TECHNISCHE UNIVERSITÄT DRESDEN

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

RESOURCE MANAGEMENT

MICHAEL ROITZSCH

AGENDA

- done: time
- today: misc. resources
  - architectures for resource management
  - solutions for specific resources
- upcoming: drivers

TECHNISCHE UNIVERSITÄT DRESDEN

KERNEL RESOURCES

PROBLEM

- kernel manages
  - tasks
  - threads
  - capabilities
  - mapping database
- it all comes down to memory
- kernel memory is limited
- opens the possibility of DoS attacks

IDEA

- memory management **policy** should not be in the kernel
- manage kernel memory in userland
- kernel provides memory control **mechanism**
- exception for bootstrapping:
  initial kernel memory is managed by kernel

SOLUTION

- kernel-memory objects in **L4.sec**
- extension of the recursive address space model
- create kernel-memory object by converting user pages
- userland apps pay for kernel memory allocated on their behalf
- converted pages no longer accessible by userland

SOLUTION

- kernel uses kernel-memory objects for storing allocated data structures
- supports map and unmap
- reconvert on unmap
- reconvert transfers object back to userland
- recursively deletes all kernel structures
- dependency graph must be cycle-free

PRINCIPLE

separate enforcement and management

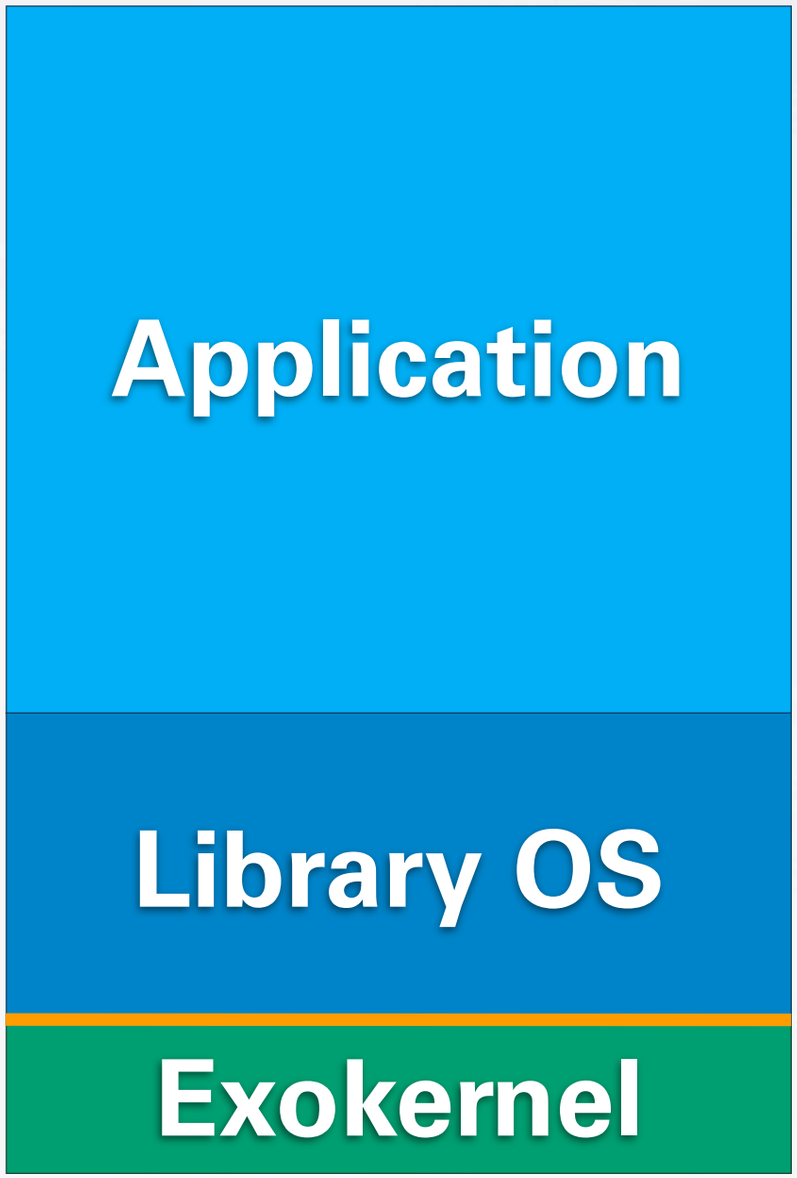
TECHNISCHE UNIVERSITÄT DRESDEN

ARCHITECTURES

EXOKERNEL

Management

Enforcement



DESIGN

- provide primitives at the **lowest possible level** necessary for protection
- use **physical names** wherever possible
- resource management primitives:
  - explicit allocation
  - exposed revocation
  - protected sharing
  - ownership tracking

CONSEQUENCES

- each application can use its own **library OS**
- library OS'es cannot trust each other
- no central management for global resources
- think of a file system
  - kernel manages disk ownership with block granularity
  - each library OS comes with its own filesystem implementation
- one partition per application?

SHARING

- invariants in shared resources must be maintained
- four mechanisms provided by the exokernel
  - **software regions** for sub-page memory protection, allows to share state
  - **capabilities** for access control
  - **critical sections**
  - **wakeup predicates:** code downloaded into the kernel for arbitrary checks

MULTISERVER

LEVELS

- different abstraction levels for resources

basic resources	**memory, CPU, IO-ports, interrupts**
hardware	**block device, framebuffer, network card**
compound resources	**file, GUI window, TCP session**

HIERARCHIES

- applications use multiple resources that depend on other resources
- resource tree of cooperating resource managers
- isolation of resources allows managers to provide real-time guarantees for their specific resource
  - DROPS:
    Dresden **R**eal-time **OP**erating **S**ystem

HIERARCHIES

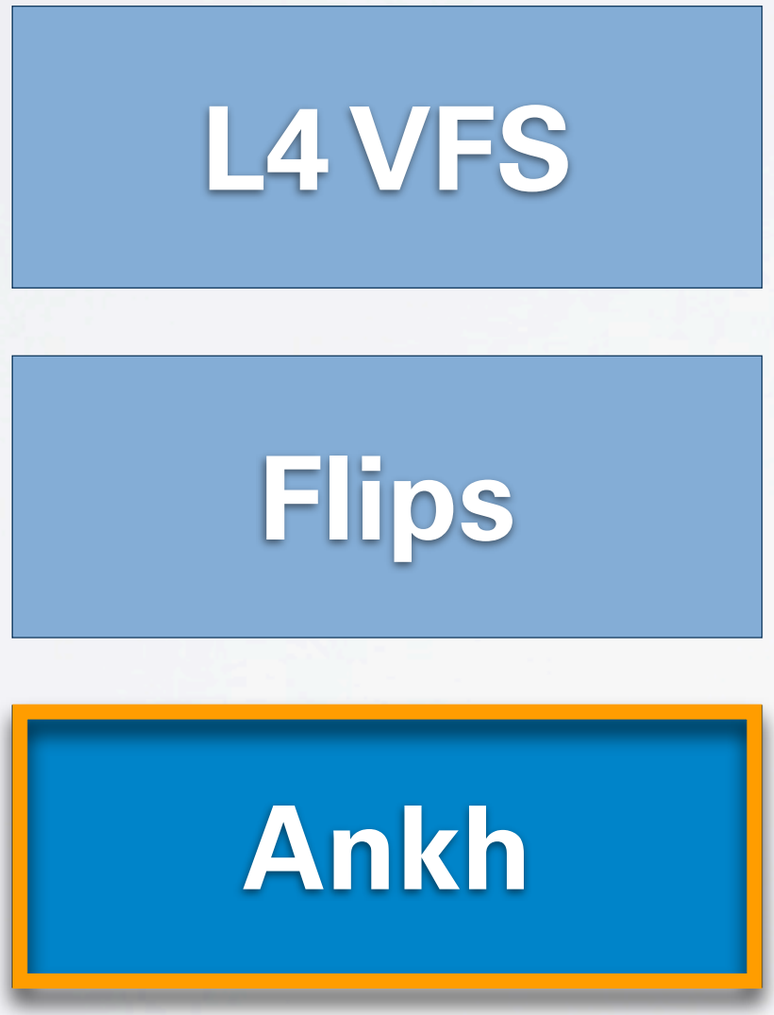
- some resources are by themselves managed in a hierarchy
  - recursive virtual memory
- hierarchy can differentiate service
  - different memory properties by different pagers

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 will cover the layered managers
  - we will outline an idea to provide real-time guarantees

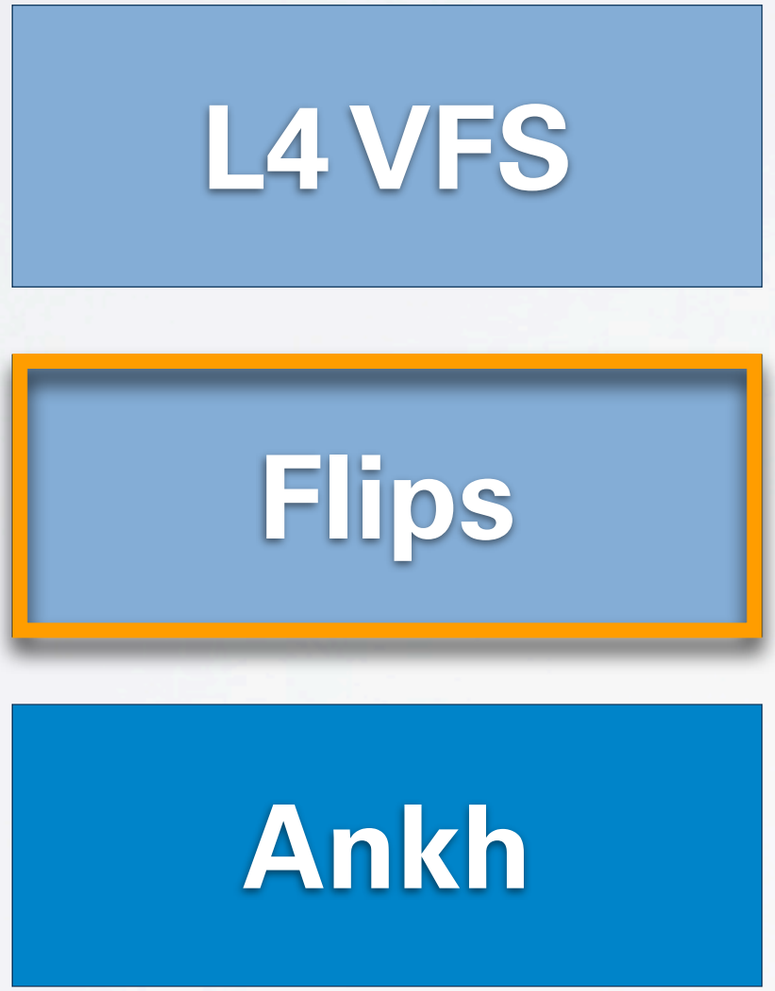
TECHNISCHE UNIVERSITÄT DRESDEN

NETWORK



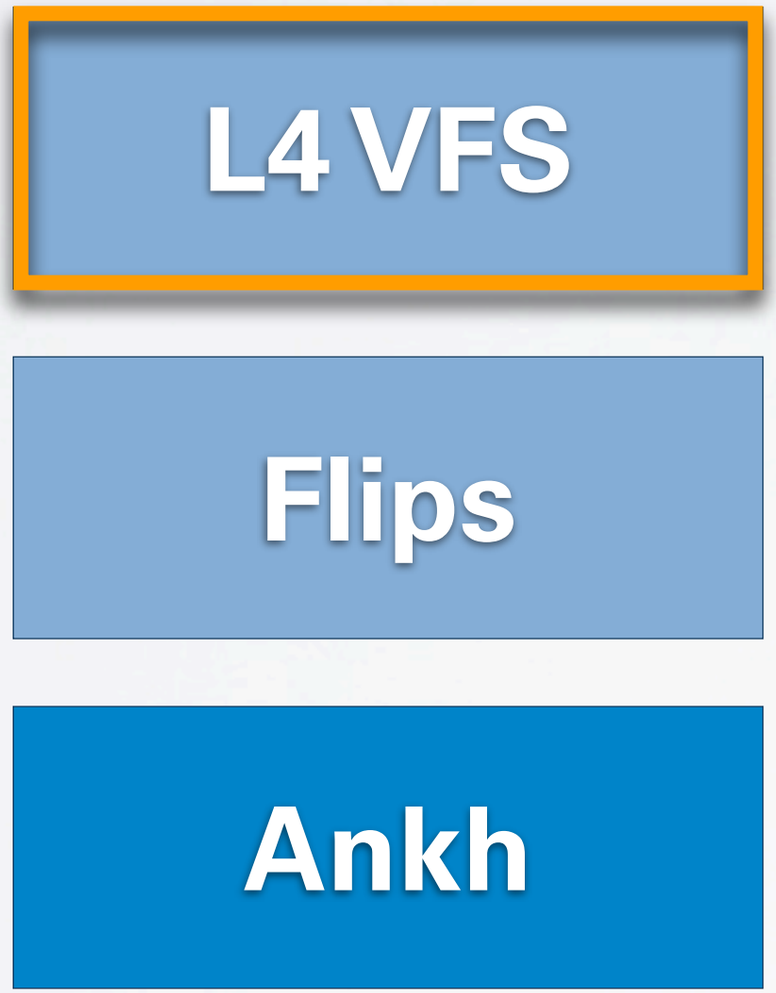
- driver for physical network card
- built with DDE using Linux 2.6 drivers
- provides multiple virtual network cards
- implements a simple virtual bridge

FLIPS



- Flexible IP Stack
- TCP/IP stack
- written by ripping the relevant code out of Linux 2.4
- uses DDE to use Linux code with little modifications

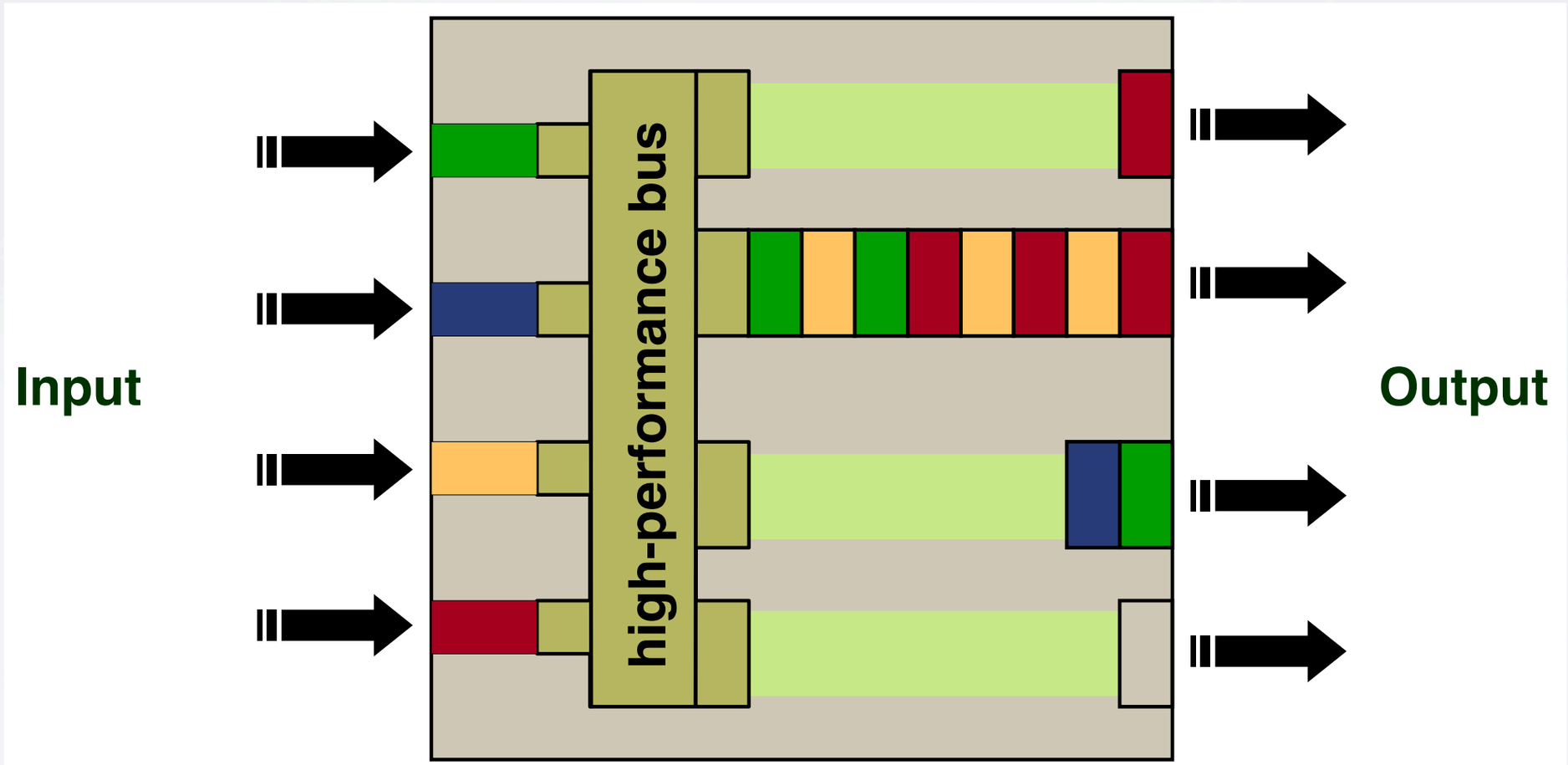
L4 VFS



- virtual file system layer
- provides POSIX file interface
- for network: BSD sockets

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

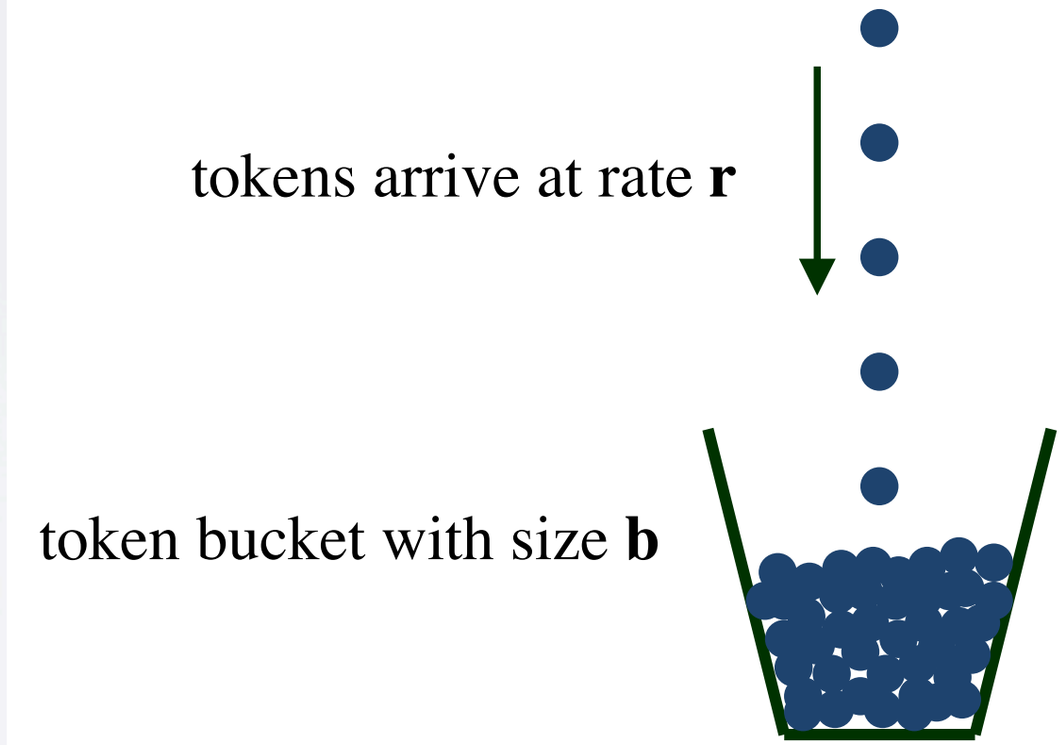


- switches use buffers on output ports
- delay bound depends on traffic to the output
- if queues get too long, frames are dropped

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
- use network calculus for shaping parameters

TOKEN BUCKET

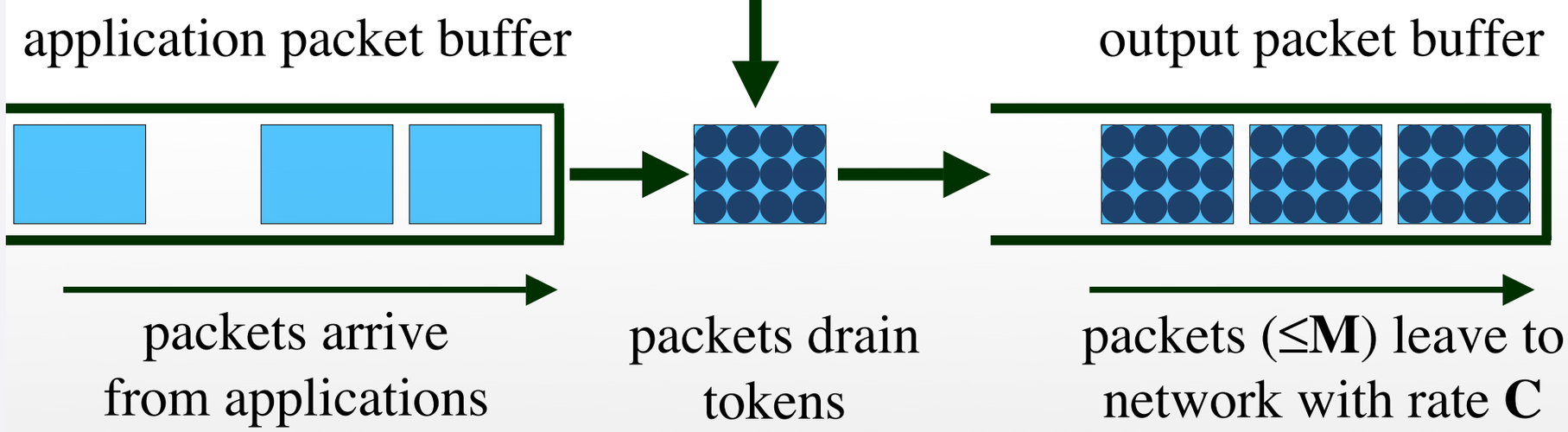


Traffic is shaped

- 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

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



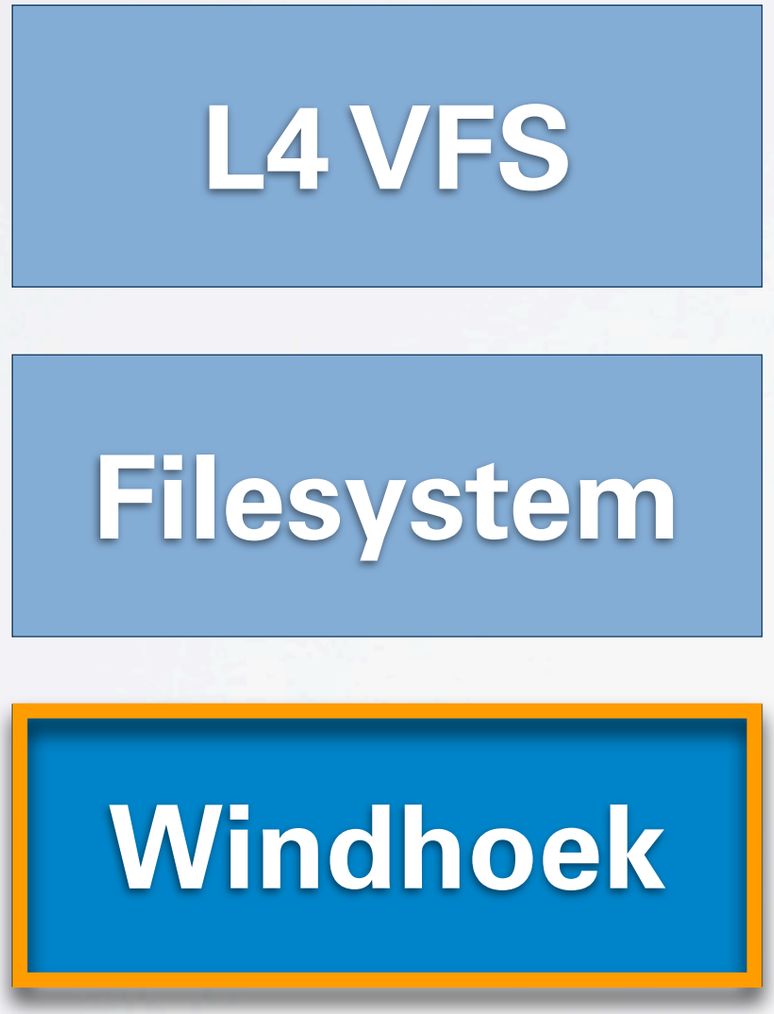
RESULTS

- 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

TECHNISCHE UNIVERSITÄT DRESDEN

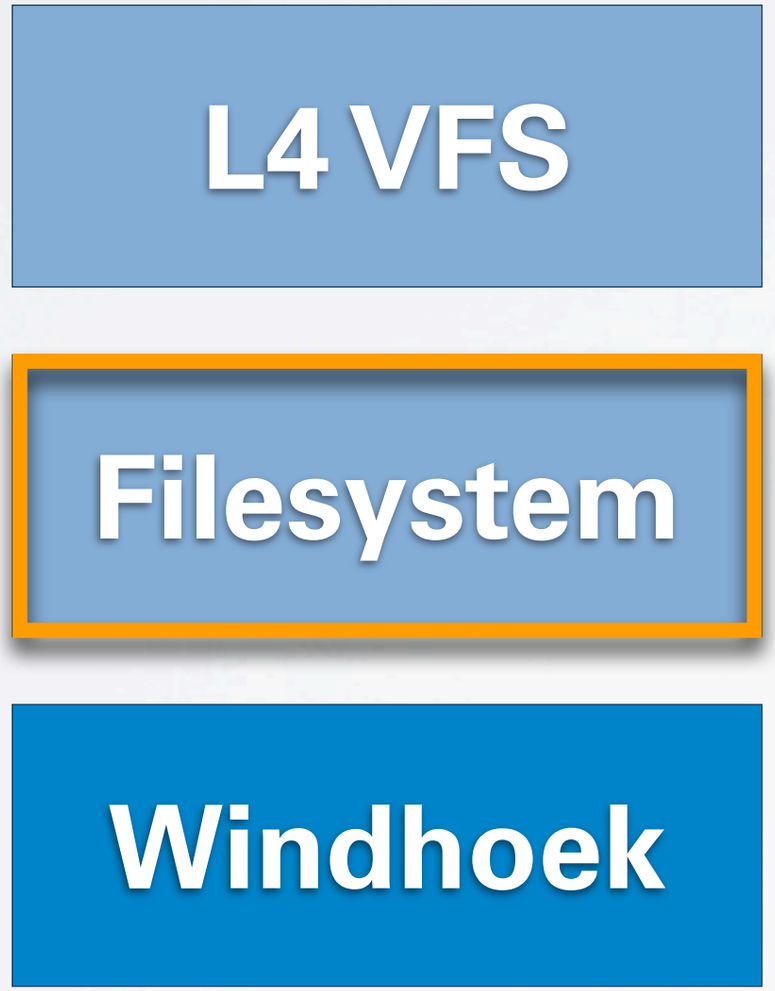
HARD DISK

WINDHOEK



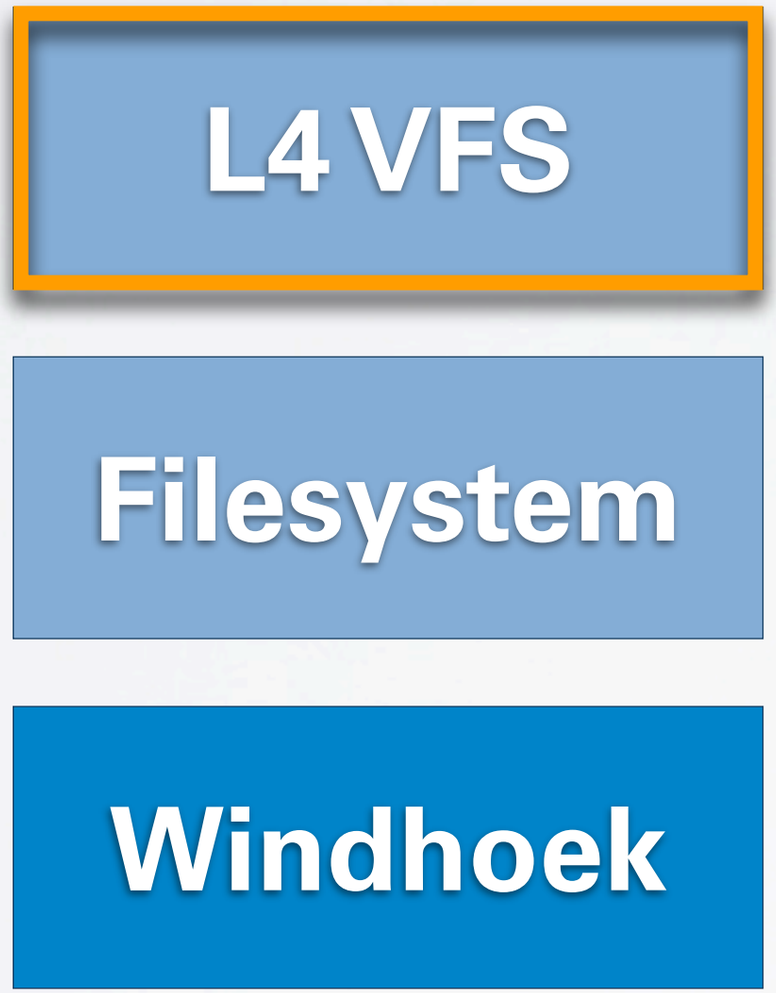
- IDE driver to access hard disks
- includes disk request scheduling
- based on DDE
- provides block device
- ongoing work on USB block devices

FILESYSTEM



- no real one implemented
- we have a tmpfs using RAM as backing store
- VPFS: securely reuse a Linux filesystem

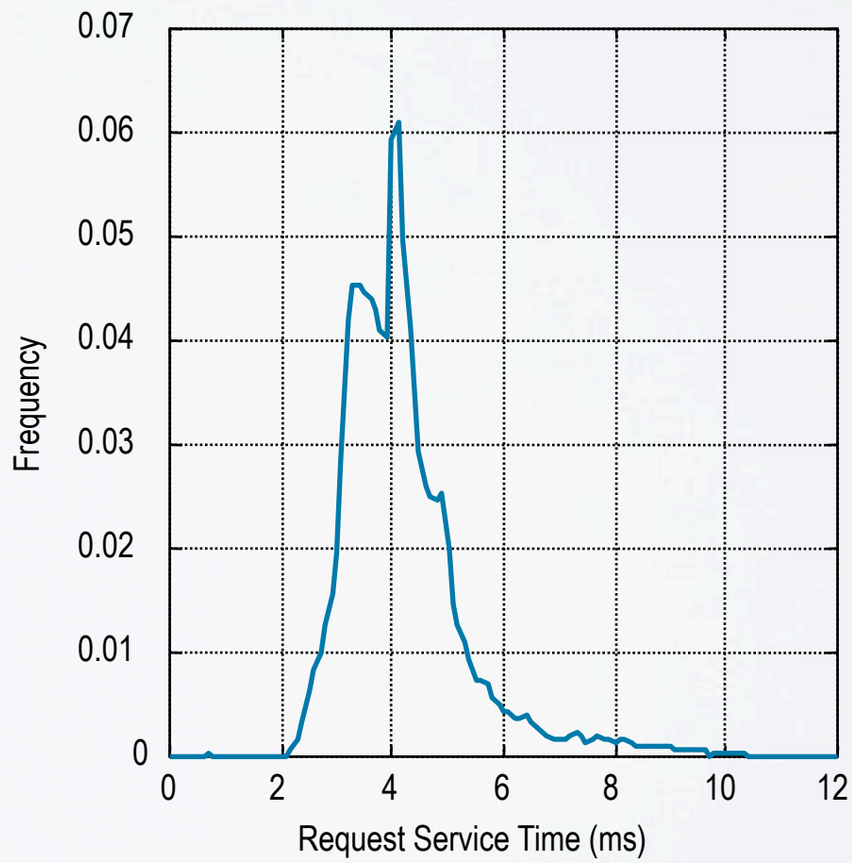
L4 VFS



- hierarchical name space
- connects subtrees to different backend servers
- aka mounting

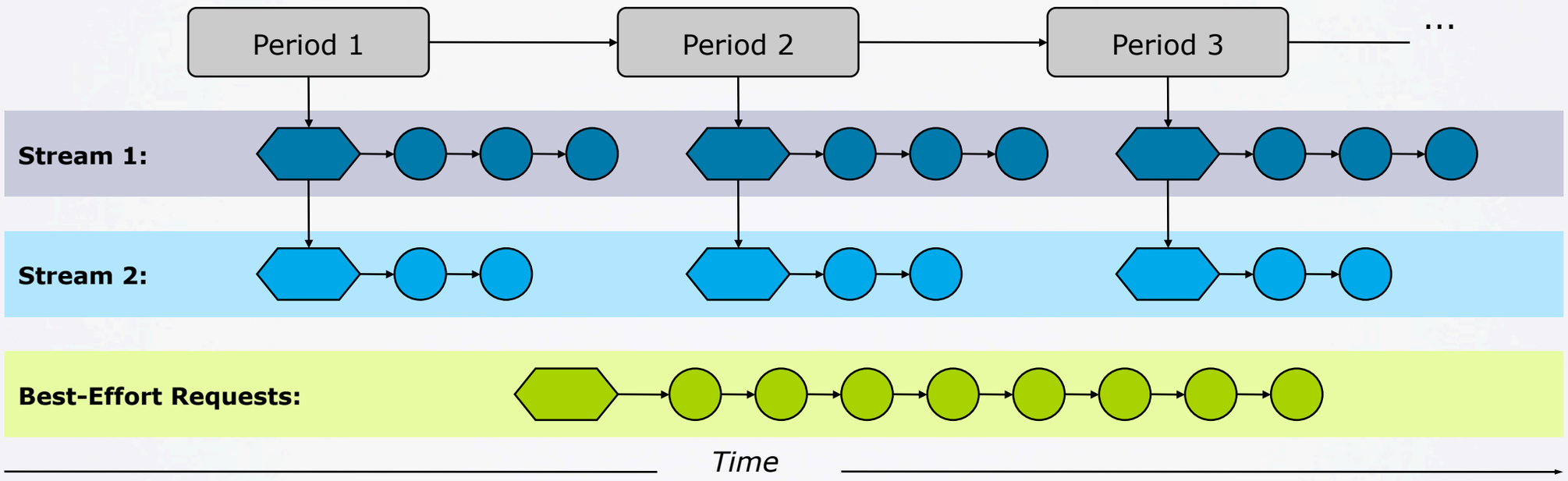
REAL-TIME

- 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 case 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

TECHNISCHE UNIVERSITÄT DRESDEN

GRAPHICS

- multiplexes the frame buffer
- no virtual desktops, but window merging
- details in the legacy / security lectures

DOPE

- widget drawing server
- also handles mouse and keyboard input
- distributed to applications
- can also operate on raw framebuffer
- real-time capable

TERMINAL

- DOpE client providing a terminal window
- VT100 emulation
- can support readline applications
  - shell
  - python

REAL-TIME

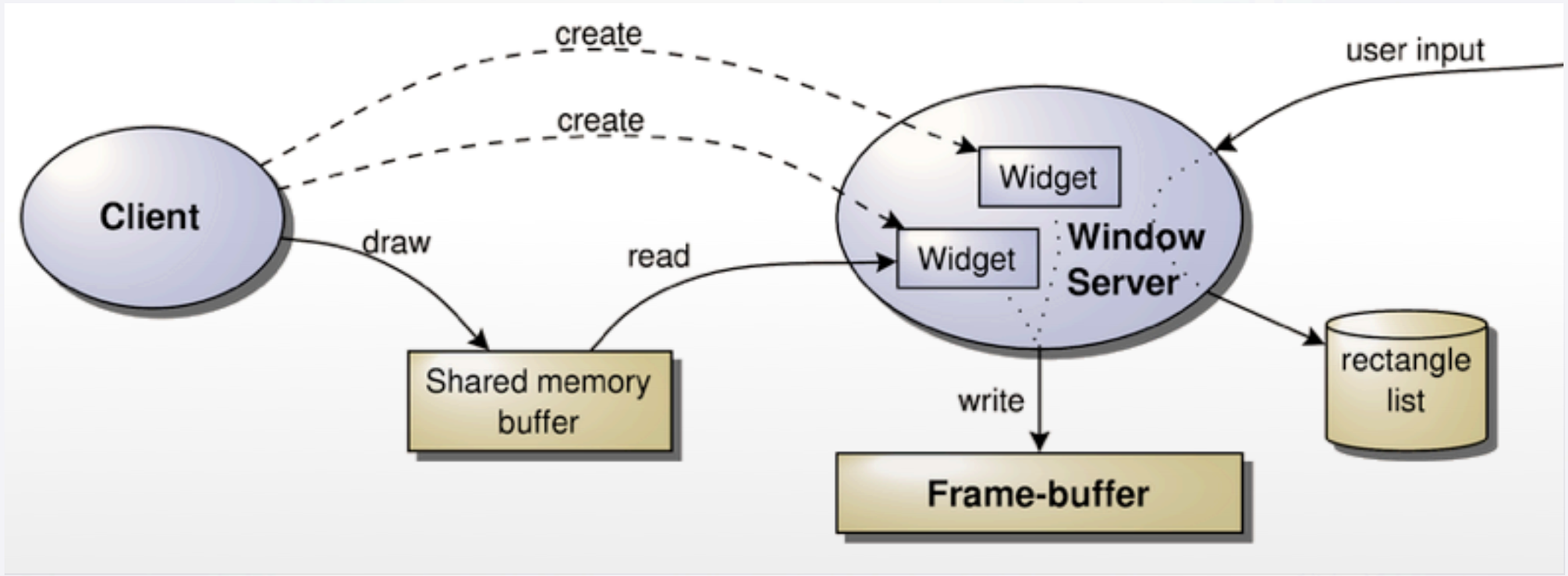
- guaranteed **update rates** of GUI elements
  - video output, animations
  - periodic jobs
  - known frame rate and drawing time
- support non-real-time applications at the same time
  - unpredictable
  - minimize latency for responsiveness

TRADITION

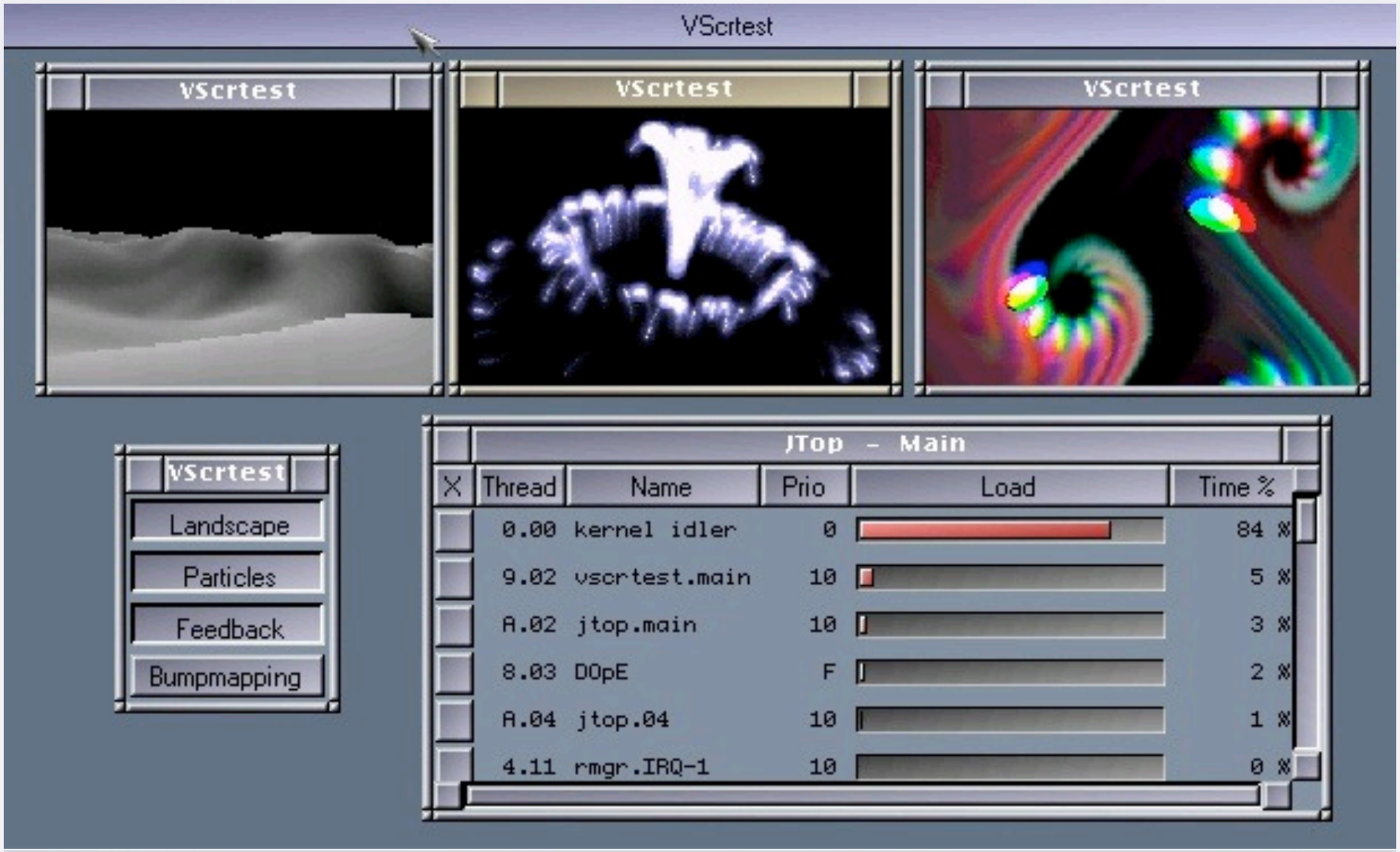
- traditional GUIs implement GUI elements („widgets“, „controls“) outside the display system
  - as a library in the application
- window system has no global view on objects involved in a redraw
  - cannot predict effects of redraw operations
  - no guarantees

DOPE

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



SCREENSHOT


VScrtest
VScrtest
VScrtest
VScrtest
VScrtest
Landscape
Particles
Feedback
Bumpmapping
JTop - Main
X
Thread
Name
Prio
Load
Time %
0.00 kernel idler 0 84 %
9.02 vscrtest.main 10 5 %
A.02 jtop.main 10 3 %
8.03 DOpE F 2 %
A.04 jtop.04 10 1 %
4.11 rmgr.IRQ-1 10 0 %

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

- 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

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

COMQUAD

GOAL

- **Com**ponents with **Qu**antitative Properties and **Ad**aptivity
- combine component technology and resource management
  - consider resource consumption of components
  - make real-time guarantees accessible to the common software engineer

SUMMARY

- kernel resource management
- basic resource management concepts
  - exokernel
  - multiserver
- management details for specific resources
  - network
  - disk
  - graphics