A kernel in a library
Genode's custom kernel approach

Martin Stein
<martin.stein@genode-labs.com>

Outline

1. Motivation
2. Overview
3. Scheduling
4. Capabilities
5. Communication

Outline

Genode on third-party Kernels

The impetus of diversity

- NOVA, Fiasco.OC, OKL4, L4ka::Pistachio, L4/Fiasco, Linux SeL4

Genode on third-party Kernels

The impetus of diversity

- NOVA, Fiasco.OC, OKL4, L4ka::Pistachio, L4/Fiasco, Linux SeL4
- Flexibility in development and application

Genode on third-party Kernels

The impetus of diversity

- NOVA, Fiasco.OC, OKL4, L4ka::Pistachio, L4/Fiasco, Linux SeL4
- Flexibility in development and application
- Versatility in testing

Genode on third-party Kernels

Kernel perspective

- Aim for comprehensive security concept

Genode on third-party Kernels

Kernel perspective

- Aim for comprehensive security concept
- Self-contained unit that mistrusts all users

Genode on third-party Kernels

Kernel perspective

- Aim for comprehensive security concept
- Self-contained unit that mistrusts all users

Perspective of Genode's Core

- Bring Kernel concept in line with Genode API

Genode on third-party Kernels

Kernel perspective

- Aim for comprehensive security concept
- Self-contained unit that mistrusts all users

Perspective of Genode's Core

- Bring Kernel concept in line with Genode API
- Must be trusted anyway

Genode on third-party Kernels

Drawbacks

- Concepts get bend in shape (Signals)

Genode on third-party Kernels

Drawbacks

- Concepts get bend in shape (Signals)
- Work is done redundantly (memory management)

Genode on third-party Kernels

Drawbacks

- Concepts get bend in shape (Signals)
- Work is done redundantly (memory management)
- Deficiencies get worked around (Capability delegation)

Creating a custom solution

Idea

- Kernel that trusts Core and is designed for Core's needs

Creating a custom solution

Idea

- Kernel that trusts Core and is designed for Core's needs
- Minimalistic library that enables Core to run as root domain

Creating a custom solution

Idea

- Kernel that trusts Core and is designed for Core's needs
- Minimalistic library that enables Core to run as root domain
- Run most critical code in the simplest manner

Outline

Kernel tasks

- Exception vectors

Kernel tasks

- Exception vectors
- Scheduling

Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts

Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
- Communication: IPC and Signals

Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
- Communication: IPC and Signals
- Capabilities

Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
- Communication: IPC and Signals
- Capabilities
- Cache and TLB maintenance

Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
- Communication: IPC and Signals
- Capabilities
- Cache and TLB maintenance
- Virtualization

Kernel interface

	Threads, VMs	PDs, Capabilities	Communication, IRQs
Core-only	thread new/del thread start thread pause thread resume thread route event thread quota vm new/del vm run vm pause	obj new/del pd new/del pd update	signal receiver new/del signal context new/del irq new/del irq ack
Common	thread pause current thread resume local thread yield	update data region update instr region cap ack cap delete	signal context kill signal submit signal await signal ack msg send request msg send reply msg await request

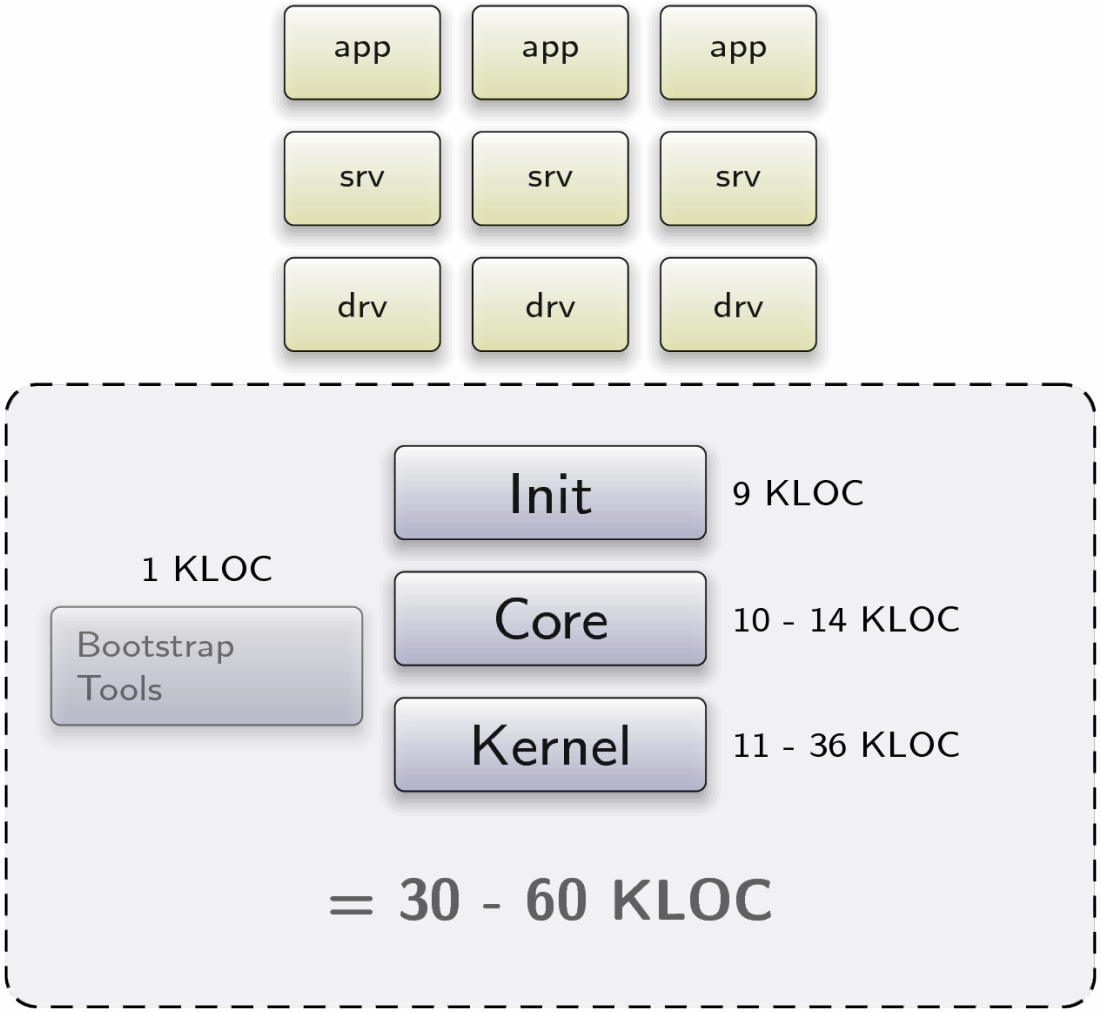
Qualities

- All dynamic memory gets accounted
  $\rightarrow$ No exhaustion

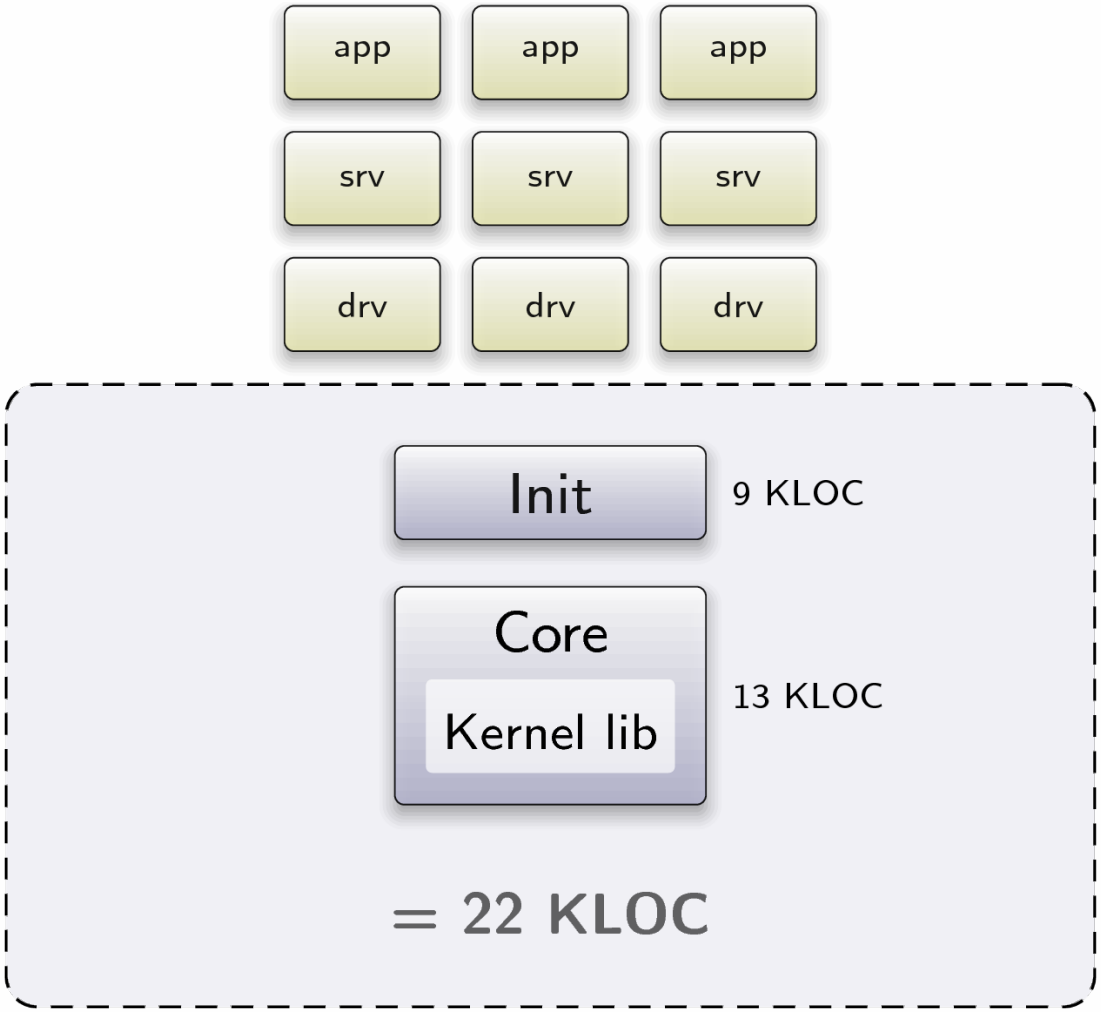
Qualities

- All dynamic memory gets accounted
  $\rightarrow$ No exhaustion
- Modeled as state machine
  $\rightarrow$ Low complexity
  $\rightarrow$ Fast kernel passes

Trusted Computing Base



Trusted Computing Base



Hardware support

- ARMv7
  - Panda Board, i.MX53 QSB, USB Armory, Wand Board, Arndale, Odroid XU, Zynq, PBXA9
  - SMP, Virtualization, Trustzone, ...

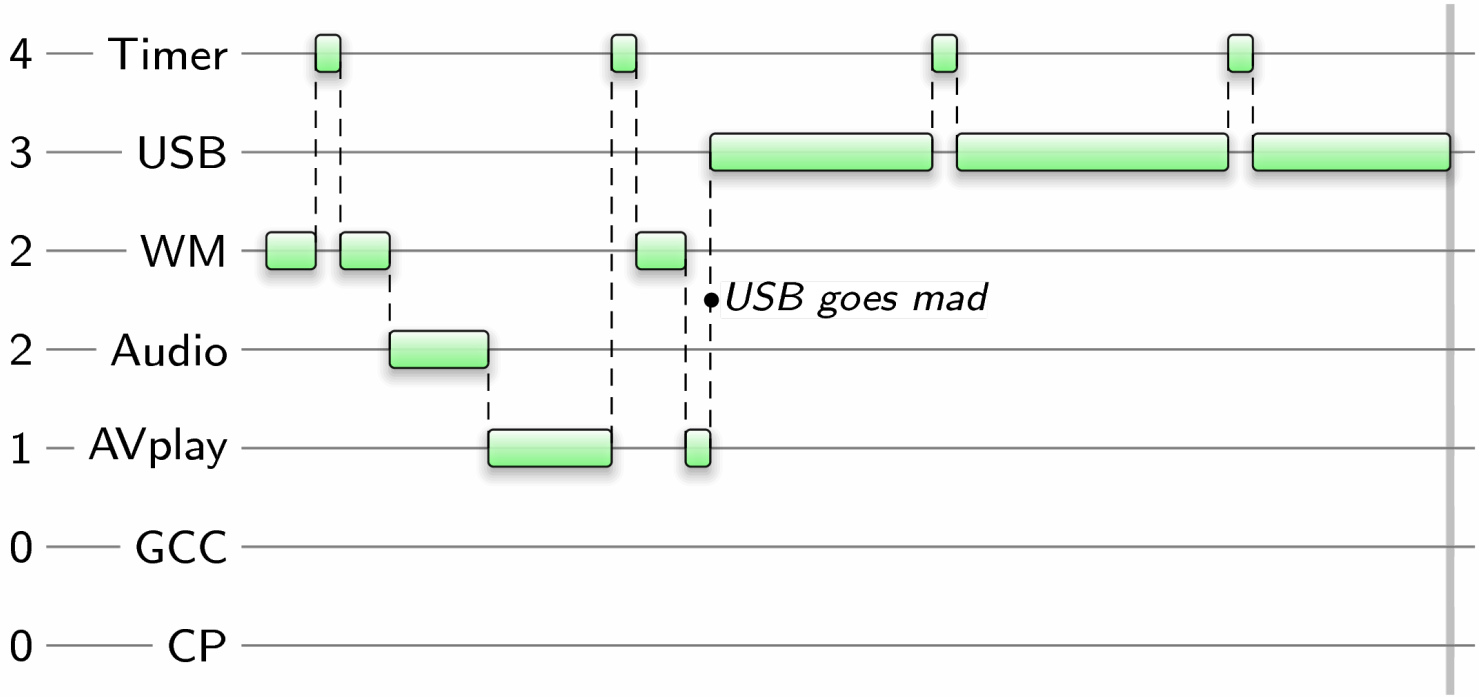
Hardware support

- ARMv7
  - Panda Board, i.MX53 QSB, USB Armory, Wand Board, Arndale, Odroid XU, Zynq, PBXA9
  - SMP, Virtualization, Trustzone, ...
- x86 64 Bit, Raspberry Pi (ARMv6), RISC-V, Muen Separation Kernel

Outline

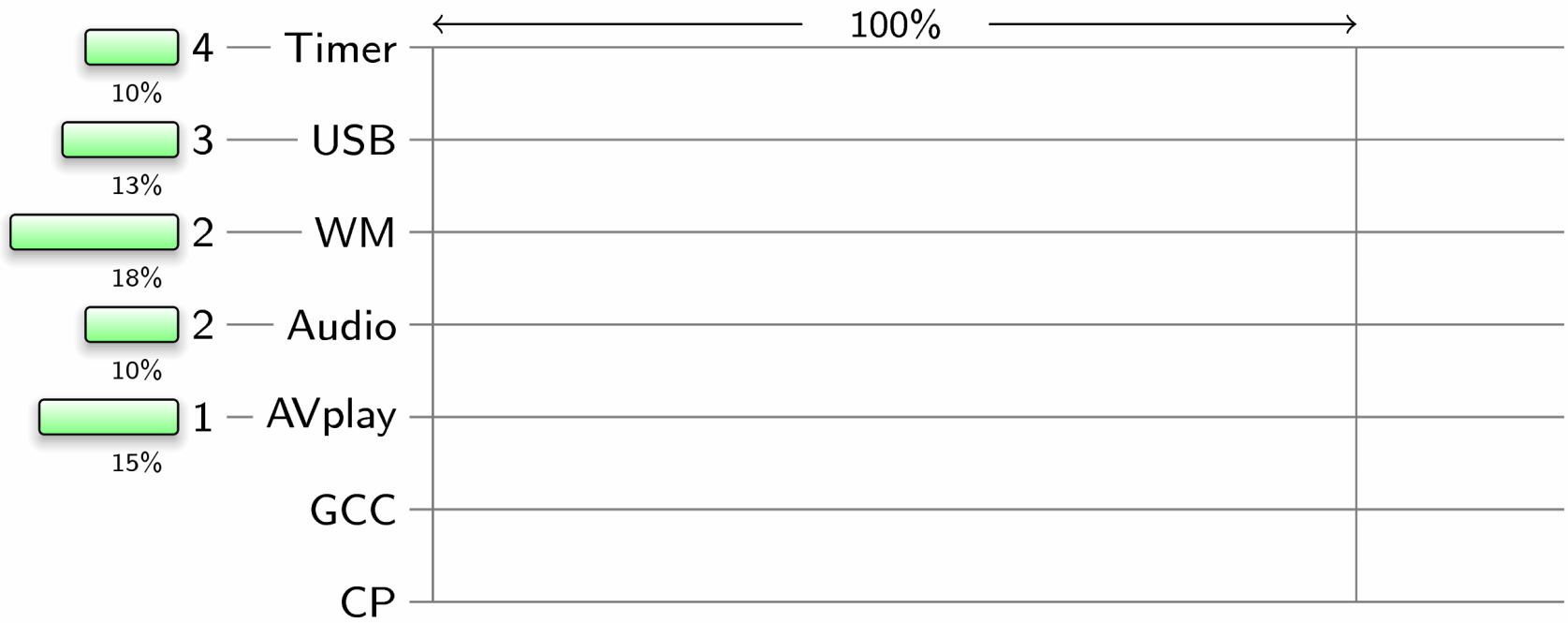
Scheduling

Absolute priorities



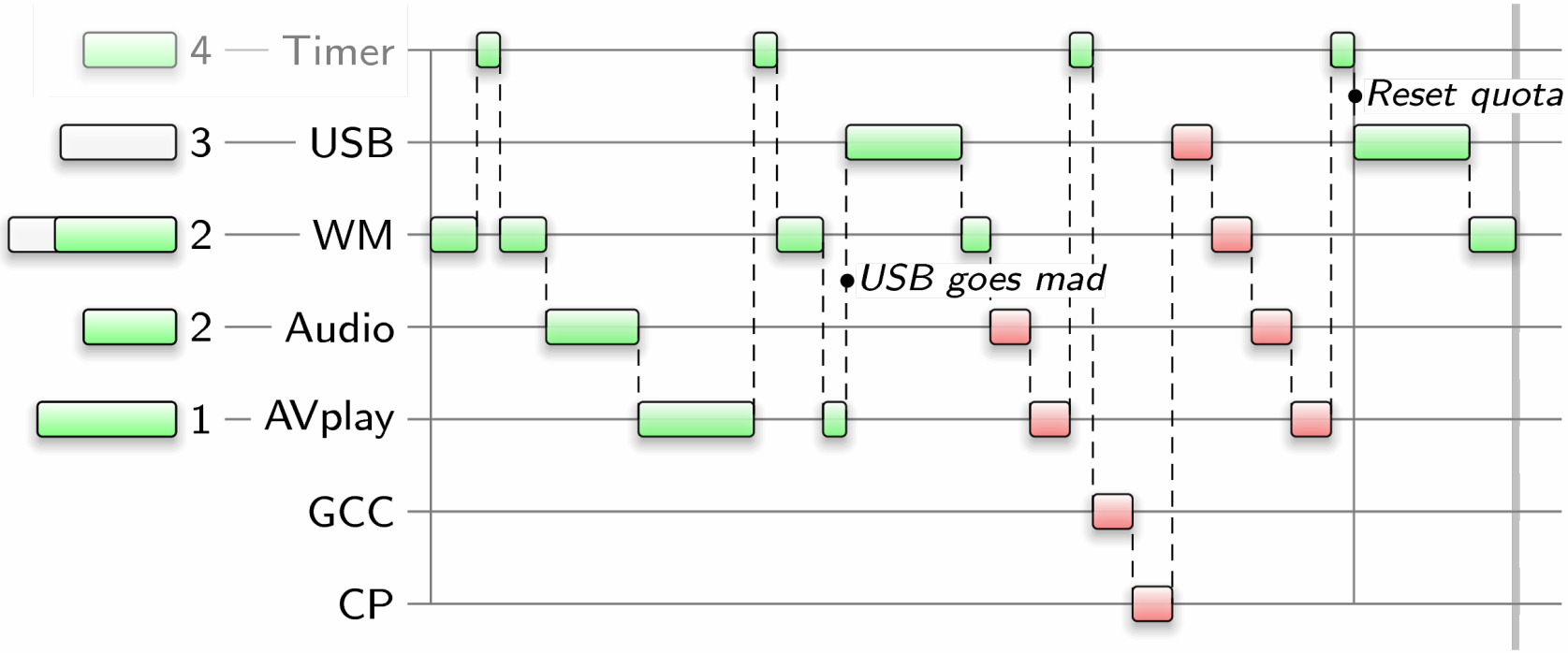
Scheduling

Quota-bound priorities



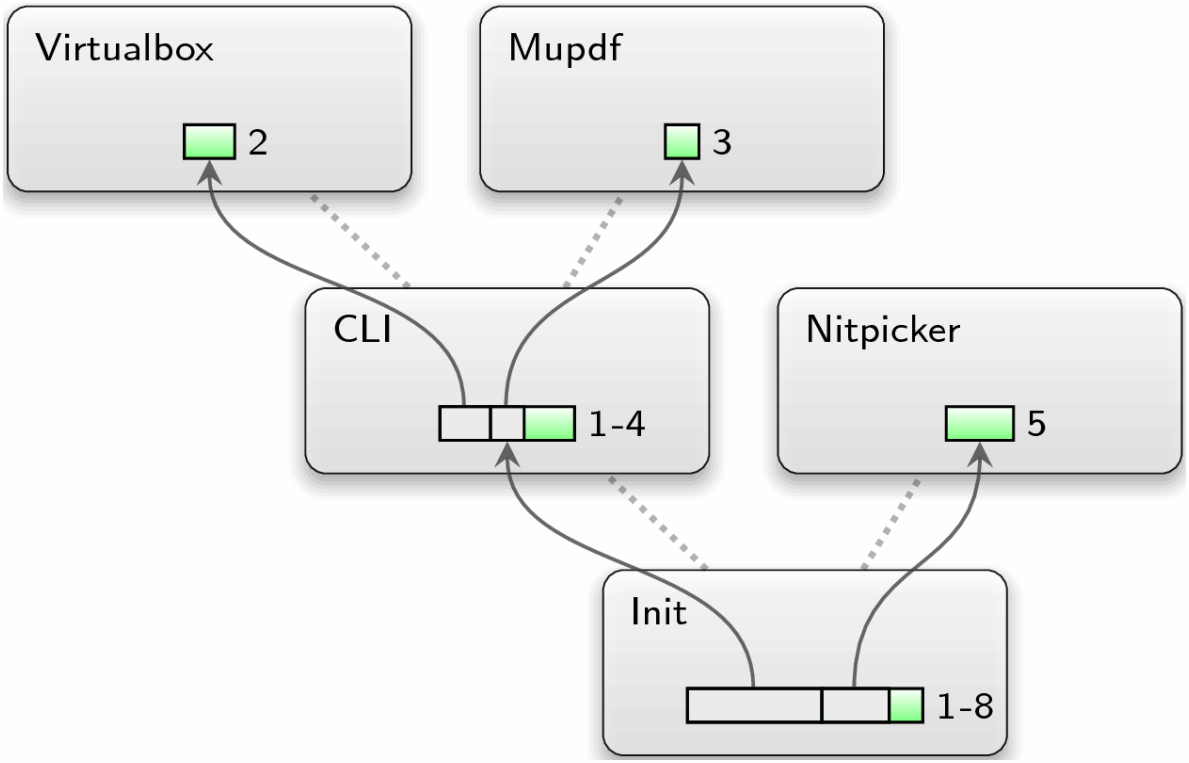
Scheduling

Quota-bound priorities



Scheduling

Donation of CPU resources from parents to their children



Outline

Capabilities

- Automatic creation or translation on IPC delegation

Capabilities

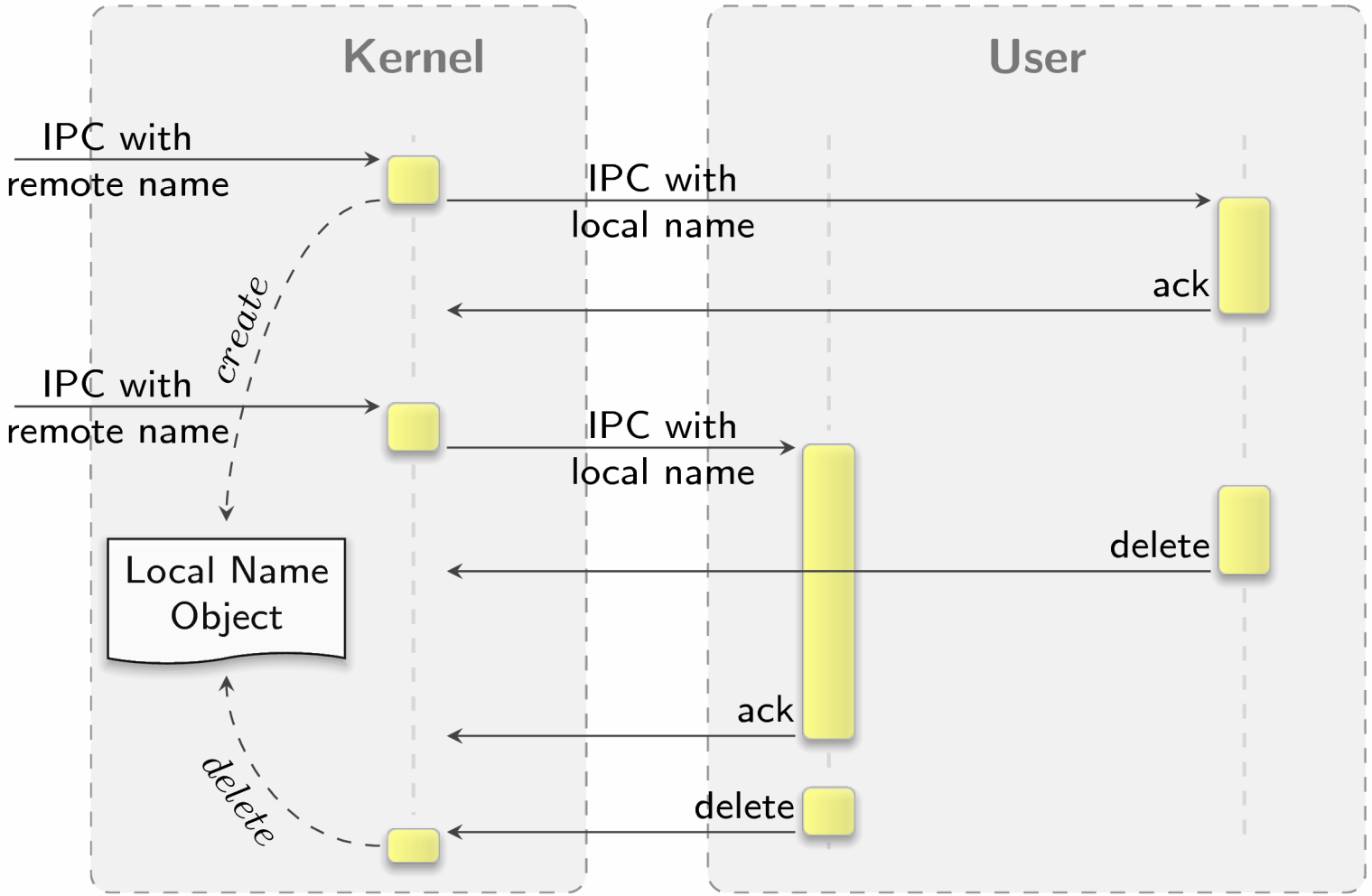
- Automatic creation or translation on IPC delegation
- No name diversity in a PD

Capabilities

- Automatic creation or translation on IPC delegation
- No name diversity in a PD
- Costs get accounted via PD session quota

Capabilities

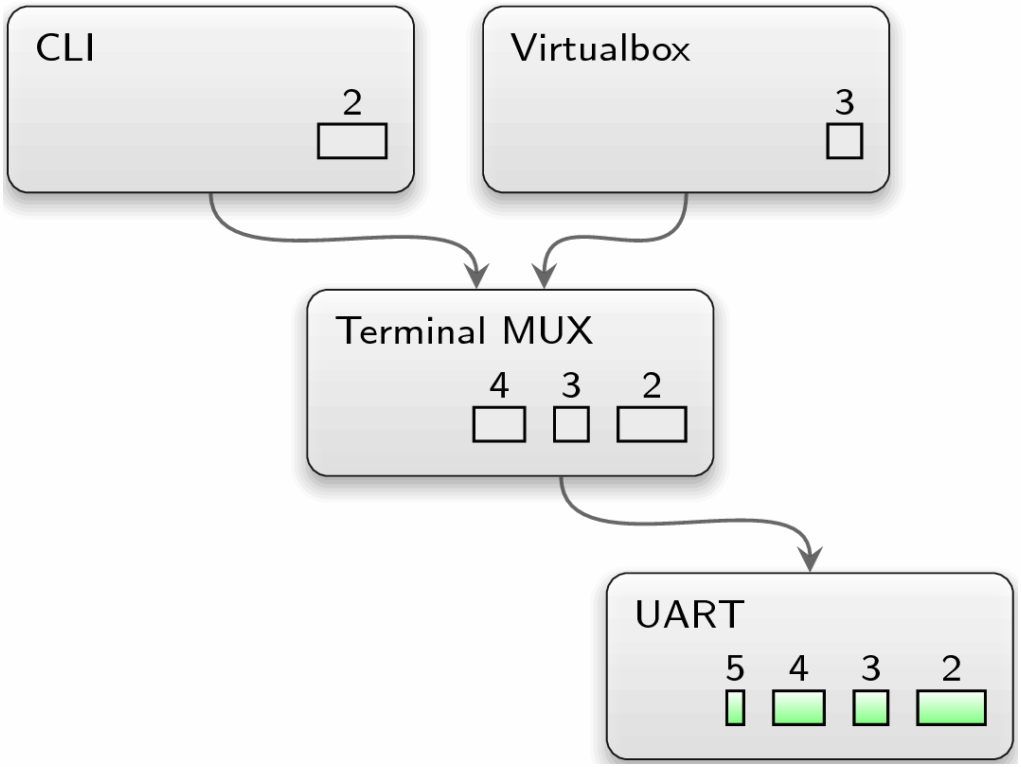
Collaborative lifetime management for Capabilities



Outline

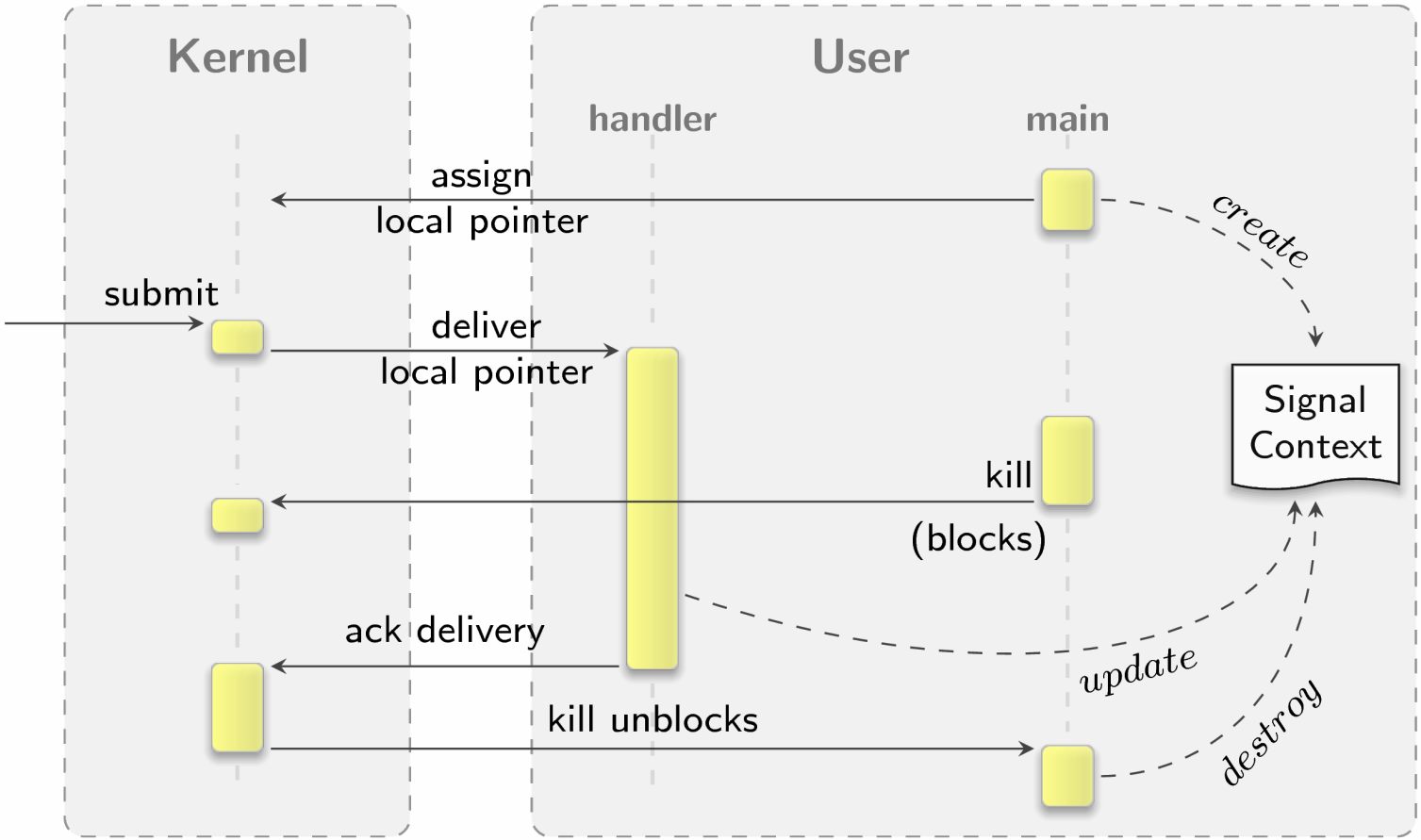
Communication

IPC implicitely delegates CPU resources



Communication

Collaborative lifetime management for Signals



Thank you!

Genode OS Framework

`http://genode.org`

Genode Labs GmbH

`http://genode-labs.com`

Source code at GitHub

`http://github.com/genodelabs/genode`