notes

Alexander

October 11, 2024

Open Systems Interconnection model 1984

The OSI model is a reference model from the ISO that 'provides a common basis for the coordination of standards development for the purpose of systems interconnection.' In the OSI reference model, the communications between systems are split into seven different abstraction layers: Physical, Data, Link, Network, Transport, Session, Presentation, and Application. The model partitions the flow of data in a communication system into seven abstraction layers to describe networked communication from the physical implementation of transmitting bits across a communications medium to the highest-level representation of data of a distributed application. Each intermediate layer serves a class of functionality to the layer above it and is served by the layer below it. Classes of functionality are implemented in software development using established communication protocols. Each layer in the OSI model has well-defined functions, and the methods of each layer communicate and interact with those of the layers immediately above and below as approriate. The Internet protocol suite as defined in RFC 1122 and RFC 1123 is a model of networking developed contemporarily to the OSI model, and was funded primarily by the U.S Department of Defense. It was the foundation for the development of the Internet. It assumed the presence of generic physical links and focused primarily on the software layers of communication, with a similar but much less rigorous structure than the OSI model. In comparison, several networking models have sought to create an intellectual framework for clarifying networking concepts and activities, but none have been as successful as the OSI reference model in becoming the standard model for discussing and teaching networking in the field of IT. The model allows tranparent communication through equivalent exchange of PDUs between two parties, through what is known as peer-to-peer networking. As a result, the OSI reference model has not only become an important piece amound professionals and non-professionals alike, but also in all networking between one or many parties, due in large part to its commonly accepted user-friendly framework.

Communication protocols enable an entity in one host to interact with a corresponding entity at the same layer in another host. Service definitions, like the OSI model, abstractly describe the functionality provided to a layer N by a layer N-1, where N is one of the seven layers of protocols operating in the local host. At each level N, two entities at the communicating devices (layer N peers) exchange PDUs by means of a layer N protocol. Each PDU contains a payload, called the service data unit (SDU), along with protocol-related headers or footers.

Data processing by two communicating OSI-compatible devices proceeds as follows:

1 The data to be transmitted is composed at the topmost layer of the transmitting device (layer N) into a PDU.

2 The PDU is passed to layer N-1, where it is known as the SDU.

3 At layer N-1 the SDU is concatenated with a header, a footer, or both, producing a layer N-1 PDU. It is then passed to layer N-2.

4 The process continues until reaching the lowermost level, from which the data is transmitted to the receiving device.

5 At the receiving device the data is passed from the lowest layers a series of SDUs while being successively stripped from each layer's header or footer until reaching the topmost layer, where the last of the data is consumed.

The host layers (4-7) refer to the layers that reside on the end systems (hosts) in the network.
The media layers (1-3) refer to the lower layers of the OSI model that deal with the physical transmission of data over a medium.

7 Application (data) High-level protocols such as for resource sharing or remote file access, e.g. HTTP.

The application layer is the layer of the OSI model that is closest to the end user, which means both the OSI application layer and the user interact directly with a software application that implements a component of communication between the client and server, such as File Explorer and Microsoft Word. Such application programs fall outside the scope of the OSI model unless they are directly integrated into the application layer through the functions of communication, as is the case with applications such as web browsers and email programs. Other examples of software are Microsoft Network Software for File and Printer Sharing and Unix/Linux Network File System Client for access to shared file resources. Application-layer functions typically include file sharing, message handling, and database access, through the most common protocols at the application layer, known as HTTP, FTP, SMB/CIFS, TFTP, and SMTP. When identifying communication partners, the application layer determines the identity and availability of communication partners for an application with data to transmit. The most important distinction in the application layer is the distinction between the application-entity and the application. For example, a reservation website might have two application-entities: one using HTTP to communicate with its users, and one for a remote database protocol to record reservations. Neither of these protocols have anything to do with reservations. That logic is in the application itself. The application layer has no means to determine the availability of resources in the network.

6 Presentation (data) Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption

The presentation layer establishes data formatting and data translation into a format specified by the application layer during the encapsulation of outgoing messages while being passed down the protocol stack, and possibly reversed during the deencapsulation of incoming messages when being passed up the protocol stack. For this very reason, outgoing messages during encapsulation are converted into a format specified by the application layer, while the conversion for incoming messages during deencapsulation are reversed. The presentation layer handles protocol conversion, data encryption, data decryption, data compression, data decompression, incompatibility of data representation between operating systems, and graphic commands. The presentation layer transforms data into the form that the application layer accepts, to be sent across a network. Since the presentation layer converts data and graphics into a display format for the application layer, the presentation layer is sometimes called the syntax layer. For this reason, the presentation layer negotiates the transfer of syntax structure through the Basic Encoding Rules of Abstract Syntax Notation One (ASN.1), with capabilities such as converting an EBCDIC-coded text file to an ASCII-coded file, or serialization of objects and other data structures from and to XML.

5 Session (data) Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes

The session layer creates the setup, controls the connections, and ends the teardown, between two or more computers, which is called a ”session”. Common functions of the session layer include user logon (establishment) and user logoff (termination) functions. Including this matter, authentication methods are also built into most client software, such as FTP Client and NFS Client for Microsoft Networks. Therefore, the session layer establishes, manages and terminates the connections between the local and remote application. The session layer also provides for full-duplex, half-duplex, or simplex operation,[citation needed] and establishes procedures for checkpointing, suspending, restarting, and terminating a session between two related streams of data, such as an audio and a video stream in a web-conferencing application. Therefore, the session layer is commonly implemented explicitly in application environments that use remote procedure calls.

4 Transport (segement, datagram) Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing

The transport layer provides the functional and procedural means of transferring variable-length data sequences from a source host to a destination host from one application to another across a network, while maintaining the quality-of-service functions. Transport protocols may be connection-oriented or connectionless. This

may require breaking large protocol data units or long data streams into smaller chunks called "segments", since the network layer imposes a maximum packet size called the maximum transmission unit (MTU), which depends on the maximum packet size imposed by all data link layers on the network path between the two hosts. The amount of data in a data segment must be small enough to allow for a network-layer header and a transport-layer header. For example, for data being transferred across Ethernet, the MTU is 1500 bytes, the minimum size of a TCP header is 20 bytes, and the minimum size of an IPv4 header is 20 bytes, so the maximum segment size is 1500-(20+20) bytes, or 1460 bytes. The process of dividing data into segments is called segmentation; it is an optional function of the transport layer. Some connection-oriented transport protocols, such as TCP and the OSI connection-oriented transport protocol (COTP), perform segmentation and reassembly of segments on the receiving side; connectionless transport protocols, such as UDP and the OSI connectionless transport protocol (CLTP), usually do not. The transport layer also controls the reliability of a given link between a source and destination host through flow control, error control, and acknowledgments of sequence and existence. Some protocols are state- and connection-oriented. This means that the transport layer can keep track of the segments and retransmit those that fail delivery through the acknowledgment hand-shake system. The transport layer will also provide the acknowledgement of the successful data transmission and sends the next data if no errors occurred. Reliability, however, is not a strict requirement within the transport layer. Protocols like UDP, for example, are used in applications that are willing to accept some packet loss, reordering, errors or duplication. Streaming media, real-time multiplayer games and voice over IP (VoIP) are examples of applications in which loss of packets is not usually a fatal problem. The OSI connection-oriented transport protocol defines five classes of connection-mode transport protocols, ranging from class 0 (which is also known as TP0 and provides the fewest features) to class 4 (TP4, designed for less reliable networks, similar to the Internet). Class 0 contains no error recovery and was designed for use on network layers that provide error-free connections. Class 4 is closest to TCP, although TCP contains functions, such as the graceful close, which OSI assigns to the session layer. Also, all OSI TP connection-mode protocol classes provide expedited data and preservation of record boundaries. An easy way to visualize the transport layer is to compare it with a post office, which deals with the dispatch and classification of mail and parcels sent. A post office inspects only the outer envelope of mail to determine its delivery. Higher layers may have the equivalent of double envelopes, such as cryptographic presentation services that can be read by the addressee only. Roughly speaking, tunnelling protocols operate at the transport layer, such as carrying non-IP protocols such as IBM's SNA or Novell's IPX over an IP network, or end-to-end encryption with IPsec. While Generic Routing Encapsulation (GRE) might seem to be a network-layer protocol, if the encapsulation of the payload takes place only at the endpoint, GRE becomes closer to a transport protocol that uses IP headers but contains complete Layer 2 frames or Layer 3 packets to deliver to the endpoint. L2TP carries PPP frames inside transport segments. Although not developed under the OSI Reference Model and not strictly conforming to the OSI definition of the transport layer, the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) of the Internet Protocol Suite are commonly categorized as layer 4 protocols within OSI. Transport Layer Security (TLS) does not strictly fit inside the model either. It contains characteristics of the transport and presentation layers.

Table 1: Detailed characteristics of TP0–4 classes are shown in the following table:

Feature name	TP0	TP1	TP2	TP3	TP4
Connection-oriented network	Yes	Yes	Yes	Yes	Yes
Connectionless network	No	No	No	No	Yes
Concatenation and separation	No	Yes	Yes	Yes	Yes
Segmentation and reassembly	Yes	Yes	Yes	Yes	Yes
Error recovery	No	Yes	Yes	Yes	Yes
Reinitiate connection (unack PDUs)	No	Yes	No	Yes	No
Multiplexing / demultiplexing over single virtual circuit	No	No	Yes	Yes	Yes
Explicit flow control	No	No	Yes	Yes	Yes
Retransmission on timeout	No	No	No	No	Yes
Reliable transport service	No	Yes	No	Yes	Yes

3 Network (packet) Structuring and managing a multi-node network, including addressing, routing and traffic control

The network layer provides the functional and procedural means of transferring packets from one node to another connected in "different networks". A network is a medium to which many nodes can be connected, on which every node has an address and which permits nodes connected to it to transfer messages to other nodes connected to it by merely providing the content of a message and the address of the destination node and letting the network find the way to deliver the message to the destination node, possibly routing it through intermediate nodes. If the message is too large to be transmitted from one node to another on the data link layer between those nodes, the network may implement message delivery by splitting the message into several fragments at one node, sending the fragments independently, and reassembling the fragments at another node. It may, but does not need to, report delivery errors. Message delivery at the network layer is not necessarily guaranteed to be reliable; a network layer protocol may provide reliable message delivery, but it does not need to do so. A number of layer-management protocols, a function defined in the management annex, ISO 7498/4, belong to the network layer. These include routing protocols, multicast group management, network-layer information and error, and network-layer address assignment. It is the function of the payload that makes these belong to the network layer, not the protocol that carries them.

2 Data link (frame) Transmission of data frames between two nodes connected by a physical layer

The data link layer provides node-to-node data transfer—a link between two directly connected nodes. It detects and possibly corrects errors that may occur in the physical layer. It defines the protocol to establish and terminate a connection between two physically connected devices. It also defines the protocol for flow control between them.

IEEE 802 divides the data link layer into two sublayers:

- Medium access control (MAC) layer – responsible for controlling how devices in a network gain access to a medium and permission to transmit data.
- Logical link control (LLC) layer – responsible for identifying and encapsulating network layer protocols, and controls error checking and frame synchronization.

The MAC and LLC layers of IEEE 802 networks such as 802.3 Ethernet, 802.11 Wi-Fi, and 802.15.4 Zigbee operate at the data link layer. The Point-to-Point Protocol (PPP) is a data link layer protocol that can operate over several different physical layers, such as synchronous and asynchronous serial lines. The ITU-T G.hn standard, which provides high-speed local area networking over existing wires (power lines, phone lines and coaxial cables), includes a complete data link layer that provides both error correction and flow control by means of a selective-repeat sliding-window protocol. Security, specifically (authenticated) encryption, at this layer can be applied with MACsec.

1 Physical (bit, symbol) Transmission and reception of raw bit streams over a physical medium

The physical layer is responsible for the transmission and reception of unstructured raw data between a device, such as a network interface controller, Ethernet hub, or network switch, and a physical transmission medium. It converts the digital bits into electrical, radio, or optical signals. Layer specifications define characteristics such as voltage levels, the timing of voltage changes, physical data rates, maximum transmission distances, modulation scheme, channel access method and physical connectors. This includes the layout of pins, voltages, line impedance, cable specifications, signal timing and frequency for wireless devices. Bit rate control is done at the physical layer and may define transmission mode as simplex, half duplex, and full duplex. The components of a physical layer can be described in terms of the network topology. Physical layer specifications are included in the specifications for the ubiquitous Bluetooth, Ethernet, and USB standards. An example of a less well-known physical layer specification would be for the CAN standard.

The physical layer also specifies how encoding occurs over a physical signal, such as electrical voltage or a light pulse. For example, a 1 bit might be represented on a copper wire by the transition from a 0-volt to a 5-volt signal, whereas a 0 bit might be represented by the transition from a 5-volt to a 0-volt signal. As a result, common problems occurring at the physical layer are often related to the incorrect media termination, EMI or noise scrambling, and NICs and hubs that are misconfigured or do not work correctly.

Cross-layer functions

Cross-layer functions are services that are not tied to a given layer, but may affect more than one layer. Some orthogonal aspects, such as management and security, involve all of the layers. These services are aimed at improving the CIA triad-confidentiality, integrity, and availability-of the transmitted data. Cross-layer functions are the norm, in practice, because the availability of a communication service is determined by the interaction between network design and network management protocols.

Specific examples of cross-layer functions include the following:

- Security service (telecommunication) as defined by ITU-T X.800 recommendation.
- Management functions, i.e. functions that permit to configure, instantiate, monitor, terminate the communications of two or more entities: there is a specific application-layer protocol, Common Management Information Protocol (CMIP) and its corresponding service, Common Management Information Service (CMIS), they need to interact with every layer in order to deal with their instances.
- Multiprotocol Label Switching (MPLS), ATM, and X.25 are 3a protocols. OSI subdivides the Network Layer into three sublayers: 3a) Subnetwork Access, 3b) Subnetwork Dependent Convergence and 3c) Subnetwork Independent Convergence. It was designed to provide a unified data-carrying service for both circuit-based clients and packet-switching clients which provide a datagram-based service model. It can be used to carry many different kinds of traffic, including IP packets, as well as native ATM, SONET, and Ethernet frames. Sometimes one sees reference to a Layer 2.5.
- Cross MAC and PHY Scheduling is essential in wireless networks because of the time-varying nature of wireless channels. By scheduling packet transmission only in favourable channel conditions, which requires the MAC layer to obtain channel state information from the PHY layer, network throughput can be significantly improved and energy waste can be avoided.

Programming interfaces

Neither the OSI Reference Model, nor any OSI protocol specifications, outline any programming interfaces, other than deliberately abstract service descriptions. Protocol specifications define a methodology for communication between peers, but the software interfaces are implementation-specific. For example, the Network Driver Interface Specification (NDIS) and Open Data-Link Interface (ODI) are interfaces between the media (layer 2) and the network protocol (layer 3).

Comparison with TCP/IP model

The design of protocols in the TCP/IP model of the Internet does not concern itself with strict hierarchical encapsulation and layering. RFC 3439 contains a section entitled ”Layering considered harmful”. TCP/IP does recognize four broad layers of functionality which are derived from the operating scope of their contained protocols: the scope of the software application; the host-to-host transport path; the internetworking range; and the scope of the direct links to other nodes on the local network

Despite using a different concept for layering than the OSI model, these layers are often compared with the OSI layering scheme in the following manner:

- The Internet application layer maps to the OSI application layer, presentation layer, and most of the session layer.
- The TCP/IP transport layer maps to the graceful close function of the OSI session layer as well as the OSI transport layer.
- The internet layer performs functions as those in a subset of the OSI network layer.
- The link layer corresponds to the OSI data link layer and may include similar functions as the physical layer, as well as some protocols of the OSI's network layer.

These comparisons are based on the original seven-layer protocol model as defined in ISO 7498, rather than refinements in the internal organization of the network layer. The OSI protocol suite that was specified as part of the OSI project was considered by many as too complicated and inefficient, and to a large extent unimplementable. Taking the "forklift upgrade" approach to networking, it specified eliminating all existing networking protocols and replacing them at all layers of the stack. This made implementation difficult and was resisted by many vendors and users with significant investments in other network technologies. In addition, the protocols included so many optional features that many vendors' implementations were not interoperable. Although the OSI model is often still referenced, the Internet protocol suite has become the standard for networking. TCP/IP's pragmatic approach to computer networking and to independent implementations of simplified protocols made it a practical methodology. Some protocols and specifications in the OSI stack remain in use, one example being IS-IS, which was specified for OSI as ISO/IEC 10589:2002 and adapted for Internet use with TCP/IP as RFC 1142.

TCP

TCP, or Transmission Control Protocol, is one of the main protocols in the Internet Protocol Suite. It ensures reliable, ordered, and error-checked delivery of data between applications running on devices connected to a network.

1 Connection Establishment

- SYN: The client sends a SYN (synchronize) packet to the server to initiate a connection.
- SYN-ACK: The server responds with a SYN-ACK (synchronize-acknowledge) packet to acknowledge the request and also to synchronize its own sequence number.
- ACK: The client sends an ACK (acknowledge) packet back to the server, confirming that the connection is established.

2 Data Transmission

- Segmentation: TCP divides the data into segments. Each segment is assigned a sequence number to ensure the correct order.
- Acknowledgments: The receiver sends back an acknowledgment (ACK) for each segment received. If a segment is lost or corrupted, the sender will retransmit it.
- Flow Control: TCP uses a mechanism called flow control (via a sliding window) to ensure that the sender does not overwhelm the receiver with too much data at once.
- Congestion Control: TCP also implements congestion control algorithms (like TCP Reno or TCP Cubic) to manage data transmission based on network conditions, avoiding congestion and packet loss.

3 Connection Termination

- FIN: One side sends a FIN (finish) packet to indicate it wants to close the connection.
- ACK: The other side acknowledges the FIN with an ACK.
- FIN: The other side then sends its own FIN.
- ACK: Finally, the first side acknowledges this FIN, and the connection is closed.

4 Key Features of TCP

- Reliability: Ensures data is delivered accurately and in order.
- Flow Control: Prevents overwhelming the receiver.
- Congestion Control: Adapts to network conditions to avoid congestion.
- Connection-oriented: Requires establishing a connection before data transfer.

UDP

UDP, or User Datagram Protocol, is a communication protocol that operates on top of the Internet Protocol (IP). Unlike TCP, UDP is connectionless and does not guarantee the reliability or order of data packets.

1 Connectionless Communication

- No Handshake: UDP does not establish a connection before sending data. There's no initial handshake like in TCP. This makes UDP faster but less reliable.

2 Data Transmission

- Datagrams: Data is transmitted in packets called datagrams. Each datagram is treated independently, meaning that each packet can take a different path to the destination.
- No Guarantees: UDP does not guarantee that:
  - Packets will arrive at the destination.
  - Packets will arrive in the order they were sent.
  - Duplicates will be eliminated.
  - Lost packets will be retransmitted.

3 Efficiency and Overhead

- Low Overhead: UDP has a smaller header size (8 bytes) compared to TCP (20 bytes), making it more efficient for certain applications where speed is crucial.
- No Flow Control: UDP does not implement flow control or congestion control, allowing for faster transmission but potentially leading to packet loss in congested networks.

4 Use Cases

- Streaming Media: Video and audio streaming, where occasional packet loss is acceptable.
- Online Gaming: Real-time games that require fast updates and can tolerate some data loss.
- VoIP: Voice over Internet Protocol, where low latency is critical.
- DNS Queries: Domain Name System lookups, which are typically brief and can be retried if necessary.

Sockets

1 Types of Sockets

- Stream Sockets (TCP): These sockets provide reliable, connection-oriented communication. They use TCP (Transmission Control Protocol) to ensure that data is transmitted in order and without errors.
- Datagram Sockets (UDP): These sockets provide connectionless communication. They use UDP (User Datagram Protocol), which is faster but does not guarantee delivery, order, or error correction.

2 Components of a Socket

- IP Address: The unique identifier for a device on a network.
- Port Number: A numerical label that identifies a specific process or service on that device.

3 Socket Programming

1 Create a Socket: Use a system call to create a socket.
2 Bind: For servers, bind the socket to a specific IP address and port number.
3 Listen: For TCP servers, listen for incoming connections.
4 Accept/Connect: Servers accept incoming connections, while clients connect to the server.
5 Send/Receive Data: Use send and receive operations to exchange data.
6 Close: Once communication is done, close the socket to free up resources.

4 Use Cases

- Web servers and browsers (HTTP/HTTPS)
- Email clients and servers (SMTP, IMAP)
- File transfer applications (FTP)
- Real-time communication (VoIP, chat applications)

1. Operating Systems

Computer Systems Overview

”An OS exploits the hardware resources of one or more processors to provide a set of services to system users. The OS also manages seconday memory and I/O devices on behalf of its users. Accordingly, it is important to have some understanding of the underlying computer system hardware before we begin our examination of operating systems.”

1 Processor: Controls the operation of the computer and performs its data processing functions. When there is only one processor, it is often referred to as the CPU

   - PC: Typically, the PC holds the address of the next instruction to be fetched. Unless instructed otherwise, the processor always increments the PC after each instruction fetch so that it will fetch the next instruction in sequence (i.e, the instruction located at the next higher memory address).
   - IR: The fetched instruction is loaded into the IR. The instruction contains bits that specify the action the processor is to take. The processor interprets the instruction and performs the required action. In general, these actions fall into four categories:

     Processor-memory: Data may be transferred from processor to memory or from memory to processor.

     Processor-I/O: Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module

     Data processing: The processor may perform some arithmetic or logic operating on data

     Control: An instruction may specify that the sequence of execution be altered. For example, the processor may fetch an instruction from location 149, which specifies that the next instruction be from location 182. The processor sets the program counter to 182. Thus, on the next fetch stage, the instruction will be fetched from location 182 rather than 150.

     An instruction’s execution may involve a combination of the these actions.
   - Execution unit
   - MAR: specifies the address in memory for the next read or write
   - MBR: contains the data to be written into memory or which receives the data read from memory
   - I/O AR: specifies a particular I/O device
   - I/O BR: is used for the exchange of data between an I/O module and the processor

2 I/O module: Move data between the computer and its external environment. The external environment consists of a variety of devices, including secondary memory devices (e.g., disks), communications equipment, and terminals.

   - Buffers

3 Main memory: Stores data and programs. This memory is typically volatile; that is, when the computer is shut down, the contents of the memory are lost. In contrast, the contents of disk memory are retained even when the computer system is shut down. Main memory is also referred to as real memory or primary memory

   - Instruction
   - Data

4 System Bus: Provides for communication amoung processors, main memory, and I/O modules

In its simplest form, instruction processing consists of two steps: The processor reads (fetches) instructions from memroy one at a time and executes each instruction. Program execution consists of repeating the process of instruction fetch and instruction execution. Instruction execution may involve several operations and depends on the nature of the instruction.

Now consider a hypothetical machine...The processor contains a single data register, called the accumulator (AC). Both instructions and data are 16 bits long, and memory is organized as a sequence of 16-bit words. the instruction format provides 4 bits for the opcode, allowing as many as $2^4 = 16$ different opcodes (represented by a single hexadecmal digit). The opcode defines the operation the processor is to perform. With the remaining 12 bits of the instruction format, up to $2^12 = 4096$ (4K) words of memory can be directly addressed.

(a) Instruction format

Opcode (0-3) Magnitude (4-15)

(b) Integer format

S (0) Address (1-15)

(c) Internal CPU registers

Program counter (PC) = Address of instruction

Instruction register (IR) = Instruction being executed

Accumulator (AC) = Temporary storage

(d) Partial list of opcodes

0001 = Load AC from memory

0010 = Store AC to memory

0101 = Add to AC from memory

1 The PC contains 300, the address of the first instruction. This instruction (the value 1940 in hex) is loaded into the IR and the PC is incremented. Note that this process involves the use of a MAR and a MBR. For simplicity,these intermediate registers are not shown

2 The first 4 bits (first hex digit) in the IR indicate that the AC is to be loaded from memory. The remaining 12 (three hex digits) specify the address, which is 940.

3 The next instruction (5941) is fetched from location 301 and the PC is incremented.

4 The old contents of the AC and the contents of location 941 are added and the result is stored in the AC.

5 The next instruction (2941) is fetched from location 302 and the PC is incremented

6 The contents of the AC are stored in location 941.

NOTE: three instructions cycles (fetch stage and an execute stage) are needed here to add the contents of 940 to 941. With a more complex set ofinstructions, fewer instruction cycles would be needed. Most modern processors include instructions that contain more than one address Thus the execution stage for a particular instruction may involve more than one reference to memory. Also, instead of memory references, an instruction may specify an I/O operation.

Classes of Interrupts:

- Program: Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction,and reference outside a user's allowed memory space.
- Timer: Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis
- I/O: Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.

- Hardware failure: Generated by a failure, such as power failure or memory parity error.

Interrupts are provided primarily as a way to improve processor utilization. For example, most I/O devices are much slower than the processor. Suppose that the processor is transferring data to a printer. After each write operation, the processor must pause and remain idle until the printer catches up. The length of this pause may be on the order of many thousands or even millions of instruction cycles. Clearly, this is a very wastful use of the processor.

NOTE: The term "hertz" is a unit of frequency that measures cycles per second. When we say something operates at 1 Hz, it means it completes one cycle in one second.

To give a specific example, consider a PC that operates at 1 GHz, which would allow roughly 10^9 instructions per second. A typical harddisk has a rotational speed of 7200 revolutions per minute for a half-track rotation time of 4 ms, which is 4 million times slower than the processor.

$\frac{7200}{60} = 120$ revolutions per second

$\frac{1}{120} \approx 0.00833$ seconds = 8.33 milliseconds

$\frac{8.33ms}{2} \approx 4.165$ ms

Instruction Time = 1 ns = 0.000001 ms (since 1 ns = 10^-6 ms)

Ratio = $\frac{4}{0.000001} = 4,0000,000$

Why half-track?: When accessing data on a hard disk, the read/write head must move to the appropriate track. In many cases, it's assumed that the average position is halfway around the disk, hence the term "half-track" rotation time. This accounts for the average time needed to position the head over the desired track.

How does a hardisk work?
A hard disk consists of one or more spinning disks called platters, coated with a magnetic material that stores data. Each platter has a read/write head that moves over its surface to access data. The heads are mounted on an actuator arm. Each platter is divided into concentric circles called tracks. Each track is further divided into smaller units called sectors, typically containing 512 bytes or 4K bytes of data. Data is stored magnetically on the surface of the platters. Each bit of data is represented by tiny magnetic fields that can be aligned in one of two directions, corresponding to binary 0s and 1s. When the hard disk is powered on, the platters spin at high speeds (e.g., 7200 RPM). The actuator arm moves the read/write head to the correct track on the platter where the data is stored. As the platter spins, the read head detects the magnetic fields of the data on the track. The read head converts these magnetic fields into electrical signals. The electrical signals are then sent to the computer's controller, which interprets them as the data stored in the corresponding sectors. Just like reading, the actuator arm positions the write head over the correct track. To write data, the write head generates a magnetic field that alters the magnetic alignment of the material on the platter surface. This changes the magnetic orientation to represent the new data (0s and 1s). Once the data is written, the head can move to the next position, and the process continues for the remaining data.

2. Operating Systems Overview

3. Process Description

1. A process is an instance of a program in execution, including the program code, its current activity (represented by the program counter), and the contents of its registers
2. The operating system maintains a data structure called the Process Control Block (PCB), which contains essential information: PID, process state, program counter, CPU registers, memory management information (e.g., page tables), I/O staus information, accounting information (e.g., CPU time, memory usage)
3. Processes can exist in serveral states: new, ready, running, waiting, terminated

Process Control

1. Process Creation: The operating system creates a new process using system calls (e.g., fork() in Unix-like systems). This involves allocating a PCB, setting up memory, and initializing state.
2. Process Scheduling: The OS decides which process runs at any given time, using scheduling algorithms (e.g., Round Robin, Priority Scheduling) to manage the CPU's time among processes.
3. Process Suspension and Resumption: Processes can be temporarily suspended (put into a waiting state) and later resumed, allowing for efficient resource utilization.
4. Inter-Process Communication (IPC): Processes often need to communicate with each other. The OS provides mechanisms like pipes, message queues, and shared memory to facilitate this.
5. Process Termination: When a process finishes execution, the OS cleans up by removing its PCB and reclaiming resources. This can be initiated by the process itself or by another process.
6. Process Control Commands: The OS provides commands for process management, such as: create, terminate, suspend, and resume.