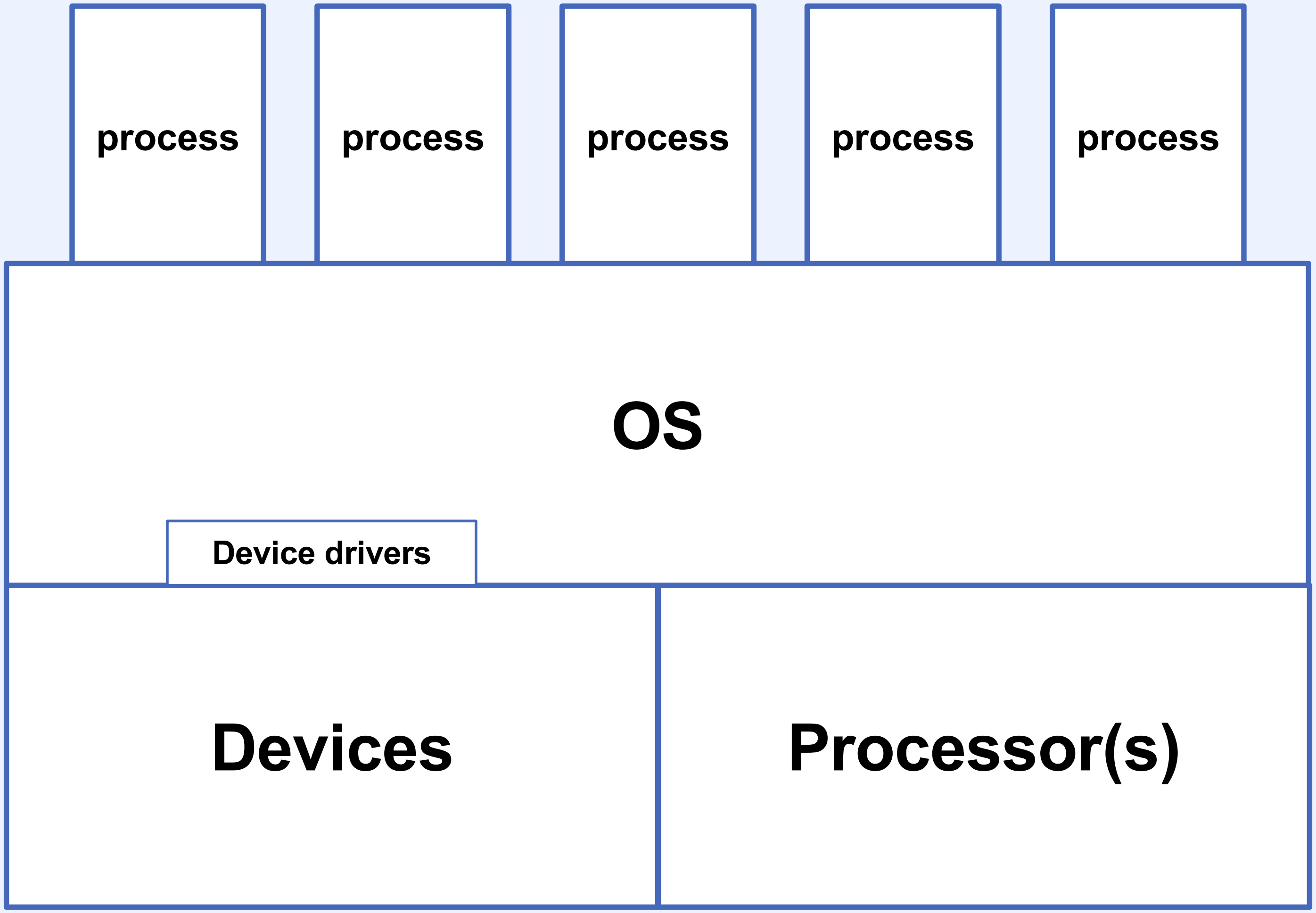
Virtual Machines

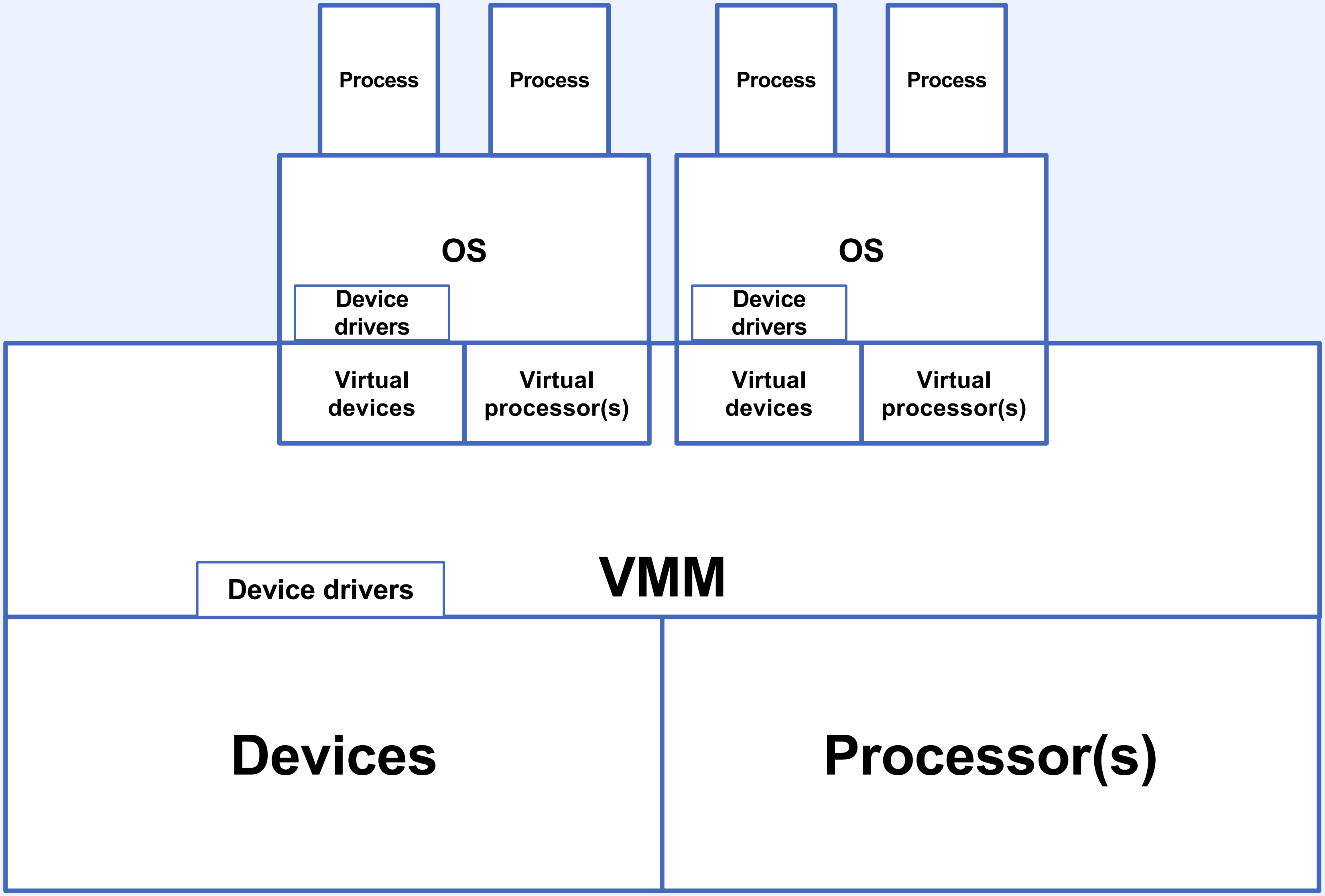
Part 2: starting ~20 years ago

(continued)

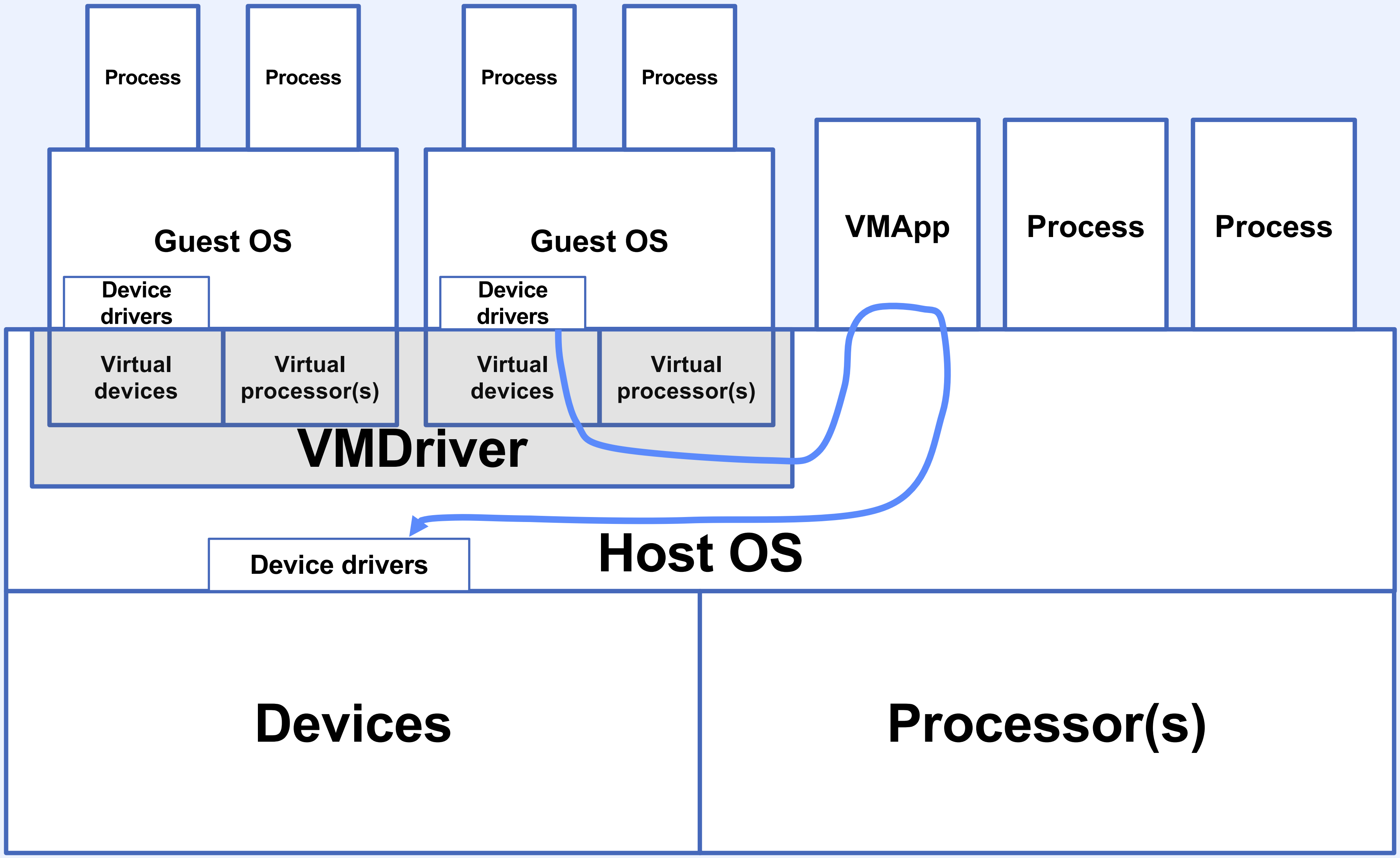
Real-Machine OS Structure



On a Virtual Machine …



VMware Workstation



Quiz 1

The host OS contains a scheduler that multiplexes the execution of threads on its processors. Each guest OS contains a scheduler that does the same thing.

Assume all threads have the same priority. Assume just one real processor.

a) Threads on the guest OS compete equally for the real processor as do threads on the host OS

b) Threads on the guest OS effectively have higher priority than host OS threads

c) Threads on the guest OS effectively have lower priority than host OS threads

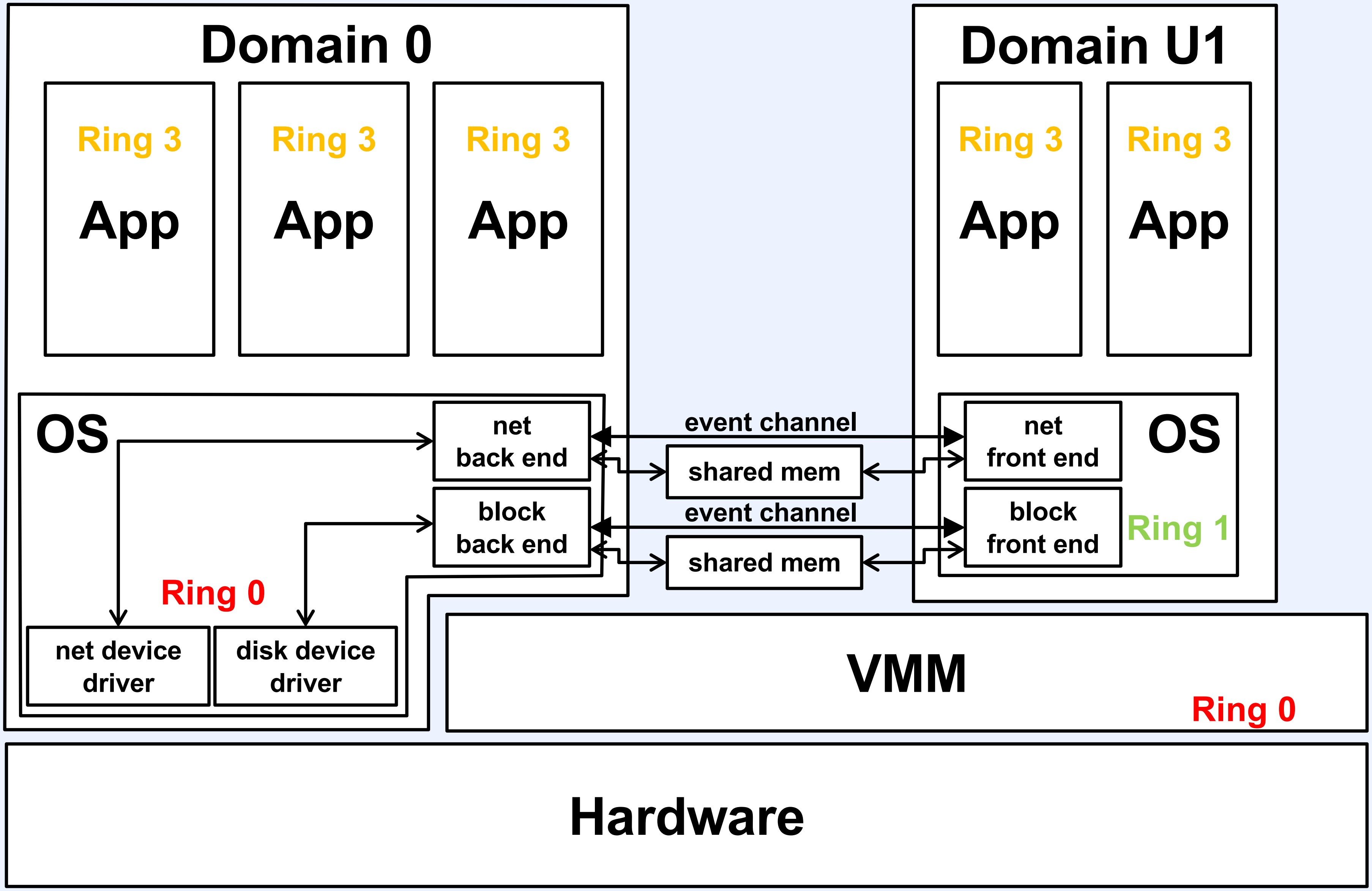
KVM/QEMU

- **KVM**
  - kernel virtual machine monitor for Linux
  - uses VMX technology (or AMD equivalent)
- **QEMU**
  - generic and open source machine emulator and virtualizer
  - does binary rewriting and caching as does VMware
  - emulates I/O devices as well
- **KVM/QEMU**
  - code executes natively until VM-exit
  - user-space QEMU code does I/O emulation

Paravirtualization

- Sensitive instructions replaced with hypervisor calls
  - traps to VMM
- Virtual machine provides higher-level device interface
  - guest machine has no device drivers

Xen

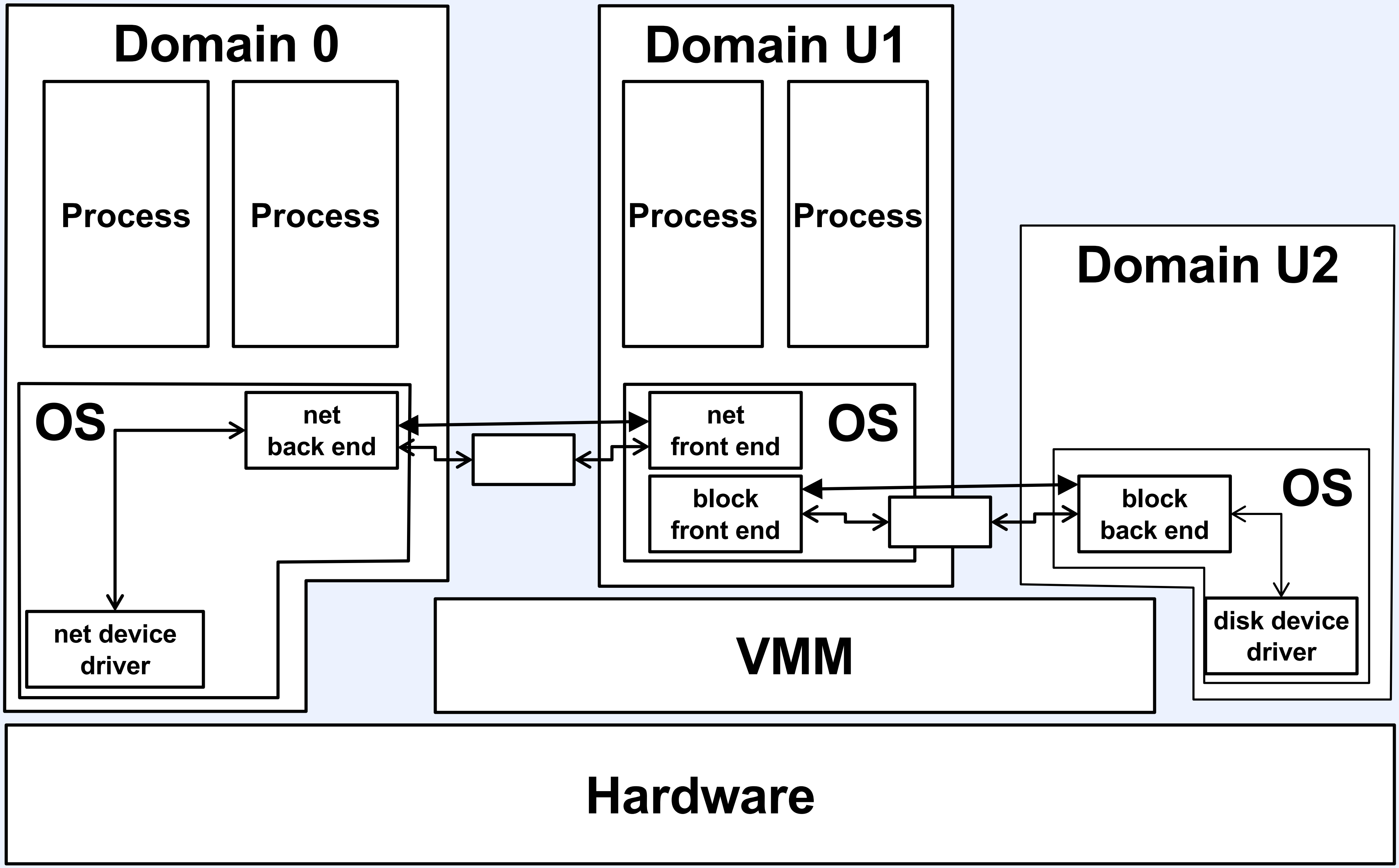


Domain 0
Ring 3
App
Ring 3
App
Ring 3
App
OS
net
back end
block
back end
Ring 0
net device
driver
disk device
driver
event channel
shared mem
event channel
shared mem
Domain U1
Ring 3
App
Ring 3
App
net
front end
OS
block
front end
Ring 1
VMM
Ring 0
Hardware

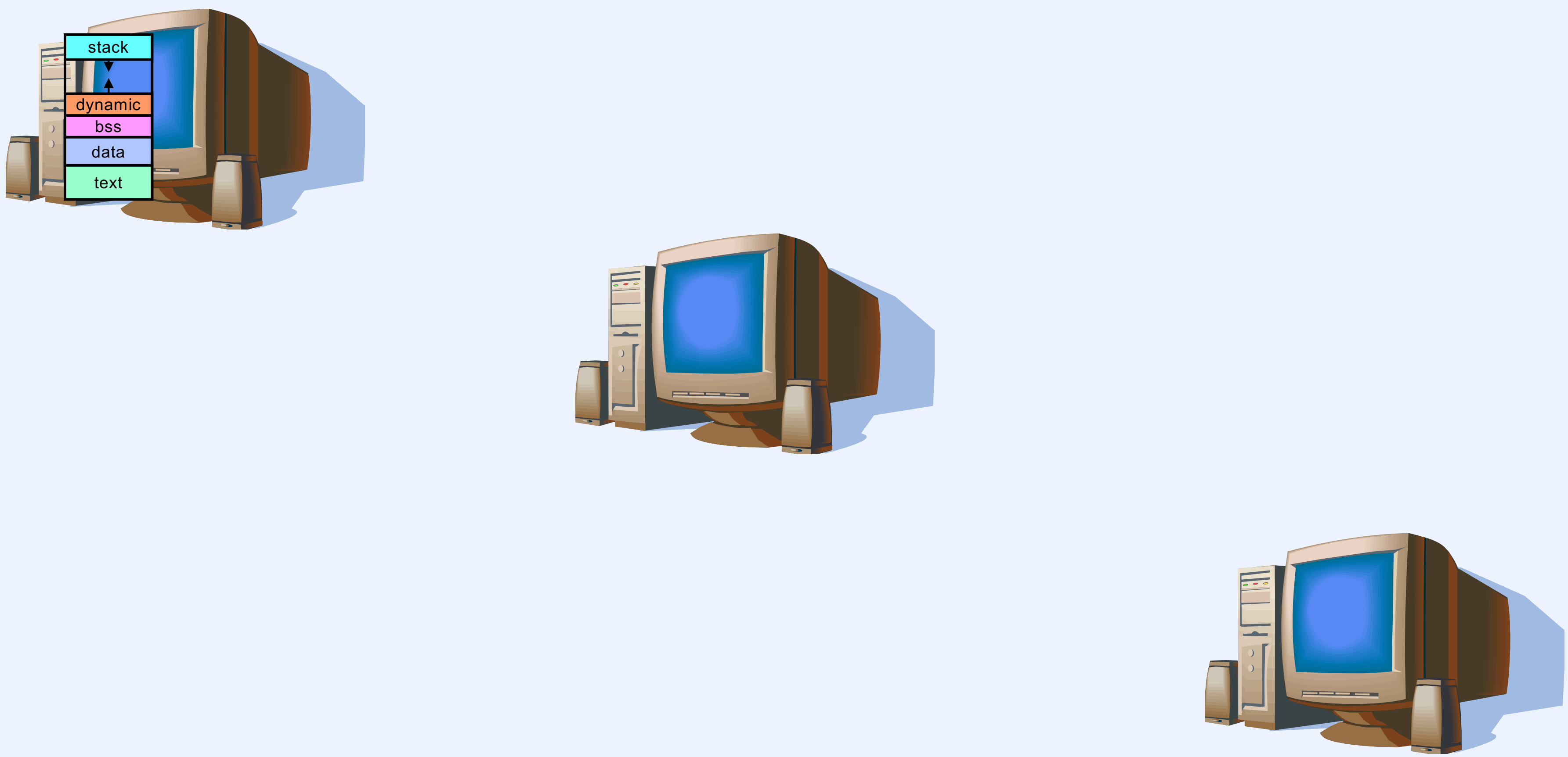
Additional Applications

- **Sandboxing**
  - isolate web servers
  - isolate device drivers
- **Migration**
  - VM not tied to particular hardware
  - easy to move from one (real) platform to another

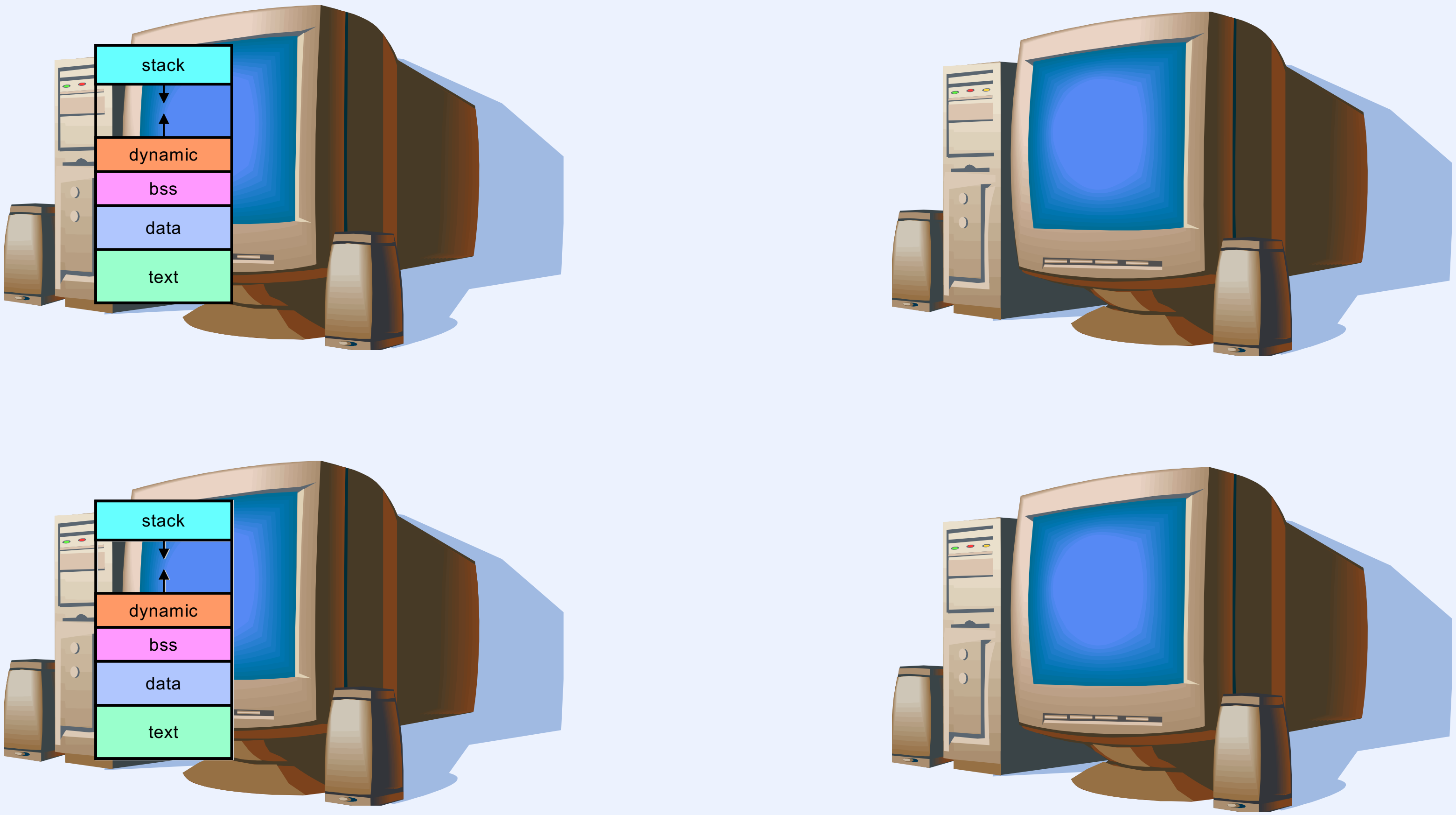
Xen with Isolated Driver



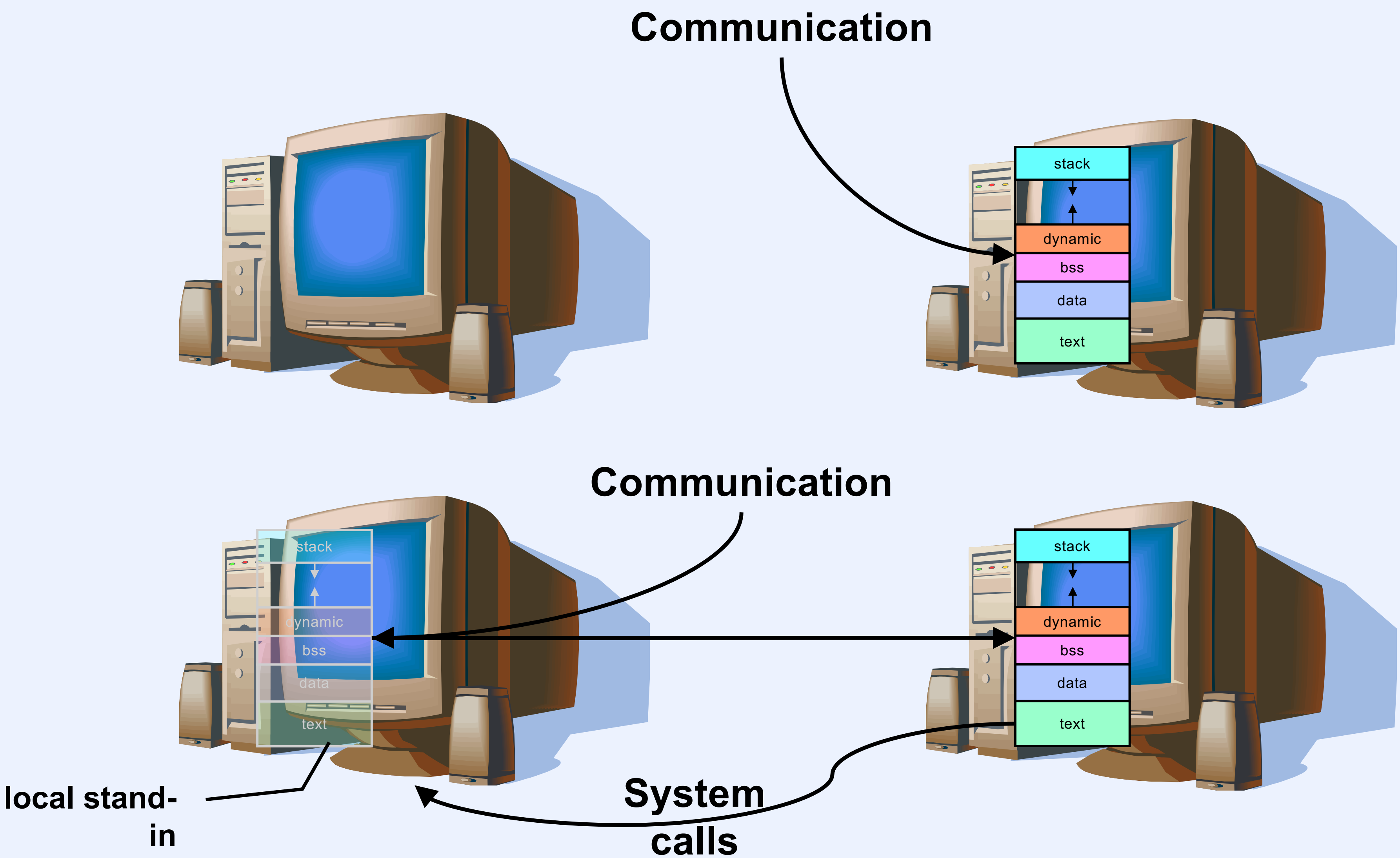
Process Migration



Approaches: Before



Approaches: After



Virtual-Machine Migration

- Virtual machines are isolated
  - by definition!
- State is well defined
  - thus easy to identify and move
  - possible exception of virtual memory

Transferring Virtual Memory

- Eager
  - all
  - dirty
    - (clean pages come from common source)
- Lazy
  - copy on reference
- Straightforward
  - flush everything to file system on source, then access file system on target
- Weird
  - precopy

Eager–Dirty

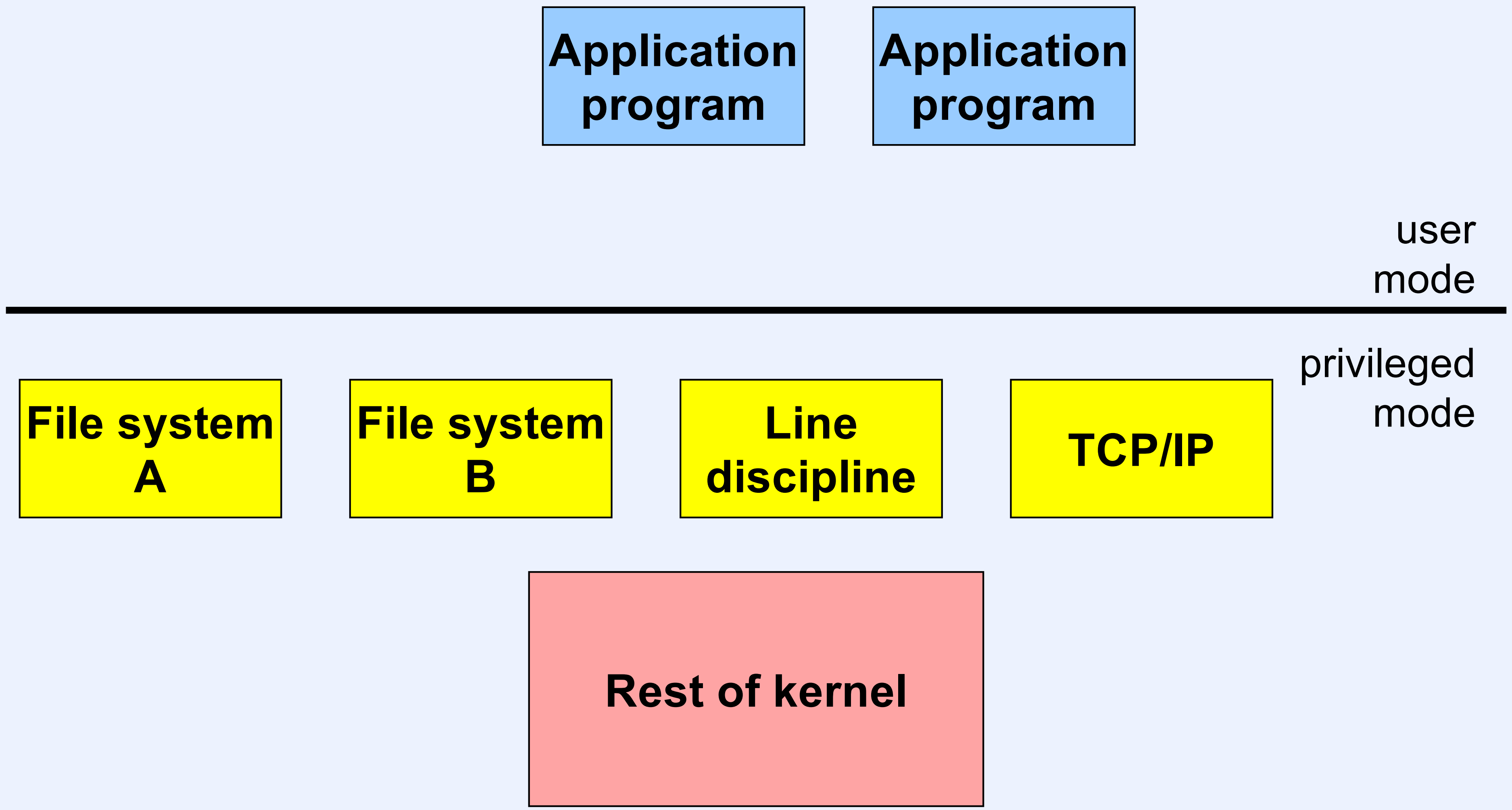
- Freeze process on source
- Transfer all dirty pages to target
- Resume process on target

Precopy

- While process still running on source
  - transfer dirty pages to target (eager–dirty)
- While more than x pages dirty on source
  - transfer newly dirtied pages to target
- Freeze process on source
- Transfer remaining dirty pages to target
- Resume process on target

Microkernels

Traditional OS Organization



Quiz 2

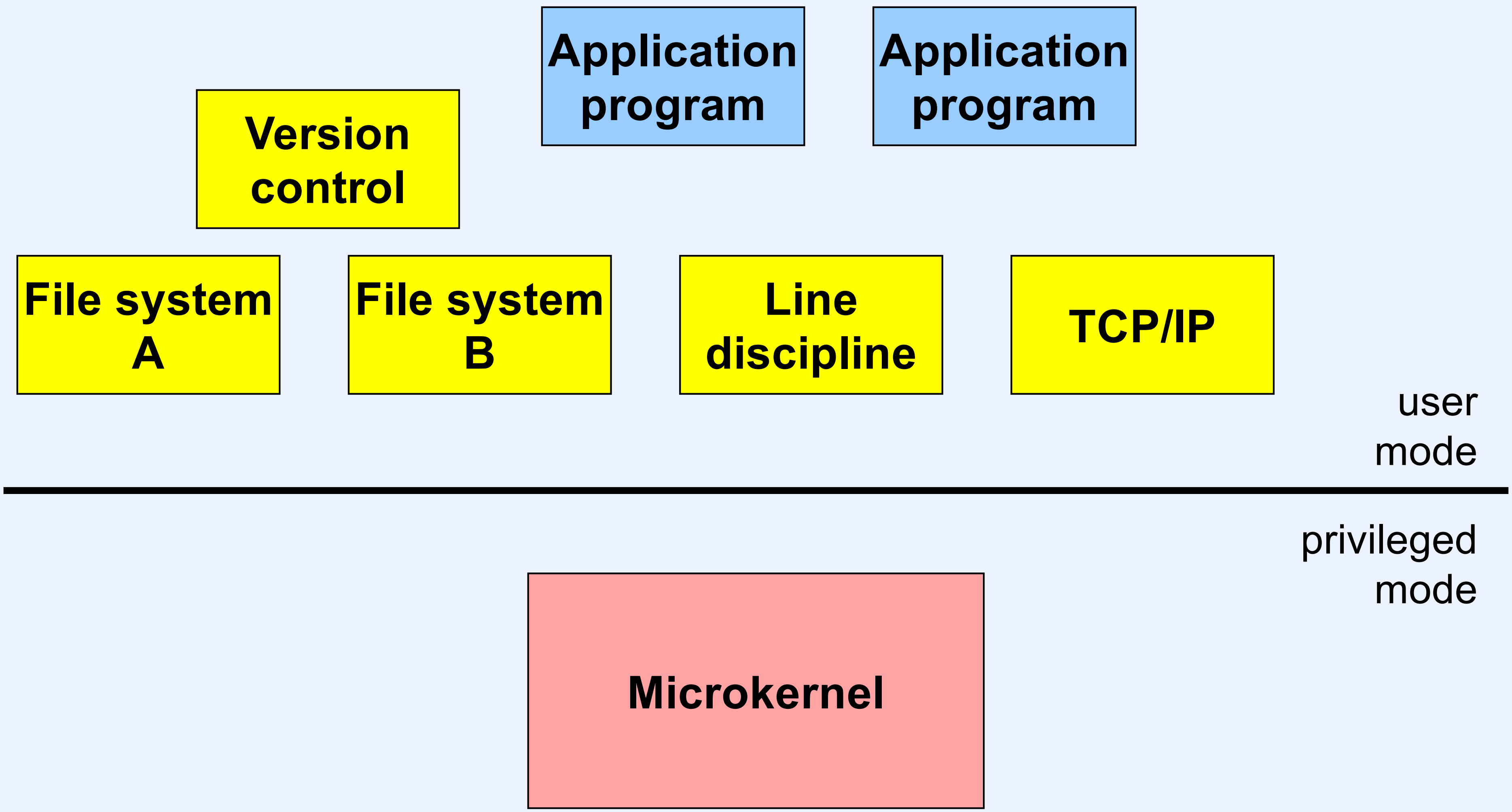
In the previous slide, assume each of the two application programs runs as a separate process. What's in the slide employs:

a) two address spaces: one for each process, with the kernel existing in a shared portion of the two process address spaces

b) three address spaces: one for each process, and one for the kernel

c) six address spaces: one for each process, one for each of the four kernel components, and one for the rest of the kernel

OS Services as User Apps



Why?

- It's cool …
- Assume that OS coders are incompetent, malicious, or both …
  – OS components run as protected user-level applications
- Extensibility
  – easier to add, modify, and extend user-level components than kernel components

Implementation Issues

- What are the building blocks?
- What is run in privileged mode?

Mach

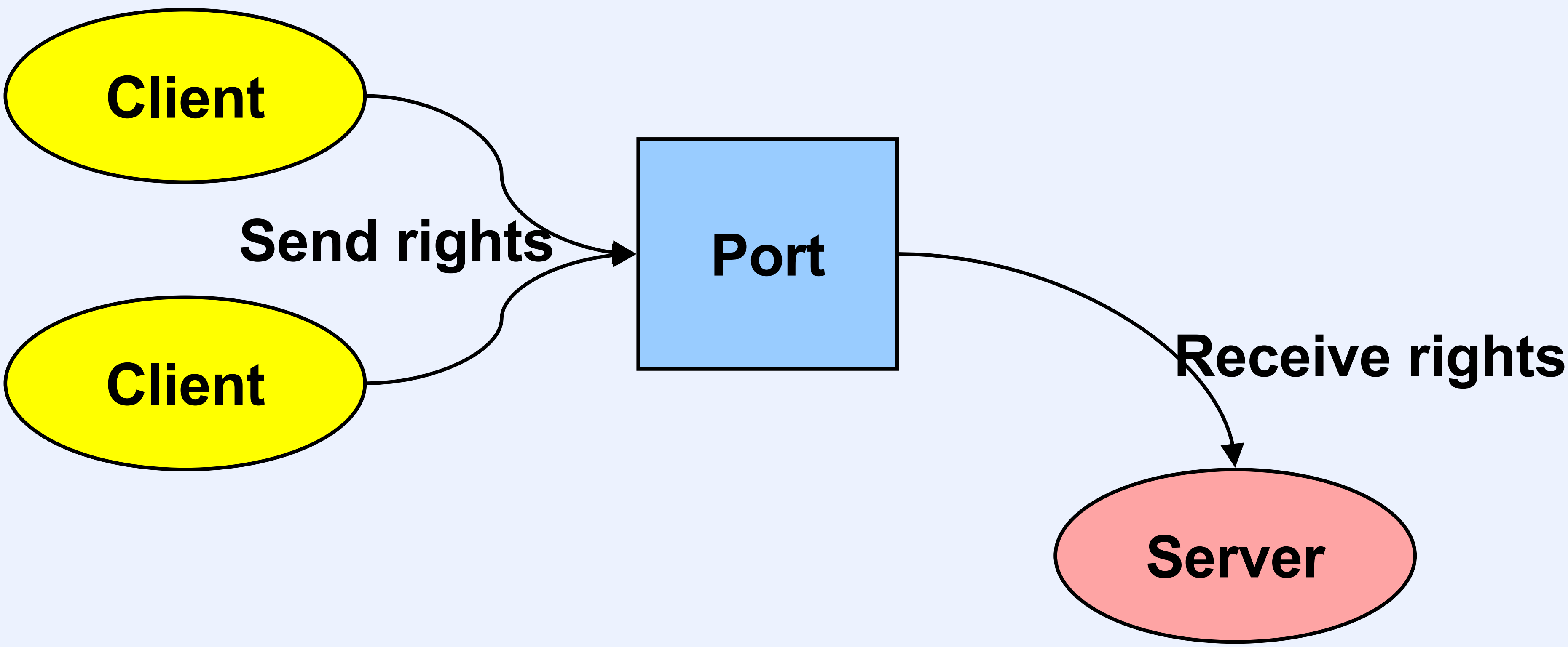
- Developed at CMU, then Utah
- Early versions shared kernel with Unix
  - basis of NeXT OS
    - basis of Apple OS X
- Later versions still shared kernel with Unix
  - basis of OSF/1
- Even later versions actually functioned as working microkernel
  - basis of GNU/HURD project
    - HURD: HIRD of Unix-replacing daemons
    - HIRD: HURD of interfaces representing depth

Mach's Building Blocks

- Tasks
  - represent services/objects
  - holders of access rights
- Threads
  - represent virtual processors
- Ports
  - communication channels and access rights
- Messages
  - carriers of data and access rights

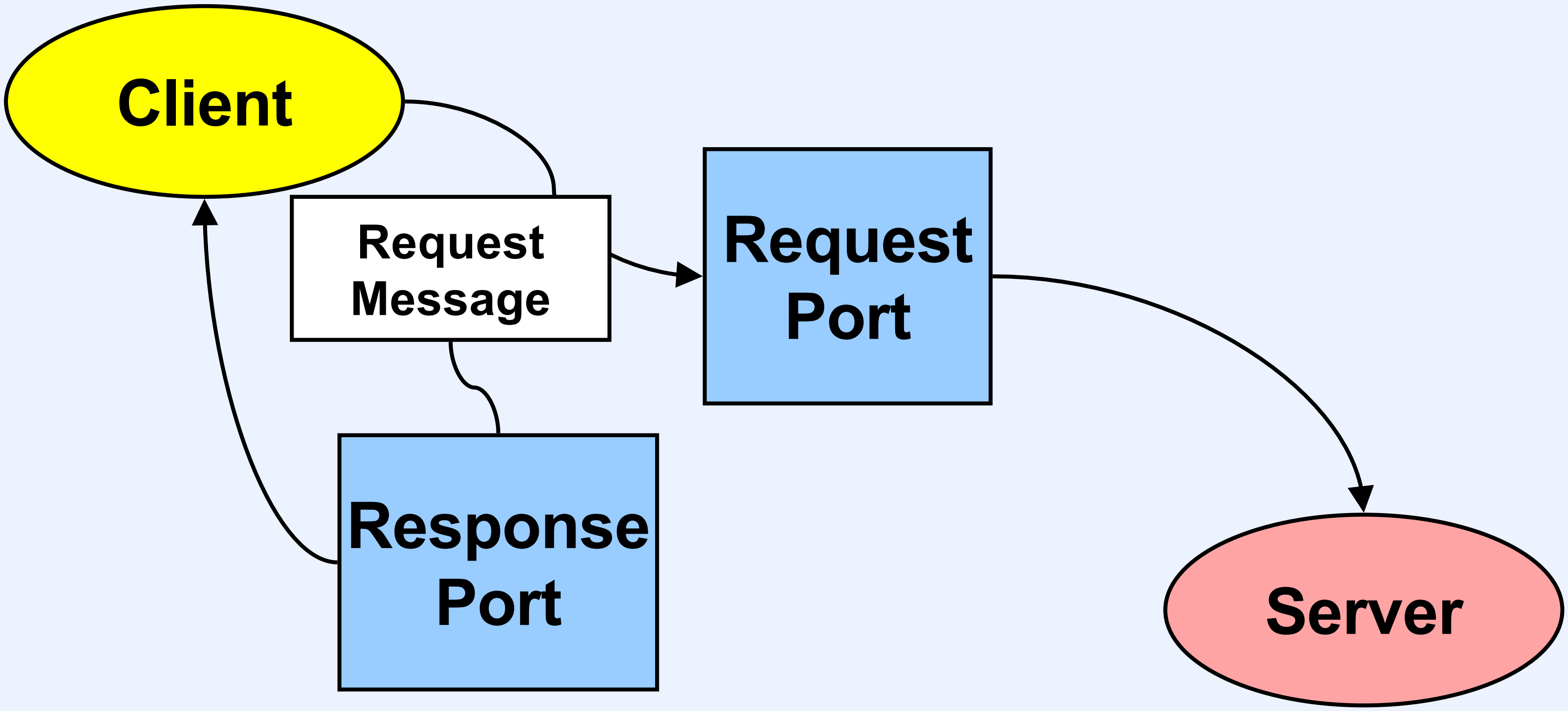
Mach Ports (1)

- Access rights



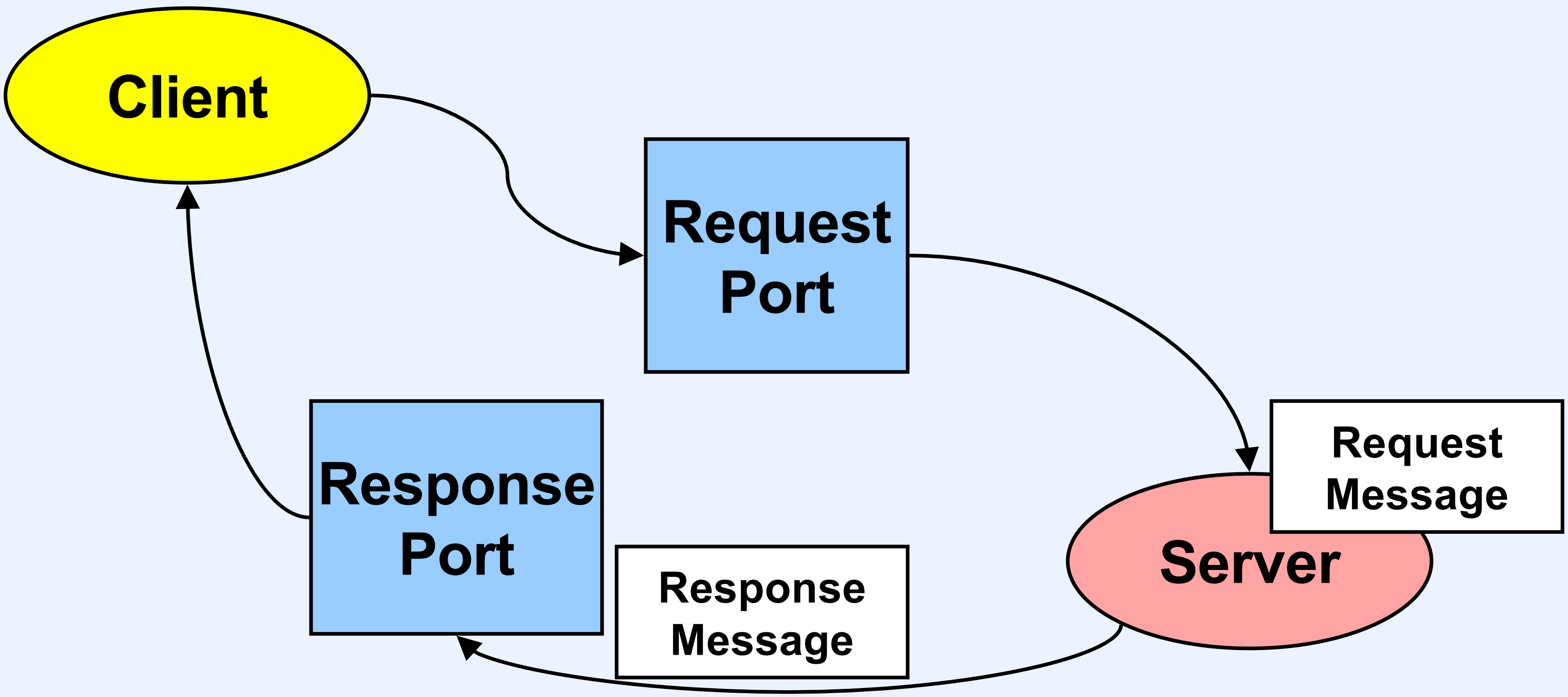
Mach Ports (2)

- Communication construct



Mach Ports (3)

- Communication construct



Method Invocation

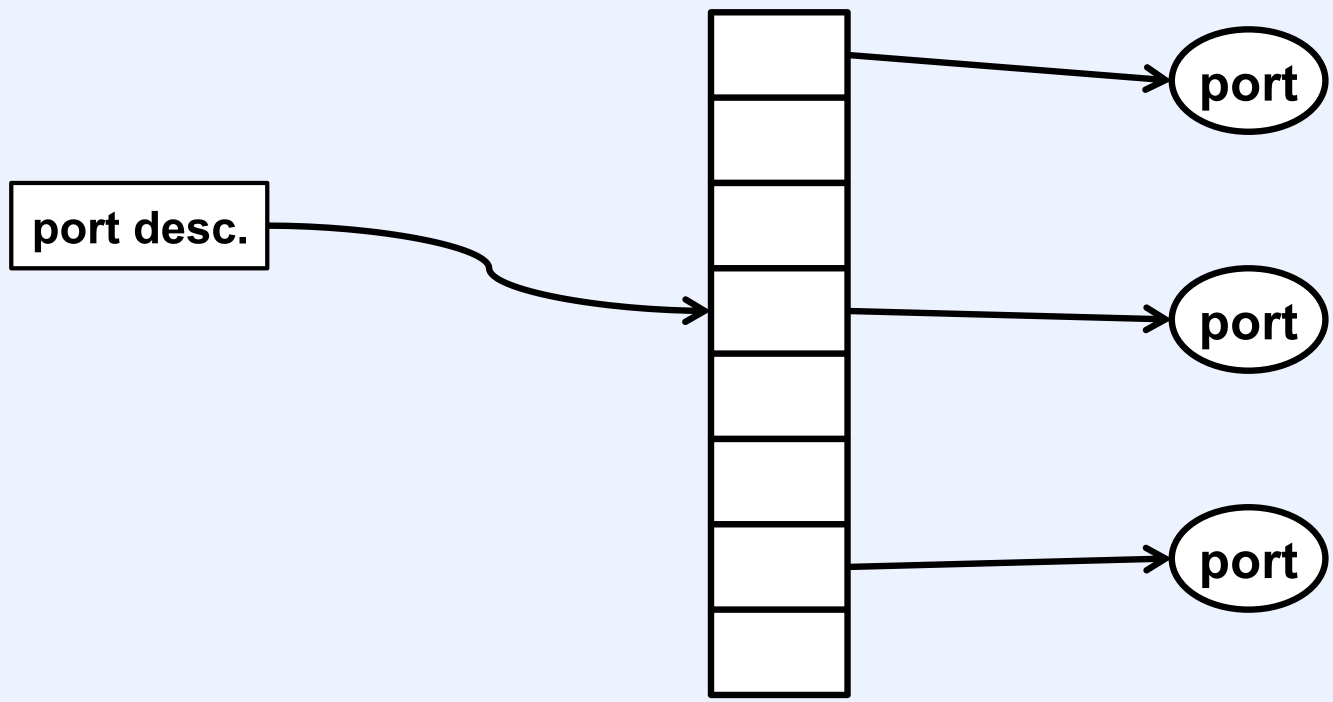
- *Tasks* implement objects
- Access rights to *ports* are secure object references
- *Messages* are method invocations and responses

Messages

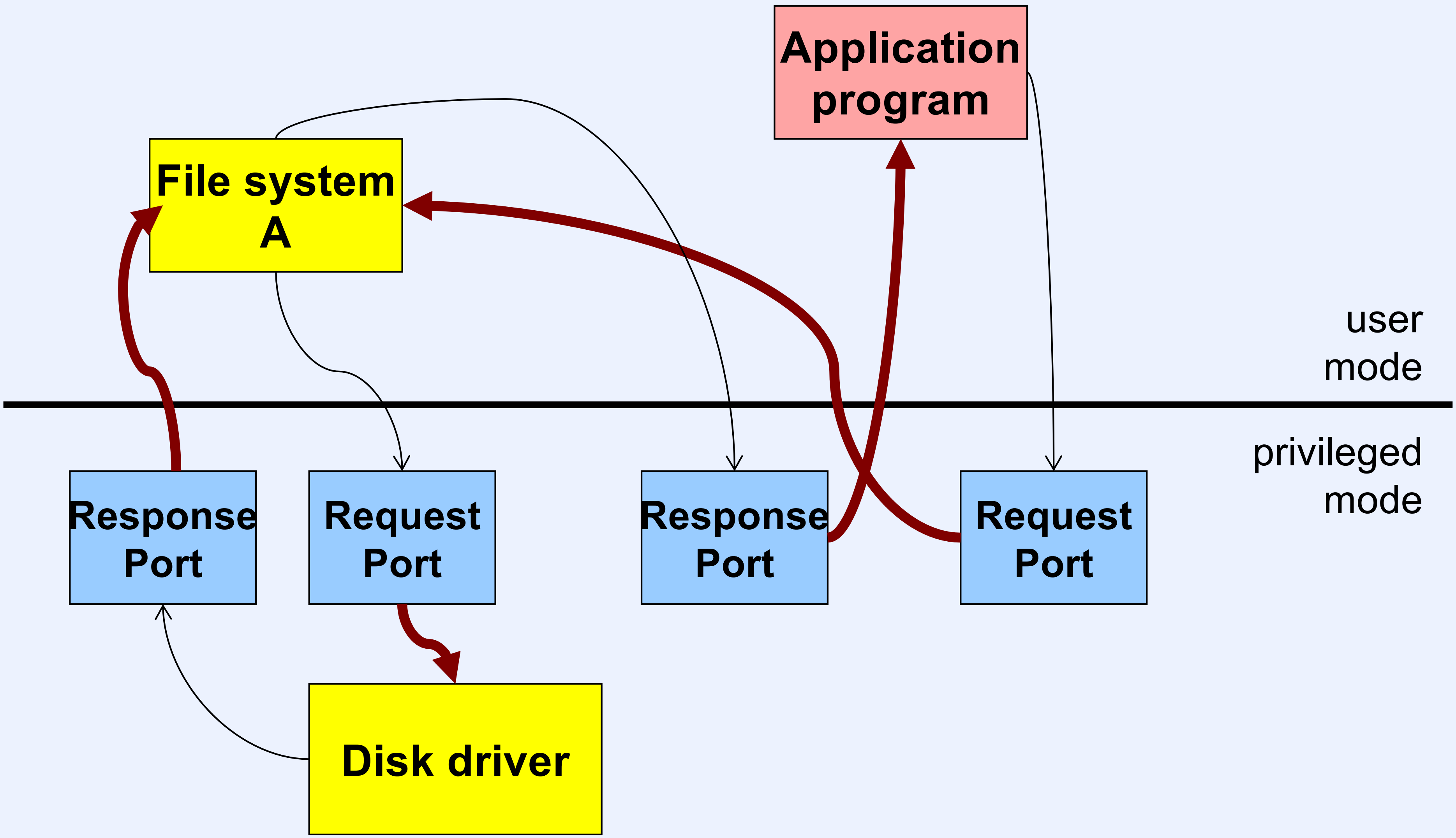
- Cost of passing messages is critical factor
- Small messages are copied
- Large messages within single address space passed by reference
- What about messages across address spaces?

Implementing Port Rights

- References to ports must be secure and unforgeable
  – how are they done?

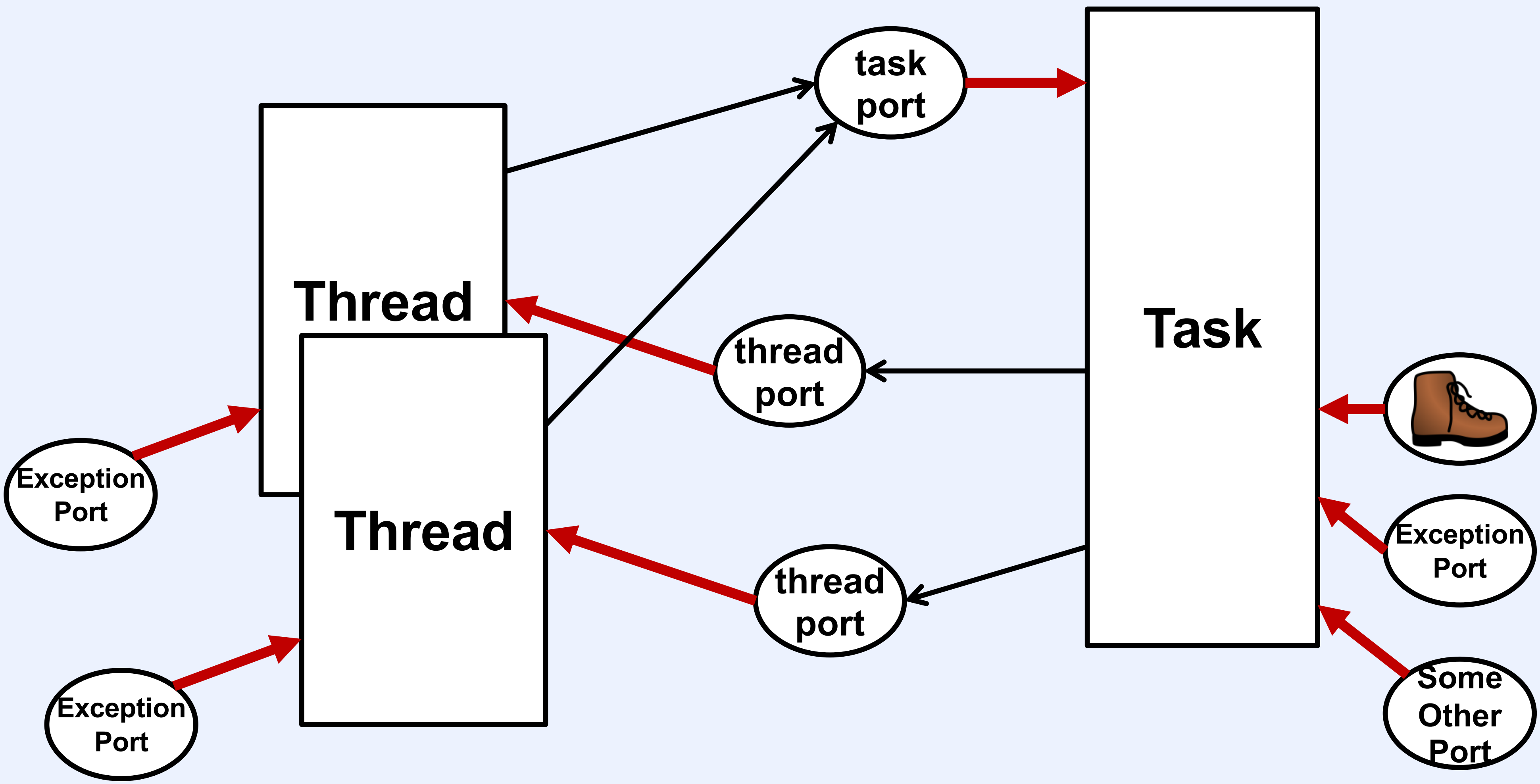


Example



Application program
File system A
user mode
privileged mode
Response Port
Request Port
Response Port
Request Port
Disk driver

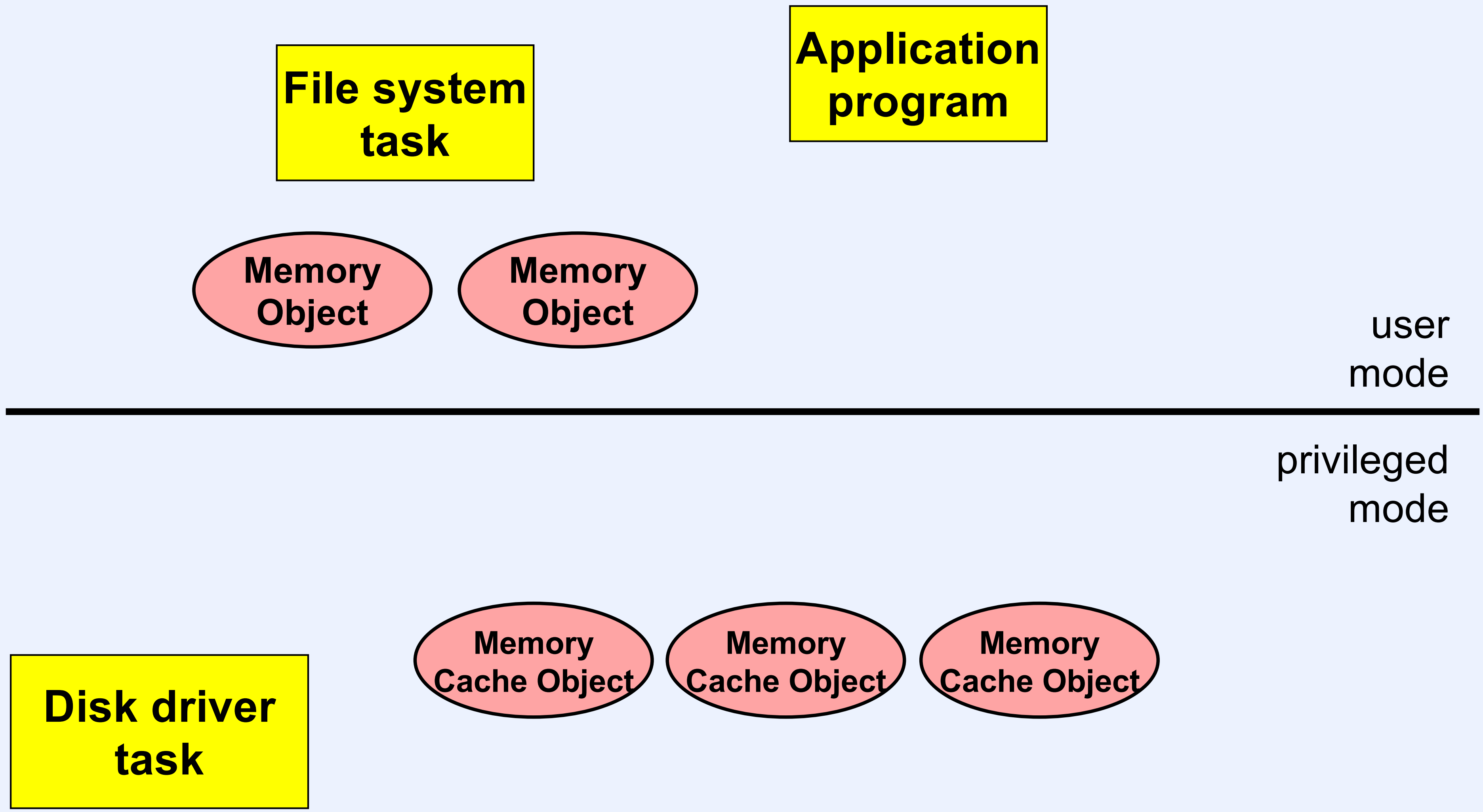
Task and Thread Objects



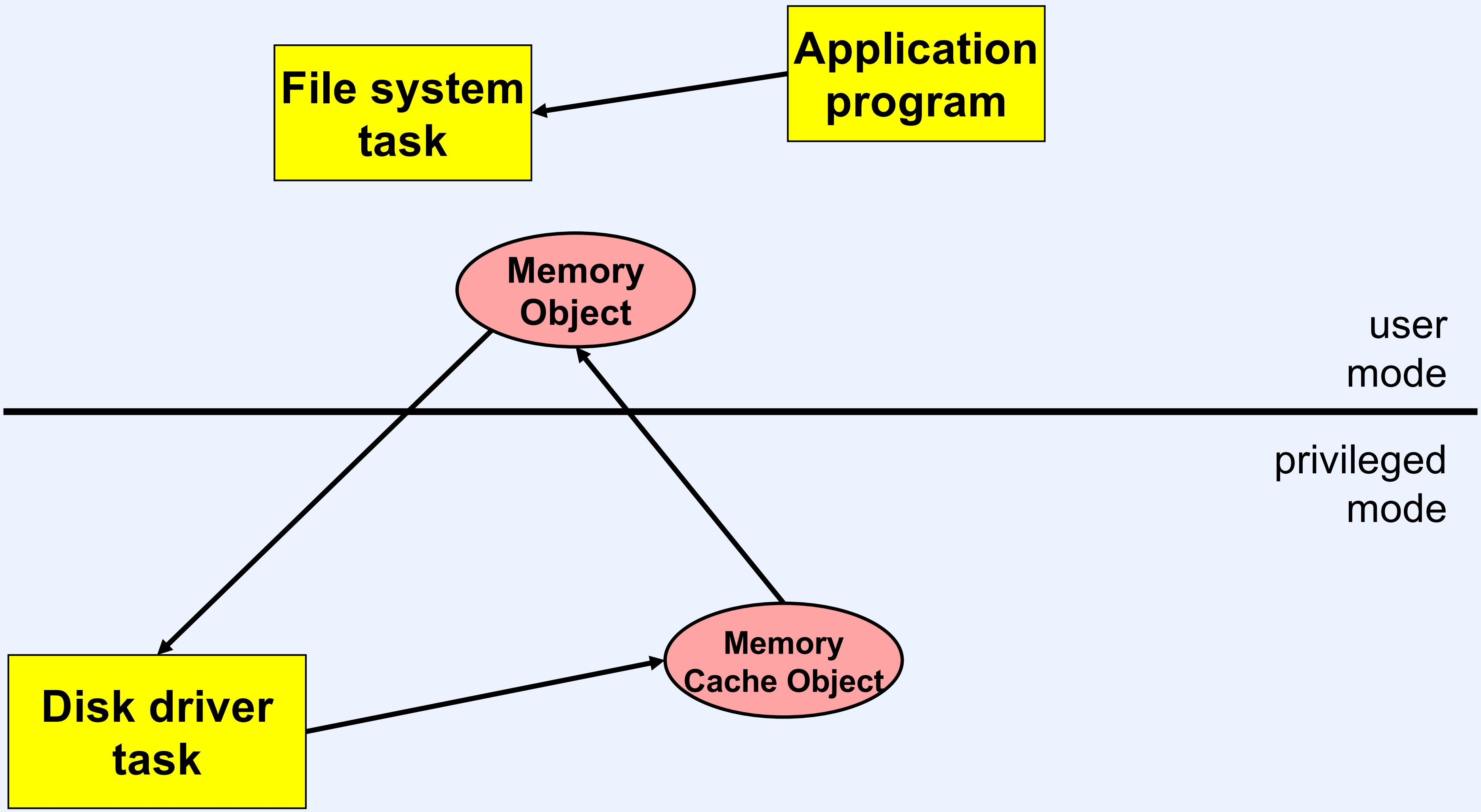
Virtual Memory

- **Memory cache objects**
  - implemented in kernel
  - represent what's in real memory
- **Memory objects**
  - implemented in kernel or as user tasks
  - represent what's mapped into real memory

Memory-Object Example



Memory-Object Example



Quiz 3

The application thread attempts to read from memory. The page containing the desired bytes is not present and must be fetched from disk. How many threads are involved?

a) one (just the application thread)
b) two
c) three or more

Devices

- Device master port exported by kernel
- Tasks holding send rights may request access to any device
  - send rights given for device port

Successful Microkernel Systems

-
-
- …

Attempts

- Windows NT 3.1
  - graphics subsystem ran as user-level process
  - moved to kernel in 4.0 for performance reasons
- Apple OS X
  - based on Mach
  - all services in kernel for performance reasons
- HURD
  - based on Mach
  - services implemented as user processes
  - no one used it, for performance reasons