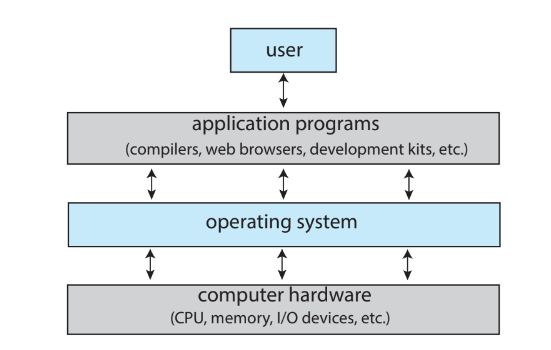
An ***operating system*** is software that manages a computer’s hardware. It acts as an intermediary between user and computer hardware. It provides an environment where programs can do useful work.

**Abstract of Computer components:**

* User
* Application programs – compilers, toolkits, web-browsers etc.
* Operating system
* Hardware



**Hardware** makes up the basic computing resources of the system (CPU, memory, I/O devices).

**Application programs** define ways in which these resources are to be used to solve users’ computing problems.

The OS manages the hardware and coordinates its use amongst various application programs for various users.

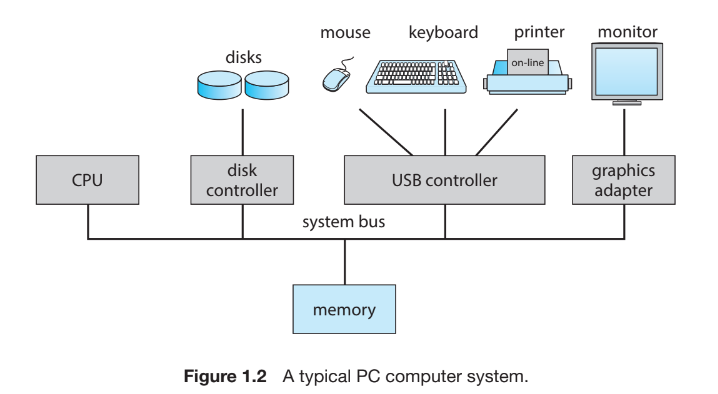
The OS is a ***control program***. A control program is responsible for managing user programs to prevent errors and improper use of the system.

**Kernel** is a program running at all the times on the computer. It is also referred as the core of CPU.

**Computer System Organization**

Modern computer system consists of one or more CPUs and multiple device controllers connected through one common bus which provides access to components and shared memory. Each **Device controller** is in charge of a specific device. It maintains local buffer storage and special-purpose registers and is responsible for moving data from the buffer and peripheral.

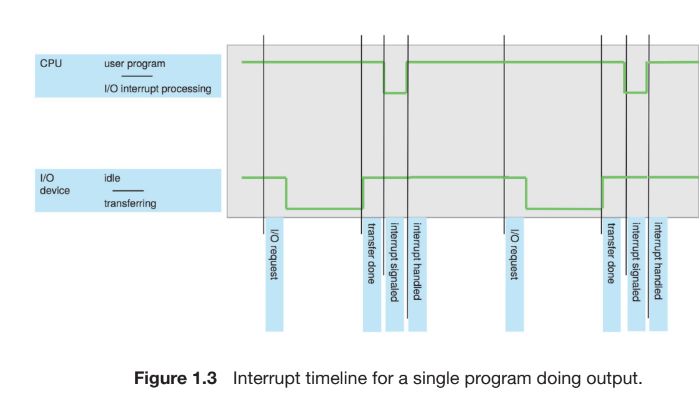
OS has a **device driver** for each device controller. It understands the controller and provides an interface to the OS of a device. Both the driver and controller compete for memory cycles. To ensure orderly access to the memory, memory controller synchronizes access to memory.



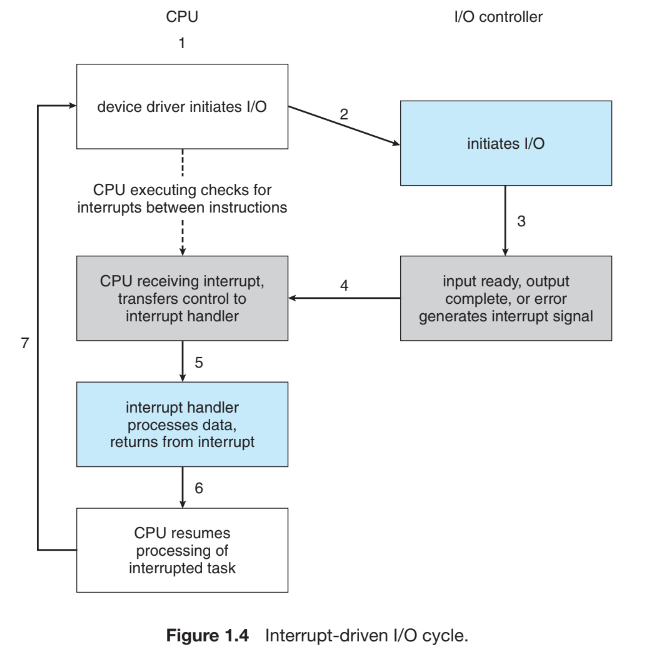
**Interrupts**

A signal sent to the CPU which causes it to stop what it is doing and transfer execution to the fixed location. The signal is usually sent through system bus.

The fixed location contains the starting address of the interrupt service routine for a particular interrupt. After execution and completion of the routine, the CPU resumes the corrected information.



Interrupts need to be handled quickly as they occur frequently. To do this, a pointer table of routines is set up containing the addresses of interrupt service routines. In this way a routine can be called through the table hence providing the necessary speed. The table is located in the starting 100 locations of the memory (low memory). This *interrupt vector* is then indexed through a unique number given with the interrupt request which provides the address of the appropriate interrupt routine.



The CPU has wire called the **interrupt-request-line** which it senses/reads after every execution. If a controller has asserted a signal (interrupt fired) then it jumps to the **interrupt-handler-routine** of the respective interrupt by using the interrupt number as an index to refer the interrupt vector. It then starts execution at the address associated with the index.

The interrupt-handler stores any state it will be changing in the processing/operation, determines the cause of interrupt, resolves it and performs a state restore and returns CPU to the execution state prior to the interrupt.

Modern CPUs have two interrupt lines; *maskable* and *non maskable. Non-*makeable interrupt is reserved for events such as unrecoverable memory, whereas the makeable is used by the CPU to turn interrupts off for a specific period of time where the sequence of instructions being executed should not be interrupted.

A single interrupt vector is not enough as a computer has more devices than addresses in the vector itself. To resolve this problem, *interrupt chaining* is utilized where each element of the vector is a head which points to the list of interrupt handlers. So when an interrupt is raised, the interrupt-handlers are checked one by one until a suitable handler is found.

Another form is interrupt is *trap* also referred as *exception.* It is generally caused by an error and is a software generated interrupt for e.g. Division by zero or accessing an invalid memory address.

**Storage Structure**

CPU must load instructions from the memory before execution. General purpose computers run most of their programs from rewriteable memory called the main memory.

Main memory (also known as **RAM**) is implemented using a semi-conductor technology hence the name **Dynamic RAM** or **DRAM**.

RAM is **volatile** – loses its content when the system is shut down or rebooted. So it can’t be trusted with significant programs like **bootstrap** – a piece of code which initializes the operating system on a computer. So the computer uses **EEPROM** (Electrical-Erasable-Programmable Read Only Memory).

**Firmware** - storage which is infrequently written to and is not volatile.

Most basic unit of computer storage is **bit**. It can contain either 0 or 1. It is used to store data in a computer. The lowest chunk of memory storage in a computer is **byte** which comprises of 8 bits. A **word** is term used for native unit of data for a computer’s architecture. Word is made up of one or more bytes. So if a computer has a 64 bit registers and memory addressing, it has 8 byte word.

* 1 KB = 1024 bytes
* 1 MB = 1024 Kb (1024)^2
* 1 GB = 1024 MB (1024)^3
* 1 TB = 1024 GB (1024)^4
* 1 PB = 1024 TB (1024)^5

**Von-Neumann architecture –** instruction fetched from the memory and stored in the instruction register. Instruction is then decoded, and it may require extra operands to be fetched. After fetching, they are stored in internal registers. Upon execution the result may be stored in register or written back to memory.

**Secondary Storage** – an extension of main memory, secondary storage is used to store large amounts of data permanently.

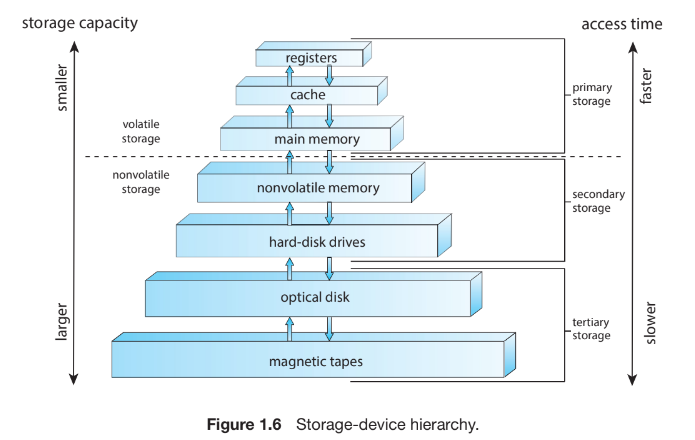
**Why the need for secondary storage?**

2 reasons:

* The main memory does not have enough capacity for holding such large data.
* Main memory is volatile

The most common secondary storage devices include **hard disk drives** (HDDs) and **Non-volatile memory** (NVM). Most system and application programs are stored in the secondary storage. They typically use the secondary storage for writing and retrieving data. Secondary storage is slow as compared to main memory hence it needs to be well managed for optimal system performance.

Another form of memory is **tertiary storage**. It is slower than secondary storage and used for special purposes like long-term archival of data or backup of data.



All storage structures provide the same basic feature; store the datum until it is required at a later stage. As a rule, smaller memory is closer to the CPU. The top four memory units are constructed through semi-conductor. They have semi-conductor based electric circuits.

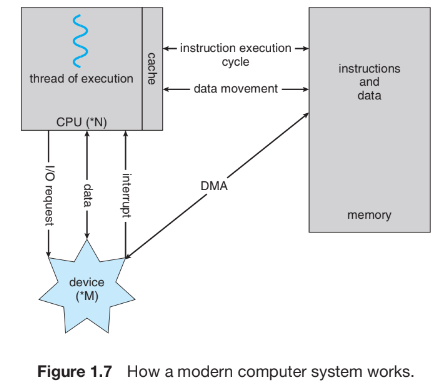
**Caches** can be used to improve performance where there is large disparity in access time or transfer rate exists between two components.

**I/O Structure**

I/O needs to be managed well to improve productivity of a system. Hence a large chunk of OS is dedicated to managing I/O structure.

Previously it was mentioned that devices sent data or receive data to CPU from a single line called bus. But that method is not suitable for large amounts of data transfer (especially involving NVS I/O). To solve this problem **DMA** (direct memory access) is used. DMA allows peripherals to transfer data directly to or from system memory without involving CPU. Only one interrupt is generated when the transfer of data is finished. During this transfer, CPU is free and can perform other tasks.

Some high-end systems use DMA instead of bus architecture. There each component communicates with other components making DMA even more effective.



**Computer-System Architecture**

**Single-processor systems** – Old systems used a processor containing one CPU with single processing core. The core was responsible for executing instructions and registers for storing data locally.

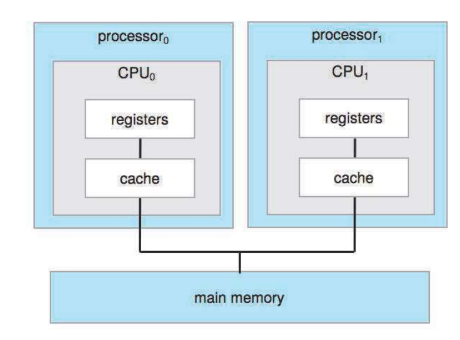
The one main CPU could execute instruction set including instruction from processes. These systems also had specific-purpose processors like device processors. They were capable of executing specific-purpose instruction set (limited instruction set) not the entire process.

Sometimes, these processors would be managed by OS. It sends information about their next task and monitors their status. The processors may implement their own scheduling algorithms and queue from there on.

**Multiprocessor Systems** – Such systems have multiple processors with each with a single core. They share system resources like computer bus, clock, memory or peripherals (Modern technology).

The main advantage of multiprocessor system is the throughput. The speedup from N processor is not N but less than N. Overhead is caused by these multiple processors as they need to work correctly, thus maintaining this correctness causes overhead which lowers the expected gain from multiprocessors.

**Symmetric multiprocessing (SMP)** – is a technique used by modern systems where each peer CPU performs all tasks (ranging from OS to user related).

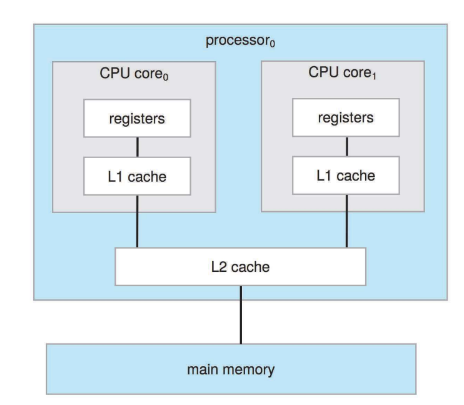


Above diagram is an illustration of a SMP. As you can see each processor has a separate CPU with those CPUs having local registers and caches but sharing the same memory.

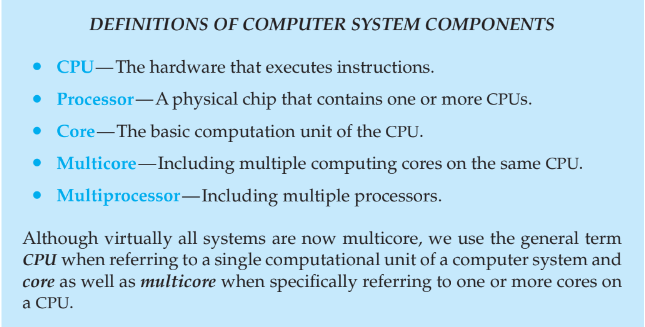
Primary advantage is that the system performance doesn’t deteriorate significantly. N CPUs can run N processes.

But a disadvantage could be that one CPU may be overloaded while the other is idle causing inefficiencies in the system. To resolve this, processors have their own data structures. This will allow dynamic sharing of memory, lowering the workload variance among the processors.

**Multicore systems** – an extension of multiprocessing systems, multicore system has a single chip with multiple CPUs inside of it. It allows faster communication between the two CPUs and less power consumption.

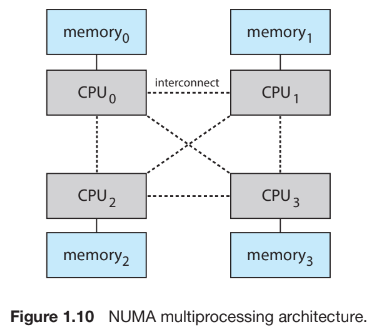


Above picture is an illustration of a multicore system. Each CPU core has its own registers and local caches, while sharing L2 cache.



CPUs in a multiprocessor system have their own local memory with a fast local bus. They are connected through **shared interconnected system** – facilitates coordination and communication between multiple cores/CPUs. Advantage is that there is no competition for system bus amongst the CPUs and the system can be up scaled easily. This approach is also known as **NUMA** (non-uniform memory access).

A potential drawback could be when a CPU has to access remote memory; it’ll take some time thus the performance will be affected.



**Blade servers** – systems who have their own processor, I/O and networking board placed in the same chassis. They have their own operating system and each processor boots independently.

**Clustered Systems** – connection of multiple systems (nodes) joined together. They gather multiple CPUs. Each system is a multicore system. Such systems are considered **loosely coupled**.

In summary, clustered systems share storage and are closely linked via **LAN** (Local Area Network).

Clustering is used to provide **high availability service** – the service will continue even if a system in the cluster fails. Cluster systems run software which allows them to monitor each other. So if the monitored system fails, the monitoring one can take over the responsibility of that system’s operations.

**Graceful degradation** - The ability to provide service proportional to surviving hardware

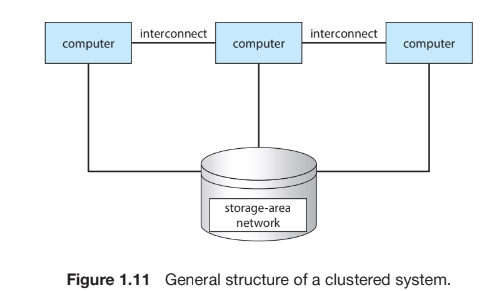
**Fault tolerant** – systems which can operate even after a failure of a component. They typically diagnose the failure and repair it through a provided mechanism.

**Types of Clustering:**

|  |  |
| --- | --- |
| **Asymmetric Clustering** | **Symmetric Clustering** |
| One machine is in **hot standby** mode while the other runs application programs.  **Hot standby** – monitors the active server. If it fails, it acts as the active server. | Two or more machines are running applications and monitoring each other at the same time. This approach is more efficient. However, it requires more than one application to be run. |

**High performance computing** – Clustered systems can deliver performance greater than both single processor and SMPs because they can run applications concurrently on all systems. Applications must be written to take advantage of the cluster.

**Parallelization** – technique which involves breaking a program down into components which run in parallel in individual computer cores or on computer systems in a cluster.



In clustering systems need shared access to access data on a server or database. This sharing needs coordination and communication between systems. The **distribution lock manager (DLM)** manages the sharing aspect of the clustered system.

Clustered systems are improving day by day. Now a cluster system can be spread over many a miles. This improvement is provided by **SAN** – Storage area network which allows many systems to be connected to a pool of storage. Through this a cluster can assign an application program in the SAN to any system connected to it. If the host fails, a different system can take over the responsibility. In a database cluster, dozens of host can have the same database increasing readability and performance.

**Operating System operations**

Bootstrap locates and loads the kernel into memory.

**System daemons** – Some system applications outside of the kernel loaded into memory during boot time. They keep on running until the kernel is running. In Linux the first program is systemsd which loads other daemons. Once they’re loaded the system is fully booted.

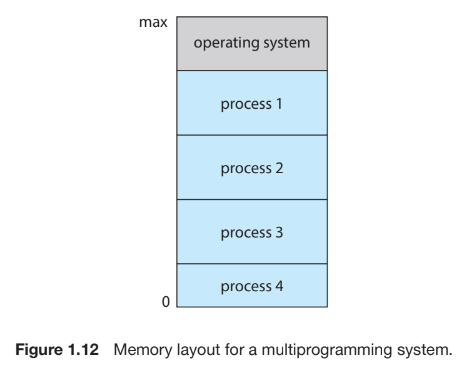
A trap or exception (type of interrupt) can also be triggered by a request from a user which can be performed by **system call**.

**Multiprogramming and multitasking**

**Multiprogramming** – Ability to run multiple programs at once. It is one of the most important aspects of operating system. It increases CPU utilization by organizing programs such that the CPU always has one program to execute.

Each program in a multiprogrammed system is a **process.** Main goal is to keep the CPU busy. The operating system does this as follows:

* Keeps several processes in memory simultaneously.
* CPU will start executing any one of the processes. When that processes needs I/O operation the CPU starts executing another process.
* The process keeps on repeating. In this way the process gets the CPU back after the I/O operation has been completed thus never leaving the CPU idle.



**Multitasking** – Logical extension of multiprogrammed systems/multiprogramming. The CPU executes the processes by switching between then frequently. This switching is responsible for a better response time to the user.

**Response time –** Measure of how long an operating system takes to respond to a user’s input or request.

If a process requires user interaction (I/O dependent) than there is a possibility for a delay in user input. During this time the operating system switches the CPU to a different process instead of letting it remain in idle state.

To maintain a reasonable response time the operating system uses **virtual memory** – technique which allows execution of programs not in the memory. It allows user to run programs larger than **physical memory** – actual memory of the system.

**CPU scheduling** – Selecting which process will run next when a bunch of them a ready for execution.

**Dual-mode and Multimode operations**

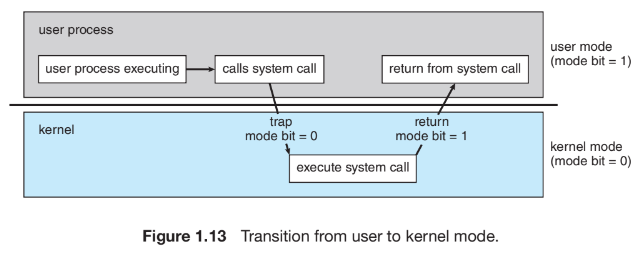
Two separate modes of execution:

* User – when a system is executing program on behalf of user application
* Kernel – when system is executing program on behalf of operating system

To distinguish between them, a **mode bit** is added to the hardware.

**Mode bit** – helps in distinguishing in which mode the system is in currently (0 = kernel, 1 = User).

If a user application request service from the kernel through a system call then a transition is made from user to kernel mode.



During boot-up the hardware starts in kernel mode. User starts user applications in user mode. Trap causes a switch from user mode to kernel mode. Before going to the kernel mode the mode bit is set to 0.

**Privileged Instructions** – Machine instructions which the user cannot access are called privileged instructions. Hardware allows these instructions to be executed in kernel mode. If an attempt is made to execute it in user mode, the hardware traps it to the OS.

Processors extend the concept of modes via *rings*. Intel has 4 rings with ring 0 being the kernel and ring 3 being the user. CPUs which support virtualization have a separate mode managed by **Virtual Machine Manager (VMM)**. It has more privileges than the user but lesser than kernel.

**System Call** – a mechanism which allows user-level processes to request services from the kernel

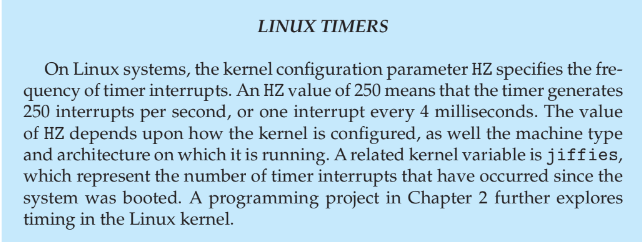
System call usually takes a trap to a specific interrupt vector. This trap can be executed by generic trap instruction. Some systems invoke system calls through “syscall” instruction. The hardware treats system calls as software interrupts. When invoked, the mode switches to kernel which then performs the steps required to resolve interrupt and returns the control back to the previous mode after executing the system call.

The trap transfers control through the interrupt vector to the operating system, just as an interrupt does. When a program error occurs, the operating system must terminate the program abnormally. This situation is handled by the same code as a user-requested abnormal termination.

**Timer**

A timer can be set to interrupt a computer after a specified period. A *variable timer* is implemented by fixed-rate clock and a counter.

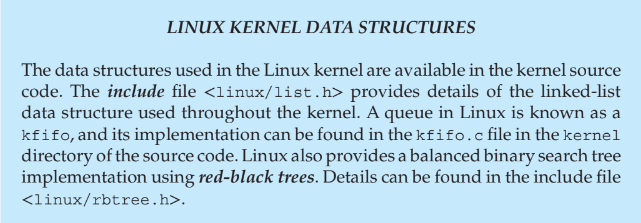
Basic use of a timer is to snap the CPU out from an infinite loop or an infinite system call.



Before handing control back to user mode, OS sets a timer to interrupt. If it interrupts, the control is transferred to OS and the interrupt is dealt with.

**Kernel Data Structures**

* **Linked lists** – items are linked to one another. Each node has a next pointer, or in cases of doubly-linked a previous pointer as well to the preceding node, to the following node. They are used directly by kernel algorithms.
* **Stack** – A data structures based on **Last in First out (LIFO)** principle. It means element entered at the last/end will be removed first.
* **Queue** – A data structure based on **First in first out (FIFO)** principle. Element added at the beginning of the queue will be removed first from the queue. Processes waiting to be run are organized as a queue.
* **Tree** - used to represent data hierarchically. A general tree may have n children. Linked through parent-child relationships
* **Binary Tree** – A binary tree has only 2 children (left and right). For better results and faster retrieval of data, Linux uses balanced binary search tree (red and black trees) where left < root < right and the tree is balanced ensuring faster data retrieval. They are typically used in CPU scheduling algorithms.
* **Maps using hash values** – Hash values generated by hash functions are unique values which act as an index on a table for quick data retrieval. To resolve collisions we can use linked lists (separate chaining) at the table locations. Hash maps are key value pairs established by the use of hash values. Once established, data can be retrieved by a key value.
* **Bitmaps** – string of n binary digits to represent status of n items. The value at the ith position is associated with the ith resource. Used in disk drives for checking the status of a disk block.



**Computing environment**

**Traditional computing:**

Companies establish **portals** – allow access to World Wide Web, to their internal servers.

**Thin clients** – systems which understand web-based computing (also called terminals) are used by companies where security and maintenance is required.

Nowadays, many home systems use the internet. Internet access, even if expensive, is relatively inexpensive in many areas across the globe. But it is not always safe so a **firewall** is used as a safety measure when browsing or surfing the internet.

**Firewall** – Limit communications between devices on a network.

|  |  |
| --- | --- |
| **Batch execution** | **Interactive execution** |
| Systems used to execute instructions in bulks with predetermined inputs from files. | Systems would wait for user input/ask for user input before processing. |

For better optimization users shared their time on these systems. These time-sharing systems used a timer and scheduling algorithms to cycle processes rapidly through the CPU, giving each user a share of the resources.

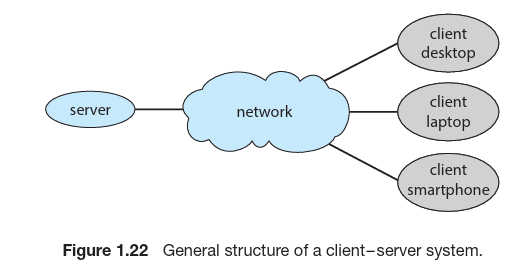
**Mobile Computing**

Computing on handheld smartphones and tablet computers is mobile computing. Mobile systems gave up screen size, memory capacity, and overall functionality in return for handheld mobile access to services such as e-mail and web browsing.

**Augmented reality** – Overlay of information on a current environment at display.

**Client-server computing**

**Client-server system** – A network architecture where computing tasks are divided between the client and the server is a client server system. A client requests data from the server which then responds by fetching the data.



Server systems are broadly categorized as:

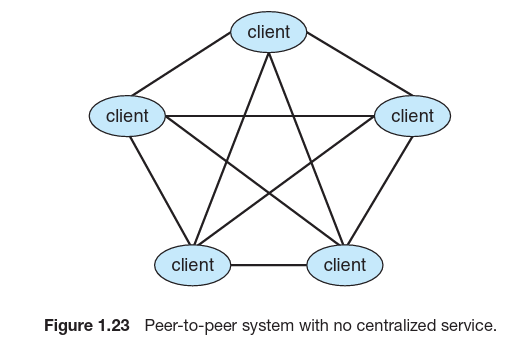
|  |  |
| --- | --- |
| **Computer-server** | **File-server** |
| Provides an interface through which the client can send request to the server which then returns the result after processing the request. | Provides a file-system interface through which client can create, update and delete files. |

**Peer-to-peer computing**

In this model, clients and servers are not distinguished from one another. Instead, all nodes within the system are considered peers, and each may act as either a client or a server, depending on whether it is requesting or providing a service.

Service determination can be done in the following 2 ways:

* When a node joins a network, it registers its service with a centralized lookup service on the network. Any node desiring a special service contacts this centralized lookup service to determine which node provides the service. The remainder of the communication takes place between the client and the service provider.
* Instead, a peer acting as a client must discover what node provides a desired service by broadcasting a request for the service to all other nodes in the network. The node (or nodes) providing that service responds to the peer making the request. To support this approach, a discovery protocol must be provided that allows peers to discover services provided by other peers in the network.



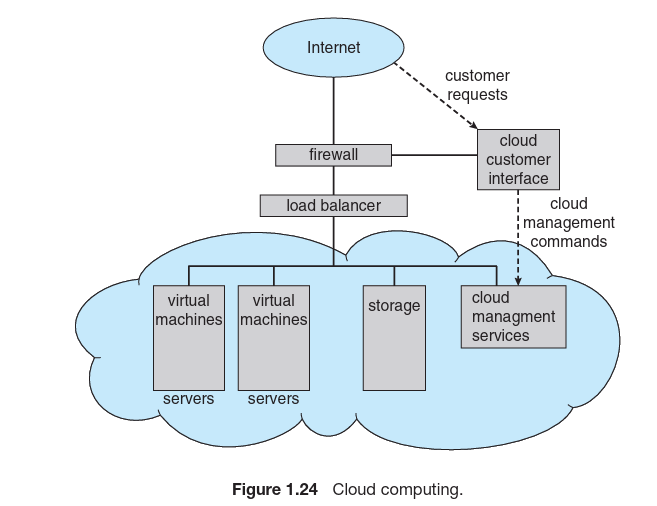
**Voice over IP (VoIP)** – It allows users to send text, images or make voice/video calls over the internet.

**Cloud Computing**

Cloud computing is a type of computing that delivers computing, storage, and even applications as a service across a network. In some ways, it’s a logical extension of virtualization, because it uses virtualization as a base for its functionality.

**Types**

* Public cloud – a cloud available via the Internet to anyone willing to pay for the services
* Private cloud – a cloud ran by a company for its own use
* Hybrid – includes both public and private cloud components
* Software as a service (SaaS) – one or more applications available via the internet
* Infrastructure as a service (IaaS) - servers or storage available over the Internet



VMMs manage the virtual machines in which the user processes run. At a higher level, them-selves are managed by cloud management tools

**Real-time OS** – It is used when rigid requirements have been placed on operation of processor or workflow of data.