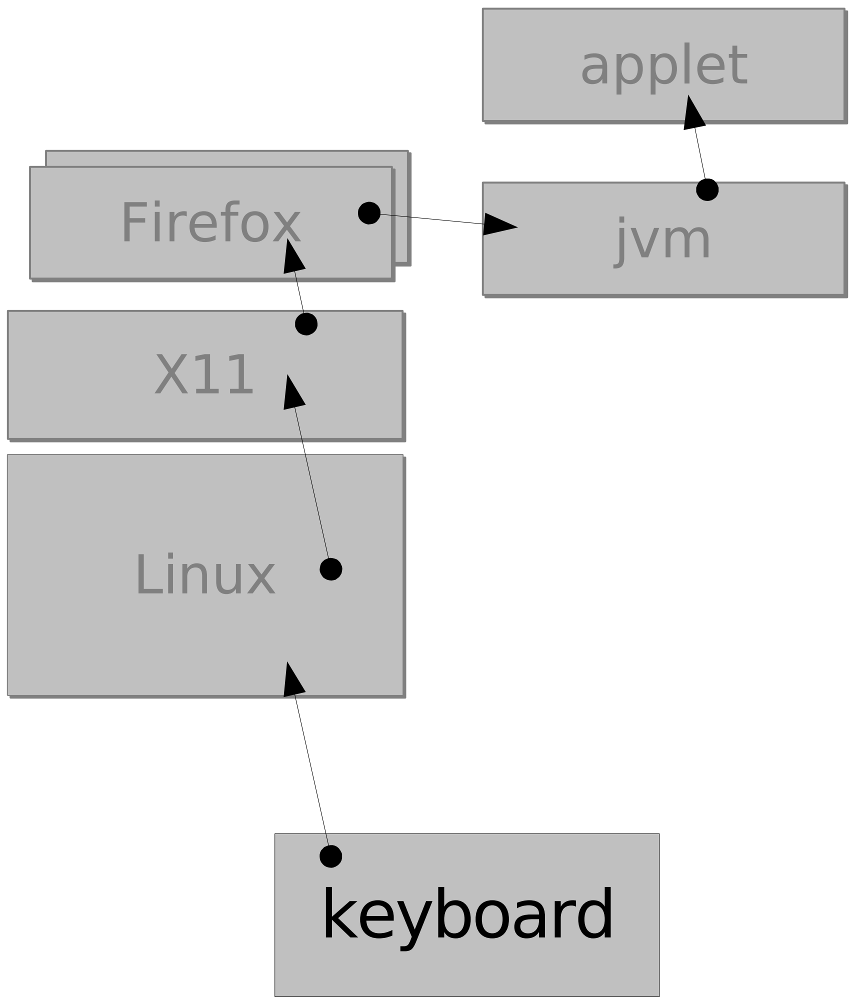
10 Years L4-Based Systems

L4/Nizza Secure-System Architecture

Hermann Härtig
et al. mult.

TU Dresden Operating Systems Group

Your Passwords, Secrets, ...



source:

Understanding Data Lifetime via Whole System Simulation Jim Chow, Ben Pfaff, Tal Garfinkel, Kevin Christopher, and Mendel Rosenblum, Stanford University Usenix Security 04

Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

Outline

L4 etc

- the microkernel vision
- early experience: MACH etc
- what is L4 ?
- L4 and legacy: L4Linux and DDE
- DROPS: L4 and Real-Time
- L4Env: a multi-server environment for L4 apps
- major L4 projects

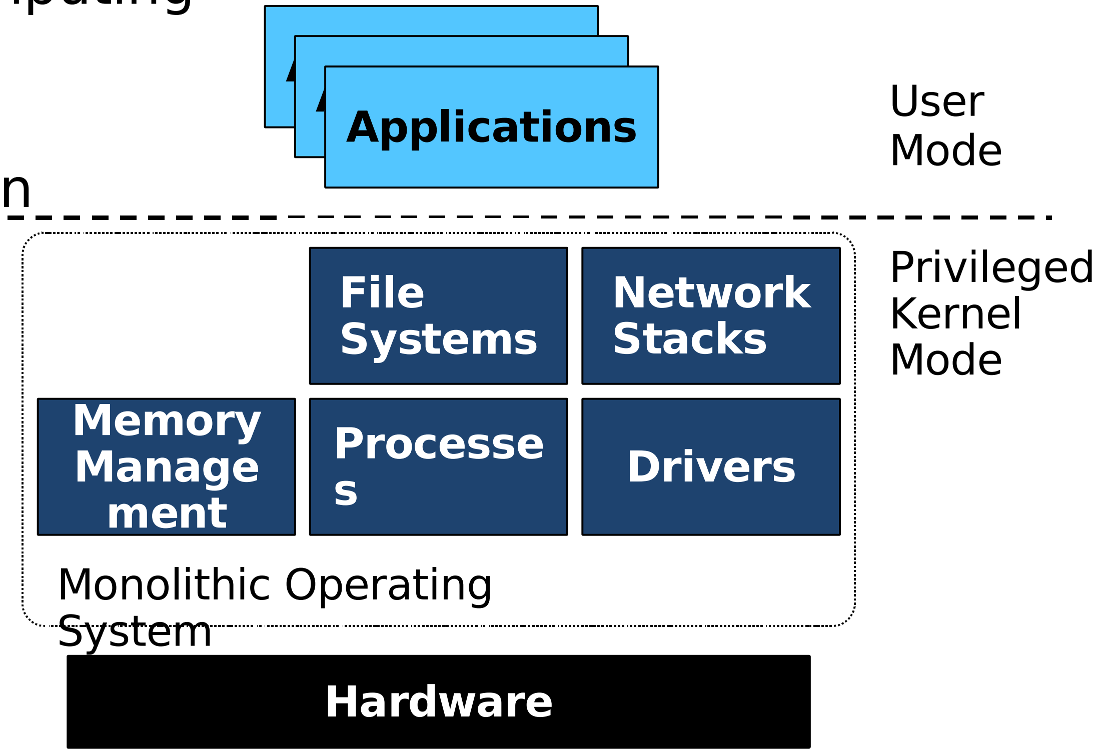
L4/Nizza Secure System Architecture

What's Up Next?

Conclusion

Microkernels - vision and earlier experience

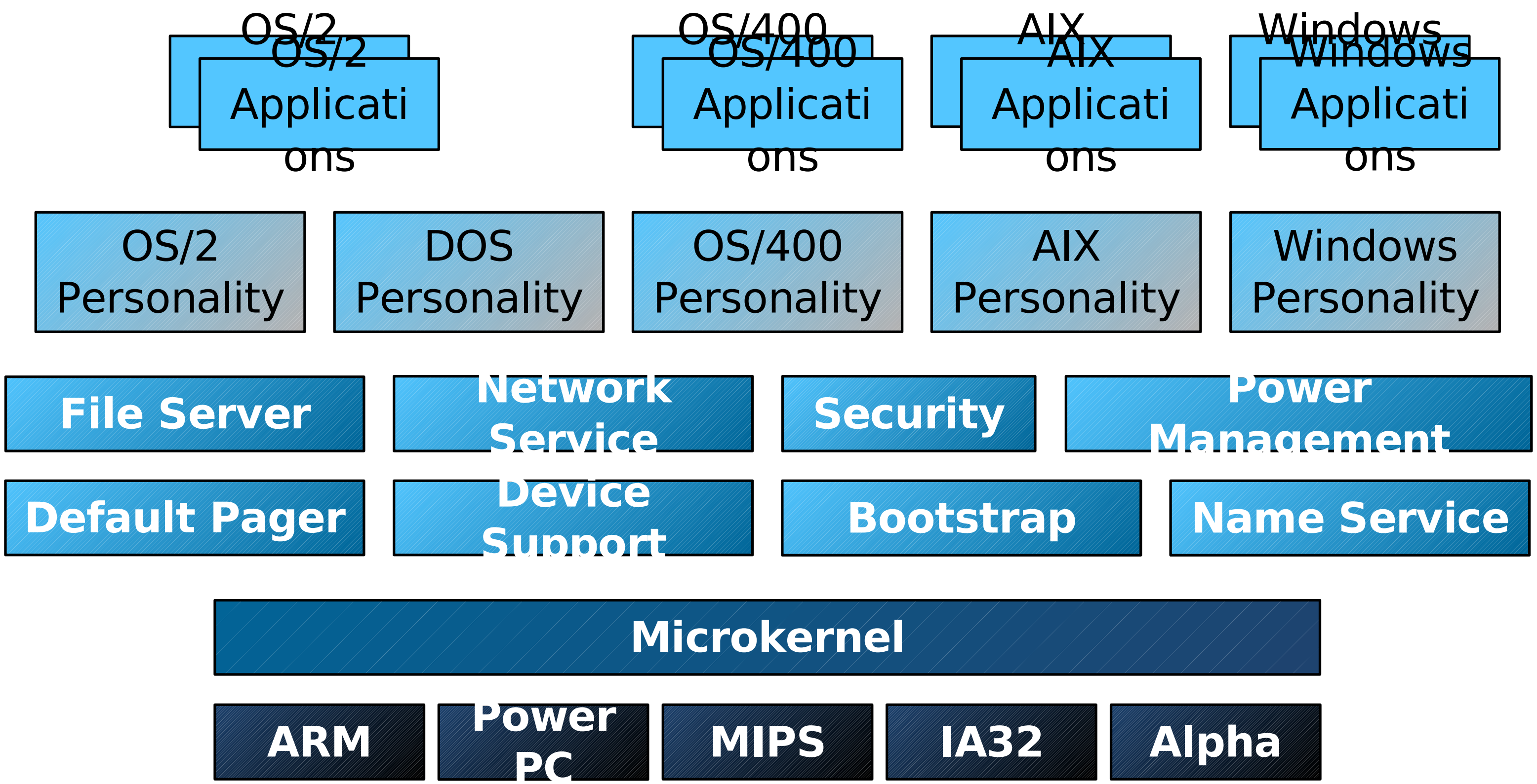
- monolithic systems
  - large
  - complex
  - hard to add real-time
  - large trusted computing bases
  - new additional components often crash system



The Microkernel Vision

- small operating system kernel
  - kernel-mode action less error prone
  - allows strict validation
- system services implemented as user-level servers with their own address spaces
  - flexibility
  - extensibility
  - customizable
- more robust systems
  - protected individual system components (e.g., drivers)
  - small trusted computing base
  - allow coexistence of different OS personalities
- reuse legacy OS (slightly modified)

IBM Workplace OS



Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

Reality in Mid 90ties: MACH-Based Systems

- disappointments
  - performance
  - complexity
  - drivers back in kernel
- e.g., IBM is said to have invested and lost over 1 Billion US $

L4 Microkernel

Jochen Liedtke(ca 96):

"A microkernel does no real work,
but does it efficiently"

- kernel provides only inevitable mechanisms
  no policies enforced by the kernel

what is inevitable?

- address spaces
- threads & scheduling
- inter process communication

L4/Fiasco(ca 98): first HLL / Real-Time scheduling

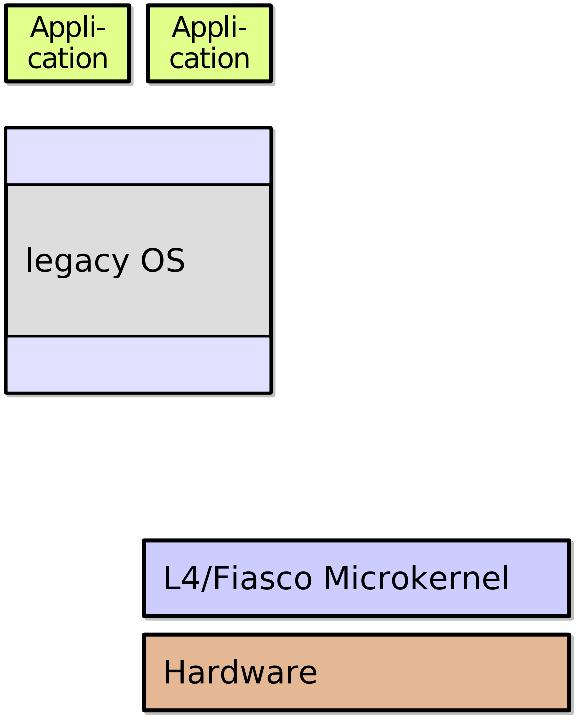
Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

TUDOS: Emphasis on Real-Time and Security

approach

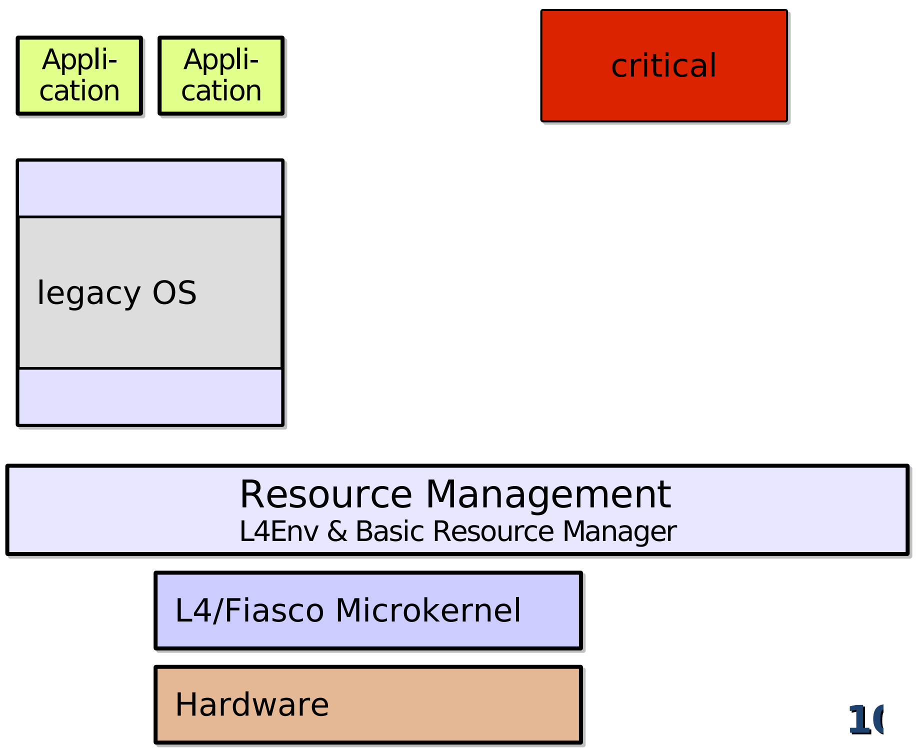
- run legacy software on legacy OS



TUDOS: Emphasis on Real-Time and Security

approach

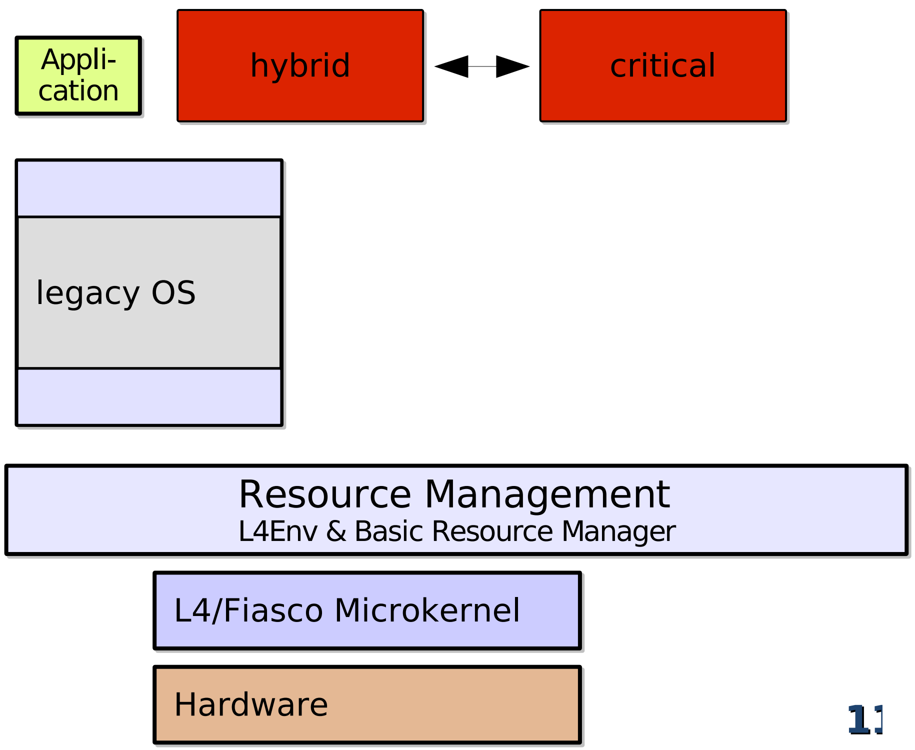
- run legacy software on legacy OS
- run critical applications besides legacy OS



TUDOS: Emphasis on Real-Time and Security

approach

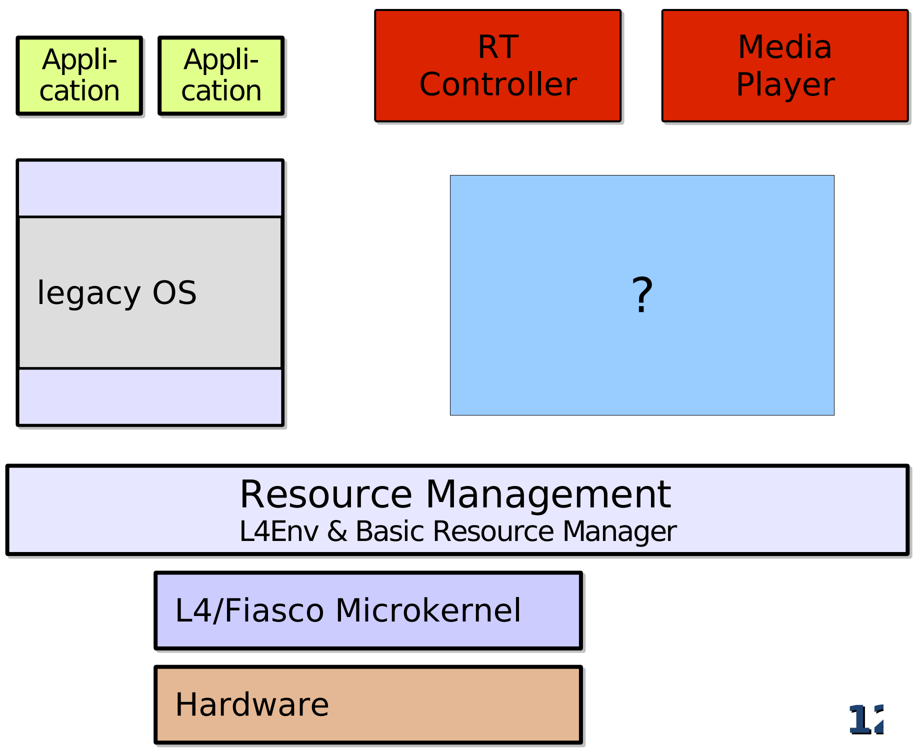
- run legacy software on legacy OS
- run critical applications besides legacy OS



TUDOS: Emphasis on Real-Time and Security

approach

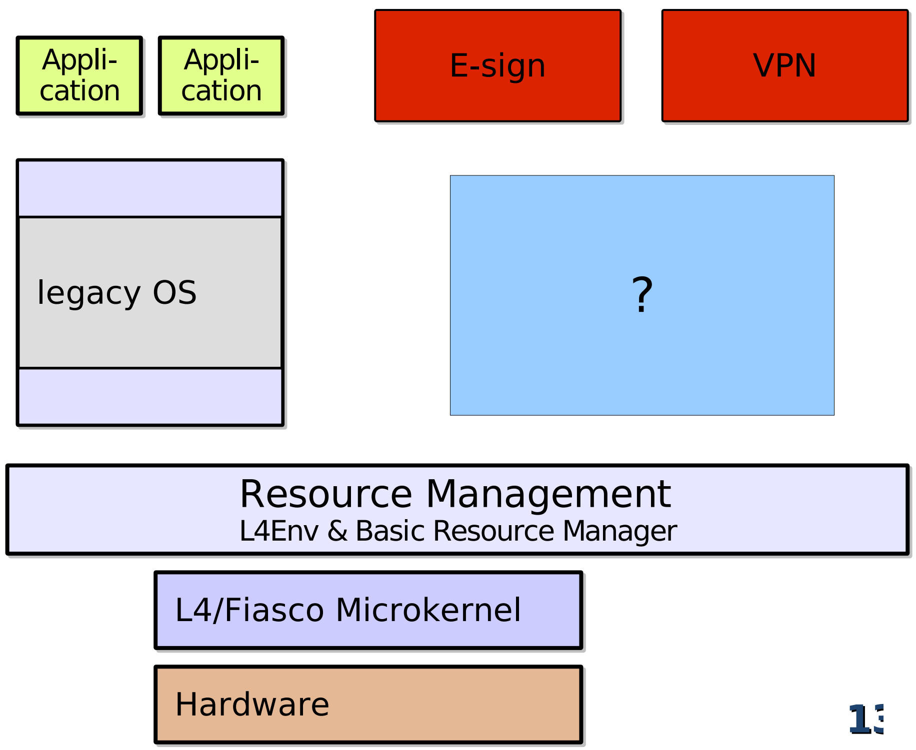
- run legacy software on legacy OS
- run critical applications besides legacy OS
  - real-time



TUDOS: Emphasis on Real-Time and Security

approach

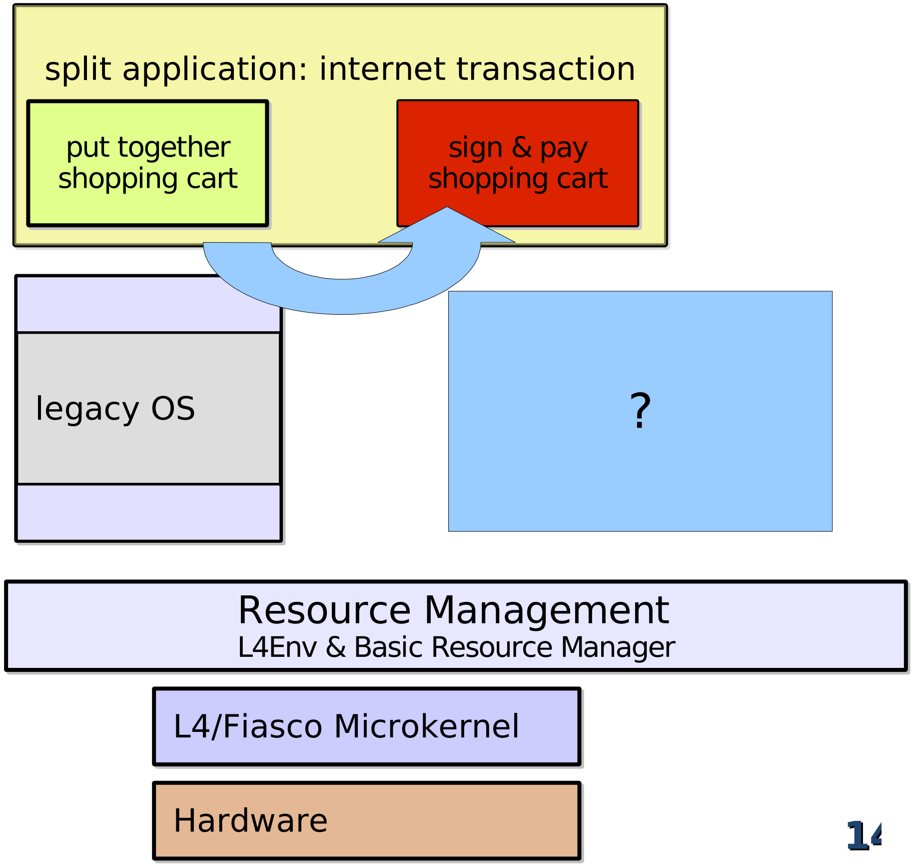
- run legacy software on legacy OS
- run critical applications besides legacy OS
  - real-time
  - high security



TUDOS: Emphasis on Real-Time and Security

approach

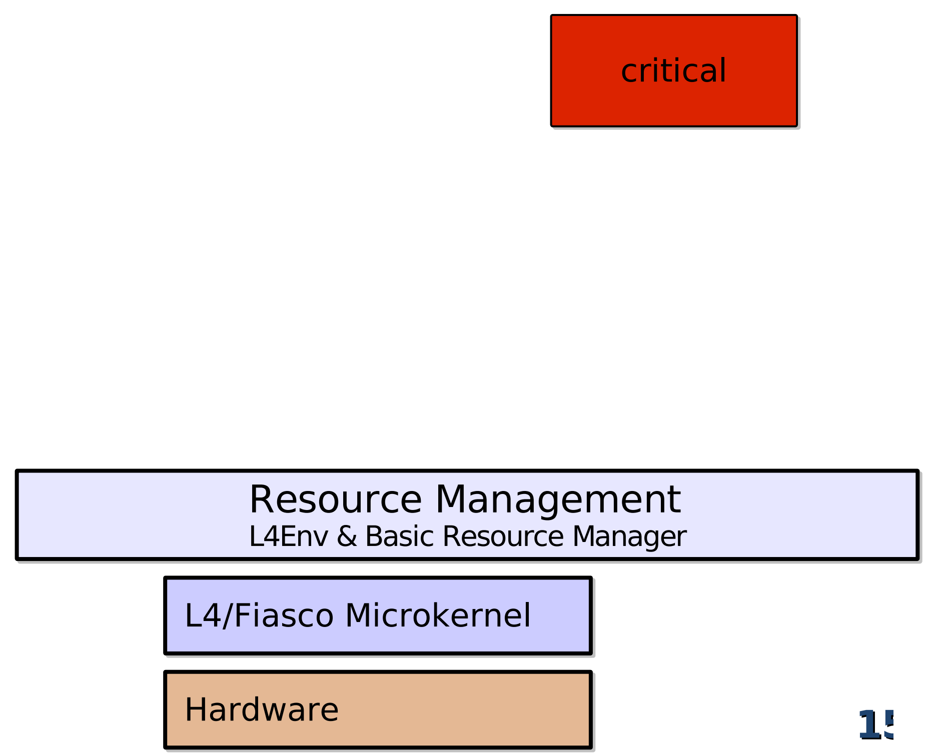
- run legacy software on legacy OS
- run critical applications besides legacy OS
  - real-time
  - high security
- split applications and reuse legacy software for uncritical parts



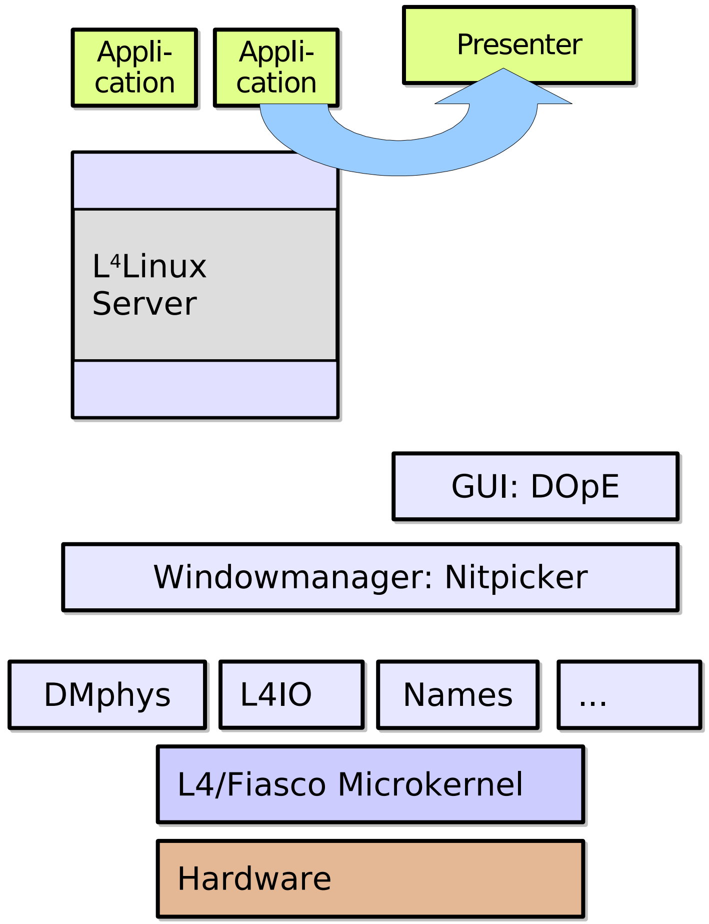
TUDOS: Emphasis on Real-Time and Security

approach

- run critical applications ithout legacy OS



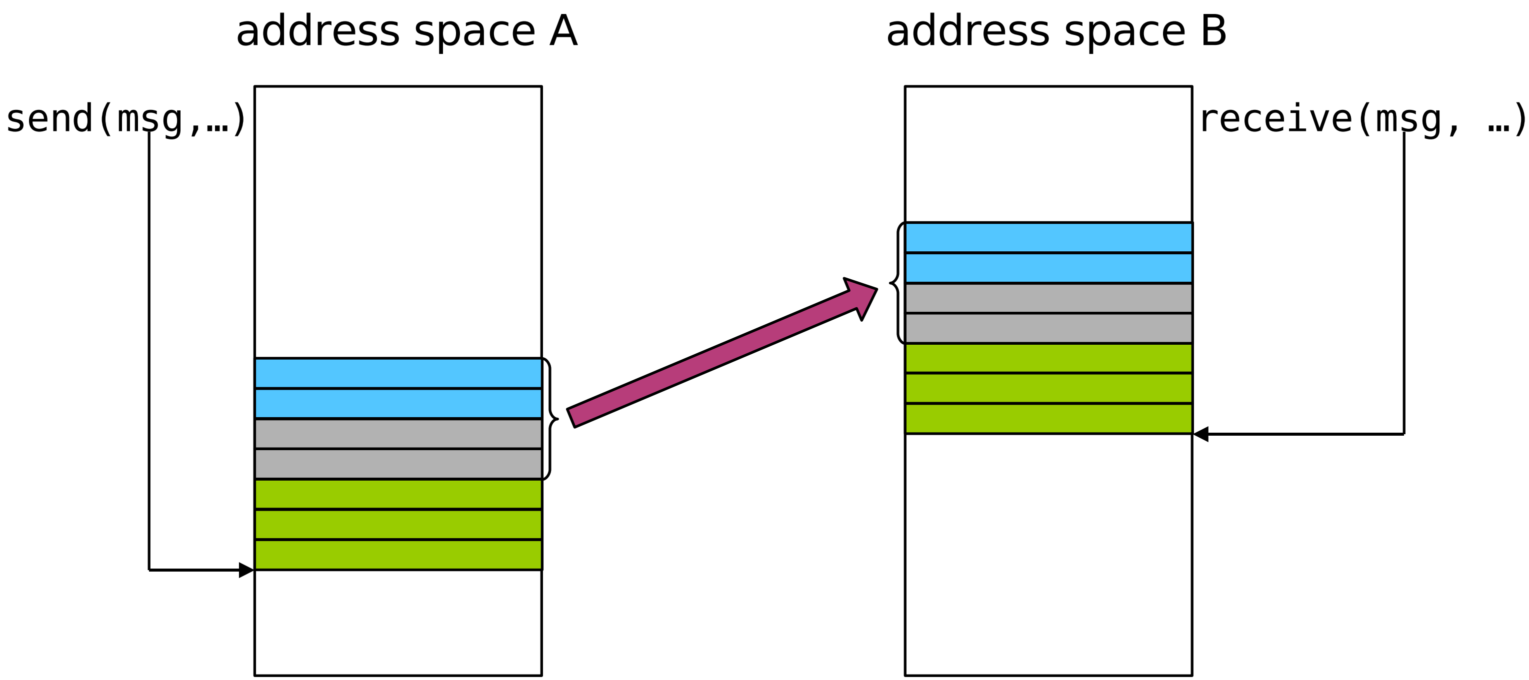
What you see ...



Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

L4 IPC



- synchronous (no buffering)
- diverse payloads

L4 IPC Payloads

- registers only (short IPC), fast
- strings (long IPC)
- access rights ("mappings")
  - memory pages
    transfer page table entries
  - IO ports
  - ...

  can be revoked ("unmap")
- faults
- interrupts

Legacy Software for L4: L^4Linux and DDE

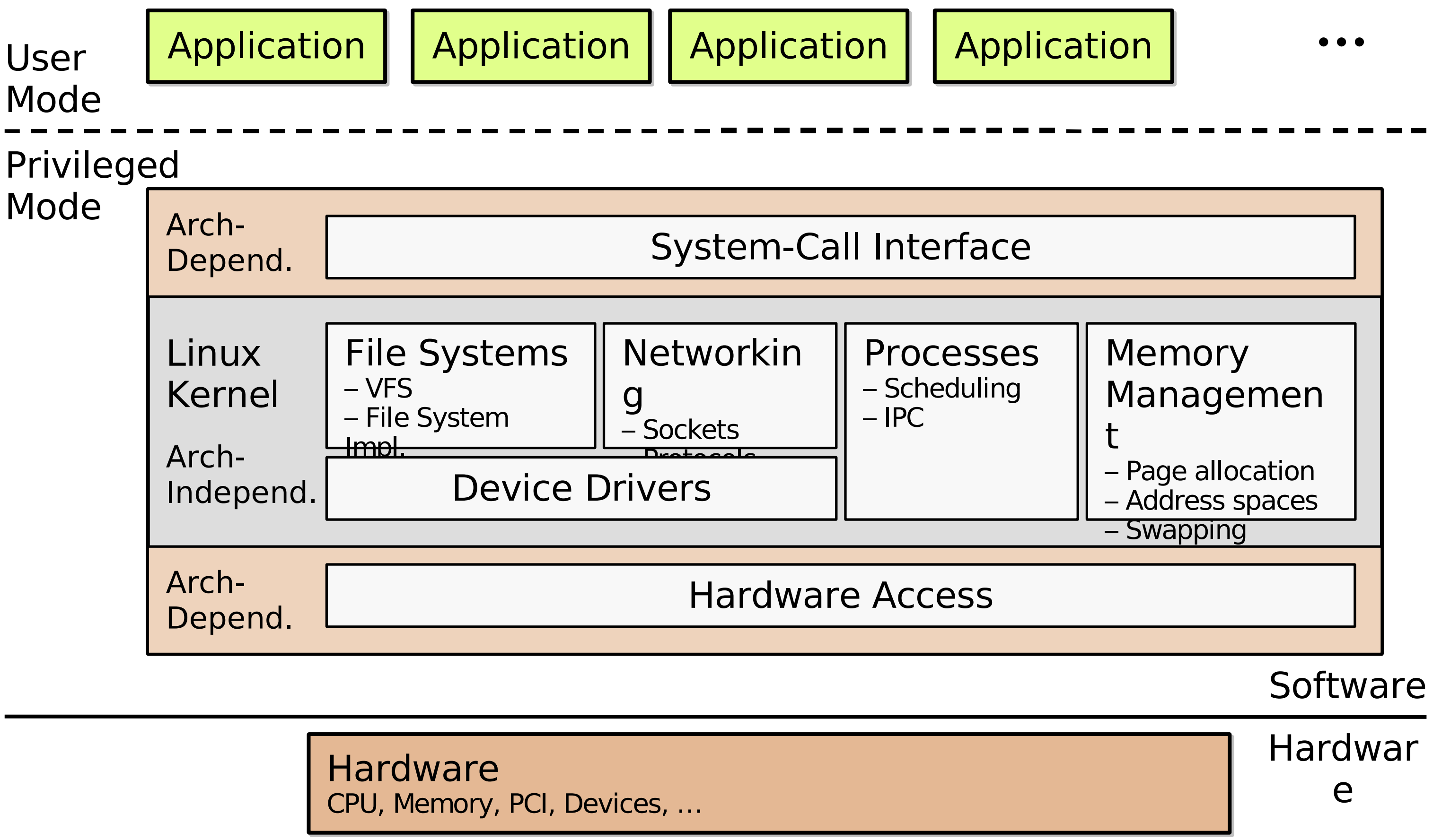
objectives

- inherit large base of legacy software binary compatible
- get it out of the way for more interesting applications
- but reuse it also for interesting applications

and

- reuse drivers

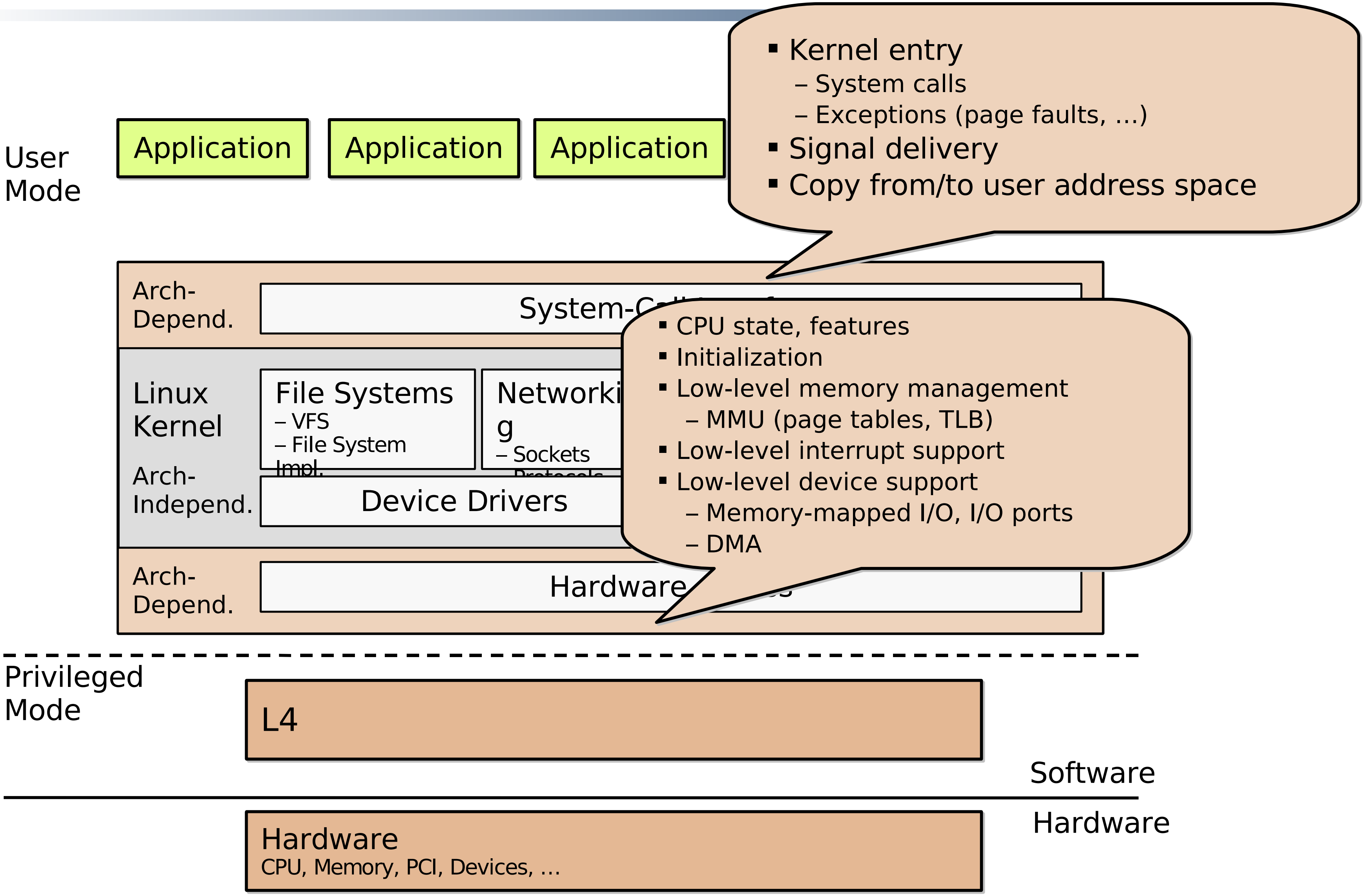
Linux Kernel Structure



Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

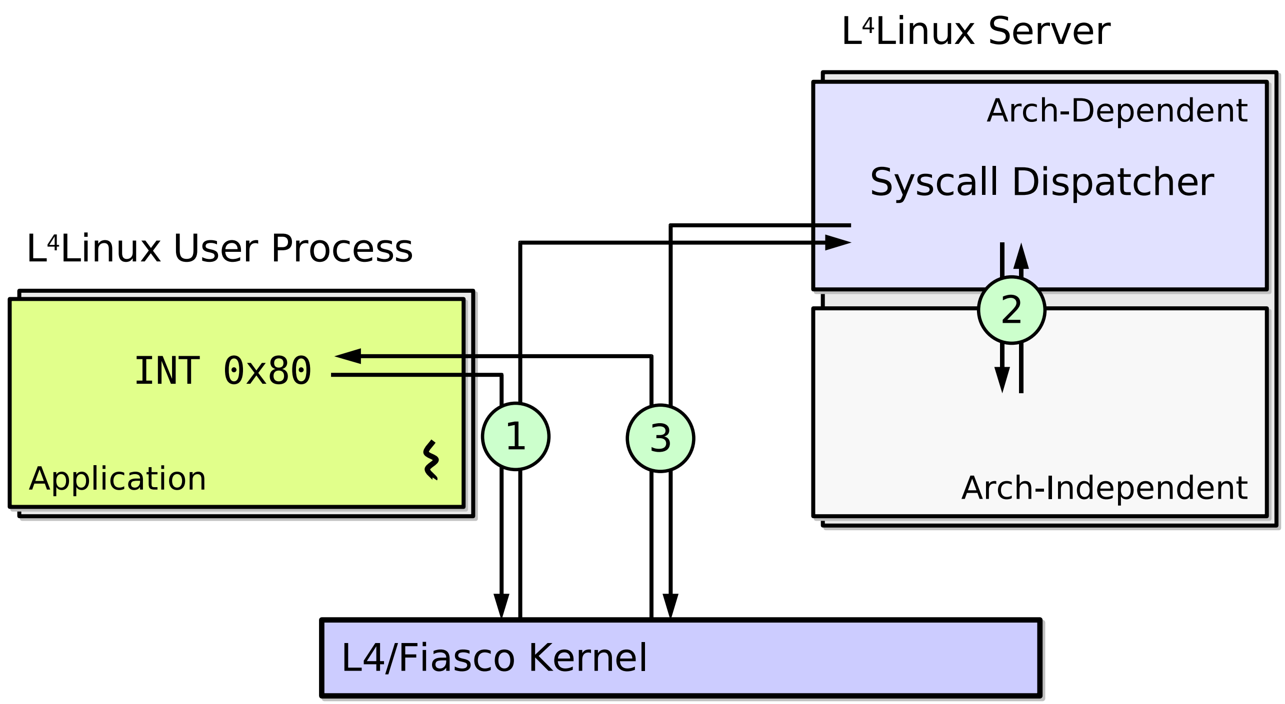
Linux Kernel Structure



Hermann Härtig et al. mult.

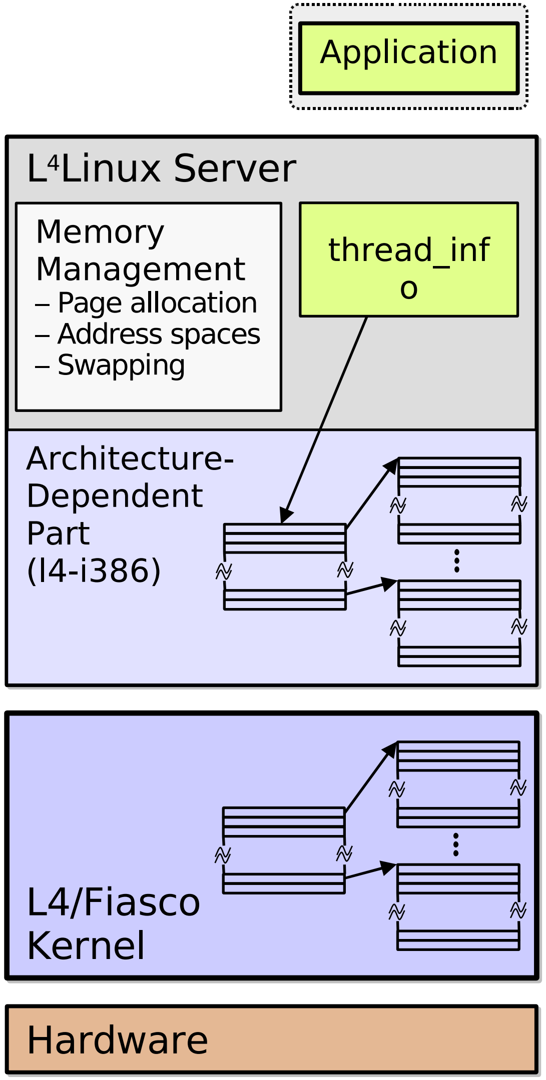
TU Dresden Operating Systems Group

Linux Systemcalls

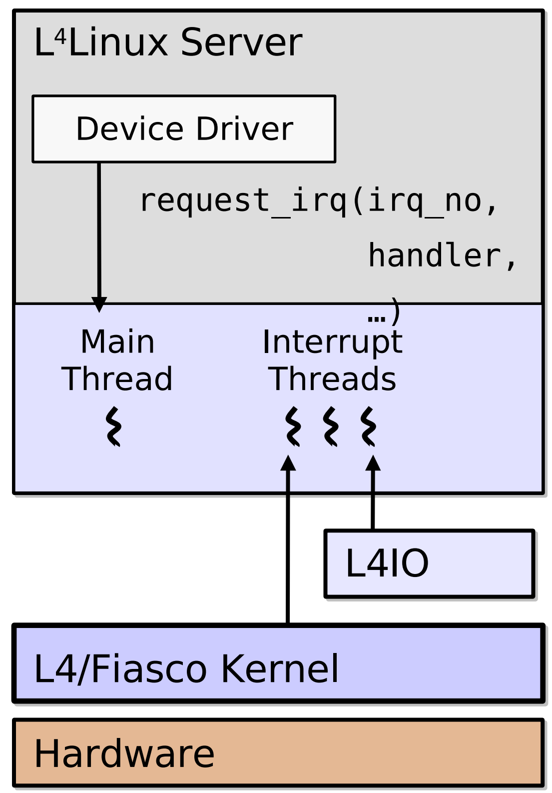


- L^{4}Linux server receives exception IPC
- L^{4}Linux server handles system call
- L^{4}Linux server sends an exception reply
- L4 kernel receives reply and sets new state of thread

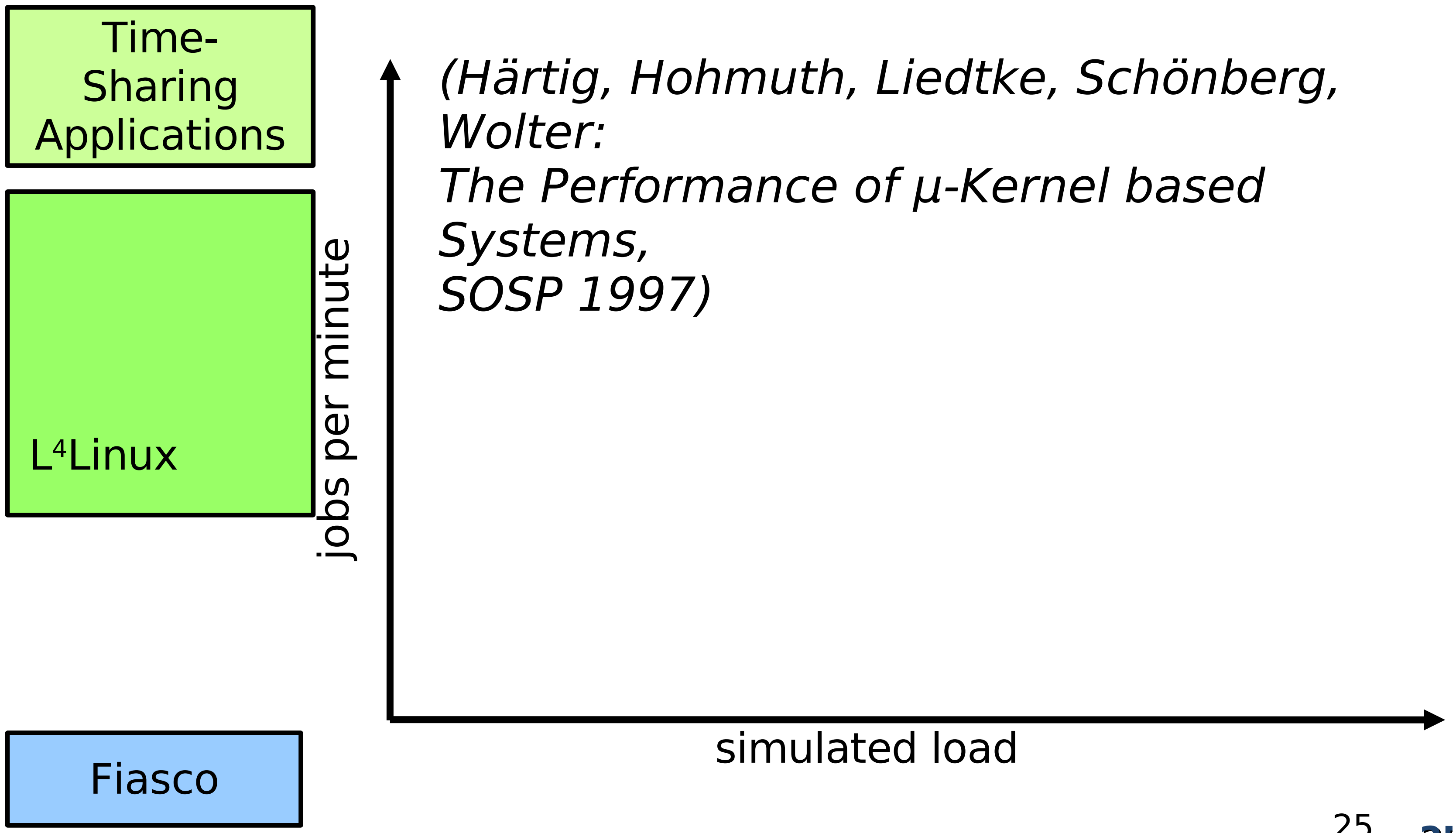
Address Spaces



Interrupts

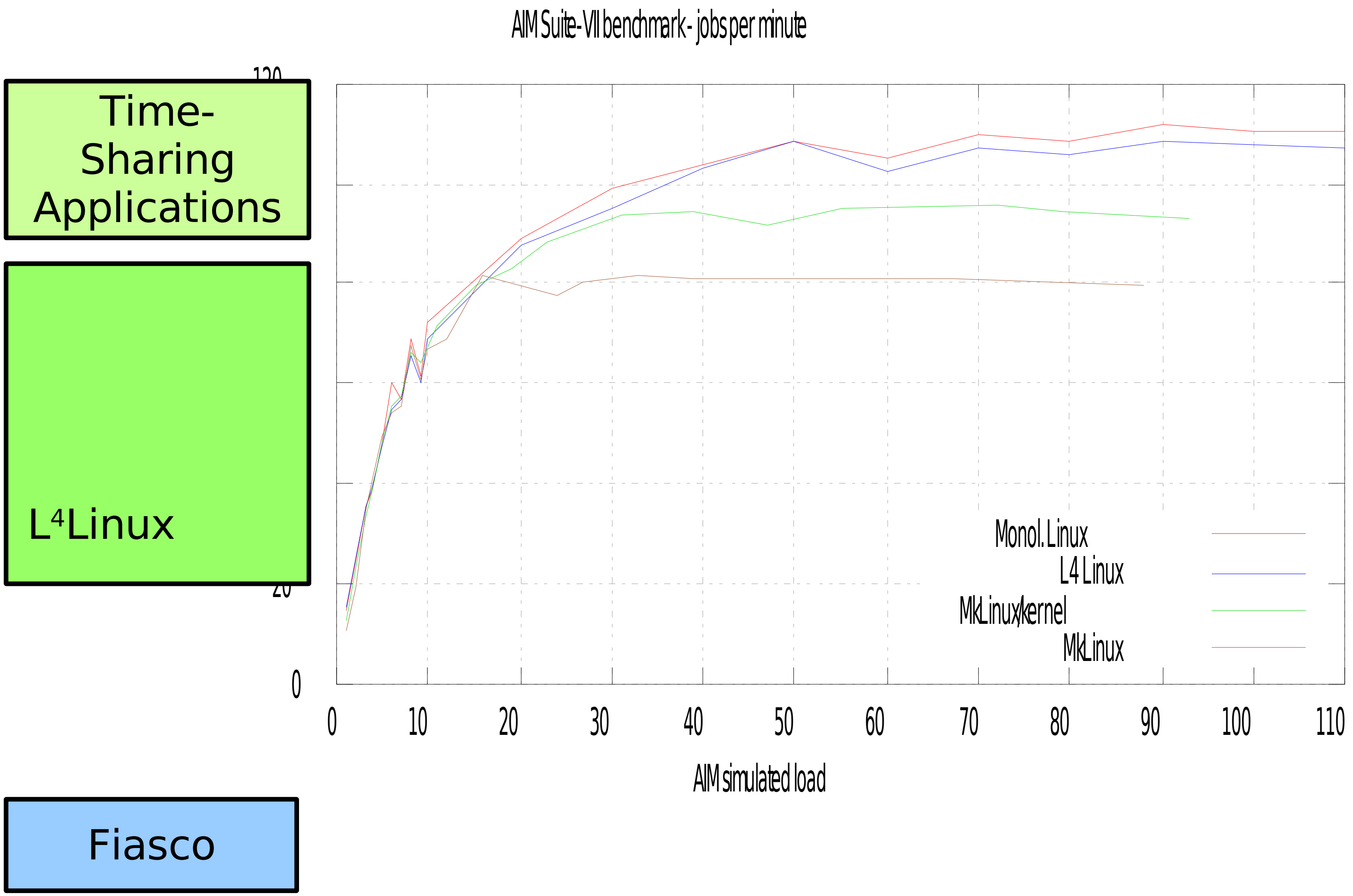

L4Linux Server
Device Driver
request_irq(irq_no,
handler,
...)
Main
Thread
Interrupt
Threads
L4IO
L4/Fiasco Kernel
Hardware

L^{4}Linux Performance



25

L4Linux Performance



Later performance results

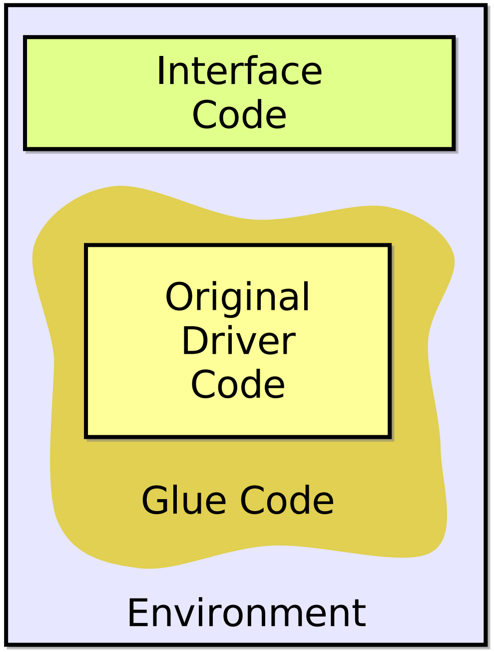
- somewhat worse
- pentium4
  - slower context switches
  - trace caches
  - ...
- L^4Env added overhead
- constant observation needed

- NICTA: L4Linux("Wombat") on ARM is faster than Linux

Legacy Drivers: Device Driver Environment

DDE structure

- unmodified source code of Linux driver is encapsulated by emulation library
- library provides implementation of Linux services as expected by driver



status:

supports Linux 2.4 drivers

L4 and Real-Time: DROPS Dresden Real-Time OPerating S.

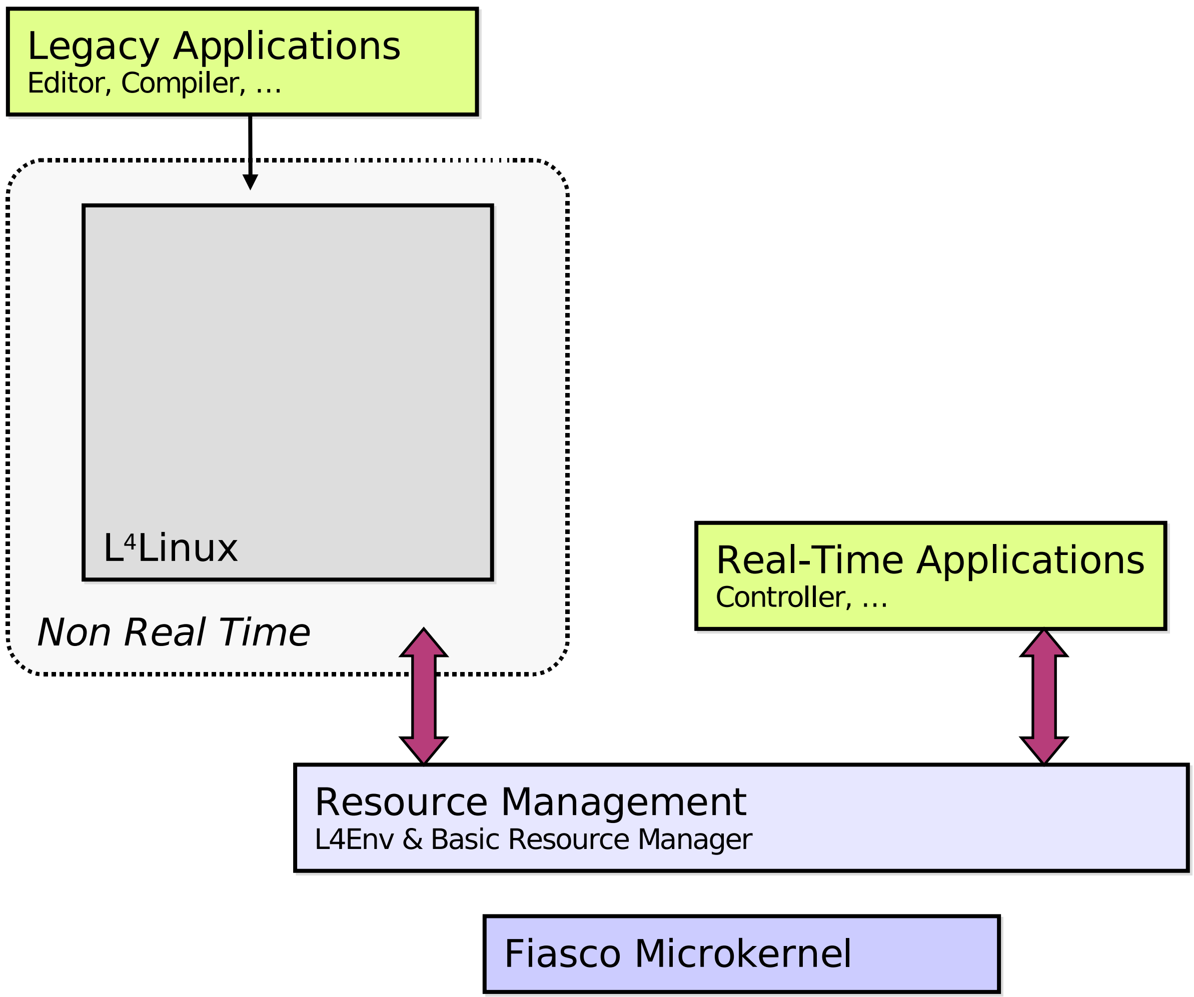
objectives & principles

- Real-Time besides Non-Real-Time L4Linux systems
- protect the RT applications against crashing legacy SW
- resource reservations thru admission procedure
- gracefully handle overload,
  also overload occasionally caused by Real-Time applications (media applications)
- manage multiple resources

Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

Real-Time Applications in Separate Address Spaces

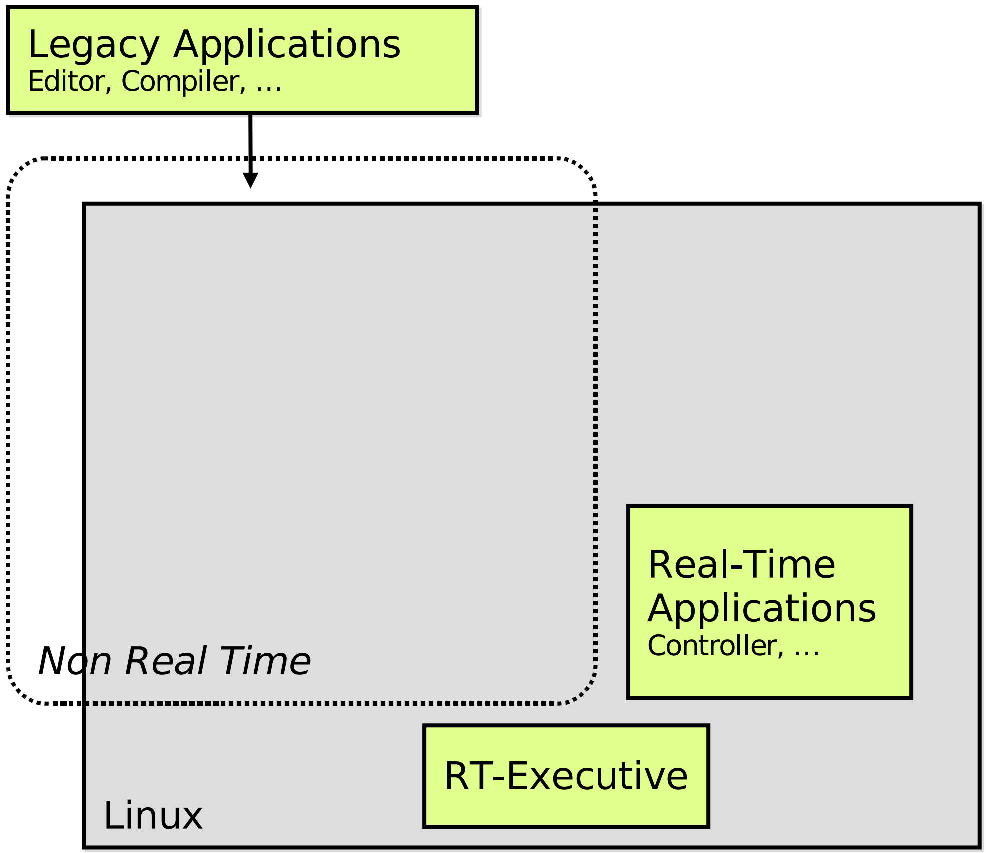


Hermann
Härtig
et al.
mult.

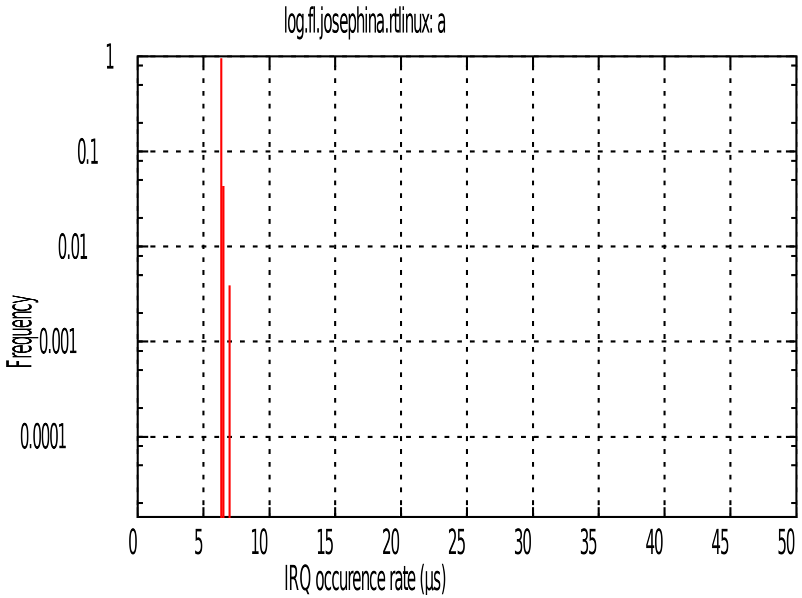
TU
Dresden
Operating
Systems
Group

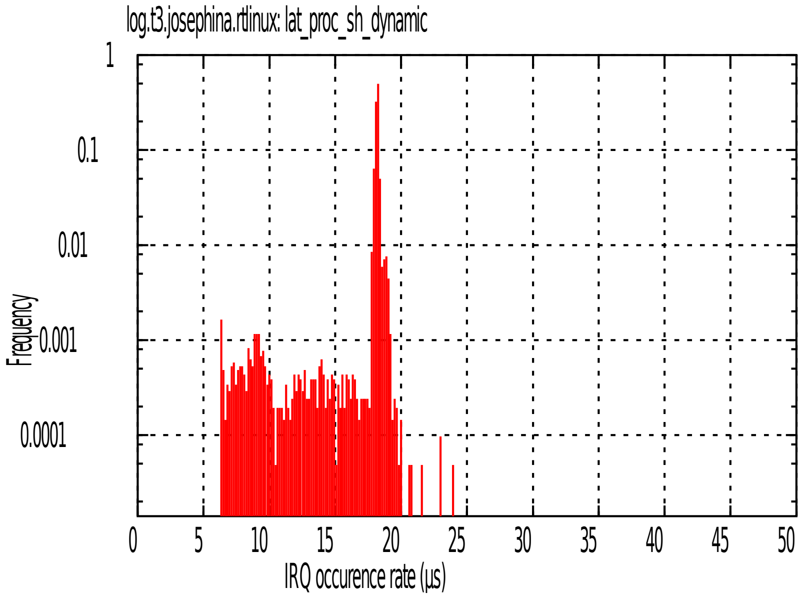
SEVECOM
Budapest
2006

Common Practice, for Example RT-Linux



RT-Linux Latencies, without protection





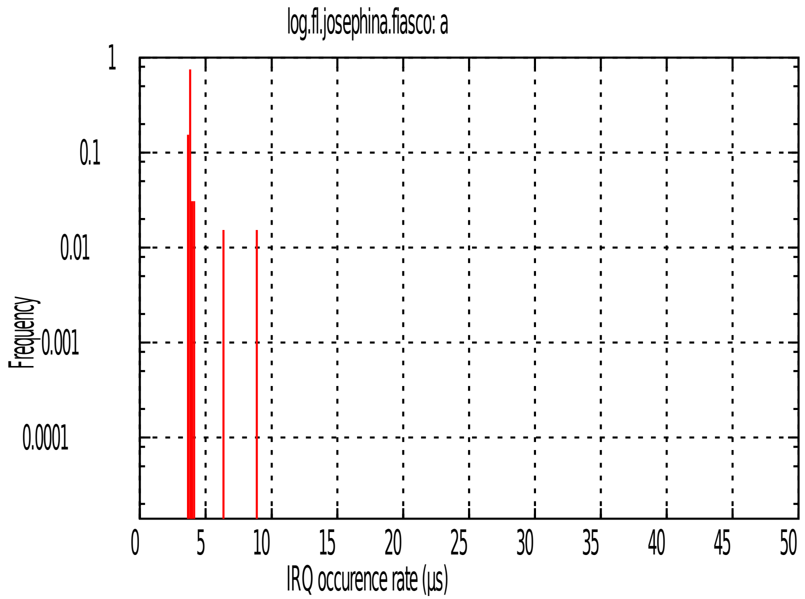
„Interrupt response time: Time from interrupt occurrence until first instruction in RT-task"

No parallel Load: 13µs (idle)

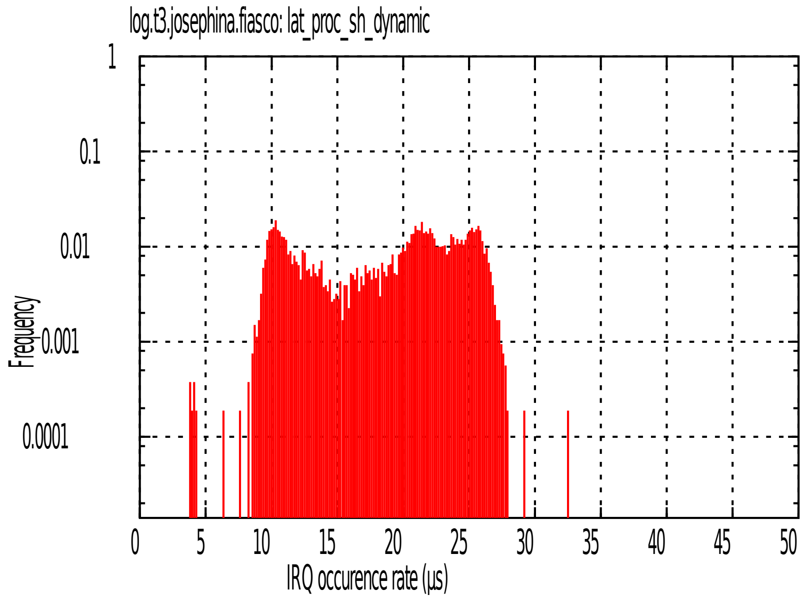
Intel P4
1.6 GHz

High parallel load: 68µs (Benchmark, Cache-Flooder)

L4Linux + RT latencies, with address space protection

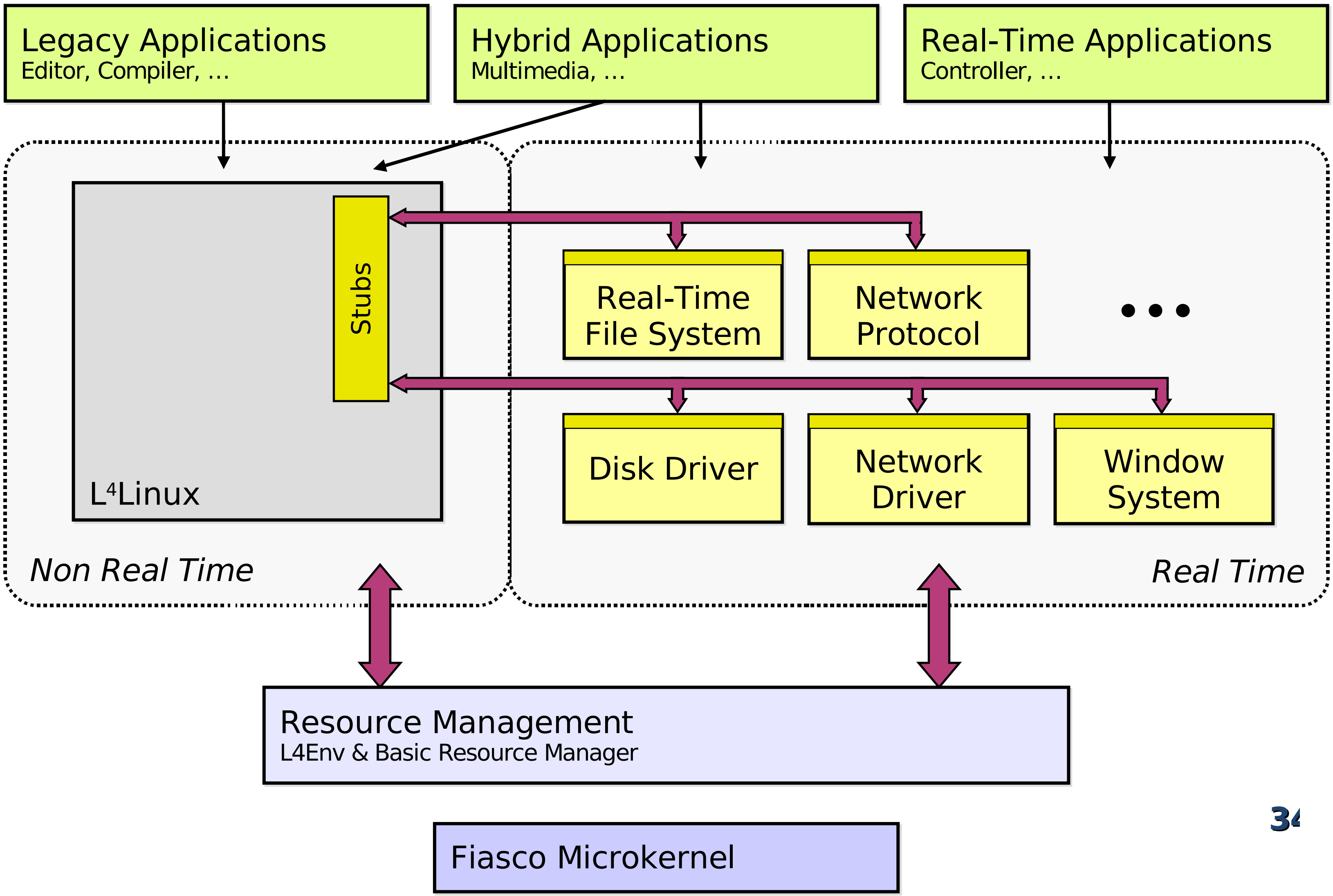


No parallel Load: 43µs
(idle)



High parallel load: 85µs
(Benchmark, Cache-Flooder)

More on DROPS Experiments



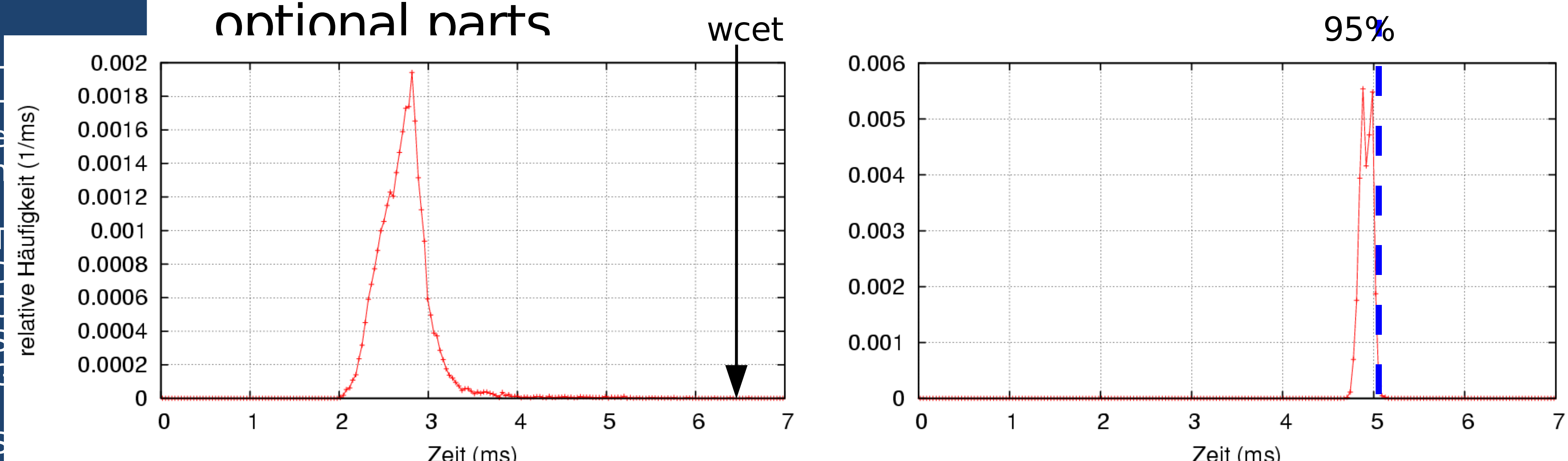
Dresden Real-Time Operating System

Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

Overload gracefully tolerated

- hard real-time must be based on worst-case analysis
- mobile systems cannot afford that in many cases (media applications)
- overload must be tolerated gracefully and predictable
- many applications can be split in mandatory and optional parts



Scheduling and admission

- admit such
  - that all deadline of mandatory applications are met
  - that requested quality (=percentage of optional parts is met)
- schedule based on budgets such that application processes can react on missed deadlines of optional parts and overused budget
- price:
  modify applications

L4Env: a multi-server environment for L4 Applications

supports

- basic resource management
- basic IO handling
- basic naming
- event handling (resource deallocation)
- loading
- ...

based on multiple L4 tasks

Major L4 projects

- IBM started (and forgot) it
- Dresden
  - L4/Fiasco: first L4 in HLL and for RT
  - L4linux, DDE
  - DROPS
  - Nizza
- Karlsruhe
  - L4/Pistacchio, fast and portable
- Sydney
  - embedded
  - portability

set of loosely coupled projects

Outline

L4 etc

L4/Nizza Secure System Architecture

- security objectives
- principles to build
- system objectives
- Nizza principles
- an example: an internet transaction
- more Nizza use cases
- Nizza and “Trusted Computing”

What's Up Next?

Conclusion

Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

SEVECOM
Budapest
2006

Objectives: Security Objectives

- confidentiality
  no unauthorized access to information
- integrity
  information is either intact, complete and up to date or it can be detected otherwise
- recoverability
  no permanent damage to information
- availability
  timeliness of service

Remember: Saltzer & Schroeder 73

- Economy of Mechanism
- Fail-safe Defaults
- Complete Mediation
- Open Design
- Separation of Privilege
- Least Privilege
- Least Common Mechanism
- Psychological Acceptability

L4 and Security: The Nizza Architecure

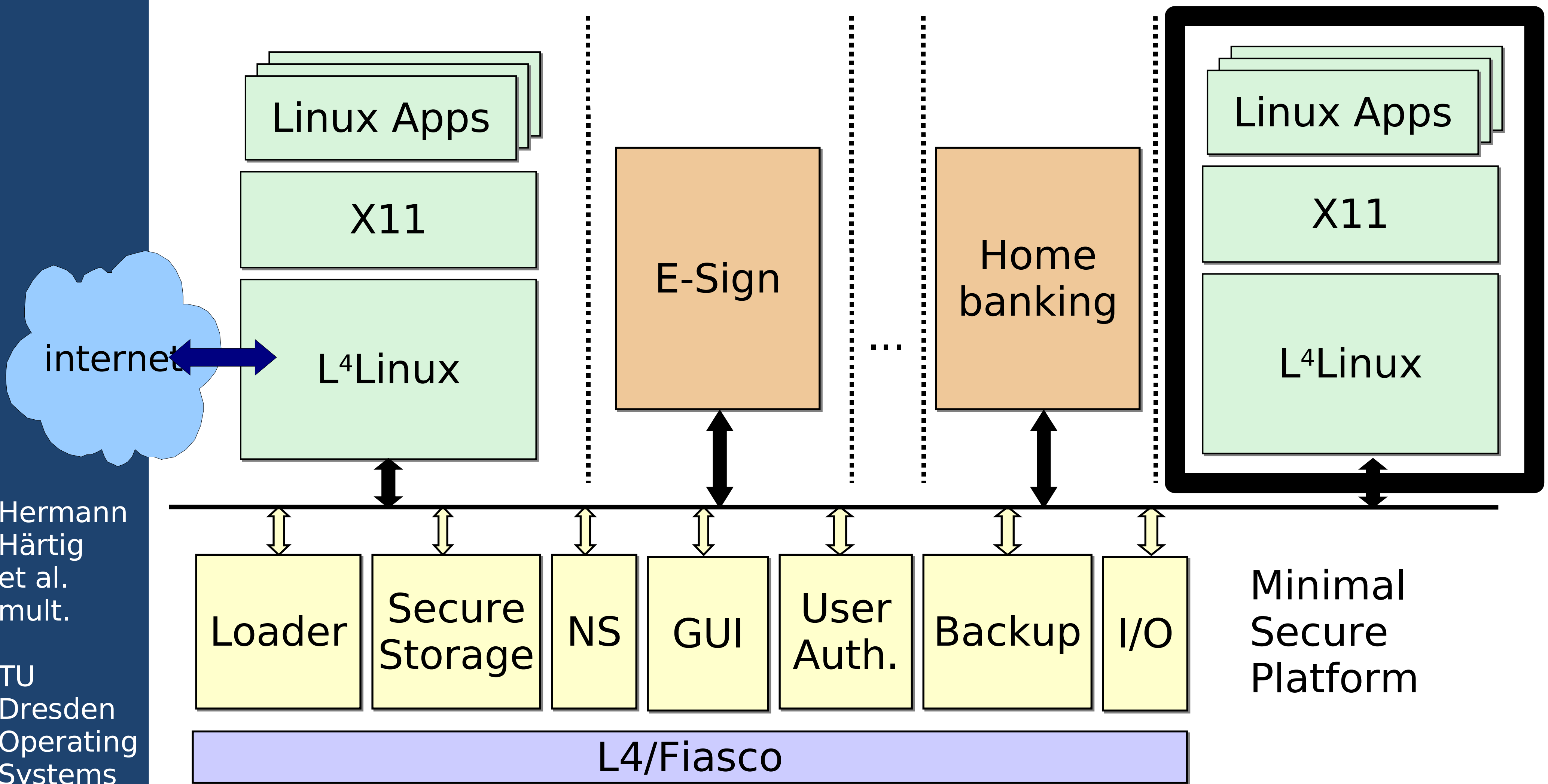
objectives

- critical applications besides L4Linux
  assume: L4Linux successfully penetrated
- reduce complexity for critical part

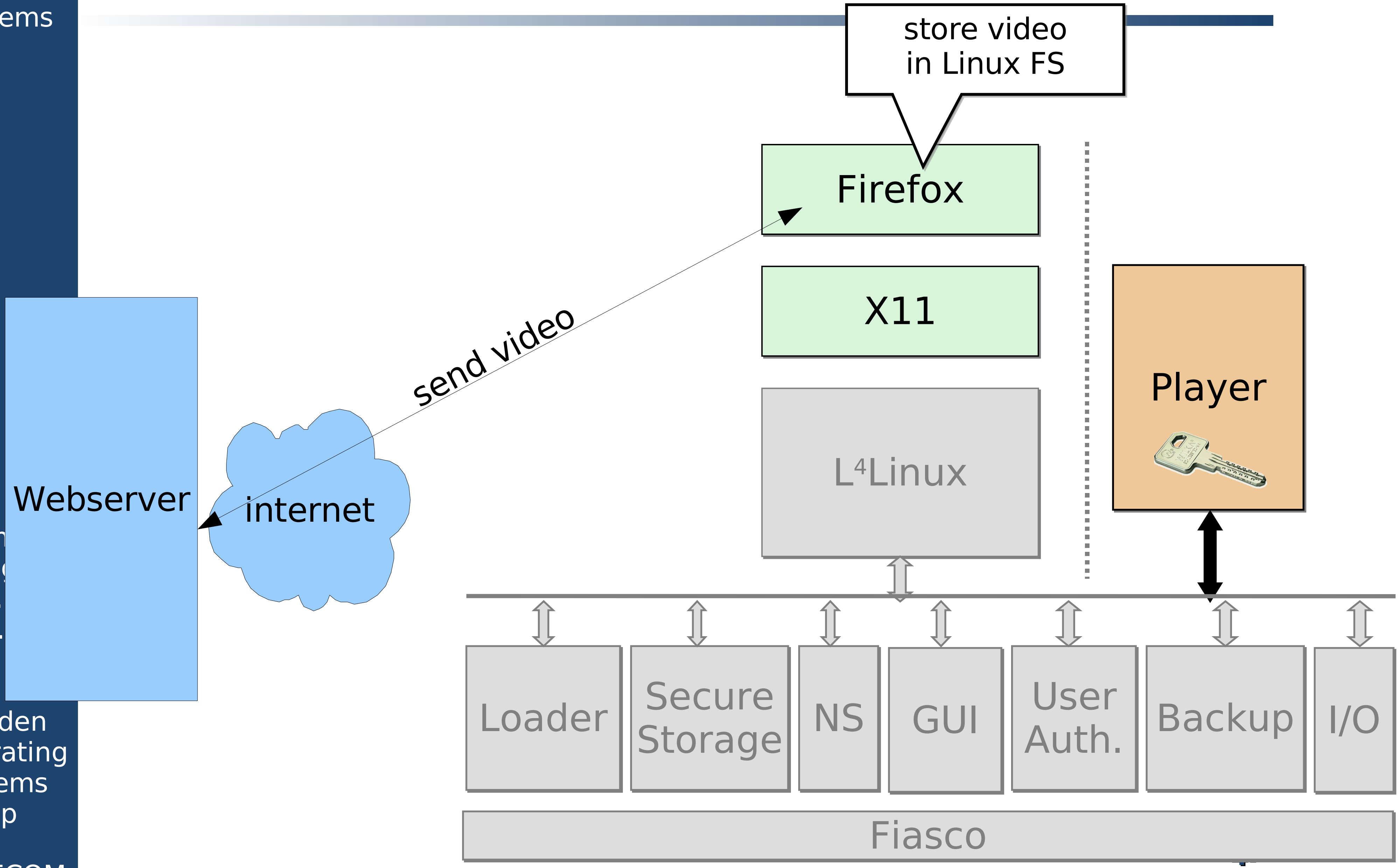
principles

- small trusted computing bases,
  application specific
- extract critical parts of applications
  -> “AppCore”
- reuse L^4Linux with trusted wrappers

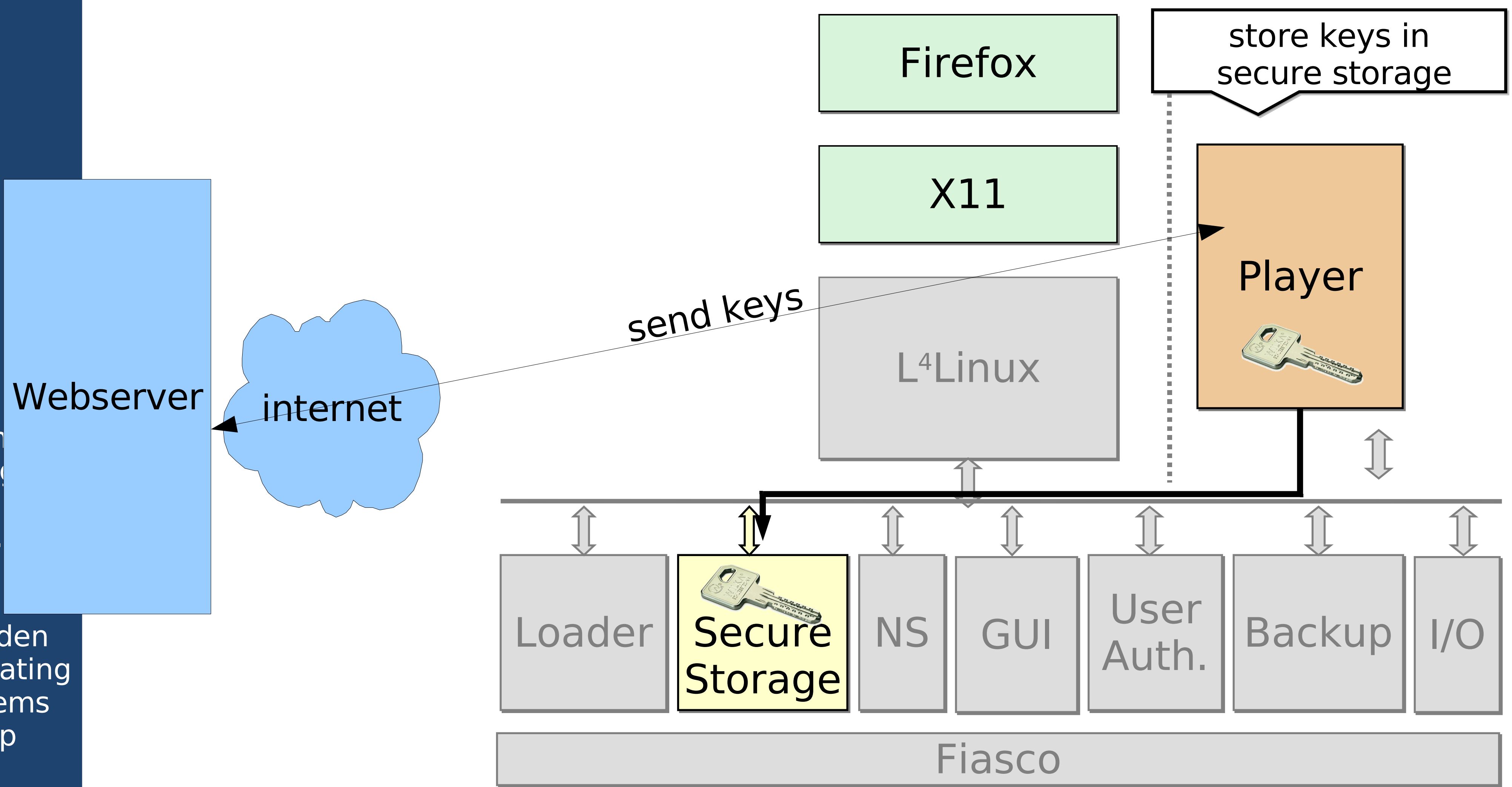
Nizza Example Scenario



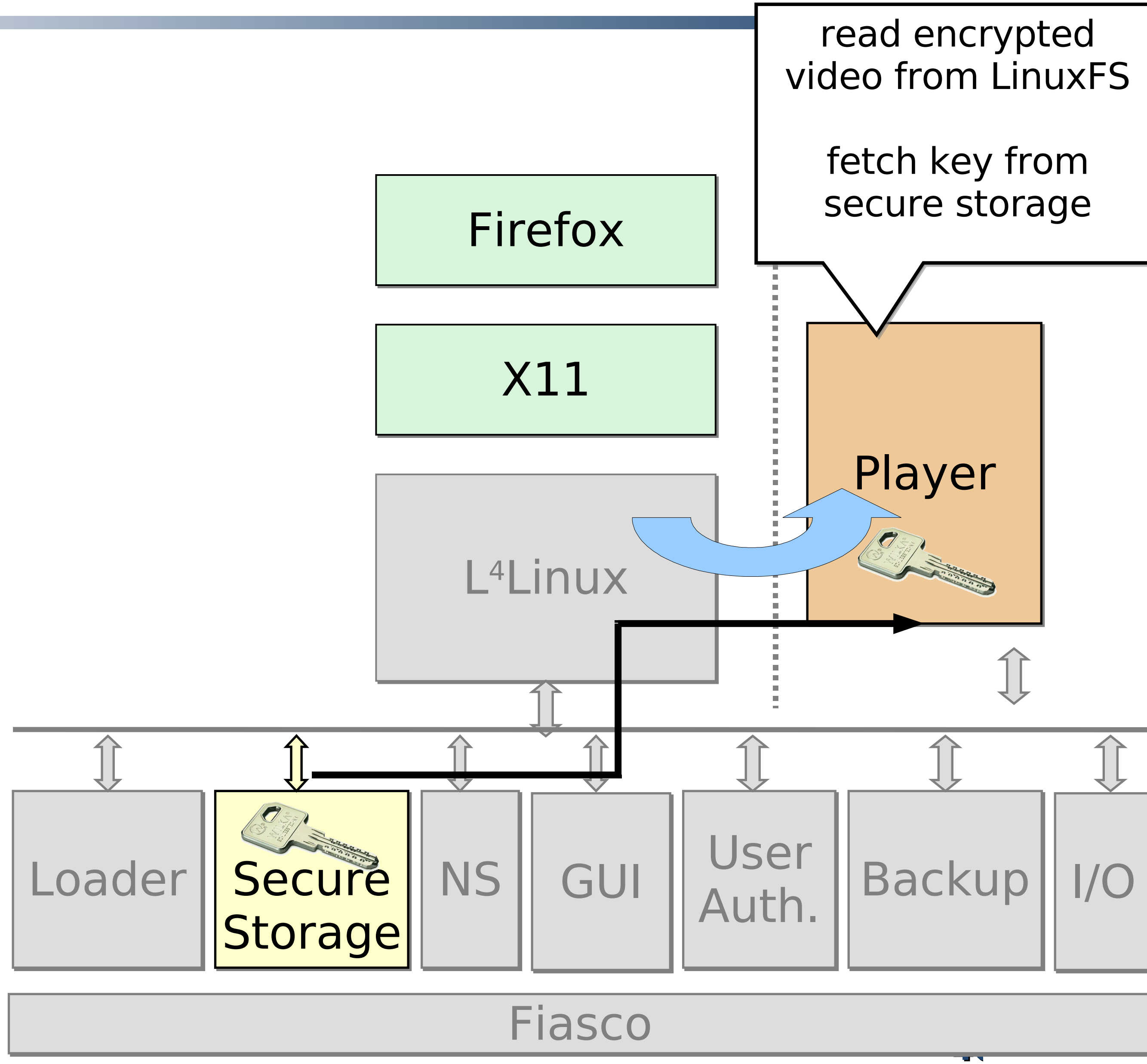
Digital Rights Management



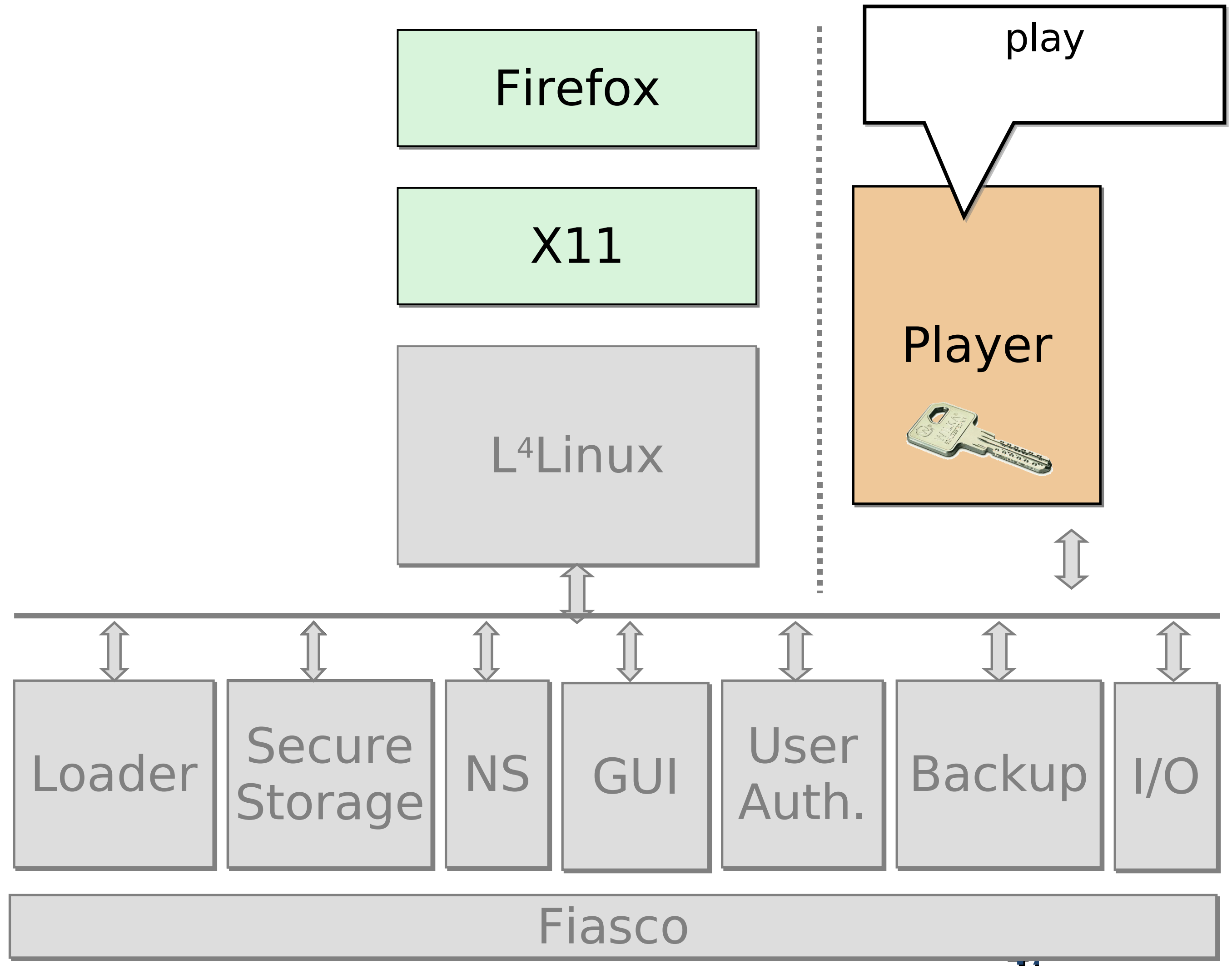
Digital Rights Management



Digital Rights Management



Digital Rights Management



Hermann Härtig et al. mult.

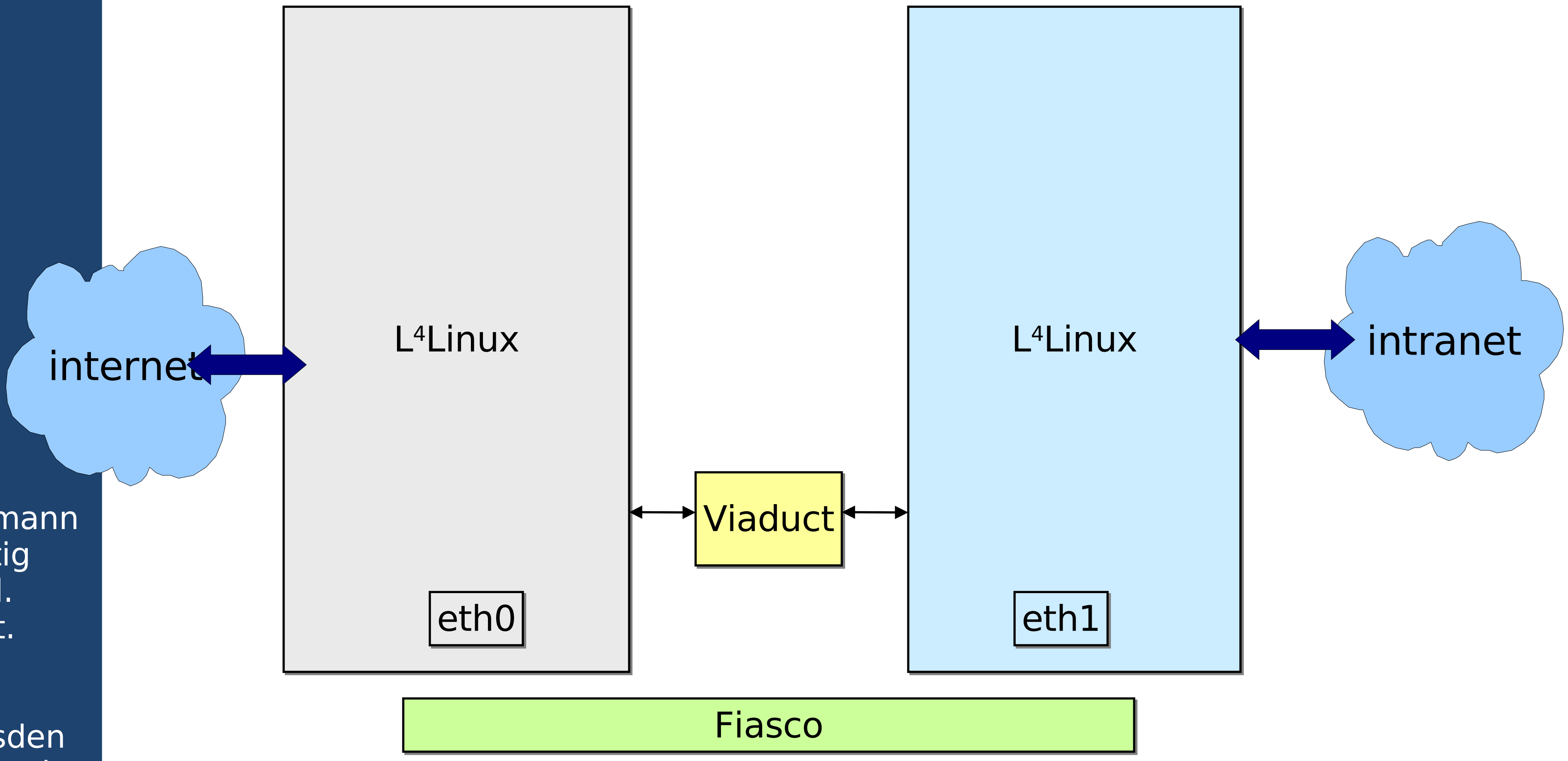
TU Dresden Operating Systems Group

Nizza and “Trusted Computing”

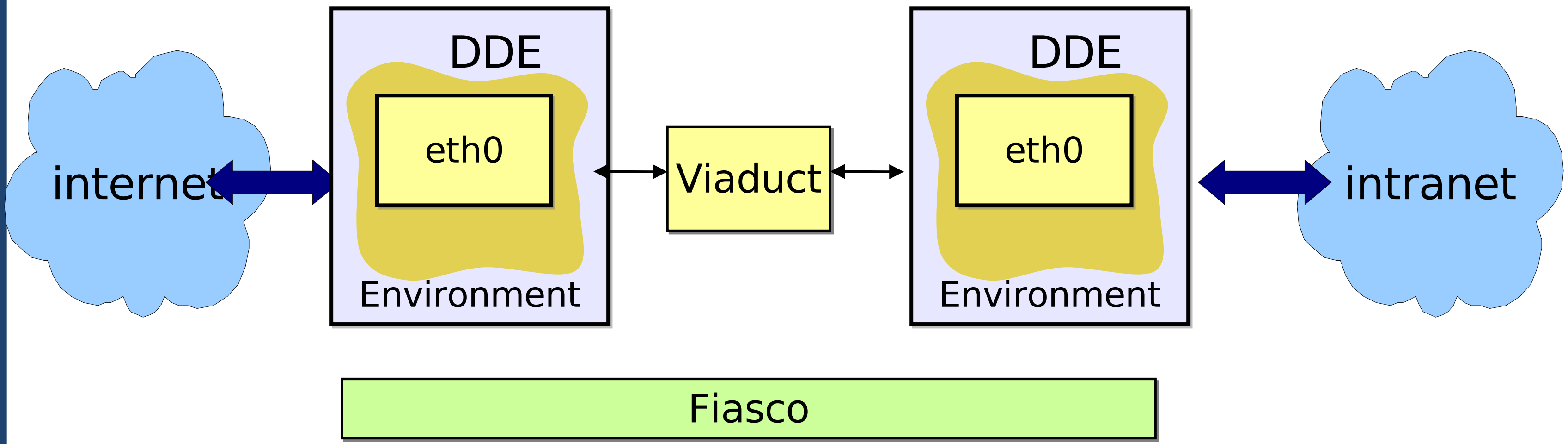
TPMs deliver

- authenticated booting
- remote attestation
- sealed memory

VPN Box


internet
L4Linux
eth0
Viaduct
L4Linux
eth1
intranet
Fiasco

VPN Box



Hermann
Härtig
et al.
mult.

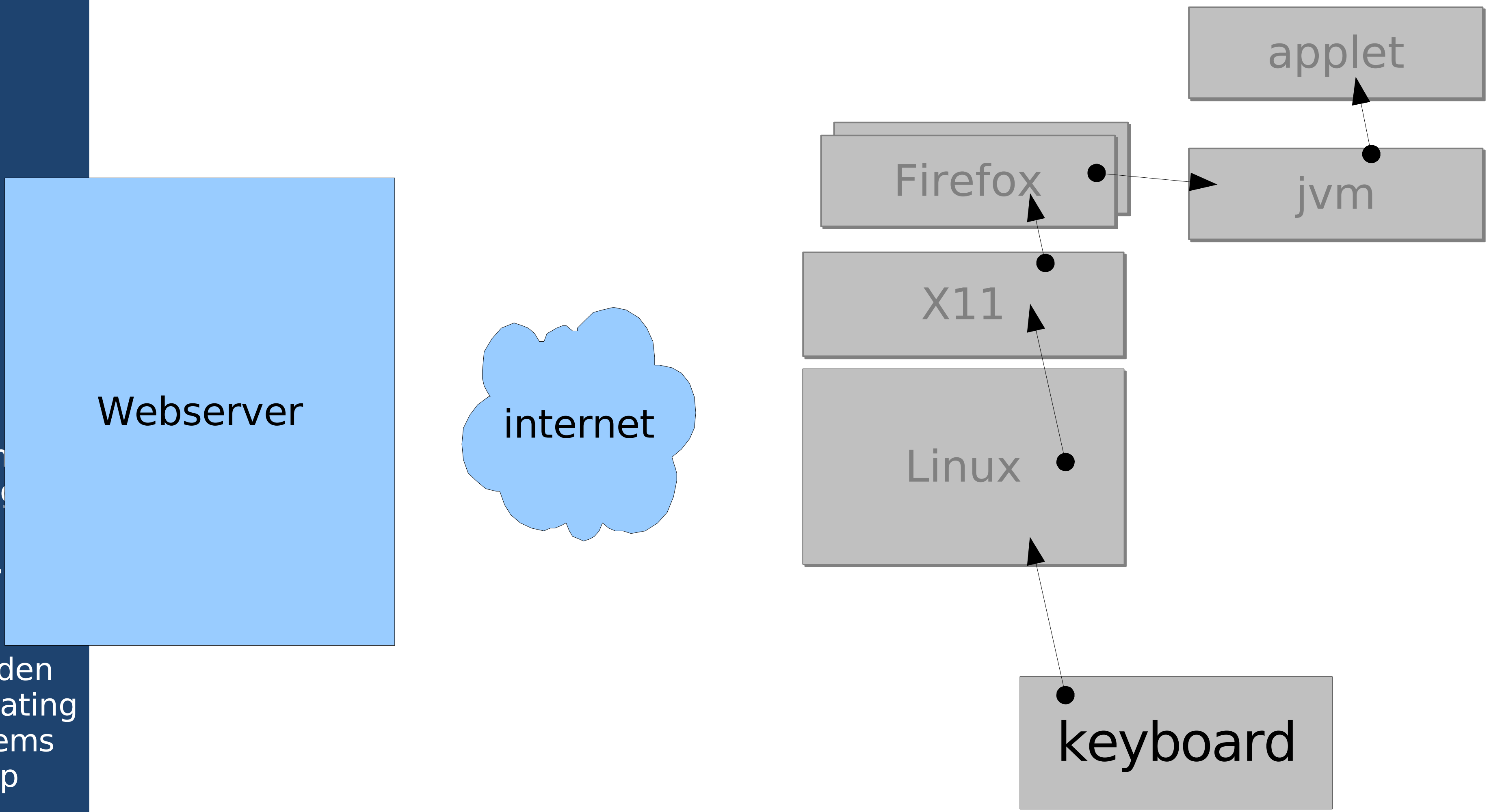
TU
Dresden
Operating
Systems
Group

Internet Transaction: Your password(s), credit card number, ...

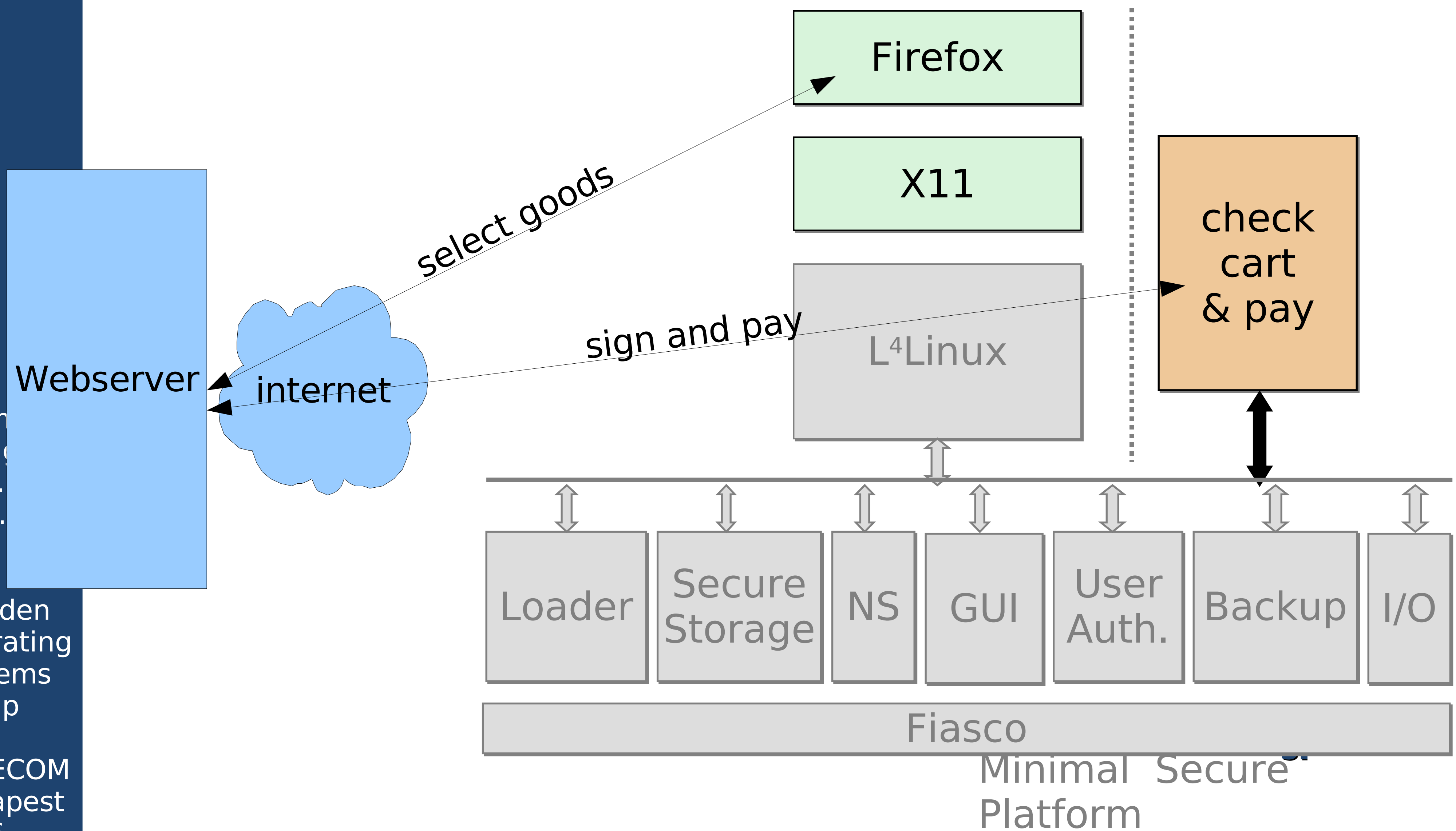
Herm
Härtig
et al.
mult.

TU Dresden Operating Systems Group

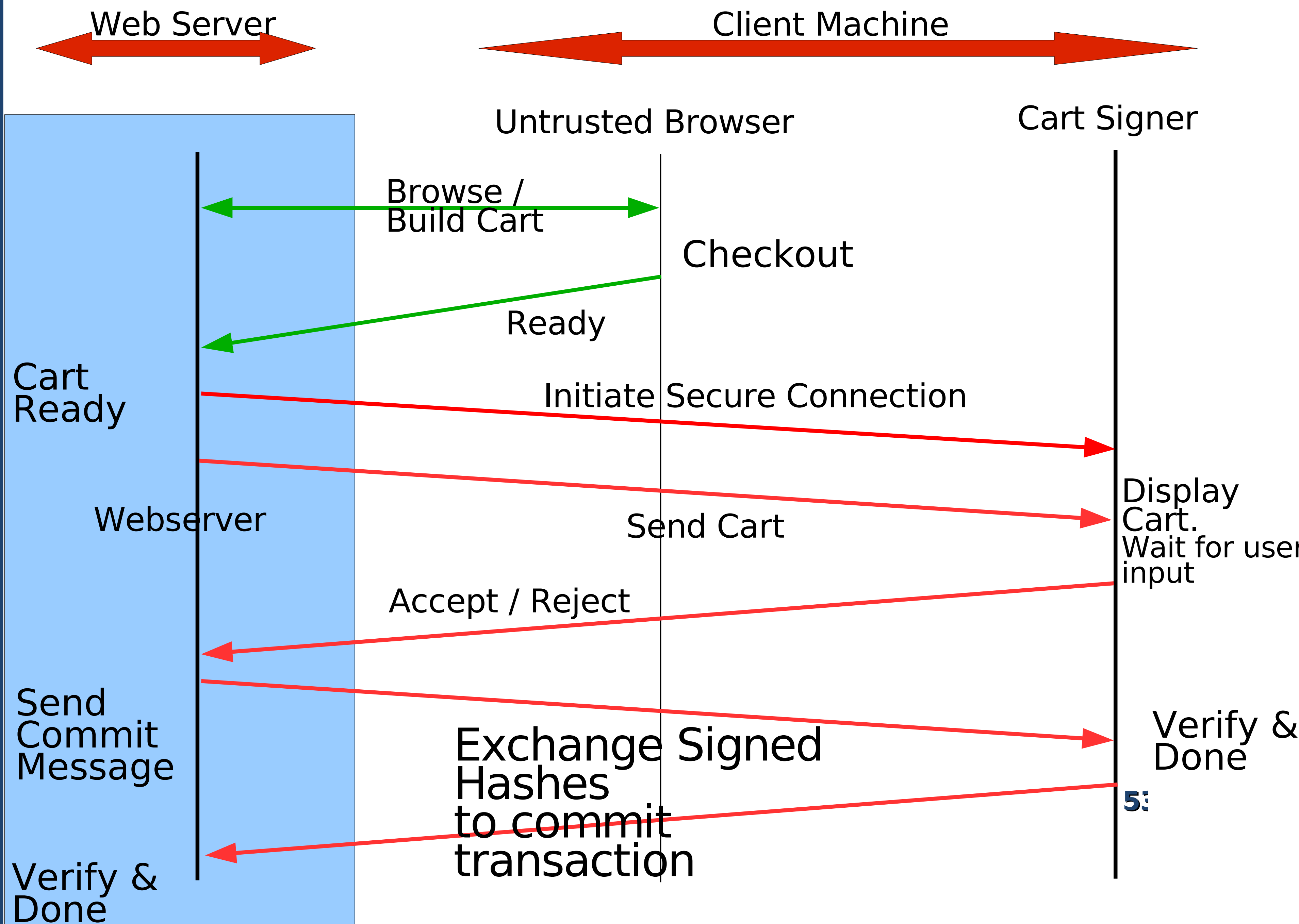
SEVECOM Budapest 2006



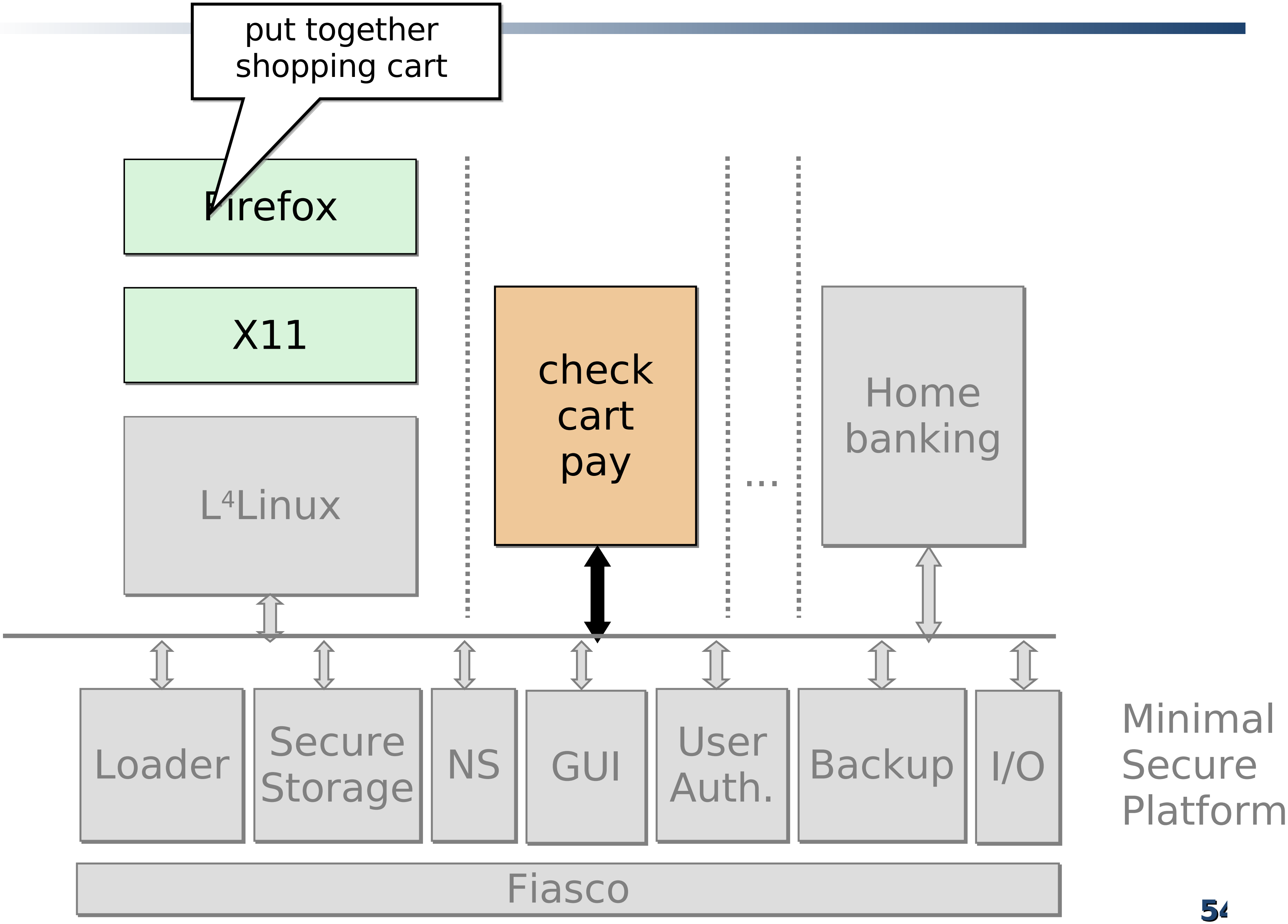
Internet Transaction: Your password(s), credit card number, ...



Split Internet Transaction



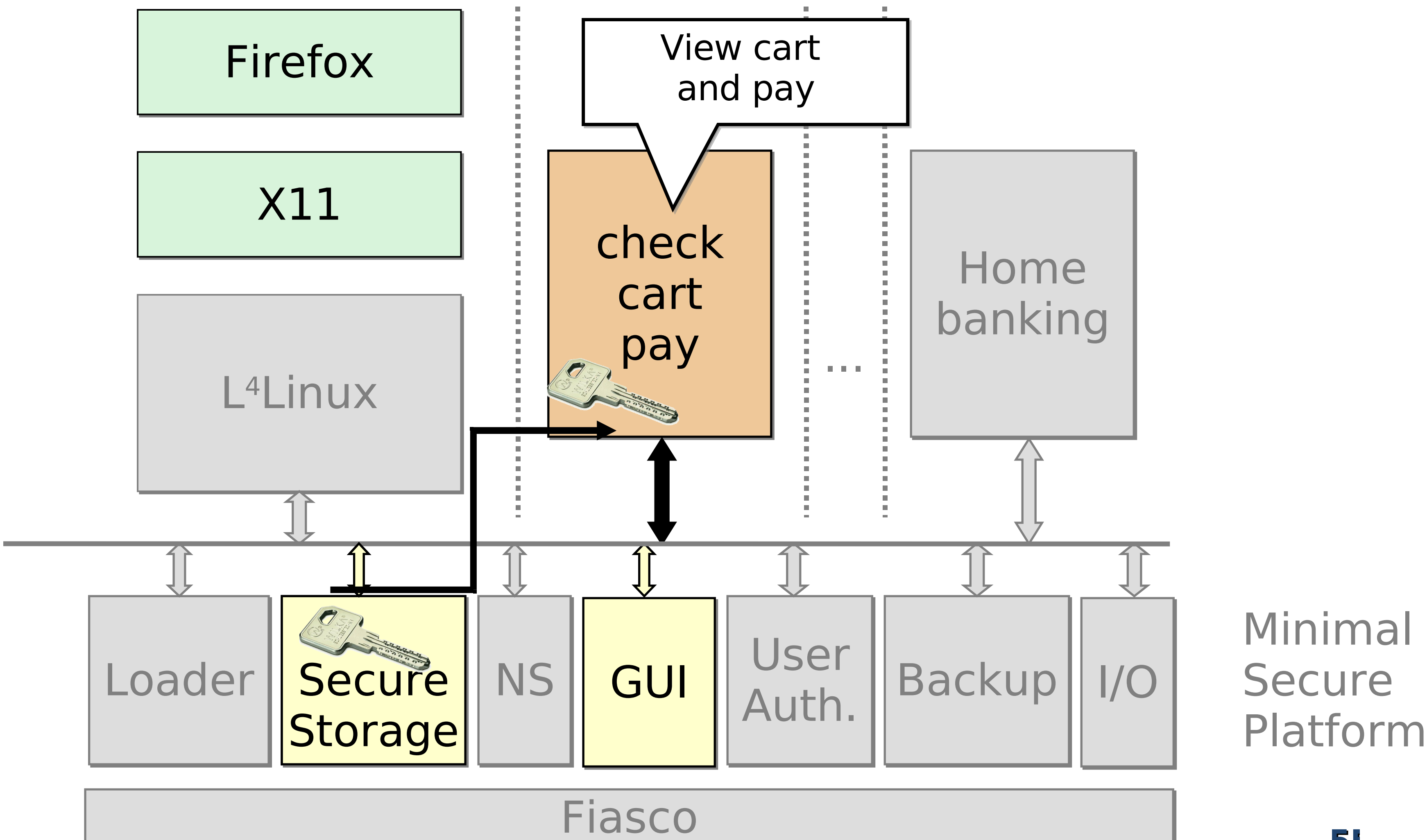
Split Internet Transaction



Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

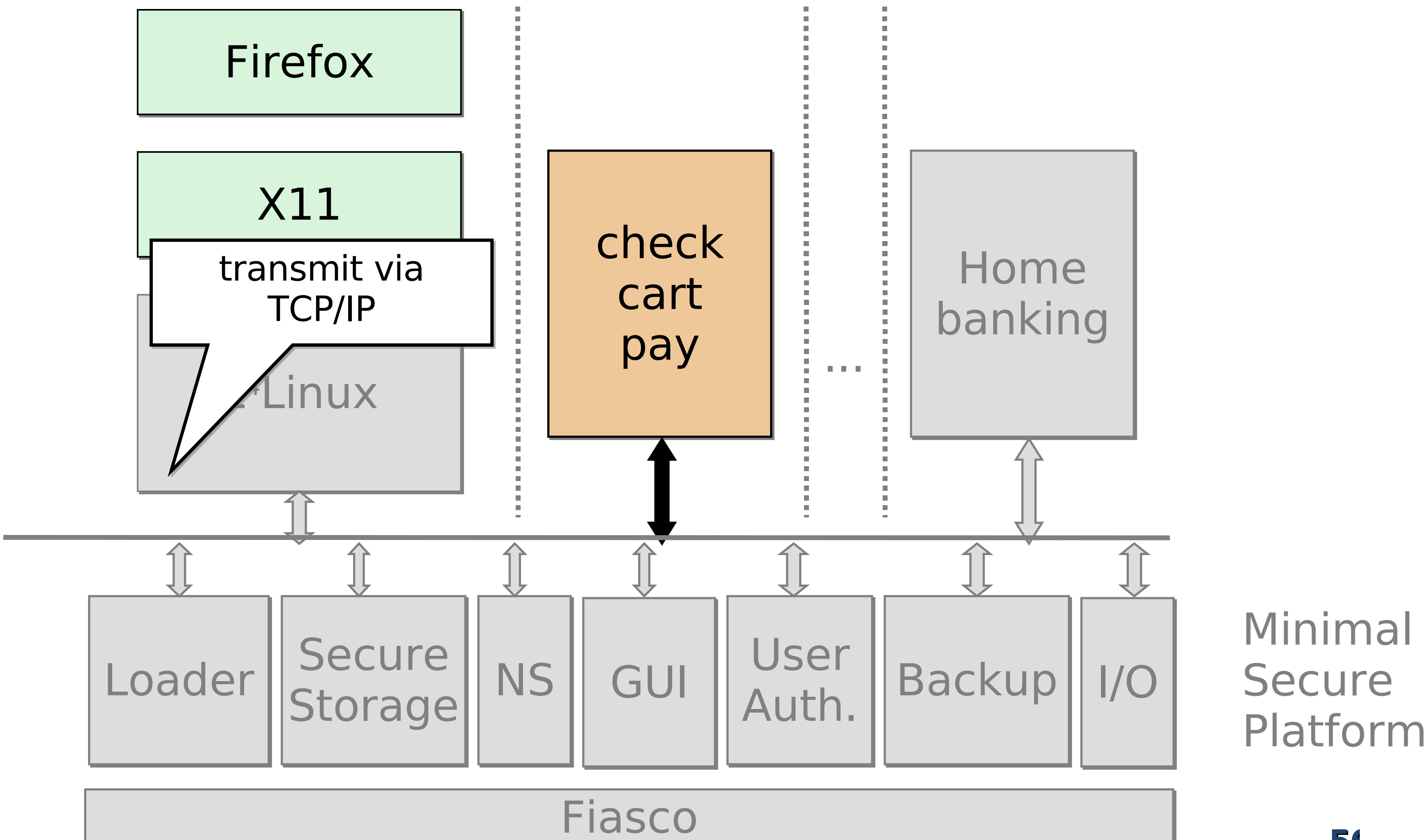
Split Internet Transaction



Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

Split Internet Transaction



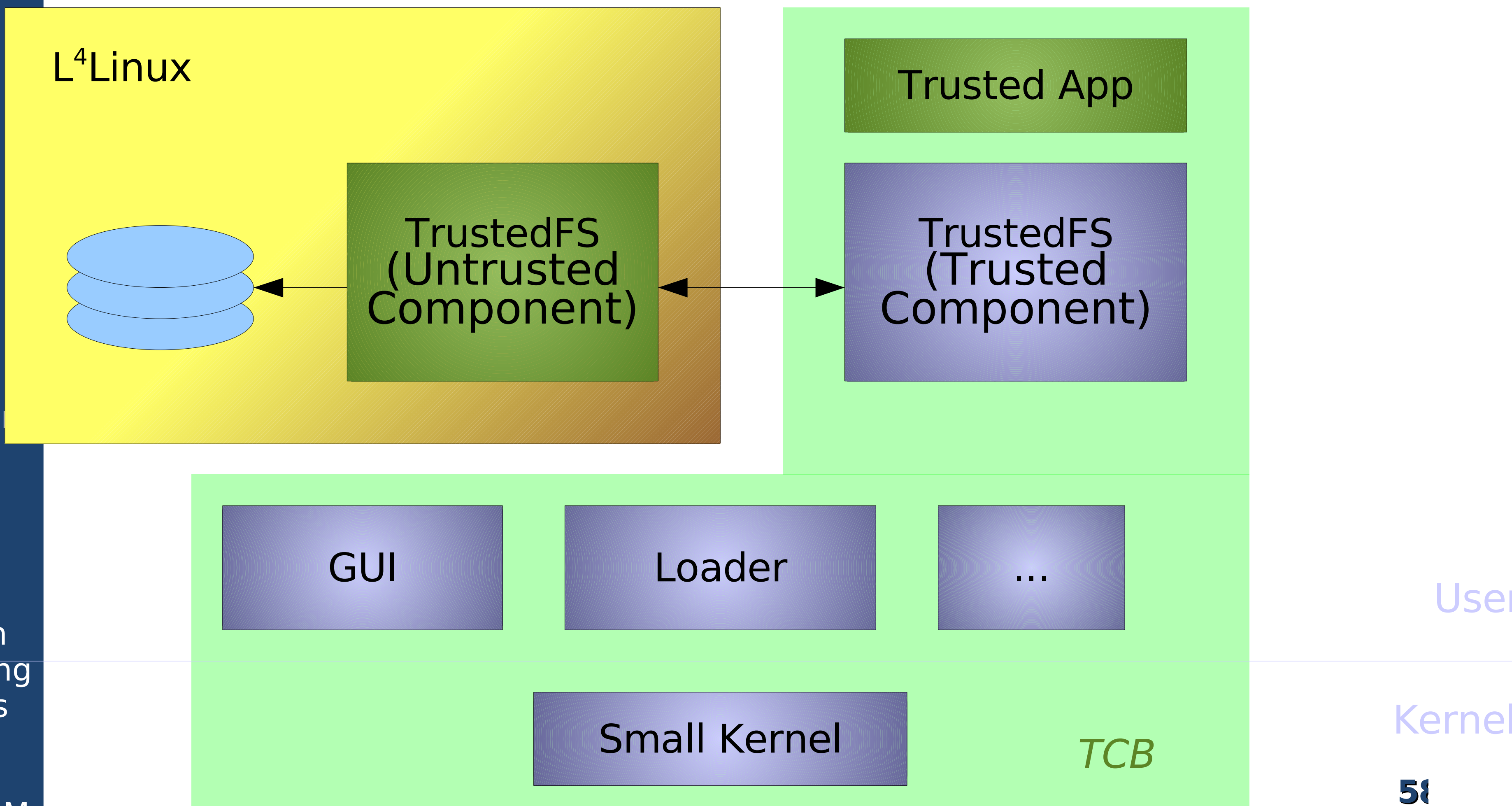
Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

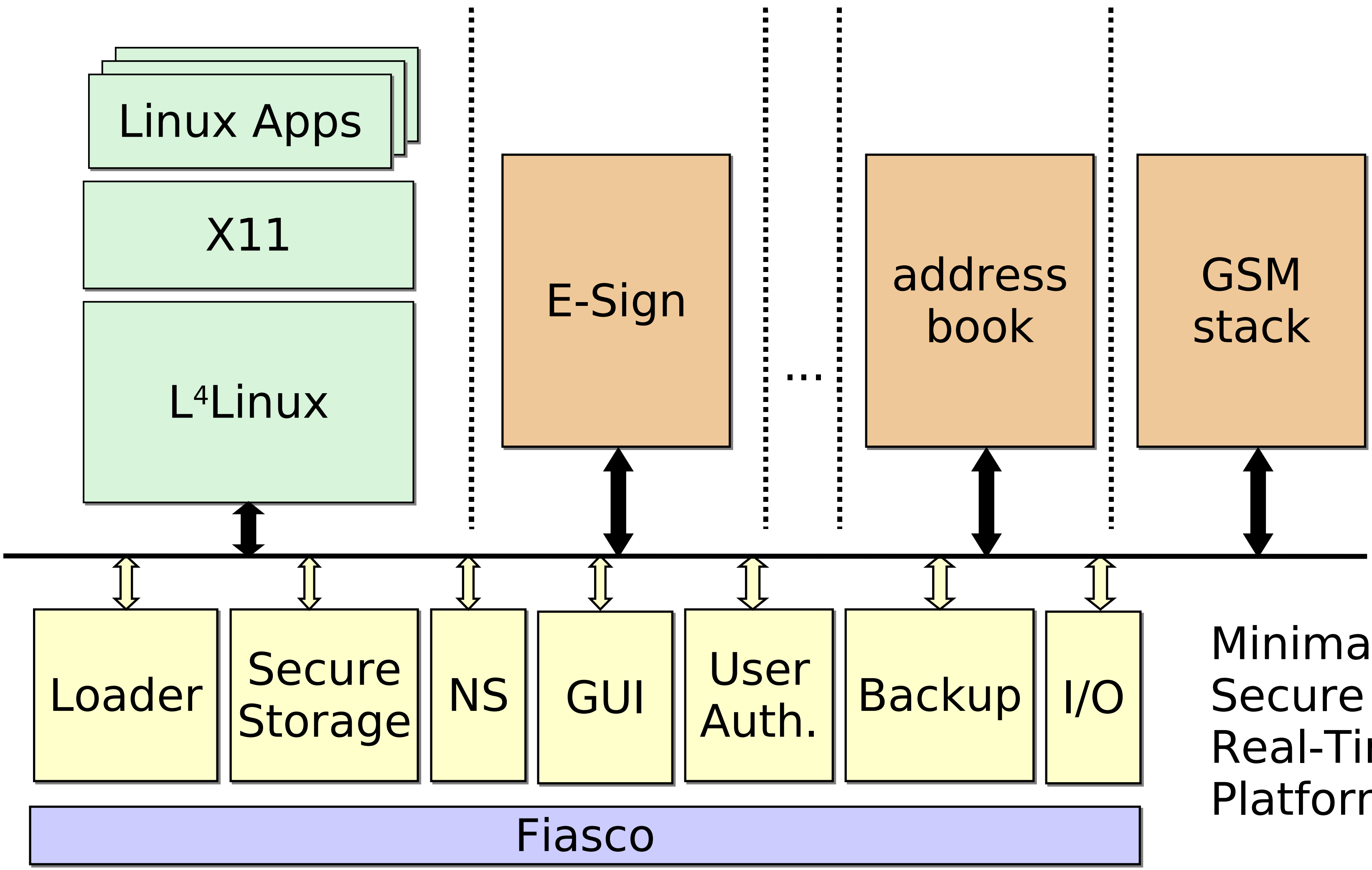
Resulting Complexity

Scenario	Original Application		AppCore		Reduction Factor
	LOC ($x10^3$)	MCC ($x10^3$)	LOC ($x10^3$)	MCC ($x10^3$)	
e-commerce (Browser)	978	151	10	1.5	100X
VPN Gateway (FreeS/WAN)	155	25	74	10	2.1X
Email signer (Thunderbird)	250	45	54	11	4.6X
TCB (Linux+Xserver)	1,485	238	100	14	14X

L4/Nizza Trusted FS



Smart Phone Scenario (RT&Sec)



Minimal
Secure +
Real-Time
Platform

Outline

L4 etc

L4/Nizza Secure System Architecture

What's Up Next?

- applications, applications, applications, ...
- hw developments:
  secure init - IOMMU - VM support
- L4 and virtual machines
- NOVA: local names, ipc control, & VM support
- Bastei: L4Env redone
- “fall back” as simple availability management
- formal specification (and verification attempts)

Conclusion

Applications: AN.ON case study

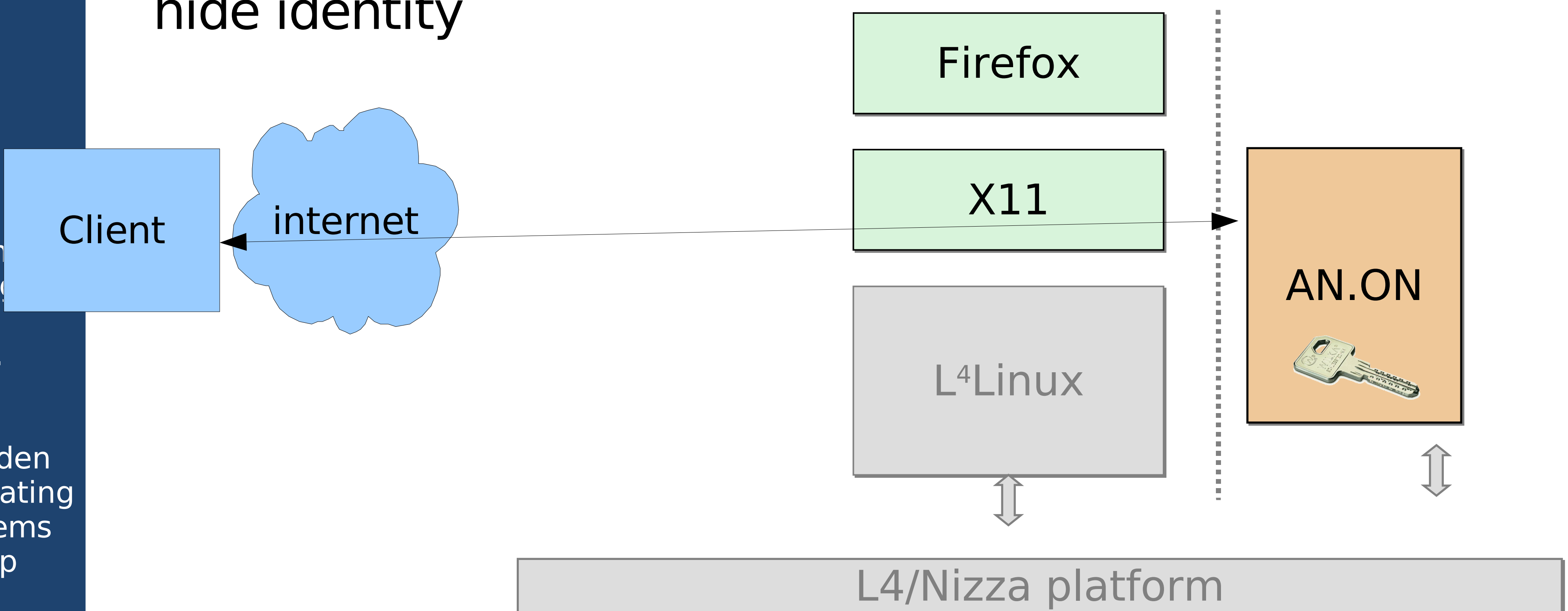
conflicting goals(superficially):

- Trusted Computing:
  establish identity, attestate SW-stack
- AN.ON
  hide identity

Applications: AN.ON case study

conflicting goals(superficially):

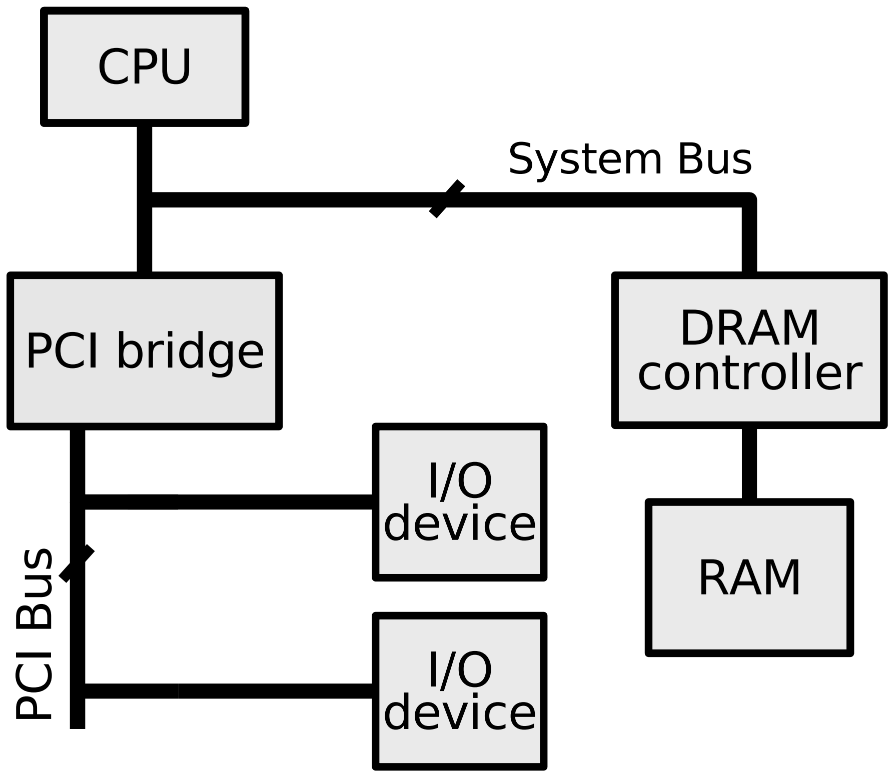
- Trusted Computing:
  establish identity, attestate SW-stack
- AN.ON
  hide identity



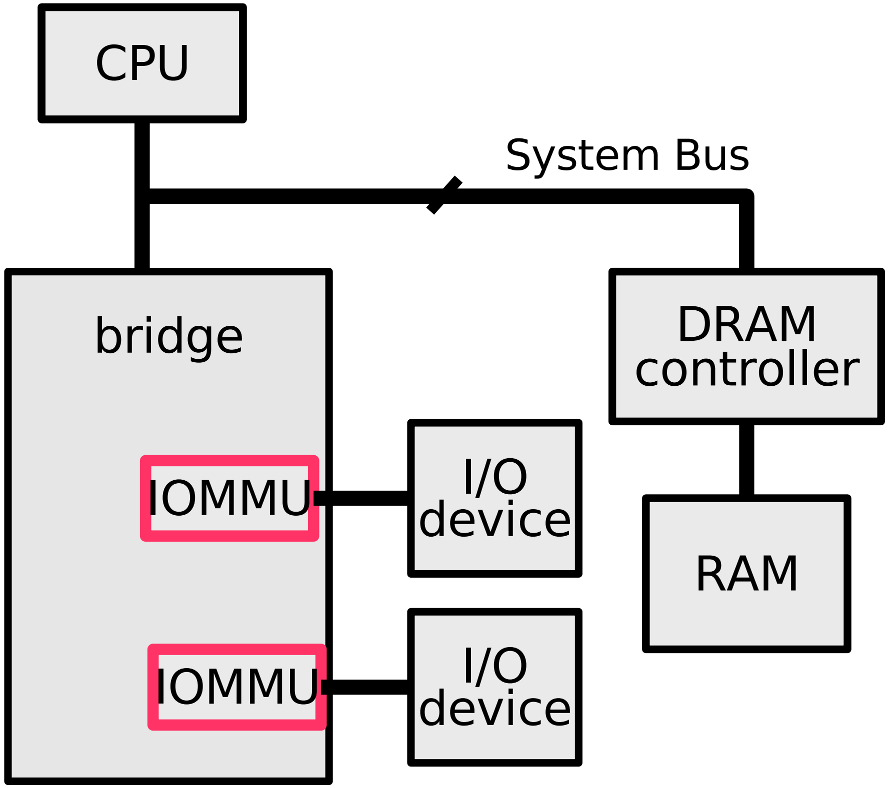
Current HW developements

- secure init (intel LaGrande, AMD Pacifica)
  - takes BIOS, Loader, etc off the TCB
- IO MMUs
  - allows real enforcement of address spaces without driver modification
- VM support
  - removes ambiguity of some X86 instructions
  - much more to support efficient virtualization

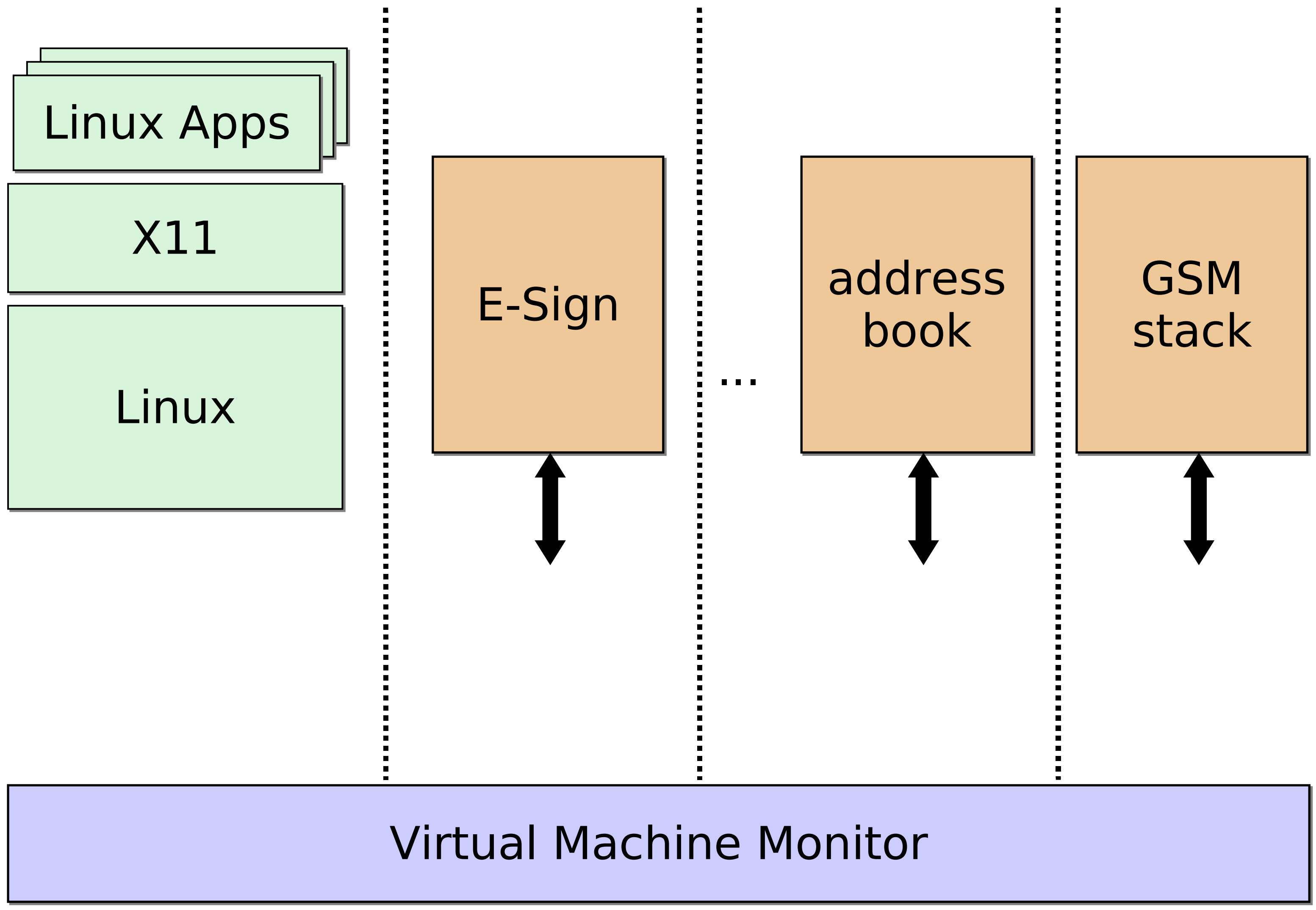
Current: Direct Memory Access



IO MMUs


CPU
System Bus
bridge
IOMMU
IOMMU
I/O
device
I/O
device
DRAM
controller
RAM

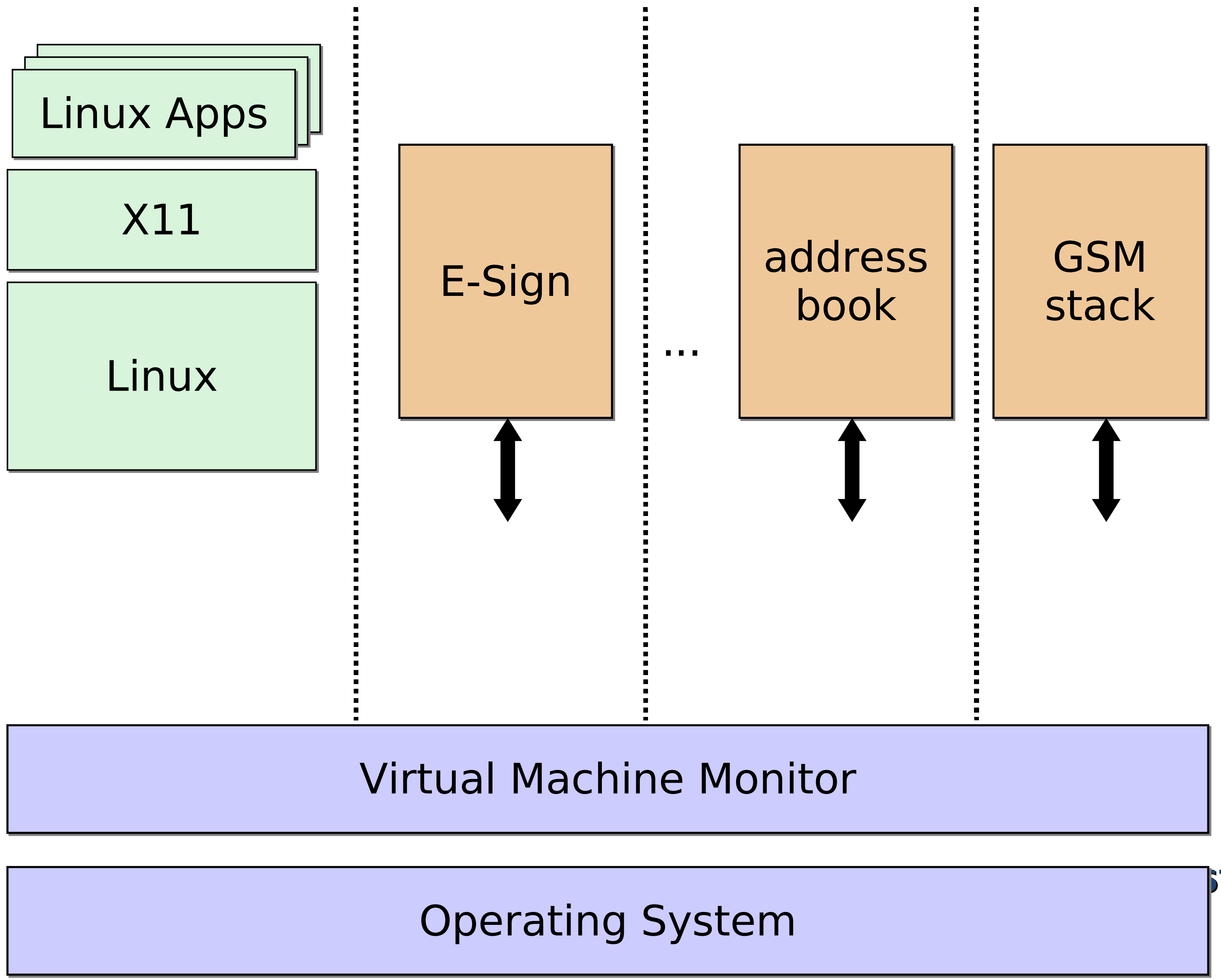
L4 and/vs. Virtual Machines



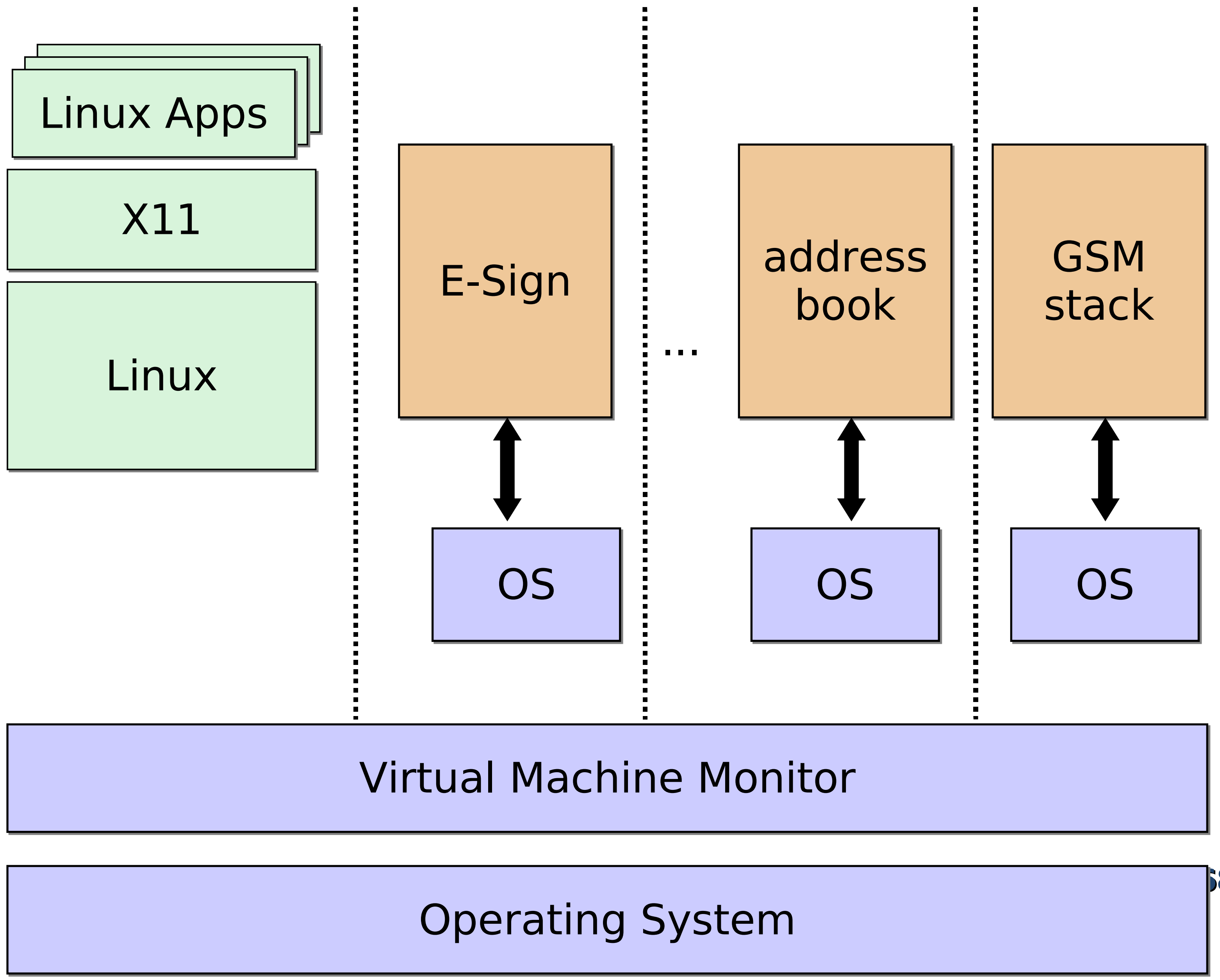
Hermann Härtig et al. mult.

TU Dresden Operating Systems Group

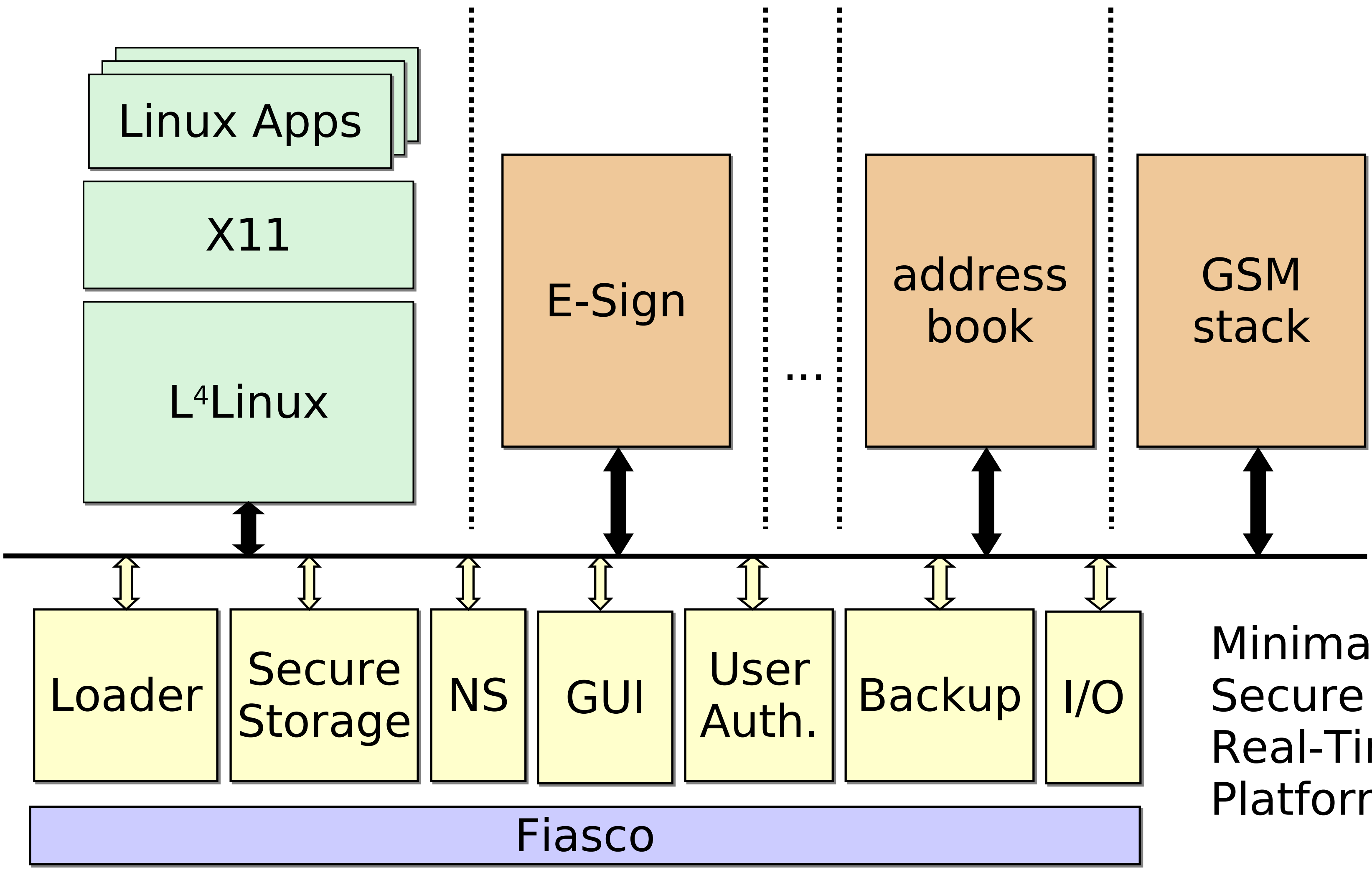
L4 and/vs. Virtual Machines



L4 and/vs. Virtual Machines



L4 and/vs. Virtual Machines

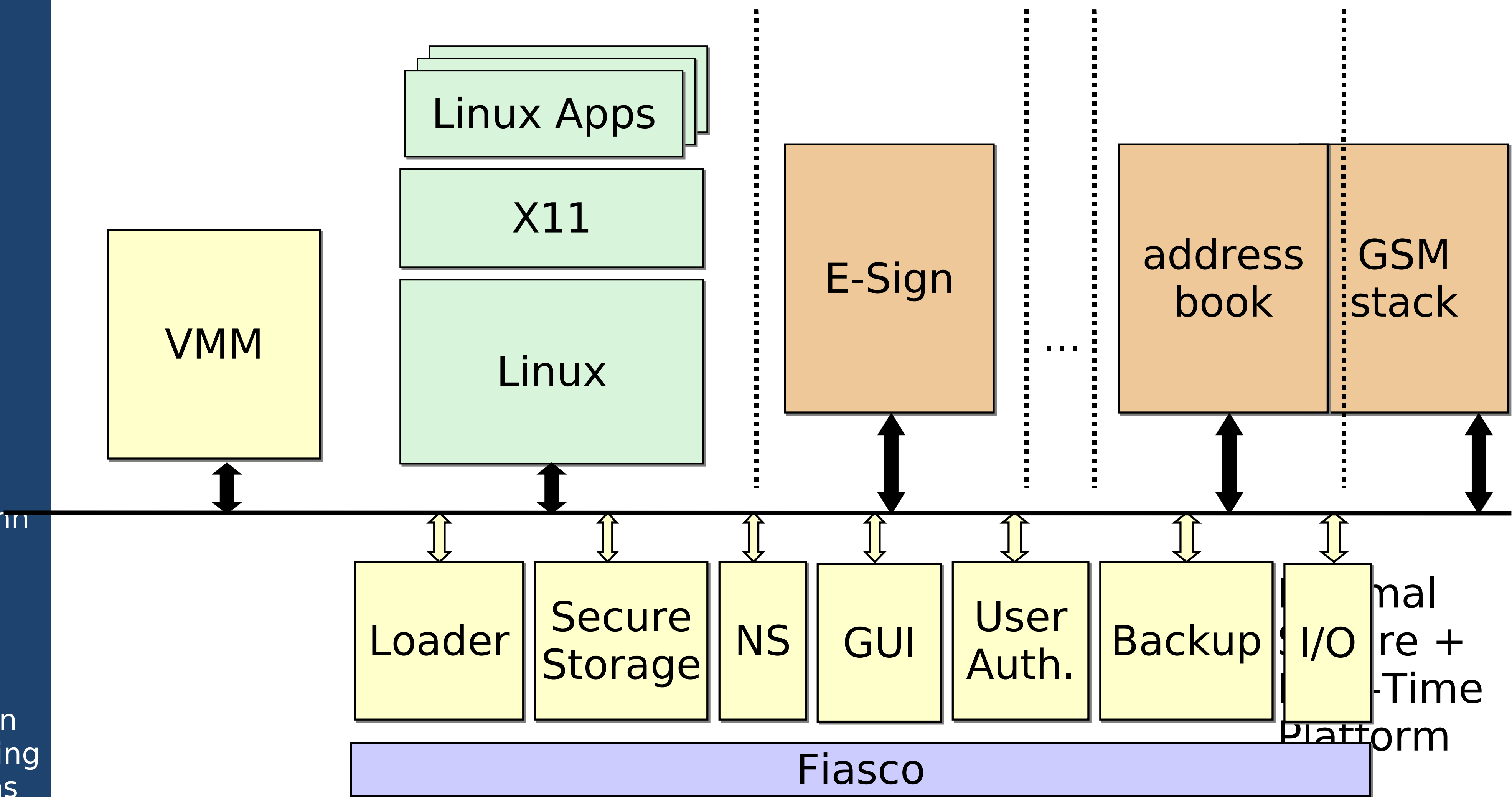


Minimal
Secure +
Real-Time
Platform

Hermann
Härtig
et al.
mult.

TU
Dresden
Operating
Systems
Group

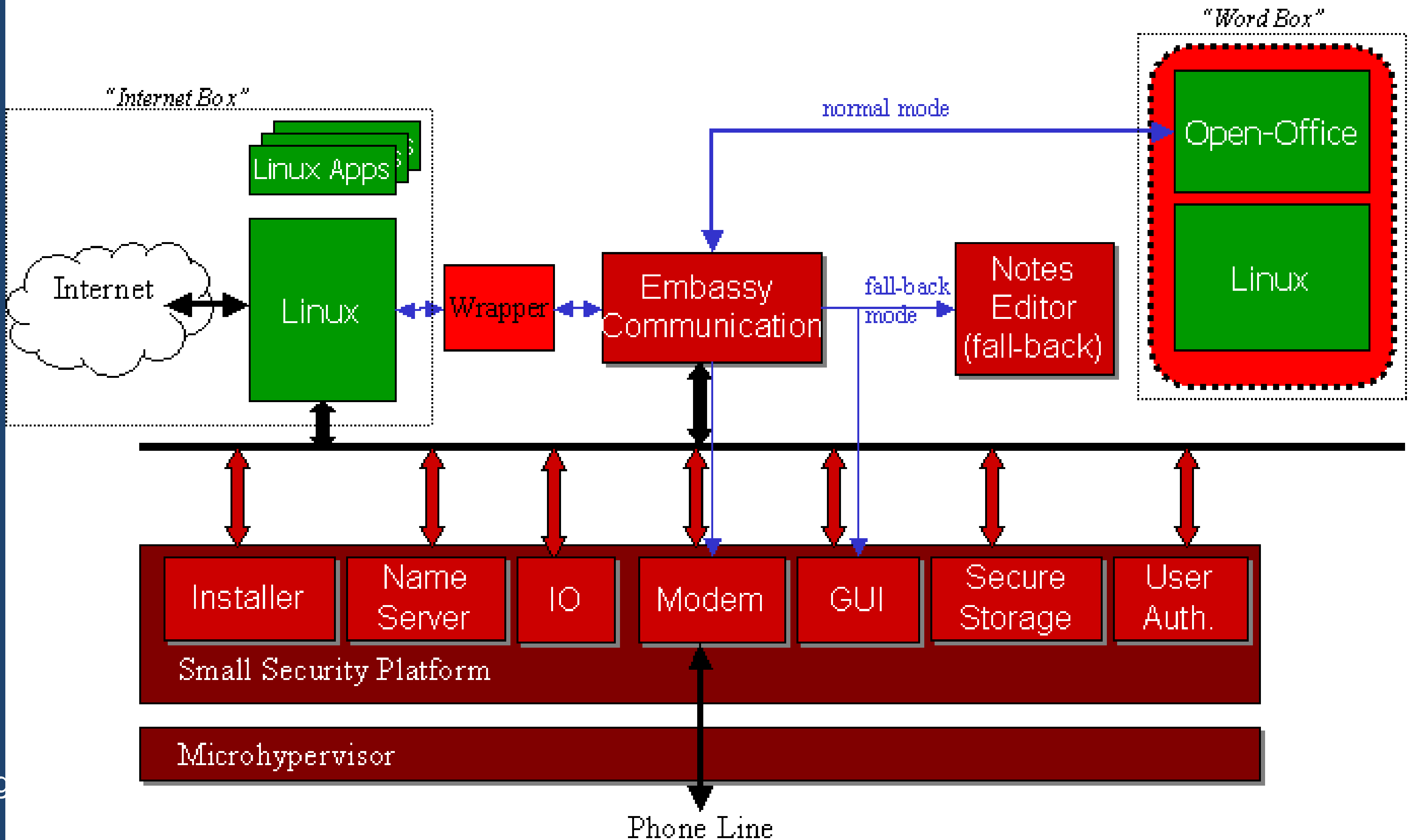
L4 and/vs. Virtual Machines



Bastei: L4Env redone

- ongoing activity, no reliable results yet
- reorganize L4Env

Fall Back as simple Availability



References, see http://tudos.org/drops/doc.html

- Reducing TCB Complexity for Security-Sensitive Applications: Three Case Studies
- The Nizza Secure-System Architecture.
- A Nitpicker's guide to a minimal-complexity secure GUI.
- Reducing TCB size by using untrusted components — small kernels versus virtual-machine monitors
- Security Architectures Revisited
- OS-Controlled Cache Predictability for Real-Time Systems
- Cost and benefit of separate address spaces in Real-Time operating systems
- The Performance of μ-Kernel-based Systems
- ...

(Some of) Our(TUDOS) Sponsors and Partners

- State of Saxony
- Deutsche Forschungsgemeinschaft (DFG)
- German Ministry for Commerce and Technology (EMSCB)
- European Commission (OpenTC, **Robin**)
- German Information Security Agency

- IBM
- infineon
- intel
- Nokia (just announced)
- secunet
- ST Microelectronics

- NICTA, University of Karlsruhe, Georgia Tech, ...

Conclusions

caveat:

- Research Experiments
- but: L4/Fiasco, L4Env ... pretty mature

L4-based systems:

- enable safe reuse of legacy SW at moderate extra cost (virtualization)
- small dedicated systems for real-time and/or security
- significant academic community pushing technology forward
- open source (GPL v2)