INDRA
Working
Paper

INDRA Note 965
TSIG 4.1
IEN 154
7th August 1980

Realization of the Yellow Book Transport Service Above TCP

C. J. Bennett

ABSTRACT: This note defines an enhancement of the service provided by the US DoD Standard Transmission Control Protocol (TCP) sufficient to meet the requirements of the UK Network Independent Transport Service (the Yellow Book). This note supersedes an earlier version (INDRA Note 959, TSIG 4.0 and IEN 153).

Department of Computer Science

University College London

1. Introduction

This document defines a means of providing the Yellow Book Transport Service [1] above the DARPA Internet facilities, in particular TCP [2], so that this can then be used to support other services such as endpoint file transfer without requiring UK hosts to implement the Internet family of protocols. It assumes familiarity with both TCP and the Yellow Book.

The basic approach taken is to enhance the TCP service along the lines suggested for enhancing X25 in Annex I of the Yellow Book, taking into account the different services provided by TCP. In addition, the note discusses how to integrate Yellow Book TCP so that it can run alongside ordinary TCP - an issue the Yellow Book ignores for Yellow Book X25.

2. Deficiencies of TCP

A comparison of the services provided by TCP and those provided by the Yellow Book reveals that TCP is unable to support directly, either in whole or in part, the following Yellow Book features:

(i) The RESET and ADDRESS primitives.

(ii) The Yellow Book multiple-domain addressing structure. The TCP address space constitutes a single naming domain in Yellow Book terms. Consequently, features involving addressing - notably ACCEPT - are inadequately supported by TCP.

(iii) Much of the subsidiary information provided with Yellow Book primitives. The fact that the source address provided with certain actions such as DISCONNECT is not provided is again a limitation of the global TCP naming domain. The Yellow Book 'Explanatory Text' parameters have no corresponding feature in TCP.

(iv) The closest equivalent to Yellow Book EXPEDITED data - theoretically requiring a priority data channel - is TCP URGENT data. However, TCP URGENT data remains in sequence, and the URGENT pointer only marks the end of the data. Its beginning is not delimited.

(v) The Yellow Book DISCONNECT is a full-duplex close, whereas the TCP CLOSE is only half-duplex. The TCP RESET is a unilateral close, used in error conditions. Connection closure

provides particularly subtle problems.

Hence in order to provide a Yellow Book service above TCP an enhancement of TCP is necessary. The remainder of this document discusses such an enhancement.

3. Principles of the Enhancement

The basic principles of the enhancement are as follows:

(i) Where a TCP function corresponds directly to a Yellow Book function that TCP function is used directly.

(ii) Where the Yellow Book function requires more information or action, the TCP function is associated with a TCP Control Message in a defined way. This message is a record of defined format containing the information required.

(iii) Where there is no TCP function even remotely corresponding to the required Yellow Book function, a control message is defined which may be used by the source and destination processes if possible, and may be forwarded into other transport domains more capable of taking the appropriate action.

4. The Yellow Book TCP Enhancement

4.1 Distinguishing the Yellow Book TCP

The services using the Yellow Book enhancement to the TCP will be identifiable through the TCP socket number used. These will be allocated for standard services as required.

4.2 Identification of Yellow Book TCP Messages

The data stream is structured into records, which in turn are substructured depending on the record type. These records are independent of TCP letter indications as the latter are purely push delimiters.

The record structure proposed is as illustrated in Figure 1. Each record is prefaced by a single octet, known as the TYPE octet. This takes a value of 0 for data records and a value of 1 for command messages. All other values are undefined.

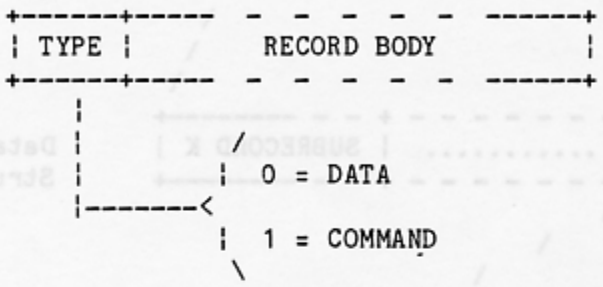

Figure 1: Letter Structure in Yellow Book TCP

4.3 Structure of TCP Data Messages

A TCP Data Message is a Yellow Book TCP record of TYPE 0.

Each record consists of a number of SUBRECORDS. Each subrecord consists of a header octet and a number of data octets, up to a limit of 127 octets.

The subrecord header is a single octet. The high-order bit of this octet, if set to 1, declares that the current subrecord is the last subrecord of the current record. The remaining seven bits define the length in octets of the current record.

This structure is illustrated in Figure 2.

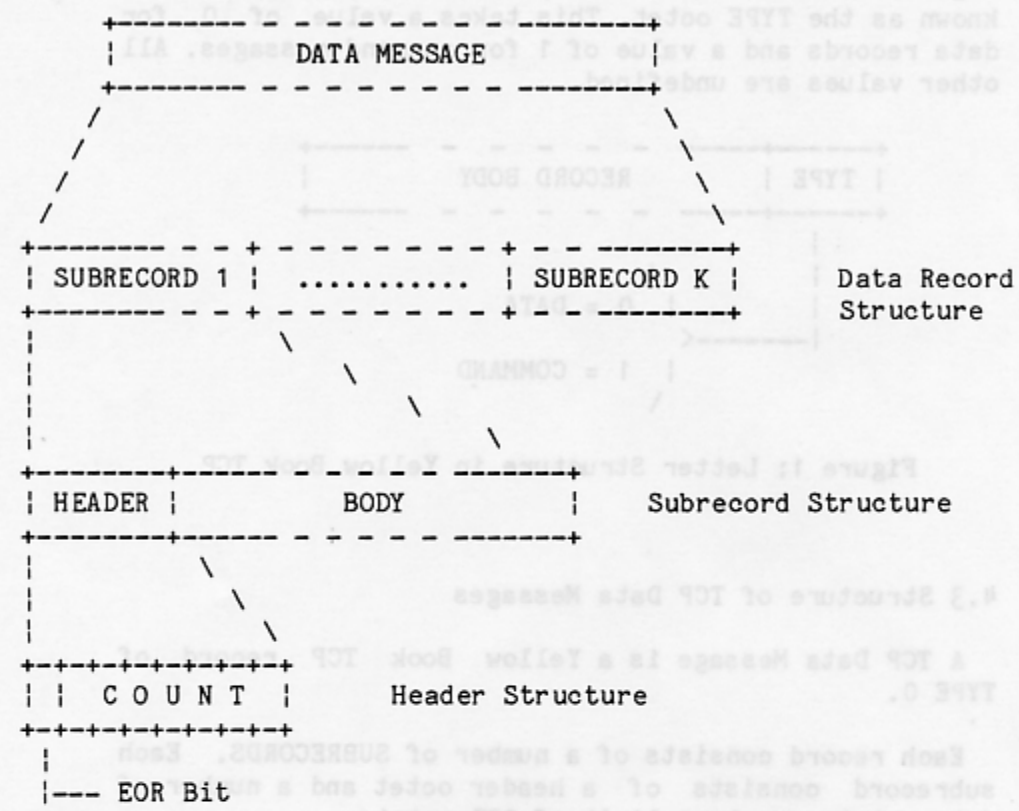

Figure 2: TCP Data Message Structure

4.4 Structure of TCP Command Messages

A TCP Command Message is a Yellow Book TCP record of TYPE 1.

The first octet of the message defines the COMMAND CODE of the message. These codes are defined in subsequent sections, and have been chosen to correspond to the command codes of the X25 Command Messages.

Following the command code is a number of PARAMETERS. The significance of these parameters is defined by their position in the parameter sequence for each command, and the command message is completed only when all parameters have been specified; thus no parameter may be omitted, though it may have a null value.

Most parameters have a free field format. For this reason each parameter is constructed of a number of FRAGMENTS. These fragments consist of a header byte and the body of the fragment, which may have a maximum length of 127 octets.

The fragment header is a single octet. The high-order bit of this octet, if set to 1, declares that the current fragment is the last fragment of the current parameter. The remaining seven bits define the length in octets of the current fragment.

This structure is summarised in Figure 3.

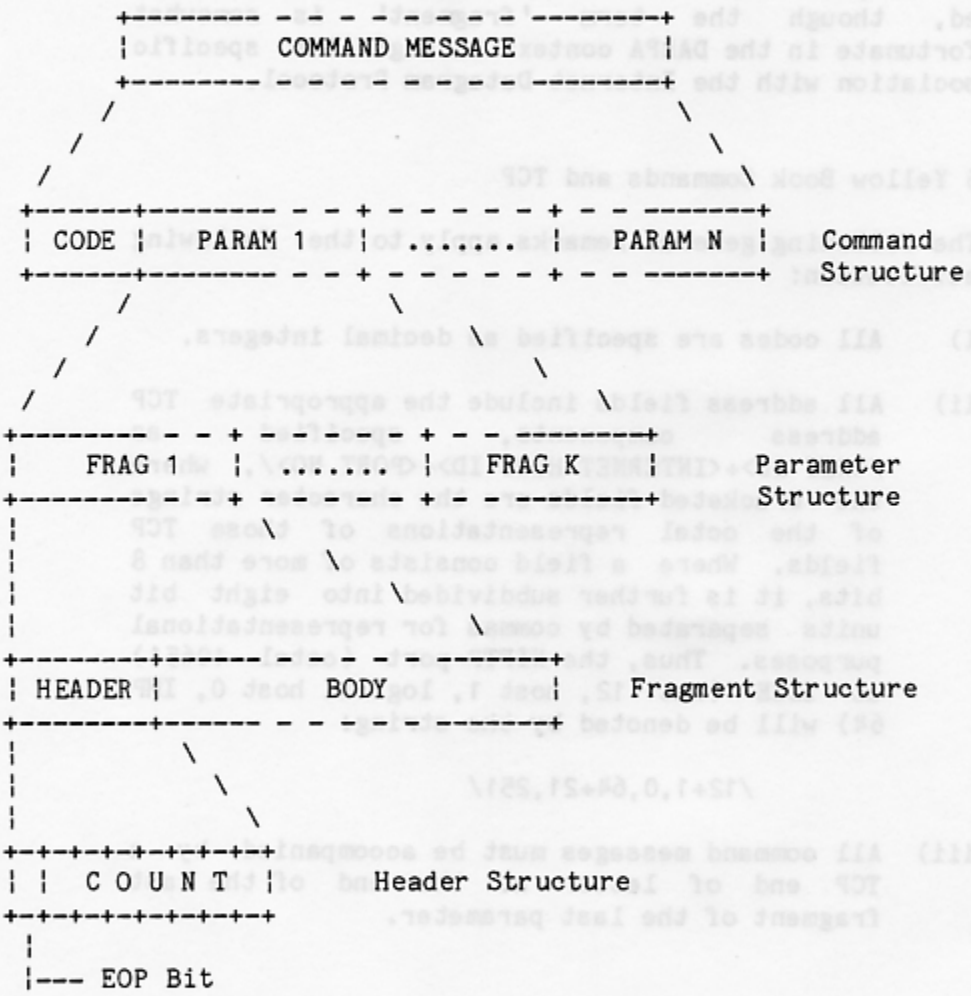

Figure 2: TCP Command Message Structure

A parameter with a null value is represented by a fragment header whose EOP bit is set to 1 and whose count field is set to 0. The rules governing the syntax of free-field parameters are the same as those defined in section 2.7 of the Yellow Book, based on the use of the IA5 character set, except where otherwise noted.

The sole difference between this structure and the X25 Command Message structure is that the count field in the fragment header is extended by one bit - this bit is used for a specific purpose in X25 CONNECT messages which does not arise with the TCP. This doubles the maximum fragment size. Because of the similarity of structure the same terminology has been used, though the term 'fragment' is somewhat unfortunate in the DARPA context through its specific association with the Internet Datagram Protocol.

4.5 Yellow Book Commands and TCP

The following general remarks apply to the following specification:

(i) All codes are specified as decimal integers.

(ii) All address fields include the appropriate TCP address components, specified as /<NET ID>+<INTERNET HOST ID>+<PORT NO>/, where the bracketed fields are the character strings of the octal representations of those TCP fields. Where a field consists of more than 8 bits, it is further subdivided into eight bit units separated by commas for representational purposes. Thus, the NIFTP port (octal 10651) at ISIE (net 12, host 1, logical host 0, IMP 64) will be denoted by the string:

/12+1,0,64+21,251/

(iii) All command messages must be accompanied by a TCP end of letter at the end of the last fragment of the last parameter.

4.5.1 CONNECT

The CONNECT command message is defined by the following code and parameters:

Code = 16
Parameter 1: Called Address
Parameter 2: Calling Address
Parameter 3: Quality of Service
Parameter 4: Explanatory Text

This message will be preceded by the usual TCP three-way handshake. Where possible or appropriate, the quality of service parameter will be used to select TCP quality of service from the options defined in the TCP specification.

The CONNECT message will be the first message sent by the calling party. It will be possible for the calling party to initiate the transfer of data before the arrival of an ACCEPT message.

4.5.2 ACCEPT

The ACCEPT command message is defined by the following code and parameters:

Code = 17
Parameter 1: Recall Address
Parameter 2: Quality of Service
Parameter 3: Explanatory Text

This message will be the first message sent by the called party after the three-way handshake, unless the call request was rejected (see DISCONNECT, below).

4.5.3 DISCONNECT

The DISCONNECT command message is defined by the following code and parameters:

Code = 18
Parameter 1: Reason
Parameter 2: Address of DISCONNECT Initiator
Parameter 3: Explanatory Text

The reason parameter is a single octet giving a machine-oriented encoding of the reason the DISCONNECT was initiated. The defined reasons are listed in Appendix B of the body of the Yellow Book. Parameter 2 is included to cover the case where the DISCONNECT was initiated by some intermediate gateway (where 'gateway' is used in the Yellow Book sense).

DISCONNECT will always cause the TCP to issue an URGENT call. On receipt of a DISCONNECT message, no further data may be sent and all data currently queued for transmission should be flushed if possible. No data will be passed across to the user after a DISCONNECT has been issued.

Beyond this, the exact DISCONNECT sequence used varies depending on the state of the connection, as follows:

(i) If the DISCONNECT is being used to reject a CONNECT request, the DISCONNECT message will be followed by a TCP RESET. This will abort the TCP connection, flushing all outstanding data. No response is expected. The URGENT pointer points to the TCP RESET.

(ii) In the normal case of closing an open connection, the DISCONNECT issued by the initiator will be followed by a TCP FIN. The remote party will respond with an optional DISCONNECT message accompanied by a FINACK and a FIN. The URGENT pointer points to the TCP FIN.

(iii) For error terminations, a DISCONNECT message should be answered with a TCP RESET. The issuer of the DISCONNECT will also issue a TCP RESET after a timeout period, if a RESET has not already been received. The URGENT pointer points to the end of the DISCONNECT message.

4.5.4 DATA

DATA is sent as a sequence of Yellow Book TCP data messages, as defined above.

4.5.5 PUSH

The PUSH function is conveyed by use of the TCP EOL function, pointing to the data octet at which the PUSH was issued.

Note that it is possible to collapse EOL indications, effectively combining PUSHes. Strictly speaking, the Yellow Book requires that a PUSH remains in sequence, but this is not necessary to obtain the required

effect. In order to propagate the PUSH, it is necessary that the TCP delivers EOL indications to the user process. The circumstances under which this occurs are currently unclear. It may be necessary in the future to define a PUSH command message.

4.5.6 EXPEDITED

The EXPEDITED command message is defined by the following code and parameter:

Code = 21
Parameter 1: EXPEDITED data

EXPEDITED data is accompanied by a TCP URGENT pointer pointing to the end of the message. There are no restrictions on the encoding of EXPEDITED data messages beyond the normal fragment structuring rules.

It should be noted that this will cause the receiver of the URGENT to process all data up to the URGENT pointer in 'fast' mode, whether EXPEDITED or not. It may or may not be possible to deliver the EXPEDITED data to the user ahead of sequence. As noted above, TCP has no direct equivalent of a priority data channel, but the mechanism described at least allows the preservation of EXPEDITED data so that it may be passed as such in subsequent networks.

4.5.7 RESET

The RESET command message is defined by the following code and parameters:

Code = 19
Parameter 1: Reason
Parameter 2: Address of RESET Initiator
Parameter 3: Explanatory Text

TCP has no equivalent of the RESET function (a TCP RESET is something else entirely). Thus the only TCP action taken with a RESET message is to accompany it with an URGENT pointer pointing to the end of the RESET message.

As with DISCONNECT, the defined RESET reasons are those listed in Appendix B of the main portion of the Yellow Book. The address parameter is again included to cater for the case where a RESET was initiated in some

intermediate network.

A RESET may only be issued if the connection is fully open and there are no RESETs already outstanding. A RESET message must always be replied to with another RESET message, leaving the connection open, or with an error DISCONNECT message followed by a TCP RESET, which will abort the connection. All data and control messages, with the exception of DISCONNECT, received after a RESET has been issued and before a RESET reply has been received, will be discarded without informing the user. In the case of DISCONNECT, the connection will be considered as having closed in an abnormal state. If a DISCONNECT has been issued, a received RESET will be ignored.

Where possible the issue of a RESET should cause the sender to flush its transmission buffers.

4.5.8 ADDRESS

The ADDRESS command message is defined by the following code and parameters:

Code = 20
Parameter 1: Address
Parameter 2: Address Qualifier

The ADDRESS qualifier is a single octet parameter taking one of the values defined in the Yellow Book:

0: The message is passing towards the addressed object.

1: The message is passing away from the addressed object.

2: An addressing error has been detected.

There is no associated TCP action taken with an ADDRESS message.

The receiver of an ADDRESS message will perform the appropriate ADDRESS transformation as defined in the Yellow Book.

It is recommended that the ADDRESS function should not be used.

5. Conclusions

One of the difficulties of writing a note such as this is that it is addressed to several audiences with different interests and not necessarily a great deal of overlap either in aim or in background.

The immediate audience is the team at University College London who are involved in implementing a 'Protocol Convertor' to make possible direct access between hosts in Britain using the CCITT and UK national standards and the hosts in the US based on the DARPA Internet standards. For this audience, the document hopefully defines an answer to what will soon be a practical need, though it is a matter for continuing debate to what extent the full enhancement defined here will be implemented.

Within Britain, the wider audience aimed at is centred on the Transport Service Implementors' Group. For this group, the aim of the document will be well understood - it is defining a service enhancement similar to the one that is already defined for X25 and the ones they are defining for their local campus networks. The aim is to provide a common transport service for all these systems. They will be unfamiliar with the detailed nature of the TCP itself, but this is not particularly important. The major interest of the document lies in the fact that the system being enhanced is not an ordinary local network, but an entire family of networks, and the resultant enhancement will make possible direct authorised access between UK and US hosts. I would also like to point out the issue of separating Yellow Book enhancements from ordinary uses of a network service. This issue is not addressed by the X25 enhancement specification.

Much of the US Internetwork Group is likewise unfamiliar with the concepts and details of the Yellow Book Transport Service. A summary of these concepts will be made available in a later IEN. For them, the document will be of interest in that it shows how to coordinate two very different approaches to internetworking. The Catenet, based on TCP, can be described as a strongly connected internetwork system, with common transport protocols and a common address space. The UK Transport Service takes an approach based on providing a common service interface, which leads to a weakly connected system with common service protocols and no common address space. Within this approach, the entire Internet system appears as a single component

network, as delimited by the TCP and its address space.

6. References

[1] - PSS User Forum Study Group Three: "A Network Independent Transport Service" SG3/CP(80)2. February 1980.

[2] - Information Sciences Institute: "Transmission Control Protocol" IEN 129.January 1980.