The B-Method for the Construction of Microkernel-Based Systems

Sarah Hoffmann (STMicroelectronics)

Julien Millot (Clearsy)

sarah.hoffmann@st.com

B2007, Besançon, 17-19 January 2007

The B4L4 Project

Goal: formally prove fundamental properties of operating systems

- started as case study to formally define API of L4 microkernel
- extended to formally prove fundamental security properties of a complete L4-based system

Outline

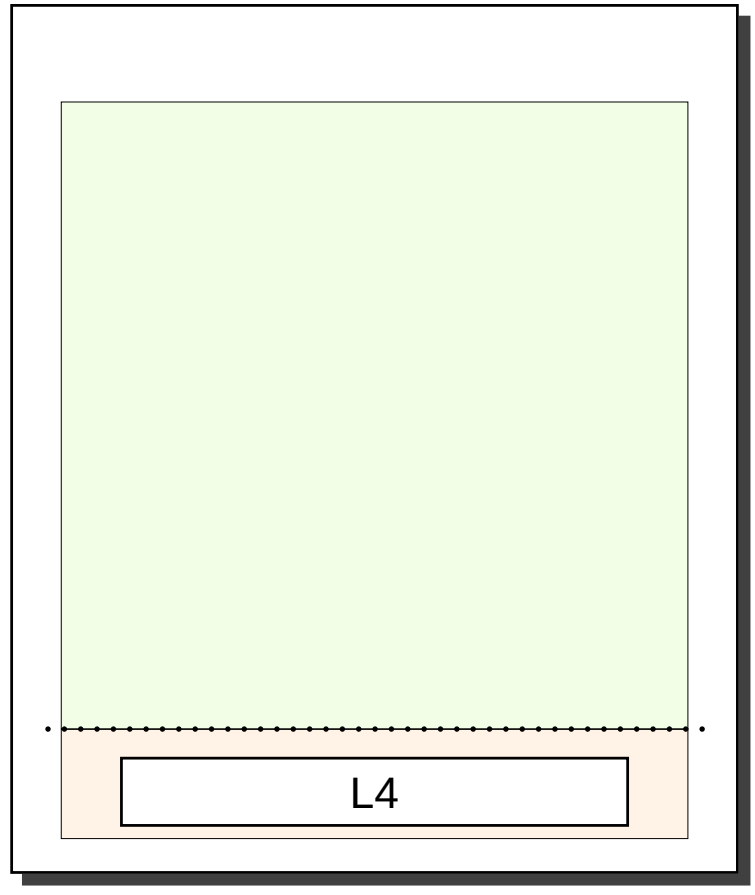
- Introduction to L4
- Model of an L4-based secure system (L4 Core)
- B Animation for test case generation

L4 Microkernel

- minimium of code in privileged mode
- abstraction of hardware
- no policy

ST-modified version of L4 X.2:

- *11 system calls*
  + protocols for page faults, interrupts, . . .
- *15.000 LOC*



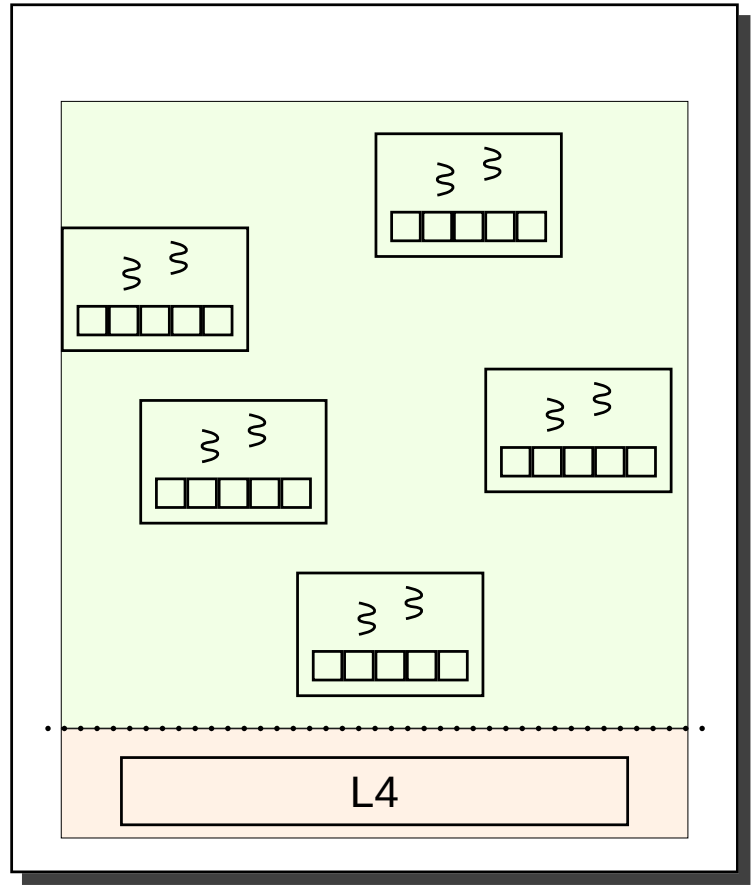
L4 Kernel API(I)

Tasks - address spaces

- virtual memory implementation using MMU
- container for all other objects (threads)

Threads - execution context

- priority-based preemptive round-robin scheduling



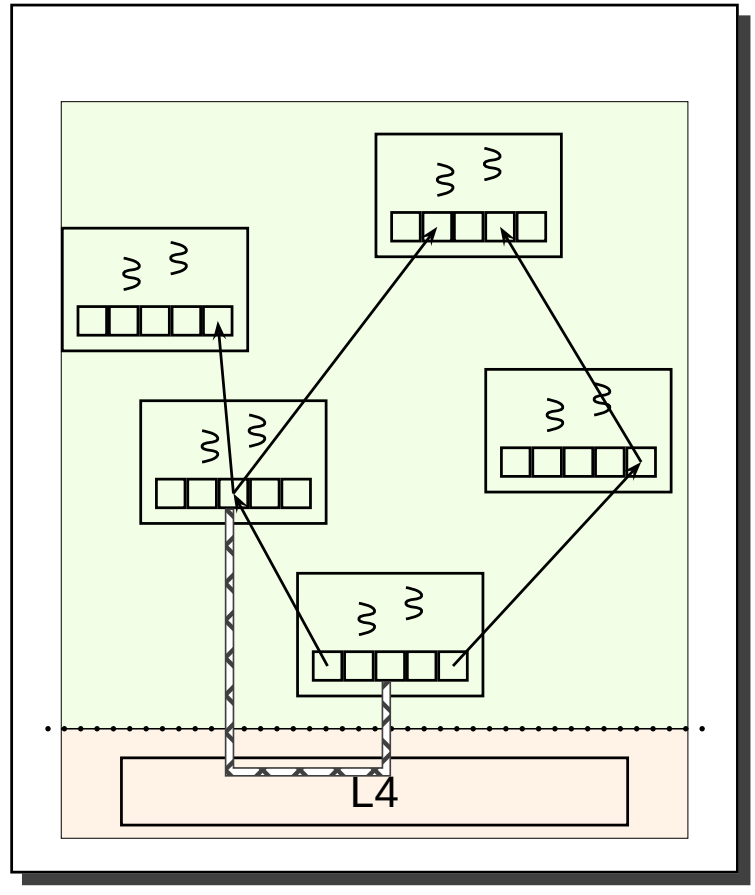
L4 Kernel API(II)

IPC - thread communication

- untyped, synchronous
- generalizes interrupts, page faults, exceptions

Mapping - memory management

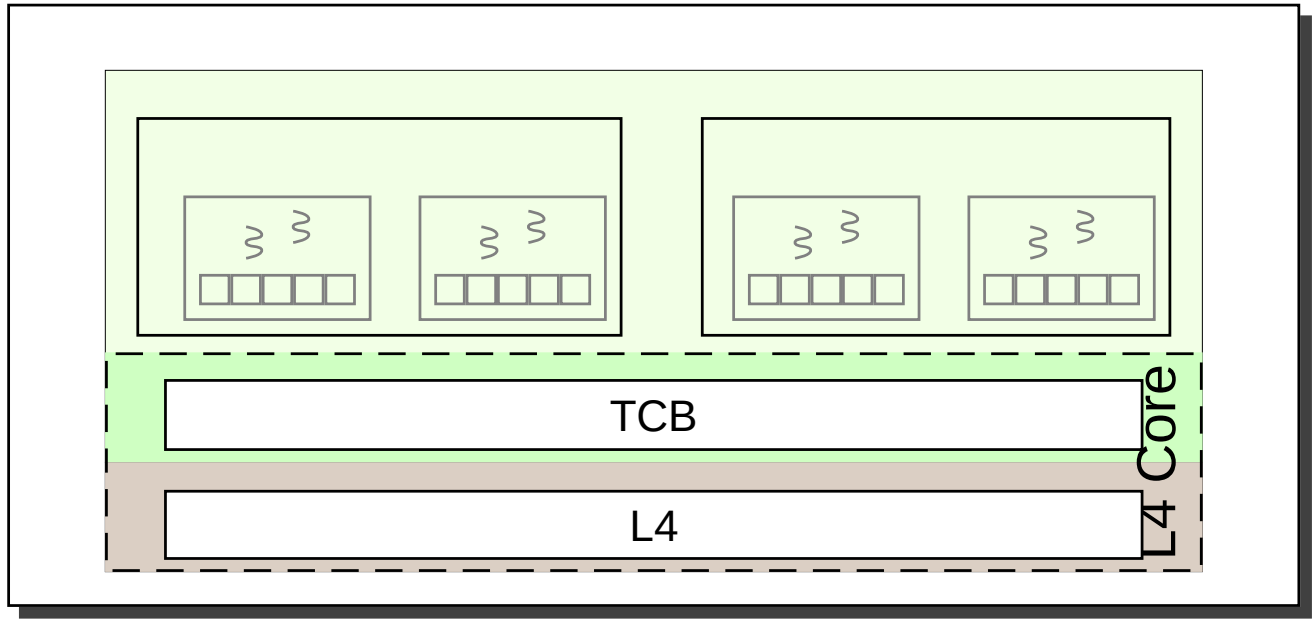
- decentralized memory transfer
- results in mapping trees for each page



Extension of the L4 Kernel

L4 Core

- add layer introducing policy and basic resource management
- flatten task structure to avoid uncontrollable dependencies between tasks



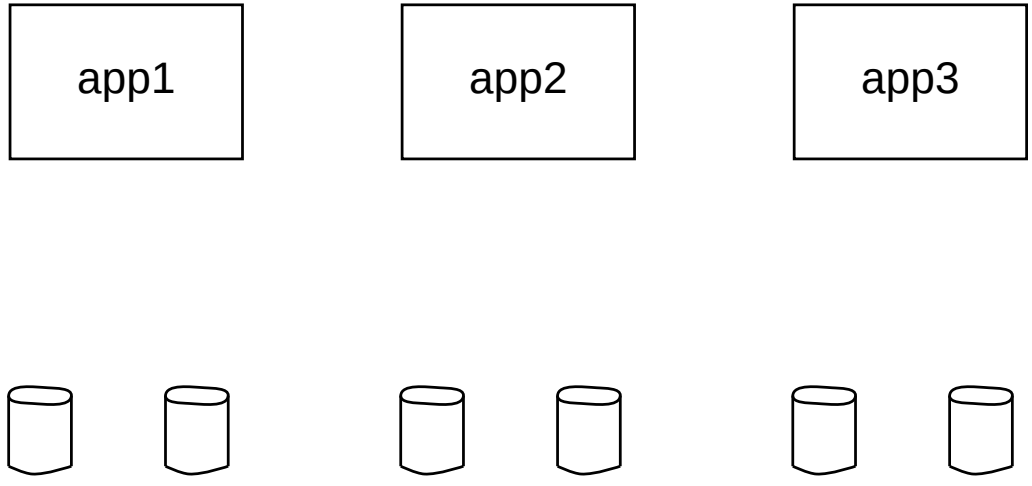
Security Properties

- from system model that fulfills basic system properties

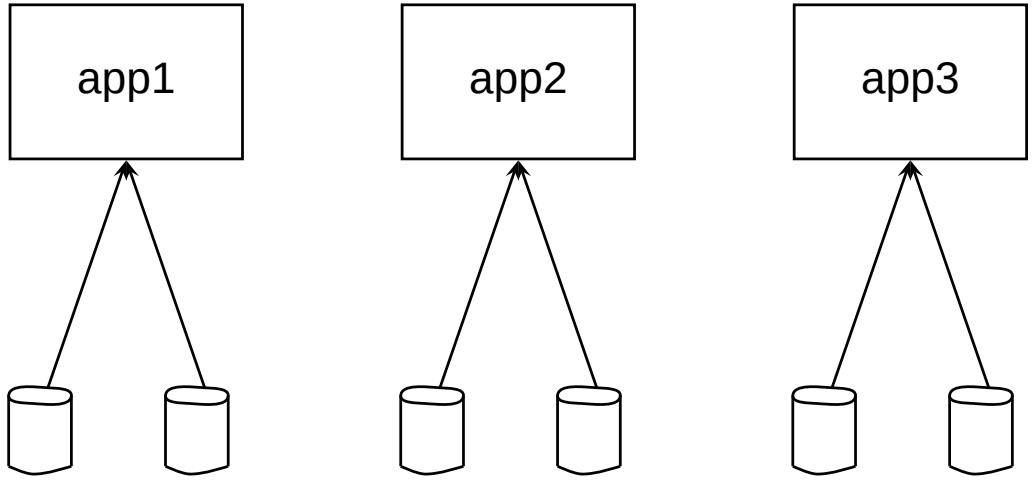
1. Every piece of information has an application as owner.
2. Information is only seen by other applications if the owner decides to share it.
3. The establishment of all communication channels is controlled by the Core.

- to refined model of an L4-based multi-task system

Core Security Model

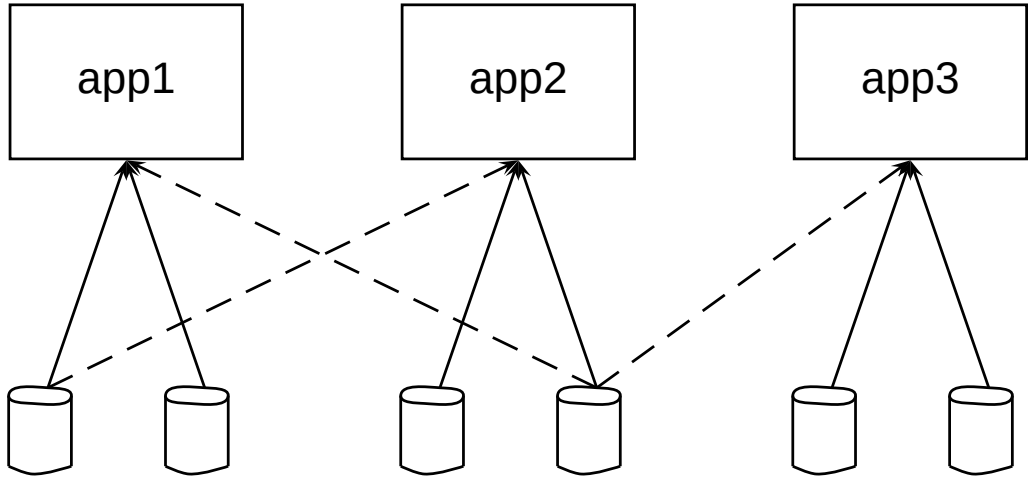


Core Security Model



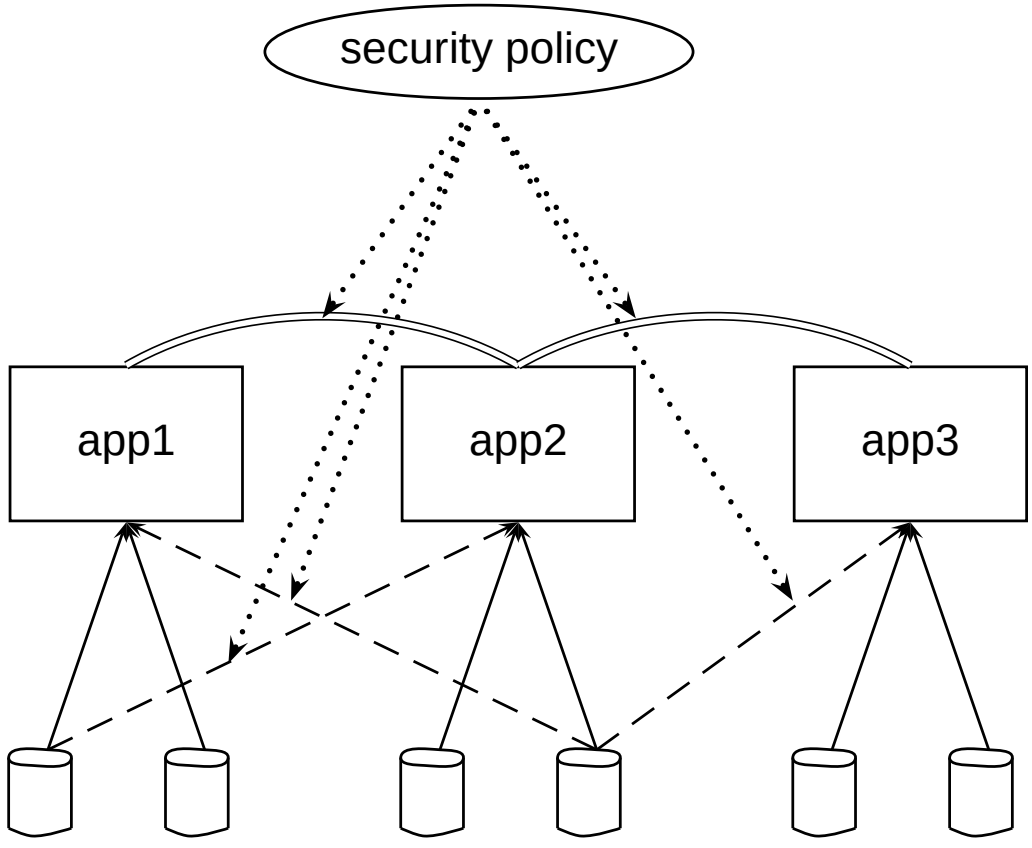
Every piece of information has an application as owner.

Core Security Model



Information is only seen by other applications if the owner decides to share it.

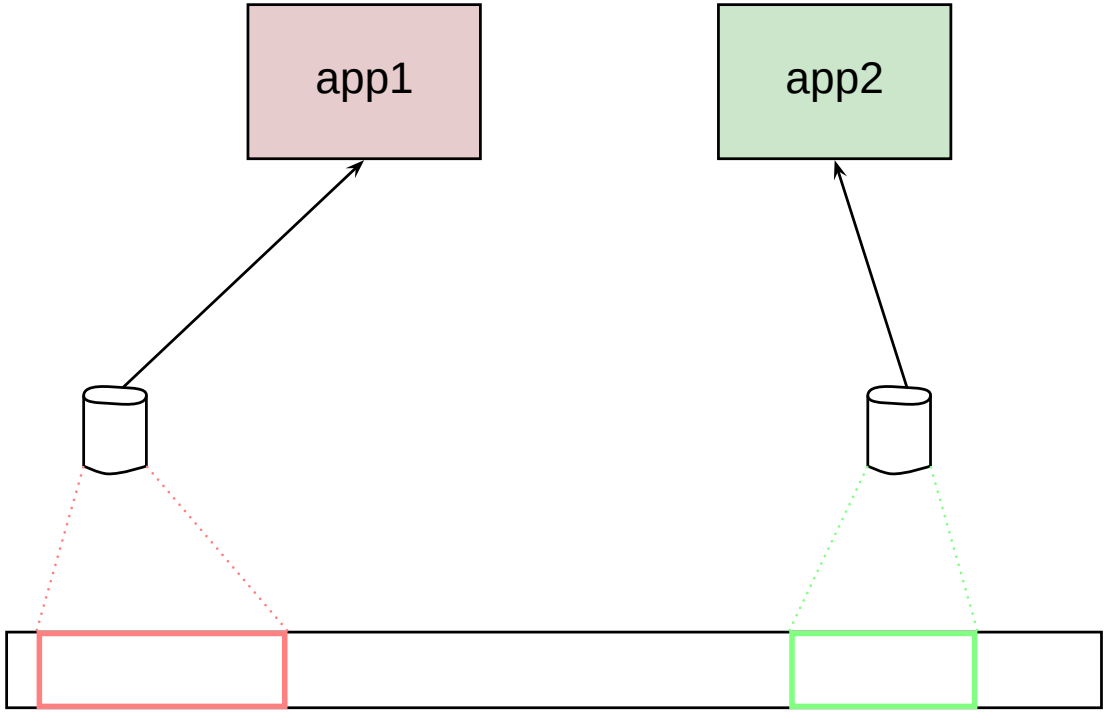
Core Security Model



The establishment of all communication channels is controlled by the Core.

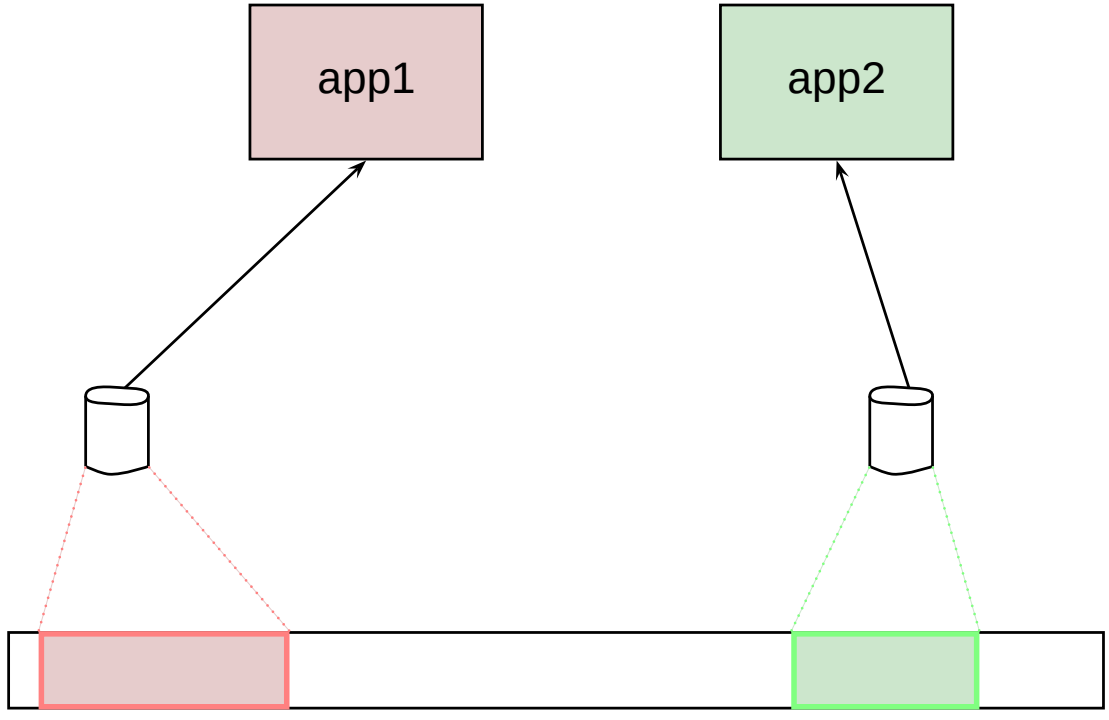
Memory Refinement

Piece of information => byte in physical memory



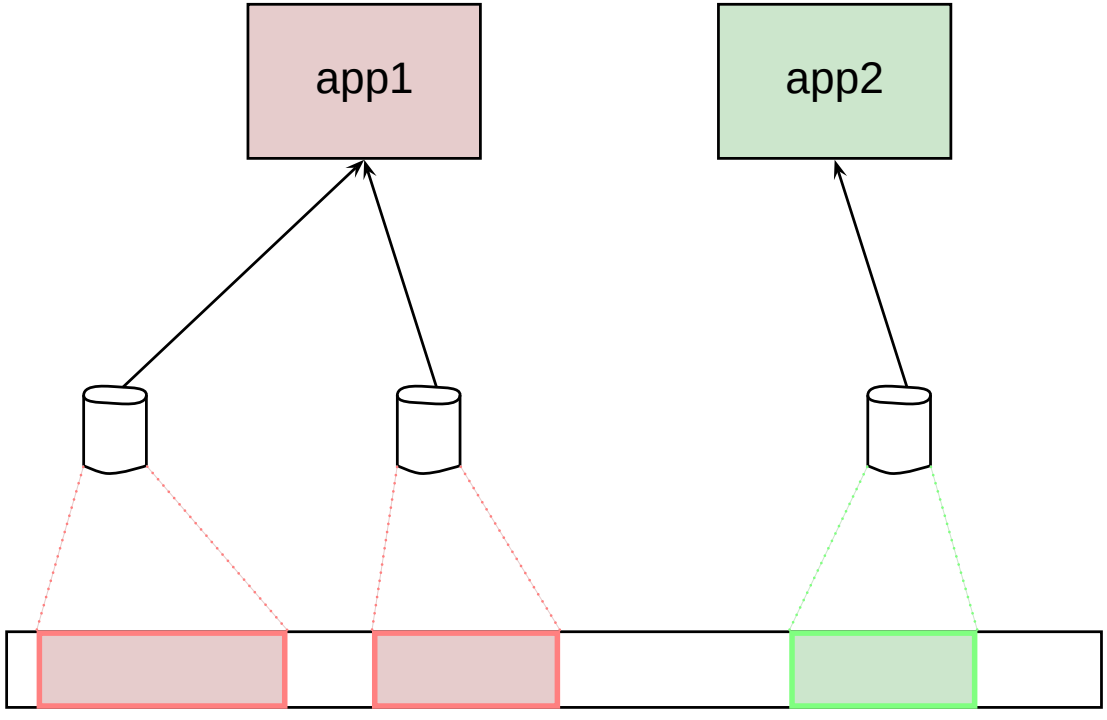
Memory Refinement

mappable = memory accessible by application



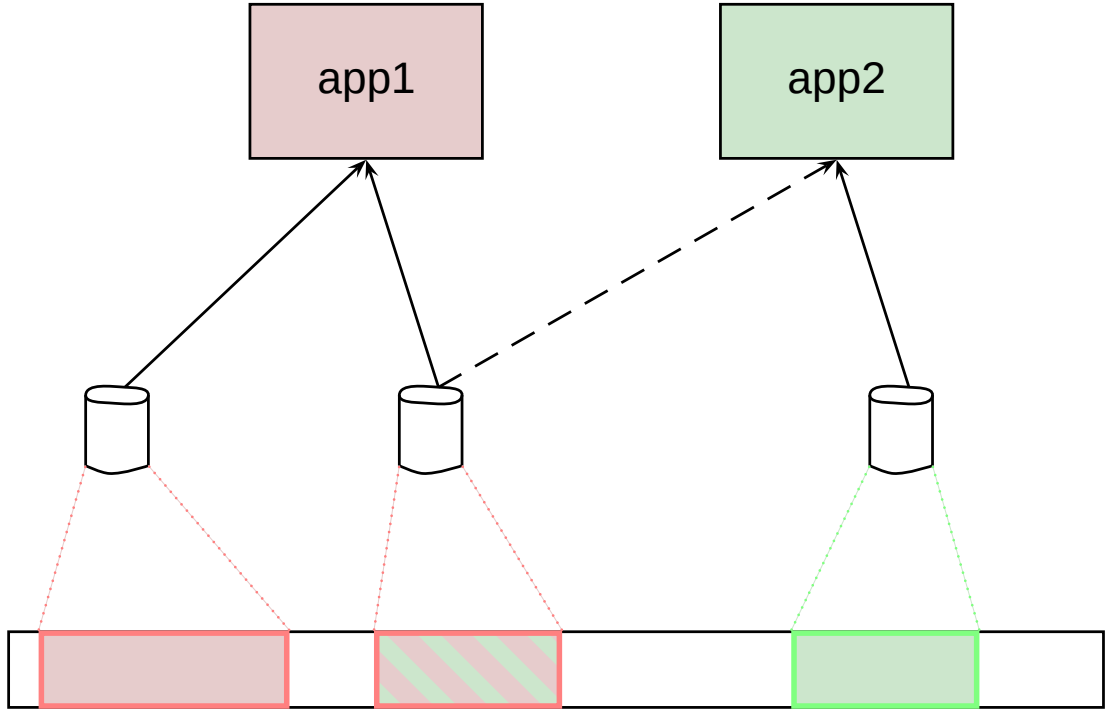
Memory Refinement

creating new information: app1 is owner and mappable



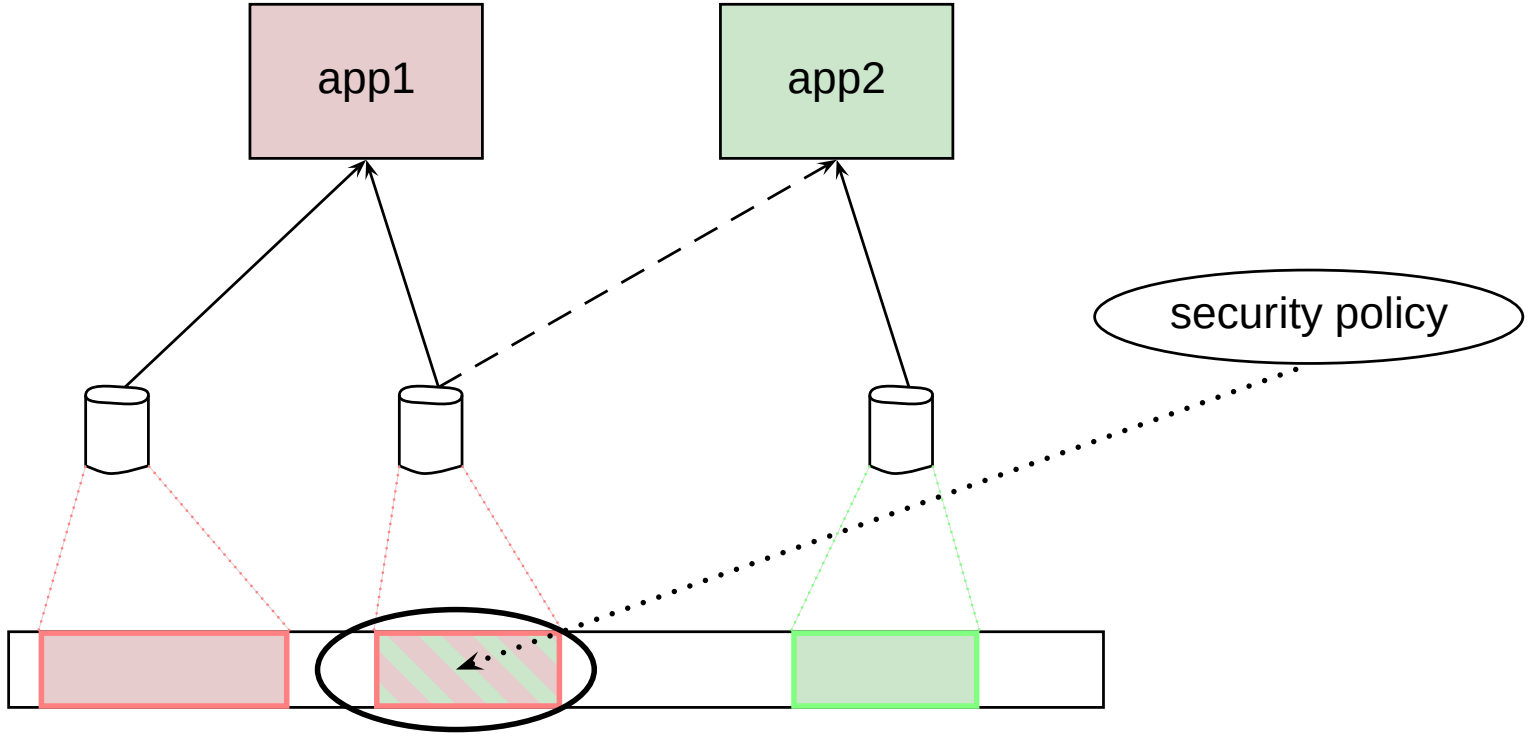
Memory Refinement

sharing this information: app1 allows app2 to be mappable



Memory Refinement

sharing this information: new communication via mapping



Communication mapping,
if intersection of mappable domain of app1 and app2 is not empty

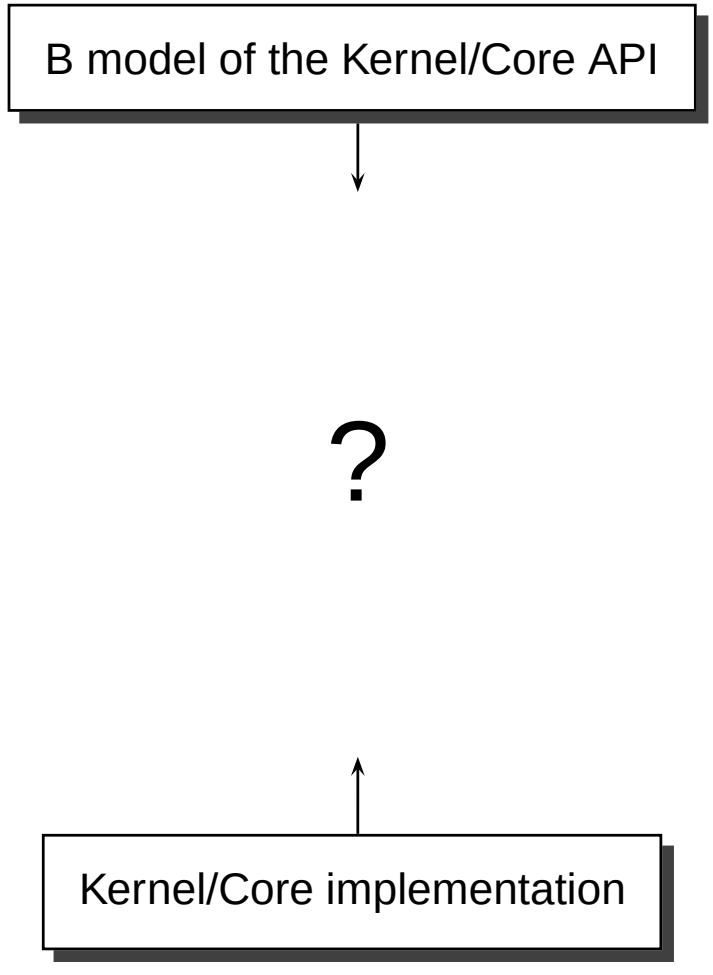
Further Refinements

- virtual memory
- hardware pages
- read/write access rights
- handle management
- compartments/tasks/threads/capabilities
- services
- binaries
- quotas

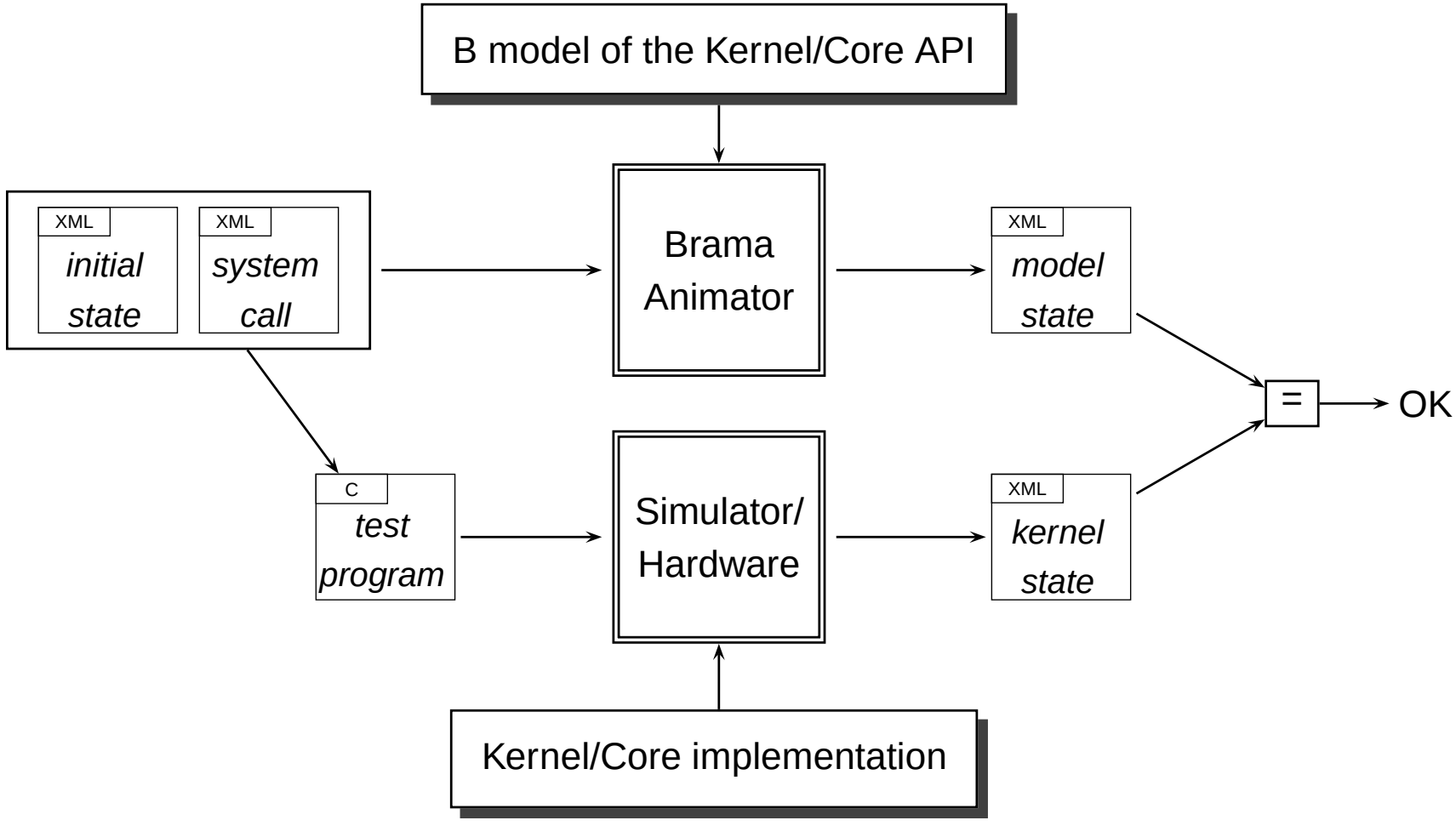
Core Model: Status

- modeled down to API level
  - 12 levels of refinement
  - 25 state variables
  - 60 events
  - final refinement: 2500 lines of code
- ongoing
  - multiple servers (refinement to code level)

Animation of the B Model



Animation of the B Model



Conclusion

- API-level formalization of micro-kernel systems possible
  - verification of completeness
  - help for construction of test cases
  - allowing higher level of certification
- application of event B in complex real-life model