4 Architectural Overview

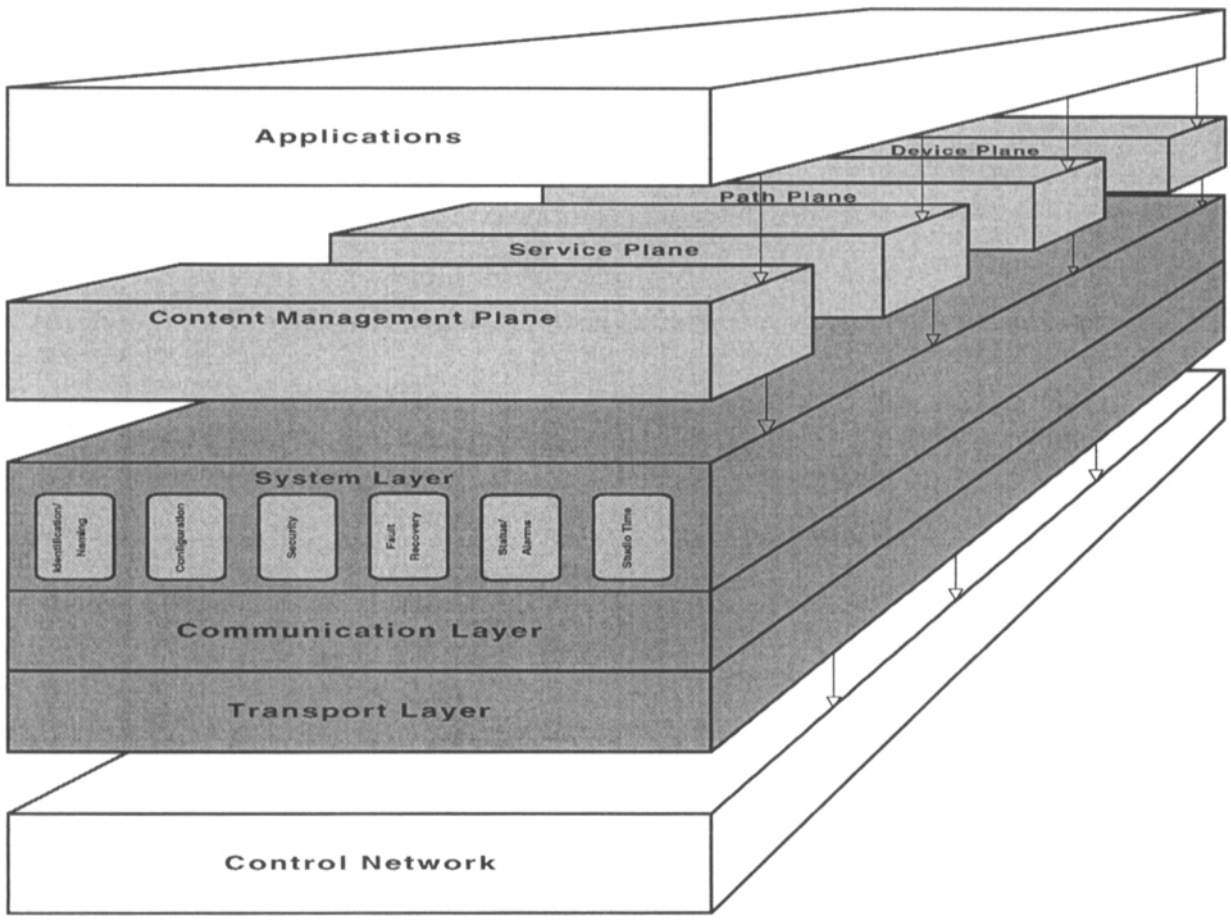
4.1 Overall Requirements

An overall architecture will need to meet the following requirements:

1. Flexible enough to accommodate varying enabling technologies where possible.
2. Independent of specific technologies where possible.
3. Scalable to a range of platforms and environments.
4. Extensible to adapt to emerging technologies.
5. Modular in nature allowing functional pieces of the system to be used as building blocks.
6. Offer a viable migration path for existing facilities.

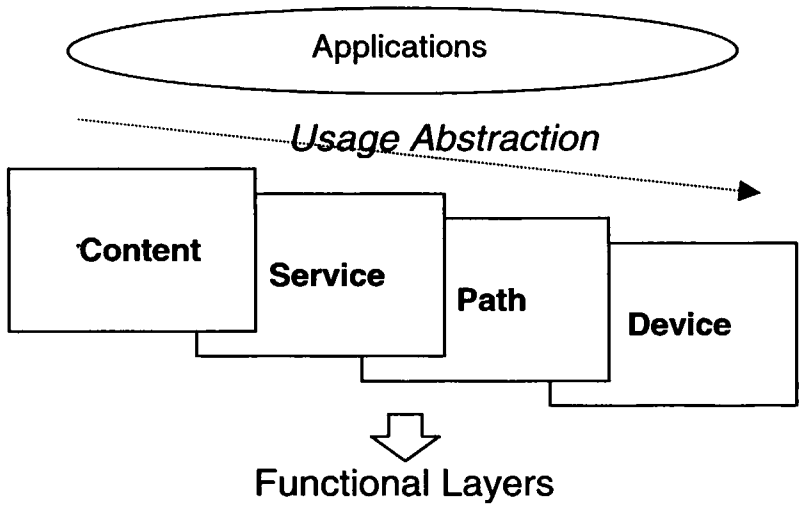
4.2 Component Diagram

This diagram shows the relationship between the key architectural *components* of the system. Each of the component *planes* and the component *layers* will be described in more detail within the remaining sections of the document. It will be helpful to refer to this diagram as each component is described.



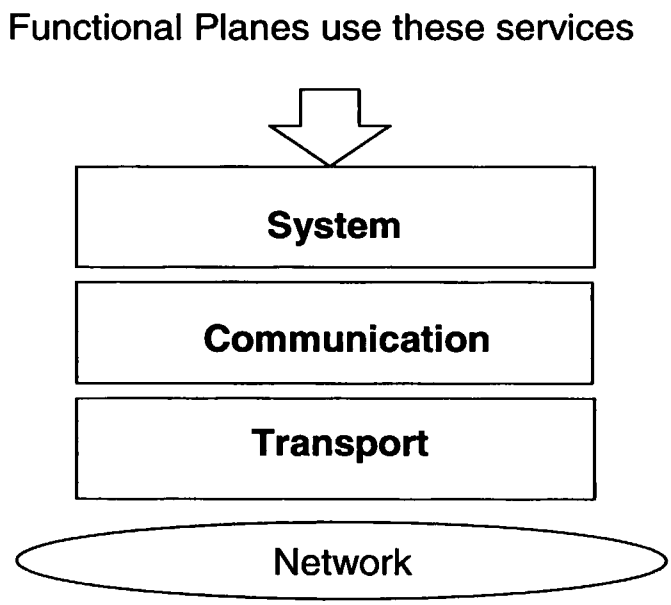
4.3 Functional Planes

This diagram shows the relationship between the component planes of the system. The planes normalize access to the primary studio *assets*, namely *content*, *network*, and *devices*. A usage abstraction from content through to device is provided to support varying levels of workflow within the studio.



4.4 Functional Layers

This diagram shows the relationship between the component layers of the system. The layers provide access to the reusable system functions as provided by the studio devices / resources. It also provides physical device independence and physical network independence where possible to allow for the highest degree of application portability.



4.5 *Functional Planes*

Functional planes exist as the front line interface between the studio applications and the functionality of the control system. The four functional planes are defined below:

1. Content Management
   - Manages content in the studio
   - Understands physical storage allocation for the content
   - Performs activities including Content Distribution, Content Creation, Scheduled Operations, and Storage Management
   - Presents a view of content as required for *data streaming*
2. Service
   - Combines multiple content streams into complete services
   - Maps services onto resources available in the *path plane*
   - Abstracts the mapping of content to individual pieces of equipment
   - Uses content sources to create paths
3. Path
   - Facilitates the physical connection between devices for the purposes of data streaming
   - Manages the physical links between devices
   - Manages the resources required for these connections
4. Device
   - Contains the interfaces used to access studio equipment
   - Provides specific information for device IO capability (ports)
   - Based on a SMPTE defined hierarchy of functional classifications
   - Provides for extensible interfaces

The planes are organized into a form of 'usage abstraction' from content, to service, to path, to device. Adjacent planes co-operate to carry out individual operations as generated by the studio applications. Though this abstraction is beneficial in most cases, it is not explicitly required. For example, applications can access functionality at any plane, or planes can access any other plane in order to get the task accomplished.

These functional planes represent logical partitioning within the control system. From an implementation point of view, the devices themselves will likely be responsible for running local software that will contribute to the functionality of one or more of these planes. For example, within a network router, software exists at runtime that is responsible for some path plane functionality.

4.5.1 Content Management Plane

The content plane is the locus of all the "higher level" applications ("business" applications) in the studio and is the first line of support for these applications. It implements *content management* classes of objects including:

- *Library Server* - A studio will have zero or more objects of this class, whose purpose is to provide a *Library Service.* If there are multiple such objects, they will link together to provide a unified, though distributed, Library Service[1].

- *StreamingElement* - objects of this class are created to present a view of content files as endpoints for streaming file transfers. These objects will be created as necessary to support scheduled operations, and be deleted after the operations end.

4.5.1.1 Rationale

Many applications manage and manipulate content within the control network. Whether content within the physical network is centralized on one library service or whether it is distributed among a set of devices, all applications need to access content in a uniform way. By collecting all content manipulation functions into a common service, the system will be able to accommodate many flexible storage schemes as well as adapt to emerging database technologies as they become available.

4.5.1.2 Dependencies

The content plane interacts with business applications in the *Service Plane*, and the *StreamNode* and various *device classes* in the *Device plane.* In particular,

- the Library Service relies on the abstract *Device object* encapsulating the media server on which a particular media recording resides, to construct StreamingElement objects to represent that recording, when asked to do so by the LibraryService.
- StreamingElement objects are instances of the StreamNode class.

4.5.1.3 Functionality

4.5.1.3.1 Library Service

The Library Service keeps track of all broadcast contents in the studio and the units of physical storage they reside on, for the purpose of conducting such activities as:

- Content ingest
- Content distribution
- Content Browsing
- Scheduled operations
- Verifications of operations
- Storage management
- Rights management

[1] A studio need not have any Library Servers (for instance, if it has only one content server, which manages its own content collection). A Library Service does not preclude or pre-empt management by content servers or storage servers from managing their own asset collections, or to manage such things as file archiving and disk free storage defragmentation.

It maintains a content[2]-aware database of the collection, and may also keep reduced-resolution versions of the content. It supports querying and browsing, from clients which can be (in the general case) either within the studio or outside it.

The Library Service is the focal point for knowledge about all the content and *content containers* (cassettes, cartridges, reels, etc) in the studio (although individual video servers would be knowledgeable about their own contents). It is responsible for acquiring, maintaining, and searching knowledge about content and containers in the studio. It keeps this knowledge in a manner which supports a Content and Container data model, which includes at least the following concepts (although it should also be extendable by the studio):

- Descriptive metadata (information about the content, without regard to how the content is encoded)
- *Essence* (content rendered, per some encoding scheme, in some frame format, at some bit rate)
- Storage media (cassettes, cartridges, reels, etc) removable from the devices which play and record them
- Non-removable storage media, e.g., hard disks (awareness of free space on)
- Recordings (instances of essence recorded on media, either fixed or removable)
- Content items (subsets of complete programs)
- Multiplexes (programs which themselves contain other programs, e.g., MPEG-2 streams)
- Intellectual property rights
- *Metadata* about content and recordings

Library Servers will handle requests for service including:

- Creating catalog entries to represent new content or containers
- Assigning UMIDs and UPIDs to new catalog entries
- Deleting content and catalog entries for it
- Querying of the database (discussed later)
- Brokering the creation of StreamingElements to represent content items for transfers. (The servers which will play or record the content items, or on which they reside or will reside, actually create and own their StreamingElements; the LibraryServer mediates this transaction for its clients.)
- Report Library Server's/Service's make and model; Service level, constituent components and their software release levels; capabilities, etc.

4.5.1.3.2 **StreamingElement**

These objects, which are constructed in anticipation of streaming of recordings (stored units of content), should contain more or less these information elements, regarding the content item and the full recording of which it is a (possibly proper) subset:

- Name / title
- UMID and/or UPID
- Encoding scheme
- Normal play bit rate
- Whether essence has constant bit rate throughout
- Host server's name in the *ServiceGateway* registry
- Recording's unique name in its host server's *namespace* (i.e., file name; "key")
- Playing time of the content item, if transmitted at normal play rate

[2] (In the sense of "Query By Content")

- *Transmission time* of the content item. (This can differ slightly from the playing (presentation) length. See Glossary.)
- StartTime - offset to beginning of the content item to be played
- StartupDelay – pre-roll latency
- Indication of whether recording is marked for deletion (recording may no longer be used, but its resources: space; directory entry have not yet been reclaimed)
- Indication of whether the recording is archived to off-line storage (and will need to be retrieved before playing)
- Indication of whether the recording has been "checked out" to any editor
- Indication of whether a recording is in the process of being received, or is complete
- Number of programs multiplexed in this recording
- List of programs multiplexed in this recording

The following operations should be supported:

- Schedule the StreamingElement for transfer to archive storage.
- Check in a recording that had previously been checked out. May involve extracting some characteristics of the recording for storage in its catalog entry.
- Copy information from an existing StreamingElement to provide content for a new StreamingElement object. This is intended for use when, for example, recording from an incoming satellite feed, where the content format is known.
- Close the stream owned by the StreamingElement, and indicate to the server owning it that the StreamingElement is no longer needed)
- Close the stream owned by the StreamingElement, and mark its underlying recording for deletion
- Allocate more storage space for a StreamingElement.
- Get information about the StreamingElementInfo object.
- Schedule an archived StreamingElement for transfer to online storage
- Update non-content-based attributes of the StreamingElement object

4.5.1.4 Implementation Considerations

Present "off-the-shelf" database technology uses the industry-standard communications protocol ODBC (and layers built on top of it, such as Java JDBC) to allow queries and updates to be posed to databases, expressed in the industry-standard object-relational language (SQL). We can choose to use these conventions, or we can use some other protocol and interface that will be more similar to that used by the other objects in the ASCA *object model*. Aside from aesthetic considerations of a non-uniform interface, the problem of SQL is that it exposes the complete table structure of the internal database. This means Library Servers would have less control over the queries posed by their clients, and no ability for their manufacturers to hide alternative data structures which give product differentiation. However, database clients with complete access to internal data, will have powerful abilities to dynamically discover the customized characteristics of the Library Server they are dealing with, without the Library Server's manufacturer having to build in any special functionality to support it. Further research is necessary before a final decision can be reached.

Use StreamingElements only for those media and containers which are actually scheduled to be used in a streaming operation, or are about to be so scheduled. Let the representation of each media recording and each container residing in the studio be its entry in the database, until such time as streaming of it is planned; then it can be represented as an object. It is typically the case that StreamingElement objects will be constructed to run on the server hosting the essence.

4.5.1.5 Legacy Considerations

- Use of legacy numbering schemes as keys/identifiers of videos in cassette libraries and players, especially robotic ones
- ISCI[3] codes (8 alphanumeric characters) as unique content identifiers
- Perhaps other recording "unique identifiers" as well
- Legacy library servers may not be able to characterize themselves according to the "capabilities" scheme we agree on

4.5.1.6 SMPTE Activities

- The Content and Container data model
- UMIDs and UPIDs (already defined or being defined)
- Identification of any structures that can be used to represent Intellectual Property rights
- Synchronization with the ongoing work of W25

[3] Standardized by the American Association of Advertising Agencies

4.5.2 Service Plane

The service plane is responsible for aggregating multiple content streams into complete services and then mapping the implementation of these services onto aggregated resources exposed by the path plane. The service plane manages multiple logical paths within the network that together represent a program, and therefore need to be treated together.

4.5.2.1 Rationale

The service plane creates a useful abstraction between implementation details and high level content scheduling by not requiring content mapping directly to individual pieces of equipment. In essence, playlists are the inputs to the service plane, and requests for path resources are the outputs. The advantage is that applications and content management services see a functional view of the network, instead of a physical view as seen by the path plane. This allows network equipment to evolve over time while exposing a consistent interface via the service plane.

4.5.2.2 Dependencies

Content Management Plane: The content plane passes down references to scheduled content events. The service plane passes up status/alarm messages applicable to these events.

Path Plane: The service plane passes down requests to implement services on aggregated path resources. The path plane passes up status and alarm messages applicable to these services. The Service Plane initiates fault recovery strategies, by requesting alternate sources or paths from the Path Plane.

Configuration System Layer: Because the service plane is purposefully unaware of detailed equipment configuration and interconnection, a high-level configuration function is necessary. Such a function would take the form of a template ("pre-set") whereby the service plane would be capable of putting portions of itself into pre-determined states. By nature of the service plane, these pre-determined states would involve the mapping of content to aggregated path resources.

Status/Alarm System Layer: The service plane needs the same capability of receiving and transmitting status/alarm messages with minimal latency and high quality of service. Status/alarms at the service plane differ from that of other planes only in their content. However, the tunneling of proprietary and/or low-level messages to/from other planes is necessary to enable the adequate presentation of information to a user (if desired by the user/application). In most cases, alarms that initiate a fault recovery are managed by the Service Plane, including attempts to configure the Path Plane for a different routing path or the Device Plane for a different device which contains the same source material. Faults which cannot be resolved by the Service Plane are passed up to the Content Management Plane.

4.5.2.3 Functionality

The generic functions of the service plane will allow translation of abstract content into physical implementation. Additionally, status/alarm messages from the path plane will enable a service plane application to ensure that services are being implemented with their desired *quality of service (QoS).* The following is a simplified example that combines these basic functions:

- Program information is passed from a station traffic system. This information consists of:

```
Channel 99-1: 10 Mbps
      20:00:00-20:12:59, "Dirt Clods Revisited", Segment 1
      20:13:00-20:13:59, "Dirt Clod Festival Ad"
      20:14:00-20:27:59, "Dirt Clods Revisited", Segment 2
```

```
Channel 99-2: 8 Mbps
      20:00:00-20:10:59, "Pocket Lint Anthology", Segment 1
      20:11:00-20:15:59, "Public Service Announcement #12345"
      20:16:00-20:27:59, "Pocket Lint Anthology", Segment 2
```

- The above programming information is passed to the service plane from the content plane. According to previously configured equipment, the content plane informs the service plane that there exists the appropriate content StreamingElement, and appropriate processing, and transmission equipment ready to be commanded between 20:00:00 and 20:27:59. Without having to issue detailed configuration commands to servers, MPEG-2 encoders, and MPEG-2 multiplexers, the service plane application is able to easily instruct the path plane to carry out the necessary functions.

- At 20:05:21, the service plane application receives an alarm from the path plane indicating a loss of video source signal on channel 99-1. At 20:05:22, the service plane responds by issuing a dual fault recovery command to the path plane. One command disables the failed path, while the second enables a pre-configured redundant path. The service plane has implemented a redundant switch over, potentially from one vendor's equipment to another, without any specific knowledge of the low level workings of the equipment involved. In a real application, it is recommended that A log of the fault and the corrective actions taken would be logged and reported.

In the context of an MPEG-2 content transfer (most often a DTV broadcast), the service plane takes on additional responsibility in the area of SI aggregation. At this plane, all of the descriptive information (PIDs, formats, rating information, etc.) concerning the content that is flowing in and out of a facility is known. Therefore, the service plane is able to combine this information into the applicable SI tables. Following is an example of SI aggregation in the service plane that continues from the above example:

The service plane understands that "Dirt Clods Revisited" and "Pocket Lint Anthology" are both being transmitted in the same DTV multiplex along with their corresponding advertisements. With exact knowledge of both the programming content and status from the path plane, the service plane is able to command the *PSI/PSIP* generator to emit the correct tables. Upon executing the redundant switch, it is possible that some information had to change (PIDs, bit rate) based on the capabilities of the redundant path. Based on the path plane's status upon acknowledging the switch, the service plane is able to update the PSI/PSIP generator with the most current information.

The Service Plane understands the requirement to organize command sequences, executed in a specific order, to various devices to avoid transitional illegal conditions between or internal to those devices or paths. It organizes these command sequences into "transactions". A database of these transactions is available in the Service Plane to permit the Management Plane to request various activities without having to know the details required to implement them.

4.5.2.4 Implementation Considerations

The Service Plane is responsible for selecting among several instances of a particular media asset available on the same or different media storage or generation devices. The selection is made according to scheduling conflicts between requesters for a specific device or path or according to asset quality, ease of accessibility, cost or redundancy and fault recovery options available in the case of a device or path failure. If the Service Plane detects a device or path failure, then it is responsible for the optimal recovery strategy, choosing an alternate source or path. If it is unable to recover on its own, the failure is passed up to the Management Plane so that it may choose an alternate asset, such as a test pattern or a rerun of an "I Dream of Jeannie" episode

All of the device and path set-up times and the status and alarm latencies must be specified beforehand. This allows the management system to enforce the minimum set-up time requirements for programming changes and to optimize its fault recovery strategies. It is also required that the service plane be able to respond to unscheduled changes in a timely fashion.

The Service Plane always assumes that any remote devices controlled over unacknowledged transmission paths require special considerations to ensure successful reception of any configuration changes, such as retransmission, and that the remote device is always successfully configured.

4.5.2.5 Legacy Considerations

The presence of the service plane makes the integration of legacy systems easier because it hides the low level equipment interconnections and configuration details from the high level content scheduling considerations. For example, a legacy traffic manager could command a collection of proprietary video transmission equipment to operate on specified content because its lack of knowledge of directly controlling video transmission equipment would be irrelevant.

4.5.2.6 SMPTE Activities

SMPTE should standardize the interfaces between the content-service planes and the service-path planes. Engineering guidelines and/or recommended practices may prove useful in explaining how these interfaces will behave in real installations. Whatever the vehicle, the industry will require direction on how content, services, and paths are modeled to allow the open exchange of such information across multiple vendors' equipment.

The following two items affect the support of managed resources:

- Vendor equipment with capabilities that exceed the standardized model require special considerations in order to utilize those capabilities. Such special considerations are beyond the scope of the standardization activities.
- Vendor equipment that lacks certain capabilities defined in the standard model require special considerations in order to properly manage those limitations. Such special considerations are also beyond the scope of the standardization activity.

Having stated these two items, any standardization activity that precludes the use of any significant percentage of vendor equipment of interest in a user facility runs the risk of being perceived as irrelevant to the user community.

4.5.3 Path Plane

The Path Plane represents the end-to-end connectivity from one device to another, or through chained devices, over a network of routers. The responsibility of the Path Plane is to transport the media content from one point to another to accomplish a particular task in a timely fashion and with a guaranteed quality of service.

A physical connectivity path is a connection between a source device and a destination device via a number of intermediate devices, i.e. a number of chained devices. The adjacent devices are connected through a network connection that is routed over the routers in the network. Control and management of the physical connectivity path involves allocation, setup, and monitoring of the components (devices and links between adjacent devices) that make up the path.

A logical and abstract representation of the connectivity path is desirable to simplify the interface for control of the content data flow. The concept of *Stream* is used to represent such a logical view and the controllable attributes of the content data transfer process. A Stream is a logical construction (object) corresponding to the transient state of content data being passed from one device to another through the physical connectivity path.

The Path Plane contains a collection of logical entities (objects) that collectively build, operate, and manage the connectivity. These entities can be categorized into two functional groups: (1) Streams that control the content data flow over the path; and (2) *Studio Resource Management (SRM)* that facilitates and manages the allocation and reservation of shared studio resources to meet the resource requirements of various tasks and achieve better resource utilization. Manageable studio resources include the devices that are registered with the device *registration repository* and the network connections and bandwidth. SRM can also provide a central point for resource *access control*. The Studio Resource Management functions may be carried out collaboratively by a number of logical entities such as Device Resource Management, Network Resource Management, and Path Construction Service that are described later in section 5.2.3..

4.5.3.1 Rationale

The path plane is a common service that provides fundamental network management. Because all network management services are abstracted by this common service, dependencies between devices can be managed. For example, specific bandwidth requirements for a path can be checked against maximum throughput for any given device that the path travels through. This device throughput can be managed as more paths are created that share the device.

4.5.3.2 Dependencies

Service Plane: The Path Plane responds to the requests from the Service Plane and returns acknowledgments along with necessary allocation information and reservation identifiers so that the Service Plane can execute their tasks over the reserved resources (path). The Path Plane also uses the device registration repository in the Service Plane as the basis for resource tracking, allocation, and reservation.

Device Plane: The Path Plane is dependent upon the Device Plane (devices and routers) in connectivity construction. Interfaces defined in the Device Plane should support the functionality of the Path Plane. The Path Plane coordinates with the Device Plane in resource allocation and reservation. In an allocation/reservation process, the SRM passes to the candidate devices detailed requests such as play time, bite rate, role (source or sink or processor), signal format, etc. An acknowledgment from the candidate devices informs the Path Plane if the request is accepted or not.

4.5.3.3 Functionality

The major functions of the Path Plane include:

- Constructing the media content connection path that satisfies the task specified by the Service Plane or high level applications.
- Providing centralized studio resource management services that
  - track studio resource usage and activity,
  - prioritize resource requests based on a set of predefined criteria, e.g. requester's authority level, task priority, and preemptiveness to ensure the availability of necessary resources to mission-critical tasks.
  - allocate shared studio resources to maximize resource utilization,
  - provide resource reservation for either an immediate or future task, ,
  - provide centralized security management and resource access control.
  - monitor studio resource status and activity, log and report alarms and failures.

4.5.3.4 Implementation Considerations

The Studio Resource Management services are important in the environment of shared resources. In some studios, however, where the resources are dedicated to individual tasks, SRM is optional.

There are two options in managing network bandwidth. One is that the Path Construction decides the network bandwidth allocation (route and priority) and communicates its decision to the Network Resource Management (NRM) for connection setup and tear down. This option gives the Path Construction complete control over network resources. It enables the Path Construction to schedule device resources in conjunction with the network connections to achieve higher utilization of the studio resources and a higher success rate in response to user requests. This option also provides the reservation capability for the network resource in the cases where the NRM is not capable of reserving bandwidth for a future time.

The second option is that the NRM allocates the network bandwidth upon requests from the Path Construction and coordinates with the Path Construction in resource reservation process. This option manages network resources separately and independently from device resources and reduces the complexity of the SRM. In cases that the network is not considered as the resource bottleneck, this option simplifies system implementation.

4.5.3.5 Legacy Considerations

- Integration/ Harmonization between switched video networks and routed data networks
- Path construction needs to address the needs that arise due to the above point
- Proxy support for non-network devices (i.e. RS232 controlled devices that participate in path construction activities)

4.5.3.6 SMPTE Activities

- Agreement on the functionality of studio resource management
- Standardize the interfaces for the Path Construction, the Device Resource Management, and the Network Resource Management

4.5.4 Device Plane

The device plane represents the collection of object classes that implement the interfaces used to access equipment throughout the studio. Devices can also represent the interfaces to software only objects that perform specific functions within the control network. These device objects need to support the following generic functionality:

- *Dynamic discovery* of the various *interface elements* used in the interface. This does not necessarily mean that the interface description information travels over any network, it may be derived from a local information database for the devices.
- Interfaces for later versions of a device need to be compatible with earlier versions of the interface.
- Interface elements need to be based on a consistent set of SMPTE defined broadcast types.
- All information that can be used to define the unambiguous operational state of a device needs to be made available in the device interface. That is, a device state can be completely represented (and therefore restored) by a snapshot of its current interface state.
- Device interfaces are made up of a hierarchical collection of sub-functions (functional blocks).
- Interfaces are extensible and therefore inherently backward compatible.

4.5.4.1 Rationale

Devices from various manufacturers will have different control interfaces. The device plane provides the necessary abstraction required to present the device to the network as a set of functions, as opposed to presenting devices by their different access interfaces. This also allows devices with similar functionality to be used interchangeably within the network affording more flexibility to the end user.

4.5.4.2 Dependencies

All Planes: Since the device layer represents a collection of object interfaces that abstract the physical device functionality, dependencies exist between all the planes and applications.

4.5.4.3 Functionality

4.5.4.3.1 Interface Identification

Interfaces to query descriptive information about the interface needs to be provided. Examples of this information are:

- Number of functional blocks. For each functional block in the interface:
  - Interface name (and location in the namespace)
  - Number of interface elements. For each interface element:
    - Element Label
    - Element Type (from SMPTE defined set)
    - ... (see properties below)}

Functions in this category will have inputs and outputs as follows:
Input – interface identifier.
Output – enumerations for functional blocks, interface elements returning descriptive info.

4.5.4.3.2 Interface Element Access

Specific functionality for reading and writing individual interface elements. Boundary condition errors and general access errors (R/O, W/O, R/W) need to be identified. Interface elements can be accessed directly or by an ordinate index representing each element of the interface.

Functions in this category will have inputs and outputs as follows:
Input – multiple input and output argument data types.
Output – output data argument or success or failure.

4.5.4.3.3 Interface Element Properties Access

A facility for querying the properties for each interface element needs to be provided. The following are example properties for each interface element:

- Element Name
- Element Description
- SMPTE data type
- Data Range
- Data Unity
- Data Calibration or Offset
- Element Access Permission (Read, Write, Create)
- Element Group Membership (Factory defined and user defined)

Functions in this category will have inputs and outputs as follows:
Input – channel and data to be sent, or buffers to contain the received data
Output – Success or Failure

4.5.4.3.4 Interface Element Group Management

A facility for collecting parameters into logical groups. Groups can be factory defined (native to the interface) or can be identified at runtime by the user. It is the intent that element grouping be used to assist in the generation of dynamic user interfaces (this grouping possibly defined through XML).

Functions in this category will have inputs and outputs as follows:
Input – interface element list + add/ remove from group operation
Output – Success or Failure

4.5.4.4 Implementation Considerations

- SNMP Support through standardized MIB extensions
- Identification of efficient mechanisms to translate existing legacy data types to the standard data types
- A mechanism for identifying the unsupported elements of an interface and the associated error reporting required for invalid accesses to the interface
- A mechanism for allowing a flat message based protocol to a device to be represented as a hierarchical collection of device sub-functions (reusable functional blocks)
- A mechanism for extending the functionality of an interface while maintaining original functionality

4.5.4.5 Legacy Considerations

- Standardizing existing legacy equipment interfaces
- Mechanism to allow legacy equipment to mimic equipment defined by a future interface.(ESLan example)
- Mapping of existing pseudo-standard/ standard devices into the new system (SMPTE Virtual Machines)

4.5.4.6 SMPTE Activities

- Standardized MIBS
- Standard list of device types with associated naming hierarchy
- Standard Interfaces for studio equipment classes
- Standard studio data types (i.e. time struct , video control , audio control , file formats, configuration)
- Migration of existing standardization (i.e. SMPTE virtual machines)
- Standard method of describing these interfaces (i.e. IDL, XML, etc.)

4.6 Functional Layers

Functional layers exist as the foundation on which the planes are built. They provide a homogeneous set of functions that facilitate the transport of control information throughout he network. The three functional layers are defined below:

- System
  - Provides common reusable system services as follows:
    - Studio Time
    - Configuration
    - Security
    - Identification and Naming
    - Fault Recovery
    - Status and Alarming
  - Abstract interface between the planes and the control network

- Communication
  - Provides a common facility for the exchange of information among communicating endpoints within a distributed network
  - Allows the use of multiple protocols simultaneously
  - Facilitates the transfer of time synchronized operations

- Transport
  - Provides for the reliable transmission of information throughout a network
  - Provides an abstraction for physical network access
  - Allows for adaptation to legacy networks

4.6.1 System Layer

4.6.1.1 Studio Time Function

The studio time service provides a representation of studio wall clock time to the other elements of the system. It consists of a collection of distributed objects that are synchronized via some appropriate means over the control network. Included in this service will be utilities to convert wall clock time to any appropriate reference signal as need by devices throughout the studio. It is assumed that some appropriate high resolution time format will be provided with appropriate accuracy for all operations within the studio.

This function within the system layer will depend upon the work already being done by the SMPTE Reference and Timecode working group. For the purposes of this overview, we are identifying a set of high level requirements for the software objects that will be used to interact with the time and reference capabilities of the control system.

4.6.1.1.1 Rationale

Within the studio, details regarding the presentation and distribution of wall clock studio time need to be hidden from the applications that require time synchronization. As new synchronization schemes develop, basic principles for managing and distributing studio time will need to change. These changes should only affect how this service is implemented as opposed to affecting the applications that depend on studio time. Also, various legacy synchronization schemes need to be handled and encapsulated by this service.

4.6.1.1.2 Dependencies

Content Plane: Studio time access for the scheduling of program and configuration material.
Service Plane: Studio time access for the purpose of scheduled resource reservation and stream latency calculations.
Path Plane: Studio time access for the purpose of logging time-referenced link activities and link latency calculations.
Device Plane: Studio time access for the purpose of synchronizing control activities within the device.
Communication System Layer: Studio time access for the purpose of synchronizing device-device transactions.

4.6.1.1.3 Functionality

4.6.1.1.3.1 Communication Layer Interface

Each communication transaction can be synchronized to the local nodes time. It is also assumed that a node is issuing a command to one or more other nodes within the system. Each of these nodes within the system may be fundamentally synchronized through signals of differing periodicity (but at some level, synchronized to one studio wall clock time).

The following types of synchronization should be provided for:
1. Immediate – perform the command as soon as possible. Disregard any fundamental synchronization period within the device.
2. Synchronized – perform the command on the next available synchronization period as appropriate for the device being controlled.
3. Delayed – perform the command on the nth synchronization period where n is the next synchronization period greater than the specified relative time.
4. Timed - perform the command on the next synchronization period that is the greater than the specified absolute time.

In the above descriptions, a typical 'synchronization period' for a device might be a vertical interval or a line interval. Also note that 2,3,4 are all 'clean' operations with respect to the processing of the data (or command) by the device.

4.6.1.1.3.2 Device Plane Interface

Device objects need to have access to the local nodes image of studio time. Functions to query studio time need to be provided. It is also assumed that devices will provide their latency information to requesting objects via a suitable API.

4.6.1.1.3.3 Path Plane Interface

Path plane objects need to manage the latency of various physical links. The studio time service needs to be able to assist the path plane in determining round trip latencies for various multi-tier network connections. This will likely include services to implement ping-response measurements across individual links within a multi-link path connection. This information would be used in command pre-roll calculations as well as help in identifying the accuracy with which any specific command could be delivered to a node in the network.

4.6.1.1.3.4 Service Plane Interface

Service plane objects need to manage the multiple overall aggregate latency (of multi-tier links). These separate path latencies are managed together along with specific device latencies to determine when and how to pre-roll each individual stream within a multi-stream program. The studio time service needs to be able to assist the service plane objects in determining the individual device latencies within the system.

4.6.1.1.3.5 Reference Binding

Functionality that indicates which of many-possible input reference signals should the studio time object be slaved on. This functionality needs to be available dynamically and should invoke required re-synchronization as required within the system.

4.6.1.1.3.6 Reference Translation

Functionality to convert any reference format to any other reference formats needs to be provided. This functionality would likely be accessed by devices in order for them to carry out studio activities.

4.6.1.1.3.7 Remote Synchronization

It is assumed that the studio time system level functionality will be distributed across several strategic nodes within the system. All other nodes can access these objects remotely (assuming network latencies are appropriate) or can shadow a slave image of the studio time service for local access. In any case, it is assumed that the master objects can negotiate a unified homogeneous image of studio time and present it to the slave objects and to the other objects within the system.

4.6.1.1.3.8 Resolution and Accuracy

It will be assumed that any device within the system has a defined synchronization granularity that is meaningful for the device. For example, where NTSC is concerned this resolution might be a field (for a frame) sync or a line (for a line sync). Where MPEG encoders are concerned, this synchronization unit may be more difficult to define (may need to be measured in clock-ticks for example). One assumption is that a device can buffer at least one (and possibly more) units depending on the device capabilities. This capability to manage chained devices will be kept to

the path plane. At the service plane, less granular units will be assumed, say a frame of NTSC video.

From a networking standpoint, the latency of any physical network will translated to an integral number of synchronization units for a specific device. In this case, the accuracy of any data packet to and from this device will need to be in the +/- 0.5 units range. This measurement is effectively the tolerable data jitter for the device and needs to be available for all devices supported in the network.

4.6.1.1.4 Implementation Considerations

- Devices will need to have access to a reference signal of some time. This reference will be converted to some presentation format and sent to a local studio time object for conversion to new studio time format, or conversion to another legacy reference format.
- A suitable accuracy needs to be defined for all operations that need to be synchronized to this studio time reference. It must be the case that the various physical networks that connect the studio devices must be capable of dynamically synchronizing communication devices to this studio time reference. This will require that the network support a data rate that can transport synchronization information between nodes faster than the smallest unit of time implied by the new studio time reference. If this is not the case, then it is required that a finite worst case latency for the link be established in order that synchronization algorithms can incorporate link delays.
- For the purposes of redundancy, the studio time objects need to operate in one of 3 ways:
  -- Synchronized to an incoming reference signal.
  – Slaved off of an existing studio time object (either local or remote).
  – Free-running – still providing reference, but accuracy would be suspect.
- For most operations, millisecond accuracy would be sufficient from a control point of view.

4.6.1.1.5 Legacy Considerations

- Translation between various legacy time reference signals and new studio time format is required
- A mechanism whereby legacy devices that recognize different time standards will need to be synchronized together
- Proxy reference objects need to be defined for legacy devices that cannot be adapted to support natively the new studio time system service

4.6.1.1.6 SMPTE Activities

- Presentation of a time format including date, time
- Synchronization protocol between objects
- Identification of which legacy formats should be supported natively and which should be proxied to
- Recommendation of new transport mechanisms as well as interaction with existing mechanisms

4.6.1.2 Fault Recovery Function

The fault recovery service exists to enable the capabilities of fault detection, logging, diagnostics, and correction in a system or across systems. Each system may provide (or implement) different levels of the fault recovery capabilities. A recovery process may involve a single system or a number of systems collaboratively, depending on the nature of the error condition and the capabilities of the involved systems. e.g. a device failure can be detected and automatically recovered by switching to a hot-stand-by redundant unit. A network failure may need reroute the traffic through a different path that involves multiple routers.

Fault recovery service also can be implemented in different layers of a system. Each layer should communicate to its higher layer about the error conditions and the correction actions that occur in the layer. This would allow the higher layer to take appropriate actions in case the lower layers are not capable of removing the error conditions to minimize the impact of the errors.

4.6.1.2.1 Rationale

To provide for the capability of automatically recognizing a network error, and successfully correcting it or routing around it, requires that devices communicate their fault status in a homogeneous way. This service provides the interface to this information as needed by the entities involved in making the decision as to how best to recover from any fault condition.

4.6.1.2.2 Dependencies

All Planes: All planes and applications are likely to be dependent on the fault recovery service.

Communication/Transport System Layer: Fault recovery dependends upon the Communication and Transport layers to deliver their diagnostic messages and correction commands across the network.

4.6.1.2.3 Functionality

The major functions of the fault recovery service are:

1. Guaranteeing that commands are properly delivered to and acknowledged by the involved management entities;
2. Logging errors and serving queries from high level control and management systems;
3. Coordinating with the Status and Alarm service in fault detection and reporting.

4.6.1.2.4 Implementation Considerations

- The timeliness and urgency required for any fault status needs to be set by the application and honored by the service

4.6.1.2.5 Legacy Considerations

- Proxies may need to be developed to handle existing fault recovery procedures and devices (i.e. GPI)

4.6.1.2.6 SMPTE Activities

- Standard data format for fault logging
- Standard interface for diagnostic and fault correction
- Modifications to existing status monitoring and diagnostic standards (SMDP)

4.6.1.3 Identification/ Naming Function

Identification and *Naming Services* exists to enable dynamic studio resource discovery and management. The *identification service* employs a unified *naming context* that uniquely identifies each studio components (e.g. individual devices). The registration service provides a centralized or distributed repository of all live studio components that are available for user access. A logical entity (object) called the *Service Gateway* is responsible for maintaining the repository and administers directory services to all the planes and applications.

4.6.1.3.1 Rationale

Studio resources will need to be managed by many services and applications, all with possibly different workflow models. Consistent naming of studio resources is required to allow these resources to share and collaborate effectively with the elements of the network.

4.6.1.3.2 Dependencies

Path Plane: Path Plane uses the Identification and Registration Service to find devices needed for mapping a high level Service Plan task into a detailed resource request. The need from the Path Plane will impact how devices should be categorized and what descriptive information about the devices need to be provided and obtained during the registration.

Device Plane: Registration is an interaction process between the Device Plane and the Service Gateway. The Device Plane should initiate the registration process along with providing the necessary information when the device is installed and activated in the studio.

4.6.1.3.3 Functionality

The generic functions of Identification and Registration Service include dynamic component registration and directory service.

Following is an example of how the Identification and Registration Service works.

A media server named 'TV01' is brought on line. It initiates the registration process by issuing a registration request to the Service Gateway and passing in the necessary information such as the identifier, object reference, device type, owner, property description and etc. The Service Gateway checks the validity of the request, e.g. if the identifier is unique, or if the required information is complete. If it passes the check, the Service Gateway adds a new entry to the repository; and acknowledges the success to the requester.

A third party user can query the Service Gateway for devices by issuing a search request and passing in the criteria, e.g. device name, type, location, owner, etc. The Service Gateway returns the object references of the devices that meet the criteria. The Service Gateway may, if needed, apply certain access control to requests.

The media server 'TV01' may be taken off line for maintenance. In such case 'TV01' should issue a request to the Service Gateway to either de-register itself or to remain registered but have its status changed to off-line. The Service Gateway checks the request validity e.g. the requester is the owner of this device; then it could either remove the 'TV01' entry from the repository or mark the status of 'TV01' as 'off-line.'

4.6.1.3.4 Implementation Considerations

The name of a studio component shall be compliant with a predefined naming context hierarchy. A decoder Device object of "Decoder01" shall be registered with a full path name of "/Components/Decoders/Device/Decoder01".

It may also be desirable that the studio naming context be compatible with the name space conventions of several public and private industry standards. Examples are:

ISO/ITU 8824-1 (ASN.1) which specifies "a human-readable notation for the precise representation of object identifiers within the ISO/ITU identifier hierarchy" (see SMPTE 298M), and/or
Open Management Group's (OMG) Common Object Request Broker Architecture (CORBA), and/or
Sun Microsystems Java/Jini.

4.6.1.3.5 Legacy Considerations

- Basic descriptive and classification information for existing legacy devices may need to be available online throughout the control system. This information will need to be provided by device manufacturers

4.6.1.3.6 SMPTE Activities

- Agreement on studio naming convention and device categorization
- Standards on Service Gateway interface
- Standards on device descriptive information format (device metadata)

4.6.1.4 Security Function

The Security Services Module of the Advanced System Control Architecture provides the following services[4]:

- Authentication: provides the verified identity of a service requestor.
- Access Control[5]: enforces access policy based on such information as the requestor's identity, the attributes of the requestor, and the attributes of the service requested.
- Confidentiality: provides privacy in the communication and storage of information.
- Integrity: verifies that communication and storage of information was unmodified.
- Non-Repudiation with Proof of Origin: ensures that a communication came from the expected source.
- Non-Repudiation with Proof of Delivery: ensures that a communication was received by the expected recipient.

4.6.1.4.1 Rationale

Security is an essential part of the network. Security services are required to help assure that resources (i.e. content, services, and devices) are used in an authorized manner. It helps to protect resources from inadvertent damage or modification, as well as protects intellectual property (i.e. content) from unauthorized use.

In addition, a scalable security model for the network is necessary to support applications that need to be portable across different implementations of the control system and to provide interoperability among these applications. The security model is scalable in that applications are able to interoperate and be portable among environments which have different security levels. Such a security model is necessary even when no security services are required by the environment in which the Control System is implemented. This enables applications to be portable among Control System implementations regardless of the level of security required by the environment. For example, where no security is required, requests for security services perform no action.

[4] The security service terminology of ISO 7498-2 Security Architecture is used.

[5] Access Control is often referred to as "authorization."

4.6.1.4.2 Dependencies

Security Services are globally available to all other objects. In general, Security Services may be used by a Module under the following circumstances:

- An application or a Module is making a request for service.

  The service provider may use authentication and access control services together to ensure that the service is being provided to an authorized requestor. An application or Module may use authentication services to ensure that the request is being made to the intended service provider. If a request for service is made over a network, then confidentiality, integrity, and non-repudiation services may also be used.

- A communication takes place over a network.

  The communication, which may be a service request or response, may make use of confidentiality, integrity, and/or non-repudiation services. The combination of all three services ensure that what was sent is not eavesdropped, that what was sent can be verified as such when received, and that the communication was both sent and received by the expected parties. Although network protocols provide some measure of data integrity, the integrity provided by network protocols does not prevent intentional modification of data.

- Data is stored on a device.

  Data storage may make use of confidentiality and integrity services. Use of both services ensures that what was stored will remain private, and that what was stored can be verified as being that which was initially stored. As is the case with network protocols, most storage devices inherently provide some level of data integrity. The integrity services provided by the Security Services Module additionally verifies that the data was not intentionally modified since the time that it was stored.

Whether a particular security service is needed by an application or Module depends on many factors. The cost of a security service, which includes the cost of implementation, the cost of use from the point of view of performance and user inconvenience, and the cost of administrating the service, needs to be balanced against the risk that the security of a resource may be compromised. The risk is estimated by computing the product of the probability that a vulnerability associated with a resource is successfully exploited and the cost of the loss which results from a successful exploitation.

Some Architecture Modules are more likely to have a need for Security Services than others. The Configuration, System Resource Management, and Fault Recovery Services are particularly critical to the safe functioning of the entire system. In addition, the Content Management Plane may require Security Services for protecting intellectual property.

4.6.1.4.3 Functionality

4.6.1.4.3.1 Authentication

Authentication operations are used to determine access to devices, content, and other services within the Control System. Authentication operations may be performed entirely within the security system which controls access to services in a manner transparent to service implementation. Authentication operations may also be available directly to the service implementation where the service implementation is solely or partially responsible for determining access to its operations. Authentication operations include:

- Obtain the identity of the requestor: This operation returns the identity, e.g., a User ID, given name of the service requestor.

- Obtain the attributes of requestor: Given the User ID of the requestor, return the attributes of the service requestor.

- Obtain the attributes of the resource: Given the resource name, this operation returns the attributes of the resource for which the operation is requested.

4.6.1.4.3.2 Access Control

Access Control operations enforce access policy to a service. Like authentication operations, controlling access to a service may be an internal operation of the security system and/or services may be wholly or partially responsible for enforcing access.

In addition to enforcing access control, access control operations include defining and managing access policies. Setting attributes of users, resources, and services are among the operations involved in managing access policies. Access control operations include:

- Enforce access to a resource: Given the identity of the requestor, the requestor's attributes, the attributes of the requested service, and optionally, other factors, determine whether access to the service in this instance is allowed.

- Manage the attributes and rules which determine access to services. There are many operations involved in such management including attribute and rule creation, modification, and deletion.

4.6.1.4.3.3 Confidentiality

Confidentiality operations provide encryption services for data. Secret (symmetric) key encryption algorithms are almost always used for bulk encryption because of their greater efficiency as compared to encryption algorithms using private/public (asymmetric) keys. Often the two types of algorithms are used together where encryption using asymmetric keys is used to encrypt the secret key which is used for bulk encryption. Confidentiality operations include:

- Encrypt data for transmission or storage: Given a block of data and a symmetric key, encrypt the data.

- Decrypt data received or accessed from storage.

4.6.1.4.3.4 Integrity

Integrity operations ensure that data is not modified during transmission or storage. As with encryption algorithms for confidentiality, the cryptographic algorithms used for integrity may be based on symmetric or asymmetric keys. Integrity operations include:

- Compute message authentication code (MAC) or digital signature of data: If a symmetric key system is used, given a secret key, compute message authentication code. If an asymmetric key system is used, compute a cryptographic hash and sign the hash with the user's private key.

- Verify message authentication code or digital signature of data. If a symmetric key system is used, given a secret key, message authentication code, and data, verify message authentication code. If an asymmetric key system is used, given a signed cryptographic hash and data, verify the signature of the hash with the public key and verify that the hash signed is the hash for the data.

4.6.1.4.3.5 Non-Repudiation

Non-repudiation operations are used to ensure that data came from the expected source. Non-Repudiation with Proof of Origin is usually implemented by the originator digitally signing the data. Non-Repudiation with Proof of Delivery is usually implemented by the recipient digitally signing a message indicating receipt of the data. In both cases, non-repudiation involves signing data using public key based cryptographic algorithms.

- Sign data using originator's private key: Given the User ID and data, compute a cryptographic hash and sign the hash using the user's private key. Note that the User ID is used within the security system to locate the user's private key for signing. Accepted practice requires that the user's private key for signing never be made available directly to applications.

- Verify signed data using originator's public key: Given the user's public key for signing, a signed hash, and data, verify that the signed hash is a cryptographic hash of the data signed by the user. Note that the user's public key for signing is readily available to applications.

4.6.1.4.4 Implementation Considerations

4.6.1.4.4.1 Control System Environment

The type of environment within which the Control System operates determines which security services are required. There are situations where no security services are required. For example, if the Control System environment consists of devices contained within the same room or building and linked by means of a local area network, then there may be no need for any security services. Traditional physical security techniques, i.e., locks and keys, may be sufficient for providing security in such an environment. In an environment where the Control System connects devices owned by the same enterprise or administrative domain, and where the connections are over a private network, minimal security services consisting of simple authentication and access control mechanisms may be sufficient. The network and the storage devices provide minimal integrity services. Even if the network in this environment is connected to a public network, a firewall can be used to prevent access to the Control System from the public network.

Requirements for security services become more substantial in an environment where the Control System connects geographically separate devices owned by different enterprises. In such an environment, the business relationships between enterprises and consequently, the levels of trust, may vary significantly. As a result, more complex authentication and access control mechanisms may be required. Furthermore, content traveling across a public network may require confidentiality and integrity services in order to protect intellectual property.

4.6.1.4.4.2 Security Technologies

There are many security specifications available for implementing secure distributed systems, e.g., Kerberos, OMG CORBA Security. Some combination of such security specifications may be needed to implement all of the Security Services described in this Section.

One of the most difficult but important choices to be made with regard to security technology is whether to use asymmetric (public) key technologies with a Public Key Infrastructure (PKI)[6] in

[6] Note that the choice not to use PKI does not mean that public key algorithms are not used within a security system. For example, SSL provides confidentiality and integrity by means of a shared secret exchanged between client and server using a public key algorithm. The term "PKI" refers not only to the use of public key algorithms, but also, to the technology and management infrastructure required to associate public/private key pairs with individuals, services, and/or processes.

providing security services. A key factor in this decision is how key management services, in terms of Registration and Certificate Authorities, will be provided. Key management services are a significant expense. Legal and liability issues are also key factors in the decision to use PKI. Laws governing electronic signatures differ from country to country and within countries. While it is possible to provide security services without using PKI, PKI provides a fully integrated solution for which there are several implementations. Furthermore, as a practical matter, PKI is the only solution for providing security services in a messaging environment over public networks, e.g., email.

4.6.1.4.5 Legacy Considerations

In developing a Control System based on this Architecture, there are almost always requirements for the integration of legacy equipment. Legacy equipment may have no security mechanisms, minimal security mechanisms insufficient for the Control System, or different security mechanisms from those of the Control System. In the first two cases, a proxy device can be created which controls the legacy device and which is compatible with the security mechanisms of the Control System. Where legacy equipment has different security mechanisms from the Control System, it may be possible to only proxy the security mechanisms of the legacy device.

4.6.1.4.6 SMPTE Activities

It is unlikely that SMPTE would need to undertake the development of new security technology in order to meet the needs of the Control System. However, SMPTE will need to choose among existing programming interfaces to security services, or develop programming interfaces to security services, so that applications can interoperate with each other and be portable among Control System implementations.

SMPTE will also need to develop environment classifications based on the level of security required. This will enable applications to interoperate and be portable among different environments, as well as, different Control System implementations. For each of these environments, appropriate profiles of existing security specifications and technologies are developed. These profiles may include:

- Developing or choosing additional programming interfaces
- Choosing authentication and access control mechanisms
- Choosing cryptographic algorithms

For example, one classification, representing the lowest level of security, would include the environment consisting of devices contained within the same room owned by a single administrative domain. The profile for this lowest level of security services would be "null," i.e., no security services. Even though there are no security services required by this environment, requests for security services could be made by applications. Such requests from applications in this environment would result in access always being allowed and/or no action being taken.

Another classification example would include the environment consisting of devices which are geographically distributed but are connected by a private network and controlled by the same enterprise. The profile for this classification might include only authentication services. All other services would be "null."

4.6.1.5 Configuration Function

Configuration services exist to enable the configuration of objects within the management system. Each plane represents a different level of configuration. Status messages will report back to the configuring application that the configuration is either successfully completed or properly scheduled to occur at the specified time. In addition to discrete configuration commands, objects within planes will be capable of accepting template configuration commands. These templates,

or "pre-sets," enable an object to load a desired state (or partial state) from a pre-determined set when commanded.

4.6.1.5.1 Rationale

By providing configuration management services through a common interface, applications and higher order services will be able to configure different types of equipment in a similar way. Through this mechanism, device manufactures can provide their specific device *configuration templates* (possibly even proprietary) to the applications that use their devices. The devices can be treated as functions within the studio leaving specific device configuration to the individual manufacturer providing the equipment.

4.6.1.5.2 Dependencies

Path Plane (or possibly all planes): Specifically for the generation of reservation, discovery, command presets requests.

Status/Alarm System Layer: Specifically for the communication of device diagnostics within the control system.

4.6.1.5.3 Functionality

The generic functions of configuration services will be capable of guaranteeing that commands are properly delivered and acknowledged (within the limits of the physical command communication medium). Following is an example of a configuration services operation. This example uses path management, but configuration services are applicable to all planes.

There could exist a variety of path plane configuration functions:

- Configure a path using a basic pre-determined state (template)
- Query configuration status of resource
- Query available configuration templates from a resource
- Associate template 'n' for a given resource with a label

The service plane application is made aware of an MPEG-2 encoding resource in the path plane and would like to use it to implement a service. A configuration command is created:

```
Configure_Path(·
      encoding_resource_identifier,
      template_identifier,
);
```

This command addresses the appropriate exposed resource and requests that it configures itself according to a stored template. However, the additional bit rate term allows the service plane to slightly alter the state setting of the encoding resource to tailor it to the specific service desired.

Configuration services take this command and send it to the correct resource (or resource proxy) in the path plane. If the resource acknowledges receipt of the command, an appropriate message is sent back to the service plane application. The same holds true when the resource acknowledges processing (successful or otherwise) of the command.

4.6.1.5.4 Implementation Considerations

Since many of the managed resources may be connected to the management system via either a Local Area Network or even a remote connection, the latency from the time a command is issued to the time the resource completes the configuration can be very long. The management system must properly account for these latencies and for the variation in these latencies. While a fixed latency is easy to manage, using such familiar mechanisms as "pre-roll", the variation in latency is more difficult. For applications where the timing of specific commands is very critical, both the

management system and the managed resource must provide mechanisms to permit these configurations to be effected with a high degree of timing accuracy (q.v. "Studio Time").

If the resource does not have a means for acknowledgment (such as in-band control of remote devices over a one-way data path), the Service Plane application is responsible for ensuring that the resource receives the command (such as multiple command transmissions) and must assume that the resource always successfully completes the configuration without any acknowledgment.

Managing unacknowledged resources requires that the management system's Service Plane application provide a mechanism for ensuring that the configuration is successfully relayed to the managed resource. One such mechanism is the repeated transmission of such a command to the managed resource. Furthermore, the management system's Service Plane application must assume that the managed resource always successfully completes every configuration command. It is the responsibility of the managed resource itself to ensure that it can successfully configure itself properly according to the management system's requirements.

Finally, there may be limits on the types of configuration transactions that can be expected of an unacknowledged remote managed resource. E.g. avoiding configuration transactions that require an ordered set of commands in order to properly configure the managed resource, since command retransmission to ensure successful reception makes the command order ambiguous.

4.6.1.5.5 Legacy Considerations

The presence of the Configuration Services makes the integration of legacy systems easier because it resolves the different configuration requirements and limitations associated with legacy systems or devices and their ability to successfully complete any management activity.

4.6.1.5.6 SMPTE Activities

- Describe syntax and facilities to implement configuration services
- Develop a set of common reference templates for each class of equipment (e.g. VTRs, routers, encoders, etc.)

4.6.1.6 Status/ Alarms Function

Status and Alarm Services exist to communicate diagnostic messages between various entities at all planes of the control system. Status/alarm messages need to be delivered in a reliable and timely fashion so that they may be acted upon appropriately. Each plane may have different requirements for message latency and the level of detail exposed by each message. However, it is important for these messages to support the encapsulation of lower level messages to enable detailed reporting if so desired by applications.

4.6.1.6.1 Rationale

Treating all diagnostic messaging in a common way, allows the development of common, re-usable applications that process these messages. Common logging applications, and event management applications can be defined and re-used throughout the control network.

4.6.1.6.2 Dependencies

Planes: All planes that participate in redundancy operations will be dependent on the Status/Alarm services.

Communication Layer: Status/Alarm services are dependent upon the communications and transport layers to deliver their content across the management system. The latencies associated with those layers have a direct impact on the management system's ability to properly respond to system anomalies. Therefore these latencies must be explicitly specified. Unacknowledged resources, which have no mechanism for sending alarms or status to the management system are assumed to always be operating properly. Any fault within such a managed resource must be resolved locally without any assistance or even the awareness of the management system.

4.6.1.6.3 Functionality

The generic functions of status/alarm services are capable of guaranteeing that such messages are properly delivered and acknowledged (within the limits of the physical command communication medium). Following is an example of a status/alarm services operation. This example uses service management, but status/alarm services are applicable to all planes.

There could exist a variety of path plane configuration functions:

- Masking/filtering/regulating alarms
- Alarm tracking
- Alarm acknowledge/clear

The path plane application detects a fault in one of the devices that is delivering a particular service. It translates this information into a higher level description that is suitable for processing by the service plane application and sends it:

```
Assert_alarm (
      service_plane_alarm_identifier,
      source_identifier,
      encapsulated_device_message
);
```

This command asserts a certain defined alarm with an associated source and destination. It also encapsulates a lower level device error message (perhaps a brief text string) that will enable a service-plane-application user to more clearly understand the nature of the fault.

The Service Plane assesses the fault and any dependencies that may exist with other programs. It also initiates recovery strategies, such as redundant sources or paths, if possible. Status/alarm

services will then take this message and send it to the correct entity in the service plane. When that entity acknowledges receipt of the message, an appropriate message will be sent back to the path plane application.

4.6.1.6.4 Implementation Considerations

Status and alarm reporting latencies must be explicitly defined since they affect the fault recovery strategies available to the management system. Unacknowledged resources must have local mechanisms for fault management and recovery or accept the possible loss of service as an operational policy. Also, standardized MIB extensions and reusable trigger objects need to be provided.

4.6.1.6.5 Legacy Considerations

Legacy systems are supported through the encapsulation of lower level messages. These messages could be current-day notifications that are not in the form of the management system's object model. In this way, a new application can communicate a resource's legacy messages to other applications in the management system.

4.6.1.6.6 SMPTE Activities

- Definition of standardized MIB extensions and reusable trigger objects
- Describe syntax and facilities to implement status and alarm services
- Develop a set of common reference status and alarm messages for each class of equipment (e.g. VTRs, routers, encoders, etc.)

4.6.2 Communication Layer

The communication layer provides for higher order communication services that are built upon the services available at the transport layer. It provides a portable facility for the exchange of control information among communicating endpoints within a distributed control network. The following general functionality needs to be provided for:

- Portable to an accepted set of protocols and environments.
- Supports *multipoint* binding across different channels of the transport layer (possibly across different physical interfaces) .
- Provides for time synchronized operation throughout the control network.

4.6.2.1 Rationale

The communication layer provides for protocol independence within the network. Services built to take advantage of the communications layer will be able to communicate over different protocols as required. As communication protocols evolve, or as facilities extend their services across Metropolitan and Wide Area Networks, different protocols will need to be incorporated into the network control structure. The communications layer provides the necessary abstraction for this.

4.6.2.2 Dependencies

The communication layer is the foundation for all control activity. Dependencies exist to all upper layers. services will depend on the interfaces that are available for each of the supported protocol drivers. The communications layer will depend on the functionality of the transport layer.

4.6.2.3 Functionality

4.6.2.3.1 Instantiation

Interfaces for creating/ allocating endpoints within the control network and enabling binding to various supported protocols. This multi-protocol support needs to be provided seamlessly. It should be possible to support multiple bindings using different protocols.

Functions in this category will have inputs and outputs as follows:
Input - arguments that specify the types of protocols or physical networks to use.
Output – a pointer or handle to the logical connection.

4.6.2.3.2 Signaling

A facility for sending meta-data regarding the status of a specific connection or status for any of the hosts involved in a connection.

Functions in this category will have inputs and outputs as follows:
Input – specific signaling data to be sent or specific information to be received. Examples of signaling information may be communication up/ down, or specific communication parameters sent during a multi-tier connection setup.
Output – Success or Failure.

4.6.2.3.3 Transactions

Transactions are collections of operations that need to be executed together, possibly at a specific timecode (see timecode in the System Layer). Transactions need to function across hosts as well as within a single host. Facilities for the following attributes need to be provided:

- Multi-tier connectivity
- Operation validation
- Operation rollback
- Batched operations
- Synchronized operations (to timecode)

Functions in this category will have inputs and outputs as follows:
Input – Connection handle + transaction context specification.
Output – Success or Failure.

4.6.2.3.4 Notifications

Device status needs to be transported via the communication services. The following functionality needs to be provided:

- Support for polled/ unsolicited notifications.
- Flexible priority assignments for notification services

Functions in this category will have inputs and outputs as follows:
Input – Connection handle and optional setup data.
Output – Notification information or Success/ Fail if setup operations.

4.6.2.4 Implementation Considerations

- Memory/ Buffer management between the transaction and notification services and the rest of the object environment.
- Transactions need to be non-blocking.
- Need to provide support for unsolicited notifications and polled notifications as necessary (as required by the underlying protocols).
- Byte ordering (network or big endian recommended).
- Ideally, protocols services can be inserted into the communication layer (i.e. as plug-ins with some glue software).
- Versioning of the resultant protocol services will need to be handled. For example, version 2.1 of the transaction services should talk to version 2.0. It is likely that major revision changes (e.g. 3.0) will not be able to talk to earlier versions (e.g. 2.1).

4.6.2.5 Legacy Considerations

- Support for existing studio control protocols
- Support for unacknowledged protocols
- Support for devices that cannot provide for time synchronized operations
- Support for manufacturers providing static data describing the communication capabilities of a device. This data will be used to configure an appropriate subset of the communication capabilities for this device
- Support for operating the facility with a subset of its interface features. For example, unacknowledged protocols or devices that cannot provide for time synchronized operations. It is necessary for the control environment to be able to query the supported functions of the facility

4.6.2.6 SMPTE Activities

- Acceptance criteria for off-the-shelf technologies
- Recommended protocols and interfaces
- Classifications for the adaptation of existing studio protocols
- Transport interface definition

4.6.3 Transport Layer

The transport layer provides for the reliable transmission of control information throughout the network via channels. A channel is a logical pipe that exists between 2 communicating endpoints on a network. An endpoint can represent a device, a port on a device, an interface to a software object, or any other source or sink point for control information.

The following general functionality needs to be provided for by these channels:

- Support for polled/ and *asynchronous* protocols.
- Support multiple different protocols/ physical networks with in the same system.
- Support for efficient block/ stream transfers (periodic data) as well as control messaging (asynchronous data).
- Scatter/ Gather capability with message order guarantees.
- Configurable prioritization.
- Support for *bandwidth reservation* and deterministic service (i.e. QoS).
- Capable of incorporating existing protocols (TCP/IP), emerging protocols, and legacy studio protocols .

4.6.3.1 Rationale

The transport layer provides for physical network independence within the network. Services built to take advantage of the transport layer will be able to communicate over different physical networks as required. As networking technology evolves, or as facilities extend their services across different physical network topologies, different physical networks will need to be incorporated into the control structure. The transport layer provides the necessary abstraction for this.

4.6.3.2 Dependencies

The transport services will depend on the interfaces that are available for each of the supported protocol drivers. The communications layer will depend on the functionality of the transport layer.

4.6.3.3 Functionality

4.6.3.3.1 Instantiation

Interfaces for creating/ allocating new channels and destroying/ de-allocating existing channels need to be provided. This also includes the ability to support multiple protocols that may exist in the system. It should be possible to support multiple protocols attached to multiple physical network interfaces simultaneously in the system.

Functions in this category will have inputs and outputs as follows:
Input - arguments that specify the types of protocols or physical networks to use.
Output – a pointer or handle to the channel for reference.

4.6.3.3.2 Binding

Binding refers to the act of creating logical connections between distributed endpoints within the studio network. Facilities for identifying endpoints, establishing the connections, as well as conditional acceptance or rejection of these connections based on other criteria (possibly security).

Functions in this category will have inputs and outputs as follows:
Input – connection (location of destination endpoint and local channel reference) to be created, removed, accepted, rejected.
Output – Success or Failure.

4.6.3.3.3 Transmission

A facility for sending data to a channel, and receiving data from a channel needs to be provided. Also, a means of unsolicited notifications for event and event group triggers.

Functions in this category will have inputs and outputs as follows:
Input – channel and data to be sent, or buffers to contain the received data.
Output – Success or Failure.

4.6.3.3.4 Transport Channel Setup

Channel setup includes the identification of specific configuration attributes for the channel. Facilities for identifying the following attributes need to be provided:
- Channel Priority
- Quality of Service (Bandwidth required, latency, etc.)
- Stream, Block, or Message type performance indication.
- Maximum data size

Functions in this category will have inputs and outputs as follows:
Input – channel handle + configuration specification.
Output – Success or Failure.

4.6.3.3.5 Transport Channel Status and Statistics

Channel statistics and status needs to be maintained by the transport facility. Functions to retrieve the following information need to be provided:
- Channel Errors
- Lost data
- Current number of endpoint bindings through channel
- Set data rate.
- Actual data rate.

Functions in this category will have inputs and outputs as follows:
Input – Channel handle.
Output – Statistics information.

4.6.3.3.6 Communication Event Management

Communication event management focuses on the detection, generation, and manipulation of transport channel events that are of particular importance to the environment. Communication events can be individually enabled, disabled, and filtered for each channel. Specific events in the system can be identified by an *event handle*. Notifications as to when the particular event has been triggered can be generated.

Examples of transport channel events are:
- Arrival of incoming data units
- Incoming connections
- Connection completion
- Disconnection completion
- Channel errors

Functions in this category will have inputs and outputs as follows:

Input – type of event to be created (from a list of supported events).
Output – event handle.

4.6.3.3.6.1 Specific Dependencies

Events will likely be used to trigger various processing operations such as redundancy changeover. For this reason, events may indirectly effect entities within the system layer.

4.6.3.3.7 Event Group Management

Events can be grouped into logical collections that are meaningful for the operation required throughout the control environment. Event group management refers to the functions that facilitate events being grouped together to form event groups which can then be identified by an event group handle. A facility for adding/ removing *event members* to/from groups needs to be provided. Notifications as to when the particular event group has been triggered can be generated.

Functions in this category will have inputs and outputs as follows:
Input – event list for group creation, or single event to be added to / removed from an existing group.
Output – group handle.

4.6.3.3.7.1 Specific Dependencies

Events will likely be used to trigger various processing operations such as redundancy changeover. For this reason, events may indirectly effect entities within the system layer.

4.6.3.4 Implementation Issues

- Memory/Buffer management between the transport services and the object based control environment
- Services in this interface will need to be non-blocking
- Ideally, protocols will easily insert into the defined protocol layering within the transport layer (i.e. as plug-ins with some glue software)
- Versioning of the resultant protocol services will need to be handled. For example, version 2.1 of the transport services should talk to version 2.0. It is likely that major revision changes (e.g. 3.0) will not be able to talk to earlier versions (e.g. 2.1)
- Need support for polled protocols as well as multi master protocols

4.6.3.5 Legacy Considerations

- Support for popular existing studio control protocols
- Support for operating the facility with a subset of its interface features. For example, unacknowledged protocols or protocols that inherently are not able to gather statistics need to be supported with a reduced functionality. It is necessary for the control environment to be able to query the supported functions of the facility

4.6.3.6 SMPTE Activities

- Acceptance criteria for off the shelf technologies (real-time characteristics)
- Recommended protocols and interfaces
- Classifications for the adaptation of existing studio protocols
- Transport interface definition
- Minimum or mandatory sets of functionality to be available in all systems