SNA Perspective On Internetworking

Volume 14, Number 7
July, 1993
ISSN 0270-7284

The single source, objective monthly newsletter covering internetworking with IBM's Systems Network Architecture

Inside APPN's New Dependent LU Support

In the November 1992 issue of SNA Perspective we presented an overview of one IBM strategy for supporting communications with dependent logical units (DLUs) across APPN networks. That technology is called the Dependent LU Requester/Server (DLUR/S) architecture. IBM has recently released the details describing how this architecture for legacy LUs will actually be implemented. In this article we'll examine the extensions that have been added to APPN to accommodate the huge installed base of DLU-based applications.

Lack of dependent LU support has probably been the single greatest obstacle in the migration from hierarchical subarea SNA networking to the newer decentralized APPN technology. The simple fact is that the vast majority of the SNA installed base is still using dependent logical units such as LU0, LU1, LU2, and LU3. They are called dependent LUs because they depend on the presence of a System Services Control Point (SSCP) to perform session activation and deactivation functions. The bottom line is that dependent LUs must have access to the services of an SSCP. The DLUs which are serviced by an SSCP are said to be in that SSCP's domain.

(continued on page 2)

An Alternative View of APPN Implementation

Almost everyone knows by now that IBM is licensing its source code for the implementation of APPN, including Network Node support. But less widely known is the fact that a portable APPN code implementation is also available from another source, Data Connection Limited (DCL). DCL is based in England and has a long history of marketing SNA software to both OEMs and end users.

In providing a generalized, portable alternative to IBM's APPN source code, the DCL software product, and possibly others which are yet to come to market, may help to change the market perception that APPN is an IBM-only technology. While

(continued on page 10)

In This Issue:

(continued from page 1)

Subarea/APPN Integration

Before we look at the DLUR/S architecture in more detail it's useful to recap the subarea/APPN integration problem that it is designed to solve. When DLUs residing in SNA peripheral nodes are attached to SNA subarea nodes as shown on the left side of figure 1, the SSCP-PU and SSCP-LU sessions with the host can pass through one or more intermediate subarea nodes - in this case a Communications Controller running the Network Control Program (NCP).

The network on the right side of figure 1 shows what happens when the direct data link connection between the peripheral node and the Communications Controller is replaced by an APPN network. The reason for this change might be to support a new distributed application, possibly a client-server application. In this migration to distributed computing, ongoing support for the applications based on the use of dependent LUs, such as 3270 applications, continues to be a requirement.

When one or more APPN Network Nodes are positioned between the SNA subarea network and the peripheral node where the DLU resides, the DLU is cut off from its SSCP's domain. Specifically, the APPN network is not capable of directly transporting the SSCP-PU and SSCP-LU sessions which are required to support the downstream DLUs. Only LU 6.2 sessions are supported.

Two Possible Solutions

There are now two solutions that are available to the problem of running DLU sessions over APPN networks. The first solution existed within the APPN architecture even before the addition of the DLUR/S extensions. APPN includes a group of option sets which are designed to handle the session initiation procedures used by DLUs. The following option sets provide these session services extensions.

1060 - Prerequisites for session services extensions

1062 - Session services extensions CP support

1063 - Session services extensions NNS support

1064 - Session services extensions PLU node support

1065 - Session services extensions CP(SLU)(SSCP) support

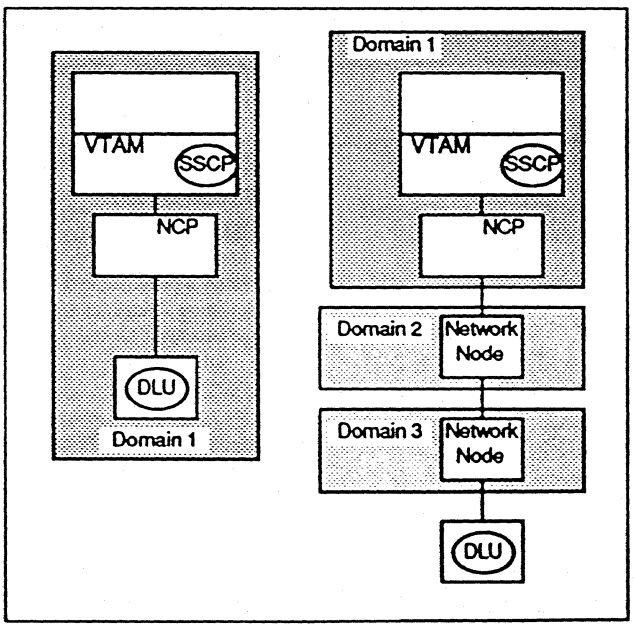


Figure 1

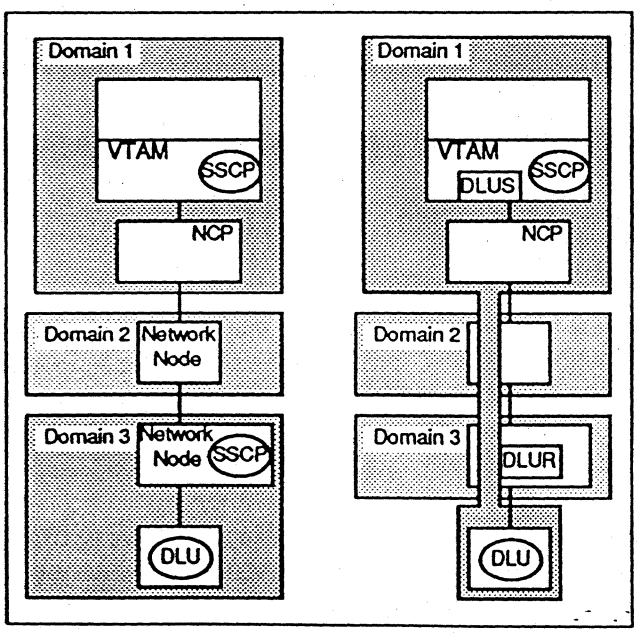


Figure 2

The session services extensions essentially allow SSCPs residing on Type 2.1 Nodes to be distributed throughout an APPN network. The result is the configuration shown on the left side of figure 2. AN SSCP now resides on the Network Node which is adjacent to the peripheral node containing the DLU.

The main drawback of this approach is the cost of putting an SSCP on each Type 2.1 Node in the network which is adjacent to DLUs which must communicate across the network. Each of these nodes would essentially have to run the equivalent of VTAM. The cost of running this software on many distributed systems is not the only issue. The proliferation of distributed SSCPs would also result in configuration management problems because each SSCP would require local configuration.

The alternative solution is to extend the reach of existing SSCPs across APPN networks rather than proliferating SSCPs. This is the approach taken by the Dependent LU Requester/Server (DLUR/S) architecture. Implementation of the DLUR/S architecture results in the environment shown on the right side of figure 2. Note that the SSCP is not adjacent to the DLU in the DLUR/S configuration. Instead, the "reach" of the SSCP is extended through the use of a Dependent LU Server (DLUS) and a Dependent LU Requester (DLUR). The DLUS and the DLUR communicate with one another to tunnel management information between the SSCP and the DLU and its supporting PU.

The benefit of the DLUR/S architecture is that SSCPs don't have to be proliferated throughout a network. Instead of putting an SSCP at locations which are adjacent to each group of DLUs, a DLUR is substituted. DLURs are considerably simpler and less resource intensive than SSCPs. DLURs, as we will see in this article, can be implemented on a wide variety of platforms including PCs and networking controllers such as the 3174.

In large enterprise networks both the DLUR/S architecture and the APPN Session Services Extensions will be employed to support users of dependent LU-based applications.

How the DLUR/S Architecture is Implemented

APPN's support for DLUs is implemented by three major components:

- Dependent LU Server (DLUS) - APPN option set 1066
- Dependent LU Requester (DLUR) - APPN option set 1067
- CP-SVR Session Pipe

DLUSs are implemented on the nodes where SSCPs reside. IBM's initial DLUS product is VTAM. DLURs are designed to be implemented on nodes where DLUs reside or on nodes which act as DLUR/S gateways to connect DLUs on downstream nodes to SSCPs via intermediate APPN networks. The CP-SVR session pipes transport encapsulated SSCP-PU and SSCP-LU traffic between DLURs and DLUSs. The CP-SVR session pipes use LU 6.2 sessions for communications between DLURs and DLUs.

Figure 3 shows some of the possible DLUR/S configurations. A DLUS must always reside on an APPN Network Node which also implements an SSCP to manage the attached DLUs. VTAM will combine the DLUS with the required SSCP and Network Node support.

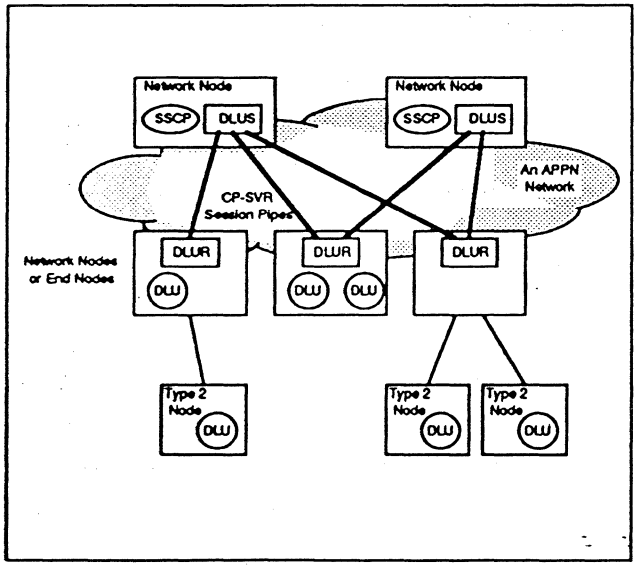


Figure 3

Dependent LU Requester Capabilities

This is a summary of some of the capabilities defined by the DLUR option set.

CP-SVR Session Pipe Operations

- Activate and deactivate CP-SVR sessions
- Support for CPSVRMGR mode and class of service
- Map local and downstream PUs to the appropriate DLUS node(s)

Handling of SSCP-PU and SSCP-LU Sessions

- Correlation of SSCP sessions to corresponding PU images based on Fully-Qualified Procedure Correlation Identifiers (FQPCID)
- Generate REQACTPU and REQ DACTPU requests to activate and deactivate SSCP-PU sessions
- Handle the encapsulation and unen capsulation of FID2 PIUs flowing on the SSCP-PU and SSCP-LU sessions
- Support dynamic definition of LUs by encapsulating Product Set ID NMVT commands generated by PUs
- Capture LU names contained in ACTLU requests to relate SLU names to their respective PU images
- Acquire PLU names from INIT-SELF requests

LU-LU Session Support

- Convert BIND requests destined for devices that do not support extended BINDs or network-qualified names
- Reject BINDs for already active DLUs
- Handle adaptive session pacing
- Handle routing of BINDs to destination PU/LU based on SLU name table
- Perform intermediate routing functions such as handling pacing protocols and session outage notification for LUs located on downstream PUs

Session Awareness Reporting

- SSCP-LU sessions
- LU-LU sessions

Each DLUS can support one or more DLURs which, in turn, support the connection of the dependent LUs. The DLURs can reside on either Network Nodes or APPN End Nodes. Note that the supported DLUs can either reside on the same node as the DLUR or on other downstream nodes. These downstream nodes would most commonly be existing Type 2 nodes such as 3270 emulation products. When downstream Type 2 nodes are supported, the DLUR acts as a gateway which allows DLUs to access SSCPs across APPN networks.

The DLURs perform a variety of functions including:

- Activation and deactivation of the CP-SVR Session Pipe
- Encapsulation of SSCP-PU and SSCP-LU session traffic
- Multiplexing and demultiplexing multiple SSCP sessions over the CP-SVR Session Pipe
- Generating requests for the activation and deactivation of SSCP-PU sessions
- Mapping local PUs and LUs to the DLUS that is associated with their owning SSCP
- Handling session-level pacing protocols between the APPN and subarea networks

- Reporting session awareness information to the SSCP
- Translating between extended and unextended BIND formats

Secondary dependent LUs can enter into sessions with primary LUs residing in either SNA subarea networks or within APPN networks. When a PLU resides in a subarea network, the SSCP employs SSCP-SSCP sessions to establish the LU-LU session across domains. If the PLU is in an APPN network the server uses APPN's Session Services Extensions protocols to initiate the LU-LU sessions.

The CP-SVR session pipes are LU 6.2 sessions which can be started on demand from either the server side of the connection or the requester side. If an ACTPU operation is initiated from VTAM, the DLUS must be manually configured with the name of the DLUR which supports the PU to be activated. When a Type 2 Node is activated it sends an indication (such as an XID command) to its DLUR. When this occurs, the DLUR must be configured with the name of the DLUS that will own the PU and LUs in the Type 2 Node. IBM says that these manually configured requesters and servers will be supplemented at some time in the future by a dynamic discovery procedure that will minimize pre-configuration requirements.

Starting the CP-SVR Session Pipe

CP-SVR session pipes are started on demand by either the requester or the server. The server can begin the activation sequence in response to a network operator command to activate a DLUR/S-supported PU. The requester would activate the session pipe in response to an activation request from one of the PU images that it services.

The CP-SVR session pipe activation sequence shown in figure 4 is initiated by the requester in response to the activation of one of its downstream PUs. This sample shows the interactions between the CP-SVR session pipe activation and the start-up of an SSCP-PU session to support the newly activated downstream PU.

The PU Type 2 node which is downstream from the requester is activated and sends a data link level XID command to its DLUR. The DLUR begins the activation sequence by determining which DLUS represents the SSCP that will "own" the downstream PU. These PU-to-DLUS/SSCP relationships must currently be configured manually, but at some point in the future IBM indicates that a dynamic discovery process will be added to the DLUR/S architecture.

Activation of the Inbound Pipe

In our example, we assume that there is no currently active session pipe to the target DLUS and, therefore, a session pipe must be started. If a pipe was already active, the pipe activation sequence in our example would not have been required.

Since the CP-SVR session pipe is a standard LU 6.2 session, the activation sequence employs the standard APPN LOCATE command to find the location of the target resource which in this case is VTAM. The session that is activated uses the DLUR/S-defined CPSVRMGR mode.

Requesting the PU Activation

After the inbound session pipe is activated, the DLUS must be informed of the fact that a PU is requesting an activation. Readers who are familiar with SNA know that there have not been any SNA-level protocols defined to request PU activations by SSCPs; this situation changes with the introduction of the DLUR/S architecture. As shown in our sample activation sequence, a new SNA command called REQACTPU has been added. The purpose of the REQACTPU command is to allow the DLUR to send a request for SSCP-PU activation to the DLUS. A new command called REQDACTPU has also been added to request the termination of SSCP-PU sessions.

The REQACTPU, like all of the SNA commands which flow through the session pipe, are encapsulated within the LU 6.2 session. The details of the encapsulation scheme will be discussed later in this article, but note that the DLUR sends an FMH5 Attach to the server to initiate the Receive_Encap_Msg_TP transaction program. This transaction program interprets the encapsulated data

stream and strips off the information which is added to the command during the encapsulation process.

The unencapsulated REQACTPU command is then sent from the DLUS to the SSCP which will actually initiate the SSCP-PU activation. If the session activation request is valid, which we assume in this example, the SSCP returns a positive response to the DLUS. This positive response must now be sent back to the DLUR which initiated the request.

Activating the Outbound Pipe

In order to send the response to the requester, an outbound session pipe will have to be activated. This procedure is essentially the mirror image of the inbound pipe initiation.

The response to the REQACTPU is encapsulated by the server and an Attach is sent to initiate the receiving transaction program. Note that this transaction program is started for each encapsulated SSCP command sent through the session pipe.

In parallel with the transmission of the REQACTPU response, the SSCP will send the ACTPU command which has been requested. The ACTPU is first sent from the SSCP to the DLUS where it is encapsulated for transmission to the DLUR which initiated the request.

When the ACTPU is received by the requester it unencapsulates the command and then uses identifiers carried by the encapsulated command to route the ACTPU to the proper downstream device.

The ACTPU response is then encapsulated and sent to the DLUS via the inbound CP-SVR session pipe which was previously started.

Activating Logical Units

The activation of SSCP-LU sessions is a straightforward encapsulation process. The example show in figure 5 employs SNA's dynamic registration of dependent LUs. The example assumes that the PU 2 device implements the Self-Defining Dependent LU support which generates an NMVT command whenever an LU activation is required. The NMVT command carries a Product Set ID which is used by the SSCP to dynamically create the appropriate type of logical unit.

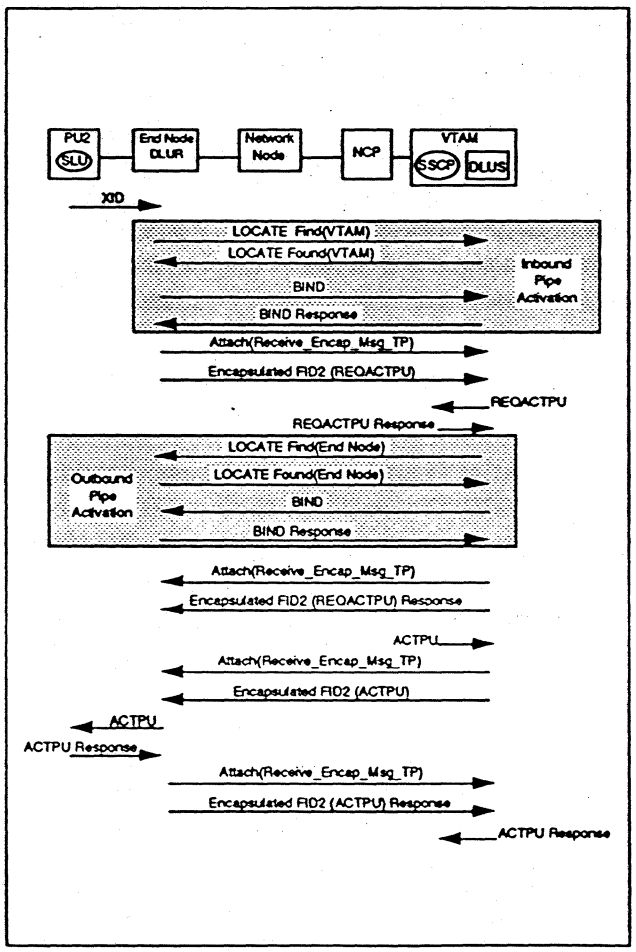


Figure 4

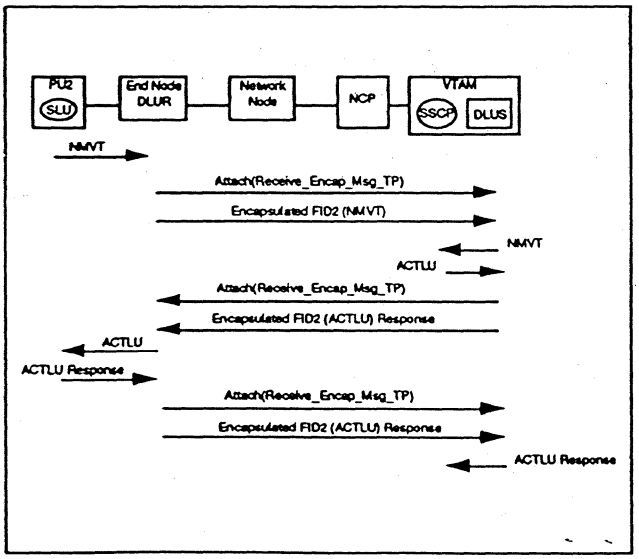


Figure 5

If the LU in our example had been predefined rather than using the Self-Defining LU support, the NMVT processing would not be performed, but the ACTLU request and response processing would be carried out as shown.

The SSCP generates an ACTLU request which is sent to the DLUS. The DLUS encapsulates the ACTLU request and sends it over the LU 6.2 session to the DLUR which unencapsulates the ACTLU and sends it to the downstream DLU node. The response to the ACTLU is encapsulated by the DLUR and sent to the DLUS which unencapsulates it and relays it to the SSCP.

LU-LU Session Activation

So far, we have activated the PU and LU in the downstream DLU node. Keep in mind that all of these sequences would be the same if the DLU were located on the same Network Node or End Node as the DLUR. The only difference would be that the SSCP session activation requests and responses would flow internally within the DLUR node.

In our example shown in figure 6 the LU-LU session is initiated by a logon from the SLU. Since the logon flows on the SSCP-LU session, it is encapsulated by the DLUR and sent to the DLUS which forwards it to the SSCP. An encapsulated response is sent back to the SLU.

The USS message which indicates that a logon is in process is encapsulated and sent to the SLU.

The SSCP must now notify the PLU that it has received a logon request. In our example, the target PLU resides within the APPN network on an APPN End Node. In order to initiate this LU-LU session, the DLUS uses the APPN Session Services Extensions. These are the protocols which we talked about earlier in this article which give APPN nodes the ability to activate DLU LU-LU sessions.

The LOCATE requests which are directed toward the PLU indicate that a CDINIT operation is being requested. When this LOCATE request is received by the Network Node it calculates the "best" route to the DLUR. Note that if multiple alternate paths are available between the PLU and the DLUR, any of those paths would be eligible to carry the LU-LU session. There is no requirement that the LU-LU sessions use the same path as the CP-SVR Session Pipelines which are used to support the SLU's SSCP-PU and SSCP-LU sessions.

Note that the BIND request is not encapsulated. It is sent directly from the PLU to the SLU over the best available route. In our sample network that route will traverse both an APPN network and an SNA subarea network. This means that the BIND request must be modified as it travels to the target SLU.

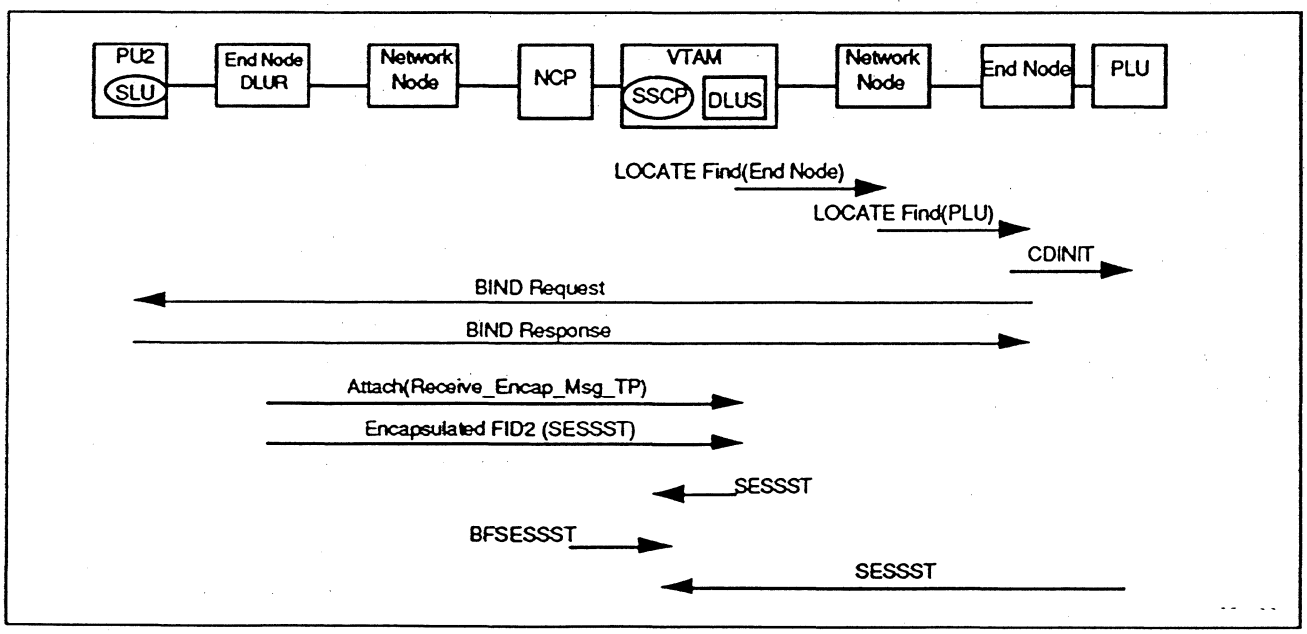


Figure 6

Since the PLU resides within an APPN network in our example, the BIND command that it generates will be an extended BIND image. These extended BINDs carry the Route Selection Control Vector (RSCV) which provides the source routing information needed to deliver the BIND across the APPN portion of the network along with other control vectors.

In most cases, these BIND requests must be "unextended" prior to being sent to the devices which support the dependent SLUs. This unextending of the BIND request is performed by the DLUR.

Multiplexing and Demultiplexing SSCP Sessions

All of the SSCP session data traffic flowing between a DLUR and a DLUS is multiplexed onto a single CP-SVR session pipe. The PIUs flowing to and from any particular PU are identified by a unique Fully-Qualified Procedure Correlation Identifier (FQPCID). When a PU activation originates at the DLUR, it is the DLUR's responsibility to generate a unique FQPCID. In this case, the FQPCID is generated just before the REQACTPU request is sent from the DLUR to the DLUS. All subsequent PIUs flowing to or from that PU will carry that PU's FQPCID. When PU activation originates at the DLUS it is the responsibility of the server to generate the unique FQPCID before sending the ACTPU request.

Each PIU flowing on an SSCP-LU session is related to a particular logical unit. PIUs encapsulated on CP-SVR session pipelines carry the standard source and destination address fields in their transmission headers. These addresses are used to relate PIUs to their respective LUs.

How PIUs Are Encapsulated

All data traffic flowing on SSCP-PU and SSCP-LU sessions must be encapsulated in LU 6.2 sessions for transport across APPN networks. This is due to the fact that APPN networks don't directly support SSCP sessions. Only LU 6.2 sessions are directly supported as we mentioned earlier. The encapsulation occurs on the CP-SVR session pipes which use LU 6.2 session protocols.

As shown in figure 7, each FID2 Path Information Unit (PIU) is encapsulated within a new Generalized Data Stream (GDS) variable type which has been assigned identifier X'1500'. This FID2 encapsulation variable also includes a PIU length field as well as some control vectors which are used to convey additional information between the dependent LU requesters and servers.

The following control vectors can be carried within the FID2 encapsulation variable along with the PIU itself:

- TG Descriptor (X'46') - indicates whether the PU is internal to its DLUR or, for externally attached PUs, it indicates the type of LAN or WAN that provides the attachment.
- DLUR/S Capabilities (X'51') - identifies the release level of the DLUR/S support, whether a node is a DLUR or DLUS, and several other capabilities.
- Fully-Qualified Procedure Correlation Identifier (FQPCID) (X'60') - a unique identifier which is used to relate encapsulated PIUs to their respective PUs.

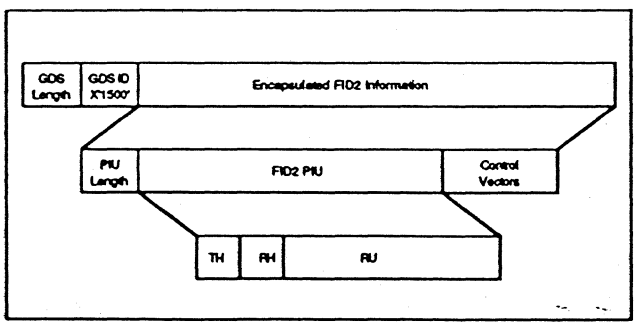


Figure 7

- XID Image (X'81') - contains the I-field information from an XID0 or XID3 SDLC response of a PU attached to a DLUR.
- Other control vectors such as Security ID Control (X'25'), Assign LU Characteristics (X'30'), and Extended SDLC Station (X'43')can also be carried along with the encapsulated PIUs.

SSCP Takeover Processing

Most SNA subarea networks are designed to provide for the takeover of resources in the event of an SSCP failure. The takeover procedure involves restarting the SSCP-PU and SSCP-LU sessions with a backup SSCP. The DLUR/S architecture must participate in the SSCP takeover procedures.

The first step in the takeover procedure is for the DLUS which represents the backup SSCP to establish new CP-SVR Session Pipes with the DLURs that represent the resources (PUs and LUs) that are to be taken over. After the CP-SVR Session Pipes are activated, the new encapsulated SSCP-PU and SSCP-LU sessions are activated by the new SSCP and its DLUS.

Identifying Resources for Network Management

The encapsulation of SSCP-PU and SSCP-LU sessions within the CP-SVR session pipes presents some new SNA network management challenges. The network operators must understand the relationships between the SSCP sessions and the CP-SVR session pipes which encapsulate them. They must also be able to relate these sessions and session pipes to the network resources which are actually involved in supporting the connections. The management architecture which will address these issues has yet to be defined, but IBM has indicated that the necessary management extensions will be added to the DLUR/S architecture.

The DLUR must also be able to relate the LU and PU names which are used by network operators to the actual resources that are being supported by the requester. These could either be DLUs located on the requester node itself or they might be PUs and LUs which reside on devices which are downstream from the requester node. In either case, it is the job of the DLUR to perform the necessary name mappings.

The DLUR captures the names of LUs and PUs at the time that they are activated. The procedure for obtaining PU names is slightly different from that used to capture LU names. The current ACTLU request format carries a Network Name control vector which contains the name of the LU being activated. Whenever the DLUR passes an ACTLU request to a DLU it learns the LU's name.

The current ACTPU request format does not include a PU name. The DLUR and DLUS compensate by modifying the ACTPU request to include a Network Name control vector. The PU name is added by the DLUS when the ACTPU is encapsulated and the DLUR will capture and strip off the PU name before passing the request to the target PU.

DLUR/S Product Packaging

IBM's DLUS support will be packaged within a new release of VTAM. We may also see DLUS support from other vendors. The likely candidates would be those vendors who already provide SNA Host (Node Type 5) support. Note that IBM has not yet published the architectural specifications which describe the DLUS.

There will be a much greater variety in the packaging of DLUR support. IBM has announced it for the 3174 and has indicated that DLUR support for OS/2 platforms will also be coming.

We expect to see DLUR support from many vendors other than IBM as well. Many vendors are already working to deliver the APPN End Node or Network Node support that is a prerequisite for implementing a DLUR.

Many vendors of 3270 emulation products will follow IBM's lead with the 3174 and add DLUR capability to their products. Internetworking product vendors who have made a commitment to the SNA market are also likely candidates for providing DLUR products. The implementation of DLUR makes sense in routers and intelligent hubs which will be required to support the attachment of legacy SNA devices.

The Market Impact of the DLUR/S Architecture

Years after IBM originally introduced APPN it remains a very small percentage of the overall SNA market. One of the most important obstacles to the acceptance of APPN has been the lack of a migration path for legacy applications. While APPN has been limited to supporting only LU 6.2 sessions, the vast majority of applications in most SNA networks are still based on the older dependent LU types such as LU0, LU1, LU2, and LU3.

Some SNA users who found APPN to be an attractive solution to support decentralized networking and dynamic network configuration, began to implement it to support new applications such as client-server applications, but these users found that they had to maintain parallel networks. The early LEN support for VTAM was not a viable solution for most customers because it did not create a seamless APPN/subarea network.

The DLUR/S architecture is one key to changing this situation. Once the DLUR/S architecture is implemented in VTAM and other products, dependent LUs will be fully supported across hybrid APPN/subarea networks. Dependent LUs can reside on both APPN and subarea networks, and their users will be able to communicate with each other freely and without any consideration for whether users are located in APPN networks or subarea networks.

The other key to supporting the migration of SNA users to APPN is already available in VTAM Version 4 Release 1. VTAM hosts can be configured as subarea nodes, APPN nodes, or combinations of the two. This will allow users to incrementally migrate their SNA backbone networks to APPN. In fact, the hybrid APPN/subarea host that we used in our examples of DLUR/S protocol exchanges is VTAM configured as both an APPN Network Node and as a subarea host. This type of hybrid configuration is aptly referred to as a migration host because it acts as a gateway between an APPN network and a subarea network.

When the DLUR/S support becomes available in VTAM and other products such as the 3174 and OS/2 platforms, SNA users will finally have the basic tools that they need to move ahead with the migration of their legacy SNA network to APPN.

(continued from page 1)

APPN's design continues to be entirely controlled by IBM, the availability of APPN implementations from multiple vendors should result in wider availability of APPN-compatible products.

We thought that it would be interesting to get a non-IBM view of the APPN implementation process and the APPN marketplace in general, so SNA Perspective (SNAP) interviewed two of the managers involved in the development and marketing of DCL's SNAP APPN product. Phil McConnell (PMcC) is director in charge of Data Connection's SNA Development Group. John Palombo (JP) is the director in charge of the SNAP APPN development team. Our interview with them follows.

SNAP: Phil, can you tell us a little about Data Connection?

PMcC: Certainly. Data Connection is a British company, based in London, England, with an office in Washington, DC. We are currently 130 people strong, with about 100 of those people in the development area. Last year, our sales were approximately $16M, some 90% of which were in the US.

Our main areas of business are in SNA and OSI communications, and graphics/multimedia and retail, with our typical customers being the development organisations of the major players in the industry (such as IBM, Microsoft and HP) and large end-users (such as the banks, government, and airlines).

In the SNA communications area, we have been delivering the SNAPS portable SNA products since 1983. The product range includes SDLC, 3270, LU6.2, PU2.1, X.25/QLLC, SNADS, and NetView support - and now SNAP APPN.

We typically license the core SNAPS products to our customers, and provide engineering and support services to assist the OEM with porting the products and adding value to their platform.

SNAP: Is the APPN product your own development?

PMcC: Yes. We developed SNAP APPN from the ground up. However, the development leaned heavily on the in-depth SNA expertise we had built up over the previous decade.

SNAP: What requirements were you trying to address when you started the development of SNAP APPN?

PMcC: We were trying to address requirements from our existing (and potential) OEMs, and from their customers - i.e., end-users who were feeding requirements back to the OEMs.

Both groups were interested in the level of APPN functionality, and in providing rich support functions for configuration, management and diagnostics.

In addition, our OEMs wanted a product that would be easy to port to their environments, and would be flexible and scalable to allow them to configure it for different market segments.

Taking each of these in turn. We decided that we wanted to provide an APPN product with the richest possible function, with a combined Network Node and End Node, and a full range of APIs on both Network Node and End Node, including LU6.2 APPC and CPI-C, LU 0, 1, 2 and 3, NOF configuration and administration, Management Services, and Data Link Control.

Full Network Node support was a requirement, including support for all the defined APPN V2 base function sets and a sensible selection of APPN V2 optional function sets.

Good support for multiple LAN and WAN types was a requirement along with a full range of network management capabilities, namely NetView, Multiple Domain Support (MDS) and SNMP MIB. We also wanted to provide fully dynamic configuration, whereby resources could be added, configured and removed on the fly.

It was clear that we had to provide a full range of query functions to provide running information about the state of the local node, and of the network.

It was also clear that the product had to be engineered for ease of use, which meant that as well as the configuration and query capabilities, it had to be built with a strong emphasis on clear and simple diagnostic aids.

Perhaps the easiest requirement (or at least the best understood given our background in developing portable SNA products) was coping with all the "normal" portability issues, such as big versus little endian, different compilers, different data mappings, etc.

Perhaps harder, but very importantly, the product had to be scalable, so that the same code could run on a wide variety of different platforms, from PCs, though workstations and routers, to mainframes - and still perform well in every environment.

For example, router OEMs are interested in providing APPN on different router models targeted at different market segments, which (in some cases) had different mixes and numbers of processors. SNAP APPN had to run on all models, and had to be flexible enough to place protocol components on different processors in different models, for example

in migrating performance critical components onto high performance line cards in the high end models.

Overall, this gave us a pretty daunting set of requirements. But it was clear that if we were to succeed (and indeed lead) with SNAP APPN, we had to meet them.

SNAP: At the time you started the SNAP APPN development, the Network Node specifications were not available. How did you go about engineering a Network Node product?

PMcC: You're right in saying the specifications weren't available when we started (in the second half of 1991) - at least not in one place. However, a large part of the required information had been published previously in various IBM and non-IBM documents.

IBM SNA manuals provided peripheral information and hints (particularly the APPN Architecture and Product Implementations tutorial). We also used IBM patent documents and various journal papers as sources of information. (See sidebar for a list of key documents)

When the Network Node specifications were released earlier this year, all of this information was brought together into one place. There were some small additions and changes in this document from the sources we had been using, but the vast majority of our researched Network Node architecture proved to be correct, and the effort to align SNAP APPN fully with the published architecture was minimal.

SNA: John, what were the key technical challenges that your team encountered during the project?

JP: Our understanding of the SNAP APPN requirements meant that the development team faced a whole range of technical challenges. First, there are the architectural issues.

There are arguments for and against staying close to the APPN Architecture. Our experience is that staying close to the defined architecture is in general worthwhile, as it generally provides a reasonable subdivision of function and tends to facilitate the incorporation of future architectural enhancements. It is also something our customers prefer.

However, there are cases where diverging from the architecture has advantages or may even be required. For example, we found we could remove some of the architecture-defined checks and gain substantially improved performance without impacting functionality. Indeed, removing checks was sometimes necessary to interoperate with existing LU6.2 LEN and End Node implementations that did not completely adhere to the architecture.

We also made slight changes to the split of function across components to get more straightforward and/or maintainable code. For example, we split TP from PS and moved session counts from SS to CS.

We introduced a number of pragmatic non-architectural changes to cope with existing implementations that did not follow the architecture. For example, Extended Services accepts binds even when the session limit is exceeded, provided the maximum session limit is not exceeded. Similarly, CICS only reactivates sessions if the session was ended using UNBIND(CLEANUP). Both these examples are architecturally incorrect, but SNAP APPN contains code to cope with them.

Where we found problems or thought there was a case for architectural changes we provided feedback to IBM - who were more than interested to get input from another set of APPN implementors.

SNAP: What about the portability and scalability of the SNAP APPN product?

PMcC: As I said earlier, we fully understand language portability issues from delivering portable SNA for the last decade. It's not easy - by any means - but we know how to do it.

In addition, we substantially reworked our existing portable operating environment technology to address the environment portability and scalability requirements.

APPN References

The following is a list of reference material, IBM patents and journal papers used during the SNAP APPN development.

Reference material

- SNA APPN Architecture Reference (SC30-3422-3)
- SNA Formats (GA27-3136)
- SNA Management Services Reference (SC30-3346-5)
- SNA LU6.2 Reference: Peer Protocols (SC31-6808)
- APPN Architecture and Product Implementations Tutorial (GG24-3669)

Patents

- Automatic Feedback of Network Topology Data (4,825,206)
- Method of Maintaining a Topology Database (4,827,411)
- Locating Resources in Computer Networks (4,914,571)
- Control Point Session Synchronisation in a network (4,926,414)
- Method of Reducing the Amount of Information Included in Topology Database Update Messages in a Data Communications Network (5,101,348)
- Adaptive Session-Level Pacing (4,736,369)
- Network Interconnection Without Integration (4,677,532)
- Automatic Update of Topology in a Hybrid Network (4,644,532)
- Method of Controlling Limited Resource Sessions in a Data Communications Network (4,972,437)
- Method of Establishing Transmission Group Numbers for Network Links (4,954,821)

Journal Papers

- A Perspective on Advanced Peer-to-Peer Networking (Green et al), IBM Systems Journal, Vol 26, no 4 (1987)
- Implementing System/36 Advanced Peer-to-Peer
- Networking (Sultan et al), IBM Systems Journal, Vol 26, no 4 (1987)
- SNA Networks of Small Systems (Baratz et al), IEEE Journal on Selected Areas in Communications, Vol sac-3, no 3, May 1985
- Subtle Design Issues in the Implementation of Distributed Dynamic Routing Algorithms (Jaffe et al), Computer Networks and ISDN Systems, Vol 12, no 3 (1986)
- The New Routing Algorithm for the Arpanet (McQuillan et al), IEEE Transactions on Communications, Vol com-28, no 5, May 1980
- Distributed Network Protocols (Segall), IEEE Transactions on Information Theory, Vol it-29, no 1, January 1983

Basically, we provide a thin, well defined mapping or isolation layer to insulate the core protocol code from the operating system. We call this layer the N-BASE, and it is all you need to modify when porting the code to a new environment. (See sidebar for an architectural overview of the SNAP APPN product)

The N-BASE forms only about 5% of the code for the overall product, thus leaving 95% of the code absolutely unchanged in different environments.

The success of this approach has been demonstrated by some of the first OEMs porting N-BASE to their proprietary environments in a week.

The N-BASE also allows us to distribute APPN components over multiple processors (which may be of different types), while retaining efficient scheduling and inter-component communication. This has proved crucial in allowing us and our OEMs to match the spread of components to the particular target architecture.

For example, in high-end routers, we can place performance- critical components (such as Path Control and the Session Connectors) on one or more specialised line cards, while placing less critical components (such as Management Services and the Control Point) on a general purpose processor.

SNAP: What was the procedure for testing the product?

JP: You can look at the test process from several angles.

The first requirement is unit testing. Given the complexity of the product, we invested a very large effort at the Unit Test phase, doing an extremely thorough test of each component in isolation before bringing the components together. To some extent this is standard industry practice (or should be), but we achieved very close to 100% branch coverage during Unit Test.

This paid off in several ways. First, we encountered a remarkably low level of problems when we brought the components together and started testing the whole product. For example, all the components were only built together a few months before Interop Spring, but performed superbly at the hot staging and during the show.

The Unit Test suite also forms the basis of our standard regression test suite. This can be run automatically (e.g., overnight), and gives 80% branch coverage. Running the regression suite overnight every night has been a major factor in maintaining quality during the later stages of test.

Functional verification is another element of the test procedure. Once the components had been integrated, we were then testing the whole product functionality - with or without associated link drivers and/or test applications dependent on the test case.

For SNAP APPN V1.0, the Functional Verification test suite contained approximately 5000 test items - where each test item specifies a single function in the product that needs to be tested.

The Network Node subset of these test items will soon be publicly available as our contribution to the IBM APPN Network Node self-certification test package.

Overlapping with the later stages of Functional Verification test, we were also performing whole network tests, stress and performance tests, and interoperability tests.

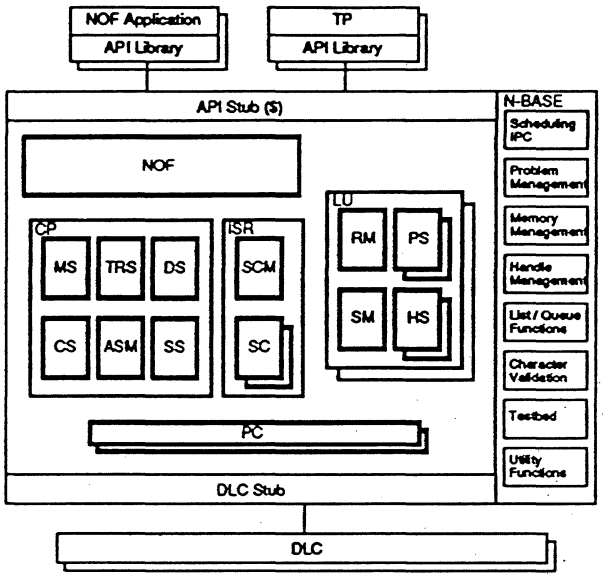
On the interopability front, we tested (and are continuing to test) against all current IBM APPN implementations, as well as gaining lots of useful (mostly positive) feedback from working with other vendor's LEN and End Node implementations during the APPC/APPN Showcase at Interop Spring.

We are also keen to get involved in the APPN conformance testing effort now that IBM have announced the availability of their conformance testing facility.

Alongside the day-to-day mechanics of testing, we also invested significantly in defect tracking

throughout the test phases. By monitoring defects per line of code, and defect arrival and removal rates, and by exercising careful judgment (i.e., using the numbers as a guide rather than an absolute measure), we had, at any moment during the process, a pretty good idea of how good the code was, and more importantly, of how good it was going to be.

SNAP APPN INTERNAL STRUCTURE



SNAP APPN is a portable product implementing the APPN Network Node and End Node protocols. SNAP APPN is inherently portable and is independent of any particular operating environment. To provide this independence, the product relies on a thin mapping layer called the N-BASE, which maps generic services required by SNAP APPN onto specific operating system functions.

SNAP APPN is structured according to the APPN Architecture Reference, and incorporates all the components (such as NOF and PC) and component groupings (such as Control Point, Logical Init and Intermediate Session Routing) as documented. Within each component the code is also structured to conform closely to the APPN Architecture Reference definition.

The N-BASE provides services such as scheduling, inter- process communication and memory management to SNAP APPN. By modifying the N-BASE implementation, it is possible to run SNAP APPN efficiently in many different types of environments.

- The N-BASE can either be single threaded, for simple, single process architectures, or can be multi-threaed to support multi-processor machines.

- Different SNAP APPN components can reside in the same or a different N-BASE. This allows different components to split across different processors or even machines. For example, in machines with a main processor, and a subsidiary communications card, it is possible to put the main data forwarding components -Path Control and the Session Connector - on the communications card, with the larger control components in the Control Point located on the main processor.

- The N-BASE can easily support different types of memory partitioning, from simple flat memory models, to architectures which partition different memory for data packets, stacks and control blocks and configuration and start-up information.

The N-BASE provides the following services.

- Scheduling and Inter Process Communication.
- Problem Management (for example error and audit logging, and trace).
- Memory Management.
- Handle Management.
- List and Queue functions.
- Character Validation (for example checking EBCDIC fields forvalidity)
- Testbed
- Various utility functions.

SNAP: What types of tools did you build to aid in development?

JP: While some of the tools were standard ones, such as the Microsoft Source Profiler coverage tool, we also invested a lot of effort in building specific test tools. For example, we developed a Unit Test tool which allows us to mimic message flows between components, such as between SS and TRS. This tool also allowed us to simulate system stress problems such as memory shortages during Unit Test.

Without this tool, we could not have reached 100% branch coverage during Unit Test, and, more importantly, would have been debugging real time system stress problems in live systems.

A high proportion of the Unit Test scripts were generated automatically from message definitions (in C) used directly by the code.

We also developed a whole range of specific test applications, most of which have rather jokey names. For example, "Stuffer" generates very large volumes of data to multiple nodes in a network, and is useful for testing buffer resource shortages, pacing, etc.

"Mole" is the APPN equivalent of the internet worm. It propagates itself through an APPN network, by monitoring the network topology. We'll makes sure it stays within the building!

"Random" generates random message sequences and message data. It is APPN protocol-aware and is biased towards generating semi-sensible sequences, which it sends to "Bounce" which reflects it back again.

"Monitor" dynamically displays the network topology using OS/2 PM graphics - and was a big hit when we showed it at Interop Spring. Unfortunately, a few people assumed we were marketing an APPN network topology monitor!

The product also contains a high proportion of diagnostic code, which can be switched on to aid problem resolution. The raw output from the embedded diagnostics can be fed (in real-time) through a filter/decoder which then presents the internal and external message flows in the architecture specification message flow format.

This was also a big hit at Interop Spring where it helped diagnose problems in other APPN implementations, including a bad outbreak of "TDU Wars" The formatted message flows fooled more than a few people to believe that we were simply listing the architecture specification, rather than tracing and displaying the internal SNAP APPN message flows in real-time.

The bottom line is that, while there were moments when we worried about the level of effort going into tools rather than product code, it has paid off many times over.

SNAP: How big a development has this been?

JP: Huge! This is the largest SNA development that Data Connection has undertaken in eleven years of writing portable SNA code. Overall we have invested tens of man-years.

The timetable for the project was:

2Q	1991	Feasibility Study
3Q	1991	Development Project Start
3Q	1992	First Integration of all components
March	1993	First demonstration at Spring Interop
June	1993	SNAP APPN V1.0 Release

SNAP APPN now comprises 80,000 lines of code, with another 5000 for the sample N-BASE provided with the product. In addition, there are over 120,000 lines of code in our regression suites.

As well as the obvious technical issues, the sheer size of the project posed major project management challenges, particularly in meeting the timescales while maintaining quality.

The bottom line is that we have proved that it is possible to develop an APPN Network Node.

However, it is very complex, and a massive investment, even for us with hundreds of man-years of SNA experience behind us.

SNAP: Are there unique features that you've added to your APPN product?

PMcC: Yes. They include rich function. we believe our implementation has the highest functionality of any APPN source code offering.

Portability and scalability which makes SNAP APPN equally applicable on PCs, workstation and routers. Superb embedded diagnostics and off-line test and integration aids are also value-added features. Quality which can be measured in terms of reliability, high performance and low occupancy is also important to our customers.

SNAP: Do you have any SNAP APPN licensees signed up yet?

PMcC: Yes. Can I tell you who they are? Unfortunately, No. At the moment we can't name our signed up OEMs, but in general we will be able to when those OEMs announce products based on our APPN product - which, for some OEMs may be reasonably soon. We'll certainly let SNA Perspective know when we can name names!

SNA: Network Node support has been announced by several router vendors. What other categories of products are likely candidates for Network Node support?

PMcC: The interesting thing here is just how many different opportunities there are. As well as the router market-place, we have encountered significant interest from workstation vendors where APPN is a developing requirement in selling into large corporate networks, and where the availability of the full range of application programming interfaces on both Network Node and End Node is very attractive

Another category which is very interesting is client/server system vendors where the ability to run one or more APPN Network Node servers plus multiple APPN End Node clients on a departmental LAN, with the clients exploiting the Virtual Routing Node capability for sessions within the LAN.

SNAP When do you estimate that APPN will be widely available from non-IBM vendors?

PMcC: Obviously at the moment, a range of vendors are offering LEN products that can inter-operate with APPN - and many such vendors demonstrated product at Interop Spring. It may be that some of those vendors will upgrade their products to support APPN End Node over the next year.

We would expect products based on SNAP APPN, whether configured as End Nodes or Network Nodes or both, to be available late this year, or (more commonly) in the first half of 1994. But to get an accurate prediction, you should really ask the vendors!

SNA: Are there any particular issues that buyers should be aware of when acquiring APPN compatible products?

PMcC: If you consider OEMs looking to buy APPN, they tend to major on speed to market which mostly comes down to portability, where our N-BASE approach gives us a terrific story to tell.

Quality is always a key issue. Quality of code, people and organisation, and support, which is the heartland of our business.

If you then consider end-users, they care about the OEM factors (because such factors translate into end-user quality and reliability), but they also care about interoperability because the product will almost certainly have to interoperate with existing IBM mainframes and AS/400s, and may have to work in a multi-vendor APPN environment

Application programming interfaces are very important to end-users. Interfaces such as LU6.2 APPC and CPI-C are required to support existing investments in transaction programs.

SNAP: What about APPN futures, such as Dependent LU Requester/Server (DLUR/S) and High Performance Routing (HPR)?

PMcC: Speaking about end-user requirements, the overriding need is for true LU 0,1,2 and 3 support within APPN.

Our APPN product (in Version 1.1) supports LUs 0,1,2 and 3 to a directly connected host, from which you can use traditional cross-domain to access other hosts. This means that many users can run existing applications and emulators over SNAP APPN.

But the real end-user requirement is to set up LU 0,1,2 and 3 sessions across a distributed APPN environment, where (potentially) the host application is on the only mainframe which is on the other side of the network. To do this requires Dependent LU Requester (DLUR) support, which we are incorporating into the next version of SNAP APPN.

High Performance Routing (HPR) will provide benefits in new technology high-speed APPN networks, such as connectionless routing, improved performance and congestion control, and reduced in-flight data storage. The final HPR specifications are not available yet, but we intend to include HPR support in SNAP APPN next year.

SNAP: Tell us about your relationship with IBM.

PMcC: Over the years, IBM has been, and continues to be, one of Data Connection's largest customers.

You probably know that, while we developed our APPN product from scratch, it does use algorithms covered by IBM patents - it has to conform to the architecture. This means that we had to license IBM's APPN intellectual property via the APPN Technology Offering Agreement (which covers APPN patents, trademarks and copyrights). That's fine and business as normal.

We have an ongoing technical relationship with the IBM SNA architecture group through feedback to the APPN architects in Raleigh, our participation in the Interop Spring APPC/APPN Showcase, and our participation in the excellent APPC/APPN Implementor's Workshops.

Another aspect of our relationship with IBM is a competitive one - as SNAP APPN competes directly with IBM's APPN OEM source code package.

Overall, we like to think we have an excellent relationship even when we're competing with IBM.

The bottom line is that IBM have actively supported what we are doing with APPN - as we both have a common goal of establishing APPN as a de facto industry standard.

SNAP: So, what would be your overall conclusion on developing an APPN Network Node?

PMcC: Well, it has been far from easy. But we've shown that it is possible to develop an APPN Network Node from scratch, and produce a product that is eminently suitable for OEMs and end-users.

We've also established a positive and cooperative relationship with IBM, that should help establish APPN as the wide-area networking standard for the future.

SNAP: Thanks very much for taking the time to give our readers your insights into the APPN product development process.

Architect's Corner

Drums in the Distance

by Dr. John R. Pickens

Do I hear drums in the distance? Is IBM preparing for (protocol) battle? Arethe preparations complete? What fronts might be opened? Will the campaign belimited or all-out?

I believe IBM is gearing up for major battle. Battle for mindshare inmultiprotocol networking architectures, performance, and products. Significant preparations have been completed; others are underway. Rumors of battle abound. Fronts can be identified. But even before the battle begins, the terms of peaceare already in place.

Battle Preparations

First, IBM has had to rid itself of old armament - subarea SNA and SDLC. While subarea SNA is long from gone, APPN has certainly been positioned to take its place. With shipment of APPN in VTAM, the stage is nearly set for the mothballing of subarea (one last detail - delivery of dependent LU server/requester).

SDLC is likewise ready to be scrapped. Much ado has been made recently by IBM about the substitution of frame relay for SDLC. Since frame relay can be used in a point-to-point peer mode (like null modems), IBM is enabling the use of frame relay both for access to frame relay WANs, and for point-to-point links between SNA nodes, effectively obsoleting SDLC as an SNA link type. In most cases frame relay can be implemented on the same adapters that today are offering SDLC.

Second, multi-vendor alliance building is now well underway. One visible evidence is the APPN Implementor's Workshop (AIW) multivendor standards forum. Of particular note is the makeup and politics (and resolution thereof) of the Data Link Switching (DLS) special interest group.

To wit.

In mid June this year the DLS working group was shaping up as a religious battle ground between IBM and Vendor_X (as in previous columns I won't name the vendor). One candidate each for DLS chair from IBM and Vendor_X was offered. A third neutral candidate was offered. By consensus the multi-vendor audience chose neither IBM nor its arch-rival Vendor_X; rather they chose the neutral candidate. Now work on standardization of DLS can begin. (It has begun - seven sub working groups were formed by the new chair.) The fragile multi-vendor alliance is in place.

Rumors of Battle

Two interesting rumors are floating regarding IBM's preparations.

First regarding TCP/IP. IBM's TCP/IP implementations have not been "me too" products. Their performance in many cases exceeds the performance of comparable products from other TCP/IP vendors. IBM is poised to battle in the TCP/IP environment (the universal standard).

Second, IBM's APPC implementations are featuring measured performance second to none. Even TCP/IP. By significant margins.

Just rumors.

The Fronts

I believe the coming battles will be fought on two fronts. High performance routing and multi-protocol routing.

The stage for the high performance battle has already been set. APPC in certain configurations outperforms other protocols, e.g. TCP (interesting to see IBM battle against itself on this one). But more significant is the next weapon, High Performance Routing. Analytic studies (not verified by real-world tests yet) show APPN/HPR (actually RTP/ANR) to be 3-10 times faster than TCP, and to

exhibit fairer graceful degradation characteristics in environments with complex LAN/WAN topologies. Also, a new high speed ATM-based switching architecture is about to be introduced (once called gigabit-APPN). This switching architecture will push the high performance envelope outward to new limits.

The stage for the multiprotocol battle is also interesting. The coming high speed ATM architecture will be protocol independent. Interfaces will be defined for ATM UNI, frame relay, token ring, ethernet, bridging, routing IP, routing SNA, routing IPX, etc. But even prior to introduction of the ATM architecture, current HPR has interesting properties which might be exploited by IBM generals. HPR is being introduced as an APPN feature, but it actually contains an architected

subcomponent, RTP/ANR, that is protocol independent. There is no reason why RTP couldn't be used as a sublayer of other protocol stacks - TCP/IP, Novell, even IBM's older dependent LU protocols (LU0, 1, 2, and 3).

Peace

Despite all these battle preparations, the terms for peace are already on the table — Multi-Protocol Transport Networking (MPTN). While vendors may battle for dominance at the transport layer, users can enjoy the benefits of protocol independence at the API layer (Sockets over SNA, CPI-C over TCP/IP). Whether motivated by transport constraints, e.g., single protocol SNA-IP-IPX backbones, or by performance requirements, e.g., stack_A faster than stack_B, customers (and NOS vendors) have a way to put themselves above the battles below.

Yes, IBM is preparing for battle. But, in my opinion, the battle will benefit customers and vendors through the delivery of higher performance, more protocol flexibility, and higher quality products. The drums already seem nearer.