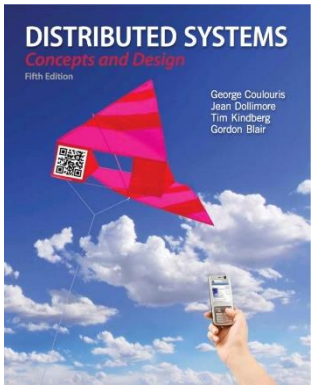
Operating System Architecture and Distributed Systems



Some concepts are drawn from Chapter 7

Dr. Rajkumar Buyya
Redmond Barry Distinguished Professor

Cloud Computing and **D**istributed **S**ystems (CLOUDS) Laboratory
School of Computing and Information Systems
The University of Melbourne, Australia
http://www.cloudbus.org/652

http://www.buyya.com/microkernel/chap2.pdf

Operating System Architecture and Distributed Systems (DS)

- 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

Open DS and System Software

- A open DS should make it possible to:
  - Run only that ("specific" components 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 particular 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.

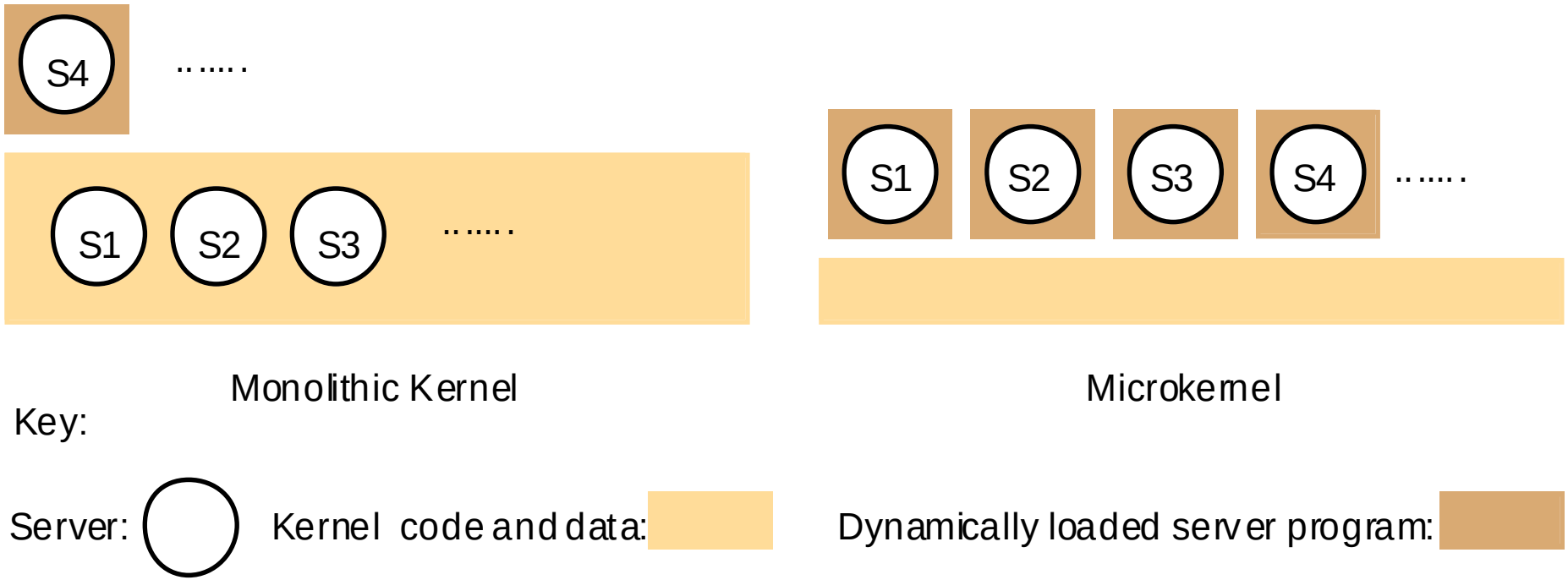
Separating Mechanisms and Policies in OS and DS

- 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.

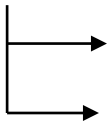
OS/Kernel Architecture

- 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.
- In literature, we find predominantly 3 types of OSs:
  - Monolithic OS
  - Layered OS
  - Microkernel-based OS

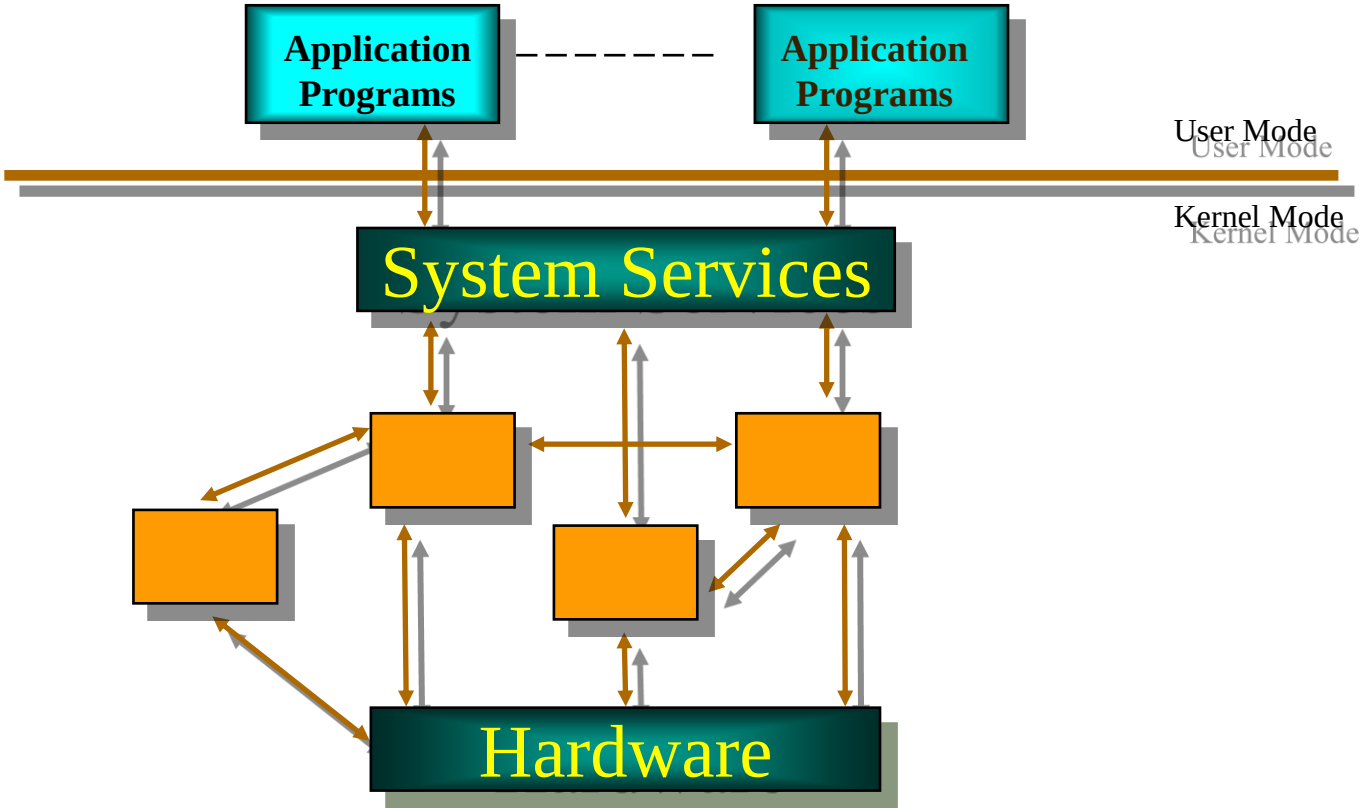
Monolithic kernel and microkernel



Operating System Models

- 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.

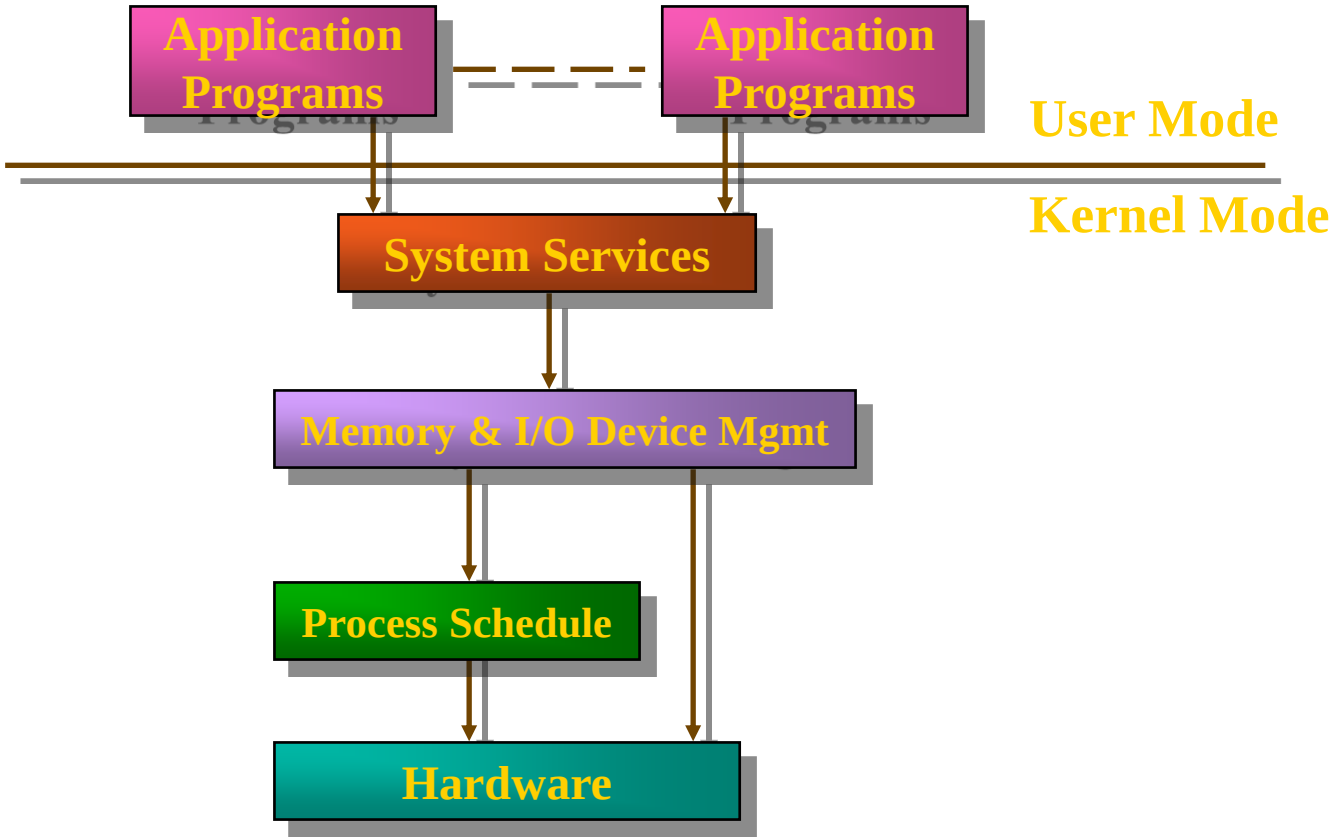
Monolithic Operating System



- **Better application Performance**
- **Difficult to extend**

Ex: MS-DOS

Layered OS



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

Ex : UNIX

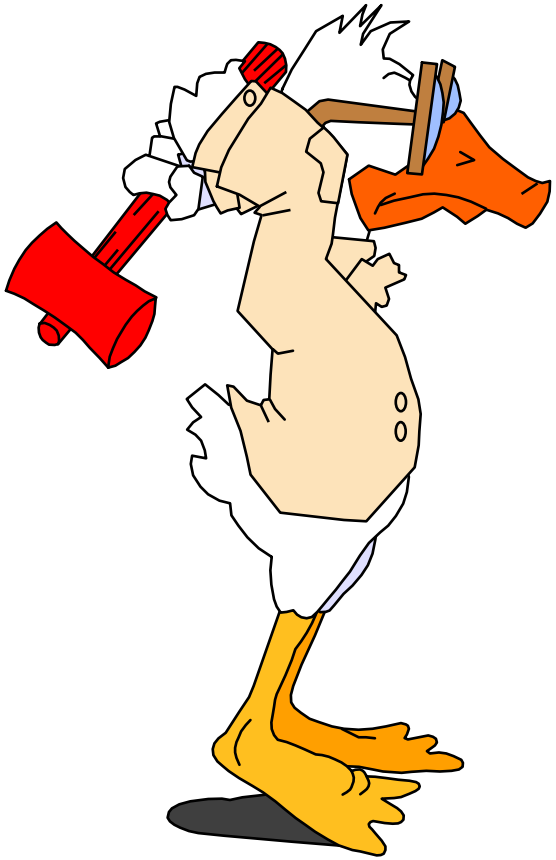
Traditional OS

Application Programs

Application Programs

User Mode

Kernel Mode



OS Designer

Disadvantages of Monolithic OS

- 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.

New trend in OS design: Separating mechanisms and policies

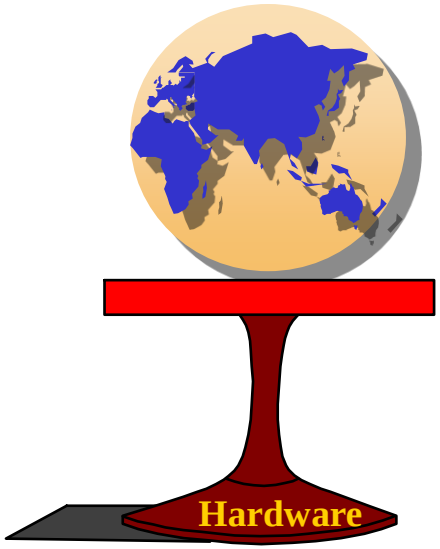
Application Programs

Servers

Application Programs

User Mode

Kernel Mode

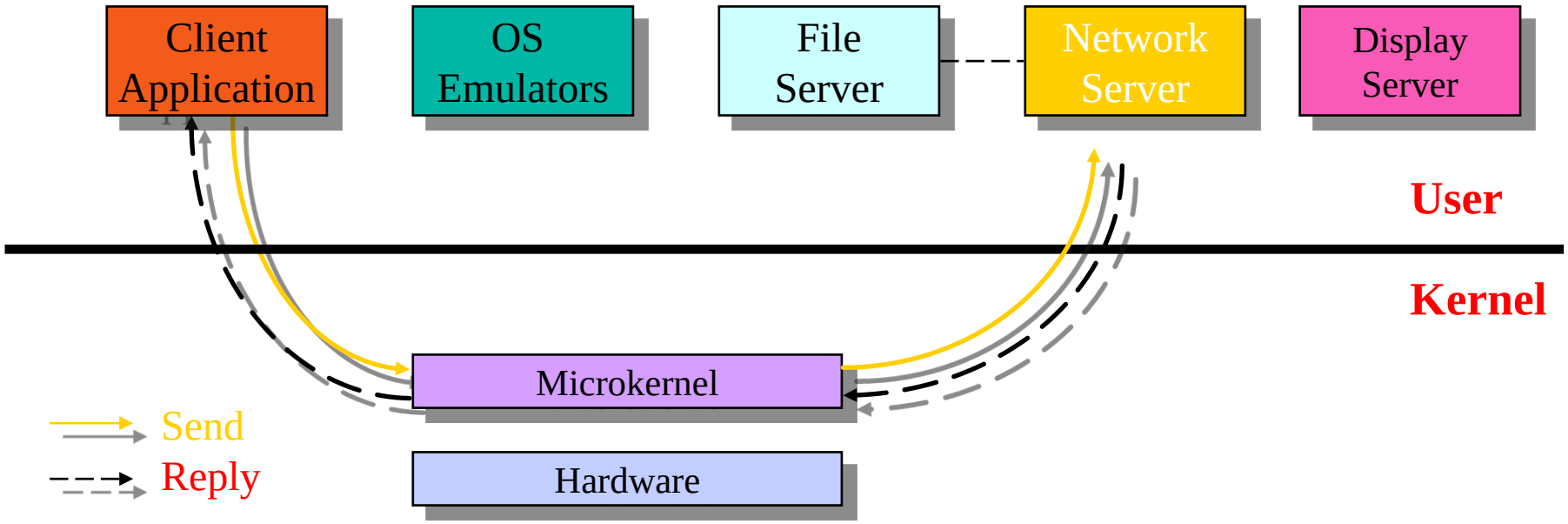


Microkernel
(very basic functions)

Micro-kernel

- 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.

Microkernel/Client Server OS

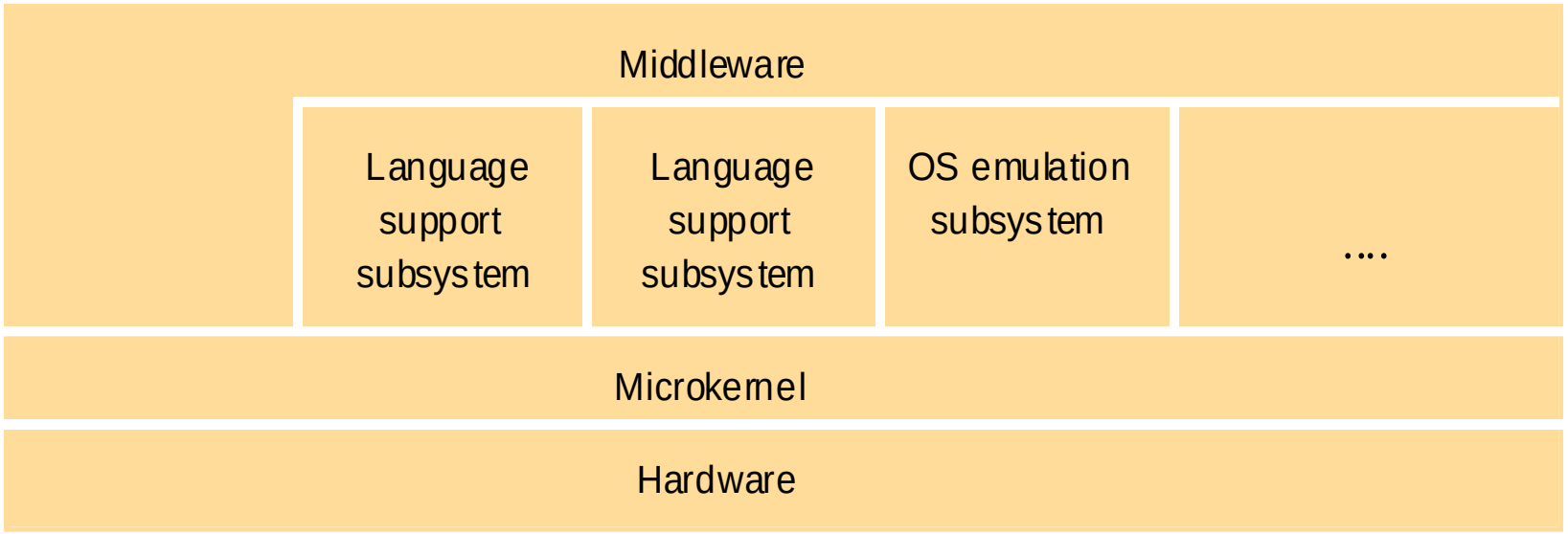


- Tiny OS kernel providing basic primitive (process, memory, IPC)
- Traditional services becomes subsystems
- OS = Microkernel + User Subsystems

Ex: Mach, QNX, Windows NT!

The role of the microkernel (MK)

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

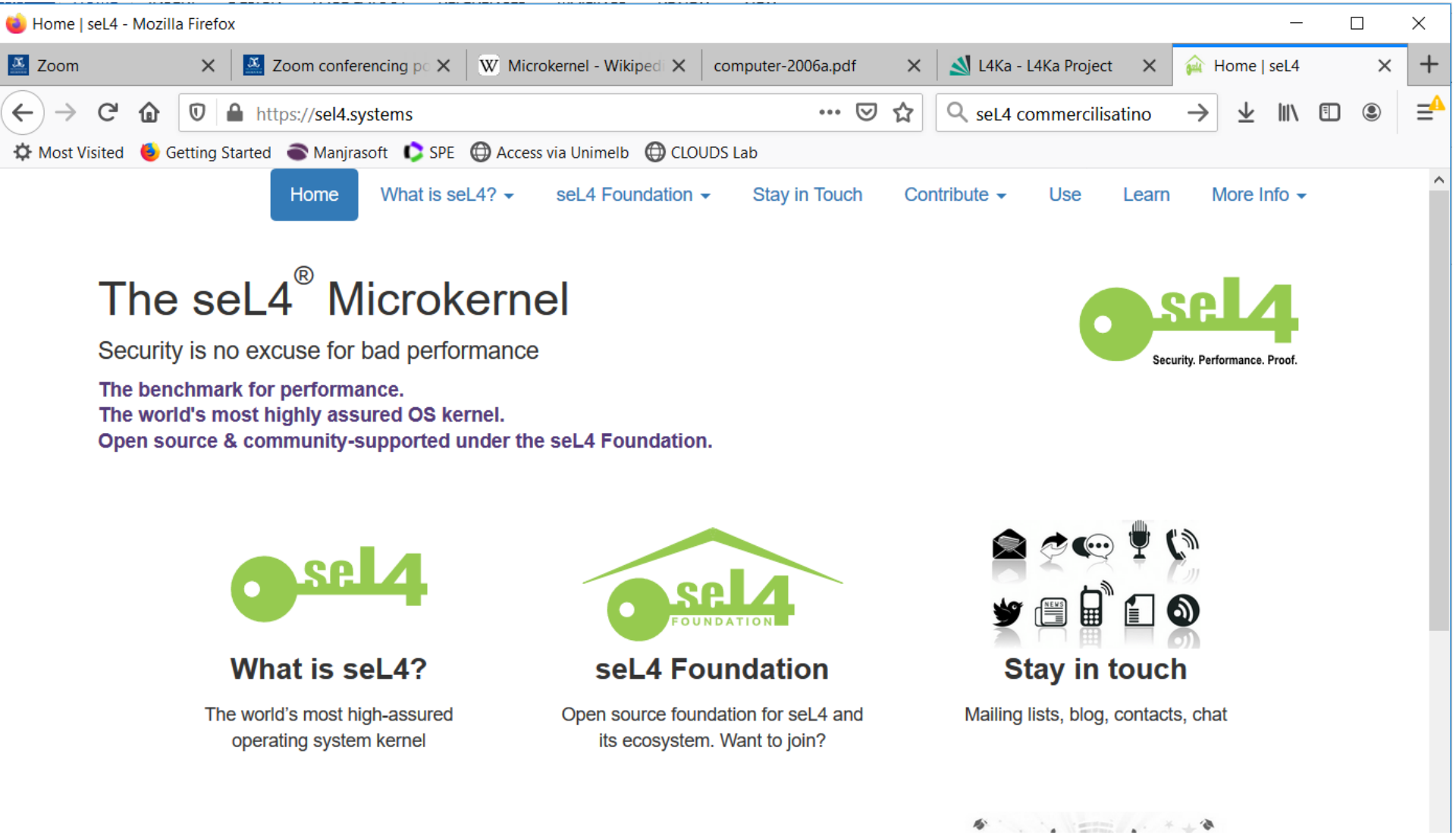


The microkernel supports middleware via subsystems

Few Popular Microkernel Systems

- MACH, CMU (Carnegie Mellon University)
  - supports OS emulators such as Unix and OS/2.
- PARAS (C-DAC, India) for PARAM Supercomputers
- ChorusOS (Sun, USA) Realtime OS (RTOS)
- seL4 (created by NICTA/Data61, Australia)
- 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.

seL4: Made in Australia!



Comparison: Monolithic and Micro-kernel OS Design

- The main advantages of a MK-based OS:
  - A relative 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.

Hybrid Approaches

- Many modern OSs follow hybrid approach in OS structure. E.g., Windows NT.
- 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.
- Some OSs (such as SPIN) use event-based model as a mechanism for interaction between modules grafted into the kernel address space.

Summary

- 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*

Additional References

- 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