A comparison of network architectures—
The ARPANET and SNA

by GILBERT FALK
Bolt Beranek and Newman Inc.
Cambridge, Massachusetts

INTRODUCTION

The ARPANET is a packet-switched data network which facilitates resource sharing and supports distributed data processing. The ARPANET technology and its descendants have had a significant impact on the direction of data communications over the past eight years. More recently, IBM announced Systems Network Architecture (SNA) which is targeted at satisfying a similar set of requirements. SNA will undoubtedly also have a major impact on the future of data communications. This paper presents a comparison of these two network concepts by discussing architectural and protocol similarities and differences employed to support process-to-process level communication. The goal is both to provide useful information and to stimulate interest in the analysis of alternate architectural approaches.

The presentation that follows identifies key properties and functions of data networks in general and discusses briefly how they are performed within each of the two network architectures. An elementary knowledge of both the ARPANET and SNA is assumed. Information describing the design and implementation of the ARPANET can be found in References 1-3. A presentation of the concepts embodied within SNA can be found in References 4-8.

Architectural similarities in these two designs result from fundamental similarities of objective. Both network architectures have as primary goals the abilities to support distributed processing and to facilitate resource sharing. In the case of the ARPANET the motivation for providing these capabilities was the need to share large hardware and software resources and to establish a facility to support computer system research. The development of SNA by IBM, on the other hand, was largely driven by the economic need to take advantage of equipment based on LSI technology. This technology provides the capability to locate many previously centralized functions in terminals, device controllers, communication front-ends, etc. Both designs have attempted to provide a single general-purpose interprocess communication facility upon which application level communication rests. A central theme in the development of both architectures is the separation of the data processing and data communication functions.

Perhaps more important than the similarities of objective are a number of fundamental differences between the ARPANET and SNA environments. These environmental differences provide a partial explanation of the individual architectural differences that are present in the two designs. The ARPANET was developed for operation within a research environment. Within such an environment it is possible to experiment a bit in the design and operation of an evolving network. In a commercial environment, on the other hand, conservatism dictates more cautious development. While SNA has been developed as a new network architecture, it nevertheless has the requirement of accommodating a large existing base of installed software systems (e.g., IMS, CICS). ARPANET development, on the other hand, was not constrained in this way. ARPANET host software was developed for the most part independent of any existing application software within the hosts. SNA was designed to provide a networking capability for equipment from a single vendor. The ARPANET, on the other hand, was developed from the start to provide a networking capability for subscriber equipment from many different vendors. Finally, SNA and the ARPANET differ in the nature of the two network concepts. SNA is an implementation independent architecture. This architecture is realized differently within different IBM computer and communication products. The ARPANET, on the other hand, is an evolving implementation of the general concept of packet-switching.

COMPARISON OF NETWORK CONCEPTS AND MECHANISMS

The following subsections provide a side-by-side comparison of a number of aspects of the ARPANET and SNA network architectures.

Physical network concept

Figure 1 illustrates typical physical network configurations within SNA and the ARPANET. Figure 1a illustrates four distinct types of physical units (PUs) that can exist within an SNA network: PU:T5, PU:T4, PU:T2, and PU:T1. Units of type PU:T5 are host computers (370s). Communication controllers (e.g., 3704s or 3705s) are described as
units of type PU.T4. Cluster controllers (e.g., 3601s) are described as units of type PU.T2. Finally, terminals (e.g., 3270s) are units of type PU.T1. A basic concept of SNA is that there is only a single SNA node description and PU types are obtained as subsets of that single description. In this sense, PU.T1, PU.T2, and PU.T4 are subsets of PU.T5. The original announcement of SNA permitted configurations consisting of only a single host as shown by the solid lines. A more recent announcement from IBM has described expanded SNA implementations consisting of multi-host configurations (indicated by the dotted lines).

Figure 1b illustrates a typical physical network arrangement that may exist in the ARPANET. The two types of network nodes are referred to as IMPs and TIPs. The IMPs provide a store-and-forward switching function in addition to providing an interface for connecting host computers (H). The TIP is an IMP with an additional front-end concentrator function providing network access for unintelligent terminals (T). Arbitrary topological interconnections of IMPs and TIPs are permitted within the ARPANET design. A Network Control Center (NCC) provides a central facility for network monitoring, diagnosis and maintenance.

Logical network concept

Figure 2 contrasts the two network architectures with respect to their basic communication facilities. Within SNA the fundamental communication system is referred to as the transmission subsystem. The transmission subsystem provides process-to-process level communication between entities referred to as Network Addressable Units (NAUs). Three types of NAUs exist. The Logical Unit (LU) provides an interface port for SNA end-users. End users may be either human operators at a terminal or application programs. Physical Units (PUs) comprise a second set of network addressable units. A distinct PU is associated with each shared communications resource (host, cluster controller, etc.) in the network. The PU is the entity to which communication is directed when one wants to communicate with the associated physical device. Finally, each SNA network has one (more in a multi-host network) System Services Control Point (SSCP). The SSCP provides central monitoring, coordination, and control of its domain in the SNA network.

In the case of the ARPANET the IMP subnetwork consisting of interconnected IMPs (and TIPs) provides the basic communication facility. At this level the network hosts are the addressable units. It is important to point out that it is
the combination of the IMP subnetwork and the Network Control Program (NCP) software within the hosts which provides the process-to-process communication analogous to the SNA transmission subsystem.

Although there are not any logical entities within the ARPANET which correspond directly to the three types of NAU under SNA, certain analogies can be drawn between the two designs. Regular host computers (including the concentrator pseudo-host associated with each TIP) are the most common type of host subscribers interfaced to the IMP subnetwork. The internal processes supported by these regular hosts correspond to the logical units within SNA. Certain fake hosts such as PARAMETER CHANGE and DEBUG, on the other hand, exist within each of the IMPs and provide a means of effecting changes in network operation. They are, therefore, analogous to the physical units within the SNA design. Finally, the NCC facility can be viewed as analogous to the SSCP in the sense that it is the single network-wide coordination entity within the ARPANET. A distinction between the SSCP and the NCC is the importance of the former for each conversation carried out over the network. In particular, it is directly involved in the initiation of LU-LU sessions as described in a later section.

Figure 3 illustrates the types of communication between addressable units under the two network concepts. A line in this figure indicates communication between the entities corresponding to its end points. We have assumed in the case of SNA that only a single SSCP exists. If multiple SSCPs exist, then these entities will clearly require communication between one another in order to synchronize their activities. In addition, in the case of the ARPANET, we have neglected end-user to fake host communication which, while possible, is only incidental to the primary communication capabilities supported by the ARPANET.

**Key architectural characteristics**

Figure 4 summarizes a number of selected architectural characteristics in the case of the two network designs. The first entry in the table indicates that both SNA and ARPANET are based on block or packet-switching technology. In the case of the ARPANET, a packet has a fixed maximum number of bits. In the case of SNA, the maximum packet size is not only implementation-dependent but can also vary from node to node within the network. Under both designs packets are passed through the network according to a routing algorithm. SNA has adopted a routing approach based on static tables within the nodes. The tables are centrally maintained by the SSCP. The ARPANET, on the other hand, has implemented a distributed adaptive routing algorithm in an attempt to dynamically adapt the flow of network traffic to changes in network topology and circuit loading. The routing algorithm is carried out periodically at each of the network nodes based on information that each network node receives from its neighbors describing the best path to each destination.

The processes of segmenting and reassembly provide the means for translating between data units (SNA PIUs and ARPANET messages) presented to the network and packets switched through the network. Under SNA, segmenting and reassembly can be performed at each node along the source-to-destination path. Within the ARPANET, packetizing is done only at the source IMP and reassembly is performed at the destination IMP after all of the packets of the message have arrived. Blocking and deblocking are associated with the combining of logically unrelated packets into a single block (superpacket) for efficiency of transmission over in-
individual network links. This concept exists only within SNA where it is used for efficiency of transmission over 370 channels. As indicated by the last entry in Figure 4, messages (BIUs) and message fragments are kept in order over their source-to-destination path within SNA, whereas ARPANET packets are routed independently and only reordered at the destination node. The ARPANET approach has potential benefits with respect to end-to-end delay and throughput but requires careful design of the control and buffering procedures in order to avoid the possibility of network lockups.

The concepts of packetizing and blocking within the two network architectures are illustrated in more detail in Figure 5. This figure illustrates the data envelopes associated with transmitting an application buffer of data from one process to another via the network. In the case of SNA the application buffer is first split into a number of Basic Information Units (BIUs), each consisting of a Request Header (RH) and a Request Unit (RU). A set of related request units is referred to as a BIU chain. (Note: A BIU chain can also be assembled from multiple application buffers.) The BIU chain is treated as a unit with respect to acknowledgment and recovery procedures. The Request Headers contain only control information to support process-to-process level communication. Application data is contained within the Request-Unit portion of the BIU. Before a Request Unit can be transmitted, a Transmission Header (TH) is attached as shown. The Transmission Header contains end-to-end control information such as destination address, sequence number, etc. The combination of Transmission Header, Request Header, and Request Unit is referred to as a Path Information Unit or PIU. If required, a BIU can be segmented into BIU segments (each of which then becomes a PIU), as shown, prior to transmission over any network link. The PIU with Data Link Control (DLC) appended at the beginning and end is ready for transmission over any network link. The PIU with Data Link Control (DLC) appended at the beginning and end is ready for transmission over any network link and is referred to as a Basic Link Unit (BLU). As illustrated in the last two lines of Figure 5a, independent PIUs can be combined or blocked at a node as described previously for purposes of transmission efficiency.

![SNA and ARPANET Diagrams](From the collection of the Computer History Museum (www.computerhistory.org))
The situation is somewhat simpler in the case of the ARPANET. Here an application buffer is split by the host into an arbitrary number of network messages, each message less than 8000 bits in length. Host-to-host protocol control information (H/JH) is added and interpreted by the network control programs in each host in order to support process-to-process level communication. Each message is then divided into no more than eight packets of approximately 1000 bits in length. These packets are routed independently through the IMP subnetwork. Host-to-IMP control information is provided by a leader attached to each message transmitted by a host. This leader is used to create the control information contained in the header of each of the data packets.

**Link control procedures**

Two types of link control are supported under SNA, the 370 channel protocol and Synchronous Data Link Control (SDLC). Analogous link control procedures supported in the ARPANET are the standard host-to-IMP interface, the very distant host interface, and the IMP-to-IMP protocol. The most appropriate of these protocols for purposes of comparison are SDLC and the ARPANET IMP-to-IMP protocol.

Although SDLC and the ARPANET IMP-to-IMP protocol both support communication over transmission circuits, these two protocols differ along several dimensions. SDLC implements data transparency by bit insertion in the data stream. Five sequential one bits in a row are always followed by a zero in the data stream to distinguish such patterns from framing patterns (which involve a sequence of six sequential ones). Transparency in the ARPANET protocol is based on doubling of DLE characters. This procedure is similar to the one implemented under IBM's older link protocol, BSC, which was the basis of the IMP-to-IMP protocol frame structure.

Another difference between SDLC and the protocol used on the links between IMPs is the diversity of the data links supported. Under SDLC a variety of data link configurations are supported, including point-to-point, multi-point, loop, etc. The ARPANET IMP-to-IMP protocol, on the other hand, is only designed for dedicated full-duplex point-to-point links. A third distinction between the two protocols is in the fundamental relation between the two ends of the link. Under SDLC each link has a primary and a secondary end. The primary end is typically associated with the station having superior processing capabilities. Under the ARPANET protocol both stations are assumed to have identical capabilities and the two ends of the link are symmetric. A minor difference between the two protocols is the length of the CRC checksum added for error detection. This checksum is 16 bits in the case of SDLC and 24 bits in the case of the ARPANET.

Perhaps the most interesting contrast between the two link control procedures is illustrated in Figure 6. As indicated, under SDLC the link control procedure implements what is fundamentally a single logical channel capable of supporting eight outstanding (unacknowledged) frames. These frames are numbered sequentially in order to enable the transmitting end to associate acknowledgments with data units. The analogous arrangement between IMPs is a sequence of eight logical channels each of which can support only one outstanding frame. In a sense these alternate approaches are identical: if all frames are acknowledged properly, both approaches support the same throughput over the link. If a frame is not acknowledged, however, the SDLC procedure requires retransmission of everything following the unacknowledged frame, whereas the IMP-to-IMP protocol conserves channel bandwidth by requiring retransmission of only the unacknowledged data. A more fundamental difference is a result of the dependency between frames which for SDLC is inherent in the sequential numbering procedure. In particular, failing to acknowledge any single frame completely blocks the flow of data over the link within eight frames of the frame which was unacknowledged. In the case of the ARPANET protocol, on the other hand, failing to acknowledge an individual frame transmitted over any logical channel does not impact the transmission of new frames over the remaining seven logical channels.

The ARPANET IMPs take advantage of this independence by permitting packet discard at the link control level. If a packet cannot be handled due to local congestion, the receiving IMP fails to acknowledge the packet as many times as necessary, being assured that its neighbor will continuously retransmit the data. Under SDLC this type of flow control cannot be carried out at the link level and an additional higher level mechanism must be incorporated for this purpose.
Error control

There are at least two types of errors which are associated with the transmission of data through packet switched networks. Each type of error is typically handled by a distinct mechanism. Corrupted data due to burst noise on the communications channels or errors within the communications hardware is detected by appending checksums to transmitted data. Lost, duplicated, and out of order data is typically detected by the addition of sequence number information to each frame transmitted. These mechanisms can be applied both over individual links and over end-to-end paths. Figure 7 illustrates the mechanisms for end-to-end error control which exist in the ARPANET and SNA designs. As shown in Figure 7a, there is a single end-to-end sequence number attached to each request unit transmitted by a process in an SNA network. This 16-bit sequence number is generated by the transmission control (TC) function of the source node. It is passed back to the source process in order that acknowledgment and error recovery information associated with that request unit can be identified. Errors due to corruption of the data are handled by checksums on the links (associated with SDLC).

In the ARPANET environment, sequencing is done on a host-to-host basis. Internal sequence numbers within the IMP subnetwork guarantee that messages are delivered to the destination host in the same order that they were submitted by the source host. A 12-bit sequence number ("message I.D.") field which is passed between the host and its IMP can be used for identification of responses (e.g., RFNMs) for multiple outstanding messages. It is also passed through to the destination host so that the destination can detect out of order errors due to subnetwork failures. As in the case of SNA, corruption of data on the lines is handled by link checksums. An additional end-to-end checksum is implemented in software in order to detect corruption of data originating within the nodes. This software checksum is computed on each packet by the IMP as it comes in from the source host and is checked at each node along the end-to-end path.

Flow control

Figure 8 illustrates the type of process-to-process flow control implemented in the two network architectures. Under SNA, two logical units are always connected by two logical full-duplex flows, a normal flow and an expedited (priority) flow. The expedited flow is typically used for the transmission of process-to-process control messages. On the normal flow a form of flow control termed pacing is applied. Pacing messages sent from the receiver to the transmitter take the form of "Send N More RUs". Such pacing requests can be sent after fewer than N RUs have been received in order to keep the end-to-end logical channel full in support of high throughput applications.

In the ARPANET process-to-process communication typically is carried out over a single pair of simplex channels. Control messages between all processes on two communicating hosts share a common control connection over which messages are handled with priority. Flow control is applied in both directions and allocations take the form of "Allocate M Messages and B More Bits." Both M and B can vary from one allocate message to the next.

Both of these flow control schemes can be described as incremental. This means that they both take the form of the receiver telling the transmitter that a certain number "more" data units are allowed. It has been argued that such incremental flow control schemes have the potential for loss of synchrony if allocated messages are lost within the communications system and no notification of this loss is provided. Windowing schemes tied more tightly to end-to-end sequence numbers have been proposed recently. Such windowing schemes have the property that they are self-synchronizing.
Addressing

A major difference between process-to-process addressing in the SNA and ARPANET architectures is the use of network names for NAUs under SNA. Names or logical addresses have the advantage of facilitating transparent movement of services between physical facilities by separating the concept of network name from the location of the named facility. Mapping between names and addresses under SNA is handled by the System Services Control Point during session initiation as described in the next section. Under the ARPANET design end-user identification is tied to physical location.

The formats of the network addresses themselves provide another interesting contrast. SNA network addresses consist of 16 bits. The 16 bits are split between a sub-area address (used for store-and-forward routing) and an element within the sub-area (used for local routing). ARPANET network addresses consist of a 16-bit IMP number (used for store-and-forward routing), an 8-bit host number (used for local routing), and a 32-bit socket number (used for process identification). In order to avoid having to transmit a pair of 32-bit socket identifications with each message transmitted between processes, an 8-bit shorthand called the "link I.D." is used in the ARPANET. This puts a (realistic) limit on the number of simultaneous conversations between any pair of hosts.

Session initiation

Figure 9 illustrates the procedures involved in setting up a logical session (connection) between two end users under the SNA and ARPANET architectures. The key point to note in this figure is that session initiation under SNA involves the SSCP as well as the two communicating processes, whereas session initiation within the ARPANET involves only the two communicating processes themselves (and their NCPs). Figure 9a illustrates the sequence of exchanges assuming a primary LU wishes to establish a connection to a secondary LU. The primary LU first sends an INITIATE command to the SSCP. The INITIATE indicates the type of session requested, the name of the LU with which the primary wishes to speak, and password information for authenticating the request from the primary. If the establishment of the requested connection is permitted by the SSCP, it will respond to the primary by the transmission of a CONTROL INITIATE command. The CONTROL INITIATE command contains a "bind image." The bind image includes all of the parameters which are subsequently sent in a BIND command from the primary to the secondary LU. The parameters in the BIND command specify the type of session being established in detail. Assuming that the secondary accepts the BIND request, the primary notifies the SSCP via a SESSION STARTED message. Session authorization checks carried out by the SSCP are associated with the allocation of reusable resources as well as the name-to-address map.

Communication between a user and a server process in the case of the ARPANET is established in one of two ways. For a service process which handles only one user process at a time, a simple pair of connection requests is exchanged. For multi-user server processes supporting multiple simultaneous conversations, a standard procedure referred to as the Initial Connection Protocol (ICP) is implemented for session initiation. The set of control messages which comprise the ICP is illustrated in Figure 9b. The complexity of the required exchange is a result of the fact that connections are fundamentally simplex under the current ARPANET protocol standards (two are typically used for process-to-process communication), and the fact that two connections cannot have either of their ends in common. The ICP works by having the server process continually listening for connections from users on a "well-known socket" (L in the figure). A user wishing to obtain the service connects to this well-known socket and is subsequently switched by the server from the well-known socket to another pair of sockets which the server chooses. The server then continues to listen on socket L for connections from other remote users. In contrast to the tailoring of process-to-process communication which under SNA is carried out as the result of the BIND exchange, tailoring of the session between user and server processes is a higher level protocol function. The ARPANET supports only a single type of process-to-process communication.

Word length mismatch

The issue of word length mismatch was of great concern to the designers of the ARPANET. This is due to the fact that the IMPs are 16-bit machines, and the hosts supported are heterogeneous. Attached host systems include IBM 370s (32-bit word length), PDP-10s (36-bit word length), PDP-15
(18-bit word length), minicomputers (16-bit word length), and CDC6600s (60-bit word length). Word length mismatch is not so important an issue under SNA because equipment from only a single vendor is involved. In addition, most IBM mainframes and peripherals have byte handling facilities.

In order to neutralize the problems of word length mismatch, the ARPANET designers provided two capabilities. The first enables the transmission of messages containing arbitrary numbers of bits (less than the maximum message size) through the network. The second facilitates the combining of multiple successive messages into a single application buffer without excessive bit shifting. The first goal was achieved by having the host/IMP interfaces add and interpret hardware padding sequences. The IMP interface at the source adds padding bits at the end of each message it receives from its host. The padding consists of a 1 followed by as many zeros as are required to fill out an IMP word. The host interface at the destination adds as many additional zero padding bits to the message as are required to fill out the word length of the receiving host. To avoid excessive bit shifting, the host-to-host protocol is defined to have a "message header." The length of this header in conjunction with the leader associated with each message insures that the text portion of the message starts in a "good place" for most machine word lengths (i.e., on a word boundary). In addition, the host-to-host protocol specifies a "connection byte size" which can be used to guarantee that the message text also ends in a "good place."

SUMMARY AND CONCLUSION

The previous sections have detailed and contrasted the mechanisms supporting a variety of network functions in the ARPANET and in SNA. An important similarity of approach is the hierarchical layering of functionality and protocols which is characteristic of both designs. Communication in both cases is characteristic of both designs. Communication in both cases is carried out between pairs of peer functional layers in the two communicating entities. This approach has advantages with respect to ease of comprehension, maintenance, debugging, and expansion. Both designs are fundamentally packet switching technology and both designs involve some amount of physical separation as well as logical separation of data communication and data processing.

More interesting than the conceptual similarities are the key architectural differences. These differences are more evident in SNA implementations than in the definition of the architecture itself. Centralization of network functionality is one such critical difference. Perhaps due to its ancestry, centralization is more clearly in evidence in the SNA design. A key example of this is the SSCP involvement in session initiation in single SSCP implementations. Symmetry of peer elements with respect to control is a second important distinction. Two communicating entities in an SNA environment often communicate as a primary to a secondary. One logical (physical) end of the communication path (link) is endowed with more control capability than the other. In the ARPANET, on the other hand, communication is typically symmetric (primary to primary). A third important distinction between the two designs is the nature of the routing strategy. The distributed adaptive approach developed for the ARPANET is one of the key features of that design. While such routing strategies are not necessarily excluded from implementation under SNA, there does not appear to be a strong leaning in this direction on the part of the SNA designers. Finally, we point to general complexity as an area where the two architectures differ. The architecture of SNA is capable of supporting both half and full duplex links, switched and non-switched links, multipoint links, and loop configurations. SNA session protocols are tailored at the transmission subsystem level for efficiency in contrast to the single process level communication protocol within the ARPANET. In addition, a variety of network elements with different physical capabilities are accounted for under SNA, whereas only a single type of network element exists in the ARPANET design.

The previous sections have attempted to present a side-by-side comparison of the ARPANET and SNA architectures. The goal of this comparison was not only to provide useful information but also to raise questions and spark some interest in comparing the relative merits of different architectural approaches. Moreover, by carefully examining two points in the space of possible architectural implementations, it is possible to gain a better understanding of additional architectural alternatives.

ACKNOWLEDGMENTS

The author would like to thank John Davidson, Alex McKenzie, Ira Richer and Eric Rosen of BBN and James P. Gray of IBM for their critical review of this paper, and Jan Mahoney and Julie Moore for their assistance in its preparation.

REFERENCES

8. IBM Corporation, "Introduction to Advanced Communications Function," Publication GC30-3033.