



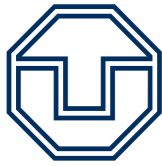
**TECHNISCHE
UNIVERSITÄT
DRESDEN**

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

RESOURCE MANAGEMENT

MICHAEL ROITZSCH

- done: time, drivers
- today: misc. resources
 - architectures for resource management
 - solutions for specific resources
 - capabilities to manage resource access
- upcoming: applications, legacy support
- next week: exercise first in INF E069



KERNEL RESOURCES

- kernel needs memory for its abstractions
 - tasks: page tables
 - threads: kernel-TCB
 - capability tables
 - IPC wait queues
 - mapping database
- kernel memory is limited
- opens the possibility of DoS attacks

- memory management **policy** should not be in the kernel
- account all memory to the application it is needed for (directly or indirectly)
- kernel provides memory control **mechanism**
- exception for bootstrapping:
initial kernel memory is managed by kernel

- **untyped memory** in **seL4**
- all physical memory unused after bootstrap is represented by untyped memory **capabilities**
- can be granted, split or retyped
- restricted to powers of 2 (see flexpages)
- initial resource manager gets all (see σ_0)
- user code decides how to use them

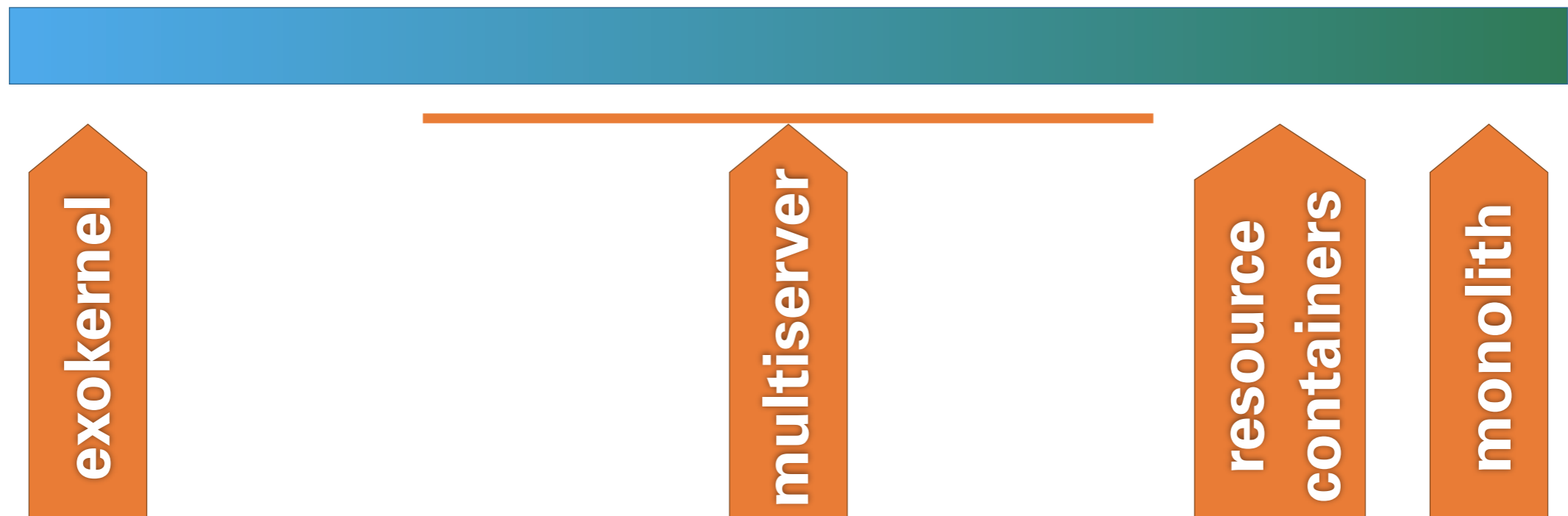
- application **retype** UM to kernel objects
 - TCB, endpoint, CNode, VNode, frame, interrupt
 - all kernel bookkeeping for the object uses the underlying physical memory
 - no implicit memory allocation by the kernel
- retyping and splitting is remembered in **capability derivation tree**
- revoking recursively destroys all derived capabilities and kernel objects

**separate enforcement and
management**

ARCHITECTURES

low-level resource abstractions
explicit management

high-level resource abstractions
implicit management



- enforcement and management implicitly tied to **process abstraction**

isolation

accounting

process

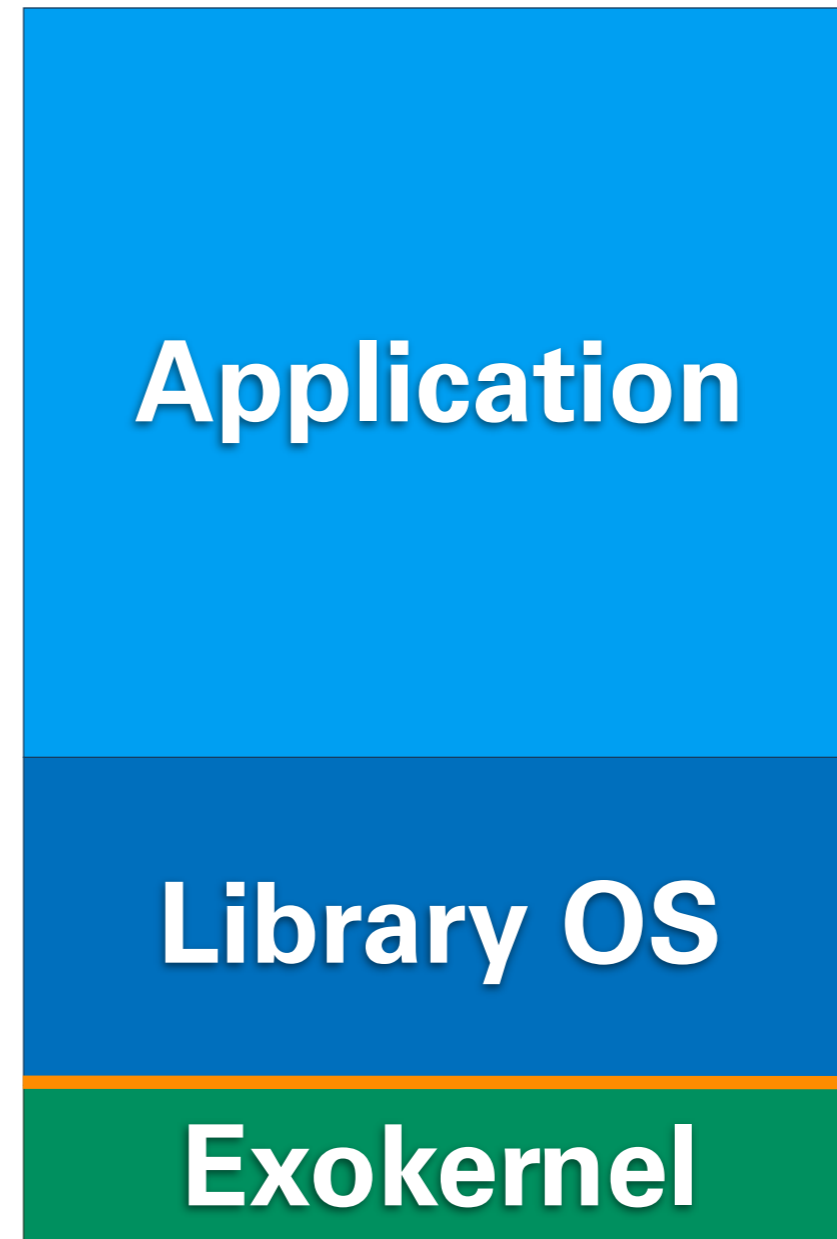
protection domain

resource container

- **resource containers** were proposed to make resource management explicit
- bags of resources assigned to subsystems

Management

Enforcement



- 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

- 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?

- 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



works on monolithic kernels too

- 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

- applications can access resource on the abstraction level they need
- servers implementing a resource can use other, lower-level resources
- isolation allows managers to provide real-time guarantees for their specific resource
- DROPS:
Dresden **R**real-time **O**Perating **S**ystem

EXAMPLES

wget

lwip

Ankh

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

wget

lwip

Ankh

- light-weight IP Stack
- TCP/IP, UDP, ICMP

wget

lwip

Ankh

- clients can use standard BSD socket interface

L4Re VFS

Filesystem

Windhoek

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

L4Re VFS

Filesystem

Windhoek

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

L4Re VFS

Filesystem

Windhoek

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

Terminal

DOpE

mag

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

Terminal

DOpE

mag

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

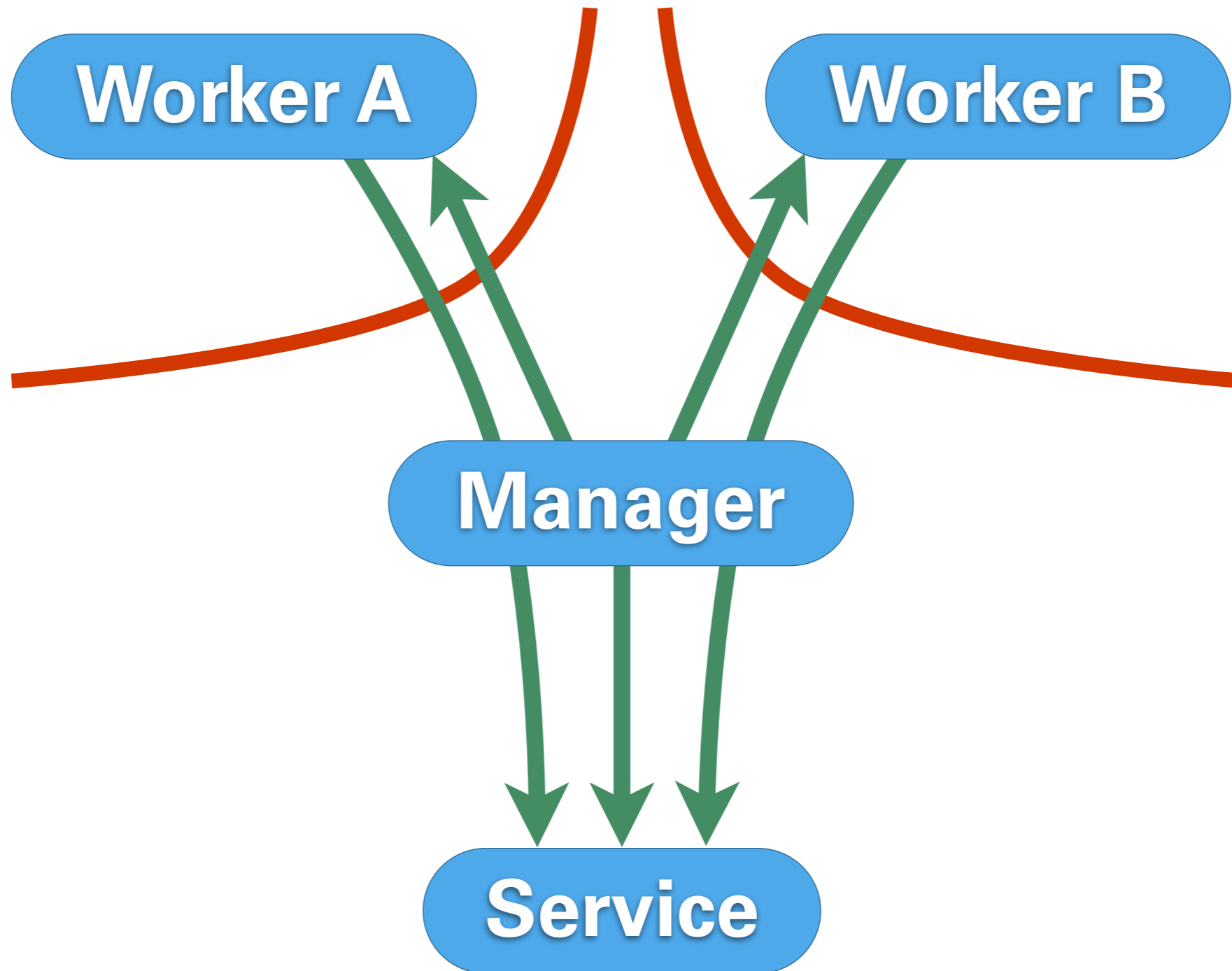
Terminal

DOpE

mag

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

RESOURCE ACCESS



- separate processes
- chrome parent
- sandboxes for tabs
- implementation on Linux: glorious mix of chroot(), clone() and setuid()
- there must be a better way...



POSIX

operations allowed
by default

some limited
restrictions apply

ambient authority

POLA

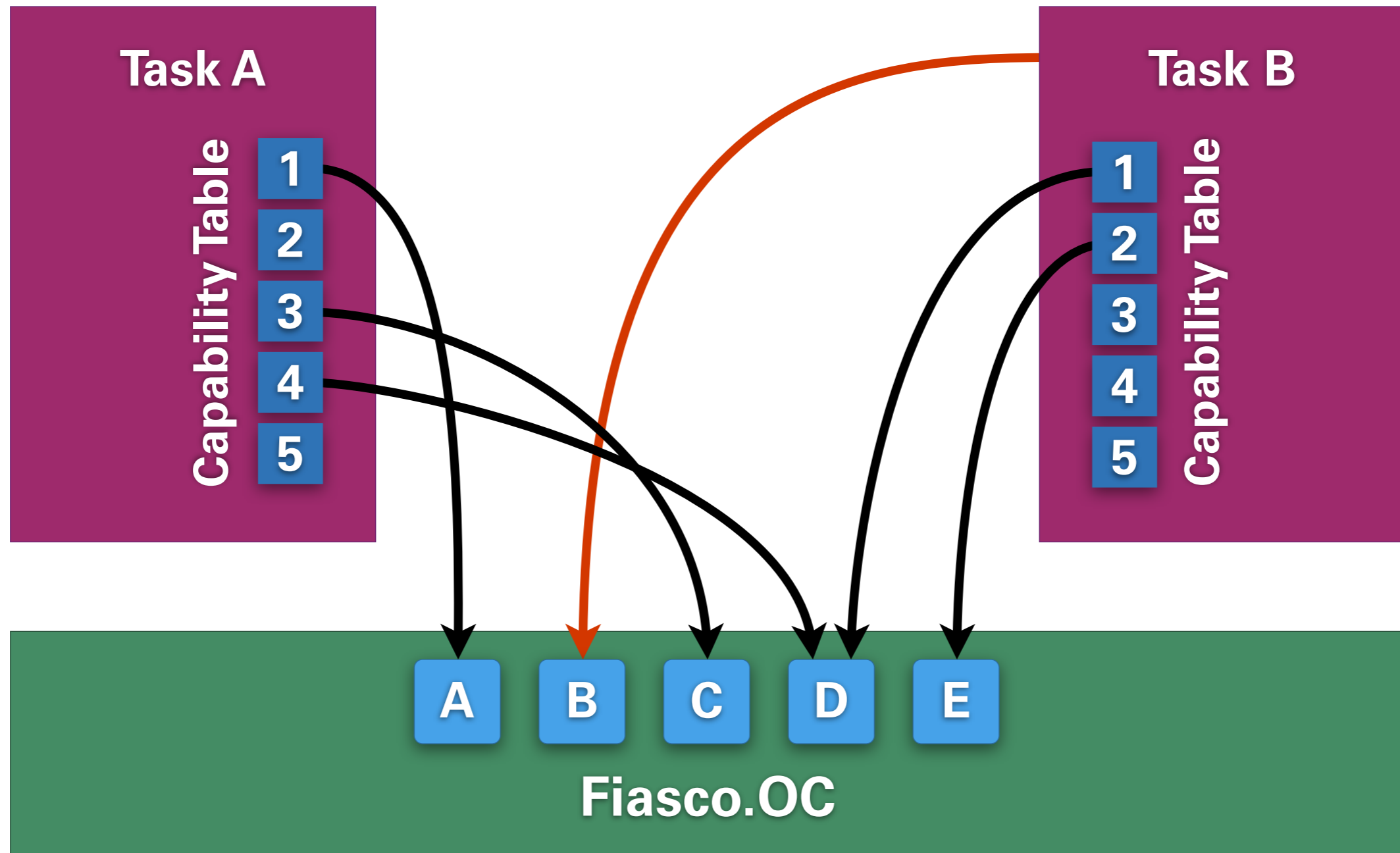
nothing allowed by
default

every right must
be granted

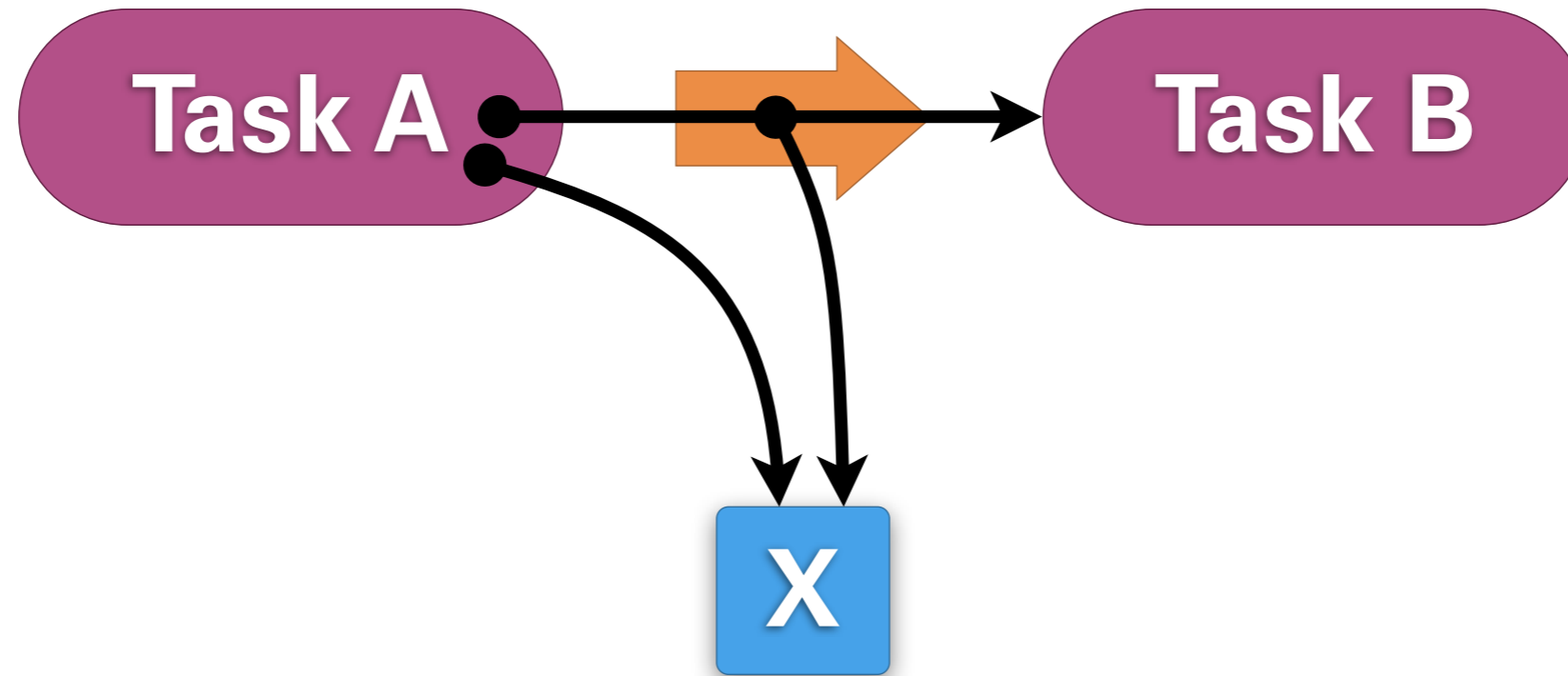
explicit authority

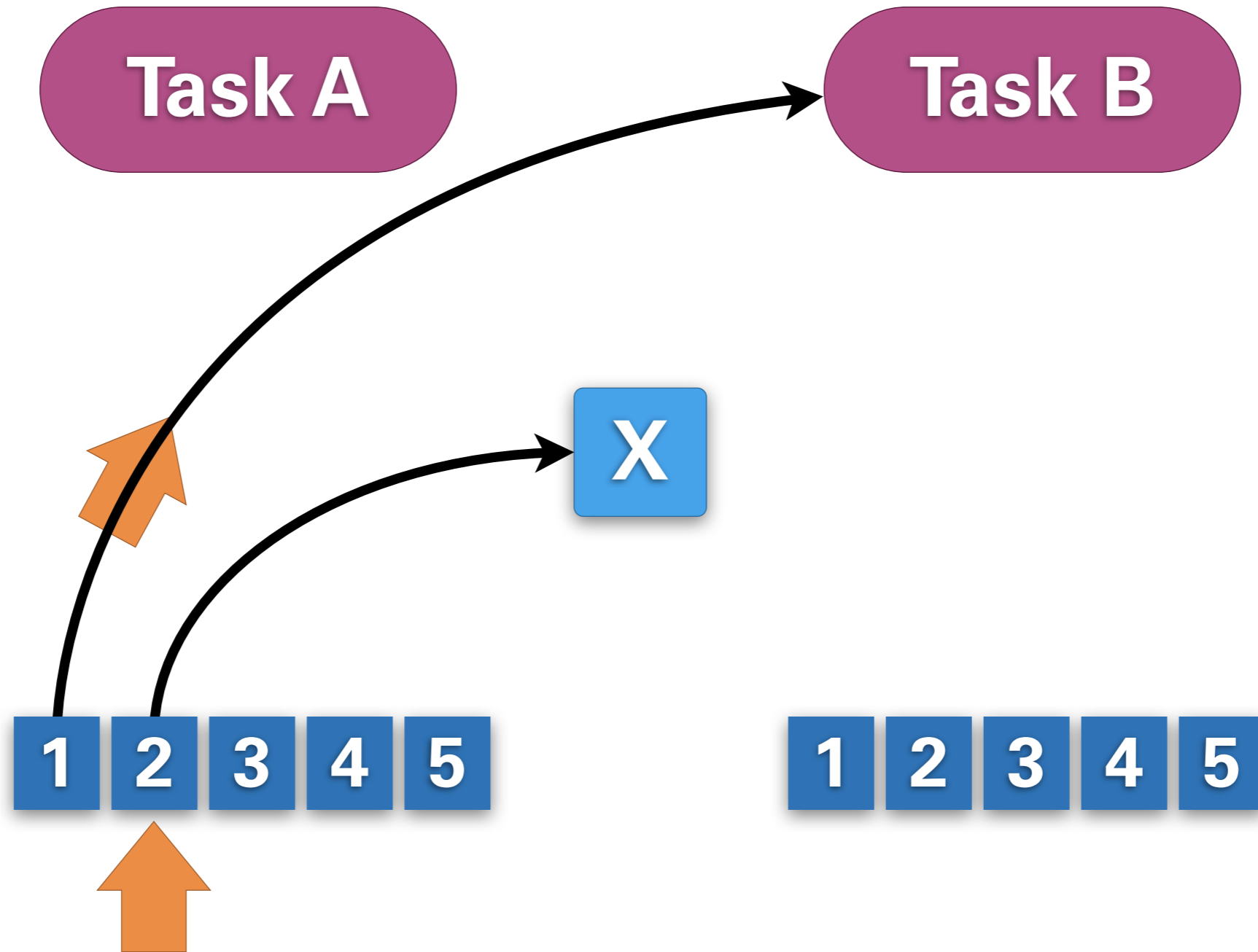
L4Re — the L4 Runtime Environment
set of libraries and system services on
top of the Fiasco.OC microkernel

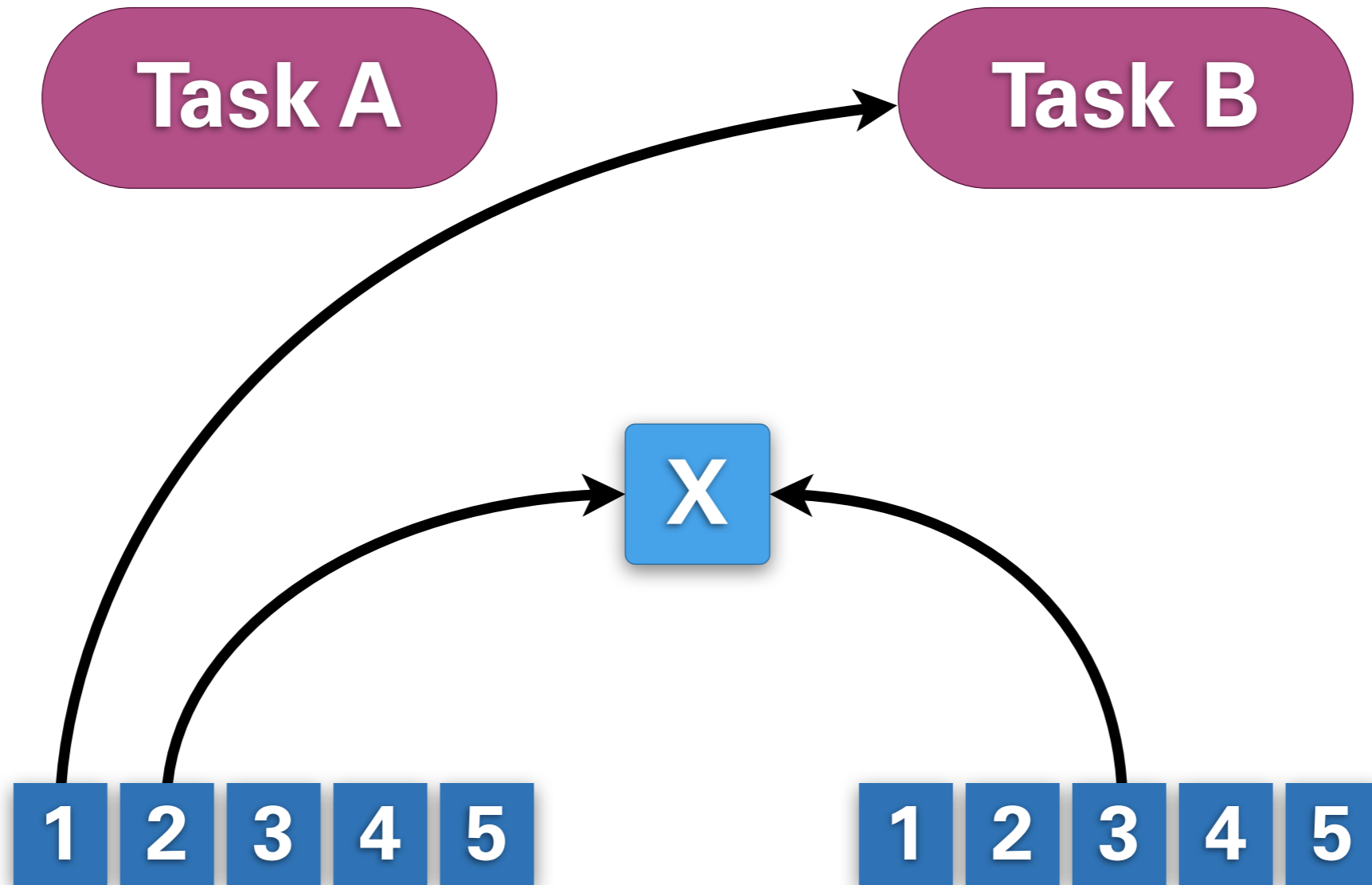
- Fiasco.OC and L4Re form an **object-capability** system
- actors in the system are **objects**
 - objects have local state and behavior
- **capabilities** are references to objects
 - any object interaction requires a capability
 - unseparable and unforgeable combination of reference and access right

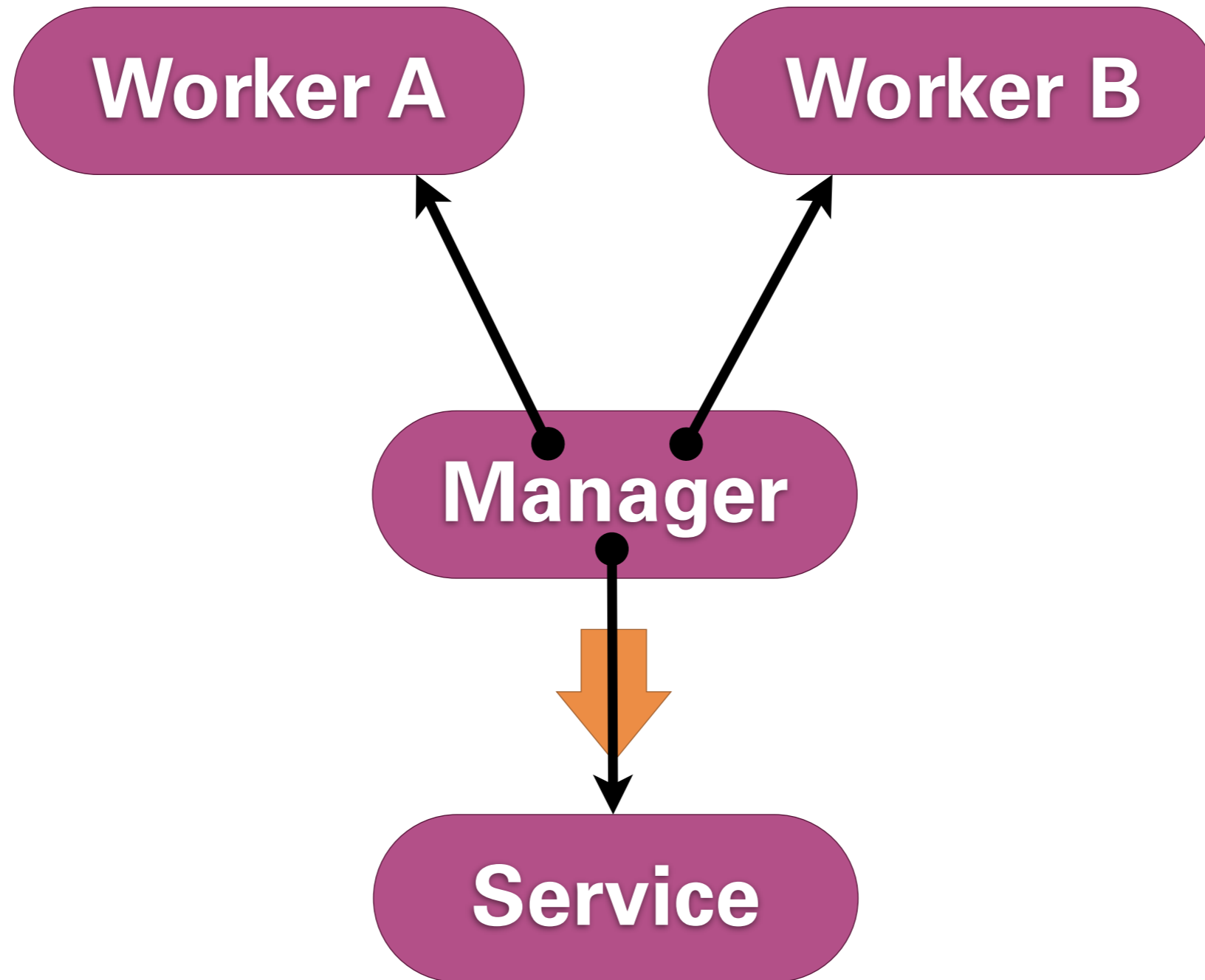


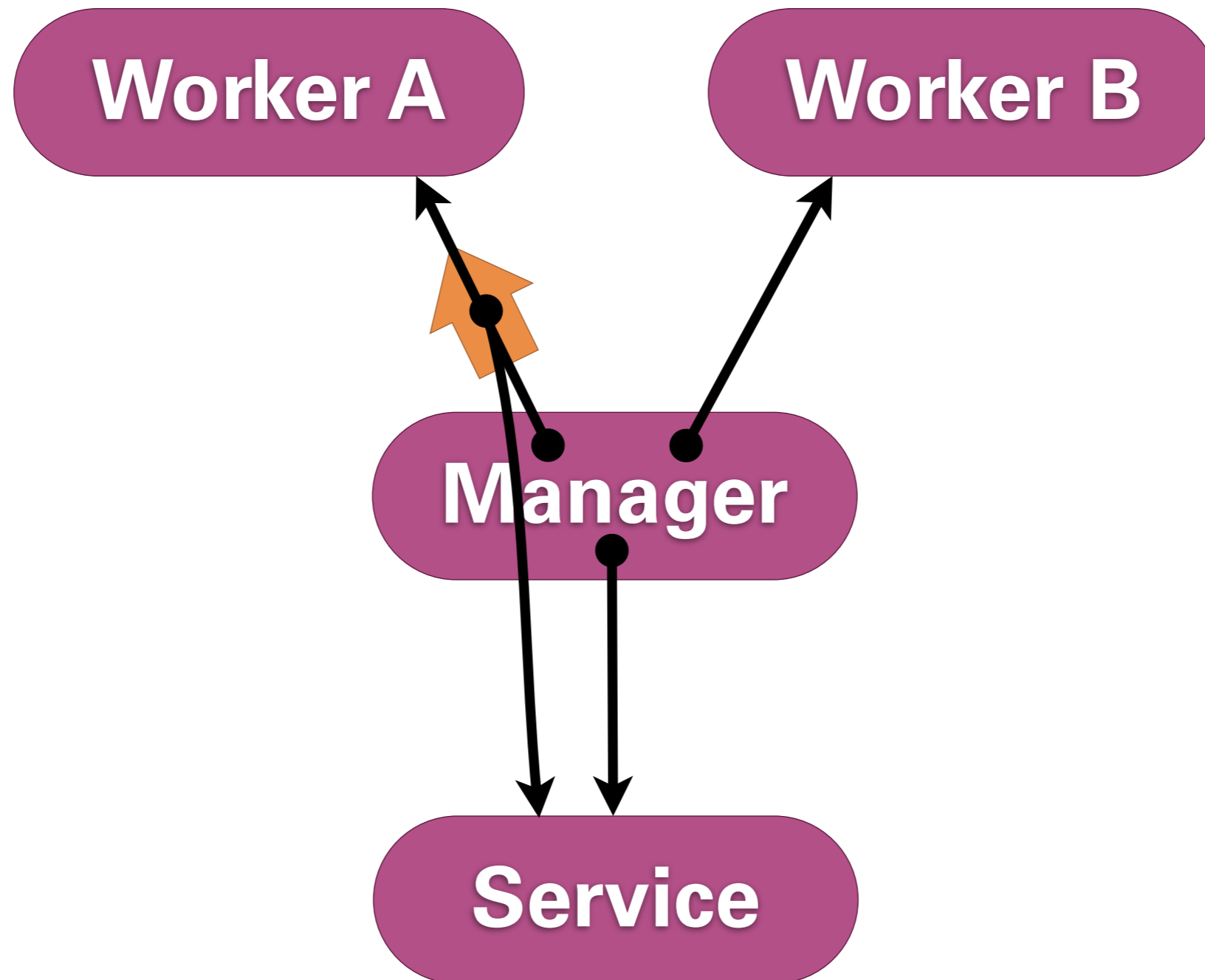
- invocation of any object requires a capability to that object
 - no global names
- no sophisticated rights representation beyond capability ownership
 - just four rights bits on objects
- C++ language integration
- capabilities passed as message payload

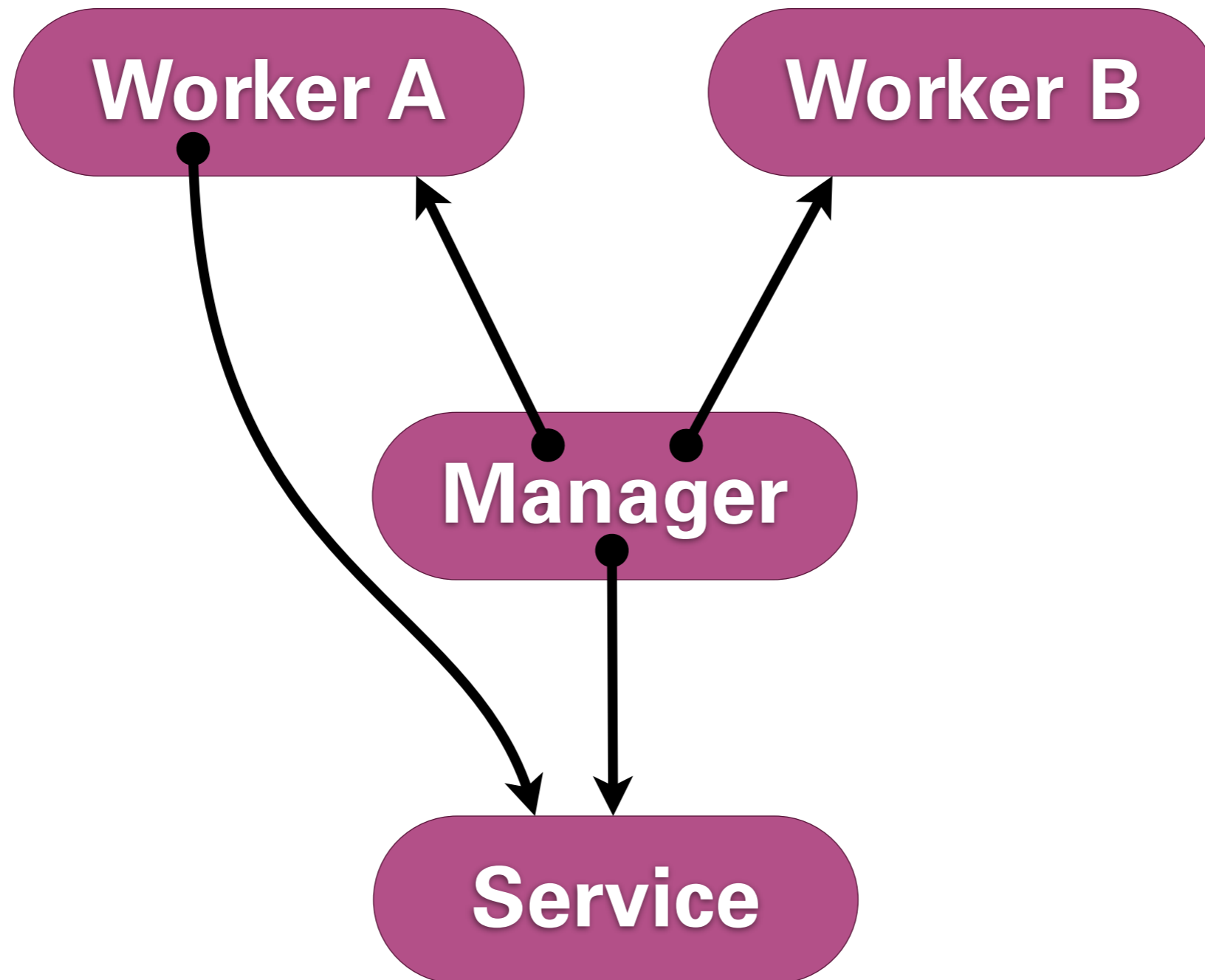


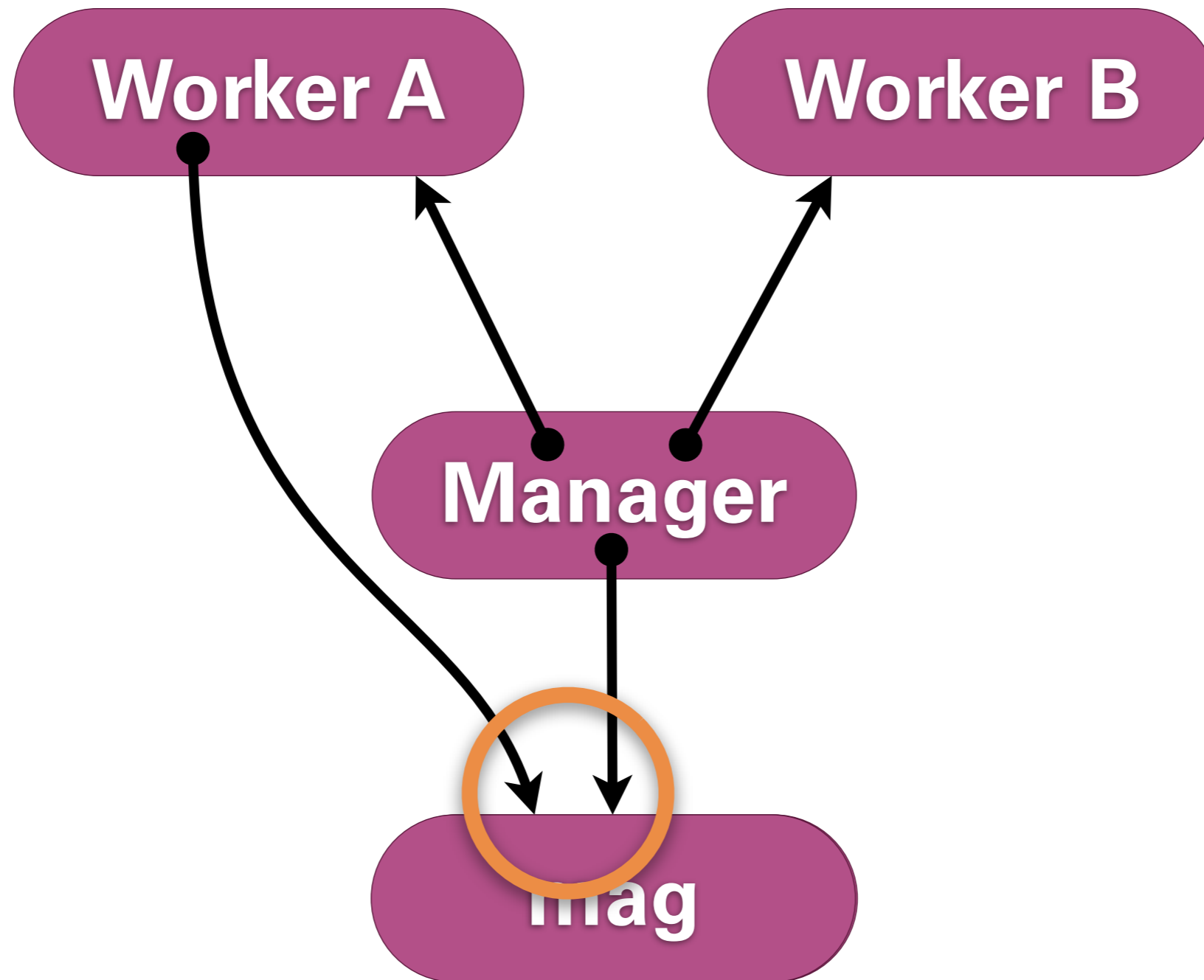


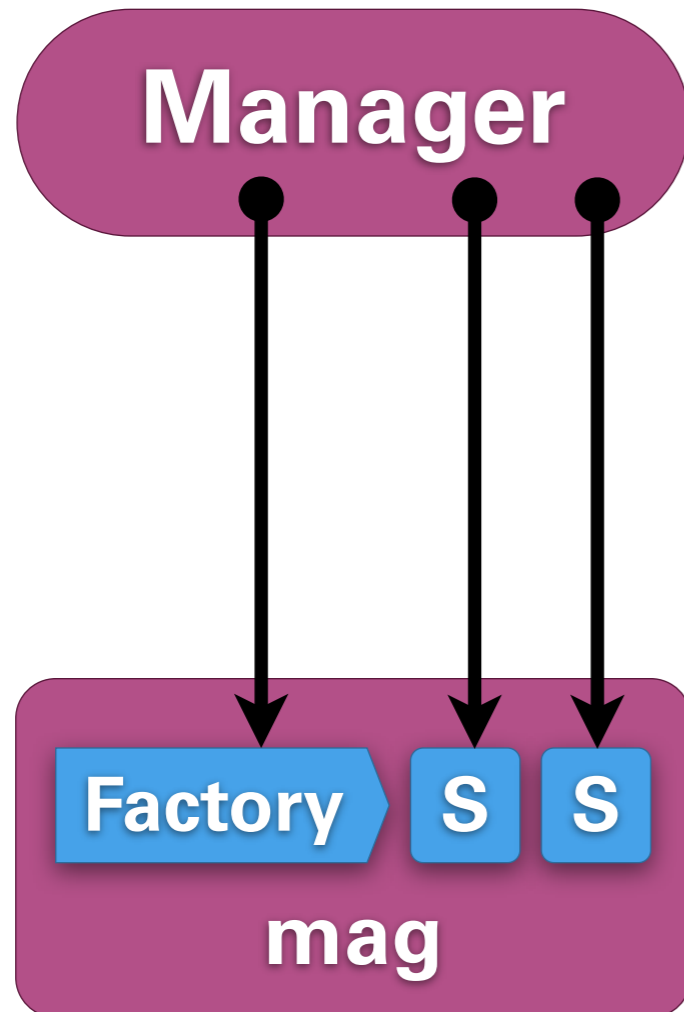




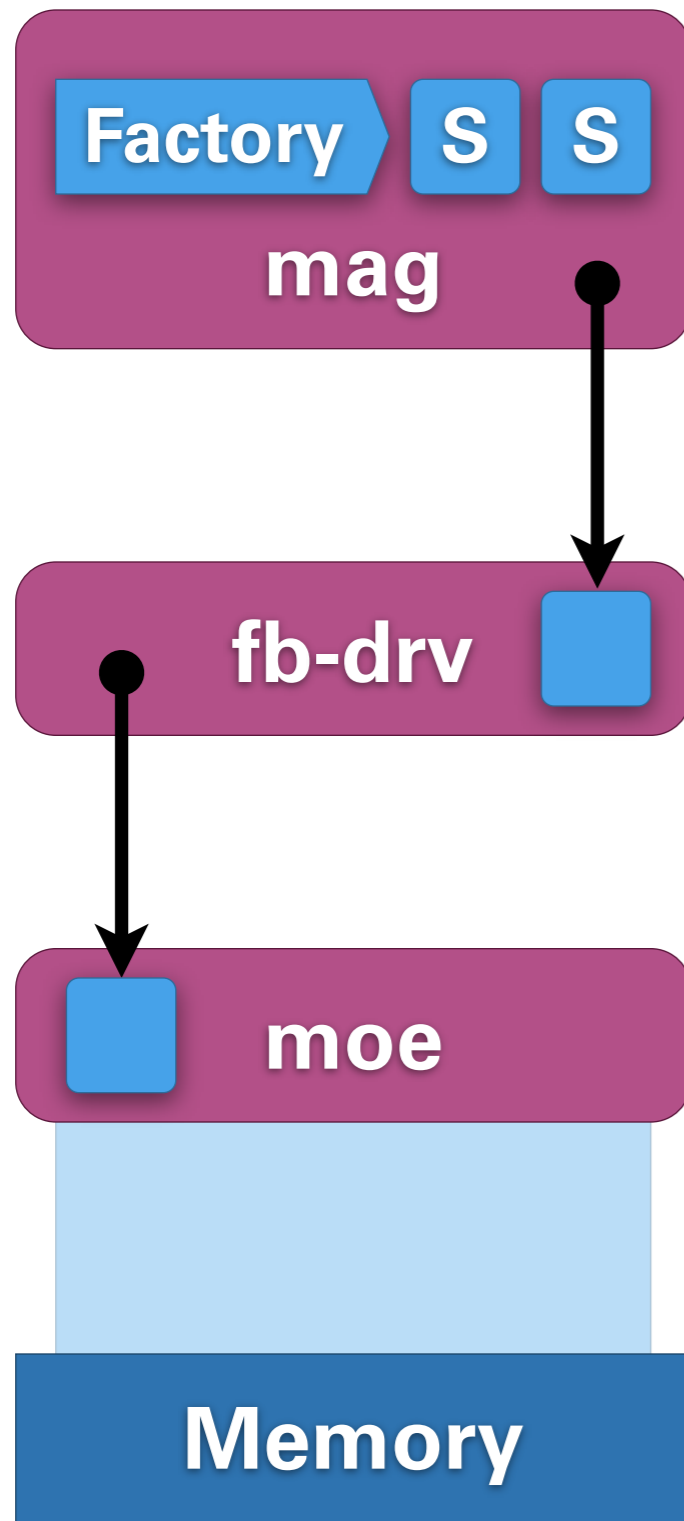








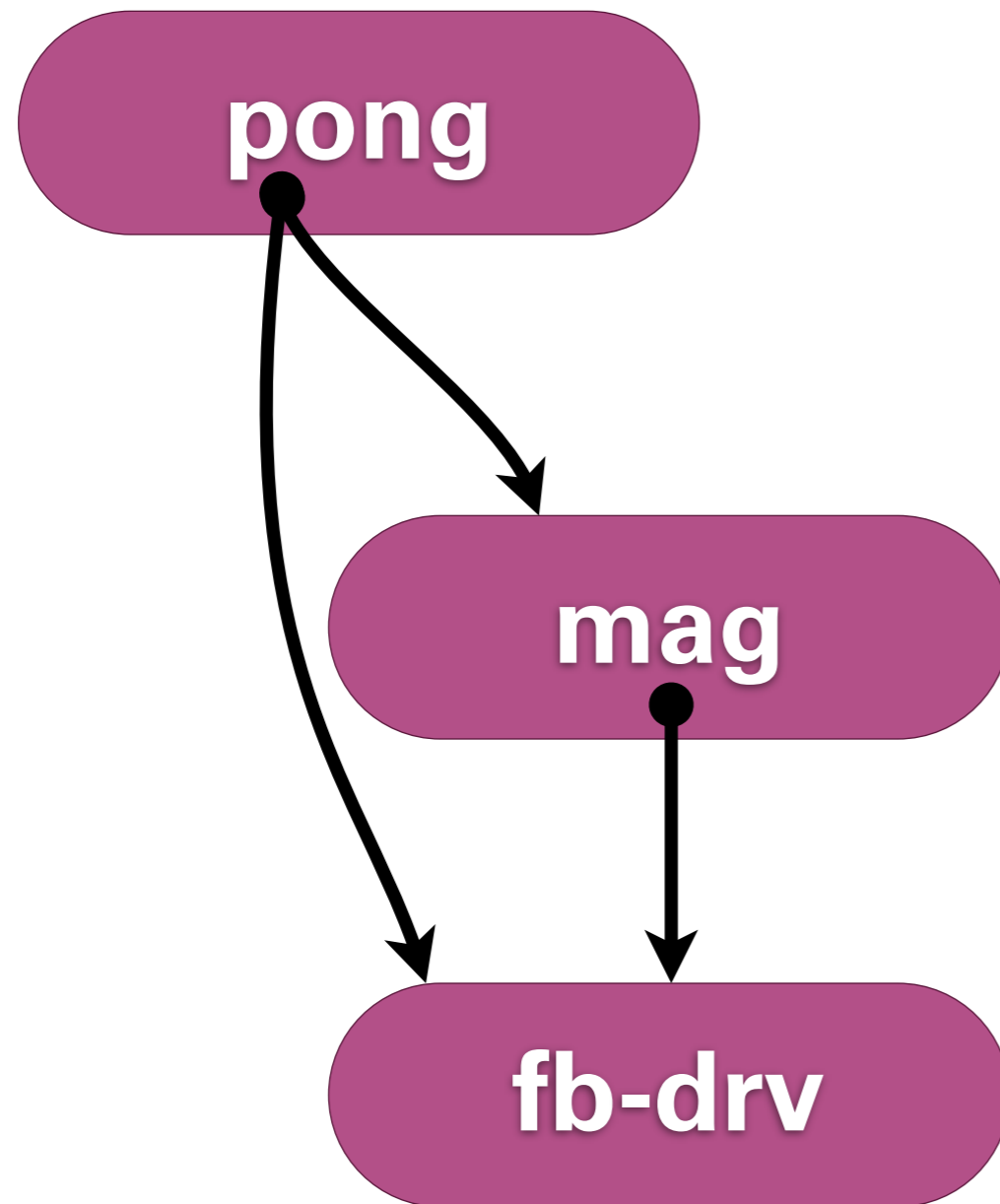
- **factory** for new framebuffer **sessions**
- session object
 - backing store memory
 - view: visible rectangle on the backing store
 - metadata, refresh method
- How does it appear on the screen?



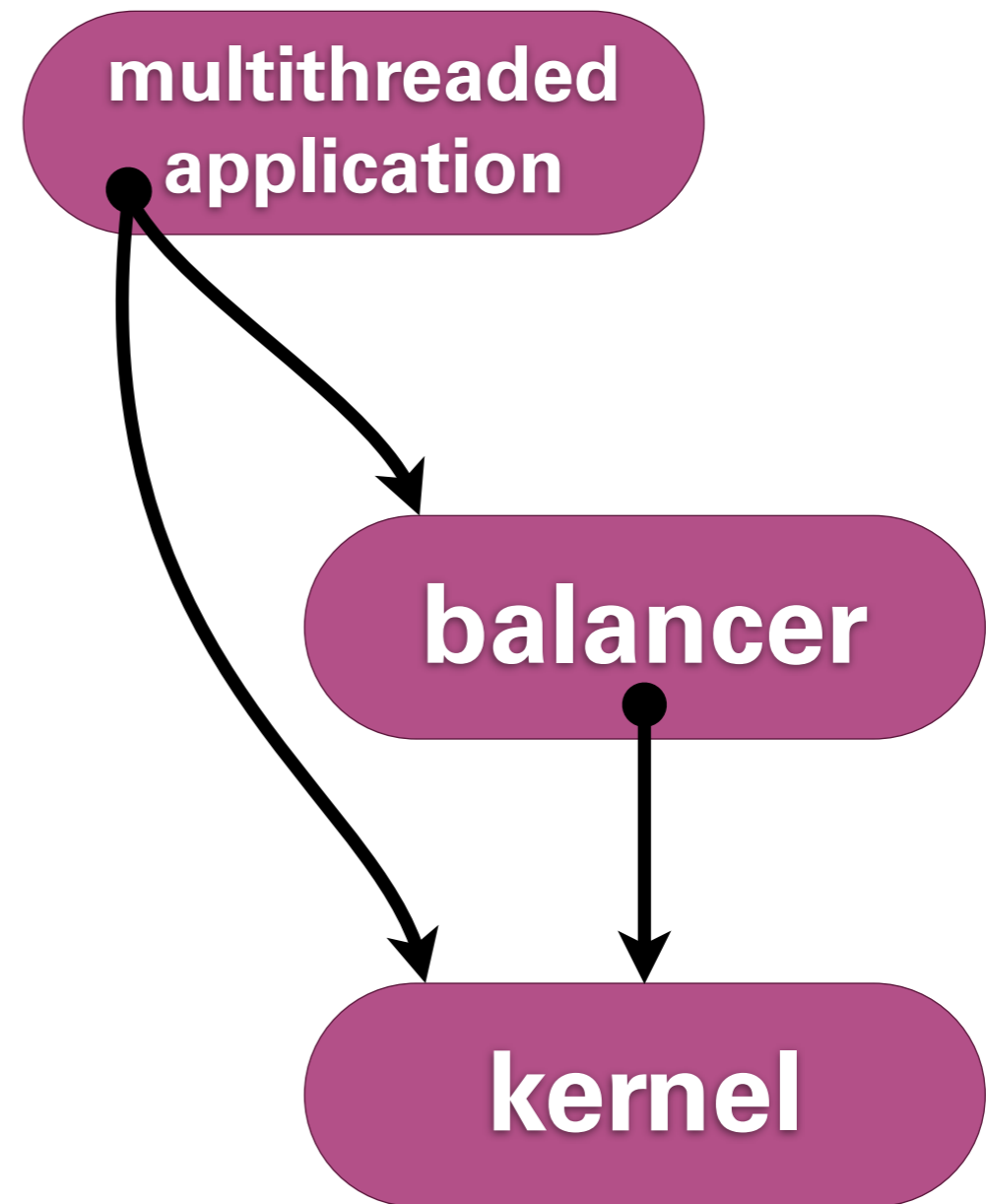
- hardware framebuffer is memory with side effect
- all memory is initially mapped to the root task
- **framebuffer driver**
 - find framebuffer memory
 - wrap in FB-interface
- same interface as mag's

- **virtualizable interfaces**
- L4Re uses one interface per resource
 - independent of the implementation
 - servers can (re-)implement any interface
- the kernel is a special server: provides low-level objects that need CPU privileges
 - minimal policy
 - userland servers can augment

Graphics



Thread scheduling



- all services provided as **objects**
- uniform access control with **capabilities**
- **invocation** is the only system call
- virtualizable: all interfaces can be interposed
- resource refinement and multiplexing transparent to clients

- kernel resource management
- basic resource management concepts
 - resource containers
 - exokernel
 - multiserver
- management details for specific resources
- object capabilities and virtualizable interfaces