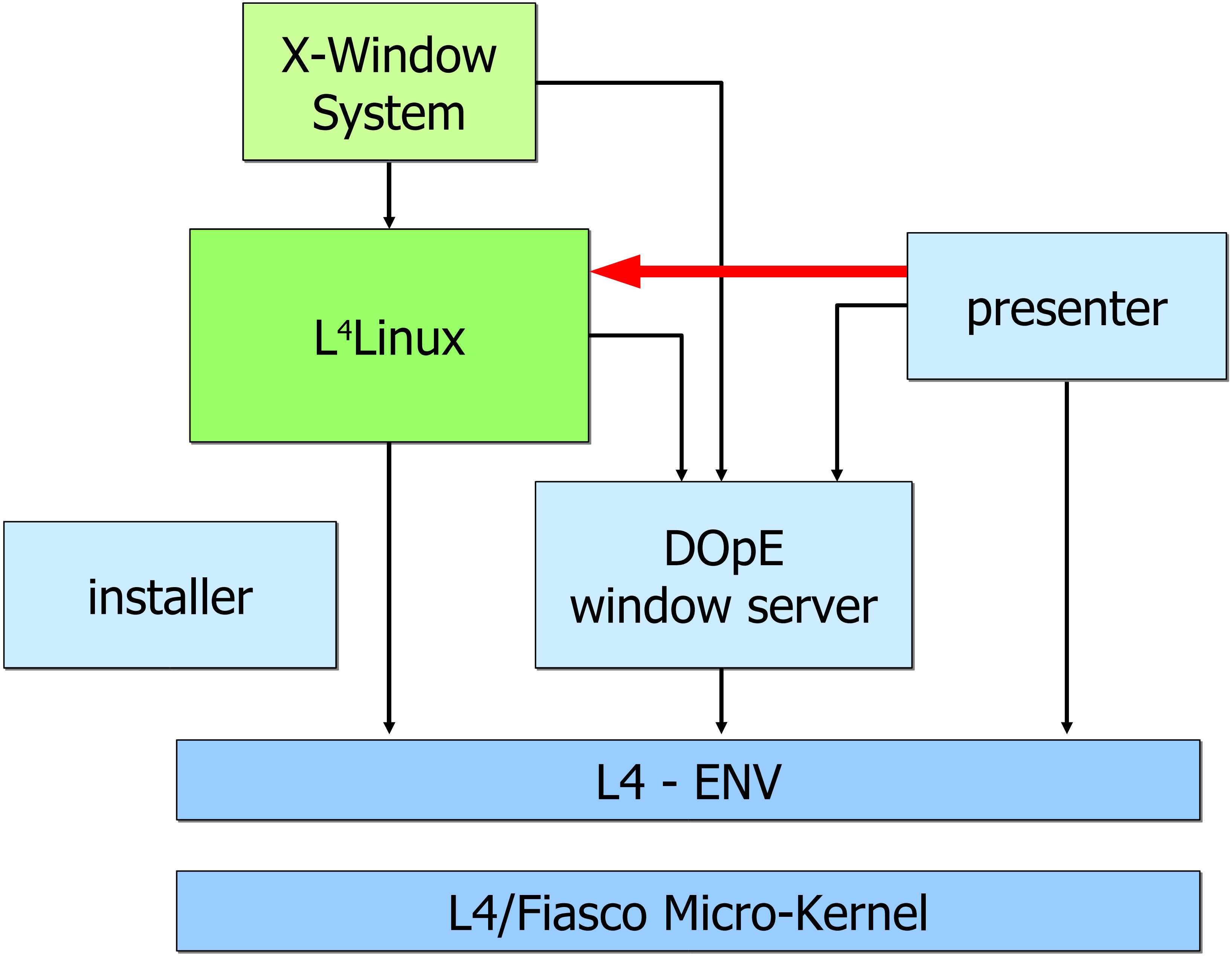
NIZZA

Hermann Härtig
TU Dresden ♦OS
August 2004


X-Window System

L^4Linux

presenter

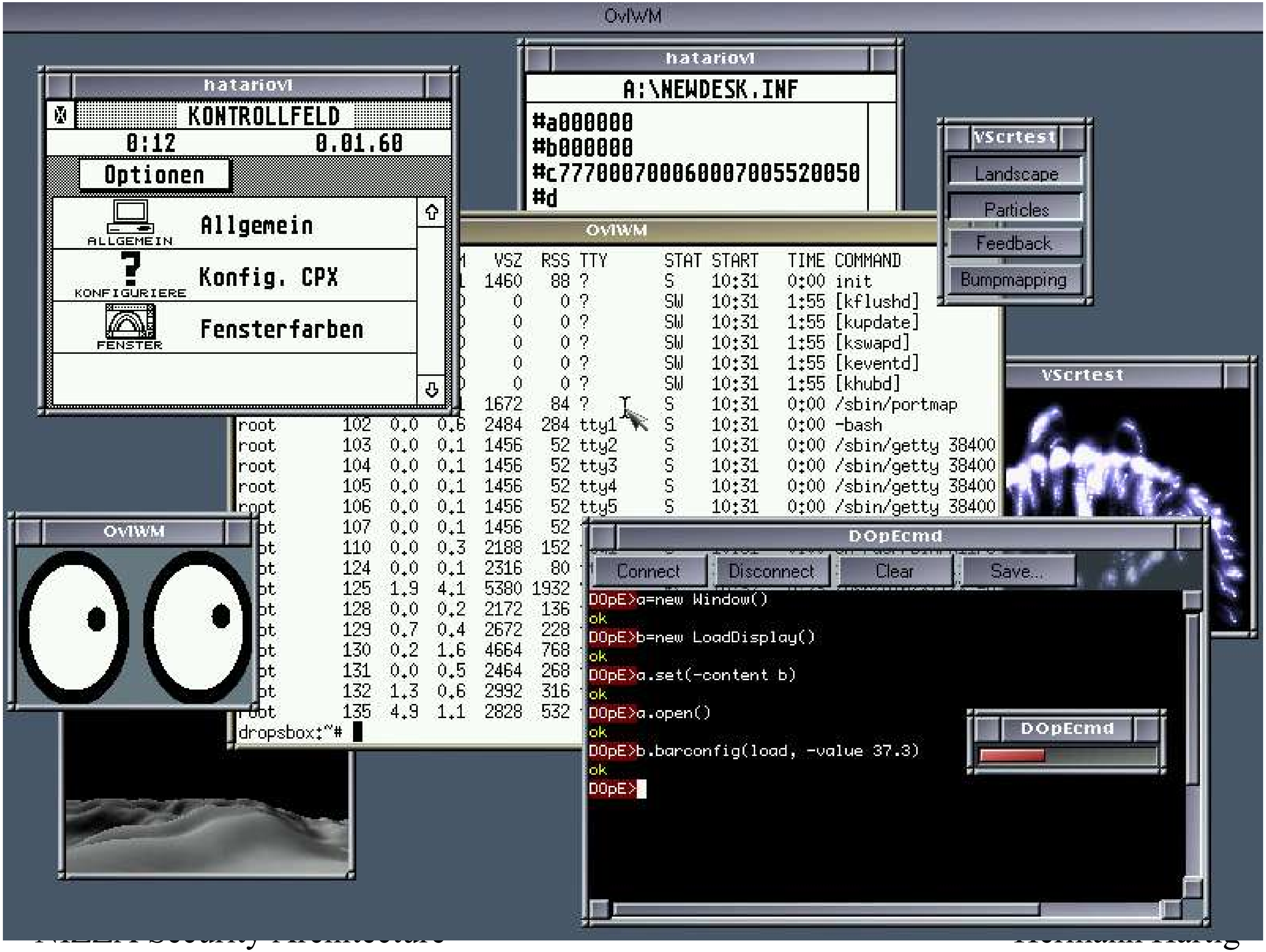
installer

DOpE window server

L4 - ENV

L4/Fiasco Micro-Kernel

Screen Shot

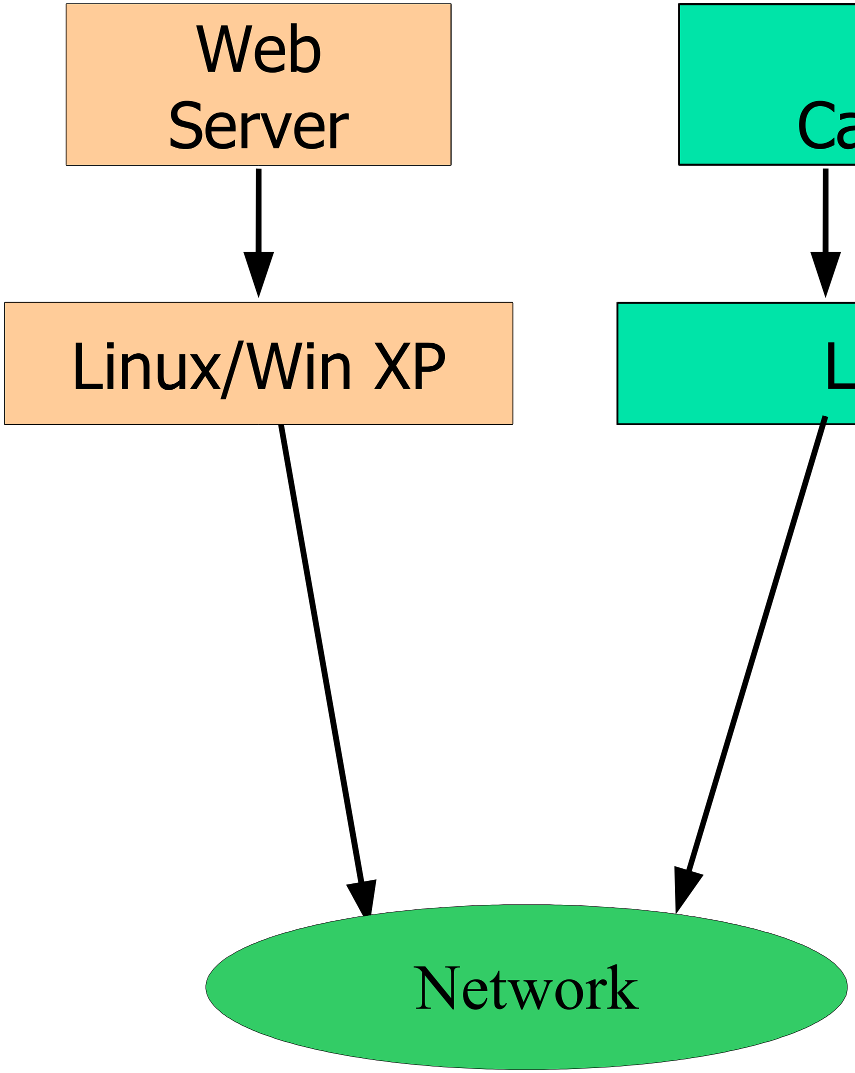


```
#a000000
#b000000
#c7770007000600070055200050
#d
```

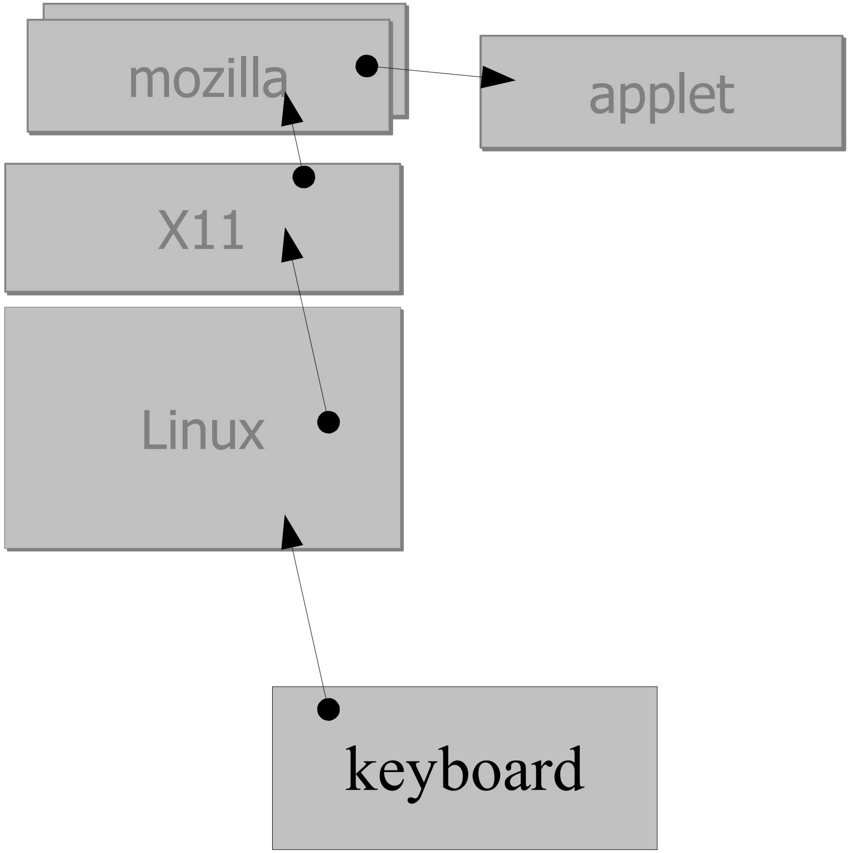
```
M     VSZ  RSS TTY      STAT START   TIME COMMAND
1    1460   88 ?        S    10:31   0:00 init
0       0    0 ?        SW   10:31   1:55 [kflushd]
0       0    0 ?        SW   10:31   1:55 [kupdate]
0       0    0 ?        SW   10:31   1:55 [kswapd]
0       0    0 ?        SW   10:31   1:55 [keventd]
0       0    0 ?        SW   10:31   1:55 [khubd]
1    1672   84 ?        S    10:31   0:00 /sbin/portmap
root       102  0.0  0.6  2484  284 tty1     S    10:31   0:00 -bash
root       103  0.0  0.1  1456   52 tty2     S    10:31   0:00 /sbin/getty 38400
root       104  0.0  0.1  1456   52 tty3     S    10:31   0:00 /sbin/getty 38400
root       105  0.0  0.1  1456   52 tty4     S    10:31   0:00 /sbin/getty 38400
root       106  0.0  0.1  1456   52 tty5     S    10:31   0:00 /sbin/getty 38400
ot         107  0.0  0.1  1456   52
ot         110  0.0  0.3  2188  152
ot         124  0.0  0.1  2316   80
ot         125  1.9  4.1  5380 1932
ot         128  0.0  0.2  2172  136
ot         129  0.7  0.4  2672  228
ot         130  0.2  1.6  4664  768
ot         131  0.0  0.5  2464  268
ot         132  1.3  0.6  2992  316
oot        135  4.9  1.1  2828  532
dropsbox:~#
```

```
DOpE>a=new Window()
ok
DOpE>b=new LoadDisplay()
ok
DOpE>a.set(-content b)
ok
DOpE>a.open()
ok
DOpE>b.barconfig(load, -value 37.3)
ok
DOpE>
```

Internet Transaction



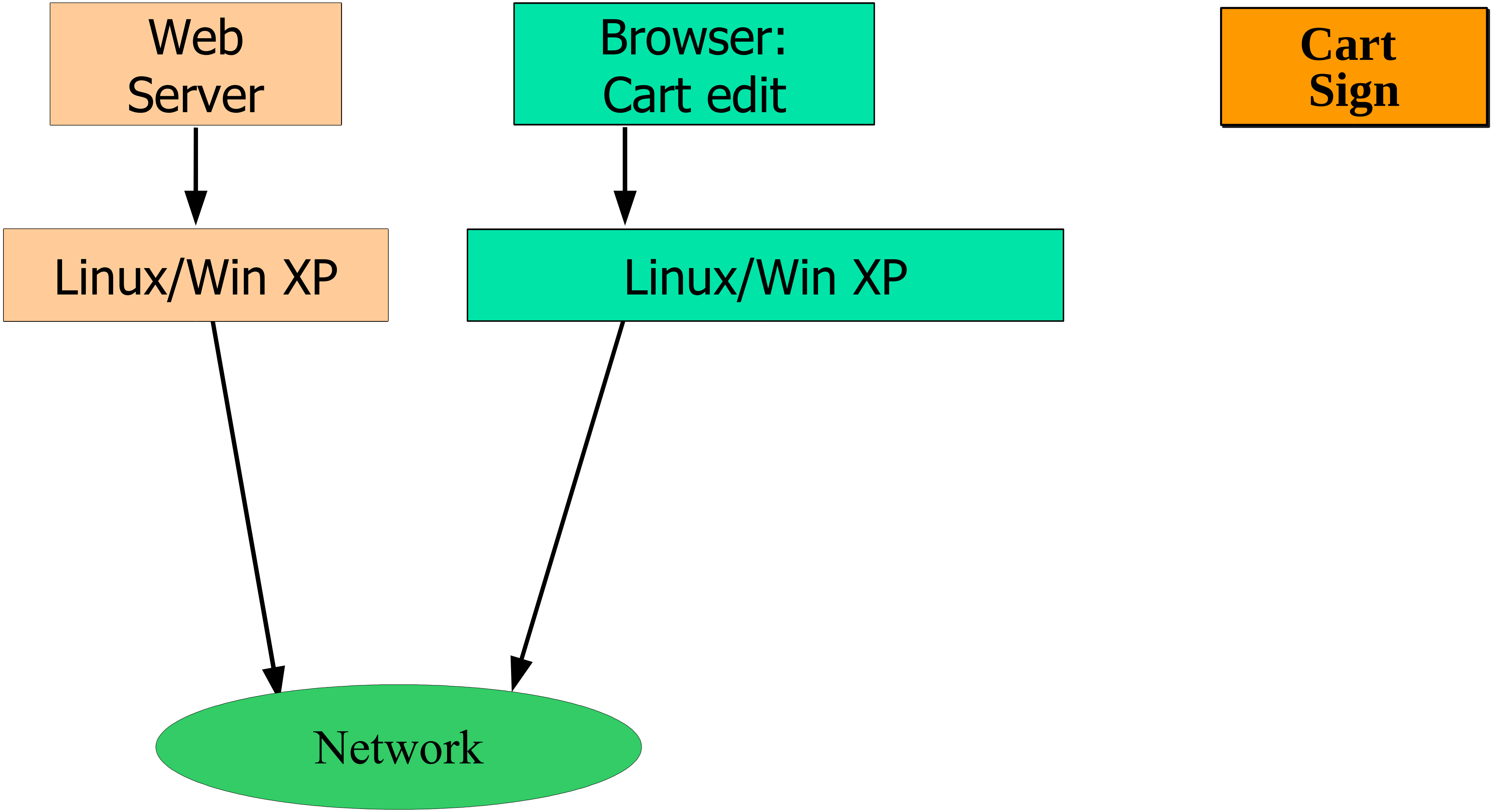
Your password(s), credit card number, ...



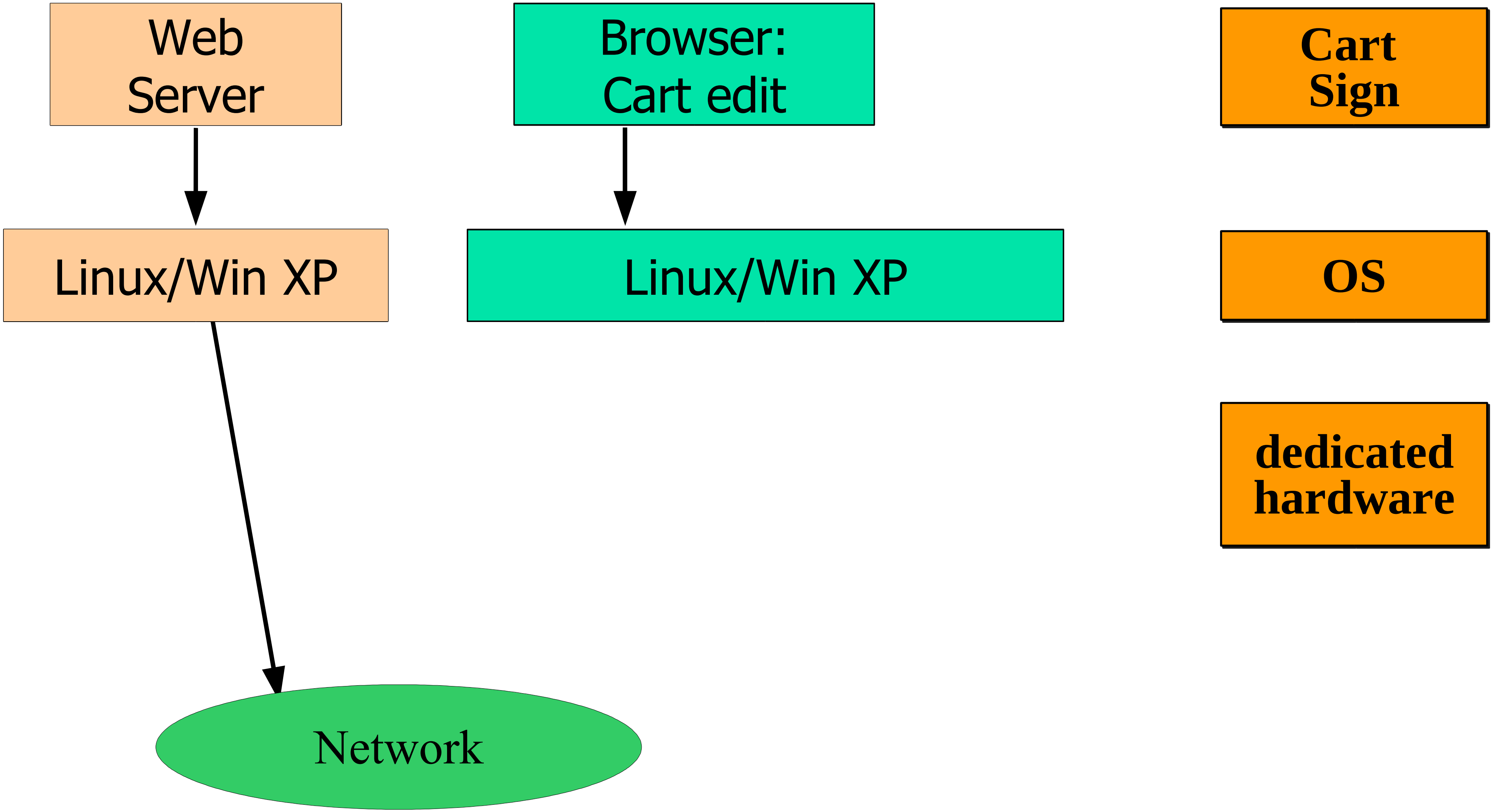
see:

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

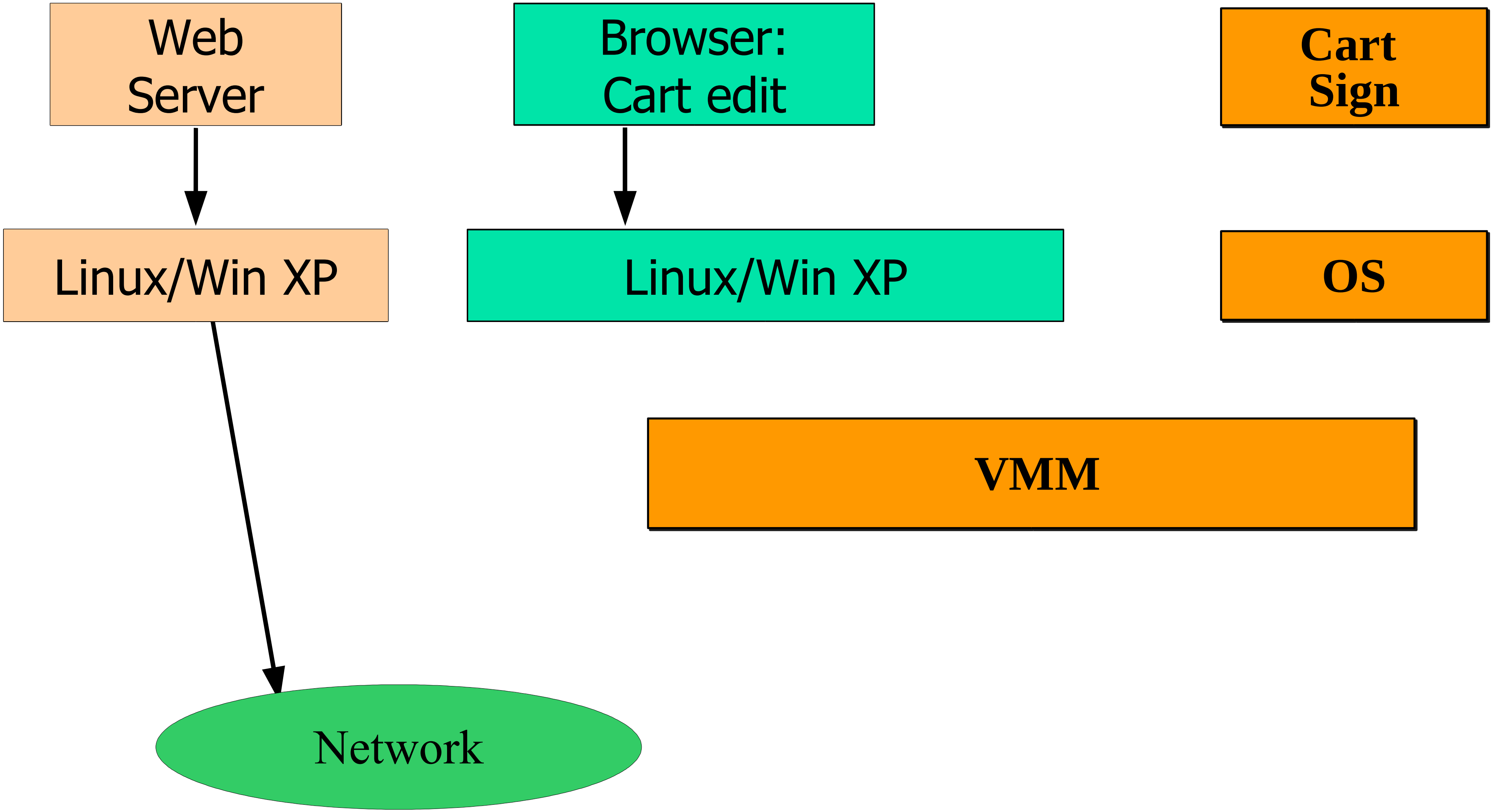
Lenin's Transaction



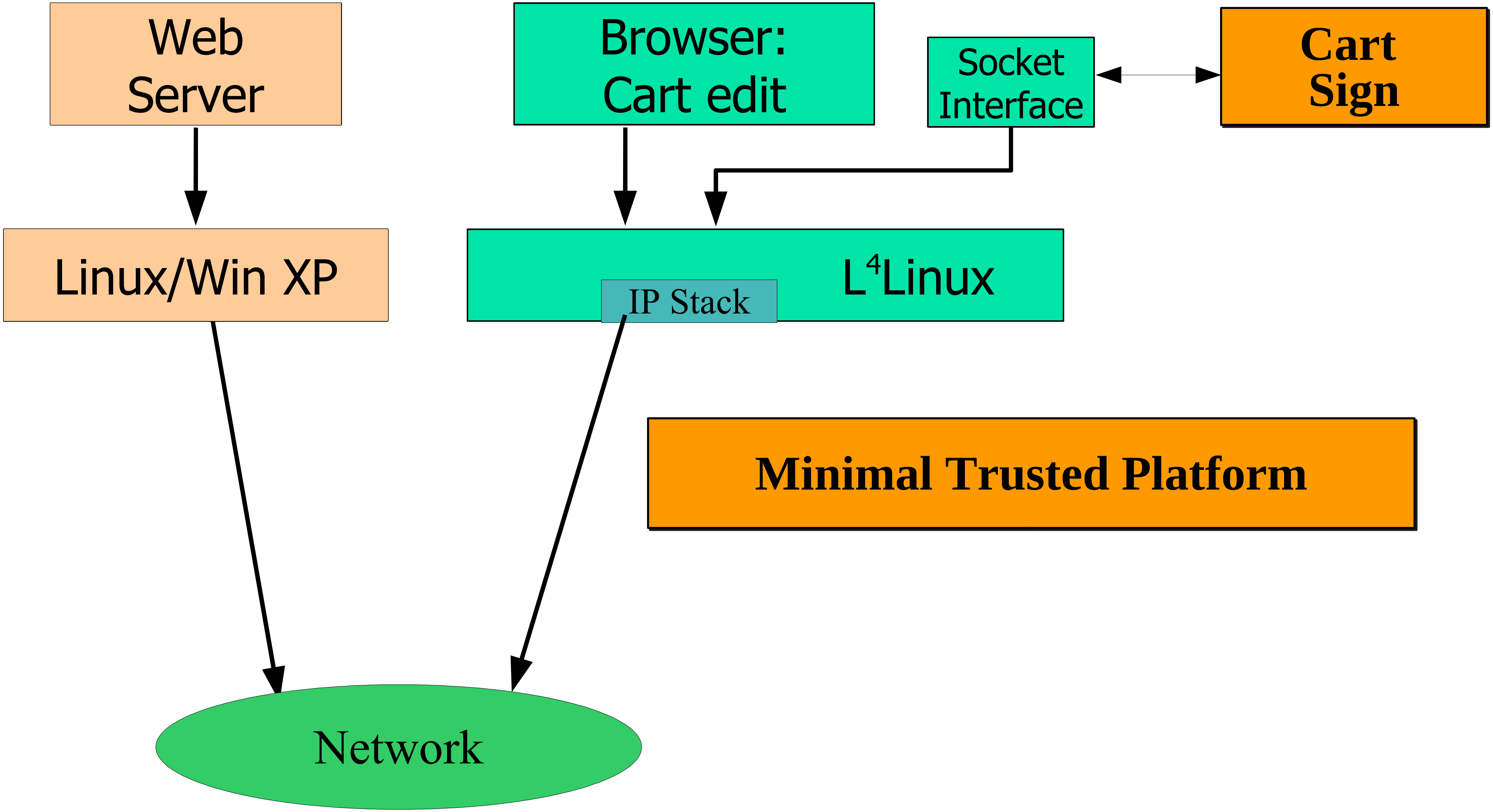
Lenin's Transaction



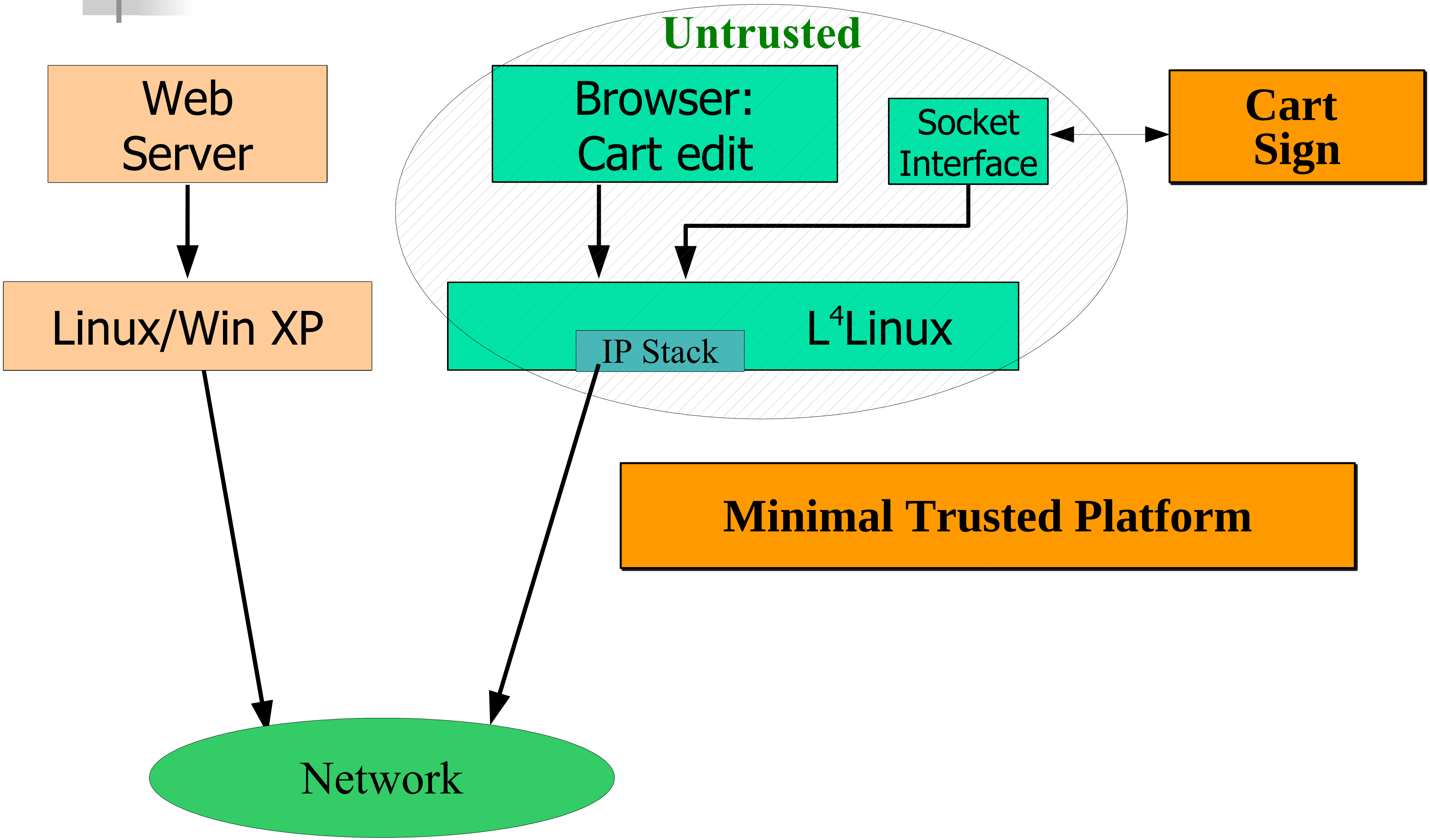
Lenin's Transaction



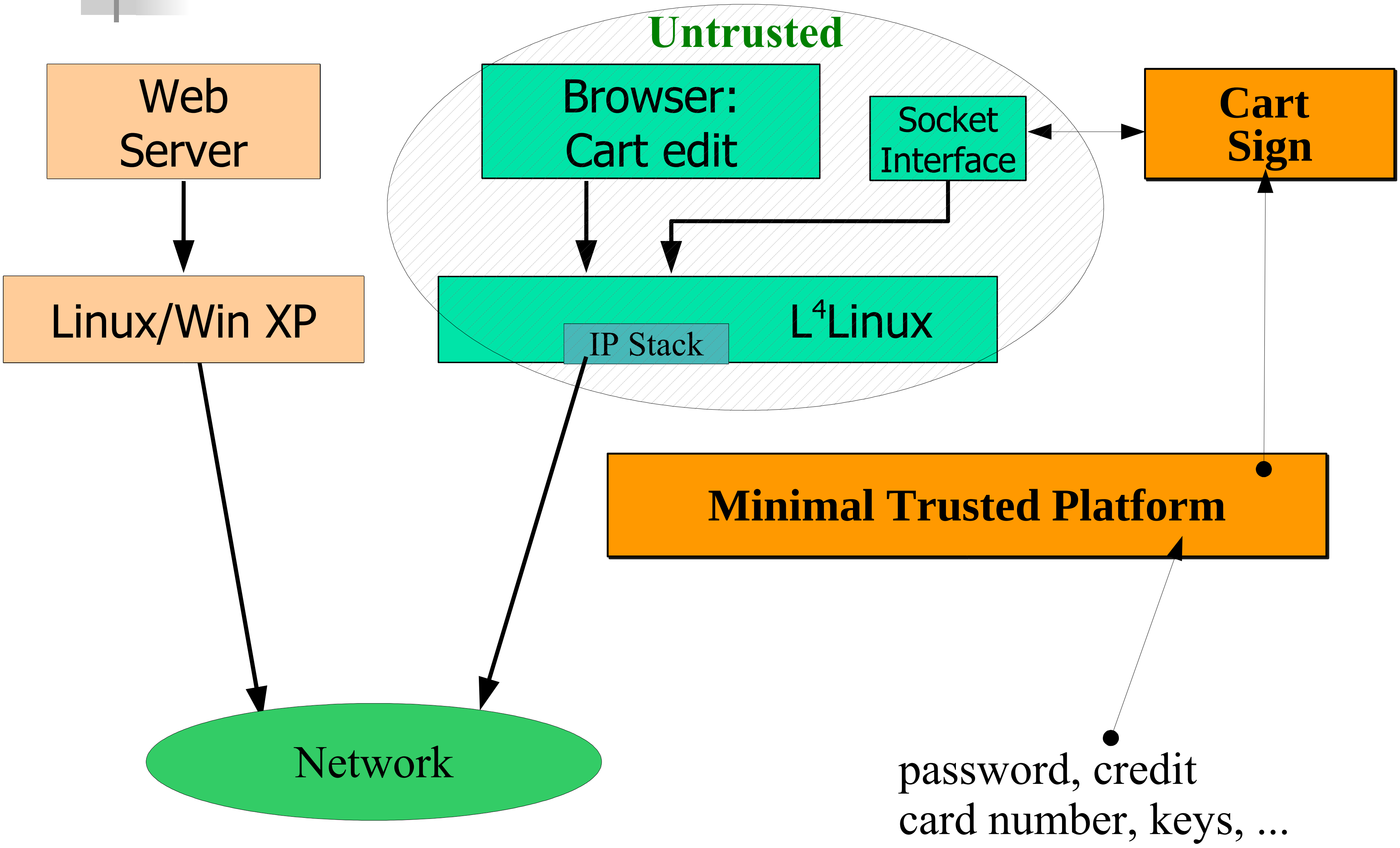
Lenin's Transaction



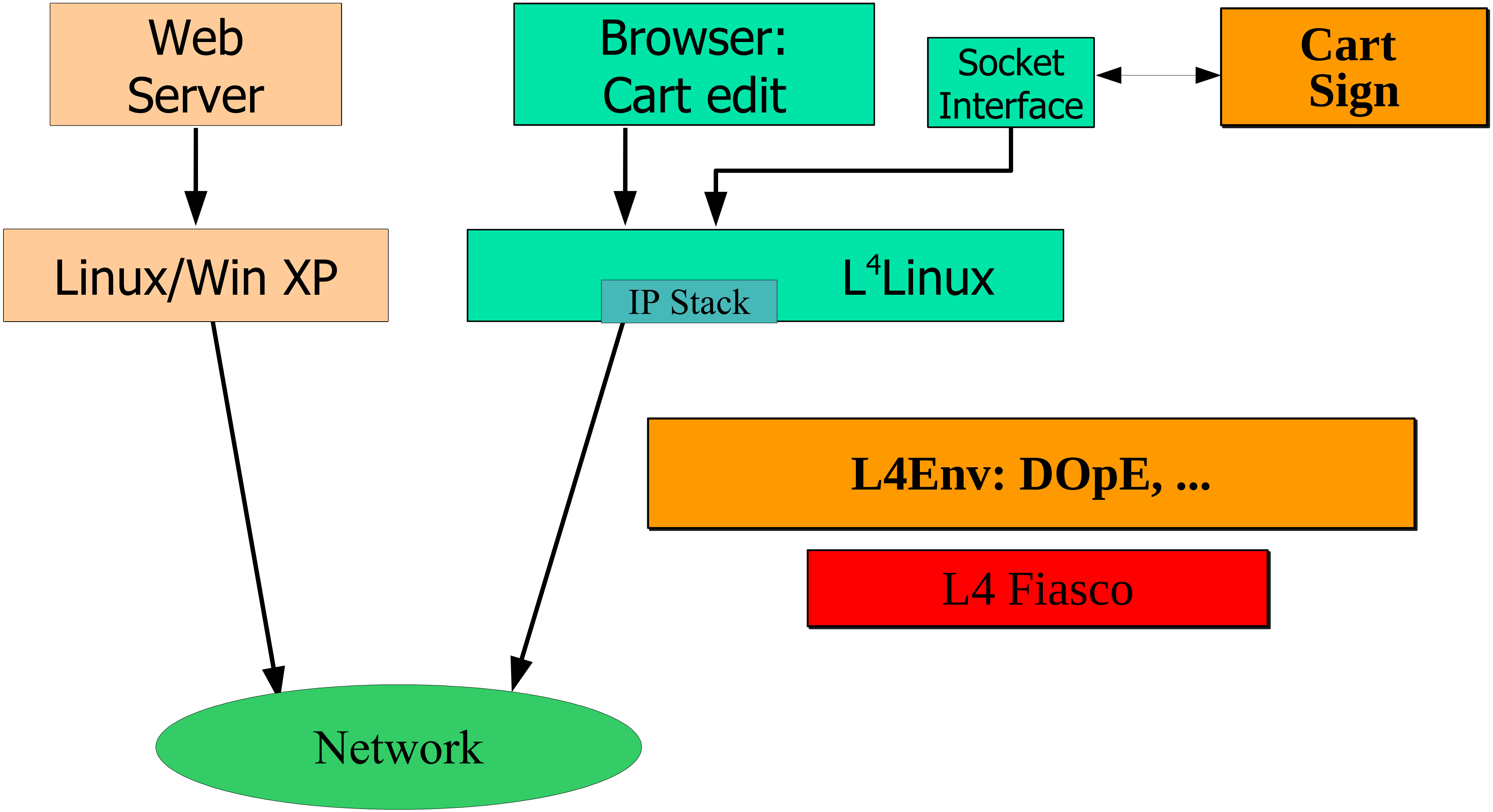
Lenin's Transaction



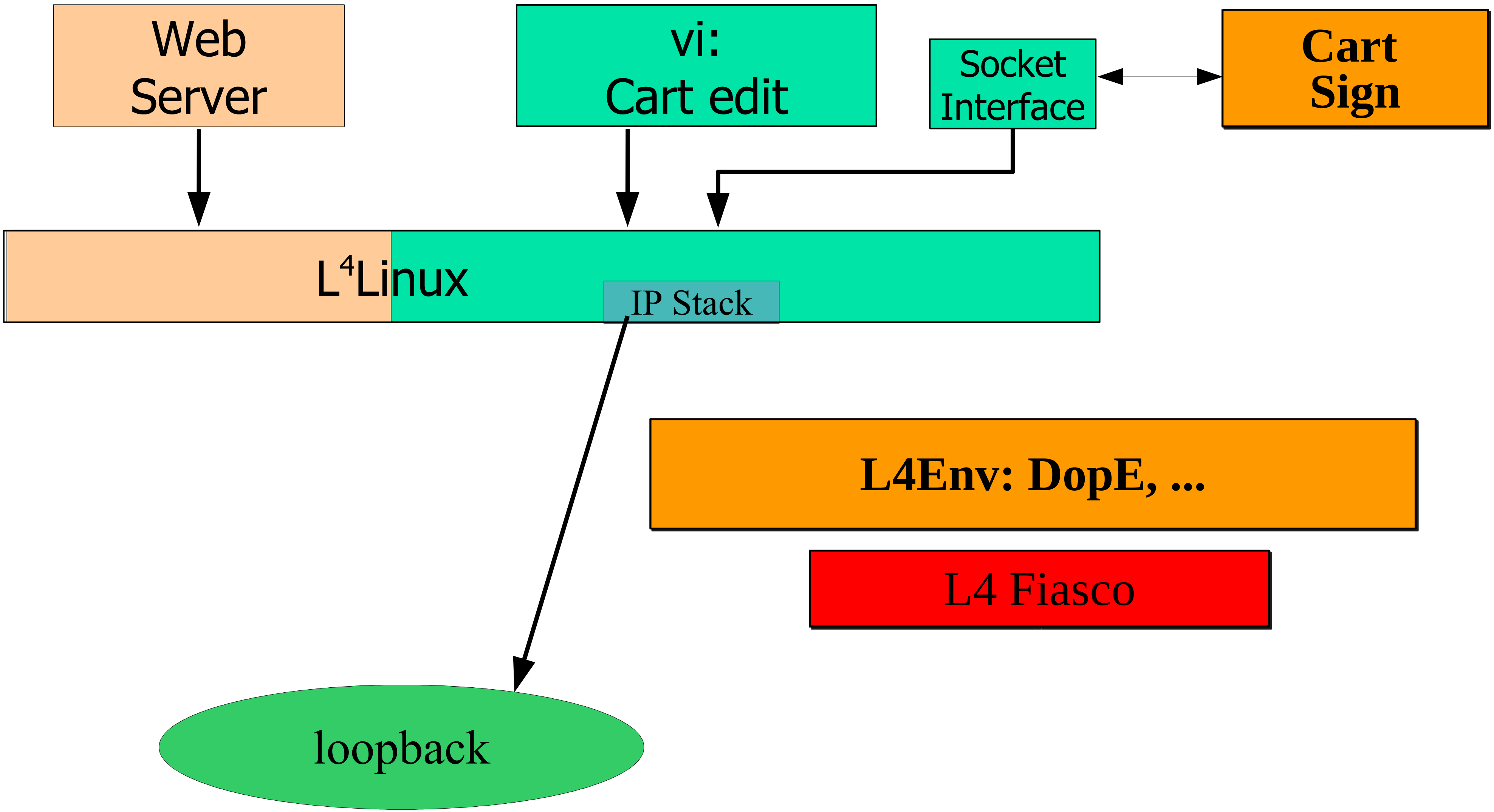
Lenin's Transaction



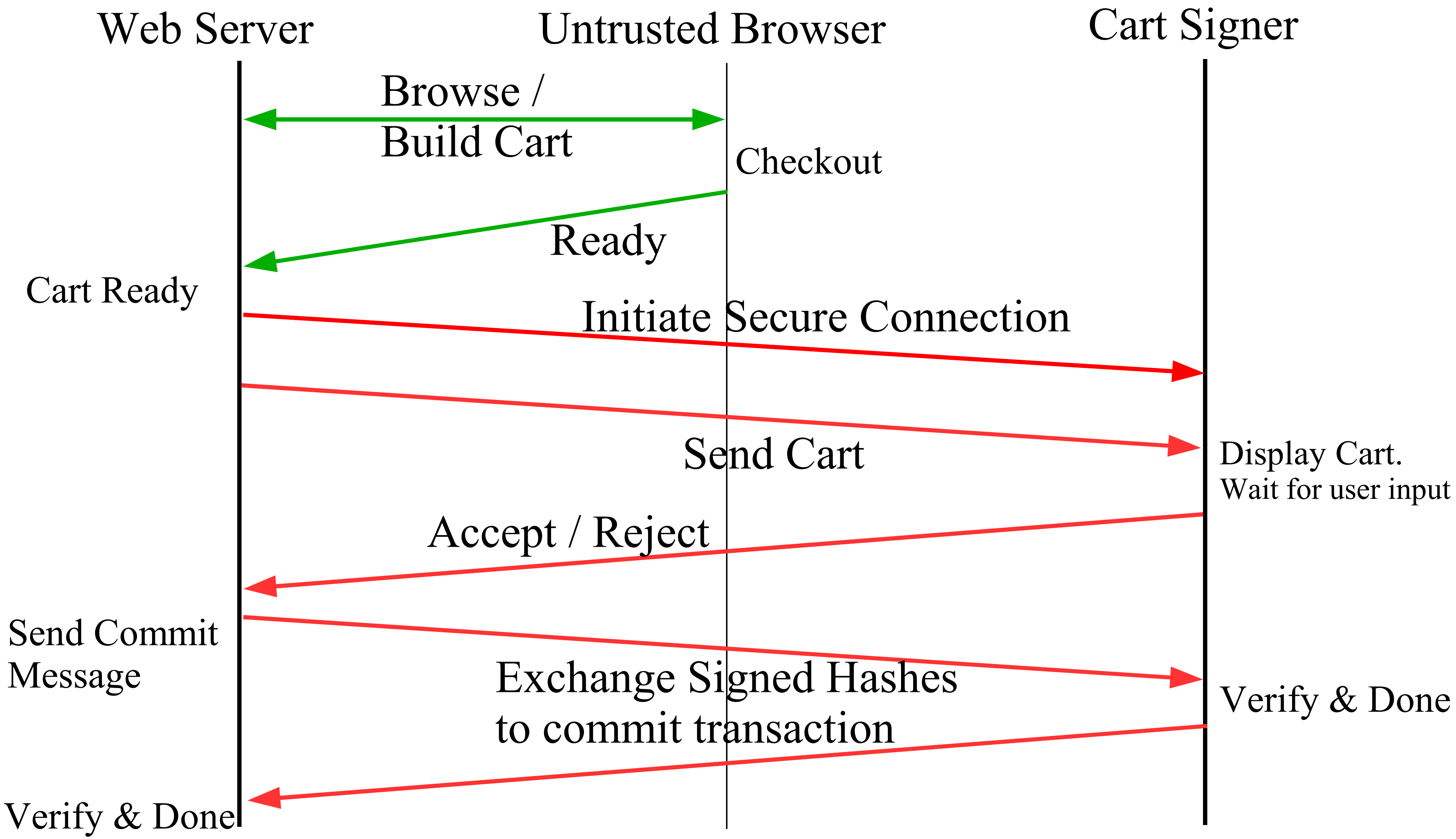
Lenin's Demo


Web Server
Browser: Cart edit
Socket Interface
Cart Sign
Linux/Win XP
L4Linux
IP Stack
L4Env: DOpE, ...
L4 Fiasco
Network

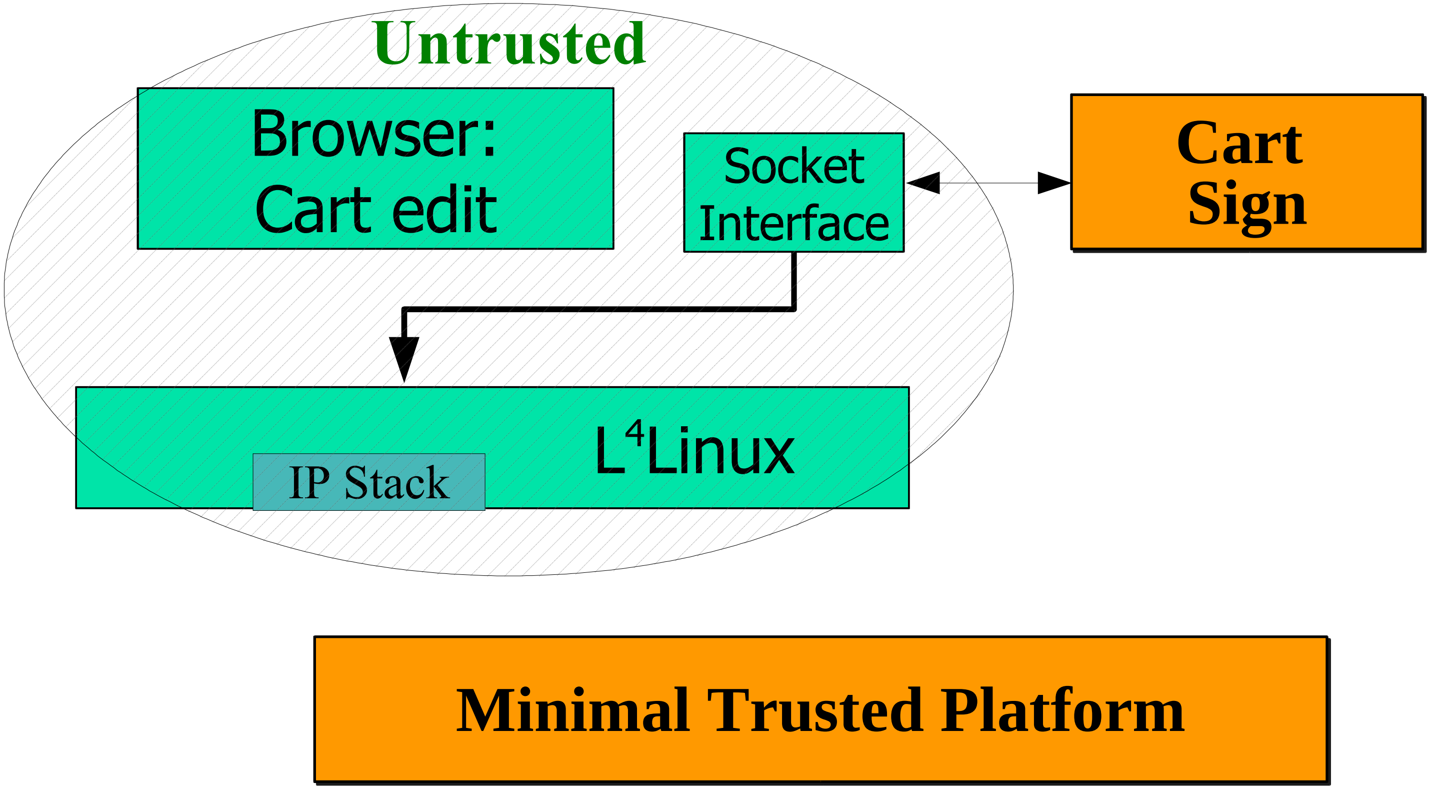
Lenin's Demo


Web Server
vi: Cart edit
Socket Interface
Cart Sign
L4Linux
IP Stack
L4Env: DopE, ...
L4 Fiasco
loopback

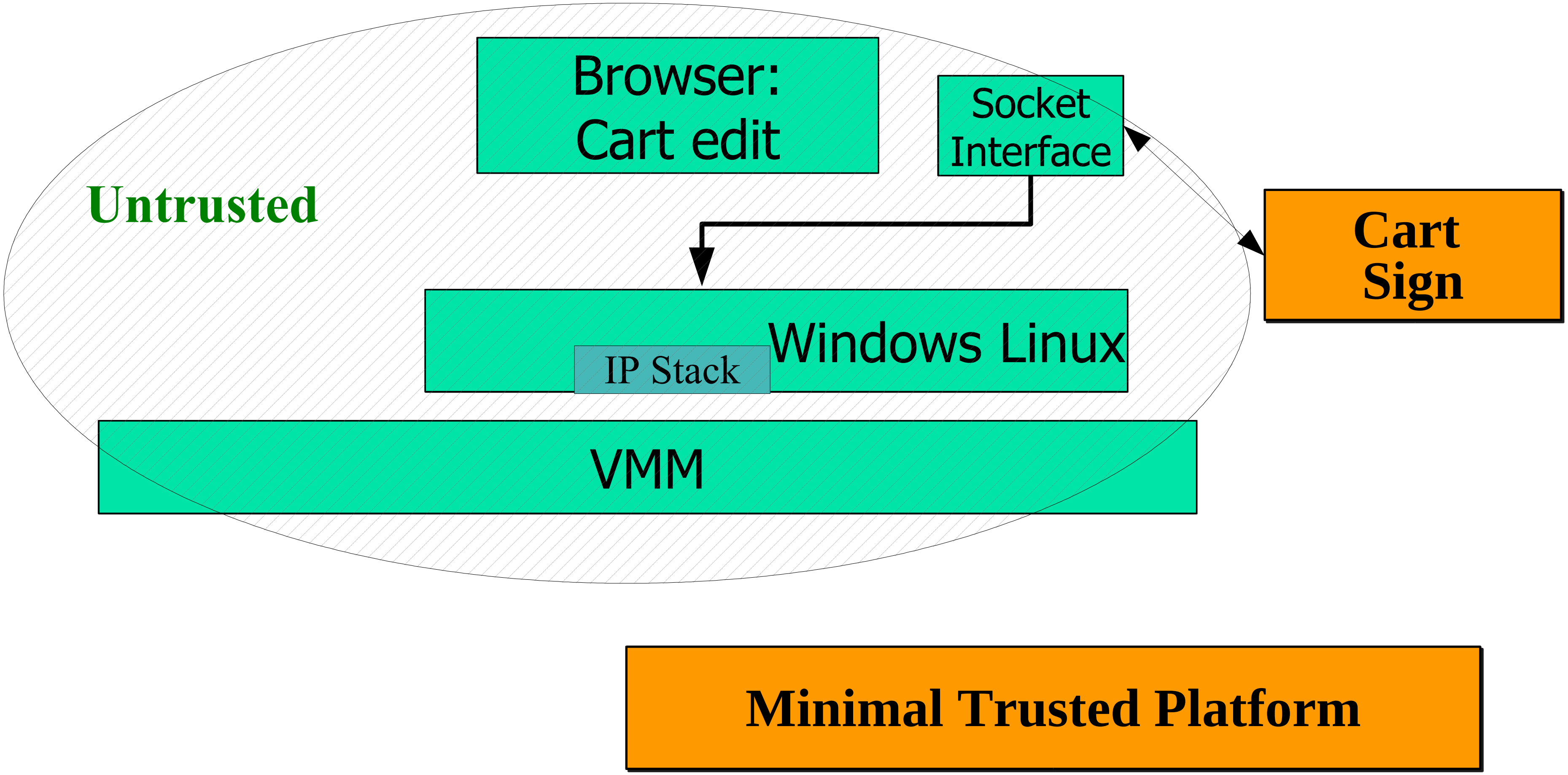
Message Sequence



Challenge: Untrusted VMM



Challenge: Untrusted VMM



Outline

- propaganda
- some names
- security and system security objectives
- design principles
- architecture and components
  - use cases
  - components: current and future
- Nizza vs. Virtual Machine Monitors and other related work
- technical risks

Names

- TUD♦OS   Technische Universität Dresden OS
  - DROPS   Dresden Real-Time OS
  - Nizza   Security Architecture
    - Micro-Sina   A Nizza Application (use case)
  - L^{4}Env   set of servers and libraries
    - DOpE   Window manager for DROPS and Nizza
  - L^{4}Linux   Linux kernel as user-level server
- L4   a micro-kernel interface
- L4/Fiasco, L4/Pistacchio: L4 implementations

Objectives: Security

- confidentiality
  no unauthorized access to information
- integrity
  no unauthorized, **unnoticed** modification of information
- recoverability
  no permanent damage to information
- availability
  timeliness of service

Objectives: System Security

- Secure and unsecure applications

Objectives: System Security

- Secure and unsecure applications
- Compatibility:
  - Legacy applications
  - Legacy/Fashionable Hardware

Objectives: System Security

- Secure and unsecure applications
- Compatibility
- Flexible sandboxing

Objectives: System Security

- Secure and unsecure applications
- Compatibility
- Flexible sandboxing
- Resource Control

Objectives: System Security

- Secure and unsecure applications
- Compatibility
- Flexible sandboxing
- Resource Control
- Small TCB:
  complexity acceptable, if for a small group
  - Each member fully understands interaction of all components.
  - Each component is understood by one member.

Principles

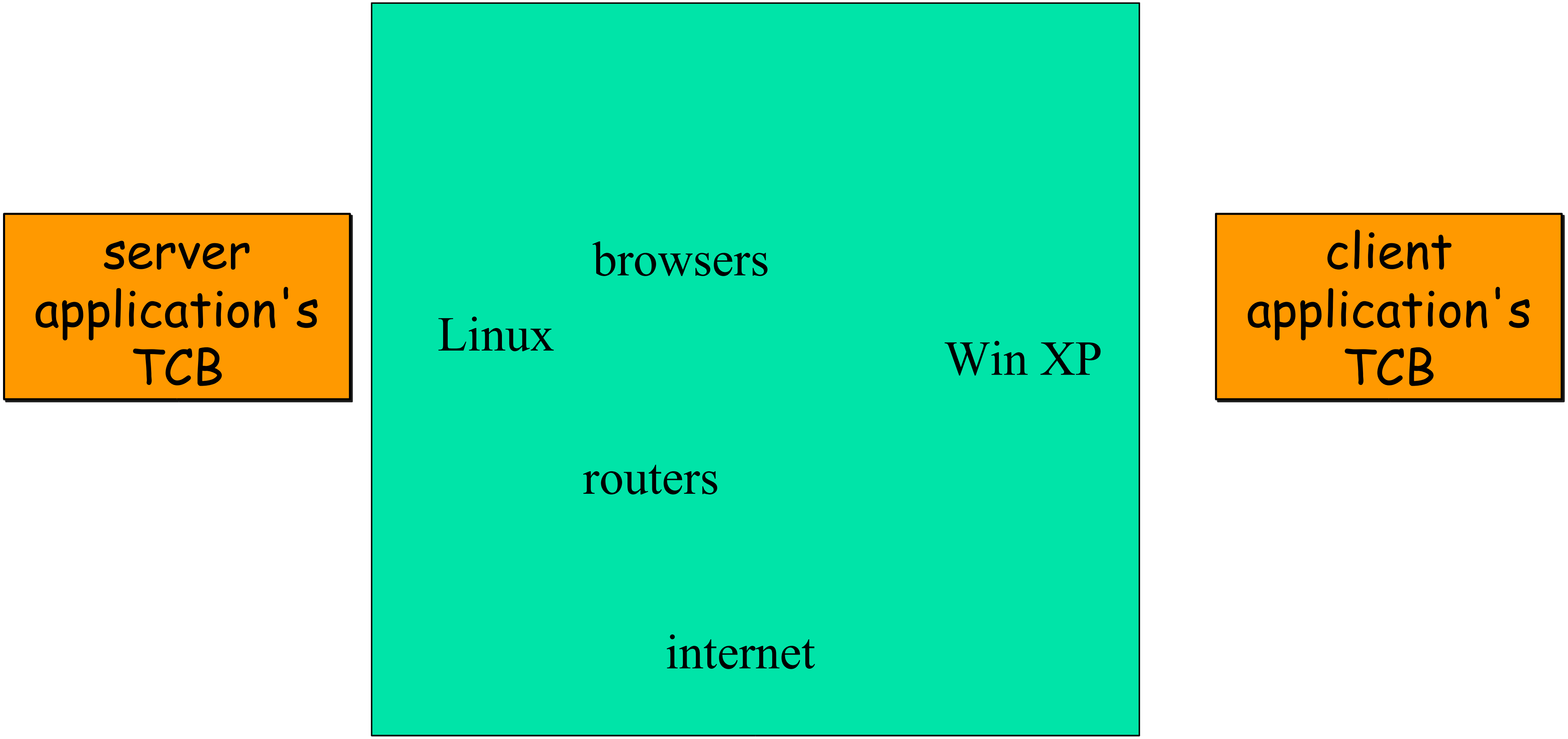
- Trusted Computing Base: per application
- platform:
  - small set of small components (servers, ...)
  - small interfaces
  - select components of platform per application

Principles, continued

- split applications and services: sensitive part in/on trusted platform and *other* part
- reuse legacy for *other* part
  –> trusted wrappers / tunneling

Principles

- push end-to-end argument to the extreme



server application's TCB

browsers

Linux

Win XP

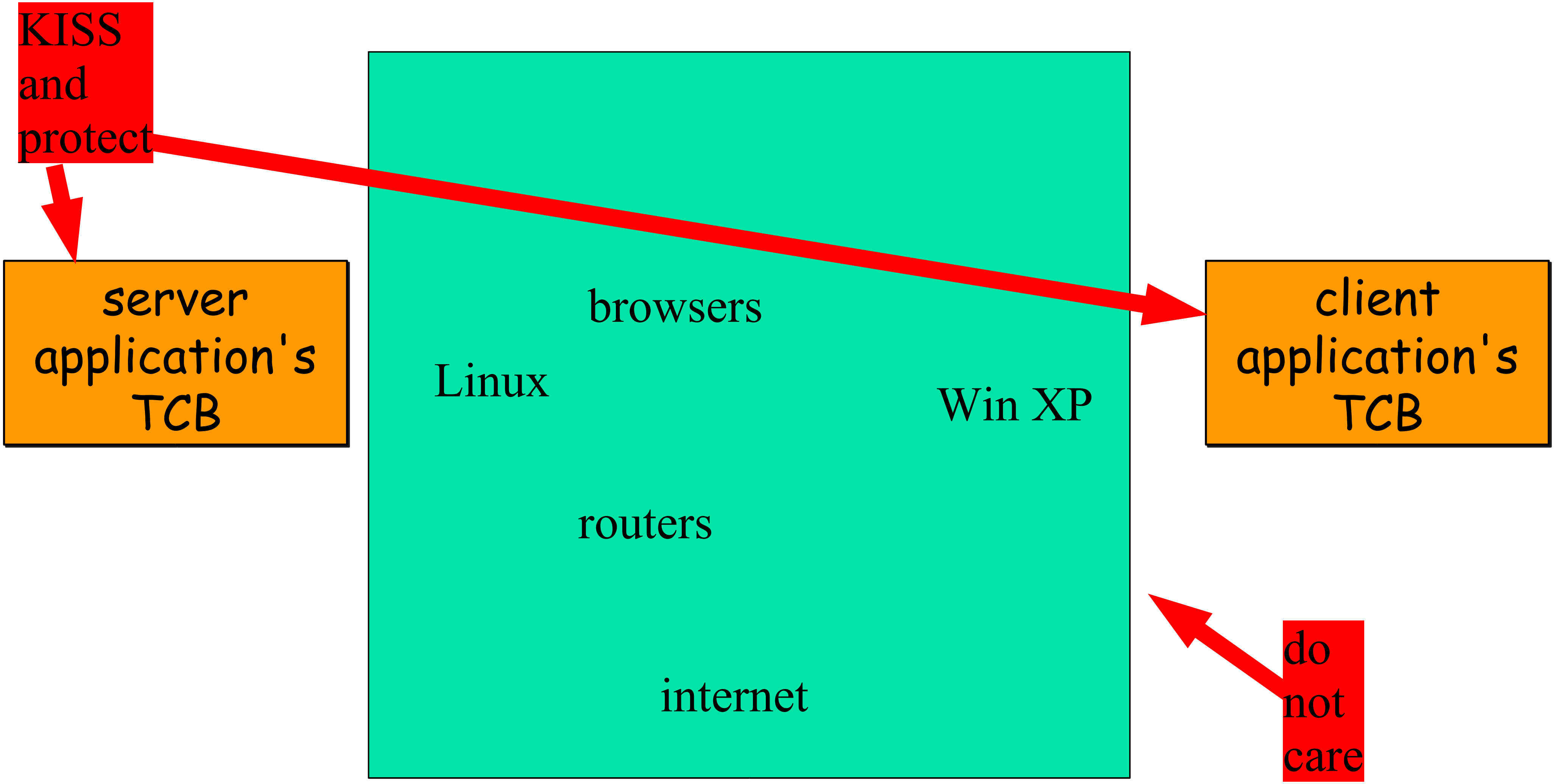
routers

internet

client application's TCB

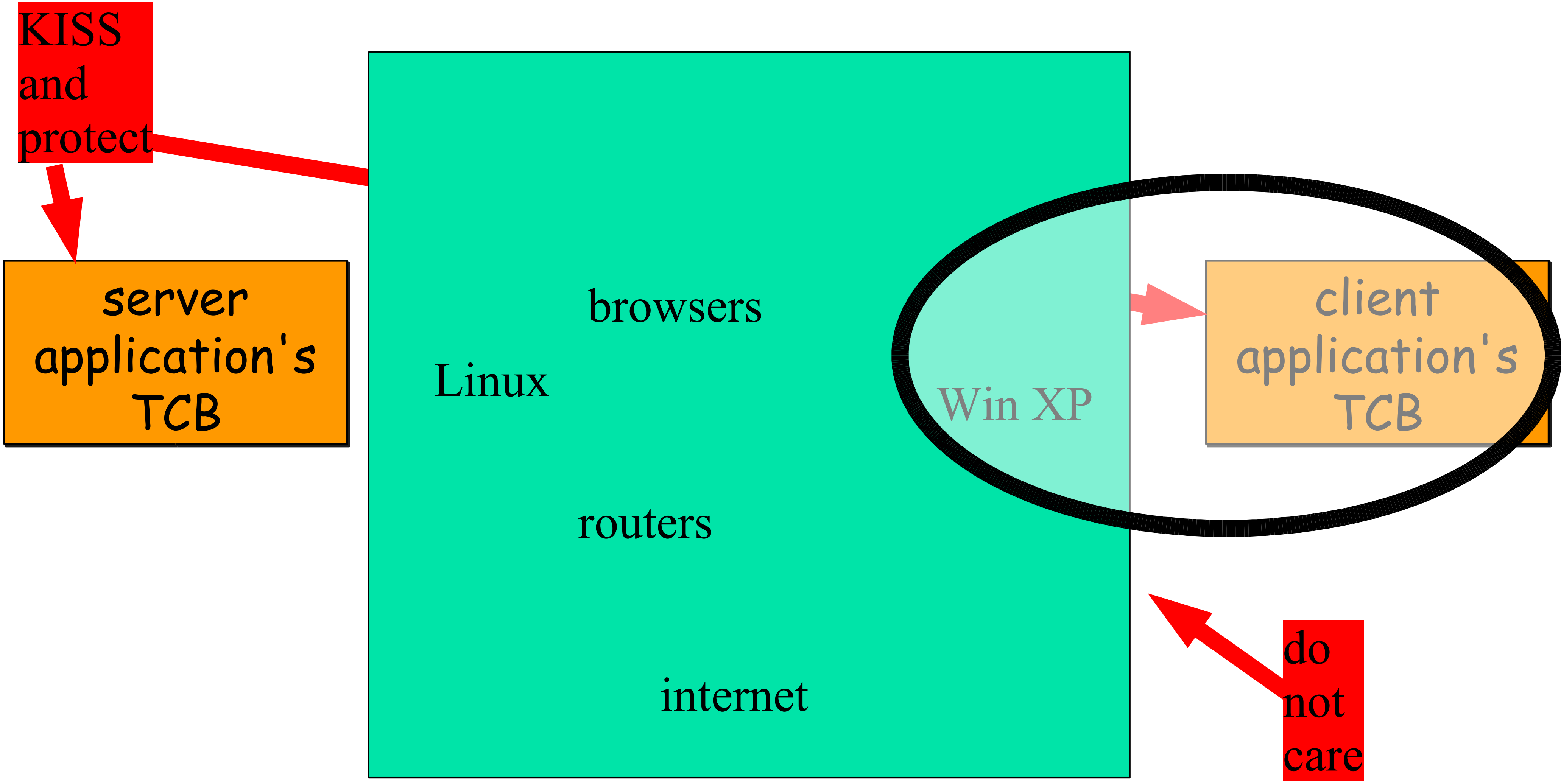
Principles

- push end-to-end argument to the extreme



Principles

- push end-to-end argument to the extreme



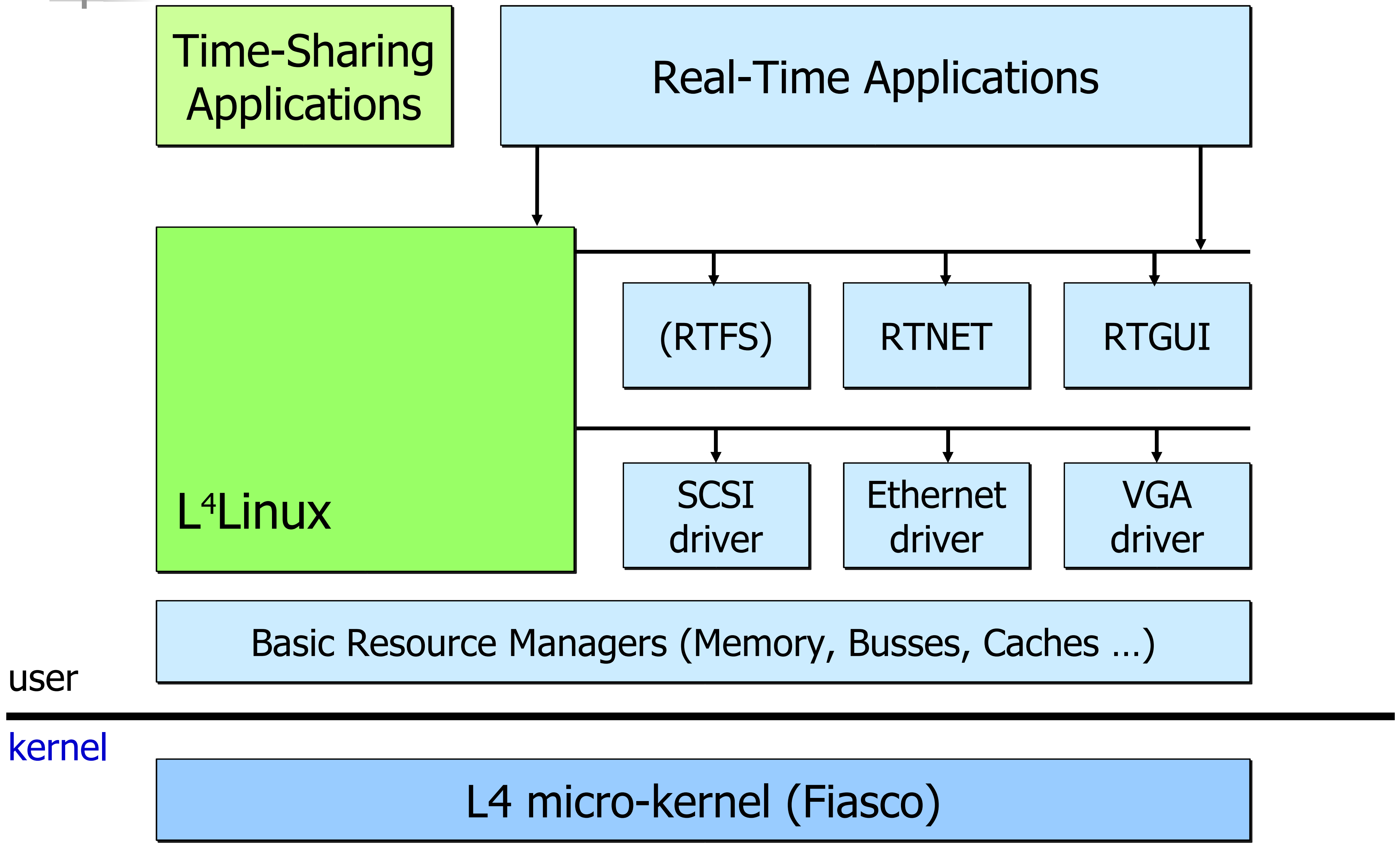
Principles and Techniques

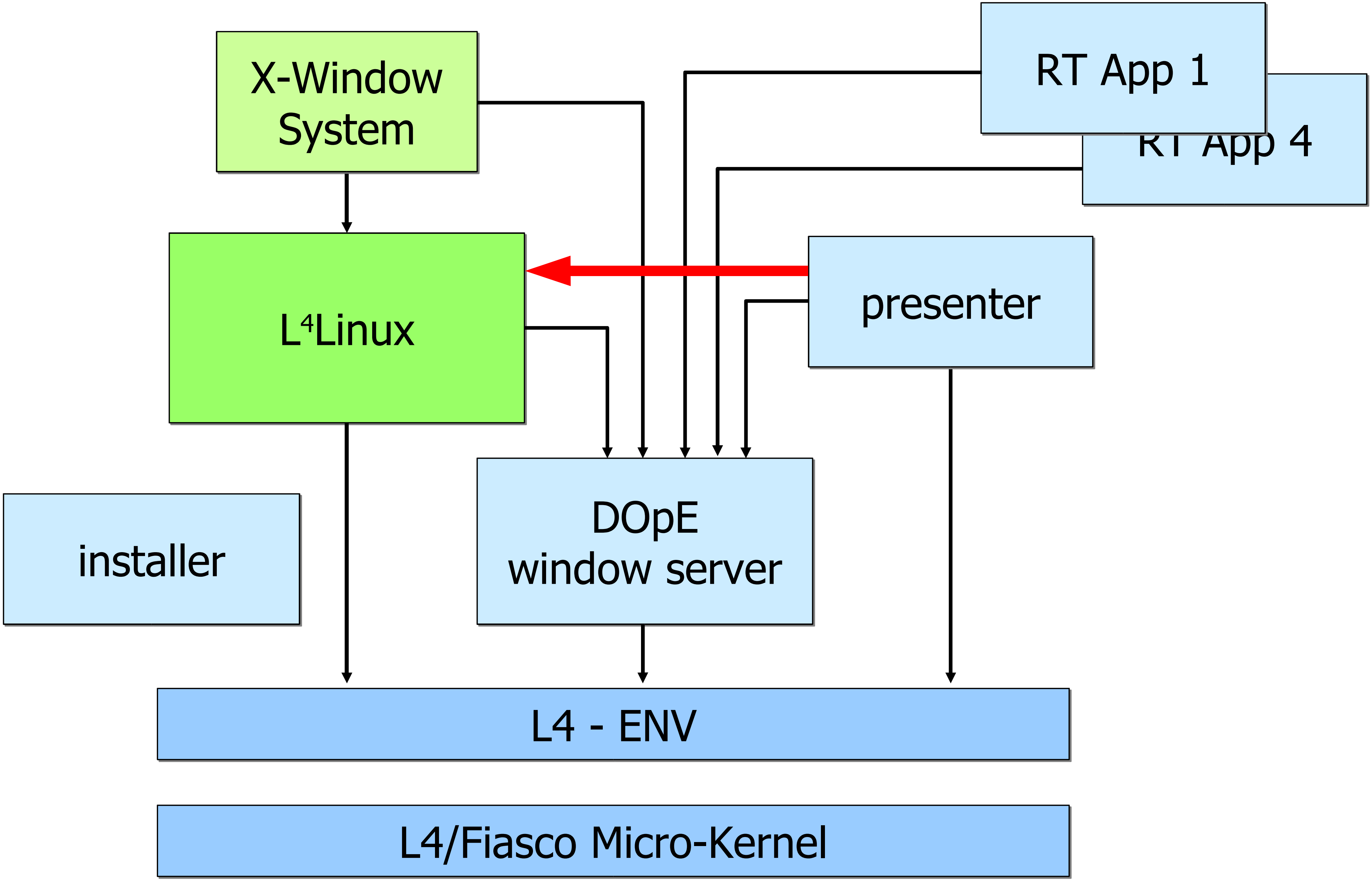
- micro kernel:
  - separates legacy from sensitive partitions
  - separates components of small platform
  - provides mechanisms for access control mediates communication
- contract-based access control
- secure booting / attestation + trusted path to user

Architecture + Components

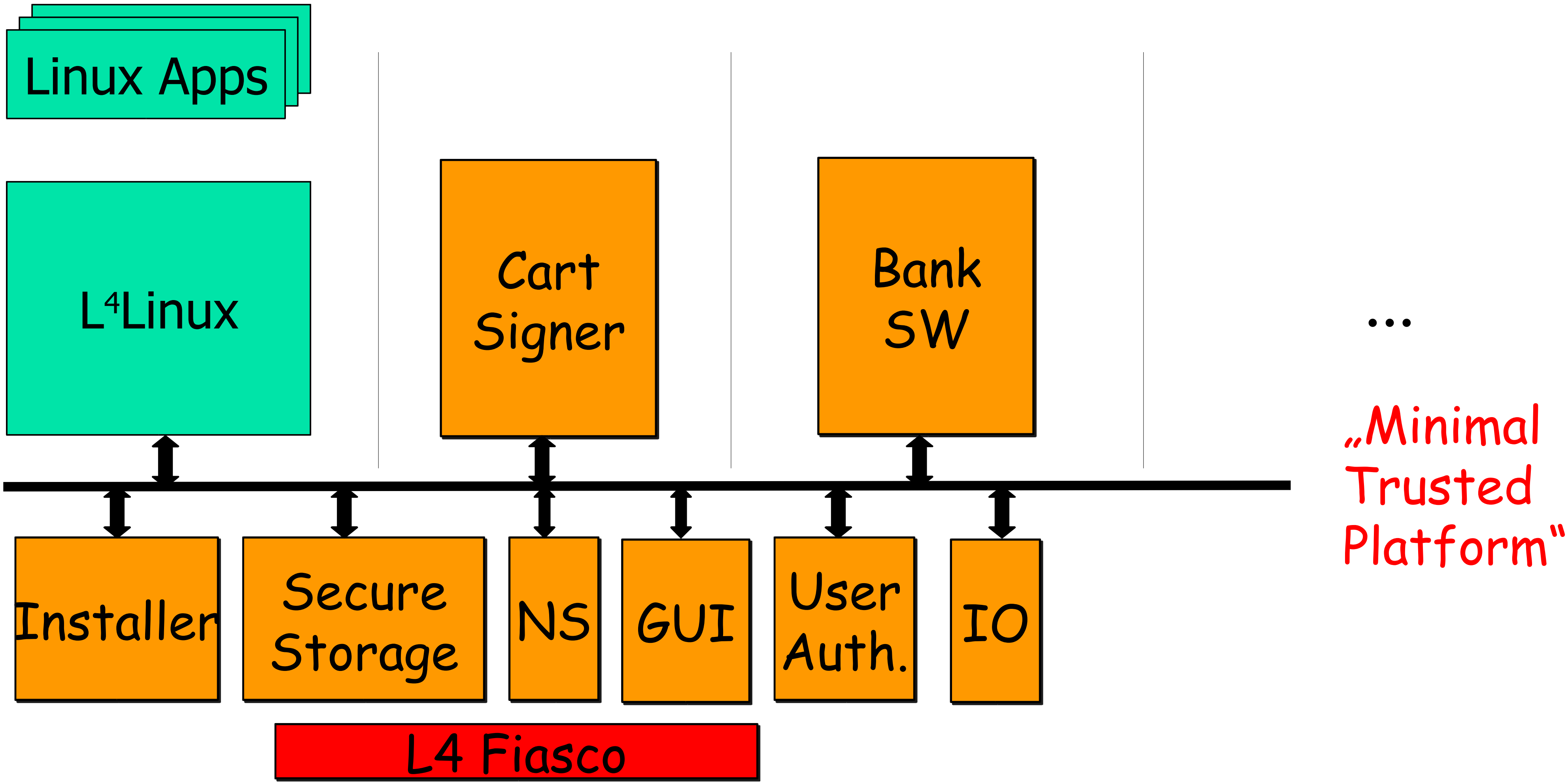
- the origin: Dresden Real-time OPerating System
- Nizza architecture
- use cases
- some Nizza components

Starting Point: DResden Real-Time OS



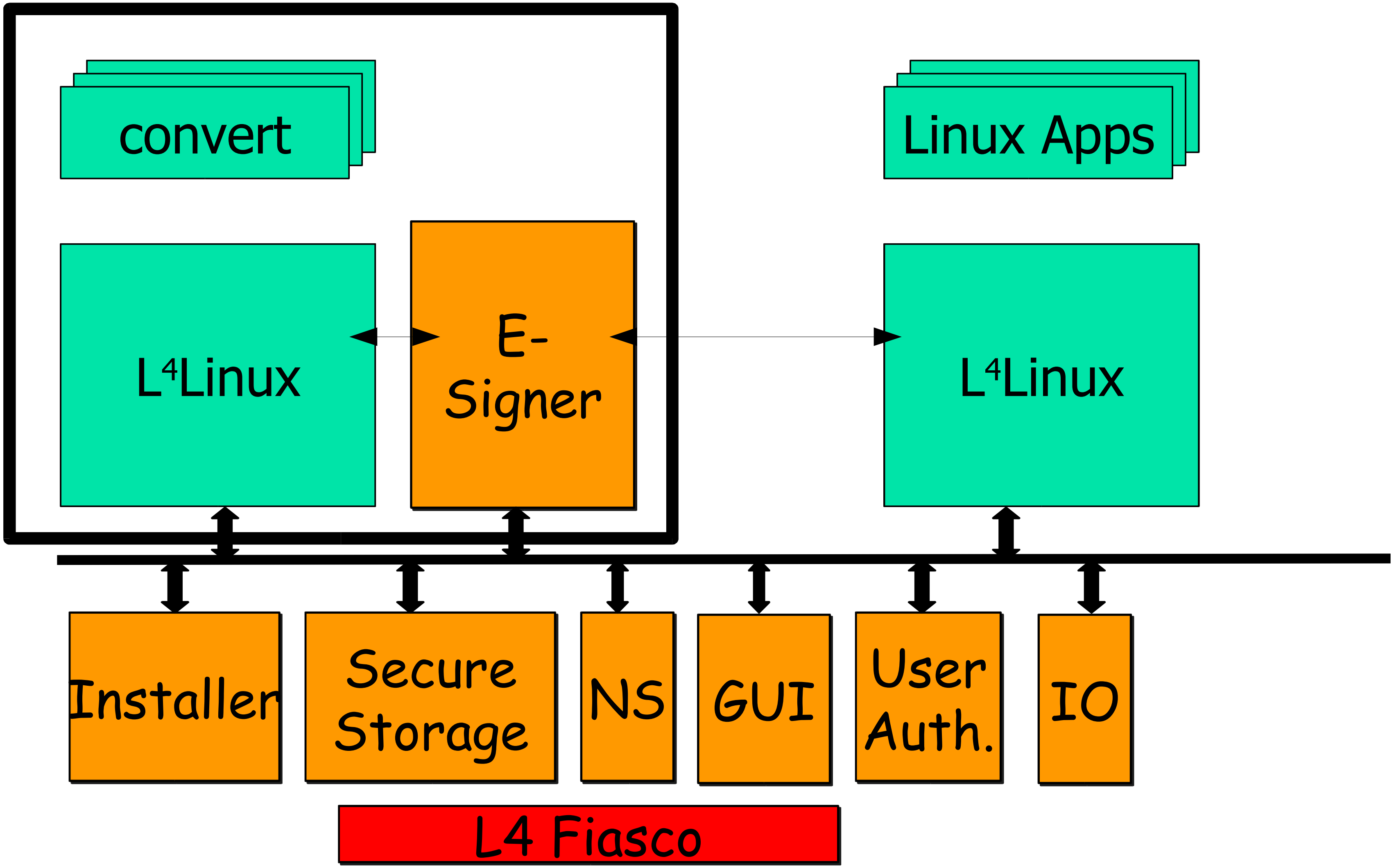

X-Window System
RT App 1
RT App 4
L⁴Linux
presenter
installer
DOpE window server
L4 - ENV
L4/Fiasco Micro-Kernel

NIZZA Architecture

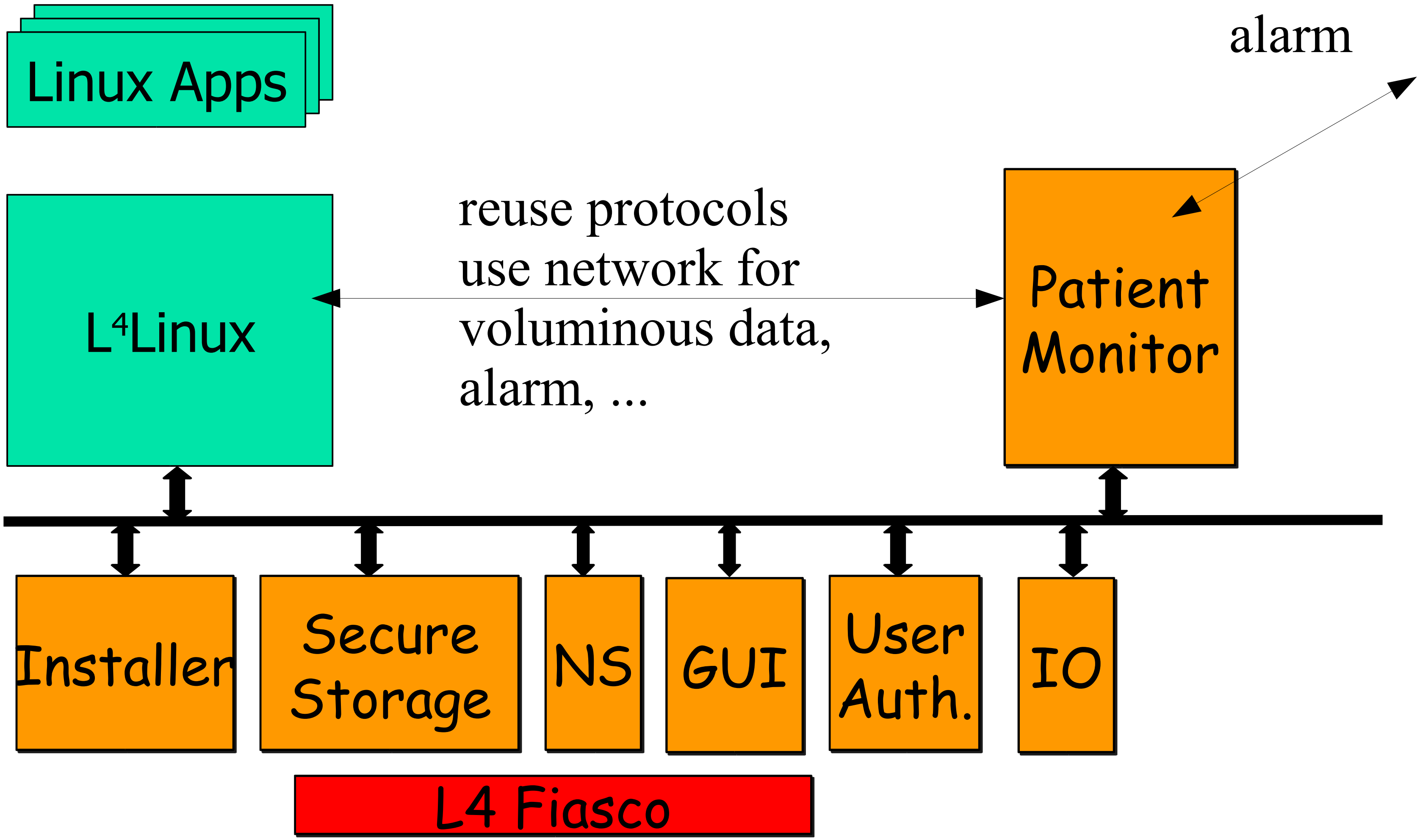


„Minimal Trusted Platform“

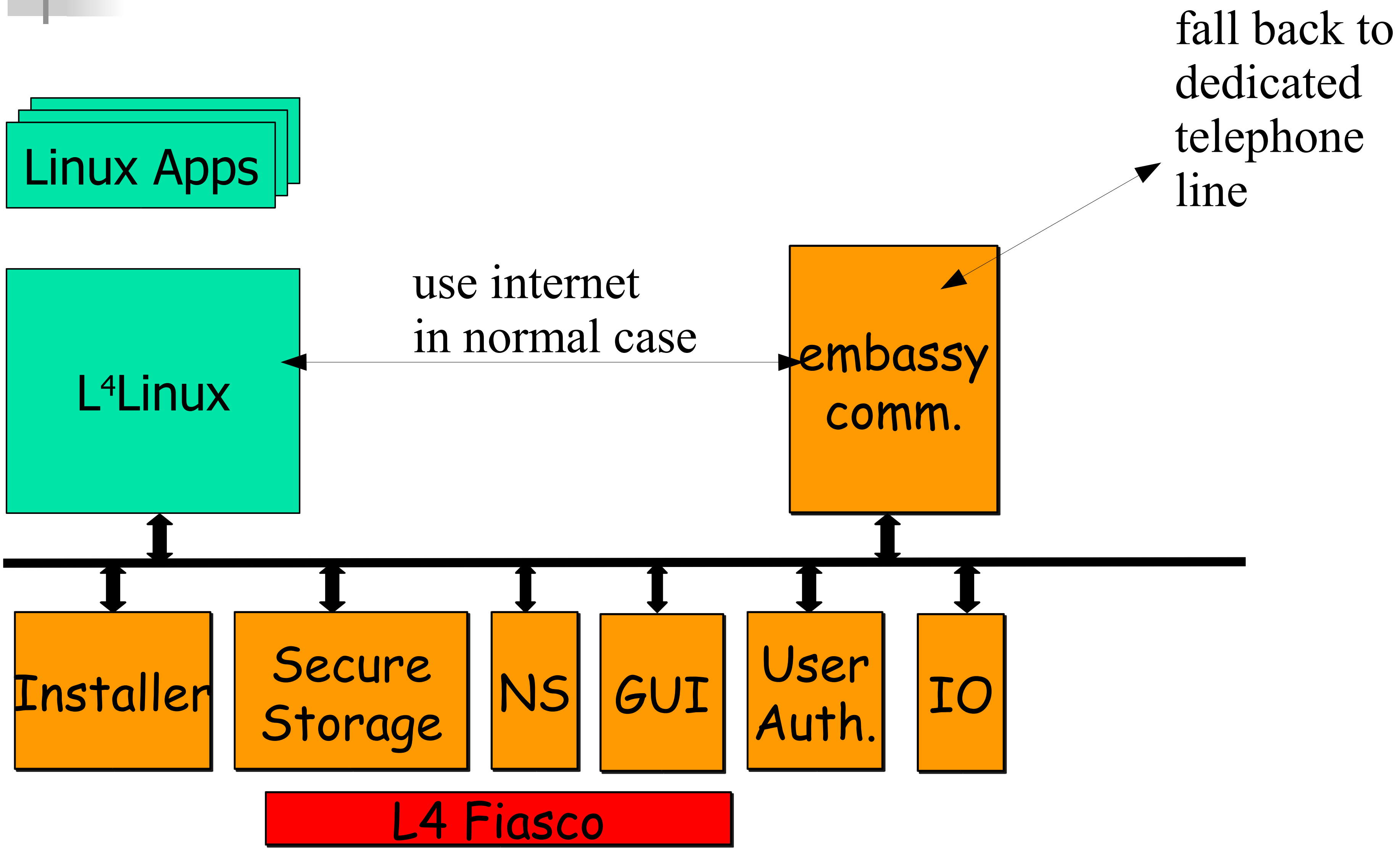
More Use Cases



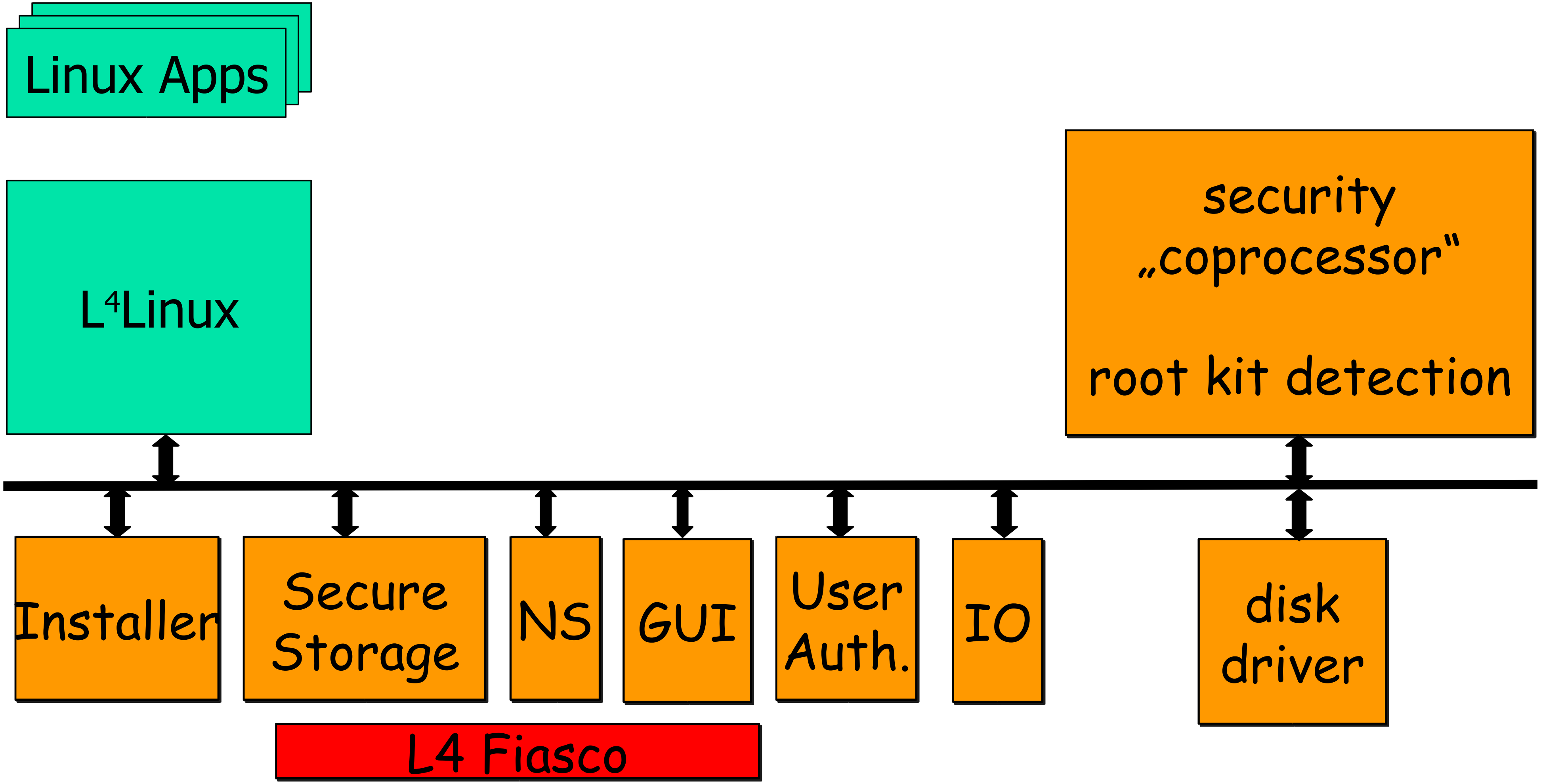
More Use Cases



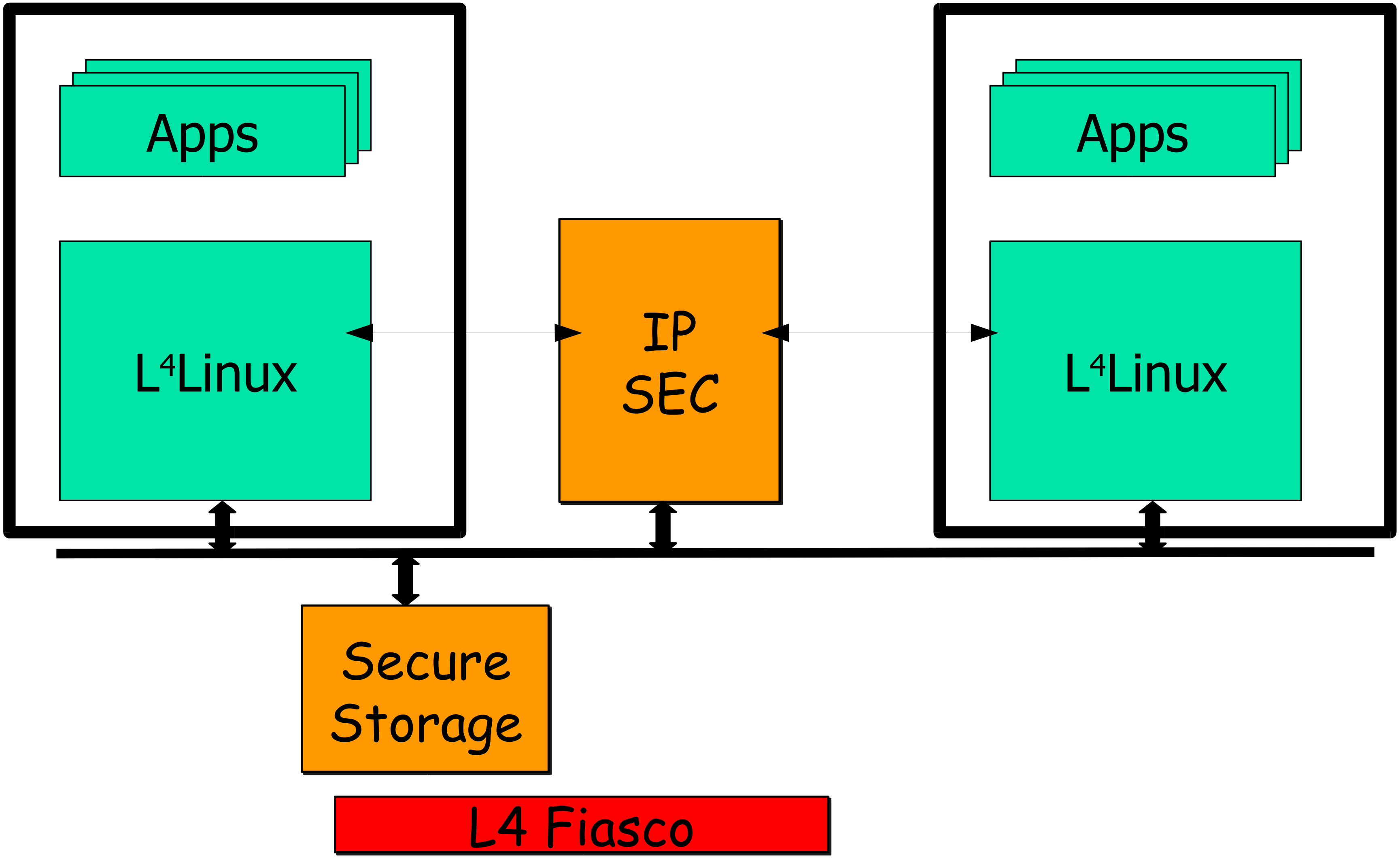
More Use Cases



More Use Cases



More Use Cases



Architecture + Components

- the origin: Dresden Real-time OPerating System
- Nizza architecture
- use cases
- **some Nizza components**
  - L4 micro kernel: present and future
  - L^{4}Linux: encapsulation and reuse
  - secure booting + trusted path
  - secure storage with small TCB (future)

L4 Micro-Kernel: evolution

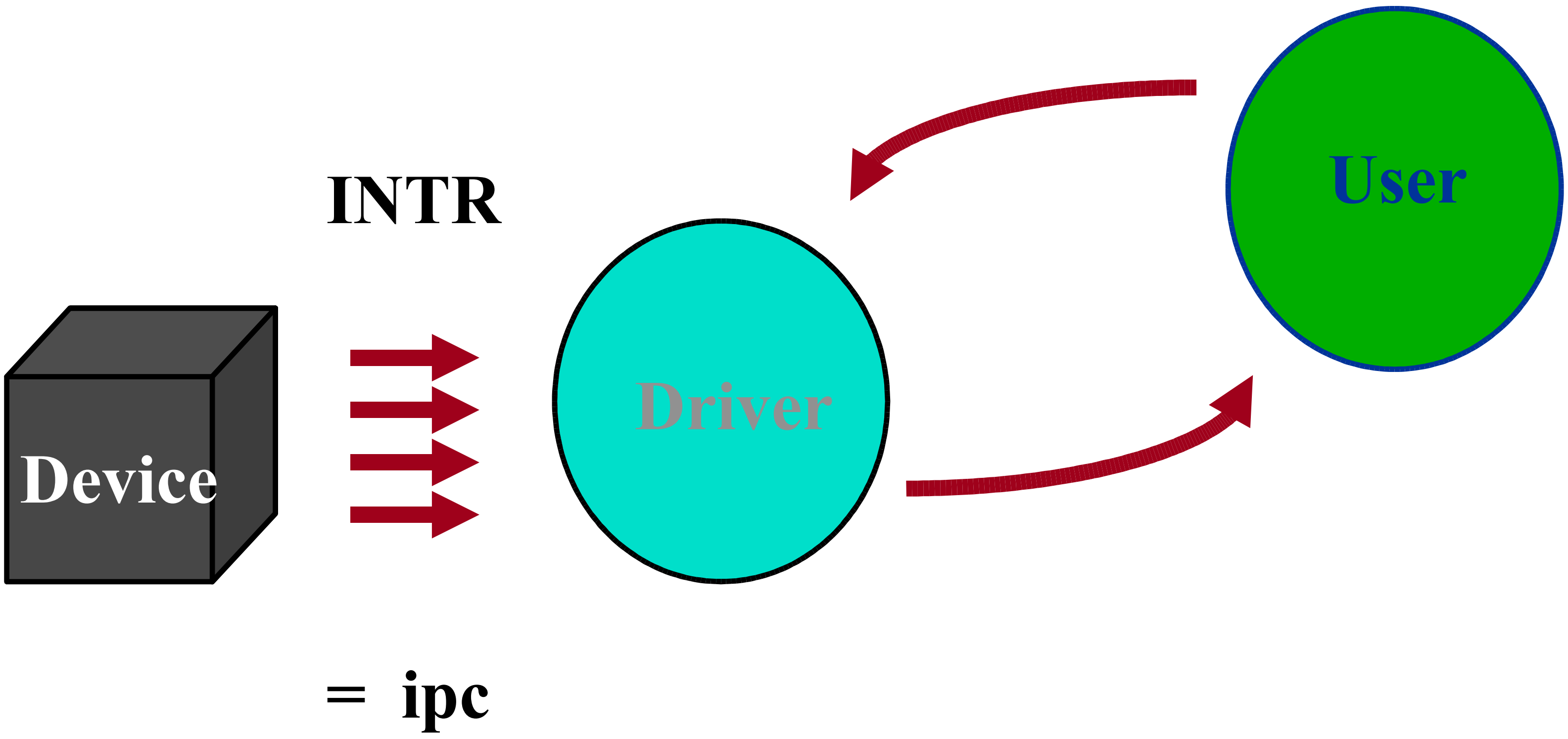
- the original:
  address spaces, threads, IPC
- L4/Fiasco-RT: real-time:
  periodic threads, fine grained scheduling support, etc
- **L4/Fiasco-X.e:**
  unified access control +
  kernel resource management
- L4-Next: virtualization support

L4 micro kernel (Jochen Liedtke)

fundamental abstractions

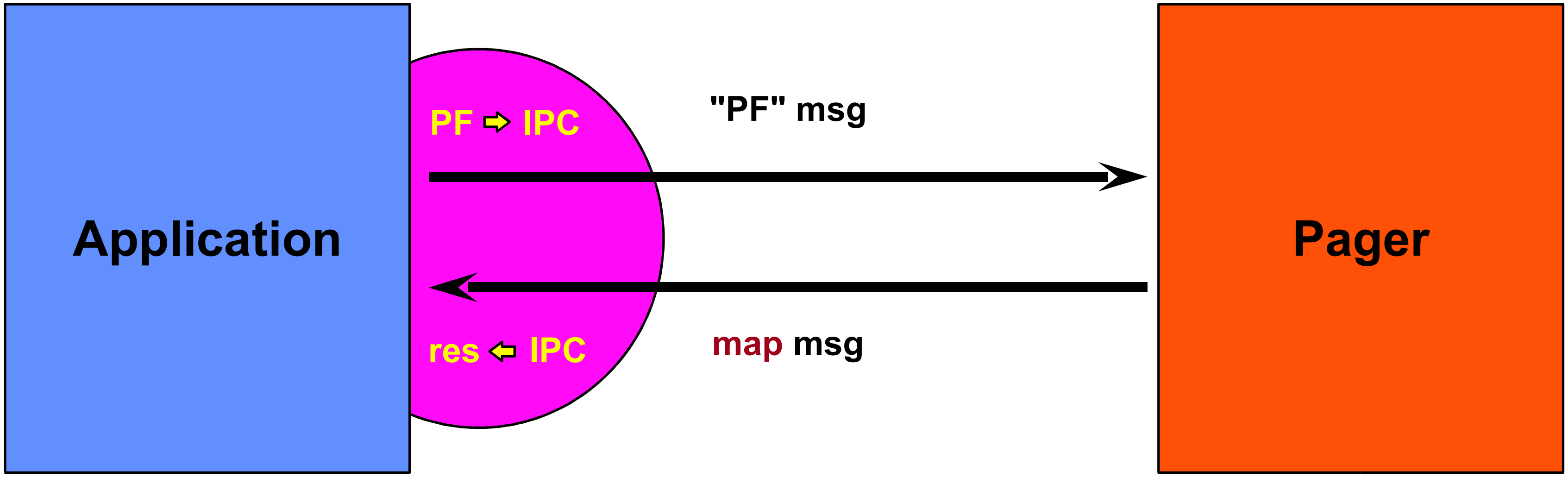
- address spaces (separation)
- threads
- inter process communication (IPC)
  - explicit
  - interrupts
  - faults and mappings

Drivers at User Level

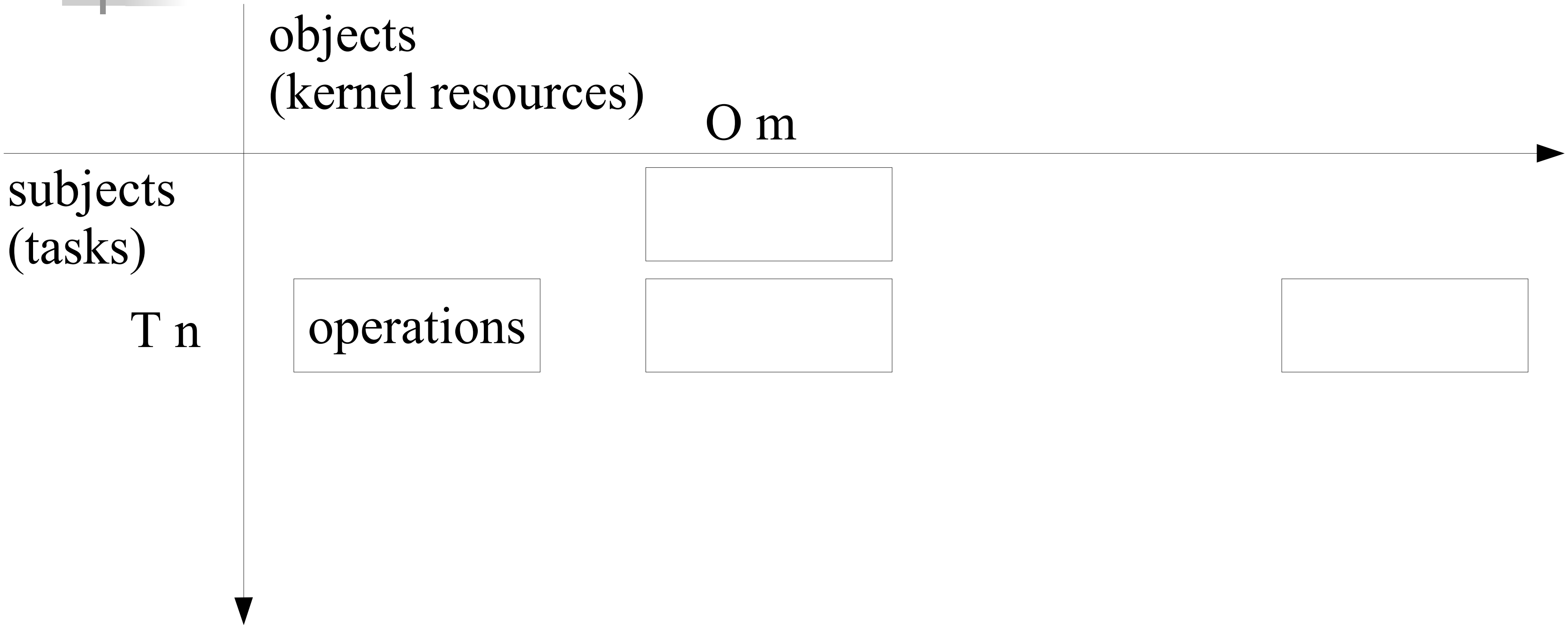


- **IO ports: part of the user address space**
- **interrupts: messages from hardware**

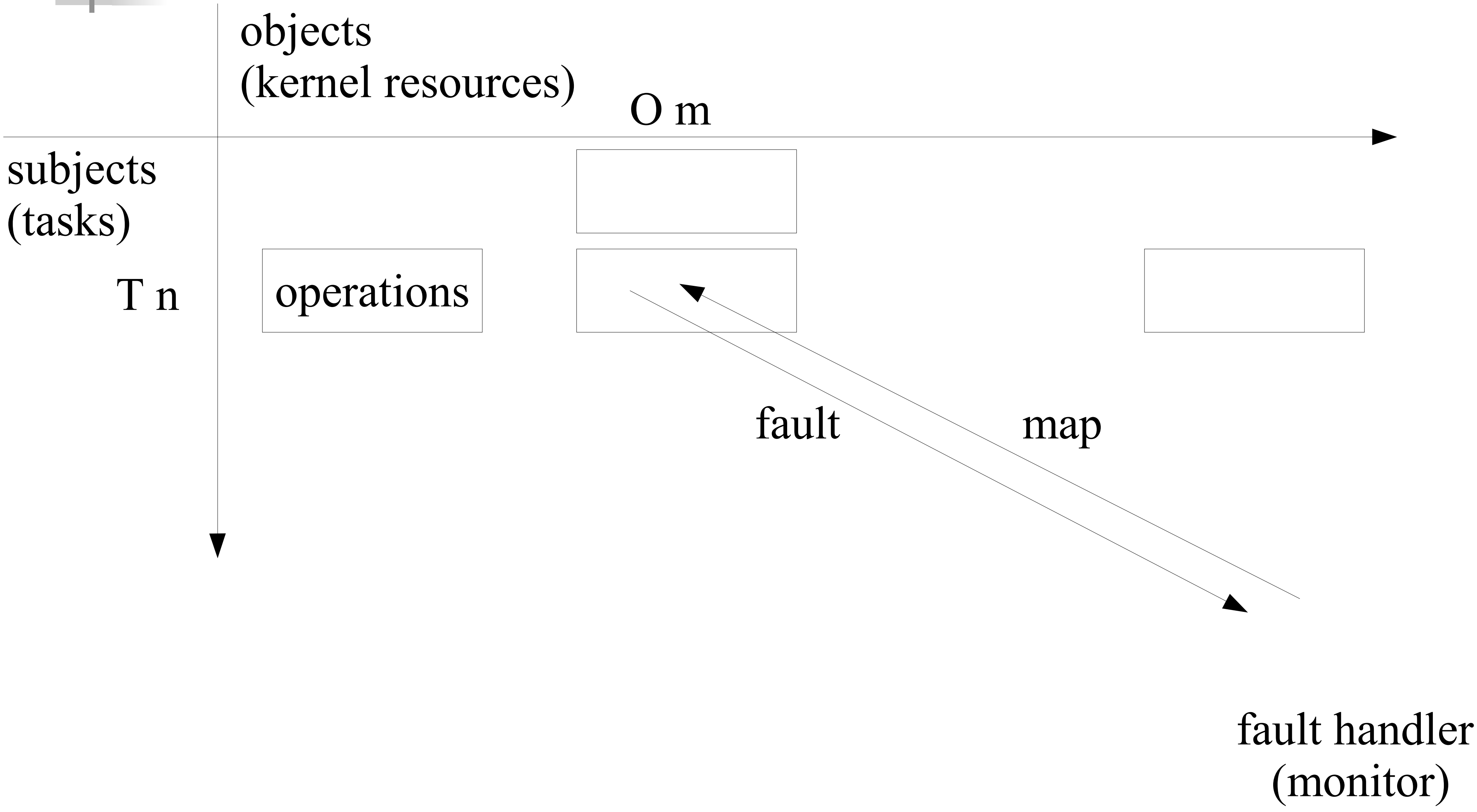
Memory Pages



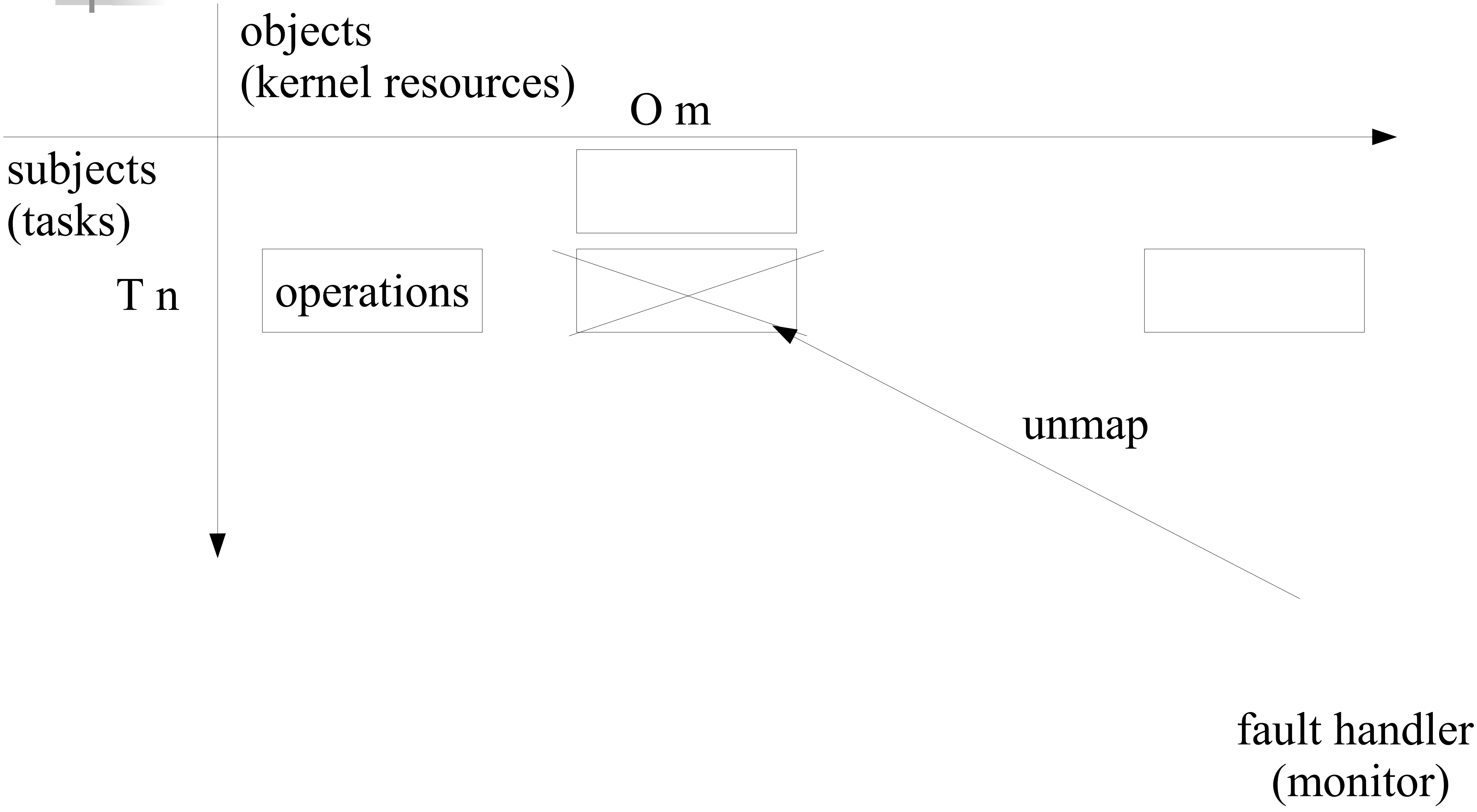
L4 X.e: Unifying Access Control



L4 X.e: Unifying Access Control



L4 X.e: Unifying Access Control

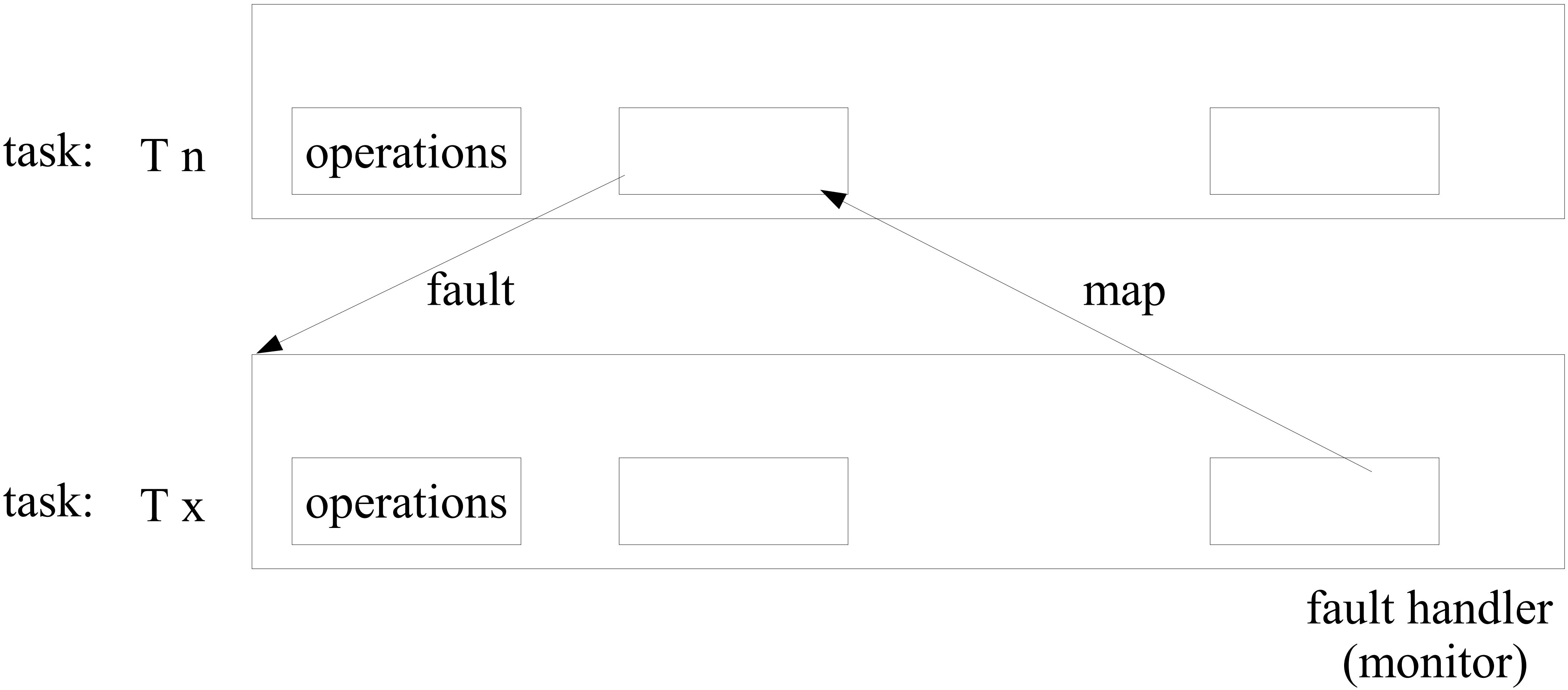


L4 X.e: Unifying Access Control

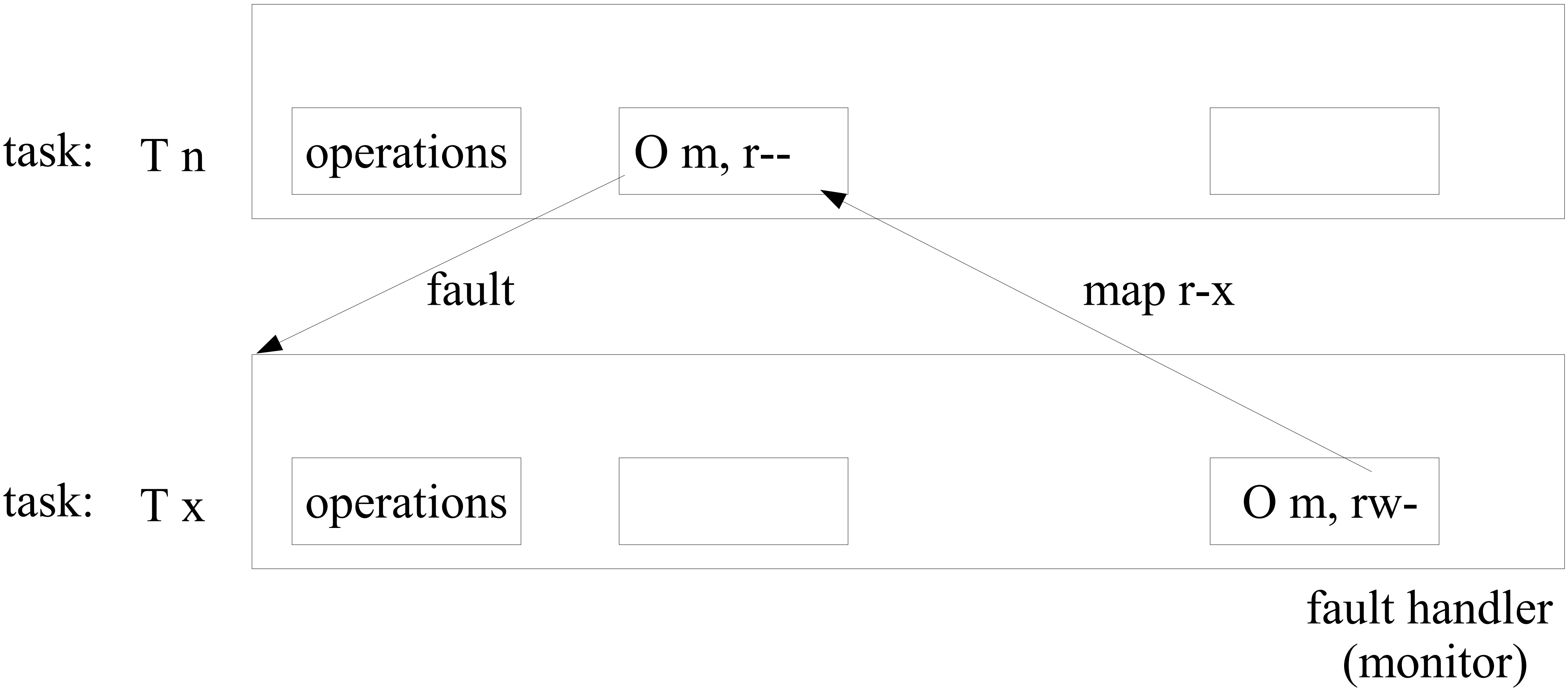
task: T n

operations

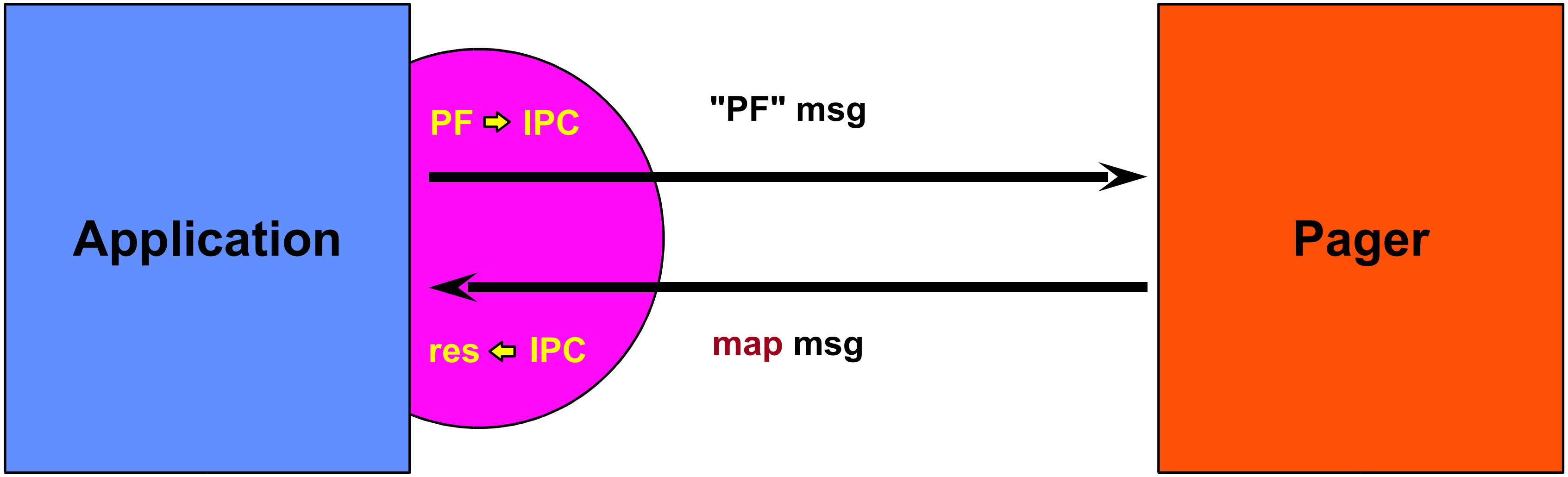
L4 X.e: Unifying Access Control



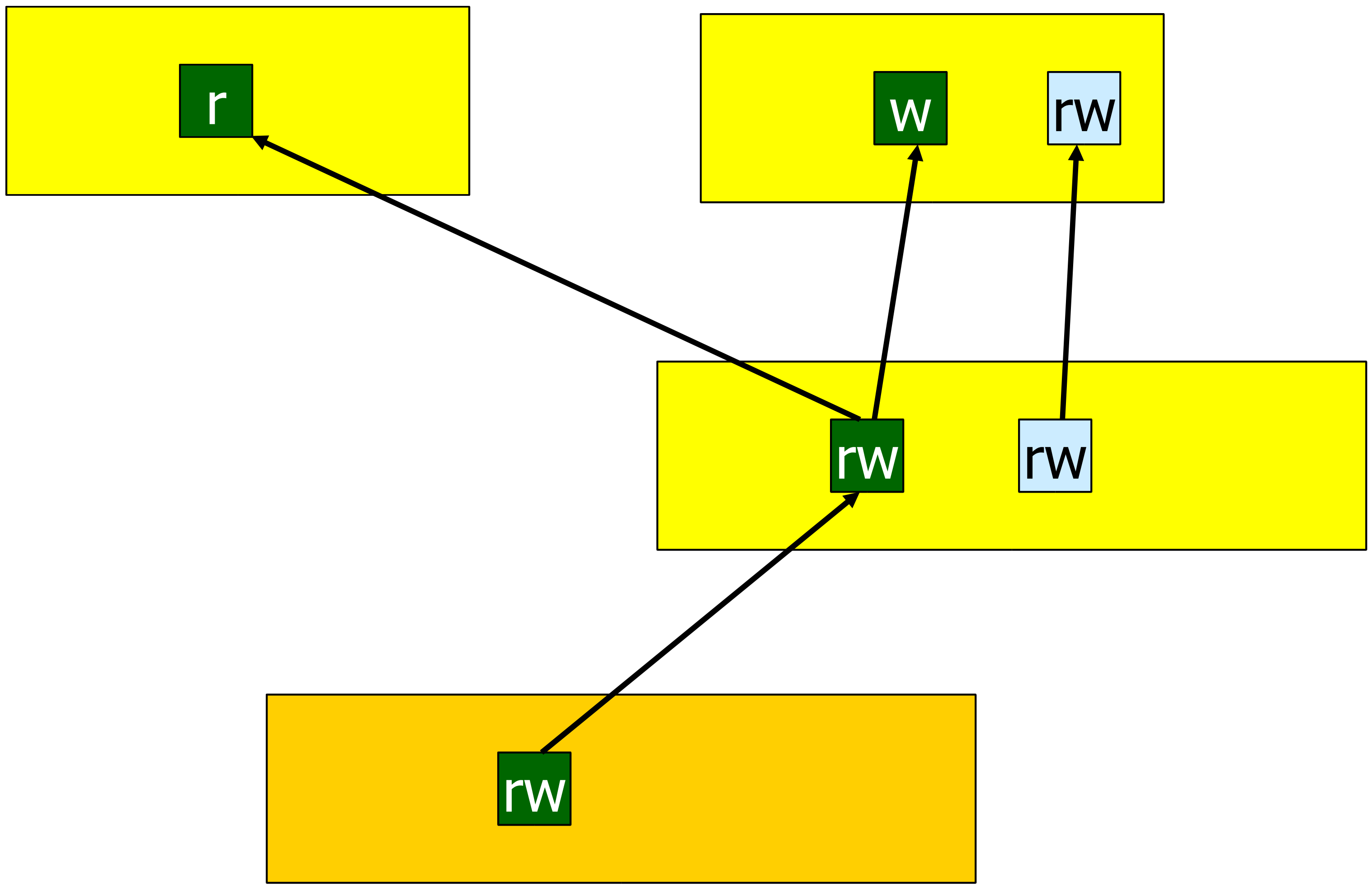
L4 X.e: Unifying Access Control



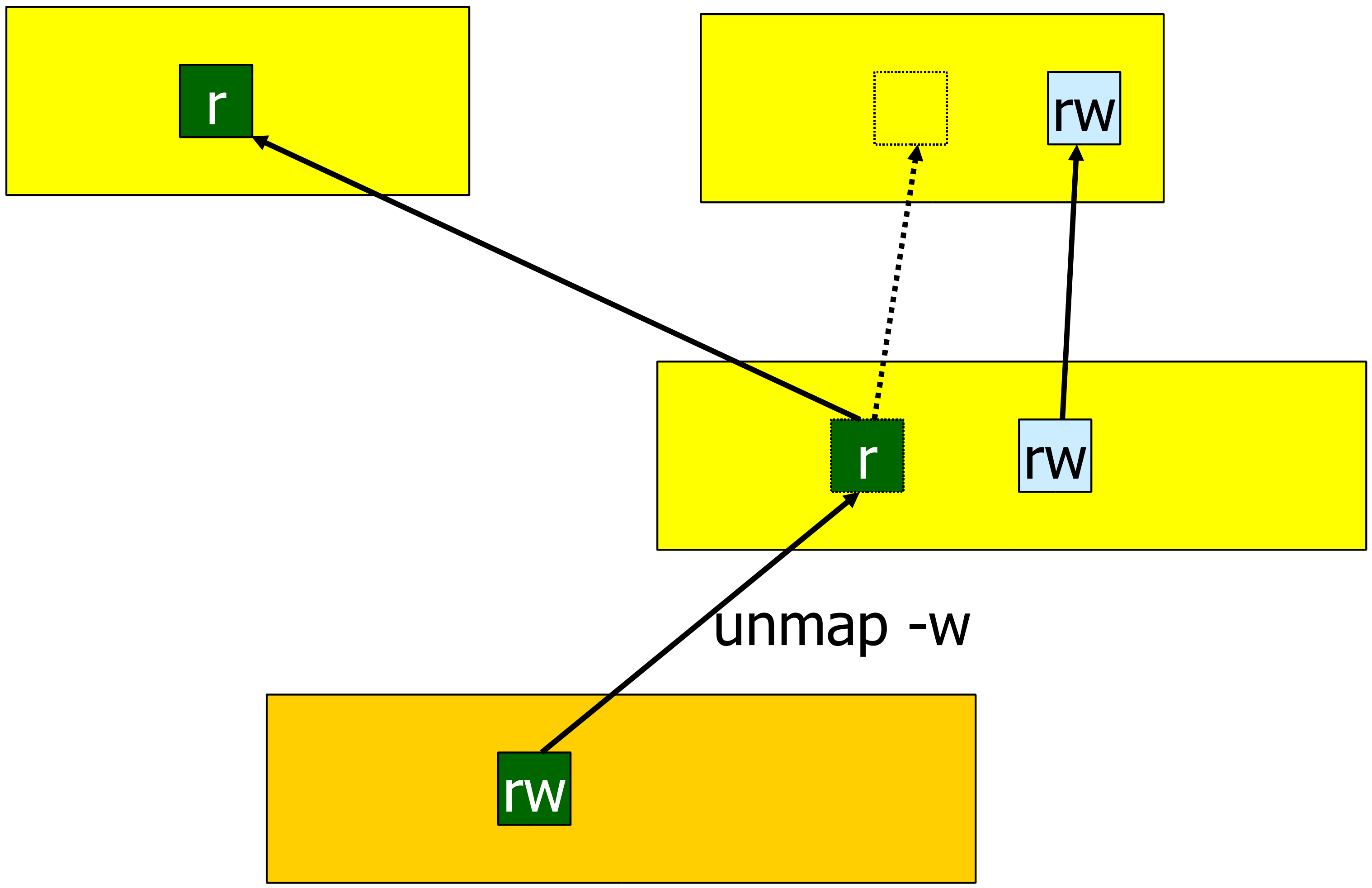
Memory Pages



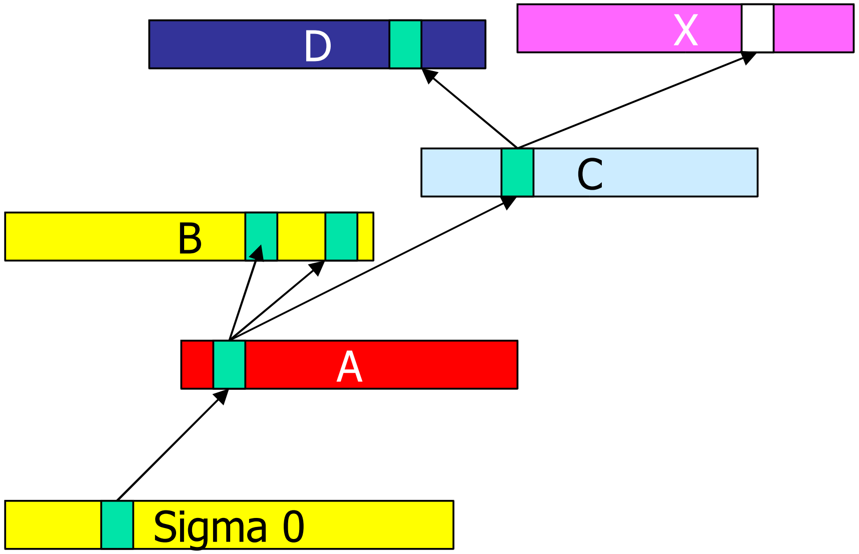
Fault, Map, Unmap

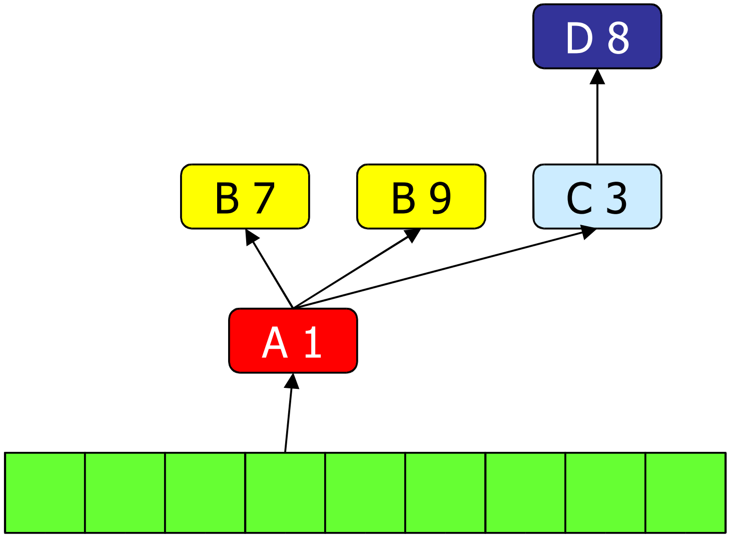


Fault, Map, Unmap



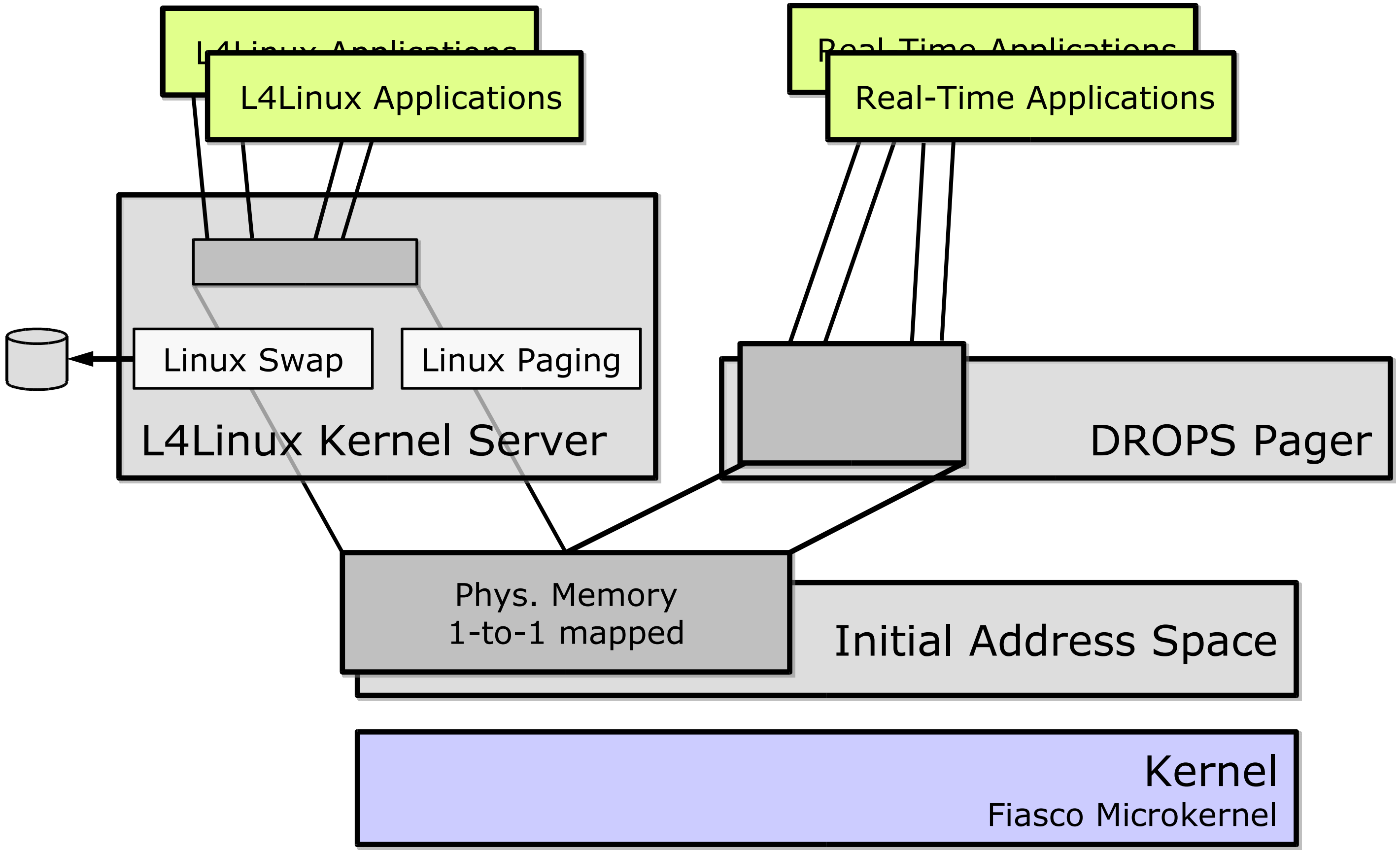
Map Trees



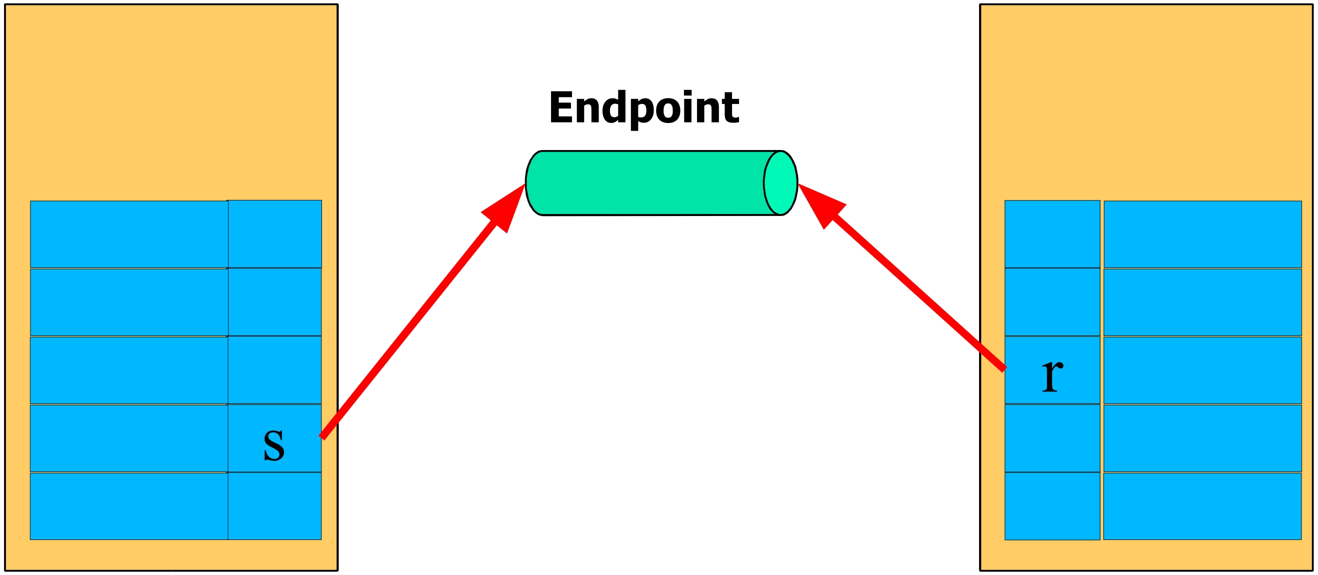


Tiles (physical memory frames)

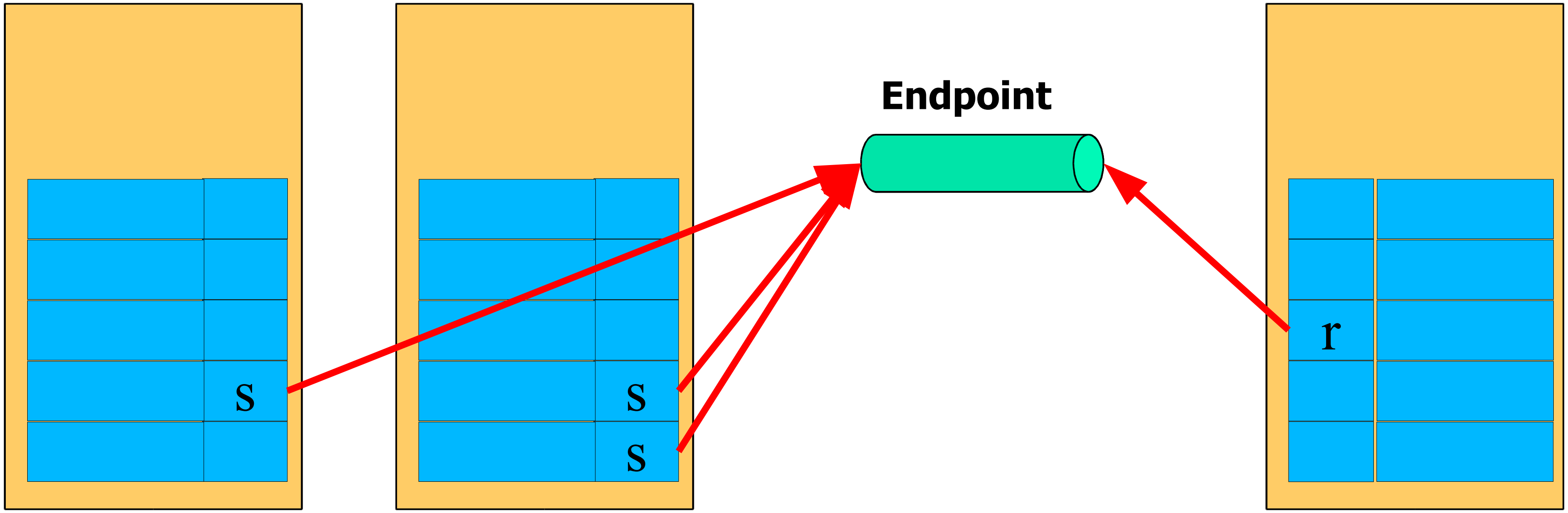
Pager Example



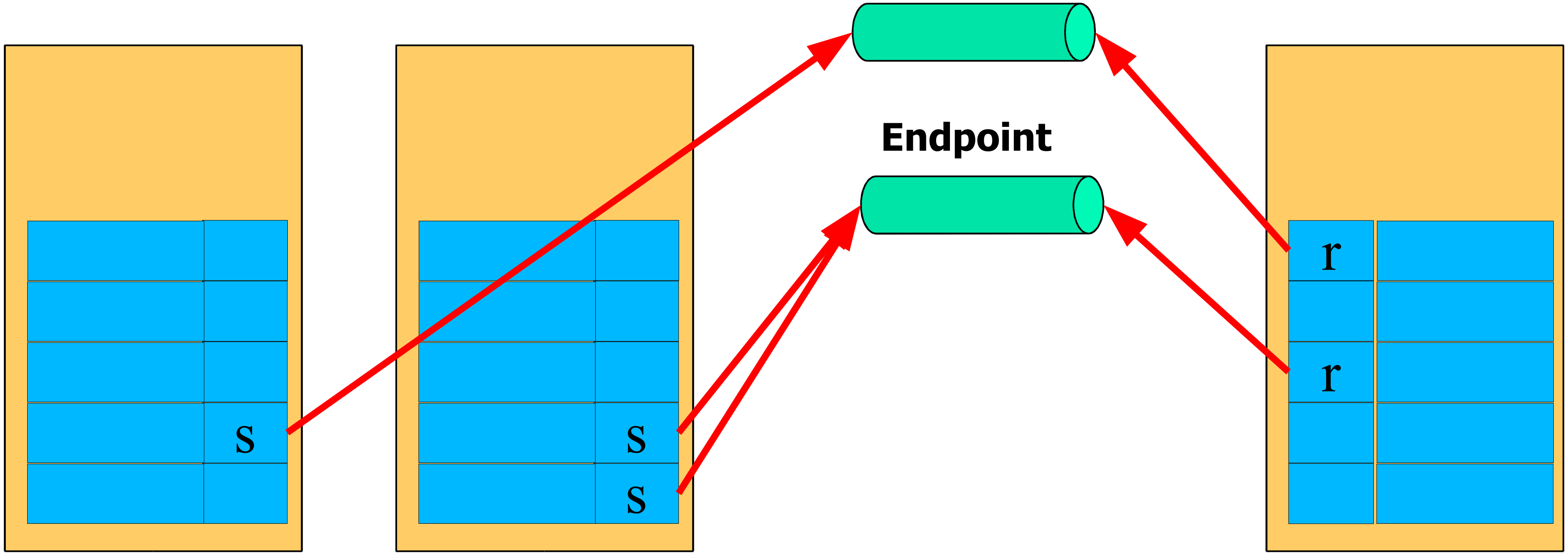
Communication



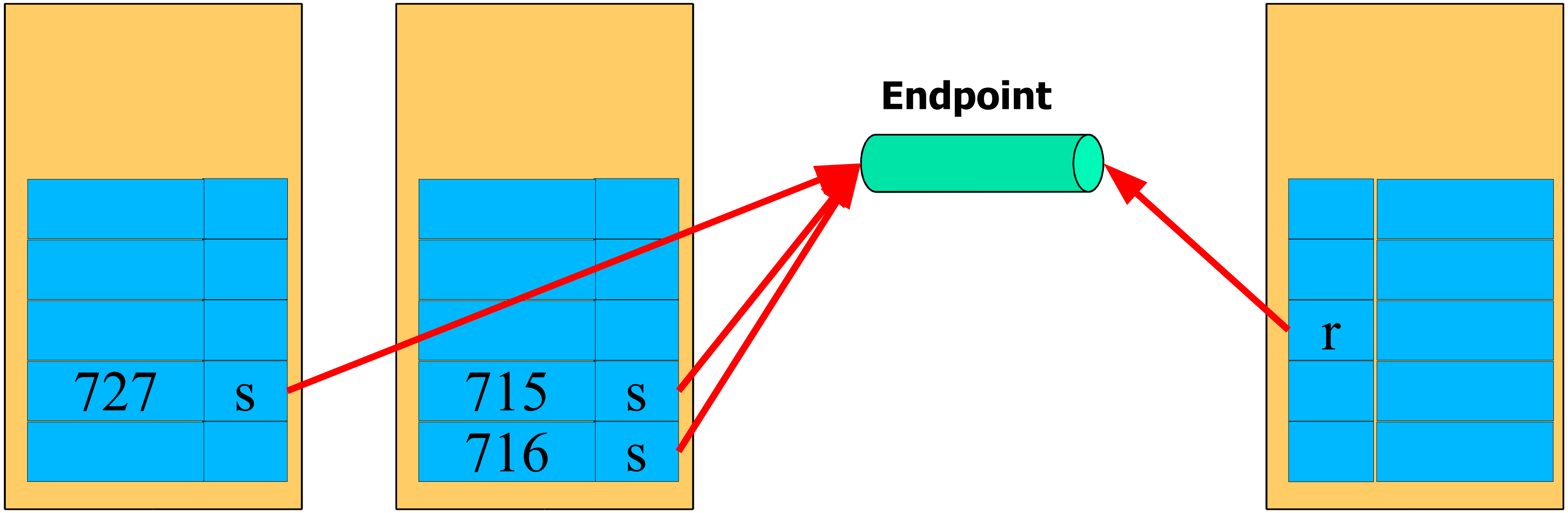
Communication



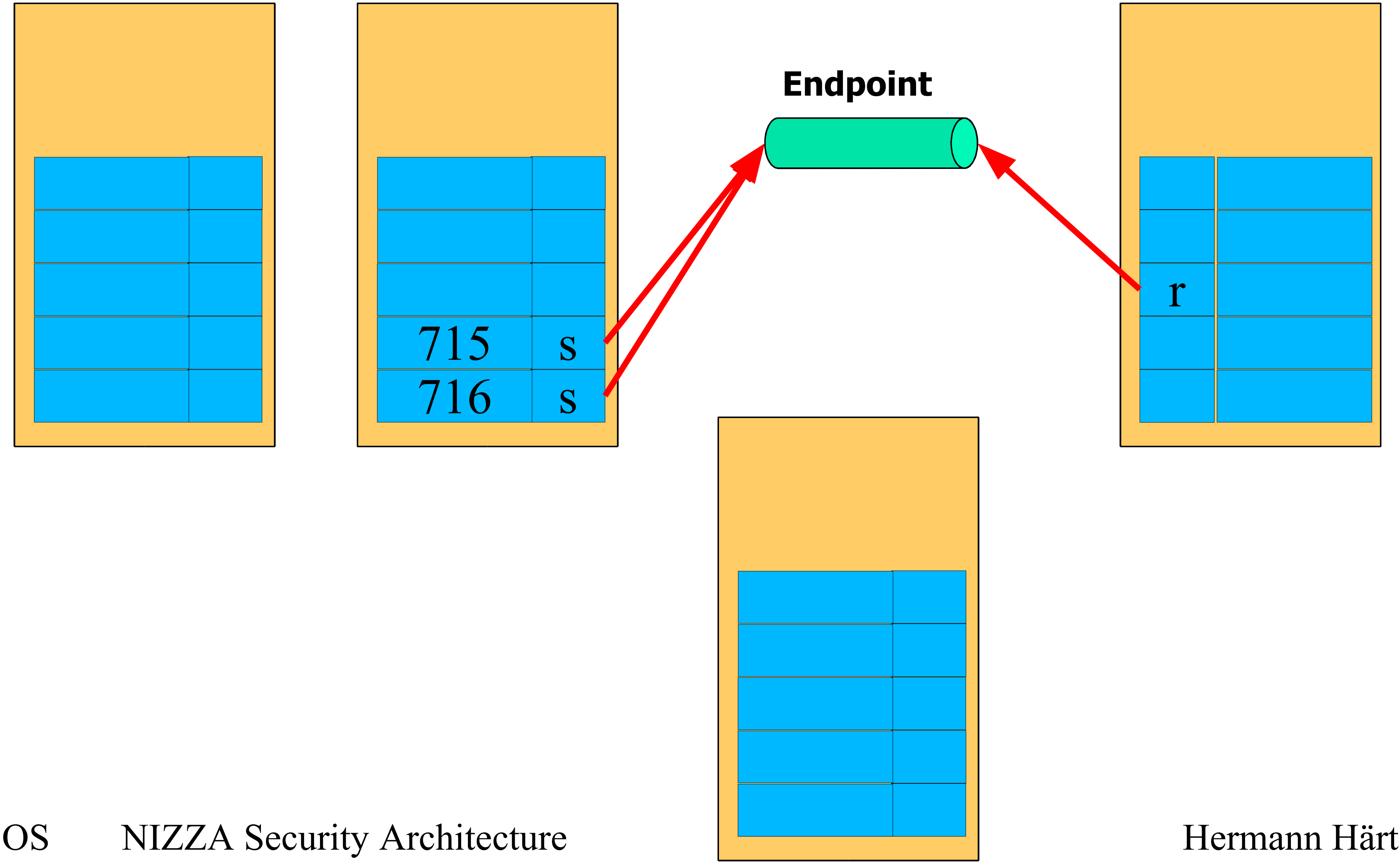
Communication



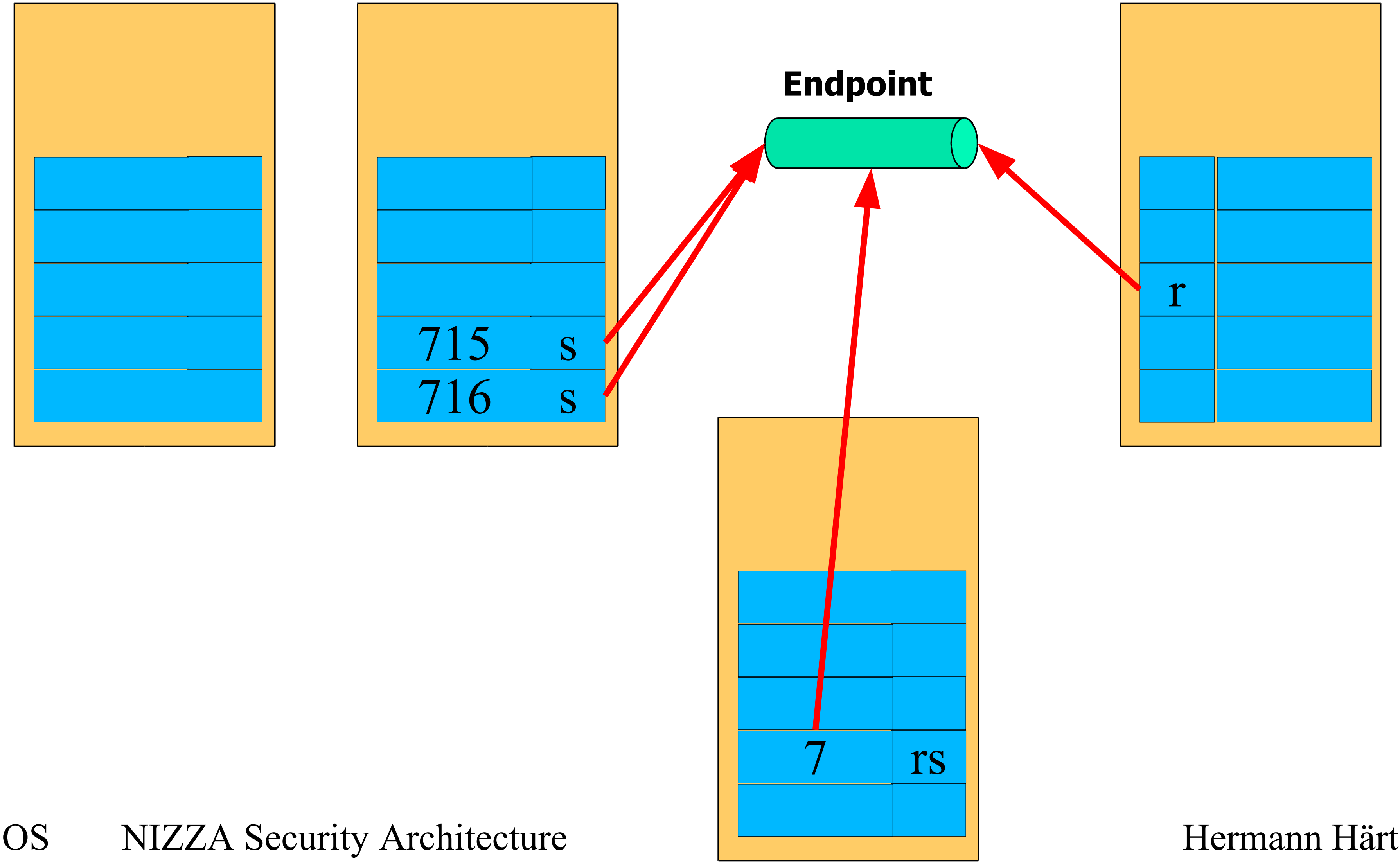
Communication



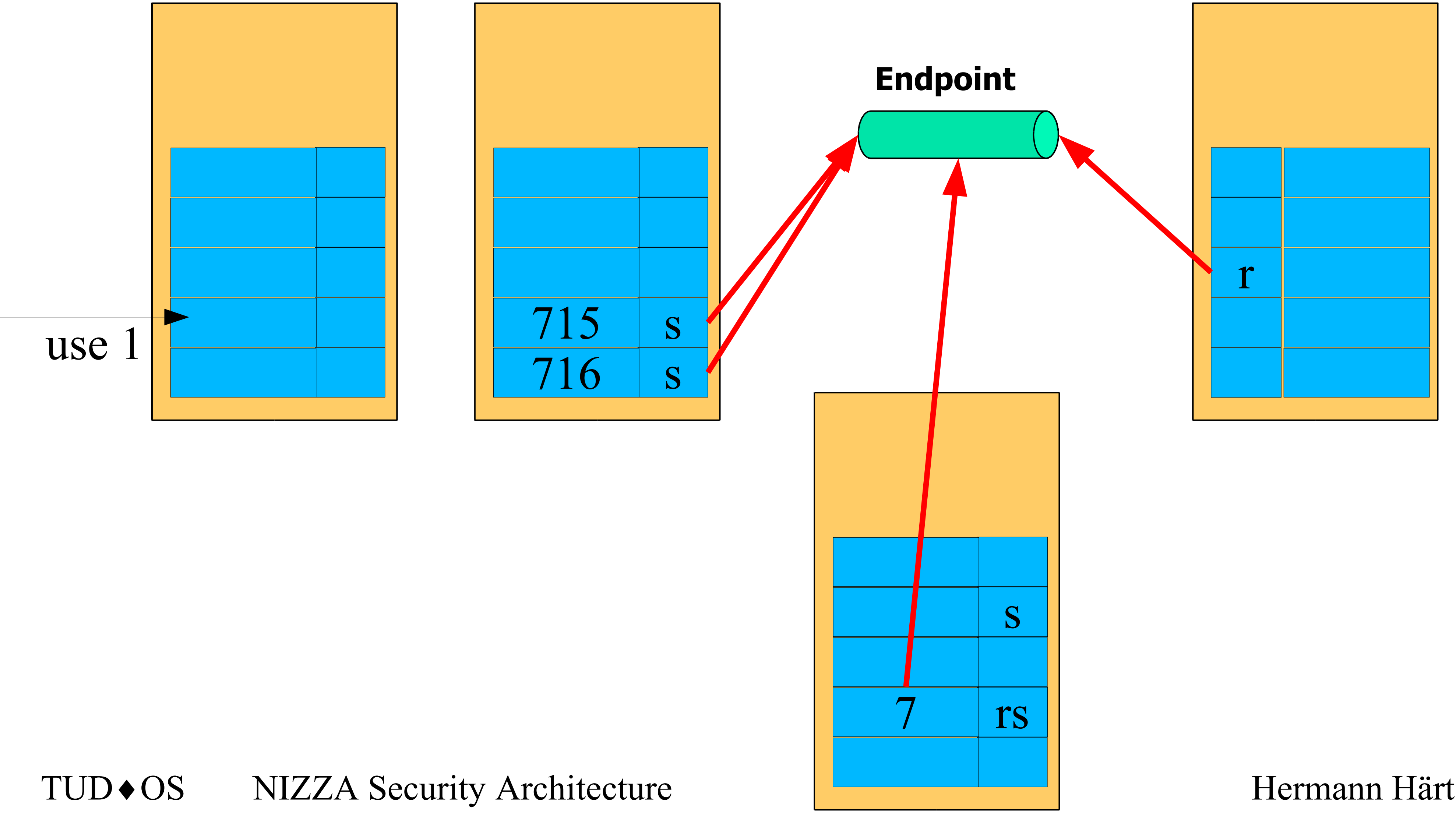
Communication



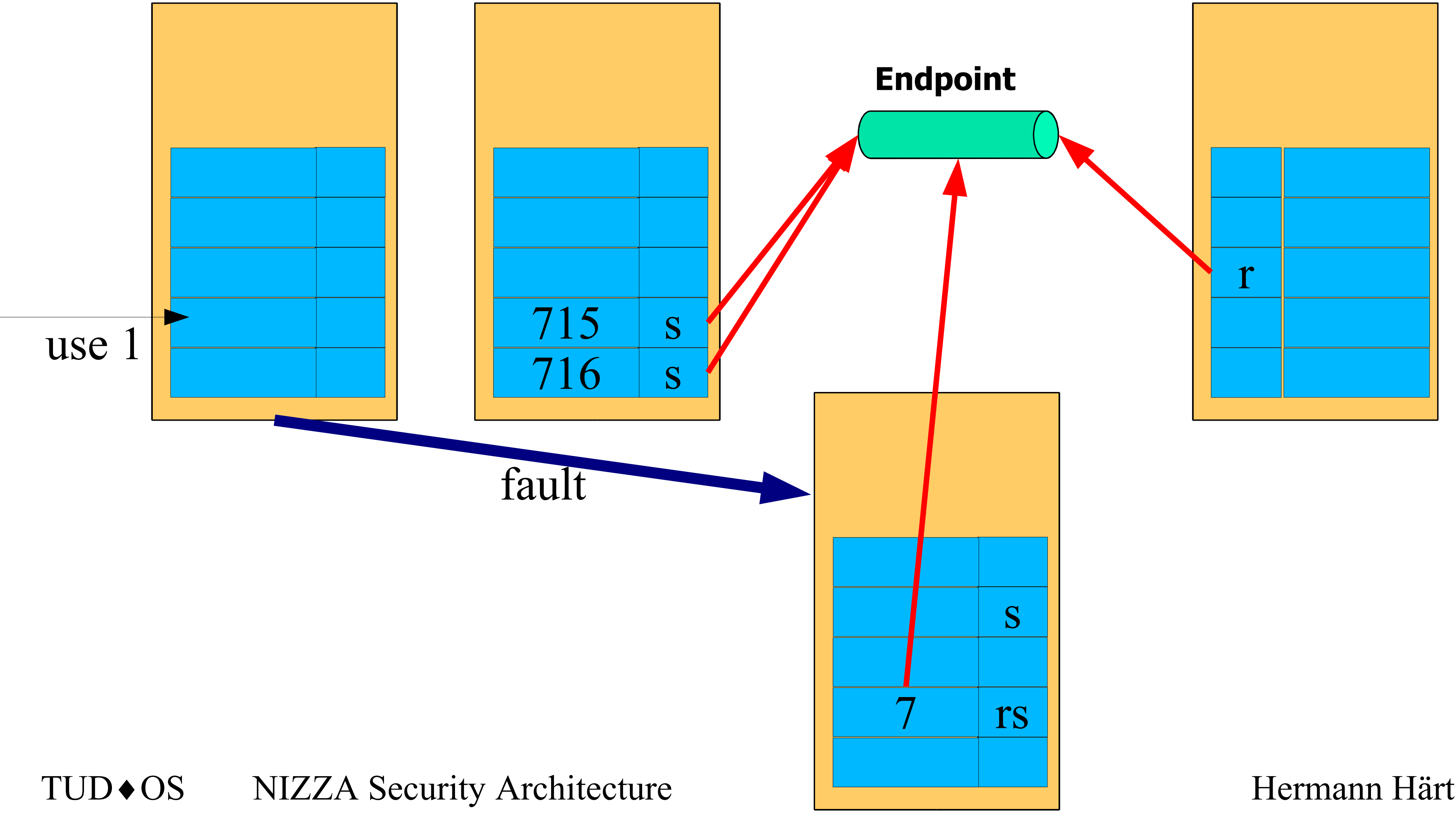
Communication



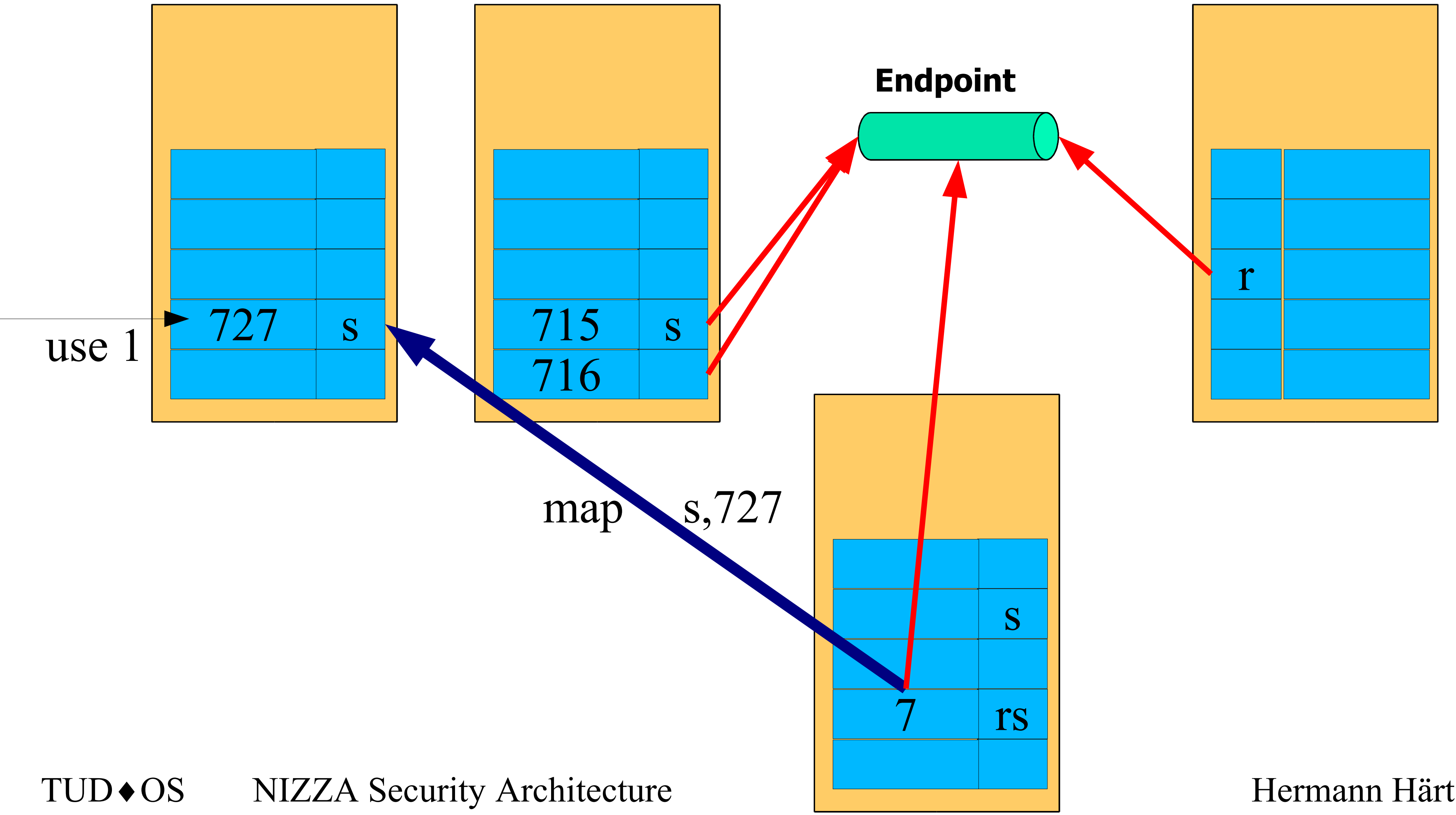
Communication

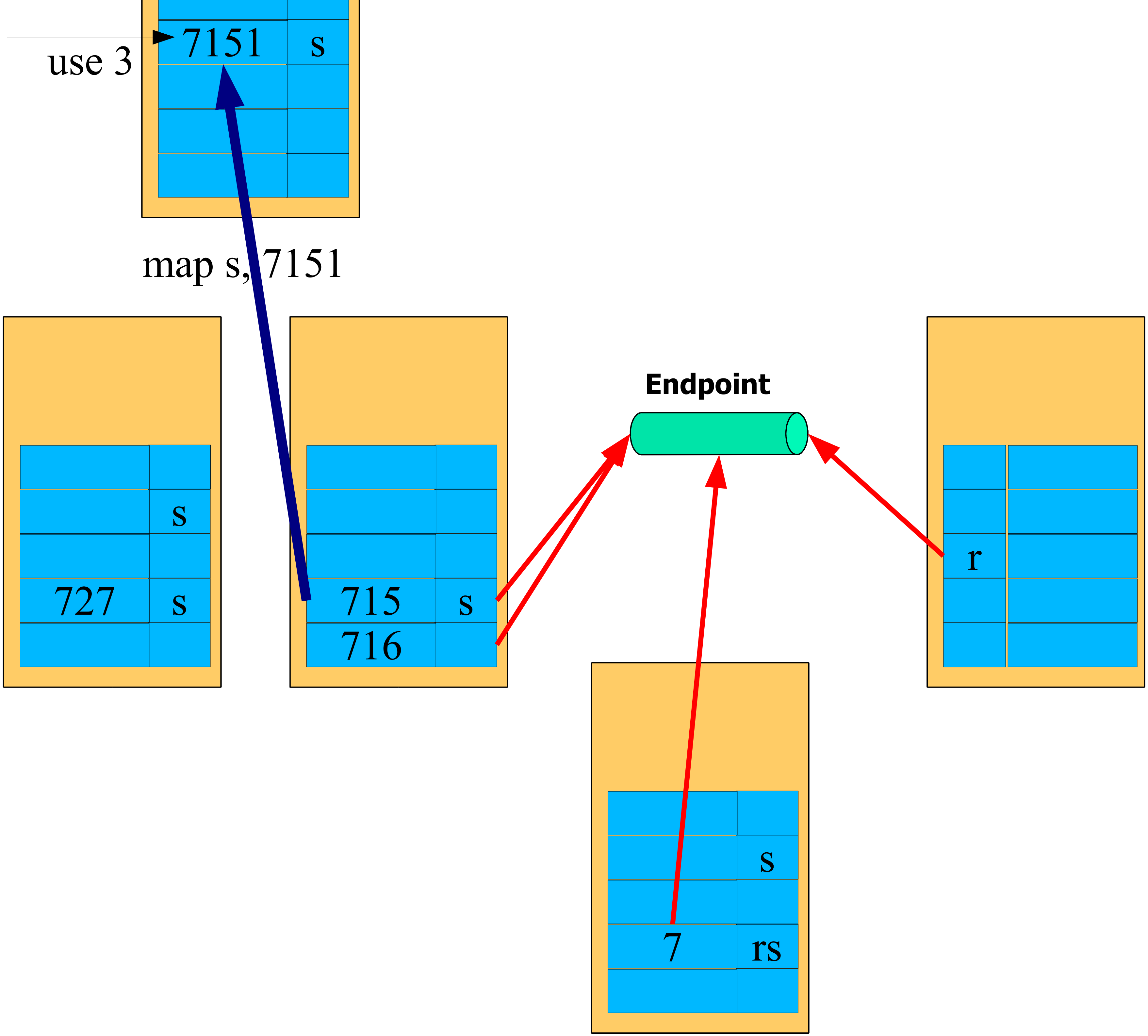


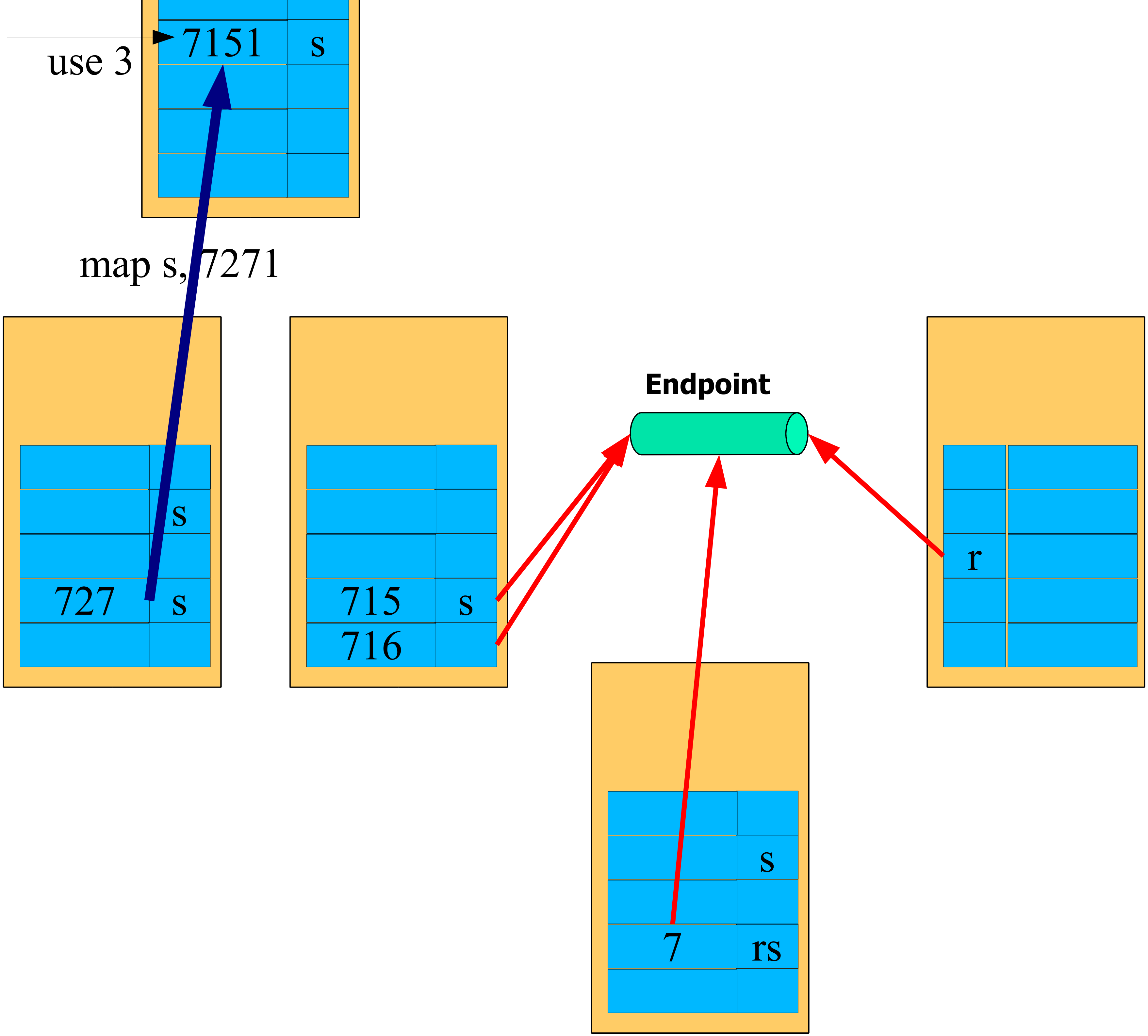
Communication



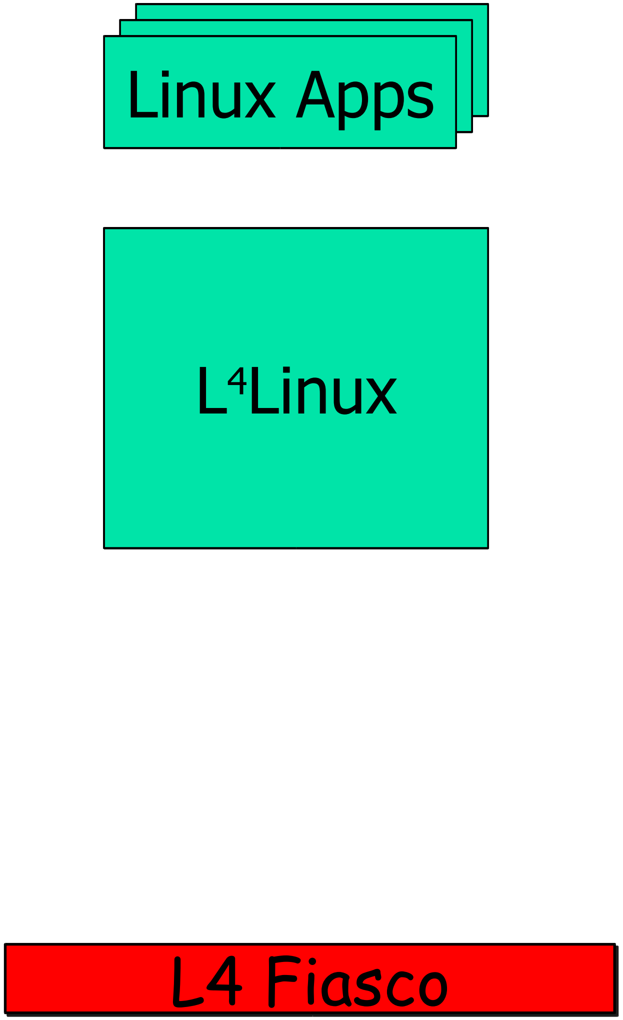
Communication



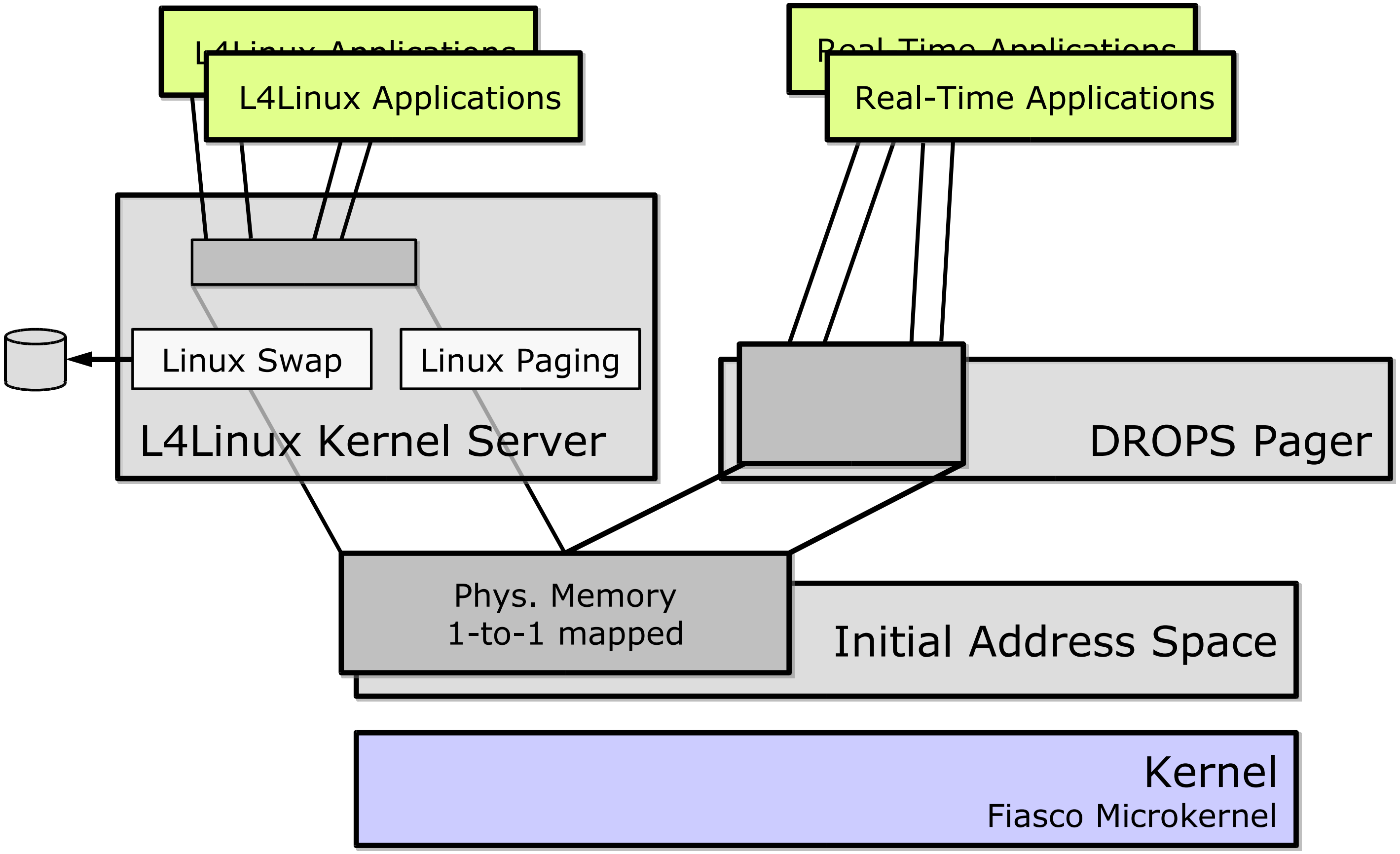

use 3
7151
s
map s, 7151
s
727
s
715
s
716
Endpoint
r
s
7
rs


use 3
7151
s
map s, 7271
s
727
s
715
s
716
Endpoint
r
s
7
rs

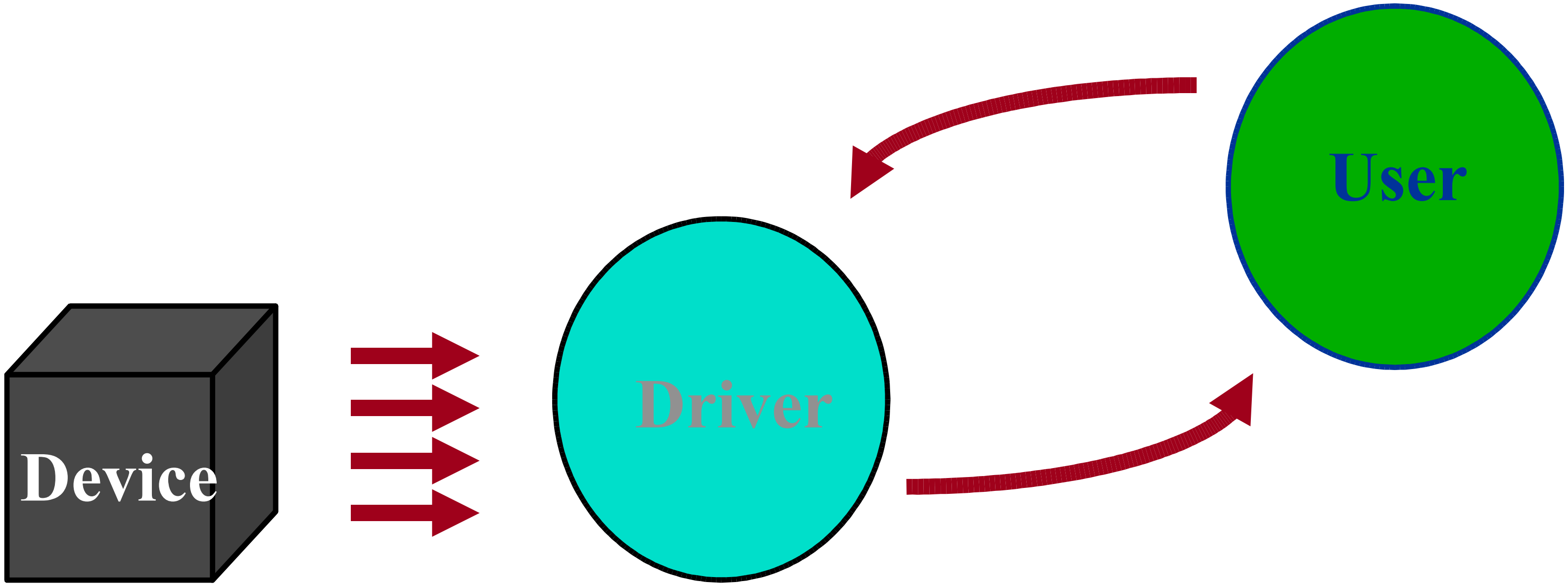
L^4Linux



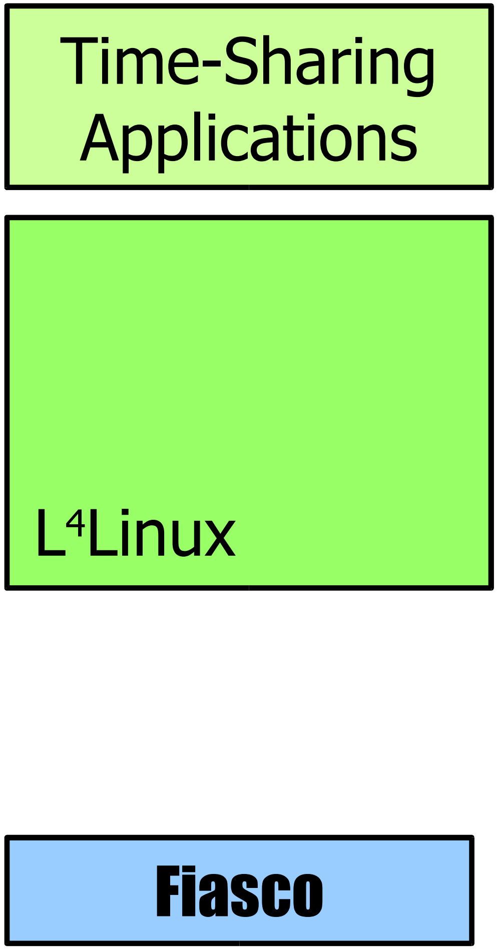
Pagers

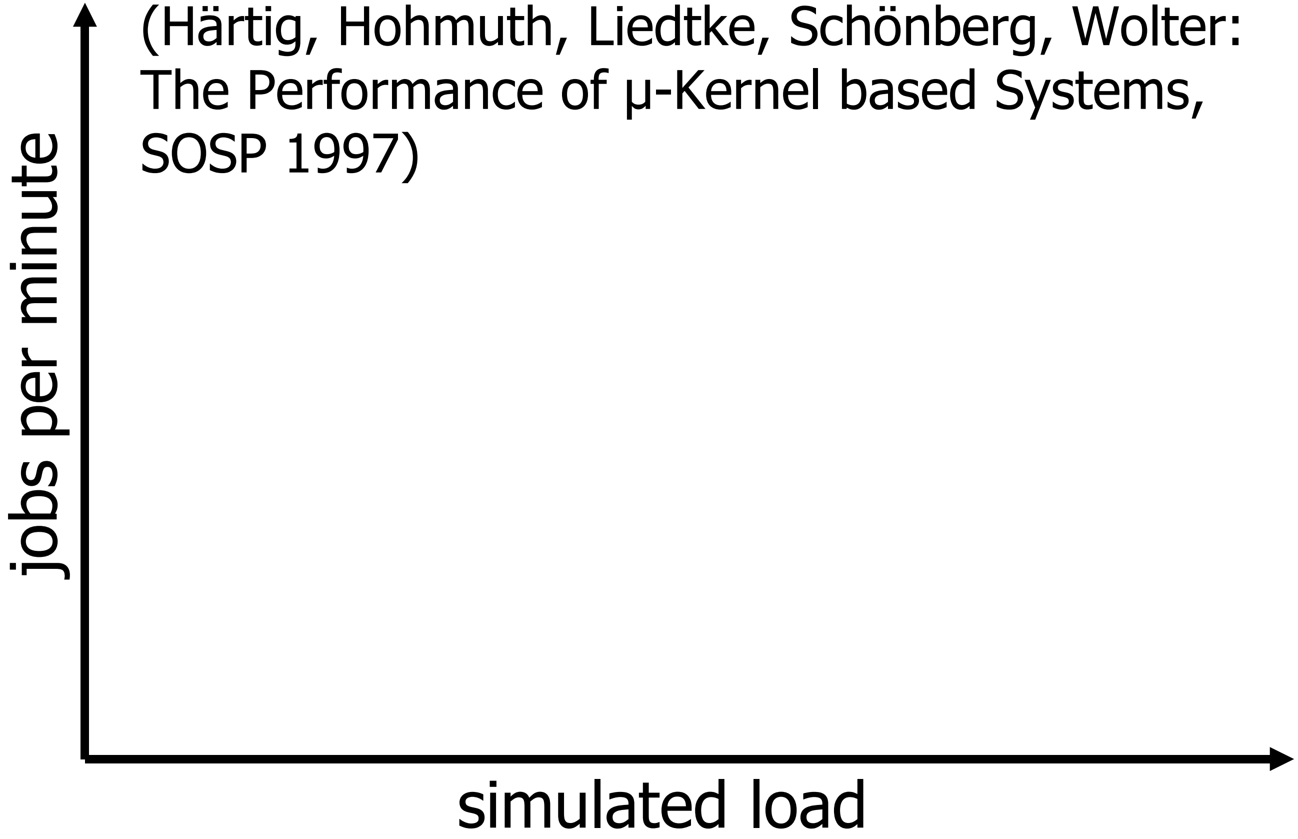

L4Linux Applications
L4Linux Applications
Real-Time Applications
Real-Time Applications
Linux Swap
Linux Paging
L4Linux Kernel Server
DROPS Pager
Phys. Memory
1-to-1 mapped
Initial Address Space
Kernel
Fiasco Microkernel

Linux "top halves" as threads



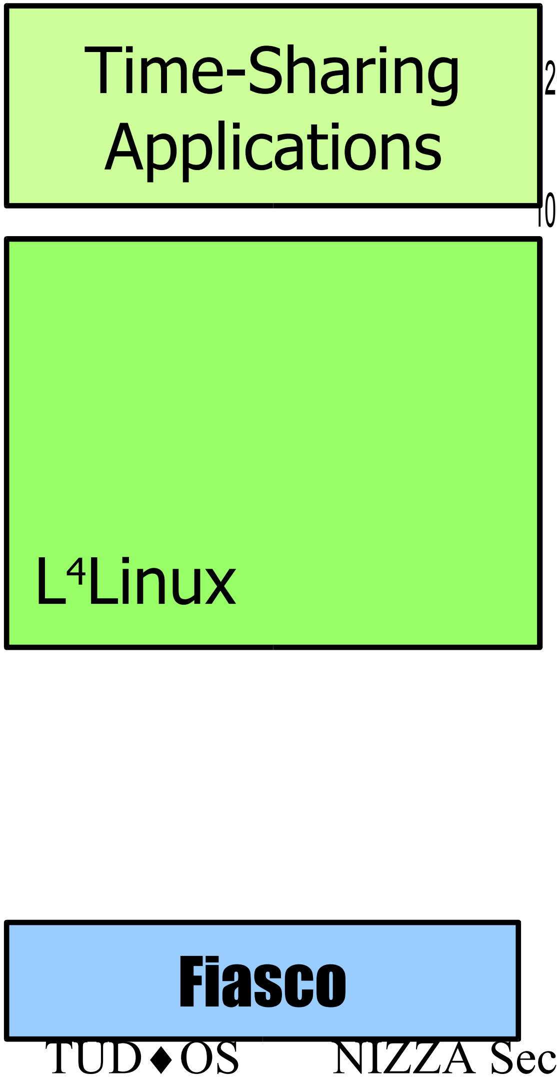
Performance





(Härtig, Hohmuth, Liedtke, Schönberg, Wolter: The Performance of μ-Kernel based Systems, SOSP 1997)

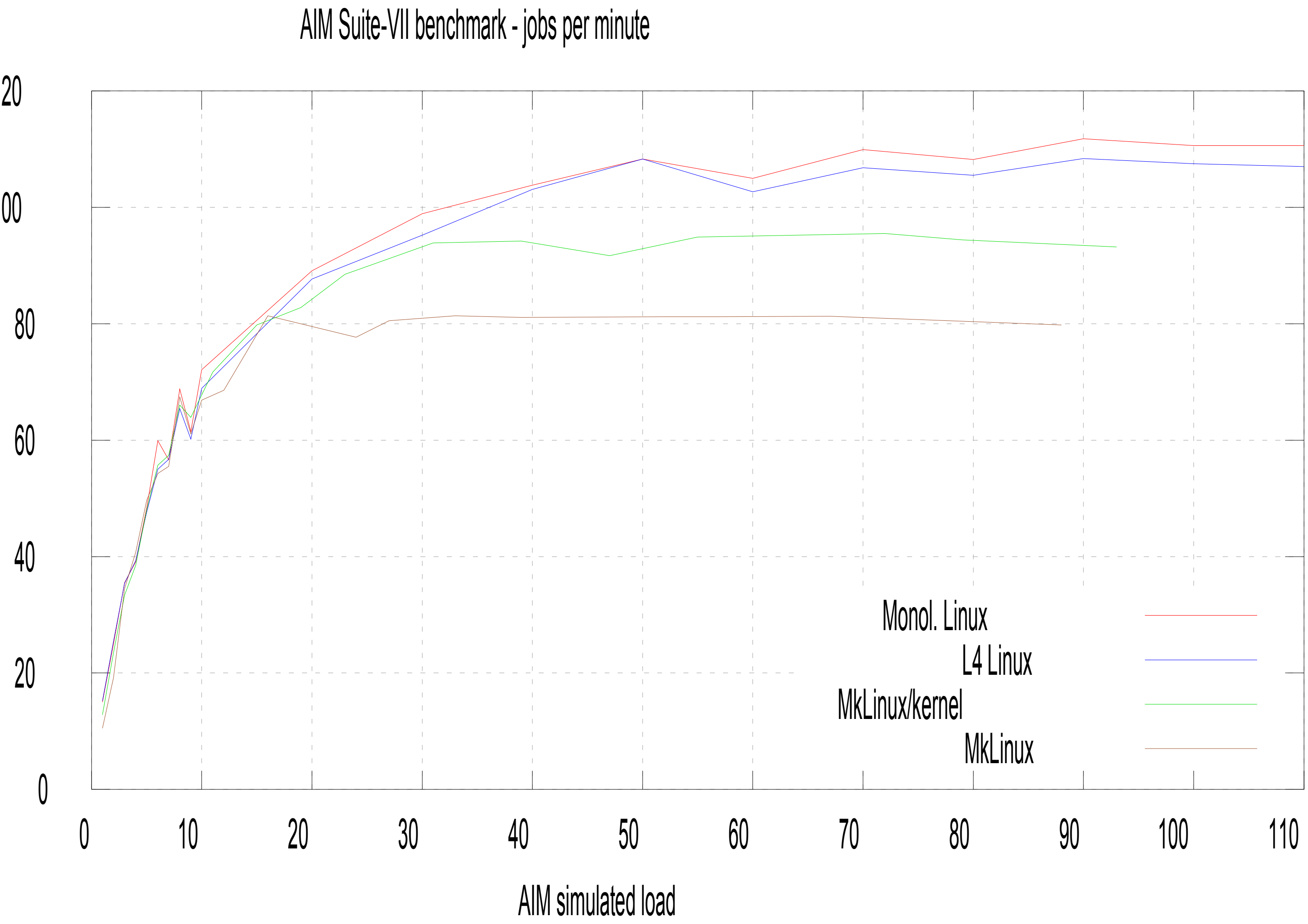
L^{4}Linux compared to MACH



Time-Sharing Applications

L^{4}Linux

Fiasco

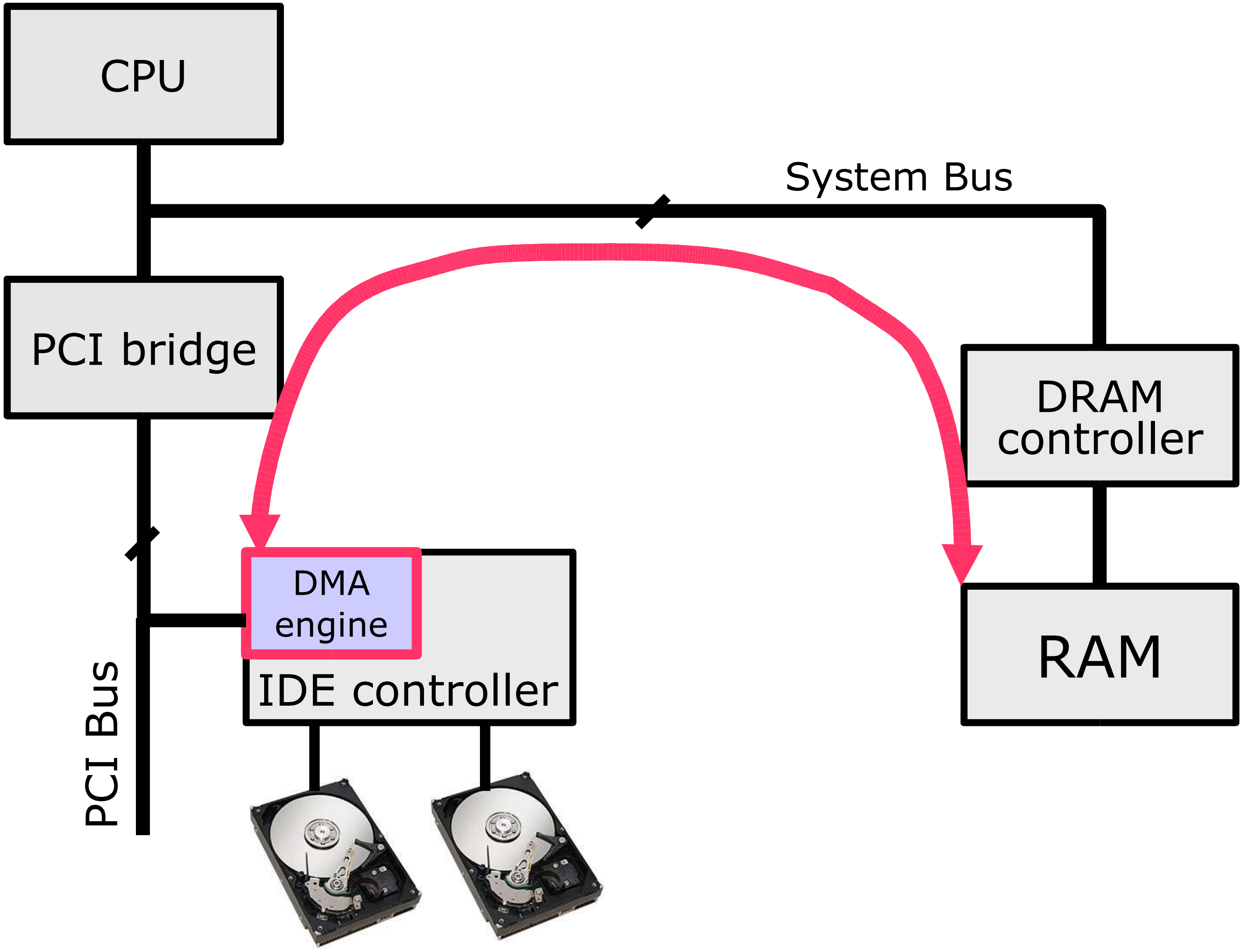


The DMA Problem

- separation is enforced by MMU
- devices access memory using bus master DMA
- DMA uses physical addresses
  (on most architectures)

➔ malicious devices
(or malicious device drivers, firmware!)
can access/modify all components of a system

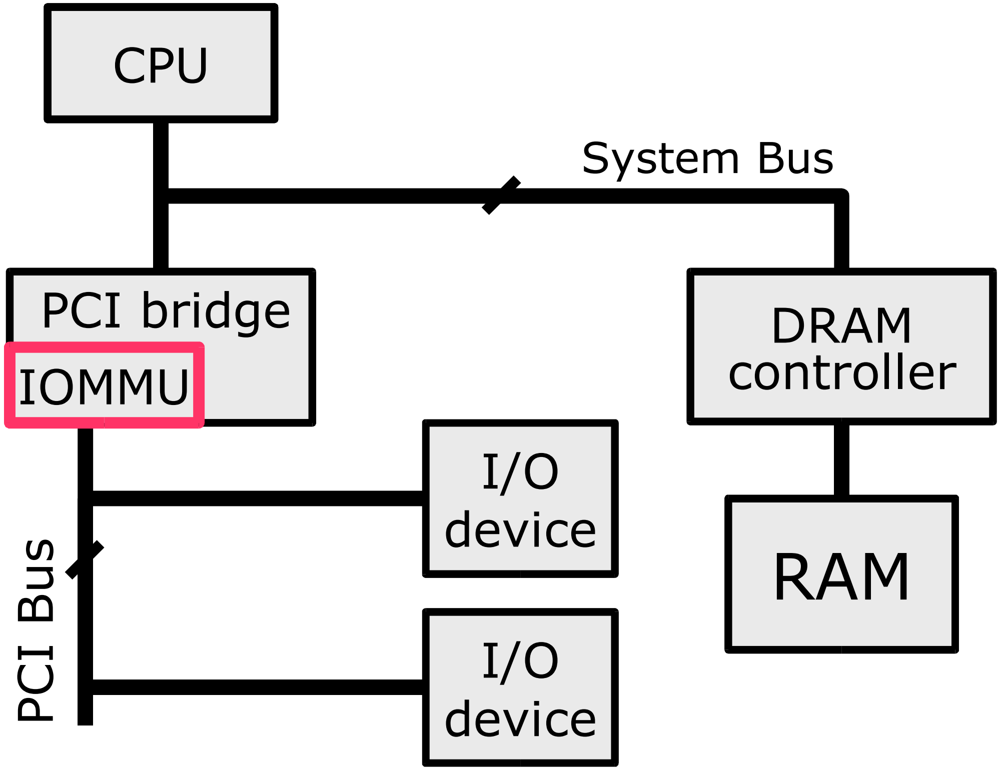
The DMA Problem



IOMMU

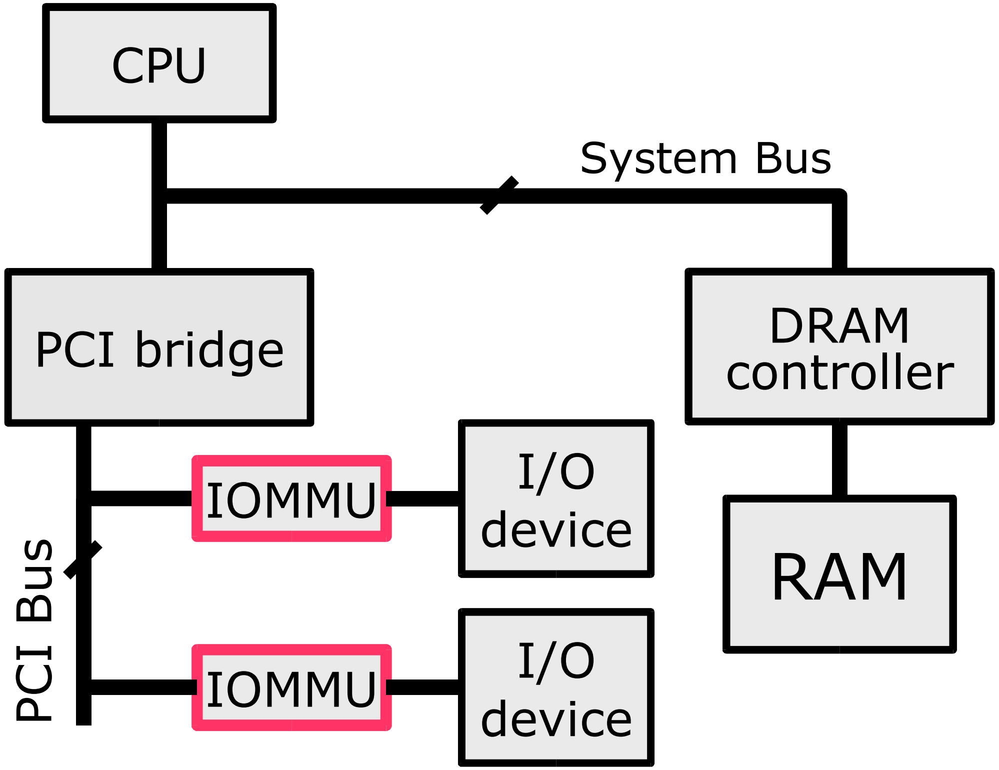
per-bus IOMMU:

- IA64 chipsets
- Opteron



per-device IOMMU:

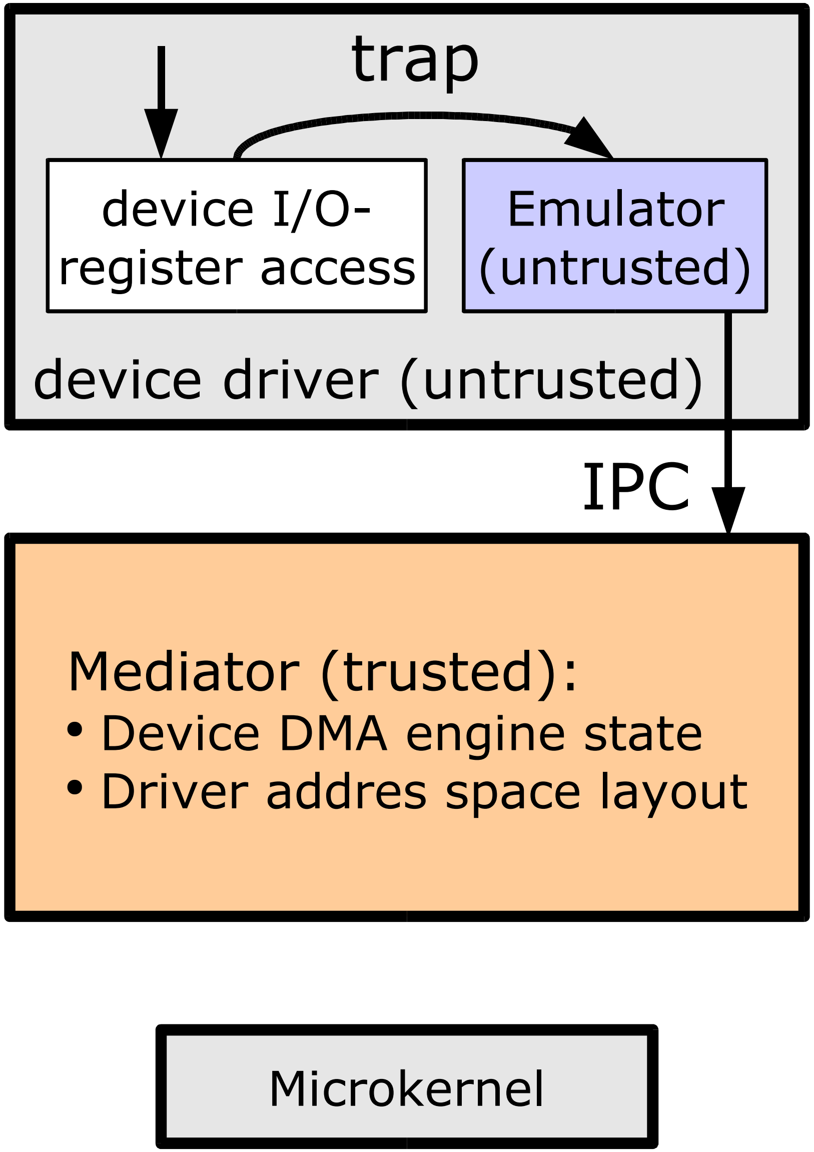
- **n/a** in current hardware!



IOMMU in Software

Make sure that device cannot perform malicious DMA:

- **Trap** read/write access to I/O-registers of the device
- An **emulator** (untrusted) determines size and value to be read/written
- A **mediator** (trusted) checks and performs the access


trap
device I/O-register access
Emulator (untrusted)
device driver (untrusted)
IPC
Mediator (trusted):
• Device DMA engine state
• Driver addres space layout
Microkernel

IOMMU in Software

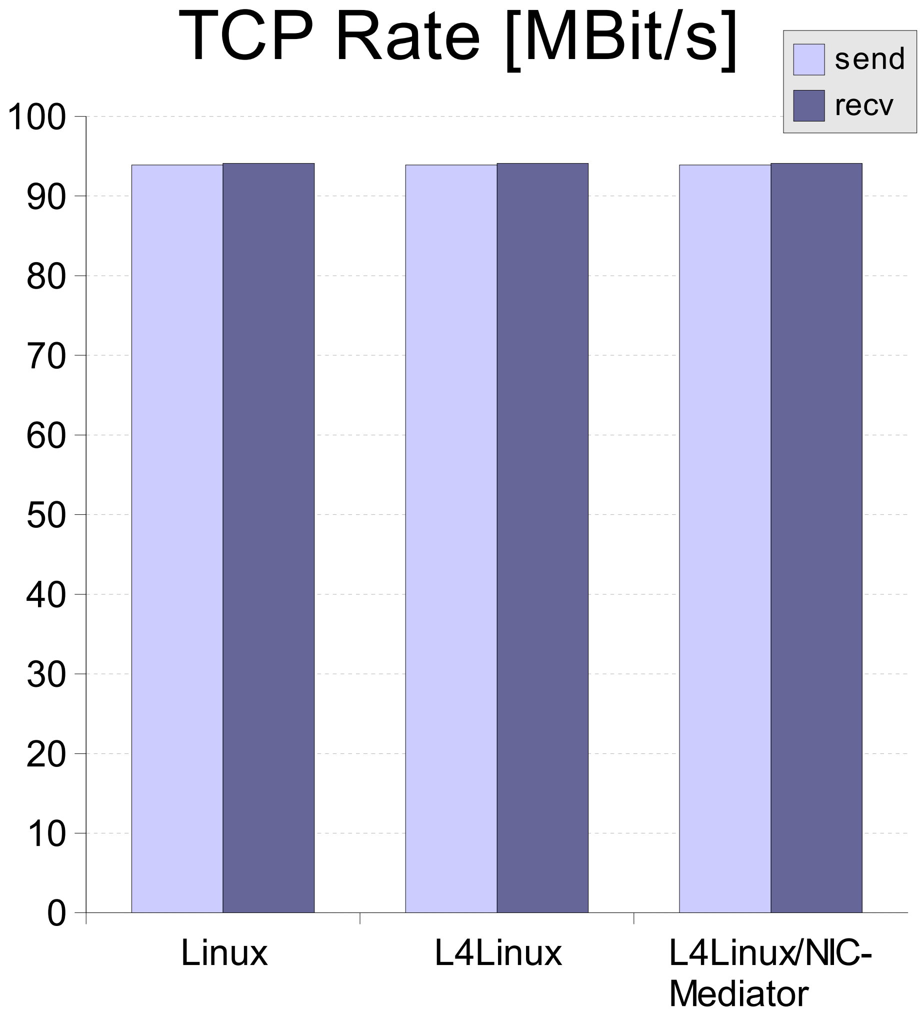
- **Emulator**:
  - in driver's address space ➞ untrusted
  - malfunction does only decrease availability of device
  - ~ 500 LoC
- **Mediator**:
  - trusted, own address space
  - specific for a device or a class of devices
  - ~ 300 LoC

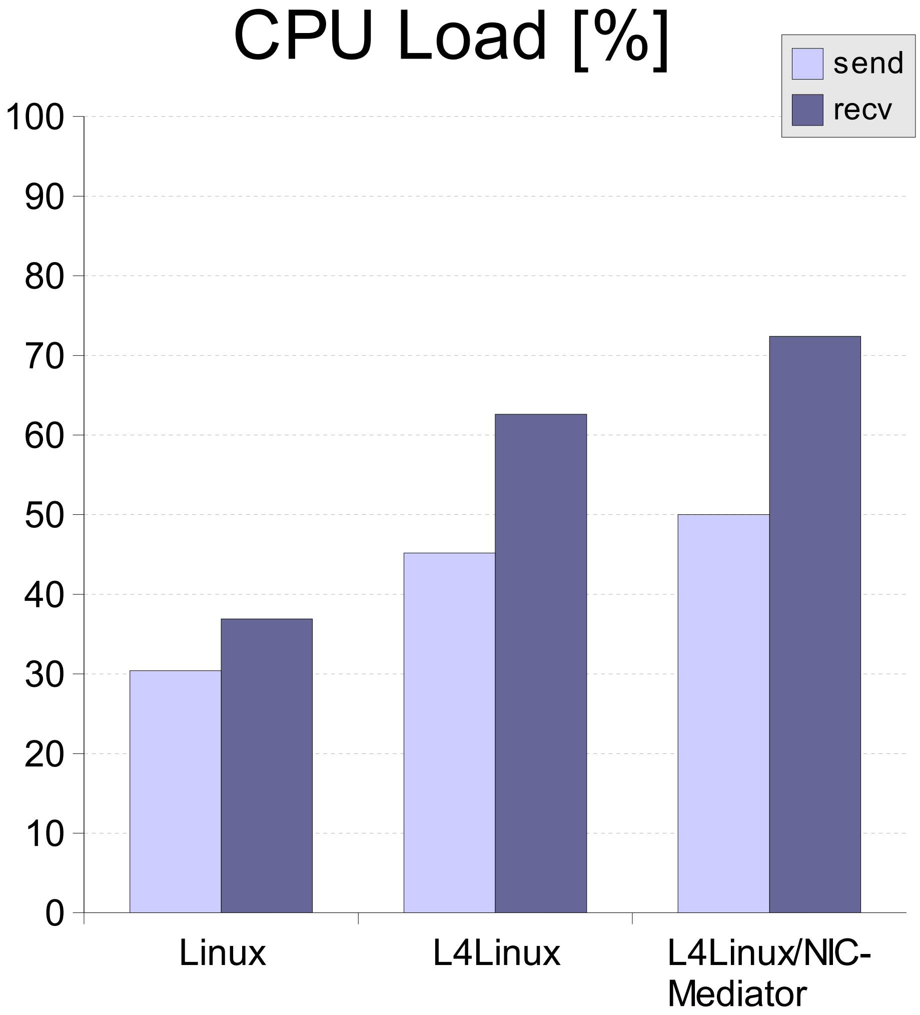
IOMMU in Software

- Implemented for:
  - Fast Ethernet card (DEC Tulip 21143)
  - ATA Controllers
- Does not work for firmware-programmable devices:
  - private interface between device driver and firmware
  - no protection mechanism between firmware and device

IOMMU in Software: NIC Performance

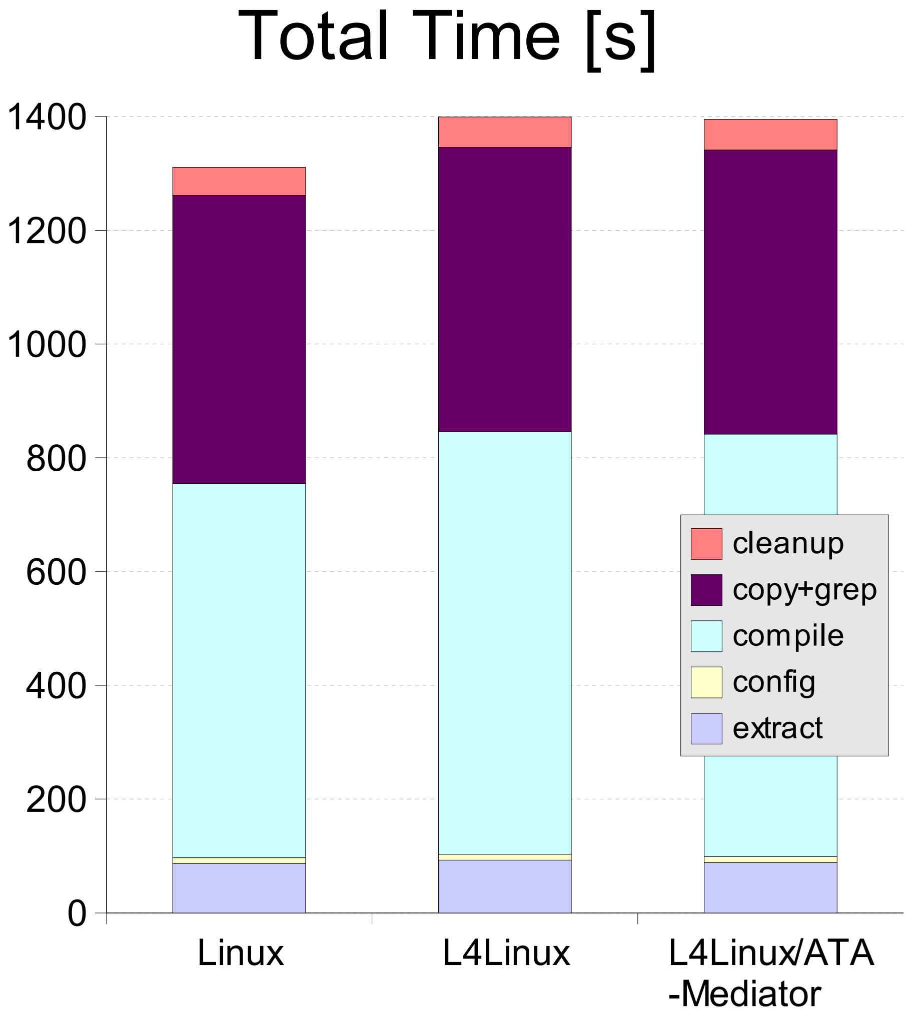
Pentium-III 800 MHz, Fast Ethernet

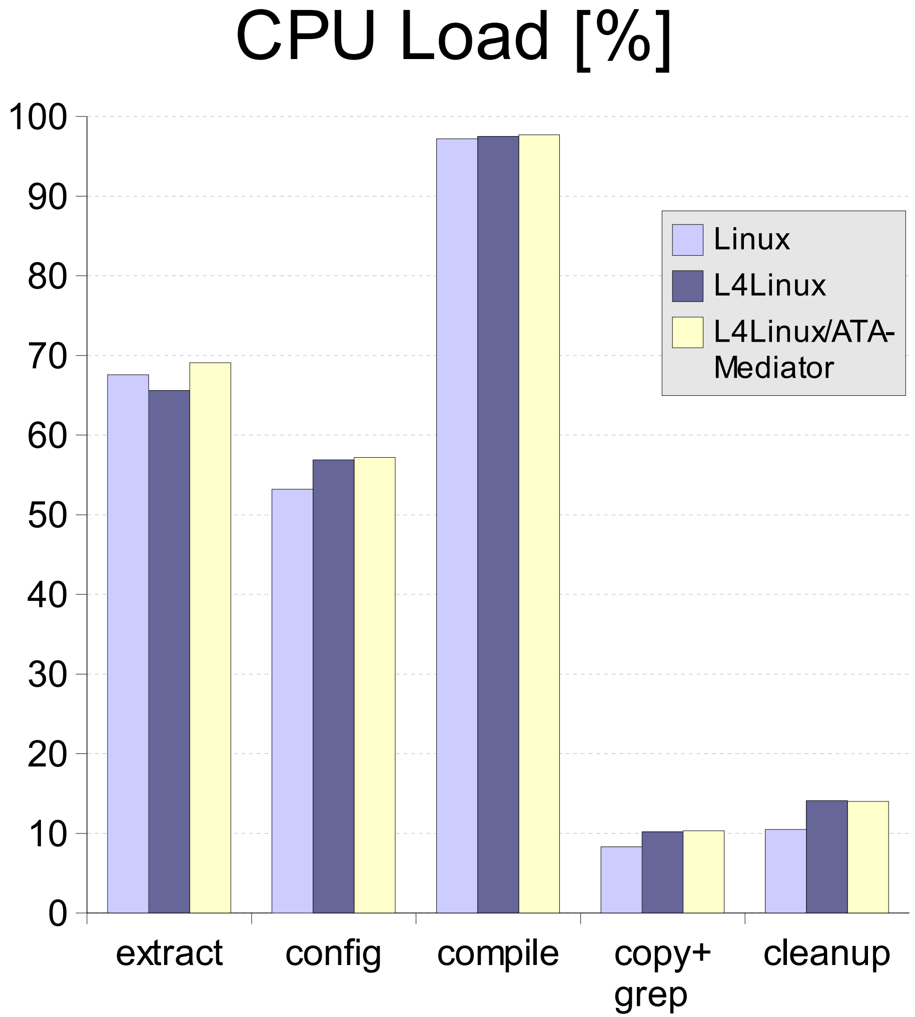

TCP Rate [MBit/s]


CPU Load [%]

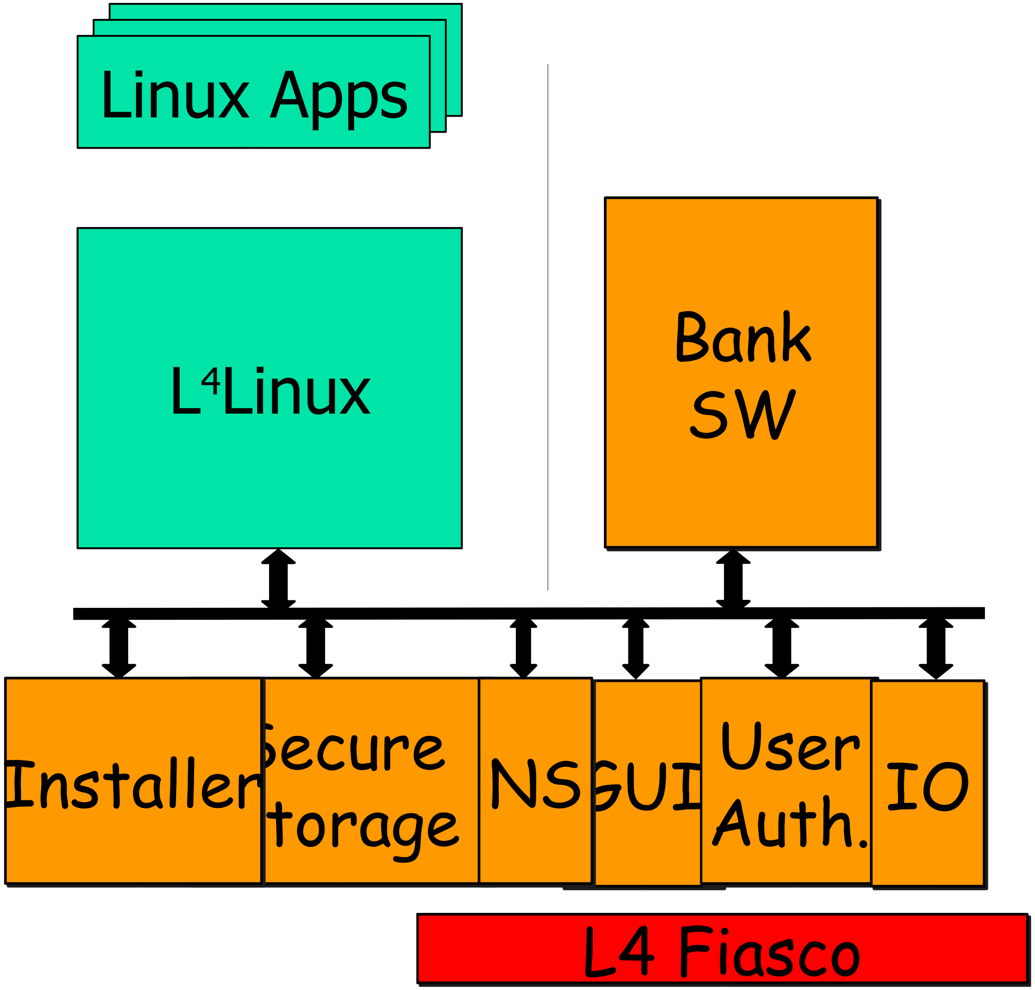
IOMMU in Software: ATA Performance

Pentium-III 800 MHz, VIA82C586 ATA Controller





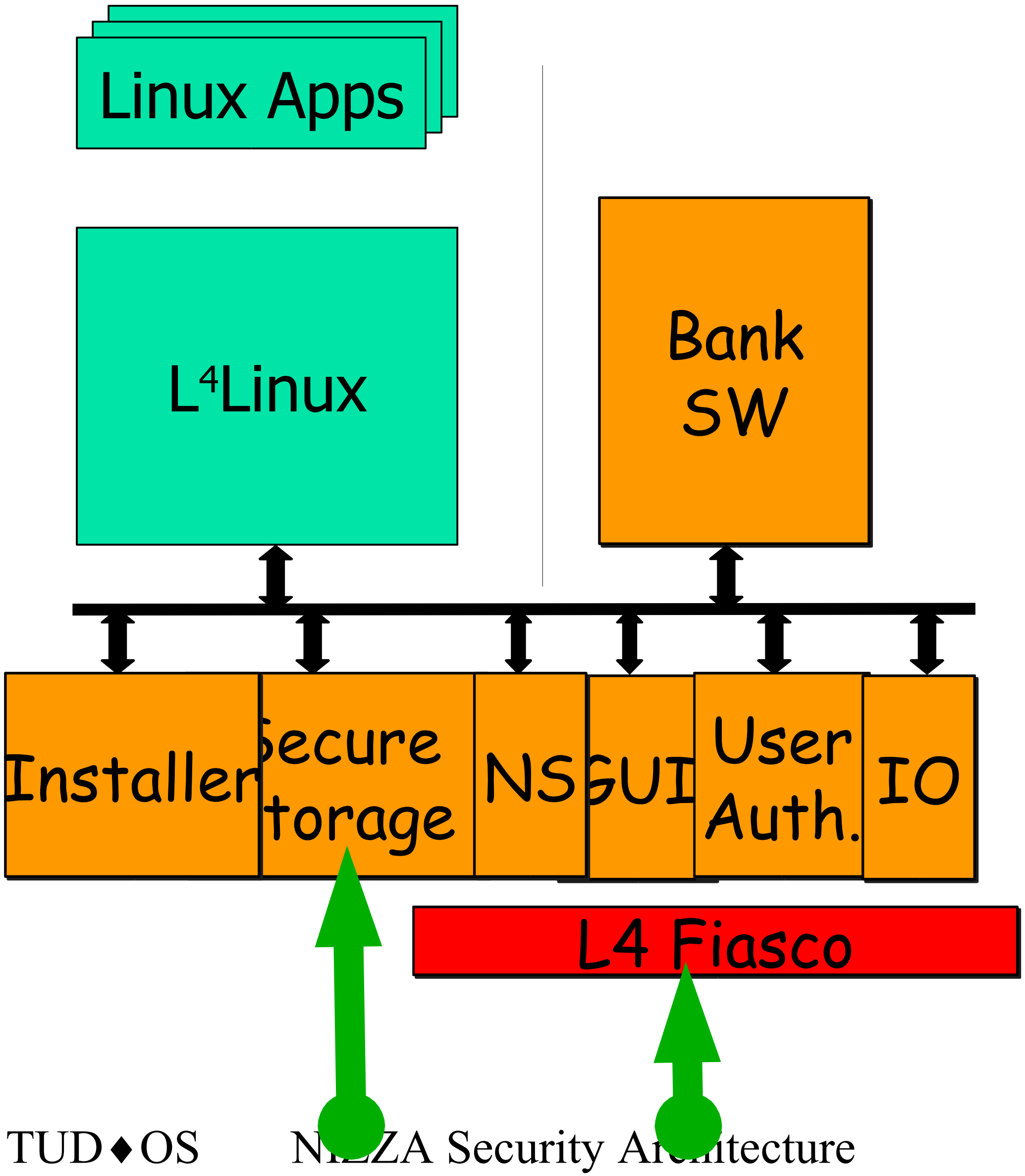
Secure Booting, Remote Attestation and Trusted Path



Authentic application/system:

- how does the remote bank know ?
- how does the local client/user know ?

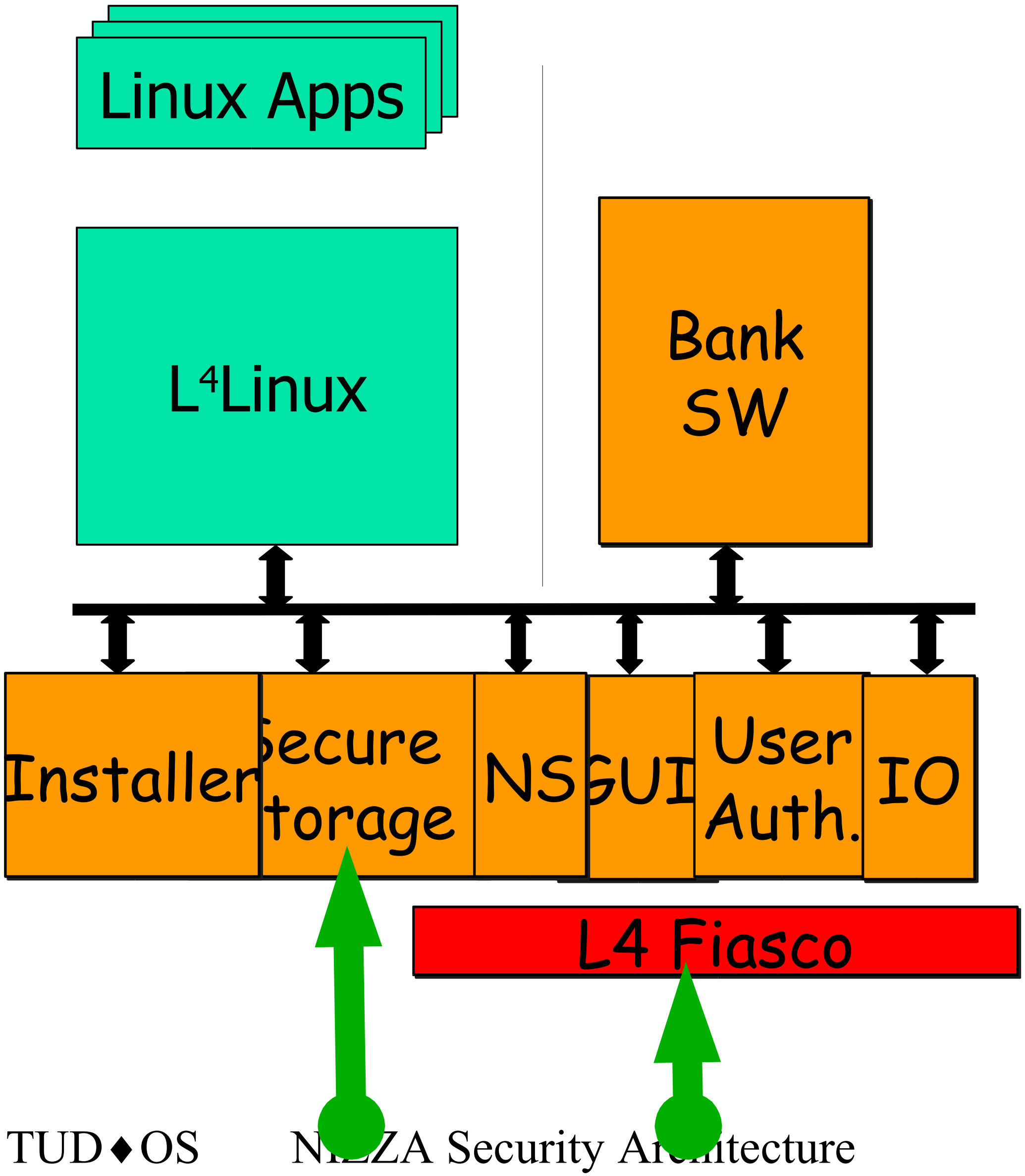
Secure Booting, Remote Attestation and Trusted Path



Authentic application/system:

- how does the remote bank know ?
- how does the local client/user know ?

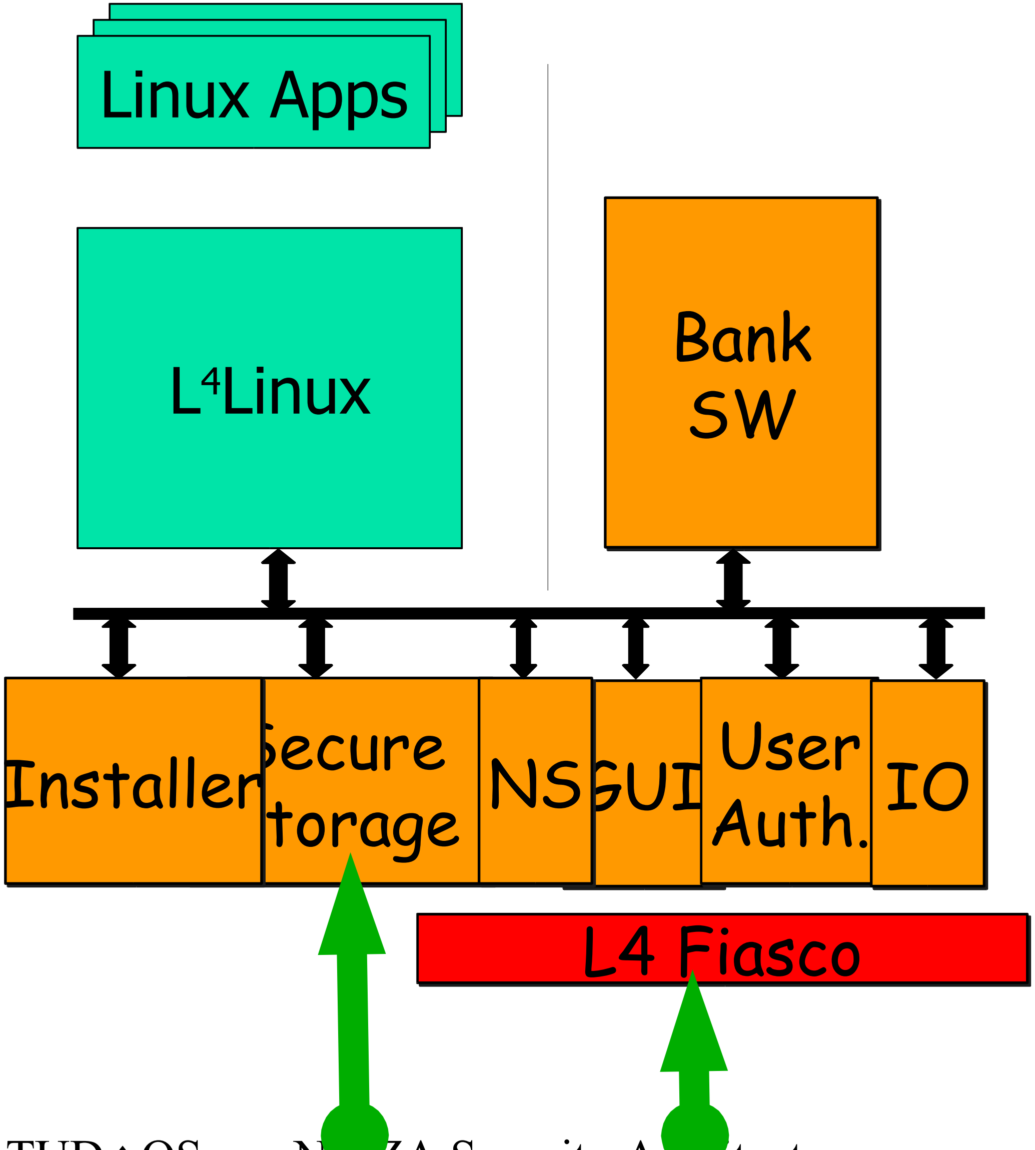
Secure Booting, Remote Attestation and Trusted Path



Authentic application/system:

- how does the remote bank know ?
  - attestation protocol up to Nizza trusted platform
  - mediate other communication thru trusted installer

Secure Booting, Remote Attestation and Trusted Path



Authentic application/system:

- how does the local client/user know ?
  - attestation protocol up to Nizza trusted platform
  - indicate “red/green”
  - handover to DOpE

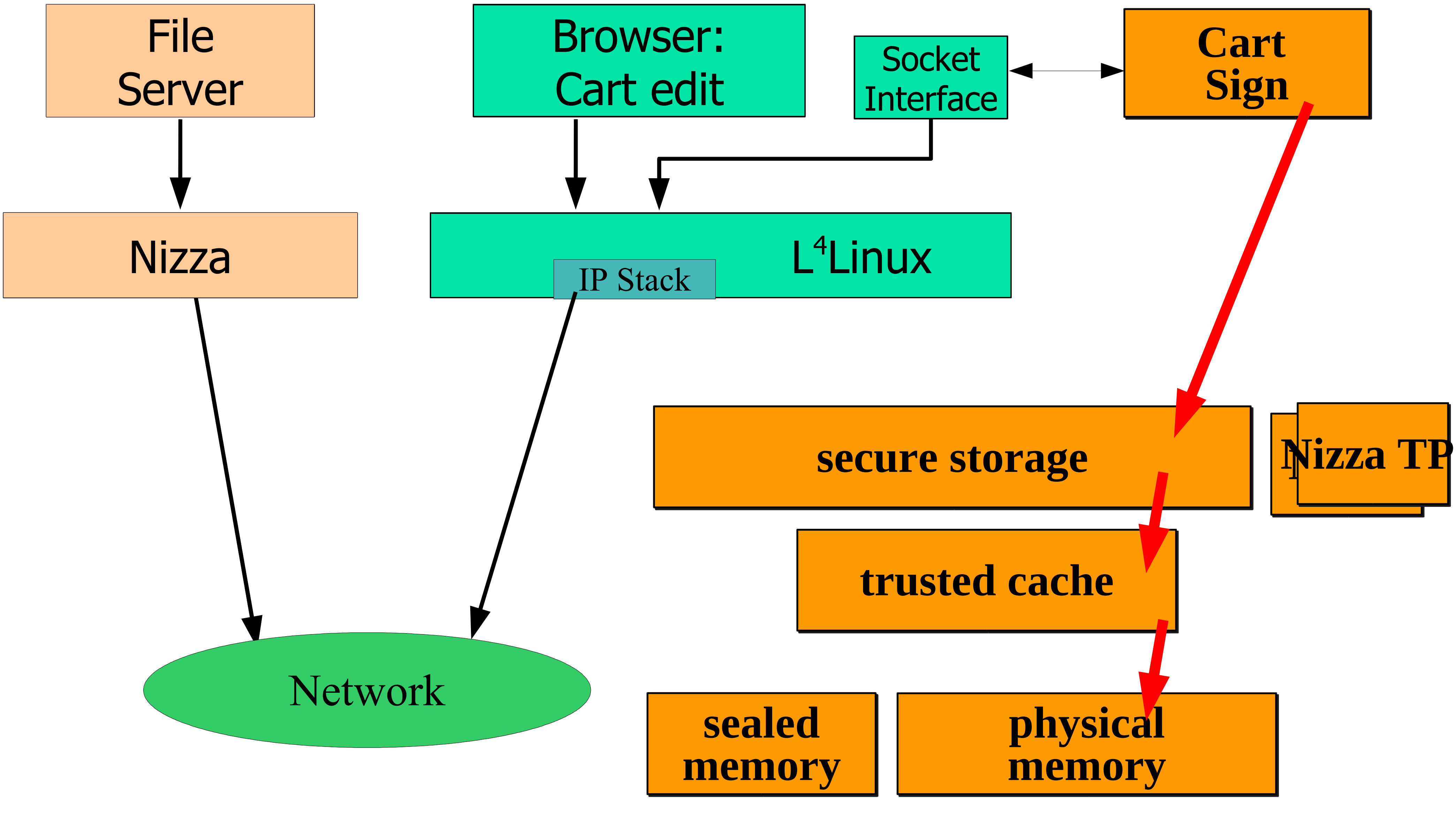
Secure Storage with small TCB (future)

- objectives:
  - security
    - confidentiality, integrity,
    - recoverability
    - availability
  - system security
    - small TCB
    - attacks:
      - theft/loss of device
      - full penetration of L4Linux
-

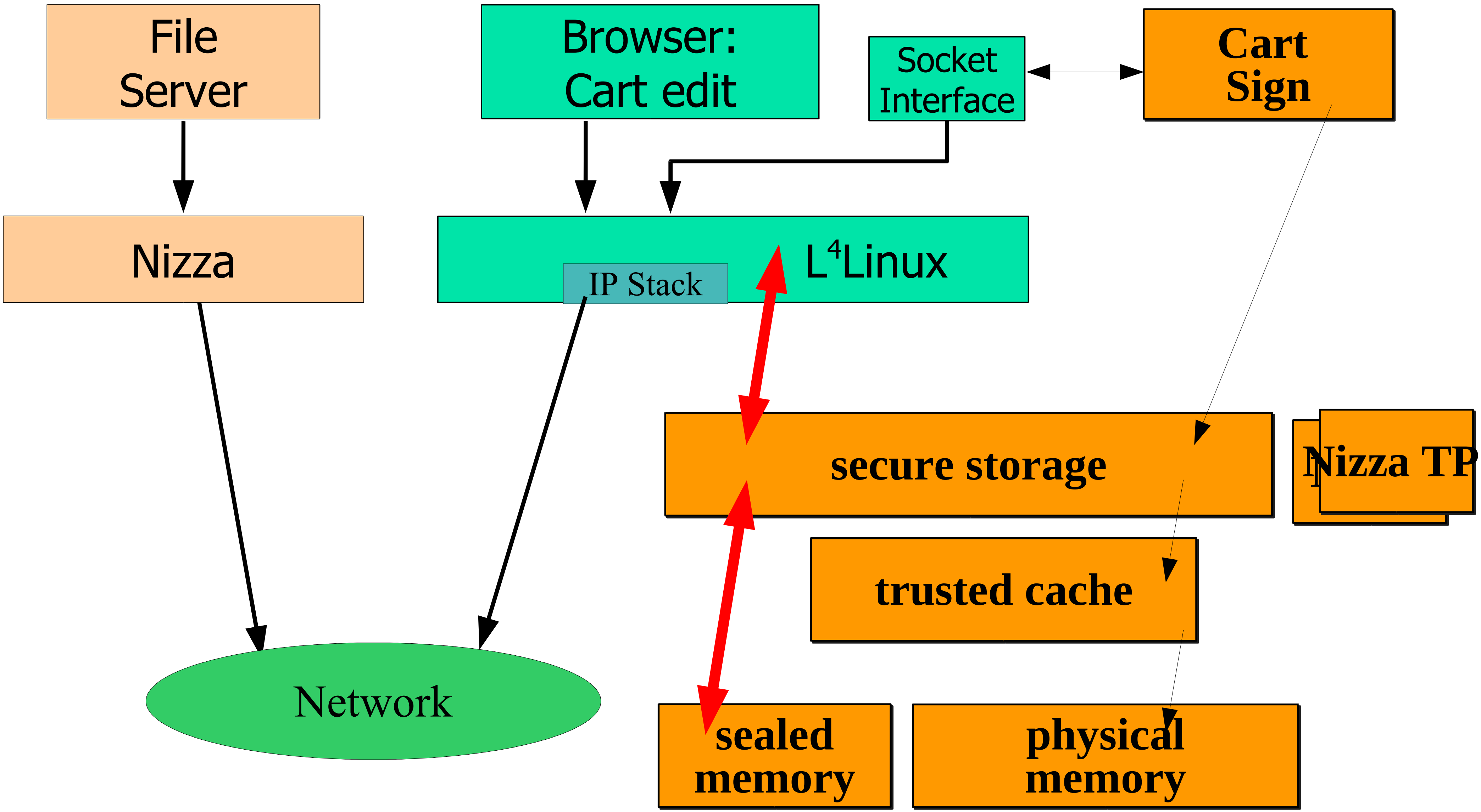
Secure Storage with small TCB (future)

- techniques:
  - use Sealed Memory as key storage
  - reuse L4Linux file system as mass storage
  - use trusted file server for recoverability
  - use resource allocation for availability

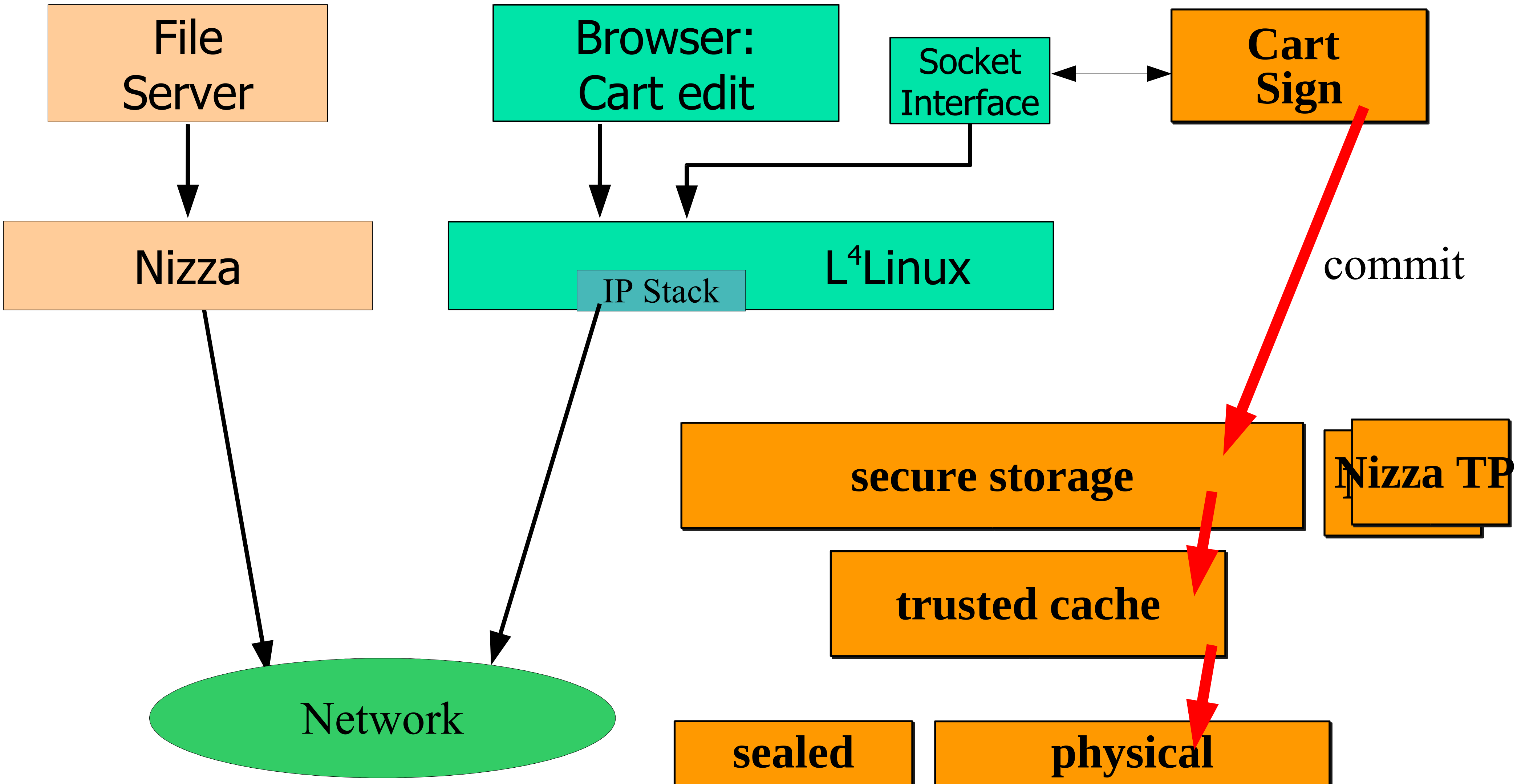
Confidentiality and Integrity



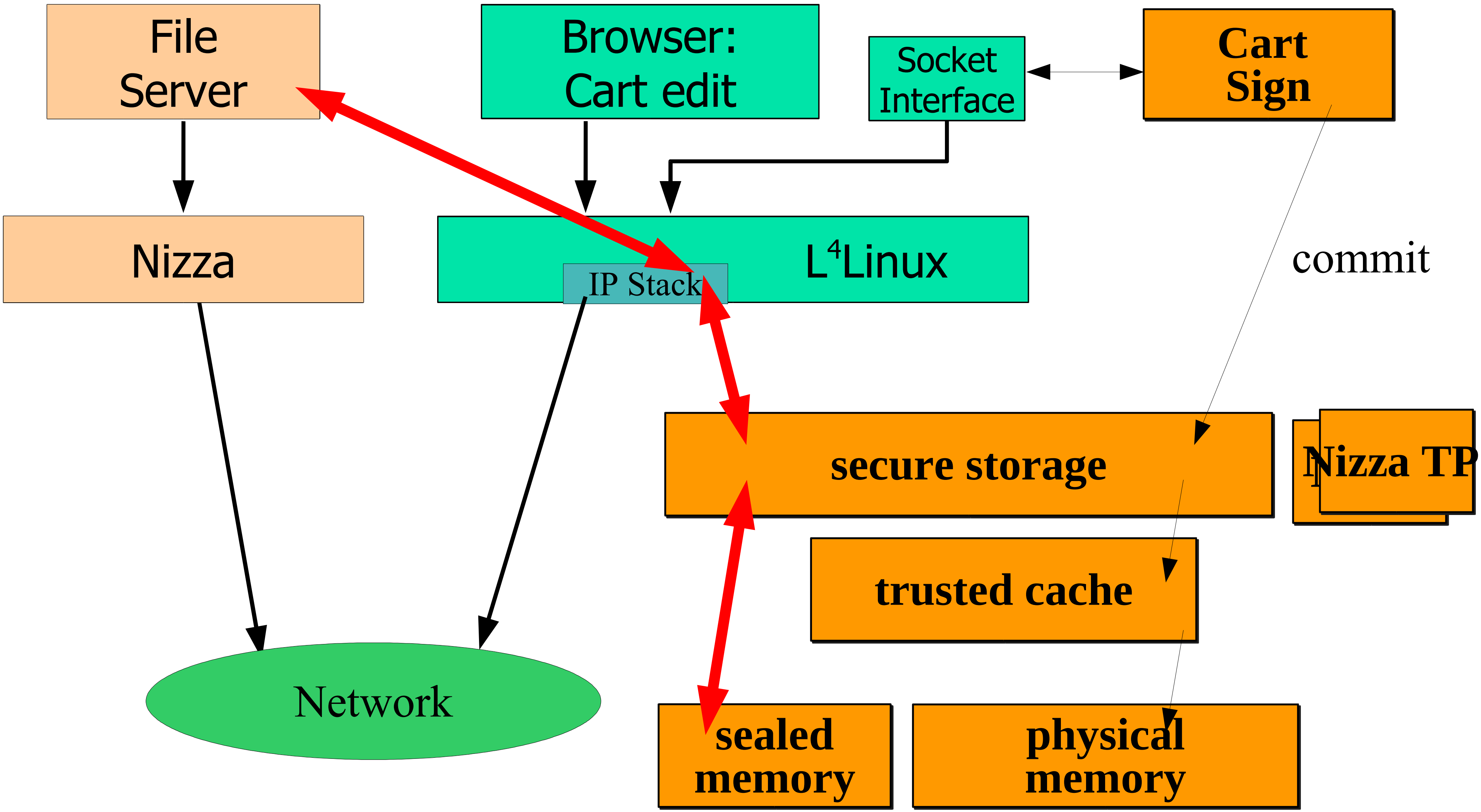
Mass Storage (File System) not part of TCB



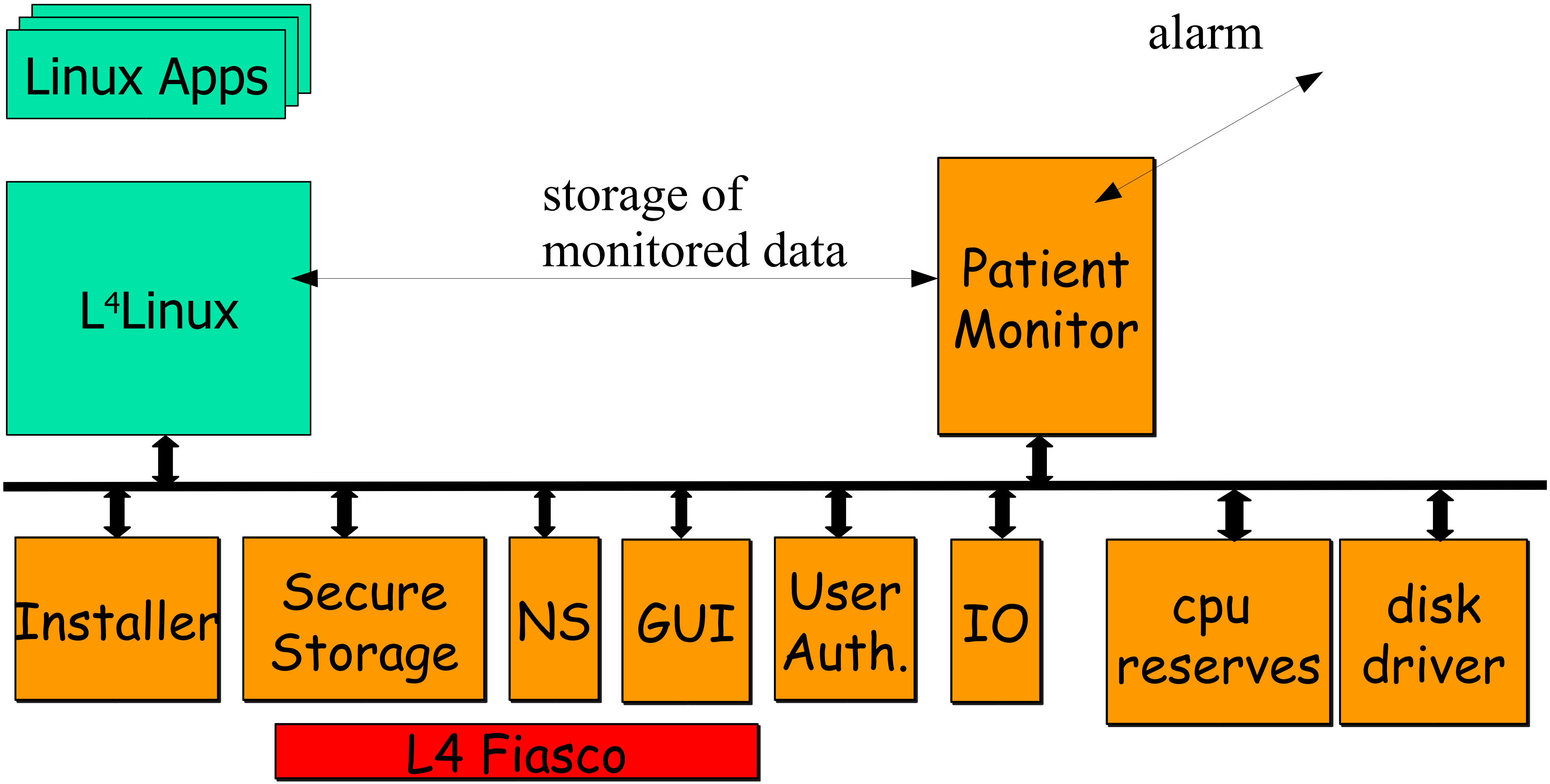
Recoverability



Recoverability



Availability for Secure Storage

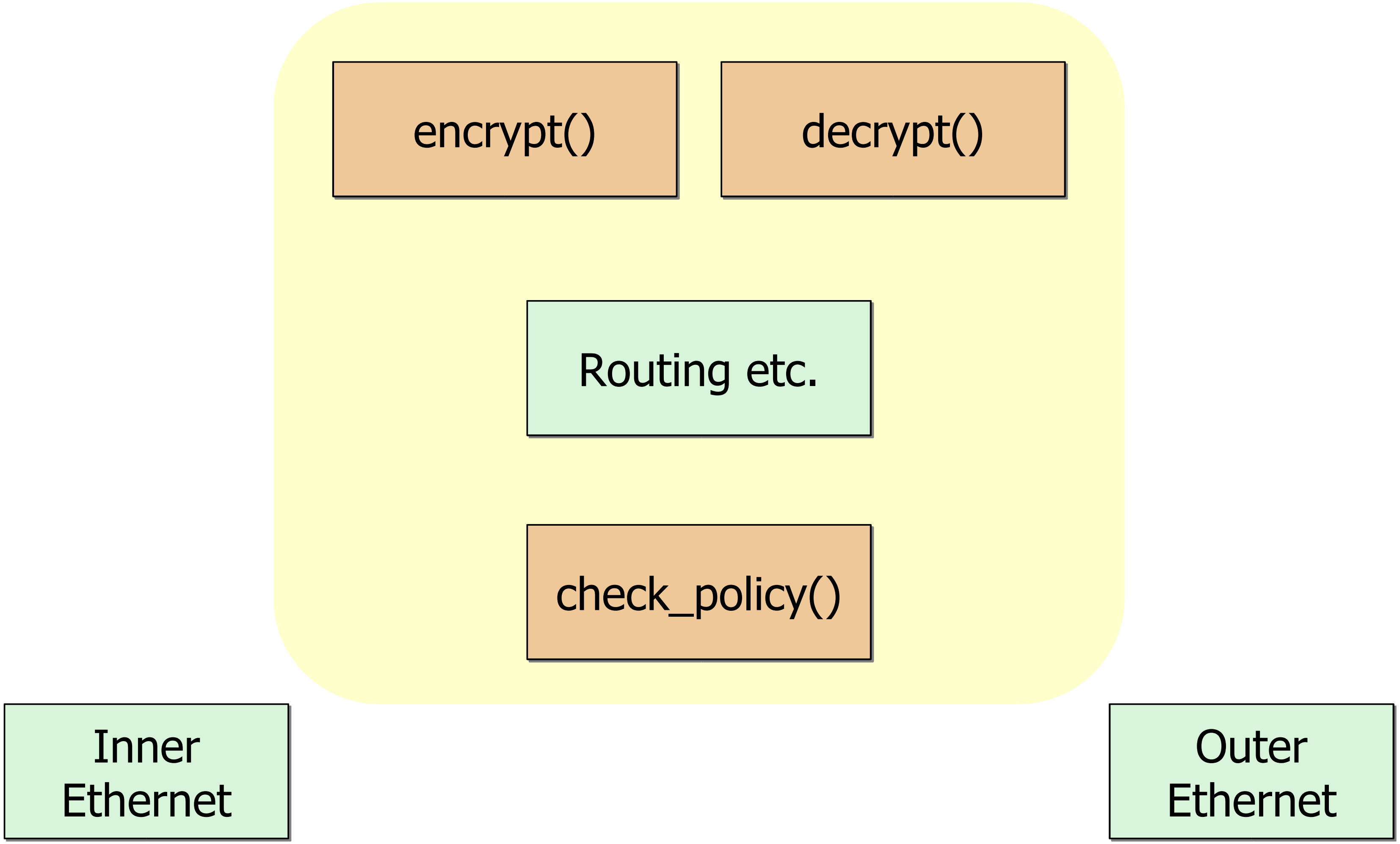


Micro-Sina

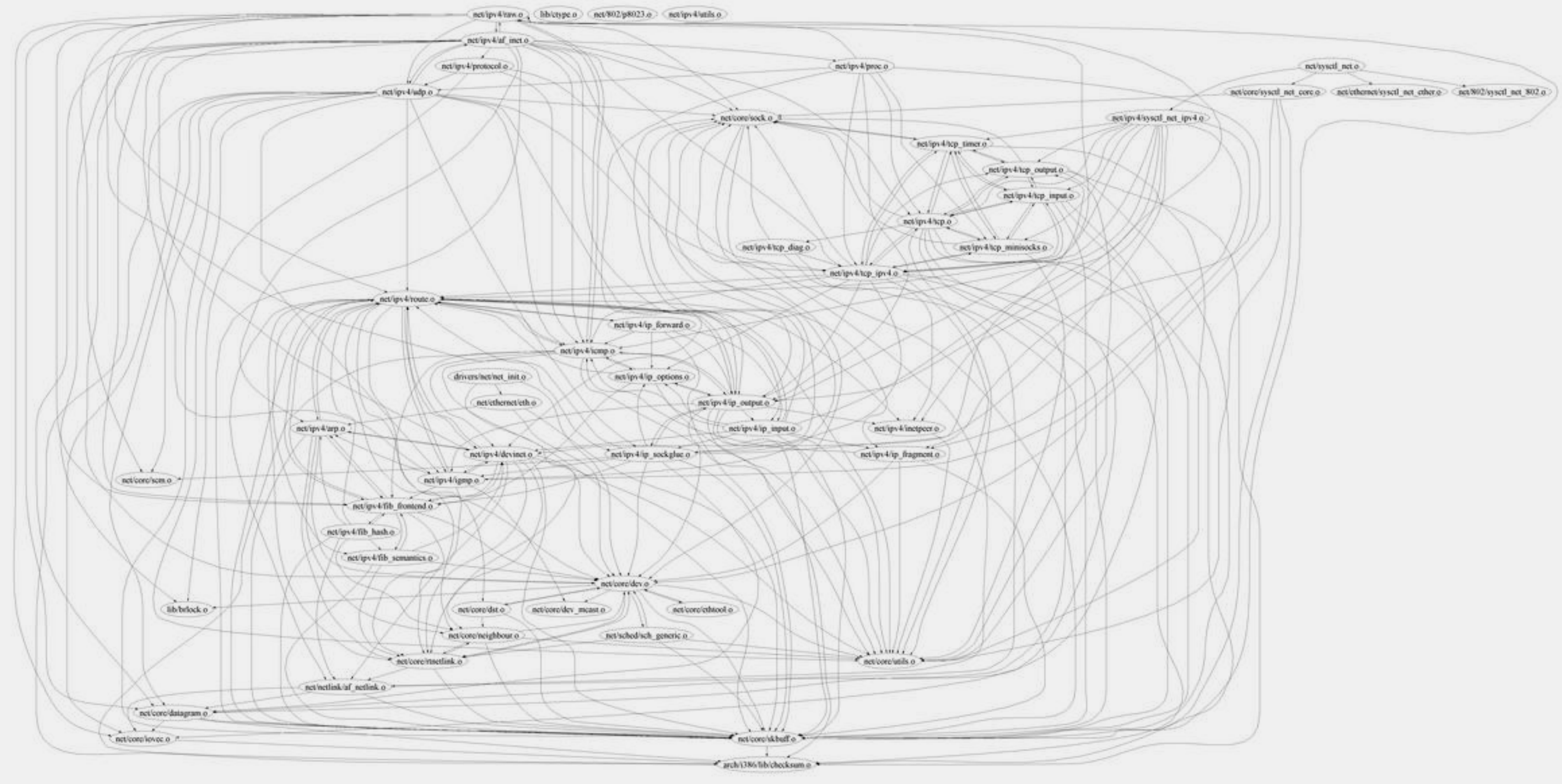
- VPN Box
- reengineering a commercial product
- first approach:
  split Linux' IP-Sec

First Approach

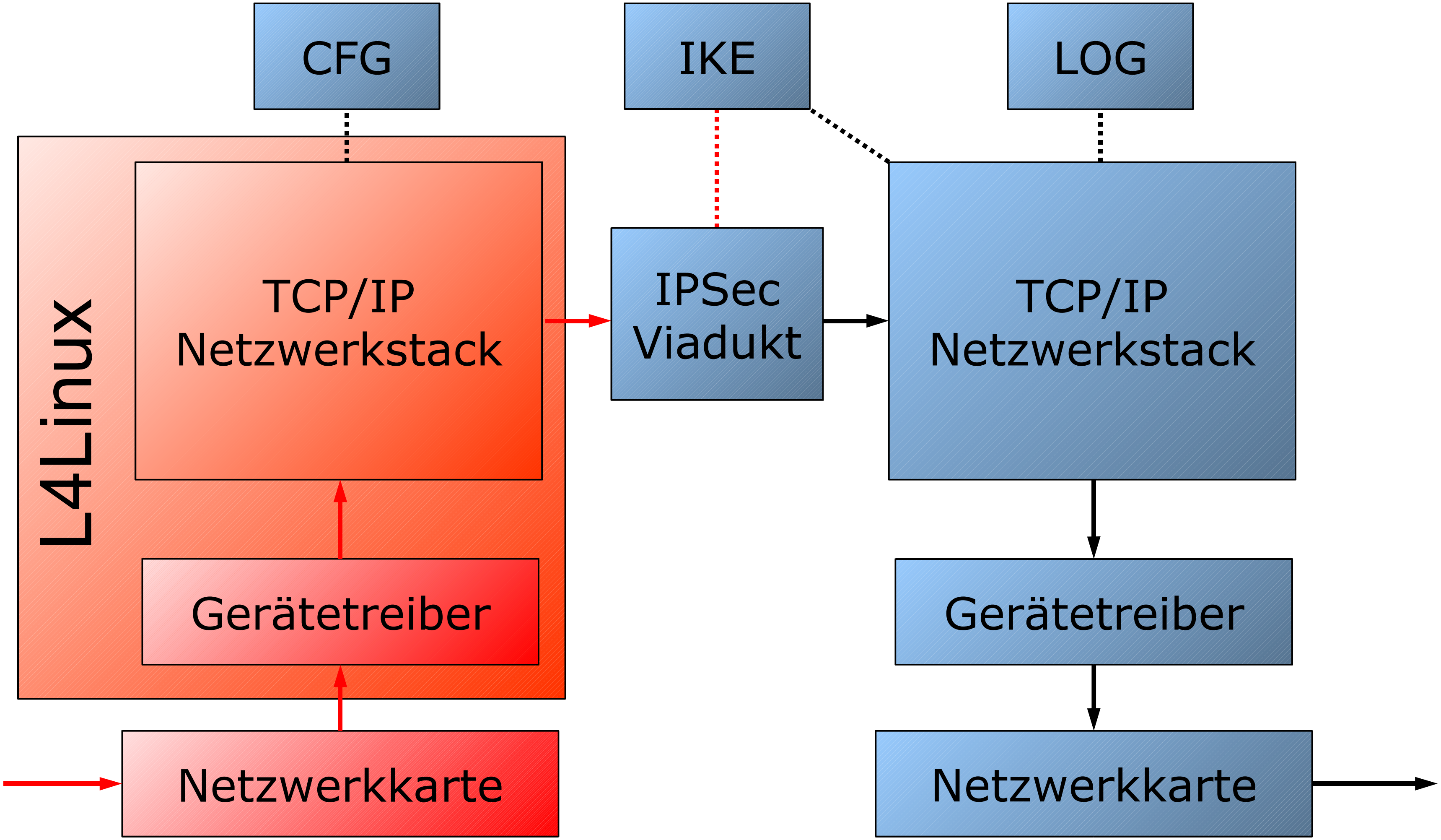
- Decompose network stack software on the basis of IP packet flow



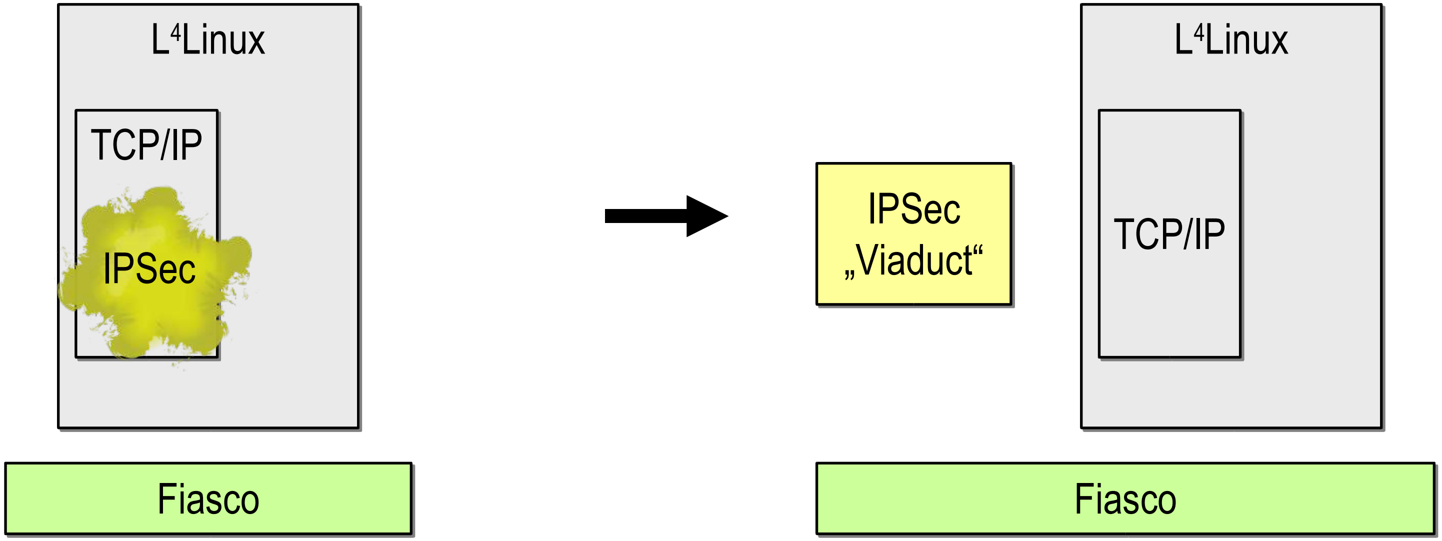
Given up


net/ipv4/raw.o
lib/ctype.o
net/802/p8023.o
net/ipv4/utils.o
net/ipv4/af_inet.o
net/ipv4/protocol.o
net/ipv4/proc.o
net/sysctl_net.o
net/ipv4/udp.o
net/core/sysctl_net_core.o
net/ethernet/sysctl_net_ether.o
net/802/sysctl_net_802.o
net/core/sock.o
net/ipv4/sysctl_net_ipv4.o
net/ipv4/tcp_timer.o
net/ipv4/tcp_output.o
net/ipv4/tcp_input.o
net/ipv4/tcp.o
net/ipv4/tcp_diag.o
net/ipv4/tcp_minisocks.o
net/ipv4/tcp_ipv4.o
net/ipv4/route.o
net/ipv4/ip_forward.o
net/ipv4/icmp.o
drivers/net/net_init.o
net/ipv4/ip_options.o
net/ethernet/eth.o
net/ipv4/ip_output.o
net/ipv4/arp.o
net/ipv4/ip_input.o
net/ipv4/inetpeer.o
net/ipv4/devinet.o
net/ipv4/ip_sockglue.o
net/ipv4/ip_fragment.o
net/core/scm.o
net/ipv4/igmp.o
net/ipv4/fib_frontend.o
net/ipv4/fib_hash.o
net/ipv4/fib_semantics.o
net/core/dev.o
lib/brlock.o
net/core/dst.o
net/core/dev_mcast.o
net/core/ethtool.o
net/core/neighbour.o
net/sched/sch_generic.o
net/core/rtnetlink.o
net/core/utils.o
net/netlink/af_netlink.o
net/core/datagram.o
net/core/iovec.o
net/core/skbuff.o
arch/i386/lib/checksum.o

Instead: Rewrite and Trusted Wrappers

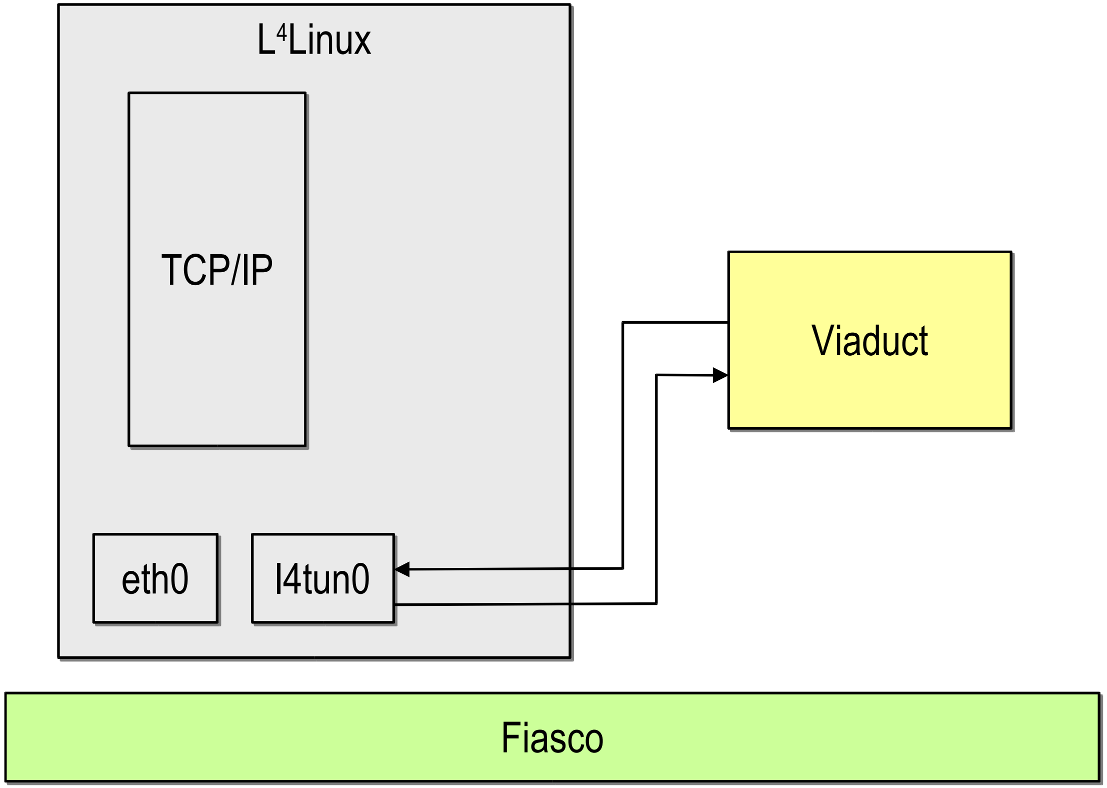


One Step Back



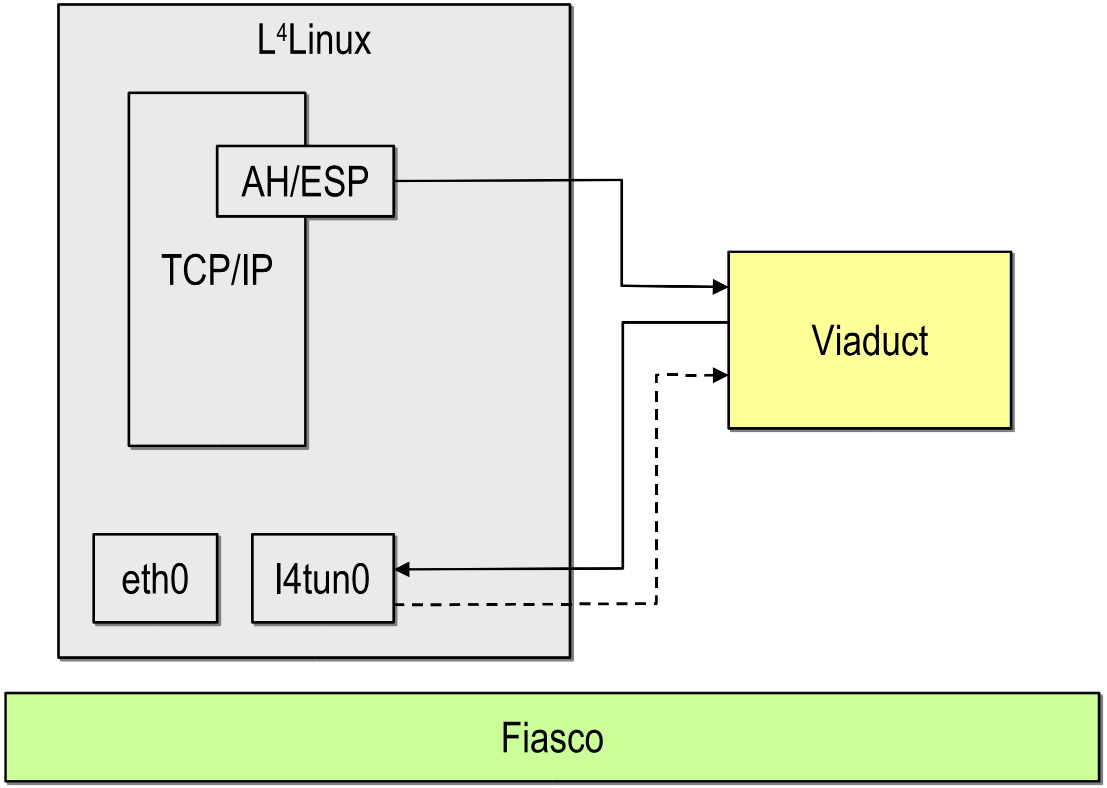
Technical Details (Viaduct)

- IP packets must be passed to the Viaduct
  - L4 IPC as Virtual network driver in L⁴Linux



Technical Details (Viaduct)

- IPSec can only handle unfragmented packets
  - Use L⁴Linux for complex reassembly



Related Work: EROS, Keykos and Nizza

- similar objectives
- moving target

Related Work: Microsoft NGSCB and Nizza

- similar objectives
- moving target

Related Work: XOM and Nizza

- XOM: take OS off trusted path

- Implementing an Untrusted Operating System on Trusted Hardware
David Lie Chandramohan A. Thekkath Mark Horowitz
SOSP 2003

Related Work: Terra and Nizza

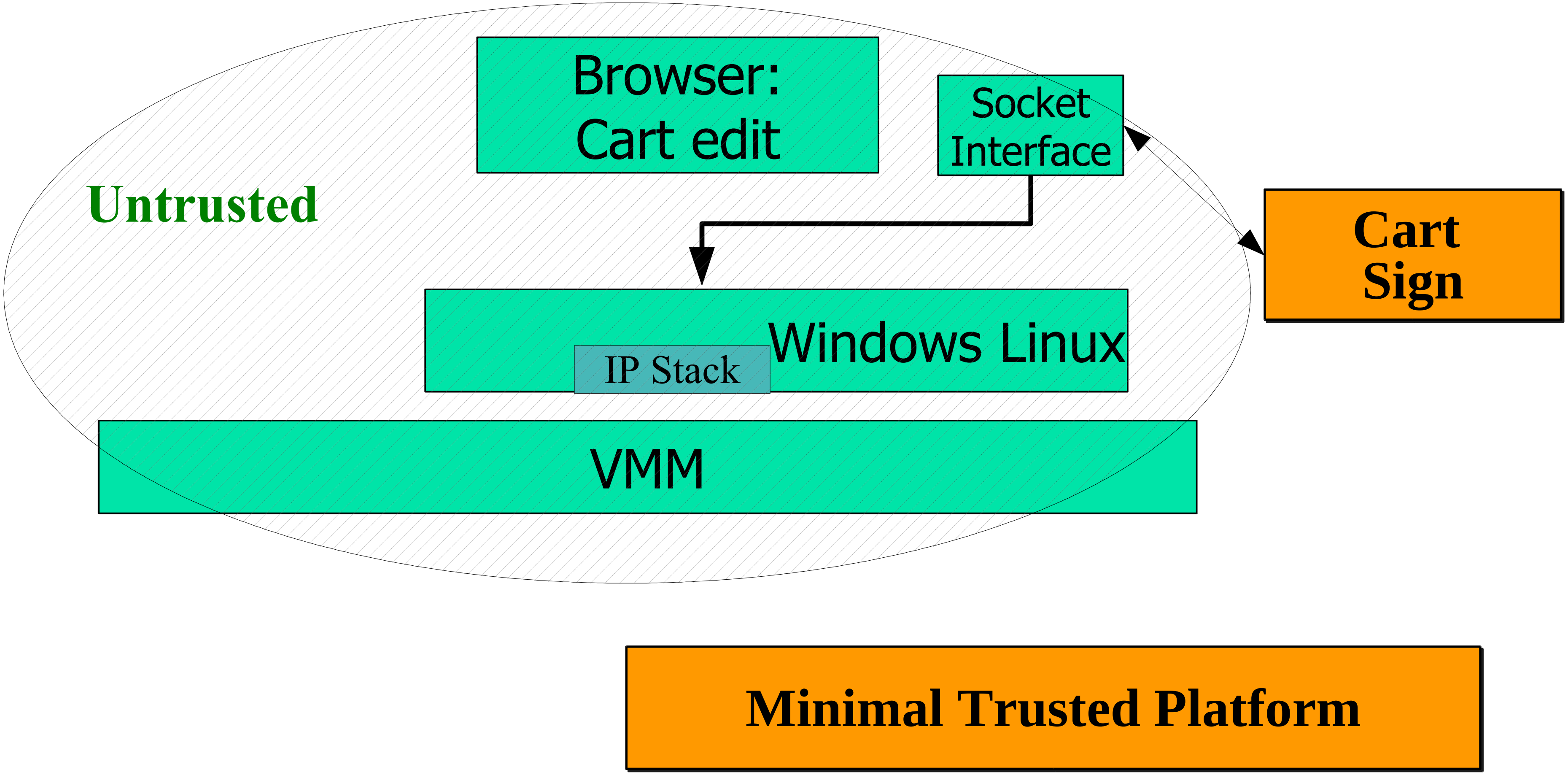
- Terra: VMM as trusted platform
- many more projects down that line

- A Virtual Machine-Based Platform for Trusted Computing
  Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum and Dan Boneh
  SOSP 2003

Nizza vs. VMM approaches

- advantages VMM:
  - support+reuse: unmodified legacy OS
  - anything else ?
- advantages Nizza
  - smaller TCP:
    no network device emulation and drivers
  - fine grained sharing
  - efficiency (optimized message passing)
- VMM untrusted on Nizza ?

Challenge: Untrusted VMM



Related work “Useful” for Nizza

- Secure Storage on Untrusted Servers Secure Untrusted Data Repository (SUNDR)
  Jinyuan Li, Maxwell Krohn, David Mazire s, and Dennis Shasha ,
  New York University
- Privtrans: Automatically Partitioning Programs for Privilege Separation
  David Brumley and Dawn Song,
  Carnegie Mellon University

Technical Risks

- performance
  - copying overhead
  - context switching time (hardware)
  - increased memory (duplication of page tables)

Context switches

- Register IPC between two address spaces (1 x send, 1 x receive; kernel entry with sysenter):

Pentium-III:	600 cycles
Opteron:	700 cycles
Prescott:	2200 cycles

??

Conclusion

technologies are in place

to build much better (securer) systems

need proper integration –> Nizza

TUD♦OS