MODULE-2

STORAGE NETWORKING TECHNOLOGIES AND VIRTUALIZATION

SAN is a high-speed dedicated network of servers and shared storage. Commom SAN deployments are:

- ✓ FC SAN
- ✓ IP SAN

2.1 Fibre Channel: Overview

- The FC architecture forms the fundamental construct of the SAN infrastructure.
- **Fibre Channel** is a high-speed network technology that runs on high-speed optical fiber cables (preferred for front-end SAN connectivity) and serial copper cables (preferred for back-end disk connectivity).
- The FC technology was created to meet the demand for increased speeds of data transfer among computers, servers, and mass storage subsystems.

2.2 Components of SAN

- Components of FC SAN infrastructure are:

1) **Node Ports,**
2) **Cabling,**
3) **Connectors,**
4) **Interconnecting Devices (Such As Fc Switches Or Hubs),**
5) **San Management Software**.

Node Ports

- In fibre channel, devices such as hosts, storage and tape libraries are all referred to as **Nodes.**
- Each node is a **source or destination** of information for one or more nodes.
- Each node requires one or more ports to provide a physical interface for communicating with other nodes.
- A port operates in full-duplex data transmission mode with a **transmit (Tx) link and a**

receive (Rx) link (see Fig 2.1).

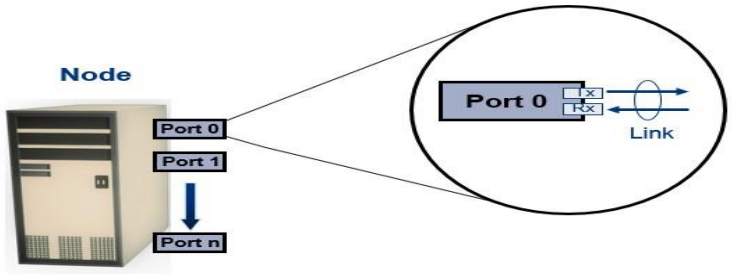


Fig 2.1: Nodes, Ports, links

Cabling

- SAN implementations use optical fiber cabling.
- Copper can be used for shorter distances for back-end connectivity
- Optical fiber cables carry data in the form of light.
- There are two types of optical cables :**Multi-Mode And Single-Mode.**

1) **Multi-mode fiber (MMF)** cable carries multiple beams of light projected at different angles simultaneously onto the core of the cable (see Fig 2.2 (a)).
    - In an MMF transmission, multiple light beams traveling inside the cable tend to disperse and collide. This collision weakens the signal strength after it travels a certain distance — a process known as *modal dispersion*.
    - MMFs are generally used within data centers for shorter distance runs
2) **Single-mode fiber (SMF)** carries a single ray of light projected at the center of the core (see Fig 2.2 (b)).
    - In an SMF transmission, a single light beam travels in a straight line through the core of the fiber.
    - The small core and the single light wave limits modal dispersion. Among all types of fibre cables, single-mode provides minimum signal attenuation over maximum distance (up to 10 km).
    - A single-mode cable is used for long-distance cable runs, limited only by the power

of the laser at the transmitter and sensitivity of the receiver.

- SMFs are used for longer distances.

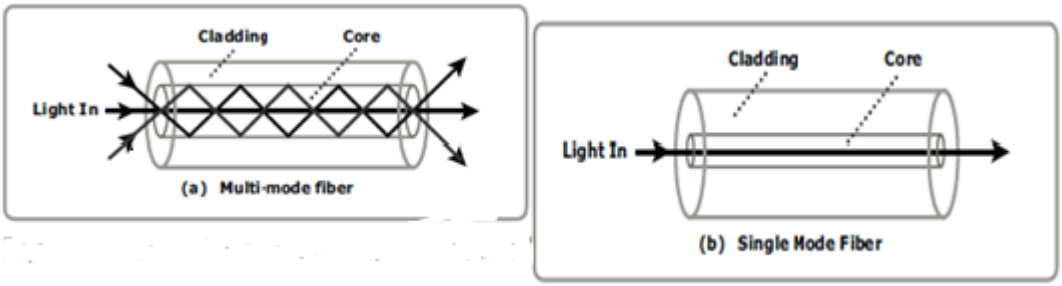


Fig 2.2: Multimode fiber and single-mode fiber

Connectors

- They are attached at the end of the cable to enable swift connection and disconnection of the cable to and from a port.
- A **Standard connector (SC)** (see Fig 2.3 (a)) and a **Lucent connector (LC)** (see Fig 2.3 (b)) are two commonly used connectors for fiber optic cables.
- An SC is used for data transmission speeds up to 1 Gb/s, whereas an LC is used for speeds up to 4 Gb/s.
- Figure 2.3 depicts a Lucent connector and a Standard connector.
- A Straight Tip (ST) is a fiber optic connector with a plug and a socket that is locked with a half-twisted bayonet lock (see Fig 2.3 (c)).

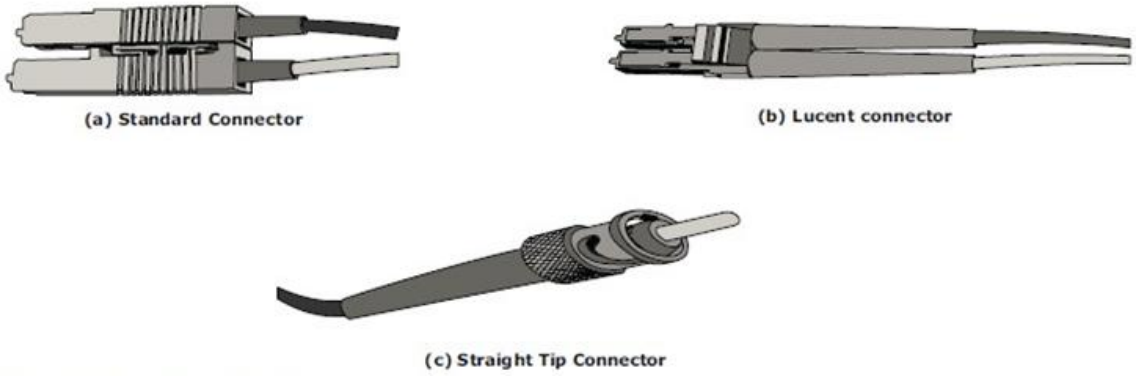


Fig 2.3: SC,LC, and ST connectors

Interconnect Devices

The commonly used interconnecting devices in SAN are

1) **Hubs,**
2) **Switches,**
3) **Directors**

- **Hubs** are used as communication devices in FC-AL implementations. Hubs physically connect nodes in a logical loop or a physical star topology.
- All the nodes must share the bandwidth because data travels through all the connection points. Because of availability of low cost and high performance switches, hubs are no longer used in SANs.
- **Switches** are more **intelligent** than hubs and directly **route data from one physical port to another**. Therefore, nodes do not share the bandwidth. Instead, each node has a dedicated communication path, resulting in bandwidth aggregation.
- Switches are available with:
  - ✓ Fixed port count
  - ✓ Modular design : port count is increased by installing additional port cards to open slots.
- **Directors are larger than switches** and are deployed for data center implementations.
- The function of directors is similar to that of FC switches, but directors have higher port count and fault tolerance capabilities.
- Port card or blade has multiple ports for connecting nodes and other FC switches

SAN Management Software

- SAN management software manages the interfaces between hosts, interconnect devices, and storage arrays.
- The software provides a view of the SAN environment and enables management of various resources from one central console.

- It provides key management functions, including mapping of storage devices, switches, and servers, monitoring and generating alerts for discovered devices, and logical partitioning of the SAN, called *zoning*

2.3 FC Connectivity

The FC architecture supports three basic interconnectivity options:

1) **Point-To-point,**
2) **Arbitrated Loop (Fc-AL),**
3) **FC Switched Fabric**

Point-to-Point

- **Point-to-point** is the simplest FC configuration — two devices are connected directly to each other, as shown in Fig 2.4.
- This configuration provides a dedicated connection for data transmission between nodes.
- The point-to-point configuration offers limited connectivity, as only two devices can communicate with each other at a given time.
- It cannot be scaled to accommodate a large number of network devices. Standard DAS uses point to- point connectivity.

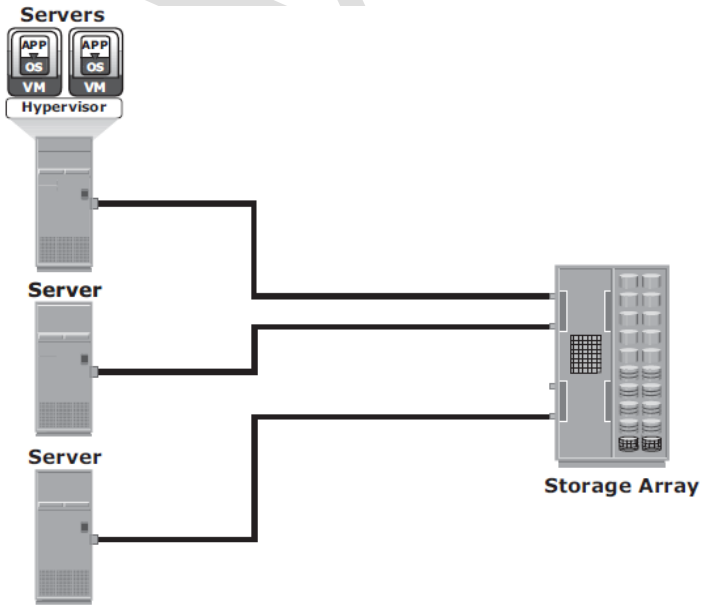


Fig 2.4: Point-to-point connectivity

Fibre Channel Arbitrated Loop

- In the FC-AL configuration, devices are attached to a shared loop, as shown in Fig 2.5.
- FC-AL has the characteristics of a **token ring topology and a physical star topology**.
- In FC-AL, each device contends with other devices to perform I/O operations. Devices on the loop must "arbitrate" to gain control of the loop.
- At any given time, only one device can perform I/O operations on the loop.
- FC-AL implementations may also use hubs whereby the arbitrated loop is physically connected in a star topology.

The FC-AL configuration has the following limitations in terms of scalability:

- FC-AL shares the bandwidth in the loop.
- Only one device can perform I/O operations at a time. Because each device in a loop has to wait for its turn to process an I/O request, the speed of data transmission is low in an FC-AL topology.
- FC-AL uses 8-bit addressing. It can support up to 127 devices on a loop.
- Adding or removing a device results in loop re-initialization, which can cause a momentary pause in loop traffic.

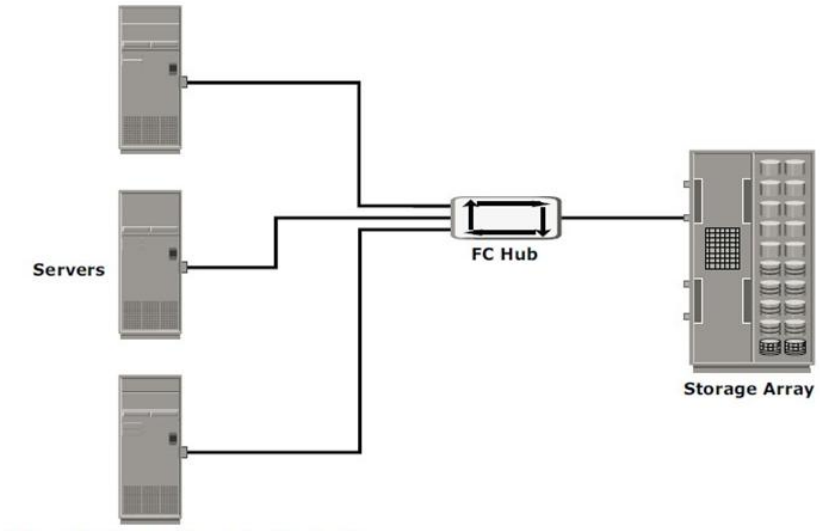


Fig 2.5: Fibre Channel Arbitrated Loop

Fibre Channel Switched Fabric(FC-SW)

- FC-SW provides dedicated data path and scalability.
- The addition and removal of a device doesnot affect the on-going traffic between other devices.
- FC-SW is referred to as **Fabric connect**.
- A Fabric is a logical space in which all nodes communicate with one another in a network. This virtual space can be created with a switch or a network of switches.
- Each switch in a fabric contains a a unique domain identifi er, which is part of the fabric's addressing scheme.
- In a switched fabric, the link between any two switches is called an *Interswitch link* (ISL).
- ISLs enable switches to be connected together to form a single, larger fabric.
- ISLs are used to transfer host-to-storage data and fabric management traffic from one switch to another.
- By using ISLs, a switched fabric can be expanded to connect a large number of nodes.

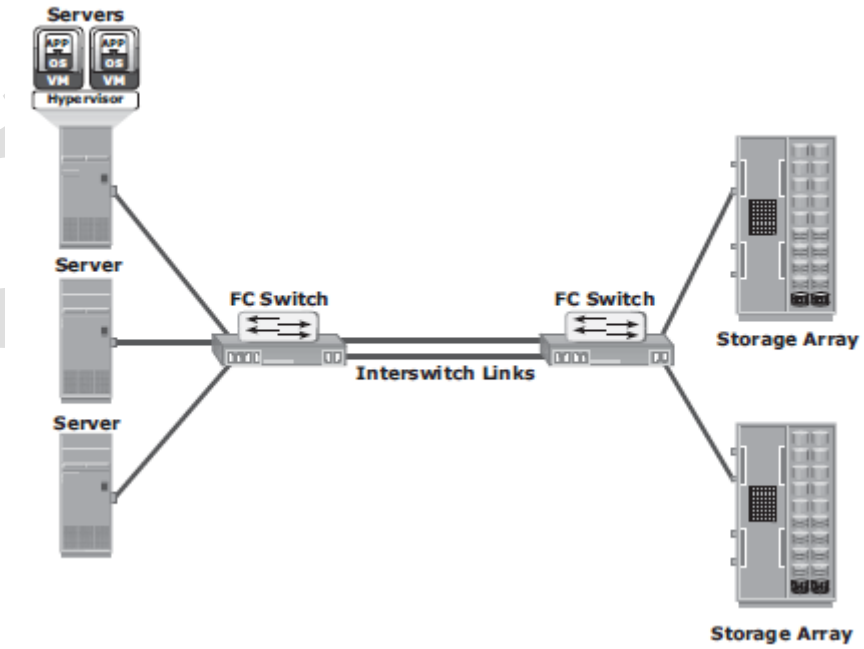


Fig 2.6: Fibre Channel switched Fabric

- A Fabric may contain tiers.
- The number of tiers in a fabric is based on the number of switches between two points that are farthest from each other

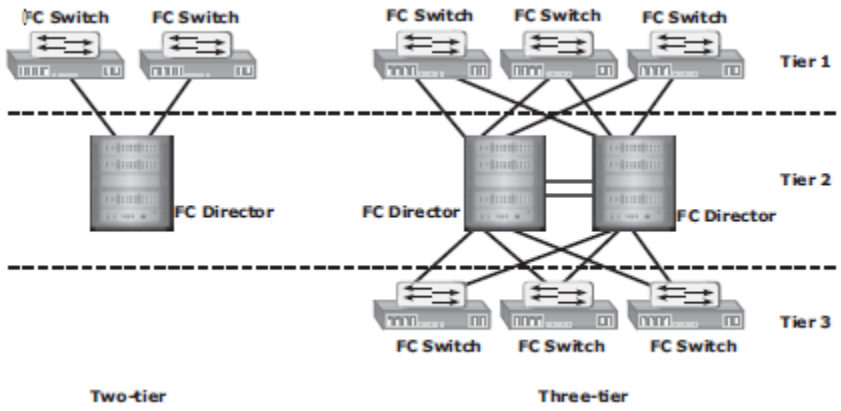


Fig 2.7: Tiered structure of Fibre Channel switched Fabric

FC-SW Transmission

- FC-SW uses switches that can switch data traffi c between nodes directly through switch ports.
- Frames are routed between source and destination by the fabric.

Node A want to communicate with Node B

1. High priority initiator, Node A inserts the ARB frame in the loop.
2. ARB frame is passed to the next node (Node D) in the loop.
3. Node D receives high priority ARB, therefore remains idle.
4. ARB is forwarded to next node (Node C) in the loop.
5. Node C receives high priority ARB, therefore remains idle.
6. ARB is forwarded to next node (Node B) in the loop.
7. Node B receives high priority ARB, therefore remains idle and
8. ARB is forwarded to next node (Node A) in the loop.
9. Node A receives ARB back; now it gains control of the loop and can start communicating with target Node B.

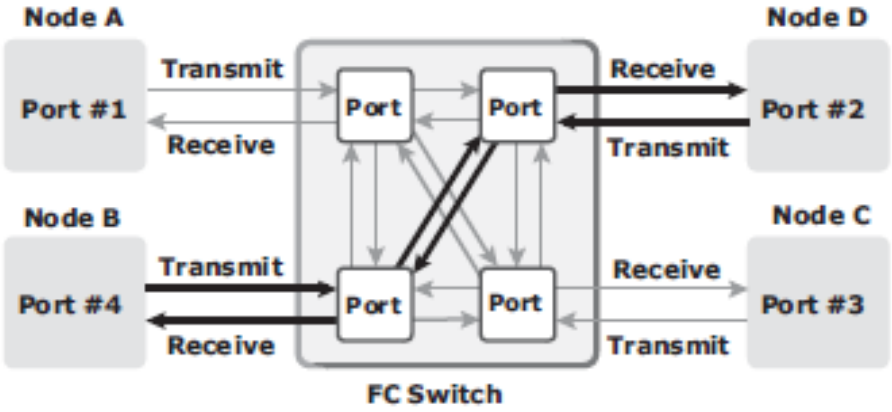


Fig 2.8: Data transmission in fibre channel switched fabric

2.4 Fibre Channel Architecture

- Connections in a SAN are accomplished using FC.
- ***Fibre Channel Protocol (FCP)* is the implementation of serial SCSI-3 over an FC network.** In the FCP architecture, all external and remote storage devices attached to the SAN appear as local devices to the host operating system.
- The key advantages of FCP are as follows:
  - Sustained transmission bandwidth over long distances.
  - Support for a larger number of addressable devices over a network.
  - Theoretically, FC can support over 15 million device addresses on a network.
  - Exhibits the characteristics of channel transport and provides speeds up to 8.5 Gb/s (8 GFC).

Fibre Channel Protocol Stack

- It is easier to understand a communication protocol by viewing it as a structure of independent layers.
- FCP defines the communication protocol in five layers:FC-0 through FC-4 (except FC-3 layer, which is not implemented).
- In a layered communication model, the peer layers on each node talk to each other through defined protocols.
- Fig 2.9 illustrates the fibre channel protocol stack.

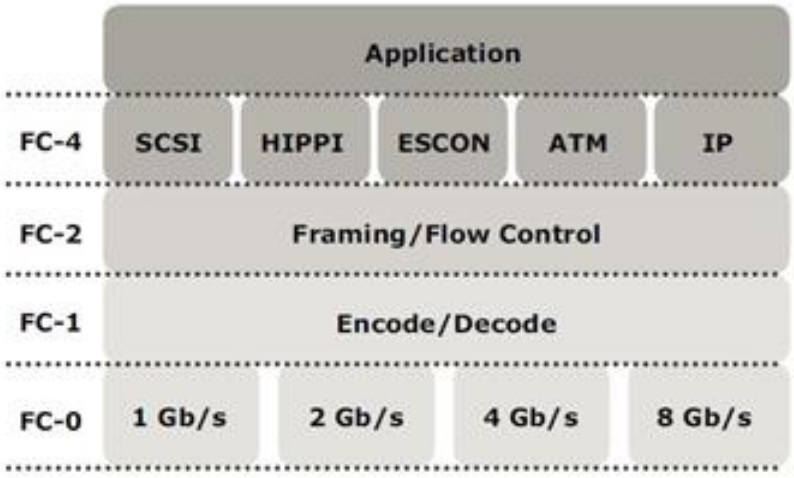


Fig 2.9: Fibre Channel Protocol Stack

- **FC-4 Upper Layer Protocol**
  - ✓ FC-4 is the uppermost layer in the FCP stack.
  - ✓ This layer defines the application interfaces and the way **Upper Layer Protocols (ULPs) are mapped to the lower FC layers.**
  - ✓ The FC standard defines several protocols that can operate on the FC-4 layer (see Fig 2.9). Some of the protocols include SCSI, HIPPI Framing Protocol, Enterprise Storage Connectivity (ESCON), ATM, and IP.
- **FC-2 Transport Layer**
  - ✓ The FC-2 is the transport layer that contains the payload, addresses of the source and destination ports, and link control information.
  - ✓ The FC-2 layer provides Fibre Channel **addressing, structure, and organization of data (frames,sequences, and exchanges).** It also defines **fabric services, classes of service,flow control, and routing.**
- **FC-1 Transmission Protocol**
  - ✓ This layer defines the transmission protocol that includes **serial encoding and decoding rules, special characters used, and error control.**
  - ✓ At the transmitter node, an 8-bit character is encoded into a 10-bit transmissions character.

- ✓ This character is then transmitted to the receiver node.
- ✓ At the receiver node, the 10-bit character is passed to the FC-1 layer, which decodes the 10-bit character into the original 8-bit character.

- **FC-0 Physical Interface**
  - ✓ FC-0 is the lowest layer in the FCP stack.
  - ✓ This layer defines the physical interface, media, and transmission of raw bits.
  - ✓ The FC-0 specification includes cables, connectors, and optical and electrical parameters for a variety of data rates.
  - ✓ The FC transmission can use both electrical and optical media.

2.5 Fibre Channel Addressing

- An FC address is **dynamically assigned** when a port logs on to the fabric.
- The FC address has a distinct format, as shown in Fig 2.10. The addressing mechanism provided here corresponds to the fabric with the switch as an interconnecting device.
- The first field of the FC address contains the domain ID of the switch (see Fig 2.10).
- A *domain ID* is a unique number provided to each switch in the fabric.
- This is an 8-bit field, there are only 239 available addresses for domain ID because some addresses are deemed special and reserved for fabric management services.
- For example, FFFFFC is reserved for the name server, and FFFFFE is reserved for the fabric login service.
- The *area ID* is used to identify a group of switch ports used for connecting nodes. An example of a group of ports with a common area ID is a port card on the switch.
- The last field, the *port ID*, identifies the port within the group.
- The maximum possible number of node ports in a switched fabric is calculated as:
  239 domains ¥ 256 areas ¥ 256 ports = 15,663,104

Fig 2.10 24-bit FC address of N_port

2.6 Zoning

- Zoning is an **FC switch function** that enables nodes within the fabric to be **logically segmented into groups** that can communicate with each other (see Fig 2.11).
- Whenever a change takes place in the name server database, the fabric controller sends a Registered State Change Notifi cation (RSCN) to all the nodes impacted by the change.
- If zoning is not configured, the fabric controller sends an RSCN to all the nodes in the fabric. Involving the nodes that are not impacted by the change results in increased fabric-management traffic.
- Zoning helps to limit the number of RSCNs in a fabric. In the presence of zoning, a fabric sends the RSCN to only those nodes in a zone where the change has occurred.

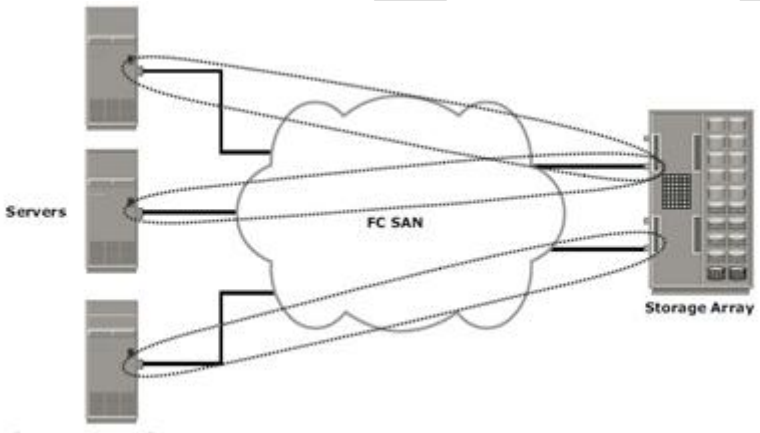


Fig 2.11 Zoning

- Multiple zone sets may be defined in a fabric, but only one zone set can be active at a time.
- A **zone set is a set of zones** and a **zone is a set of members**.
- A member may be in multiple zones. Members, zones, and zone sets form the hierarchy defined in the zoning process (see Fig 2.12).
- **Members** are nodes within the SAN that can be included in a zone.
- **Zones** comprise a set of members that have access to one another. A port or a node can be a member of multiple zones.
- **Zone sets** comprise a group of zones that can be activated or deactivated as a single entity in a fabric. Only one zone set per fabric can be active at a time.

- Zone sets are also referred to as *zone configurations*.

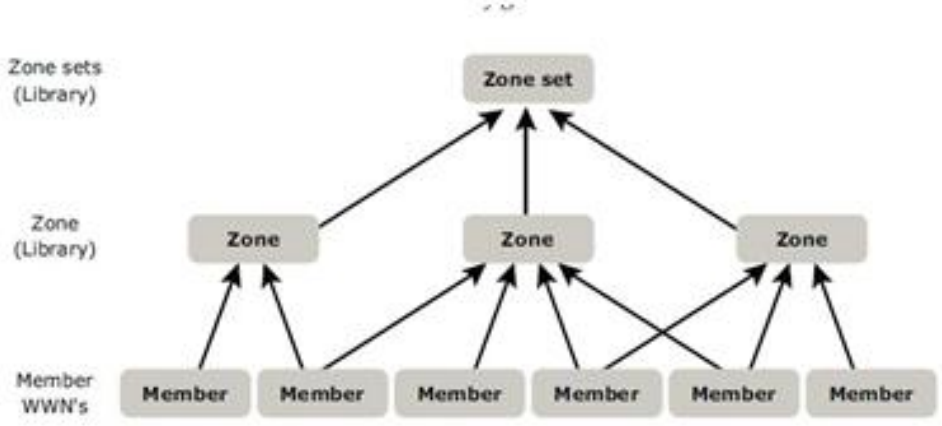


Fig 2.12: Members,Zones, and Zone sets

Types of Zoning

Zoning can be categorized into three types:

1) **Port zoning**
2) **WWN zoning**
3) **Mixed zoning**

Port zoning:

- It uses the **FC addresses** of the physical ports to define zones.
- In port zoning, access to data is determined by the physical switch port to which a node is connected.
- The **FC address is dynamically** assigned when the port logs on to the fabric. Therefore, any change in the fabric configuration affects zoning.
- Port zoning is also called **hard zoning.**
- Although this method is secure, it requires updating of zoning configuration information in the event of fabric reconfiguration.

WWN zoning:

- It uses World Wide Names to define zones.
- WWN zoning is also referred to as **soft zoning**.

- A major advantage of WWN zoning is its flexibility.
- It allows the SAN to be recabled without reconfiguring the zone information. This is possible because the WWN is static to the node port.

Mixed zoning:

- It combines the qualities of both WWN zoning and port zoning.
- Using mixed zoning enables a specific port to be tied to the WWN of a node.

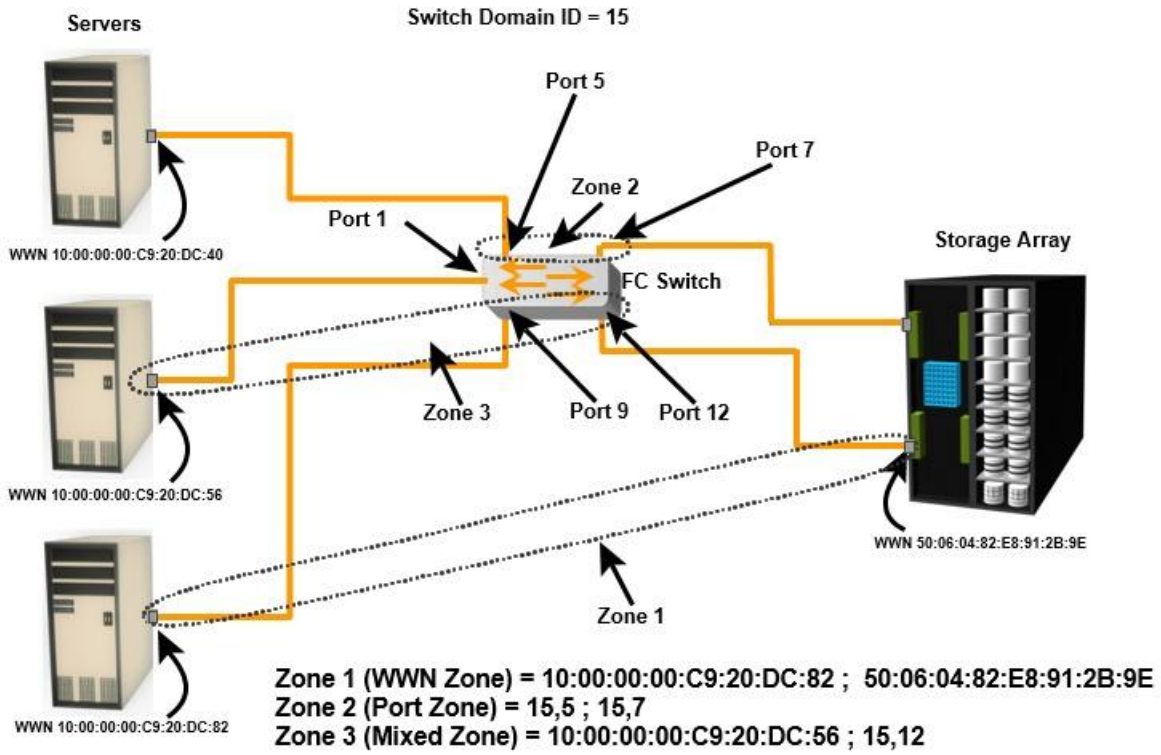


Fig 2.14: Types of Zoning

- Zoning is used in conjunction with LUN masking for controlling server access to storage. However, these are two different activities. Zoning takes place at the fabric level and LUN masking is done at the array level.

2.7 FC Topologies

- Fabric design follows standard topologies to connect devices. There are two types of topologies.
  - **Mesh Topology**
  - **Core-Edge Fabric**

Mesh Topology

- In a mesh topology, **each switch is directly connected to other switches by using ISLs**.
- This topology promotes enhanced connectivity within the SAN.
- When the number of ports on a network increases, the number of nodes that can participate and communicate also increases.
- A mesh topology may be one of the two types: **full mesh or partial mesh.**
- In a **full mesh, every switch is connected to every other switch** in the topology.
- Full mesh topology may be appropriate when the number of switches involved is small. A typical deployment would involve up to four switches or directors, with each of them servicing highly localized host-to-storage traffic. In a full mesh topology, a maximum of one ISL or hop is required for host-to-storage traffic.
- In a **partial mesh** topology, several hops or ISLs may be required for the traffic to reach its destination. Hosts and storage can be located anywhere in the fabric, and storage can be localized to a director or a switch in both mesh topologies. A full mesh topology with a symmetric design results in an even number of switches, whereas a partial mesh has an asymmetric design and may result in an odd number of switches. Fig 2.15 depicts both a full mesh and a partial mesh topology.

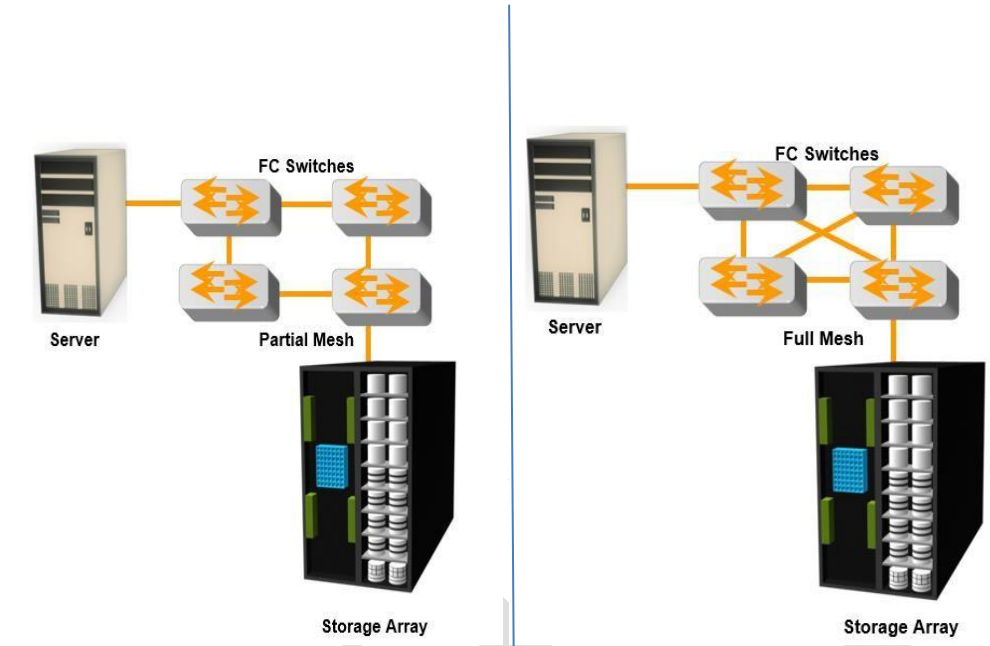


Fig a :Partial **Fig b:Full**

Fig 2.15: Partial and Full mesh Topologies

Core-Edge Fabric

- In the **core-edge fabric** topology, there are two types of switch tiers in this fabric.
- The **edge tier** usually comprises switches and offers an inexpensive approach to adding more hosts in a fabric. The tier at the edge fans out from the tier at the core. The nodes on the edge can communicate with each other.
- The **core tier** usually comprises **enterprise directors** that ensure high fabric availability. Additionally all traffic has to either traverse through or terminate at this tier.
- In a two-tier configuration, all storage devices are connected to the core tier, facilitating fan-out.
- The host-to-storage traffic has to traverse one and two ISLs in a two-tier and three-tier configuration, respectively.
- The core-edge fabric topology increases connectivity within the SAN while conserving overall port utilization. If expansion is required, an additional edge switch can be connected to the core. This topology can have different variations.
- In a **single-core topology**, all hosts are connected to the edge tier and all storage is

connected to the core tier. Fig 2.16 depicts the core and edge switches in a single- core topology.

- A **dual-core topology** can be expanded to include more core switches. However, to maintain the topology, it is essential that new ISLs are created to connect each edge switch to the new core switch that is added. Fig 2.17 illustrates the core and edge switches in a dual-core topology.

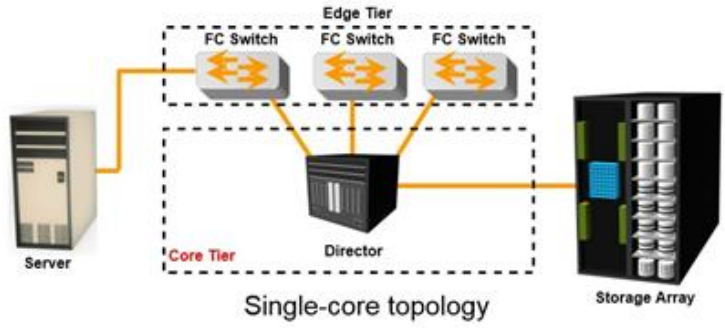


Fig 2.16: Single-core topology

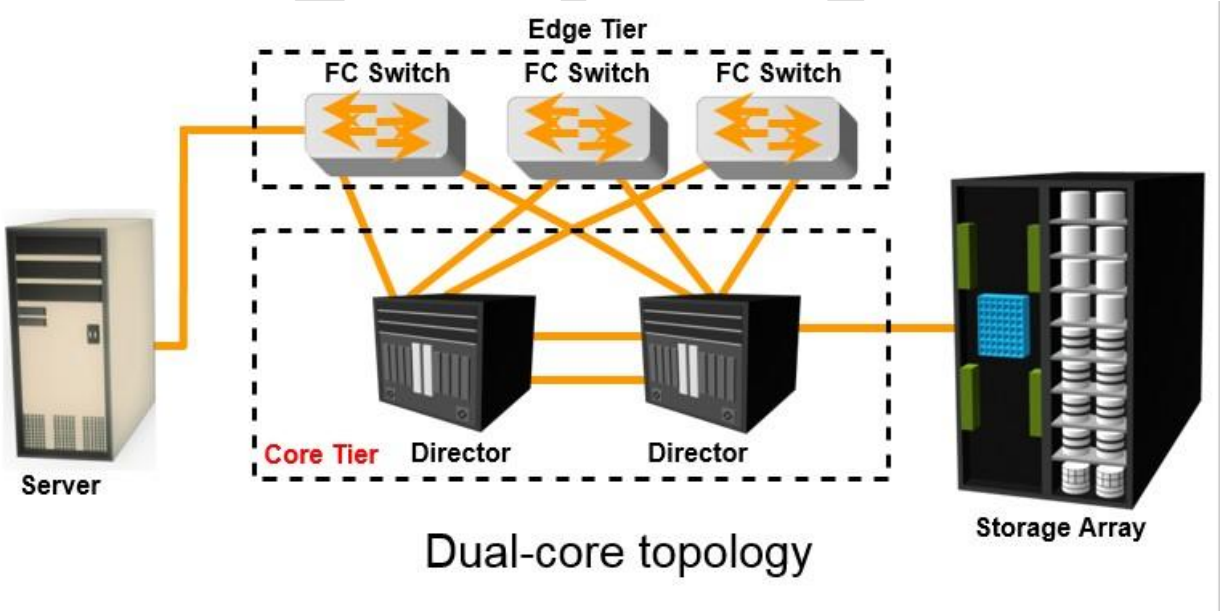


Fig 2.17: multi-core topology

Benefits and Limitations of Core-Edge Fabric

- The core-edge fabric provides one-hop storage access to all storage in the system. Because traffic travels in a deterministic pattern (from the edge to the core), a core-edge provides easier calculation of ISL loading and traffic patterns.
- Because each tier's switch is used for either storage or hosts, one can easily identify which resources are approaching their capacity, making it easier to develop a set of rules for scaling and apportioning.
- Core-edge fabrics can be scaled to larger environments by linking core switches, adding more core switches, or adding more edge switches.
- However, the core-edge fabric may lead to some performance-related problems because scaling a core-edge topology involves increasing the number of ISLs in the fabric.
- As more edge switches are added, the domain count in the fabric increases.

As the number of cores increases, it is prohibitive to continue to maintain ISLs from each core to each edge switch. When this happens, the fabric design is changed to a **compound or complex core-edge design.**

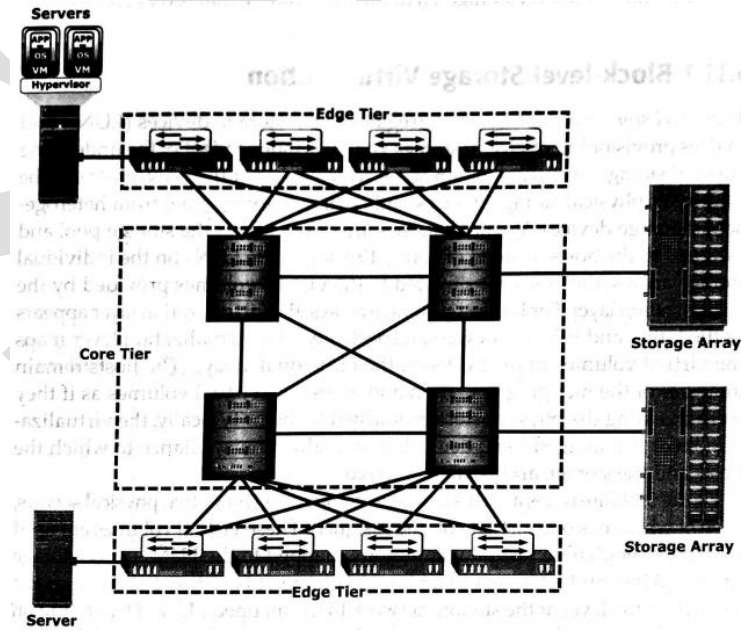


Fig 2.18: Compound core-edge topology

2.8 SAN based virtualization and VSAN technology

There are two network-based virtualization techniques in a SAN environment:

- block-level storage virtualization
- virtual SAN (VSAN).

Block-level Storage Virtualization

- *Block-level storage virtualization* aggregates block storage devices (LUNs) and enables provisioning of virtual storage volumes, independent of the underlying physical storage.
- A virtualization layer, which exists at the SAN, abstracts the identity of physical storage devices and creates a storage pool from heterogeneous storage devices.
- Virtual volumes are created from the storage pool and assigned to the hosts.
- Instead of being directed to the LUNs on the individual storage arrays, the hosts are directed to the virtual volumes provided by the virtualization layer.
- For hosts and storage arrays, the virtualization layer appears as the target and initiator devices, respectively.
- The virtualization layer maps the virtual volumes to the LUNs on the individual arrays.
- The hosts remain unaware of the mapping operation and access the virtual volumes as if they were accessing the physical storage attached to them.
- Typically, the virtualization layer is managed via a dedicated virtualization appliance to which the hosts and the storage arrays are connected.
- Fig 2.19 illustrates a virtualized environment. It shows two physical servers, each of which has one virtual volume assigned. These virtual volumes are used by the servers. These virtual volumes are mapped to the LUNs in the storage arrays.
- When an I/O is sent to a virtual volume, it is redirected through the virtualization layer at the storage network to the mapped LUNs.

- Depending on the capabilities of the virtualization appliance, the architecture may allow for more complex mapping between array LUNs and virtual volumes.

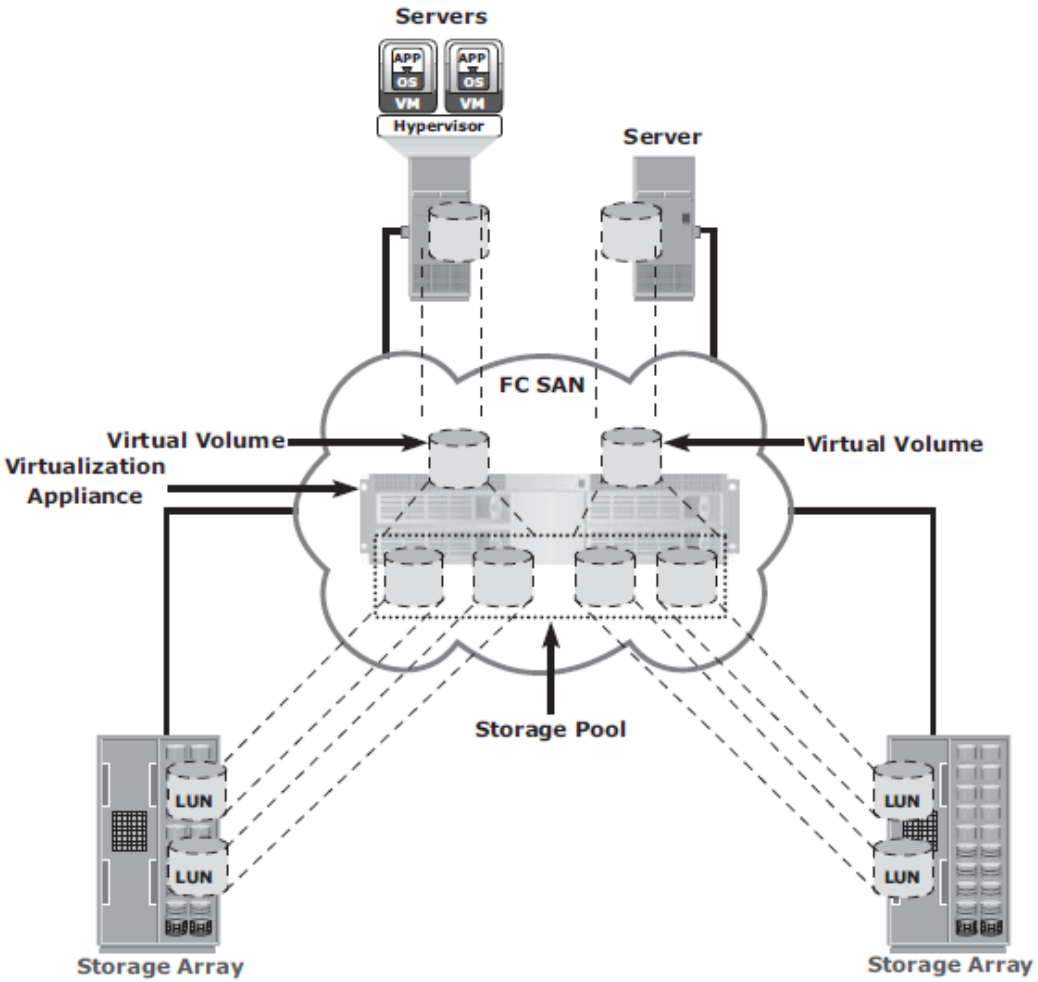


Fig 2.19 Block-level storage virtualization

- Block-level storage virtualization also provides the advantage of nondisruptive data migration.
- In a traditional SAN environment, LUN migration from one array to another is an offline event because the hosts needed to be updated to reflect the new array configuration.
- In other instances, host CPU cycles were required to migrate data from one array to the other, especially in a multivendor environment.
- With a block-level virtualization as a solution, the virtualization layer handles the back-end

migration of data, which enables LUNs to remain online and accessible while data is migrating.

- No physical changes are required because the host still points to the same virtual targets on the virtualization layer.
- Previously, block-level storage virtualization provided nondisruptive data migration only within a data center. The new generation of block-level storage virtualization enables nondisruptive data migration both within and between data centers.
- It provides the capability to connect the virtualization layers at multiple data centers. The connected virtualization layers are managed centrally and work as a single virtualization layer stretched across data centers (Fig 2.20). This enables the federation of block-storage resources both within and across data centers. The virtual volumes are created from the federated storage resources.

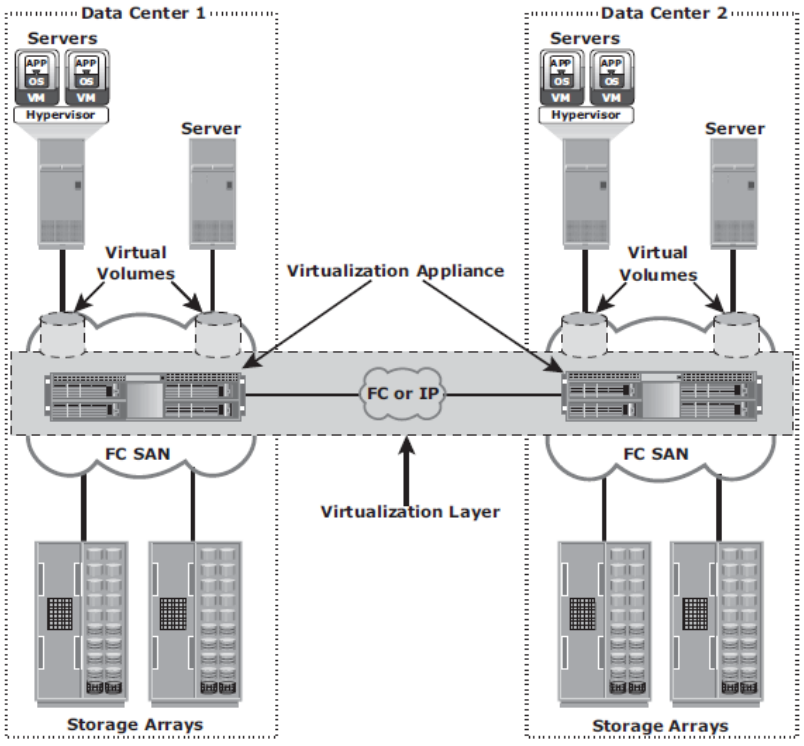


Fig 2.20 Federation of block storage across data centers

Virtual SAN (VSAN)

- *Virtual SAN* (also called *virtual fabric*) is a logical fabric on an FC SAN, which enables communication among a group of nodes regardless of their physical location in the fabric.
- In a VSAN, a group of hosts or storage ports communicate with each other using a virtual topology defined on the physical SAN.
- Multiple VSANs may be created on a single physical SAN.
- Each VSAN acts as an independent fabric with its own set of fabric services, such as name server, and zoning.
- Fabric-related configurations in one VSAN do not affect the traffic in another.
- VSANs improve SAN security, scalability, availability, and manageability.
- VSANs facilitate an easy, flexible, and less expensive way to manage networks.
- Configuring VSANs is easier and quicker compared to building separate physical FC SANs for various node groups.
- To regroup nodes, an administrator simply changes the VSAN configurations without moving nodes and recabling.

2.9 iSCSI

- iSCSI is an IP based protocol that establishes and manages connections between host and storage over IP, as shown in Fig 2.21.
- iSCSI encapsulates SCSI commands and data into an IP packet and transports them using TCP/IP.
- iSCSI is widely adopted for connecting servers to storage because it is relatively inexpensive and easy to implement, especially in environments in which an FC SAN does not exist.

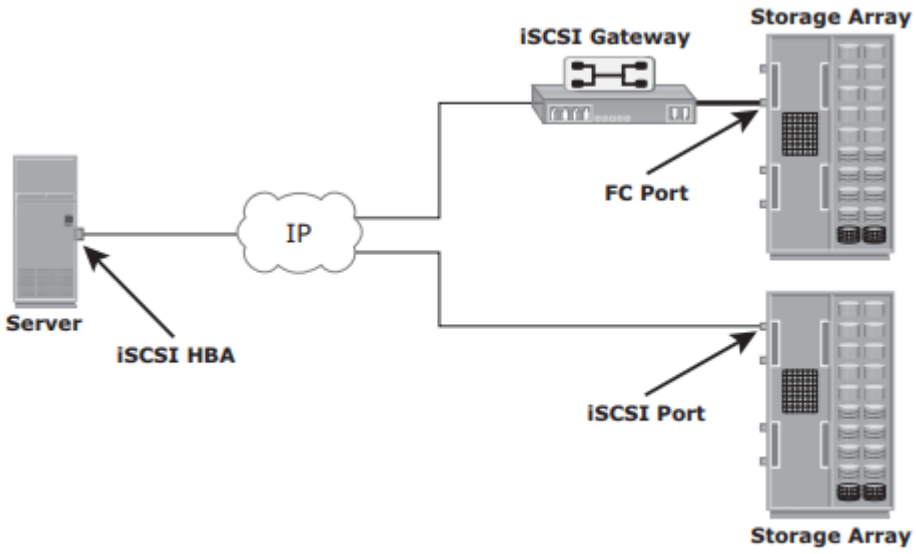


Fig 2.21: iSCSI implementation

2.9.1 Components of iSCSI

- An initiator (host), target (storage or iSCSI gateway), and an IP-based network are the key iSCSI components.
- If an iSCSI-capable storage array is deployed, then a host with the iSCSI initiator can directly communicate with the storage array over an IP network.
- However, in an implementation that uses an existing FC array for iSCSI communication, an iSCSI gateway is used.
- These devices perform the translation of IP packets to FC frames and vice versa, thereby bridging the connectivity between the IP and FC environments.

2.9.2 iSCSI Host Connectivity

The three iSCSI host connectivity options are:

- A standard NIC with software iSCSI initiator,
- a TCP offload engine (TOE) NIC with software iSCSI initiator,

- an iSCSI HBA

- The function of the iSCSI initiator is to route the SCSI commands over an IP network.
- A **standard NIC with a software iSCSI** initiator is the simplest and least expensive connectivity option. It is easy to implement because most servers come with at least one, and in many cases two, embedded NICs. It requires only a software initiator for iSCSI functionality. Because NICs provide standard IP function, encapsulation of SCSI into IP packets and decapsulation are carried out by the host CPU. This places additional overhead on the host CPU. If a standard NIC is used in heavy I/O load situations, the host CPU might become a bottleneck. TOE NIC helps reduce this burden.
- A **TOE NIC** offloads TCP management functions from the host and leaves only the iSCSI functionality to the host processor. The host passes the iSCSI information to the TOE card, and the TOE card sends the information to the destination using TCP/IP. Although this solution improves performance, the iSCSI functionality is still handled by a software initiator that requires host CPU cycles.
- An **iSCSI HBA** is capable of providing performance benefi ts because it offloads the entire iSCSI and TCP/IP processing from the host processor. The use of an iSCSI HBA is also the simplest way to boot hosts from a SAN environment via iSCSI. If there is no iSCSI HBA, modifi cations must be made to the basic operating system to boot a host from the storage devices because the NIC needs to obtain an IP address before the operating system loads. The functionality of an iSCSI HBA is similar to the functionality of an FC HBA.

2.9.3 iSCSI Topologies

- Two topologies of iSCSI implementations are **native and bridged**.
- Native topology does not have FC components.
- The initiators may be either directly attached to targets or connected through the IP network.
- Bridged topology enables the coexistence of FC with IP by providing iSCSI-to-FC bridging functionality.
- For example, the initiators can exist in an IP environment while the storage remains in an FC

environment.

Native iSCSI Connectivity

- FC components are not required for iSCSI connectivity if an iSCSI-enabled array is deployed.
- In Fig 2.22(a), the array has one or more iSCSI ports configured with an IP address and is connected to a standard Ethernet switch.
- After an initiator is logged on to the network, it can access the available LUNs on the storage array.
- A single array port can service multiple hosts or initiators as long as the array port can handle the amount of storage traffic that the hosts generate.

Bridged iSCSI Connectivity

- A bridged iSCSI implementation includes FC components in its configuration.
- Fig 2.22(b), illustrates iSCSI host connectivity to an FC storage array. In this case, the array does not have any iSCSI ports. Therefore, an external device, called a gateway or a multiprotocol router, must be used to facilitate the communication between the iSCSI host and FC storage.
- The gateway converts IP packets to FC frames and vice versa.
- The bridge devices contain both FC and Ethernet ports to facilitate the communication between the FC and IP environments.
- In a bridged iSCSI implementation, the iSCSI initiator is configured with the gateway's IP address as its target destination.
- On the other side, the gateway is configured as an FC initiator to the storage array.
- **Combining FC and Native iSCSI Connectivity:** The most common topology is a

combination of FC and native iSCSI. Typically, a storage array comes with both FC and iSCSI ports that enable iSCSI and FC connectivity in the same environment, as shown in Fig 2.22(c).

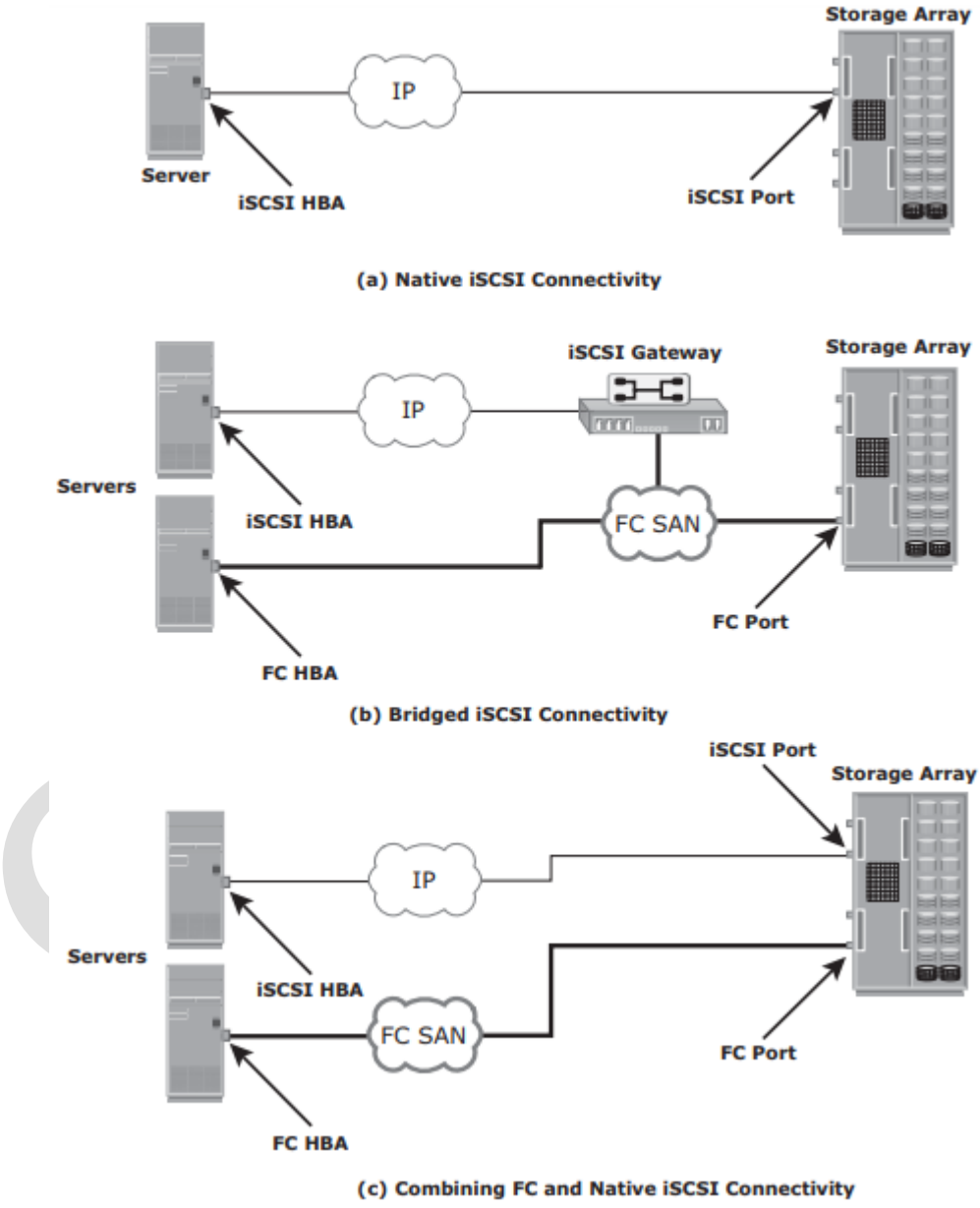


Fig 2.22 : iSCSI Topologies

2.9.4 iSCSI Protocol Stack

- Fig 2.23 displays a model of the iSCSI protocol layers and depicts the encapsulation order of the SCSI commands for their delivery through a physical carrier.

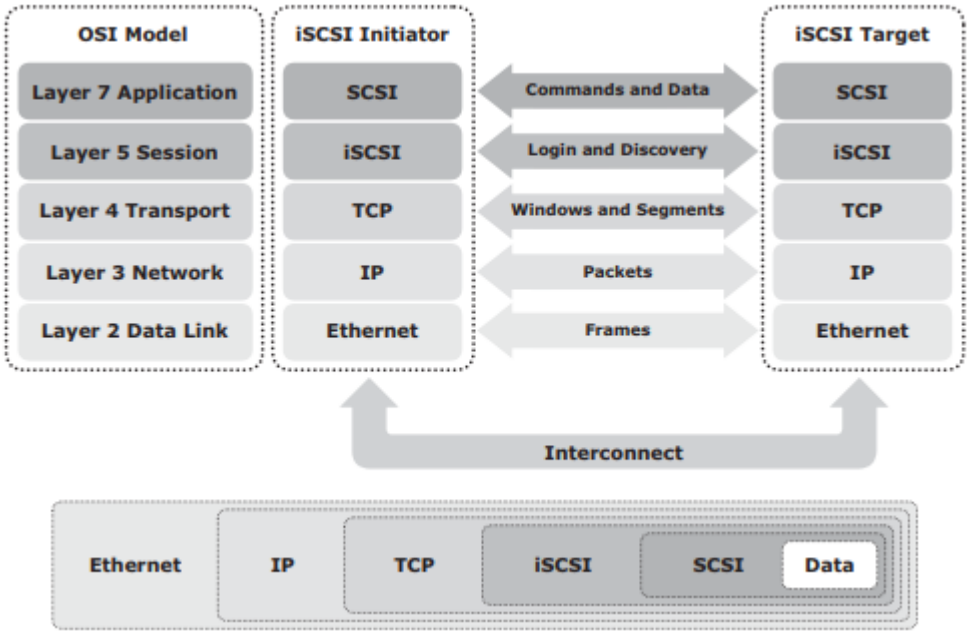


Fig 2.23: iSCSI protocol stack

- SCSI is the command protocol that works at the application layer of the Open System Interconnection (OSI) model.
- The initiators and targets use SCSI commands and responses to talk to each other.
- The SCSI command descriptor blocks, data, and status messages are encapsulated into TCP/IP and transmitted across the network between the initiators and targets.
- iSCSI is the session-layer protocol that initiates a reliable session between devices that recognize SCSI commands and TCP/IP.
- The iSCSI session-layer interface is responsible for handling login, authentication, target discovery, and session management.
- TCP is used with iSCSI at the transport layer to provide reliable transmission.

- TCP controls message flow, windowing, error recovery, and retransmission.
- It relies upon the network layer of the OSI model to provide global addressing and connectivity.
- The Layer 2 protocols at the data link layer of this model enable node-to-node communication through a physical network.

2.9.5 iSCSI PDU

- A *protocol data unit* (PDU) is the basic "information unit" in the iSCSI environment.
- The iSCSI initiators and targets communicate with each other using iSCSI PDUs. This communication includes establishing iSCSI connections and iSCSI sessions, performing iSCSI discovery, sending SCSI commands and data, and receiving SCSI status.
- All iSCSI PDUs contain one or more header segments followed by zero or more data segments.
- The PDU is then encapsulated into an IP packet to facilitate the transport.
- A PDU includes the components shown in Fig 2.23.
- The IP header provides packet-routing information to move the packet across a network.
- The TCP header contains the information required to guarantee the packet delivery to the target.
- The iSCSI header (basic header segment) describes how to extract SCSI commands and data for the target. iSCSI adds an optional CRC, known as the *digest*, to ensure datagram integrity. This is in addition to TCP checksum and Ethernet CRC.
- The header and the data digests are optionally used in the PDU to validate integrity and data placement.

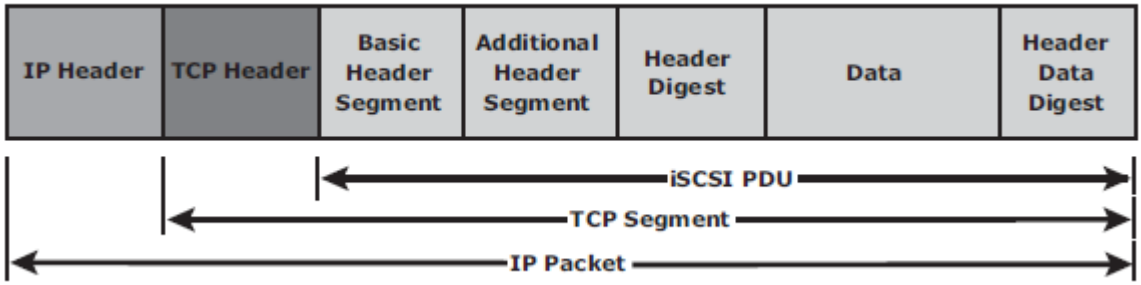


Fig 2.23 iSCSI PDU encapsulated in an IP packet

2.9.6 iSCSI Discovery

- An initiator must discover the location of its targets on the network and the names of the targets available to it before it can establish a session.
- This discovery can take place in two ways:
  - **SendTargets discovery**
  - **internet Storage Name Service (iSNS).**
- In *SendTargets discovery*, the initiator is manually configured with the target's network portal to establish a discovery session. The initiator issues the SendTargets command, and the target network portal responds with the names and addresses of the targets available to the host.
- iSNS (Fig 2.24) enables automatic discovery of iSCSI devices on an IP network. The initiators and targets can be configured to automatically register themselves with the iSNS server. Whenever an initiator wants to now the targets that it can access, it can query the iSNS server for a list of available targets.
- The discovery can also take place by using service location protocol (SLP). However, this is less commonly used than SendTargets discovery and iSNS.

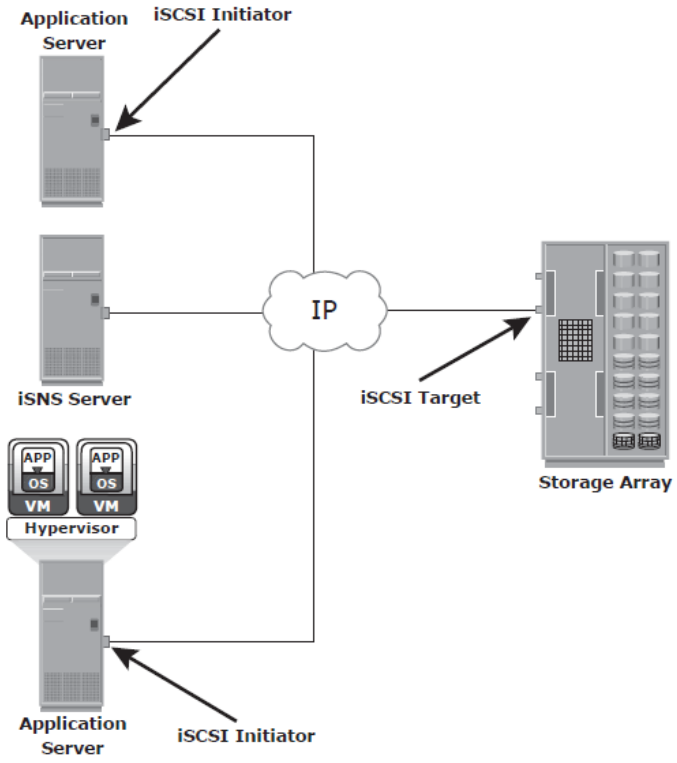


Fig 2.24 : Discovery using iSNS

2.9.7 iSCSI Names

- A unique worldwide iSCSI identifier, known as an *iSCSI name*, is used to identify the initiators and targets within an iSCSI network to facilitate communication.
- The unique identifier can be a combination of the names of the department, application, or manufacturer, serial number, asset number, or any tag that can be used to recognize and manage the devices.
- Following are two types of iSCSI names commonly used:

  - **iSCSI Qualified Name (IQN):**
  - **Extended Unique Identifier (EUI)**

- **iSCSI Qualified Name (IQN):** An organization must own a registered domain name to generate iSCSI Qualifi ed Names. This domain name does not need to be active or resolve to an address. It just needs to be reserved to prevent other organizations from using the same domain name to generate iSCSI names. A date is included in the name to avoid potential conflicts caused by the transfer of domain names.

  An example of an IQN is iqn.2008-02.com.example:*optional_string.* The *optional_string* provides a serial number, an asset number, or any otherdevice identifiers.

- **Extended Unique Identifi er (EUI):** An EUI is a globally unique identifier based on the IEEE EUI-64 naming standard. An EUI is composed of the eui prefix followed by a 16-character hexadecimal name, such aseui.0300732A32598D26.

- In either format, the allowed special characters are dots, dashes, and blank spaces.

2.9.8 iSCSI Session

- An iSCSI session is established between an initiator and a target, as shown in Fig 2.25.
- A session is identified by a session ID (SSID), which includes part of an initiator ID and a target ID.
- The session can be intended for one of the following:
  - The discovery of the available targets by the initiators and the location of a specific target on a network
  - The normal operation of iSCSI (transferring data between initiators and targets)
- There might be one or more TCP connections within each session. Each TCP connection within the session has a unique connection ID (CID).
- An iSCSI session is established via the iSCSI login process. The login process is started when the initiator establishes a TCP connection with the required target either via the well-known port 3260 or a specified target port.

- During the login phase, the initiator and the target authenticate each other and negotiate on various parameters.
- After the login phase is successfully completed, the iSCSI session enters the full-feature phase for normal SCSI transactions. In this phase, the initiator may send SCSI commands and data to the various LUNs on the target.
- The final phase of the iSCSI session is the connection termination phase, which is referred to as the logout procedure.
- The initiator is responsible for commencing the logout procedure; however, the target may also prompt termination by sending an iSCSI message, indicating the occurrence of an internal error condition.
- After the logout request is sent from the initiator and accepted by the target, no further request and response can be sent on that connection.

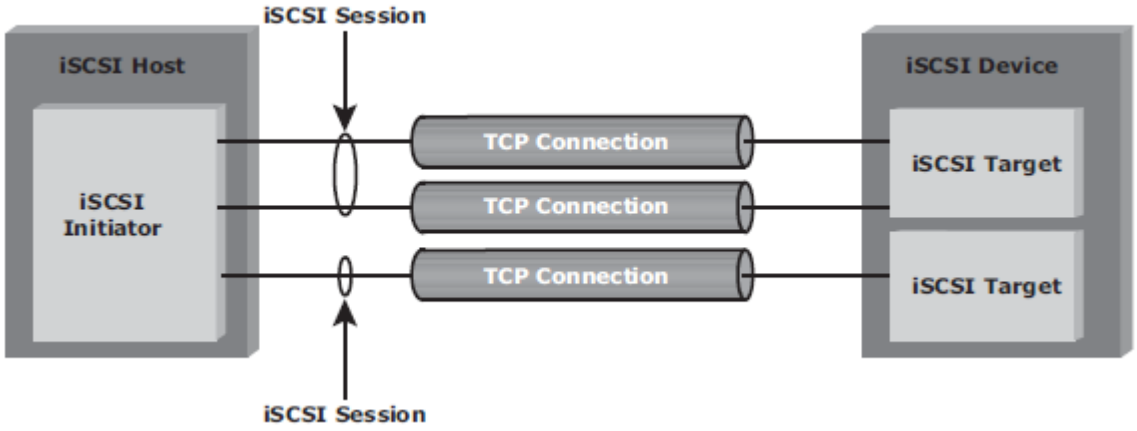


Fig 2.25 iSCSI session

2.9.9 Command Sequencing

- The iSCSI communication between the initiators and targets is based on the request-response command sequences.
- A command sequence may generate multiple PDUs.

- A ***command sequence number*** **(CmdSN**) within an iSCSI session is used for numbering all initiator-to-target command PDUs belonging to the session.
- This number ensures that every command is delivered in the same order in which it is transmitted, regardless of the TCP connection that carries the command in the session.
- Command sequencing begins with the first login command, and the CmdSN is incremented by one for each subsequent command.
- The iSCSI target layer is responsible for delivering the commands to the SCSI layer in the order of their CmdSN.
- Similar to command numbering, a ***status sequence number*** **(StatSN)** is used to sequentially number status responses, as shown in Fig 2.26.
- These unique numbers are established at the level of the TCP connection.
- A target sends ***request-to-transfer*** **(R2T)** PDUs to the initiator when it is ready to accept data.
- A *data sequence number* (DataSN) is used to ensure in-order delivery of data within the same command.
- The DataSN and R2TSN are used to sequence data PDUs and R2Ts, respectively.

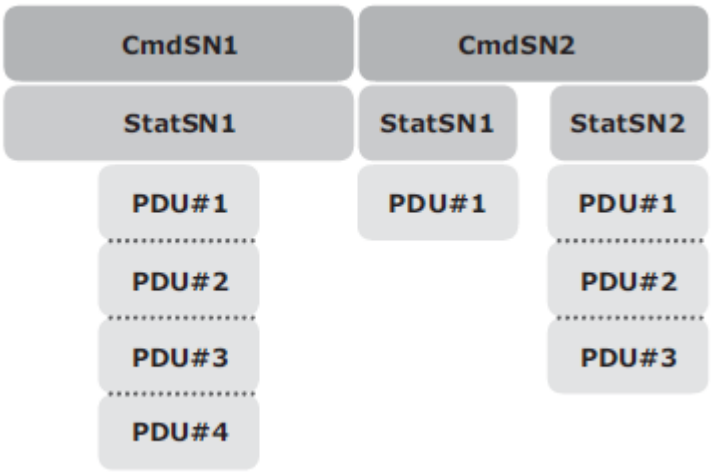


Fig 2.26 Command and status sequence number

2.10 FCIP (Fibre channel over IP)

- FCIP is a IP-based protocol that is used to connect distributed FC-SAN islands.
- Creates virtual FC links over existing IP network that is used to transport FC data between different FC SANS.
- It encapsulates FC frames into IP packet.
- It provides disaster recovery solution.

2.10.1 FCIP Protocol Stack

- The FCIP protocol stack is shown in Fig 2.27. Applications generate SCSI commands and data, which are processed by various layers of the protocol stack.
- The upper layer protocol SCSI includes the SCSI driver program that executes the read-and-write commands.
- Below the SCSI layer is the Fibre Channel Protocol (FCP) layer, which is simply a Fibre Channel frame whose payload is SCSI.
- The FCP layer rides on top of the Fibre Channel transport layer. This enables the FC frames to run natively within a SAN fabric environment. In addition, the FC frames can be encapsulated into the IP packet and sent to a remote SAN over the IP.

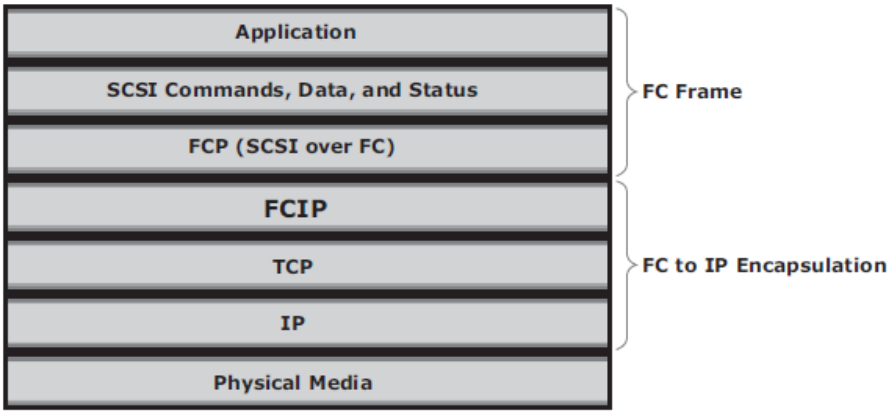


Fig 2.27 FCIP protocol stack

- The FCIP layer encapsulates the Fibre Channel frames onto the IP payload and passes them to the TCP layer (see Fig 2.28). TCP and IP are used for transporting the encapsulated information across Ethernet, wireless, or other media that support the TCP/IP traffic.
- Encapsulation of FC frame into an IP packet could cause the IP packet to be fragmented when the data link cannot support the maximum transmission unit (MTU) size of an IP packet.
- When an IP packet is fragmented, the required parts of the header must be copied by all fragments.
- When a TCP packet is segmented, normal TCP operations are responsible for receiving and re-sequencing the data prior to passing it on to the FC processing portion of the device.

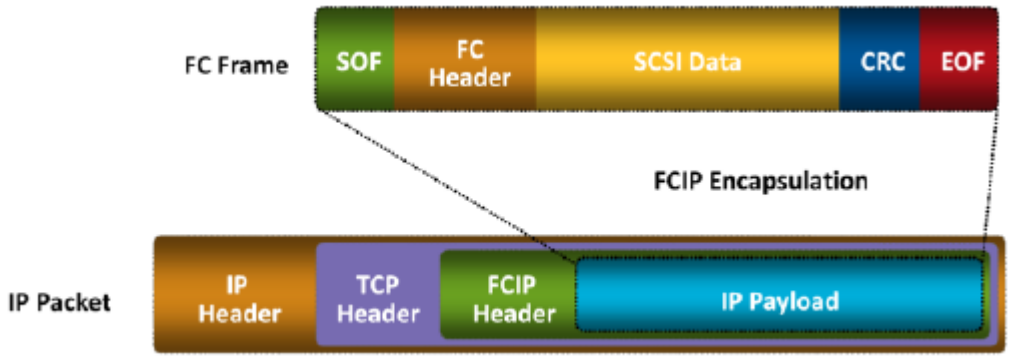


Fig 2.28 FCIP encapsulation

2.10.2 FCIP Topology

- In an FCIP environment, an FCIP gateway is connected to each fabric via a standard FC connection (Fig 2.29).
- The FCIP gateway at one end of the IP network encapsulates the FC frames into IP packets.
- The gateway at the other end removes the IP wrapper and sends the FC data to the layer 2 fabric.
- The fabric treats these gateways as layer 2 fabric switches.
- An IP address is assigned to the port on the gateway, which is connected to an IP network. After the IP connectivity is established, the nodes in the two independent fabrics can communicate

with each other.

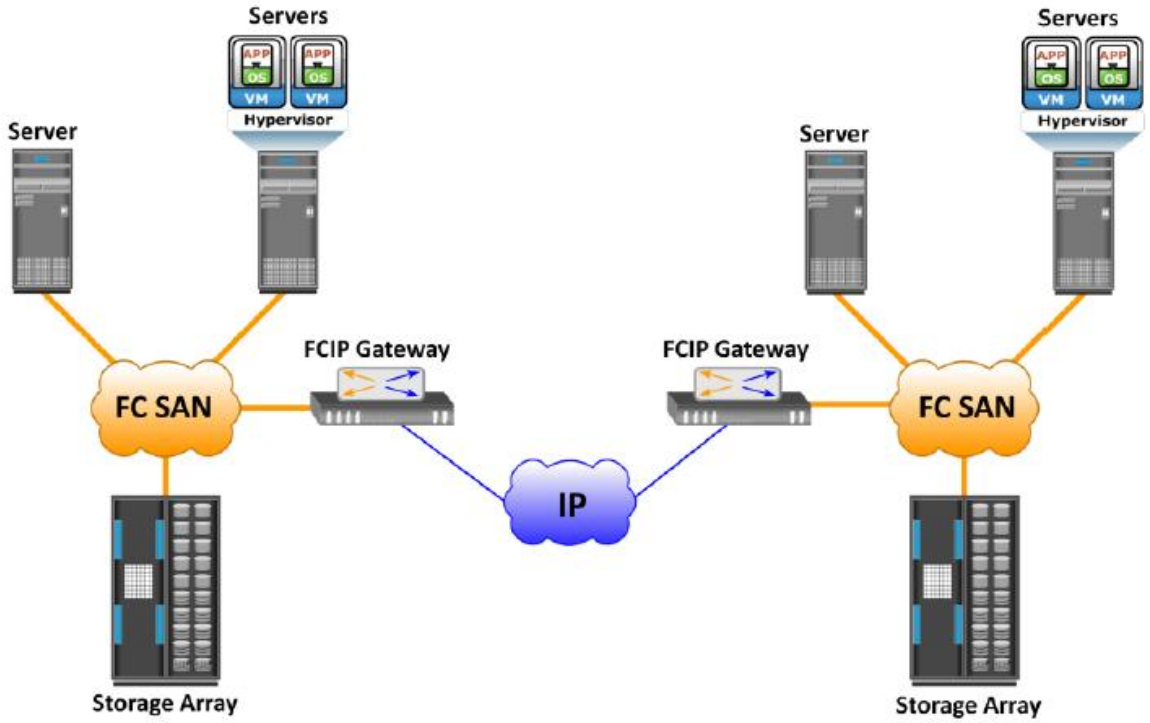


Fig 2.29 FCIP topology

2.11 FCoE (Fibre Channel over Ethernet)

- Data centers typically have multiple networks to handle various types of I/O traffic — for example, an Ethernet network for TCP/IP communication and an FC network for FC communication.
- TCP/IP is typically used for client-server communication, data backup, infrastructure management communication, and so on.
- FC is typically used for moving block-level data between storage and servers.
- To support multiple networks, servers in a data center are equipped with multiple redundant physical network interfaces — for example, multiple Ethernet and FC cards/adapters. In addition, to enable the communication, different types of networking switches and physical cabling infrastructure are implemented in data centers.

- The need for two different kinds of physical network infrastructure increases the overall cost and complexity of data center operation.
- Fibre Channel over Ethernet (FCoE) protocol provides consolidation of LAN and SAN traffic over a single physical interface infrastructure.
- FCoE helps organizations address the challenges of having multiple discrete network infrastructures.
- FCoE uses the Converged Enhanced Ethernet (CEE) link (10 Gigabit Ethernet) to send FC frames over Ethernet.

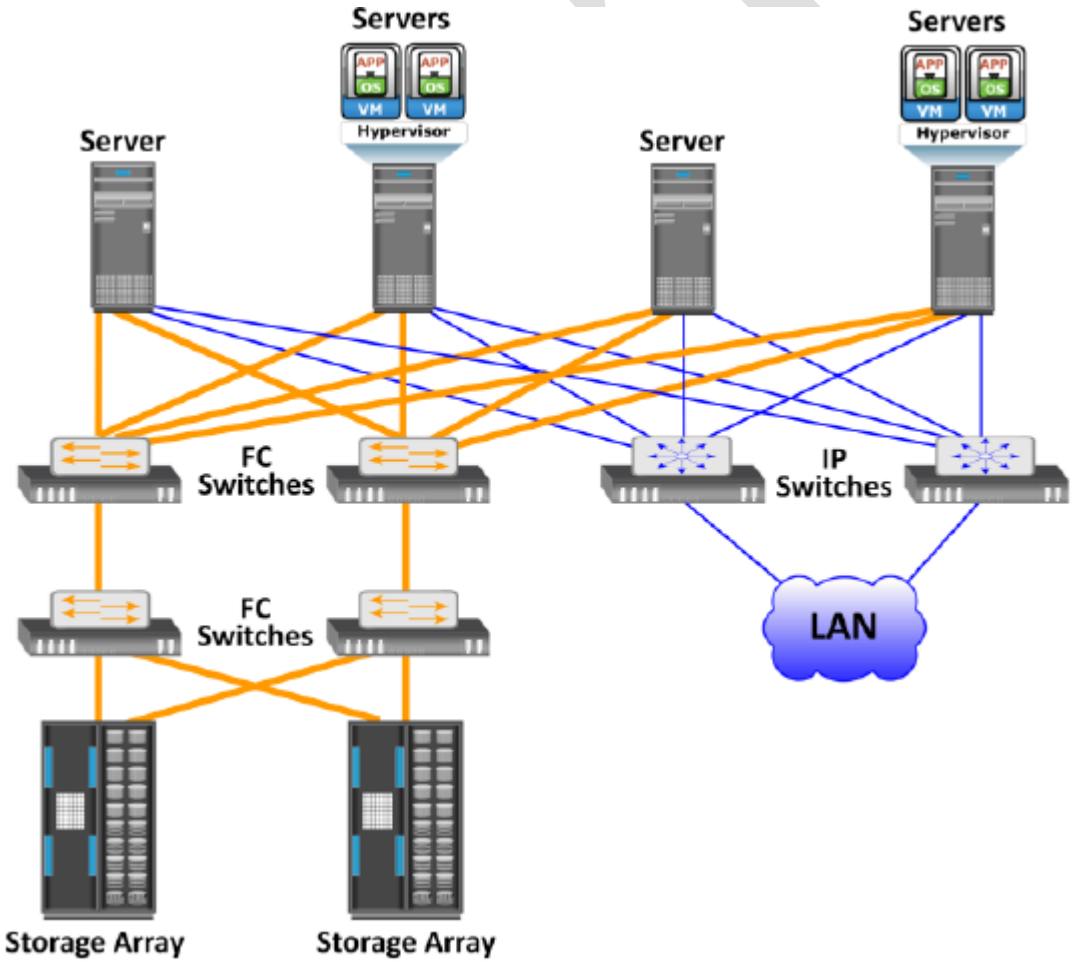


Fig 2.30 Before using FCOE

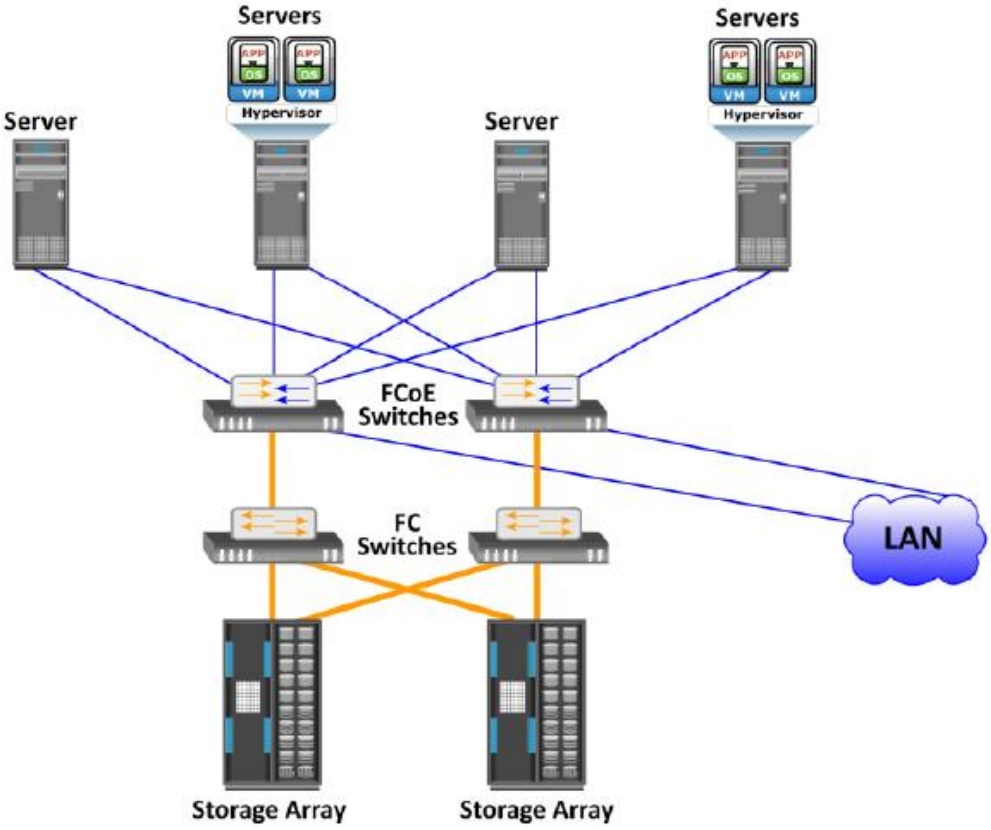


Fig 2.30 After using FCOE

2.11.1 Components of FCOE

The key components of FCOE are :

- Converged Network Adaptors(CNA)
- Cables
- FCOE Switches

Converged Network Adaptors(CNA)

- A CNA provides the functionality of both a standard NIC and an FC HBA in a single adapter and consolidates both types of traffic. CNA eliminates the need to deploy separate adapters and cables for FC and Ethernet communications, thereby reducing the required number of server slots and switch ports.

- As shown in Fig 2.31, a CNA contains separate modules for 10 Gigabit Ethernet, Fibre Channel, and FCoE Application Specific Integrated Circuits (ASICs). The FCoE ASIC encapsulates FC frames into Ethernet frames. One end of this ASIC is connected to 10GbE and FC ASICs for server connectivity, while the other end provides a 10GbE interface to connect to an FCoE switch.

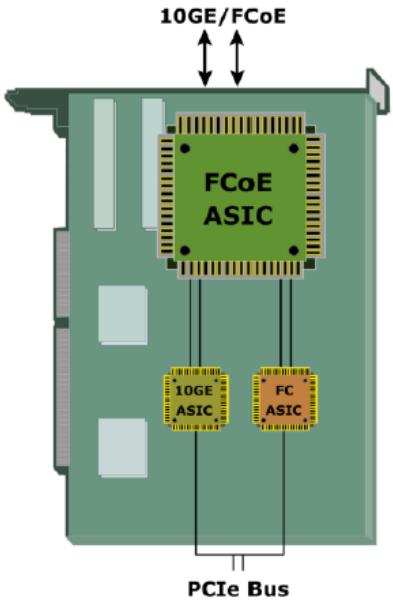


Fig 2.31 Converged Network Adapter

Cables

- There are two options available for FCoE cabling:
    1. Copper based Twinax
    2. standard fiber optical cables.
- A Twinax cable is composed of two pairs of copper cables covered with a shielded casing. The Twinax cable can transmit data at the speed of 10 Gbps over shorter distances up to 10 meters. Twinax cables require less power and are less expensive than fi ber optic cables.
- The Small Form Factor Pluggable Plus (SFP+) connector is the primary connector used for FCoE links and can be used with both optical and copper cables.

FCoE Switches

- An FCoE switch has both **Ethernet switch** and **Fibre Channel switch** functionalities.
- As shown in Fig 2.32, FCoE switch consists of:
    1. *Fibre Channel Forwarder (FCF),*
    2. *Ethernet Bridge,*
    3. set of Ethernet ports
    4. optional FC ports
- The function of the FCF is to encapsulate the FC frames, received from the FC port, into the FCoE frames and also to de-encapsulate the FCoE frames, received from the Ethernet Bridge, to the FC frames.
- Upon receiving the incoming traffic, the FCoE switch inspects the **Ethertype** (used to indicate which protocol is encapsulated in the payload of an Ethernet frame) of the incoming frames and uses that to determine the destination.
    - If the Ethertype of the frame is FCoE, the switch recognizes that the frame contains an FC payload and forwards it to the FCF. From there, the FC is extracted from the FCoE frame and transmitted to FC SAN over the FC ports.
    - If the Ethertype is not FCoE, the switch handles the traffic as usual Ethernet traffic and forwards it over the Ethernet ports.

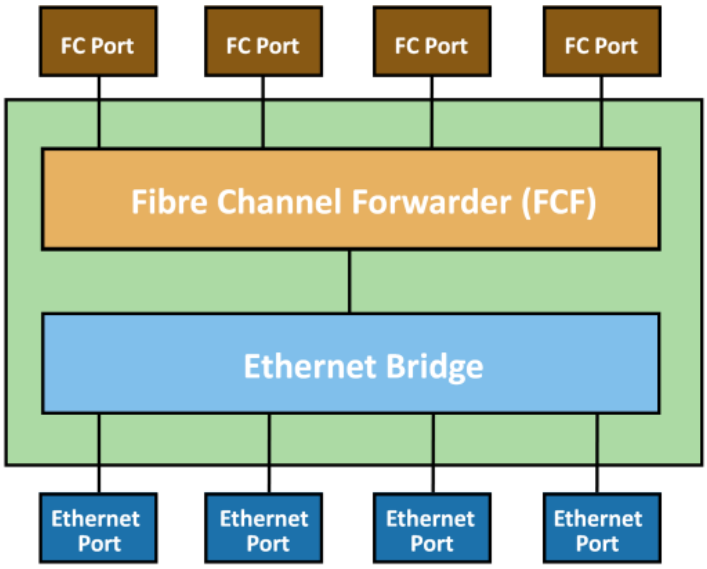


Fig 2.32 FCoE switch generic architecture

2.12 NETWORK ATTACHED STORAGE (NAS)

File Sharing Environment

- File sharing enables users to share files with other users
- In file-sharing environment, the creator or owner of a file determines the type of access to be given to other users and controls changes to the file.
- When multiple access a shared file at the same time, a locking scheme is required to maintain data integrity and also make this sharing possible. This is taken care by file-sharing environment.
- Examples of file sharing methods:
  - File Transfer Protocol (FTP)
  - Distributed File System (DFS)
  - Network File System (NFS) and Common Internet File System (CIFS)
  - Peer-to-Peer (P2P)

What is NAS?

- NAS is an IP based dedicated, high-performance file sharing and storage device.
- Enables NAS clients to share files over an IP network.
- Uses network and file-sharing protocols to provide access to the file data.
- Ex: Common Internet File System (CIFS) and Network File System (NFS).
- Enables both UNIX and Microsoft Windows users to share the same data seamlessly.
- NAS device uses its own operating system and integrated hardware and software components to meet specific file-service needs.
- Its operating system is optimized for file I/O which performs better than a general-purpose server.
- A NAS device can serve more clients than general-purpose servers and provide the benefit of server consolidation.

2.12.1 Components of NAS

- NAS device has *two* key components (as shown in Fig 2.33): **NAS head** and **storage**.
- In some NAS implementations, the storage could be external to the NAS device and shared with other hosts.
- NAS head includes the following components:
    - CPU and memory
    - One or more network interface cards (NICs), which provide connectivity to the client network.
    - An optimized operating system for managing the NAS functionality. It translates file-level requests into block-storage requests and further converts the data supplied at the block level to file data
    - NFS, CIFS, and other protocols for file sharing

- Industry-standard storage protocols and ports to connect and manage physical disk resources

➢ The NAS environment includes clients accessing a NAS device over an IP network using file-sharing protocols.

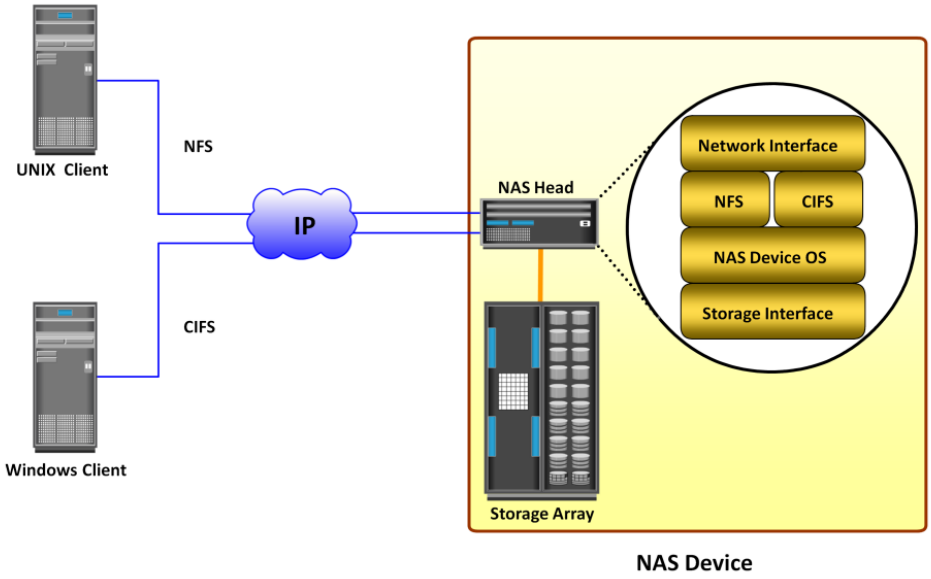


Fig 2.33 Components of NAS

2.12.2 NAS I/O Operation

➢ NAS provides *file-level data access* to its clients. File I/O is a high-level request that specifies the file to be accessed.

➢ Eg: a client may request a file by specifying its name, location, or other attributes. The NAS operating system keeps track of the location of files on the disk volume and converts client file I/O into block-level I/O to retrieve data.

➢ The process of handling I/Os in a NAS environment is as follows:

1. The requestor (client) packages an I/O request into TCP/IP and forwards it through the network stack. The NAS device receives this request from the network.
2. The NAS device converts the I/O request into an appropriate physical storage request,

which is a block-level I/O, and then performs the operation on the physical storage.

3. When the NAS device receives data from the storage, it processes and repackages the data into an appropriate file protocol response.
4. The NAS device packages this response into TCP/IP again and forwards it to the client through the network.

- Fig 2.34 illustrates the NAS I/O operation

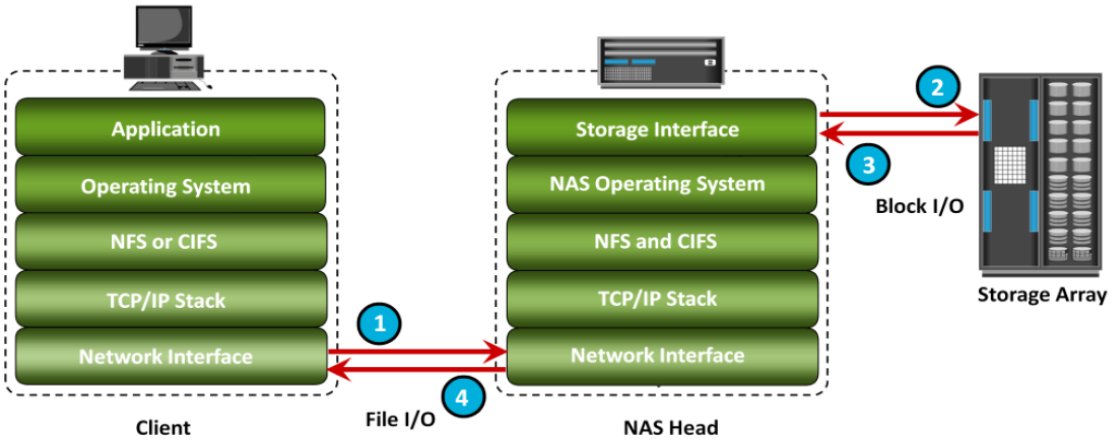


Fig 2.34 NAS I/O Operation

2.12.3 NAS File Sharing Protocols

- NAS devices support multiple file-service protocols to handle file I/O requests
- Two common NAS file sharing protocols are:
  - Common Internet File System (CIFS)
  - Network File System (NFS)
- NAS devices enable users to share file data across different operating environments
- It provides a means for users to migrate transparently from one operating system to another

Network File System (NFS)

- NFS is a **client-server protocol** for file sharing that is commonly used on **UNIX systems**.
- NFS was originally based on the connectionless *User Datagram Protocol (UDP).*
- It uses *Remote Procedure Call (RPC)* as a method of inter-process communication between two computers.
- The NFS protocol provides a set of RPCs to access a remote file system for the following operations:
    - Searching files and directories
    - Opening, reading, writing to, and closing a file
    - Changing file attributes
    - Modifying file links and directories
- NFS creates a connection between the client and the remote system to transfer data.
- NFSv3 and earlier is a stateless protocol
- It does not maintain any kind of table to store information about open files and associated pointers. Each call provides a full set of arguments - a file handle, a particular position to read or write, and the versions of NFS - to access files on the server .
- Currently, three versions of NFS are in use:
    1. **NFS version 2 (NFSv2):** Uses *UDP* to provide a *stateless* network connection between a client and a server. Features, such as locking, are handled outside the protocol.
    2. **NFS version 3 (NFSv3):** Uses *UDP or TCP*, and is based on the *stateless protocol* design. It includes some new features, such as a 64-bit file size, asynchronous writes, and additional file attributes to reduce refetching.
    3. **NFS version 4 (NFSv4):** Uses TCP and is based on a *stateful protocol* design. It offers enhanced security. The latest NFS version 4.1 is the enhancement of NFSv4 and includes some new features, such as session model, parallel NFS (pNFS), and data retention.

Common Internet File System (CIFS)

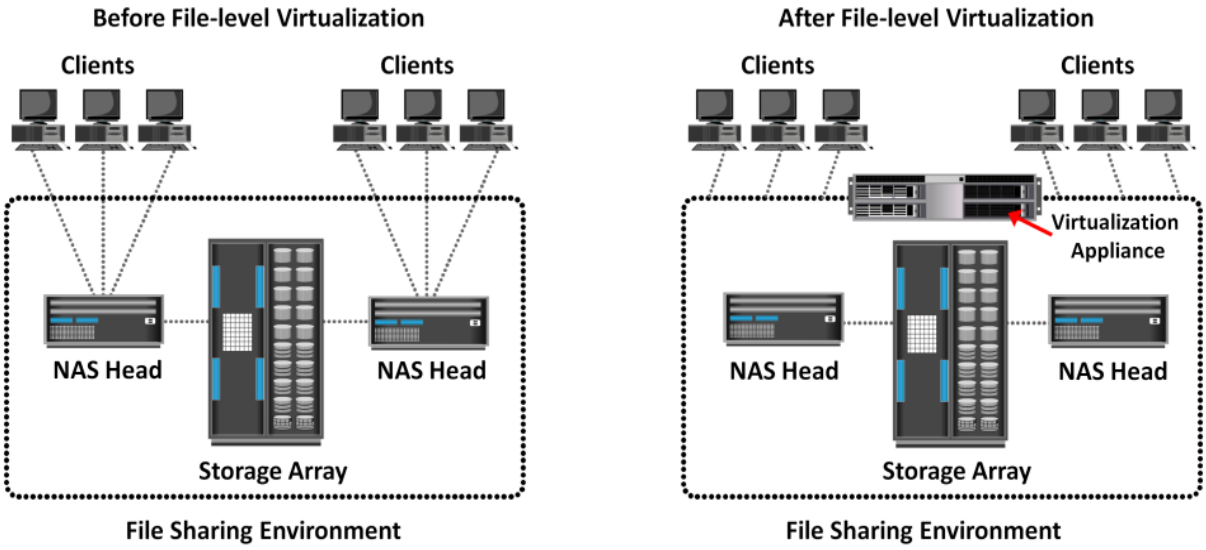
- CIFS is a *client-server application* protocol
- It enables clients to access files and services on remote computers over **TCP/IP**.
- It is a public, or open, variation of **Server Message Block (SMB)** protocol.
- It provides following features to ensure data integrity:
    - It uses file and record locking to prevent users from overwriting the work of another user on a file or a record.
    - It supports fault tolerance and can automatically restore connections and reopen files that were open prior to an interruption. This feature depends on whether an application is written to take advantage of this.
    - CIFS is a stateful protocol because the CIFS server maintains connection information regarding every connected client. If a network failure or CIFS server failure occurs, the client receives a disconnection notification. User disruption is minimized if the application has the embedded intelligence to restore the connection. However, if the embedded intelligence is missing, the user must take steps to reestablish the CIFS connection.
- Users refer to remote file systems with an easy-to-use file-naming scheme:
- Eg: \\server\share or \\servername.domain.suffix\share

2.13 File-level Virtualization

- File-level virtualization, implemented in NAS or the file server environment, provides a simple, non disruptive file-mobility solution.
- It eliminates the dependencies between data accessed at the file level and the location where the files are physically stored.
- It creates a logical pool of storage, enabling users to use a logical path, rather than a physical path, to access files.
- A global namespace is used to map the logical path of a file to the physical path names. File-

level virtualization enables the movement of files across NAS devices, even if the files are being accessed.

Before and After File-level Virtualization



- Dependency between client access and file location
- Underutilized storage resources
- Downtime is caused by data migrations

- Break dependencies between client access and file location
- Storage utilization is optimized
- Non-disruptive migrations

Object-Based Storage & Unified Storage Platform

- **Object-based storage** is a way to store file data in the form of objects based on its *content and other attributes* rather than the name and location.
- Recent studies have shown that more than 90 percent of data generated is unstructured.
- Traditional NAS, which is the dominant solution for storing unstructured data, has become inefficient.
- This demands a smarter approach to manage unstructured data based on its content rather than metadata about its name, location, and so on.

- Due to varied application requirements, organizations have been deploying storage area networks (SANs), NAS, and object-based storage devices (OSDs) in their data centers.
- Deploying these disparate storage solutions adds management complexity, cost and environmental overhead.
- An ideal solution is to have an integrated storage solution that supports block, file, and object access.
- **Unified storage** has emerged as a solution that consolidates *block*, *file*, and *object-based access* within one unified platform.
- It supports multiple protocols for data access and can be managed using a single management interface.

2.14 Object-Based Storage Devices (OSD)

- An OSD is a device that *organizes and stores* unstructured data, such as movies, office documents, and graphics, as objects.
- **Object-based storage** provides a scalable, self-managed, protected, and shared storage option. OSD stores data in the form of objects.
- OSD uses *flat address space* to store data. Therefore, there is no hierarchy of directories and files; as a result, a large number of objects can be stored in an OSD system (see Fig 2.35).

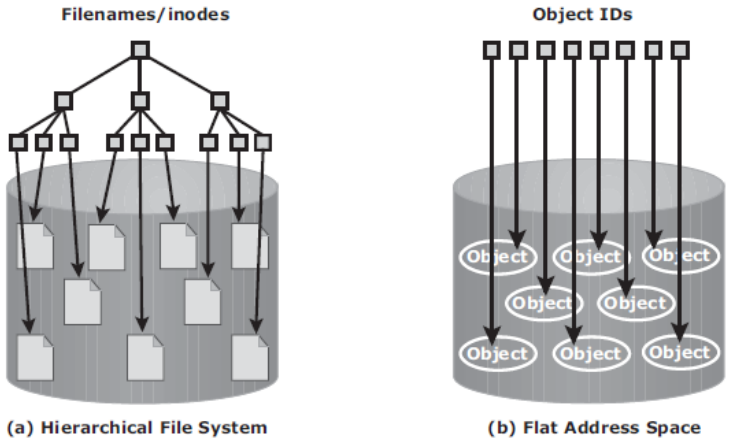


Fig 2.35 Hierarchical file system versus flat address space

- An object might contain user data, related metadata (size, date, ownership, and so on), and other attributes of data (retention, access pattern, and so on); See Fig 2.36.
- Each object stored in the system is identified by a unique ID called the **object ID**.
- The object ID is generated using specialized algorithms such as hash function on the data and guarantees that every object is uniquely identified.

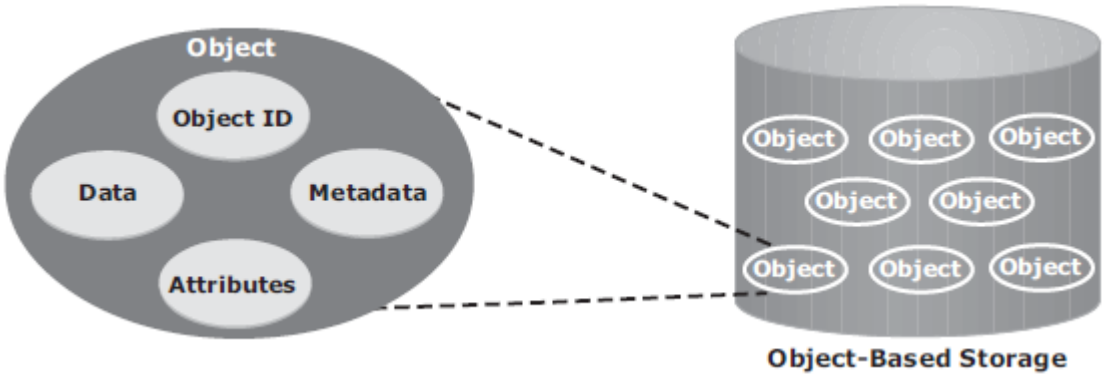


Fig 2.36 Object Structure

2.14.1 Object-Based Storage Architecture

- An I/O in the traditional block access method passes through various layers in the I/O path.
- The I/O generated by an application passes through the file system, the channel, or network and reaches the disk drive.
- When the file system receives the I/O from an application, the file system maps the incoming I/O to the disk blocks. The block interface is used for sending the I/O over the channel or network to the storage device. The I/O is then written to the block allocated on the disk drive.
- Fig 2.37 (a) illustrates the block-level access.
- The file system has two components: *user component* and *storage component*.

    1. The user component of the file system performs functions such as hierarchy management, naming, and user access control.
    2. The storage component maps the files to the physical location on the disk drive.

- When an application accesses data stored in OSD, the request is sent to the file system user component. The file system user component communicates to the OSD interface, which in turn sends the request to the storage device.
- The storage device has the OSD storage component responsible for managing the access to the object on a storage device.
- Fig 2.37 (b) illustrates the object-level access.
- After the object is stored, the OSD sends an acknowledgment to the application server.
- The OSD storage component manages all the required low-level storage and space management functions. It also manages security and access control functions for the objects.

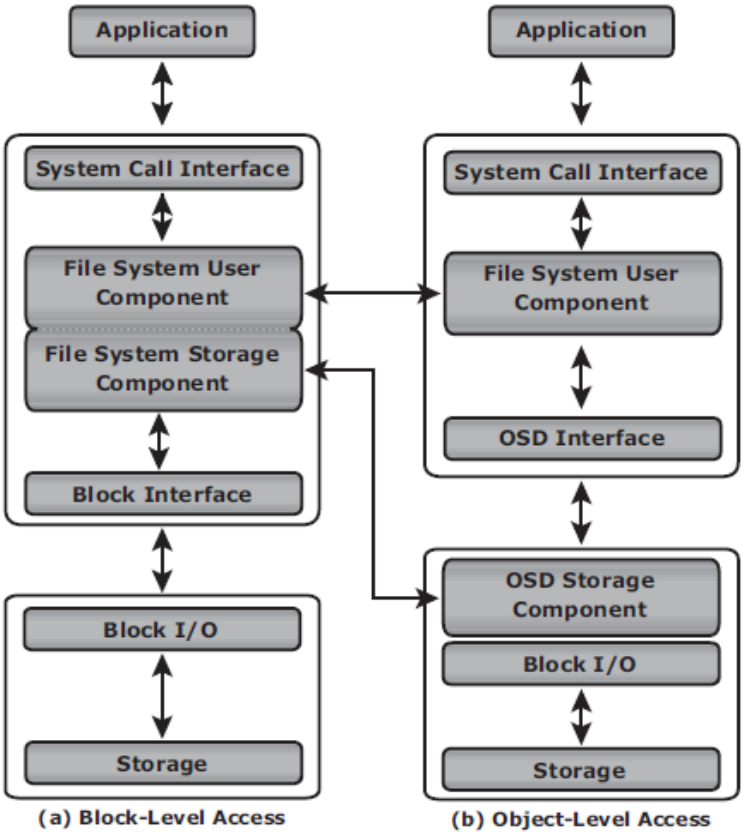


Fig 2.37 Block-level access versus object-level access

2.14.2 Components of OSD

- The OSD system is typically composed of three key components:
    1. nodes
    2. private network
    3. storage
- Fig 2.38 illustrates the components of OSD.
- A **node** is a server that runs the *OSD operating environment* and provides *services* to store, retrieve, and manage data in the system. The OSD system is composed of one or more nodes.
- The OSD node has *two* key services:
    1. The ***metadata service*** is responsible for generating the object ID from the contents of a file. It also maintains the mapping of the object IDs and the file system namespace.
    2. The ***storage service*** manages a set of disks on which the user data is stored.
- The OSD nodes connect to the storage via an **internal network (private network)**. The internal network provides node-to-node connectivity and node-to-storage connectivity.
- The application server accesses the node to store and retrieve data over an *external network*.
- For **storage,** OSD typically uses low-cost and high-density disk drives to store the objects. As more capacity is required, more disk drives can be added to the system.

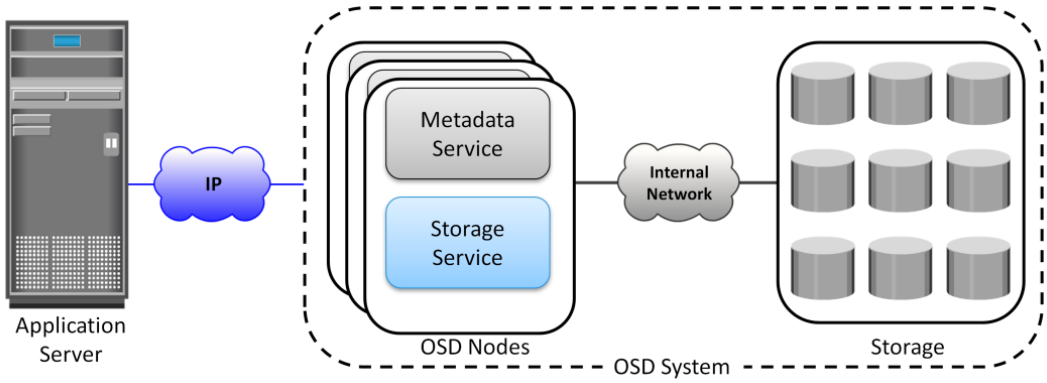


Fig 2.38: OSD Components

2.14.3 Object Storage and Retrieval in OSD

- The process of storing objects in OSD is illustrated in Fig 2.39.
- The data storage process in an OSD system is as follows:

1. The application server presents the file to be stored to the OSD node.
2. The OSD node divides the file into two parts: **user data** and **metadata**.
3. The OSD node generates the **object ID** using a specialized algorithm. The algorithm is executed against the contents of the user data to derive an ID unique to this data.
4. For future access, the OSD node stores the metadata and object ID using the ***metadata service***.
5. The OSD node stores the user data (objects) in the storage device using the ***storage service***.
6. An acknowledgment is sent to the application server stating that the object is stored.

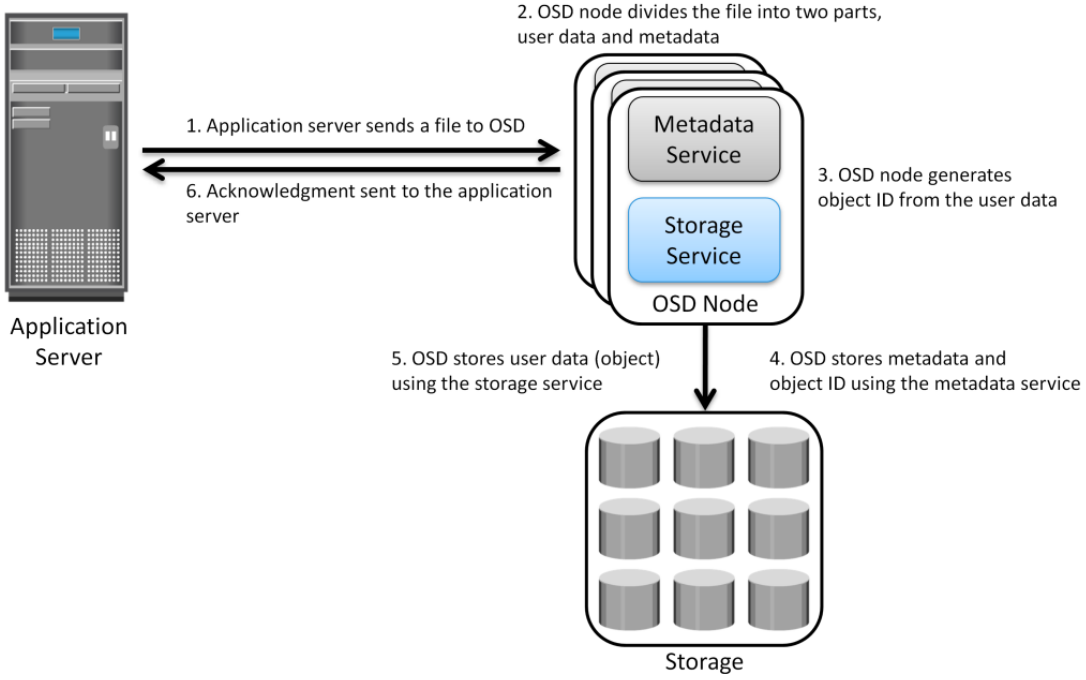


Fig 2.39: Storing objects on OSD

- A user accesses the data stored on OSD by the same filename.
- The application server retrieves the stored content using the **object ID**. This process is transparent to the user.
- The process of retrieving objects in OSD is illustrated in Fig 2.40. The process of data retrieval from OSD is as follows:

1. The application server sends a ***read request*** to the OSD system.
2. The metadata service retrieves the object ID for the requested file.
3. The metadata service sends the object ID to the application server.
4. The application server sends the object ID to the OSD storage service for object retrieval.
5. The OSD storage service retrieves the object from the storage device.
6. The OSD storage service sends the file to the application server.

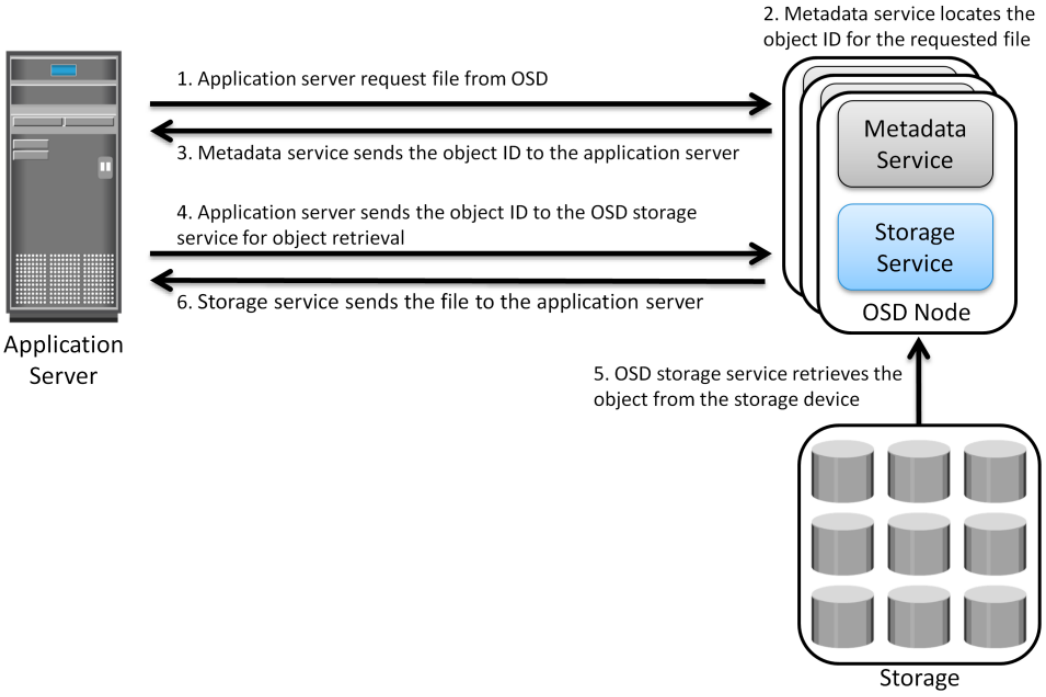


Fig 2.40: Object retrieval from an OSD system

2.14.4 Benefits of Object-Based Storage

- The key benefits of object-based storage are as follows:
    - **Security and reliability:** Data integrity and content authenticity are the key features of object-based storage devices. OSD uses *specialized algorithms* to create objects that provide strong data encryption capability. In OSD, request authentication is performed at the storage device rather than with an external authentication mechanism.
    - **Platform independence:** Objects are abstract containers of data, including metadata and attributes. This feature allows objects to be shared across heterogeneous platforms locally or remotely. This makes object-based storage the best candidate for cloud computing environments.
    - **Scalability:** Due to the use of flat address space, object-based storage can handle large amounts of data without impacting performance. Both storage and OSD nodes can be scaled independently in terms of performance and capacity.
    - **Manageability:** Object-based storage has an inherent intelligence to manage and protect objects. It uses self-healing capability to protect and replicate objects. Policy-based management capability helps OSD to handle routine jobs automatically.

2.14.5 Common Use Cases for Object-Based Storage

- A **data archival solution** is a promising use case for OSD. Data integrity and protection is the primary requirement for any data archiving solution. Traditional archival solutions - CD and DVD-ROM - do not provide scalability and performance. OSD stores data in the form of objects, associates them with a unique object ID, and ensures high data integrity. Along with integrity, it provides scalability and data protection. These capabilities make OSD a viable option for long term data archiving for fixed content.
- Cloud-based storage is another use case of OSD. OSD uses a web interface to access storage resources. OSD provides inherent security, scalability, and automated data management. It also enables data sharing across heterogeneous platforms or tenants while ensuring integrity of data.

These capabilities make OSD a strong option for cloud-based storage. Cloud service providers can leverage OSD to offer storage-as-a-service.

- OSD supports web service access via **representational state transfer (REST)** and **simple object access protocol (SOAP)**.
- REST and SOAP APIs can be easily integrated with business applications that access OSD over the web.

2.15 Unified Storage

- Unified storage consolidates *block, file,* and *object* access into one storage solution.
- It supports multiple protocols, such as CIFS, NFS, iSCSI, FC, FCoE, REST (representational state transfer), and SOAP (simple object access protocol).

2.15.1 Components of Unified Storage

- A unified storage system consists of the following key components:
  - storage controller,
  - NAS head,
  - OSD node,
  - storage.
- Fig 2.41 illustrates the block diagram of a unified storage platform.
- The **storage controller or storage processor** provides block-level access to application servers through iSCSI, FC, or FCoE protocols. It contains the corresponding front-end ports for direct block access. The storage controller is also responsible for managing the back-end storage pool in the storage system.
- The controller configures LUNs and presents them to application servers, NAS heads, and OSD nodes. The LUNs presented to the application server appear as local physical disks. A file system is configured on these LUNs and is made available to applications for storing data.

- **A NAS head** is a dedicated file server that provides file access to NAS clients. The NAS head is connected to the storage via the storage controller typically using a FC or FCoE connection. The system typically has two or more NAS heads for redundancy.
- The **LUNs** presented to the NAS head appear as physical disks. The NAS head configures the file systems on these disks, creates a NFS, CIFS, or mixed share, and exports the share to the NAS clients.
- **The OSD node** also accesses the storage through the storage controller using a FC or FCoE connection.
- The LUNs assigned to the OSD node appear as physical disks. These disks are configured by the OSD nodes, enabling them to store the data from the web application servers.

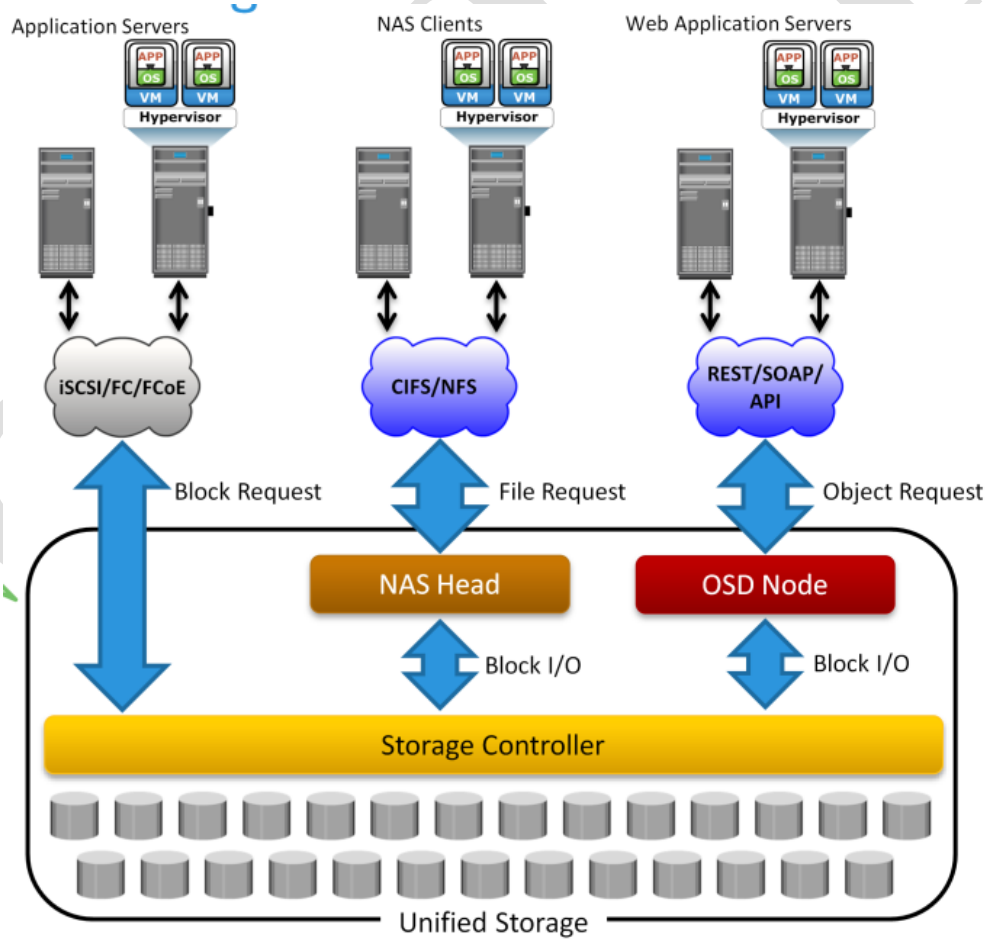


Fig 2.41 Unified Storage Platform

Data Access from Unified Storage

- In a unified storage system, block, file, and object requests to the storage travel through different I/O paths. Fig 2.41 also illustrates the different I/O paths for block, file, and object access.
- **Block I/O request:** The application servers are connected to an FC, iSCSI, or FCoE port on the storage controller. The server sends a block request and the storage processor (SP) processes the I/O and responds to the application server.
- **File I/O request:** The NAS clients send a file request to the NAS head using NFS or CIFS protocol. The NAS head receives the request, converts it into a block request, and forwards it to the storage controller. Upon receiving the block data, the NAS head again converts the block request back to the file request and sends back it to the clients.
- **Object I/O request:** The web application servers send an object request, typically using REST or SOAP protocols, to the OSD node. The OSD node receives the request, converts it into a block request, and sends it to the disk through the storage controller. The controller in turn processes the block request and responds back to the OSD node, which in turn provides the requested object to the web application server.