Periphals and Interfaces

Sandhya Dwarkadas

I/O Issues

Addressing

Interconnection

Interface

I/O Devices

Block devices - info stored in fixed-size blocks that are addressable

Character devices - accepts or delivers a stream of characters - no block structure or seek operation - e.g., printers, mice, network interfaces

I/O unit can be divided into -

- Electro-Mechanical component
- Electronic component - device controller - PCB in slot on motherboard - usually with a standardized interface

Magnetic Disks

Block device - 512-32K byte blocks

Device-independent interfaces - IDE (Integrated Drive Electronics) or SCSI (Small Computer System Interface) disk controller interfaces

Disk organization - tracks, sectors, cylinders

Contributors to disk access time

- seek time
- rotational latency
- transfer time
- controller overhead

Buses

A bus is a shared communication link with one set of "wires" to connect multiple parts of the computer

- Processor-Memory Buses
- Backplane Buses
- I/O Buses

Bus Performance

Limited by physical factors - length of the bus and number of devices

Minimize bus latency by streamlining the communication path

Maximize bus bandwidth using buffering and larger blocks of data

Bus Transactions

A bus transaction has two parts:

- sending the address
- receiving or sending the data

Bus Design Options

Decisions depend on the cost and performance goals

Option	High Performance	Low Cost
Bus width	Separate address and data lines	Multiplex address and data lines
Data width	Wide	Narrow
Transfer size	Multiple-word reduces overhead	Single-word is simpler
Bus masters	Multiple	Single
Split transaction	Yes	No
Clocking	Synchronous	Asynchronous

Synchronous Versus Asynchronous Buses

Synchronous - includes a clock in the control lines and a fixed protocol for address and data relative to the clock

- simple, therefore fast
- all devices must run at same speed
- clock-skew limits the length of the bus if speed is desired

Asynchronous - self-timed, handshaking protocol used between the sender and the receiver - `DataReq, DataRdy,` and `Ack` control lines

- easier to accommodate a wide variety of devices
- length of the bus can be increased
- slower due to handshaking

Bus Arbitration

Bus Request, Bus Grant, and Bus Release signals to control access to bus

- Daisy chain
- Centralized, parallel (PCI bus uses this)
- Distributed, self-selection
- Distributed, collision detection

Methods of Addressing I/O Devices

Memory-mapped I/O - e.g., MIPS, SPARC, Motorola processors

Alternate address space with special I/O instructions - e.g., IBM, Intel processors
Example: "IN acc, port #"

Techniques for Performing I/O

Programmed I/O with busy waiting

Interrupt-driven I/O

DMA (Direct-Memory Access) I/O

I/O using data channels

Interrupts

An asynchronous electrical event that uses a hardware interrupt request line (IRQ)

Causes processor to halt execution of current program and execute I/O code

IRQ mapped to interrupt vector, which locates corresponding interrupt service software

User-Space I/O Software

System calls made via library procedures
Example: count = write(fd, buffer, nbytes)
printf, scanf - does more work

E.g., Printer:

- Create a special process called a daemon
- Create a special **spooling** directory
  - deals with dedicated I/O devices in a multiprogrammed system
  - prevents starvation

Principles of I/O Software

- Device independence
- Uniform naming

Structure

- User-level software called by user process (top) - make I/O calls, format I/O, spooling
- Device-independent operating systems software - naming, protection, blocking, buffering, allocation
- Device drivers - set up device registers, check status
- Interrupt handlers - wake up driver when I/O is complete
- Hardware - perform I/O operation

Hardware Actions on Interrupt

1. Assert interrupt line
2. CPU then asserts int. ack. line when ready
3. Device controller then places interrupt vector on data lines
4. CPU saves interrupt vector temporarily
5. CPU pushes PC and PSW onto stack
6. CPU locates new PC by using interrupt vector as index into table at a fixed location in memory

Software Actions on Interrupt

1. Save all registers on stack or in system table (must disable interrupts while doing this)
2. Read device register for exactly which device - generally shared by all devices of a given type
3. Read status codes of device
4. Handle any I/O errors
5. Update data structures
6. Tell device interrupt has been processed if necessary
7. Restore registers
8. Execute return from interrupt

Interconnection Networks

Types

- RS232 standard - slow but cheap dedicated wires for terminal network
- Massively parallel processor network (MPP) - max. distance 25 meters
- Local area networks (LAN) - ethernet, FDDI - few km.
- Wide area networks (WAN) - e.g., ARPANET (1st), Internet - 1000s of km

Interconnection Network Media

Twisted pair

Coaxial cable

Fiber optic

Network Topology

Shared - e.g., ethernet - CSMA/CD (Carrier-Sense Multiple Access/Collision Detect)

Switched - point-to-point

- star - e.g., ATM
- ring - e.g., Token-Ring, FDDI (Fiber Distributed Data Interface)
- crossbar
- omega (multistage network), fat tree

Network Architecture

Design and Implementation Guide

Principle of Abstraction - layering of protocols

A protocol provides a communication service to the next higher-level layer

- Service interface - e.g., send and receive
- Peer interface - form and meaning of messages exchanged between peers

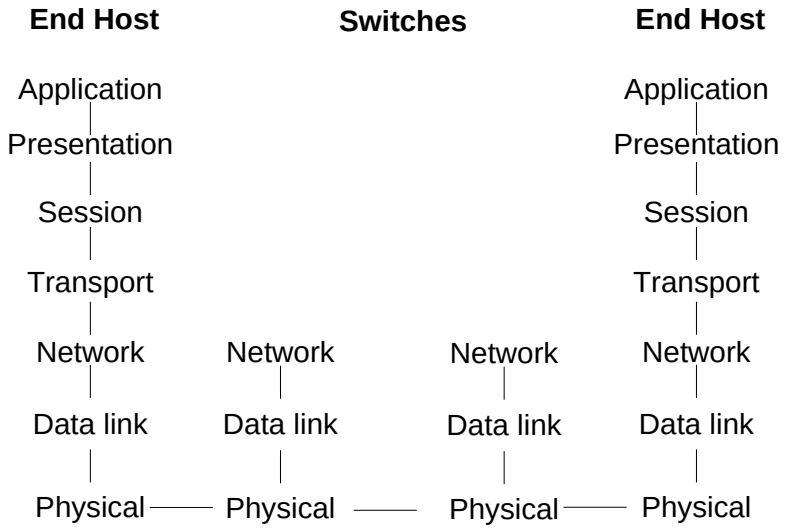
Network Architecture Issues

Encapsulation - header and body of message

Multiplexing/demultiplexing - keys in headers to identify target application

OSI Architecture

Partitioning of network functionality into seven layers - reference model for a protocol graph



Issues affecting software interface to network or I/O device

- Consistency with processor's cache
- Programmed I/O or DMA
- Polling versus interrupts

Network Performance Issues

Effective bandwidth/total latency is a function of -

- Sender Overhead - buffering, error-code generation, OS/controller actions
- Receiver overhead - buffering, error checking, OS/controller actions
- Time of flight (latency)
- Transmission time (a function of available bandwidth)