



COMP90015 Distributed Systems

Semester 2, 2023

Topic: Operating System Architecture and Distributed Systems

Dr Tawfiq Islam
School of Computing and Information Systems (CIS)
The University of Melbourne, Australia



- Explore the architecture of a kernel suitable for a distributed system.
- A key principle of DS is openness and with this in mind, let us examine the major kernel architectures:
 - Monolithic kernels
 - Layered architecture-based kernels
 - Micro-kernels

- An open DS should make it possible to:
 - Run only that (“**specific**” component of) system software at each computer that is necessary for its particular role in the system architecture.
 - ❖ For example, system software needs of laptops and dedicated servers are different and loading redundant modules wastes memory resources.
 - Allow the software implementing any service to be changed independent of other facilities.
 - Allow for **alternatives** of the same service to be provided, when this is required to suit different users or applications.
 - Introduce **new services** without harming the **integrity** of existing ones.



Separating Mechanisms and Policies in OS and DS

- A Guiding Principle of OS design:
 - The separation of fixed resource management “**mechanisms**” from resource management “**policies**”, which vary from application to application and service to service.
 - For example, an ideal scheduling system would provide mechanisms that enable a multimedia application such as videoconferencing to meet its real-time demands while coexisting with a non-real-time application such as web browsing.
- That is kernel would provide only the most basic mechanisms upon which the general resource management tasks at a node are carried out.
- Server modules would be dynamically loaded as required, to implement the required RM policies for the currently running applications.

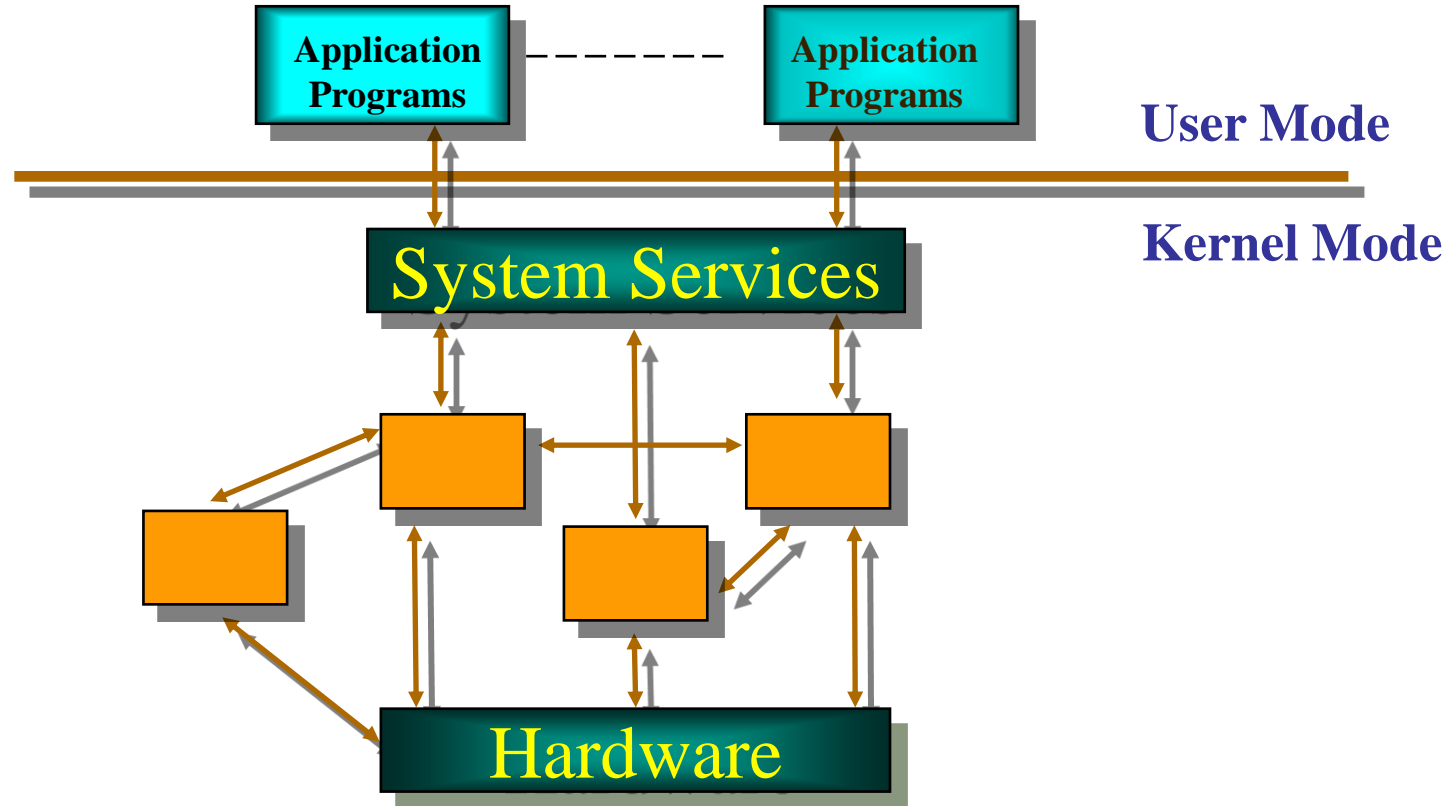
- **The kernel** is the core component of an operating system. It's a piece of software that manages system resources, controls hardware devices, and provides essential services to user-level applications and processes.
- The two key examples of kernel design approaches are:
 - Monolithic
 - Microkernel
- These two designs differ primarily in the decision as to what functionality belongs in the kernel and what is left to server processes that can be dynamically loaded to run on top of it.

- Serve as frameworks that unify capabilities, services and tasks to be performed
- Three approaches to building OS....
 - Monolithic OS
 - Layered OS
 - Microkernel based OS
 - ✓ Client server OS
 - ✓ Suitable for distributed systems
- Simplicity, flexibility, and high performance are crucial for OS.

User Mode vs Kernel Mode

- **User mode** and **kernel mode** represent two different privilege levels within an operating system. User mode is for user-level processes with restricted access, while kernel mode is for the operating system's kernel and critical system services with full access to hardware and resources. Transition between these modes occurs through system calls, ensuring that user programs can safely interact with the kernel.
- The separation between user mode and kernel mode is essential for the security and stability of the operating system. It prevents user-level processes from interfering with the critical operations of the kernel and other processes.

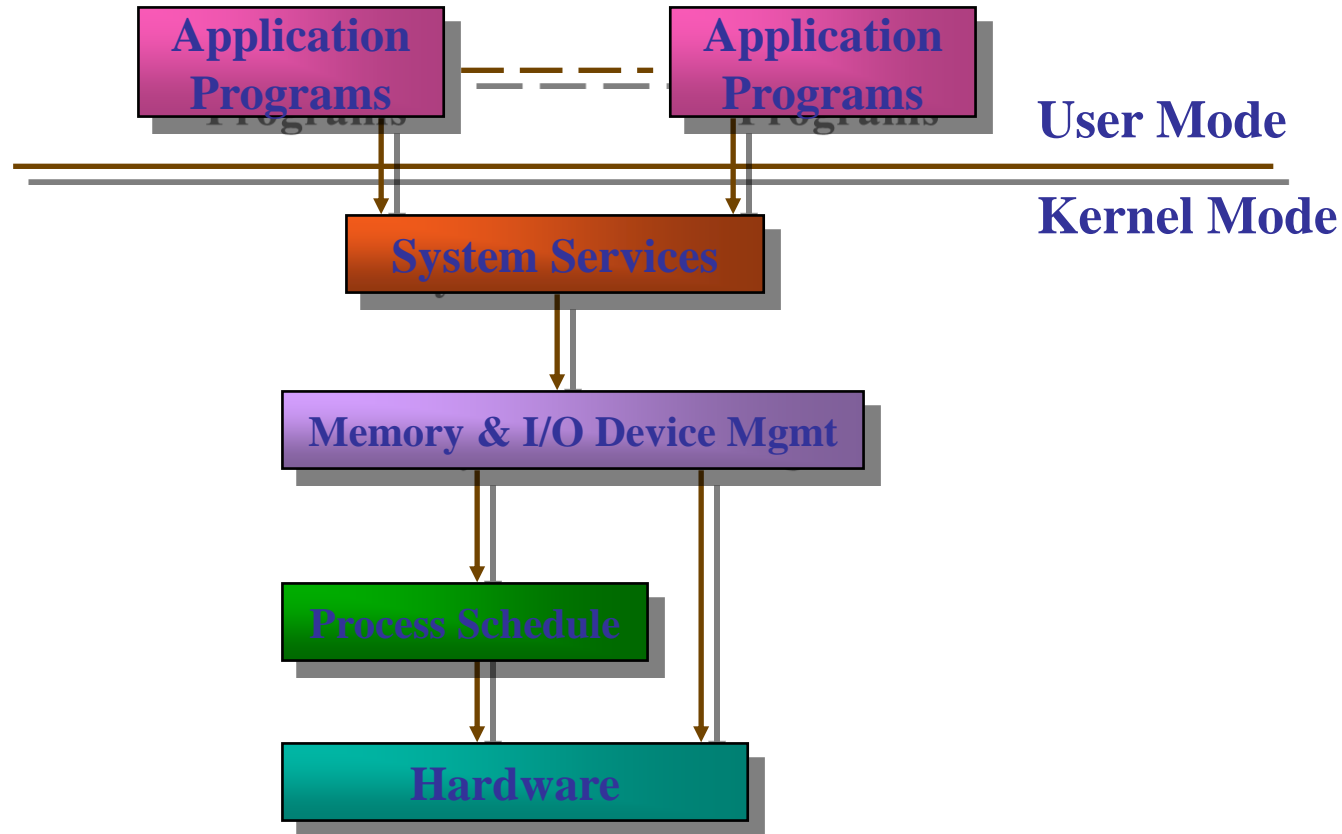
Monolithic Operating System



- Better application Performance
- Difficult to extend

Ex: MS-DOS

- **It is massive:**
 - It performs all basic OS functions and takes up in the order of megabytes of code and data
- **It is undifferentiated:**
 - It is coded in a non-modular way (traditionally) although modern ones are much more layered.
- **It is intractable:**
 - Altering any individual software component to adapt it to changing requirements is difficult.



- Easier to enhance
- Each layer of code access lower-level interface
- Low-application performance

Ex: UNIX/Linux

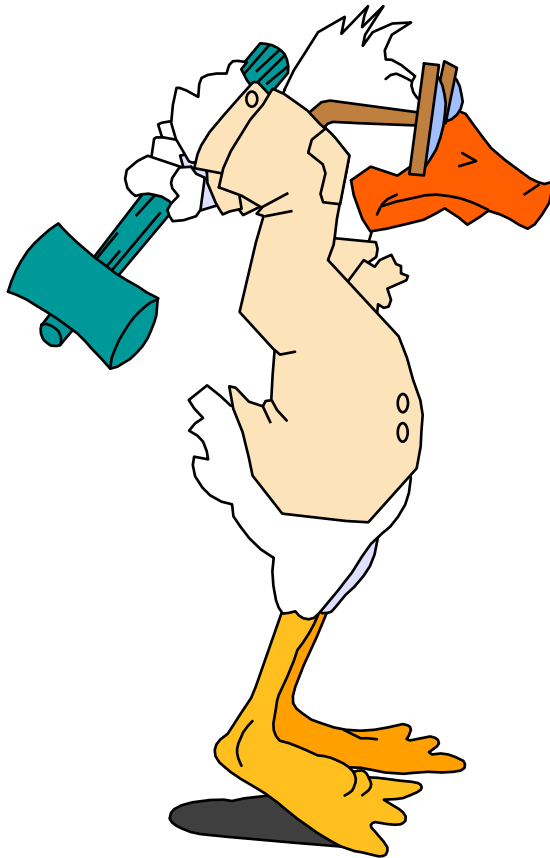
Traditional OS

Application
Programs

Application
Programs

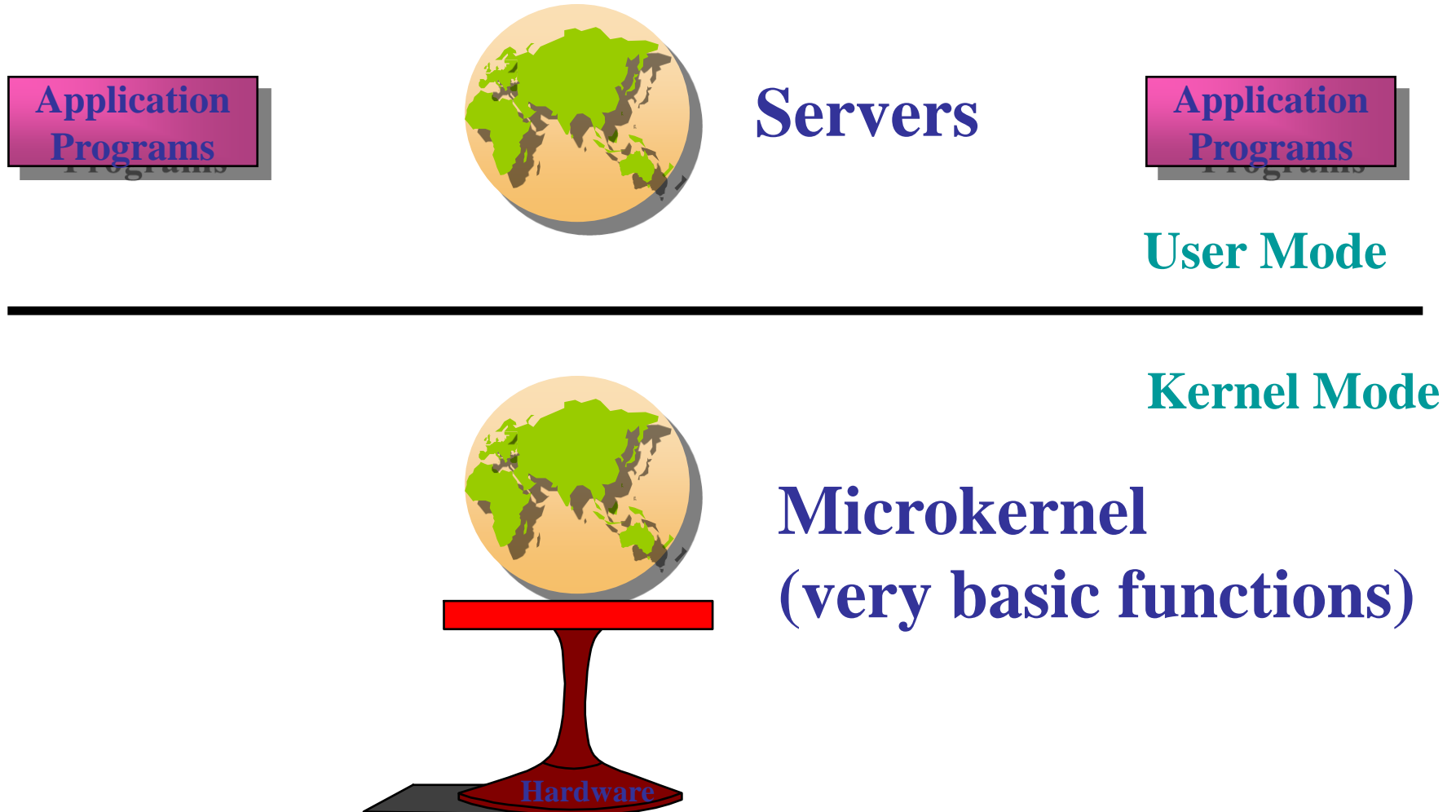
User Mode

Kernel Mode



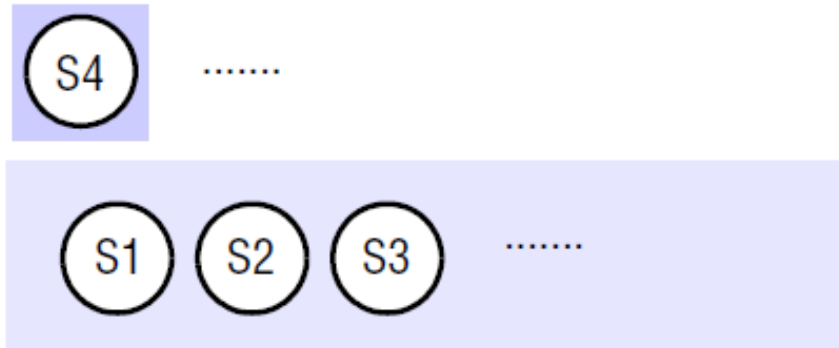
OS Designer

New trend in OS design: Separating mechanisms and policies





- Compared to monolithic, microkernel design provides only the most basic abstractions,
 - address space, threads and local IPC.
- All other system services are provided by servers that are dynamically loaded precisely on those computers in the DS that require them.
- Clients access these system services using the kernel's message-based invocation mechanisms.

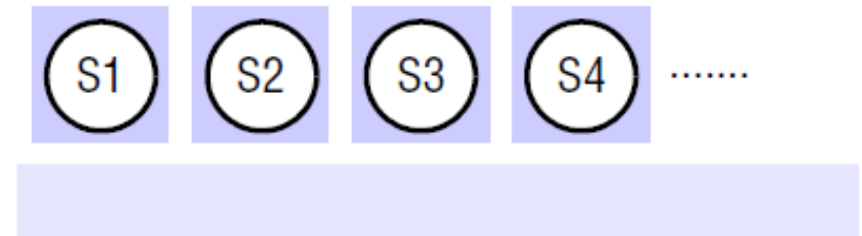
Monolithic kernel vs microkernel




Monolithic kernel

Key:

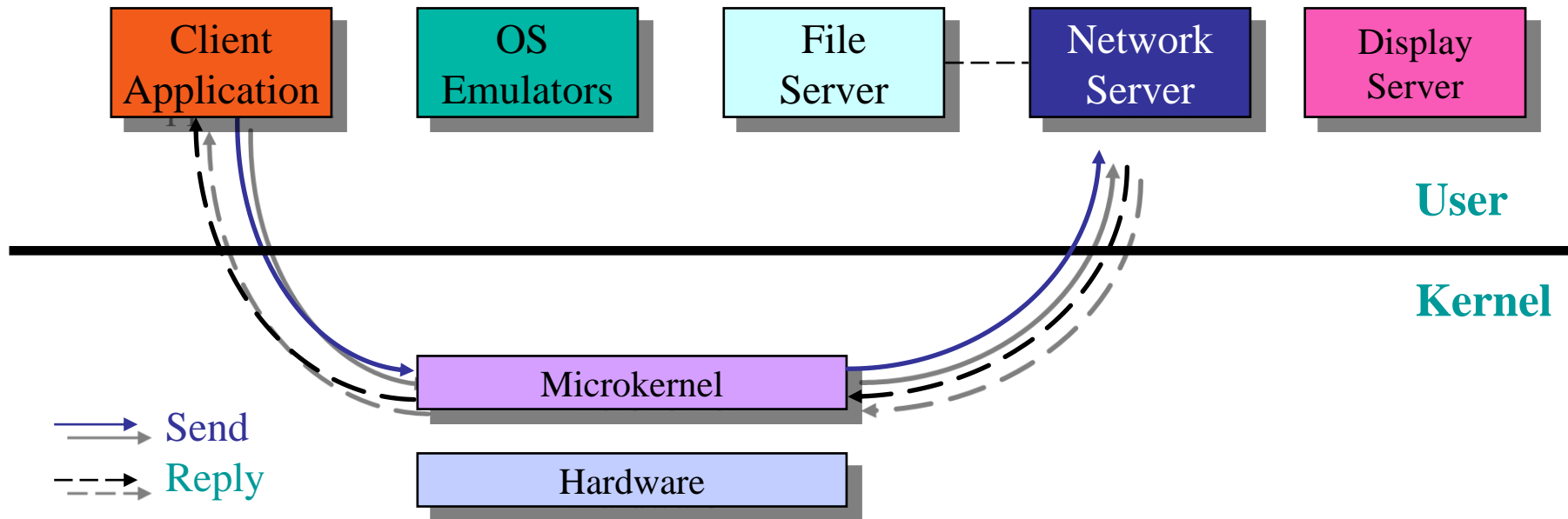
Server:  Kernel code and data: 



Microkernel

Dynamically loaded server program: 

Microkernel/Client Server OS

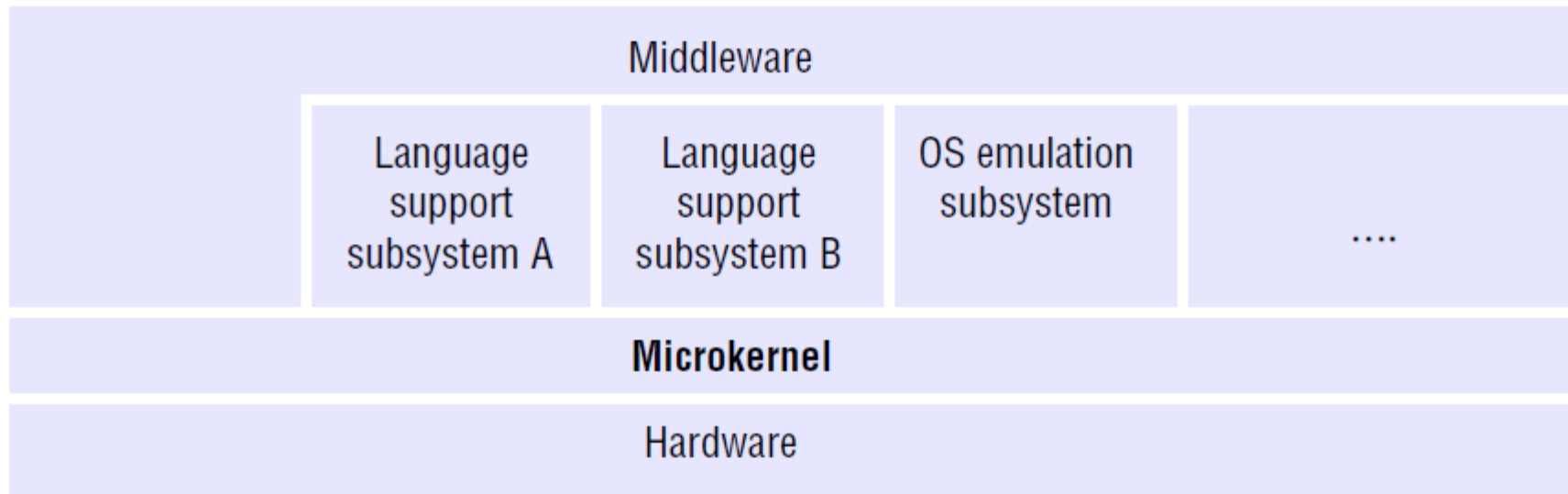


- Tiny OS kernel providing basic primitive (process, memory, IPC)
- Traditional services becomes subsystems
- OS = Microkernel + User Subsystems

Ex: Mach, QNX, Windows NT!

The role of the microkernel (MK)

- MK appears as a layer between Hardware and a layer of major system components (subsystems). If performance, rather than portability is goal, then middleware may use facilities of MK directly.

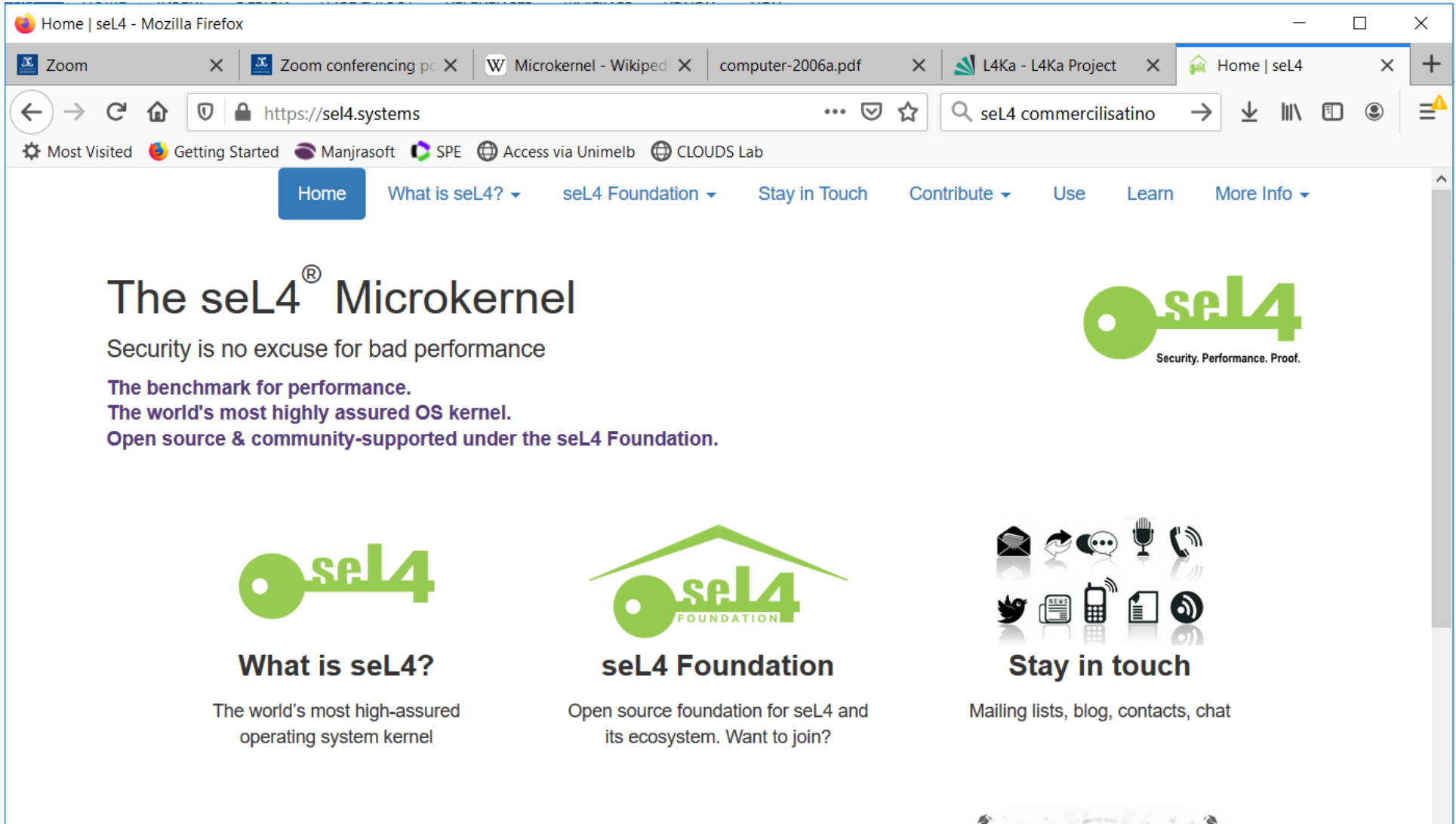


The microkernel supports middleware via subsystems

Few Popular Microkernel Systems

- MACH, CMU (Carnegie Mellon University)
 - supports OS emulators such as Unix and OS/2.
- MINIX (modular microkernel by Tanenbaum)
- PARAS (C-DAC, India) for PARAM Supercomputers
- ChorusOS (Sun, USA) Realtime OS (RTOS)
- seL4 (created by NICTA/Data61, Australia)
 - Google KateOS for embedded devices (uses seL4)
- QNX - Unix-like RTOS (Canada, BlackBerry)
 - used in a variety of devices including cars and mobile phones (e.g., BlackBerry).
 - Intel x86, MIPS, PowerPC, StrongARM.
- Windows NT – original design.





Home | seL4 - Mozilla Firefox

Zoom X Zoom conferencing pc X Microkernel - Wikipedi X computer-2006a.pdf X L4Ka - L4Ka Project X Home | seL4 X

← → ↻ 🏠 🔒 https://sel4.systems 🔍 seL4 commercialisatio ⬇️ 📄 📖 🔔


⚙️ Most Visited 🌈 Getting Started 🟣 Manjrasoft 🟣 SPE 🌐 Access via Unimelb 🌐 CLOUDS Lab

Home What is seL4? ▾ seL4 Foundation ▾ Stay in Touch Contribute ▾ Use Learn More Info ▾


The seL4[®] Microkernel

Security is no excuse for bad performance

The benchmark for performance.
The world's most highly assured OS kernel.
Open source & community-supported under the seL4 Foundation.




Security. Performance. Proof.




What is seL4?

The world's most high-assured operating system kernel



seL4 Foundation

Open source foundation for seL4 and its ecosystem. Want to join?



Stay in touch

Mailing lists, blog, contacts, chat

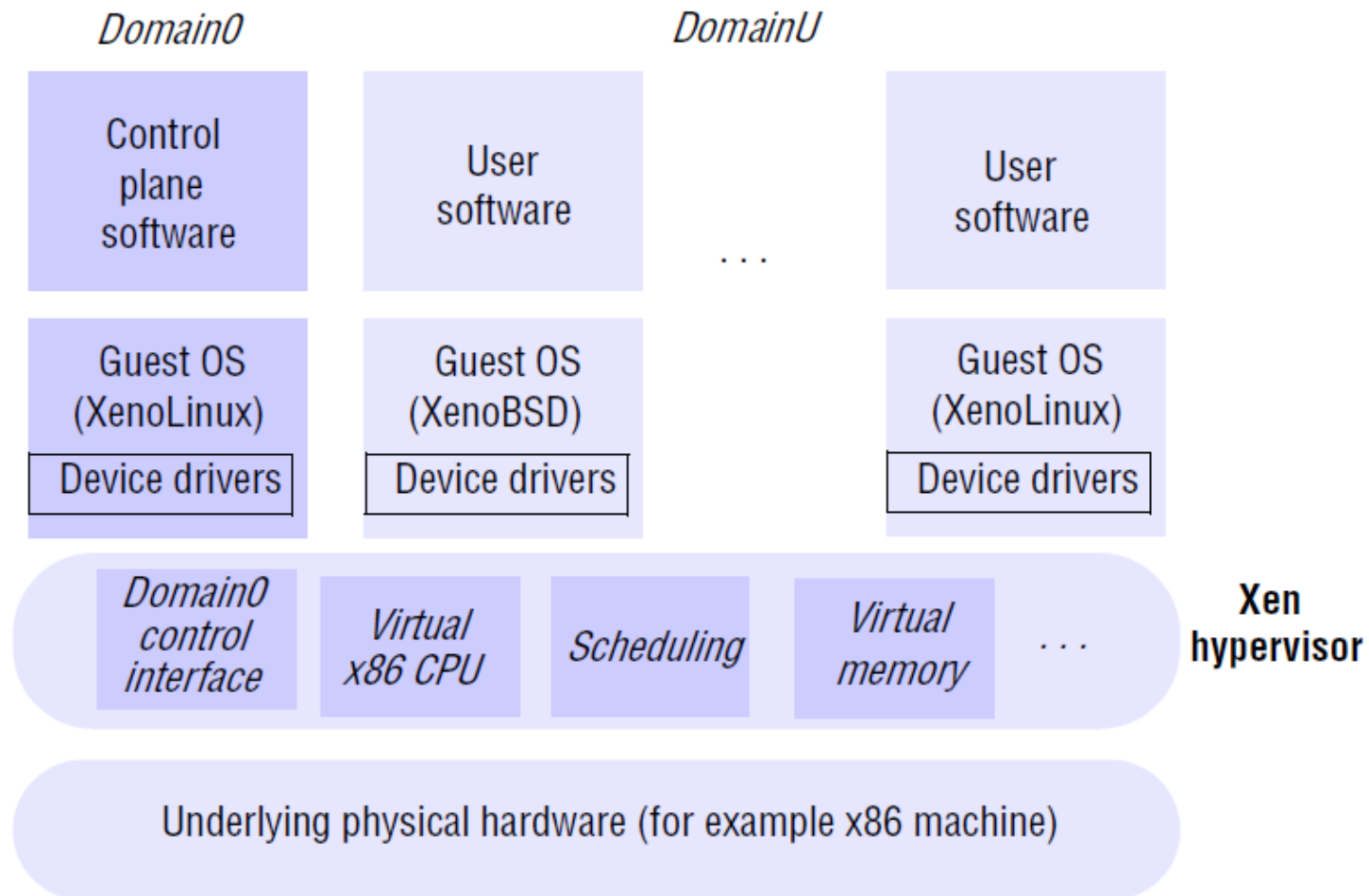
Comparison: Monolithic and Micro-kernel OS Design

- **The main advantages of a MK-based OS:**
 - A relatively small kernel is more likely to be free of bugs than one that is larger and complex.
 - Extensibility and its ability to enforce modularity behind memory protection boundaries
- **The advantage of a monolithic OS:**
 - Relative efficiency with which operations can be invoked is high because even invocation to a separate user-level address space on the same node is more costly.



- Many modern OSs follow hybrid approach in OS structure. E.g., Windows NT.
- Hybrid approaches incorporates elements of both a monolithic kernel and a microkernel system.
- Pure microkernel OSs such as Chorus & Mach have changed over time to allow servers to be loaded dynamically into the kernel address space or into a user-level address space.

OS-level Virtualization (Xen Case study)



- OSs provide various types of facilities/services to support middleware for distributed system:
 - encapsulation, protection, and concurrent access and management of node resources.
- Three types of OS:
 - Monolithic OS
 - Layered OS
 - Microkernel-based OS
- New OS designs provide flexibility in terms of separating mechanisms from policies.
- Microkernel based systems are flexible
 - Quite popular model for OS design for embedded systems
 - New Emerging optimized Kernels like *nanokernel* or *picokernel*

- Rajkumar Buyya, The Design of PARAS Microkernel, Centre for Development of Advanced Computing (C-DAC), 1998.
 - <http://www.buyya.com/microkernel/chap2.pdf>
- Gernot Heiser, Gerwin Klein, June Andronick, *seL4 in Australia: From Research to Real-World Trustworthy Systems*, Communications of the ACM, April 2020.
 - <https://cacm.acm.org/magazines/2020/4/243641-sel4-in-australia/fulltext>