Software Kinetics

VOLUME 2
SOFTWARE DETAILED DESIGN DOCUMENT FOR THE INTERNETWORK GATEWAY PROJECT

Submitted to: C.R.C.
Ottawa, Ontario

SKL Document #1500-15-031.02.0
Copy #3 05 May 1988

VOLUME 2
SOFTWARE DETAILED DESIGN DOCUMENT FOR THE INTERNETWORK GATEWAY PROJECT

Submitted to: C.R.C.
Ottawa, Ontario

SKL Document #1500-15-031.02.0
Copy #3 05 May 1988

Vol.2 RELEASABLE
DOC-CR-RC-88-008

SOFTWARE DETAILED DESIGN DOCUMENT

FOR THE

INTERNETWORK GATEWAY PROJECT

VOLUME 2

Contract No. 36001-6-3535/02-ST

05 May 1988

Prepared for:

Communications Research Centre
Ottawa, Ontario

Prepared by:

Software Kinetics Ltd.
65 Iber Road, P.O. Box 680
Stittsville, Ontario Canada
K0A 3G0

SKL Document #1500-15-031.02.0

[illegible]98 17096
DL 9348751

QA
76.9
.S6
S6474
1988
v.2

Document Approval Sheet

for the

Internetwork Gateway Project

Document No: 1500-15-031.02.0

Document Name: Software Detailed Design Document for the Internetwork Gateway Project

Approvals	Signature	Date
Project Engineer:	[illegible] R. D. Bradford	5 May 1988
Project Manager:	[illegible] T. M. Symchych	May 5/88
Technical Authority:	[illegible] P. Labbe - CRC	7 June 88

Document Revision History

Revision	Description of Changes	Origin Date
01	New Document Issued	23 September 1987
02	Coding and Integration Revisions	05 May 1988

TABLE OF VOLUMES

TABLE OF CONTENTS

3.3.2 IP TLC Detailed Design

The IP TLC implements the IP protocol, the ICMP protocol, and packet filtering. The IP TLC accepts datagrams received by one of the network hardware interfaces and forwards the datagrams to the appropriate network interfaces, generating ICMP datagrams as required according to the ICMP protocol. The IP TLC also sends IP and ICMP statistics related to the transmission and reception of datagrams to the STAT TLC.

3.3.2.1 IP TLC Architecture

The IP TLC operations are controlled by the IP_CONTROL unit which, in turn, calls additional units to send the datagram to the next hop in the destination, to rebuild the route table, and to send the statistics to the STAT TLC. In sending the datagram, units are called to determine the route from the route table, filter out certain datagrams, maintain the time to live of the datagram, fragment and reassemble as required, forward the datagram, and update the statistics message buffer. The IP TLC consists of the following Units as shown in Figure 3-2. The Units are:

1) Calc_Cksum Unit - This unit verifies that the checksum specified for the data equals the calculated checksum.

2) Check_Ttl Unit - This unit decrements the time to live of the datagram and drops the datagram when the

#1500-15-031.02.0

```
                                    +------+
                                    | IP   |
                                    | TLC  |
                                    +--+---+
                                       |
      +--------------+----------------+-+-----------+-----------+--------------+
      |              |                 |            |           |              |
+-----+------+ +-----+-----+ +----+----+ +----+----+ +----+-----+              |
| Calc_Cksum | | Check_Ttl | | Do_Icmp | | Fail_IP | | Fragment |              |
+------------+ +-----------+ +---------+ +---------+ +----------+              |
                                                                               |
      +-----------+---------+-----------+--------------+-----------------------+
      |           |         |           |              |                       |
+-----+-----+ +---+---+ +---+---+ +-----+------+ +-----+-----+                 |
| Get_Route | | Htonl | | Htons | | IP_Control | | IP_Filter |                 |
+-----------+ +-------+ +-------+ +------------+ +-----------+                 |
                                                                               |
     +------------+-----------+---------+----------+---------------------------+
     |            |           |         |          |                           |
+----+----+ +-----+-----+ +---+---+ +---+---+ +----+----+                      |
| IP_Send | | Net_Class | | Ntohl | | Ntohs | | Options |                      |
+---------+ +-----------+ +-------+ +-------+ +---------+                      |
                                                                               |
      +-----------------+---------------+-----------+--------------------------+
      |                 |               |           |                          |
+-----+------+ +--------+---------+ +---+---+ +-----+------+                   |
| Reassemble | | Return_To_Sender | | Route | | Route_Init |                   |
+------------+ +------------------+ +-------+ +------------+                   |
                                                                               |
        +------------------+----------------+------------------+---------------+
        |                  |                |                  |               |
+-------+-------+ +--------+--------+ +-----+------+ +---------+--------+      |
| Send_IP_Stats | | Send_Icmp_Stats | | Time_Stamp | | Update_IP_Stats |       |
+---------------+ +-----------------+ +------------+ +-----------------+       |
                                                                               |
          +-----------------+--------------------------------------------------+
          |                 |
+---------+---------+ +-----+-----+
| Update_Icmp_Stats | | Ver_Cksum |
+-------------------+ +-----------+
```

Figure 3-2

time to live has expired.

3) IP_Control Unit - This unit waits for a message from the EGP TLC, for a datagram to arrive from a hardware interface, or for a timeout to occur indicating the statistics must be sent.

4) Do_Icmp Unit - This unit satisfies the ICMP requirements for ICMP messages received by the IGW.

5) Fail_Ip Unit - This unit initiates the return of an ICMP message to the originator of the datagram.

6) Fragment Unit - This unit divides a datagram that exceeds the maximum transmission unit of the next network.

7) Get_Route Unit - This unit searches the IP route table for the entry which identifies the next hop for the datagram.

8) IP_Filter Unit - This unit enforces fixed and variable packet filtering.

9) IP_Send Unit - This unit handles the processing of a datagram.

10) Net_Class Unit - This unit returns the class of the specified network address.

11) Options Unit - This unit satisfies IP specifications for the IGW requirements related to the variable options that may be found in the internet header.

12) Reassemble Unit - This unit assembles datagrams marked as fragmented into complete datagrams.

13) Return_To_Sender Unit - This unit returns the current message to the sender with an ICMP message of the appropriate type and code.

14) Route Unit - This unit initiates the processing of the datagram depending on the previously determined route.

15) Route_Init Unit - This unit uses the gateway, network, and EGP route tables to update the IP route table.

16) Send_Icmp_Stats Unit - This unit sends an ICMP message buffer to the STAT TLC and re-initializes the ICMP message buffer.

17) Send_Ip_Stats Unit - This unit sends an IP message buffer to the STAT TLC and re-initializes the IP message buffer.

18) Time_Stamp Unit - This unit obtains the current IGW time and modifies the time to indicate the time is not an IP standard time stamp.

19) Update_Icmp_Stats Unit - This unit creates a new entry in the ICMP message buffer and sends the ICMP message buffer when the buffer is full.

20) Update_Ip_Stats Unit - This unit creates a new entry in the IP message buffer and sends the IP message buffer when the buffer is full.

21) Ver_Cksum Unit - This unit verifies that the checksum for the data equals the calculated checksum.

22) Htonl Unit - This unit byte swaps a 32 bit integer.

23) Htons Unit - This unit byte swaps a 16 bit integer.

24) Ntohl Unit - This unit byte swaps a 32 bit integer.

25) Ntohs Unit - This unit byte swaps a 16 bit integer.

3.3.2.2 IP Global Data

The following is a list of global data items for the IP TLC:

1) Icmp_Stats_Table - This item is a Stats_Buffer containing a list of ICMP_Stats_Entry entries. The Icmp_Stats_Table accumulates statistics related to ICMP messages sent and received by the IP TLC.

2) Ip_Stats_Table - This item is a Stats_Buffer containing a list of IP_Stats_Entry entries. The Ip_Stats_Table accumulates statistics related to IP messages sent and received by the IP TLC.

3) Resources - This item is global to the IP TLC and maintains an array of datagram entries for the assembly of fragmented datagrams. Each datagram entry contains the following fields:

   1) R_msg_hdr - Header for message buffer to hold incoming fragments for reassembly.

   2) R_Timer - Timer to expire old reassembly entries.

   3) R_frb_Table - This field consists of an array of bytes for which each bit, starting at bit zero, is set to indicate the placement of fragmented datagram units of 8 bytes into the buffer of a complete datagram.

   4) R_Next - Pointer to next entry in reassembly list.

   5) R_Prev - Pointer and previous entry in reassembly list.

4) Hash_Table_A - This item is global to the IP TLC and consists of an array of pointers to the IP_Route_Table. Each pointer in the hash table corresponds to one class A network and the hash table points to the first occurrence of that network in the IP_Route_Table.

   Subsequent route entries of the same network in the IP_Route_Table are linked together in the IP_Route_Table. Hash_Table_A is capable of holding 128 pointers.

5) Hash_Table_B - This item is global to the IP TLC and consists of an array of pointers to the IP_Route_Table. Each pointer in the hash table corresponds to one class B network and the hash table points to the first occurrence of that network in the IP_Route_Table.

   Subsequent route entries of the same network in the IP_Route_Table are linked together in the IP_Route_Table. Hash_Table_B is capable of holding 256 pointers.

6) Hash_Table_C - This item is global to the IP TLC and

consists of an array of pointers to the IP_Route_Table. Each pointer in the hash table corresponds to one class C network and the hash table points to the first occurrence of that network in the IP_Route_Table.

Subsequent route entries of the same network in the IP_Route_Table are linked together in the IP_Route_Table. Hash_Table_C is capable of holding 128 pointers.

7) IP_Stat_Timeout - This global IP input is a 32 bit integer which maintains the periodic time interval for the update of the IP statistics.

8) ICMP_Stat_Timeout - This global IP input is a 32 bit integer which maintains the periodic time interval for the update of the ICMP statistics.

9) IP_Stat_Hdr - This global IP output is a Message_Header for a message containing an IP_Stats_Table.

10) ICMP_Stat_Hdr - This global IP output is a Message_Header for a message containing an ICMP_Stats_Table.

11) IP_Stats_Table - This global IP output is a Stats_Buffer for a message containing a list of IP_Stats_Entry records to be sent to the STAT TLC.

12) ICMP_Stats_Table - This global IP output is a Stats_Buffer for a message containing a list of ICMP_Stats_Entry records to be sent to the STAT TLC.

13) ASSEMBLY_RESOURCES - This constant defines 100 datagram entries for the reassembly.

14) NO_CLASS (0) - indicates the network class is undefined.

15) CLASS_A (1) - indicates a class A network.

16) CLASS_B (2) - indicates a class B network.

17) CLASS_C (2) - indicates a class C network.

18) CONTINUE (0) - indicates processing can continue

19) RESEND (1) - indicates processing is to be attempted again.

20) NEXT_PACKET (2) - indicates that the all processing is completed.

21) IO_NOP (0x01) - identifies the no option Option.

22) IO_SECURITY (0x82) - identifies the security Option.

23) IO_TS (0x44) - identifies the timestamp Option.

24) IO_SSRR (0x89) - identifies the strict source route Option.

25) IO_LSRR (0x83) - identifies the loose source route Option.

26) IO_RRP (0x07) - identifies the record route Option.

27) IO_SATID (0x88) - identifies the stream identifier Option.

28) PROT_UNSUPPORTED (1) - protocol unsupported.

29) HOST_UNREACHABLE (2) - host unreachable.

30) TIME_EXCEEDED (3) - time is exceeded.

31) DONT_FRAGMENT (4) - dont fragment.

32) SOURCE_ROUTE_FAILED (6) - source route failed.

33) ICMP_ECHO (8) - the ICMP ECHO message.

34) ICMP_EREPLY (0) - the ICMP ECHO REPLY message.

35) ICMP_TIMESTAMP (13) - the ICMP TIMESTAMP message.

36) ICMP_TREPLY (14) - the ICMP TIMESTAMP REPLY message.

37) ICMP_UREACH (3) - the ICMP DESTINATION UNREACHABLE message.

38) ICMP_TIMEXCEED (11) - the ICMP TIME EXCEEDED message.

39) ICMP_PPROBLEM (12) - the ICMP PARAMETER PROBLEM message.

40) ICMP_SQUENCH (4) - the ICMP SOURCE QUENCH message.

41) ICMP_REDIRECT (5) - the ICMP REDIRECT message.

42) ICMP_IREQUEST (15) - the ICMP INFORMATION REQUEST message.

43) ICMP_IREPLY (16) - the ICMP INFORMATION REPLY message.

3.3.2.3 IP LLC Design

There are no lower level components defined for the IP TLC.

3.3.2.4 IP Units

This section contains the descriptions of the IP TLC units.

3.3.2.4.1 Calc_Cksum Unit

Calc_Cksum verifies that the checksum specified for the data equals the calculated checksum.

3.3.2.4.1.1 Inputs

The following inputs are required by the Calc_Cksum unit:

1) Data - This input parameter is a pointer to a data set for which the checksum is specified and the checksum is to be calculated.

2) Count - This input parameter is a 32 bit integer containing the number of 16 bit words in Data.

3.3.2.4.1.2 Outputs

The following outputs are returned by the Calc_Cksum unit:

1) Cksum - This output function value is a 32 bit integer containing the checksum calculated for Data.

3.3.2.4.1.3 Local Data

No local data is defined for the Calc_Cksum unit.

3.3.2.4.1.4 Processing

Cksum = the 16 bit one's complement of the one's complement sum of all 16 bit words
Return the Cksum

3.3.2.4.1.5 Limitations

No limitations are defined for the Calc_Cksum unit.

3.3.2.4.2 Check_Ttl Unit

Check_Ttl decrements the time to live and either generates an ICMP for time exceeded or recalculates the checksum.

3.3.2.4.2.1 Inputs

The following inputs are required by the Check_Ttl unit:

1) Data_Start - This input parameter is a pointer to the set of data for which the checksum is to be calculated.

2) Data_Count - This input parameter is a 32 bit integer containing the number of 16 bit words for which the checksum is to be calculated.

3) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

4) Rc_Datagram - This global IP input is a Pointer to a Dgram_Message.

3.3.2.4.2.2 Outputs

The following output is returned by the Check_Ttl unit:

1) Status - This output function value is a 32 bit integer which indicates the amount of time to live for the datagram:

   CONTINUE - The time to live has not expired
   NEXT_PACKET - The time to live has expired

2) Rc_Msg_Hdr - This global IP output is a Message_Header for a message containing a Rc_Datagram.

3) Rc_Datagram - This global IP output is a Pointer to a Dgram_Message.

3.3.2.4.2.3 Local Data

No local data is defined for the Check_Ttl unit.

3.3.2.4.2.4 Processing

```
Decrement the Rc_Datagram Internet_Header time to live field TTL
If the time to live TTL field is zero
    Call Fail_Ip( TIME_EXCEEDED )
    Set the Status to NEXT_PACKET
Else
    Set the Rc_Datagram checksum Sum to zero
    Sum = Calc_Cksum( Data_Start, Data_Count )
    Set the status to CONTINUE
Endif
Return the status
```

3.3.2.4.2.5 Limitations

No limitations are defined for the Check_Ttl unit.

3.3.2.4.3 Ip_Control Unit

Ip_Control forwards datagrams while maintaining the IP route table and managing the collection and submission of IP and ICMP statistics to the STAT TLC.

3.3.2.4.3.1 Inputs

The following inputs are required by the Ip_Control unit:

1) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global IP input is a Pointer to a Dgram_Message.

3) IP_Stat_Timeout - This global IP input is a 32 bit integer which maintains the periodic time interval for the update of the IP statistics.

4) ICMP_Stat_Timeout - This global IP input is a 32 bit integer which maintains the periodic time interval for the update of the ICMP statistics.

3.3.2.4.3.2 Outputs

The following outputs are returned by the Ip_Control unit:

1) IP_Stat_Hdr - This global IP output is a Message_Header for a message containing an IP_Stats_Table.

2) ICMP_Stat_Hdr - This global IP output is a Message_Header for a message containing an ICMP_Stats_Table.

3) P_Stats_Table - This global IP output is a Stats_Buffer for a message containing a list of IP_Stats_Entry records to be sent to the STAT TLC.

4) ICMP_Stats_Table - This global IP output is a Stats_Buffer for a message containing a list of ICMP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.3.3 Local Data

No local data is defined for the Ip_Control unit.

3.3.2.4.3.4 Processing

```
/* Initialize IP message header, statistic table and timer */

Call Message_Get( address of IP_Stat_Hdr )
If No Error
    Set the IP_Stat_Hdr M_Offset to the number of bytes in the
     IP_Stats_Table Descriptor field
    Set the IP_Stat_Hdr M_Length to the IP_Stat_Hdr M_Offset
    Set the IP_Stat_Hdr M_From to the sending process qid IP_Q_ID
    Set the IP_Stat_Hdr M_Qid to the quid to deliver to STAT_Q_ID
    Set the IP_Stats_Table Descriptor Entries to zero
    Set the IP_Stats_Table Descriptor Operation to UPD
    Set the IP_Stats_Table Descriptor Statistic to IP
    Set the IP_Stats_Timeout to the periodic ip statistics interval
```

```
Else
    Mark IP STATS Table as invalid
    SET IP_Stats_Timeout to STAT_NOBUF_TIMEOUT
Endif

/* Initialize ICMP message header, statistics table and timer */

Call Message_Get( address of ICMP_Stat_Hdr )
If No Error
    Set the ICMP_Stat_Hdr M_Offset to the number of bytes in the
     ICMP_Stats_Table Descriptor field
    Set the ICMP_Stat_Hdr M_Length to the ICMP_stat_Hdr M_Offset
    Set the ICMP_Stat_Hdr M_From to the sending process qid IP_Q_ID
    Set the ICMP_Stat_Hdr M_Qid to the quid to deliver to STAT_Q_ID
    Set the ICMP_Stats_Table Descriptor Entries to zero
    Set the ICMP_Stats_Table Descriptor Operation to UPD
    Set the ICMP_Stats_Table Descriptor Statistic to ICMP
    Set the ICMP_Stats_Timeout to the periodic icmp statistics interval
Else
    Mark ICMP Stats Table as Invalid
    Set ICMP_Stats_Timeout to STAT_NOBUF_TIMEOUT
Endif

Call Route_Init
Status = Open_Message_Queue( IP_Q_ID, IP_Q_SIZE )
If status is not NOERROR
    Call Oi_Messages( status )
    Call exit ()
Endif
    Start timer = Get_Time
    Loop forever
        Status = Message_Receive( IP_QUEUE, Rc_msg_hdr )
        If Status is NO ERROR
            If OFFSET Equals ROUTE_INIT
                Call Message_discard(Rc_msg_hdr)
                Call IP_Route_Init()
            Else
                Set Rc_datagram to Address of Rc_msg_hdr data
                Call IP_Send()
            Endif
        Else If Status is M_QEMPTY
            Call er_wait_timeout( MESG_Arrive,
              Min(IP_Stat_Timeout, ICMP_Stat_Timeout))
            Cur_Time = Get_Time()
            Decrement Start_Time by Cur_Time
            Add Start_Time to IP_Stat_Timeout
            Add Start_Time to ICMP_Stat_Timeout
            Move Cur_Time to Start_Time
            If IP_Stat_Timeout <=0
```

```
                Set IP_Stat_Timeout to IP_STAT_TIMEOUT
                Call Send_IP_STATS()
            Endif
            If ICMP_Stat_Timeout <=0
                Set ICMP_Stat_Timeout to ICMP_STAT_TIMEOUT
                Can Send_ICMP_Stats()
            Endif
        Endif
    Endloop
Endif
```

3.3.2.4.3.5 Limitations

No limitations are defined for the Ip_Control unit.

3.3.2.4.4 Do_Icmp Unit

Do_Icmp satisfies the ICMP requirements for ICMP messages received by the IGW.

3.3.2.4.4.1 Inputs

The following inputs are required by the Do_Icmp unit:

1) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global IP input is a Pointer to a Dgram_Message containing an ICMP message which is to be replaced by a reply.

3.3.2.4.4.2 Outputs

The following outputs are returned by the Do_Icmp unit:

1) Rc_Msg_Hdr - This global IP output is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global IP output is a Pointer to a Dgram_Message which contains the ICMP message reply to the original ICMP message.

3.3.2.4.4.3 Local Data

The following local data is defined for the Do_Icmp unit:

1) ICMP_Datagram - This local data item contains the fields required to form a ICMP message. The ICMP_Datagram contains the following fields:

    1) Type - This field is a byte which contains an ICMP types.

    2) Code - This field is a byte which specifies a version of the ICMP message type.

    3) Cksum - This field is a 16 bit integer which contains the ICMP header checksum.

    4) Ident - This field is a 16 bit integer.

    5) Seq - This field is a 16 bit integer.

    6) Data - This field is a variable length set of bytes.

3.3.2.4.4.4 Processing

```
Set a pointer to the Rc_Datagram Data section which equivalences the
 Rc_Datagram Data section to ICMP_Datagram structure
Set byte count to the total Rc_Datagram length LEN
If the byte count is odd
    Add a zero byte to the end to the Rc_Datagram
    Increment the byte count
Endif

Checksum = Ver_Cksum( address of ICMP datagram,
 byte count / 2 less the Rc_Datagram internet header length VHL * 2
If checksum is not equal CONTINUE
    Return 0
Endif

Case ICMP_Datagram Type
    ICMP_ADRMASK:   Reset ICMP_Datagram Type to ICMP_RADRMASK
                    Set the ICMP_Datagram Data to the IGW gateway mask
                    Increment byte count by four
                    Increment Rc_msg_hdr.M_Length by 4
                    Increment IP Packet Length by 4

    ICMP_ECHO:      Reset ICMP_Datagram Type to ICMP_EREPLY
                    Break

    ICMP_TIMESTAMP: Set the ICMP_Datagram Data bytes 4 - 7 = Time_Stamp
                    Reset the ICMP_Datagram Type to ICMP_TREPLY
                    Break

    Default:        Return 0
Endcase
Swap SRL & DST Addresses
Recompute Checksum
Return 1
```

3.3.2.4.4.5 Limitations

No limitations are defined for the Do_Icmp unit.

3.3.2.4.5 Fail_Ip Unit

Fail_Ip initiates the return of an ICMP message to the originator of the datagram.

3.3.2.4.5.1 Inputs

The following inputs are required by the Fail_Ip unit:

1) Reason - This input parameter is a 32 bit integer indicating the reason for the ICMP message.

3.3.2.4.5.2 Outputs

No outputs are returned by the Fail_Ip unit.

3.3.2.4.5.3 Local Data

No local data is defined for the Fail_Ip unit.

3.3.2.4.5.4 Processing

```
Case Reason
    TIME_EXCEEDED:     Call Return_To_Sender( ICMP_TIMEXCEED, 0, 0 )

    PROT_UNSUPPORTED:  Call Return_To_Sender( ICMP_UREACH, 2, 0 )

    HOST_UNREACHABLE:  Call Return_To_Sender( ICMP_UREACH, 1, 0 )

    DONT_FRAGMENT:     Call Return_To_Sender( ICMP_UREACH, 4, 0 )

    SOURCE_ROUTE_FAILED: Call Return_To_Sender( ICMP_UREACH, 5, 0 )

    PARAMETER_PROBLEM:   Call Return_To_Sender( ICMP_PPROBLEM, 0, 0 )
Endcase
Return
```

3.3.2.4.5.5 Limitations

No limitations are defined for the Fail_Ip unit.

3.3.2.4.6 Fragment Unit

Fragment divides a datagram that exceeds in size the maximum transmission unit of the next network. Each fragmentation datagram contains the maximum allowable size as specified by the maximum transmission unit, except for the last fragmentation datagram which contains the rest of the original datagram.

3.3.2.4.6.1 Inputs

The following inputs are required by the Fragment unit:

1) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global IP input is a Pointer to a Dgram_Message.

3) Net_Table - This global data input is a table of network entries.

4) Network - This input parameter is a pointer to a NET_Table entry which contains the interface quid to which the datagram is sent.

3.3.2.4.6.2 Outputs

No outputs are returned by the Fragment unit.

3.3.2.4.6.3 Local Data

The following local data is defined for the Fragment unit:

1) Fr_Msg_Hdr - This local data item is a Message_Header for a message containing a Fr_Datagram.

2) Fr_Datagram - This local data item is an Dgram_Message.

3) Fr_Offset - This local data item is a 32 bit integer which contains the offset of the fragmented datagram from the start of the original unfragmented datagram in units of eight bytes.

4) Surplus - This local data item is a 32 bit integer which contains the number of bytes yet to be assigned to a fragmented datagram.

5) Option_Start_Adr - This local data item is a pointer to the Rc_Datagram Options section and is used to step through the options in the Rc_Datagram.

6) New_Start_Adr - This local data item is a pointer to the Fr_Datagram Options section and is used to step through for the placement of Rc_Datagram options in the Fr_Datagram.

7) Option_End_Adr - This local data item is a pointer to the end of the last option in the Rc_Datagram.

3.3.2.4.6.4 Processing

```
If the Rc_Datagram OFF field and IP_DF indicate dont fragment
    Call Fail_Ip( DONT_FRAGMENT )
    Return
Endif

Fr_Offset in units of eight bytes = ( Net_MTU of
 Net_Table entry pointed to by Network less the length VHL*4 ) / 8
Surplus bytes after the first block = Ntohs( Rc_Datagram LEN field )
 less the Rc_Datagram length VHL*4 less Fr_Offset*8
If Surplus >0
    Set Rc_Datagram LEN field = Htons( Fr_Offset*8 + the length field
     VHL*4 )
    Increase the Fr_Offset by Ntohs( Rc_Datagram OFF field )
    Set the Rc_Datagram OFF field more fragment bit = IP_MF
    Set the Rc_Datagram checksum Sum to zero
    Rc_Datagram Sum = Calc_Cksum( address of Rc_Datagram Internet_Header,
     the length VHL field *2 )
Endif

While the Surplus bytes exceeds zero
    Call Message_Get( address of Fr_Msg_Hdr )
    If Error
         Call message_discard(Rc_msg_hdr)
         Return
    Endif
    Set the Fr_Datagram Out_Dest to the Rc_Datagram Out_Dest
    Set the Fr_Datagram Internet_Header fields type of service TOS,
     identifier ID, time to live TTL, Prot, Dst IP address, and Src
     IP address from the Rc_Datagram Internet_Header

    /* Copy selected options to the fragmented datagram */

    Set a New_Start_Adr to the Fr_Datagram Data field
    Set a Option_Start_Adr to the Rc_Datagram Data field
    Set the Option_End_Adr = ( the length VHL field * 4 ) - ( 5*4 )
    While Option_Start_Adr is less than the Option_End_Adr
        If the option type at the first byte of Option_Start_Adr
         equals the IO_NOP option
            Increment the Option_Start_Adr by one byte
            Continue
        Endif

        If bad option length
            Call message_discard(Fr_msg_hdr)
            Call Fail_IP(PARAMETER_PROBLEM)
            Return
        Endif

         If most significant bit in the option type at the first byte
         of Option_Start_Adr is set to one
            Obtain the option size from the next byte
            Copy the option size bytes from Option_Start_Adr to
             New_Start_Adr and increment both by the number of bytes
        Endif
    Endwhile
    Add a series of zero bytes to the Fr_Datagram at New_Start_Ptr
     such that the Fr_Datagram Internet_Header length LEN field ends
     on a 32 bit boundary
    Set the Fr_Datagram version and header length VHL field and the
     offset OFF field to Htons( Fr_Offset )

    /* Set the more fragment bit and the offset to zero in the first
       datagram, the more fragment bit set and the offset > 0 for
       subsequent datagrams except for the last datagram which has
       the more fragment bit set to zero.                            */

    Set a data length to ( Net_MTU of Net_Table entry pointed to by
     Network less the Fr_Datagram length VHL*4   )
    If ( ( the data length ) / 8 ) * 8 is less than the Surplus
        Set the Fr_Datagram length LEN = Htons( the data length less
         the Internet_Header length VHL*4 )
        Set the more fragment bit in the Fr_Datagram offset OFF field
         to IP_MF
        Increase the Fr_Offset by ( data length / 8 )
        Decrease the Surplus by the data length
    Else
        Set the Fr_Datagram length LEN field = Htons( the Fr_Datagram
         Internet_Header length VHL*4 plus the Surplus )
        Set the Surplus to zero
        Set the more fragment bit in Fr_Datagram OFF to zero
    Endif
    Copy the data from the Rc_Datagram to the Fr_Datagram
    Set the Internet_Header checksum Sum to zero
    Sum = Calc_Cksum( address of Fr_Datagram Internet_Header,
     the Fr_Datagram Internet_Header length VHL * 2 )

    /* Set the interface quid over which the fragmented datagram
       is to be sent and send the datagram                      */
    Call IP_update_IP_Stats(Fr_datagram IP header)
    Increment current_IF
    If End of Interface List reached
       Set Current_IF equals 0
    Endif
    Call Message_Send (Queue ID referenced by Current_IF,
     fr_msg_hdr)
    If Error
        Call message_discard(Fr_msg_hdr)
        Call message_discard(Rc_msg_hdr)
        Return
    Endif
Endwhile

/* Send the remaining datagram on the already determined interface */

Call Update_IP_STATS(Rc_datagram IP hdr)
Increment_Current_IF
If end of Interface List reached
    Set Current_IF =0
Endif
Call Message_Send(Queue ID Referenced by Current_IF, Rc_msg_hdr)
If Error
    Call message_discard(Rc_msg_hdr)
Endif
Return
```

3.3.2.4.6.5 Limitations

No limitations are defined for the Fragment unit.

3.3.2.4.7 Get_Route Unit

Get_Route searches the IP_Route_Table for the entry which identifies the next hop for the datagram.

3.3.2.4.7.1 Inputs

The following inputs are required by the Get_Route unit:

1) Redirect - This input parameter is a 16 bit integer which indicates whether redirection has been enabled or not.

2) IP_Route_Table - This global data input is a table of routing entries required by the IP TLC.

3) Hash_Table_A - This global IP input is an array of pointers to the IP_Route_Table for class A networks.

4) Hash_Table_B - This global IP input is an array of pointers to the IP_Route_Table for class B networks.

5) Hash_Table_C - This global IP input is an array of pointers to the IP_Route_Table for class C networks.

6) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

7) Rc_Datagram - This global IP input is a Pointer to a Dgram_Message.

3.3.2.4.7.2 Outputs

The following output is returned by the Get_Route unit:

1) Network - This output function value is a 32 bit integer which contains either a pointer to a NET_Table entry as determined from the IP_Route_Table, or one of the route flags:

   DEBUG
   ROUTE_FOR_ME
   ROUTE_NO_ROUTE

3.3.2.4.7.3 Local Data

No local data is defined for the Get_Route unit.

3.3.2.4.7.4 Processing

```
Case Net_Class( first byte of the Rc_Datagram Dst IP address )
    CLASS_A:  Set a pointer to the IP_Route_Table entry which is
               pointed to by the Hash_Table_A entry indexed by
               the most significant byte of the destination address of
               the datagram
    CLASS_B:  Set a pointer to the IP_Route_Table entry which is
               pointed to by the Hash_Table_B entry indexed by
               the second most significant byte of the destination
               address of the datagram
    CLASS_C:  Set a pointer to the IP_Route_Table entry which is
               pointed to by the Hash_Table_C entry indexed by
               the most significant byte of the destination address of
               the datagram
    Default:   Return ROUTE_NO_ROUTE
Endcase

For two iterations through the IP_Route_Table
    For the set of linked IP_Route_Table entries starting at the
     pointer set to the IP_Route_Table

        If the network part of Rc_Datagram Dst as determined by
         the IP_Route_Table entry Mask equals the Net in the
         IP_Route_Table entry
            If the IP_Route_Table entry Flag is marked IP_NOROUTE
                Return ROUTE_NO_ROUTE
            Endif

        If the IP_Route_Table entry Flag is marked IP_TCPONLY
                Case on Rc_Datagram Protocol
                    Default:  Set the Rc_Datagram equal the
                               IP_Route_Table entry Gw
                              Return address of the NET_Table entry as
                               Network which is pointed to by the
                               IP_Route_Table pointer
                    IH_ICMP:  Continue
                    IH_EGP:   Continue
                Endcase
        Endif

        If the IP_Route_Table entry is marked IP_DIRECT
            If the Rc_Datagram Dst equals the IP_Route_Table Gw
                Return ROUTE_FOR_ME
            Endif
            Set the Rc_Datagram Out_Dest to the Rc_Datagram Dst
        Else
                Set the Rc_Datagram Out_Dest to the IP_Route_Table Gw
        Endif
                Set Network to that entry in the NET_Table pointed to
                 by the IP_Route_Table entry
                If the redirect is enabled and the Rc_Datagram Protocol
                 is not IH_ICMP and ( Rc_Datagram Src and Mask is equal
                 to the Net and Mask )
                    Call Return_To_Sender( ICMP_REDIRECT, 1,
                     the Rc_Datagram Out_Dest )
                Endif
                Return Network
        Endif
        Endfor
        Reset pointer to the Hash_Table_A entry zero to determine
         a route based on the default GW_Table entry
    Endfor
Return ROUTE_NO_ROUTE
```

3.3.2.4.7.5 Limitations

No limitations are defined for the Get_Route unit.

3.3.2.4.8 Htonl Unit

Htonl swaps two sets of bytes of a 32 bit word for data going from host to network.

3.3.2.4.8.1 Inputs

The following inputs are required by the Htonl unit:

1) Data - This input parameter is a 32 bit integer to be byte swapped.

3.3.2.4.8.2 Outputs

The following outputs are returned by the Htonl unit:

1) Data - This output function value is the 32 bit data with both sets of bytes swapped.

3.3.2.4.8.3 Local Data

No local data is defined for the Htonl unit.

3.3.2.4.8.4 Processing

Swap both sets of bytes in Data Return Data

3.3.2.4.8.5 Limitations

No limitations are defined for the Htonl unit.

3.3.2.4.9 Htons Unit

Htons swaps a set of bytes of a 16 bit word for data going from host to network.

3.3.2.4.9.1 Inputs

The following inputs are required by the Htons unit:

1) Data - This input parameter is a 16 bit integer to be byte swapped.

#1500-15-031.02.0

3.3.2.4.9.2 Outputs

The following outputs are returned by the Htons unit:

1) Data - This output function value is the 16 bit data with the set of bytes swapped.

3.3.2.4.9.3 Local Data

No local data is defined for the Htons unit.

3.3.2.4.9.4 Processing

Swap both sets of bytes in Data
Return Data

3.3.2.4.9.5 Limitations

No limitations are defined for the Htons unit.

3.3.2.4.10 Ip_filter Unit

Ip_filter enforces the fixed packet filtering such that a datagram is not sent to the originating network via the IGW, and variable packet filtering as specified by the packet filtering table.

3.3.2.4.10.1 Inputs

The following inputs are used by the ip_filter unit:

1) p_key - This input parameter is a 32 bit integer containing the primary key for packet filtering.

2) s_key - This input parameter is a 32 bit integer containing the secondary key for packet filtering.

3) Pf_table - This global data input is a table of variable packet filters.

3.3.2.4.10.2 Outputs

The following outputs are produced by the ip_filter unit:

1) status - This return value indicates the result of the packet filtering.

3.3.2.4.10.3 Local Data

No local data is defined for the ip_filter unit.

3.3.2.4.10.4 Processing

```
Convert p_key to host byte order
Convert s_key to host byte order
If p_key < s_key
    Swap p_key and s_key
Endif
If key match in quick lookup list
    Return CONTINUE
Endif
If p_key and s_key are on the same network
    Return NEXT_PACKET
Endif
Perform binary search for HOST - HOST match
If no match
    Perform binary search for HOST - NET match
    If no match
        Perform binary search for HOST - ANY match
        If no match
            Perform binary search for NET - HOST match
            If no match
                Perform binary search for NET - NET match
                If no match
                    Perform binary searcg for NET - ANY match
                    If no match
```

```
                        Perform binary search for ANY - HOST match
                        If no match
                            Perform binary search for ANY - NET match
                            If no match
                                Add p_key and s_key to quick lookup list
                                Return CONTINUE
                            Endif
                        Endif
                    Endif
                Endif
            Endif
        Endif
    Endif
Endif
If matched entry mode is PF_ALLOW
    Add p_key and s_key to quick list
    Return CONTINUE
Endif
Return NEXT_PACKET
```

3.3.2.4.10.5 Limitations

No limitations are defined for the ip_filter Unit.

3.3.2.4.11 Ip_Send Unit

Ip_Send handles the processing of a datagram.

3.3.2.4.11.1 Inputs

The following inputs are required by the Ip_Send unit:

1) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global IP input is an Dgram_Message.

3.3.2.4.11.2 Outputs

No outputs are returned by the Ip_Send unit.

3.3.2.4.11.3 Local Data

The following local data is defined for the Ip_Send unit:

1) Status - This local data item is a 32 bit integer which is used to indicate the success of a particular operation:

   CONTINUE - processing on the datagram may continue
   NEXT_PACKET - processing on the datagram is finished
   RESEND - attempt to send the datagram again

2) Network - This local data item is a 32 bit integer which contains either a pointer to the network table or a route flag:

   ROUTE_FOR_ME
   ROUTE_NO_ROUTE

3) Option_Start_Adr - This local data item is a pointer in the Rc_Datagram Data fields which steps through the options.

4) Option_End_Adr - This local data item is a pointer in

the Rc_Datagram Options field which indicates the location of the last option.

5) Net_Table - This global data input is a table of network entries.

3.3.2.4.11.4 Processing

```
Verify the Rc_Datagram Internet_Header VHL version and length
Status = Ver_Cksum( address of the Rc_Datagram,
 the Rc_Datagram Internet_Header Sum, the length in VHL * 2 )
If Status is NEXT_PACKET
Call message_discard(Re_msg_hdr)
    Return
Endif
Status = Check_Ttl()
If Status is NEXT_PACKET
    Return
Endif
Set status to RESEND
While the Status is RESEND

    /* Find the corresponding network table entry.  The next hop is
       set by Get_Route                                            */

    Network = Get_Route( redirection enabled )

    /* Process the options in the IP datagram */

    Set the Option_Start_Adr to the address of the Rc_Datagram Options
     field
    Set the Option_End_Adr to the address of the Rc_Datagram plus the
     Options field byte count calculated as ( Rc_Datagram VHL*4 - 5*4 )
    While Option_Start_Adr is less than the Option_End_Adr
        Status = Options( Option_Start_Adr, Network )
        If Status is NEXT_PACKET
            Return
        Endif
        If Bad option Length
            Call fail_IP (PARAMETER_PROBLEM)
            Return
        Endif
        Increment the Option_Start_Adr by the byte count
         at Option_Start_Adr(1)
    Endwhile
```

#1500-15-031.02.0

```
    /* Enforce fixed and variable packet filtering */

    Status = Ip_Filter( Rc_Datagram Src, Rc_Datagram Dst )
    If Status is NEXT_PACKET
    Call Message_discard (RL_msg_hdr)
        Continue
    Endif

    /* Forward the datagram */

    Status = Route( Network )
    If Status is not CONTINUE
        Continue
    Endif

    /* Decrement the time to live */

    Status = Check_Ttl( address of the Rc_Datagram, the length in
     VHL*2 )
    If Status is NEXT_PACKET
        Continue
    Endif

    /* Set the interface quid */

    If NTOHS(total length of datagram)>NET_MTU)
        Call Fragment(network)
    Else
        Call Update_IP_STATS (Rc_datagram IP header)
        Increment Current_IF
        Ifend of Interface List
            Set Current_If equals 0
    Endif
        Call message_send(Queue IP or Current_Interfacing
          RC_msg_hdr)
    IF ERROR
        Call message_discard(Rc_msg_hdr)
    Endif
Endif


Endwhile
Return
```

3.3.2.4.11.5 Limitations

No limitations are defined for the Ip_Send unit.

3.3.2.4.12 Net_Class Unit

Net_Class returns a class of the specified network address.

3.3.2.4.12.1 Inputs

The following inputs are required by the Net_Class unit:

1) Network - This input parameter is a byte containing the network address from which the class is to be determined.

3.3.2.4.12.2 Outputs

The following output is returned by the Net_Class unit:

1) Class - This output function value is a 32 bit integer identifying the class to which the network belongs:

CLASS_A -	a class A network address
CLASS_B -	a class B network address
CLASS_C -	a class C network address
NO_CLASS -	the class is none of the above

#1500-15-031.02.0

3.3.2.4.12.3 Local Data

No local data is defined for the Net_Class unit.

3.3.2.4.12.4 Processing

```
If the most significant bit in Network is zero
    Return the Status set to CLASS_A
Endif
If the two most significant bit in Network are one and zero respectively
    Return the Status set to CLASS_B
Endif
If the three most significant bit in Network are one, one and zero
 respectively
    Return the Status set to CLASS_C
Endif
Return the Status set to NO_CLASS
```

3.3.2.4.12.5 Limitations

No limitations are defined for the Net_Class unit.

3.3.2.4.13 Ntohl Unit

Ntohl swaps two sets of bytes of a 32 bit word for data going from network to host.

3.3.2.4.13.1 Inputs

The following inputs are required by the Ntohl unit:

1) Data - This input parameter is a 32 bit integer to be byte swapped.

3.3.2.4.13.2 Outputs

The following outputs are returned by the Ntohl unit:

1) Data - This output function value is the 32 bit data with both sets of bytes swapped.

3.3.2.4.13.3 Local Data

No local data is defined for the Ntohl unit.

3.3.2.4.13.4 Processing

Swap both sets of bytes in Data Return Data

3.3.2.4.13.5 Limitations

No limitations are defined for the Ntohl unit.

3.3.2.4.14 Ntohs Unit

Ntohs swaps a sets of bytes of a 16 bit word for data going from network to host.

3.3.2.4.14.1 Inputs

The following inputs are required by the Ntohs unit:

1) Data - This input parameter is a 16 bit integer to be byte swapped.

3.3.2.4.14.2 Outputs

The following outputs are returned by the Ntohs unit:

1) Data - This output function value is the 16 bit data with the set of bytes swapped.

3.3.2.4.14.3 Local Data

No local data is defined for the Ntohs unit.

3.3.2.4.14.4 Processing

Swap both sets of bytes in Data
Return Data

3.3.2.4.14.5 Limitations

No limitations are defined for the Ntohs unit.

3.3.2.4.15 Options Unit

Options satisfies IP specifications for the IGW requirements related to the variable options that may be found in the internet header. Any options in the datagram are found immediately following the datagram destination address.

3.3.2.4.15.1 Inputs

The following inputs are required by the Options unit:

1) Option_Adr - This input parameter is a pointer to the current option in the Rc_Datagram.

2) Network - This input parameter is a 32 bit integer which contains a pointer to the NET_Table as determined by the IP_Route_Table, or one of the route flags:

   ROUTE_FOR_ME
   ROUTE_NO_ROUTE

3) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing an Rc_Datagram.

4) Rc_Datagram - This global IP input is a Dgram_Message

5) NET_Table - This global input is a table of network entries

3.3.2.4.15.2 Outputs

1) Network - This output parameter is a 32 bit integer which contains a pointer to the NET_Table as determined by the IP_Route_Table or one of the route flags:

   ROUTE_FOR_ME
   ROUTE_NO_ROUTE

2) Rc_Msg_Hdr - This global IP output is a Message_Header for a message containing a Rc_Datagram.

3) Rc_Datagram - This global IP output is an Dgram_Message.

4) Status - This output function value is a 32 bit integer which indicates the success of completing the options:

   CONTINUE - the option is successfully completed

NEXT_PACKET - processing is completed for this datagram

3.3.2.4.15.3 Local Data

No local data is defined for the Options unit.

3.3.2.4.15.4 Processing

```
Case the first Option_Adr byte is the option type
    IO_NOP:         Set status to CONTINUE

    IO_SECURITY:    Set status to CONTINUE

    IO_TS:          Set status to CONTINUE

    IO_SSRR:        Set status to CONTINUE
                    If Network is ROUTE_FOR_ME
                        If an IP address is not in the option; the
                         option pointer Option_Adr(2)+3 exceeds the
                         option length Option_Adr(1)
                            Call Fail_Ip( PARAMETER_PROBLEM )
                            Set status to NEXT_PACKET
                        Else
                            Set the Rc_Datagram Internet_Header Dst IP
                             address to the option IP address
                            Network = Get_Route( disable redirect )
                            If Network is ROUTE_NO_ROUTE or the
                             Rc_Datagram Dst IP address is not the gw IP
                             address determined by Get_Route ( route is
                             not a direct hop )
                                Call Fail_Ip( SOURCE_ROUTE_FAILED )
                                Set status to NEXT_PACKET
                            Endif
                        Endif
                    Endif

    IO_LSRR:        Set status to CONTINUE
                    If Network is ROUTE_FOR_ME
                        If an IP address is not in the option; the
                         option pointer Option_Adr(2)+3 exceeds the
                         option length Option_Adr(1)
                            Call Fail_Ip( PARAMETER_PROBLEM )
```

```
                                        Set status to NEXT_PACKET
                                    Else
                                        Set the Rc_Datagram Internet_Header Dst IP
                                         address to the option IP address
                                        Network = Get_Route( disable redirect )
                                        If Network is ROUTE_NO_ROUTE
                                            Call Fail_Ip( SOURCE_ROUTE_FAILED )
                                            Set status to NEXT_PACKET
                                        Endif
                                    Endif
                                Endif

    IO_RR:                      Set status to CONTINUE
                                If an IP address is not in the option; the option
                                 pointer Option_Adr(2)+3 exceeds the option length
                                 Option_Adr(1)
                                    Call Fail_Ip( PARAMETER_PROBLEM )
                                    Return NEXT_PACKET status
                                Else
                                    Copy the NET_Table Net_IP_Addr pointed to by
                                     Network to the Option_Adr route field
                                Endif

    Default:                    Call Fail_Ip( PARAMETER_PROBLEM )
                                Set status to NEXT_PACKET
Endcase
Return status
```

3.3.2.4.15.5 Limitations

No limitations are defined for the Options unit.

3.3.2.4.16 Reassemble Unit

Reassemble processes fragmented datagrams. Any datagram that has either the more fragments bit set (not zero) or a fragment offset greater than zero is considered a fragmented datagram. A set of fragmented datagrams which all have identical Identification, Protocol, Source IP Address, and Destination IP Address fields in the internet header are assembled as a complete datagram where the internet header of the first fragmented datagram (more fragments bit is one and fragment offset is zero) is used and the total length of the complete datagram is determined from the fragment offset of the last datagram (more fragments bit is zero and fragment offset is greater than zero).

The datagram is considered complete when each byte of the datagram is placed in the allocated resource entry. The filled byte positions are recorded in a fragment received bit table which is associated with the allocated resource Entry.

3.3.2.4.16.1 Inputs

The following inputs are required by the Reassemble unit:

1) TIMER_LOWER_BOUND - This global data input is a 32 bit integer which contains the lower bound time to live for the datagram being assembled.

2) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

3) RC_Datagram - This global IP input is a pointer to an Dgram_Message which may be fragmented and placed in the resource table.

4) Resources - This global IP input is an array of datagram entries used for the assembly of fragmented datagrams.

3.3.2.4.16.2 Outputs

The following outputs are returned by the Reassemble unit:

1) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global IP output is a pointer to a complete Dgram_Message which can be forwarded as the protocol indicates.

3) Status - This output parameter is a 32 bit integer which indicates the next operation for the datagram:

   CONTINUE - the datagram can be forwarded
   NEXT_PACKET - datagram is processed, accept another

5) Resources - This global IP output is an array of datagram entries used for the assembly of fragmented datagrams.

3.3.2.4.16.3 Local Data

No local data is defined for the Reassemble unit.

3.3.2.4.16.4 Processing

```
Set diff to difference between current time and saved time
Save current time
For each reassembly buffer allocated
    Decrement reassembly timer by diff
    If reassembly timer expired
        Free reassembly resources
    Endif
Endfor
For each reassembly buffer allocated
    If there is a  match of Id, Protocol, Src, and Dst between
     incoming packet and reassembly entry
        Move current reassembly entry pointer to cur_reas
        Exit loop
    Endif
Endwhile
If no fragment offset or IP_MF bit is clear
    If reassembly resources have been allocated for current datagram
        Release reassembly resources for current datagram
    Endif
    Return CONTINUE
Endif
If no match reassembly buffer has been found
    Allocate a reassembly buffer
    If error in buffer allocation
        Call er_message_discard(Rc_msg_hdr pointer)
        Return NEXT_PACKET
    Endif
    Set H_len field of reassembly datagram to MAX_IP_HDR_LEN
    Set Id, Protocol, Src, and Dst fields of reassembly datagram
     to values obtained from incoming datagram
Endif
Call bcopy(pointer to incoming IP datagram data,
 pointer to location in reassembly datagram to place data,
 number of bytes to copy to reassembly datagram)
Set bits in Frb_table indicating the 8 byte blocks that have arrived
If IP_MF bit is clear
```

```
    Set length of reassembly message buffer
    Set Len field of reassembly datagram
Endif
If offset is 0
    Fill data empty option entries in reassembly datagram header
     with IO_EOL
    Set Version, Tos, Ttl, options in reassembly datagram from
     incoming IP datagram
    Clear Off field of reassembly datagram
    Calculate checksum of reassembly datagram
Endif
If last fragment has arrived
    If Frb_table indicates that all fragments have arrived
        Reassign reassembly buffer to Rc_msg_hdr by swaping
         appropriate fields in the two headers
        Release reassembly resources
        Return CONTINUE
    Endif
Endif
If Ttl field of incoming datagram is greater than current timer
    Set current reassembly timer to Ttl field of incoming datagram
Endif
Call er_message_discard(Rc_msg_hdr pointer)
Return NEXT_PACKET
```

3.3.2.4.16.5 Limitations

No limitations are defined for the Reassemble unit.

3.3.2.4.17 Return_To_Sender Unit

Return_To_Sender returns the current message to the sender with an ICMP message of the appropriate type and code.

3.3.2.4.17.1 Inputs

The following inputs are required by the Return_To_Sender unit:

1) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global IP input is a Pointer to a Dgram_Message.

3) Type - This input parameter is a 32 bit integer which identifies the ICMP message type.

4) Code - This input parameter is a 32 bit integer which identifies the ICMP message subtype.

5) Other - This input parameter is a 32 bit integer which is not used.

3.3.2.4.17.2 Outputs

No outputs are returned by the Return_To_Sender unit.

3.3.2.4.17.3 Local Data

The following local data is defined for the Return_To_Sender unit:

1) Cp_Msg_Hdr - This local data item is a Message_Header for a message containing a Cp_Datagram used so that the original datagram is not destroyed.

2) Cp_Datagram - This local data item is an Dgram_Message used so the original datagram is not destroyed.

3) Tp_Msg_Hdr - This local data item is a Message_Header for a message containing a Tp_Datagram to encompass either the Rc_Msg_Hdr or the Cp_Msg_Hdr.

4) Tp_Datagram - This local data item is an Dgram_Message used to encompass the Rc_Datagram or the Cp_Datagram.

3.3.2.4.17.4 Processing

```
If the Rc_Datagram  Internet_Header Protocol is IH_ICMP
    If type is ICMP_REDIRECT
        Call message_discard
    Endif
    Return
Endif

If the Type is ICMP_REDIRECT
    Call Message_Get( address of Cp_Msg_Hdr )
    If ERROR
        Return
    Endif
    Set the Cp_Msg_Hdr fields to the Rc_Msg_Hdr fields
    Set the Cp_Datagram Out_Dest equal to the Tp_Datagram Out_Dest
    Set the Tp_Msg_Hdr to point to the Cp_Msg_Hdr
    Set the Tp_Datagram to point to the Cp_Datagram
Else
    Set the Tp_Msg_Hdr to point to the Rc_Msg_Hdr
    Set the Tp_Datagram to point to the Rc_Datagram
Endif

/* Set up the data portion of the ICMP message */

Set a copy pointer to start of Tp_Datagram Internet_Header plus
 20 bytes for the Internet_Header for ICMP plus 8 bytes for the
 ICMP fields in the Tp_Datagram Data field
Data to copy equals the minimum of ( Rc_Datagram length VHL*4 + 64 )
 or Ntohs( Rc_Datagram LEN )
Copy the data to copy bytes from Rc_Datagram Internet_Header to the
 copy pointer

/* Set up the internet header portion of the ICMP message */

Set the Tp_Datagram VHL version and internet header length LEN,
 Protocol to IH_ICMP, TTL, checksum Sum equal zero, the offset
 OFF equal zero, the destination address Dst from the Rc_Datagram source
 Src, and the source Src to the IGW address
Set the length in the message pointer to the ICMP message
Call Calc_Cksum( address of Tp_Datagram Internet_Header, 10 )

/* Set up the ICMP fields */

Copy the Type and Code to the zeroth and first bytes of the Tc_Datagram
 Data section, and the Others to the fourth through seventh bytes of the
 Tc_Datagram Data section
Set the Tc_Datagram Data bytes two through three to zero
Call Calc_Cksum( address of Tc_Datagram Data, (data to copy+8)/2 )

Set the Tp_Msg_Hdr M_Qid to IP_Q_ID
Call Message_Send( address of Tp_Msg_Hdr, Tp_Msg_Hdr M_Qid )
If the Error
    Call Message_Discard( address of Tp_Msg_Hdr )
Return
```

3.3.2.4.17.5 Limitations

No limitations are defined for the Return_To_Sender unit.

3.3.2.4.18 Route Unit

Route initiates the processing of the packet depending on the previously determined route.

3.3.2.4.18.1 Inputs

The following input is required by the Route unit:

1) Network - This input parameter is a 32 bit integer which contains either a pointer to a NET_Table entry as determined by the IP_Route_Table, or one of the route flags: ROUTE_FOR_ME or ROUTE_NO_ROUTE.

2) Rc_Msg_Hdr - This global IP input is a Message_Header for a message containing a Rc_Datagram.

3) Rc_Datagram - This global IP input is a pointer to a Dgram_Message.

3.3.2.4.18.2 Outputs

The following output is returned by the Route unit:

1) Status - This output parameter is a 32 bit integer which indicates processing success of packet:

   CONTINUE - processing on the datagram may continue
   NEXT_PACKET - drop the current datagram
   SEND - resend the datagram

2) Rc_Msg_Hdr - This global IP output is a Message_Header for a message containing a Rc_Datagram.

3.3.2.4.18.3 Local Data

No local data is defined for the Route unit.

3.3.2.4.18.4 Processing

```
Set the Status to CONTINUE
If Network is ROUTE_TO_ME
    Status = Reassemble( address of Rc_Datagram )
    If Status is not CONTINUE
        Return Status
    Endif
    Case on the Rc_Datagram Protocol
        IH_ICMP: Call Update_Icmp_Stats(address of Rc_Msg_Hdr, "R")
                 If Do_Icmp( ) == False
                                 Call er_message_discard (Re_msg_hdr)
                                 Return(NEXT_PACKET)
                                 Endif
                 Return(RESEND)

        IH_EGP:  Set the Rc_Msg_Hdr M_Qid to EGP_Q_ID
                 Call Message_Send( Rc_Msg_Hdr, EGP_Q_ID )
                 If no_error
                     Call Update_Ip_Stats( address of Rc_Msg_Hdr, "T" )
```

```
                    Else
                         Call message_discard(Re_msg_hdr)
                    Endif
                    Set the Status to NEXT_PACKET

         Default:   Call Fail_Ip( PARAMETER_PROBLEM )
                    Set the Status to NEXT_PACKET
    Endcase
Endif

If Network equals ROUTE_NO_ROUTE
    Call Fail_Ip( HOST_UNREACHABLE )
    Set the Status to NEXT_PACKET
Endif

Return Status
```

3.3.2.4.18.5 Limitations

No limitations are defined for the Route unit.

3.3.2.4.19 Route_Init Unit

Route_Init uses the gateway, the network, and the EGP route tables to update the IP_Route_Table.

3.3.2.4.19.1 Inputs

The following inputs are required by the Route_Init unit:

1) GW_Table - This global data input contains a list of gateway entries.

2) NET_Table - This global data input contains a list of network entries.

3) EGP_Table - This global data input contains a list of EGP route entries.

3.3.2.4.19.2 Outputs

The following outputs are returned by the Route_Init unit.

1) Hash_Table_A - This global data output contains a list of pointers to the class A networks in the IP_Route_Table.

2) Hash_Table_B - This global data output contains a list of pointers to the class B networks in the IP_Route_Table.

3) Hash_Table_C - This global data output contains a list of pointers to the class C networks in the IP_Route_Table.

4) IP_Route_Table - This global data output contains a list of route entries which are required the the IP TLC.

3.3.2.4.19.3 Local Data

No local data is defined for the Route_Init unit.

3.3.2.4.19.4 Processing

```
For each entry in the GW_Table
    If the entry is marked GW_REROUTE
        Set the IP_Route_Table Net to GW_Table ( GW_Dst_Net and
         GW_Mask )
        Set the IP_Route_Table Gw to the GW_Table GW_Addr
        Set the IP_Route_Table Nwindex to the GW_Table GW_Number
        Set the IP_Route_Table Mask to the GW_Table GW_Mask
        Set the IP_Route_Table entry Flag to (IP_VALID or
             IP_TCPONLY)
        Increment IP_Route_Table entry pointer
    Endif
Endfor

For each entry in the NET_Table
    Mark NET_Table entry as NW_UP
    Set the IP_Route_Table Net to ( Net_IP_Addr and Net_Mask)
    Set the IP_Route_Table Gw to the Net_IP_Addr
    Set the IP_Route_Table Nwindex to the NET_Table index
    Set the IP_Route_Table Mask to the Net_Mask
    Set the IP_Route_Table entry Flag to ( IP_VALID or IP_DIRECT )
    Increment IP_Route_Table entry pointer
Endfor

For each entry in the GW_Table
    If the ( GW_Table entry and GW_GW ) equals GW_GW
        Set the IP_Route_Table Net to GW_Table ( GW_Dst_Net and
         GW_Mask )
        Set the IP_Route_Table Gw to the GW_Table GW_Addr
        Set the IP_Route_Table Nwindex to the GW_Table GW_Number
        Set the IP_Route_Table Mask to the GW_Table GW_Mask
        Set the IP_Route_Table Flag to IP_VALID
        Increment IP_Route_Table entry pointer
    Endif
Endfor

For each entry in the EGP_Table
    Set the IP_Route_Table Net to EGP_Route_Net
    Set the IP_Route_Table Gw to the EGP_Route_GW
    Set the IP_Route_Table Mask to Net_Class( first Net byte )
    Set the IP_Route_Table Nwindex to EGP_IF
    Mark the IP_Route_Table entry Flag as IP_VALID or IP_EGP
    Increment IP_Route_Table entry pointer
Endfor

Mark the last IP_Route_Table entry Flag as IP_LASTALLOC
Initialize Hash_Table_A, Hash_Table_B, and Hash_Table_C entries to zero
For each entry in the IP_Route_Table
    Case Net_Class( first byte of IP_Route_Table Net )
        CLASS_A:  Obtain the pointer to the Hash_Table_A entry indexed
                  by the least significant byte of the IP_Route_Table
                  entry

        CLASS_B:  Obtain the pointer to the Hash_Table_B entry indexed
                   by the second least significant byte of the
                   IP_Route_Table entry

        CLASS_C:  Obtain the pointer to the Hash_Table_C entry indexed
                   by the third least significant byte of the
                   IP_Route_Table entry

        Default:  Continue
    Endcase
    Trace and update the pointer from the hash table through the
     IP_Route_Table which terminates at a zero value for the pointer
    Set the pointer to the address of the IP_Route_Table entry and
     initialize the next IP_Route_Table entry pointer to zero
Endfor
Return
```

3.3.2.4.19.5 Limitations

No limitations are defined for the Route_Init unit.

3.3.2.4.20 Send_Icmp_Stats Unit

Send_Icmp_Stats sends the ICMP_Stats_Table message buffer to the STAT TLC and resets the ICMP_Stats_Table descriptor and list of entries to an empty state.

3.3.2.4.20.1 Inputs

The following inputs are required by the Send_Icmp_Stats unit:

1) ICMP_Stat_Hdr - This global IP input is a Message_Header for a message containing an ICMP_Stats_Table.

2) ICMP_Stats_Table - This global IP input is a Stats_Buffer for a message containing a list of ICMP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.20.2 Outputs

The following outputs are returned by the Send_Icmp_Stats unit:

1) ICMP_Stat_Timeout - This global IP output is a 32 bit integer which maintains the periodic time interval for the update of the ICMP statistics.

2) ICMP_Stat_Hdr - This global IP output is a Message_Header for a message containing an ICMP_Stats_Table.

3) ICMP_Stats_Table - This global IP output is a Stats_Buffer for a message containing a list of ICMP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.20.3 Local Data

No local data is defined for the Send_Icmp_Stats unit.

3.3.2.4.20.4 Processing

```
/* Send the ICMP statistics message buffer */
If ICMP_STATS_Table is valid
    Call Message_Send( address of  STAT_Q_ID, ICMP_Stat_Hdr )
    If Error
        Call message_discard(ICMP_STAT_hdr)
    Endif
Endif

/* Create a new ICMP statistics message buffer and initialize */

Call Message_Get( address of ICMP_Stat_Hdr )
If no error
    Set the ICMP_Stat_Hdr M_Offset to the number of bytes in the
     ICMP_Stats_Table Descriptor field
    Set the ICMP_Stat_Hdr M_Length to the ICMP_stat_Hdr M_Offset
    Set the ICMP_Stat_Hdr M_From to the sending process qid IP_Q_ID
    Set the ICMP_Stat_Hdr M_Qid to the quid to deliver to STAT_Q_ID
    Set the ICMP_Stats_Table Descriptor Entries to zero
    Set the ICMP_Stats_Table Descriptor Operation to UPD
    Set the ICMP_Stats_Table Descriptor Statistic to ICMP

/* Reset the timer */

    Reset the ICMP_Stat_Timeout to the periodic icmp statistic interval
Else
    Mark ICMP STATS Table as invalid
    Set ICMP_STAT_Timeout to STAT_NOBUF_Timeout
Endif
Return
```

3.3.2.4.20.5 Limitations

No limitations are defined for the Send_Icmp_Stats unit.

3.3.2.4.21 Send_Ip_Stats Unit

Send_Ip_Stats sends the IP_Stats_Table message buffer to the STAT TLC and resets the IP_Stats_Table descriptor and list of entries to an empty state.

3.3.2.4.21.1 Inputs

The following inputs are required by the Send_Ip_Stats unit:

1) IP_Stat_Hdr - This global IP input is a Message_Header for a message containing an IP_Stats_Table.

2) IP_Stats_Table - This global IP input is a Stats_Buffer for a message containing a list of IP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.21.2 Outputs

The following outputs are returned by the Send_Ip_Stats unit:

1) IP_Stat_Timeout - This global IP output is a 32 bit integer which maintains the periodic time interval for the update of the IP statistics.

2) IP_Stat_Hdr - This global IP output is a Message_Header for a message containing an IP_Stats_Table.

3) IP_Stats_Table - This global IP output is a Stats_Buffer for a message containing a list of IP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.21.3 Local Data

No local data is defined for the Send_Ip_Stats unit.

3.3.2.4.21.4 Processing

```
/* Send the IP statistics message buffer */
If IP_STATS Table is valid
    Call Message_Send( address of IP_Stat_Hdr, STAT_Q_ID )
    If error
        Call message_discard(IP_STAT_hdr)
    Endif
Endif
/* Create a new IP statistics message buffer and initialize */

Call Message_Get( address of IP_Stat_Hdr )
If No Error
    Set the IP_Stat_Hdr M_Offset to the number of bytes in the
     IP_Stats_Table Descriptor field
    Set the IP_Stat_Hdr M_Length to the IP_Stat_Hdr M_Offset
    Set the IP_Stat_Hdr M_From to the sending process qid IP_Q_ID
    Set the IP_Stat_Hdr M_Qid to the quid to deliver to STAT_Q_ID
    Set the IP_Stats_Table Descriptor Entries to zero
    Set the IP_Stats_Table Descriptor Operation to UPD
    Set the IP_Stats_Table Descriptor Statistic to IP

/* Reset the timer */

    Reset the IP_Stat_Timeout to the periodic ip statistic interval
Else
    Mark IP_STATS Table as invalid
    Set IP_STAT_Timeout to STAT_NOBUF_Timeout
Endif
Return
```

3.3.2.4.21.5 Limitations

No limitations are defined for the Send_Ip_Stats unit.

3.3.2.4.22 Time_Stamp Unit

Time_Stamp obtains the current IGW system time in hundredth of seconds for use in ICMP datagrams. The most significant bit is set to indicate the time is non-standard.

3.3.2.4.22.1 Inputs

No inputs are required by the Time_Stamp unit.

3.3.2.4.22.2 Outputs

The following output is returned by the Time_Stamp unit:

1) Time - This output function value is a 32 bit integer in which the IGW time is placed.

3.3.2.4.22.3 Local Data

No local data is defined for the Time_Stamp unit.

3.3.2.4.22.4 Processing

```
Time = Get_Time
Set the most significant bit in Time to indicate a non-standard time
Return(htonl(Time))
```

3.3.2.4.22.5 Limitations

No limitations are defined for the Time_Stamp unit

3.3.2.4.23 Update_Icmp_Stats Unit

Update_Icmp_Stats places datagram information into the STAT TLC message buffer for ICMP stats. The message buffer is sent when the message buffer is full.

3.3.2.4.23.1 Inputs

The following inputs are required by the Update_Icmp_Stats unit:

1) Direction - This input parameter is a byte which indicates the acquisition stat of the Dgram_Datagram as one of the following single characters:

   R - The Dgram_Datagram has been received by ICMP TLC.
   T - The Dgram_Datagram has been transmitted by ICMP TLC.

2) Dgram_Msg_Hdr - This global IP input is a Message_Header for a message containing an Dgram_Datagram.

3) Dgram_Datagram - This global IP input is an ICMP_Datagram for a message containing a list of ICMP_Stats_Entry records to be sent to the STAT TLC.

4) ICMP_Stat_Hdr - This global IP input is a Message_Header for a message containing an ICMP_Stats_Table.

5) ICMP_Stats_Table - This global IP input is a Stats_Buffer for a message containing a list of ICMP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.23.2 Outputs

The following outputs are returned by the Update_Icmp_Stats unit:

1) ICMP_Stat_Hdr - This global IP output is a Message_Header for a message containing an ICMP_Stats_Table.

2) ICMP_Stats_Table - This global IP output is a Stats_Buffer for a message containing a list of ICMP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.23.3 Local Data

No local data is defined for the Update_Icmp_Stats Unit.

3.3.2.4.23.4 Processing

```
If ICMP STATS Table is valid
    Increment the ICMP_Stats_Table Descriptor Entries
    Increase the ICMP_Stat_Hdr byte count M_Length by seven
    Set the ICMP_Stats_Table entry Entries fields to the Direction and the
    Dgram_Datagram fields IP_Address, Type, and Code
    If the ICMP_Stats_Table Descriptor Entries equals STATS_RESOURCES
        Call Send_Icmp_Stats
      Endif
Endif
Return
```

3.3.2.4.23.5 Limitations

No limitations are defined for the Update_Icmp_Stats unit.

3.3.2.4.24 Update_Ip_Stats Unit

Update_Ip_Stats places datagram information into the STAT TLC message buffer for IP stats. The message buffer is sent when the message buffer is full.

3.3.2.4.24.1 Inputs

The following inputs are required by the Update_Ip_Stats unit:

1) Dgram_Msg_Hdr - This input parameter is a Message_Header for a message containing an IP datagram.

2) IP_Stat_Hdr - This global IP input is a Message_Header for a message containing an IP_Stats_Table.

3) IP_Stats_Table - This global IP input is a Stats_Buffer for a message containing a list of IP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.24.2 Outputs

The following outputs are returned by the Update_Ip_Stats unit:

1) IP_Stat_Hdr - This global IP input is a Message_Header for a message containing an IP_Stats_Table.

2) IP_Stats_Table - This global IP input is a Stats_Buffer for a message containing a list of IP_Stats_Entry records to be sent to the STAT TLC.

3.3.2.4.24.3 Local Data

No local data is defined for the Update_Ip_Stats unit.

3.3.2.4.24.4 Processing

```
If IP_STATS_Table is valid
    Increment the IP_Stats_Table Descriptor Entries
    Increase the IP_Stat_Hdr byte count M_Length by thirteen
    Set the IP_STATS_Entry fields IP_Address_A and
      IP_Address_B to the Source and Destination
        addresses of the IP datagram
    Set the IP_STATS_Entry field Direction to Transmit
    Set the IP_STATS_Entry field Pkt_Size to IP
      datagram length - 4* IHL
    If the IP_Stats_Table Descriptor Entries equals STATS_RESOURCES
        Call Send_Ip_Stats
    Endif
Endif
Return
```

3.3.2.4.24.5 Limitations

No limitations are defined for the Update_Ip_Stats unit.

3.3.2.4.25 Ver_Cksum Unit

Ver_Cksum verifies that the checksum specified for the data equals the calculated checksum.

3.3.2.4.25.1 Inputs

The following inputs are required by the Ver_Cksum unit:

1) Data - This input parameter is a pointer to a data set for which the checksum is specified and the checksum is to be calculated.

2) Count - This input parameter is a 32 bit integer containing the number of 16 bit words in Data.

3.3.2.4.25.2 Outputs

The following outputs are returned by the Ver_Cksum unit:

1) Status - This output function value is a 32 bit integer which indicates the success of the checksum verification:

   CONTINUE - The checksum is correct
   NEXT_PACKET - The checksum did not verify

3.3.2.4.25.3 Local Data

No local data is defined for the Ver_Cksum unit.

3.3.2.4.25.4 Processing

```
Status is set to CONTINUE
Calculated checksum = Calc_Cksum( Data, Count )
If the Cksum does not equal 0
    Status is set to NEXT_PACKET
Endif
Return the Status
```

3.3.2.4.25.5 Limitations

No limitations are defined for the Ver_Cksum unit.

3.3.3 EGP TLC Detailed Design

The EGP TLC implements the EGP protocol.

3.3.3.1 EGP TLC Architecture

The EGP TLC consist of the following Units as shown in Figure 3-3. The Units are:

1) Allocatep Unit - This unit returns a message header for a message which will contain an EGP_Datagram.

2) Egp_Ac Unit - This unit generates a neighbour acquisition message and sends the message to the neighbour gateway.

3) Egp_Control Unit - This unit processes incoming EGP_Datagrams when messages are on the queue, updates the states of existing neighbours, and manages the collection and submission of EGP statistics to the

```
                            +-----+
                            | EGP |
                            | TLC |
                            +--+--+
                               |
      +-----------+-----------++-----------+------------+---------------+
      |           |           |            |            |               |
+-----+-----+ +---+----+ +----+----+ +-----+------+ +---+----+          |
| Allocatep | | EGP_Hi | | EGP_Nrp | | EGP_Update | | Egp_Ac |          |
+-----------+ +--------+ +---------+ +------------+ +--------+          |
                                                                        |
       +---------------+------------+----------+------------------------+
       |               |            |          |                        |
+------+------+ +------+------+ +---+----+ +---+---+                    |
| Egp_Control | | Egp_Request | | Finden | | Htons |                    |
+-------------+ +-------------+ +--------+ +-------+                    |
                                                                        |
            +-------------------------+---------------------+-----------+
            |                         |                     |           |
+-----------------------+ +-----------+------------+ +------+-------+   |
| Neighbour_Acquisition | | Neighbour_Reachability | | Network_Poll |   |
+-----------------------+ +------------------------+ +--------------+   |
                                                                        |
            +----------------+------------+-----------------+-----------+
            |                |            |                 |           |
+-----------+----------+ +---+---+ +------+-------+ +-------+--------+  |
| Network_Reachability | | Ntohs | | Route_Update | | Send_Egp_Stats |  |
+----------------------+ +-------+ +--------------+ +----------------+  |
                                                                        |
    +---------------+---------------------------------------------------+
    |               |
+---+----+ +--------+---------+
| Sendit | | Update_Egp_Stats |
+--------+ +------------------+
```

Figure 3-3

STAT TLC.

4) Egp_Hi Unit - This unit generates a neighbour reachability message and sends the message to the neighbour gateway.

5) Egp_Nrp Unit - This unit generates a poll command and sends the message to the neighbour gateway.

6) Egp_Request Unit - This unit processes an EGP_Datagram, generating an appropriate response to a received message.

7) Egp_Update Unit - This unit updates the states for each of the neighbour gateways listed in the neighbour table. The main neighbour with an UP state is polled for an EGP update message.

8) Finden Unit - This unit locates an EGP table entry or finds the first entry marked for delete when the entry is not in the table.

9) Neighbour_Acquisition Unit - This unit processes the request, cease, reply, refuse and cease acknowledgement forms of the neighbour acquisition EGP_Datagram.

10) Neighbour_Reachability Unit - This unit returns an I-HEARD-YOU in response to a HELLO EGP_Datagram.

11) Network_Poll Unit - This unit creates an Update message which is sent to the polling neighbour gateway.

12) Network_Reachability Unit - This unit processes an Update message from the router gateway neighbour.

13) Route_Update Unit - This unit processes a received update message. The gateway and network address pairs are extracted from the update message and placed in the EGP Route Table which is to be sent to the IP TLC.

14) Send_Egp_Stats Unit - This unit sends an EGP statistics buffer to the STAT TLC.

15) Sendit Unit - This unit completes the both the message header and the datagram header fields of the EGP_Datagram. The message is sent to the IP TLC.

16) Update_Egp_Stats Unit - This unit adds an entry to the Egp_Stats_Table. When full, the Egp_Stats_Table is sent to the STAT TLC.

17) Ntohs Unit - This unit is described in the IP TLC.

18) Htons Unit - This unit is described in the IP TLC.

19) Net_Class - This unit returns the class of the specified network address. This unit is described in the detailed design of the IP TLC.

20) Ver_Cksum - This unit verifies that the specified data contains the correct checksum. The unit is described in the detailed design of the IP TLC.

21) Calc_CKSUM - This unit computes the one's complement checksum of a block of data. This unit is described in the detailed design of the IP TLC.

3.3.3.2 EGP Global Data

The following is a list of global structures defined for EGP:

1) The NA_Message provides the additional fields to the EGP_Datagram to complete the neighbour acquisition EGP Message. The following fields are defined for the NA_Message:

   Hello_Interval - This message field is a 16 bit integer which contains the time in seconds between neighbour reachability hello messages. This message field exists only for neighbour acquisition Codes REQUEST and REPLY.

   Poll_Interval - This message field is a 16 bit integer which contains the time in seconds between poll messages. This message field only exists for neighbour acquisition Codes REQUEST and REPLY.

2) The Poll_Message provides the additional fields to the EGP_Datagram to complete the poll command EGP Message. The following fields are defined for the Poll_Message:

Reserved - This message field is a 16 bit integer which is reserved for future use.

IP_Source_Network - This message field is a 32 bit integer which contains the network address common to the neighbouring gateways.

3) The Update_Message provides the additional fields to the EGP_Datagram to complete the network reachability EGP Message. The following fields are defined for the Update_Message:

Int_Gw - This message field is a byte which contains the total number of interior gateways listed in the Gateway_List section.

Ext_Gw - This message field is a byte which contains the total number of exterior gateways listed in the Gateway_List section.

Ip_Source_Network - This message field consists of a 32 bit integer which contains the network address about which reachability is being supplied.

Gateway_List - This section is variable length and contains a list of Gateways, each identified by up to three host address bytes. Each Gateway contains one Distance byte totalling the number of sets of distances for the particular Gateway, followed by a list of the actual distances. For each of the Distance sets, the Hop count takes one byte and the number of networks at the Hop distance takes a second byte. The Net number of each network reachable via the Gateway is listed in up to three bytes each, depending on the class of the network.

4) The Error_Message provides the additional fields to the EGP_Datagram to complete the error EGP Message. The following fields are defined for the Error_Message:

Reason - This message field is a 16 bit integer which identifies the reason for the error:

ERR_UNSPEC	(0)	- unspecified
ERR_BAD_HDR	(1)	- bad EGP header
ERR_BAD_DATA	(2)	- bad EGP data field format
ERR_REACH_UNAU	(3)	- reachability info unavailable
ERR_XPOLL_RATE	(4)	- excessive polling rate
ERR_NO_RESPONSE	(5)	- no response

Error_Header - This message field is a set of three 32 bit words which contains the first three words of the EGP header.

5) EGP_Datagrams contain neighbour acquisition, neighbour reachability, network reachability, updates, or error messages which are sent from and received by the EGP TLC. An EGP_Datagram is constructed as follows:

EGP_Version - This message field is a byte which contains the current version of the EGP protocol.

Type - This message field is a byte which contains the integer which defines the EGP message type:

EGP_NR	(1)	- network reachability
EGP_NRPOLL	(2)	- network poll
EGP_NA	(3)	- neighbour acquisition
EGP_HELLO	(5)	- neighbour reachability
EGP_ERROR	(8)	- error

Code - This message field is a byte which contains an indicator identifying the EGP Message subtype. The subtypes for different EGP Messages are zero except for the following:

NA_Message	CODE_REQUEST	(0)	- request
	CODE REPLY	(1)	- reply (confirm)
	CODE_REFUSE	(2)	- refuse
	CODE_CEASE	(3)	- cease
	CODE CACK	(4)	- cease acknowledge
NR_Message	CODE_HELLO	(0)	- hello
	CODE_IHU	(1)	- i_heard_you

Info - This message field is a byte which contains an indicator identifying the EGP Message status. The EGP Message status codes are defined as:

NA_Message	INFO_UNSPEC	(0)	- unspecified
	INFO_ACTIVE	(1)	- active mode
	INFO_PASSIVE	(2)	- passive mode
	INFO_INSUFF	(3)	- insufficient resources
	INFO_PROHIB	(4)	- administratively prohibited
	INFO_GDOWN	(5)	- going down
	INFO_PROBLEM	(6)	- parameter problem

	INFO_PVIOLATION	(7)	- protocol violation
NR_Message	INFO_INDETERM	(0)	- indeterminate
	INFO_UP	(1)	- up
	INFO_DOWN	(2)	- down
Poll_Message	INFOR_INDETERM	(0)	- indeterminate
	INFO_UP	(1)	- up
	INFO_DOWN	(2)	- down
Update_Message	INFO_INDETERM	(0)	- indeterminate
	INFO_UP	(1)	- up
	INFO_DOWN	(2)	- down
	INFO_UNSOLIC	(3)	- unsolicited message
Error_Message	INFO_INDETERM	(0)	- indeterminate
	INFO_UP	(1)	- up
	INFO_DOWN	(2)	- down
	INFO_UNSOLIC	(3)	- unsolicited message

Checksum - This message field is a 16 bit integer which contains a checksum on the neighbour reachability message only.

AS - This message field is a 16 bit integer which contains the Autonomous System number in which the IGW resides.

Sequence - This message field is a 16 bit integer which contains the current sequence number of the neighbour reachability message.

Data - This field consists of data in the form of one of the EGP Messages NA_Message, Poll_Message, Update_Message, or Error_Message. The data field may be empty (ie no message data).

The following is a list of global data items for the EGP TLC:

1) Rc_Msg_Hdr - This item is a Message_Header for a message containing a Rc_Datagram.

2) Tx_Msg_Hdr - This item is a Message_Header for a message containing a Tx_Datagram.

3) Stat_Msg_Hdr - This item is a Message_Header for a

message containing an Egp_Stats_Table.

4) Rc_Datagram - This item is an Dgram_Message which has been received on the EGP queue.

5) Tx_Datagram - This item is an Dgram_Message which has been created for transmission.

6) Egp_Stats_Table - This item is a Stats_Buffer containing a list of EGP_Stats_Entry entries. The Egp_Stats_Table accumulates statistics related to messages sent and received by the EGP TLC.

7) Sequence - This item is a 32 bit integer which contains the current sequence number to be used for Tx_Datagram.

8) EGP_Table - This item is the address of EGP route table entries.

9) Stats_Timeout - This item is a 32 bit integer which maintains the periodic update interval for the update of the Egp_Stats_Table.

10) Router_Timer - This item is a 32 bit integer which maintains the last time an update message was received.

11) EGP_EXPIR_TIME - This constant defined as 100 defines the number of seconds that must pass from last hearing about the gateway before the gateway can be marked down.

12) EGP_ROUTE_TIME - This constant defined as 180 defines the number of seconds between requests for route information from the main gateway neighbour.

13) Ip-Rtr - This global EGP item points to the Internet header in the global EGP RC_Msg_Hdr message buffer.

14) EGP_Rtr - This global EGP item points to the EGP datagram, which is contained in the Internet header pointed to by Ip_Rtr, of the global EGP Rc_Msg_Hdr message buffer.

15) Ip_Ttr - This global EGP item points to the Internet header in the global EGP Tx_Msg_Hdr message buffer.

16) Egp_Ttr - This global EGP item points to the EGP

datagram, which is contained in the Internet header pointed to by Ip_Ttr, of the global EGP Tx_Msg_Hdr message buffer.

3.3.3.3 EGP LLC Design

There are no lower level components defined for the EGP TLC.

3.3.3.4 EGP Units

This section describes the EGP units.

3.3.3.4.1 Allocatep Unit

Allocatep returns a message header which points to an Tx_Datagram message buffer.

3.3.3.4.1.1 Inputs

No inputs are required by the Allocatep unit.

3.3.3.4.1.2 Outputs

The following outputs are produced by the Allocatep unit:

1) Tx_Msg_Hdr - This global EGP output consists of a Message_Header for a message which contains a Tx_Datagram.

2) Status - This output function value is a 16 bit integer which indicates the success in allocating space for the Tx_Datagram message.

3) Ip_Ttr - This global EGP output points to the Internet header in the global EGP Tx_Msg_Hdr message buffer.

4) EGP_Ttr - This global EGP output points to the EGP datagram, which is contained in the Internet header pointed to by IP_Ttr, of the global EGP Tx_Msg_Hdr message buffer.

3.3.3.4.1.3 Local Data

No local data is defined for the Allocatep unit.

3.3.3.4.1.4 Processing

```
Status = Message_Get( address of Tx_Msg_Hdr )
Set Ip_Ttr and Egp_Ttr pointers
Return Status
```

3.3.3.4.1.5 Limitations

No limitations are defined for the Allocatep unit.

3.3.3.4.2 Egp_Ac Unit

Egp_Ac generates a neighbour acquisition message and sends the message to the neighbour gateway.

3.3.3.4.2.1 Inputs

The following inputs are required by the Egp_Ac unit:

1) Code - This input parameter is a 32 bit integer which contains a neighbour acquisition message code.

2) Source - This input parameter is a 32 bit integer which contains an IP address indicating the source of the IP datagram.

3) Destination - This input parameter is a 32 bit integer which contains an IP address of the neighbour gateway to which the neighbour acquisition message is sent.

4) Sequence - This input parameter is a 16 bit integer which contains the current sequence number of the EGP_Datagram.

5) Egp_Ttr - This global EGP input points to the EGP datagram, which is contained in the Internet header pointed to by Ip_Ttr, of the global EGP Tx_Msg_Hdr message buffer.

3.3.3.4.2.2 Outputs

The following output is produced by the Egp_Ac unit:

1) Sequence - This global EGP output is a 16 bit integer which contains the current sequence number of the EGP_Datagram.

2) Tx_Msg_Hdr - This global EGP output is a Message_Header for a message containing a Tx_Datagram.

3) Tx_Datagram - This global EGP output is an Dgram_Message which will contain a neighbour acquisition EGP_Datagram.

3.3.3.4.2.3 Local Data

The following local data is defined for the Egp_Ac unit:

1) Length - This local data item is a 16 bit integer which contains the number of bytes in the EGP_Datagram of the Tx_Datagram.

2) Status - This output return parameter is a 16 bit integer which identifies either a successful or unsuccessful acquisition of the message header.

3.3.3.4.2.4 Processing

```
Status = Allocatep
If Status is NOERROR

    /* Create the appropriate neighbour acquisition message */

    Set the Tx_Datagram Type to EGP_NA
    Set the Tx_Datagram Code to the input Code
    Set the Tx_Datagram Info to the integer value INFO_PASSIVE
    Case Code
    CODE REQUEST: Set the Tx_Datagram Sequence = Htons( Sequence
                   number incremented )
                  Set the Tx_Datagram Hello_Interval = Htons( 180 )
                  Set the Tx_Datagram Poll_Interval = Htons( 180 )
                  Set the Length to fourteen

    CODE REPLY:   Set the Tx_Datagram Sequence = Htons(Sequence number)
                  Set the Tx_Datagram Hello_Interval = Htons( 180 )
                  Set the Tx_Datagram Poll_Interval = Htons( 180 )
                  Set the length to fourteen

    CODE CEASE:   Set the Tx_Datagram Sequence = Htons( Sequence
                   number incremented )
                  Set the Length to ten

    CODE REFUSE:  Set the Tx_Datagram Sequence = Htons( Sequence
                   number incremented )
                  Set the Length to ten

    CODE CACK:     Set the Tx_Datagram Sequence = Htons( Sequence
                   number incremented )
                  Set the Length to ten
    Endcase
    Call Sendit( Length, Source Destination )
Endif
Return
```

3.3.3.4.2.5 Limitations

No limitations are defined for the Egp_Ac unit.

3.3.3.4.3 Egp_Control Unit

Egp_Control processes datagrams containing EGP messages when messages are on the queue, updates the states of existing neighbours, and manages the collection and submission of EGP stats to the STAT TLC.

3.3.3.4.3.1 Inputs

The following inputs are required by the Egp_Control unit:

1) Rc_Msg_Hdr - This global EGP input is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global EGP input is a Dgram_Message which will contain an EGP_Datagram.

3) Stats_Timeout - This global EGP input is a 32 bit integer which maintains the periodic time interval for the update of the EGP statistics.

3.3.3.4.3.2 Outputs

The following outputs are returned by the Egp_Control unit:

1) Stat_Msg_Hdr - This global EGP output is a Message_Header for a message containing the Egp_Stats_Table.

2) Egp_Stats_Table - This global EGP output is a Stats_Buffer for a message containing a list of EGP_Stats_Entry records to be sent to the STAT TLC.

3.3.3.4.3.3 Local Data

The following local data is defined for the Egp_Control unit:

1) Egp_Timeout - This local data item is a 32 bit integer which maintains a timer for the update of states on all active EGP neighbour table entries.

3.3.3.4.3.4 Processing

```
Call Message_Get( address of Stat_Msg_Hdr )
Set the Stat_Msg_Hdr M_Offset to the number of bytes in the
 Egp_Stat_Table Descriptor field
Set the Stat_Msg_Hdr M_Length to the Stat_Msg_Hdr M_Offset
Set the Egp_Stat_Table Descriptor Entries to zero
Set the Egp_Stat_Table Descriptor Operation to UPD
Set the Egp_Stat_Table Descriptor Statistic to EGP
Status = Open_Message_Queue( EGP_Q_SIZE, EGP_Q_ID )
If status is not NOERROR
    Call Oi_Messages( status )
Else
    Egp_Timeout = thirty seconds
    Stats_Timeout = zero
    Start timer = Get_Time
    Loop forever
        Status = Message_Receive( address of Rc_Msg_Hdr, EGP_Q_ID )
        If status is M_QEMPTY
            Egp_Timeout = Update_Timeout - (Get_Time - start timer)
            Stats_Timeout = Stats_Timeout + ( Get_Time - start timer )
            Start timer = Get_Time
            If the min(Stats_Timeout) has not expired
                Set Event_flag to MSG_ARRIVE
                Event_Status = Wait_Timeout(Event flag,
                 min(Stats_Timeout, EGP_Timeout)
                If the event status indicates a timeout occurred
```

```
                    Set the Egp_Timeout to expired
                Endif
            Endif
            If the Stats_Timeout has expired
                Call Send_Egp_Stats
                Reset the Stats_Timeout
                Update the Egp_Timeout
            Endif
            If the Egp_Timeout has expired
                Call Egp_Update
                Reset the Egp_Timeout to thirty seconds
                Update the Stat_Timeout
                Start timer = Get_Time
            Endif
        Else
            Call Egp_Request
            Call Message_Discard( address of Rc_Msg_Hdr )
        Endif
    Endloop
Endif
```

3.3.3.4.3.5 Limitations

No limitations are defined for the Egp_Control unit.

3.3.3.4.4 Egp_Hi Unit

Egp_Hi generates a neigbhour reachability message and sends the message to the neighbour gateway.

3.3.3.4.4.1 Inputs

The following inputs are required by the Egp_Hi unit:

1) Code - This input parameter is a 32 bit integer which contains a neighbour reachability code.

2) Source - This input parameter is a 32 bit integer which contains an IP address indicating the source of the IP datagram.

3) Destination - This input parameter is a 32 bit integer which contains an IP address of the neighbour gateway to which the neighbour reachability message is sent.

4) Sequence - This global EGP input is a 16 bit integer which contains the current sequence number of the EGP_Datagram.

5) Egp_Ttr - This global EGP input points to the EGP datagram, which is contained in the Internet header pointed to by Ip_Ttl, of the global EGP Tx_Msg_hdr message buffer.

3.3.3.4.4.2 Outputs

The following output is produced by the Egp_Hi unit:

1) Sequence - This global EGP output is a 16 bit integer which contains the current sequence number of the EGP_Datagram.

2) Tx_Msg_Hdr - This global EGP output is a Message_Header for a message containing a Tx_Datagram.

3) Tx_Datagram - This global EGP output is an Dgram_Message which will contain a neighbour reachability EGP_Datagram.

3.3.3.4.4.3 Local Data

The following local data is defined for the Egp_Hi unit:

1) Length - This local data item is a 16 bit integer which contains the number of bytes in the EGP_Datagram of the Tx_Datagram.

2) Status - This local data item is a 16 bit integer which identifies either a successful or unsuccessful acquisition of the message header.

3.3.3.4.4.4 Processing

```
Status = Allocatep
If the Status is NOERROR
    Set the Tx_Datagram Type to EGP_HELLO
    Set the Tx_Datagram Code to the input Code
    Set the Tx_Datagram Info to the integer INFOR_UP
    Case Code
        HELLO:  Set the Tx_Datagram Sequence = Htons( Sequence number
                 incremented )
        IHU:    Set the Tx_Datagram Sequence = Htons( Sequence number )
    Endcase
    Set the Length to ten
    Sendit( Length, Source Destination )
Endif
Return Status
```

3.3.3.4.4.5 Limitations

No limitations are defined for the Egp_Hi unit.

3.3.3.4.5 Egp_Nrp Unit

Egp_Nrp generates a poll command and sends the message to the neighbour gateway.

3.3.3.4.5.1 Inputs

The following inputs are required by the Egp_Nrp unit:

1) Source - This input parameter is a 32 bit integer which contains an IP address indicating the source of the IP datagram.

2) Destination - This input parameter is a 32 bit integer which contains an IP address of the neighbour gateway to which the poll command is sent.

3) Sequence - This global EGP input is a 16 bit integer which contains the current sequence number of the EGP_Datagram.

3.3.3.4.5.2 Outputs

The following output is produced by the Egp_Nrp unit:

1) Status - This output function value is a 16 bit integer which identifies either a successful or unsuccessful completion.

2) Tx_Msg_Hdr - This global EGP output is a Message_Header for a message containing a Tx_Datagram.

3) Tx_Datagram - This global EGP output is a Dgram_Message which will contain a poll command.

4) Sequence - This global EGP output is a 16 bit integer which contains the current sequence number of the EGP_Datagram.

3.3.3.4.5.3 Local Data

The following local data is defined for the Egp_Nrp unit:

1) Length - This local data item is a 16 bit integer which contains the length of the EGP_Datagram in the Tx_Datagram.

3.3.3.4.5.4 Processing

```
Status = Allocatep
If the Status is NOERROR
    Set the Tx_Datagram Type to EGP_NRPOLL
    Set the Tx_Datagram Code to zero
    Set the Tx_Datagram Info to one
    Set the Tx_Datagram Sequence = Htons( Sequence number incremented )
    Set the Tx_Datagram IP_Source_Network = EGP_NET
    Set the Length to sixteen
    Sendit( Length, Source Destination )
Endif
Return Status
```

3.3.3.4.5.5 Limitations

No limitations are defined for the Egp_Nrp unit.

3.3.3.4.6 Egp_Request Unit

Egp_Request processes an EGP_Datagram, generating an appropriate response/indication when necessary.

3.3.3.4.6.1 Inputs

The following inputs are required by the Egp_Request unit:

1) Rc_Msg_Hdr - This global EGP input is a Message_Header for a message containing a Rc_Datagram.

2) Rc_Datagram - This global EGP input is a Dgram_Message message buffer which will contains one of a neighbour acquisition, neighbour reachability, network poll, network reachability, or error message.

3.3.3.4.6.2 Outputs

The following output is produced by the Egp_Request unit:

1) Ip_Rtr - This global EGP output consists of a pointer to the Internet header of the global EGP Rc_Msg_Hdr message buffer.

2) Egp_Rtr - This global EGP output consists of a pointer to the EGP datagram, which is contained in the Internet header pointed to by Ip_Rtr, of the Rc_Msg_Hdr message buffer.

3.3.3.4.6.3 Local Data

The following local data is defined for the Egp_Request unit:

1) NB_Entry - This local data item is a pointer to a entry which has been found or assigned in the EGP neighbour table NB_Table.

2) Format_String - This local data item is a pointer to an ASCII text string which contains the format of the message.

3.3.3.4.6.4 Processing

```
Set the pointer Egp-Ptr to the EGP_Datagram which immediately follows the
 Internet_Header in the Rc_Datagram
If the EGP_Datagram EGP_Version number is not two
    Return
Endif
If the Rc_Datagram contains an odd number of bytes as indicated by the
 total Rc_Datagram Internet_Header LEN field
    Add a zero filled byte to the end of the Rc_Datagram
    Increment the Rc_Datagram LEN field
Endif
Calculate the EGP_Datagram checksum = Calc_Cksum( EGP_Datagram pointer,
 Ntohs( Rc_Datagram Internet_Header LEN ) -
 4 * Rc_Datagram Internet_Header VHL )
If the EGP_Datagram checksum is not zero and not FFFF
    Return
Endif
Call Update_Egp_Stats( Ip_Rtr, Egp_Rtr, "R" )
NB_Entry = Finden( Rc_Datagram Internet_Header Src IP address )

Case on Rc_Datagram Type
    EGP_NA:         Call Neighbour_Acquisition( NB_Entry )

    EGP_NRPOLL:     Call Network_Poll( NB_Entry )

    EGP_NR:         Call Network_Reachability( NB_Entry )

    EGP_HELLO:      Call Neighbour_Reachability( NB_Entry )

    EGP_ERROR:      Call Oi_Print( Format_String, IP Address of the host
                from which EGP packet was received )
               Call Oi_Print( Format_String, Rc_Datagram Code,
                 Rc_Datagram Info, Htons( Rc_Datagram Reason ) )
    Default:    Call Oi_Print( Format_String, IP Address of the host
                 from which EGP_Datagram was received )
                Call Oi_Print( Format_String, Rc_Datagram Type,
                 Rc_Datagram Code, Rc_Datagram Info, Rc_Datagram AS )
                Endcase
Return
```

3.3.3.4.6.5 Limitations

No limitations are defined for the Egp_Request unit.

3.3.3.4.7 Egp_Update Unit

Egp_Update updates the states for each of the neighbour gateways listed in the neighbour table. The latest main neighbour with an UP status is polled for an EGP update message.

3.3.3.4.7.1 Inputs

The following inputs are required by the Egp_Update unit:

1) NB_Table - This global data input contains a list of neighbour gateways that implement EGP.

2) NB_Router - This global EGP input is an entry of the NB_Table with which the IGW communicates via EGP.

3) Route_Timer - This global EGP input is a 32 bit integer which indicates the last time an update message was received.

3.3.3.4.7.2 Outputs

The following output is produced by the Egp_Update unit:

1) NB_Table - This global data output contains a revised list of neighbour gateways that implement EGP.

2) NB_Router - This global EGP output is an entry of the NB_Table with which the IGW communicates via EGP.

3.3.3.4.7.3 Local Data

The following local data is defined for the Egp_Update unit:

1) NB_Main - This local data item is a NB_Table entry which contains the NB_Table entry defined as the latest main neighbour.

3.3.3.4.7.4 Processing

```
Set the current time = Get_Time
For all entries in the NB_Table
    If the entry Flags is not NB_VALID
        Continue with the next NB_Table entry
    Endif

    If the entry Time has timed out with respect to the current time
        If the entry Flags is NB_UP
            If the entry Flags is NB_REMOTE ( neighbour initiated
             acquisition )
                Mark the entry as deleted by clearing all Flags
                Break
            Endif
            Remove the STATE( NB_ACQUIRING, NB_UP, NB_DOWN )
             and NB_ROUTER Flags
            Set the NB_DOWN Flags
            Reset the entry Time to current time
        Endif
        If the entry Flags is NB_ACQUIRING
            Remove the STATE( NB_ACQUIRING, NB_UP, NB_DOWN )
             and NB_ROUTER Flags
            Set the NB_DOWN Flags
            Reset the entry Time to the current time
        Endif
        If the entry Flags is NB_DOWN
            Remove the STATE( NB_ACQUIRING, NB_UP, NB_DOWN )
             and NB_ROUTER Flags
        Endif
    Endif

    If the Flags main neighbour NB_MAIN_NEIGHBOUR is set and
     neither NB_UP nor NB_DOWN are set
        Set the NB_ACQUIRING Flags
        Call Egp_Ac( CODE REQUEST, EGP_ADDR, NB_Table IP_Address )
    Else
        If the flag for this entry is already NB_ACQUIRING
            Call Egp_Ac( CODE REQUEST, EGP_ADDR, NB_Table IP_Address )
            Endif
    Endif

    If the NB_Main has not been determined and the current NB_Table
     entry Flags is defined as a NB_MAIN_NEIGHBOUR and NB_UP
        Set NB_Main to the current NB_Table entry
    Endif

    If the NB_Table entry Flags is defined as NB_UP
        Call Egp_Hi( CODE HELLO, EGP_ADDR, NB_Table IP_Address )
    Endif
Endfor

If the NB_Main is defined and the NB_Router is not defined as NB_Main
    Clear NB_CURRENT Flag referenced by NB_Router
    Set NP_Router to NB_Main
    Set NB_Current Flag referenced by NB_Router
Endif

Time since the last update = Get_Time - Route_Timer
If the time since the last update > EGP_ROUTE_TIME
    Call Egp_Nrp(  EGP_ADDR, IP_Address in the NB_Router )
Endif
Return
```

3.3.3.4.7.5 Limitations

No limitations are defined for the Egp_Update unit.

3.3.3.4.8 Finden Unit

Finden locates an EGP neighbour table entry as identified by an IP address. When the entry cannot be found, the first EGP neighbour table entry marked for delete is assigned or when the table is already full, a zero value is returned.

3.3.3.4.8.1 Inputs

The following inputs are required by the Finden unit:

1) IP_Address - This input parameter is a 32 bit integer which contains an IP address of the entry to be found in the EGP neighbour table NB_Table.

2) NB_Table - This global data input is a table of EGP neighbour entries.

3.3.3.4.8.2 Outputs

The following output is produced by the Finden unit:

1) Location - This output function parameter is a pointer to an EGP neighbour table entry. A value of zero indicates an existing entry is not in the table and the table is full.

2) NB_Table - This global data output is a table of EGP neighbour entries.

3.3.3.4.8.3 Local Data

No local data is defined for the Finden unit.

3.3.3.4.8.4 Processing

```
Assign Location zero
For each entry in the NB_Table
    If Flags equals zero indicating the entry is marked for delete
        If an entry marked for delete has not been located earlier
            Assign the Location this NB_Table entry
        Endif
        Continue with the next NB_Table entry
    Endif

    If the IP_Address matches the NB_Table entry Gateway IP address
       Set the Location to the current NB_Table entry
       Return the Location
    Endif
Endfor
```

```
If an entry marked for delete has been located
    Set the NB_Table entry gateway address to the IP_Address
    Set the NB_Table entry Flags to NB_VALID
    Set the Location to the NB_Table entry
Endif
Return the Location
```

3.3.3.4.8.5 Limitations

No limitations are defined for the Finden unit.

3.3.3.4.9 Neighbour_Acquisition Unit

Neighbour_Acquisition processes the request, cease, reply, refuse, and cease acknowledgement forms of the neighbour acquisition EGP messages.

3.3.3.4.9.1 Inputs

The following inputs are required by the Neighbour_Acquisition unit:

1) NB_Entry - This input parameter is a pointer to a entry which has been found or assigned in the neighbour table NB_Table.

2) NB_Table - This global data input is a table of EGP neighbour entries.

3) Rc_Msg_Hdr - This global EGP input is a Message_Header for a message containing a Rc_Datagram.

4) Rc_Datagram - This global EGP input is an Dgram_Message which will contain a neighbour acquisition EGP_Datagram.

5) Ip-Rtr - This global EGP input points to the IP datagram in the global EGP Rc_Msg_Hdr message buffer.

6) Egp_Rtr - This global EGP input points to the EGP datagram, which is in the IP datagram pointed to by IP-Rtr, in the global EGP Rc_Msg_Hdr message buffer.

3.3.3.4.9.2 Outputs

No output is produced by the Neighbour_Acquisition unit.

3.3.3.4.9.3 Local Data

The following local data is defined for the Neighbour_Acquisition unit:

1) Format_String - This local data item is a pointer to an ASCII text string which contains the format of the message.

3.3.3.4.9.4 Processing

```
Case on Rc_Datagram Code
    CODE REQUEST:
              If the NB_Entry is found or assigned a NB_Table
                    entry
                  If the NB_Table entry Flags is NB_VALID only
                      Add the NB_REMOTE NB_Table Flags
                  Endif
                  Set the NB_Table entry Flags to the NB_UP state
                  Set the NB_Table entry Time = Get_Time
                  Re-set the Ceases counter to zero
                  Call Egp_Ac( CODE REPLY, Rc_Datagram Internet_Header
                   Dst IP address, Rc_Datagram Internet_Header
                   Src IP address )
              Else
                  Call Egp_Ac( CODE REFUSE, Rc_Datagram
                      Internet_Header Dst IP address,  Rc_Datagram
                      Internet_Header Src IP address )
                 Endif

    CODE CEASE:
                 If the NB_Entry is found or assigned a NB_Table entry
                     If the NB_REMOTE Flags is set
                         Mark the NB_Table entry as deleted by setting
                          the Flags to zero
                     Else
                         Remove the STATE(NB_UP, NB_DOWN, NB_ACQUIRING) and
                          NB_ROUTER Flags
                         Set the NB_Table entry Flags to the NB_DOWN state
                         Set the NB_Table entry Time = Get_Time
                         Increment the Ceases counter
                         If the entry Ceases counter exceeds four
                             Call Oi_Print( Format_String )
                             Break
                         Endif
                     Endif
                 Endif
                 Call Egp_Ac( CODE CACK, Rc_Datagram Internet_Header
                  Dst IP address,  Rc_Datagram Internet_Header Src
                  IP address )

    CODE REPLY:
                 If the NB_Table entry does not exist
                     Break
                 Endif
                 Set the NB_Table entry Time = Get_Time
                 Remove the NB_DOWN and NB_ACQUIRING Flags
                 Set the NB_Table Flags to the NB_UP state

    CODE REFUSE:
                 If the NB_Table entry does not exist
                     Break
                 Endif
                 Set the NB_Table entry Time = Get_Time
                 Remove the NB_DOWN, NB_ACQUIRING, and NB_ROUTER Flags
                 Set the NB_DOWN Flags

    CODE CACK:
                 If the NB_Table entry does not exists
                     Break
                 Endif
                 Set the NB_Table entry Time = Get_Time
                 Remove the NB_DOWN, NB_ACQUIRING, and NB_ROUTER Flags
                 Set the NB_DOWN Flags
Endcase
```

Return

3.3.3.4.9.5 Limitations

No limitations are defined for the Neighbour Acquisition unit.

3.3.3.4.10 Neighbour_Reachability Unit

Neighbour_Reachability returns an I-HEARD-YOU in response to a HELLO EGP_Datagram.

3.3.3.4.10.1 Inputs

The following inputs are required by the Neighbour_Reachability unit:

1) NB_Entry - This input parameter is a pointer to a entry which has been found or assigned in the neighbour table NB_Table.

2) NB_Table - This global data input is a table of EGP neighbour entries.

3) Rc_Msg_Hdr - This global EGP input is a Message_Header for a message containing a Rc_Datagram.

4) Rc_Datagram - This global EGP input is a message buffer which will contain a neighbour acquisition EGP_Datagram.

5) Ip_Rtr - This global EGP data input points to the IP datagram in the global EGP Rc_Msg_Hdr message buffer.

6) Egp_Rtr - This global EGP data input points to the EGP datagram, which is contained in the IP datagram pointed to by IP_Rtr, of the global EGP Rc_Msg_Hdr message buffer.

3.3.3.4.10.2 Outputs

No output is produced by the Neighbour_Reachability unit.

3.3.3.4.10.3 Local Data

No local data is defined for the Neighbour_Reachability unit.

3.3.3.4.10.4 Processing

```
If the NB_Entry is zero
    Call Egp_Ac( CODE CEASE, EGP_ADDR, Rc_Datagram
     Internet_Header Src IP address )
Else

    /* The Neighbour Table entry is found or assigned. */

    If the NB_Table entry Flags indicate the state is NB_UP
        Reset the entry Time = Get_Time
    Endif
    If the Rc_Datagram EGP datagram code is a HELLO message
    Call Egp_Hi( CODE IHU, Rc_Datagram IP datagram Dst IP
            address, Rc_Datagram IP datagram Src IP address )
Endif
Return
```

3.3.3.4.10.5 Limitations

No limitations are defined for the Neighbour_Reachability unit.

3.3.3.4.11 Network_Poll Unit

Network_Poll builds an Update message which is sent to the polling neighbour gateway.

3.3.3.4.11.1 Inputs

The following inputs are required by the Network_Poll unit:

1) NB_Entry - This input parameter is a pointer to a entry which has been found or assigned in the neighbour table NB_Table.

2) NB_Table - This global data input is a table of EGP neighbour entries.

3) Net_Table - This global data input is a table of Network entries.

4) GW_Table - This global data input is a table of Gateway entries.

5) Rc_Msg_Hdr - This global EGP input is a Message_Header for a message containing a Rc_Datagram.

6) Rc_Datagram - This global EGP input is an Dgram_Message which contains a received poll command.

7) IP_Rtr - This global EGP input points to the Internet header in the global EGP Rc_Msg_Hdr message buffer.

8) Egp_Rtr - This global EGP input points to the EGP datagram, which is contained in the Internet header

pointed to by IP_Rtr, of the global EGP Rc_Msg_Hdr message buffer.

9) Egp_Ttr - This global EGP input points to the EGP datagram, which is contained in the Internet header pointed to by IP_Ttr, of the global EGP Tx_Msg_Hdr message buffer.Om

3.3.3.4.11.2 Outputs

The following output is produced by the Network_Poll unit:

1) Tx_Msg_Hdr - This global EGP output is a Message_Header for a message containing a Tx_Datagram.

2) Tx_Datagram - This global data output is a message buffer which will contain an update message.

3.3.3.4.11.3 Local Data

No local data is defined for the Network_Poll unit.

3.3.3.4.11.4 Processing

```
If the NB_Entry is zero
    Call Egp_Ac( CODE CEASE,EGP_ADDR, Rc_Datagram Internet_Header
     Src IP address )
    Return
Endif
Status = Allocatep
If status is not NOERROR
   Return
Endif
Define the Tx_Datagram Internet_Header Dst IP address as the
 Rc_Datagram Internet_Header Src IP address
Define the Tx_Datagram Type as EGP_NR
Define the Tx_Datagram Code as zero
Define the Tx_Datagram Info as zero
Define the Tx_Datagram Sequence number as the Sequence number
 of the Rc_Datagram
Define the Tx_Datagram IP_Source_Network number as the Rc_Datagram
 IP_Source_Network number
Set the Tx_Datagram Length to 16 bytes
Set a pointer to the EGP_Datagram section of the Tx_Datagram
Set the number of exterior gateways Ext_Gw to zero
If the first byte of the IP_Source_Network does not correspond
 to the EGP_CORE_NET
    Set the number of interior gateways Int_Gw to zero
Else

    /* Identify the networks this gateway knows about */

    Set the Gateway IP address = Net_Class( gateway address byte )
    Set the number of interior gateways Int_Gw to one
    Set the number of distances Gw_Distance to one
    Set the Distance to zero
    For each entry in the Net_Table marked NET_UP and NET_VALID
        Increment the Num_Nets
    Endfor
    For each entry in the Net_Table
        If the Net_Table entry is NET_UP and NET_VALID
            Class = Net_Class( Net_Table network address )
            Place class bytes of the Net_Table network address in the
             Tx_Datagram Net_Address field
        Endif
    Endfor

    /* Identify the networks in the gateway table reachable from this
       gateway */

    For each entry in the GW_Table
        If the GW_Table entry is GW_VALID
            Increment the Gw_Distances
            Place the Hop count in the Tx_Datagram Distance field
            Set the Num_Net to one in the Tx_Datagram
            Class = Net_Class( GW_Table Network_Address )
            Place class bytes of the GW_Table Network_Address
             in the Tx_Datagram Net field
        Endif
    Endfor

    /* Send the completed Update message to the polling gateway */

    Calculate the final message Length of the Tx_Datagram
    If the message Length is odd
        Add one additional byte to the Tx_Datagram
        Increment Tx_Datagram Length by one
    Endif
    Call Sendit( Length, the Rc_Datagram Internet_Header Dst IP
     address, Rc_Datagram Internet_Header Src IP address )
Endif
Return
```

3.3.3.4.11.5 Limitations

No limitations are defined for the Network_Poll unit.

3.3.3.4.12 Network_Reachability Unit

Network_Reachability processes an Update message from the router gateway neighbour.

3.3.3.4.12.1 Inputs

The following inputs are required by the Network_Reachability unit:

1) NB_Entry - This input parameter is a pointer to a entry which has been found or assigned in the neighbour table NB_Table.

2) NB_Router - This global EGP input is an entry of the NB_Table with which the IGW communicates via EGP.

3) NB_Table - This global data input is a table of EGP neighbour entries.

4) Rc_Msg_Hdr - This global EGP input is a Message_Header for a message containing a Rc_Datagram. 5) Rc_Datagram - This global EGP input is a message buffer which will contain a neighbour acquisition EGP message.

3.3.3.4.12.2 Outputs

The following output is produced by the Network_Reachability unit:

1) Router_Timeout - This global EGP output is a 32 bit integer which contains the time at which the last route update was received.

3.3.3.4.12.3 Local Data

No local data is defined for the Network_Reachability unit.

3.3.3.4.12.4 Processing

```
If the NB_Table entry was not found
    Call Egp_Ac( CEASE, EGP_ADDR, Rc_Datagram Internet_Header Src IP address )
Else
    If the NB_Table entry is not the NB_Router entry
        Return
    Endif
    Reset the Router_Timeout = Get_Time
    Call Route_Update
Endif
Return
```

3.3.3.4.12.5 Limitations

No limitations are defined for the Network_Reachability unit.

3.3.3.4.13 Route_Update Unit

Route_Update processes a received Update message. The gateway and network address pairs are extracted from the Update message and placed in the EGP Route Table which is to be sent to the IP TLC.

3.3.3.4.13.1 Inputs

The following inputs are required by the Route_Update unit:

1) Rc_Datagram - This global EGP input is a EGP_Datagram which contains a network reachability Update message used to revise the EGP_Table.

2) EGP_Table - This global EGP data input is the address of EGP route table entries. Each entry consists of one pair of gateway and network IP addresses which are required by the IP TLC.

3) Egp_Ptr - This global EGP input points to the EGP datagram in the global EGP data Rc_Msg_Hdr message buffer.

3.3.3.4.13.2 Outputs

The following output is produced by the Route_Update unit:

1) EGP_Table - This global EGP data output is the address of EGP route table entries. Each entry consists of one pair of gateway and network IP addresses which are required by the IP TLC.

3.3.3.4.13.3 Local Data

No local data is defined for the Route_Update unit.

3.3.3.4.13.4 Processing

```
Indicate no changes to the EGP_Table
Class = Net_Class( Rc_Datagram IP_Source_Network )
Set a pointer to the first Gateway address in the Rc_Datagram
 Gateway_List

Total the Int_Gw interior and Ext_Gw exterior gateways in the
 Rc_Datagram
For each Gateway in the Gateway_List of the Rc_Datagram
    Concatenate the Gateway host address to the IP_Source_Network
     IP address
    If changes to the EGP_Table have not been indicated
        If this Gateway IP address is not the Gw_Address in the
         EGP_Table
            Indicate changes to the EGP_Table
        Endif
    Endif

    Obtain the number of Gw_Distances for this Rc_Datagram Gateway
    For each of the GW_Distances of the Gateway
        Obtain the first Distance
        Obtain the number of nets Num_Nets for this Gateway at this
         Distance
        For each of the Num_Net Net_Addresses
            Set the EGP_Table Gw_Address to the Rc_Datagram Gw_Address
            Class = Net_Class( Rc_Datagram Net_Address )
            For each of the class bytes in the Rc_Datagram Net_Address
                If corresponding bytes in the Net_Address fields of
                 both the Rc_Datagram and EGP_Table do not match
                    Indicate changes to the EGP_Table have occurred
                Endif
                Copy Net_Address byte from the Rc_Datagram to the
                 EGP_Table
            Endfor
            Fill the EGP_Table Net_Address host part with zeroes
            Increment counter of entry total in EGP Route Table
            If the EGP_Table is not full
                Increment the EGP_Table position
            Endif
        Endfor
    Endfor
Endfor

Set the current EGP_Table Gw_Address to zero
Set the current EGP_Table Net_Address to zero
If EGP_Table changes are indicated
    Status = Message_Get( address of the Rt_Msg_Hdr )
    Set the RT_Msg_Hdr Offset to IP_ROUTE_INIT
    Set the RT_Msg_Hdr Length to zero
    Status = Message_Send(  address of IP_QUEUE, RT_Msg_Hdr )
    If status not equal NOERROR
        Call Message_Discard( RT_Msg_Hdr )
    Endif
Endif
Return
```

3.3.3.4.13.5 Limitations

No limitations are defined for the Route_Update unit.

3.3.3.4.14 Send_Egp_Stats Unit

Send_Egp_Stats sends the EGP_Stats_Table message buffer to the STAT TLC and resets the EGP_Stats_Table descriptor and list of entries to an empty state.

3.3.3.4.14.1 Inputs

The following inputs are required by the Send_Egp_Stats unit:

1) Stat_Msg_Hdr - This global EGP input is a Message_Header for a message containing an Egp_Stats_Table.

2) Egp_Stats_Table - This global EGP input is a Stats_Buffer which contains a list of EGP_Stats_Entry records.

3.3.3.4.14.2 Outputs

The following output is produced by the Send_Egp_Stats unit:

1) Stat_Msg_Hdr - This global EGP output is a Message_Header for a message containing an Egp_Stats_Table.

2) Egp_Stats_Table - This global EGP output is a Stats_Buffer which contains a list of EGP_Stats_Entry records.

3) Stats_Timeout - This global EGP output is a 32 bit integer which maintains the periodic time interval for the update of the EGP statistics.

3.3.3.4.14.3 Local Data

No local data is defined for the Send_Egp_Stats unit.

3.3.3.4.14.4 Processing

```
/* Send the message buffer */

Call Message_Send( address of Stat_Msg_Hdr, STAT_Q_ID )

/* Create a new message buffer for the EGP statistics */

Call Message_Get( address of Stat_Msg_Hdr )
Set the Stat_Msg_Hdr M_Offset to the number of bytes in the
 Egp_Stat_Table Descriptor field
Set the Stat_Msg_Hdr M_Length to the Stat_Msg_Hdr M_Offset
Set the Egp_Stat_Table Descriptor Entries to zero
Set the Egp_Stat_Table Descriptor Operation to UPD
Set the Egp_Stat_Table Descriptor Statistic to EGP

/* Reset the timer */

Reset the Stats_Timeout to STAT_TIME_OUT
Return
```

3.3.3.4.14.5 Limitations

No limitations are defined for the Send_Egp_Stats unit.

3.3.3.4.15 Sendit Unit

Sendit completes both the message header and internet header fields in the Tx_Datagram message buffer. The message is sent to the IP TLC for transmission the neighbour gateway and the statistics are taken.

3.3.3.4.15.1 Inputs

The following inputs are required by the Sendit unit:

1) Length - This input parameter is a 16 bit integer which contains the number of bytes in the EGP_Datagram.

2) Source_Ipadr - This input parameter is a 32 bit integer which identifies the gateway from which the message is generated.

3) Destin_Ipadr - This input parameter is a 32 bit integer which identifies the gateway to which the message is to be sent.

4) Tx_Msg_Hdr - This global EGP input is a Message_Header for a message containing a Tx_Datagram.

5) Tx_Datagram - This global EGP input is an Dgram_Message which contains one of a neighbour acquisition, neighbour reachability, network poll, network reachability, or error message.

6) Ip_Ttr - This global EGP input points to the Internet header of the global EGP Tx_Msg_Hdr message buffer.

7) Egp_Ttr - This global EGP input points to the EGP datagram, which is contained in the Internet header pointed to by IP_Ttr, of the global EGP Tx_Msg_Hdr message buffer.

3.3.3.4.15.2 Outputs

The following output is produced by the Sendit unit:

1) Tx_Msg_Hdr - This global EGP output is a Message_Header for a message containing the Tx_Datagram which is sent to the IP TLC.

2) Tx_Datagram - This global EGP output is the completed Dgram_Message which contains one of a neighbour acquisition, neighbour reachability, network poll, network reachability, or error message.

3.3.3.4.15.3 Local Data

No local data is defined for the Sendit unit.

3.3.3.4.15.4 Processing

```
/* Complete the definition of the datagram internet header */

Define the Tx_Datagram Internet_Header Src IP address as the
 input Source_Ipadr
Define the Tx_Datagram Internet_Header Dst IP address as the
 input Destin_Ipadr
Define the version and length subfields of the Tx_Datagram
 Internet_Header VHL as four and five respectively
Define the Tx_Datagram Internet_Header TTL as 10
Define the Tx_Datagram Internet_Header Protocol as eight for EGP
Define the Tx_Datagram Internet_Header LEN as Htons( twenty bytes in the
 Tx_Datagram Internet_Header plus Length bytes in the EGP_Datagram )
Set the Tx_Datagram Internet_Header Chksum to zero
Define the Tx_Datagram Internet_Header Chksum = Calc_Cksum( Tx_Datagram
 Internet_Header pointer, ten 16 bit words in the Internet_Header )

/* Complete the definition of the datagram EGP datagram */

Define the Tx_Datagram EGP_Version as two
Define the Tx_Datagram AS as Htons( EGP_AS )
Set the Tx_Datagram Calc_cksum to zero
Define the Tx_Datagram Checksum = Cksum( address of EGP_Datagram,
 number of 16 bit words in the EGP_Datagram is Length divided by two )

/* Complete the definition of the message header */

Set Tx_Msg_Hdr Offset to twenty
Set Tx_Msg_Hdr Length to the Tx_Msg_Hdr Offset and Length of EGP
 datagram
Update_Egp_Stats( IP_Ttr, Egp_Ttr, "T" )
Status = Message_Send( IP_QUEUE,address of Tx_Msg_Hdr )
Return
```

3.3.3.4.15.5 Limitations

No limitations are defined for the Sendit unit.

3.3.3.4.16 Update_Egp_Stats Unit

Update_Egp_Stats adds an entry to the Egp_Stats_Table. When the Egp_Stats_Table is full, the list of entries is sent to the STAT TLC.

3.3.3.4.16.1 Inputs

The following inputs are required by the Update_Egp_Stats unit:

1) Direction - This input parameter is a byte which indicates the acquisition state of the Dgram_Datagram as one of the following single characters:

   R - The Dgram_Datagram has been received by EGP TLC.

   T - The Dgram_Datagram has been transmitted by EGP TLC.

2) Ip_Ptr - This input parameter is a pointer to the IP datagram in either the global receive or transmit IP datagram in the global EGP Rc_Msg_Hdr or Tx_Msg_Hdr message buffer.

3) Egp_Ptr - This input parameter is a pointer to the EGP datagram in either the global receive or transmit IP datagram in the global EGP Rc_Msg_Hdr or Tx_Msg_Hdr message buffer.

4) Stat_Msg_Hdr - This global EGP input is a Message_Header for a message containing the Egp_Stats_Table.

5) Egp_Stats_Table - This global EGP data input is a message buffer which contains a descriptor followed by a list of Egp_Stats_Entry records.

3.3.3.4.16.2 Outputs

The following output is produced by the Update_Egp_Stats unit:

1) Stat_Msg_Hdr - This global EGP output is a Message_Header which contains a pointer to the Egp_Stats_Table message buffer.

2) Egp_Stats_Table - This global EGP data output is a message buffer which contains a descriptor followed by a list of Stat_Table_Entry records.

3.3.3.4.16.3 Local Data

No local data is defined for the Update_Egp_Stats unit.

3.3.3.4.16.4 Processing

```
Increase the Stat_Msg_Hdr byte count M_Length by nine
Set the Egp_Stats_Table entry Entries fields to the Direction;
 the Dgram_Datagram fields Internet_Header Src IP address as
 pointed to by Ip_Ptr; and the AS, the Type, and the Code as
 pointed to by Egp_Ptr
Increment the EGP_Stats_Table Descriptor Entries
If the EGP_Stats_Table Descriptor Entries equals STATS_RESOURCES
    Call Send_Egp_Stats
Endif
Return
```

3.3.3.4.16.5 Limitations

No limitations are defined for the Update_Egp_Stats unit.

SOFTWARE DETAILED DESIGN DOCUMENT FOR THE INTER-NETWORK GATEWAY PROJECT