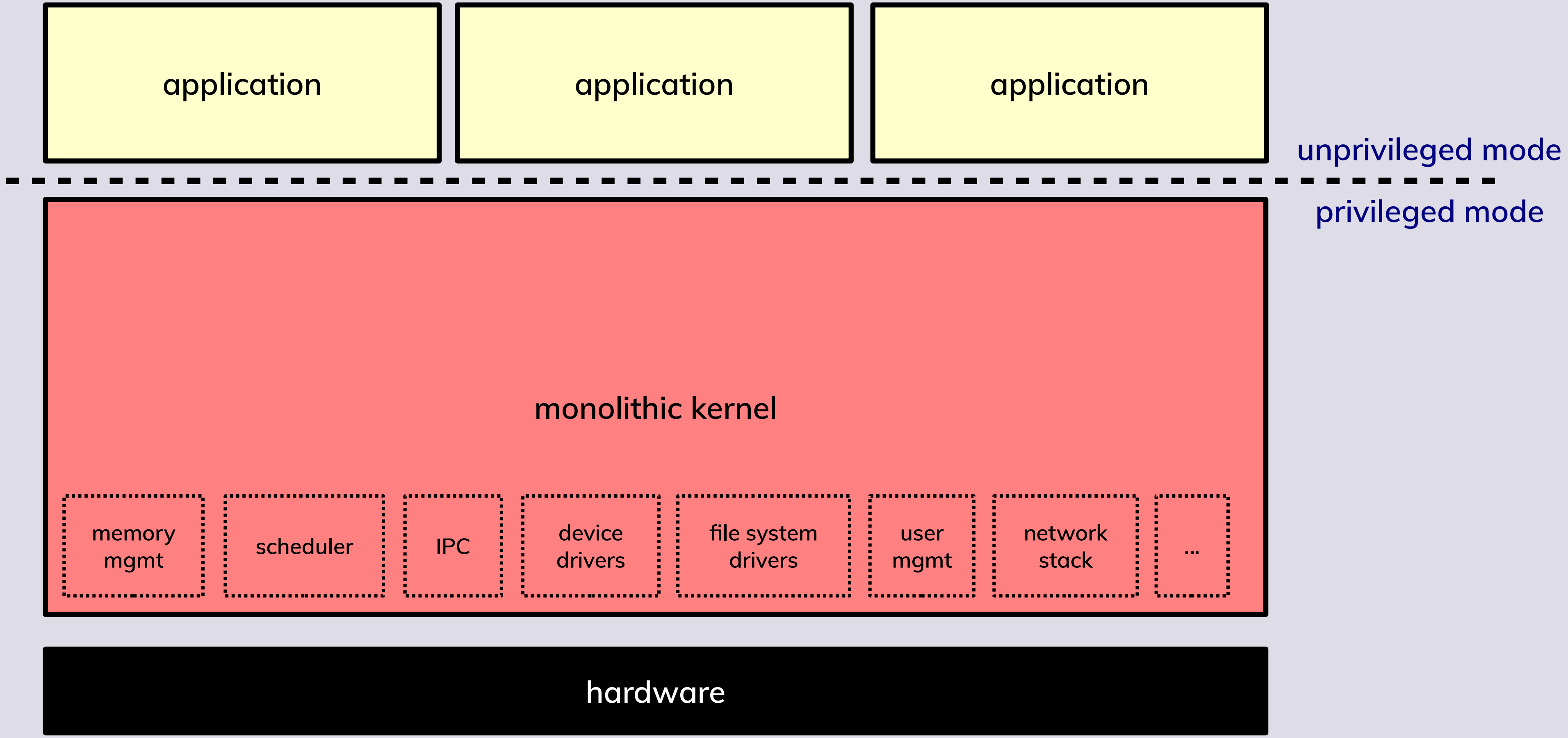
Advanced Operating Systems
Summer Semester 2022/2023

Martin Děcký

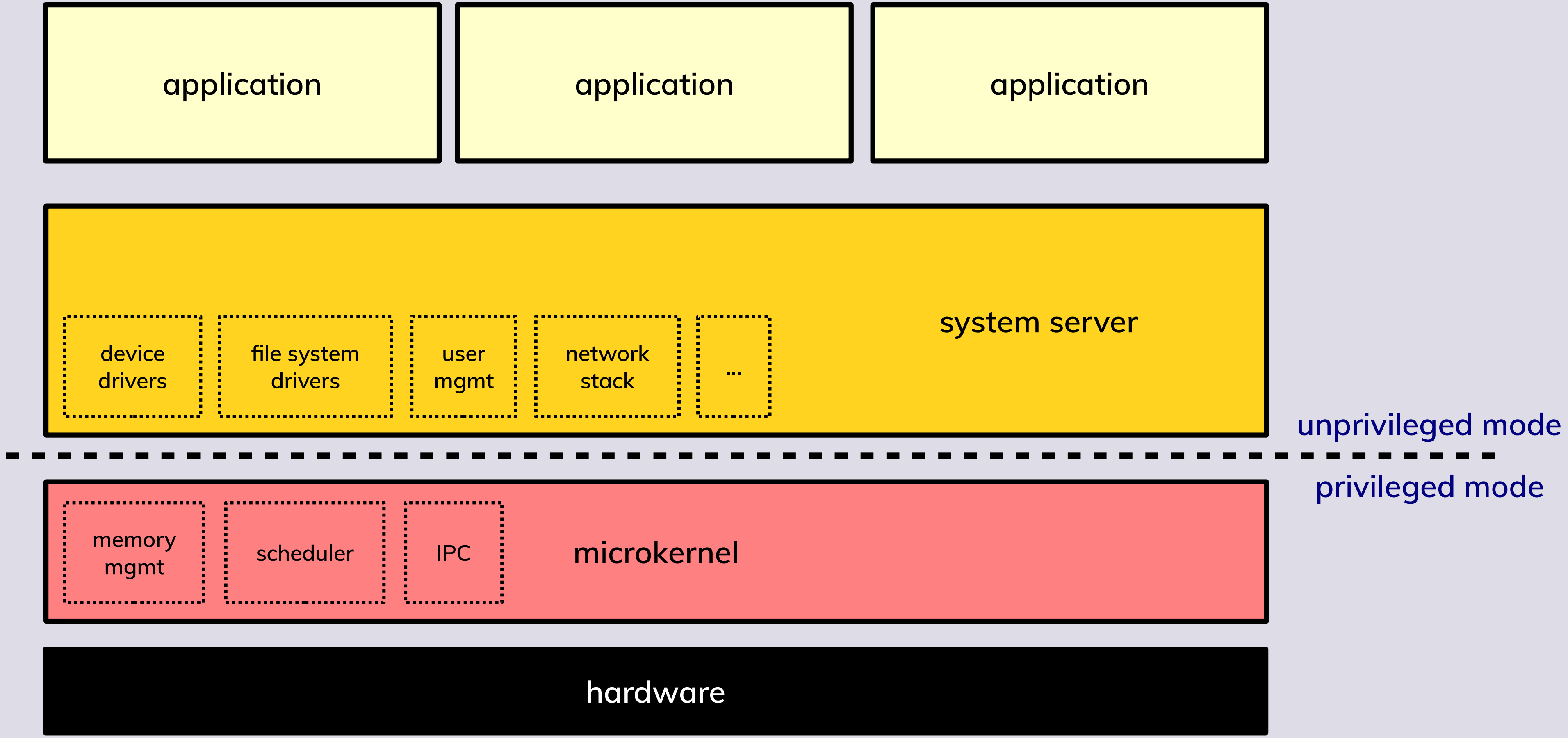
9

Virtualization

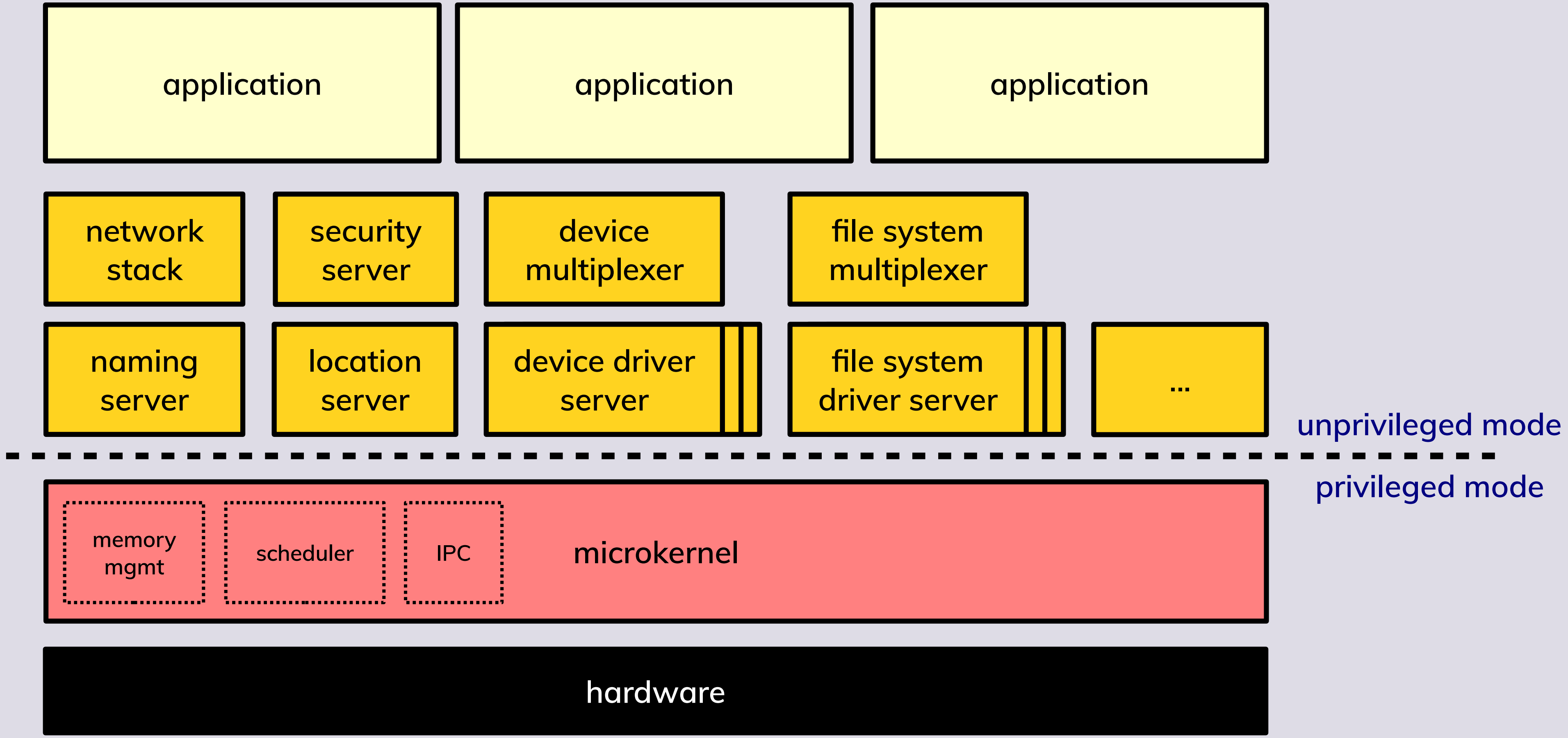
Monolithic Kernel



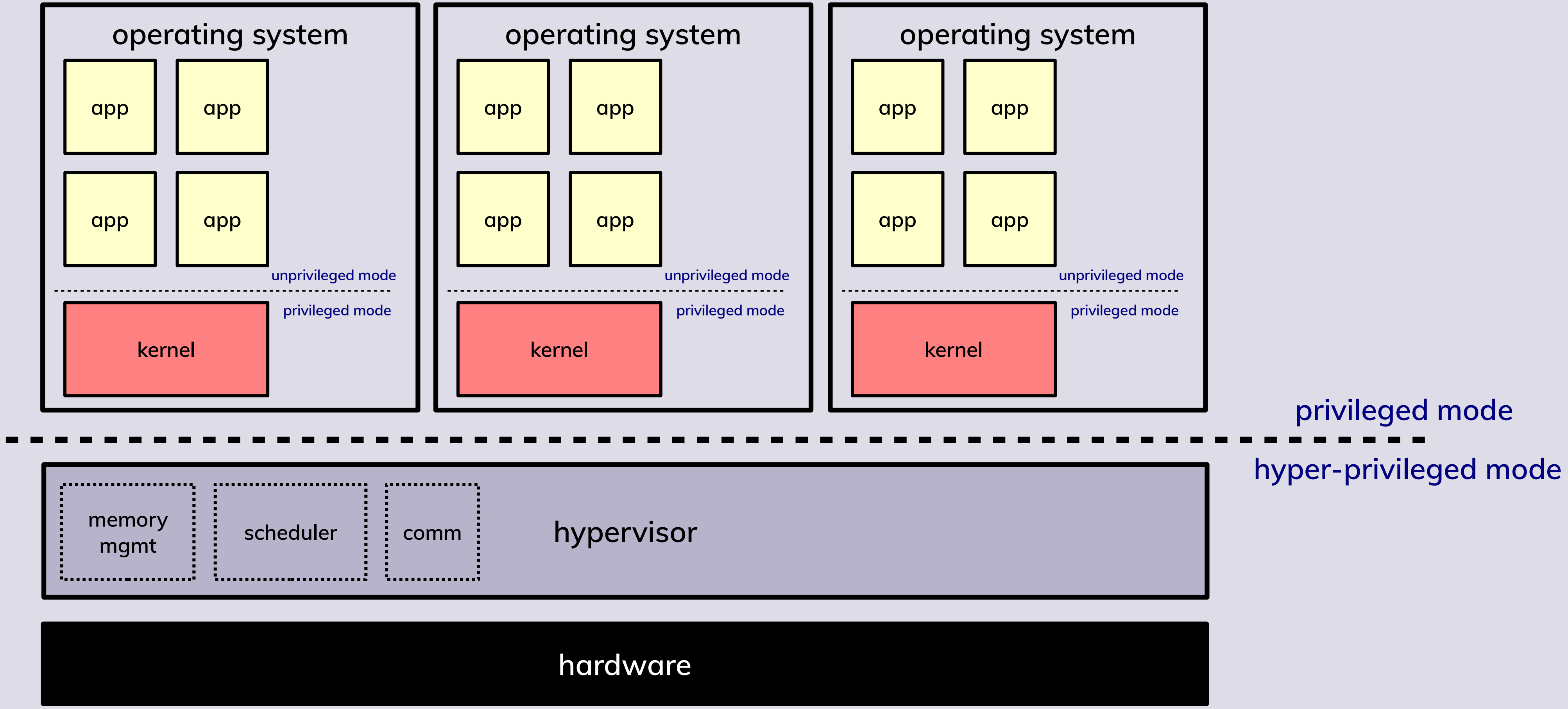
Single-server Microkernel



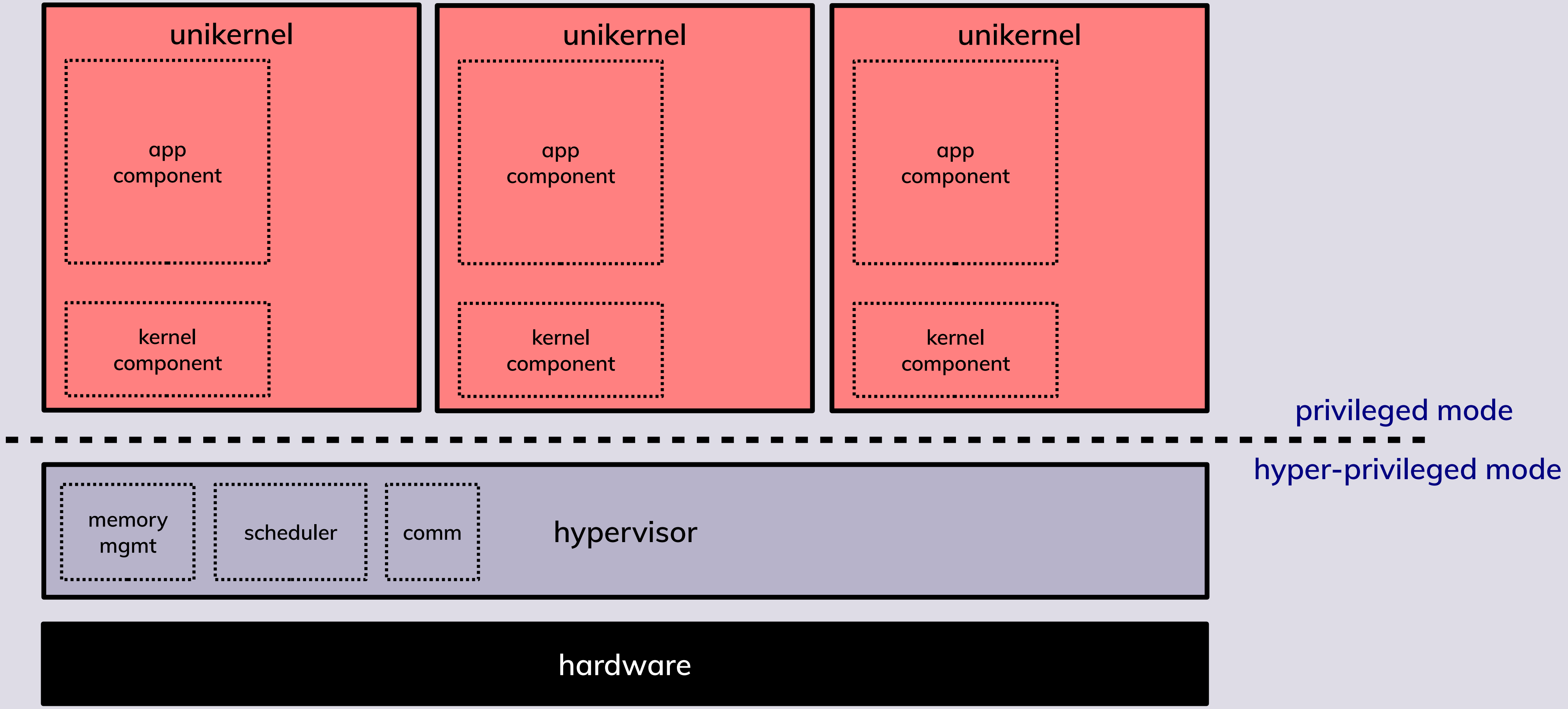
Multiserver Microkernel



Hypervisor (Type 1)



Hypervisor (Type 1) with Unikernels



Effective Virtualization

- **Popek/Goldberg conditions on instruction set effective virtualization**
  - Concerns instruction sets with a privileged and a non-privileged mode
  - Definitions
    - Virtualizable instructions
      - Instructions that always trap when executed in non-privileged mode
    - State-altering instructions
    - State-affected instructions
  - Instruction set is virtualizable if every state-altering and state-affected instruction is also a virtualizable instruction
    - Classical IA-32 contains several *critical* instructions that do not meet this condition
      - `SGDT`, `SIDT`, `SLDT`, `POPF`, `PUSHF`, `POP`, `PUSH`, `MOV`, `CALL`, `JMP`, `INT`, `RET`

Virtualization without Effective Virtualization

- **Non-transparent**
  - Partitioning
    - “Shared kernel virtualization”
      - Logical separation of user space tasks into containers
      - No true VM abstractions
        - Traditional OS abstractions with additional layer of resource management and object visibility
  - Paravirtualization
    - Voluntary cooperation between VM and hypervisor
      - VM replaces state-altering instructions with hypercalls and adapts the output of state-affected instructions
      - Also usable as a performance improvement (e.g. I/O) for transparent virtualization

Virtualization without Effective Virtualization

- **Transparent**
  - Emulation
  - Dynamic translation
    - More efficient emulation that tries to separate *critical* and *non-critical* instructions
      - Whenever a code page is executed, *critical* instructions are replaced by explicit traps
      - VM usually provided with a read-only shadow copy to maintain integrity
      - Complicated by the fact that many non-effectively virtualizable instruction sets also do not provide other efficient features (e.g. non-executable pages)

Virtualization without Effective Virtualization

- **Transparent**
  - Special hardware privileged mode
    - Turning *critical* instructions into virtualizable instructions
    - Usually somewhat limited in scope (e.g. V86 on IA-32)
  - Hyper-privileged mode
    - Mode that affects the behavior of the privileged mode (which is then not fully privileged)
      - Usually associated with an analogous set of control registers as the privileged mode
      - Instructions that might be *critical* w.r.t. non-privileged mode are virtualizable using the hyper-privileged mode
    - PL2 (ARM), EL2/EL3 (ARM64), M-mode (RISC-V)

Virtualization without Effective Virtualization

- **Transparent**
  - Orthogonal virtualization modes
    - Separate control registers (control structures) and control instructions
      - Nested virtualization possible if the control instructions are self *non-critical*
    - No traditional traps, but VM exits (and VM entries)
    - Intel VT-x (VMX), AMD AMD-V (SVM)
      - Root mode (hypervisor)
      - Non-root mode (guest VM)
    - Hypervisor Extension (RISC-V)
      - HS-mode (hypervisor-extended supervisor mode)
      - VU-mode (virtual user mode), VS-mode (virtual supervisor mode)

Operating System Virtualization Abstraction

- **vCPU (virtual CPU)**
  - Logical extension of the (user) thread abstraction
    - Entity that keeps the computational context state
    - Besides the usual user context, it also tracks the privileged context
      - Paravirtualization
        - User context: Guest user thread running inside the VM
          - Exceptions, page faults, IRQs, IPC, etc., switch to the privileged context
        - Privileged context: Guest paravirtualized kernel (running in a different address space) that provides the environment for the guest user threads in the VM (including thread scheduling, etc.)
      - Transparent virtualization
        - User context: Virtual machine monitor (VMM, running in a different address space)
        - Privileged context: Context of the entire guest VM
          - Regular exceptions (including standard page faults) handled internally
          - IRQs, some state-altering instructions and other conditions switch to the user context (VM exit)

Intel VT-x (VMX)

- **Crucial instructions**
  - `VMXON` / `VMXOFF`
    - Enter / exit root mode
    - 4 KiB physical location for virtualization bookkeeping (opaque)
  - `VMPTRLD`
    - Load a Virtual-Machine Control Structure (VMCS) as current
      - 4 KiB physical location that stores the vCPU privileged context
        - Mostly opaque, fields accessed strictly via the `VMREAD` / `VMWRITE` instructions
        - Control fields (affecting the features / behavior of the virtualization, events that trigger VM exits, nested paging configuration, etc.)
          - vCPU ID
        - Guest fields (context of the guest VM, i.e. privileged context of the vCPU)
          - `RSP`, `RSP`, `RIP`, `RFLAGS`, selectors, control registers, MSRs, interrupt/activity state
          - Does not store most of the GPRs
        - Host fields (context of the VMM, i.e. user context of the vCPU)
          - Analogy of the guests fields (for efficiently switching to the VMM)
        - Read-only fields (information about the VM exit)
          - VM exit reason, interruption (IDT vectoring) state, guest-physical address of a nested page fault, I/O instruction information, etc.

Intel VT-x (VMX)

- **Crucial instructions**
  - `VMLAUNCH` / `VMRESUME`
    - Launch / resume the current VMCS (i.e. execute a VM entry)
      - If there is no error on the VM entry, the instruction eventually transfers to the host state of the VMCS when a VM exit occurs
  - `INVEPT` / `INVVPID`
    - Invalidate the TLB for the nested paging based on the Extended Page Table root pointer or on the vCPU ID
  - `VMCALL`
    - Hypercall to the VMM
  - `VMFUNC`
    - Possible hardware acceleration of certain VMM operations (without a VM exit)
      - Currently only the Extended Page Table root pointer switching (among preset list of possible values)
      - Can be used to implement efficient hardware-assisted address space switching for IPC [1]

Intel VT-x (VMX)

- **Crucial VM exits**
  - Exception or NMI
  - External interrupt
  - Triple fault / INIT signal (i.e. reset) / start-up IPI
  - SMI events
  - Interrupt / NMI window (VM is in a state where it can handle the event)
  - Task switch, control register access, debug register access, `CPUID`, `RDMSR`, `WRMSR`, `GETSEC`, `HLT`, `INVD`, `INVLPG`, `MWAIT`, `MONITOR`, `PAUSE`, `XSETBV`, `XSAVES`, `XRSTORS`, `PCONFIG`, etc.
  - I/O instruction
  - APIC access
  - EPT violation
  - `VMCALL` (i.e. hypercall)
  - VMX instruction (i.e. nested virtualization)
  - Preemption timer
  - Page-modification log full

References

[1] Mi Z., Li D., Yang Z., Wang X., Chen H.: *SkyBridge: Fast and Secure Inter-Process Communication for Microkernels*, in Proceedings of the 14th EuroSys Conference, ACM, 2019

D3S

Thank you!

Questions?