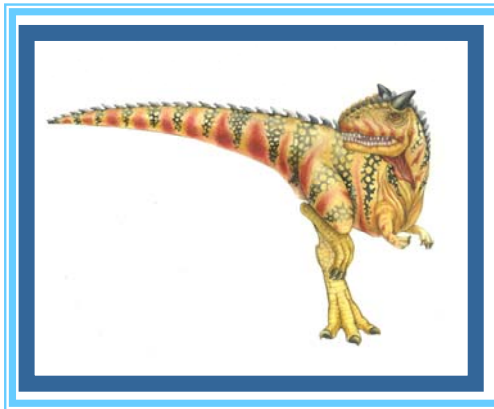# Chapter 13: I/O Systems

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

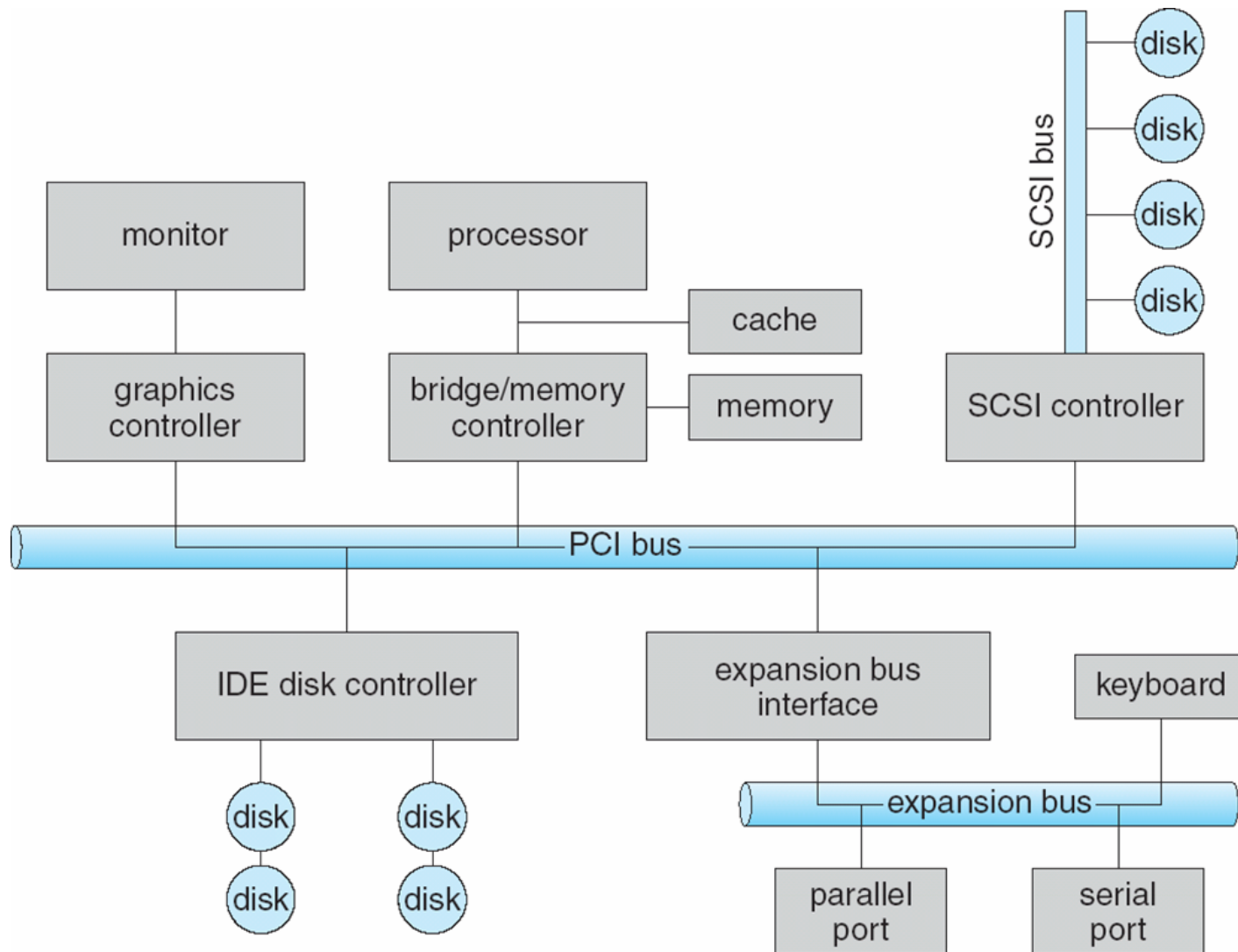- Provide details of the performance aspects of I/O hardware and software

I/O Hardware

- Incredible variety of I/O devices

- Common concepts
    - Port
    - Bus (daisy chain or shared direct access)
    - Controller (host adapter)
- I/O instructions control devices
- Devices have addresses, used by
    - Direct I/O instructions
    - Memory-mapped I/O

A Typical PC Bus Structure

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)

Polling

- Determines state of device
  - command-ready
  - busy
  - Error
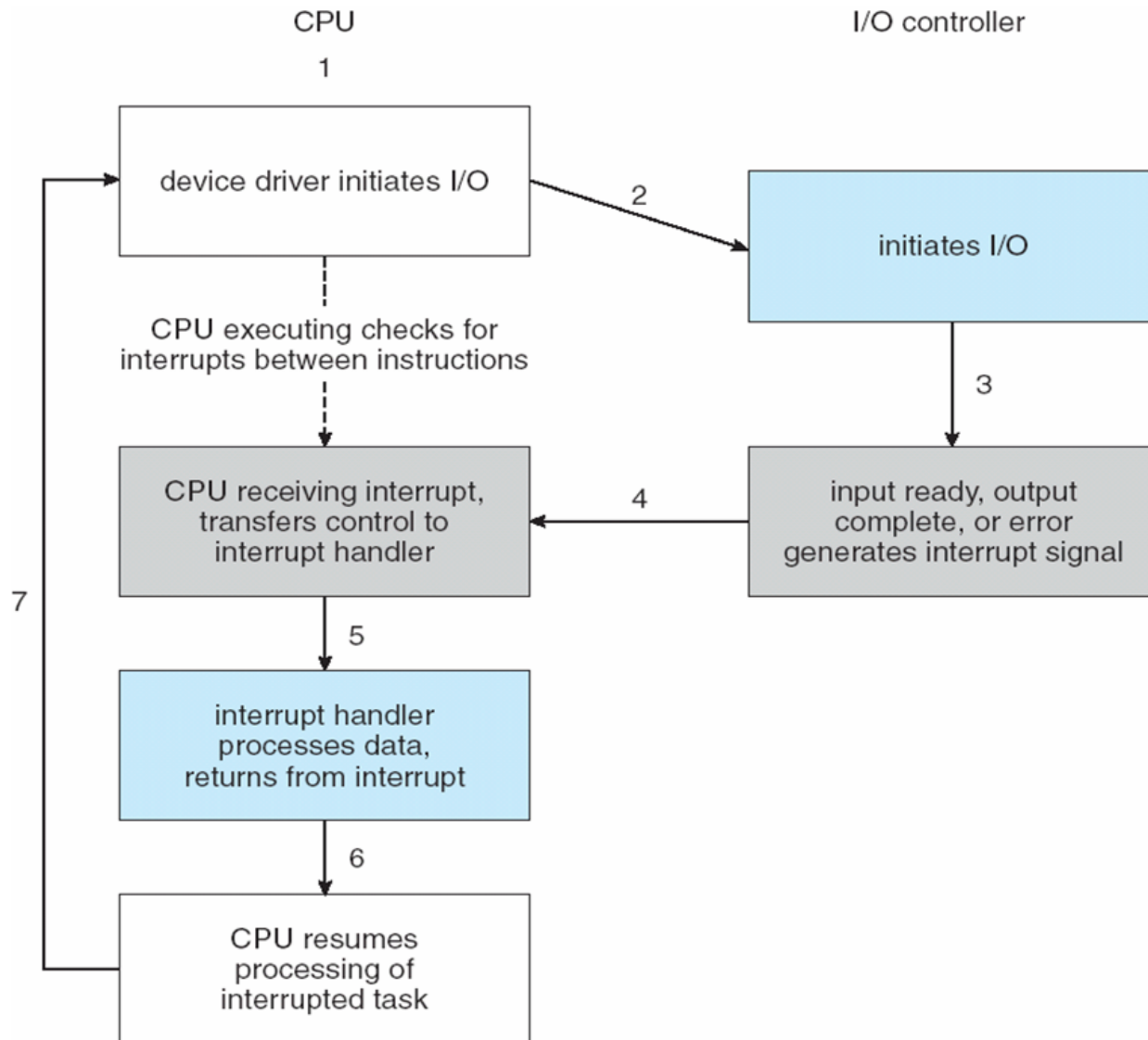- Busy-wait cycle to wait for I/O from device

Interrupts

- CPU Interrupt-request line triggered by I/O device

- Interrupt handler receives interrupts

- Maskable to ignore or delay some interrupts

- Interrupt vector to dispatch interrupt to correct handler
  - Based on priority
  - Some nonmaskable

- Interrupt mechanism also used for exceptions

Interrupt-Driven I/O Cycle

CPU

I/O controller

1 device driver initiates I/O

2

initiates I/O

CPU executing checks for interrupts between instructions

3

4 CPU receiving interrupt, transfers control to interrupt handler

input ready, output complete, or error generates interrupt signal

5 interrupt handler processes data, returns from interrupt

6

7 CPU resumes processing of interrupted task

Intel Pentium Processor Event-Vector Table

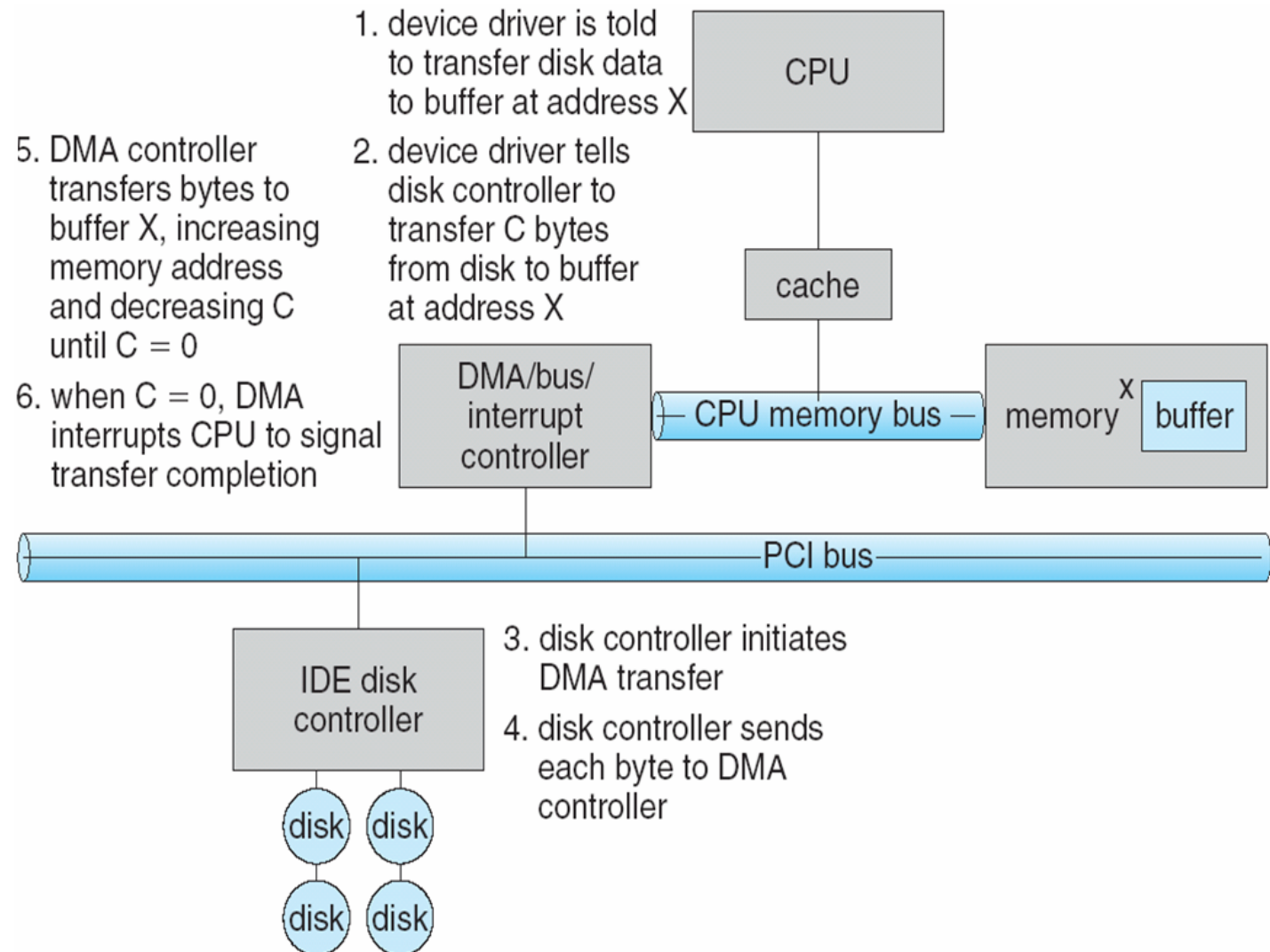| vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
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

- Used to avoid programmed I/O for large data movement

- Requires DMA controller

- Bypasses CPU to transfer data directly between I/O device and memory

1. device driver is told to transfer disk data to buffer at address X

2. device driver tells disk controller to transfer C bytes from disk to buffer at address X

5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0

6. when C = 0, DMA interrupts CPU to signal transfer completion

CPU

cache

DMA/bus/ interrupt controller

CPU memory bus

memory ^{X} buffer

PCI bus

IDE disk controller

3. disk controller initiates DMA transfer

4. disk controller sends each byte to DMA controller

disk disk

disk disk

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

- Devices vary in many dimensions

  - Character-stream or block

  - Sequential or random-access

  - Sharable or dedicated

  - Speed of operation

  - read-write, read only, or write only

A Kernel I/O Structure
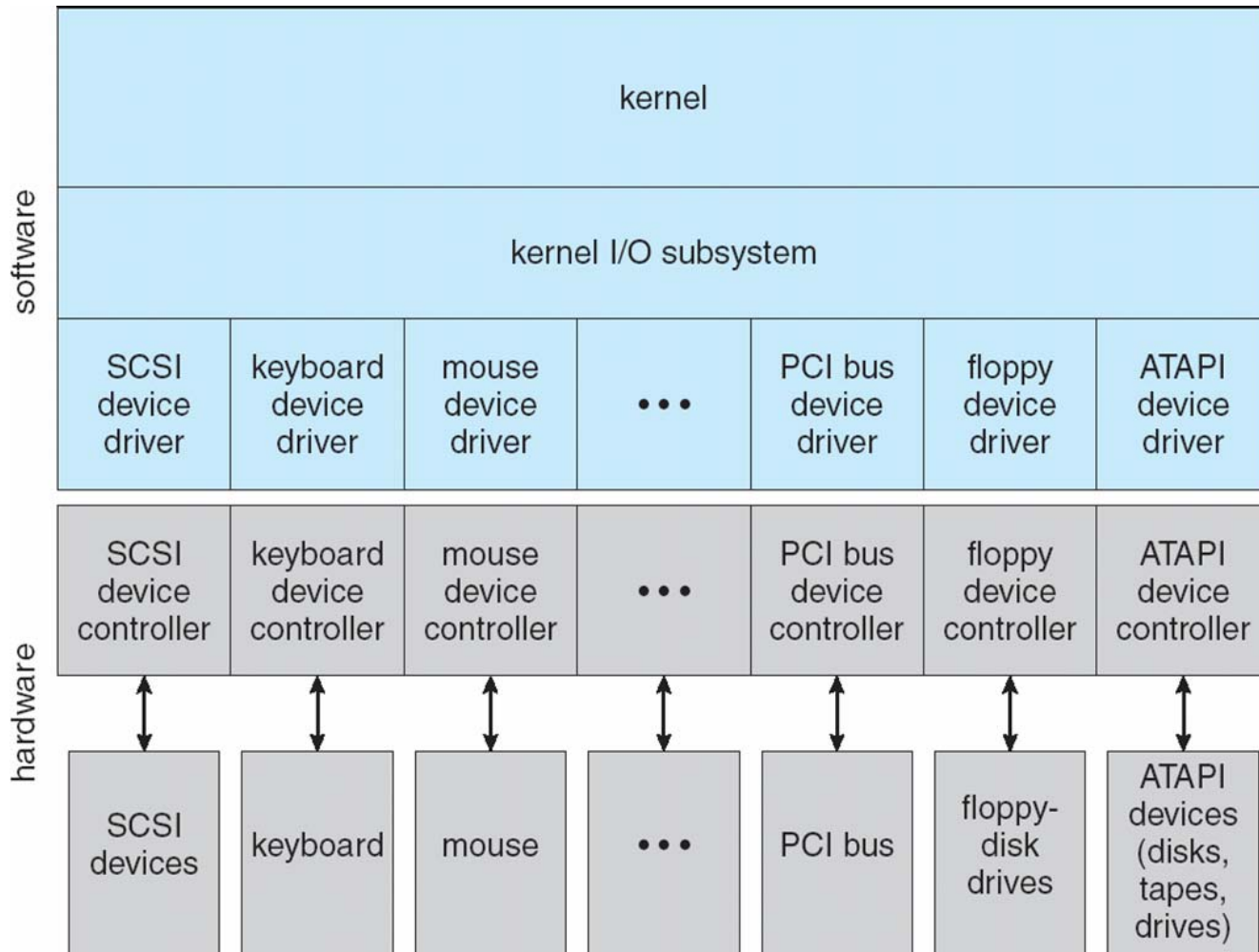
Characteristics of I/O Devices

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
  - Memory-mapped file access possible

- 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

- 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

- Programmable interval timer used for timings, periodic interrupts

- `ioctl()` (on UNIX) covers odd aspects of I/O such as clocks and timers

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

(a) Synchronous

(b) Asynchronous
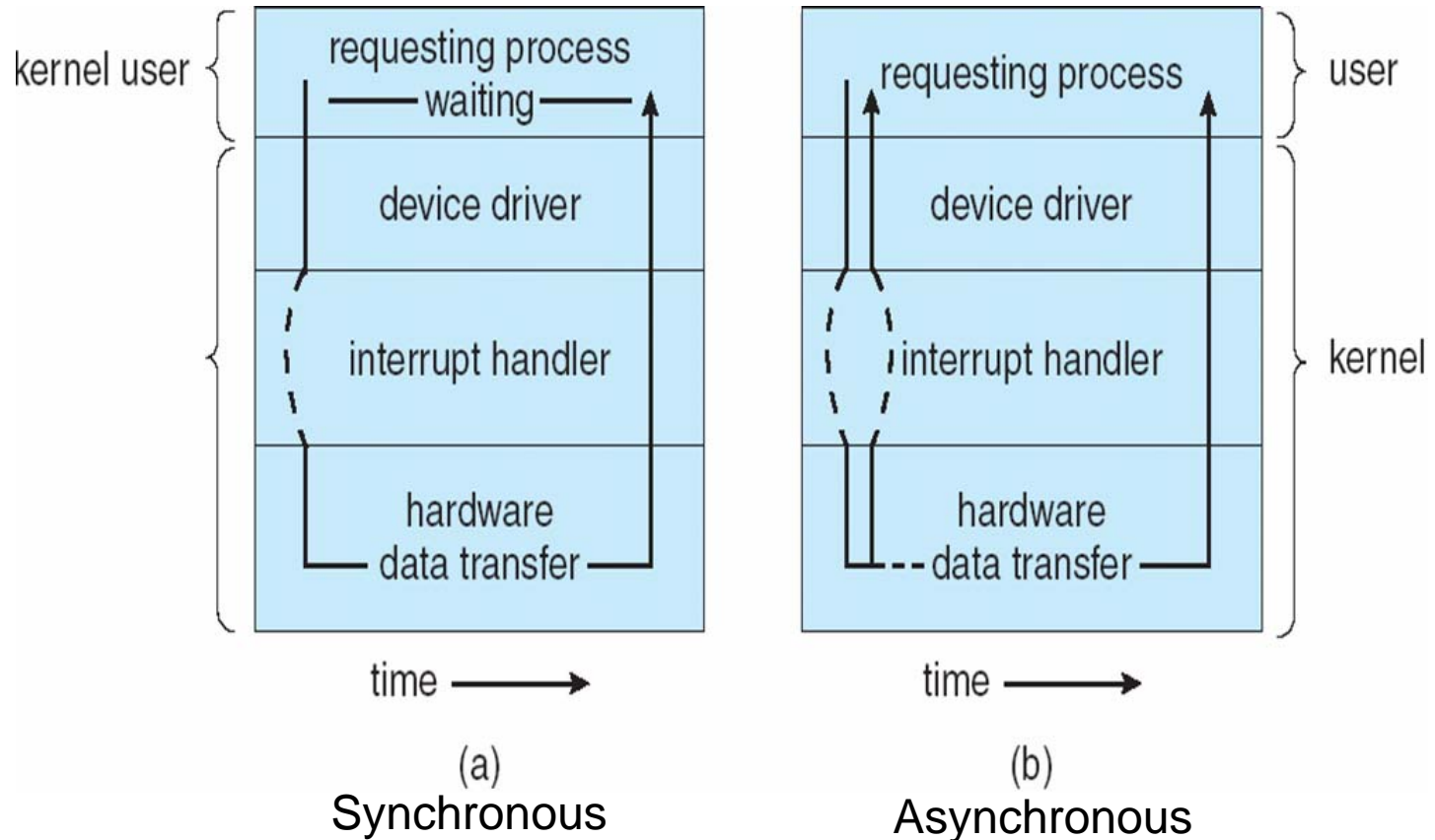
Kernel I/O Subsystem

- Scheduling
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness

- Buffering - store data in memory while transferring between devices
  - To cope with device speed mismatch
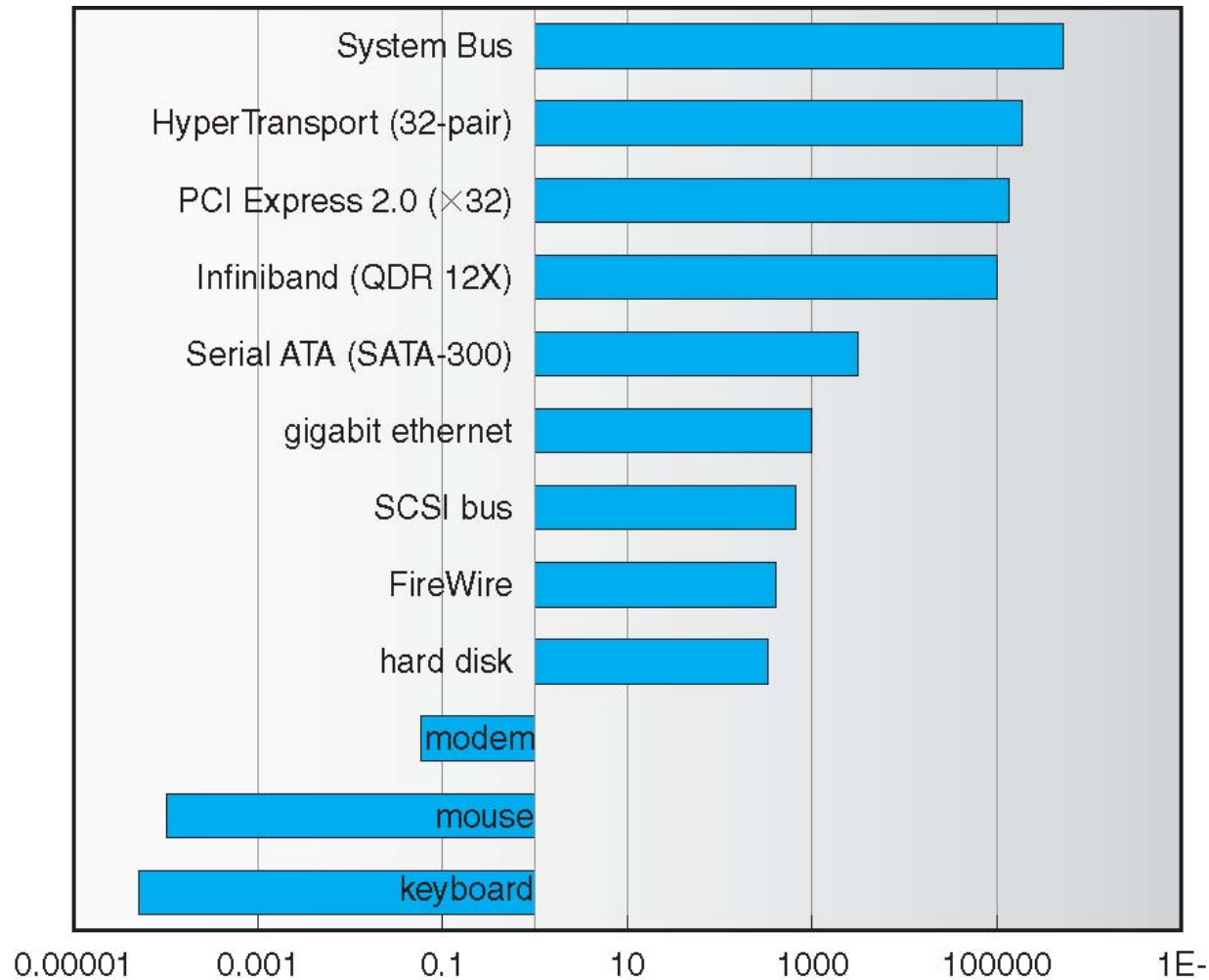 - To cope with device transfer size mismatch
  - To maintain "copy semantics"

Device-status Table
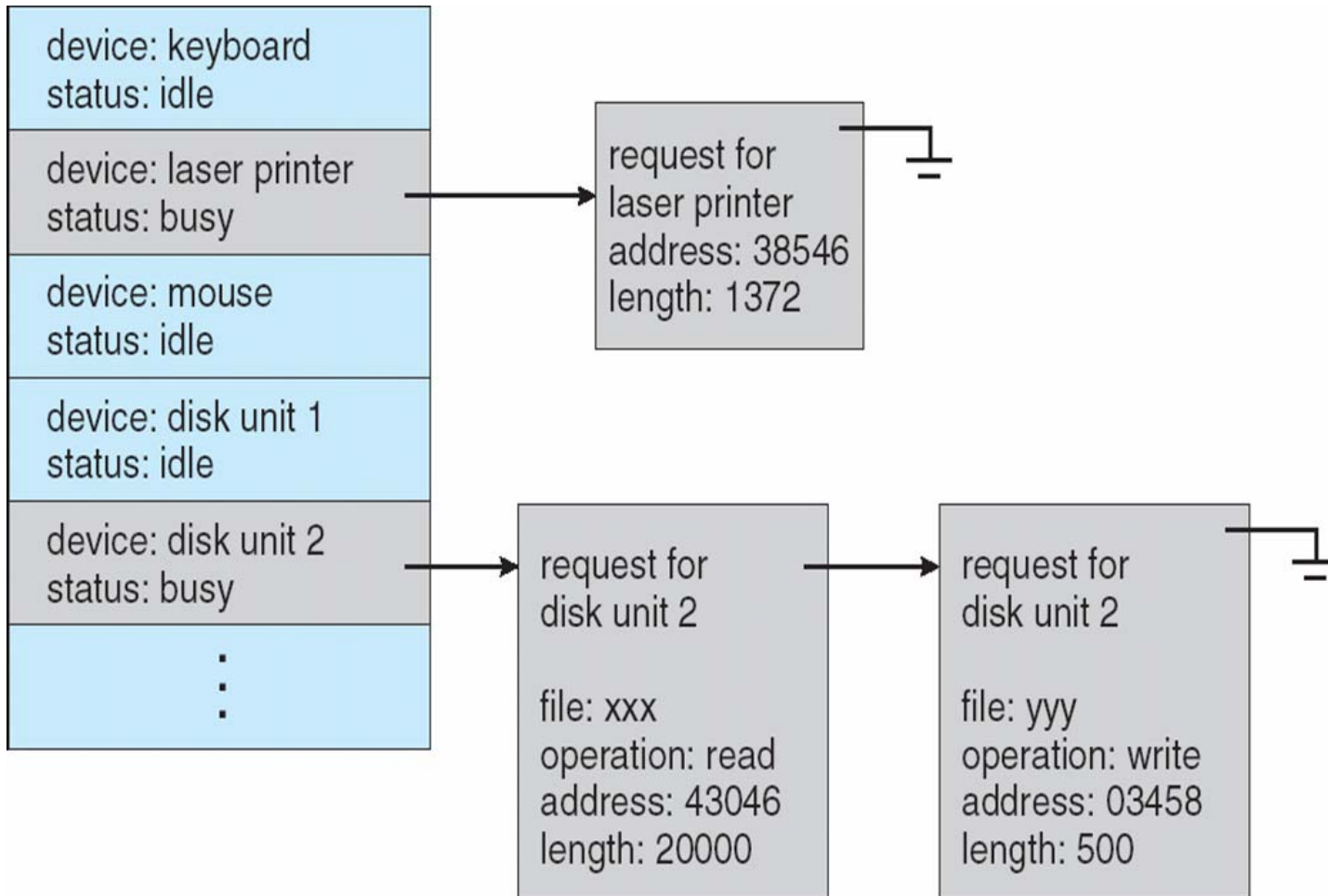
Sun Enterprise 6000 Device-Transfer Rates

Kernel I/O Subsystem

- **Caching** - fast memory holding copy of data
    - Always just a copy
    - Key to performance

- **Spooling** - hold output for a device
    - If device can serve only one request at a time
    - i.e., Printing

- **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

    - Memory-mapped and I/O port memory locations must be protected too

Use of a System Call to Perform I/O

kernel

① trap to monitor

case n

read

② perform I/O

③ return to user

system call n

user program
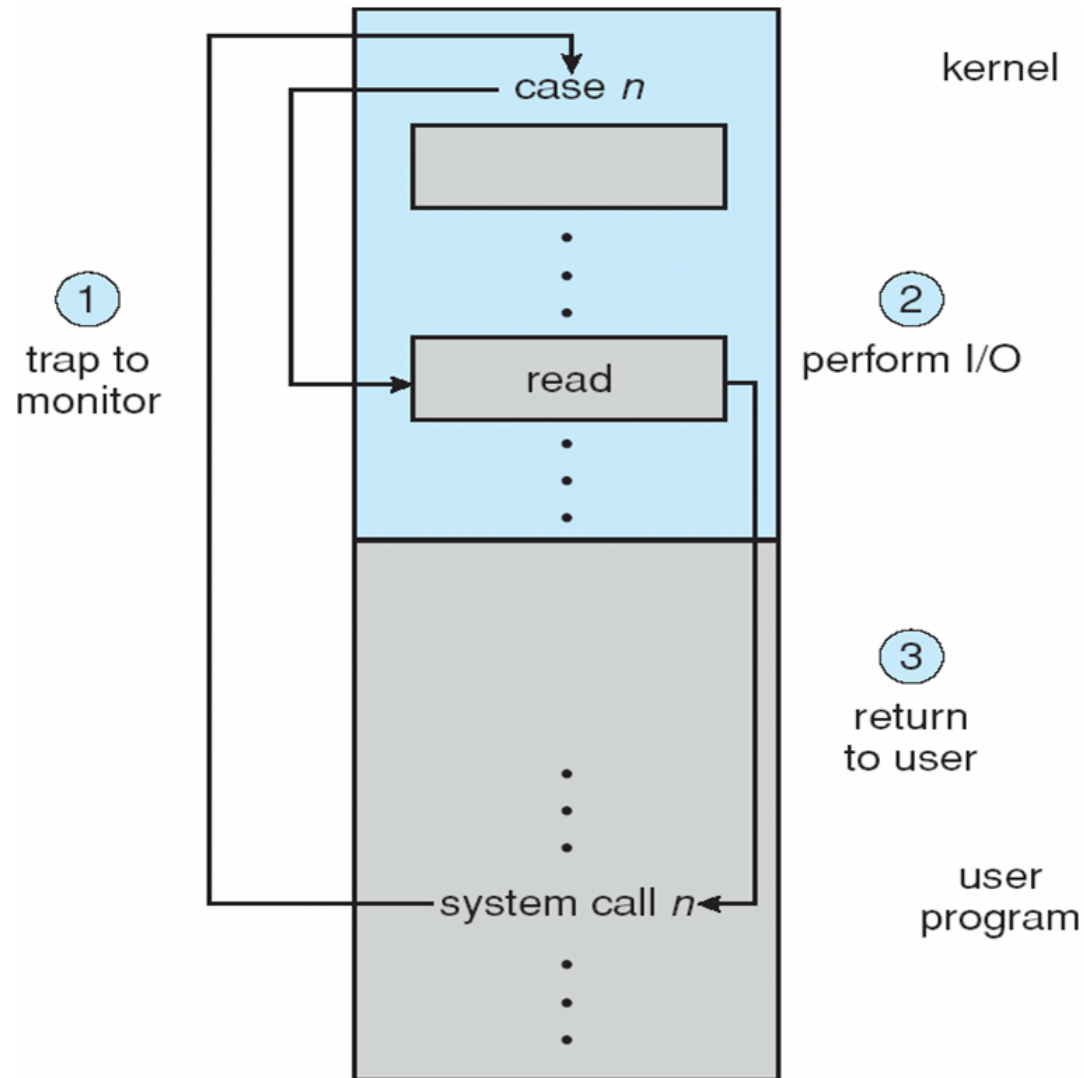
Kernel Data Structures

- 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
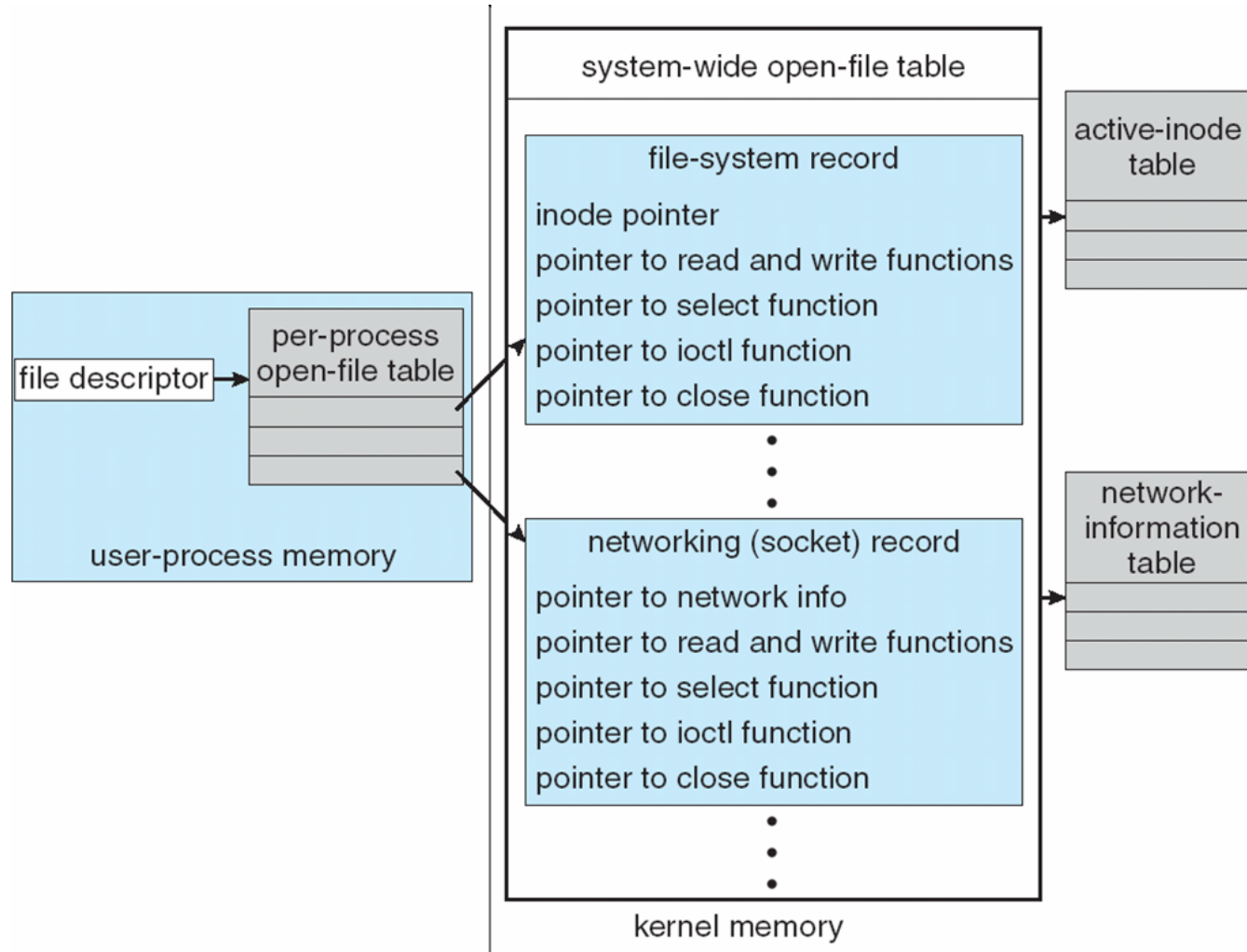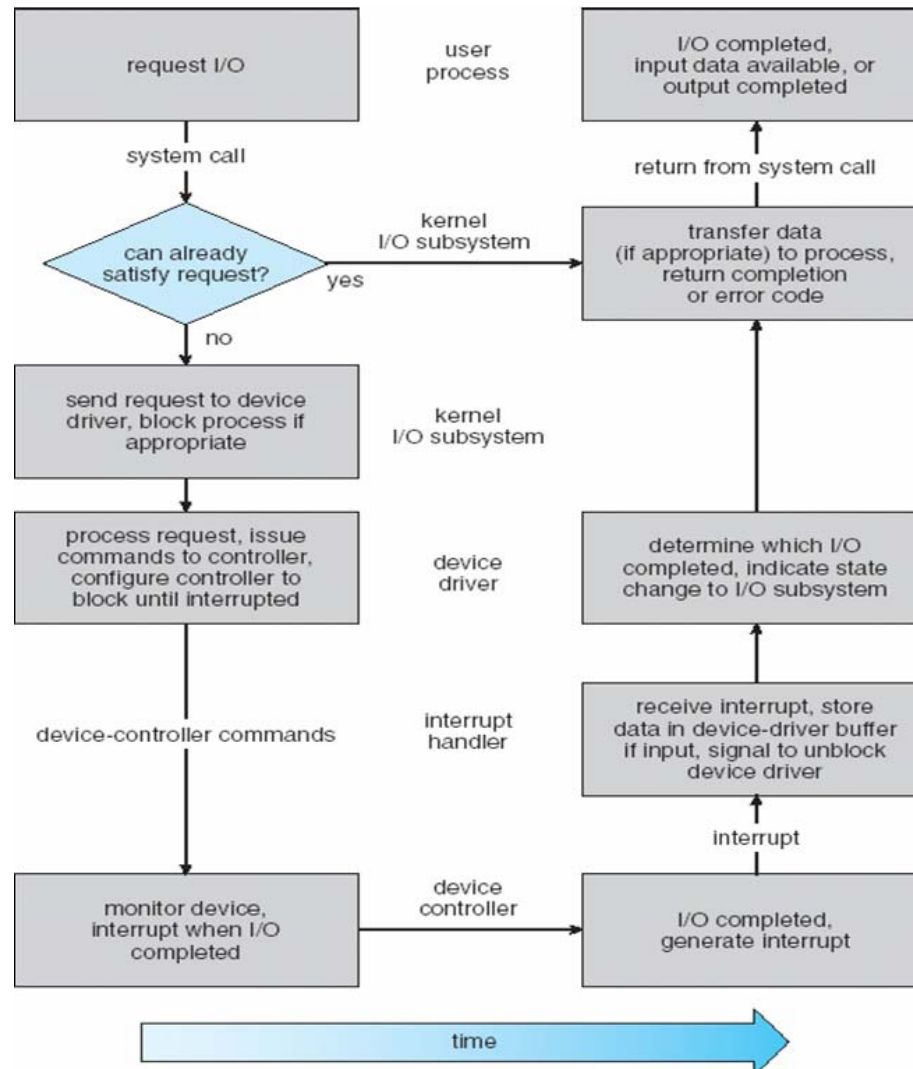
I/O Requests to Hardware Operations

- 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
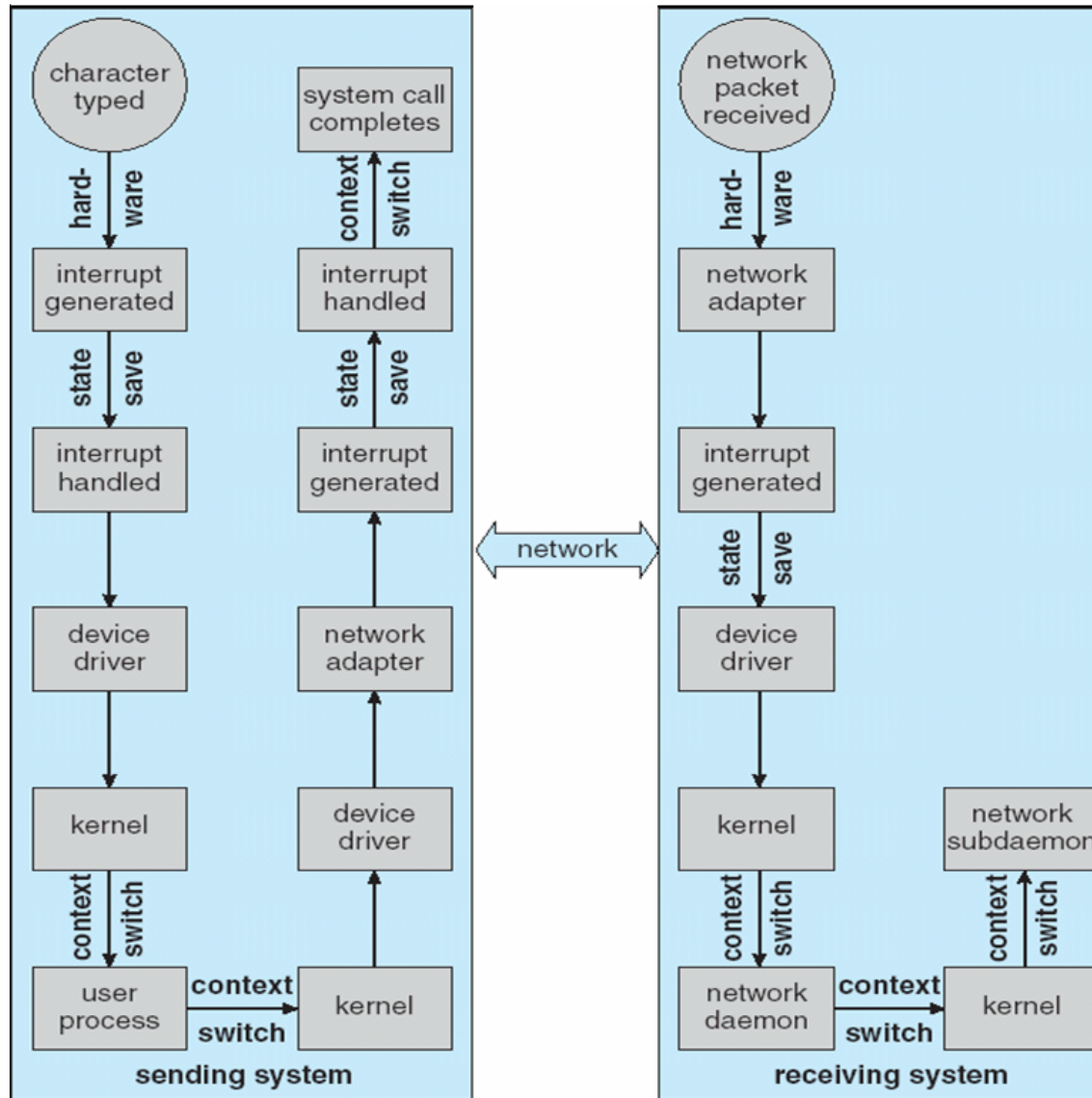
The STREAMS Structure
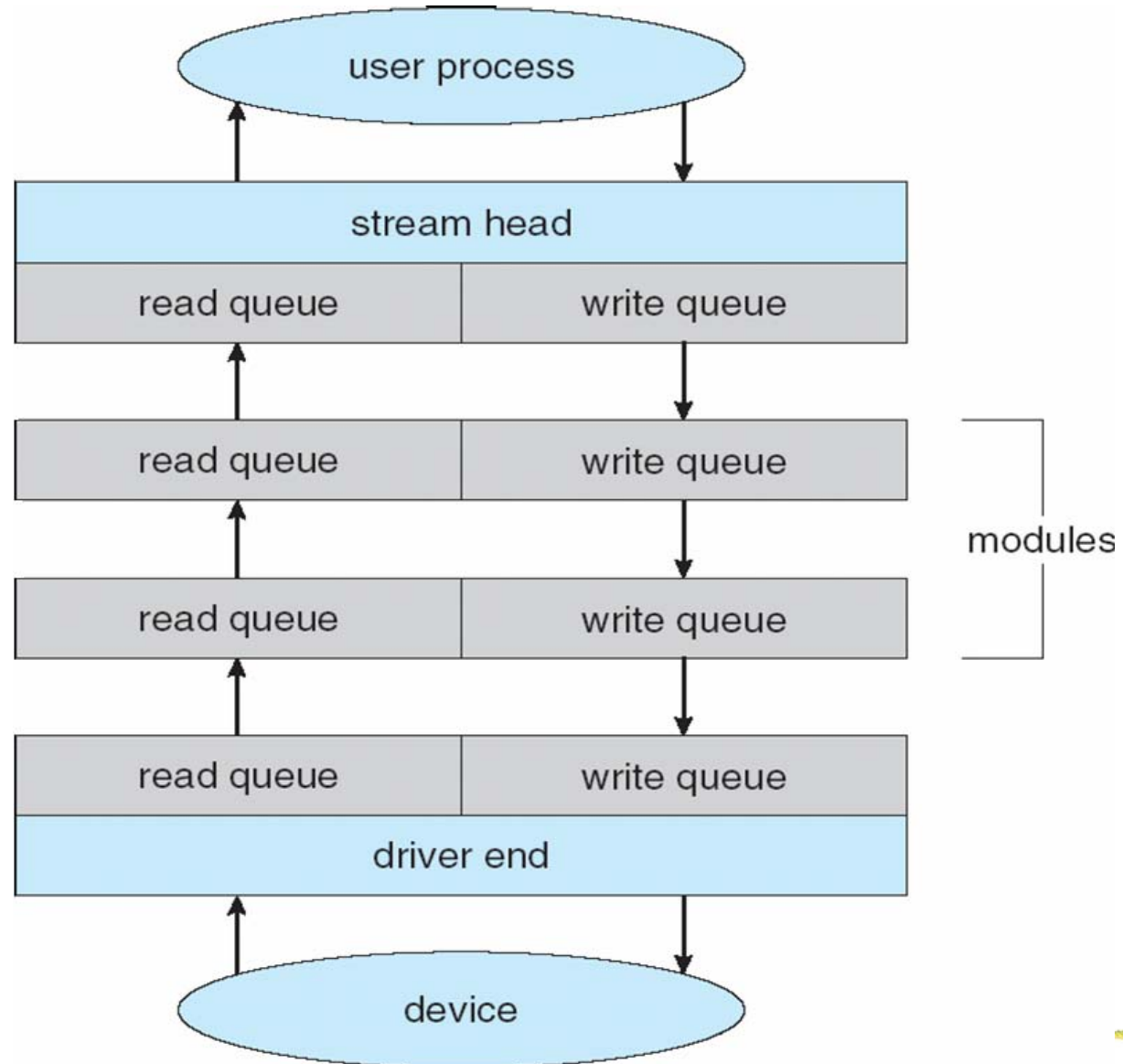
Performance

- 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

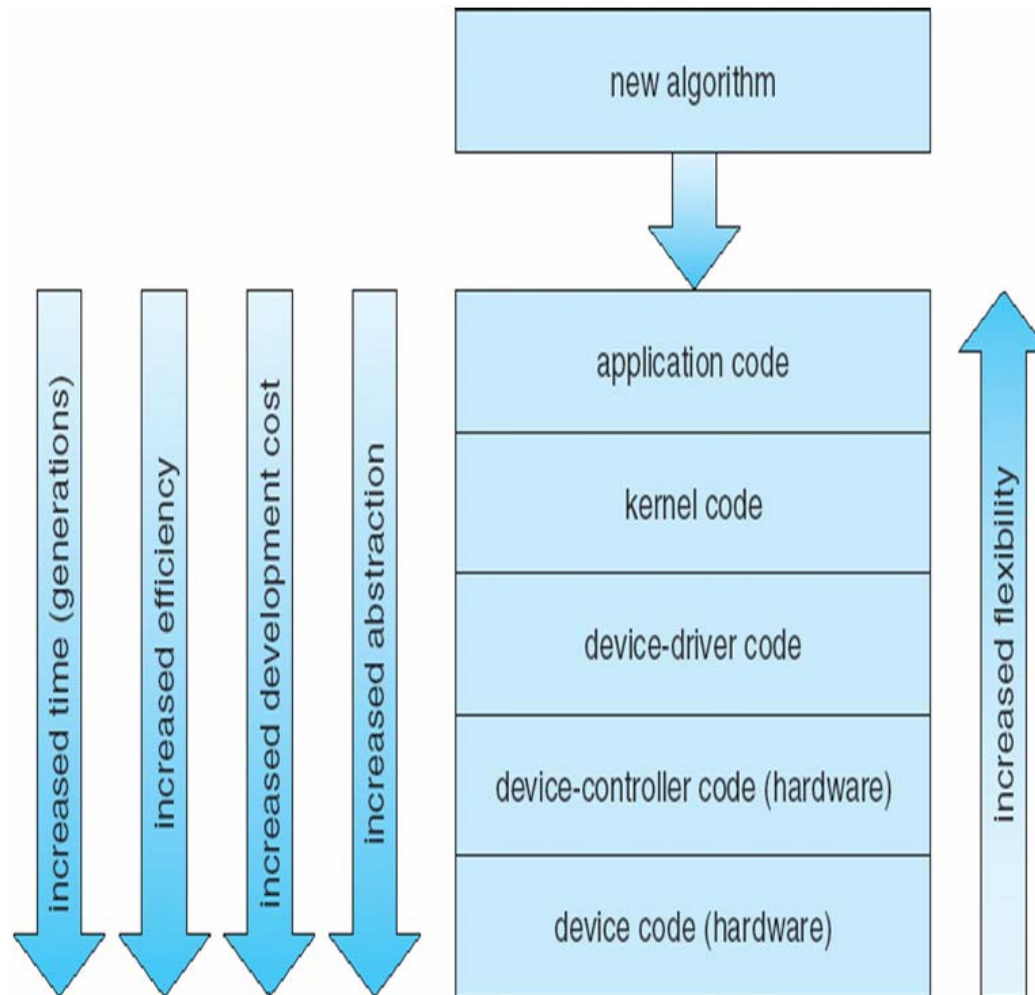# Intercomputer Communications

Improving Performance

- Reduce number of context switches

- Reduce data copying

- Reduce interrupts by using large transfers, smart controllers, polling

- Use DMA

- Balance CPU, memory, bus, and I/O performance for highest throughput

Device-Functionality Progression

End of Chapter 13