ABSTRACT

Computer networks provide the basic mechanism for accessing programs and data. Effective use of networks requires appropriate protocols to preserve the meaning of data being transmitted between heterogeneous systems. Currently existing protocols were developed to support the requirements of scientists and engineers. Information processing support, for example remote database access, requires substantially more sophisticated protocols. This paper describes a spectrum of data sharing protocols, identifies the need for a Structure Transport Protocol (STP), describes the nature of an STP, and discusses its utilization in the context of remote database access.

This work is a contribution of the National Bureau of Standards and is not subject to copyright. Partial funding for the preparation of this paper was provided by the U.S. Air Force Rome Air Development Center (RADC) under Contract No. F 30602-77-F-0068. Certain commercial products are identified in this paper in order to adequately specify the procedures being described. In no case does such identification imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the material identified is necessarily the best available for the purpose.

1. INTRODUCTION

Data sharing is a major application of computer communication networks. Currently existing protocols limit such sharing to the transmission of character files or binary strings [ARPAN 76]. This is inadequate for meeting the data sharing requirements for information processing (as opposed to scientific computing).

This paper: i) discusses alternative levels of data sharing, ii) establishes the need for a Structure Transport Protocol (STP), and iii) shows how this protocol fills a key requirement for supporting effective network access to remote Database Management Systems (DBMSs).

Our discussion of the general nature of Data Transfer Protocols (DTPs) also identifies the information support requirements essential to their operation. For an STP, these requirements are sufficiently complex to suggest the desirability of a Network Operating System (NOS) [KIMBS 78A] as an effective means of providing the required additional support mechanisms for maintaining, transmitting and transforming STP related data.

Providing an appropriate framework for discussing these issues requires an understanding of network protocols and their structure as developed in section two. Section three discusses different levels of data sharing and establishes the relationship between an STP protocol and other data transport protocols while section four structures its major components. Section five shows how this protocol can be used to support remote access to multiple, heterogeneous DBMSs and describes some additional required functions. Section six contains some concluding remarks.

2. PROTOCOLS

A protocol, as discussed in [CROCS 72], defines the format and relative timing of messages exchanged between two cooperating processes in a computer network. For the purposes of this paper, a protocol may be viewed as consisting of four basic components: i) information requirements for its operation, ii) commands for its utilization, iii) responses from its utilization, and iv) error messages. Protocol support requirements are also discussed.

U.S. Government work not protected by U.S. copyright.
A protocol may be used to transfer data, such as files, or to provide a means for implementing a common set of commands to be used across a collection of network accessible computers, e.g. a common network-wide job control language. This paper is concerned with Data Transfer Protocols (DTPs). The operation of such a protocol proceeds in four stages: a) establishing a connection between the user (or requestor) process in the source computer and the server process in the destination computer, b) initializing the connection once established, e.g. negotiating conventions such as data representations, c) transferring data or commands, and d) terminating the connection.

Protocols are required to support the exchange of data between processes. Since the user and server portions of the protocols may be produced by different vendors at different times, there is a substantial requirement that a protocol specification be unambiguous, correct, and self contained. Thus the interest in protocol specification, protocol validation to ensure that a given specification is logically correct, and protocol verification to ensure that a given implementation conforms to the specification.

The remainder of this section describes some of the protocols which have been developed within the Arpanet and establishes the basic concepts needed for our subsequent discussion of DTPs.

2.1 Some Current Protocols

The 'Arpanet Protocol Handbook' [ARPAN 76] lists a variety of protocols which have been developed for use within the Arpanet. Many of these protocols, e.g. the graphics protocol, are intended to support a particular application. Three are of widespread general interest: the host-host protocol, Telnet, and FTP.

The Arpanet host-host protocol is the essential building block for most of the other Arpanet protocols. Its function is to provide a reliable, virtually error free communication path between source and destination computer. This is accomplished through the utilization of sophisticated error detection and correction capabilities [KIMBS 75].

Perhaps the simplest way of supporting sharing of remotely accessible resources is to permit the terminal user of one system to access other, remote systems just as if the user were a local user of those systems. The Telnet protocol, provides such a mechanism.

Implementation of the Telnet protocol is based on a Virtual Terminal Protocol defining a network-wide set of terminal functions and character encodings (ASCII). The source computer (system to which the user is logged in) maps the functions and character encodings which it uses into the corresponding VTP functions and encodings. The destination computer (remote system being accessed by the user) maps from these VTP functions and encodings into those which it supports. This general approach of mapping from a local host representation to a common network representation and back to a local host representation is used in the other data transfer protocols discussed below.

The need to transmit files between systems is evident. This function is provided by the File Transfer Protocol (FTP). FTP, as currently implemented, only supports transmission of text (character string) or binary files. Although this limited functionality may have been sufficient for the needs of the scientists and engineers, it is insufficient for supporting the data processing needs of the organization.

FTP is also of interest in that it utilizes two different communication paths: a data transfer path constructed via the host-host protocol and a control path using the Telnet protocol. This approach permits the FTP protocol to utilize out of band, rather than in band, signalling. This is desirable to permit fast response to interrupts which, otherwise, would have to wait for the processing of all data transmitted prior to their issuance.

2.2 Data Transfer Protocols

The underlying objective of a DTP is preservation of meaning in transmitting data between systems. That is, a block of data within the source computer is to be transmitted to the destination computer and transformed, as appropriate, to ensure preservation of meaning.

Within a system, a block of data (data element, COBOL record, PL/1 or C structure, file) is represented as a binary string. This representation is termed the block buffer representation (BBR). For data consisting of a given structure and set of values, the BBR may be system, language and compiler dependent.

To avoid writing approximately \( N^2 \) translators for each possible combination of heterogeneous system architectures, when implementing a data transmission
protocol, an intermediate representation is used. Thus, the BBR at the source computer, say i, is transformed into a Common Network Representation (CNR) which is then transformed at the destination computer, say j, into the appropriate BBR. The chain of events is 

\[ \text{BBR}(i) \rightarrow \text{CNR} \rightarrow \text{BBR}(j) \]

The difficulty of building a DTP is driven by the complexity of the required translators. Their construction requires knowledge of: i) the structure of the block, ii) the data types of individual elements of the block, iii) the algorithm for relating the structure to the BBR and conversely, and iv) the BBR. The difficulty of constructing the translator is driven by the relative complexity of (i-iii).

3. DATA TRANSMISSION ALTERNATIVES

A DTP provides the basic mechanism for transmitting a block of data between source and destination computers in a way which preserves meaning. The prerequisite to a DTP is a means for describing the block of data to be transmitted. Three levels of difficulty can be distinguished depending on whether the block is generated by: i) a given data element type, ii) a pointer free structure, or iii) a structure containing pointers. Examples of the first two levels are a file consisting of (character) text and a COBOL record. An example of the third level would be a PL/1 structure containing pointers.

Case (i) corresponds to the support provided by the Arpanet FTP for files consisting of characters or binary strings. Note that the term "structure", as used in C and PL/1, is a generalization of the term record as used in COBOL.

Implementation of a DTP generated by a given data element type (sometimes called a File Transport Protocol) is straightforward since the transformation from BBR to CNR can be performed on an instance by instance basis. The graph describing the structural interrelationship among the data elements of the block, the data types of the individual elements, and the algorithm for relating block structure to the BBR are all conceptually simple and require minimum supporting mechanism.

Transmission of a pointer-free structure requires a description of the graph of the structure, the types of the structure's data elements, and the algorithm mapping the structure into its corresponding BBR. It follows that a Structure Transmission Protocol (STP) requires substantially more information as well as significantly more complex programs to manipulate this information than does a DTP corresponding to a single data element type.

Preserving meaning in transmitting structures containing pointers is likely to be hard. This reflects the architectural dependence which can exist between the interpretation of the pointer and its representation.

Given an STP, developing an FTP for structured files consisting of some number of instances of a given structure type is straightforward.

It is reasonable to consider developing a database transmission protocol. Unfortunately, most databases contain pointers which are heavily system dependent for their proper interpretation and utilization. (Note that this is a major area of concern within the database translation community [FRYJP 74].) As a result, such a protocol seems unlikely. However, an FTP for files generated by a given structure would be a useful first step in transmitting databases between heterogeneous computers and is straightforward, given an STP.

4. A STRUCTURE TRANSFER PROTOCOL

The major components of a Structure Transfer Protocol (STP) are the information requirements for its operation, commands for its invocation, responses to these commands, and error messages. The following discussion of these issues is based upon a prototype implementation of such a protocol as one of the components of the NBS Experimental Network Operating System (XNOS) [KIMBS 78A].

4.1 Information Requirements

Transforming the source BBR of a structure into the CNR requires knowledge of the graph, data element types and algorithm for mapping the graph into the BBR. Assuming the availability of this information, the basic BBR \( \rightarrow \) CNR transformation is conceptually straightforward. Via the available information, one can determine the type of the next data element to be processed and its bit locations. Thus, the value can be determined and mapped into the corresponding representation of this value for the CNR.
It is important to bear in mind that three BBRs are actually involved in the preceding discussion. These are: i) the BBR corresponding to the block of data at the source computer, ii) the BBR corresponding to the block of data given its CNR representation, and iii) the BBR corresponding to the block of data with the same meaning at the destination computer. Rearrangement is required whenever one of these BBRs stores data elements in a different sequence than that used by the subsequent representation.

4.2 STP Commands

The basic STP commands support record access and status determination. In addition, some options negotiation may be required depending upon the sophistication of a particular implementation.

4.2.1 Record Access

Accessing a remote record requires specification of the host, directory and file. Accessing the host requires an appropriate account. Given an account, the Arpanet provides a model for developing a fully qualified file name having the generic form:

[hostname]<directoryname>filename

Thus, a valid designator would be:
[NBS-1B]<JONES>testfile.

Having identified the file containing the record, the next issue is identifying the record. We assume that it is identified via either a key if random access is supported or the keyword 'NEXT' if sequential access is employed. The following section discusses additional issues which must be considered in accessing databases supported by Database Management Systems (DBMSs).

The minimum set of commands required to support these functions are straightforward. They are: OPEN, RETRIEVE (specified-key or NEXT), and CLOSE together with those commands required to gain access to the system.

4.2.2 Status

In a networking environment, a non-negligible probability exists that either the destination host or the communication network will function incorrectly with the resultant likelihood of missing messages. Thus the need for status determination capabilities to establish what happened to missing data. If updates are involved, these capabilities must be able to determine after a failure has occurred whether the transmitted update has actually taken place. Although the fact that record access is usually supported via a DBMS aids in ensuring integrity of data within the destination host, there is still a substantial requirement that the process within the source host have exact information regarding the extent of processing of the update in case of network or destination host failure.

4.3 Responses

The responses to the three basic commands just described are straightforward and consist of either: i) notification that the desired file has been opened, and thus addressability has been established, ii) a copy of the requested structure in some common network format, or iii) an error message.

4.4 Error Responses

It is relatively easy to generate error responses indicating that a host computer is down or that the PSCN is malfunctioning. However, errors are often more subtle to detect and describe than are those caused by outages. Sharing of network accessible data would be substantially facilitated through providing informative, non-cryptic error messages permitting the user to identify the precise nature of the problem. Because of the inconsistency of errors across systems, this implies the need for developing a collection of uniform error responses which are transmitted to the remote user instead of, or in addition to, the local form of error messages.

4.5 Support Requirements

STP support requirements can be divided into two major categories: i) interfacing to the program requesting the remote record, and ii) simplifying network use.
4.5.1 Program Interfacing

Because of the likelihood that a response may not be received coupled with the generally slower access times to remote data, a message based approach to communication between remote processes is preferable to a CALL/RETURN based approach.

Using a CALL/RETURN based approach implies that the CALLing process enters the wait state upon issuance of the CALL and is awakened upon RETURN. Thus, the CALLing process is unable to continue with other useful processing pending receipt of the RETURN. (This approach is conceptually analogous to the CALL/RETURN mechanism used for subroutine invocation within an individual computer system.)

A message based approach, in contrast, supports transmission of a message containing the desired command but, in contrast with the CALL/RETURN approach, permits the CALLing process to continue processing pending receipt of the response. That is, the message is transmitted, processing continues, and the CALLing process is interrupted upon receipt of the response.

Note that the actual commands issued in either case are similar. The differences are the CALLing process options, the nature of the required support within the computer containing the CALLing process, and the manner in which the program corresponding to the CALLing process should be designed.

4.5.2 Support Requirements

The preceding discussion has shown that operation of an STP requires a substantial amount of descriptive information. It is naturally desirable to simplify the collection and maintenance of this information together with the requirements for accessing individual systems.

Network Operating Systems (NOSs) provide a useful collection of functions for simplifying network usage including uniform user-system and system-system interfaces. An STP has been implemented as part of an Experimental Network Operating System (XNOS) which has been implemented at the National Bureau of Standards [KIMBS 78A]. Through its utilization, one can establish common naming conventions across systems, reference files without having to give fully qualified path names, and provide support mechanisms to facilitate status determination and error handling.

Given that an STP is available, a key requirement for supporting effective network access to data is at hand. We now explore additional issues in meeting this objective.

5. NETWORK DATABASE ACCESS

Two polar approaches to supporting network database access can be identified. In the first approach, a single DBMS is implemented across a collection of networked computers. In the second, one seeks to provide a uniform network viewpoint across a collection of individual DBMSs implemented on different systems.

These two approaches have substantially different protocol requirements. In the first, the user is presented with a single logical port into the system, and all operations of the DBMS are effectively hidden from the user. Thus, the only protocol requirement is that implicit in communicating with the DBMS and, in principle, this is different than remote communication with any DBMS.

The second approach, in contrast, requires extensive utilization of protocols and translation mechanisms. In comparing the merits of these two approaches note that the first is cleaner and provides an absolutely uniform viewpoint since there is only one actual database implemented across several systems. The disadvantage of this approach is the need for database conversion if existing applications are to be accommodated. The second approach provides a straightforward means for accommodating existing DBMSs and for support of a gradual process of expansion. The disadvantage is the attendant need for protocols and translation mechanisms. These issues are currently being evaluated via an Experimental Network Data Manager (XNDM) being constructed at the National Bureau of Standards. The following discussion is based on insights derived through this process.

5.1 Multiple Independent Database Support

The basic scenario underlying the development of XNDM is that of assisting a user (process) in communicating with multiple (existing) DBMSs located in remote hosts. That is, the user issues commands in a common, network wide data manipulation language (DML), which are: i) decomposed into commands against the individual DBMSs, ii) translated from the network DML into the DML appropriate to
the individual DBMSs, iii) processed by the individual DBMSs, and iv) the collection of resulting responses are then aggregated and returned to the user process. (Note that transmitting responses requires an STP.)

5.2 Network Level DML Issues

A basic objective in establishing a common DML to be used by the network user across DBMSs is that it be non-procedural. The network user should be presented with a collection of data structures and, using the common DML, should be able to manipulate them to effect the desired queries or updates.

The preceding statement argues for providing a network view of accessible data as a collection of (virtual) data structures because of their effective elimination of access path considerations, coupled with a relational calculus for the DML. XNOM uses an Experimental Network Query Language (XNQL) which is a subset of SEQUEL [CHAMB 76] coupled with extensions required to support access across a collection of DBMSs. Selection of SEQUEL was made because: i) it was not the DML of any of the DBMSs being considered (MRDS, RDMS, INGRES, MIDS, UNIBASE), ii) it has been subjected to human factors investigations aimed at improving its usability [ETSP 76], and iii) a precise definition is available in the literature [CHAMB 76]. A more extensive discussion of XNQL is given in [WANGP 78] and a discussion of the precise collection of data structures seen by the network user is described in [KIMBS 78B].

5.3 Query Decomposition and Translation

Decomposing an XNQL query involves: i) decomposing the query into subqueries to be applied against individual DBMSs, ii) coordinating the execution of these subqueries, and iii) performing inter-DBMS operations, as required.

Translation of an XNQL query proceeds in two stages. The query is first decomposed into an intermediate representation. This representation is then used as the basis for the translation process. Currently, translators are being constructed between XNQL and the relational calculus system available on Multics (MRDS), a relational algebra system available on Multics (RDMS), and a DBTG system available on Multics (MIDS). While it is relatively clear that such translators can be constructed, it is substantially less clear how to factor in optimizing information that may be available from the local DBMS.

5.4 Local Processing

Support of local processing requires some ability to map locally generated error messages into a form minimizing the amount of DBMS-specific knowledge required for network user understanding. Moreover, strategies need to be developed to characterize the local DBMS in a way which supports determination of efficient DML mappings.

5.5 Response Aggregation

As responses are generated by individual DBMSs, they must be aggregated with others. The ultimate result is generation of the response to the user's command. Support of this aggregation process requires a Structure Transfer Protocol.

The preceding discussion has indicated the major functions which must be performed to support network access to remote DBMSs. Although achieving this objective requires a significant amount of effort, an STP provides a major facilitating tool.

6. CONCLUDING REMARKS

This paper has articulated the major issues in constructing a Structure Transport Protocol and has discussed the role of this protocol in the context of supporting network access to remote DBMSs. Our comments have been based upon a prototype implementation of an STP within XNOS. Our conclusion is that such a protocol is feasible and that its utilization is simplified if a Network Operating System exists.

REFERENCES


