L4.sec Implementation

Kernel Memory Management

Bernhard Kauer
kauer@os.inf.tu-dresden.de

Diplomverteidigung 2005-05-20

Motivation

Why L4.sec?

Motivation

Why L4.sec?

L4 does not have a security model and hence needs one.

Hermann Härtig

Lack of security

- kernel-resource management
- IPC control

New Concepts

L4.sec - a new L4 API?

aim

Build an improved L4 that allows to

- implement access control
- manage kernel memory
- flexible communication

New Concepts

L4.sec - a new L4 API?

aim

Build an improved L4 that allows to

- implement access control — Capabilities
- manage kernel memory — Kernel-Memory objects
- flexible communication — Endpoints

New Concepts

L4.sec - a new L4 API?

aim

Build an improved L4 that allows to

- implement access control — Capabilities
- manage kernel memory — Kernel-Memory objects
- flexible communication — Endpoints

previous state

- L4.sec ideas
- but: Implementation missing

Capability

What are capabilities?

Capability

What are capabilities?

definition

Capability = Object + Permissions

Objects	thread, task, endpoint, kernel-memory object, ...
Permissions	object dependent rights like read, write or send, recv

Capability

What are capabilities?

definition

Capability = Object + Permissions

Objects	thread, task, endpoint, kernel-memory object, ...
Permissions	object dependent rights like read, write or send, recv

capability space

- translate task local number to capabilities
- extension of the recursive address space model
  - `map`
  - `unmap`

Kernel-Memory Objects

How to handle kernel memory?

Kernel-Memory Objects

How to handle kernel memory?

requirements

- managing the used kernel-memory
- no predefined amount of kernel memory
- map and unmap kernel-memory

Kernel-Memory Objects

How to handle kernel memory?

requirements

- managing the used kernel-memory
- no predefined amount of kernel memory
- map and unmap kernel-memory

solution

- creating kernel-memory objects *converts* user-pages
- converted pages are inaccessible
- objects are *created* with kernel-memory (objects)
- reconvert at *unmap* of user-page or kmem object
- delete all objects within it at reconvert

Delete Objects

Dependencies

Delete Objects

Dependencies

delete named objects

- last capability removed
- memory revoked

Delete Objects

Dependencies

delete named objects

- last capability removed
- memory revoked

$\Rightarrow$ have to delete all dependent objects

Delete Objects

Dependencies

delete named objects

- last capability removed
- memory revoked

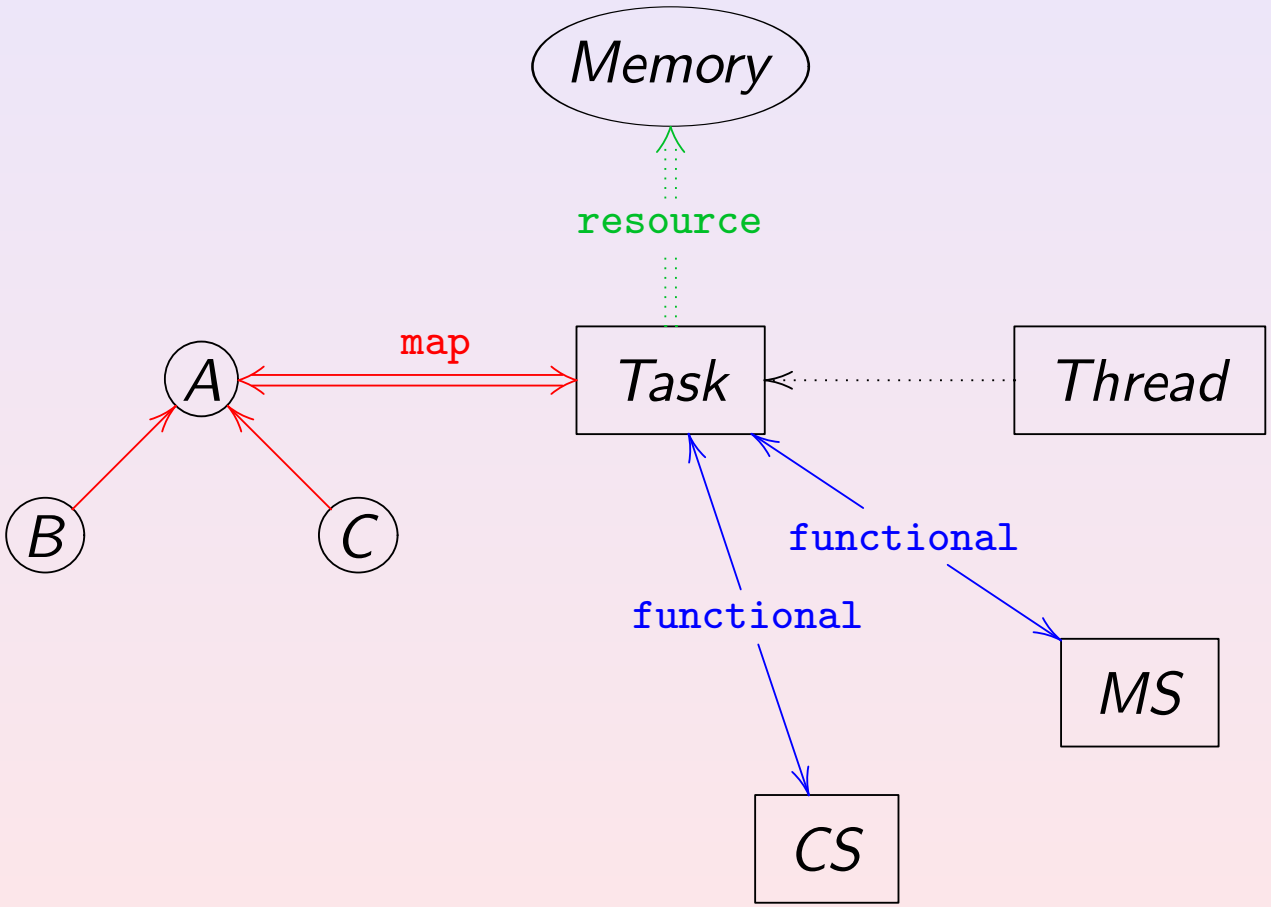
⇒ have to delete all dependent objects

Dependencies

Resource	Who provides the resources for this object?
Functional	What needs this object to work?
Map	Where it is mapped?

Dependencies

Example: Task



Dependencies

Iterating

Dependencies

Iterating

Avoid cycles

- directed dependency graph
- only a tree walk needed
- simplifies locking and implementation

Dependencies

Iterating

Avoid cycles

- directed dependency graph
- only a tree walk needed
- simplifies locking and implementation

Solutions

Bind	thread to task
Collocate	named object and mapping node
Redirection	change the pointers

The W5 binding

A syscall binding for L4.sec

The W5 binding

A syscall binding for L4.sec

idea

- parameters in virtual registers
- param words as abstraction for in- and output-parameters
- up to 255 param words could be used
- transfer at least 5 param words in registers

The W5 binding

A syscall binding for L4.sec

idea

- parameters in virtual registers
- param words as abstraction for in- and output-parameters
- up to 255 param words could be used
- transfer at least 5 param words in registers

properties

- used with `iret`, `sysenter`, `syscall`
- architecture independently
- fast and simple

Current state

What is working?

Current state

What is working?

missing

- Long-IPC
- fine-grained locking
- native ia32 and interrupts

Current state

What is working?

missing

- Long-IPC
- fine-grained locking
- native ia32 and interrupts

working

- send messages and page a task
- convert user-pages and create objects
- mapping capabilities and unmapping objects

Evaluation

Code complexity

Evaluation
Code complexity

Number of SLOC for the *UX*-versions:

name	L4.V2	L4.sec	difference
kernel	17805	14421	−3384
jdb	14656	3950	−9626
unit tests	400	1762	+1362
kernel memory and allocators	1521	972	−549
mapping database and spaces	1994	2614	+620
thread and syscalls	4347	2265	−2082
api	1067	889	−178
reused code	7287	6985	−302
others	1589	696	−893

Conclusion

The important thinks

Conclusion

The important thinks

experiences

Three quarter of the code is present

- similar code complexity
- fundamental change of Fiasco
- it is feasible to build L4.sec

Conclusion

The important thinks

experiences

Three quarter of the code is present

- similar code complexity
- fundamental change of Fiasco
- it is feasible to build L4.sec

conclusion

- problem
- idea
- implementation
- application?

Conclusion

The important thinks

experiences

Three quarter of the code is present

- similar code complexity
- fundamental change of Fiasco
- it is feasible to build L4.sec

conclusion

- problem
- idea
- implementation
- application?

Endpoint

What is an endpoint?

communication channel

- replaces direct IPC to a given thread
- many threads could send and receive from an endpoint
- works over task boundaries

differences

- sender or receiver wait at an endpoint
- only open wait at a single endpoint possible
- a `call` uses two different endpoints