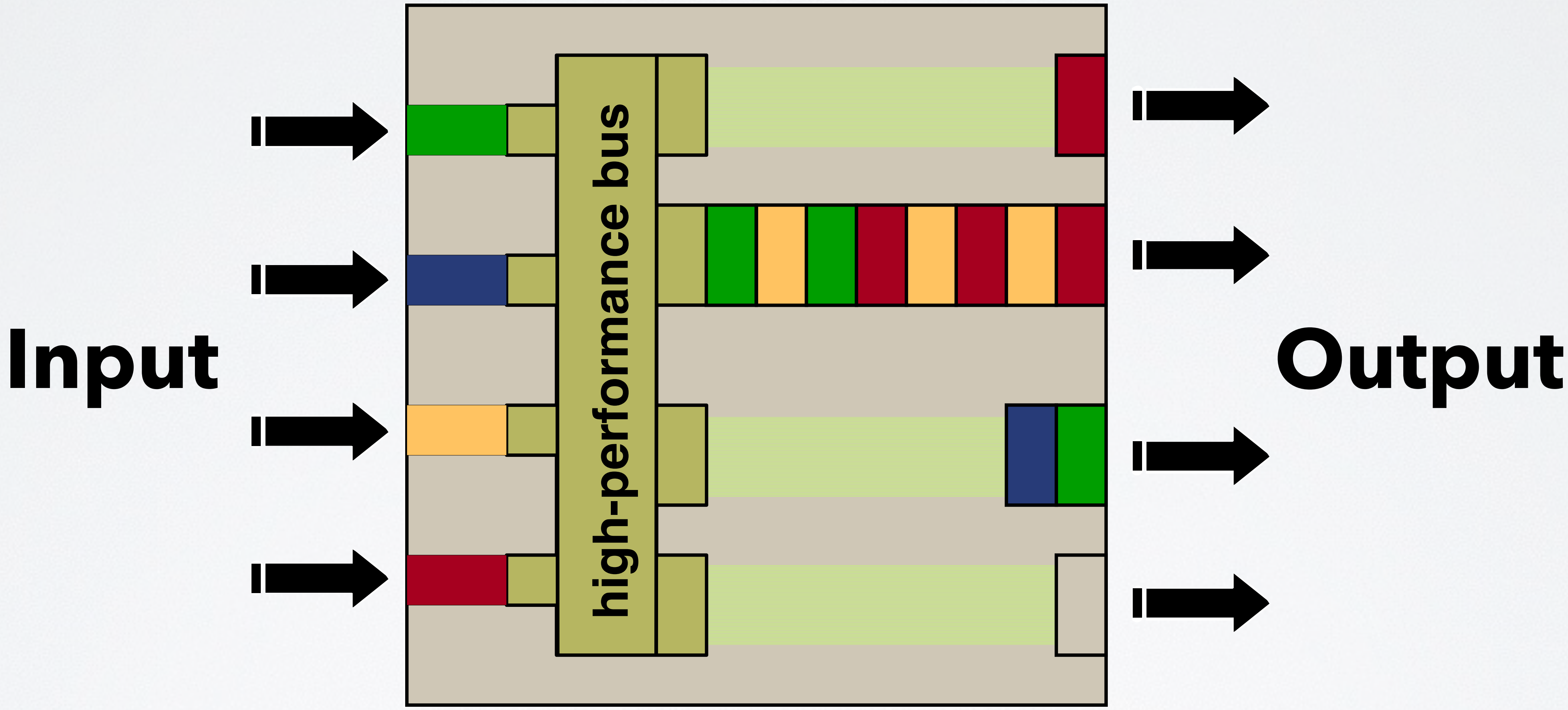
- often research only deals with generic management concepts we just discussed
- drilling down is required for usable systems
- coming up next:
  specific resources in DROPS (aka TUD:OS)
- for each resource we…
  - outline the real-time guarantee
  - sketch an idea for reservation

NETWORK

REAL-TIME

- 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

ETHERNET

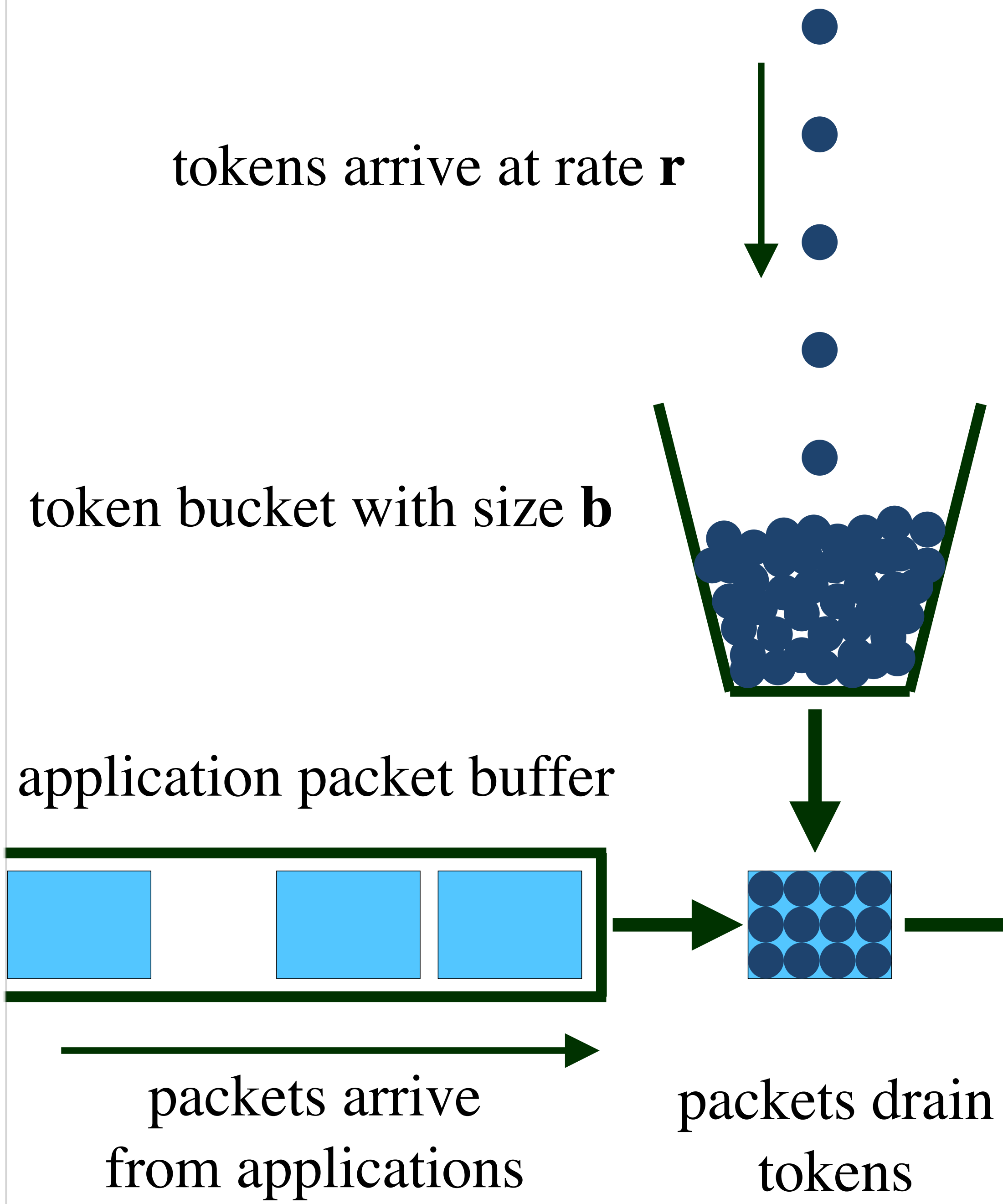


- switches use buffers on output ports
- delay bound depends on traffic to output
- if queues overflow, frames are dropped

IDEA

- 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
- network calculus for shaping parameters

TOKEN BUCKET



Traffic is shaped

- by the token bucket (b+r*t)
- by the medium (M+C*t).

Resulting traffic bound for a duration t at the switch: T-SPEC

$$\min(b + rt, M + Ct)$$

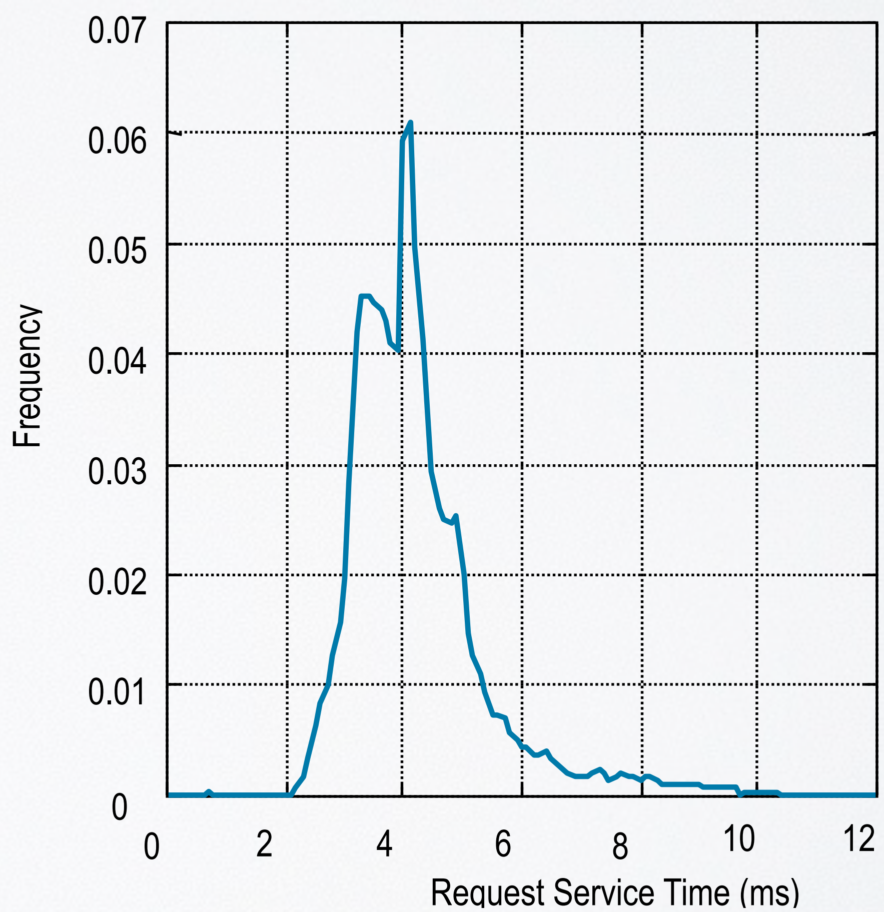
RESULTS

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

HARD DISK

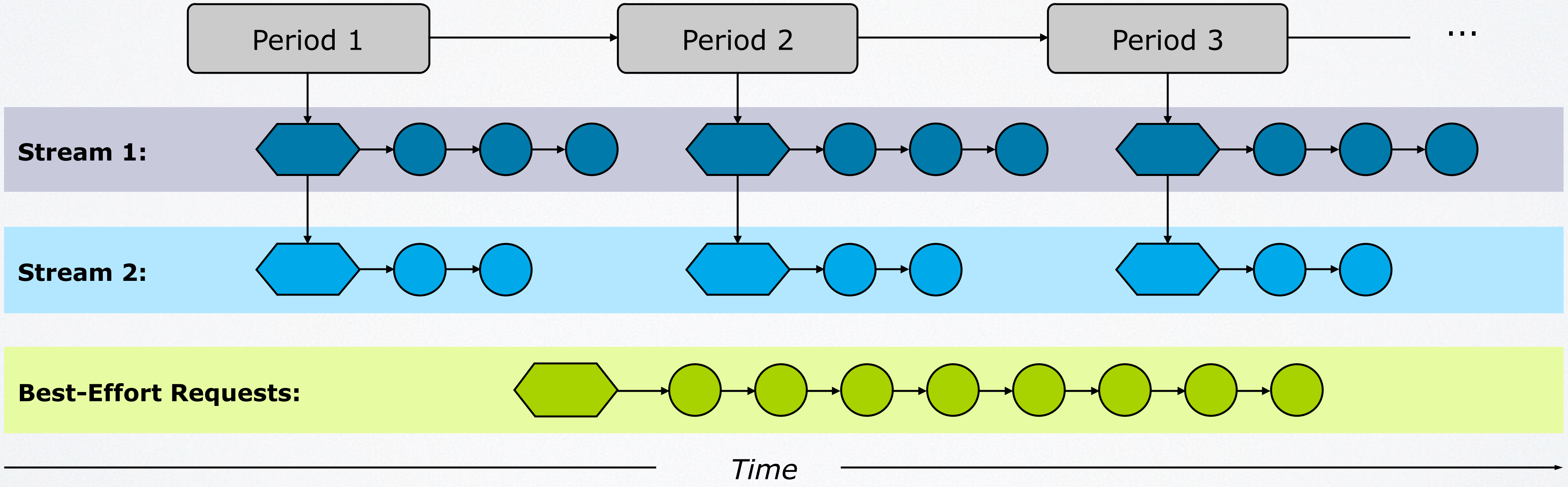
REAL-TIME

- 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 and average case
  - average: 4ms
  - worst: 30ms



IDEA

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



SCHEDULING

- 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

OPTIMIZE

- 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

OPTIMIZE

- 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

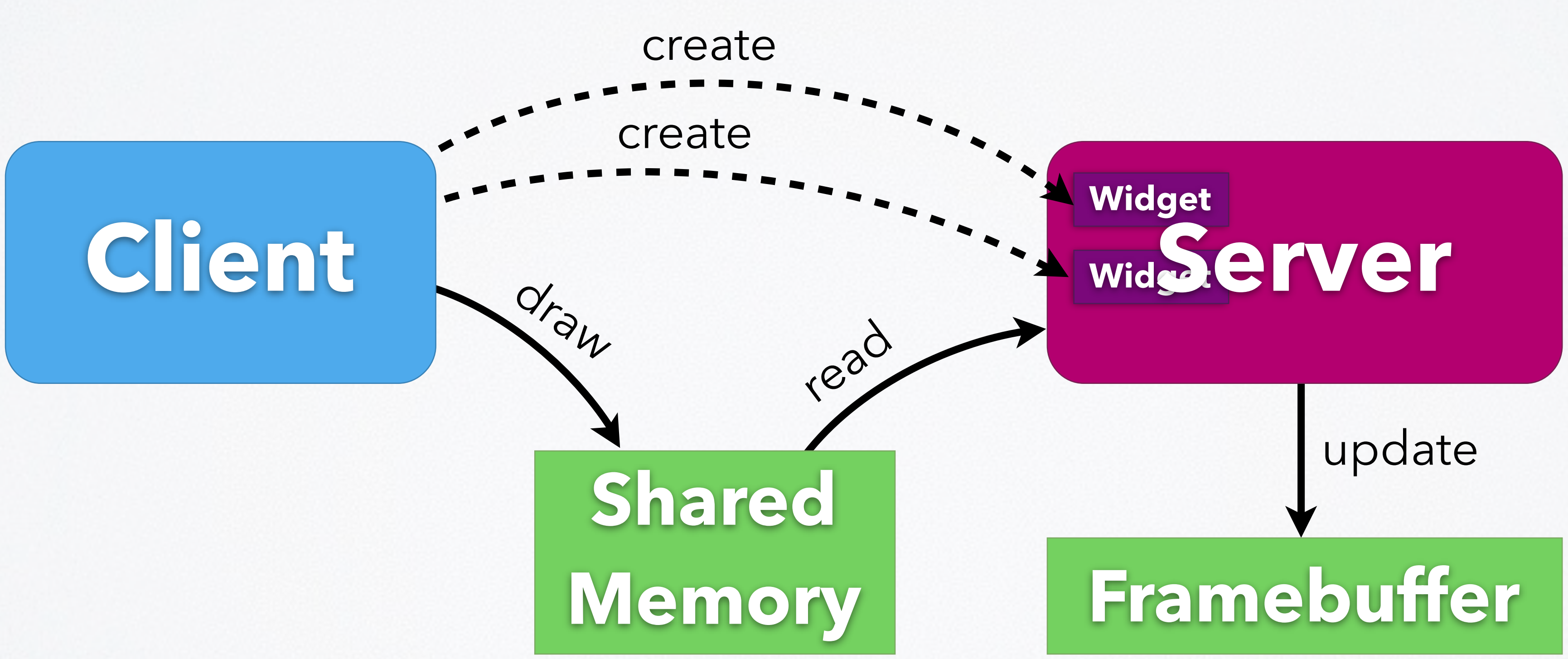
REAL-TIME

- 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

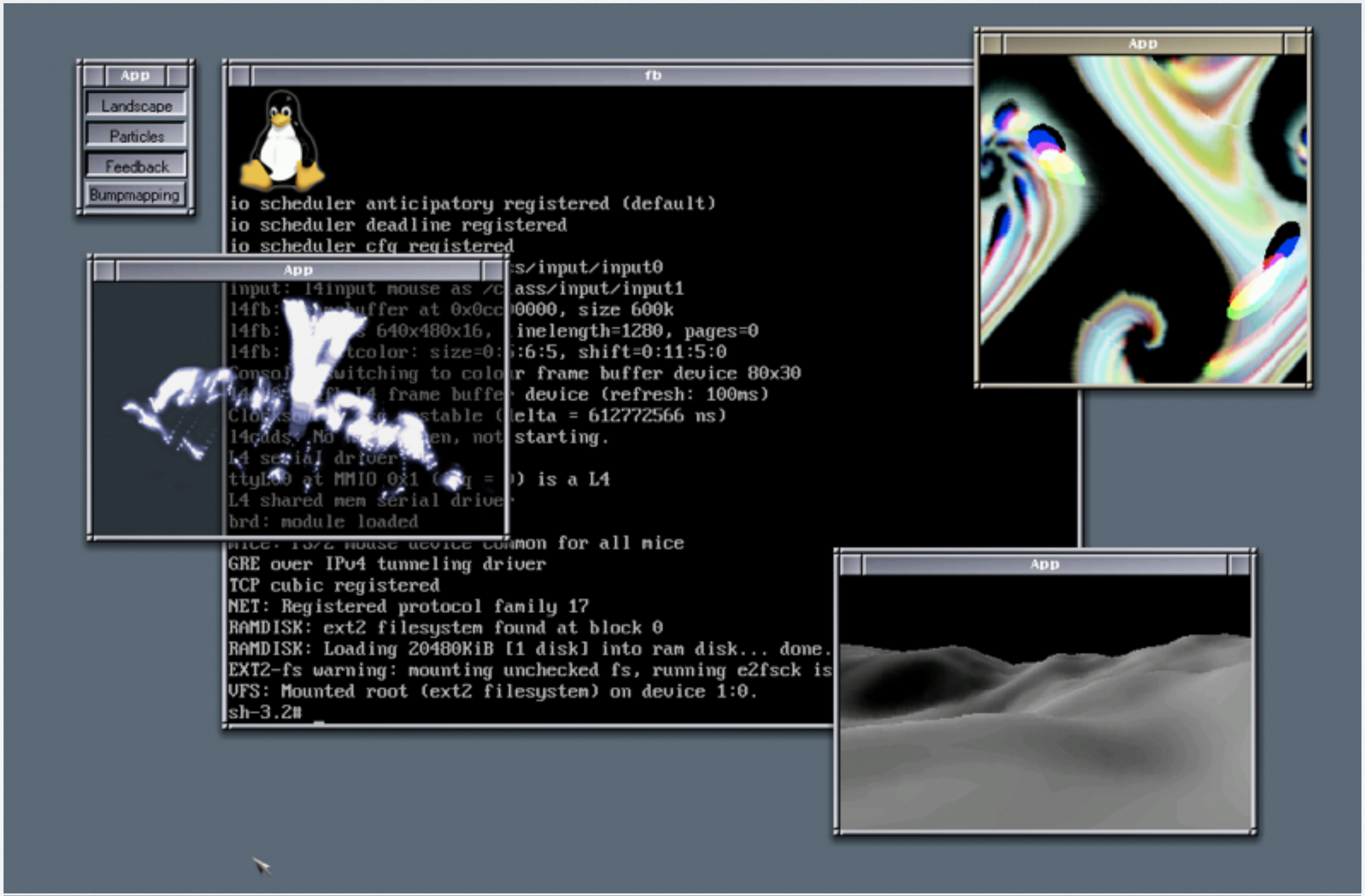
TRADITION

- 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 (Desktop Operating Environment) implements widgets in the window server
  - shared memory buffers for transfer
  - no client interaction for redraw operations



SCREENSHOT


App
Landscape
Particles
Feedback
Bumpmapping
fb
io scheduler anticipatory registered (default)
io scheduler deadline registered
io scheduler cfq registered
L4 shared mem serial driver
brd: module loaded
GRE over IPv4 tunneling driver
TCP cubic registered
NET: Registered protocol family 17
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 20480KiB [1 disk] into ram disk... done.
VFS: Mounted root (ext2 filesystem) on device 1:0.
sh-3.2#
App
App
App

SCHEDULING

- 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

SCHEDULING

- split complex non-real-time redraws
- 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

TRENDS

- 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!)

TECHNISCHE UNIVERSITÄT DRESDEN

SUMMARY

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