Operating System

Ch12: I/O systems

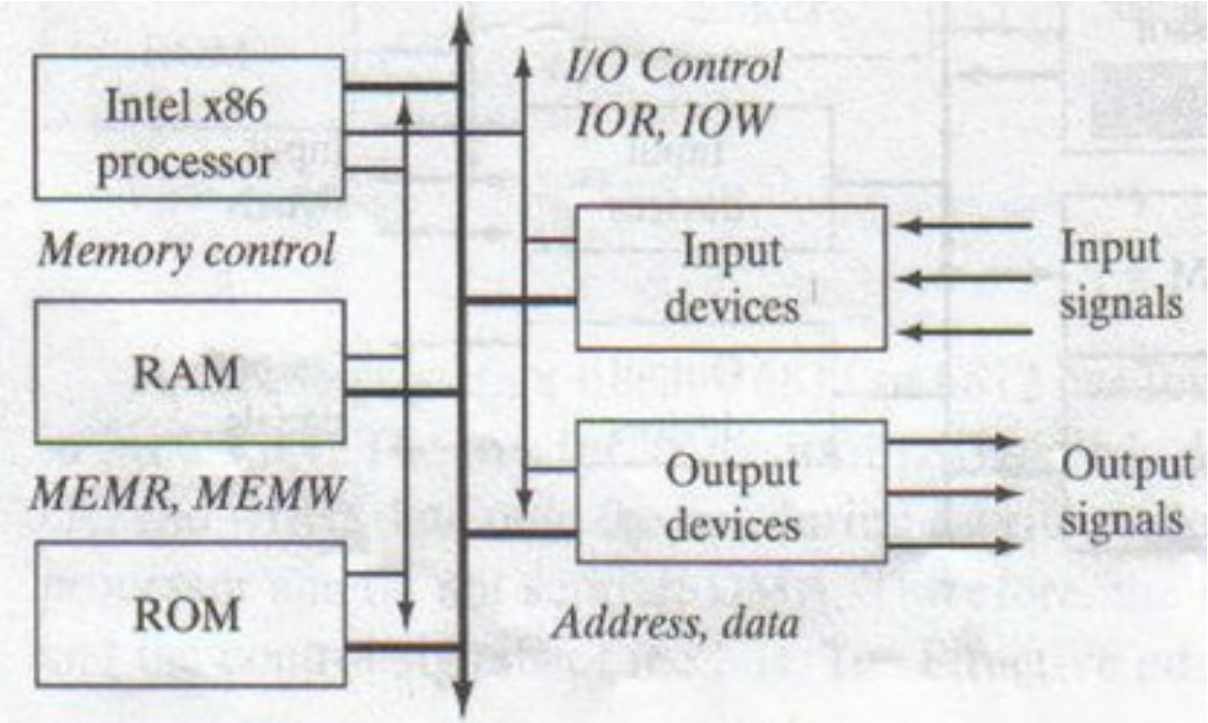
BeomSeok Kim
Department of Computer Engineering
KyungHee University
passion0822@khu.ac.kr

Revisited: Computer System Operation

- I/O operation
  - ✓ I/O request via I/O instruction
    - ➢ Direct I/O vs. Memory-mapped I/O
    - ➢ Communicates with registers in I/O controller
    - ➢ Typically, IR (Instruction Register) & DR (Data Register)
  - ✓ I/O method
    - ➢ Programmed I/O
    - ➢ Interrupt
    - ➢ DMA (Direct Memory Access)

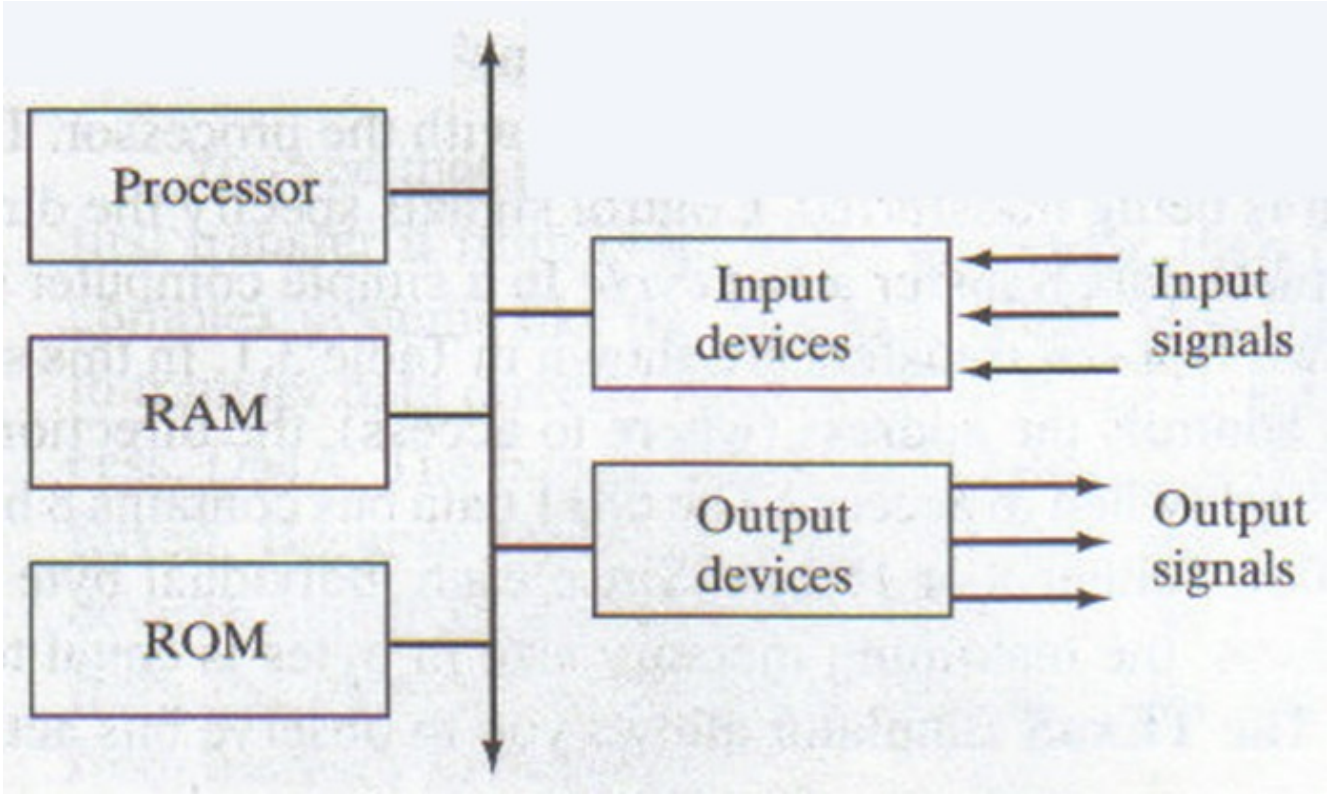
Direct I/O

- Separate addresses: memory vs. I/O
- Separate instructions
  - ✓ MEMR / MEMW (Load / Store)
  - ✓ IOR / IOW (IN / OUT)

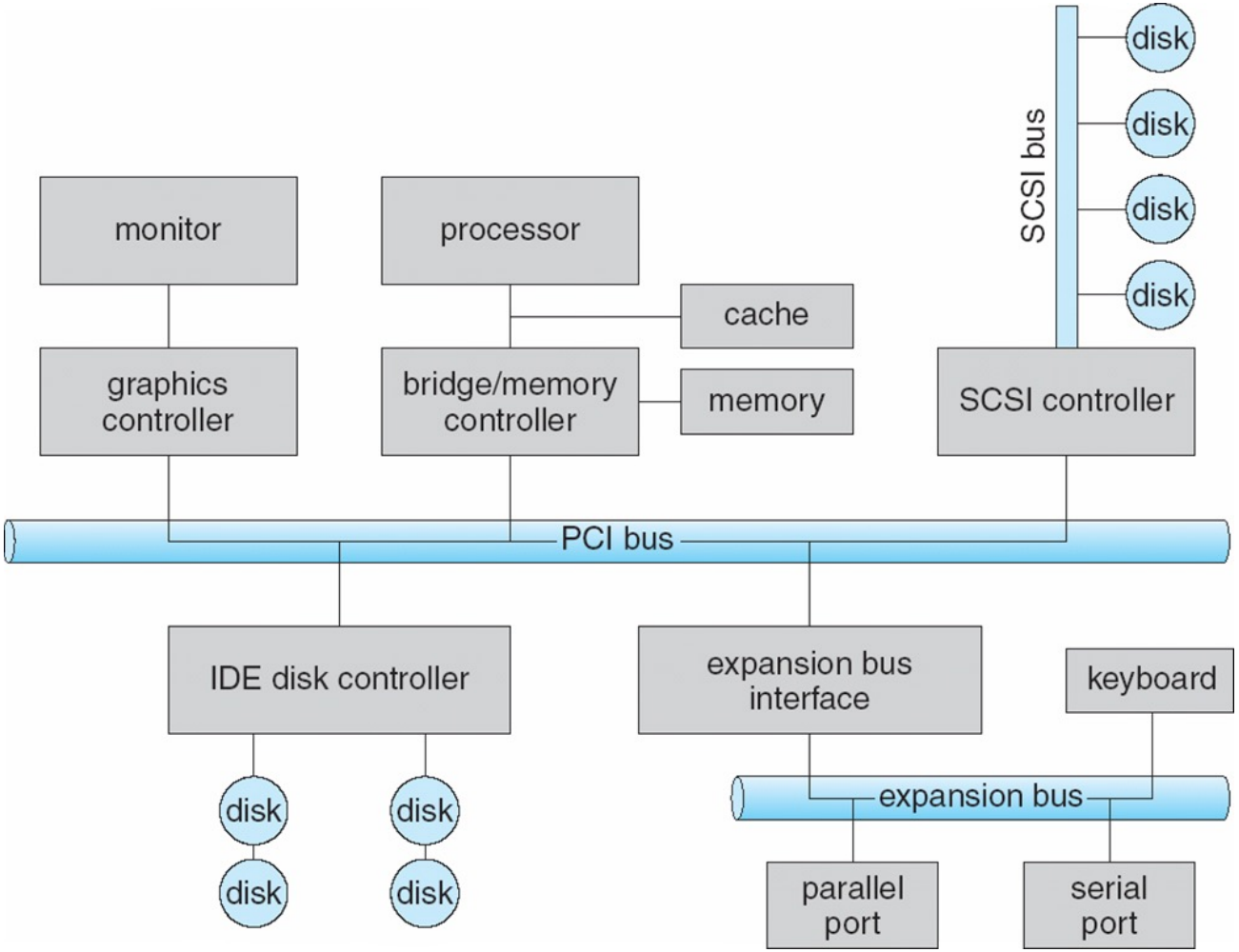


Memory-Mapped I/O

- Integrated addresses: memory & I/O
- Integrated instructions
  - ✓ MEMR / MEMW (Load / Store)



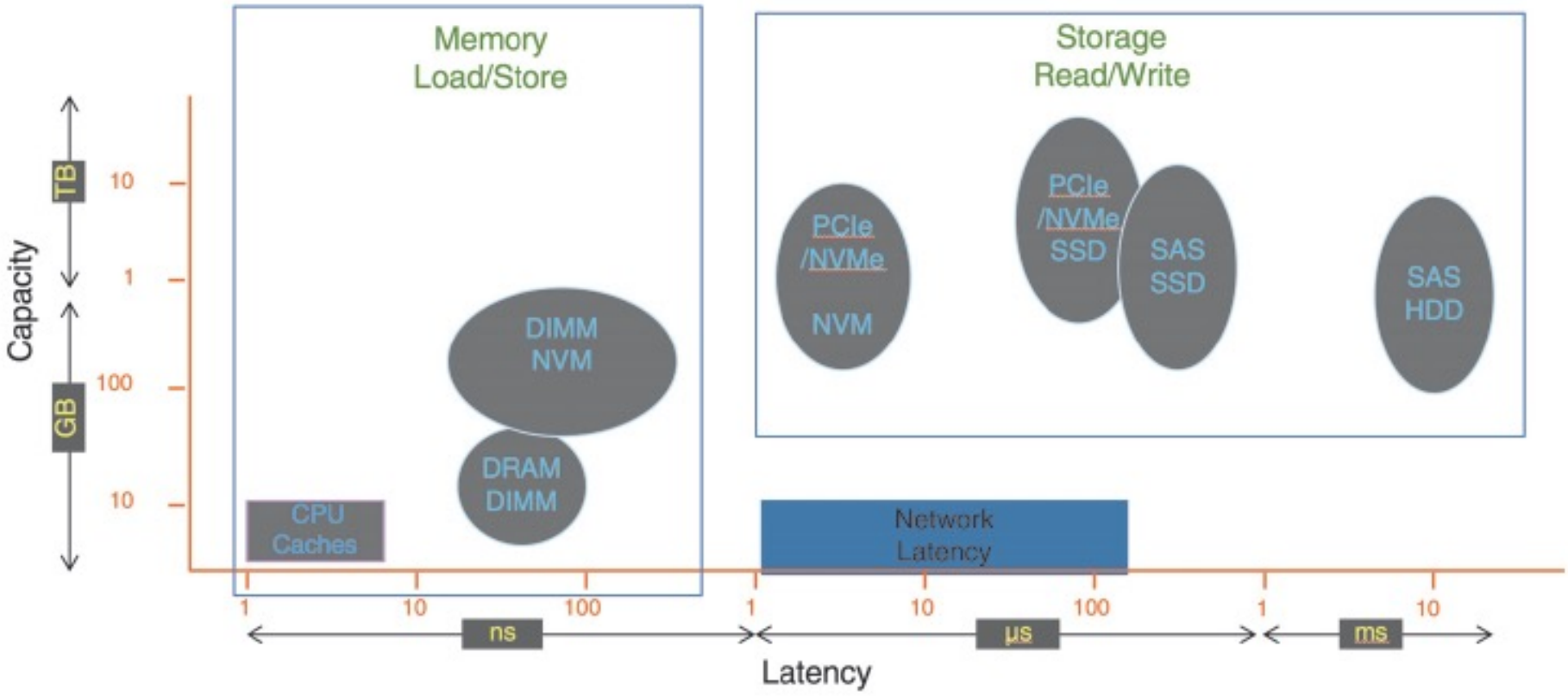
A Typical PC Bus Structure



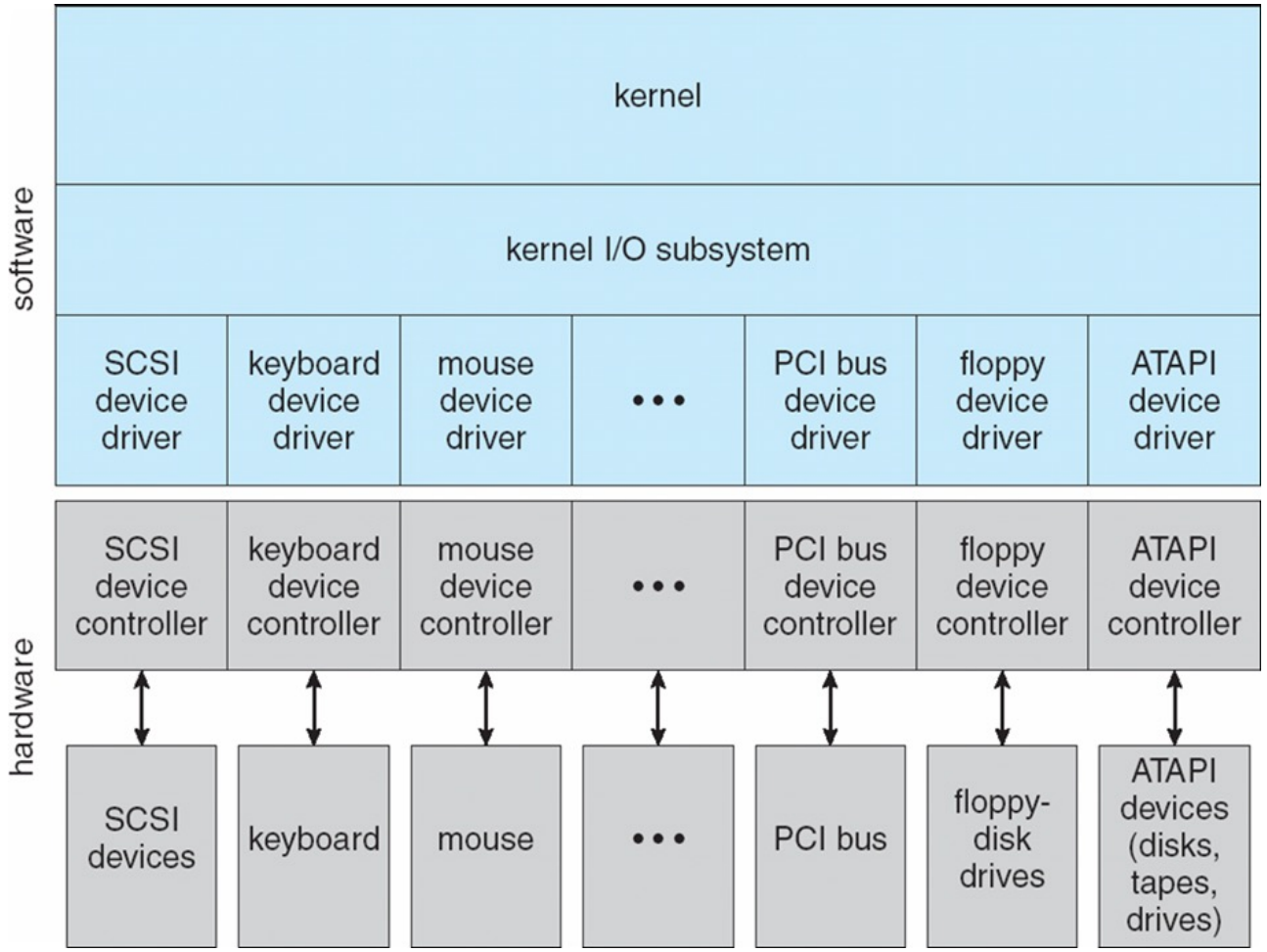
Device Controller

- Device controller (or host adapter)
  - ✓ I/O devices have components:
    - ➢ Mechanical component
    - ➢ Electronic component
  - ✓ The electronic component is the device controller
    - ➢ May be able to handle multiple devices
  - ✓ Controller's tasks
    - ➢ Convert serial bit stream to block of bytes
    - ➢ Perform error correction as necessary
    - ➢ Make available to main memory

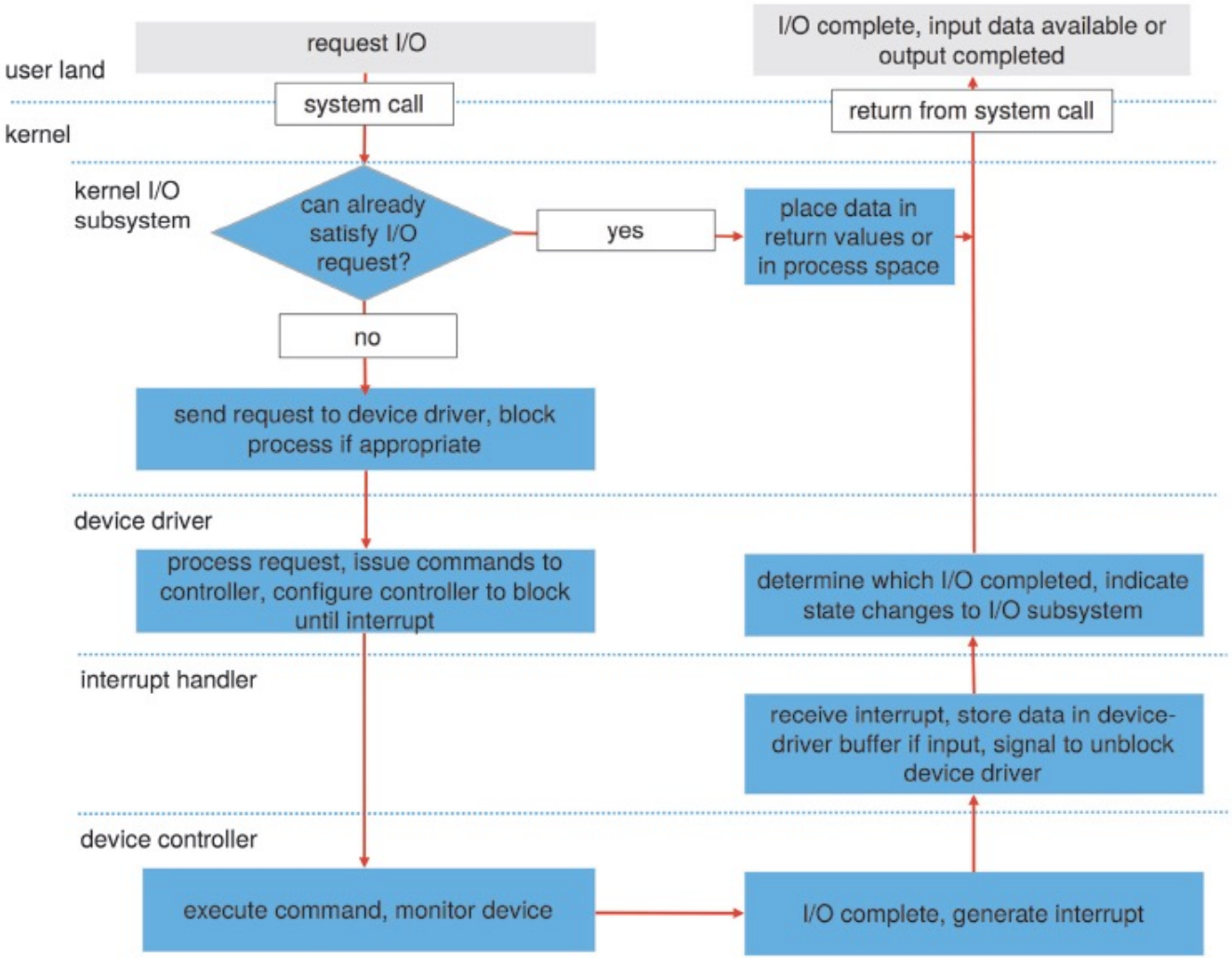
I/O Performance of Storage and Network Latency



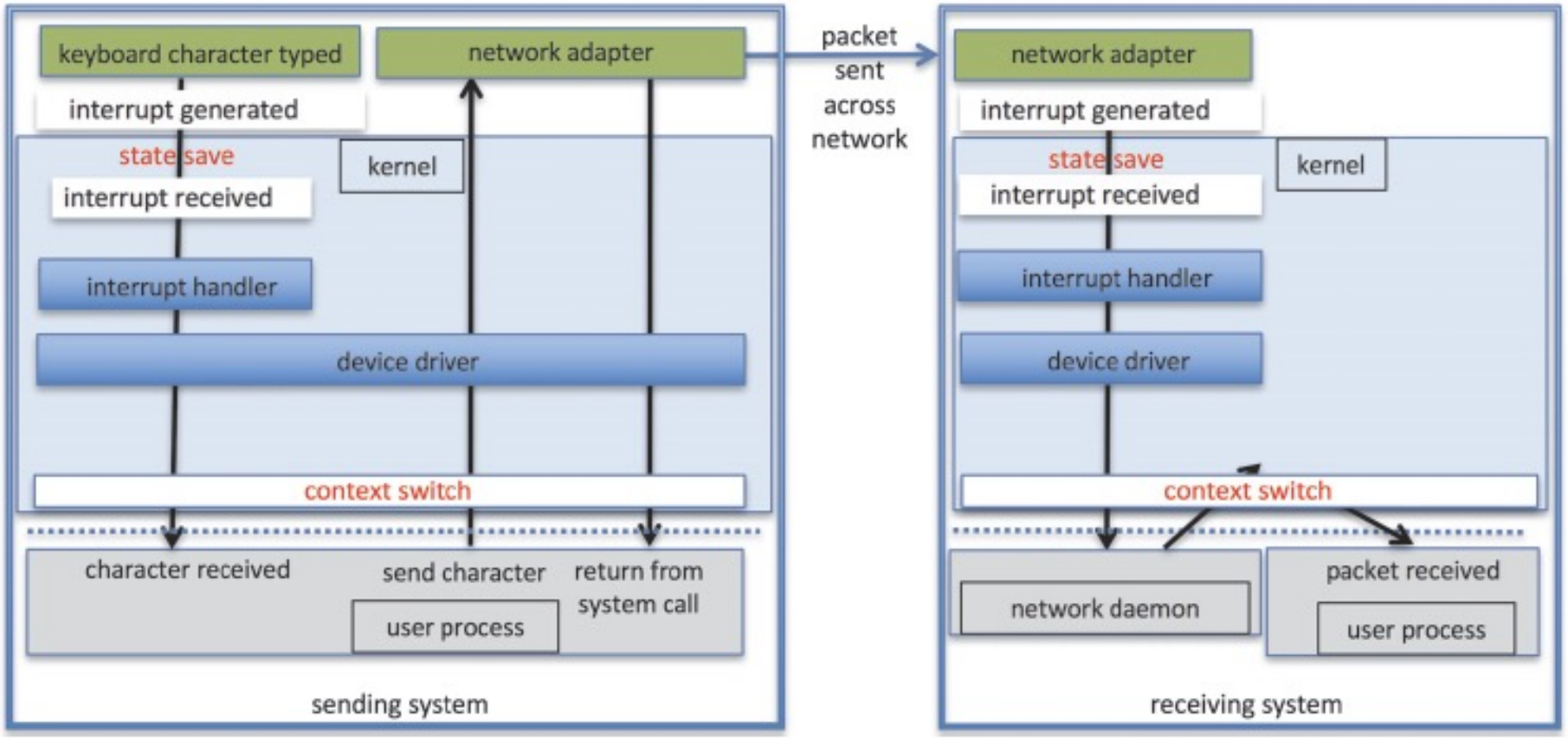
Kernel I/O Structure



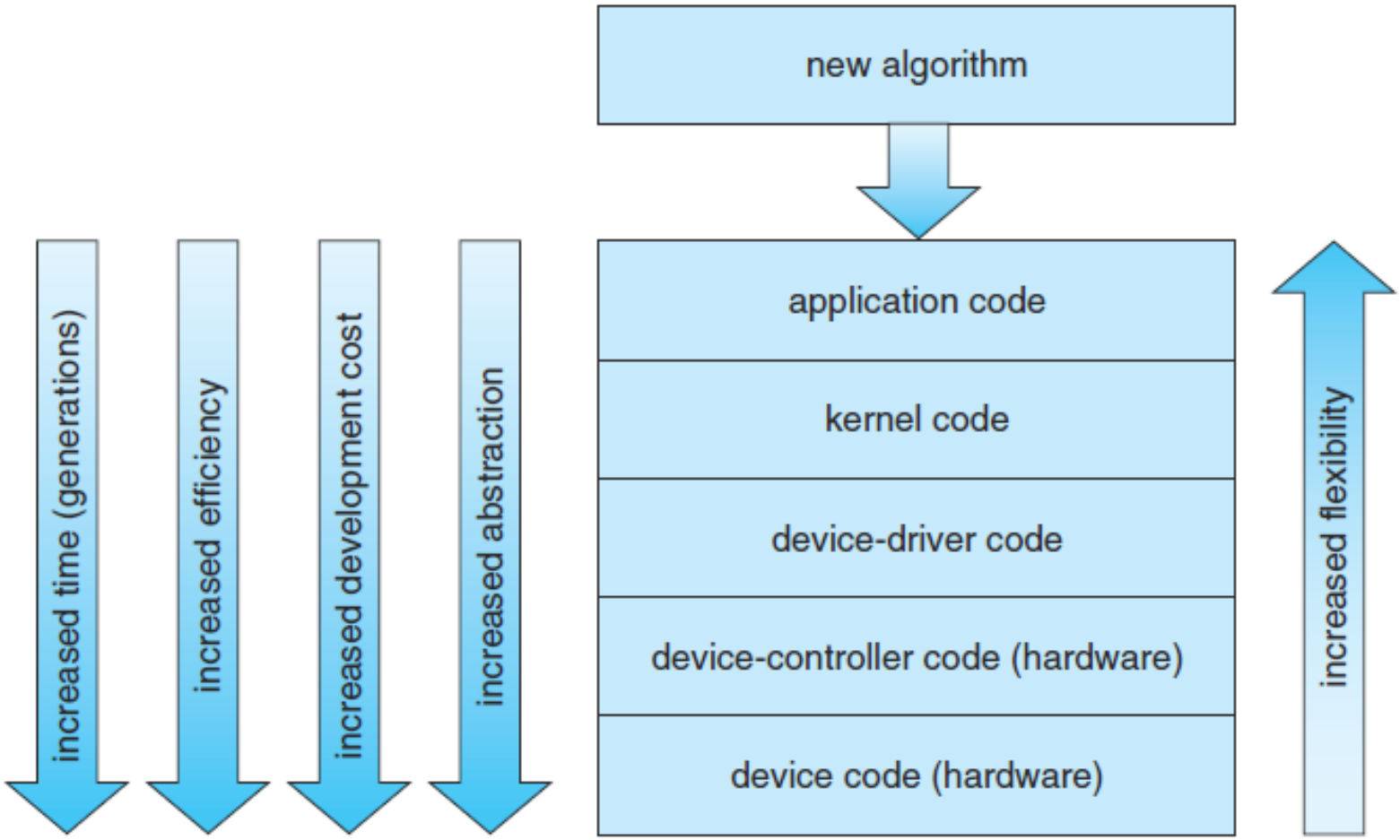
Life Cycle of An I/O Request



Intercomputer Communications



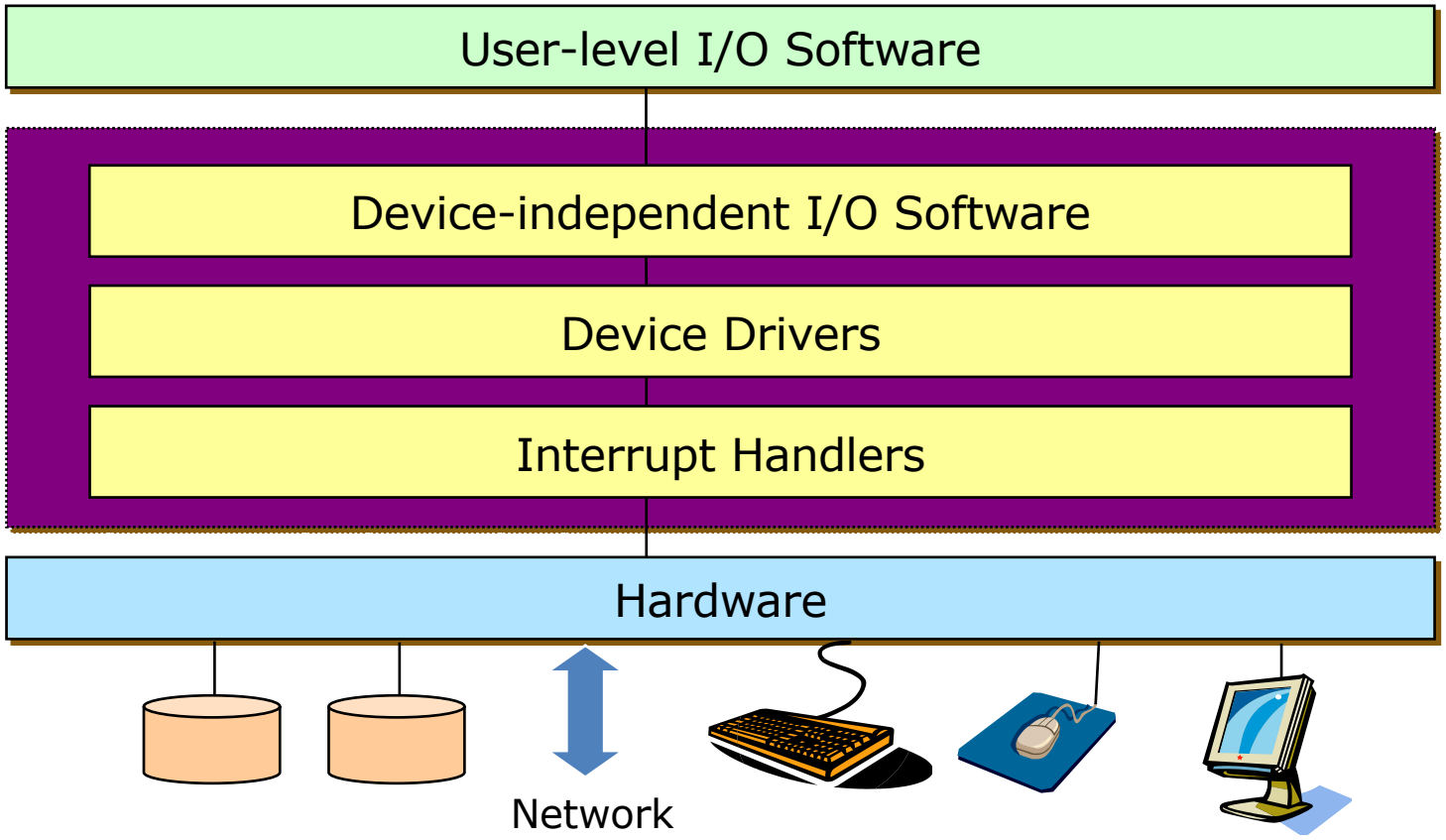
Device-Functionality Progression



Goals of I/O Software

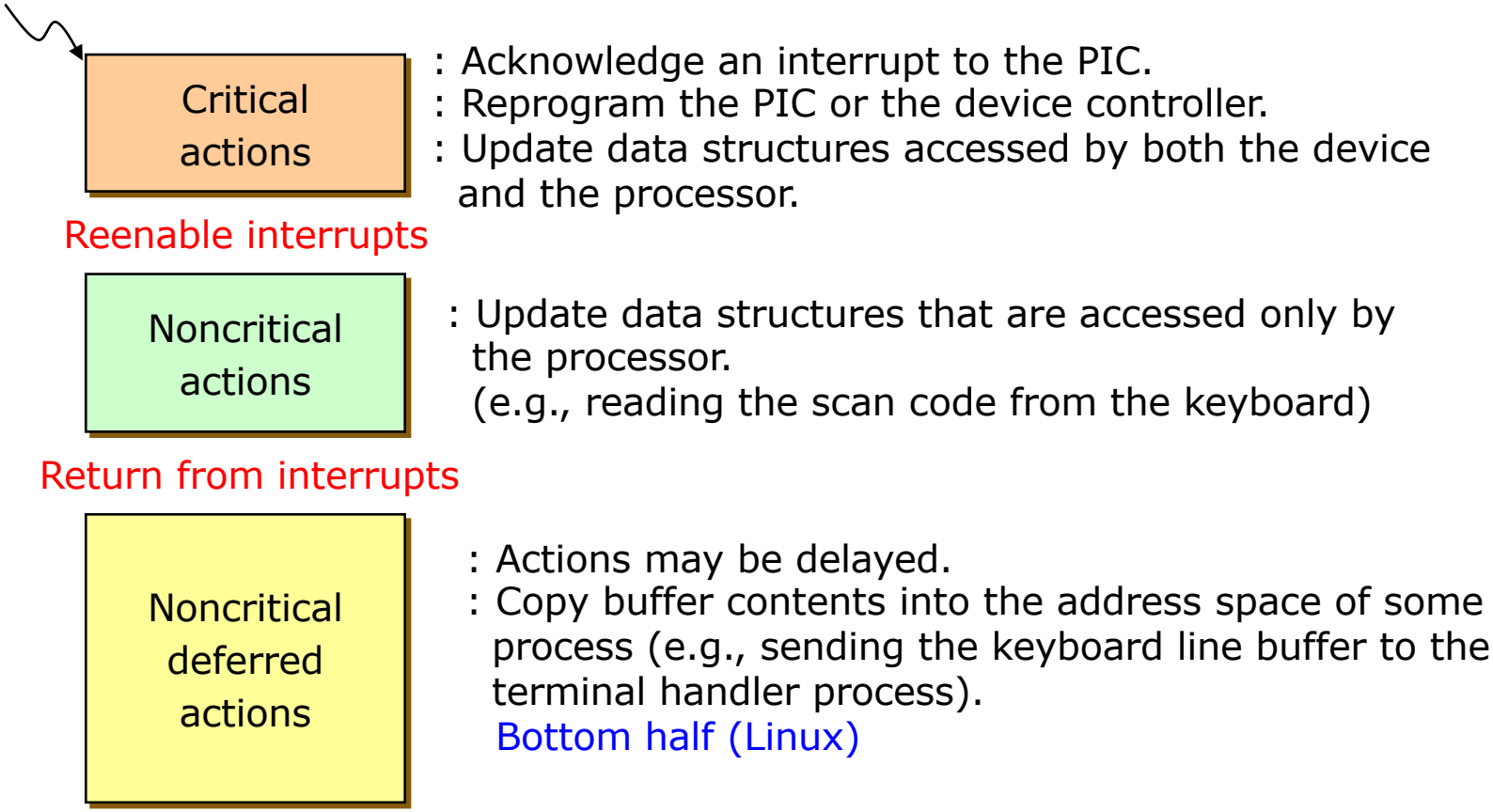
- Goals
  - ✓ Device independence
    - ➢ Programs can access any I/O device without specifying device in advance
  - ✓ Uniform naming
    - ➢ Name of a file or device should simply be a string or an integer
  - ✓ Error handling
    - ➢ Handle as close to the hardware as possible
  - ✓ Synchronous vs. asynchronous
    - ➢ blocked transfers vs. interrupt-driven
  - ✓ Buffering
    - ➢ Data coming off a device cannot be stored in final destination
  - ✓ Sharable vs. dedicated devices
    - ➢ Disks vs. tape drives
    - ➢ Unsharable devices introduce problems such as deadlocks

I/O Software Layers



Interrupt Handlers

- Handling interrupts



Device Drivers

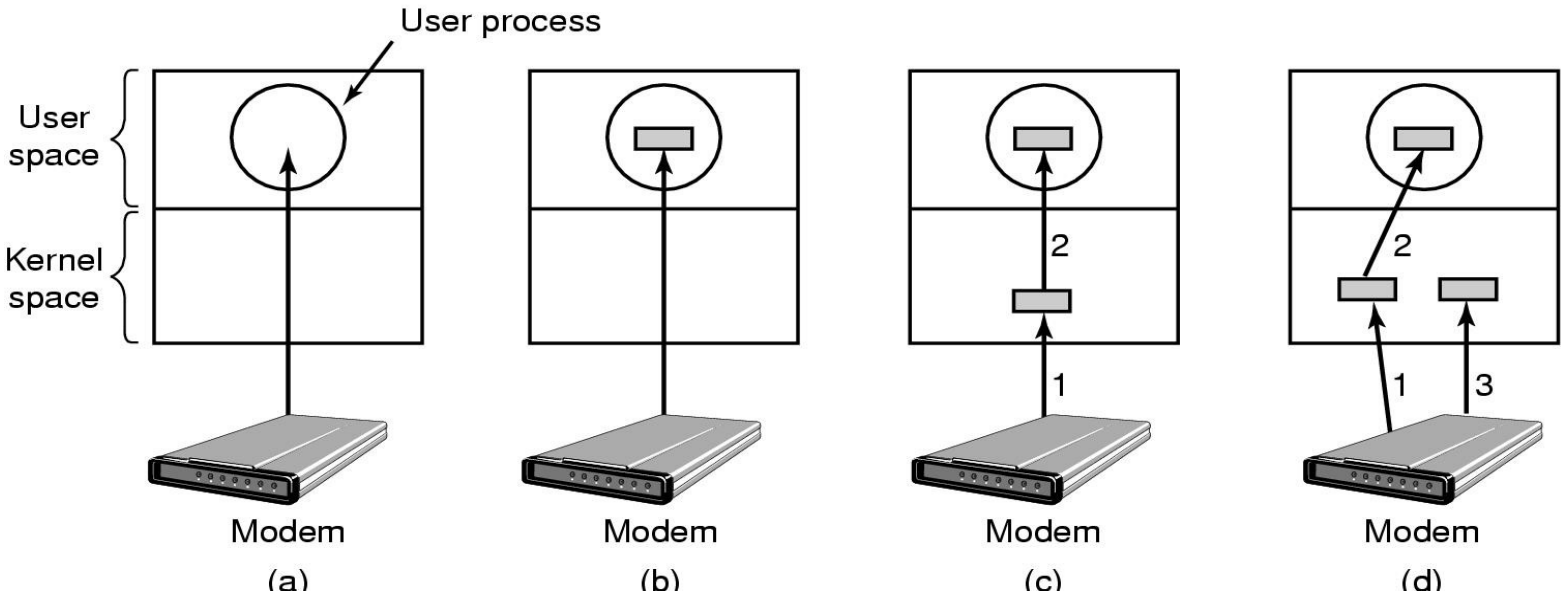
- Device drivers
  - ✓ Device-specific code to control each I/O device interacting with device-independent I/O software and interrupt handlers
  - ✓ Requires to define a well-defined model and a standard interface of how they interact with the rest of the OS
  - ✓ Implementing device drivers:
    - ➢ Statically linked with the kernel
    - ➢ Selectively loaded into the system during boot time
    - ➢ Dynamically loaded into the system during execution (especially for hot pluggable devices)

Device-Independent I/O Software

- Uniform interfacing for device drivers
  - ✓ In Unix, devices are modeled as special files
    - ➢ They are accessed through the use of system calls such as open(), read(), write(), close(), ioctl(), etc.
    - ➢ A file name is associated with each device
  - ✓ Major device number locates the appropriate driver
    - ➢ Minor device number (stored in i-node) is passed as a parameter to the driver in order to specify the unit to be read or written
  - ✓ The usual protection rules for files also apply to I/O devices

Device-Independent I/O Software

- Buffering
  - ✓ (a) Unbuffered
  - ✓ (b) Buffered in user space
  - ✓ (c) Buffered in the kernel space
  - ✓ (d) Double buffering in the kernel



Device-Independent I/O Software

- Error reporting
  - ✓ Many errors are device-specific and must be handled by the appropriate driver, but the framework for error handling is device independent
  - ✓ Programming errors vs. actual I/O errors
  - ✓ Handling errors
    - ➢ Returning the system call with an error code
    - ➢ Retrying a certain number of times
    - ➢ Ignoring the error
    - ➢ Killing the calling process
    - ➢ Terminating the system

Device-Independent I/O Software

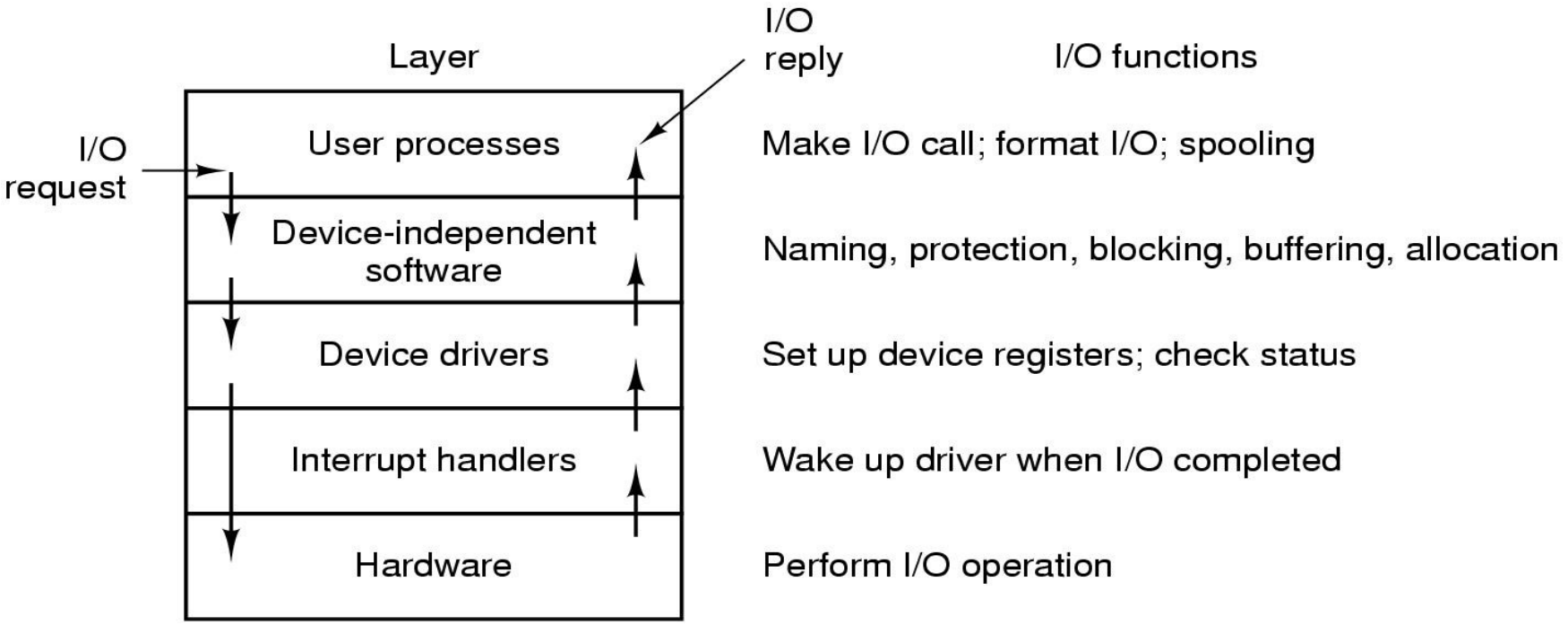
- Allocating and releasing dedicated devices
  - ✓ Some devices cannot be shared
  - (1) Require processes to perform open()'s on the special files for devices directly
    - ➢ The process retries if open() fails
  - (2) Have special mechanisms for requesting and releasing dedicated devices
    - ➢ An attempt to acquire a device that is not available blocks the caller

- Device-independent block size
  - ✓ Treat several sectors as a single logical block
  - ✓ The higher layers only deal with abstract devices that all use the same block size

User-Space I/O Software

- Provided as a library
  - ✓ Standard I/O library in C
    - ➢ fopen() vs. open()
- Spooling
  - ✓ A way of dealing with dedicated I/O devices in a multiprogramming system
  - ✓ Implemented by a daemon and a spooling directory
  - ✓ Printers, network file transfers, USENET news, mails, etc.

I/O Systems Layers



Thank You!
Q&A