This paper introduces the input/output facilities of DPPX, the Distributed Processing Programming Executive for the IBM 8100 Information Processing System. Design requirements and alternatives are discussed, as well as the general structure of the services that implement the I/O facilities. Services that support specific I/O resources, such as disk storage and communication devices, are related to the general structure. The paper considers some of the problems in designing a general structure to support a wide range of services, and it briefly describes the interface architecture used to solve these problems.

I/O facilities of the Distributed Processing Programming Executive (DPPX)

by H. R. Albrecht and L. C. Thomason

The Distributed Processing Programming Executive (DPPX) is a new communications based operating system designed for IBM's multipurpose 8100 Information Processing System.[1] As a control system for distributed processing, DPPX supports both horizontal distribution among interconnected 8100 systems and vertical distribution between the 8100 and System/370, 303X, and 4300 processors.[2] DPPX design concepts are oriented toward a general structure that is adaptable to a broad set of distributed processing requirements. Both the structure and the current implementation are designed to be readily adaptable to future enhancements as new requirements emerge.

This paper introduces the input/output facilities of DPPX, and it discusses the structure of the services that form those facilities. Emphasis is placed on why the I/O services are structured as they are, rather than how the structure is implemented. Descriptions of the design and implementation are readily available elsewhere.[3] Although the paper concentrates on the structure, it also introduces key elements of the architecture and interfaces implemented within the I/O facilities of the system.

In the discussion that follows, the first section, *Requirements*, introduces the primary design requirements for the I/O structure and outlines how those requirements were met. The second section, *General structure*, introduces the I/O facilities and the general structure of the supporting services. It describes a general conceptual service structure and indicates why the approach was chosen. The third section, *I/O service facilities*, introduces the specific I/O service structures that support each of the facilities. The fourth section, *I/O service architecture*, identifies important aspects of the I/O services architecture.

Requirements

It was clear during the initial design stage that the success of DPPX would hinge on satisfying a broad set of requirements at a significantly lower cost than System/360 and System/370 operating systems. Further, the initial system was expected to be the base for future software development.

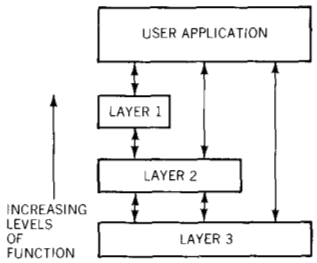
The functional requirements for DPPX were multidimensional.[4] They encompassed both decentralized data processing and distributed processing. They included batch and interactive communications, data formatting for display terminals, and transaction based processing with data recovery.[2] In addition, the system was to support a device-independent I/O interface for applications written in high-level languages (principally COBOL) while permitting several other interface levels for programs written in PL/DS (a high-level language for use by system programmers)[5] or assembler language.

There are three major alternatives in designing a system or a set of I/O facilities. The first is an unstructured, single-package approach. The second is a subsystem approach in which each subsystem provides its own set of tailored services. The third is a layered structure with a common set of well defined services.

The first approach is characterized by modules of code through which many functions are interwoven, and the interfaces between functions are loose or informal. It appears to be advantageous for a small set of external functions with a single level of interface, since it has some advantages in terms of performance and initial development cost. However, with this approach it is difficult to add new functions later, and loosely defined and informal interfaces are difficult to manage if more than a few people are involved. For the broad set of functional requirements facing DPPX, this approach was not considered feasible.

The second approach (tailored services) has two advantages. It offers maximal flexibility for developing new functions indepen-

Figure 1 Functional layers

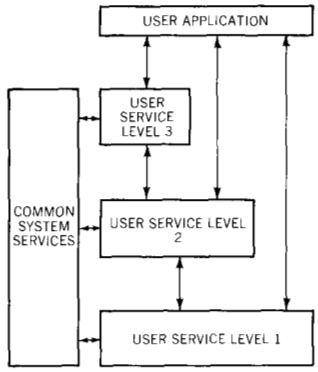


dently (as separate subsystems), and it allows for services designed specifically for the particular subsystem. CICS (the Customer Information Control System) is a good example, with its unique storage management, program management, and task management services.[6] However, this approach has two major disadvantages. One is the complexity (to the user) of dealing with several subsystems, each with its own user interfaces, configuration restrictions, tuning requirements, and installation and service characteristics. The other disadvantage is that many functions must be duplicated, so more code must be written.

The third alternative is a layered structure in which hierarchical functions are provided. Each major function is implemented as a separate layer which occupies a particular level in the hierarchy of layers, as shown in Figure 1. The function is used, without change, both by higher-level functions and by users who wish to bypass higher levels of function (or exclude them entirely). This approach tends to minimize redundancy while maximizing flexibility for changing or extending a particular function (layer) without affecting the various users, so long as the interface to the function is preserved.

The hierarchy in Figure 1 represents increasing function and ease of use. The hierarchy provides increasing levels of service to the user (application program). Each layer itself requires system services (such as storage and task management) in implementing its function. By providing a common set of system services, as shown in Figure 2, redundancy is reduced further. The combination of a layered structure of functional services and a common set of system services minimizes redundancy.

Figure 2 Common system services



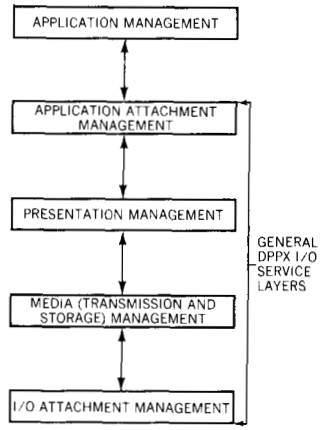
Consideration of the factors discussed above led to selection of a layered structure for DPPX as the most effective way to minimize the cost of a broad set of distributed processing functions.

General structure

IBM's Systems Network Architecture (SNA),[7] introduced in 1973, is a conceptual framework and information format that allows communication software to be developed in a layered structure. Before SNA was developed, layering was difficult to achieve because line control and device control procedures were merged in a single operation. Since then, the understanding of layered I/O software design concepts has advanced.[7,8] Thus in the development of DPPX it was possible to refine and apply the concept of layering to an even broader scope of problem, encompassing not only SNA communications, but also non-SNA communications and data management. This refinement led to the general definition of I/O layers given in Figure 3.

The Application Management layer encompasses both the user's application programs and the management services that support them. The application programs are command processors or transaction processors, usually written in COBOL (FORTRAN, PL/DS, and assembler language are also available). The Application Management services currently provided by DPPX include terminal monitoring functions as well as application scheduling functions. They are provided by the Command and Transaction Management facilities of DPPX.[1]

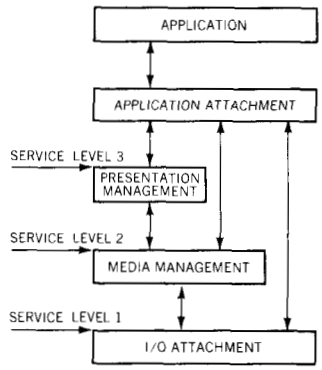
Figure 3 General layer service functions



The Application Attachment Management layer directly supports the application layer's interface with the I/O services. The applications use macro instructions for establishing connections with I/O resources and sending (output) and receiving (input) data. These services also provide integrity checking, synchronization between application and I/O processes, and event-driven interfaces for notifying the application of asynchronous conditions. By providing these services as a separate layer, redundant implementations are avoided in the other layers. Figure 4 illustrates the support provided for several levels of service by the same Application Attachment layer.

The Presentation Management layer allows an application to be device-independent. It provides for the presentation of data from an input source to an application, and from an application to an output sink. The principal general-purpose source and sink is a keyboard display terminal. Communication between application programs also is supported. In addition, DPPX controls the presentation of logical records to and from data storage media.

Figure 4 Common Application Attachment layer

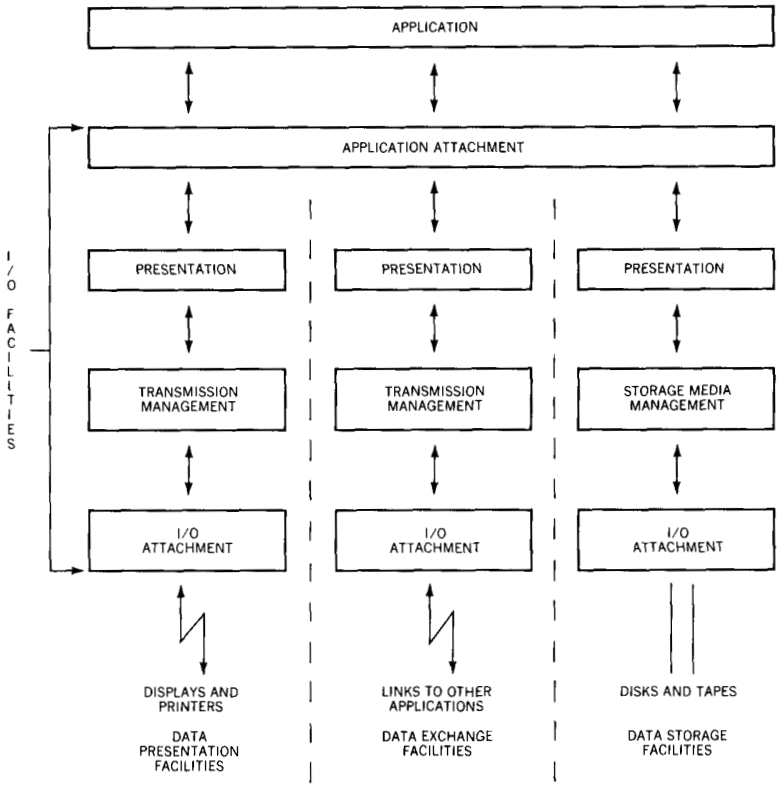


The Media Management layer controls both transmission and storage media. Thus Media Management includes both a portion of the SNA transmission management functions[7] and a portion of the traditional data management function. It provides for the sharing of a single physical transmission path among many logical transmission paths (SNA sessions).[9,10] It also provides for the sharing of physical storage media among many data sets and of a data set among several users.

The I/O Attachment layer supports physical I/O adapters, including communication adapters. It encompasses the data link control functions of SNA, and it includes support of directly attached terminals and storage devices (disk, diskette, and tape).

During the evolution of the system design, it became clear that the simple layer terms used above did not adequately describe the unique purpose of different combinations of layers. For example, there is commonly a need to discuss data management functions (as contrasted with data communication functions) as a single related collection. The problem was to allow discussion of unique

Figure 5 DPPX I/O facilities



sets of functions such as data management while retaining the notion of common layer functions such as presentation services. To solve this problem, the concept of *I/O facilities* was added to the system design. Each I/O facility provided by the system denotes a particular set of I/O services and physical I/O resources that are used to accomplish a particular purpose.

There are three major purposes for I/O. The first is data storage and retrieval (data management). The second is the presentation of data to an end user by some external medium such as a display terminal or printed output, and also the input of data from a user. The third is the exchange of data between application programs. These three purposes are accomplished, respectively, by data storage facilities, data presentation facilities, and data exchange facilities. Each uses a common, general I/O service structure, as shown in Figure 5.

As a result, the application layer perceives a common interface mechanism (including macro instructions and control blocks) for all three facilities. The application program may, in fact, use a

device-independent interface that allows the type of facility to be transparent. Thus the application may communicate with a data set during one invocation and a terminal during another without change to the application program. The application program may, alternatively, use higher- or lower-function interfaces that are unique to the particular facility while still using a common set of basic interface mechanisms.

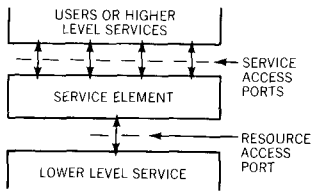
As a layered system is being developed, the internal design of each layer can proceed independently of any attempt to achieve commonality with other layers (except for those common interfaces that must match). Although this approach tends to allow initial freedom and speed in design (indeed, independent design is one of the reasons for a layered approach), it leads to a number of difficulties. If software functions built with widely different structures are combined (as required for grouping services into a higher-level function), the result can be costly and can lead to poor performance. For example, if the design requires that one layer of a system have access to resources by means of a CALL macro interface, and another by means of a COMMAND interface, it is difficult (and in some cases impractical) for the function that supports the macro to invoke the command in a hierarchical sense. To solve this problem, the I/O layers of DPPX are designed to use a common set of internal concepts and interfaces when appropriate. These common concepts and interfaces are discussed under *Service elements* and *Resource Manager elements*, below.

In general, I/O software provides two basic types of functions, those associated with handling requests for service and those associated with managing the resources of the layer. The distinction between these two types of functions is made explicit in DPPX by the terms *service* and *resource manager*. The service portion of a layer includes the elements that support requests and responses to requests from users of the I/O layer. These functions are commonly requested by SEND and RECEIVE macros (equivalent to GET/PUT and READ/WRITE macros in other systems). The Resource Manager portion of a layer includes those elements of an I/O layer that control the use of resources and services provided by the layer. Those functions normally are requested by commands that define resources (such as data sets and terminals) and allow users to access resources (CONNECT and DISCONNECT macros).

service elements

The service element of each layer provides I/O service to connected users (or higher-level service layers). Each connection is represented by a service access port. The user of a higher-level service views each access port as a separate resource. Early in the design of DPPX it became clear that a way was needed to distinguish various types of service functions.

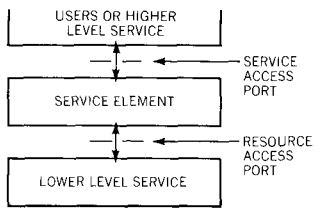
Figure 6 Resource sharing structure



Some services provide a resource sharing function. Examples are Storage Media Services, which provides shared access to data sets and to direct-access volumes, and Transmission Media Services, which provides shared access to data links. Such service elements have the capacity to relate many service access ports to a few resource access ports, as shown in Figure 6.

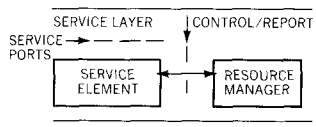
Other services do not support resource sharing. Generally, these are services that provide only transformation or a particular mode of access to a resource. For such service elements there is always a one-for-one mapping of service access ports to resource access ports, as shown in Figure 7. Such mapping is characteristic of the current Presentation Services and Application Attachment layers in DPPX.

Figure 7 Simple access structure



The I/O Attachment layer supports services of both types, although the majority of supported attachments do require resource sharing. In some cases, multiple devices share a single hardware adapter. (In this sense an adapter is a hardware unit that moves data and commands from the processor to an attached device. Generally it can be used by only one device at a time.) In such cases, many service access ports may map to a single hardware resource (adapter). The hardware adapter is not a standard DPPX software port, but each adapter does represent shared resource access points and thus is treated as a shared access port to maintain the common concept.

Figure 8 Resource status reporting and control

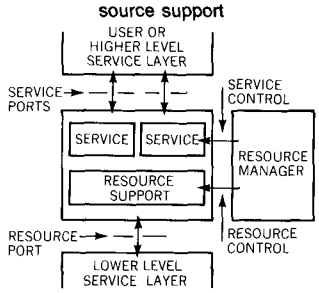


As discussed above, the primary function of service elements is to support service requests from connected users. A secondary function is to provide status reporting and resource control on behalf of the Resource Manager. The interface for these secondary functions is shown in Figure 8 as the control/report interface between the service and Resource Manager elements of a layer.

Resource Manager elements

The Resource Manager portion of each layer controls the use of services provided by the service element. Thus it controls the setting up and taking down of connections (service ports) as requested by the user.

Figure 9 Resource Manager relation to service and resource support



The Resource Manager also controls the activation and deactivation of shared resources. The service element, under control of the Resource Manager, provides the actual creation and deletion of resource sharing support (such as modules and control blocks). This function applies only to those layers that support sharing.

Figure 9 illustrates relationships discussed above. Examples of resource control are:

- Creation of storage media support for a direct-access storage volume.

- Creation of transmission support for a communication link.
- Creation of I/O Attachment support for a communication adapter.

Examples of service control are:

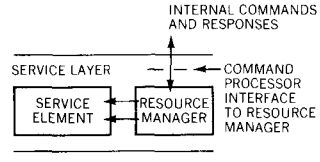
- Creation of a data set access port in the storage media layer (many data sets share a direct-access storage volume).
- Creation of a communication access port in the transmission layer (many communication sessions share a communication link).
- Creation of a communication terminal port in the I/O Attachment layer when several terminals are connected to one adapter.

Resource control functions generally are provided by the Resource Manager as requested by the system or network administrator. In their external form, these requests are viewed as operator commands. In their internal form (after parsing), they are viewed as Resource Manager commands.

Service control functions generally are requested on behalf of application programs or higher-level service layers as a result of CONNECT or DISCONNECT macro commands. In their internal form, they also constitute Resource Manager commands.

Figure 10 depicts the command processor interface for both types of commands discussed above. For example, an internal command that requests support for a named data link would go to the I/O Attachment-Link Resource Manager, resulting in a CREATE SUPPORT command to the service element with appropriate link parameters.

Figure 10 Resource Manager command interface



Many Resource Managers use resource profile or service profile data sets, or both. Resource profiles can be characterized as descriptions of specific, named resources. For example, the Resource Manager for storage media uses resource profiles for volumes and data sets. Both types are maintained in the system catalog. Other Resource Managers use separate data sets to maintain their profiles.

Service profiles can be characterized as descriptions of a selectable level of service or mode of access. A given service profile may apply to a number of resources and service ports. Some Resource Managers maintain service profile information with each resource profile. Others maintain separate service profiles.

Figure 11 illustrates the Resource Manager's access to profile data sets. The Resource Manager uses the standard data set services or catalog services.

Figure 11 Resource Manager profile data set interface

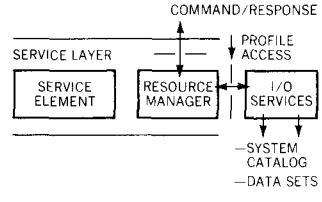
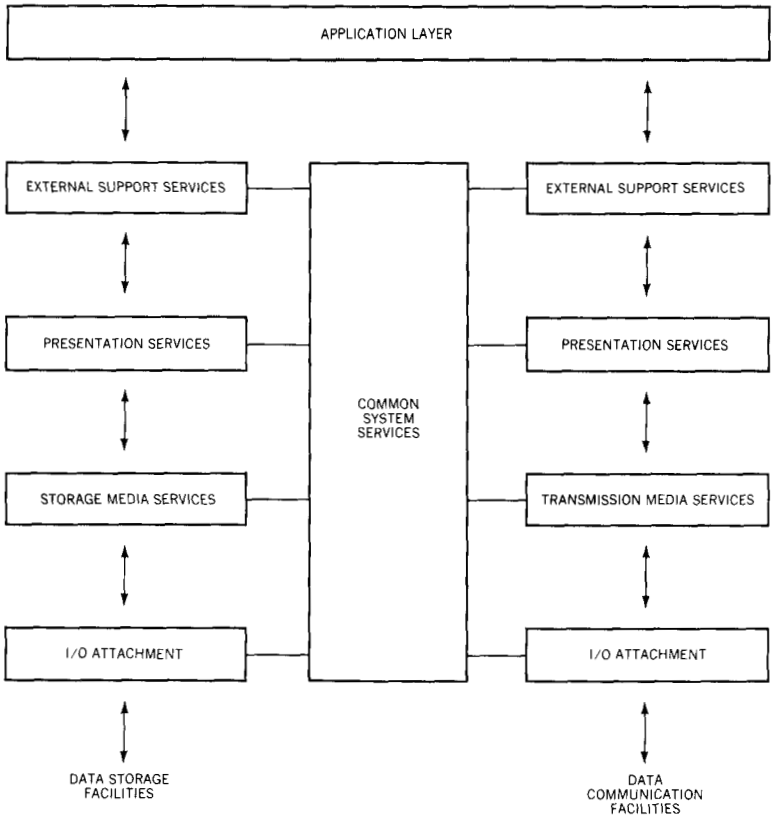


Figure 12 Data storage and communication facilities



I/O service facilities

The following paragraphs describe the services and structures that support the I/O facilities introduced earlier. Unique features of each facility are identified, and shared services and mechanisms are discussed.

Figure 12 shows two major types of I/O facilities and the services and resources that support them. For simplicity in the early part of this discussion, data communication facilities are assumed to include both data presentation and data exchange, each of which is discussed in more detail later. The major distinction between data storage and data communication facilities is that their service and hardware architectures are different. For example, the data storage facility is based on a fixed-block storage architecture and a DPPX logical-block architecture for storing and retrieving data. The data communication facilities, on the other hand, are based on SNA,[7] which encompasses physical data transmission protocols (synchronous data link control, or SDLC), as well as logical end-to-end transmission protocols and network resource management.

Although the architectures are quite different, several internal DPPX services are shared among the storage and communication facilities. As shown in Figure 12, all the service layers share a common set of system services, including buffer management, program management, I/O process management, and interlayer linkage services which are optimized for high-performance I/O functions. The development of common system services has eliminated the need for redundant implementations within any of the layers.

device-independent interface

A common requirement in the design of I/O facilities is to have a simple, high-level interface for use by applications. An additional goal is for applications using that interface to operate with a wide variety of devices. In DPPX these requirements are satisfied by a combination of External Support Services (ESS) and Presentation Services (PS) for each of the I/O facilities.

ESS provides a common I/O interface mechanism (including macros and control blocks) for all I/O facilities. Each PS implementation provides the functions required to interpret the device-independent requests for each facility. As an example, this capability allows a user to write an application that normally uses a pair of data sets (one input and one output) and occasionally uses a terminal, instead, simply by changing resource assignment commands that are external to the application program.[11] In addition, this capability allows system-provided applications (such as the COPY command processor) to easily support many different devices.

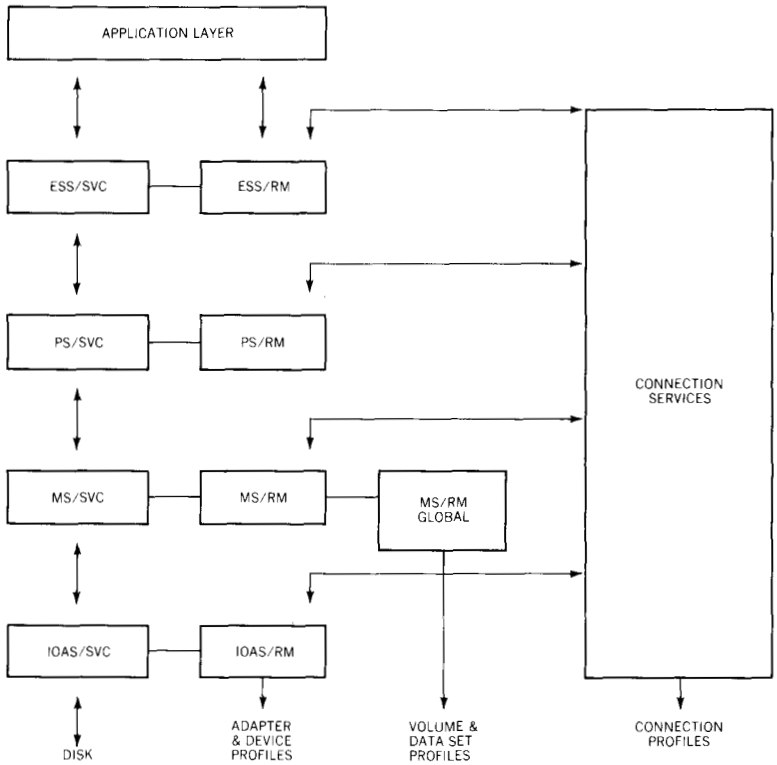
data storage facility

Figure 13 illustrates the structure of the data storage facility. The service portion of each layer is at the left, and the Resource Manager (RM) portion is at the right. Each of the Resource Managers interfaces with Connection Services, one of the common system services.

Connection Services controls the setting up and taking down of support (modules and control blocks) within a layer and the establishing of interfaces between layers. The specific layers and the sequence of Resource Manager invocation are controlled by connection profiles, maintained in a profile data set. As an example, the interface between MS (Media Services) and IOAS (I/O Attachment Services) is set up when a storage device is activated, and the ESS-to-PS and PS-to-MS interfaces are set up when an application requests a connection to a data set. Two different connection profiles support these connections, one supporting the device activation, and the other the data set connection.

Whenever a device is activated, the IOAS Resource Manager is invoked (by means of the appropriate connection profile). The IOAS Resource Manager accesses the device profile for the named device and sets up the service portion of the IOAS layer. This

Figure 13 Structure of the data storage facility

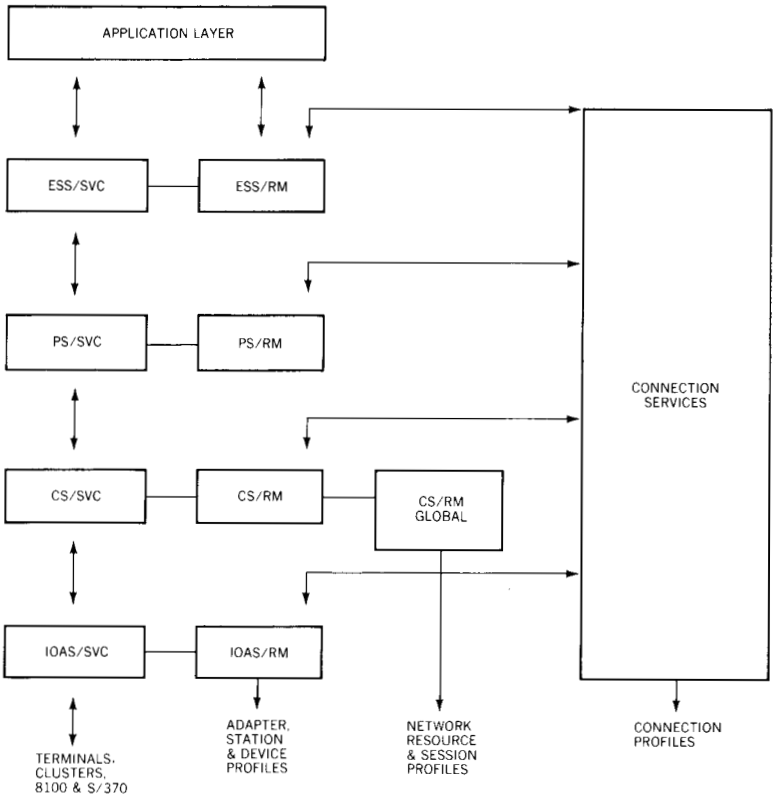


procedure includes loading the appropriate modules as well as building and initializing the required control blocks as determined by the profile for the particular device. Once IOAS has been set up, the MS Resource Manager is invoked to set up the volume management support. IOAS Resource Manager support is common to the data storage and data communication facilities. In addition to the savings associated with this commonality, this design approach separates the physical I/O resource profiles from the logical resource profiles, discussed next.

Whenever a data set connection is requested, the MS Resource Manager is invoked under the control of the appropriate connection profile. The MS Resource Manager causes the global data storage Resource Manager to access the profile for the named data set, then causes service support to be set up, based on the profile. The PS Resource Manager is then invoked; it sets up Presentation Services based on the same data set profile. Finally ESS is set up, and the interlayer interfaces between the application and the user data set are complete.

Connection Services can set up a variable number of layers as determined by connection profiles, not just the specific layer con-

Figure 14 Structure of the data communication facilities



figuration shown above. Thus the system is readily extendible to support other configurations, including new or replacement layers, by adding or replacing connection profiles. The global data storage Resource Manager is the catalog service of DPPX. This service is unique to the data storage facility. Several data set access capabilities are provided by different implementations of the PS layer. These capabilities include sequential access, access by relative record number, and keyed access either with or without recovery. For further information, see the accompanying paper by Fitzgerald and Goodrich.[12]

data communication facilities

Figure 14 illustrates the structure of the data communication facilities, which support both data presentation and data exchange between applications. Some differences between the two are discussed at the end of this section.

It should be noted that the structure is similar to that described previously for data storage. This similarity is an aid to understanding and maintaining the system. Media Services (MS) is replaced here by Communication Services (CS). The physical I/O resource is a terminal or cluster of terminals, or another processor, rather than a data set on a storage device.

Much of the previous discussion of Resource Managers applies here as well. However, there are a few important differences. There is an additional flow of data between the ESS and CS Resource Managers. This interface supports requests for session initiation and termination. Sessions can be initiated by the application layer or by a user (a terminal user or another application). When the application requests a connection to another resource (known as a logical unit in SNA), the request for connection is passed to the CS Resource Manager, which in turn sends a session initiation request to the global communications Resource Manager (known as the System Services Control Point in SNA). The global Resource Manager determines whether the resource (logical unit) is available. If the session can be initiated, a session profile appropriate to the particular application and resource is retrieved and passed back to the CS and ESS Resource Managers. Connection Services is then invoked to cause layers that support data communication to be set up as described above for connection to a data set.

The key difference is that the global Resource Manager is accessed before the layers are set up, rather than after. This difference has a significant advantage. It allows information from the session profile to be used in selecting the connection profile (for the appropriate Presentation Services). For data sets, the corresponding information comes from the resource assignment statement associated with the application.

The CS Resource Manager is made up of two components. The first is *logical unit services*, which support session initiation and termination, described above. The second is *physical unit services*, which support the activation and deactivation of physical paths. When a device is to be activated, the global Resource Manager invokes the CS Resource Manager, which in turn invokes the IOAS Resource Manager by means of Connection Services. The IOAS Resource Manager accesses the appropriate station or device profile and sets up IOAS accordingly. When IOAS is set up, the CS Resource Manager sets up the path control support within CS. This support varies somewhat, depending on the type of SNA physical unit.

As indicated in the above discussion, many Resource Manager functions are common to data storage and communication facilities. These common functions include the IOAS Resource Manager, Connection Services, and much of the ESS Resource Manager. In addition, most ESS service functions are common to both types of facilities.

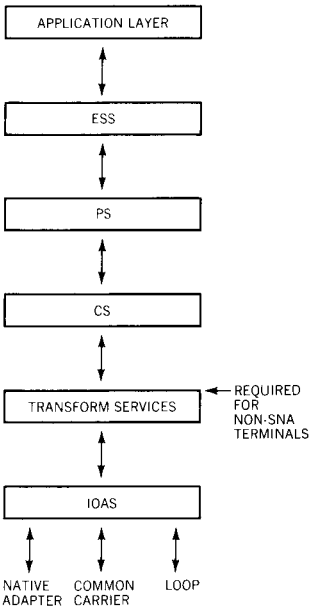
The remainder of this section focuses on the differences between data presentation and data exchange, primarily in terms of their service functions.

data presentation facility

Figure 15 illustrates the structure of the data presentation facility, including support for non-SNA terminals. Most devices, other than data storage devices, are managed as SNA resources and are generically referred to here as *terminals*. The Transform Services (XS) layer is required for those terminals that do not fully implement an SNA protocol. Included are some terminals that attach directly to an adapter, for which IOAS provides the support that is normally provided by the device controller. Also included are some terminals that support binary synchronous or start/stop communication-link protocols. In addition, some SDLC terminals are included that do not support standard SNA protocols. By providing transformations that mask the protocol and attachment differences from the higher-level service layers, it is possible for all of these terminals to access many of the presentation management and application management functions that are available to SNA terminals. This approach minimizes the cost of developing special support for non-SNA terminals.

Those terminals that require a Transform Service are identified as such in the device profile, and the name of the Transform Service also is provided there. This mechanism allows for the easy addition of new Transform Services without changing the structure of the I/O facility.

Figure 15 Data presentation facility that supports non-SNA terminals



Two major Presentation Services are provided for terminals. The first supports the device-independent logical record interface described earlier. It includes support for keyboard-display terminals and keyboard-printer terminals as well as card read/punch and printer devices.

The other Presentation Service supports a field-formatted record interface for keyboard-display and printer terminals. For the display, it provides full-screen formatting of application data under the control of tables external to the application program. For the printer, it formats application record fields into page format. The data layout definition is provided by a programmer in an interactive process that is separate from the application coding process. In addition, formatting for the device is performed while the application is executing, but outside the application program. This technique allows one application program to access different devices using different tables and also allows the format of the data to be changed without changing the application code (for example to support a new terminal).

To provide these features, the Presentation Services (PS) layer maps fields from an application record to appropriate locations within the display screen or printer page. Constant values, such as heading and keyword values, are added under map control. Conversely, on input from the display screen, data fields are

mapped to the desired application record format. The record is then passed to the application when a RECEIVE operation is performed.

data exchange facility

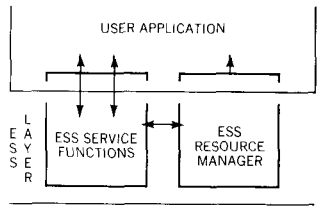
For data exchange between systems, two different Presentation Services are provided. One supports communication with CICS or IMS (the Information Management System)[13] in a System/370. It performs certain SNA recovery functions required by these data base/data communication systems. The other Presentation Service is a simple variation on the Presentation Service described earlier for terminals. It supports the device-independent interface, described earlier, between applications. Both of these services use a common CS layer interface. Thus they are independent of the particular variation of SNA that is used for communication with the other application, as illustrated in Figure 15.

For data exchange configurations, both the CS and IOAS services are the same as the data presentation configurations that support terminals. This commonality has advantages both in minimizing the cost of initial development and in potentially improving the configurations supported.

application attachment

External Support Services (ESS) provide a single implementation for attaching DPPX applications to the I/O facilities. As with past access-method support such as VSAM and VTAM,[14,15] the ESS definition makes no assumption about application structure. It supports application interface functions that have been proved appropriate by these past implementations and defined in higher-level languages such as FORTRAN and COBOL. The effect is that ESS provides for an interface between the DPPX environment and the I/O facilities. The application running in the environment is then responsible for the relation of the I/O connections and other environment resources such as programs, processes, and storage.

Figure 16 ESS relation to user application



One view of ESS is that it adapts an unstructured application program to the common DPPX service and Resource Manager structure. This view is used in the following discussion of ESS functions. Figure 16 illustrates the design concept. The resource managed by ESS is the connection (or connections) between an environment and the I/O facilities. The service supplied for this resource is connection request/response handling.

The functions supplied by the ESS Resource Manager are receiving and controlling application requests for connection to and disconnection from I/O facilities (similar to OPEN and CLOSE), and receiving, interfacing, and controlling requests for connection to and disconnection from the application. In the latter case, when requests are for connection, the application can be viewed as a service, and the request can be viewed as a request for connection to the service (for example, LOGON). And when the

request is for disconnection, the application can be operating as either a user or a service, and disconnection is initiated by an event outside the application (for example, "lost terminal" or LOGOFF).

Queued interfaces are established by ESS for connection functions that require asynchronous (random in time) notification. Such functions include LOGON, LOGOFF, and "lost terminal."

The ESS service functions support the handling of requests and responses for all I/O services. In summary, the functions supplied are:

- Mapping the application-program-defined requests to and responses from an internal request/response form called the I/O block and service command list. This request/response form is common to all DPPX I/O Services.
- Synchronous request/response operation by the application. (*Synchronous* means that the appplication process waits until the request is complete to continue processing.)
- Asynchronous request/response operation by the application. (*Asynchronous* means that the application issuing the I/O request continues to execute while the request is being processed, and the application is later notified that the request is complete.)
- Exit and error status information functions relating to the request/response flow.

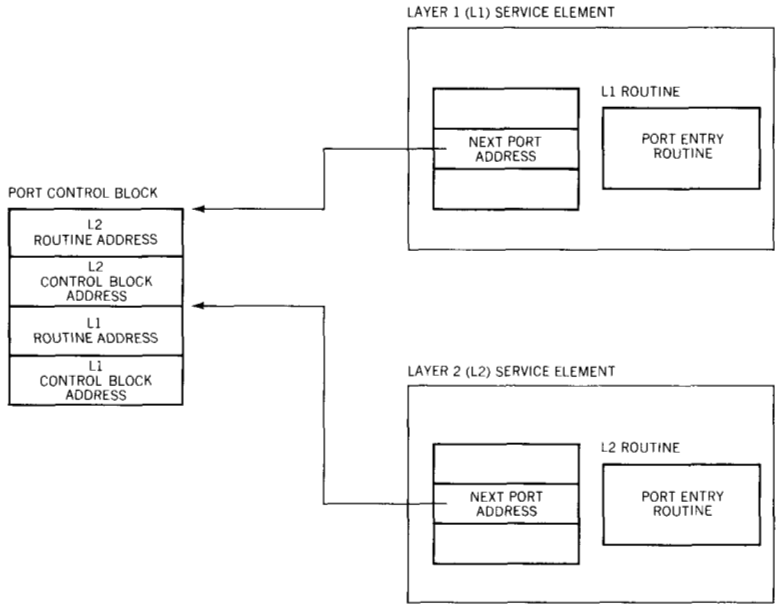
ESS provides a queued interface to the application for I/O service functions that require random entry into the application environment. Such functions are related to *attention* and *unsolicited data* conditions initiated by the terminal operator.

ESS performs validity checking to insure that all requests issued by an application are to services and resources that are validly connected to the application environment. This validity checking assists in maintaining the integrity of the system.

I/O service architecture

Both the key concepts of the layered I/O structure and the specific structures that support the major I/O facilities of DPPX have been described. Those concepts and structures may now be viewed as a set of requirements for the system design. The successful implementation of those requirements depended on establishing a common framework in two areas. The first was defining the rules and supporting mechanisms for interlayer communication, and the second was defining the commonly required system and Resource Manager services so as to facilitate the implementation of func-

Figure 17 Port linkage mechanism



tions within a layer. These definitions constitute the DPPX I/O architecture, some key elements of which are described next.

One of the important concepts of the structure, described earlier, was designing a single layer to provide external support services for all the I/O facilities, including the multiple levels of interface within the facilities. The problem was to provide a common set of interface rules and supporting implementation mechanisms, so that ESS could communicate with various layers in a consistent way. The set of rules and mechanisms is referred to as the *port interface*.

port interface

The port interface is defined in three parts. First is the interlayer linkage mechanism, also referred to as the port control block interface. Second is the set of rules, or protocols, called port control functions, that synchronize interlayer connections. Third is the set of rules, called the request/response mechanism, for interlayer data flow.

The port linkage mechanism consists of a port control block and a convention for passing control from the port exit routine of one layer to the port entry routine of another. The port control block supports bidirectional requests and responses between connected layers, as illustrated in Figure 17. It anchors the address of each layer's entry routine. It also anchors two layer-dependent parameters, which are used (by the prospective layer upon entry) to

identify the specific service or resource control block. Both the Layer 1 and Layer 2 control blocks point to the port control block.

These relationships are set up at connect time. This structure provides for high-performance interlayer linkage while making the invoker independent of the identity of the invoked layer. Thus either layer can be replaced or omitted in any particular connection without affecting the other.

The normal mechanism for passing control information from one layer to another is a macro instruction, PCBGO, which results in either a branch instruction or a supervisor-assisted transfer of control. A supervisor-assisted linkage is required if the port entry routine is a transient module, or if the invoking layer's port exit routine occupies a system transient area. In either case, the means of invocation is transparent to both layers. This mechanism permits layers to be configured either as transient modules or as resident modules, depending on storage and performance considerations.

The port synchronization functions and the request/response functions use a common I/O block. When one layer invokes another, the I/O block address is passed, along with the address of the port control block.

port synchronization functions

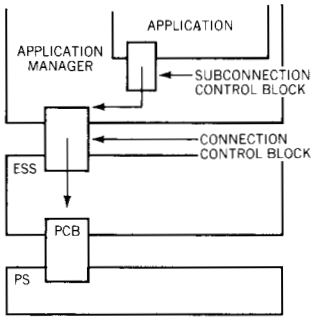
The port synchronization functions control the status of a connection between layers. Three major functions are supported by all layers, as described below:

- START—issued by the higher-level service to start or restart normal request/response handling.
- STOP—issued by the higher-level service to terminate normal request/response handling. Outstanding requests must be forced to completion. STOP does not imply DISCONNECT. It may be followed by a START request.
- INOPERATIVE—issued by the higher-level service to initiate abnormal termination of the request/response flow. Usually the processing is the same as for STOP, except that an error code is implied.

In addition to the functions above, some layers also must support the receipt of inoperative requests from layers below. For example, communication layers may have to deal with externally generated inoperative conditions that arise as the result of a communication-link failure or an application failure in another node.

The ESS and PS layers of the communication facilities support all of the above functions, plus additional functions that support sub-

Figure 18 Application Management subconnections



connections. Subconnections allow an Application Management layer to share a connection with a series of application programs, one at a time. The Application Manager establishes a connection with a terminal (or another application), and when the terminal user invokes a particular transaction or command processor (application), the application may borrow the terminal connection for the duration of the transaction, then return it to the Application Manager. This relationship is shown in Figure 18. It provides a high-performance mechanism for borrowing a connection while presenting a standard connection interface to the application, thus minimizing the special support that is common in earlier systems.

The subconnection capability requires additional synchronization functions, as described below:

- CHANGE CONNECTION—causes control of a connection to be switched from the Application Manager environment to an application subenvironment, or the converse.
- START SUBCONNECTION—causes control of a connection to be switched to an application subconnection within the same environment.
- STOP SUBCONNECTION—terminates a subconnection and returns control to the Application Manager connection or to a previous subconnection. (Multiple subconnections are permitted within the same environment.)

common request/response mechanism

When a connection (or subconnection) is in an active (START) state, the series of layers that support the connection are prepared to handle requests for service from the user. A service request typically involves processing in several layers. Also, a given service request may be supported by several different facilities made up of different combinations of layers. Both of these factors dictate a common interaction structure for processing service requests and their associated responses.

The major information structure for request/response handling is the I/O block, which carries the common information required to process any request. It also contains a work area for each layer that may be involved in processing the request. This approach may result in unused work-area space for some types of connections and requests, but it facilitates the correlation of requests and responses within each layer, and it eliminates the performance overhead of obtaining work areas dynamically within each layer. Generally, an I/O block is obtained from a buffer pool by the layer that originates the request (for example, ESS or IOAS) and is returned to the free pool by the layer that completes the response processing. Thus the I/O block is required only for the duration of a request/response processing cycle.

The service-dependent parameters and the user data associated with a request are carried by a subsidiary structure, the service command list, which is made up of one or more service commands and associated data.

Summary

The DPPX I/O facility has been designed and implemented to provide maximal functional capability at minimal cost and to provide a structure in which functions designed to satisfy future requirements can be implemented easily.

The general structure of the I/O facilities has been discussed from two points of view. First is the functional concept of data storage, data presentation, and data exchange facilities. Second is the concept of layers and their functional role in the system. The general structure of each layer has been explained, along with the service and Resource Manager functions, and the common interface architecture of the layer structure has been described.

It is difficult, of course, to predict the nature of future requirements for I/O functions; however, it is generally agreed that the requirements will change over a period of years. Two factors lead to the conclusion that the structure described here is a good base upon which to construct software that will satisfy future requirements. First is the observation that during the development of DPPX, the requirements did change or were extended, as is usual when large software systems are being developed. With each change, the structure proved to be a suitable framework for implementing new functions. The second observation is that the structure is soundly based on current I/O concepts and experience.

Future work is needed to study the conceptual (perhaps theoretical) aspects of such structures and to provide a better understanding of the scope of their applicability and to enhance the generality of their architecture.

ACKNOWLEDGMENTS

The general structure of the DDPX I/O facility was developed during 1976 and 1977 by a team of system designers, whose contributions we acknowledge with gratitude. Especially significant were the contributions of Dr. Ed Harrison, Brian Goodrich, Fred Banks, and Hal Opdyke. We give special recognition to the encouragement and stimulating ideas of Steve Kiely.

CITED REFERENCES

1. *An Introduction to the IBM 8100 Information System*, IBM Systems Library, order number GA27-2875, available through IBM branch offices.
2. A. L. Scherr, "Distributed data processing," *IBM Systems Journal* **17,** No. 4, 324-343 (1978).
3. *Distributed Processing Programming Executive Base Diagnosis: Logic*, IBM Systems Library, order number GA27-2875, available through IBM branch offices.
4. S. C. Kiely, "An operating system for distributed processing—DPPX," *IBM Systems Journal* **18**, No. 4, 507-525 (1979, this issue).
5. *Distributed Processing Development System, Programming Language for Distributed Systems Reference*, IBM Systems Library, order number SC27-0446, available through IBM branch offices.
6. *CICS/VS General Information*, IBM Systems Library, order number GC33-0066, available through IBM branch offices.
7. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15,** No. 1, 4-23 (1976).
8. E. W. Dijkstra, "The structure of the 'THE'-multiprogramming system," *Communications of the ACM* **11,** No. 5, 341-346 (1968).
9. P. G. Cullum, "The transmission subsystem in Systems Network Architecture," *IBM Systems Journal* **15,** No. 1, 24-38 (1976).
10. H. R. Albrecht and K. D. Ryder, "The Virtual Telecommunications Access Method: A Systems Network Architecture perspective," *IBM Systems Journal* **15,** No. 1, 53-80 (1976).
11. *Distributed Processing Programming Executive Base Commands: General Use*, IBM Systems Library, order number SC27-0404, available through IBM branch offices.
12. A. K. Fitzgerald and B. F. Goodrich, "Data Management for the Distributed Processing Programming Executive (DPPX)," *IBM Systems Journal* **18,** No. 4, 547-564 (1979, this issue).
13. *IMS/VS General Information Manual*, IBM Systems Library, order number GH20-1260, available through IBM branch offices.
14. *Planning for Enhanced VSAM Under OS/VS*, IBM Systems Library, order number GC26-3842, available through IBM branch offices.
15. *Introduction to VTAM*, IBM Systems Library, order number GC27-6987, available through IBM branch offices.

GENERAL REFERENCES

D. E. Freeman and O. R. Perry, *I/O Design: Data Management in Operating Systems*, Hayden Book Company Inc., Englewood Cliffs, New Jersey (1977).

R. J. Cypser, *Communications Architecture for Distributed Systems*, Addison-Wesley Publishing Co., Reading, Massachusetts (1978).

The authors are located at the IBM System Communications Division laboratory, Neighborhood Road, Kingston, NY 12401.

Reprint Order No. G321-5108.