10

The SNIA Shared Storage Model

The fact that there is a lack of any unified terminology for the description of storage architectures has already become apparent at several points in previous chapters. There are thus numerous components in a storage network which, although they do the same thing, are called by different names. Conversely, there are many systems with the same name, but fundamentally different functions.

A notable example is the term 'data mover' relating to server-free back-up (Section 7.8.1) in storage networks. When this term is used it is always necessary to check whether the component in question is one that functions in the sense of the 3rd-party SCSI Copy Command for, for example, a software component of back-up software on a special server, which implements the server-free back-up without 3rd-party SCSI.

This example shows that the type of product being offered by a manufacturer and the functions that the customer can ultimately expect from this product are often unclear. This makes it difficult for customers to compare the products of individual manufacturers and find out the differences between the alternatives on offer. There is no unified model for this with clearly defined descriptive terminology.

For this reason, in 2001 the Technical Council of the Storage Networking Industry Association (SNIA) introduced the so-called Shared Storage Model in order to unify the terminology and descriptive models used by the storage network industry. Ultimately, the SNIA wants to use the SNIA Shared Storage Model to establish a reference model, which will have the same importance for storage architectures as the seven-tier OSI model has for computer networks.

In this chapter, we would first like to introduce the disk-based Shared Storage Model (Section 10.1) and then show, based upon examples (Section 10.2), how the model can be

Storage Networks Explained U. Troppens R. Erkens W. Müller
 ISBN: 0-470-86182-7

used for the description of typical disk storage architectures. In Section 10.3 we introduce the extension of the SNIA model to the description of tape functions. We then discuss examples of tape-based back-up architectures (Section 10.4). Whilst describing the SNIA Shared Storage Model we often refer to text positions in this book where the subject in question is discussed in detail, which means that this chapter also serves as a summary of the entire book.

10.1 THE MODEL

In this book we have spoken in detail about the advantages of the storage-centric architecture in relation to the server-centric architecture. The SNIA sees its main task as being to communicate this paradigm shift and to provide a forum for manufacturers and developers so that they can work together to meet the challenges and solve the problems in this field. In the long run, an additional reason for the development of the Shared Storage Model by SNIA was the creation of a common basis for communication between the manufacturers who use the SNIA as a platform for the exchange of ideas with other manufacturers. Storage-centric IT architectures are called shared storage environments by the SNIA. We will use both terms in the following.

First of all, we will describe the functional approach of the SNIA model (Section 10.1.1) and the SNIA conventions for graphical representation (Section 10.1.2). We will then consider the model (Section 10.1.3), its components (Section 10.1.4) and the layers 'file/record layer' and 'block layer' in detail (Section 10.1.5 to Section 10.1.8). Then we will introduce the definitions and representation of concepts from the SNIA model, such as access paths (Section 10.1.9), caching (Section 10.1.10), access control (Section 10.1.11), clustering (Section 10.1.12), data (Section 10.1.13) and resource and data sharing (Section 10.1.14). Finally, we will take a look at the service subsystem (Section 10.1.15).

10.1.1 The functional approach

The SNIA Shared Storage Model first of all describes functions that have to be provided in a storage-centric IT architecture. This includes, for example, the block layer or the file/record layer. The SNIA model describes both the tasks of the individual functions and also their interaction. Furthermore, it introduces components such as server ('host computer') and storage networks ('interconnection network').

Due to the separation of functions and components, the SNIA Shared Storage Model is suitable for the description of various architectures, specific products and concrete installations. The fundamental structures, such as the functions and services of a shared storage environment, are highlighted. In this manner, functional responsibilities can be

assigned to individual components and the relationships between control and data flows in the storage network worked out. At the same time, the preconditions for interoperability between individual components and the type of interoperability can be identified. In addition to providing a clear terminology for the elementary concepts, the model should be simple to use and, at the same time, extensive enough to cover a large number of possible storage network configurations.

The model itself describes, on the basis of examples, possible practicable storage architectures and their advantages and disadvantages. We will discuss these in Section 10.2 without evaluating them or showing any preference for specific architectures. Within the model definition, however, only a few selected examples will be discussed in order to highlight how the model can be applied for the description of storage-centred environments and further used.

10.1.2 Graphical representations

The SNIA Shared Storage Model further defines how storage architectures can be graphically illustrated. Physical components are always represented as three-dimensional objects, whilst functional units should be drawn in two-dimensional form. The model itself also defines various colours for the representation of individual component classes. In the black and white format of the book, we have imitated these using shades of grey. A coloured version of the illustrations to this chapter can be found on our home page http://www.storage-explained.com. Thick lines in the model represent the data transfer, whereas thin lines represent the metadata flow between the components.

10.1.3 An elementary overview

The SNIA Shared Storage Model first of all defines four elementary parts of a shared storage environment (Figure 10.1):

1. File/record layer
   The file/record layer is made up of database and file system.
2. Block layer
   The block layer encompasses the storage devices and the block aggregation. The SNIA Shared Storage Model uses the term ‘aggregation’ instead of the often ambiguously used term ‘storage virtualization’. In Chapter 5, however, we used the term ‘storage virtualization’ to mean the same thing as ‘aggregation’ in the SNIA model, in order to avoid ambiguity.
3. Services subsystem
   The functions for the management of the other components are defined in the services subsystem.

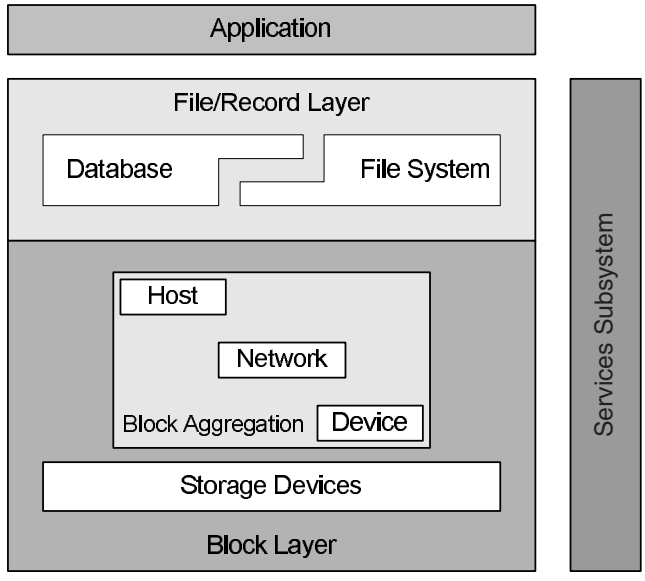

Figure 10.1 The main components of the SNIA Shared Storage Model are the file/record layer, the block layer and the services subsystem. Applications are viewed as users of the model

4. Applications
   Applications are not discussed further by the model. They will be viewed as users of the model in the widest sense.

10.1.4 The components

The SNIA Shared Storage Model defines the following components:

- Interconnection network
  The interconnection network represents the storage network, i.e. the infrastructure, that connects the individual elements of a shared storage environment with one another. The interconnection network can be used exclusively for storage access, but it can also be used for other communication services. Our definition of a storage network (Section 1.2) is thus narrower than the definition of the interconnection network in the SNIA model.

  The network must always provide a high-performance and easily scaleable connection for the shared storage environment. In this context, the structure of the interconnection network – for example redundant data paths between two components to increase fault-tolerance – remains just as open as the network techniques used. It is therefore a prerequisite of the model that the components of the shared storage environment are connected over a network without any definite communication protocols or transmission techniques being specified.

In actual architectures or installations, Fibre Channel, Fast Ethernet, Gigabit Ethernet, InfiniBand and many other transmission techniques are used (Chapter 3). Communication protocols such as SCSI, Fibre Channel FCP, TCP/IP, RDMA, CIFS or NFS are based upon these.

- Host computer
  Host computer is the term used for computer systems that draw at least some of their storage from the shared storage environment. According to SNIA, these systems were often omitted from classical descriptive approaches and not viewed as part of the environment. The SNIA shared storage model, however, views these systems as part of the entire shared storage environment because storage-related functions can be implemented on them.

  Host computers are connected to the storage network via host bus adapters or network cards, which are operated by means of their own drivers and software. Drivers and software are thus taken into account in the SNIA Shared Storage Model. Host computers can be operated fully independently of one another or they can work on the resources of the storage network in a compound, for example, a cluster (Section 6.4.1).
- Physical storage resource
  All further elements that are connected to the storage network and are not host computers are known by the term 'physical storage resource'. This includes simple hard disk drives, disk arrays, disk subsystems and controllers plus tape drives and tape libraries. Physical storage resources are protected against failures by means of redundant data paths (Section 6.3.1), replication functions such as snapshots and mirroring (Section 2.7) and RAID (Section 2.5).
- Storage device
  A storage device is a special physical storage resource that stores data.
- Logical storage resource
  The term 'logical storage resource' is used to mean services or abstract compositions of physical storage resources, storage management functions or a combination of these. Typical examples are volumes, files and data movers.
- Storage management functions
  The term 'storage management function' is used to mean the class of services that monitor and check (Chapter 8) the shared storage environment or implement logical storage resources. These functions are typically implemented by software on physical storage resources or host computers.

10.1.5 The layers

The SNIA Shared Storage Model defines four layers (Figure 10.2):

I. Storage devices
II. Block aggregation layer

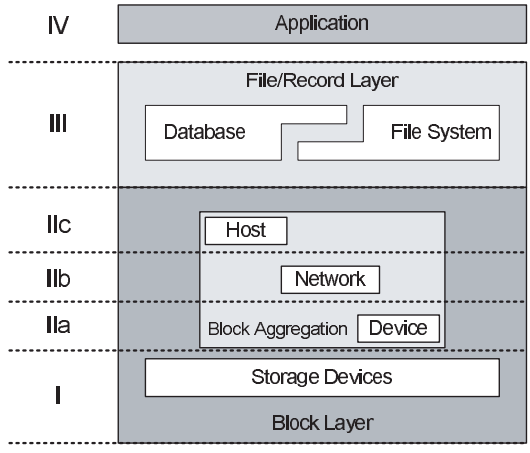

Figure 10.2 The SNIA Shared Storage Model defines four layers

III. File/record layer
  IIIb. Database
  IIIa. File system
IV Applications

Applications are viewed as users of the model and are thus not described in the model. They are, however, implemented as a layer in order to illustrate the point in the model to which they are linked. In the following we'll consider the file/record layer (Section 10.1.6), the block layer (Section 10.1.7) and the combination of both (Section 10.1.8) in detail.

10.1.6 The file/record layer

The file/record layer maps database records and files on the block-oriented volume of the storage devices. Files are made up of several bytes and are therefore viewed as byte vectors in the SNIA model. Typically, file systems or database management systems take over these functions. They operate directories of the files or records, check the access, allocate storage space and cache the data (Chapter 4). The file/record layer thus works on volumes that are provided to it from the block layer below. Volumes themselves consist of several arranged blocks, so-called block vectors. Database systems map one or more records, so-called tuple of records, onto volumes via tables and table spaces:

$$\text{Tuple of records} \longrightarrow \text{tables} \longrightarrow \text{table spaces} \longrightarrow \text{volumes}$$

In the same way, file systems map bytes onto volumes by means of files:

$$\text{Bytes} \longrightarrow \text{files} \longrightarrow \text{volumes}$$

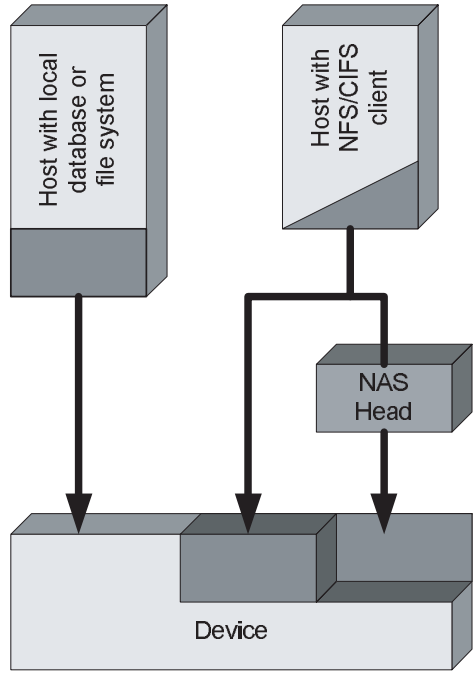

Figure 10.3 The functions of the file/record layer can be implemented exclusively on the host or distributed over a client and a server component

Some database systems can also work with files, i.e. byte vectors. In this case, block vectors are grouped into byte vectors by means of a file system – an additional abstraction level. Since an additional abstraction level costs performance, only smaller databases work in a file-oriented manner. In large databases the additional mapping layer of byte to block vectors is dispensed with for performance reasons.

The functions of the file/record layers can be implemented at various points (Figure 10.3, Section 5.6):

- Exclusively on the host
  In this case, the file/record layer is implemented entirely on the host. Databases and the host-based file systems work in this way.
- Both in the client and also on a server component
  The file/record layer can also be implemented in a distributed manner. In this case the functions are distributed over a client and a server component. The client component is realized on a host computer, whereas the server component can be realized on the following devices:
  - NAS/file server
    A NAS/file server is a specialized host computer usually with a locally connected, dedicated storage device (Section 4.2.2).

- NAS head
  A host computer that offers the file serving services, but which has access to external storage connected via a storage network. NAS heads correspond with the devices called NAS gateways in our book (Section 4.2.2).

In this case, client and server components work over network file systems such as NFS or CIFS (Section 4.2).

10.1.7 The block layer

The block layer differentiates between block aggregation and the block-based storage devices. The block aggregation in the SNIA model corresponds to our definition of the virtualization on block level (Section 5.5). SNIA thus uses the term 'block aggregation' to mean the aggregation of physical blocks or block vectors into logical blocks or block vectors.

To this end, the block layer maps the physical blocks of the disk storage devices onto logical blocks and makes these available to the higher layers in the form of volumes (block vectors). This either occurs via a direct (1 : 1) mapping, or the physical blocks are first aggregated into logical blocks, which are then passed on to the upper layers in the form of volumes (Figure 10.4). In the case of SCSI, the storage devices of the storage device layer exist in the form of one or more so-called logical units (LU).

Further tasks of the block layer are the labelling of the logical units using so-called logical unit numbers (LUNs), caching and – increasingly in the future – access control. Block aggregation can be used for various purposes, for example:

- Volume/space management
  The typical task of a volume manager is to aggregate several small block vectors to form one large block vector. On SCSI level this means aggregating several logical units

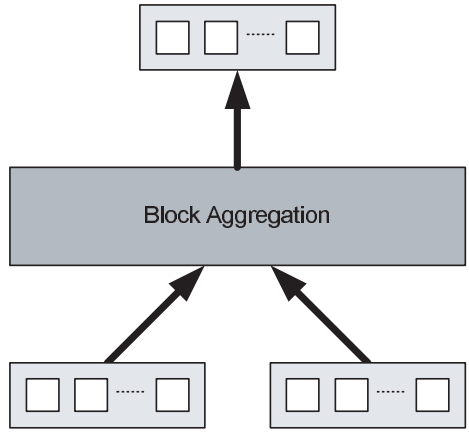

Figure 10.4 The block aggregation layer aggregates physical blocks or block vectors into logical blocks or block vectors

to form a large volume, which is passed on to the upper layers such as the file/record layer (Section 4.1.4).

- Striping
  In striping, physical blocks of different storage devices are aggregated to one volume. This increases the I/O throughput of the read and write operations, since the load is distributed over several physical storage devices (Section 2.5.1).
- Redundancy
  In order to protect against failures of physical data carriers, RAID (Section 2.5) and remote mirroring (Section 2.7.2) are used. Snapshots (instant copies) can also be used for the redundant storage of data (Section 2.7.1).

The block aggregation functions of the block layer can be realized at different points of the shared storage environment (Section 5.6):

- On the host
  Block aggregation on the host is encountered in the form of a logical volume manager software, in device drivers and in host bus adapters.
- On a component of the storage network
  The functions of the block layer can also be realized in connection devices of the storage network or in specialized servers in the network.
- In the storage device
  Most commonly, the block layer functions are implemented in the storage devices themselves, for example, in the form of RAID or volume manager functionality.

In general, various block aggregation functions can be combined at different points of the shared storage environment. In practical use, RAID may, for example, be used in the disk subsystem with additional mirroring from one disk subsystem to another via the volume manager on the host computer (Section 4.1.4). In this setup, RAID protects against the failure of physical disks of the disk subsystem, whilst the mirroring by means of the volume manager on the host protects against the complete failure of a disk subsystem. Furthermore, the performance of read operations is increased in this set-up, since the volume manager can read from both sides of the mirror (Section 2.5.2).

10.1.8 Combination of the block and file/record layers

Figure 10.5 shows how block and file/record layer can be combined and represented in the SNIA shared storage model:

- Direct attachment
  The left-hand column in the figure shows storage connected directly to the server, as is normally the case in a server-centric IT architecture (Section 1.1).

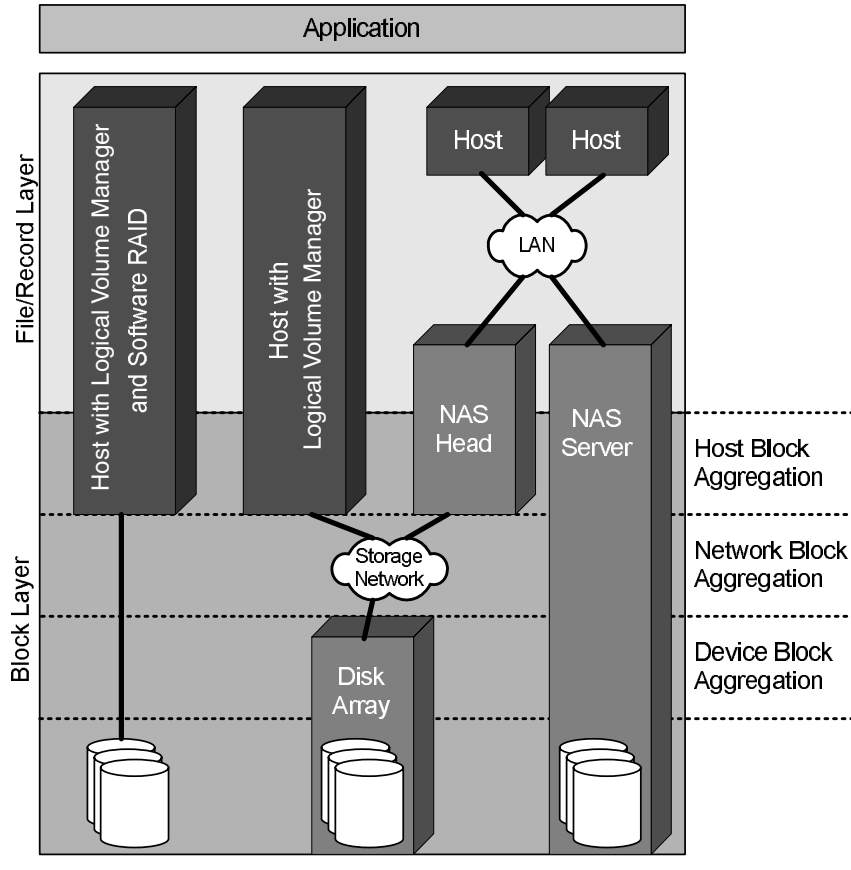

Figure 10.5 Block and file/record layer can be combined in different ways

- Storage network attachment
  In the second column we see how a disk array is normally connected via a storage network in a storage-centric IT architecture, so that it can be accessed by several host computers (Section 1.2).
- NAS head (NAS gateway)
  The third column illustrates how a NAS head is integrated into a storage network between SAN storage and a host computer connected via LAN.
- NAS server
  The right-hand column shows the function of a NAS server with its own dedicated storage in the SNIA Shared Storage Model.

10.1.9 Access paths

Read and write operations of a component on a storage device are called access paths in the SNIA Shared Storage Model. An access path is descriptively defined as the list

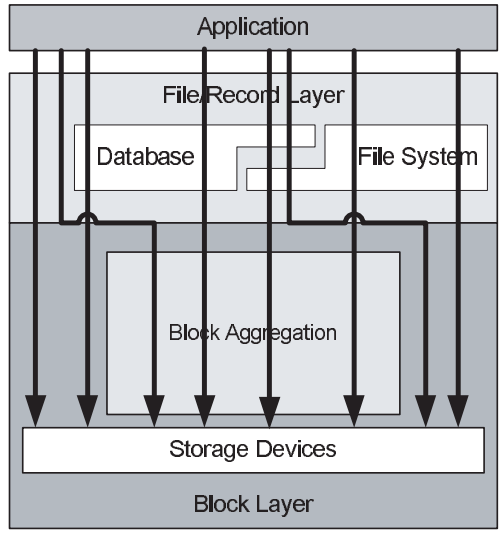

Figure 10.6 In the SNIA Shared Storage Model, applications can access the storage devices via eight possible access paths

of components that are run through by read and write operations to the storage devices and responses to them. If we exclude cyclical access paths, then a total of eight possible access paths from applications to the storage devices can be identified in the SNIA Shared Storage Model (Figure 10.6):

1. Direct access to a storage device.
2. Direct access to a storage device via a block aggregation function.
3. Indirect access via a database system.
4. Indirect access via a database system based upon a block aggregation function.
5. Indirect access via a database system based upon a file system.
6. Indirect access via a database system based upon a file system, which is itself based upon a block aggregation function.
7. Indirect access via a file system.
8. Indirect access via a file system based upon a block aggregation function.

10.1.10 Caching

Caching is the method of shortening the access path of an application – i.e. the number of the components to be passed through – to frequently used data on a storage device. To this end, the data accesses to the slower storage devices are buffered in a faster cache storage. Most components of a shared storage environment can have a cache. The cache can be implemented within the file/record layer, within the block layer or in both.

In practice, several caches working simultaneously on different levels and components are generally used. For example, a read cache in the file system may be combined with a write cache on a disk array and a read cache with pre-fetching on a hard disk (Figure 10.7). In addition, a so-called cache-server (Section 5.7.2), which temporarily stores data for other components on a dedicated basis in order to reduce the need for network capacity or to accelerate access to slower storage, can also be integrated into the storage network.

However, the interaction between several cache storages on several components means that consideration must be given to the consistency of data. The more components that use cache storage, the more dependencies arise between the functions of individual components. A classic example is the use of a snapshot function on a component in the block layer, whilst another component stores the data in question to cache in the file/record layer. In this case, the content of the cache within the file/record layer, which we will assume to be consistent, and the content of a volume on a disk array that is a component of the block layer can be different. The content of the volume on the array is thus inconsistent. Now, if a snapshot is taken of the volume within the disk array, a virtual

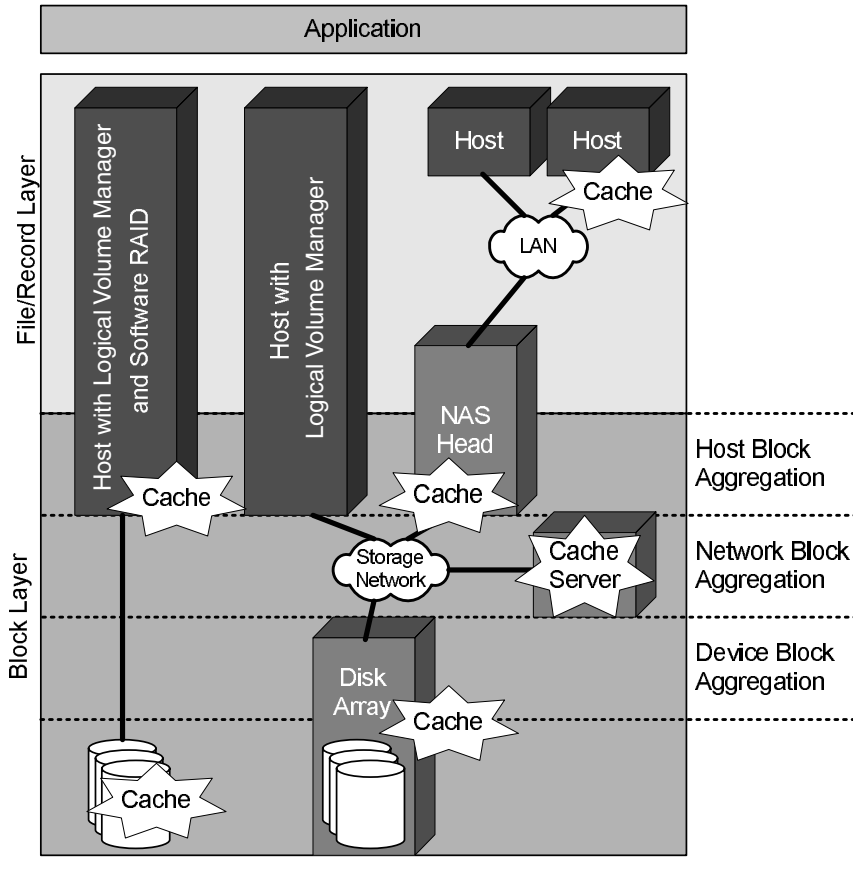

Figure 10.7 Caching functions can be implemented at different levels and at different components of a shared storage environment

copy is obtained of an inconsistent state of the data. The copy is thus unusable. Therefore, before the snapshot is made within the block layer, the cache in the file/record layer on the physical volume must be destaged, so that it can receive a consistent copy later.

10.1.11 Access control

Access control is the name for the technique that arranges the access to data of the shared storage environment. The term access control should thus be clearly differentiated from the term access path, since the mere existence of an access path does not include the right to access. Access control has the following main objectives:

- Authentication
  Authentication establishes the identity of the source of an access.
- Authorization
  Authorization grants or refuses actions to resources.
- Data protection
  Data protection guarantees that data may only be viewed by authorized persons.

All access control mechanisms ultimately use a form of secure channel between the data on the storage device and the source of an access. In its simplest form, this can be a check to establish whether a certain host is permitted to have access to a specific storage device.

Access control can, however, also be achieved by complicated cryptographic procedures, which are secure against the most common external attacks. When establishing a control mechanism it is always necessary to trade off the necessary protection and efficiency against complexity and performance sacrifices.

In server-centric IT architectures, storage devices are protected by the guidelines on the host computers and by simple physical measures. In a storage network, the storage devices, the network and the network components themselves must be protected against unauthorized access, since in theory they can be accessed from all host computers. Access control becomes increasingly important in a shared storage environment as the number of components used, the diversity of heterogeneous hosts and the distance between the individual devices rise.

Access controls can be established at the following points of a shared storage environment:

- On the host
  In shared storage environments, access controls comparable with those in server-centric environments can be established at host level. The disadvantage of this approach is, however, that the access rights have to be set on all host computers. Mechanisms that reduce the amount of work by the use of central instances for the allocation and distribution of rights must be suitably protected against unauthorized access. Database systems and file systems can be protected in this manner. Suitable mechanisms for the block layer are currently being planned. The use of encryption technology for the host's network protocol stack is in conflict with performance requirements. Suitable

offload engines, which process the protocol stack on the host bus adapter themselves, are available for some protocols.

- In the storage network
  Security within the storage network is achieved in Fibre Channel SANs by zoning and virtual storage networks (Virtual SAN (VSAN), Section 3.4.2) and in Ethernet-based storage networks by so-called virtual LANs (VLAN). This is always understood to be the subdivision of a network into virtual subnetworks, which permit communication between a number of host ports and certain storage device ports. These guidelines can, however, also be defined on finer structures than ports.
- On the storage device
  The normal access control procedure on SAN storage devices is the so-called LUN masking, in which the LUNs that are visible to a host are restricted. Thus, the computer sees only those LUNs that have been assigned to it by the storage device (Section 2.7.3).

10.1.12 Clustering

A cluster is defined in the SNIA Shared Storage Model as a combination of resources with the objective of increasing scalability, availability and management within the shared storage environment (Section 6.4.1). The individual nodes of the cluster can share their resources via distributed volume managers (multi-node LVM) and cluster file systems (Figure 10.8, Section 4.3).

10.1.13 Storage, data and information

The SNIA Shared Storage Model differentiates strictly between storage, data and information. Storage is space – so-called containers – provided by storage units, on which the data is stored. The bytes stored in containers on the storage units are called data. Information is the meaning – the semantics – of the data. The SNIA Shared Storage Model names the following examples in which data–container relationships arise (Table 10.1).

10.1.14 Resource and data sharing

In a shared storage environment, in which the storage devices are connected to the host via a storage network, every host can access every storage device and the data stored upon it (Section 1.2). This sharing is called resource sharing or data sharing in the SNIA model, depending upon the level at which the sharing takes place (Figure 10.9).

If exclusively the storage systems – and not their data content – are shared, then we talk of resource sharing. This is found in the physical resources, such as disk subsystems and tape libraries, but also within the network.

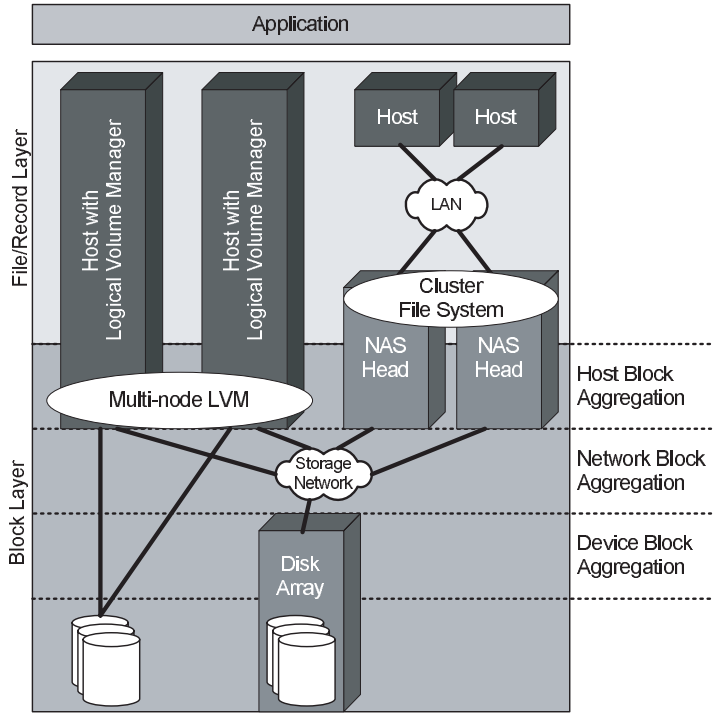

Figure 10.8 Nodes of a cluster share resources via distributed volume managers or cluster file systems

Table 10.1 Data–container relationships

Relationship	Role	Remark
User	Data	Inputs via keyboard
Application	Container	Input buffer
User	Data	Input buffer file
File system	Container	Byte vector
File system	Data	A file
Volume manager	Container	Blocks of a volume
Volume manager	Data	Mirrored stripe set
Disk array	Container	Blocks of a logical unit

Data sharing denotes the sharing of data between different hosts. Data sharing is significantly more difficult to implement, since the shared data must always be kept consistent, particularly when distributed caching is used.

Heterogeneous environments also require additional conversion steps in order to convert the data into a format that the host can understand. Protocols such as NFS or

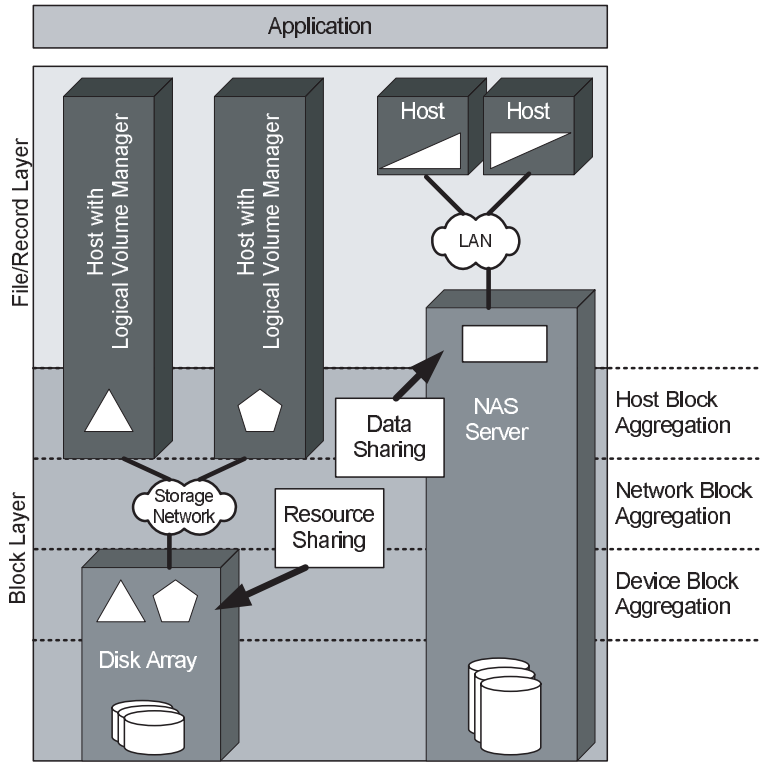

Figure 10.9 In resource sharing, hosts share the physical resources – in this case a disk array – which make a volume available to each host. In data sharing, hosts access the same data – in this case the NAS server and its data

CIFS are used in the more frequently used data sharing within the file/record layers (Section 4.2).

For data sharing in the block layer, server clusters with shared disk file systems or parallel databases are used (Section 4.3, Section 6.2.3).

10.1.15 The service subsystem

Up to now we have concerned ourselves with the concepts within the layers of the SNIA Shared Storage Model. Let us now consider the service subsystem (Figure 10.10). Within the service subsystem we find the management tasks which occur in a shared storage environment and which we have, for the most part, already discussed in Chapter 8.

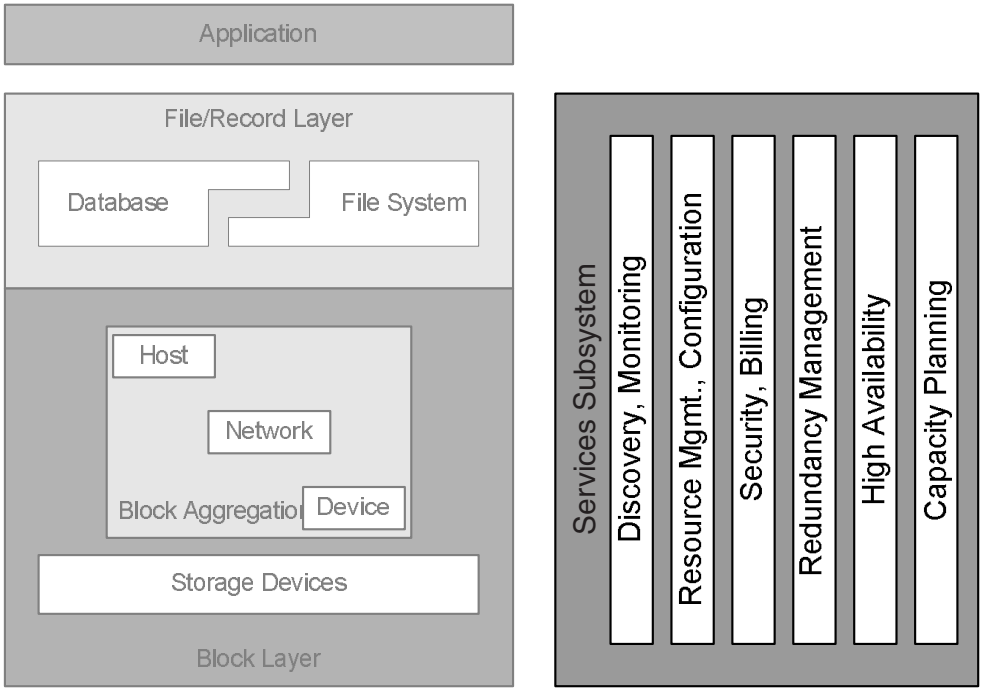

Figure 10.10 In the services subsystem, the SNIA defines the management tasks in a shared storage environment

In this connection, the SNIA Technical Council mention:

- discovery and monitoring
- resource management
- configuration
- security
- billing (charge-back)
- redundancy management, for example, by network back-up
- high availability
- capacity planning.

The individual subjects are not yet dealt with in more detail in the SNIA Shared Storage Model, since the required definitions, specifications and interfaces are still being developed (Section 8.7.3). At this point we expressly refer once again to the check list in the Appendix B, which reflects a cross-section of the questions that crop up here.

10.2 EXAMPLES OF DISK-BASED STORAGE ARCHITECTURES

In this section we will present a few examples of typical storage architectures and their properties, advantages and disadvantages, as they are represented by the SNIA in the Shared Storage Model. First of all, we will discuss block-based architectures, such as the direct connection of storage to the host (Section 10.2.1), connection via a storage network (Section 10.2.2), symmetric and asymmetric storage virtualization in the network (Section 10.2.3 and Section 10.2.4) and a multi-site architecture such as is used for data replication between several locations (Section 10.2.5). We then move on to the file/record layer and consider the graphical representation of a file server (Section 10.2.6), a NAS head (Section 10.2.7), the use of metadata controllers for asymmetric file level virtualization (Section 10.2.8) and an object-based storage device (OSD), in which the position data of the files and their access rights is moved to a separate device, a solution that combines file sharing with increased performance due to direct file access and central metadata management of the files (Section 10.2.9).

10.2.1 Direct attached block storage

Figure 10.11 shows the direct connection from storage to the host in a server-centric architecture. The following properties are characteristic of this structure:

- No connection devices, such as switches or hubs, are needed.
- The host generally communicates with the storage device via a protocol on block level.
- Block aggregation functions are possible both in the disk subsystem and on the host.

10.2.2 Storage network attached block storage

The connection from storage to host via a storage network can be represented in the Shared Storage Model as shown in Figure 10.12. In this case:

- Several hosts share several storage devices.
- Block-oriented protocols are generally used.
- Block aggregation can be used in the host, in the network and in the storage device.

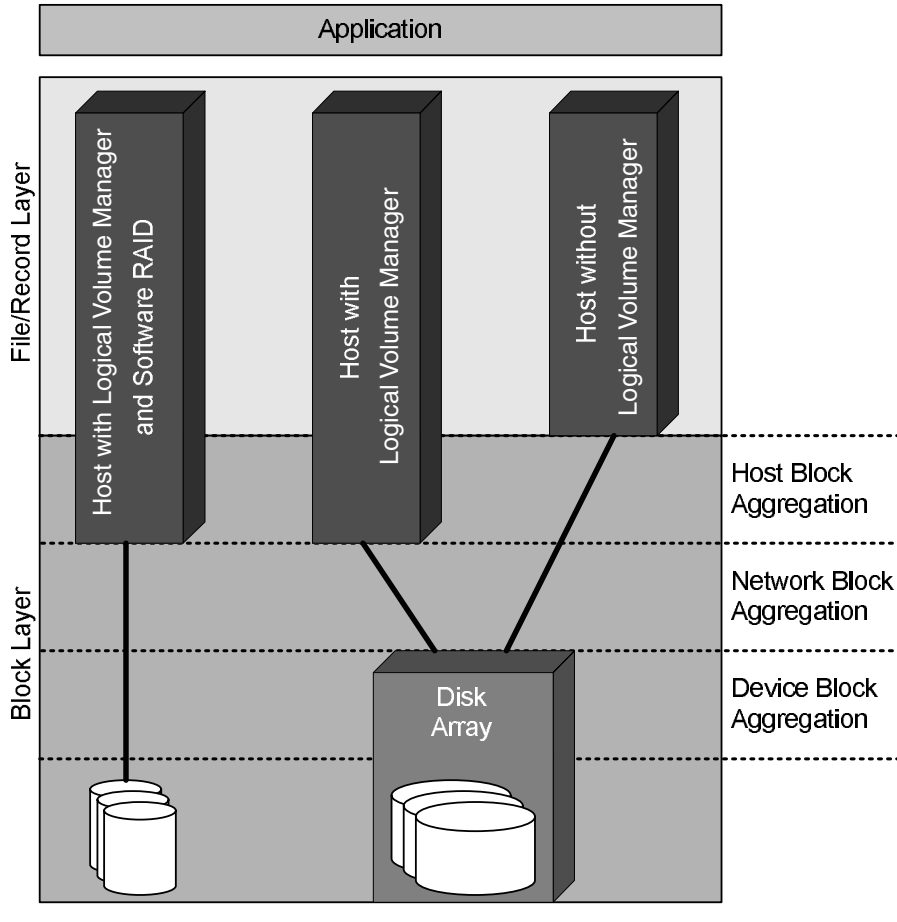

Figure 10.11 In direct attachment, hosts are connected to storage devices directly without connection devices such as switches or hubs. Joint use of data or resources is not possible without additional software

10.2.3 Block storage aggregation in a storage device: SAN appliance

Block aggregation can also be implemented in a specialized device or server of the storage network in the data path between hosts and storage devices, as in the symmetric storage virtualization (Figure 10.13, Section 5.7.1). In this approach:

- Several hosts and storage devices are connected via a storage network.
- A device or a dedicated server – a so-called SAN appliance – is placed in the data path between hosts and storage devices to perform block aggregation, and data and metadata traffic flows through this.

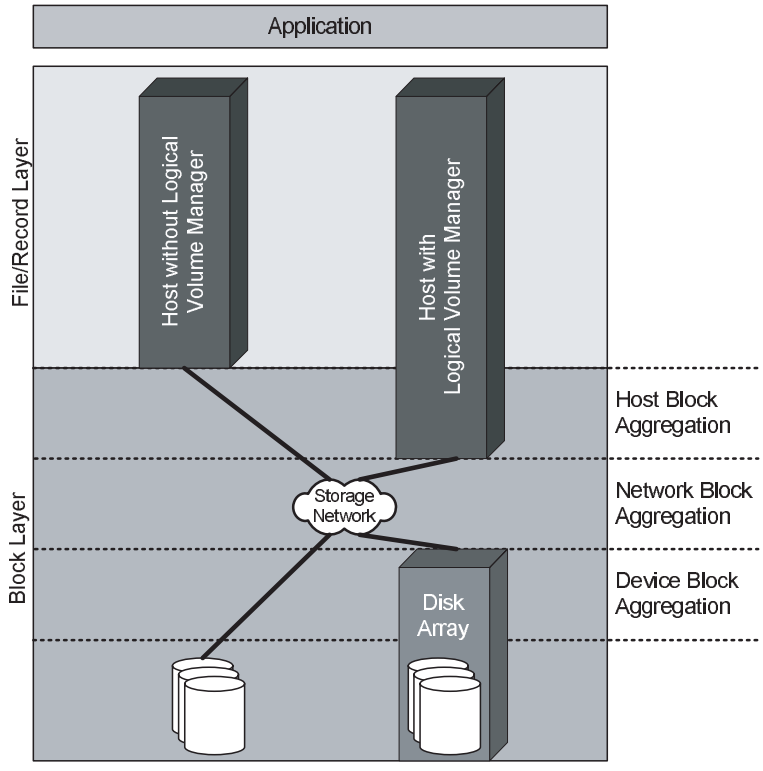

Figure 10.12 In storage connected via a storage network, several hosts share the storage devices, which are accessed via block-oriented protocols

10.2.4 Network attached block storage with metadata server: asymmetric block services

The asymmetric block services architecture is identical to the asymmetric storage virtualization approach (Figure 10.14, Section 5.7.2):

- Several hosts and storage devices are connected over a storage network.
- Host and storage devices communicate with each other over a protocol on block level.
- The data flows directly between hosts and storage devices.
- A metadata server outside the data path holds the information regarding the position of the data on the storage devices and maps between logical and physical blocks.

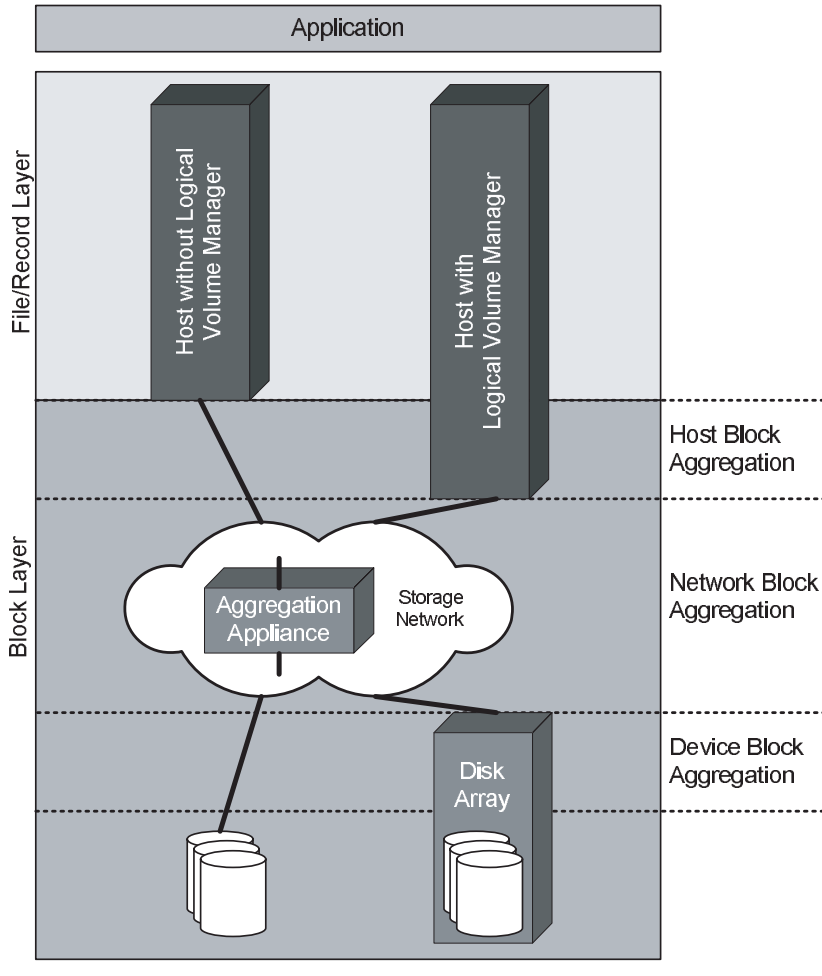

Figure 10.13 In block aggregation on a specialized device or server in the storage network, a SAN appliance maps between logical and physical blocks in the data path in the same way as symmetric virtualization

10.2.5 Multi-site block storage

Figure 10.15 shows how data replication between two locations can be implemented by means of WAN techniques. The data can be replicated on different layers of the model using different protocols:

- between volume managers on the host;
- between specialized devices in the storage network; or
- between storage systems, for example disk subsystems.

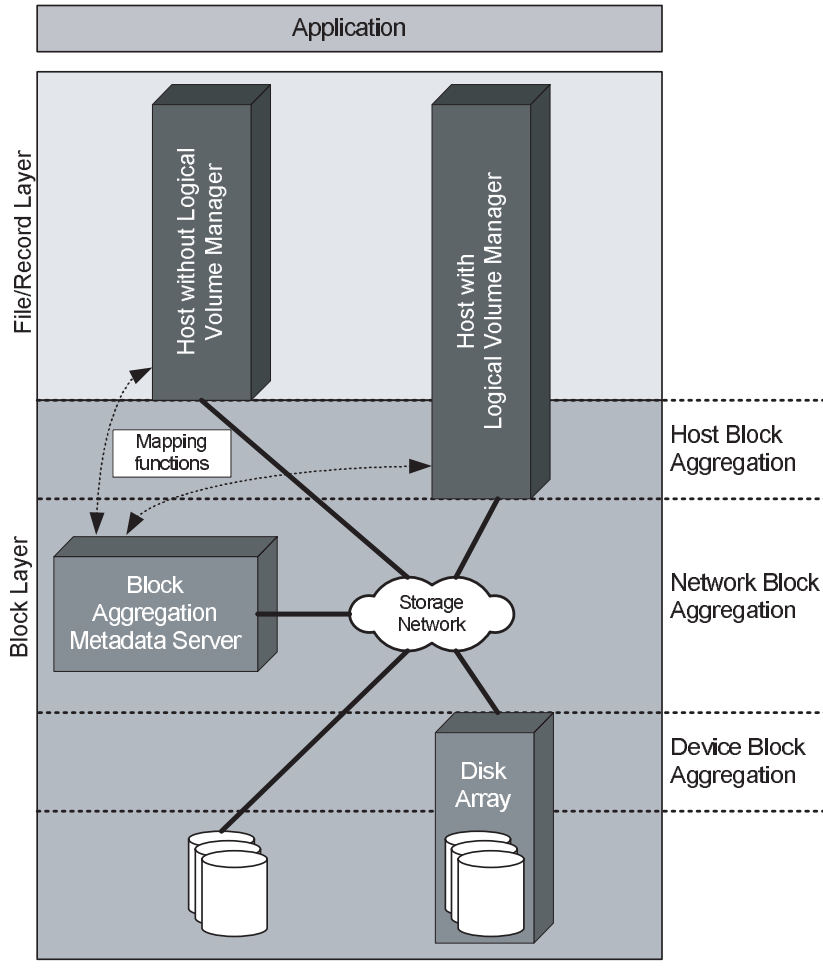

Figure 10.14 In an asymmetric block services architecture a metadata server outside the data path performs the mapping of logical to physical blocks, whilst the data flows directly between hosts and storage devices

If the two locations use different network types or protocols, additional converters can be installed for translation.

10.2.6 File server

A file server (Section 4.2) can be represented as shown in Figure 10.16. The following points are characteristic of a file server:

- the combination of server and normally local, dedicated storage;
- file sharing protocols for the host access;

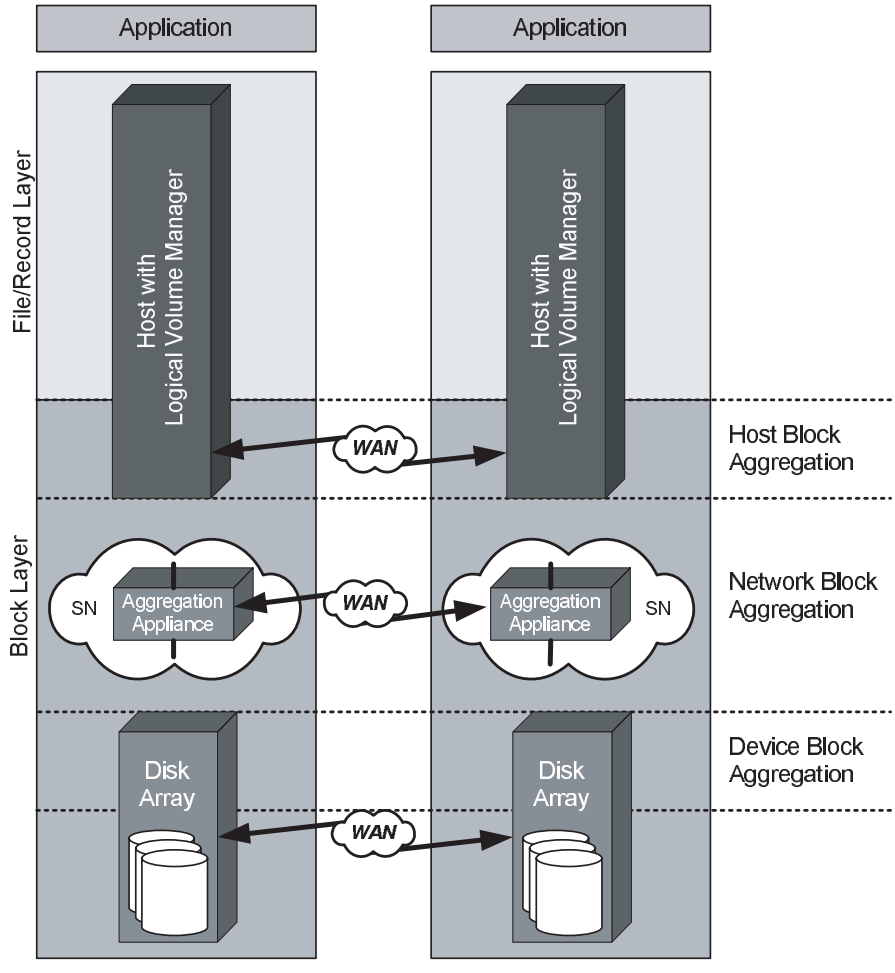

Figure 10.15 Data replication between two locations by means of WAN technology can take place at host level between volume managers, at network level between specialized devices, or at storage device level between disk arrays

- normally the use of a network, for example, a LAN, that is not specialized to the storage traffic;
- optionally, a private storage network can also be used for the control of the dedicated storage.

10.2.7 File server controller: NAS heads

In contrast to file servers, NAS heads (Figure 10.17, Section 4.2.2) have the following properties:

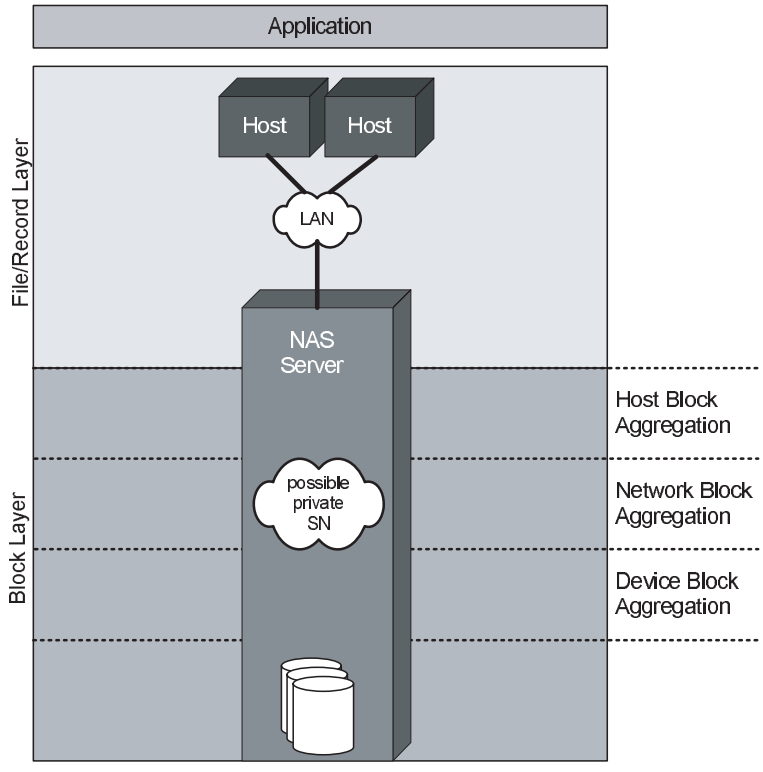

Figure 10.16 A file server makes storage available to the hosts via a LAN by means of file sharing protocols

- They separate storage devices from the controller on the file/record layer, via which the hosts access.
- Hosts and NAS heads communicate over a file-oriented protocol.
- The hosts use a network for this that is generally not designed for pure storage traffic, for example a LAN.
- When communicating downwards to the storage devices, the NAS head uses a block-oriented protocol.

NAS heads have the advantage over file servers that they can share the storage systems with other hosts that access them directly. This makes it possible for both file and block services to be offered by the same physical resources at the same time. In this manner, IT architectures can be designed more flexibly, which in turn has a positive effect upon scalability.

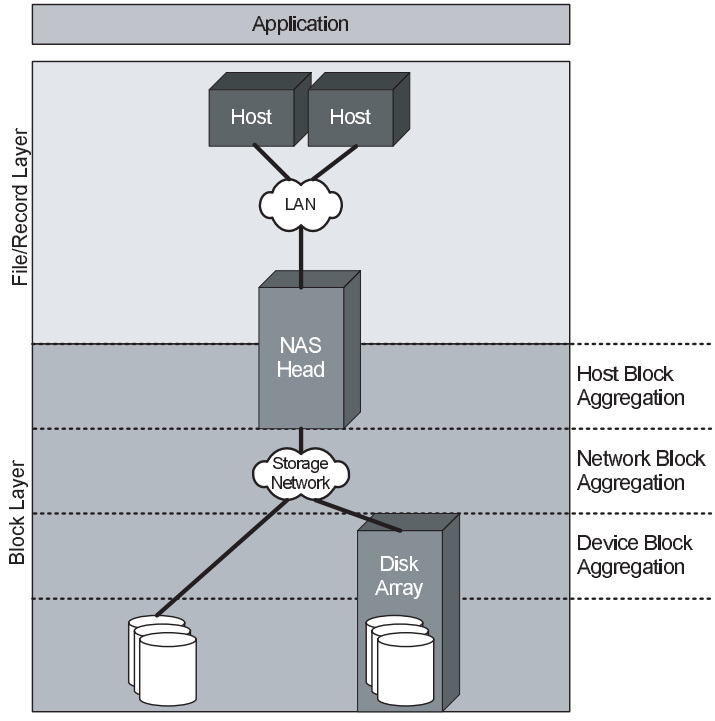

Figure 10.17 A NAS head separates the storage devices from the hosts and thereby achieves better scalability and a more efficient use of resources

10.2.8 Asymmetric file services: NAS/file server metadata manager

A file server metadata manager (Figure 10.18) works in the same way as asymmetric storage virtualization on file level (Section 5.7.2):

- Hosts and storage devices are connected via a storage network.
- A metadata manager positioned outside the data path stores all file position data, i.e. metadata, and makes this available to the hosts upon request.
- Hosts and metadata manager communicate over an expanded file-oriented protocol.
- The actual user data then flows directly between hosts and storage devices by means of a block-oriented protocol.

This approach offers the advantages of fast, direct communication between host and storage devices, whilst at the same time offering the advantages of data sharing on

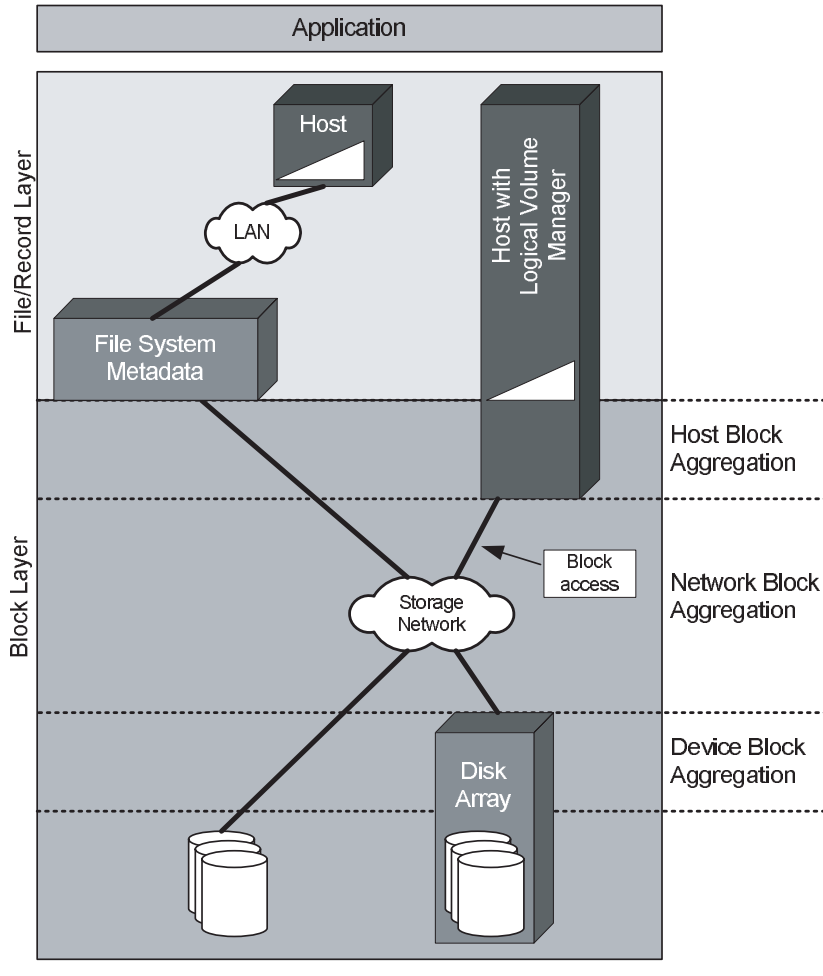

Figure 10.18 A file server metadata manager holds all position data of the files on the storage devices and makes this available to the hosts upon request. Then the hosts can exchange their useful data with the storage devices directly over the storage network. In addition, a metadata manager can offer classical file sharing services in a LAN

file level. In addition, in this solution the classic file sharing services can be offered in a LAN over the metadata manager.

10.2.9 Object-based storage device (OSD)

The SNIA Shared Storage Model defines the so-called object-based storage device (OSD). The idea behind this architecture is to move the position data of the files and the access

rights to a separate OSD. OSD offers the same advantages as a file sharing solution, combined with increased performance due to direct access to the storage by the hosts, and central metadata management of the files. The OSD approach functions as follows (Figure 10.19):

- An OSD device exports a large number of byte vectors instead of the LUNs used in block-oriented storage devices. Generally, a byte vector corresponds to a single file.
- A separate OSD metadata manager authenticates the hosts and manages and checks the access rights to the byte vectors. It also provides appropriate interfaces for the hosts.
- After authentication and clearance for access by the OSD metadata manager, the hosts access the OSD device directly via a file-oriented protocol. This generally takes place via a LAN, i.e. a network that is not specialized for storage traffic.

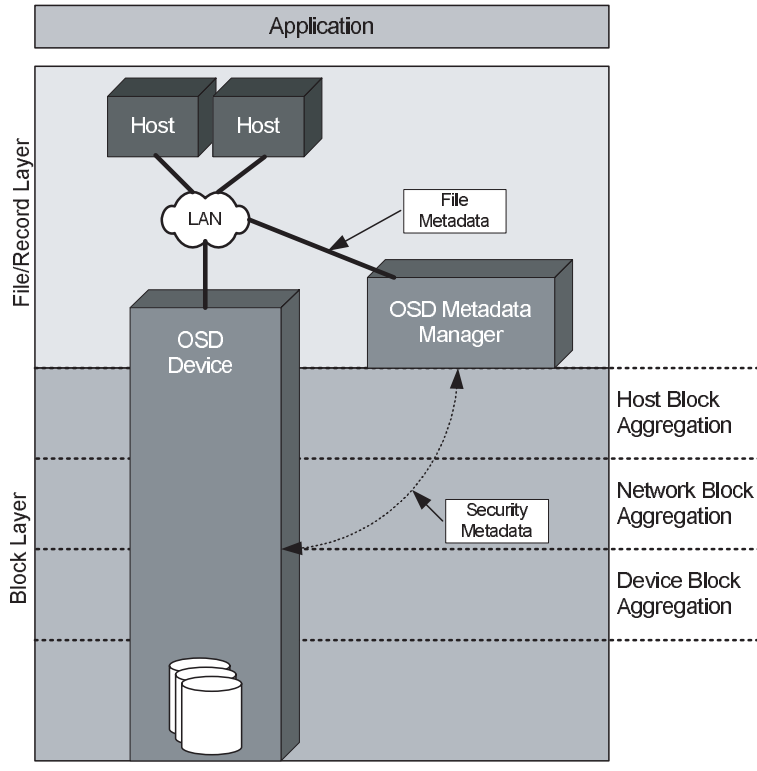

Figure 10.19 Object-based storage devices offer file sharing and facilitate direct I/O between hosts and storage. A metadata manager authenticates the hosts and controls access

10.3 EXTENSION OF THE SNIA SHARED STORAGE MODEL TO TAPE FUNCTIONS

The SNIA Shared Storage Model described previously concentrates upon the modelling of disk-based storage architectures. In a supplement to the original model, the SNIA Technical Council defines the necessary extensions for the description of tape functions and back-up architectures.

The SNIA restricts itself to the description of tape functions in the Open Systems environment, since the use of tapes in the mainframe environment is very difficult to model and differs fundamentally from the Open Systems environment. In the Open Systems field, tapes are used almost exclusively for back-up purposes, whereas in the field of mainframes tapes are used much more diversely. Therefore, the extension of the SNIA model concerns itself solely with the use of tape in back-up architectures.

Only the general use of tapes in shared storage environments is described in the model. The SNIA does not go into more depth regarding the back-up applications themselves. We have already discussed network back-up in Chapter 7. More detailed information on tapes can be found in Section 9.2.1.

First of all, we want to look at the logical and physical structure of tapes from the point of view of the SNIA Shared Storage Model (10.3.1). Then we will consider the differences between disk and tape storage (10.3.2) and how the model is extended for the description of the tape functions (10.3.3).

10.3.1 Logical and physical structure of tapes

Information is stored on tapes in so-called tape images, which are made up of the following logical components (Figure 10.20):

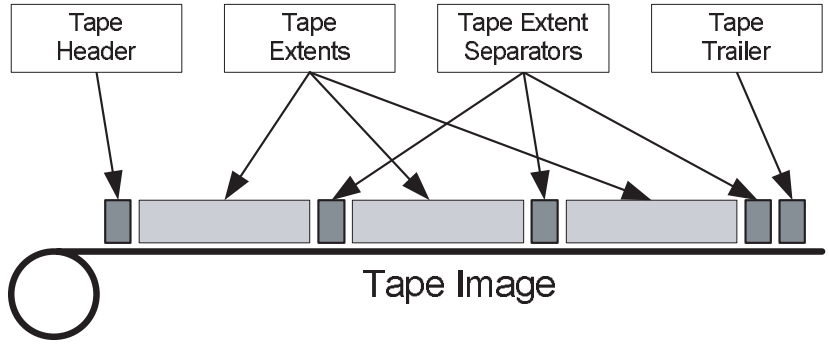

Figure 10.20 Logically, a tape image is made up of tape extents and tape extent separators. A tape header and trailer may optionally mark the start and end of a tape image respectively

- Tape extent
  A tape extent is a sequence of blocks upon the tape. A tape extent is comparable with a volume in disk storage. The IEEE Standard 1244 (Section 9.5) also uses the term volume but it only allows volumes to reside exactly on one tape and not span multiple tapes.
- Tape extent separator
  The tape extent separator is a mark for the division of individual tape extents.
- Tape header
  The tape header is an optional component that marks the start of a tape.
- Tape trailer
  The tape trailer is similar to the tape header and marks the end of a tape. This, too, is an optional component.

In the same way as logical volumes of a volume manager extend over several physical disks, tape images can also be distributed over several physical tapes. Thus, there may be precisely one logical tape image on a physical tape, several logical tape images on a physical tape, or a logical tape image can be distributed over several physical tapes. So-called tape image separators are used for the subdivision of the tape images (Figure 10.21).

10.3.2 Differences between disk and tape

At first glance, disks and tapes are both made up of blocks, which are put together to form long sequences. In the case of disks these are called volumes, whilst in tapes they are called extents. The difference lies in the way in which they are accessed, with

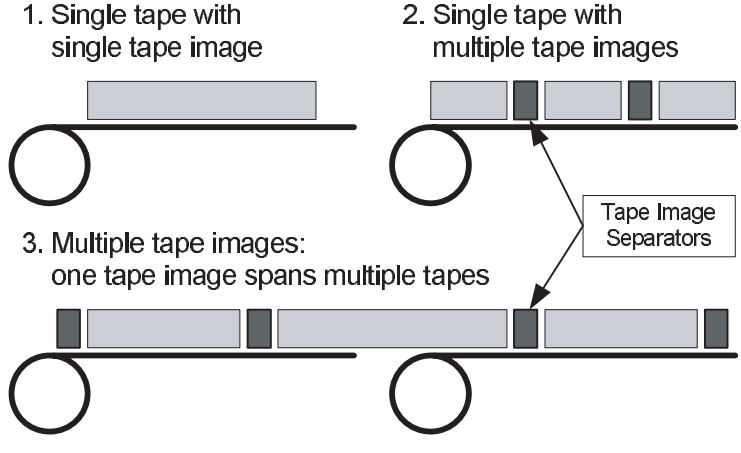

Figure 10.21 Physically, a tape image can take up on precisely one tape (1), several tape images can share a tape (2), or a tape image can extend over several tapes (3). Tape image separators separate the individual tape images

disks being designed for random access, whereas tapes can only be accessed sequentially. Consequently, disks and tapes are also used for different purposes. In the Open Systems environment, tapes are used primarily for back-up or archiving purposes. This is completely in contrast to their use in the mainframe environment, where file structures – so-called tape files – are found that are comparable to a file on a disk. There is no definition of a tape file in the Open systems environment, since several files are generally bundled to form a package, and processed in this form, during back-up and archiving. This concept is, therefore, not required here.

10.3.3 Extension of the model

The SNIA Shared Storage Model must take into account the differences in structure and application between disk and tape and also the different purposes for which they are used. To this end, the file/record layer is expanded horizontally. The block layer, which produces the random access to the storage devices in the disk model, is exchanged for a sequential access block layer for the sequential access to tapes. The model is further supplemented by the following components (Figure 10.22):

- Tape media and tape devices
  Tape media are the storage media upon which tape images are stored. A tape devices is a special physical storage resource, which can process removable tape media. This differentiation between media and devices is particularly important in the context

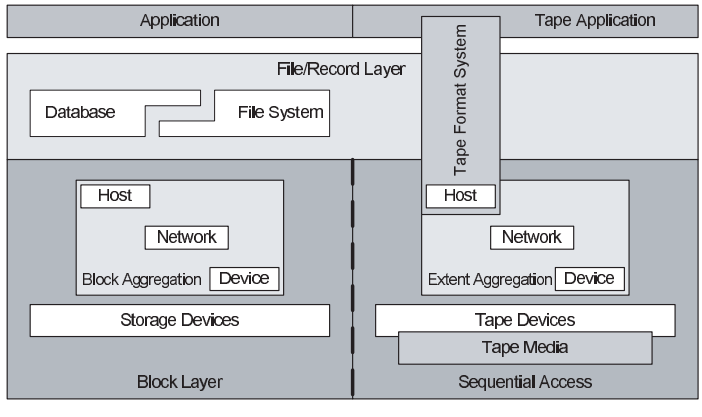

Figure 10.22 The extension of the SNIA model to tape functions expands the file/record layer in the horizontal direction, exchanges the block layer for a sequential access block layer and adds the required components of a tape architecture

of removable media management (Chapter 9). The applicable standard, IEEE 1244, denotes tape media as cartridge and tape device as drive.

- Tape applications
  The SNIA model concentrates upon the use of tapes for back-up and archiving. Special tape applications, for example, back-up software, are used for back-up. This software can deal with the special properties of tapes.
- Tape format system
  In the tape format system, files or records are compressed into tape extents and tape images. Specifically in the Open Systems environment, the host generally takes over this task. However, access to physical tape devices does not always have to go through the tape format system. It can also run directly via the extent aggregation layer described below or directly on the device.
- Extent aggregation layer
  The extent aggregation layer works in the same way as the block aggregation layer (Section 10.1.7), but with extents instead of blocks. However, in contrast to the random access of the block aggregation layer, access to the physical devices takes place sequentially. Like the access paths, the data flows between the individual components are shown as arrows.

10.4 EXAMPLES OF TAPE-BASED BACK-UP TECHNIQUES AND ARCHITECTURES

First of all, we want to examine four examples that illustrate back-up techniques. At the forefront are the access paths and the interaction of the individual components with the UNIX tool *tar* in the file back-up (Section 10.4.1), file system volume back-up using *dump* (Section 10.4.2), the volume back-up using *dd* (Section 10.4.3) and the use of virtual tapes (Section 10.4.4).

We then concentrate upon the data flow between the individual components of a back-up architecture with the disk, first of all discussing the two classical approaches to back up to tape: tape connected directly to the host (Section 10.4.5) and the data flow in a back-up over LAN (Section 10.4.6). We then consider typical approaches for tape sharing in a shared storage environment, such as tape library sharing (10.4.7) and tape library partitioning (Section 10.4.8).

Next we see how tape virtualization by means of a virtual tape controller (Section 10.4.9) and supplemented by a disk cache (Section 10.4.10) changes the data flow. In addition to a virtual tape controller, a data mover can also be positioned in the storage network to permit the realization of server-free back-up. As in LAN-free back-up, in addition to the LAN and the back-up server this also frees up the host performing the back-up (Section 10.4.11).

We will then look at two variants of the NDMP local back-up with local (Section 10.4.12) and external (Section 10.4.13) storage. Finally, we will consider an architecture in which the NDMP is used with a data mover for the realization of server-free back-up (Section 10.4.14).

10.4.1 File back-up

The example shows how a file back-up using the UNIX tool *tar* functions (Figure 10.23):

1. *Tar* reads files from the file system.
2. *Tar* compresses the files in the integral tape format system.
3. It finally writes them to tape.

In the restore case the access paths are turned around:

1. *Tar* reads the file packages from tape.

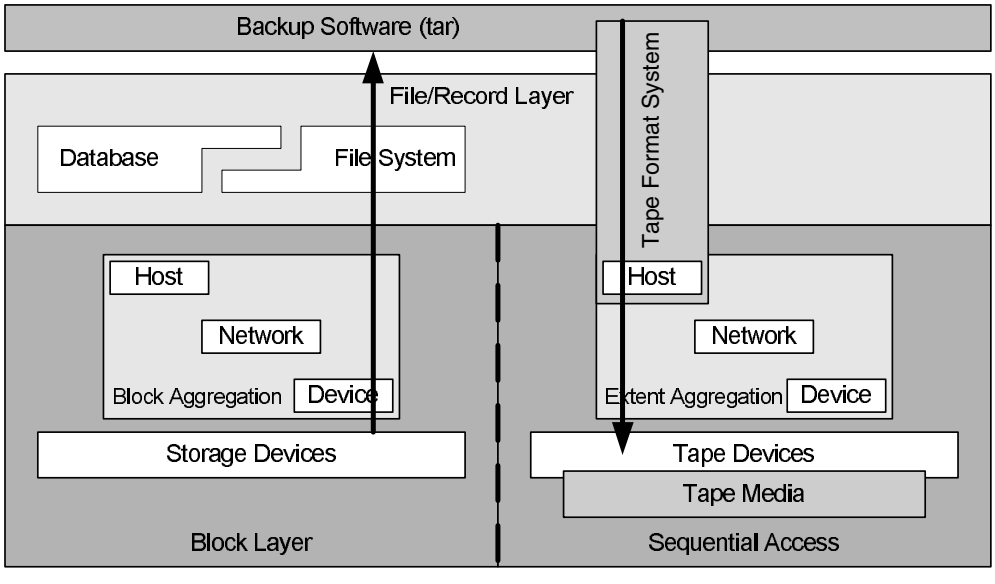

Figure 10.23 *Tar* carries out a file back-up by reading data from the file system, then compressing it in the integral tape format system and writing it to tape. In the restore case, the access paths are reversed

2. *Tar* extracts them by means of the integral tape format system.
3. It writes them into the file system.

10.4.2 File system volume back-up

Using the file system back-up tool *dump* it is possible to use the file system to back up a logical volume – and thus the files contained within it – bypassing the file system (Figure 10.24). The meta information of the file system is also backed up, so that it is possible to restore individual files later. *Dump*, like *tar*, has an integral tape format system for the compression and extraction of the files during back-up or restore.

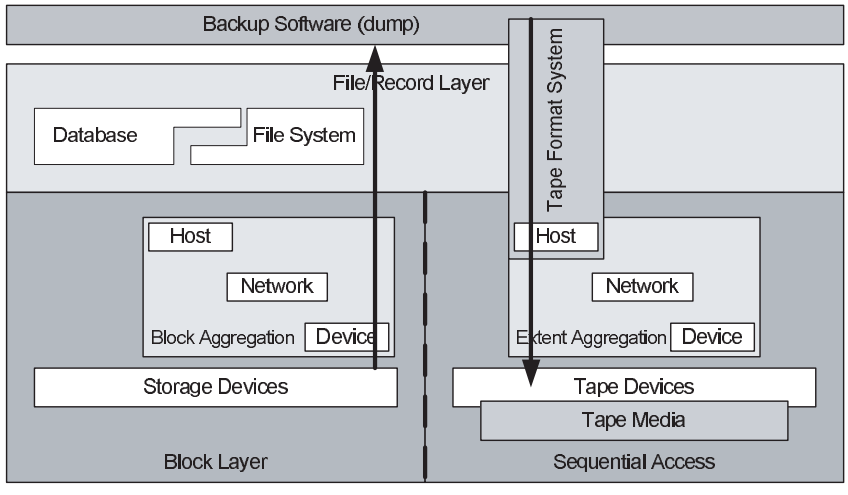

Figure 10.24 With *dump*, files can be backed up directly from a logical volume, bypassing the file system. As is the case for *tar*, an integral tape format system looks after the compression and extraction during restore or back-up

10.4.3 Volume back-up

The program *dd* represents the simplest way of creating a copy of a logical volume and writing it directly to tape (Figure 10.25). *dd* writes the information it has read to tape 1 : 1 without previously sending it through a tape format system. The restore can be represented in a similar way by reversing the access paths.

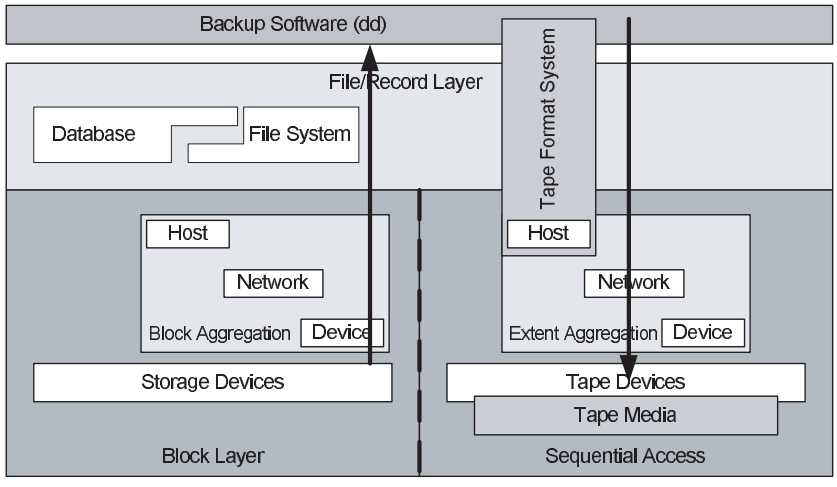

Figure 10.25 The *dd* program creates a copy of a logical volume on tape without the use of a tape format system

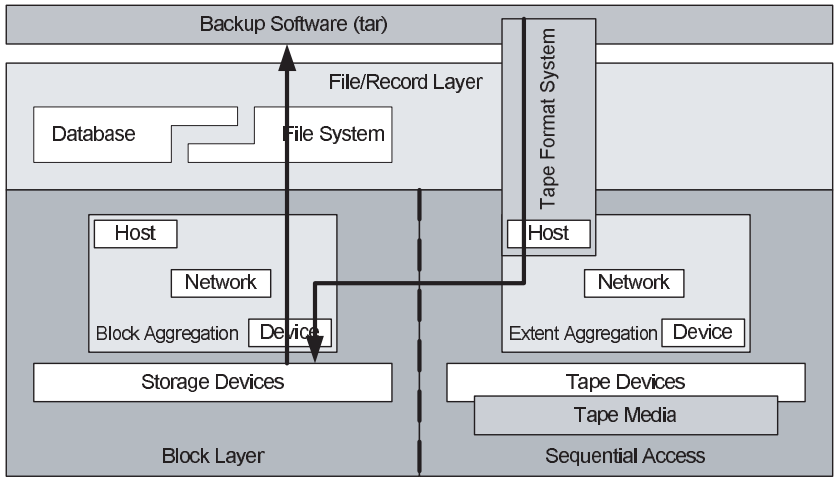

Figure 10.26 By the emulation of a virtual tape, the sequential access of the *tar* command in the extent aggregation layer is diverted into the block aggregation layer of a disk-based storage system, which permits random access

10.4.4 File back-up to virtual tape

The concept of virtual tapes can also be described using the SNIA model. Figure 10.26 uses the example of the *tar* command to show how a disk-based storage system is used to emulate a virtual tape. The sequential tape access of *tar* is diverted via the tape format system in the extent aggregation layer to the block aggregation layer of a disk storage system, where random access can take place.

10.4.5 Direct attached tape

The simplest back-up architecture is the direct connection of the tape to the host, in which the data flows from the disk to the tape library via the host (Figure 10.27).

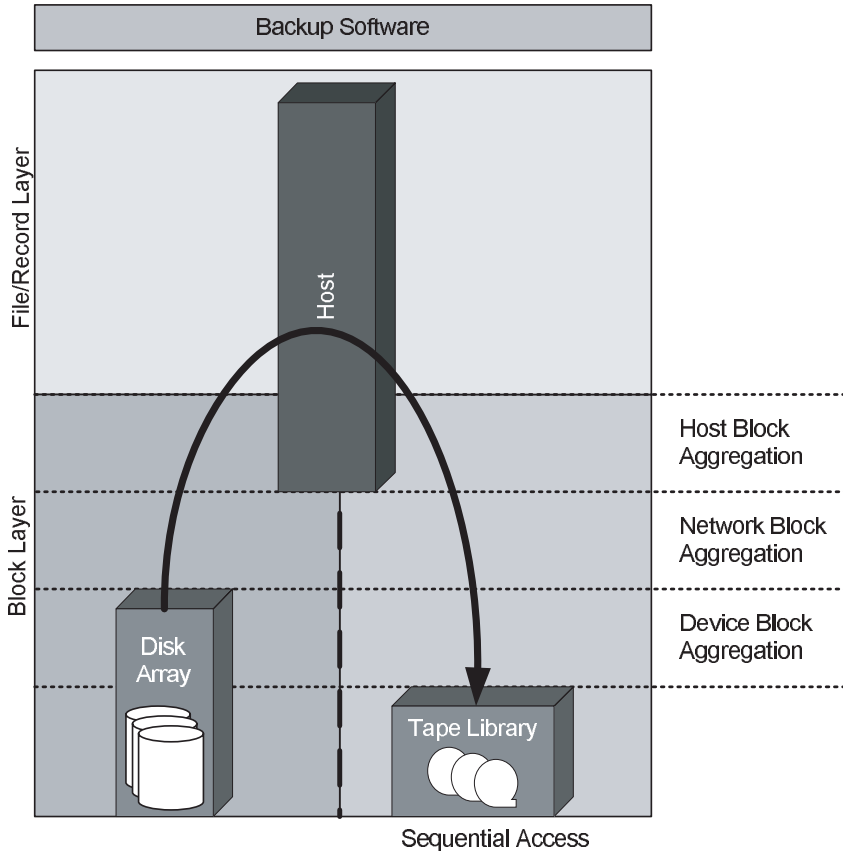

Figure 10.27 In direct attached tape the data flows from the disk to the tape library via the host, as shown by the arrow

10.4.6 LAN attached tape

LAN attached tape is the classic case of a network back-up (Section 7.2), in which a LAN separates the host to be backed up from the back-up server, which is connected to the tape library. The back-up data is moved from the host, via the LAN, to the back-up server, which then writes to the tape (Figure 10.28).

10.4.7 Shared tape drive

In tape library sharing, two hosts use the same tape drives of a library. In this approach, the hosts dynamically negotiate who will use which drives and tape media. To achieve

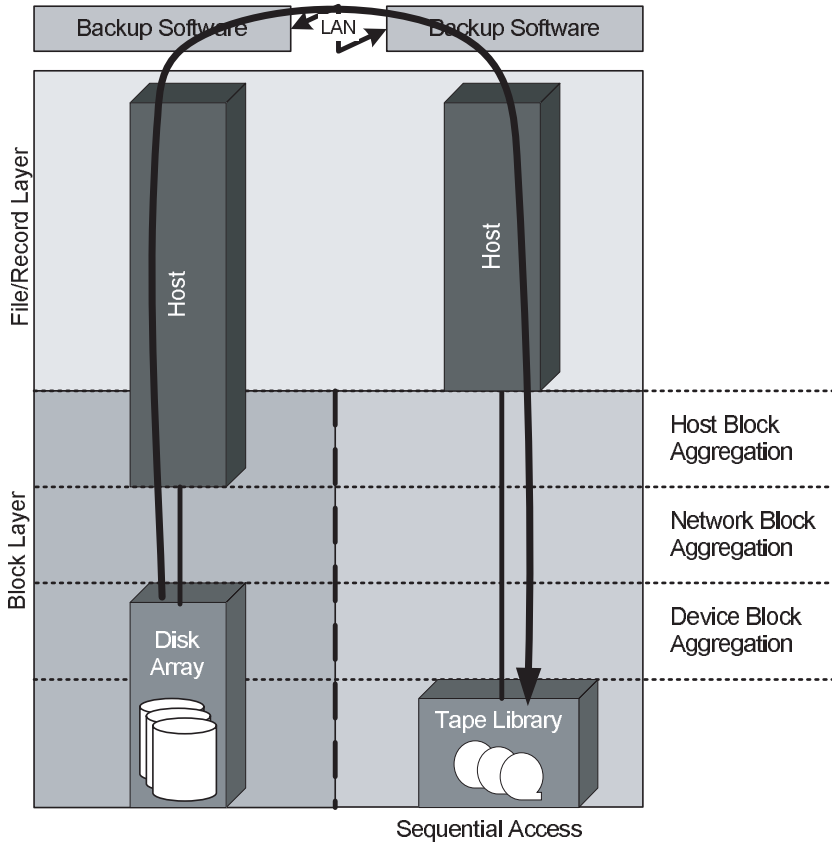

Figure 10.28 In classical network back-up, the data must be moved from the host to be backed up, via the LAN, to the back-up server, which then writes to the tape

this, one server acts as library master, all others as library clients. The library master co-ordinates access to the tapes and the tape drives (Figure 10.29). In this manner, a LAN-free back-up can be implemented, thus freeing up the LAN from back-up traffic (Section 7.8.2).

10.4.8 Partitioned tape library

In library partitioning a library can be broken down into several virtual tape libraries (Section 7.8.4). Each host is assigned its own virtual library to which it works. In this manner, several back-up servers can work to the library's different tape drives simultaneously.

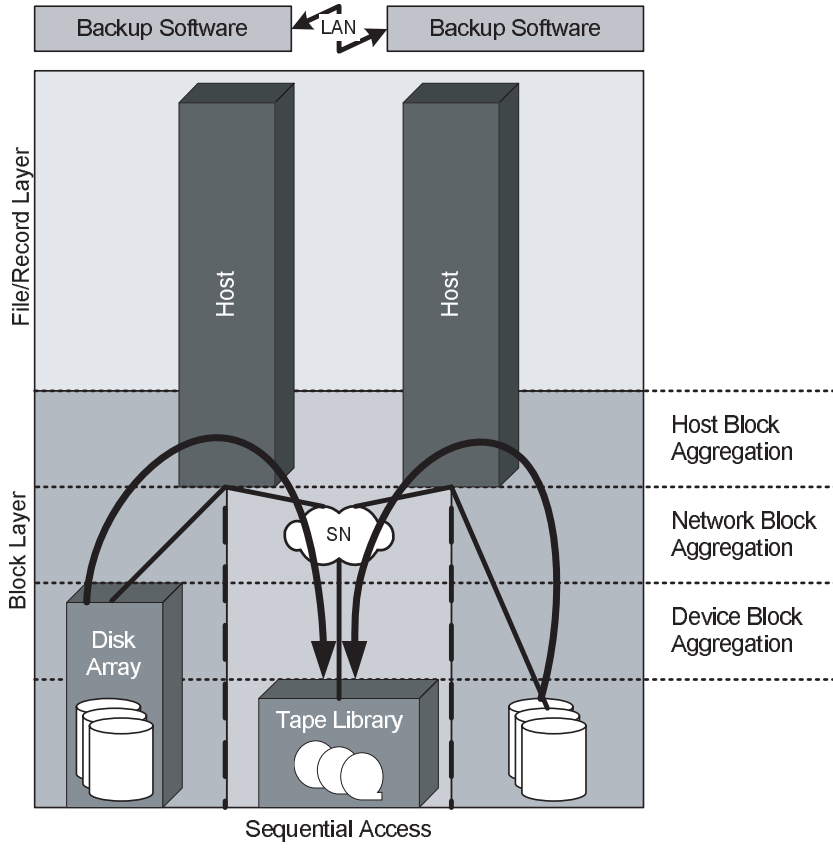

Figure 10.29 A shared tape drive facilitates the implementation of LAN-free back-up, which frees the LAN from back-up traffic

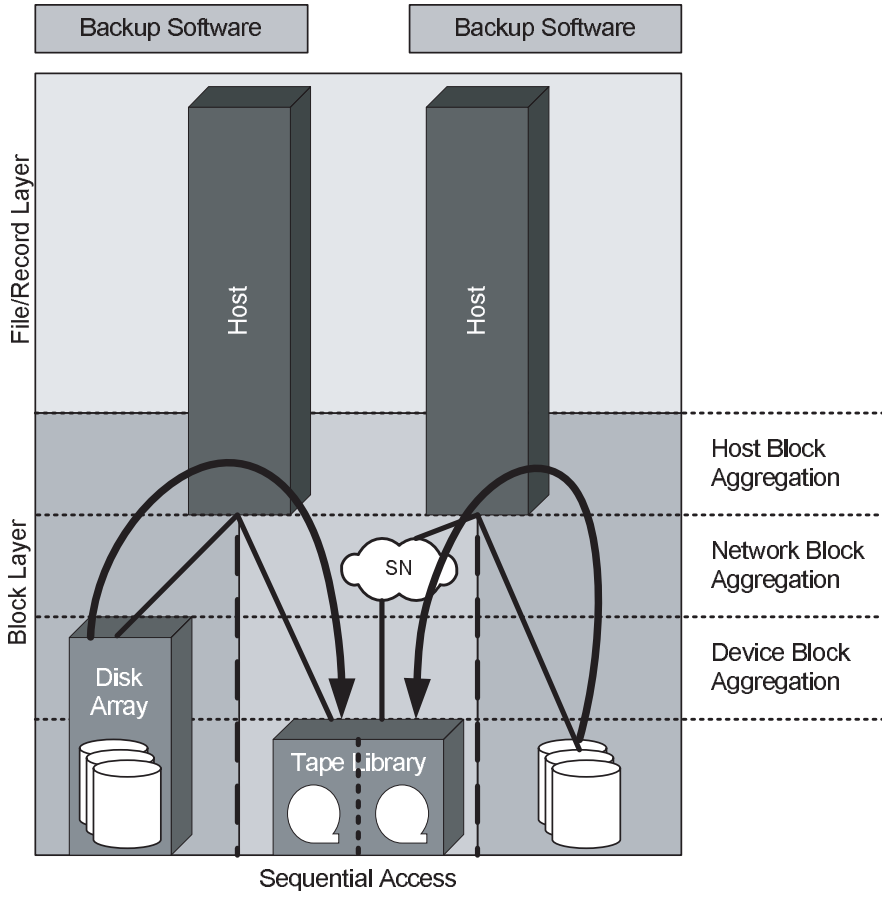

Figure 10.30 In a partitioned tape library, several hosts work to virtual tape libraries that consist of different physical tape drives, but which share a common robot

The library co-ordinates the parallel accesses to the media changer (Figure 10.30) independently.

10.4.9 Virtual tape controller

Additional back-up functionality now comes into play in the storage network! A virtual tape controller in the storage network permits the virtualization of tape devices, media and media changer. Thus, different interfaces can be implemented and different tape devices emulated. However, the back-up data still runs directly from the hosts to the drives (Figure 10.31).

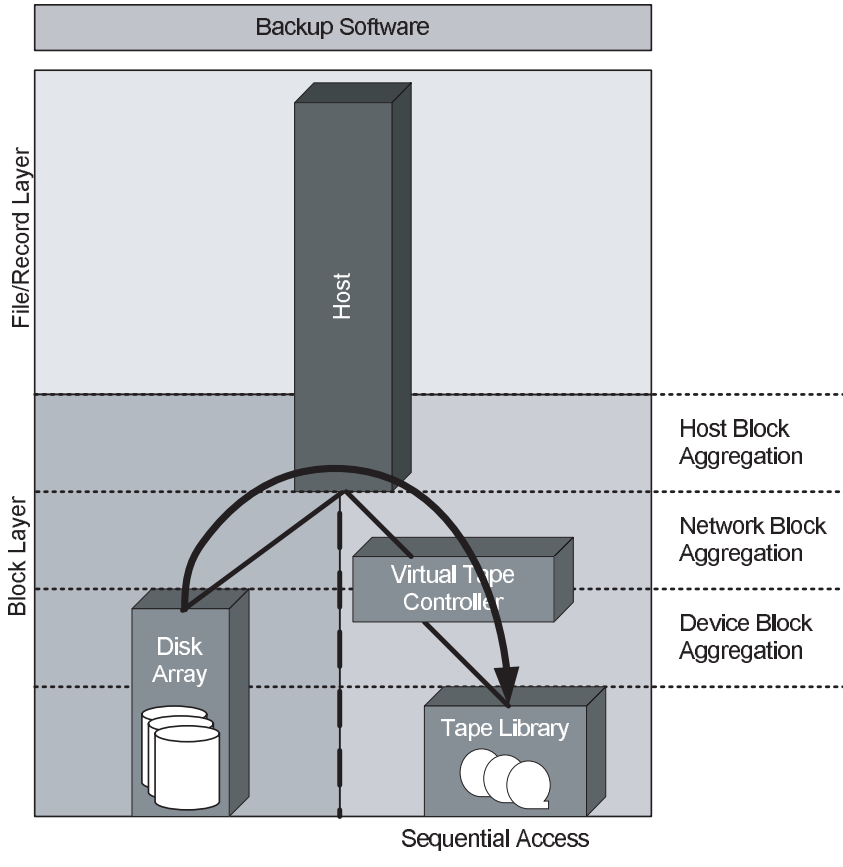

Figure 10.31 A virtual tape controller virtualizes tape devices, media and media changer

10.4.10 Virtual tape controller with disk cache

The approach using a virtual tape controller can be expanded to include an additional disk cache (Figure 10.32). This yields the following three-stage process for a back-up:

1. First of all, the host reads the data to be backed up from disk.
2. This data is first written to a disk belonging to the virtual tape controller, the so-called disk cache.
3. Finally, the data is moved from the disk cache to tape.

In this manner, a back-up can benefit from the higher performance of the disk storage. This is especially useful when backed up data must be restored: Most restore requests deal with data which was backed up within the last one or two days.

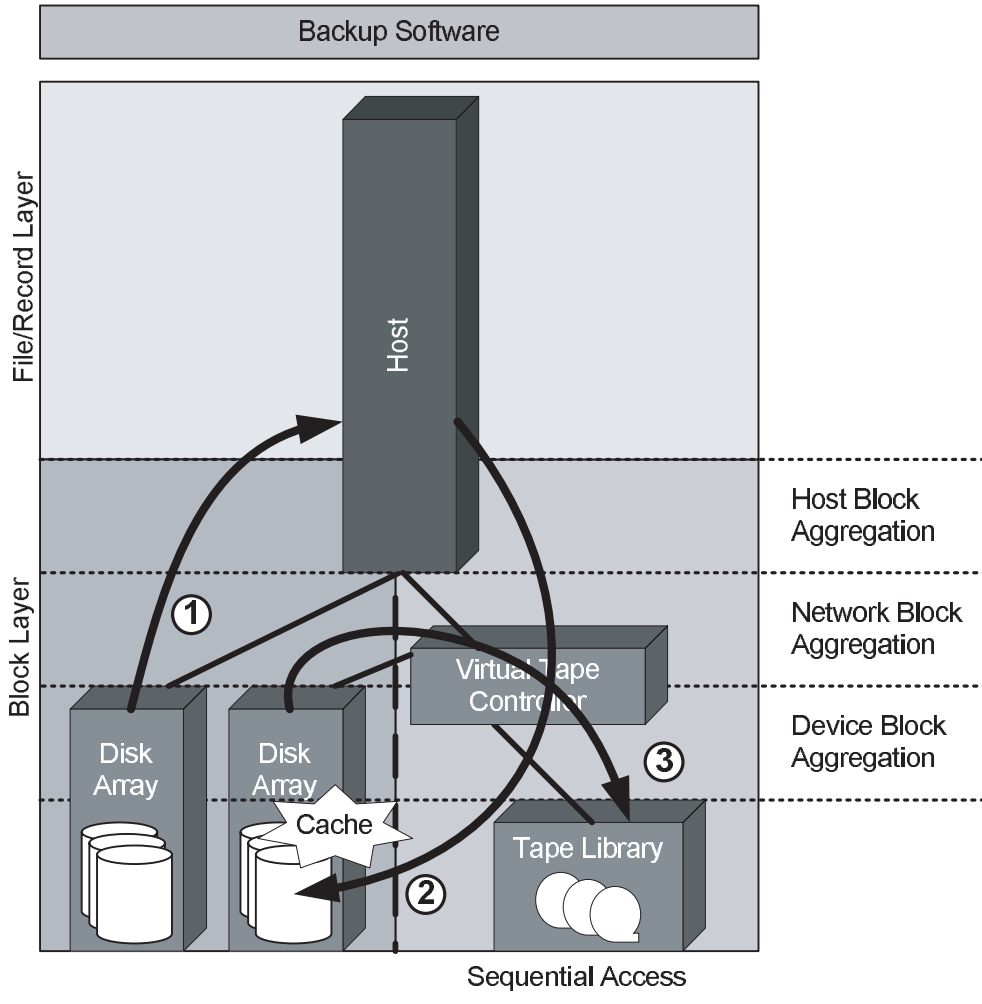

Figure 10.32 If the virtual tape controller is extended to include a disk cache, the back-up software can benefit from the higher disk performance

10.4.11 Data mover for tape

With an additional data mover in the storage network that moves the data from disk to tape, server-free back-up can be implemented. This frees up both the LAN and also the participating hosts from back-up traffic (Section 7.8.1). The back-up servers only have to control and check the operations of the data mover (Figure 10.33).

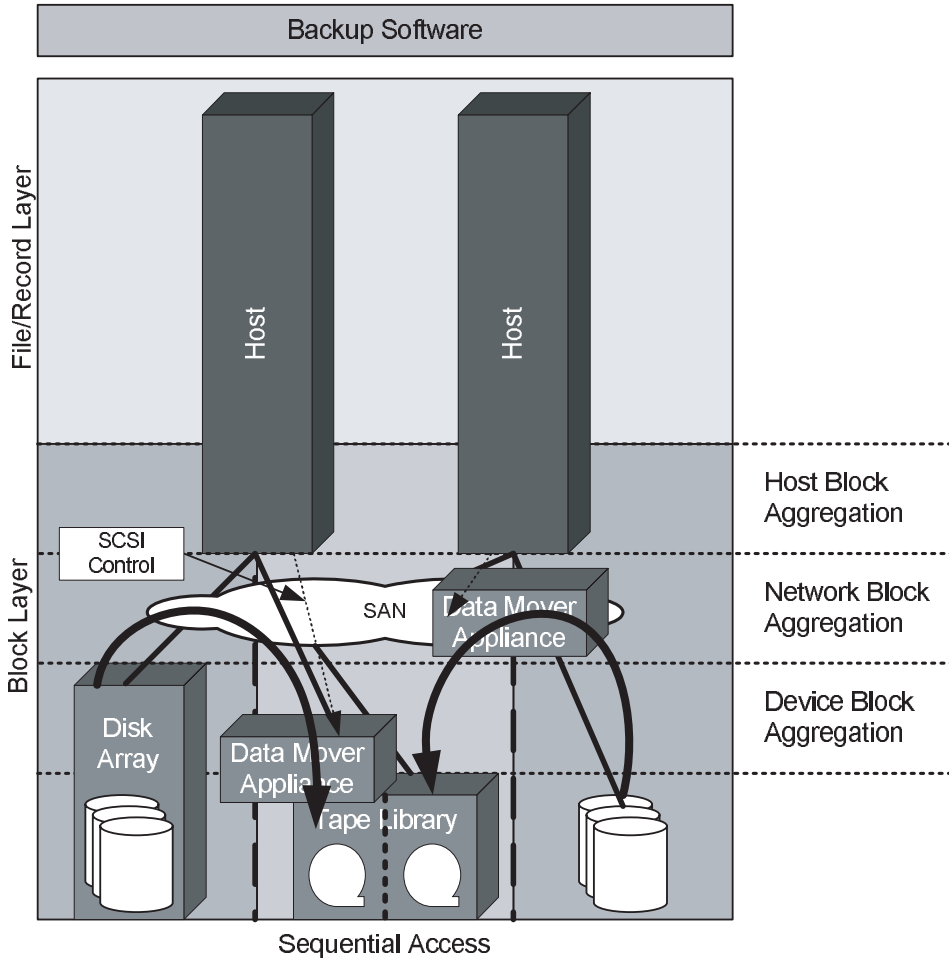

Figure 10.33 Additional data movers in the network implement the server-free back-up, which frees both the LAN and the hosts from back-up traffic at the same time

10.4.12 File server with tape drive

Figure 10.34 shows the implementation of the NDMP local back-up (Section 7.9.4). In this approach, the file server itself transports the data from disk to tape, which in this case is even locally connected. External back-up software checks this process and receives the meta-information of the backed up data via a LAN connection by means of the NDMP protocol.

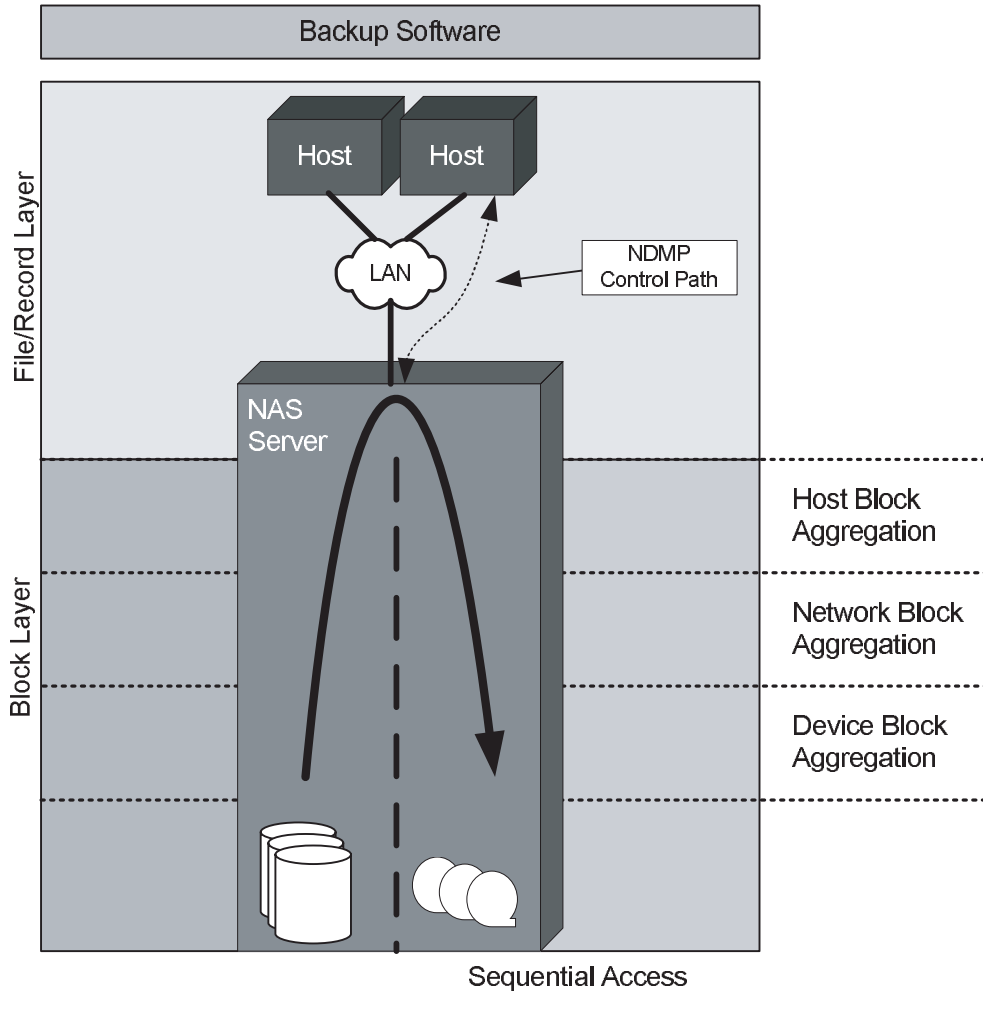

Figure 10.34 In the NDMP local back-up the NAS server takes over the transport of the data from disk to tape, which in this case is even locally connected

10.4.13 File server with external tape

If the NAS server in Section 10.4.12 is exchanged for a NAS head with external disk and tape storage, then the back-up software additionally checks the functions of the tape library on the host. Again, additional meta information on the backed up information flows from the NAS head to the back-up server (Figure 10.35).

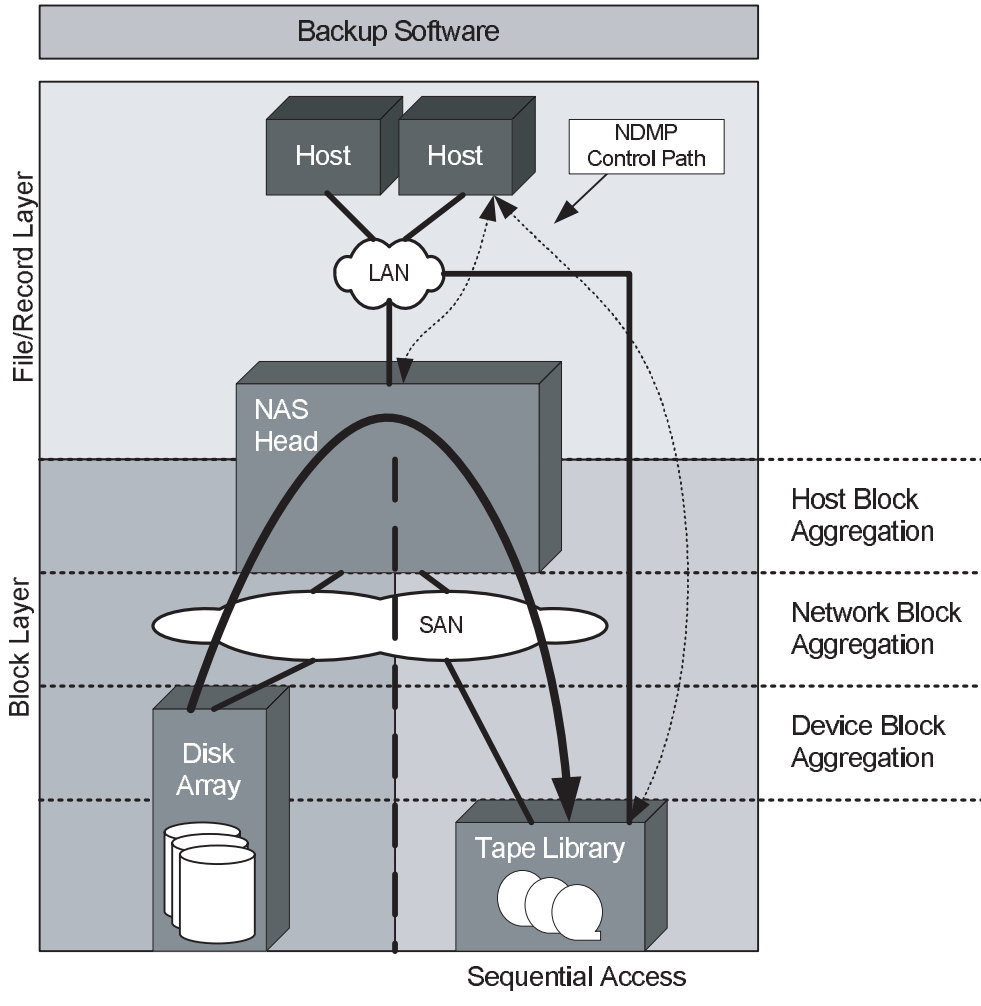

Figure 10.35 The NDMP local back-up can also be implemented for external disk and tape storage on a NAS head

10.4.14 File server with data mover

An additional data mover in the storage network (Figure 10.36), which takes over the transport of the back-up data from the NAS head with external storage, also implements server-free back-up (Section 7.8.1) on file server level. LAN and back-up software are already freed from data transport by the use of NDMP (Section 7.9.4).

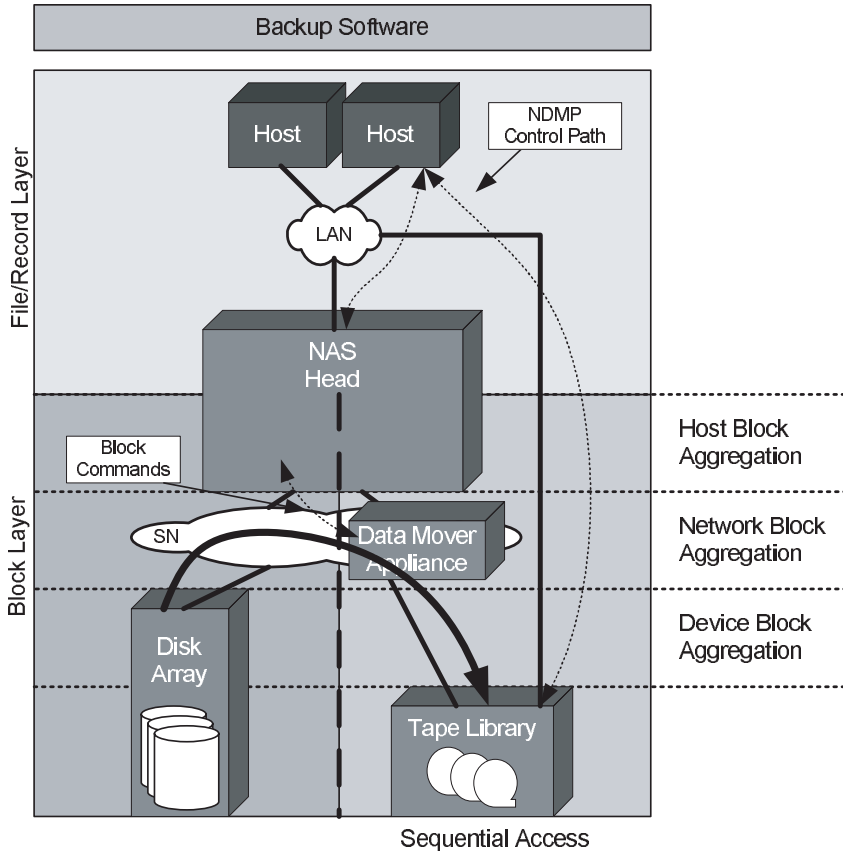

Figure 10.36 Combined use of NDMP and a data mover frees up the LAN, the back-up server due to NDMP, and frees up the NAS head from the transport of the back-up data by the implementation of server-free back-up at file server level

10.5 SUMMARY

The SNIA Shared Storage Model permits architectures to be described and compared with one another in a value-neutral manner and discussed using a consistent vocabulary. This makes it easier for manufacturers to present the differences between their products and competing products to the customer on the basis of a common vocabulary. The customer interested in the actual functionality finds it easier to compare and choose between different product alternatives. He benefits from the function-centred approach of the SNIA Shared Storage Model, which puts the entire functionalities of the Shared Storage environment in the foreground and only highlights the components on the basis of which these are implemented as a secondary consideration.