# *NanoKernel and Hypervisors*

*Advanced Operating Systems*

Luca Abeni

luca.abeni@santannapisa.it

# Traditional OS Protection

- Traditional view
  - CPU: (at least) 2 privilege levels $\rightarrow$ distinction between user programs and kernel
  - User programs: low privilege
  - Kernel: high privilege mode, must be trusted
- So, we can see $2$ *protection domains*
  - Protection domain $\neq$ address space
  - User code and kernel can run in the same address space, but page access rights might be different
    - This was a good idea until Meltdown / Spectre!!!

# Address Spaces and Protection Domains

- Address space: characterized by the mapping between virtual addresses and physical addresses
  - Page table
  - In general, space → mapping between virtual resources and physical ones
- Protection domain: characterized by the policies in allowing access to resources
  - If a (virtual) memory page is mapped in physical memory, can it be accessed?
- Traditionally, in supervisor mode everything can be accessed

# Multiple Protection Domains

- Why using only $2$ protection domains?
  - Because this model maps naturally to the “least common denominator” provided by different hw architectures
- If we extend this concept (allowing multiple protection domains), we can have a more flexible architecture
  - We can split different OS components in different domains...
  - ...Or we can have different OSs / OS kernels running in different domains!
- How to switch between protection domains?

# Spaces, Domains and... Portals!

- Portal: abstraction used to switch betwen different domains
  - In traditional OSs, syscall and interrupts used to securely change privilege level
  - With multiple protection domains, the concept must be extended!
- Associated to interrupt / exception / trap / page fault
  - Specifies the domain handling it $\rightarrow$ used to move execution between domains
- Lower-level abstractions respect to “traditional” OSs and Kernels

# SPACE “NanoKernel”

- Provides only $3$ abstractions
  - Space: translation of virtual addresses/resources into physical ones (and/or portals)
  - Domain: access policy (determine how to map the space addresses: in physical addresses or portals?)
  - Portal: mechanism to move between domains
- The nanokernel provides `portal_entry` and `resume_pcb`
- Everything else can implemented by code running in different domains!!!

# Adeos and SPACE

- Adeos implements some of the notions from the SPACE kernel
  - With focus on interrupt management
  - Many protection mechanisms (example: memory protection) are not considered, for efficiency / simplicity
- Different domains, for different kernels
  - At least Linux and some real-time executive (RTAI, Xenomai nucleus, ...)
  - More complex setups are possible
- Interrupt portals to build the interrupt pipeline

# Adeos as Support for Application-Specific Kernels

- The original SPACE desing provided support for application-specific OSs
  - So that resource allocation can be optimized for specific applications!
- Possible by executing different OSs in different protection domains
  - SPACE provides protection and security
- Adeos focuses on optimizing an OS kernel for real-time
  - Again, protection and security are not considered...

# Application-Level Resource Management

- The “exokernel” idea proposed something similar
- Resource management moved from the OS kernel to user applications
  - Small *exokernel* allowing to do this in a secure way
  - Most of the OS kernel linked to user applications as a “library Operating System”
- Adeos focuses on a similar idea: the real-time executive / real-time applications are in charge of managing their resources
  - Management delegated to the Linux kernel for non real-time resources
  - Again, no focus on protection / security

# Adeos as a Hypervisor - 1

- Adeos calls itself a “nanokernel”
  - Following naming from some scientific papers
  - Tries not to be a “Hardware Abstraction Layer” (HAL)
- SPACE idea: multiple OSs in multiple protection domains
  - Two domains: one for Linux and one for the real-time executive
- Relationship woth hypervisors?
  - Similar idea
  - (Protection) Domain $\leftrightarrow$ VM
  - Ah... This is where the Xen domains come from!!!

# Adeos as a Hypervisor - 2

- Can Adeos be seen as a hypervisor?
  - Two OS kernels in two domains...
- Hypervisor for para-virtualized kernels
  - A kernel must be modified to run on Adeos
  - Patched Linux kernel, Xenomai, RTAI, ...
- Hypervisor without complete control of the system resources (performance trade-off?)
  - Linux and the RT kernel can crash the system...
  - ...Including other para-virtualized kernels!
- Hosted hypervisor
  - Does not boot on baremetal, but uses functionalities from the Linux kernel

# Xtratum

- Another “hypervisor”, very similar to Adeos
  - Again, started as an RTLinux replacement!
- Can run (paravirtualized) Linux and some paravirtualized RTOSs
- Big difference: Xtratum is a bare-metal hypervisor
  - Does not rely on Linux (or other kernels’) functionalities
  - Loaded by a bootloader, can start Linux and other kernel after booting Xtratum