POCS 2021
Exokernel Recitation

Rishabh Iyer

30.09.2021

Paper Recap

The idea (as in the paper)

- Operating Systems multiplex and abstract H/W resources
- No universally good abstraction that OS can provide
- Exterminate all OS abstractions

The idea (in POCS terms)

- Kernel and the OS should be distinct modules
- Kernel
  - Multiplexes resources securely
  - Privileged operations
- OS
  - Provide high-level hardware-agnostic abstractions
  - Can be unprivileged (e.g., libOS, microkernel servers)

Benefits (as in the paper)

- Efficiency
  - Applications can implement their own abstractions
- Reliability
  - Smaller OS implies fewer bugs
- Flexibility
  - Empowers users, easy to modify/implement new abstractions

Benefits (in POCS terms)

- Decomposing a monolith into modules improves
  - Flexibility and Efficiency
    - Can add/remove/replace individual modules as desired
  - Reliability
    - Faults isolated in smaller modules

Discussion

Exokernels, μKernels and VMs

- Apart from the exokernel, what other OS designs are you aware of?

- How do exokernels differ from

  - μKernels?

  - Virtual Machines?

- Today VMs are everywhere. Why not exokernels?

Enforcing resource modularity

- What are *secure bindings*?
  - How does Aegis multiplex memory, network, CPU?
- Why does the exokernel provide visible revocation?
  - How is it implemented for the above resources?
- Why does the exokernel need an abort protocol?
  - How flexible can the abort protocol be?

Capabilities (basics)

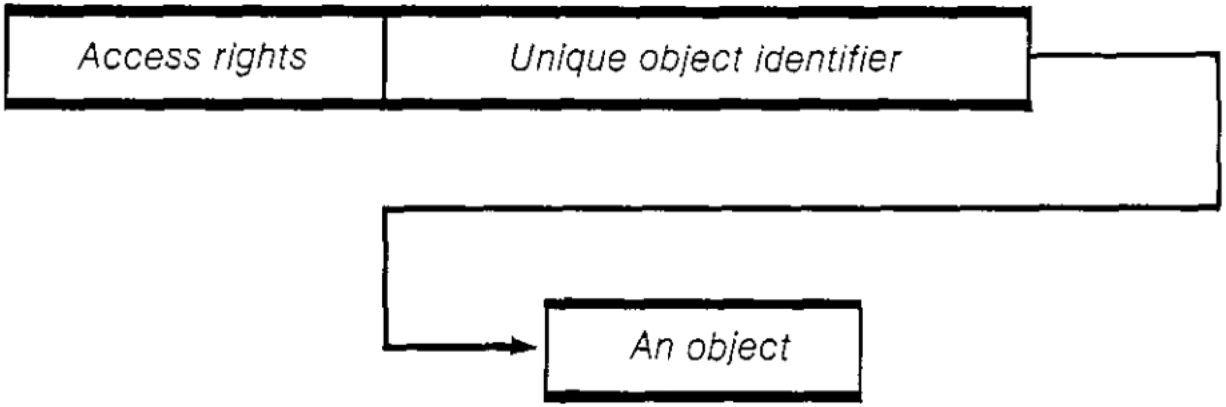
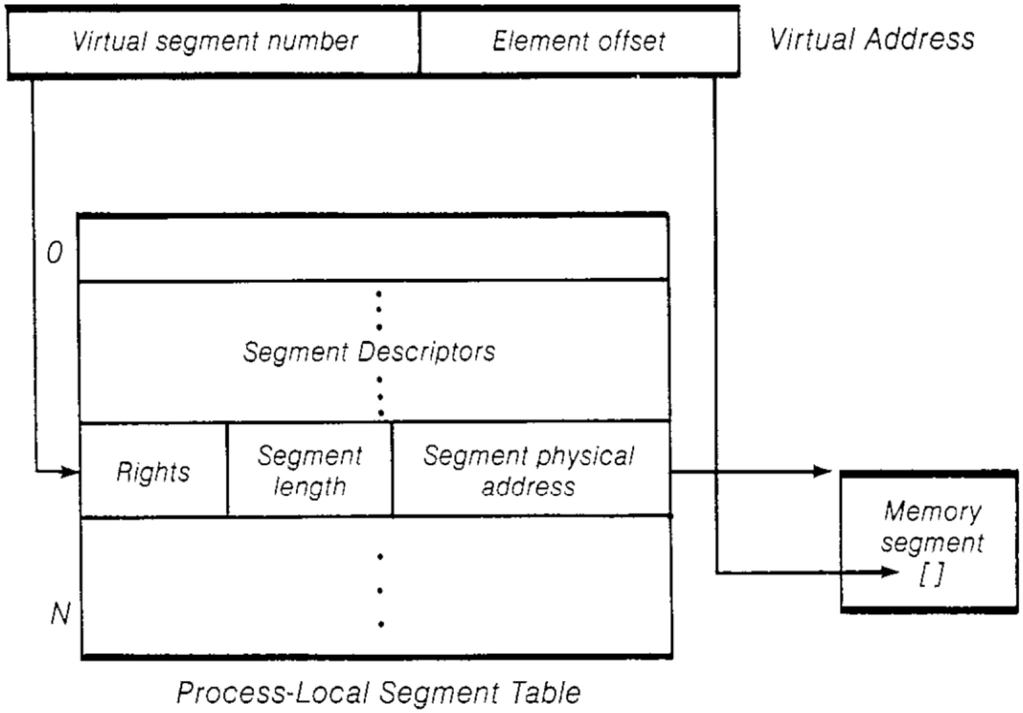


Figure 1-1: A Capability

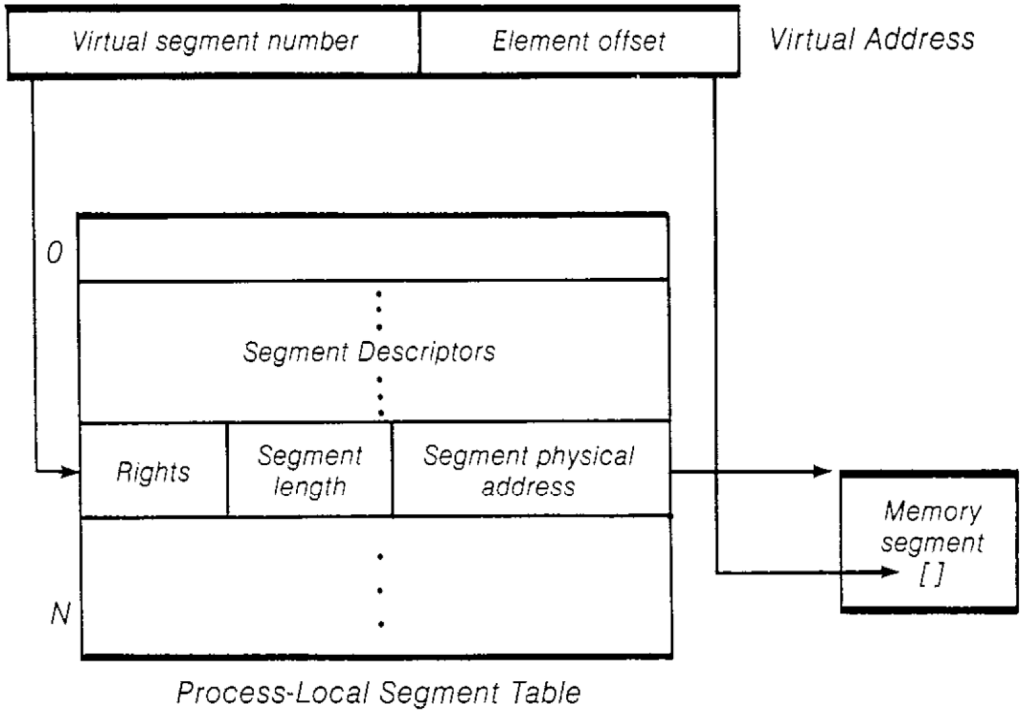
- Capabilities define an object and access rights to that object
- e.g., virtual memory area, read/write/execute permissions

Capabilities for physical memory



- Segment num indexes into the process-local segment table
- Length field in the entry checks that offset is within bounds
- Rights field stores the access rights.

Capabilities for physical memory



- Protecting capabilities
  - ❖ Writing to process-local segment table is a privileged instruction
  - ❖ Hardware capability registers for improved performance
    - Think equivalent registers to x86's CRs

Multiplexing physical memory in Aegis

- Aegis provides a few guaranteed mappings
  - e.g., exception handlers, page tables
- On TLB miss:
  - If guaranteed mappings -> Aegis handles miss
  - If ordinary user segment -> exception forwarded to process
    - TLB cache for improved perf
- Process requests installation of TLB entry using capability
  - No checking capability on data path
- Key enabler: Software managed TLB
  - How would you implement this for a HW-managed TLB?

Multiplexing the network

- Key question: Which process does this packet belong to?
- Easy to answer if kernel speaks networking protocols
  - But exokernel should not manage resources!
- Solution: Packet filters downloaded into the kernel
  - Does this break modularity?
- Can you think of similar concepts in today's OSes?

Talking PL

o Languages are moving to higher abstractions? Does this contradict exokernel principles?

Being precise about abstractions

- Can the CISC vs RISC debate be resolved using arguments similar to that in the exokernel paper?

Group design exercise

Pick one particular application and discuss how to redesign it to take advantage of the exokernel. Come up with **concrete** abstractions you would implement in your libOS to improve performance. You can assume a POSIX interface as the baseline

Backup

Changing assumptions

- If designing a kernel for machines that run a single app, what changes would you make to the exokernel?

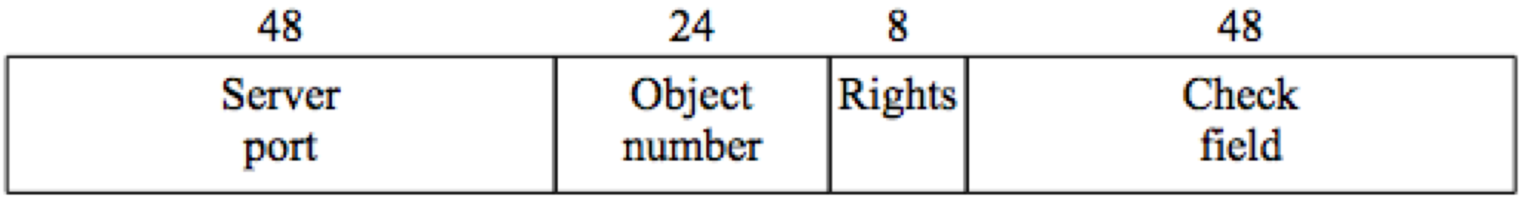
Will my app benefit from the exokernel?

- The paper has several examples of how porting an application to the exokernel and defining application-specific abstractions greatly improves performance. Can you think of applications for which this will not hold true?

Exokernel abstraction

- Is the exokernel the lowest level of abstraction an OS can provide? Can we go lower?
- Can we bake into HW a low-level of abstraction?

Cryptographically protected capabilities



48	24	8	48
Server port	Object number	Rights	Check field

- First described in the Amoeba OS [Tanenbaum '90]
  - Client-server based, object based OS
- Server port identifies client (process)
- Object number indexes into server's cap table
- Check field is one-way function of rights XOR random number