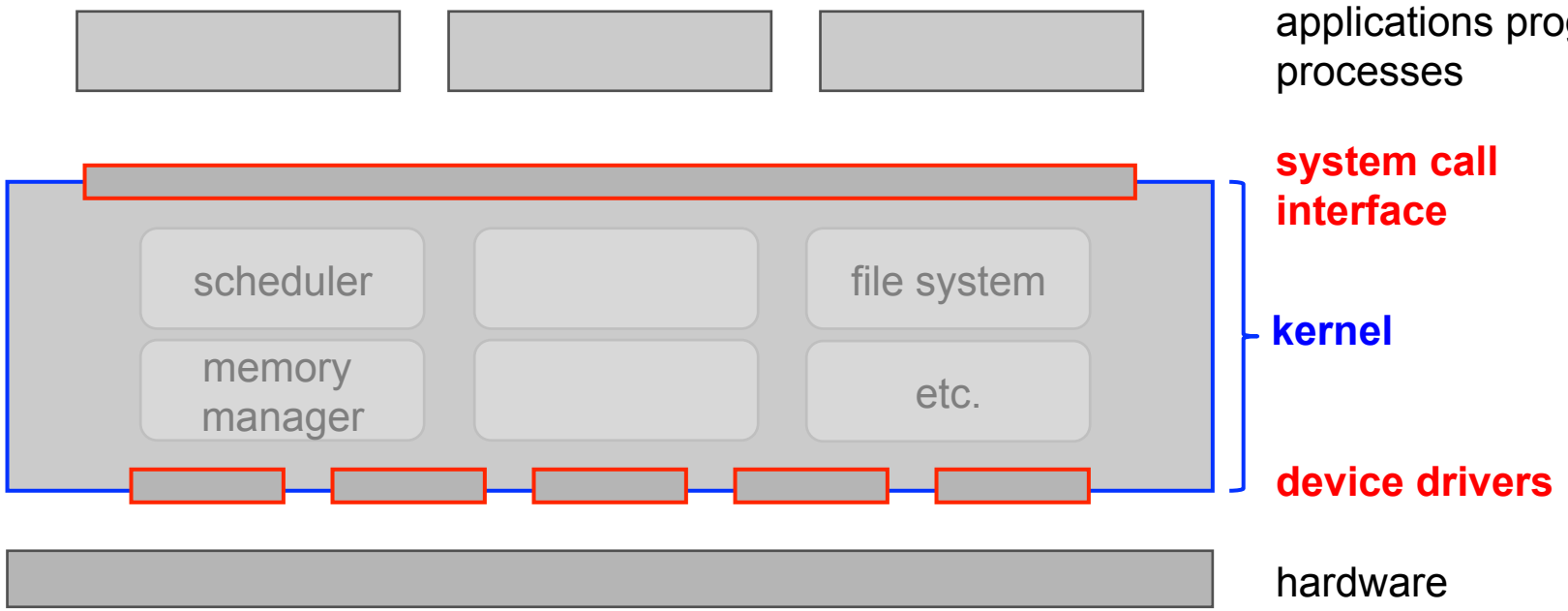
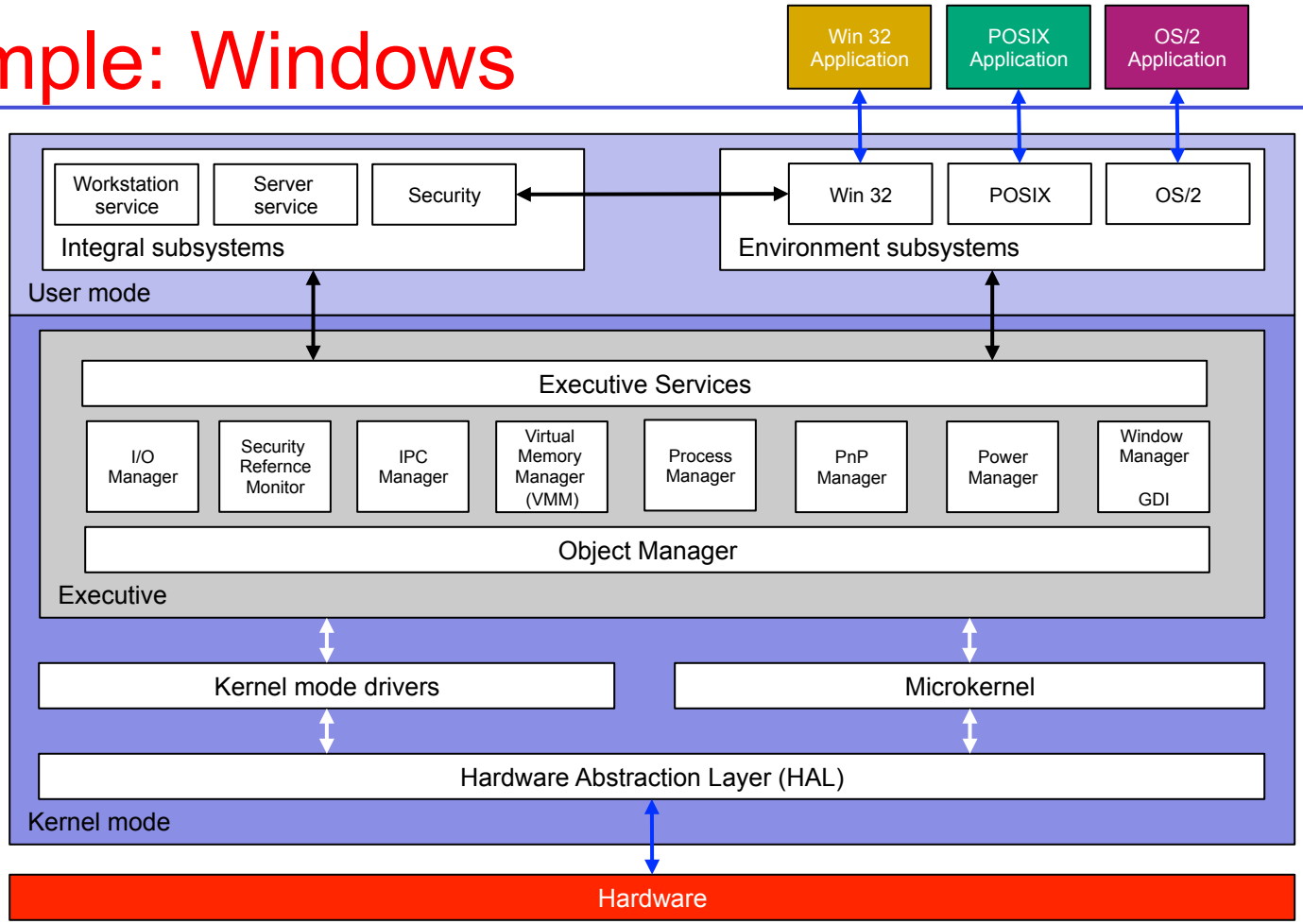
Operating Systems Structure

1. External Structure
2. Operating System Interfaces
   - System Calls
   - Device Drivers
3. Internal Structure: Layering and Monolithic Kernels
4. Alternative Structures:
   - Microkernels
   - Exokernels and Library Operating Systems

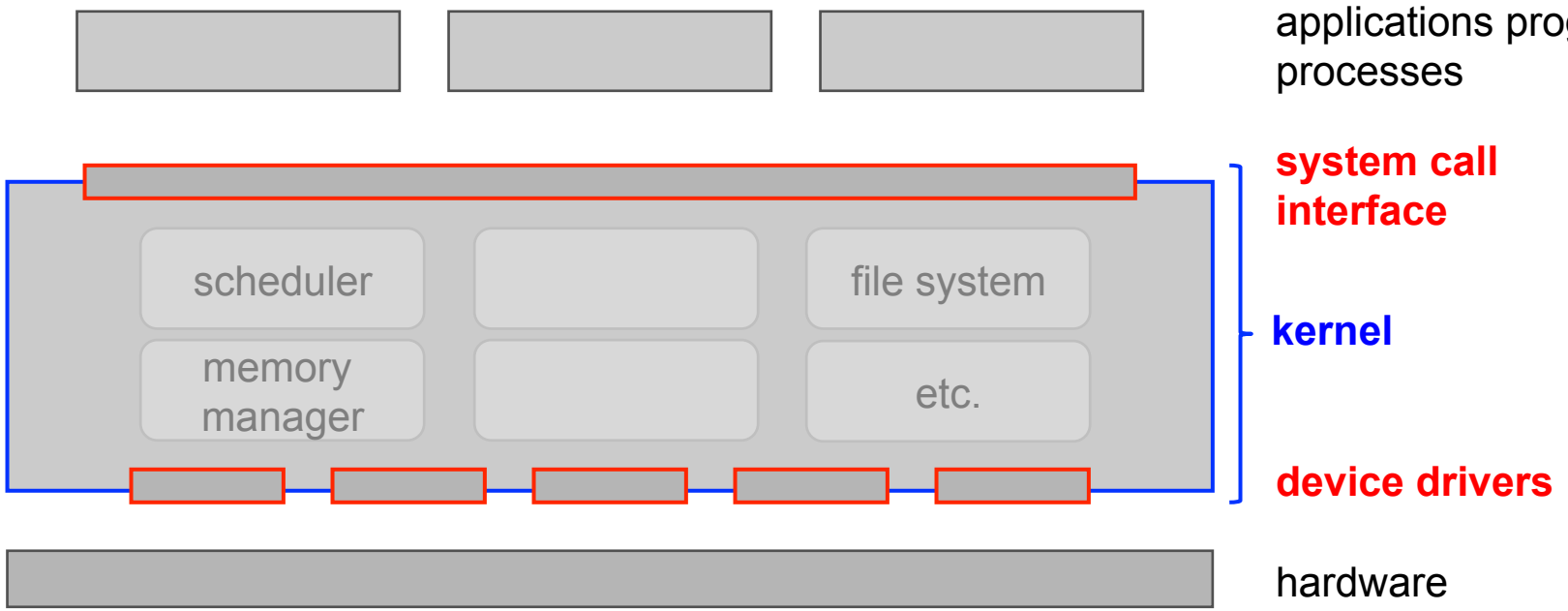
External Structure of an OS



Example: Windows

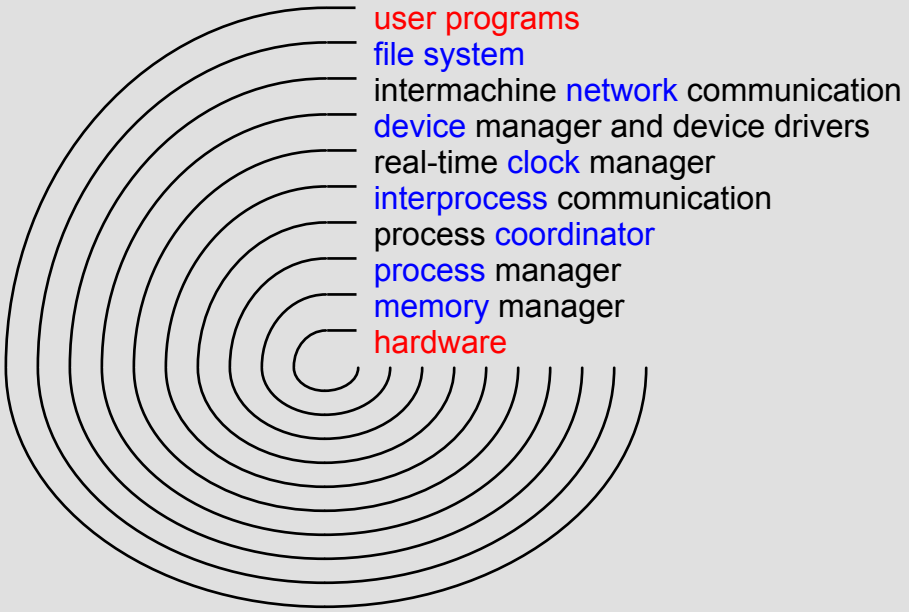


Internal Structure of an OS



Internal Structure: Layered Services

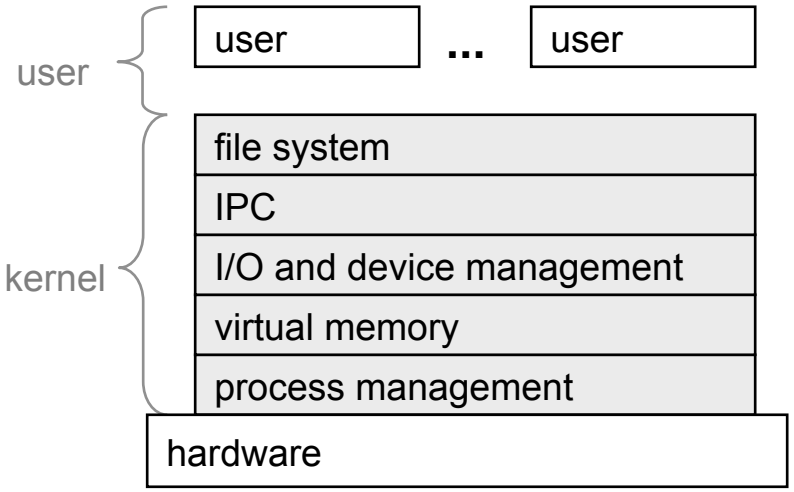
The "Onion View" of the OS.



Xinu [Comer 1984]

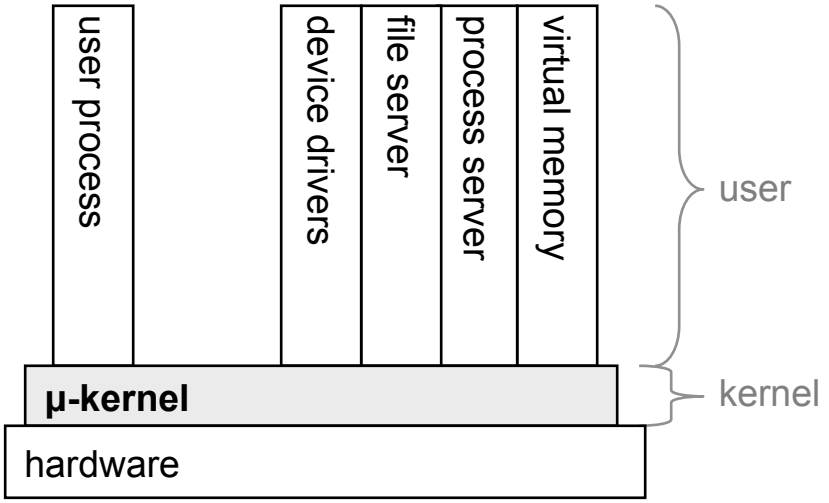
Internal Structure: Monolithic *vs.* µ-Kernels

Layered Kernel



Hierarchical decomposition.
Leads to monolithic kernels.

Microkernel

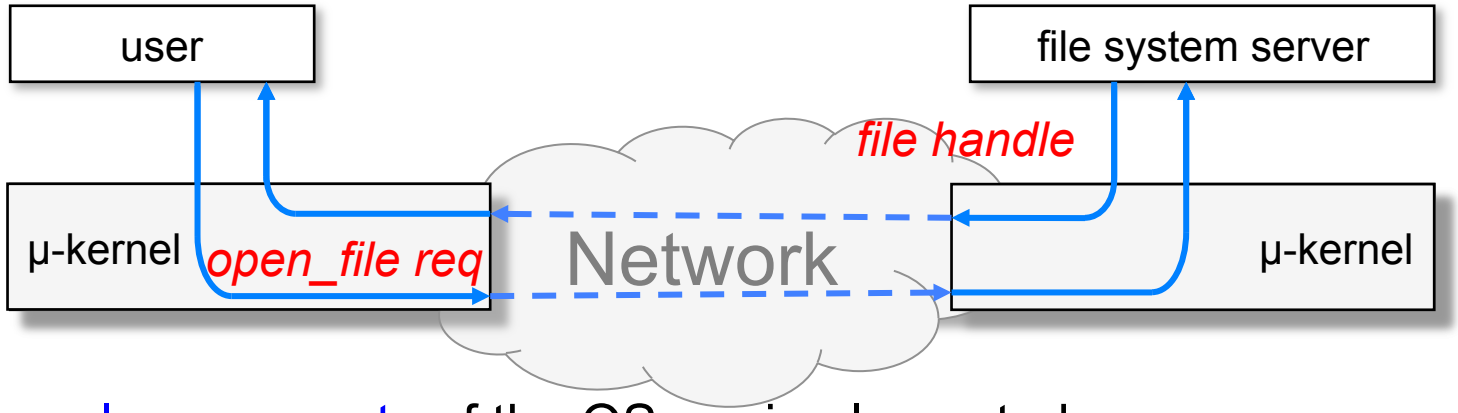


Kernel has only core operating system functions.
Other functions run in *server processes* in user space.

Operations in a µ-Kernel

- Non-kernel components of the OS are implemented as server processes.
- Communication between user and servers using messages through kernel.
- "Client-server computing in within a single computer."
- Examples: Mach, Windows NT/2000/XP/7/8/10/…, Chorus, L4, …

Client-Server Computing with μ-Kernel

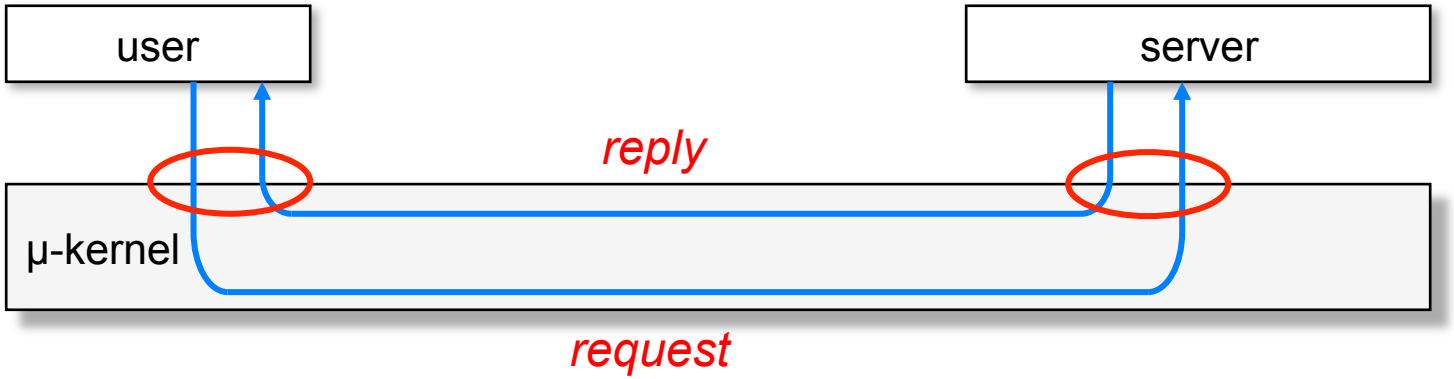


- Non-kernel components of the OS are implemented as server processes.
- Communication between user and servers using messages through kernel.
- "Client-server computing in within a single computer."
- Examples: Mach, Windows NT/2000/XP/7/8/10/…, Chorus, L4, …

Benefits of μ-Kernels

- Extensibility:
  - New services can be added by adding server processes.
- Flexibility:
  - Services can be customized.
- Portability:
  - Kernel small, with well-defined interface.
- Robustness:
  - Servers can fail “locally”
- Distributed System Support:
  - Interface between users and services is message-based.

µ-Kernels: Performance is Problem

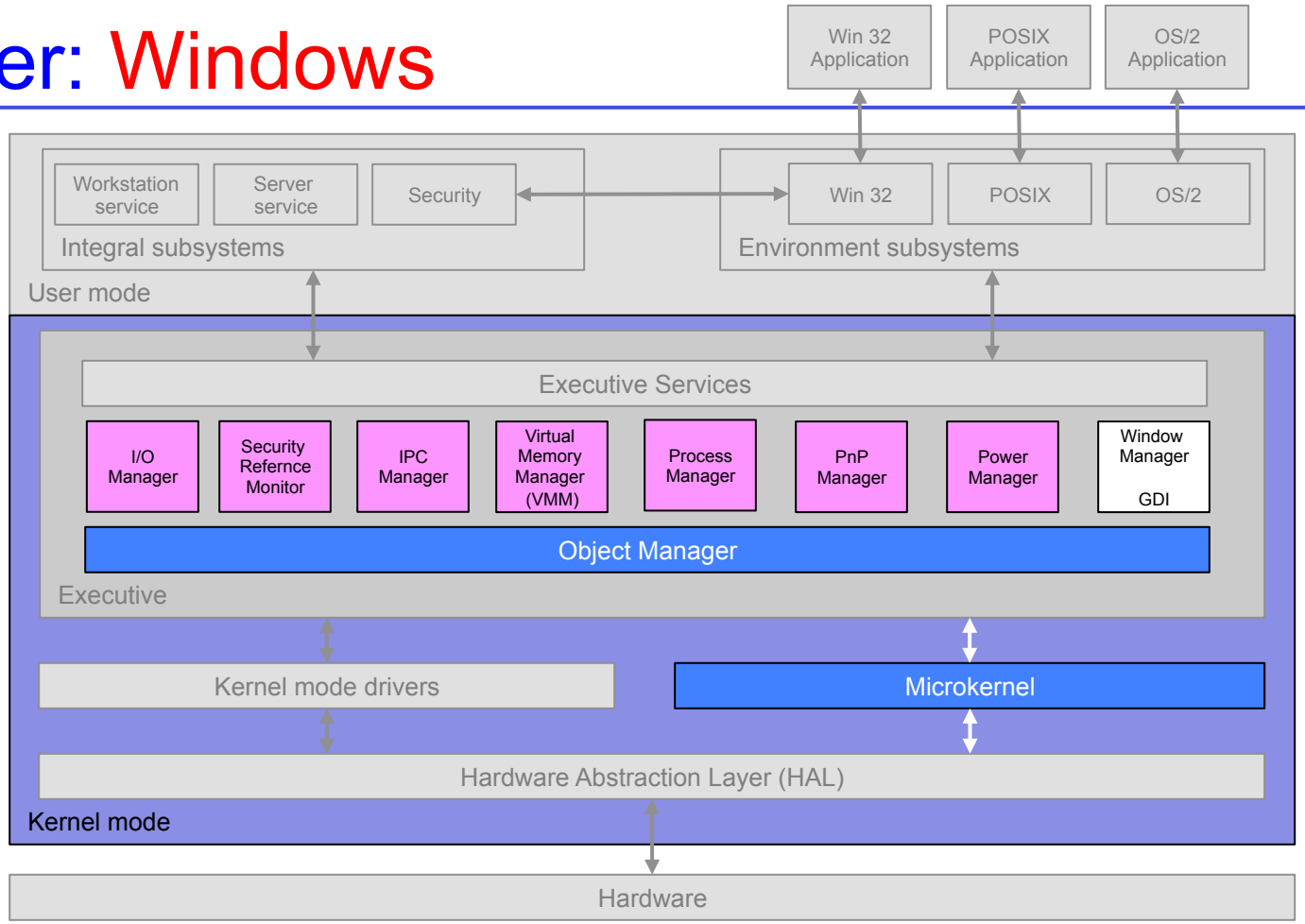


Request traverses user/kernel boundary twice, same for reply.

Solution approaches:

- Move critical services back into the kernel (“make kernel bigger”)
- Make kernel “smaller”

Bigger: Windows



Smaller: Exokernel

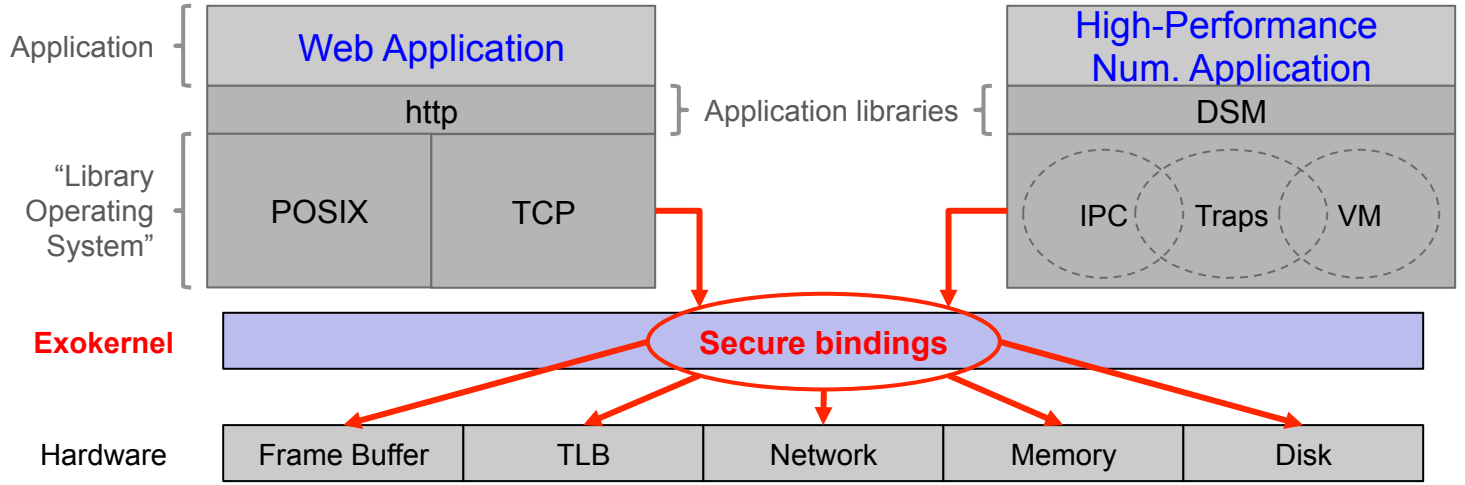
Claim: Providing abstractions of resources or virtual machines for applications to execute is the wrong approach!

The Exokernel approach: Export hardware resources securely (!) instead of emulating them.

Expose everything about resources, but do it securely, e.g.:

- Allow application to access HW resources as efficiently as possible.
- Export physical names of resources (e.g., addresses)
- Allow applications to allocate specific resources

Exokernel: Secure Binding



Library OSs manage hardware resources.

Exokernel protects Library OSs from each other.

Summary: Operating Systems Structure

1. External Structure
2. Operating System Interfaces
   - System Calls
   - Device Drivers
3. Internal Structure: Layering and Monolithic Kernels
4. Alternative Structures:
   - Microkernels
   - Exokernels and Library Operating Systems