R. Hinden
Bolt Beranek and Newman Inc.
January 1981

Design of TCP/IP for the TAC

1. Introduction

This document is a working design document for the development of the Transmission Control Protocol (TCP) and Internet Protocol (IP) for the Terminal Access Controller (TAC). The TAC is a terminal controller that supports the TCP and NCP host to host protocols. It will run in an H-316 computer with a Multi-Line Terminal Controller (MLC) and an 1822 host interface. It is based in part on the existing H-316 TIP.

This document is meant as a guide for the implementation of the TAC. The intent is not to write a specification that describes everything in fine detail, but to describe the overall system and how its pieces interact with each other. Also, it discusses changes to parts of the existing H-316 TIP.

Everything in this document is subject to change. As the implementation and debugging proceed, new things will be learned. This will force a re-evaluation of the design decisions made here. Undoubtedly, some things will change.

The document is written assuming a working knowledge of the 316 TIP, Internet Protocol, Transmission Control Protocol, and Network Control Protocol. All numbers in this document are in decimal.

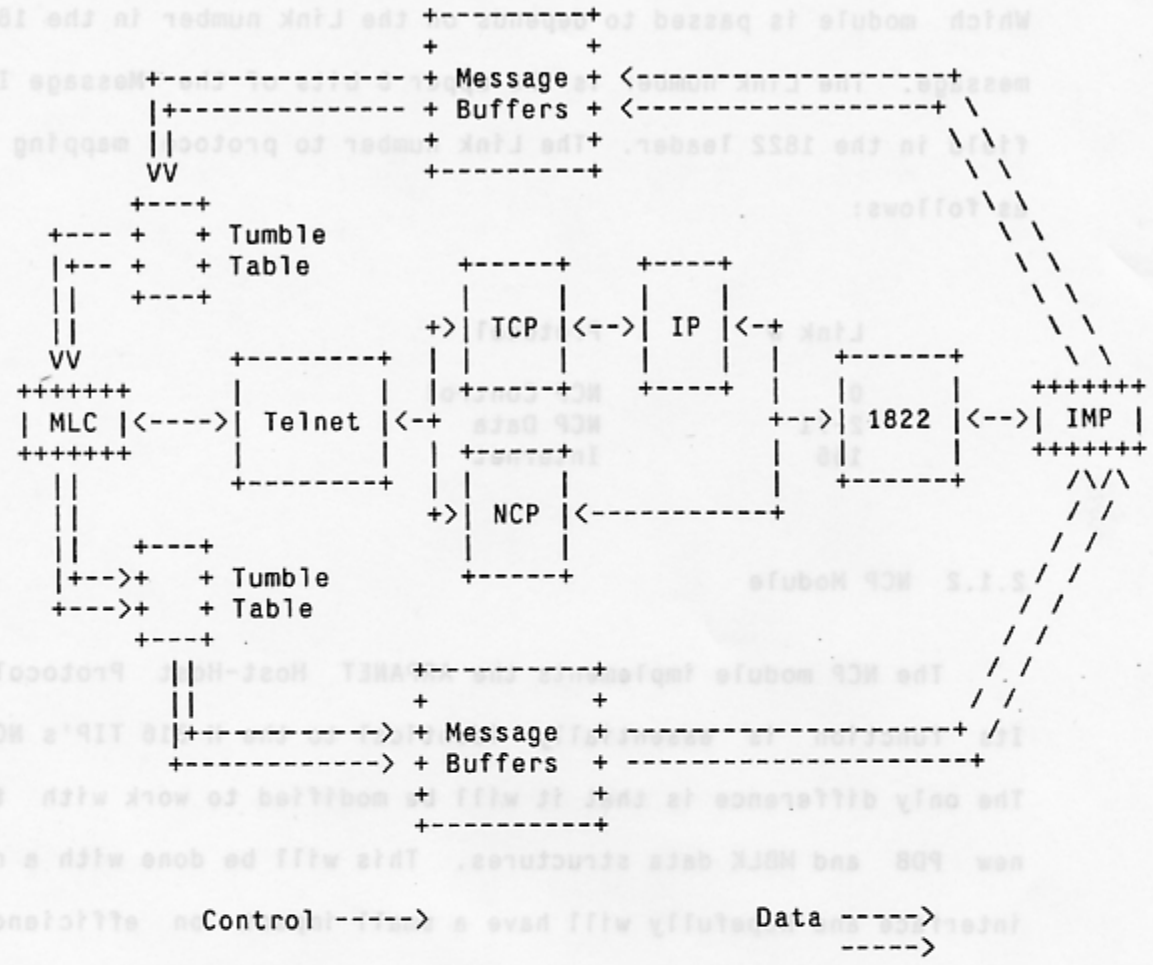
2. Overall Data Flow

A basic premise in the design of TAC is that data should not be moved between buffers, rather the pointers to the data should be passed between program modules. Thus, when a message is read into a buffer, pointers to it are passed between the different protocol modules. When a character is read from the MLC and put into a buffer, the protocol modules manipulate the buffer pointers, not the data itself. This is illustrated in Figure 1.

2.1 Receiving Data

To receive data from the network, a message is read from the 1822 host interface into a Message Block (MBLK). If the message will not fit in one MBLK, the remainder will be read into other free MBLKs until the message has been completely read in. All the MBLKs containing the same message will be linked together. A Protocol Data Block (PDB) will be created to point to the MBLKs. The pointers that are passed between the protocol modules will point to the PDBs. More details on the PDBs and MBLKs can be found in the section on "Data Structures".

R. Hinden
Bolt Beranek and Newman Inc.
January 1981



2.1.1 1822 Module

The 1822 module is given a pointer to a PDB. This module will act directly on the message if it is an 1822 control message (i.e., a RFNM). It will update the appropriate data structure to initiate the action to be taken. If the PDB contains an 1822

data message, it will be passed on to the next protocol module. Which module is passed to depends on the Link number in the 1822 message. The Link number is the upper 8 bits of the "Message ID" field in the 1822 leader. The Link number to protocol mapping is as follows:

Link #	Protocol
0	NCP Control
2-71	NCP Data
155	Internet

2.1.2 NCP Module

The NCP module implements the ARPANET Host-Host Protocol. Its function is essentially identical to the H-316 TIP's NCP. The only difference is that it will be modified to work with the new PDB and MBLK data structures. This will be done with a new interface and hopefully will have a small impact on efficiency. When the NCP module is done with a message, it will pass the pointer to the PDB to the Telnet Module.

2.1.3 Internet Module

When the Internet Protocol (IP) module gets a pointer to a PDB it first checks the checksum in the IP header and then checks that the destination address is correct (it should be the address

of the TAC running the code). If either of these checks fails, the datagram is discarded. Also, if the destination address was incorrect an IP error report will be sent to the source of the datagram. The next check is whether or not the datagram is fragmented. If so, then the IP module will perform reassembly. This is described in detail in the section on "Internet Protocol". When the IP module gets a complete datagram (either received whole or reassembled) it will pass it on to the next protocol module. Which module it is depends on the "Protocol" field in the Internet header. If a datagram is received for a protocol that is not supported, it will be discarded and an IP error report sent to the datagram source. The protocols supported are as follows:

Protocol #	Protocol Name
3	Gateway to Gateway Protocol
6	Transmission Control Protocol
71	Packet Core
20	TAC Monitoring

2.1.4 TCP Module

When the Transmission Control Protocol (TCP) receives a PDB it first checks the checksum of the message and the validity of the TCP header. If the message that passes this check is for a valid connection, and its sequence number is in the receiving window, the TCP module will set it up for the open connection.

The data will be sequenced if necessary at this point. This is described in detail in the section on the "Transmission Control Protocol". The next module is then informed that there is data to be processed.

Flow control is implemented by using the TCP Acknowledgement (ACK) and the Window size parameters. A fixed number of characters will be buffered for each connection. As characters are accepted they will be ACKed until the limitation is reached. As the ACK value is advanced, the window will be shrunk correspondingly. When the next module takes data from the message, buffer space becomes available. This causes TCP to advance its window, thus allowing the distant host to send more data.

Normally the new ACK and window values will be sent out with the next data message from that connection. If nothing is pending for this connection, a message with just the updated ACK and window values will be sent. This is described in more detail in the section "Transmission Control Protocol".

2.1.5 Telnet Module

The Telnet module is given a pointer to a PDB when there is data to be processed. This data may be from the NCP or TCP

protocol modules. It takes characters out of the MBLKs, looks for Telnet commands, and outputs them to the MLC. This output is done using the existing TIP's "Tumble Tables". This will work using "OIs", which means that every time an "OI" comes in for a port, Telnet will check if there is another character to output.

2.2 Sending Data

Data that is sent out to the network normally comes in from the MLC and is received in "Tumble Table" format. This is a block which is filled by the MLC. Its format is one word for each character input. The low order byte of the word is the character and the high order byte is the line number that the character came in on.

When the block is received it is passed to the Telnet module. This module takes the characters out and processes them. As this is happening another block is being filled by the MLC.

2.2.1 Telnet Module

When the Telnet Module gets a character for a port, it first checks if there is an open connection for that port. If not, it discards the character and outputs a bell character to the port.

Next, it checks to see if there is room in the MBLK for another character. If not, then the character is discarded and the bell rung. If there is room, the character is put into the MBLK and the proper pointers are advanced. Telnet then indicates to the next protocol module that there is data to send. Depending on which protocol is being used for the connection, this is either the NCP or TCP module.

2.2.2 TCP Module

When the TCP module gets a signal that there is new data that should be sent, it first checks if there is room in the sending window to send more data. This done by checking if the last sent but unacknowledged data is at the right edge of the sending window. If there is no room, then nothing will be sent. Otherwise, the TCP module will adjust the pointers in the PDB and MBLKs to point to the correct data and update the TCP header.

Flow control in the sending direction is done by maintaining three pointers in the PDB. These are pointers to data in the MBLKs. They are pointers to the last ACKed character, the last sent but unACKed character, and the last not-yet-sent character in the MBLK. As data is ACKed, sent, or put into the MBLK, the appropriate pointer is advanced.

When the TCP module is ready to send the data, it checks to see if the 1822 module can send the data (i.e., there are not more than eight outstanding messages). If the data can be sent, then the TCP module will compute a checksum for the message and pass a PDB pointer to the IP module.

2.2.3 Internet Module

When the IP module gets a pointer to a PDB it first checks to see if it knows where to send the datagram. If the destination is on the same network as the TAC, the Internet module will use that as the address. If the destination is on a different network, then it will send it to a gateway. The procedure to decide which gateway to use is discussed in more detail in the section "Internet Protocol".

The IP module will then build an IP leader in the MBLK and compute the checksum of the leader. It will then pass a pointer to the message to the 1822 module.

2.2.4 1822 Module

When the 1822 module gets a pointer to a PDB, it will always send the message it contains to the destination specified in the PDB. The destination host will either be a server host or a

gateway. The 1822 module will keep track of the number of outstanding (no RFNMs received) messages sent to a host. This will be used by the NCP and TCP protocol modules to insure that the IMP will never block the TAC's host interface due to having more than eight outstanding messages.

When the 1822 module sends the message, it will build an 1822 leader in the MBLK. It will then send the message to the IMP via the host interface hardware.

3. Control and Priority

The code in the TAC will run either at the interrupt level or at the background loop. The interrupt routines will support the host interface, MLC, and clock. In addition, high priority protocol routines will run at the task interrupt level.

The background loop will contain most of the TAC code. The protocol modules will run here. They will be executed in the following order: 1822 Input, IP Input, TCP Input, NCP Input, Telnet Input, Telnet Output, NCP Output, TCP Output, IP Output, and 1822 Output.

Each protocol module will have an input queue. When it runs, it checks for an entry on its queue. If it finds something, it takes it off the queue and processes it. Some of

the protocol modules will be written to process all entries on their queue before exiting; others will process one entry and then exit. The NCP, TCP, and Telnet modules will process one entry. The 1822 and IP modules will process all entries.

4. Data Structures

A new system of buffers will be used in the TAC. It consists of two types of blocks, the Message Block (MBLK) and Protocol Data Block (PDB). These are used both for receiving and transmitting messages and for buffering characters on input and output.

The structure of these buffers is such that when a protocol module is passed a message it is given a pointer to a PDB. The PDB includes a link to the first MBLK. The main function of the PDB is to save frequently accessed things in the message and to point to the message. The MBLKs contain the actual message. They also have fields to facilitate reassembly and sequencing.

4.1 Message Block

The main function of the Message Block (MBLK) is to hold messages. It will be used for all protocols. If a message will not fit in one MBLK, then the remainder will be put into a second

MBLK. The second will be linked to the first. The length of the MBLK will be either 30 or 60 words. The 30 word MBLK is used for sending data and the 60 word MBLK is used for receiving.

The header of the MBLK consists of 4 words. See Figure 2 for the format of the block. The "Link" is used to point to other MBLKs. The "Offset" field is used for reassembling IP fragments and sequencing TCP data. During these operations it contains the offset of where this data is relative to the data in the previous MBLK. Zero means that there is no missing data. This is discussed in detail in the section on "Reassembly".

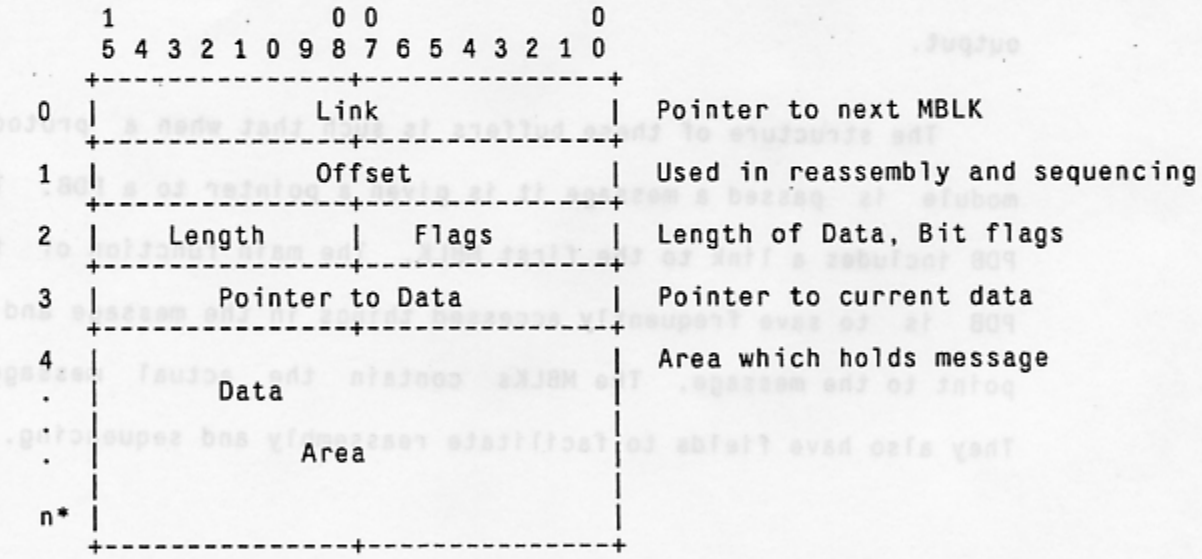


Figure 2 . Message Block Format

* Where "n" is either 29 or 59, depending whether the block is used for sending or receiving.

The "Length" and "Pointer to Data" fields are used to indicate where and how much data is in the MBLK. The meaning of these is always relative to the protocol module currently processing the message. For example: when a message is read in from the host interface the "Pointer to Data" will point to the 1822 leader and the "Length" will be the length of all the data in this MBLK. When the 1822 module is ready to pass the data to the next protocol module, it adjusts these fields to refer to the data after the 1822 leader. In this way, a protocol module need not know what, if any, protocol preceded it.

The "Flags" is a bit field containing such things as End of message, I/O in progress, Read or Write, small or large block, etc. The "Data Area" is where the actual message is stored. The small size MBLK (30 words) is sized to contain an 1822, IP, and TCP leader, but no data. The large size (60 words) can contain the leaders plus up to 60 bytes of data.

4.2 Protocol Data Block

The Protocol Data Block is a header block for one or more MBLKs that make up a message. It contains pointers to the first MBLK, pointers to specific leaders in the MBLKs, frequently accessed items from the message, and a link to the next PDB.

As previously stated, it is pointers to PDBs that are passed between protocol modules. When a protocol module gets a PDB, it expects to find one PDB, which points to one or more MBLKs. The data in the MBLKs is expected to be in sequence and non-fragmented. This requires that each protocol module insure that the data it passes to the next module be contiguous. This is best described with the following example:

When the Internet module gets two fragments of the same datagram, it needs to reassemble them before it can pass them to the next protocol module. What it does is to take the MBLKs containing the second fragment and link them into the proper places in the list of MBLKs of the first fragment. As it does this, it adjusts the fields in the MBLKs to point to the correct data. When it has linked in all the MBLKs from the second fragment, it puts the PDB, which controlled the second fragment, back on the free list of PDBs.

The TCP module performs a similar operation to sequence the data before it passes it to the Telnet module. The format of the PDB is shown in Figure 3.

The first field in the PDB, "Link to next PDB", is a pointer to another PDB. This is used for reassembly and sequencing. The next field in the PDB is the address field. This is either the source of the message if it was received or the destination if it

R. Hinden
Bolt Beranek and Newman Inc.
January 1981

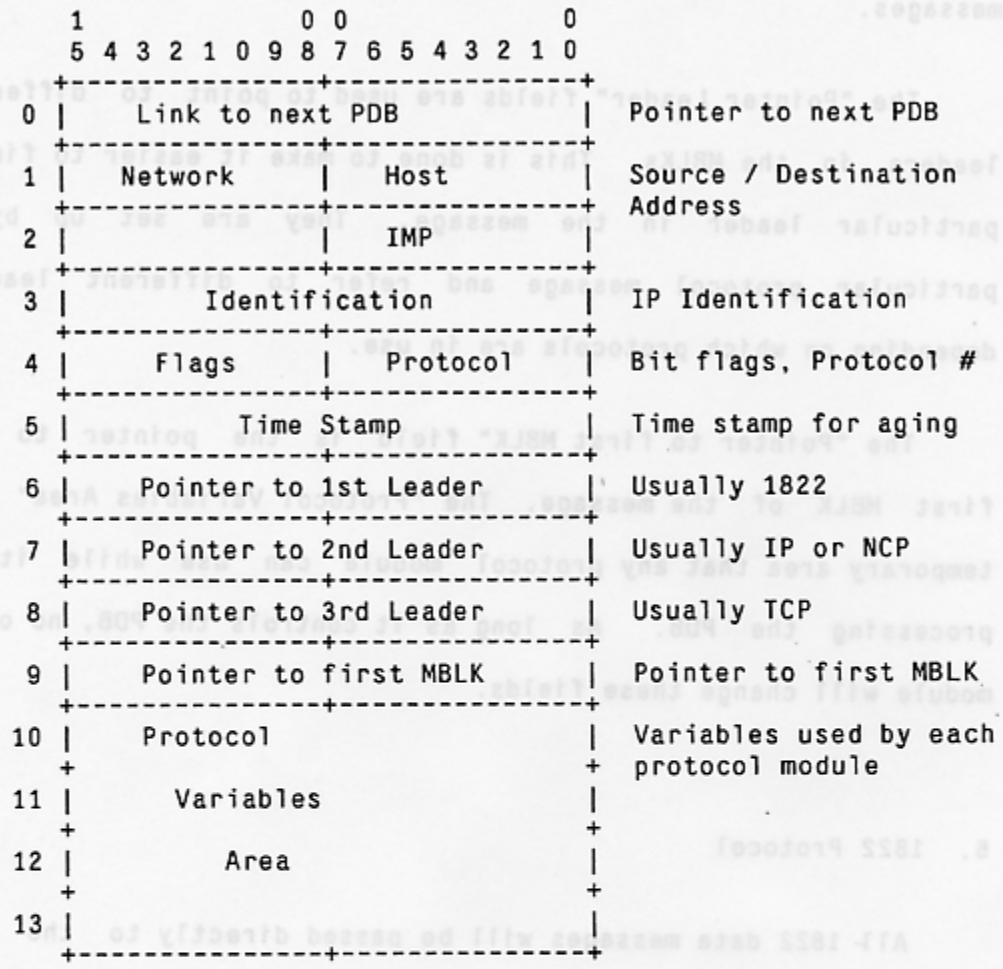


Figure 3 . Protocol Data Block Format

is to be sent. The "Identification" field is the internet identification which is used in assembling internet fragments. The "Flags" field is a bit array used for things like datagram complete, EOL, Urgent, Read or Write, block free, in use, hole, etc. The "Protocol" field is the host-to-host protocol the

message is for. The "Time Stamp" field is used for timing out messages.

The "Pointer Leader" fields are used to point to different leaders in the MBLKs. This is done to make it easier to find a particular leader in the message. They are set up by a particular protocol message and refer to different leaders depending on which protocols are in use.

The "Pointer to first MBLK" field is the pointer to the first MBLK of the message. The "Protocol Variables Area" is a temporary area that any protocol module can use while it is processing the PDB. As long as it controls the PDB, no other module will change these fields.

5. 1822 Protocol

All 1822 data messages will be passed directly to the next protocol module. When the 1822 module gets a control message it will call a routine supplied to it by the next protocol module. For example, the NCP module will supply a routine to be called when the 1822 module receives a "RFNM" on an NCP link number. The routines will be called with a pointer to the PDB of the message. When the routine returns the 1822 module will discard the message.

6. Internet Protocol

6.1 Identifier Assignment

When the Internet module gets a message to send, it generates a value for the "Identifier" (ID) field in the internet header. It does this by keeping a 16-bit counter called the ID counter. When it needs a new value it increments the counter by one and uses the result. The ID counter will not be initialized when the TAC is reloaded or restarted, to insure that the values are sequential.

6.2 Option Support

The Internet module will only actively support the "Error Report" IP option. None of the other currently defined options will require any action to be performed by the TAC.

6.3 Reassembly

When the Internet module gets a PDB which is a datagram fragment, it must reassemble it. It first looks at the fragments on the Internet-Reassembly queue to see if there are any other fragments of the same datagram. It does this by comparing the source address, ID, and protocol number of the two fragments. If

it does not find a match, it adds the new PDB to the queue. At this point, it also puts a time stamp in the PDB. This will be used to timeout unassembled fragments.

The actual reassembly process consists of adding the MBLKs of the new datagram to the list of MBLKs of the datagram on the queue. This is done using the "Offset", "Length" and the "Pointer to Data" fields in the MBLK. At this point in the processing of the fragment, the fragment consists of one or more MBLKs linked together. The IP header will always be in the first MBLK. The "Offset" fields in the MBLKs are all zero (there is no missing data in the new fragment itself). The "Length" fields contain the length of the IP data (not including the IP header) in each MBLK. The "Pointer to Data" fields point to the IP data in each MBLK.

The MBLKs of the new datagram are added to the datagram on the queue by comparing the "Fragment Offset" in the IP header of the new datagram to the "Fragment Offset" in the IP header of the datagram on the reassembly queue. This is done by taking the first MBLK on the list and adding the "Fragment Offset" from the IP header to the "Length" and "Offset" fields in the first MBLK. If this sum is greater than the "Fragment Offset" in the IP header of the new datagram, then the MBLKs of the new datagram should go before the first MBLK of the datagram in the original

fragment. If not, then they should be added in after. In this case, the comparative process is repeated with the rest of the MBLKs on the list. When the proper place is found, the new MBLKs are linked in and the fields of the new and old MBLKs are adjusted. If the new fragment overlaps the existing, then by adjusting the "Pointer to Data" and "Length" fields, the overlap can be skipped. This may result in one or more MBLKs being discarded.

When the datagram is completely reassembled, it can then be taken off the Reassembly queue and passed to the next protocol module.

6.4 Routing

The decision about where to send a datagram is twofold. If the destination host is on the same net as the TAC, then the datagram is sent directly to that host. If not, then it must be sent to a gateway.

The Internet module will maintain a table that will facilitate routing messages to hosts on other networks. The table is a list of all networks (256) and the gateways to get to the networks. This table, called the Network-Gateway table, will be initially loaded into the TAC and will be dynamically

maintained by the TAC.

When the Internet module needs to find an address, it looks in the Network-Gateway table to get the gateway address for the network it wants to send to. If it finds an address it uses it. If the entry for the network it wants is empty (i.e., no gateway address specified), then the Internet module will use an arbitrary gateway.

If the Internet module receives a Redirect message from a gateway, it will update the Network-Gateway table to indicate the correct gateway. This will insure that the Network-Gateway table contains current information.

If a gateway goes down during a connection the Internet module will clear that network's entry in the Network-Gateway table. It will then try an arbitrary gateway. If at a later time, the Internet module receives a Redirect message telling it to use a new gateway to get to a network, it will set that network's entry in Network-Gateway table to the new gateway's address.

6.5 Gateway to Gateway Messages

The Internet module will support the "Gateway to Gateway" protocol in a passive sense. If it receives a "Destination

Unreachable Packet" or a "Redirect Packet" it will take appropriate action. If it receives anything else, it will discard the message. In particular, if the Internet module receives a "Source Quench Packet" it will discard the message. This strategy is used due to the TAC's limited amount of buffering. The buffers would soon fill up because of the data not being acknowledged (in TCP). This will effectively limit the transmission.

6.6 Timeouts

Internet fragments will be discarded if they are not reassembled within 60 seconds. When a new fragment is reassembled into an existing one, the "Timeout" field in PDB of the existing fragment will be updated with the current time. This will, in effect, reset the timer for that fragment.

7. Transmission Control Protocol

7.1 Connection Opening and Closing

The TCP module will have a finite state machine which will be used for establishing and closing connections. The procedure to open a connection is to pass the required information (Destination address, socket, etc.) to the TCP module. It will

then run the finite state machine, which will set up the required data structures and open the connection. Closing the connection will be done in a similar way. The states of the finite state machine will be similar to what is described in "IEN-129, DOD Standard Transmission Control Protocol".

All TCP Port numbers assigned by the TAC will consist of the upper 8 bits set to the terminal number for the connection (1-64.) and the lower 8 bits set to 23. All connections made to or from the TAC will use this format.

7.2 Initial Sequence Number Assignment

The TCP module will maintain a 32-bit counter that will be used to generate Initial Sequence Numbers (ISN). The counter will be incremented by a constant value every time the H-316 clock ticks, which is every 25.6MS. The counter will be incremented by 64. This will then wrap around approximately every 4.55 hours. When the TCP module needs an ISN it reads the counter and gets a value.

7.3 Option Support

The TCP module will understand the format of all TCP options. It will support the "No-Operation" and "End of Option

List" options. It will not support the "Buffer Size" option. If it receives a "Buffer Size" option with anything greater than size one, it will not accept the connection but will reset it.

7.4 Urgent Data

When the TCP module receives a message with a valid Urgent Pointer, it sets a bit in the "Flag" word in the PDB and saves the offset to the end of the urgent data. When the next protocol module takes data out of the MBLK it will get an indication that the data it is getting is urgent.

Likewise, when the TCP module is given data to send, the protocol module supplying the data can include an indication that the data is urgent. The TCP module will include this information in all messages it sends until the urgent data is sent.

7.5 End of Letter Handling

The TCP module will not do anything special when it receives a message with End of Letter (EOL) set. The TCP module always presents all data to the next protocol module as soon as it is available. Consequently, no special handling is necessary.

The TCP module will accept data to be sent with an EOL indication. It will send this data with EOL set in the message. No additional data will be sent in the message. If the data is required to be retransmitted, it will be transmitted with EOL preserved.

7.6 Retransmissions

Data that is transmitted by the TCP module will be held until it has been ACKed by the remote host. If an ACK is not received for the data within three seconds it will be retransmitted. All data in the buffer that is not ACKed will be retransmitted (except if EOL is set; see previous section). If the data is still not ACKed for another seven seconds, retransmission will occur again. This procedure will continue using the series 3,7,15,15,30 . If there is still no ACK, then the user will be notified. The retransmissions will continue every 30 seconds until either the user closes the connection or the TCP module receives an ACK.

7.7 Acknowledgement and Window Strategy

The Acknowledgement (ACK) and Window parameters are used to control how much data the remote host can send to the TAC. The

TCP module will ACK data up to the limit it is willing to buffer for a connection. Data received after this limit is reached will be discarded. As the data is received, but before the next protocol module takes it out of the buffer, the window will be closed by the amount received. When the buffer limit is reached, the TCP module will not ACK any new data and will be advertising a zero window. When the next protocol module takes data out of the buffer, the window will be opened. This will allow the remote host to send more data.

When data is sent on the connection, the current ACK and Window values are always included in the same message. In the case where no data is being sent and the ACK and/or Window values have changed, a different strategy is used. When the ACK pointer is advanced, the TCP module will wait one second to see if there is data to send. If there has been no data sent for one second, then the ACK will be sent without data. A new window value will not be sent until the buffer is at least half empty. This strategy is designed to insure that the remote host sends blocks of data and to eliminate unnecessary retransmissions.

7.8 Sequencing

When the TCP module gets a data message for a connection, it must insure that the data is sequenced before it passes the data

to the next protocol module. The sequencing is done by comparing the sequence number of the message to the current "Left Window Edge" (LWE) and manipulation of the "offset", "length", and "pointer to data" fields of the MBLKs that make up the message. For each connection a list is maintained of MBLKs that are being sequenced. The MBLKs are in order but there may be missing data.

When the TCP module is ready to sequence the data, the message consists of a PDB, followed by one or more MBLKs linked together. The "pointer to data" field points to the actual TCP data. The "offset" fields are all zero because the data in the message is sequential relative to itself. The "length" field is the amount of data in each MBLK.

The sequencing is done by linking the MBLKs of the new message into the sequencing list for the connection for which the data message is intended. This is done via the following steps:

1. Subtract the LWE from the Sequence number of the message. The result is the offset from the LWE. Then set the "offset" field of the first MBLK to this result. If it is zero, this means that there is no missing data. Otherwise, the result is the amount of missing data.

2. Add the MBLKs to the existing list. If there are no MBLKs on the list, then the new MBLKs become the list. Otherwise, the insertion is done in the following steps:

    a) Find the position in which to add the new MBLK by adding together the "offset" and "length" of first MBLK in the list. If "offset" of new MBLK is less than the sum, then the new MBLK goes before the MBLK in the list. Otherwise, add