Chapter 13: I/O Systems

Chapter 13: I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Streams
- Performance

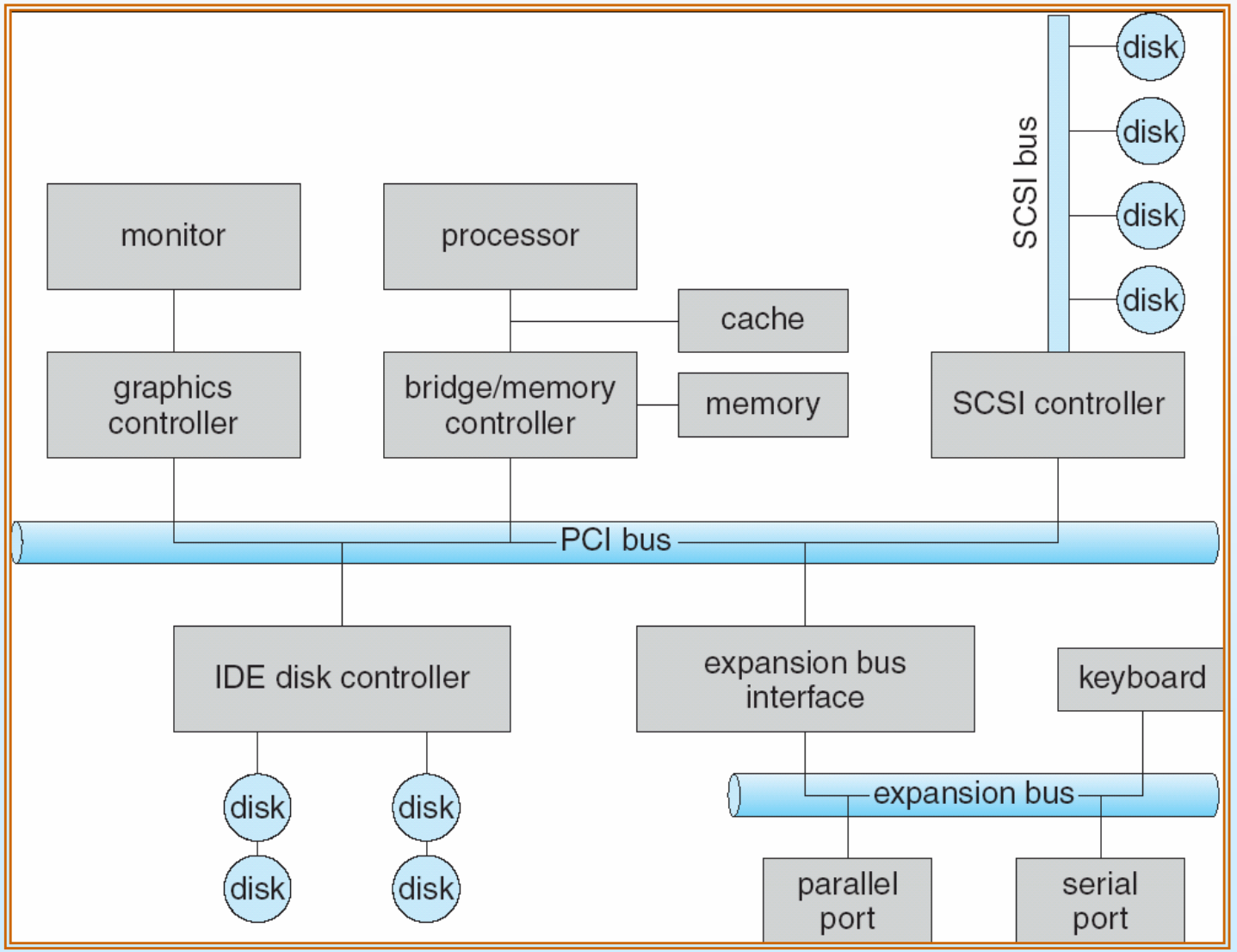
Objectives

- Explore the structure of an operating system's I/O subsystem
- Discuss the principles of I/O hardware and its complexity
- Provide details of the performance aspects of I/O hardware and software

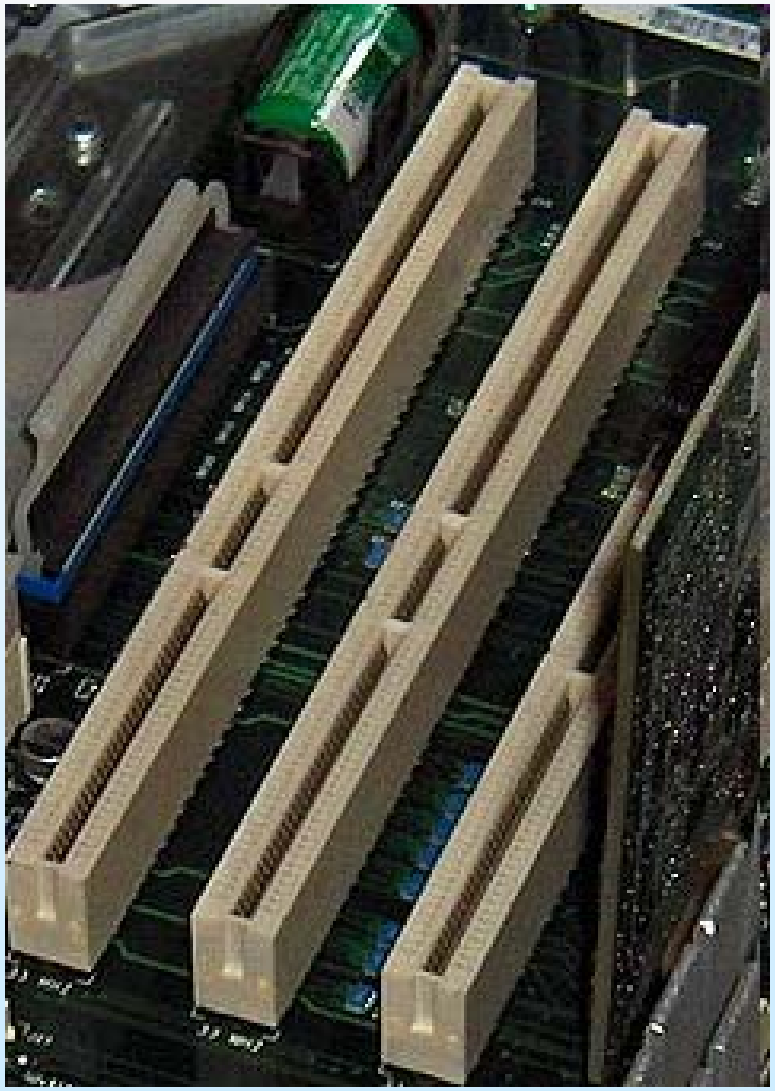
I/O Hardware

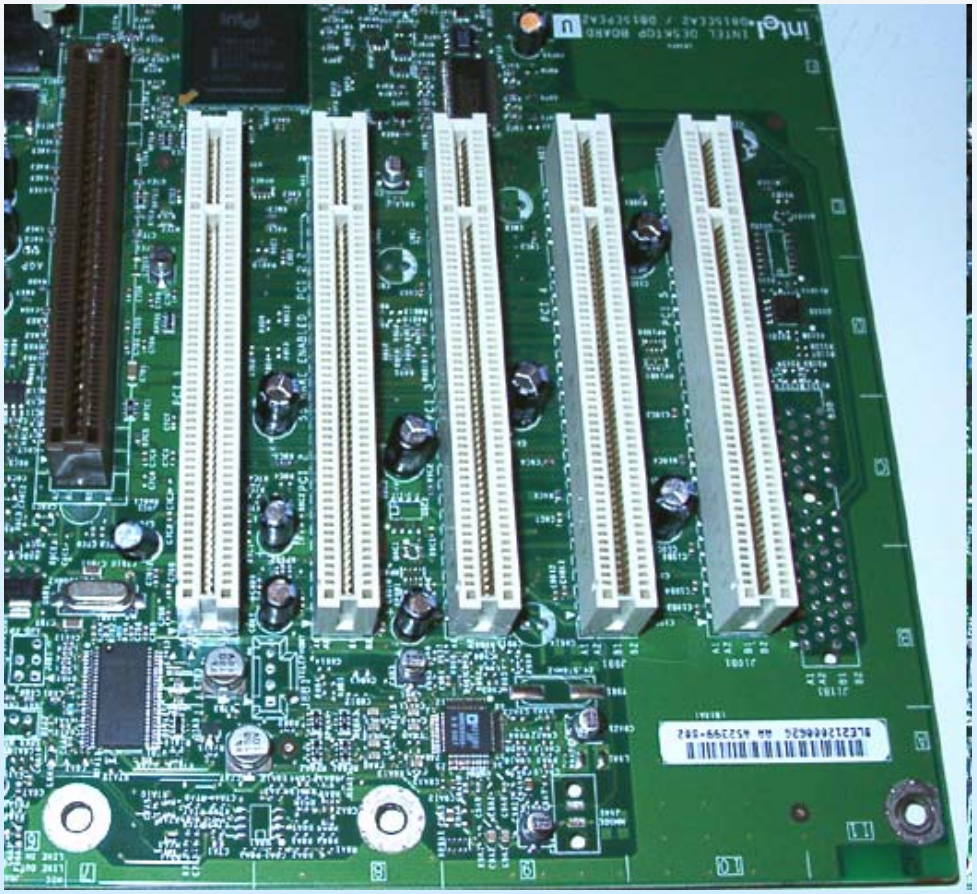
- Incredible variety of I/O devices
- Common concepts
  - **Port**
  - **Bus (ex. PCI)**
    - (daisy chain or shared direct access)
  - **Controller** (host adapter)
    - **device registers**
- I/O instructions control devices
- Devices have addresses, used by
  - Direct I/O instructions
  - Memory-mapped I/O

A Typical PC Bus Structure



32 and 64-bit PCI





Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

I/O Port

- An I/0 port typically consists of four registers, called the (1) status, (2) control, (3) data-in, and (4) data-out registers:

- The **data-in** register is read by the host to get input.
- The **data-out** register is written by the host to send output.
- The **status** register contains bits that can be read by the host. These bits indicate states, such as whether the current command has completed, whether a byte is available to be read from the data-in register, and whether a device error has occurred.
- The **control register** can be written by the host to start a command or to change the mode of a device.

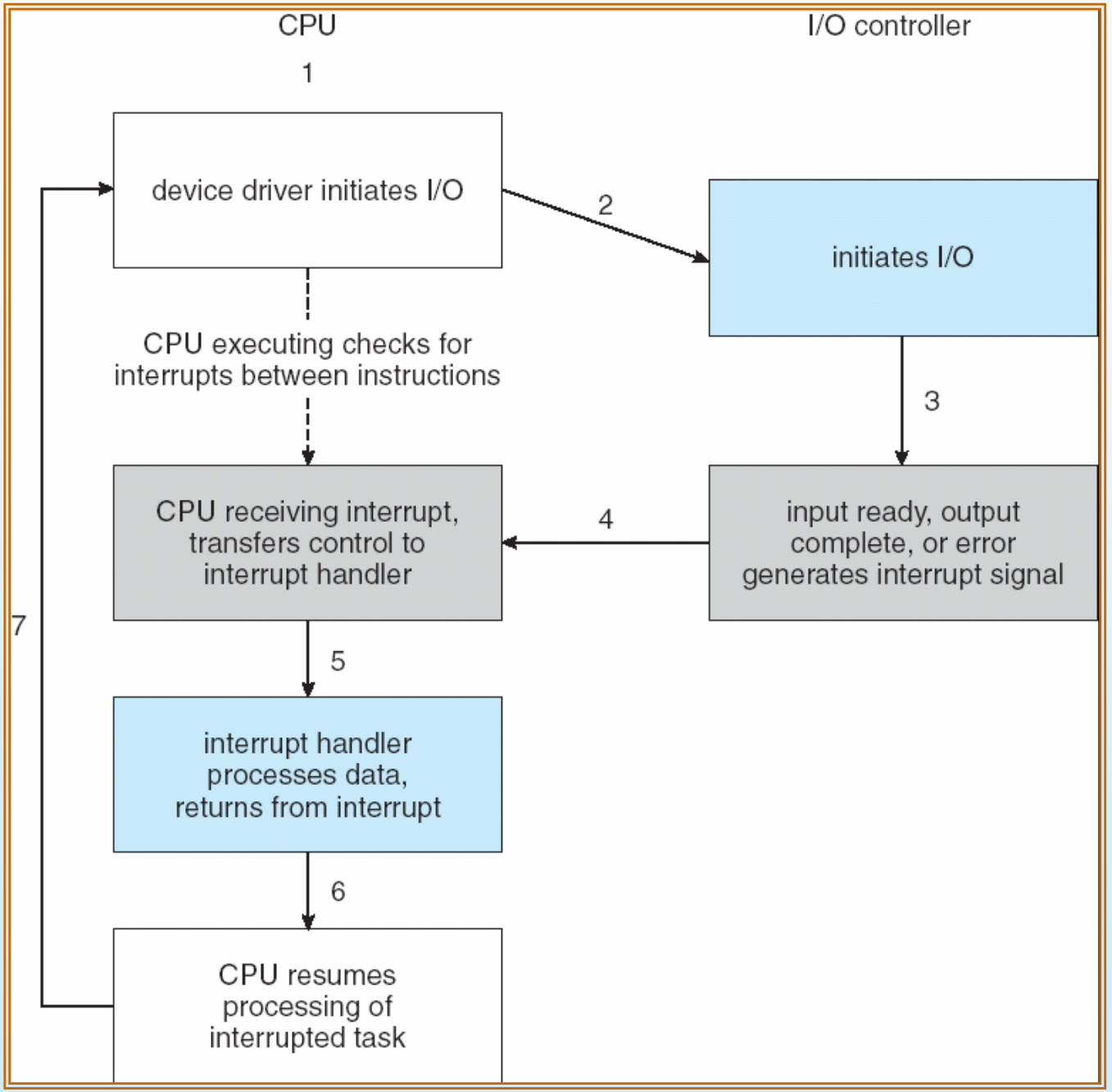
Busy Waiting

- Busy bit
- The host repeatedly reads the ***busy* bit** until that bit becomes clear.
- State of device:
  - command-ready
  - busy
  - error
- **Busy-wait** cycle to wait for I/O from device

Interrupts

- CPU has an **Interrupt-request line** triggered by I/O device
- **Interrupt handler** receives interrupts
- **IRL**
  - **Nonmaskable:** for unrecoverable memory errors
  - **Maskable:** before the execution of critical operations, the IRL can be turned off
- **Interrupt vector** to dispatch interrupts to correct handlers
  - Interrupts have priority
  - Some **nonmaskable**
- Interrupt mechanism also used for **exceptions**
  - Division by zero

Interrupt-Driven I/O Cycle



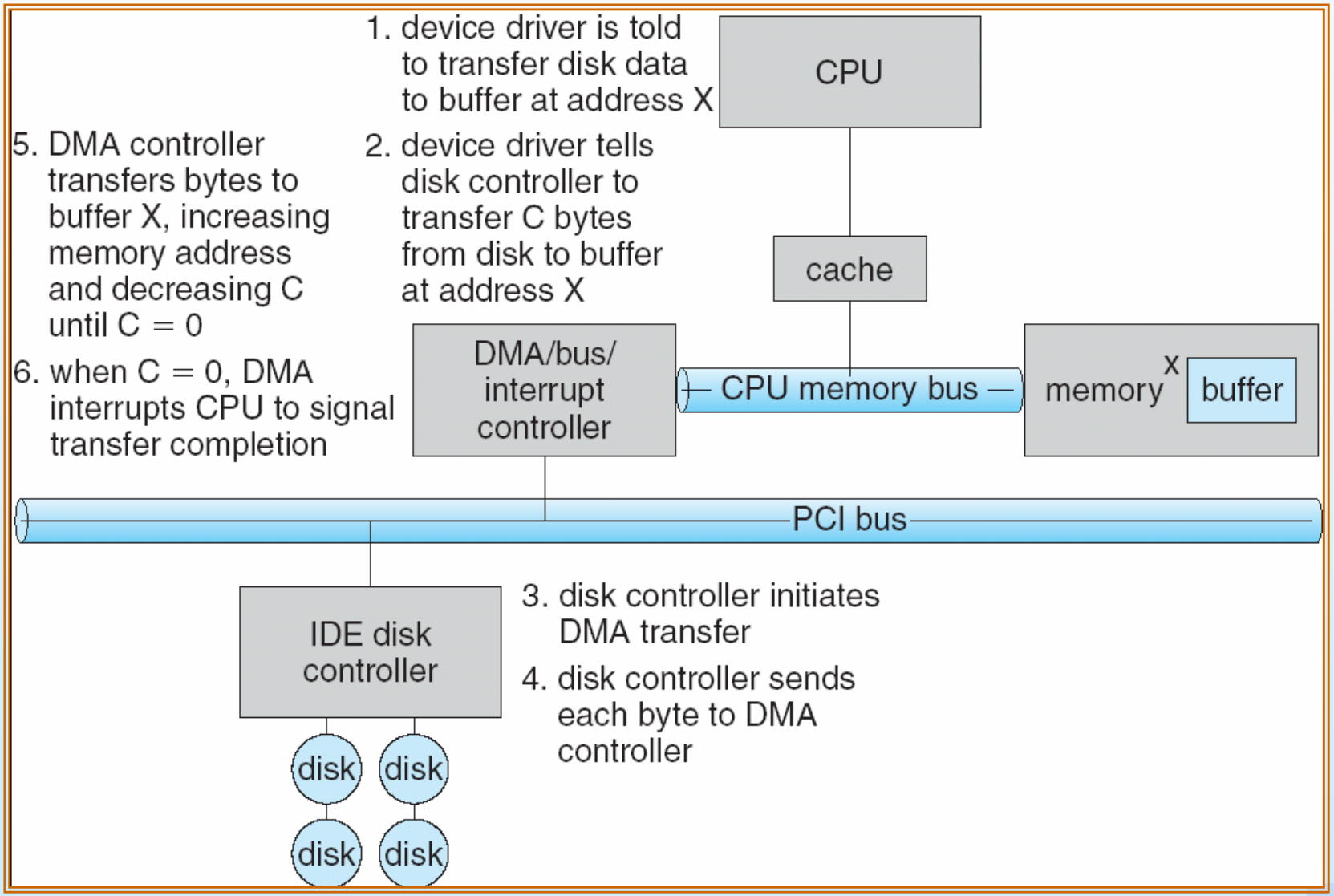
Intel Pentium Processor Event-Vector Table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Direct Memory Access

- **Programmed I/O**
  - General purpose CPU deals with I/O
- **DMA** used to avoid **programmed I/O** for large data movement
- Requires **DMA** controller: a special processor
- Bypasses CPU to transfer data directly between I/O device and memory

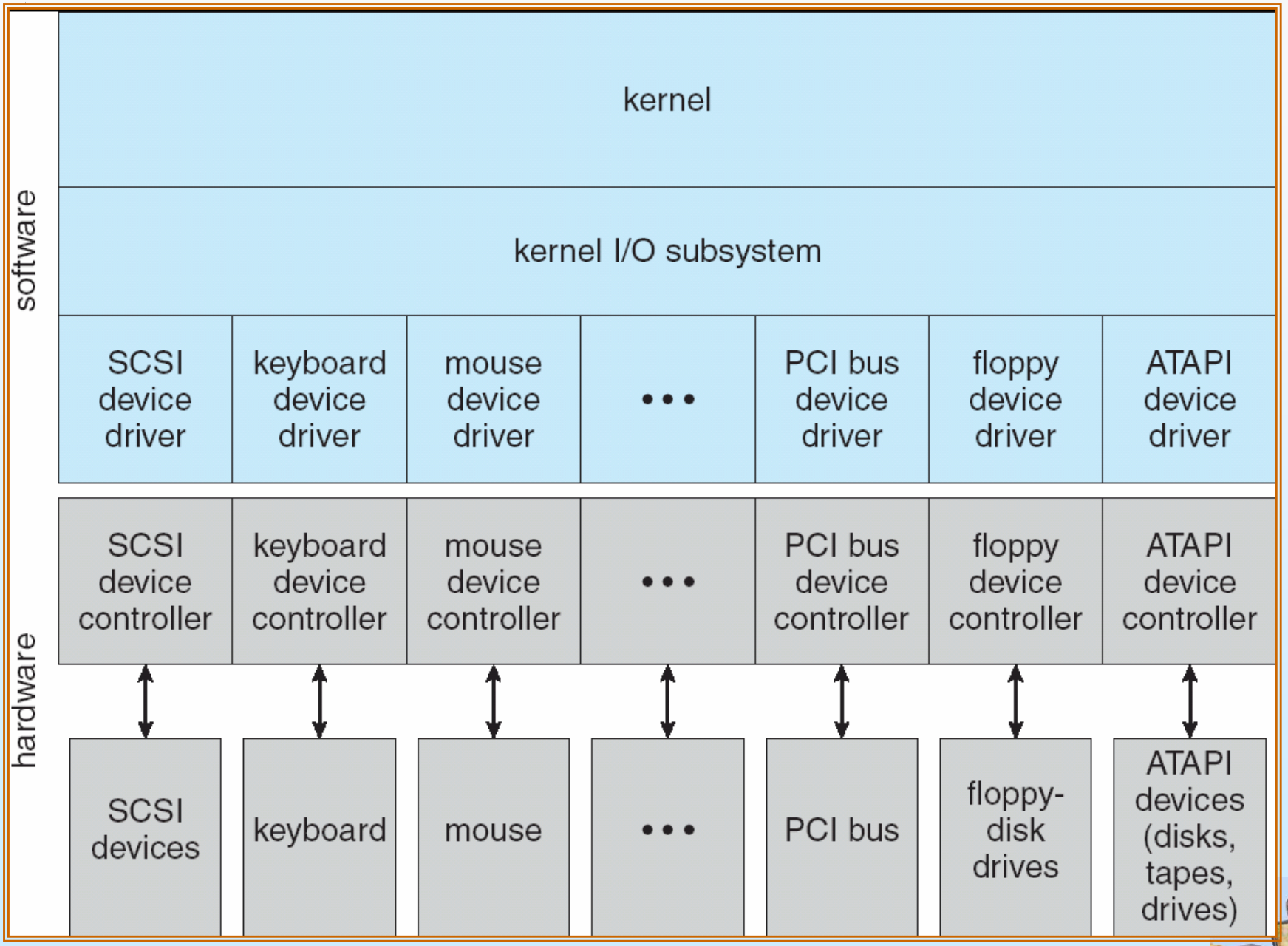
Six Step Process to Perform DMA Transfer



Application I/O Interface

- I/O system calls
  - encapsulate device behaviors in generic classes
- Device-driver layer
  - hides differences among I/O controllers from kernel

A Kernel I/O Structure



Devices

- Devices vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential or random-access**
  - **Synchronous or asynchronous.**
  - **Sharable or dedicated**
  - **Speed of operation**
  - **read-write, read only,** or **write only**

Characteristics of I/O Devices

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Block and Character Devices

- **Block devices** include disk drives
  - Commands include read, write, seek
  - **Raw** I/O or file-system access
- **Character devices** include keyboards, mice, serial ports
  - Commands include `get, put`
  - Libraries layered on top allow line editing

Network Devices

- Varying enough from block and character to have own interface
- Most OSs provide Network Interfaces
- Unix and Windows NT/9*x*/2000 include **socket interface**
  - Separates network protocol from network operation
  - Includes `select` functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

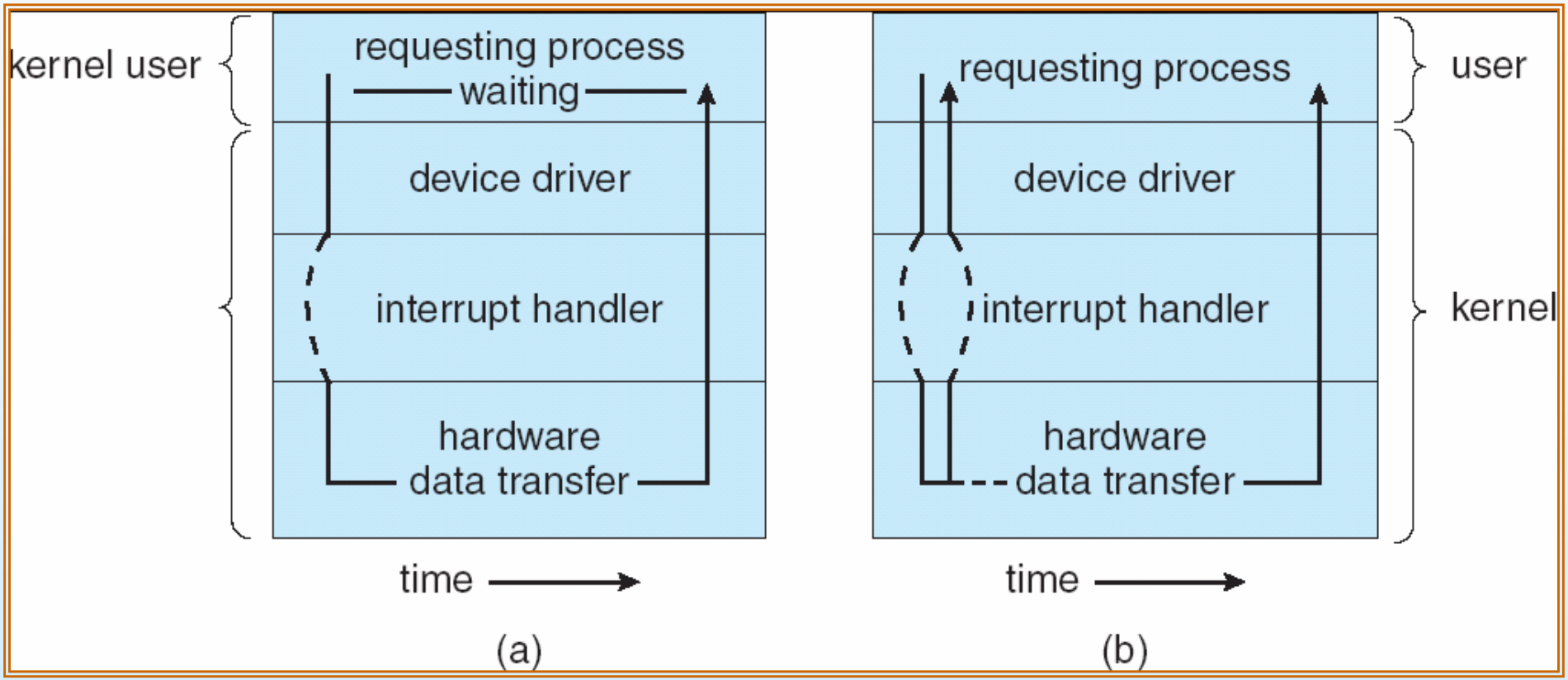
Clocks and Timers

- Provide current time, elapsed time, timer to trigger operation
- **Programmable interval timer** used for timings, periodic interrupts
- Timers used by
  - Scheduler
  - Disk I/O subsystem
  - Network subsystem
  - User processes can also use them

Blocking and Nonblocking I/O

- **Blocking** - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
- **Nonblocking** - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
- **Asynchronous** - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed

Two I/O Methods



Synchronous

Asynchronous

Kernel I/O Subsystem

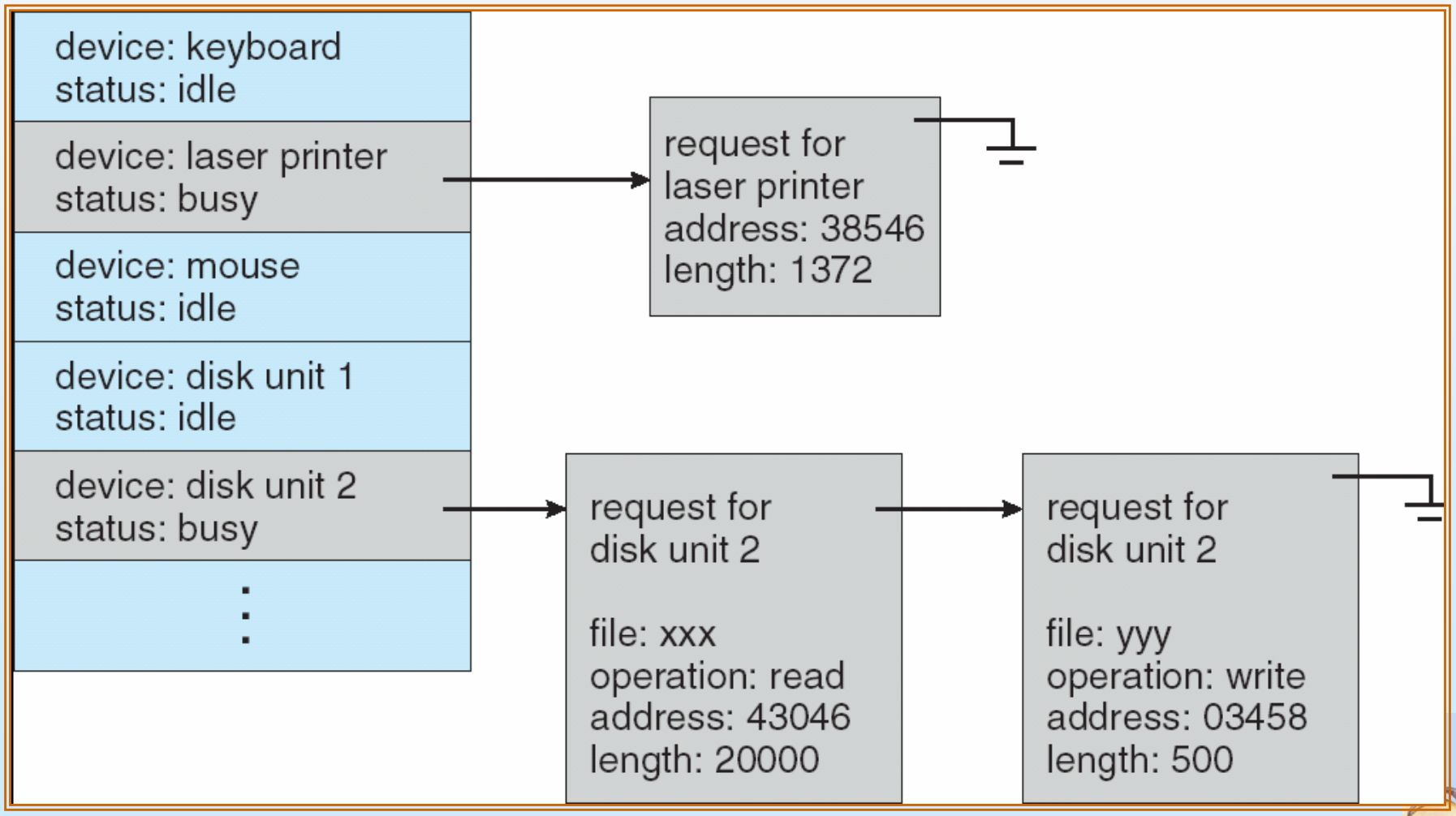
- I/O Scheduling
  - Some I/O request ordering
  - Some OSs try fairness

I/O Scheduling

- Scheduling can:
  - improve overall system performance
  - share device access fairly among processes
  - reduce the **average waiting time for I/O to complete**.
- Suppose that a disk arm is near the **beginning** of a disk and that three applications issue blocking read calls to that disk.
- Application 1 requests a block near the end of the disk, application 2 requests one near the beginning, and application 3 requests one in the middle of the disk.
- The operating system can reduce the distance that the disk arm travels by serving the applications in the order 2, 3, 1.
- Rearranging the order of service in this way is the essence of **I/0 scheduling.**

Device-status Table

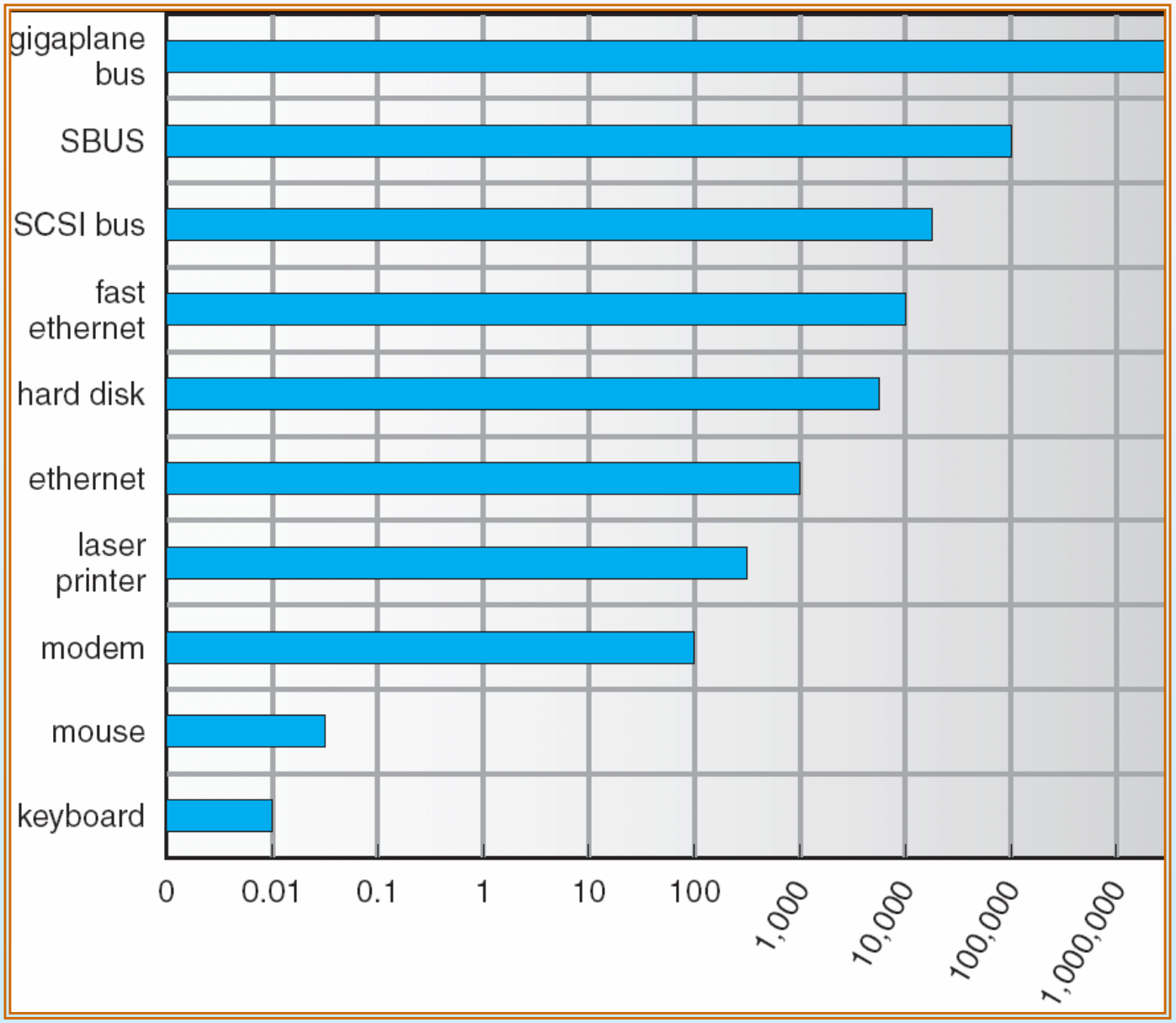
When a kernel supports asynchronous I/O, it must be able to **keep track** of many I/O requests at the same time. For this purpose, the operating system might attach the wait queue to a **device-status table**.



Buffering

- Store data in memory while transferring between devices
  - To cope with device **speed mismatch**
  - To cope with device transfer **size mismatch**
  - To maintain “copy semantics”
    - With copy semantics, the version of the data written to disk is guaranteed to be the version at the time of the application system call, independent of any subsequent changes in the application's buffer.

Sun Enterprise 6000 Device-Transfer Rates



Caching

- A cache is a region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
- For instance, the instructions of the currently running process are stored on disk, cached in physical memory, and copied again in the CPU's secondary and primary caches.
- The difference between a buffer and a cache is that a buffer may hold **the only existing copy** of a data item, whereas a cache, by definition, just holds **a copy on faster storage** of an item that resides elsewhere.

Spooling

- A spool is a buffer that holds output for a device, such as a printer that cannot accept interleaved data streams.
- Although a printer can serve only one job at a time, several applications may wish to print their output concurrently, without having their output mixed together.
- The operating system solves this problem by **intercepting** all output to the printer.
- Each application's output is spooled to a **separate disk file**.
  - When an application finishes printing, the spooling system **queues** the corresponding spool file for output to the printer.
- The spooling system copies the queued spool files to the printer one at a time.
- In some operating systems, spooling is managed by a **system daemon process**.

Device reservation

- **Device reservation** - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

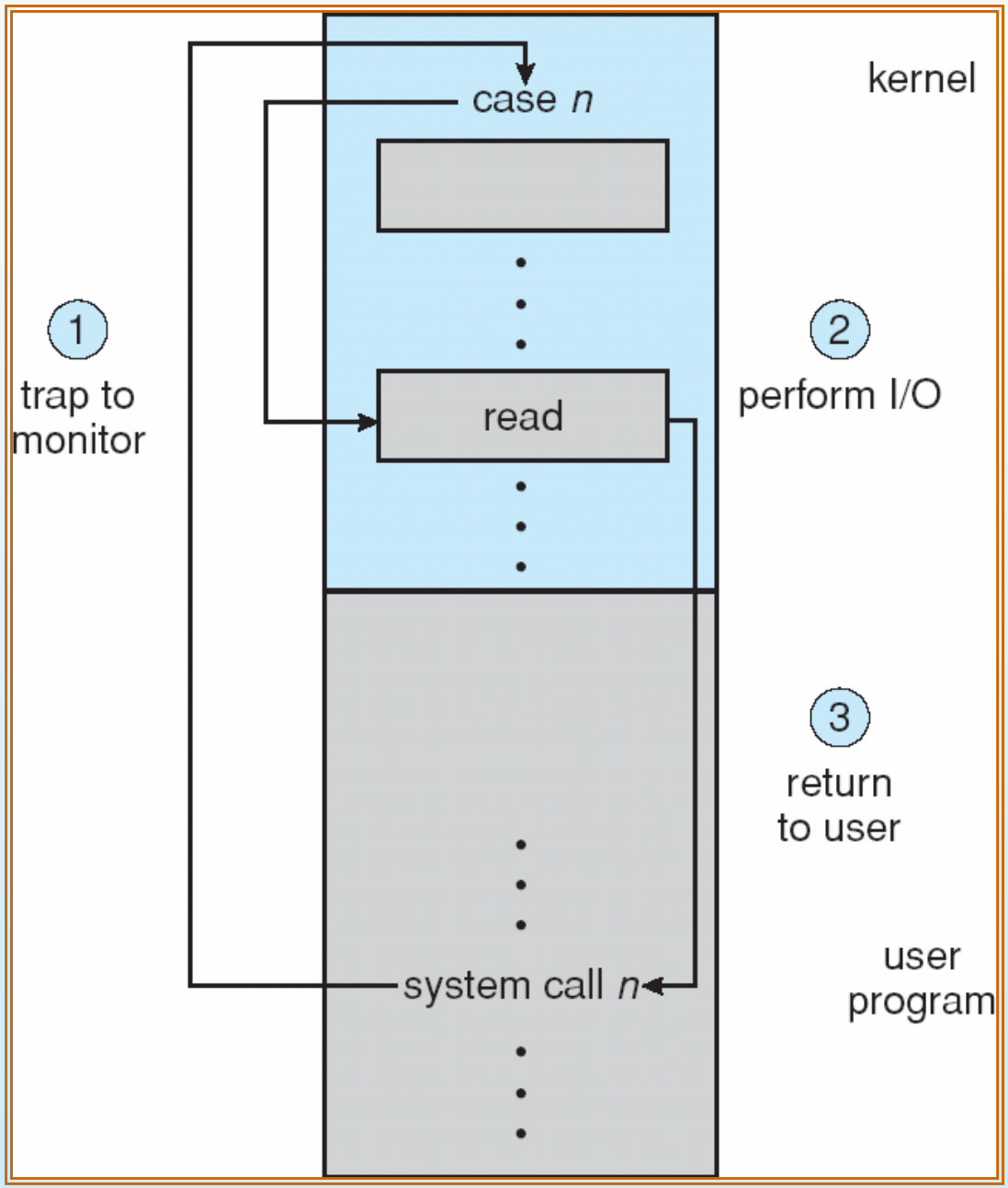
Error Handling

- OS can recover from disk read, device unavailable, transient write failures
- Most return an error number or code when I/O request fails
- System error logs hold problem reports

I/O Protection

- User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions
  - All I/O instructions defined to be **privileged**
  - I/O must be performed via **system calls**
    - **Thus the OS can check them**
- Memory-mapped and I/O port memory locations must be protected too

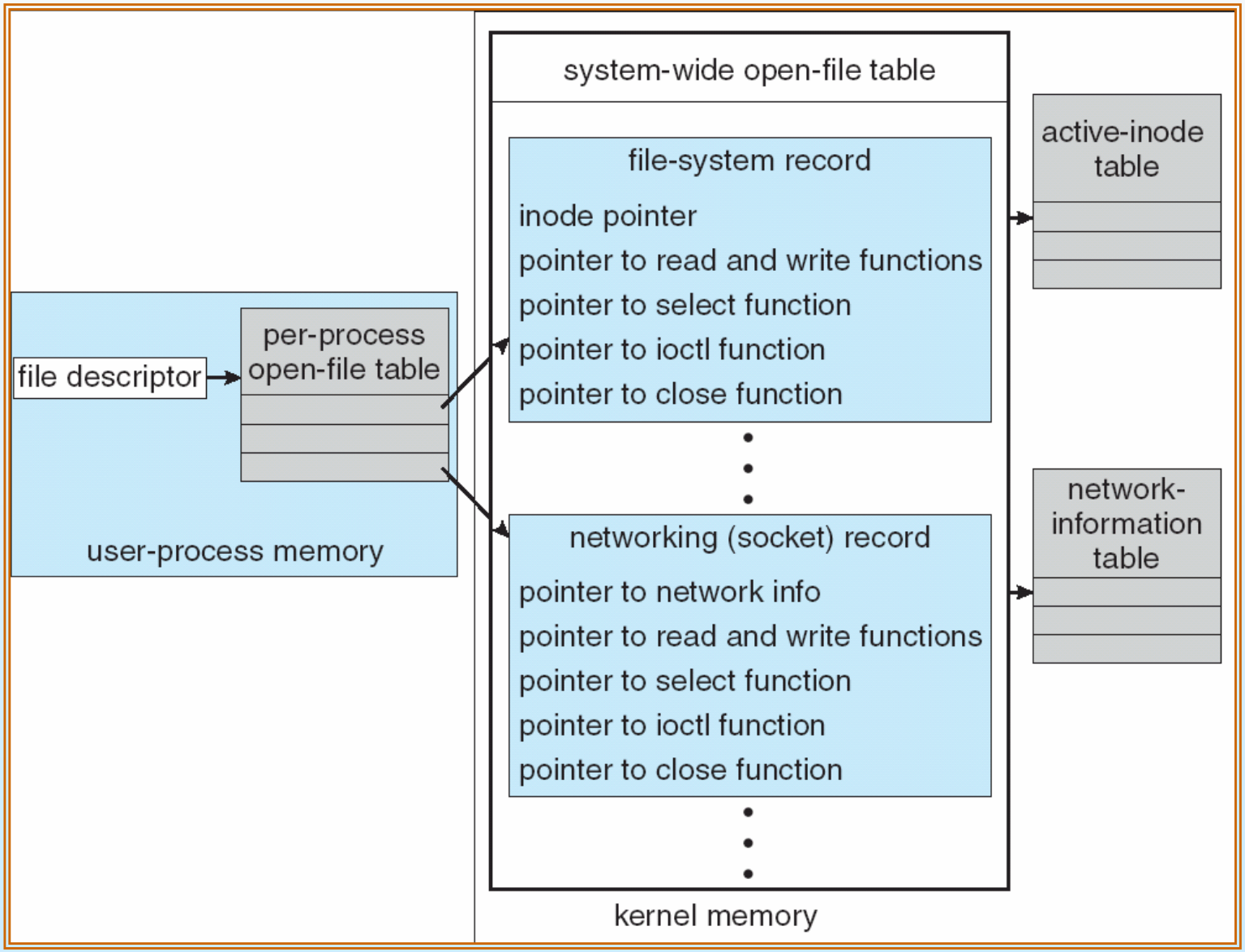
Use of a System Call to Perform I/O



Kernel Data Structures

- Kernel keeps state info for I/O components, including:
  - open file tables
  - network connections
  - character device state
- Many, many complex data structures to track buffers, memory allocation, “dirty” blocks
- Some use object-oriented methods and message passing to implement I/O

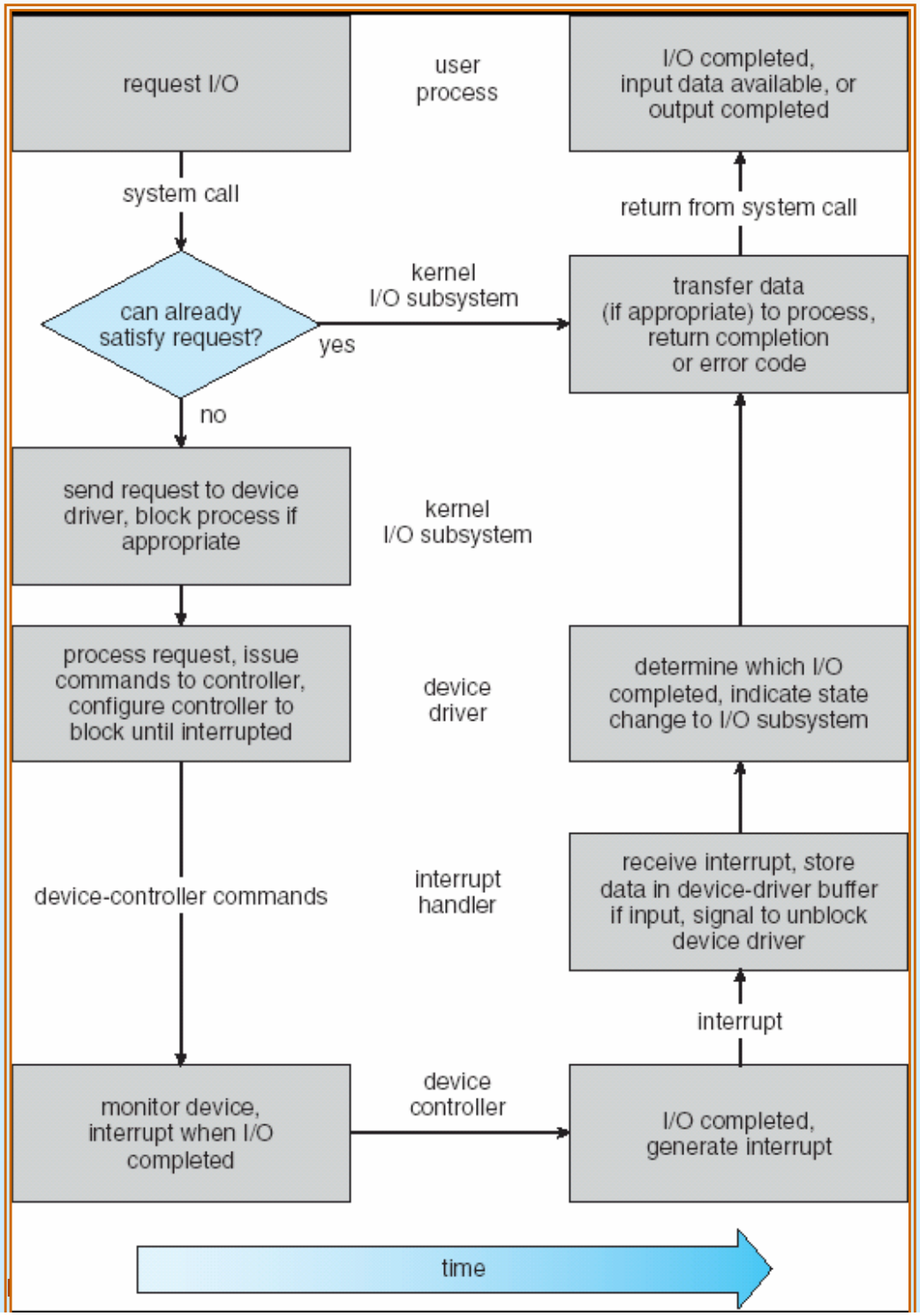
UNIX I/O Kernel Structure



I/O Requests to Hardware Operations

- Consider reading a file from disk for a process:
  - Determine device holding file
  - Translate name to device representation
  - Physically read data from disk into buffer
  - Make data available to requesting process
  - Return control to process

Life Cycle of An I/O Request



STREAMS

- **STREAM** – a full-duplex communication channel between a user-level process and a device in Unix System V and beyond

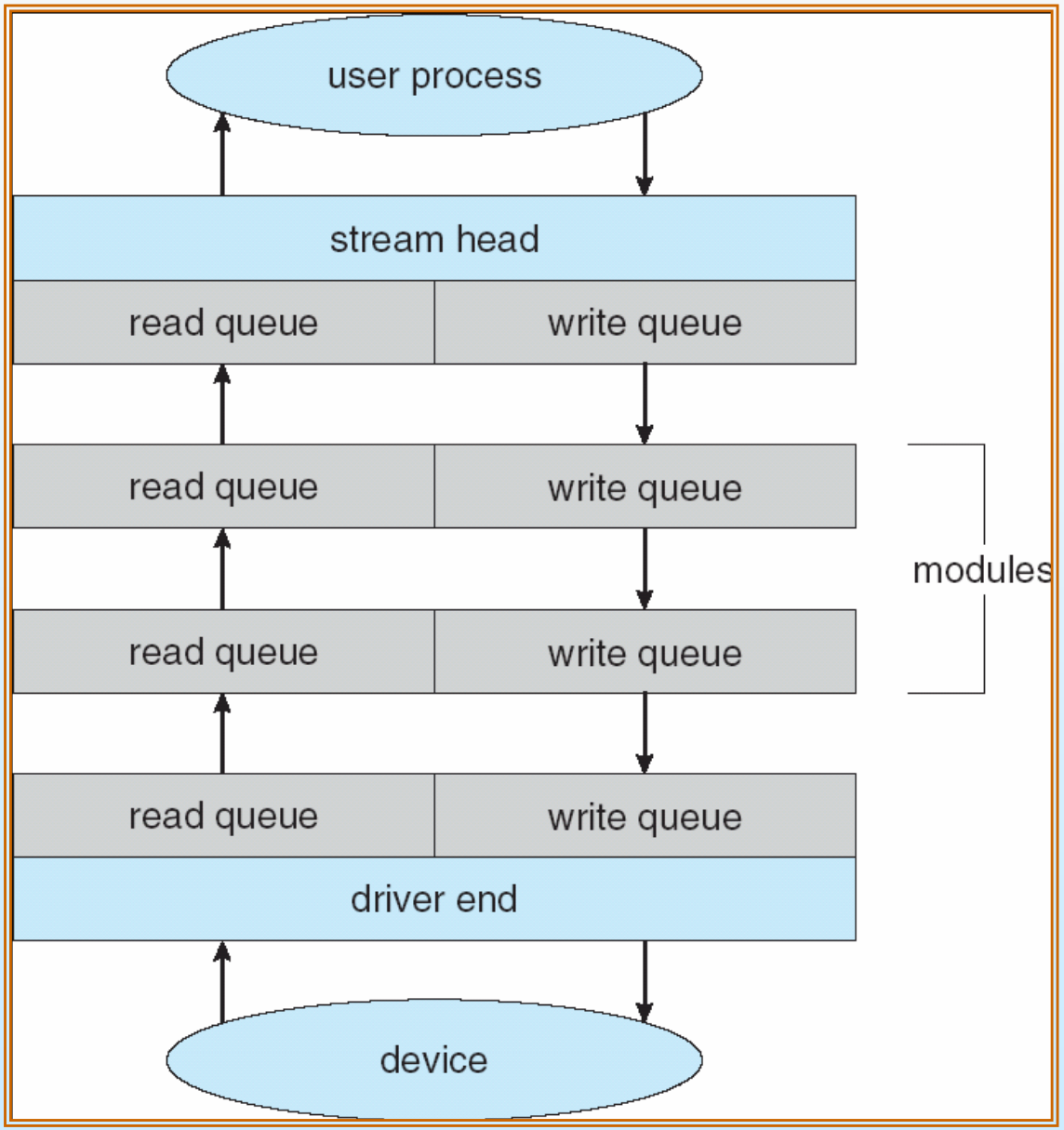
- A STREAM consists of:

  - **STREAM head** interfaces with the user process
  - **driver end** interfaces with the device
  - **zero or more** STREAM modules between them.

- Each module contains a **read queue** and a **write queue**

- Message passing is used to communicate between queues

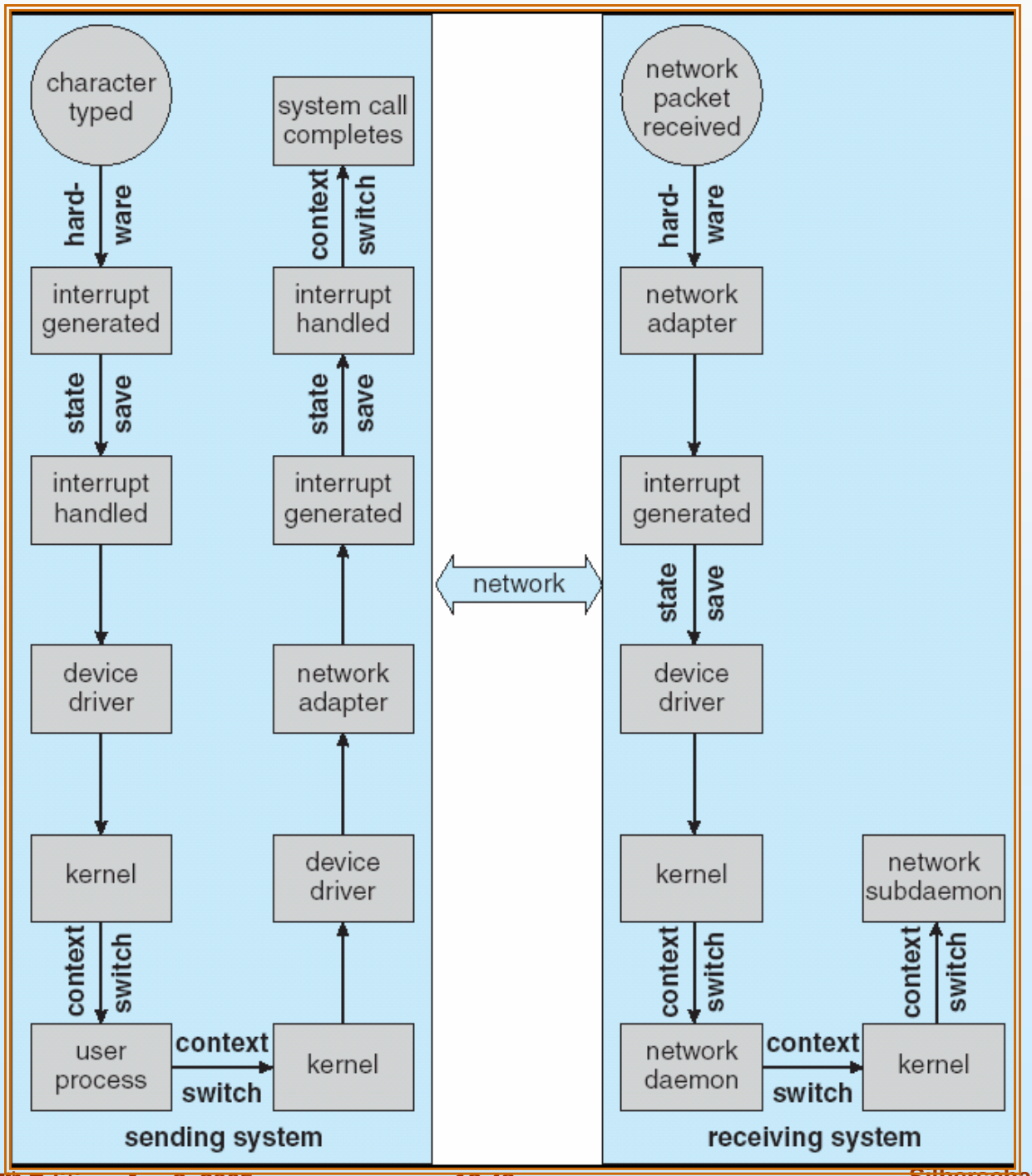
The STREAMS Structure



Performance

- I/O a major factor in system performance:
  - Demands CPU to execute device driver, kernel I/O code
  - Context switches due to interrupts
  - Data copying
  - Network traffic especially stressful

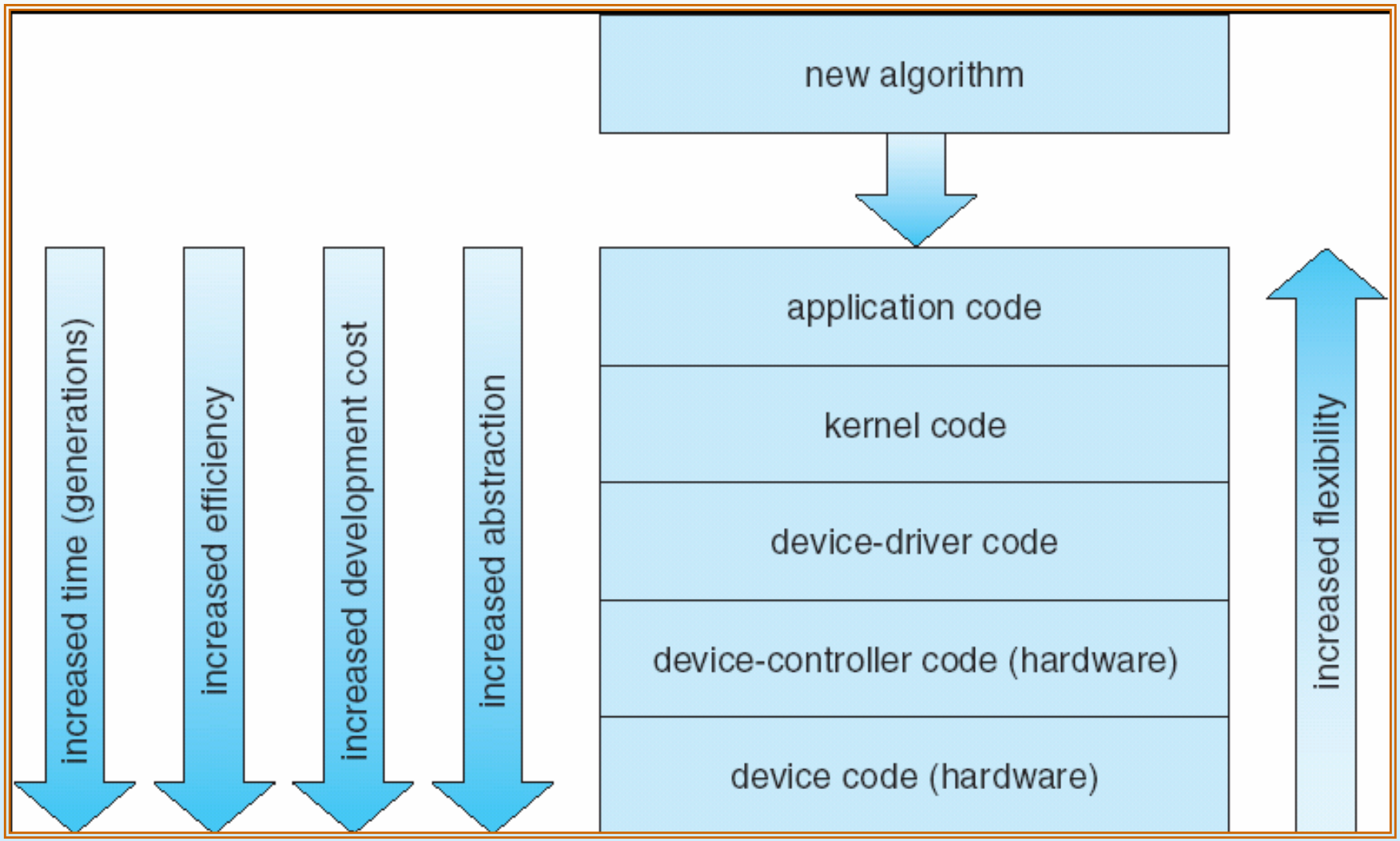
Intercomputer Communications



Improving Performance

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using large transfers, smart controllers
- Use DMA
- Balance CPU, memory, bus, and I/O performance for highest throughput

Device-Functionality Progression



End of Chapter 13

Exam Rules

- I appreciate the efforts of some of you to have studied hard, but numbers are what matter in the final decision.
  - Everybody should be given the possibility to correct his mistakes!!!
  - You will have one Bonus of 20 pts in the final exam
    - Positive answers added to midterm grade ☺
    - Negative answers deducted from final exam ☹
    - No replies in Bonus Part will not change anything.

GOOD LUCK!