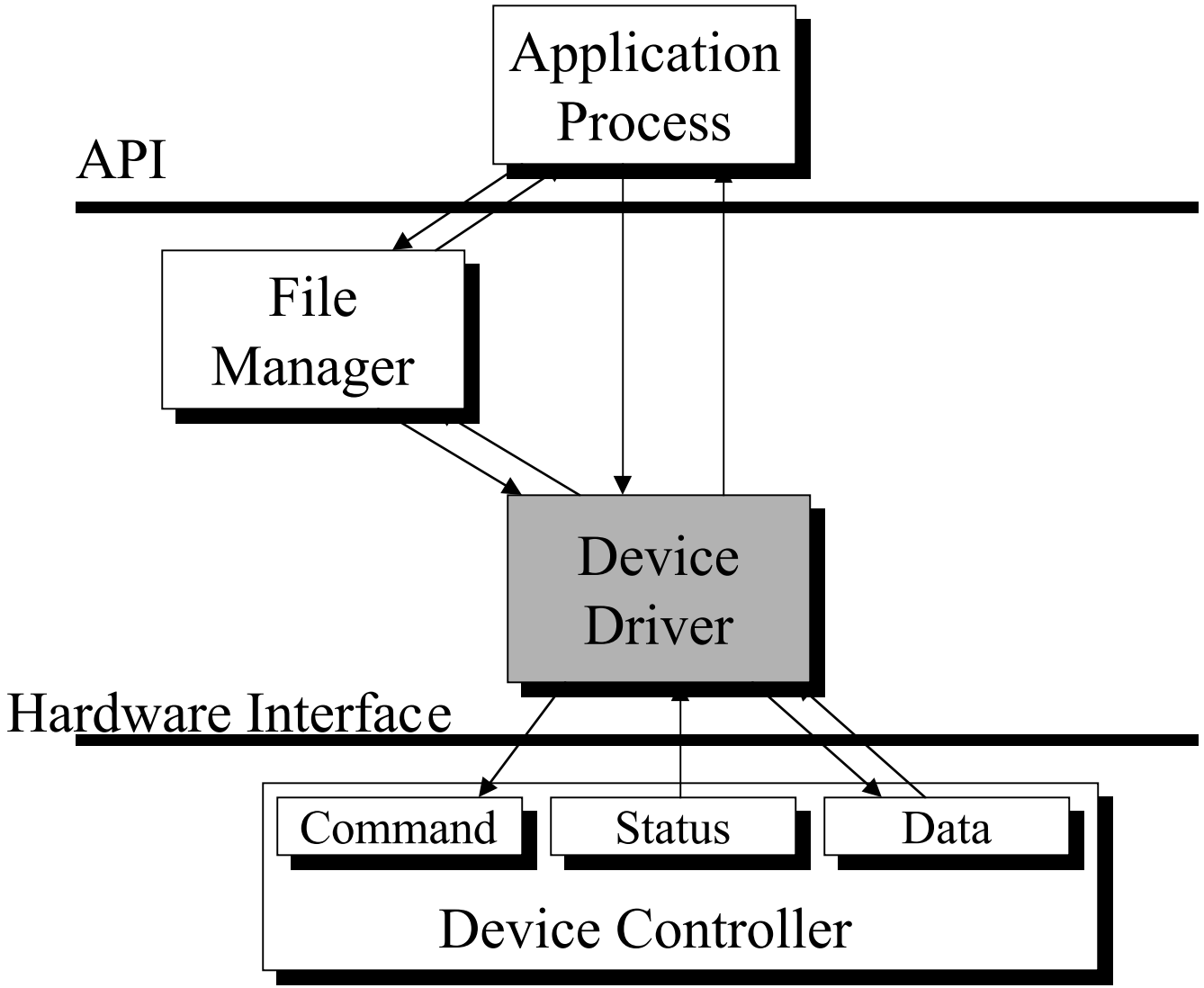
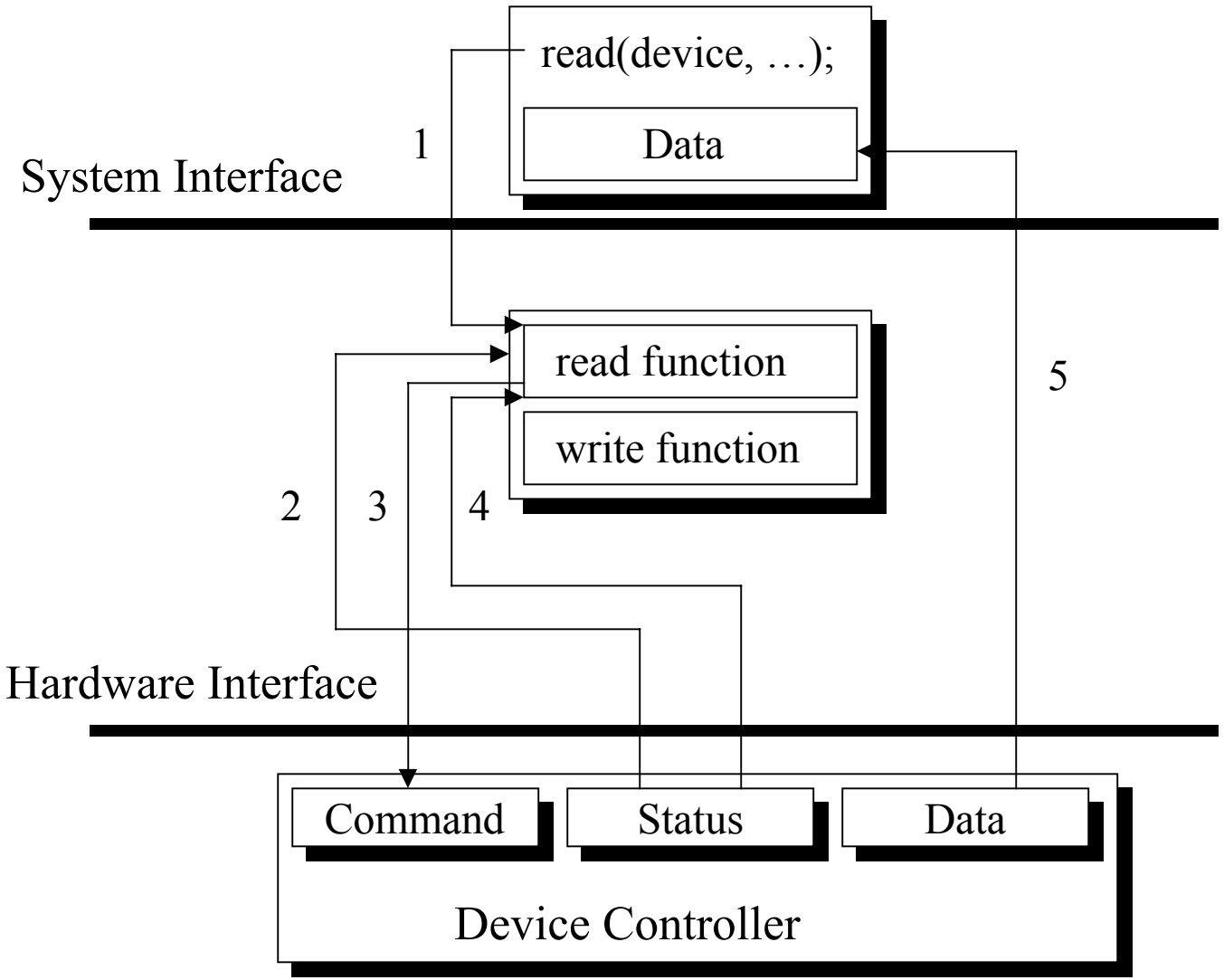
Device Management

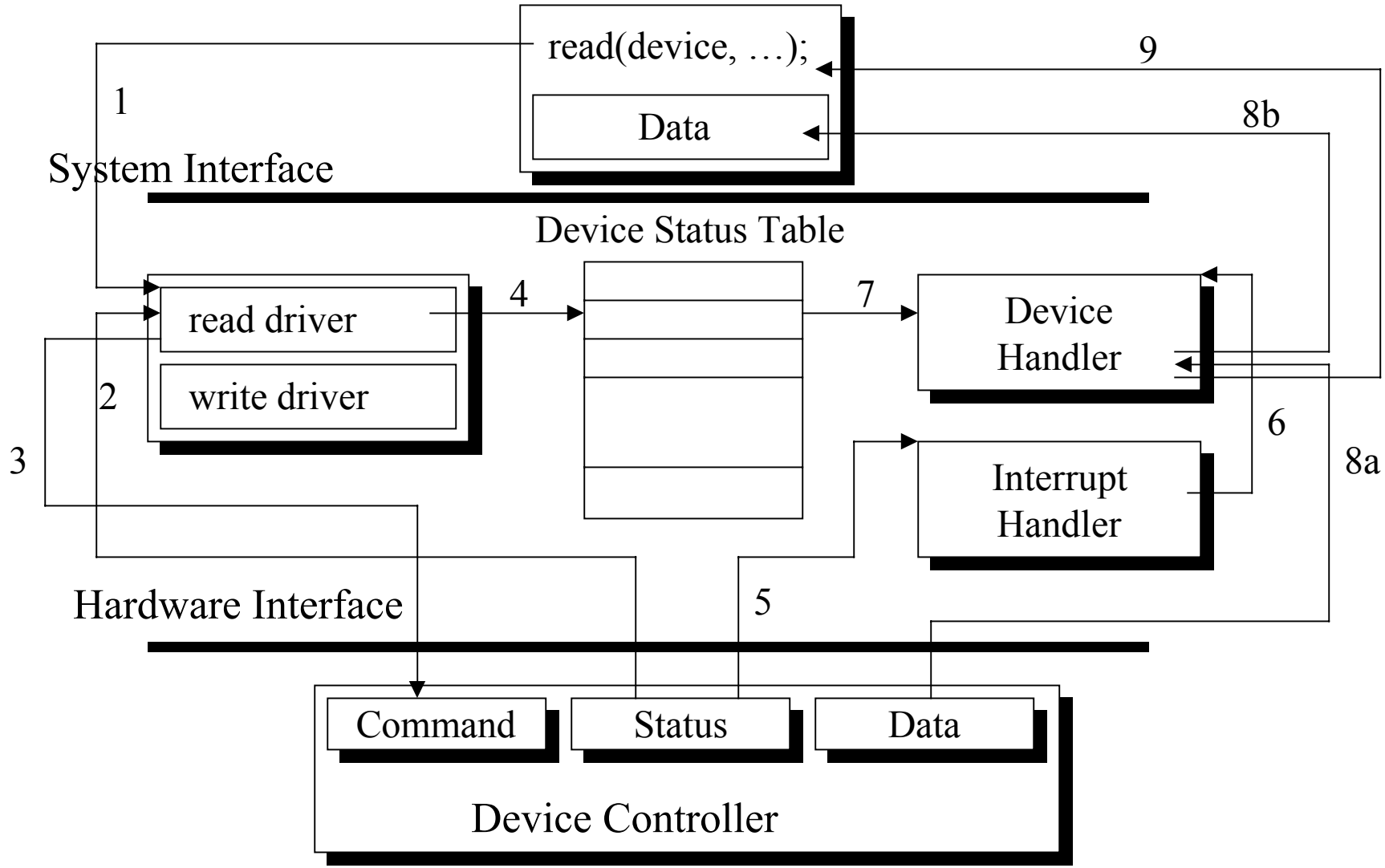
Device Management Organization



Read with Polling



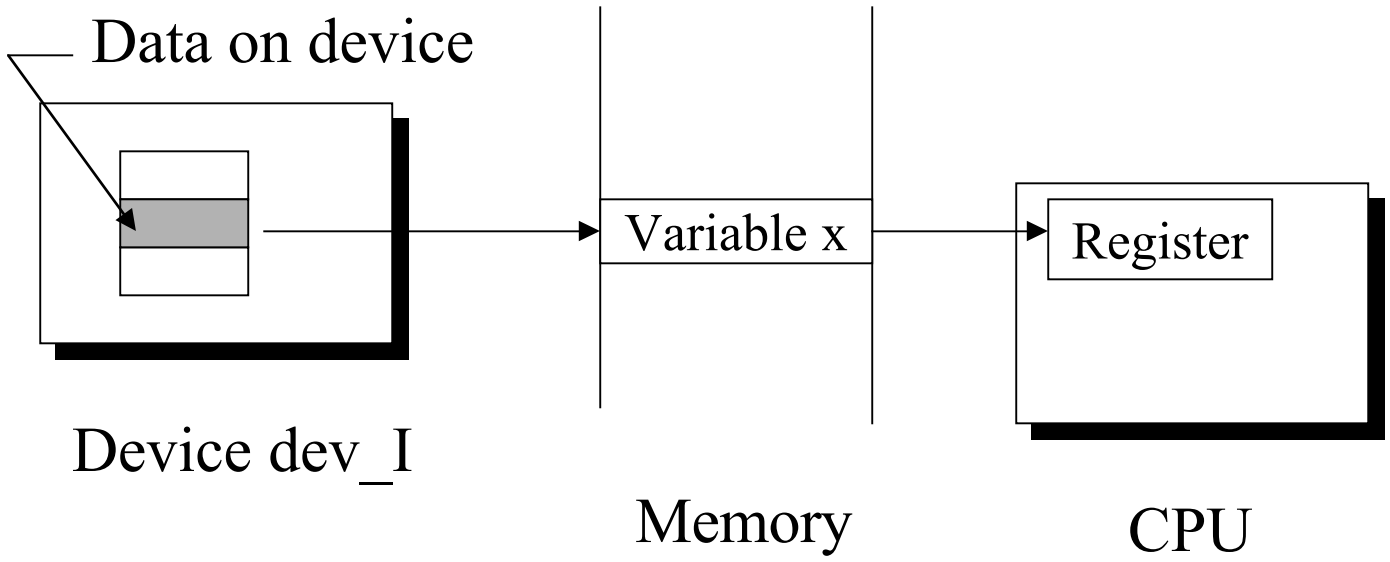
Read Using Interrupts



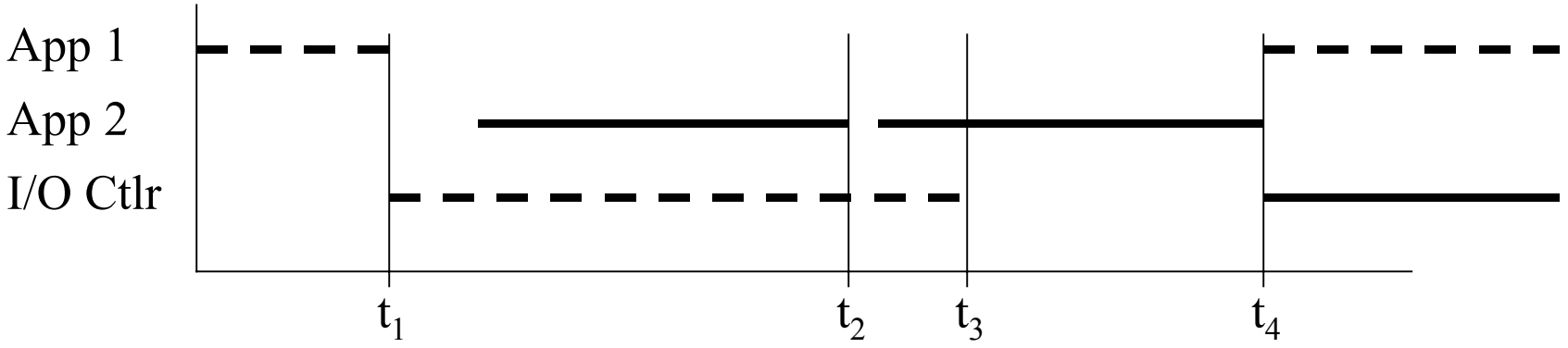
CPU-I/O Overlap

```
. . .
read(dev_I, "%d", x);
y = f(x)
. . .
```

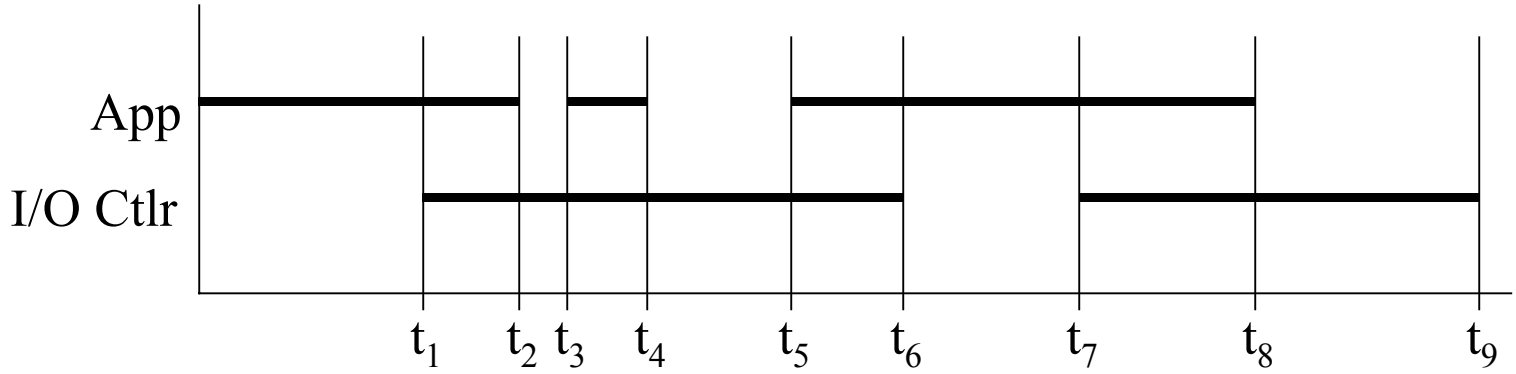
```
. . .
startRead(dev_I, "%d", x);
. . .
While(stillReading()) ;
y = f(x)
. . .
```



I/O - CPU Overlap

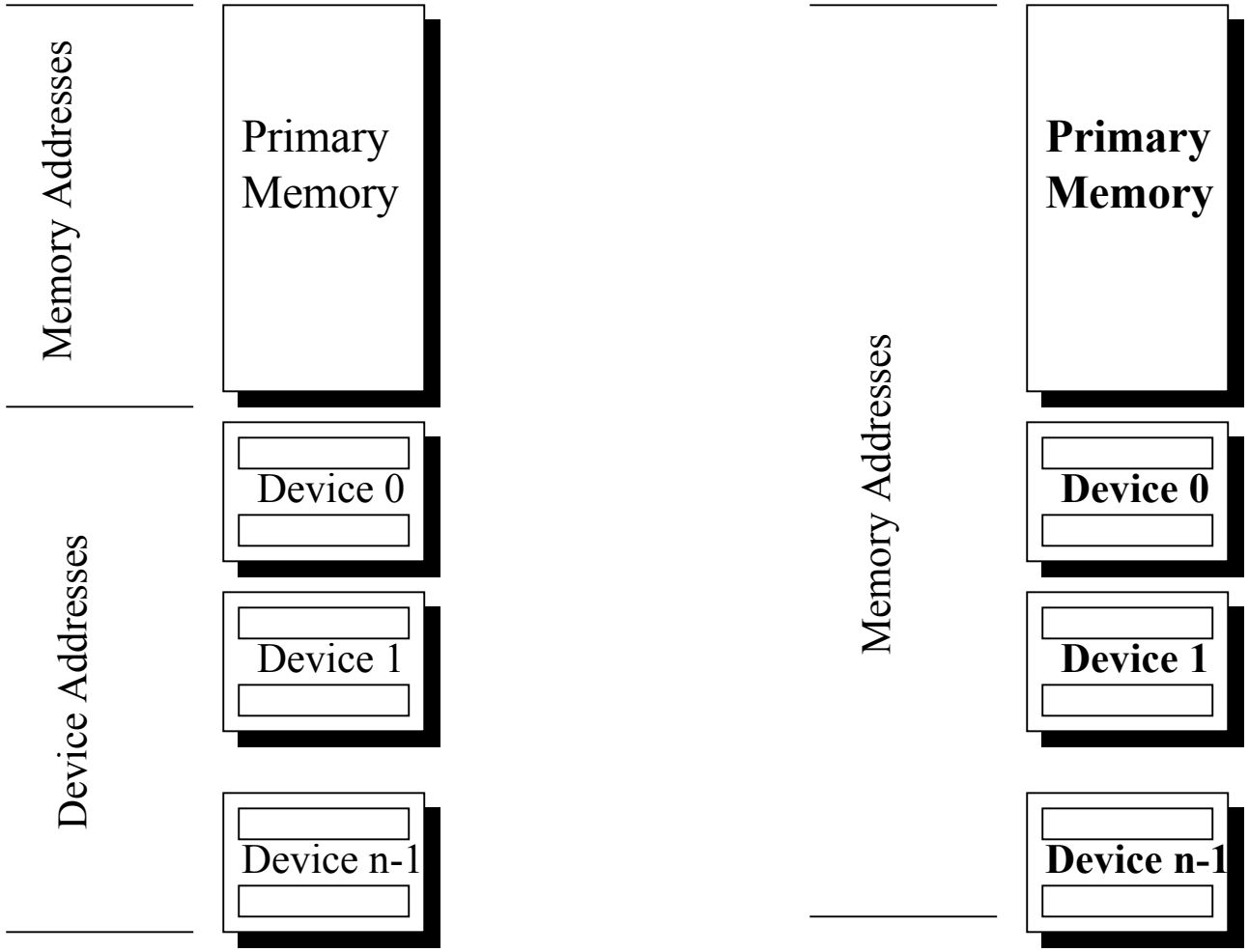


Overlapping App 1's I/O with App 2

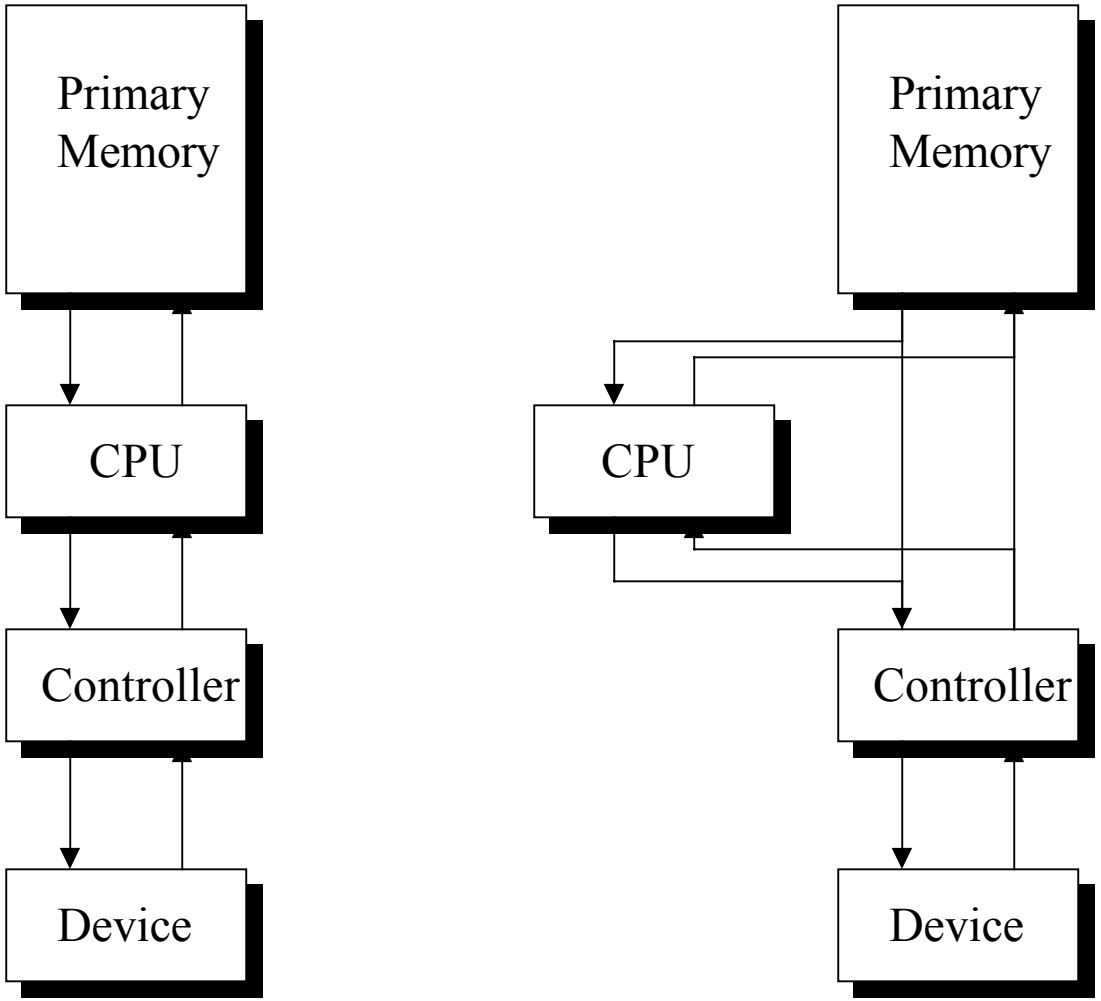


Overlapping App CPU with its own I/O

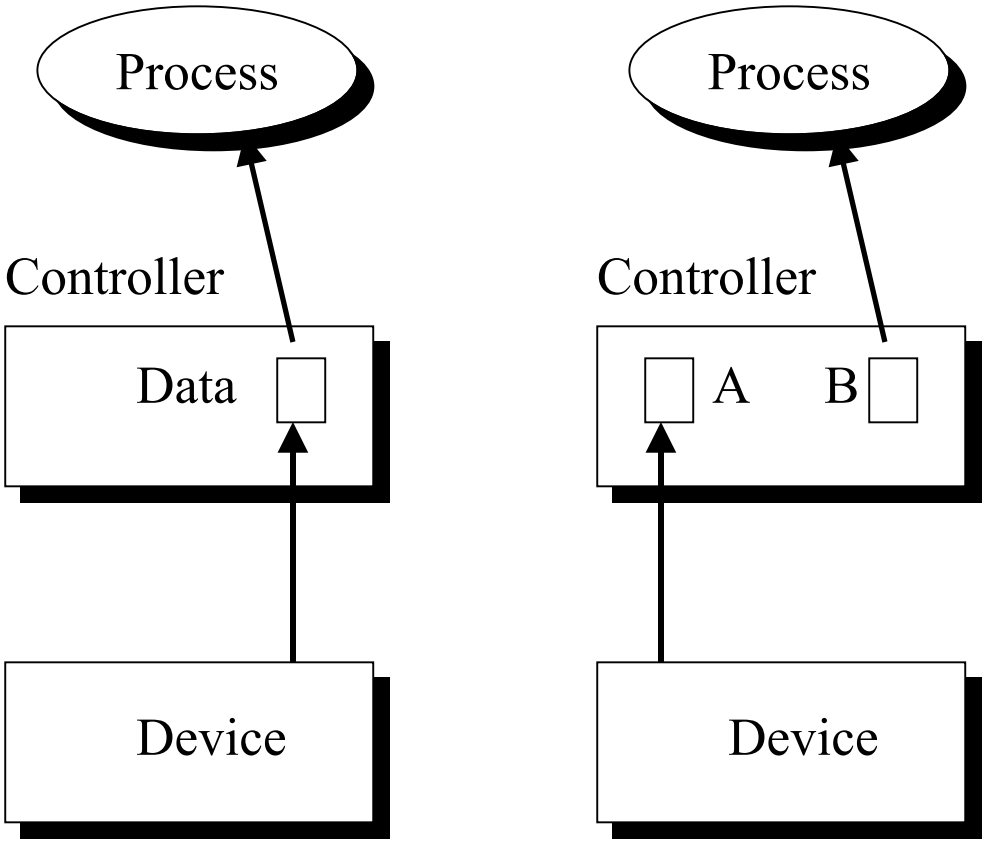
Memory Mapped I/O



Direct Memory Access



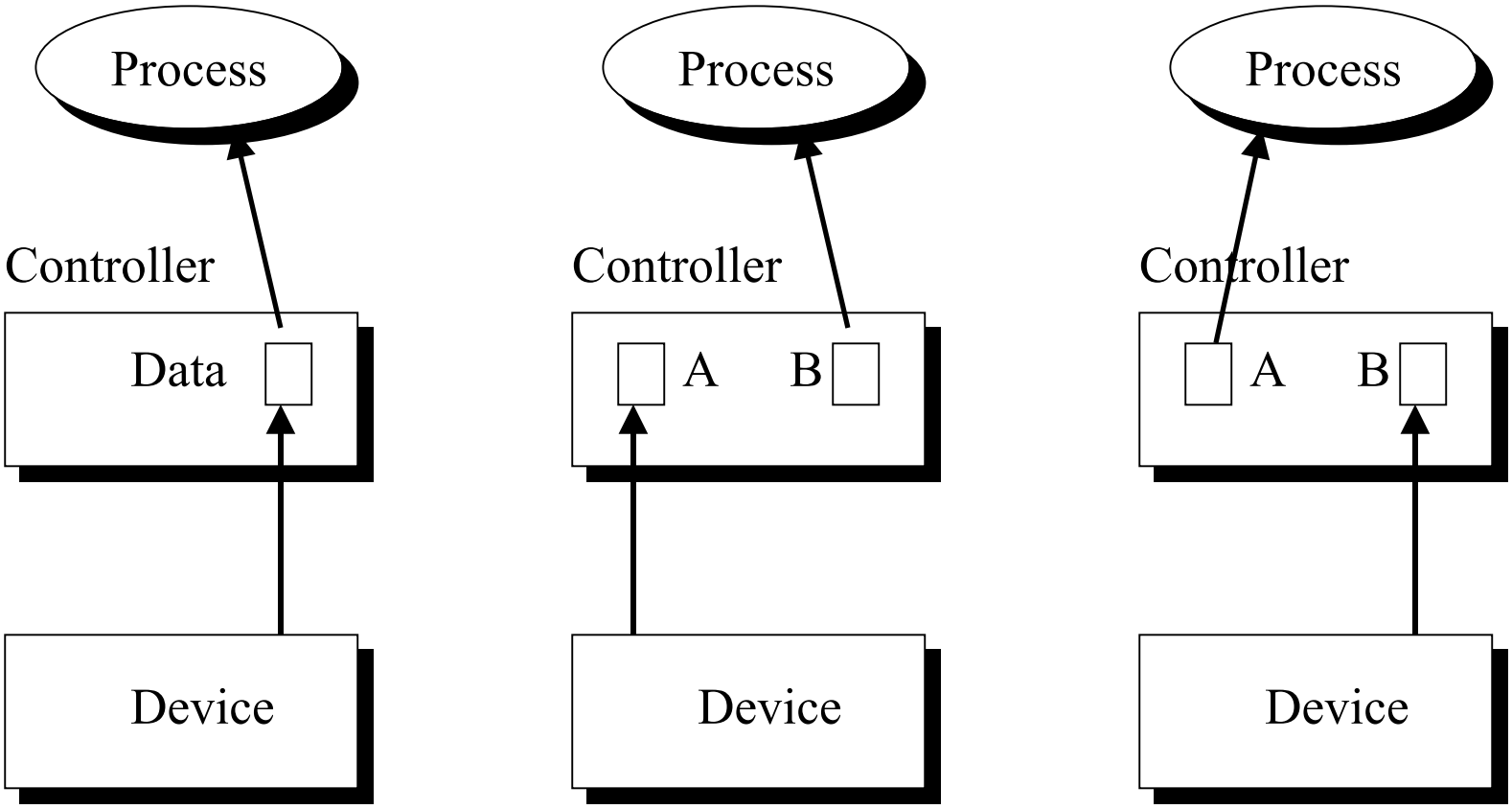
Hardware Buffering



Unbuffered

Process reads b_{i-1}
Controller reads b_i

Hardware Buffering

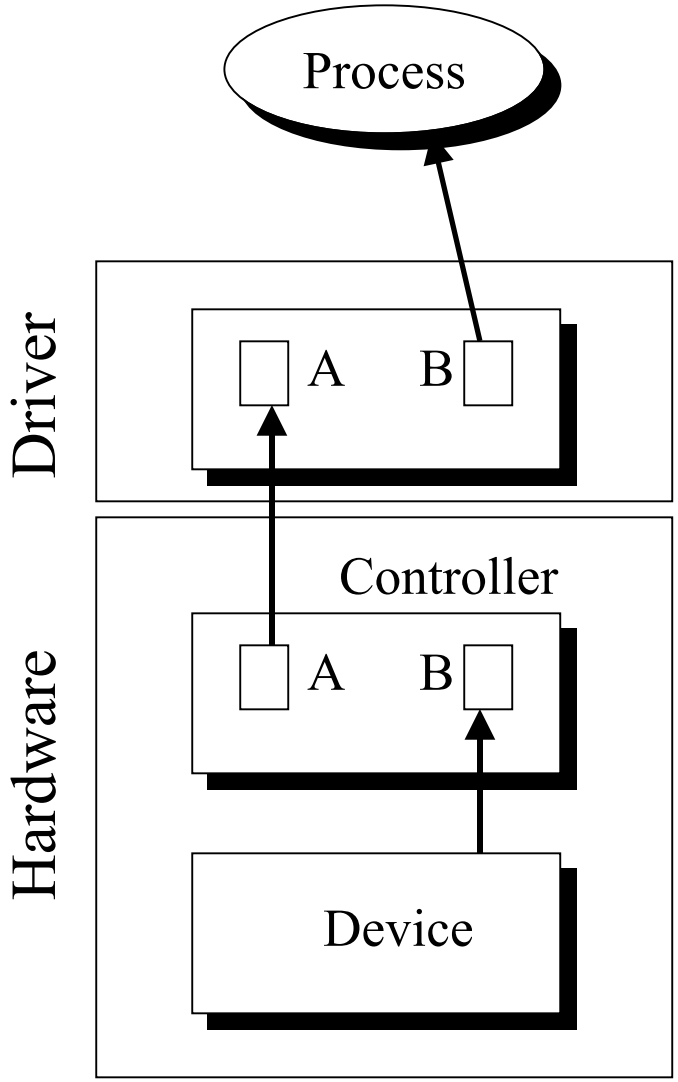


Unbuffered

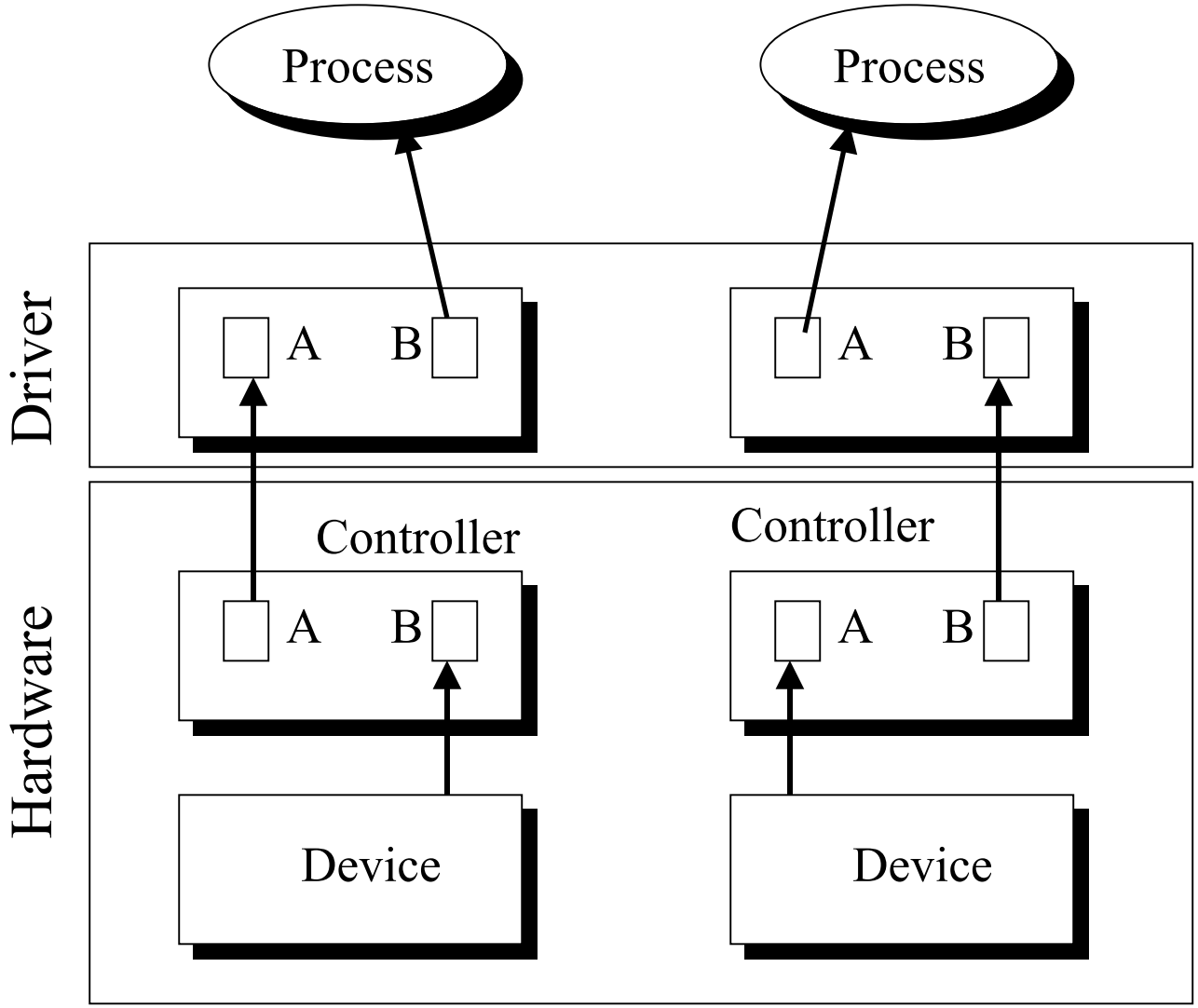
Process reads b_{i-1}
Controller reads b_i

Process reads b_i
Controller reads b_{i+1}

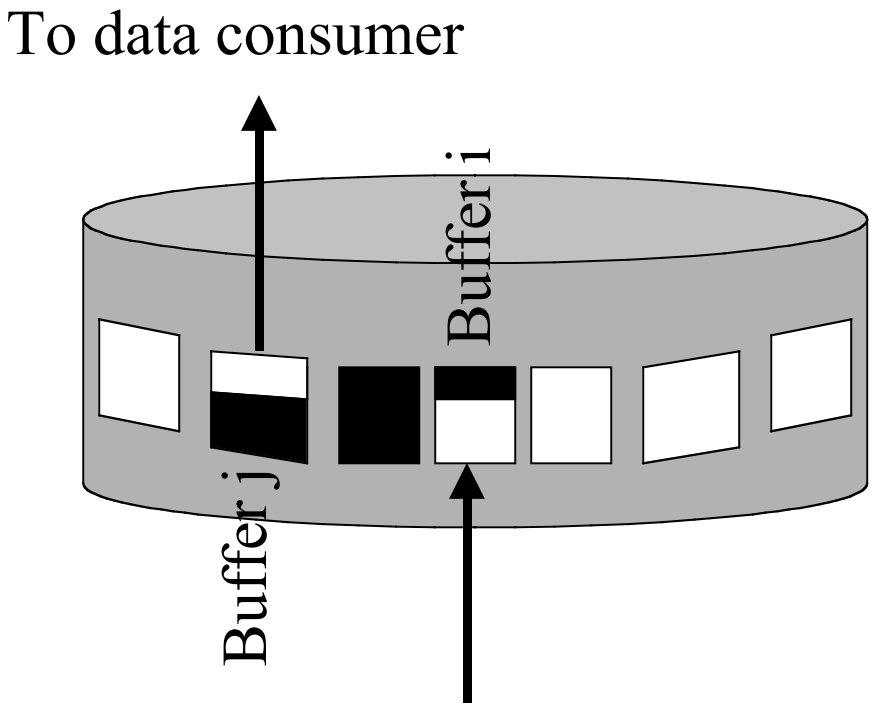
Buffering in the Driver



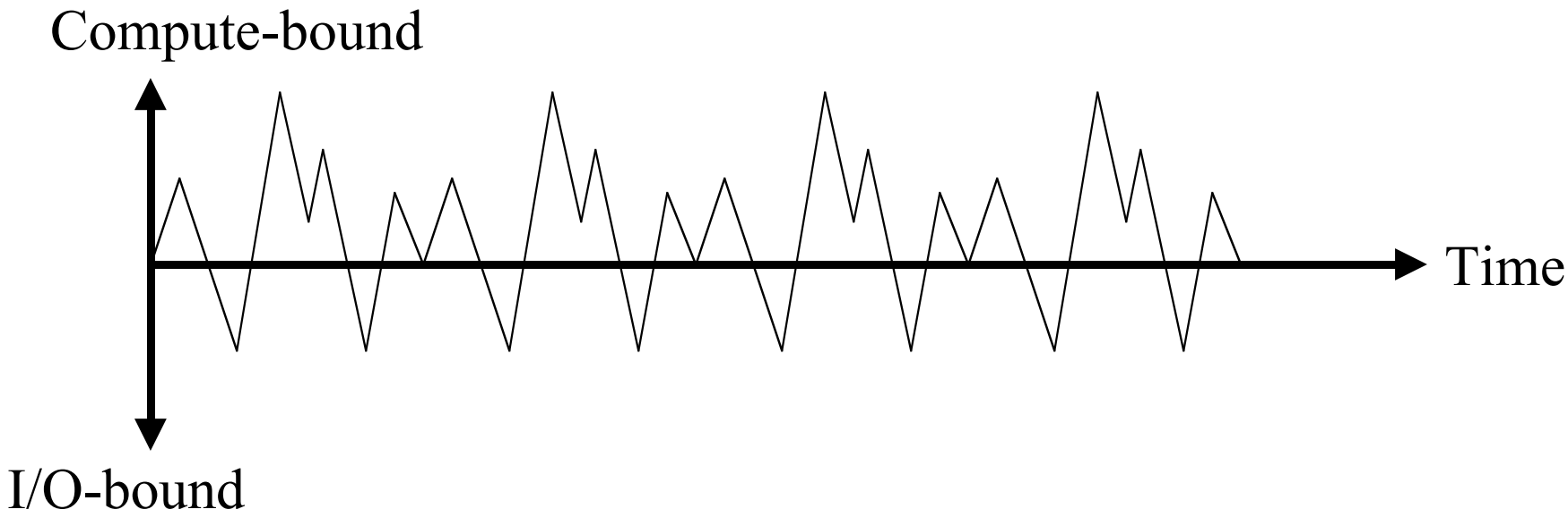
Buffering in the Driver



A Ring Buffer



Compute vs I/O Bound



Application Programming Interface

- Functions available to application programs
- Abstract all devices to a few interfaces
- Make interfaces as similar as possible
  - Block vs character
  - Sequential vs direct access
- Device driver implements functions (one entry point per API function)

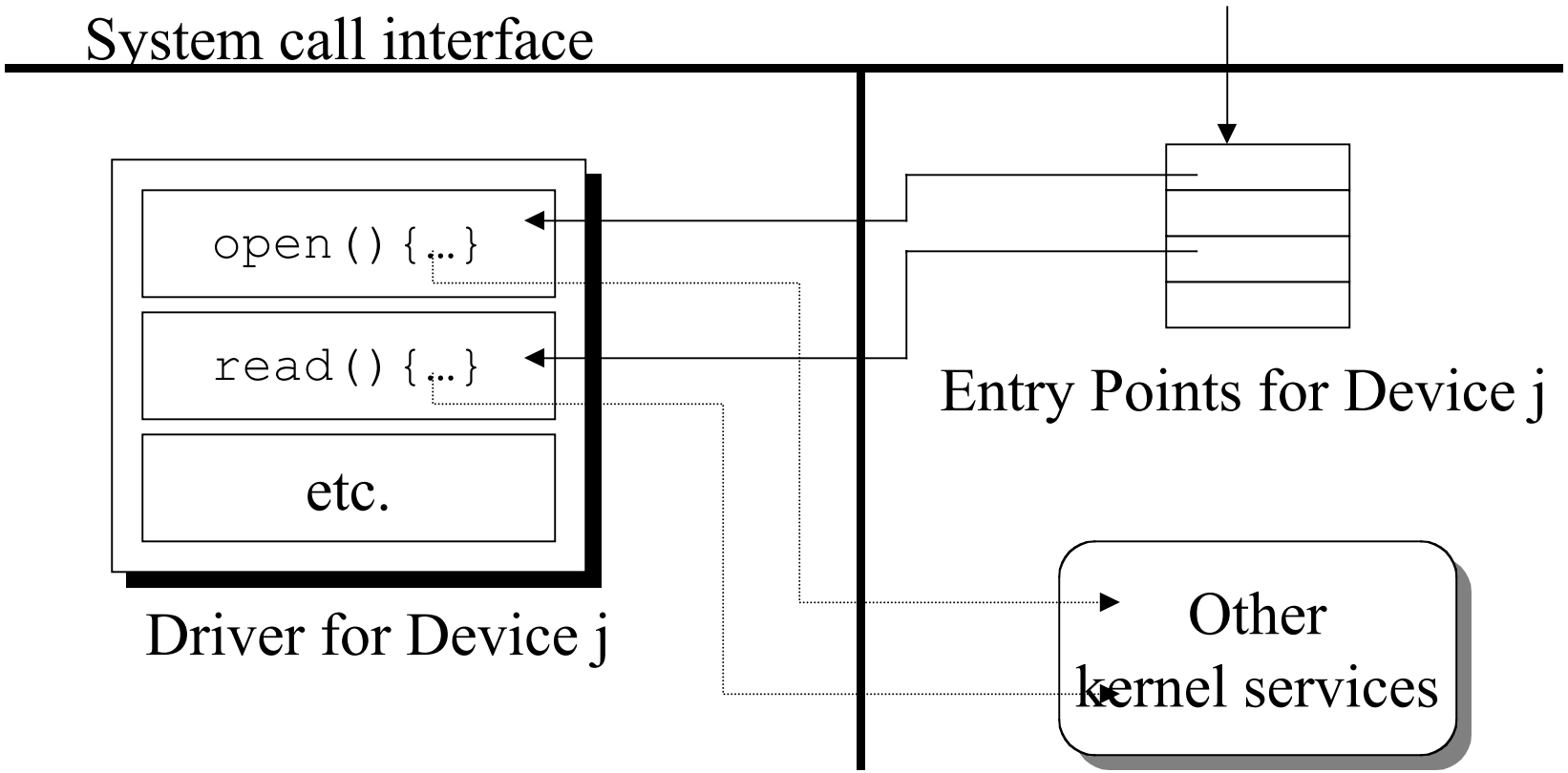
BSD UNIX Driver

`open`	Prepare dev for operation
`close`	No longer using the device
`ioctl`	Character dev specific info
`read`	Character dev input op
`write`	Character dev output op
`strategy`	Block dev input/output ops
`select`	Character dev check for data
`stop`	Discontinue a stream output op

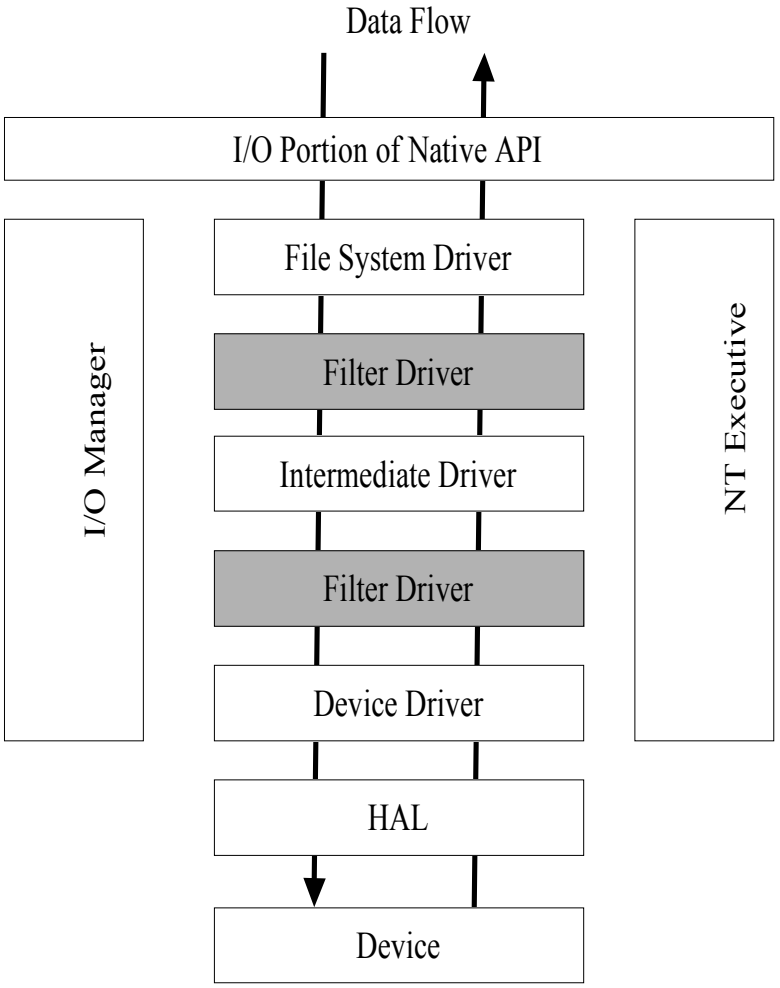
Driver-Kernel Interface

- Drivers separate from rest of kernel
- Kernel makes calls on specific functions, drivers implement them
- Drivers use other kernel functions for:
  - Device allocation
  - Resource (e.g., memory) allocation
  - Scheduling
  - etc. (varies from OS to OS)

Reconfigurable Drivers



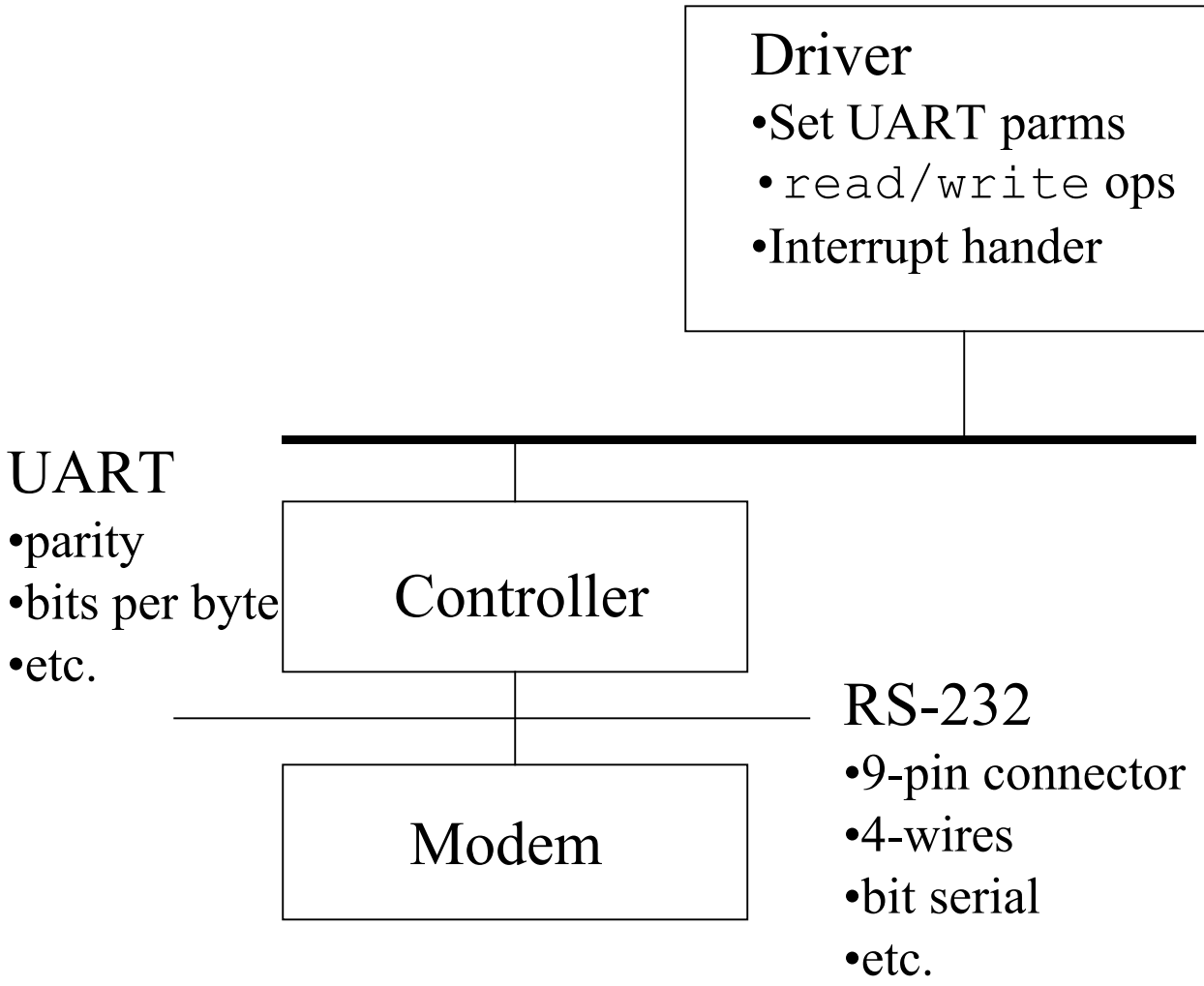
NT Driver Organization



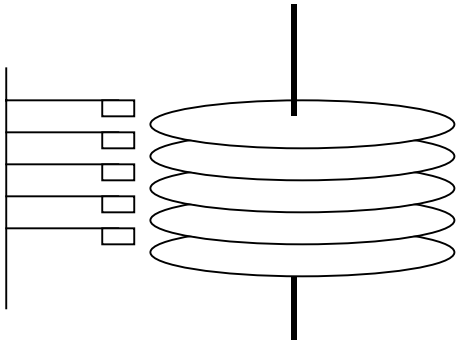
NT Device Drivers

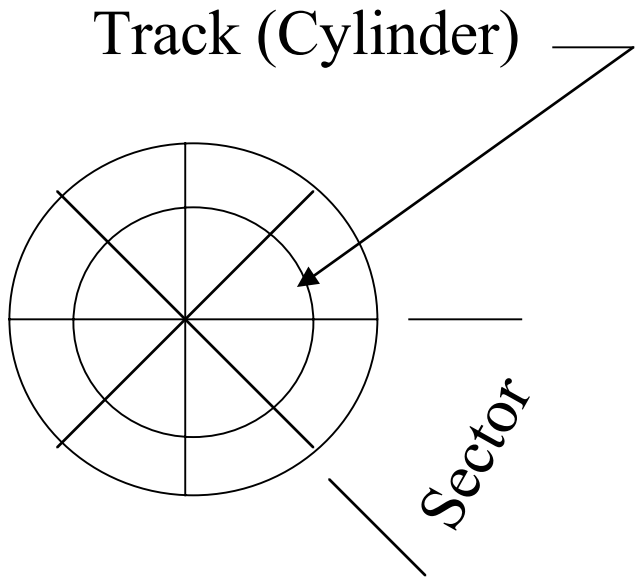
- API model is the same as a file
- Extend device management by adding modules to the stream
- Device driver is invoked via an Interrupt Request Packet (IRP)
  - IRP can come from another stream module
  - IRP can come from the OS
  - Driver must respond to minimum set of IRPs
- See Part I of notes

Serial Communication Device



Rotating Storage





Top View of a Surface

MS Disk Geometry

0x00	0x02	<a jump instruction to 0x1e>
0x03	0x0a	Computer manufacturer name
0x0b	0x0c	Sectors per cluster (discussed in Exercise 11)
0x0d	0x0f	Reserved sectors for the boot record
0x10	0x10	Number of FATs
0x11	0x12	Number of root directory entries
0x13	0x14	Number of logical sectors
0x15	0x15	Medium descriptor byte (used only on old versions of MS-DOS)
0x16	0x17	Sectors per FAT
0x18	0x19	Sectors per track
0x1a	0x1b	Number of surfaces (heads)
0x1c	0x1d	Number of hidden sectors
0x1e	…	Bootstrap program

Disk Optimizations

- Transfer Time: Time to copy bits from disk surface to memory
- Disk latency time: Rotational delay waiting for proper sector to rotate under R/W head
- Disk seek time: Delay while R/W head moves to the destination track/cylinder
- Access Time = seek + latency + transfer

Optimizing Seek Time

- Multiprogramming on I/O-bound programs => set of processes waiting for disk
- Seek time dominates access time => minimize seek time across the set
- Tracks 0:99; Head at track 75, requests for 23, 87, 36, 93, 66
- FCFS: 52+ 64 + 51 + 57 + 27 = 251 steps

Optimizing Seek Time (cont)

- Requests = 23, 87, 36, 93, 66
- SSTF: (75), 66, 87, 93, 36, 23
  - 11 + 21 + 6 + 57 + 13 = 107 steps
- Scan: (75), 87, 93, 99, 66, 36, 23
  - 12 + 6 + 6 + 33 + 30 + 13 = 100 steps
- Look: (75), 87, 93, 66, 36, 23
  - 12 + 6 + 27 + 30 + 13 = 87 steps

Optimizing Seek Time (cont)

- Requests = 23, 87, 36, 93, 66
- Circular Scan: (75), 87, 93, 99, 23, 36, 66
  - 12 + 6 + 6 + home + 23 + 13 + 30 = 90 + home
- Circular Look: (75), 87, 93, 23, 36, 66
  - 12 + 6 + home + 23 + 13 + 30 = 84 + home