The Distributed Processing Programming Executive (DPPX) operating system has network configuration requirements placed on it. This paper discusses those requirements and the way in which they are met, including those that result from the various configurations possible with a DPPX system. In addition, the unique way in which terminal resources are supported in DPPX and the dynamic approach to resource definition in the DPPX system are described. Finally, application definition and application usage of the network configuration capabilities of DPPX are discussed.

Distributed processing communications software support for operation within an SNA network

by E. S. Harrison

The Distributed Processing Programming Executive (DPPX), an operating system for the IBM 8100 Information System,[1] is based on SNA (Systems Network Architecture)[2] and has been developed as IBM's distributed processing system.[3] As such, it has to meet various network configuration requirements. One requirement is a stand-alone configuration in which the DPPX system supports applications and terminals in a single 8100 Information System. This configuration allows useful work to be carried out at a remote geographic site, essentially independent of a System/370-like host system.

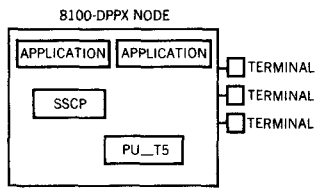
DPPX must also attach to existing host system networks to permit DPPX users to be able to communicate with existing host application programs, and by this means, take advantage of already developed host application programs. Periods of attachment to a host may then be based on whether the host data base needs to be updated to reflect the processing carried out at the remote 8100 location. This intermittent attachment may be at the end of a working day when the host data base is updated with the results of transactions executed at the remote site. The stand-alone capability of DPPX and the general way in which all terminals are supported by a common structure are described in the first section of this paper. The host attachment capability of DPPX is described in a later section.

Interconnection of two or more 8100 Information Systems allows a DPPX user greater network configuration capability within a network made up of 8100 systems. It will, for instance, be possible for the user to distribute DPPX systems to different geographic locations while maintaining overall network control of the distributed systems at one specific DPPX node. DPPX applications will then be able to communicate between any two adjacent 8100 systems, thus allowing the user a distributed processing capability within a network consisting of DPPX nodes. This ability of DPPX to support the interconnection of two or more 8100 Information Systems is described in this paper.

An important function in any communications system is the way in which a system operator defines network resources to the system. A movement towards a more dynamic approach to network definition has been taken by the DPPX system and is discussed in a later section.

Finally, the way in which DPPX applications can access the network communications functions of a DPPX system and the way in which DPPX has expanded the concept of dynamic definition to DPPX application programs are described.

Figure 1 8100-DPPX system operating in stand-alone mode



Stand-alone capability

To be able to have one DPPX node supporting terminals and application programs independently of a host system requires DPPX to have its own System Services Control Point (SSCP), which is the configuration manager of resources in an SNA domain. Residing with an SSCP in the same node is a physical unit called a *physical unit type 5* (PU_T5). The stand-alone configuration is illustrated in Figure 1. In this configuration, DPPX application programs can communicate with terminals attached to the DPPX system and with other DPPX applications running in the same stand-alone DPPX system.

terminal support

DPPX supports many different kinds of terminals attached to the system by many different physical attachments. Table 1 lists some of the terminals supported.

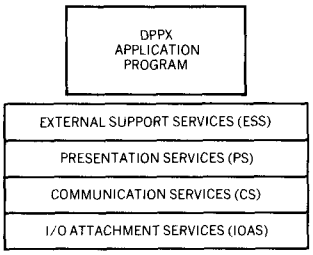
The support for SNA-type terminals is well-defined, and there are SNA-defined protocols for activating and deactivating terminals and for initiating and terminating connections to application programs; these are well-documented in SNA. However, many different protocols exist for non-SNA terminals. These different protocols add complexity in the support of non-SNA terminals.

One method of supporting non-SNA terminals is to have a non-SNA equivalent of the SSCP such that the terminals are supported

Table 1 Some terminals supported by DPPX

Terminal type	*Example*
SNA	IBM 3276 display IBM 8775 display IBM 3767 keyboard printer IBM 3287 printer
BSC (Binary Synchronous Communication)	IBM 2780/3780-like terminals
Locally attached	IBM 3277 display
SDLC attached non-SNA	IBM 364X plant communication family IBM 3289 line printer
Start/Stop	IBM 2741 communication terminal Teletype 33/35 (TTY)

Figure 2 Typical DPPX I/O layer structure



as two sets: SNA and non-SNA. This approach is feasible but has the following disadvantages:

- Tends to duplicate function.
- Presents a dual interface for the system operator: two interfaces exist—one for SNA terminals and one for non-SNA terminals.
- Complexities arise when SNA and non-SNA terminals are attached to the same link; for example, determining which manager controls the link.
- Presents a dual interface to the application program.

Obviously, from network usability considerations it is desirable to have all terminals supported with the same external network interface. The procedures needed to define all terminals to the DPPX system and the network procedures needed to activate and deactivate these terminals should be the same.

This result has been achieved within DPPX by extending the I/O layer design of DPPX. All terminals are treated in DPPX as if they are SNA terminals. In order to more fully understand how this has been achieved, it will be necessary to briefly explain the I/O layer concept of DPPX. A fuller description of the internal I/O structure of DPPX is contained in Reference 4.

I/O layer structure

Layers of software function exist in DPPX, referred to as I/O layers, which correspond very closely to the functional layers defined in SNA. A connection to an SNA terminal from a DPPX application program will typically have an internal I/O layer structure as shown in Figure 2.

The layers are responsible for the following functions:

- ESS (External Support Services) essentially is the application program interface where data and control blocks entered by the application as part of a request are changed into control block formats required by the internals of DPPX.
- PS (Presentation Services) is an I/O service layer which may or may not be present in the I/O structure for a connection. It provides function to format the user data ready for presentation to the session partner and includes the Presentation Services and Data Flow Control functions of SNA.
- CS (Communication Services) is the layer that provides the SNA transmission control and path control support. It is responsible for the end-to-end routing of data between session partners.
- IOAS (I/O Attachment Services) is the layer that contains the I/O support for interfacing with the I/O hardware and, in SNA terms, contains the Data Link Control (DLC) functions. Note that there are many different types of IOAS support, depending on the type of terminal supported. For SNA terminals, the IOAS layer is called the DLC IOAS.

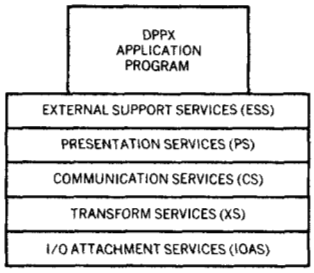
The I/O interface between CS and IOAS for terminal resources consists of an SNA transmission header whose format depends on the physical unit type of the SNA terminal. For other IOAS layers (non-DLC IOAS layers), the interface is different and based usually on the data interface to the particular terminal.

An additional I/O layer (called the Transform Layer) inserted between CS and IOAS for non-SNA terminals can match the outbound and inbound interfaces between the CS and IOAS layers and do whatever transformation of the data is necessary to maintain the interfaces. By this means, the CS layer and all other SNA functions above the CS layer will view the non-SNA terminal as an SNA terminal. Any function designed to work for the SNA terminals will then be equally valid for the non-SNA terminals supported by the I/O Transform Layer.

SNA request units flow in the process of activating and deactivating SNA terminals and in connecting and disconnecting with application programs. In order for a non-SNA terminal to maintain the appearance to the CS layer of an SNA terminal, these request units are intercepted and responded to by the I/O Transform Layer.

One way of describing the function of the I/O Transform Layer is to recognize that SNA terminals contain intelligence that allows them to respond to SNA protocols in an SNA-defined manner. Non-SNA terminals, however, do not contain this intelligence, and the Transform Layer can be viewed as a way of providing it on behalf of the terminal. The DPPX I/O layer structure for a non-

Figure 3 DPPX transform services I/O layer

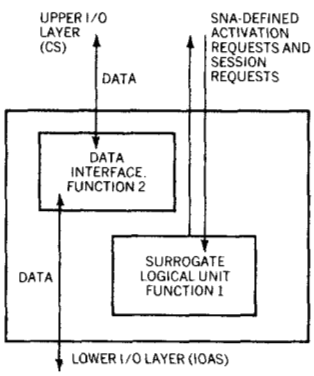


SNA terminal in Figure 3 shows the positioning of the Transform Layer in the I/O layer structure.

The Transform Layer has two basic functions:

- One to supply the SNA intelligence on behalf of the non-SNA terminal by responding appropriately to SNA requests not supported by the non-SNA terminal. The component that performs this processing is often referred to as a Surrogate Logical Unit and provides the logical unit services function on behalf of the non-SNA terminal.
- Another to ensure that the CS interface and the non-DLC IOAS interface are maintained during the request flow to and from the terminal. This function may be referred to as a Data Interface component.

Figure 4 Transform Layer structure



These functions of the Transform Layer are depicted in Figure 4.

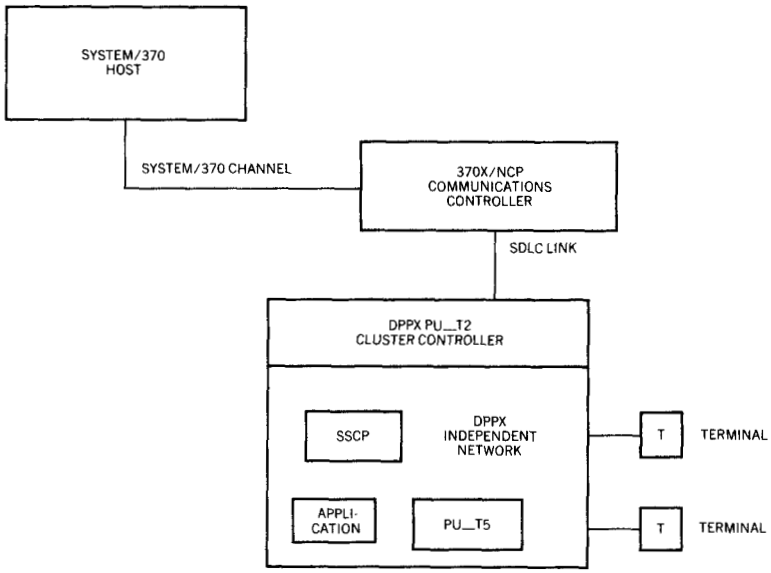
With the introduction of the Transform Layer concept into the DPPX network communication function, all terminals, whether SNA or non-SNA, are supported in exactly the same manner. Thus, the system operator need only understand one means of defining terminal resources to the system and one means by which these terminals can be activated and deactivated. DPPX applications also see no difference in interface when communicating with SNA and non-SNA terminals, and hence their application logic need not be concerned with the different protocols that actually exist to the different terminals. Transform Layers can also be user supplied so that a terminal not supported by DPPX can be supported within the DPPX system.

In DPPX development activity, new function, which may be added in the future, can essentially be used for both SNA and non-SNA terminals, since the structural support for both types of terminals is the same. This common support will manifest itself in lower design and development costs for functions added to DPPX and the 8100 Information System.

System/370-like host attachment capability

DPPX can attach to a host system, and DPPX application programs can communicate with already existing host application subsystems such as CICS (Customer Information Control System) and IMS (Information Management System). DPPX attaches to a System/370-like system as a cluster controller: in SNA terms, this means attaching as a physical unit type 2 (PU_T2).[5] The 8100 node attaches to an adjacent IBM 370X/NCP (Network Control Program) communications controller that is either locally or remotely attached to the host system. The communications controller shown in Figure 5 is locally attached to the host system.

Figure 5 8100-DPPX attached to host system (8100-DPPX as a cluster controller to host System/370 while also containing independent network management functions within same 8100-DPPX node to support attached terminals)

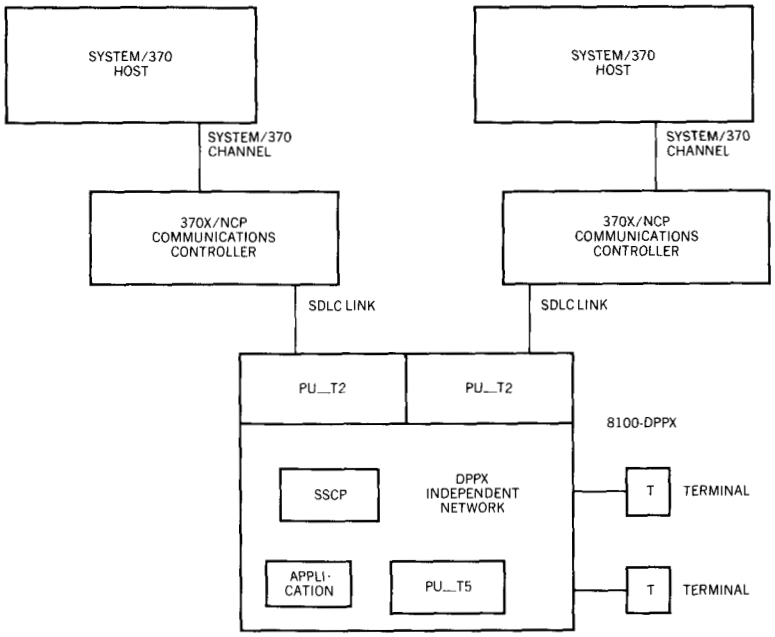


With the 8100 and DPPX acting as a cluster controller in a host system network, host applications can communicate with DPPX terminals by a pass-through application program that provides the function of passing the data from the host application through to the terminal and from the terminal back to the application program. This capability is discussed more fully later in this paper. Figure 5 shows the 8100-DPPX system containing one physical unit type 2. It is possible for the 8100 node to support more than one host attachment capability, however. For example, two or more cluster physical units can be contained within one 8100 node; these can be attached to the same or to different host systems and are attached by separate links to adjacent communications controllers. This capability is generally referred to as the multihost attachment capability within DPPX. Figure 6 is a typical example and shows two physical units in one 8100-DPPX node attached via different links to two communications controllers belonging to different host systems. This capability allows the user considerable flexibility in being able to communicate directly, from the same 8100 node, with application subsystems running in different host systems.

Interconnected capability

One DPPX system can interconnect to another DPPX system. The physical interconnection is by means of a link (SDLC, IBM's net-

Figure 6 Multihost attach configuration



work protocol, synchronous data link control) and may be point-to-point or multipoint.

There were two principal architectural and design alternatives for interconnecting 8100 nodes:

1. As a cluster controller which attaches in much the same way as the 8100 attaches to a host system (PU_T2)
2. As a communications controller (PU_T4)

The second alternative was taken since in the long term this will give a better base on which to build a distributed DPPX network. The communications controller is called a PU_T4 node in SNA terms and is identical to a PU_T5 except that it does not exist in the same physical node as its SSCP. However, the protocols from the SSCP are exactly the same as those for the PU_T5. Hence, it is possible to use the same physical unit design to implement a communications controller (PU_T4) as that used in the PU_T5 (since the PU_T4 is simply a distributed form of the PU_T5). The same control code is used in DPPX for both the PU_T4 and PU_T5 physical units.

Terminals physically distributed from the SSCP node are supported by the PU_T4 DPPX node so that applications in one DPPX node can communicate with applications or terminals located in an adjacent node. Note that these distributed terminals can be both SNA and non-SNA terminals (supported by a Transform

Figure 7 *Simple interconnected 8100-DPPX systems*

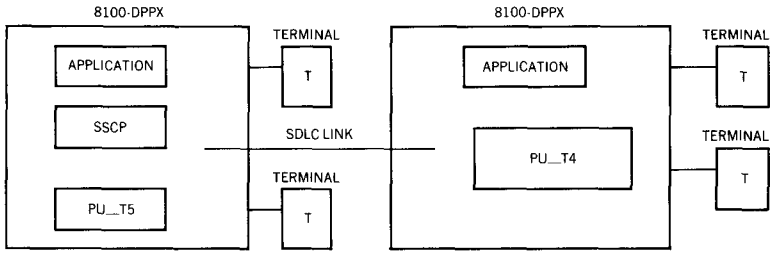
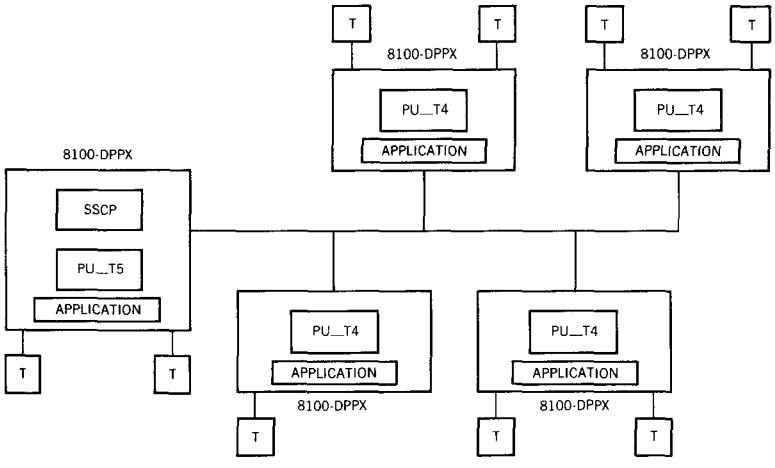


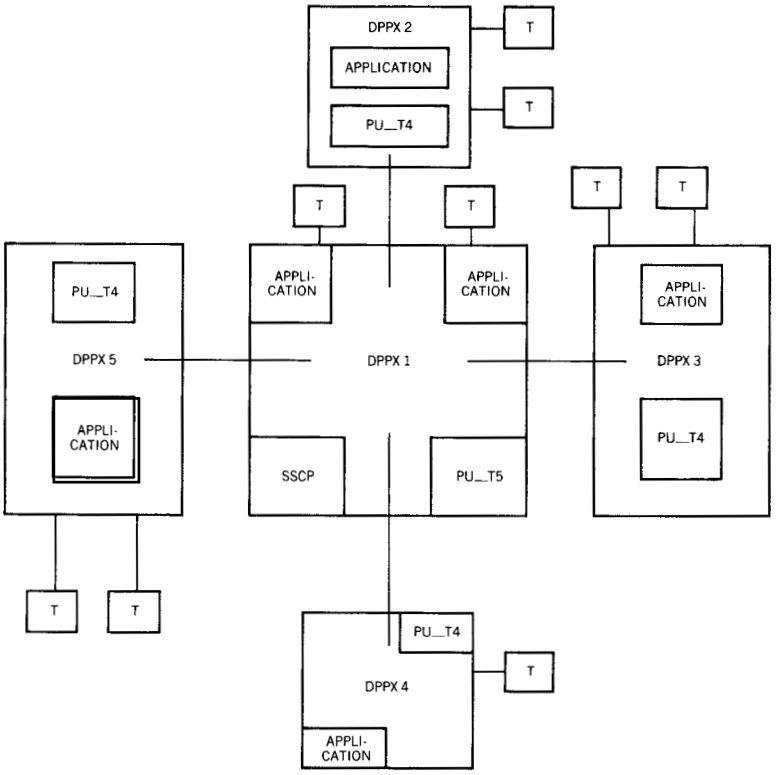
Figure 8 Interconnected multidropped 8100-DPPX systems



Layer). Applications in the PU_T4 DPPX node can also communicate directly with terminals attached to the same PU_T4 node. In this case, both the terminal and the application program are completely distributed from the SSCP. This support, distributed from the SSCP, allows great flexibility in configuring a DPPX network. The interconnected DPPX PU_T4 and PU_T5 nodes are shown in Figure 7, where the configuration allows any application shown to communicate with any terminal shown and any application to communicate with any other application.

Figure 8 shows the capability of having many interconnected 8100 systems off the same link in a multidrop configuration. This configuration provides for high link utilization and low link costs. The PU_T5 node provides primary link-station support, and the PU_T4 nodes are defined as secondary link stations. Applications in the PU_T5 node can communicate with applications or terminals in any of the PU_T4 nodes. Similarly, applications in any of the PU_T4 nodes can communicate with applications or terminals in the PU_T5 node.

Figure 9 DPPX star network



One DPPX system can interconnect with a number of adjacent DPPX systems to form a star-like DPPX network where the central system supports adjacent DPPX systems and acts as the configuration manager. This star-like DPPX network is depicted in Figure 9, where DPPX 1 is the configuration manager. Applications existing in DPPX 1 can connect directly with applications or terminals in any of the other DPPX systems since DPPX 2, DPPX 3, DPPX 4, and DPPX 5 are all interconnected to DPPX 1.

With the utilization of this interconnection capability, it is possible to build up an independent network of DPPX systems and still maintain a host connection capability by having one or more DPPX systems include a cluster controller attachment capability to the host system. The user can then distribute 8100 systems to different geographic locations and maintain control of the network at one specific DPPX location. For instance, DPPX 1 in Figure 9 can have a host attachment capability, and once all the information has been retrieved from the adjacent DPPX nodes and its central DPPX data base has been updated, data can then be transmitted to the host system for a final update of the host data base.

System definition capability

In existing IBM systems the system definition capability by which resources are defined to the system has usually been in the form of a system generation procedure. With this procedure a list of macros are executed to produce a system generation deck which is then read by the system. This deck defines the network configuration to the system and is typically accessed by the configuration manager (SSCP) when the network configuration is being activated.

One of the problems inherent in this approach is that it becomes very difficult, if not impossible, to change the network configuration of the system without requiring a new system definition for every node in the network. Changing the network configuration results in the existing configuration being shut down such that no useful work can be carried out by the system during that time. In DPPX, the network configuration definition has been made dynamic such that all network resources are defined by DPPX commands. The definition required for the SSCP and physical units in a DPPX system is discussed below.

implementation and definition of SSCP and physical units

It should be clear at this point that it is possible in certain user configurations to have many different physical units present in one DPPX system. For example, in Figure 6, there exist three physical units, and more can exist depending on whether the same DPPX system needs to interconnect to other DPPX systems. It is apparent that a variable number of physical units can exist in one DPPX system, and it is also desirable to be able to dynamically activate and deactivate these physical units. For example, a DPPX system may not require a host attachment capability (PU_T2) until late in the working day and may only require the capability to be operational for the time it takes to update the host data base. Implementing physical units as DPPX control applications[6] seems to meet all these requirements and has the followng advantages:

1. The physical units, as application programs, can be started and stopped dynamically like any other DPPX application programs. This capability meets the requirement of dynamically activating and deactivating different physical units as required. The basic DPPX commands, START and STOP,[7] are used to accomplish this function.
2. New keywords on the DPPX START command allow system definition information relating to the physical units to be entered dynamically at start time; no other system definition information is necessary.
3. Since the physical units are applications from a system point of view, they can maintain connections with the SSCP as application-to-application connections.

4. Numerous development advantages exist in being able to develop the physical units as separate and distinct development packages. The application programs are written to well-defined and well-documented interfaces.
5. There are no design restrictions on the number of physical units existing in one DPPX system; the restriction will normally be one of storage availability within the DPPX system.

The SSCP is also implemented as an application program. The SSCP (by SNA definition) maintains connections with each network resource in its domain (except SNA links). These connections will simply be DPPX application connections. As an example, the SSCP connection to a DPPX physical unit (which is itself an application program) will be an application-to-application connection within DPPX. The system definition parameters required by the SSCP itself are entered dynamically on the START command, and hence no special system definition is required for the SSCP. It can be seen that the definition, activation, and deactivation of the SSCP and PUs control functions within DPPX and are therefore completely dynamic.

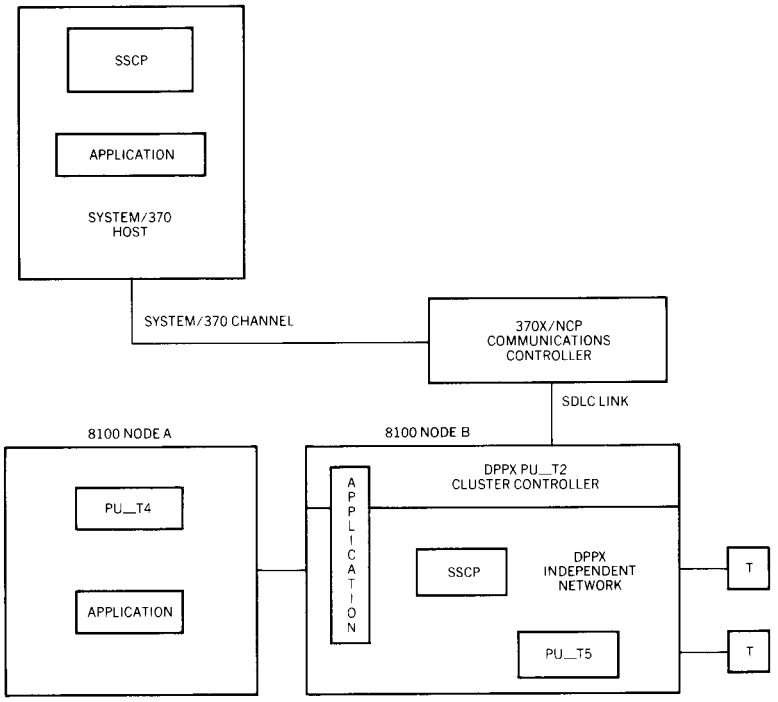
terminal definition

As described earlier, all terminal resources, including an interconnected DPPX system, are defined to the managing DPPX system in the same basic way. It is appropriate at this time to describe the way in which these terminal resources are defined in DPPX.

DPPX includes a command system by which users of the system can enter either system or user-defined commands. One of the system commands available is DEFINE.NET,[7] by which the user can define a particular network resource to the configuration manager (SSCP). When a user is defining a network consisting of a number of resources, a DEFINE command for each network resource is entered to the command system. For a large network, these individual commands will normally be entered as a list of commands which is then stored on a data set. This list can then be entered for execution to the command system by entering another DPPX command.

Once the initial DPPX system has been defined, it may be necessary to change the network configuration either by adding new terminals or perhaps by adding an interconnected DPPX system. In this case, the DPPX configuration can be changed dynamically by simply entering the appropriate DEFINE commands to the command system. While the network configuration definition is being updated, the previously defined network is not affected and can continue operation. It is not necessary for there to be any down time for the system, so that as the user grows and adds more capability within the DPPX network, the definition process does not cause a major upheaval of the user's operations; useful work can still be carried out at the system location.

Figure 10 Typical DPPX configuration



Once the configuration has been defined, the new resources can be activated and are then available for use within the network.

Application network usage capability

Up to this point, the three basic ways in which the DPPX system can be configured have been discussed, and certain system definition aspects of the DPPX system have been described. This section will concentrate on DPPX application usage of the network configurations and will discuss the DPPX definition process for application programs.

It may often be necessary to have more than one of the network configuration capabilities present in the same DPPX system at once, depending on the user's configuration requirements. A more complex DPPX configuration is depicted in Figure 10. Shown is one DPPX node (Node B) consisting of a host attachment capability in addition to a stand-alone function. In this configuration the DPPX application in 8100 Node B can communicate with the following resources: (1) host application programs, (2) DPPX application programs in both 8100 nodes, and (3) attached terminals in both 8100 nodes. The DPPX application in 8100 Node A can com-

municate with the following resources: (1) DPPX application programs in both 8100 nodes and (2) attached terminals in both 8100 nodes.

A typical function needed with this configuration is to transmit input from a terminal to a host application program and transmit data from the host application program to the terminal. This procedure can be accomplished by having a pass-through application program that supports connections to host applications and DPPX terminals. In this way the user at the DPPX-supported terminal can appear, externally at least, to be directly connected to the host system—the interface seen at the terminal is as if the terminal were directly connected to the host. In Figure 10 the terminal can be attached to either 8100 Node A or 8100 Node B. IBM 3270 Data Stream Compatibility is an IBM-licensed program that provides such a capability.

It is appropriate at this point to discuss some network communication aspects of how this is made possible within DPPX.

As can be seen in Figure 10, the host application program resides within the host domain, and the DPPX terminal resides within the DPPX domain. The terminal and host application program thus belong to two different domains, and the intermediate DPPX application program, often referred to as a pass-through application, is maintaining connections in two different domains. To accomplish these connections without requiring cross domain support, the DPPX application program has to be represented in each of the domains as a network resource (in this case a logical unit (LU)). It must be possible for the DPPX application program to be assigned a logical unit in both the DPPX domain and the host domain in this particular example. Since in the general case many different domains may be represented in one DPPX system, it is important to be able to allocate to a DPPX application program a logical unit in any particular domain or domains so that it can communicate with other resources (terminals or other application programs) existing in the same domain. This has been achieved as explained below.

application sign-on capability

For a DPPX application to become a member of a DPPX domain, the application issues a macro (SIGNON) that causes the following internal DPPX processing to be carried out:

- Dynamically creates a logical unit within the domain specified on the macro.
- Dynamically informs the DPPX SSCP (via the SSCP-PU connection) of the creation of a new network resource within its domain, at which time the SSCP adds the resource definition to its list of network resources.
- The SSCP then causes the new resource to be made active and thus usable within the domain.

All information relating to the new resource and required by the SSCP is either entered directly on the SIGNON macro, or provided internally by DPPX communications processing. The system programmer does not have to enter any definition commands to the DPPX system on behalf of the application program. The information required by the SSCP is transmitted by the physical unit representing the particular domain at that node. Note that the domain must be represented at a particular node; otherwise an application cannot become a member of the domain and cannot communicate with other resources in that domain.

For example, in Figure 10, two domains exist in 8100 Node B: one is a host domain and the other is a DPPX domain represented by the PU_T5. Any application in this DPPX node has the capability of becoming part of only these two specific domains. It cannot communicate with any resources that are part of another domain not represented at the 8100 Node B.

If the application program is moved to another 8100 node in which the same domains are represented, the same processing will occur and no change will be required of the application program. DPPX applications, from a network configuration viewpoint, are hence dynamic and portable and can effectively be shifted around within the network.

For attaching to a host network, the application can utilize the same external interface (SIGNON macro), but in this case no dynamic definition occurs to the host SSCP during the internal processing of the SIGNON function. In this case resources have to be predefined to the host and DPPX systems to meet the existing host system definition procedures. For host networks, all logical units have to be defined as part of the host system generation procedure, and all DPPX logical units defined to the host system are built by DPPX when the particular physical unit (in this case PU_T2) representing the host network is started within DPPX. These resources must exist before any DPPX applications issue SIGNON requests since they are not built and activated dynamically as part of SIGNON.

The allocation and activation of a logical unit (supporting a DPPX application program) within a DPPX unique network is completely dynamic, and no prior network definition of the application program is required. The SIGNON macro is in effect the network definition procedure for a DPPX application program within a DPPX network. Any change in the definition of the application can be achieved by alteration of the SIGNON parameters.

Summary

It has been shown that the 8100 Information System and DPPX operating system can assume the external appearance of three

different physical unit types (PU_T2, PU_T5, PU_T4), and these physical units can be dynamically started and stopped within the same DPPX system. The DPPX system has the capability of attaching to a host system, operating as a stand-alone system independent of any other system, and operating as an interconnected system.

The capabilities of DPPX in this area are such that it allows a wide variety of configurations to be supported and is also extendable for any future additional configuration requirements.

The DPPX system has set a precedent towards the dynamic definition of network resources both for terminals and application programs. Terminal resources and application programs are both defined dynamically in the DPPX system, and formal system generation is not required. It is easy to install an 8100-DPPX system, and once the system is running, it is a simple matter to change the configuration in a dynamic manner.

The basic support for different types of terminals was shown to be the same externally and internally to the application program and system operator, and it will be possible to support new terminals within the same structure.

The overall solution to these problems within a general network design framework will allow the 8100 system and DPPX to provide a solid base for IBM in distributed processing.

ACKNOWLEDGMENTS

The author is grateful to the team of experienced system designers who worked on the communications design of DPPX. In particular, the author expresses his thanks to L. C. Thomason, H. R. Albrecht, and A. F. Banks for their creative contributions to the DPPX communications structure.

CITED REFERENCES

1. S. C. Kiely, "An operating system for distributed processing—DPPX," *IBM Systems Journal* **18,** No. 4, 507-525 (1979).
2. *Systems Network Architecture General Information*, GA27-3102, IBM Corporation; available through the local IBM branch office.
3. A. L. Scherr, "Distributed data processing," *IBM Systems Journal* **17,** No. 4, 324-343 (1978).
4. H. R. Albrecht and L. C. Thomason, "I/O facilities of the Distributed Processing Programming Executive (DPPX)," *IBM Systems Journal* **18,** No. 4, 526-546 (1979).
5. *SNA Format and Protocol Reference Manual*, SC30-3112, IBM Corporation; available through the local IBM branch office.
6. *Distributed Processing Programming Executive Base (DPPX/BASE): Guide to System Services*, SC27-0405, IBM Corporation; available through the local IBM branch office.
7. *Distributed Processing Programming Executive Base (DPPX/BASE): Commands: Configurations and Operations*, SC27-0511, IBM Corporation; available through the local IBM branch office.

GENERAL REFERENCES

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

J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15**, No. 1, 4-23 (1976).

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).

C. R. Blair and J. P. Gray, "IBM's Systems Network Architecture," *Datamation* **21**, No. 4, 51-56 (April 1975).

Distributed Processing Programming Executive Base (DPPX/BASE): General Information, GC27-0400, IBM Corporation; available through the local IBM branch office.

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

Reprint Order No. G321-5122.