Internet Experiment Note: 195

COMMENTS ON NBS TRANSPORT PROTOCOL PROPOSAL [1]

by Carl Sunshine
University of Southern California
Information Sciences Institute
August 1981

1 INTRODUCTION

1a The U.S. National Bureau of Standards is sponsoring the development of protocols for several levels of computer network architecture. Numerous documents have been produced, mostly by Bolt Beranek and Newman, as part of this program. The purpose of this note is to comment on the recent proposal for a tranpsort protocol [1], concerning both the general specification methodology and the particular mechanisms suggested for this protocol.

1b These comments stem from a moderately detailed reading of the NBS transport protocol [1] and specification methodology [2] documents, and a general familiarity with other work in the NBS program. They necessarily focus on objections and problems rather than compliments, although lack of comment should not be taken as agreement since it has not been possible to completely review these documents. My comments fall into four broad categories: service specification, protocol specification, architecture, and comparison with TCP. The final section should be of special interest to readers who are familiar with U.S. Department of Defense protocol development efforts but not with the details of the NBS proposals.

2 SERVICE SPECIFICATION

2a One of my first points must be that the service specification is very unsatisfactory. I understand there was a decision to be less formal in this area than with the protocol itself, but I think that decision should be reconsidered. In my view, there is no reason that the layer as a whole cannot be viewed as a "service machine" and services defined in essentially the same way as for the protocol. The recent work by SDC on DoD IP and TCP [3,4] provide strong support for this view. An alternative approach based on sequencing expressions has been extensively developed by Schindler and applied to a variety of OSI protocols and services [5].

2b Even if the informal "time line" figures and text now in use for service specifications are maintained, major deficiencies must be corrected. The figures purport to show the relative timing of service primitives within a given service. They do NOT show the relative timing allowed for interaction primitives of

different services (e.g. that T_DATA can only follow T_CONNECT). They do NOT show the relative timing allowed for multiple instances of the same service (e.g. several T_UNIT_DATA data requests may be made, and their corresponding indiciations may

occur in a different order than the requests, but with normal T_DATA, the indications must be in the same order as the requests). They do NOT show that some services can be interrupted or completed by other services (e.g. T_CONNECT service primtives can be followed at various stages by T_DISCONNECT primitives, for example Connect.Req -> Connect.Ind -> Disconnect.Req -> Disconnect.Ind, which is not shown in any figure).

2c There is no indication in the service primitives of how they are associated by the receiver with a particular connection. For example, the DATA primitives have only the data as a parameter. Apparently the connection ID is an implicit parameter of all service primitives. This should be discussed explicitly. For example, what happens when two users try to connect to each other at the same time (using the same transport addresses)? Is this a simultaneous T_CONNECT situation which is not shown, or is it considered two different connections? What if two remote entities issue T_CONNECT requests to the same third party transport address? Are multiple connections to the same address allowed? When a T_UNIT_DATA confirmation is returned to the user, how is it to be associated with the proper initial request (there may be several to different destinations, or even to the same destination in progress)?

2d There is no explicit statement that the parameter values of a request are related to those of the corresponding indication. Presumably they are to be copied, but is this always true?

2e There are many questions about addressing and connection establishment. Some have been mentioned two paragraphs above. Another is that the T_CONNECT service definition specifies that a request will be followed by an indication to the remote address, but what if this address is invalid, or "no one is there" somehow, so the indication cannot be delivered (for example, there may be no process bound to the specified address on the remote system, and it may not operate to create processes on demand). Will this result in a T_DISCONNECT from the remote entity (reason address unknown or unreachable)?

2f Specifying the to and from parameters when the service as a whole is defined (e.g., page 28-29 for T_UNIT_DATA) seems redundant with the specification of all parameters for each service primitive (page 31), and is in fact misleading since it implies that every primitive will carry all of these parameters.

2g There is no discussion or specification anywhere of flow control which is mentioned as one element of the service provided (page 15). In fact the event model used seems to allow an unlimited number of primitives to be requested by the user or

indicated by the entity. There is no mention of any primitives explicitly intended to indicate cease/resume type flow control, or any limit on event queue sizes that could be checked by either party or block acceptance of primitives. Again in Section 6.3 describing an X.25 interface machine, the X.25 RR/RNR flow control packets are accepted, but have no specified effect on the using layer. The machine apparently contains an unlimited size queue of data to be sent (snd_buf), there are no enablng conditions of a flow control nature on user data sending primitives, and no explicit flow control is passed from the X.25 layer up to the user.

2h The expedited data is specified to be "unsynchronized with

respect to data in the normal flow," meaning (a) that the service does not tell the receiver where in the normal data stream the expedited data was generated, and (b) it may be delivered AFTER a subsequently sent normal data unit. I think point b is highly undesirable, and point a is rather inconvenient since it forces users to put synchroniztion markers into the normal data stream. A more powerful service should be adopted in this area.

2i The time figures seem to provide minimal information beyond the text, and in fact all correspond to a request-followed-by-indication model. They could be omitted entirely, or their size greatly reduced. If they are retained, they should be placed in the document so they appear AFTER their citations in the text (many appear before).

3 PROTOCOL SPECIFICATION

3a Most of the comments in this section apply to the extended class protocol found in Section 5.2 of the specification.

3b Sequence numbers for normal data are discussed in Section 5.1.8, but sequence numbers for expedited data are never discussed. Only in the section on header formats do we discover that this field is 7 bits.

3c Section 5.1.12 states that the transport connection will be disconnected if the underlying net connection is closed "unexpectedly." I would think it is an important service feature to protect users from network level difficulties, and to maintain the transport level connection, at least in the extended class, by opening another network connection if necessary. Indeed, the main purpose of the transport layer is to mask network level errors and provide reliable service--giving up in this case seems exactly counter to this goal.

3d The explanation of the difference between two-way and three-way connection establishment in Section 5.1.16 is nice.

3e Section 5.1.17 essentially defines the default processing for all unspecified conditions to be disconnection. It is important to make this notion more rigorous, and to discuss it when defining the basic machine model. It should be extended to include processing of unspecified user requests as well as PDUs. That is, the model should be extended to include definition of an "else" transition that applies if no other transition is matched, with its precise actions given as for other transitions. It is not clear that the desired action is always disconnection--simply ignoring or discarding certain inputs is often desired.

3f The data types defined in Section 5.2.1 are never used later. For example, the type given for the "reason" field of the machine on page 48 is "int_type" rather than the "reason_type" defined on page 46. The types actually used in the bulk of the specification (int, data, adr, and stat), on the other hand, are not defined formally anywhere. The adr_type in particular seems to be a catch-all for several types of addresses (e.g. transport and network address fields are both specified as adr_type).

3g Sections 5.2.3, 5.2.7, and 5.2.8 have similar information and purposes, and probably should appear together. There are far too many items that are "primitives" and hence not defined formally. For example, only two out of perhaps 25 predicates are defined.

3h The function of the "reference" numbers assigned to each side of a connection is never explanined clearly. I am guessing that they are intended to function as a sort of incarnation number, and serve to prevent old packets between the same addresses from being confused with newer ones. This is intended to allow connections to always start with sequence number zero, since they will have different reference numbers. Hence reference numbers must not be reused too frequently, leading to the REF_WAIT state whenever a connection is closed, and the timeout discussed on page 66. As noted, if a host crashes, it must not start ANY connections for the appropriate time period if it forgets the reference numbers used before the crash. All of this should be clarified, with a few references to the relevant literature. The reference numbers also appear to serve an addressing function, being used (implicitly?--see next) to associate inputs with the correct state machine instance.

3i There is a major omission concerning how to associate incoming packets and user commands with a particular connection. Section 5.1.1 states that reference number pairs serve to identify connections, but this is not reflected in the formal spec in any way. Technically, the interfaces should be extended to include these fields, and the enabling conditions should say

([from N:PDU.dst_ref] = src_ref) for an arriving PDU, or
([from U:Src_ref]=src_ref) for a user request, or
([from S:Src_ref]=src_ref) for a timer indication

Interestingly, the X.25 Interface machine in Section 6.3.3.3.3 does include an explicit check of this type on the logical

channel. However, this will not work for initial connection requests where no machine has been esatblished yet--in this case, a new one must be created. This is not modeled formally either.

3j There are problems with events not associated with any specific connection. For example, some timer requests are made before any reference numbers have been assigned, such as in transition 9, page 82. How can the Current State item on page 82 refer to "The transport connection" when there is no tranport level connection associated with this event yet (a specific transport address will only be identified when the CR PDU arrives as a subsequent event)? How is the timeout specified in transition 11, page 84, associated with the correct protocol machine, particularly if more than one N_Connect indication has been received? Similarly, how is the N_Accept indication in tranition 2, page 70, associated with the proper machine?

3k Transitions 12 (p. 85) and 71-2 (pp. 183-6) are applicable to the state set "listening", but the first action (cancelling the timer) is only relevant in the CALLED state, not the CLOSED state. I have not had the time to check all transitions in detail, but if this is indicative of sloppy treatment of state classes, there may be many other bugs of this sort.

3l The 81 transitions are presented in a seemingly unstructured and haphazard fashion. This makes it difficult to understand the protocol, and nearly impossible to check its completeness (has the possibility of every event in every state been considered?). With the default connection termination suggested in Section 5.1.17, technically every specification is complete, but the reader would have much greater confidence if the spec was presented systematically by types of events, or by states, and "error" inputs were explicitly listed, so a syntactic

completeness check could be performed.

3m It would be extremely helpful to provide some sort of index to the numerous transitions, perhaps a conventional state transition diagram with the transition numbers written on the arcs, or a listing of all the transition line specs (with no procedures) together. This would allow the reader to trace interesting scenarios much more easily.

3n The use of overlapping enabling conditions seems dangerous. For example, see pages 367 and 373 where an incoming DATA unit is received from X.25. Combined with the lack of ordering of the transitions noted above, it is difficult for the reader (and the designer) to remember that implicit in transition 24 is the fact that packets with bad sequence numbers have been "filtered out" and handled in transition 18 (which has a higher priority). Perhaps these transitions should be combined, or else the enabling conditions expanded to be mutually exclusive so each stands complete on its own. In the current format, it would be all too easy to modify the specification in one place, forgetting

that it has an effect on another transition that appears pages away.

3o The syntax chosen for transitions, listing the new state first and the old state second seperated by a right-to-left arrow, is at odds with every other example of transition specifications I have seen. The conventional method is to write old -> new, left to right. The authors should at least highlight this difference and provide some justification for it in Section 3.3.3.

3p When default values are passed for parameters, those parameters should not be specified explicitly (e.g. Subscript and Datum parameters at the end of transition two, page 70). This will simplify the specification. Otherwise, why define defaults?

4 ARCHITECTURE

4a Both basic class and extended class protocols are designed to make use of a virtual circuit or connection oriented network level service. This may make sense for basic class, but certainly does not for extended class. Presumably, the reason for selecting extended class mode of operation would be in the situation where unreliable and/or datagram network service was being used. In this case, the transport level goes through significant effort to manage network level "connection" setup and clearing, only to have the interface sublayer undo all of this work for a datagram system! The design of extended class should clearly be optimized for operation over datagram nets. This would allow substantial simplification of both the transport protocol (CALLED and CALLING states, and transitions 1-2, 9-11, 59-60, etc. would be eliminated) and the interface sublayer (essentially null for datagram nets).

4b There seems to be little relation between basic class and extended class protocols. In particular, basic class is in no sense a "subset" of extended class so that an extended class implementation could also communicate with basic class by refraining from use of certain PDUs. While there is some similarity in the interaction primitives of the two classes, it seems that these are essentially two different protocols. Acceptance of this fact would allow redesign of each protocol for

greater efficiency. In particular, the attempt to base extended class on connection oriented network service could be abandoned with great savings as discussed above.

4c The wisdom of defining two classes of protocol at all is open to question. It clearly introduces the possibility of inability to communicate between different "class" users. The extended class protocol is motivated by a view of underlying nets as not fully reliable, providing minimal services, a situation which clearly applies "in general." Basic class is motivated by a view of highly reliable end-to-end connections available at the

network level. Even if some public networks are expected to provide this service, the reality of user-to-user interconnections will in most cases also involve private or local networks, leading to serious questions about the validity of the situations for which basic class is optimized.

4d Masking the datagram nature of the network level from the extended class tranport protocol leads to another inefficiency. The ARPA IP datagram protocol allows the user to supply a datagram "identifier" which is used to reassemble fragments at the destination. If the transport level is aware of this feature, it can purposely use the same ID on retransmissions of the same packet, thereby increasing chances of correct reassembly at the destination (fragments of different retransmissions will be merged). When this feature is hidden from the transport level and the interface sublayer provides a new ID to each packet (including retransmissions) from the transport layer (as specified on p. 395), this is not possible.

4e A difficulty in designing the interaction among layers is apparent in use of the datagram interaface sublayer defined in Section 6.4. When the "connection" is closed at one end (transactions 6,8,10 on pages 407,9,11), no network level disconnect is sent to the remote partner. Hence the remote interface layer machine remains in the OPEN state indefinitely, even though the transport layer has closed its connection.

5 COMPARISON WITH TCP

5a The NBS extended class tranport protocol bears some resemblance to the DoD TCP protocol. Because there is some interest in the applicability of public standards to DoD, I will try to highlight some of the differences.

5b The NBS protocol is designed to operate above connection oriented network services. In DoD, datagram services predominate. Hence the NBS protocol can be expected to be more costly and less efficient in this architecture as discussed above. In addition to the extra states and transitions for network level connection management (see 4a above), the NBS protocol will be less robust since it drops transport connections when there are network problems (see 3c), and cannot support the merging of retransmissions by the IP (see 4d).

5c The TCP uses carefully selected initial sequence numbers to prevent confusion of packets from different incarnations of connections. The NBS protocol uses reference numbers which also perform an addressing function in a fashion not fully explained (see 3h-3i). The NBS protocol allows a less reliable two way handshake to establish connections in addition to the three way

handshake used in TCP.

5d The addressing features are not sufficiently well-defined in the NBS protocol to allow a clear comparison with TCP. Consistent with its use of underlying connection services, only the initial call request PDU includes transport addresses (format and semantics unspecified), while subsequent PDUs carry only the reference number or numbers (presumably--this is left unspecified), and no network level addresses. In TCP all packets carry transport level ports (16 bits). Certain ports are assigned to "well-known" applications with TCP, for which there seems no counterpart in the NBS protocol. The NBS protocol preserves the boundaries between sent transport service data units when they are delivered (i.e., provides record boundaries to users), while the latest version (August 1981) of the TCP is stream oriented and provides no record boundaries. Instead the TCP provides a "Push" service feature to force the prompt delivery of all data sent up to that point.

5e The unit of sequencing and flow control in TCP is 8-bit bytes, while in the NBS protocol it is protocol data units (PDUs) whose maximum size is defined at connection establishment. This follows the stream oriented nature of TCP data transmission versus the record oriented aspect of the NBS protocol. Sequence and acknowledgement numbers are 32 bits in TCP, and 15 bits in the NBS protocol.

5f The NBS protocol maintains a minimum level of activity by sending Acks if necessary in either direction. Periodic transmission, particularly when a zero window is opened, is only suggested in TCP. Acks do not carry a sequence number in the NBS protocol, seemingly allowing them to be received and processed out of order more easily than with TCP. Acks can not be "piggybacked" on data packets in the NBS protocol, seemingly leading to more packets exchanged (see section 5.4.2 giving format of TPDUs). Error packets have no sequence or acknowledgement numbers in the NBS protocol, seemingly allowing duplicates to be accepted.

5g A single unit of "expedited data" may be transmitted at one time in the NBS protocol whose position and delay relative to the normal data units is unknown (see 2h above). The latest TCP (August 1981) provides an "Urgent" service feature which indicates a point in the normal data stream at which "urgent" or priority data has been sent. The urgent indication is presented to the receiver at least as soon as the urgent data.

5h The features for terminating connections (both "graceful" and immediate disconnect) in the NBS protocol appear to duplicate those in TCP. The security, compartment, and precedence features in the NBS protocol also seem to duplicate those in TCP although the fields are smaller and their semantics are left undefined.

REFERENCES

[1] J. Burruss and others, Specification of the Transport Protocol, Report No. ICST/HLNP-81-1, National Bureau of Standards, February 1981.

[2] R. Tenney and T. Blumer, An Automated Formal Specification Technique for Protocols, Report No. 4581, Bolt Beranek and Newman, February 1981.

[3] G. A. Simon and M. M. Bernstein, DCEC Protocols Standardization Program/ Protocol Specification Report, TM-7038/204/00, System Development Corp., July 1981. (Also see DARPA Internet Experiment Note No. 186.)

[4] M. M. Bernstein, DCEC Protocols Standardization Program/ Preliminary Transport Protocol Specification Report, TM-7038/207/01, System Development Corp., August 1981.

[5] S. Schindler, U. Flasche, and D. Altenkrueger, "The OSA Project: Formal Specification of the ISO Transport Service," Proc. Computer Networking Symposium, National Bureau of Standards, December 1980.