SUBSYSTEM:	Local Area Network Controller
COMPONENT:	Transport Layer
PLANNED RELEASE:	Mod400 Release 4.0
SPECIFICATION REVISION NUMBER:	Working Draft
DATE:	January 8,1986
AUTHOR:	ken C. Yu

This specification describes the current definition of the subject software component, and may bye revised in order to incorporate design improvements.

TABLE OF CONTENTS

REFERENCES

1. 09-0016-00 Bridge Communications,Inc. ESPL, Software Technical Reference Manual, Vol 1,Kernel and Support Software

2. 6014981 Enginerring Product Specification LAN Software EPS-1, R. Dhondy, November,1985

3. 60149766 Engineering Product Specification, Local Area Controller Subsystem (LACS) EPS-1, A. Hirtle, March,1985

4. 60149824 DPS6 Local Area Network Controller PFS

5. LAN Software Component Specification,Interface Software direct Memory Acess, K. Yu, July,1985

6. LAN Software Component,Specification, Interface Software I/O Dispatcher, K. Yu, July,1985

7. ISO 7498, Open System Interconnection,Basic Reference Model

8. ISO 8072, OSI Services of the Transport Layer

9. ISO 8073, OSI Transport Protocol Specification

10. ISO Draft Proposed Addendum to ISO 8073 to enable Class Four Operation over Connectionless Mode Service as defined in ISO 8348,DAD 1

11. LAN Software Component Specification, System Management D.OShaughnessy, August,1985

12. LAN Software Component Specification, Lacs Driver Megabus Services, P. Stopera, Aug. 16,1985

13. LAN Software Component Specification, Lacs Driver Interface Services, P. Stopera,Aug. 16, 1985

14. ISO 8473 Connectionless Network Specification

1. INTRODUCTION AND OVERVIEW

1.1 BACKGROUND

ISO Class 4 Transport component is required to develop as part of the software to support the Local Area Network Controller Subsystem (Lacs). It provides a connection oriented services to the Session Layer or similar user above this transport layer and uses the services of the connectionless network layer below this transport layer. The understanding of the ISO/DIS 8073 specification is a prerequisite to this component specification. The implementation of this module is based on DDI CLass 4 ISO Transport Service module. This component describes the area in the product that is outside the DDI transport module. No attempt is made to document the design of the DDI transport module at the time. However, the documents should be available in the next release. Although this component will be fully conformant with the ISO/DIS 8072 and 8073 specifications, it can be used as the sub-net layer to provide a connection oriented subnetwork services to user who requires a reliable link connection service. Therefore , any descriptions on connectionless network services in this document will also apply to logical link control service

1.2 BASIC PURPOSE

The basic purpose of this component is to describe the designs of the ISO Class 4 Transport services, ISO layer 4 which basically provides the following sevices to users.

* Connection Establishment
  To establish a transport connection between two transport users.

* Data Transfer
  Manages a normal and expedited data transfer between two transport users. Functions such as error detection,error recovery,segmenting and reassembly,flow control are applied all the times.

* Connection Release
  To disconnect the transport connection

This transport layer uses the connectionless network layer to communicate with its peer layer.

A transport layer management service is incorporated in this component to communicate with System Management in the Lacs. This service responds to the System Management primitives such as read statistics, tsap creation and deletion and initiates event indication in case of an unusual event occurs.

1.3 BASIC STRUCTURE

Figure 1 shows the basic structure of the Transport layer. It shows the relateship with its external interfaces as well as its internal structure.

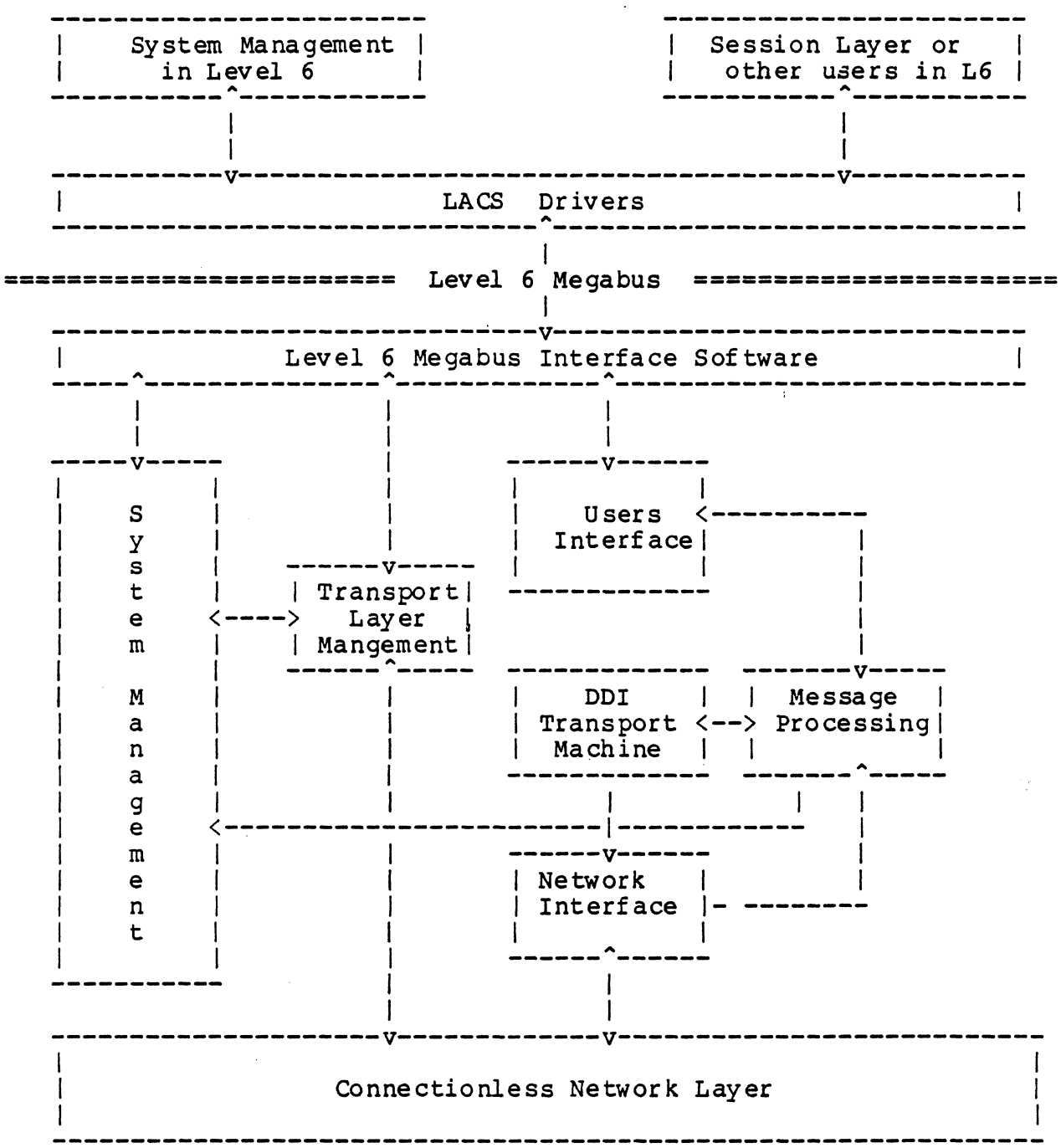

Figure 1 Transpart Layer Interface structure

The Transport Layer is phsically split across the Level 6 Megabus into two separate functions. The LACS Driver in the Level 6 provides the interface to the Transport users to access the transport service provider in the controller Requests are made to LACS Driver via $RQIO/LNJ interface. The LACS Driver then issues the LCB to the transport layer in controller to request specific action. The LCB contains primitives to request the transport layer to establish transport connection, data transfer or release transport connection. As shown in Figure 1 the transport layer consists of 2 separate processes with DDI Transport as its kernel. It provides interfaces to the Session layer in 16 via LACS driver, the System Management and the connectionless network layer in the LACS.

1.4 OPERATION

Before any connection establishment request can be honored the transport layer creates all local and remote tsap tables which are specified by the System Management in the LACS. Next the user issues an activate local and remote TSAP calls to identify the potential link between this local and a peer tsap.An activate remote call will get a logical remote tsap address back which must be used in the subsequent connect request primitive.

Before data can be transferred between a local tsap and a remote tsap the user must issue a connect request primitive to establish a transport connection between two tsaps. The transport layer then performs those functions necessary to establish a connection between two tsap. Negotiation with the remote peer entity determines protocol data unit size; selecting function to use during data transfer phase; identify different transport connections. Upon the completion of connection the user will be informed by returning the connect request primitive with a connection identifier which must be used in all subsequent data transfer primitives for this transport connection.

When a remote connect request arrives,the Transport layer will locate the local TSAP and identify the corresponding tsap in the remote tsap table and inform the user of this connection request by meant of the user supplied tsap event lcb. If the TSAP is not found in the remote table and the local tsap permits dynamic remote tsap creation, it will create a remote tsap and add the entry into its remote table for this local tsap. The dynamic configuration will not be suported until next release.

Transport will provide normal data transfer services as well as expedited data transfer. The Write CO data or Read Co data primitives allow the user to transfer data to a peer user on a transport connection. The transport will use segmenting and reassembling,concatenating and separation,flow control,error detection,transport connection identification,error recovery, normal and expedited data transfer function to accomplish the data transfer function.

Transport will release the transport connection on receiving the disconnect request primitive locally or remotely. When a remote disconnect request is received the transport will inform the user of the disconnect via the event connection indication LCB

1.4.1 Memory Management

In general each process is responsible for its memory allocation and memory release. In the Transport Process when a tpdu is sent to the network layer, it will give a copy of the buffer descriptor to the network layer with no confirmation expected from network layer. The transport process will keep that tpdu until it is acknowledged by its peer. In the receive function incoming tpdus are relayed to level 6 after processing. It is the responsibiltiy of the transport process to release the buffer to memory pool. Currently DDI copies the user data into its own buffer for retransmission and awaiting acknowledgement purposes. before returning the buffer to the user. This must be changed to use the prepend/append kernel to improve performance. Each time a protocal data unit is coming from Level 6 the transport process must allocate the data buffer big enough for the pdu as well as for all the header and padding areas that each layer may use to append its header information before passing down to next layer. This will improve performance and efficiency.

1.4.2 Flow Control

The transport layer will comply the flow control principles specified in the Lan Software EPS and ISO transport protocol specifications. The flow control of the transport layer between the local tsap and its remote tsap is in the DDI transport module. There is no flow control between the network layer and this transport layer.

1.4.2.1 Flow control of tsap event

One and only one tsap event LCB is queued in a local tsap at any given time. Addition tsap event LCB arrives will cause the previous LCB be returned with the new mask but without any event indication. The only event request mask that this transport supports is the connection indication and tsap deactivated.

1.4.2.2 Flow control of write CO data call

There is one write credit count for each local tsap. The tsap write credit count represent the number of additional write data primitives that a user can issue on this tsap. There are two two write credit counts for the connection: one for normal data and one or expedited data. They represent the number of additional write normal data and expedited data on this connection. The summation of the connection credits must equal to the tsap write credit count. The flow control between the user and the transport is handled by the transport server in the Level 6. Credits oversubscription is also handled by the Level 6 transport server. Therefore write credits cannot be exceeded within this transport layer. The initial credit is allocated by the system management. Each write primitive causes the count be decremented and each write completion will increment the count as well as return a credit to user. Any write primitive will be returned with status indicating credit exceeded when the tsap write credit is equal to zero. Initial write credits returned in the connection indicate or connection request confirmation will be the smaller of the system administratively set and the connect request/connect confirm primitives.

1.4.2.3 Flow control of read CO data call

There is one read credit count for each local tsap. This credit represents the read data pending orders outstanding. The read credits must be equal to all connection credits for this tsap. There are two read credits counts for each connection: one for normal read data and one for expedited read data.
Any read data primitive issued to this local tsap will be returned if the tsap read credit count become zero. Each read order completion will increment the tsap read credit count. The transport will not acknowledge the protocol data unit until it has been transferred to the user,s buffer in the Level 6 specified in the read order LCB. Therefore this transport does not buffered any sdu in the controller and this maybe be done at the transport layer server in level 6.

1.4.2.4 Flow control between DDI and Transmit data flow

There is an explicit flow control between the DDI and Transmit function. When DDI transport module cannot transmit data because of its peer flow control it shall call the Transmit function to stop sending data until its peer acknowledges or increases its credits at later time.
The Transmit function will not issue any data request until it receives a resume message from DDI transport although it may still move data across the Megabus to be queued in it transmit queue.

1.4.2.5 Flow control between DDI and Receive data flow

There is a mechanism to flow control the DDI incoming data for the user. If for some reasons there is no buffer pending or buffer size is insufficient for the protocol data unit, the receive function will not return a favorable indication to DDI to accept the pdu when DDI indicates data is available for user. This will control the data flow until the condition is cleared.

1.5 Statistics

The transport layer maintains statistics on a tsap basis. Each local tsap when created contains counts such as number of data sent, number of TPDU resent and transport protocol error. These counts will forever increment. System Mangement may issue read statistics to gather all these statistics.

1.6 Timers

The transport layer maintains five timers for its transport operation usages. The fives timers are: retransmission, window, inactivity, reference and giveup timers. The values of these timers are the responsibility of the System Management. They must be setup during local tsap creation. System Mangement may issue read attributes to gather these values.

2. EXTERNAL SPECIFICATION

2.1 Owned Data Structures

The ISO transport layer uses several data structures in it operation. These structures are illustrated in Figure 2 as shown below

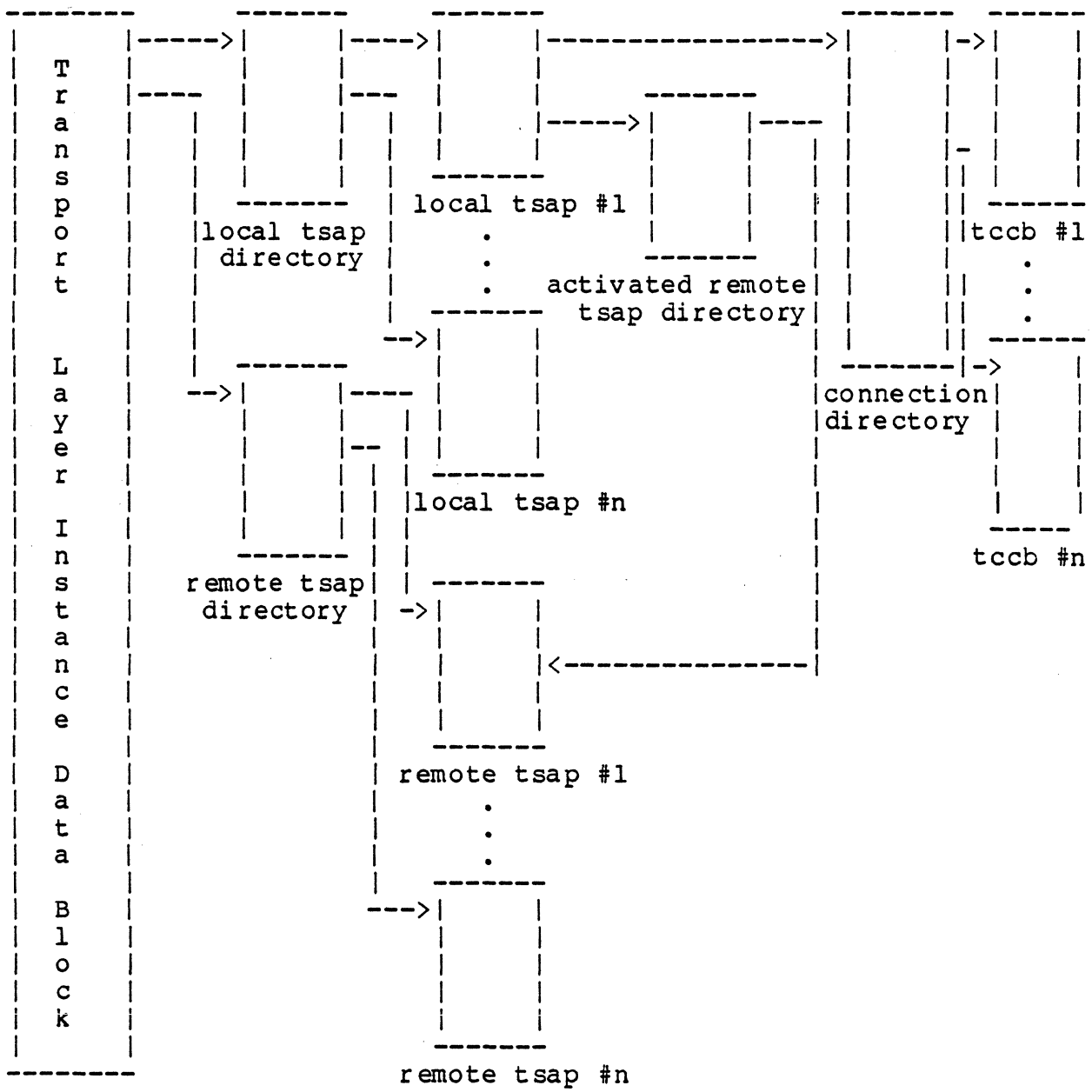

Figure 2

2.1.1 Transport Layer Instance Data Block - TLIDB

This data structure contains some variables that are used in the transport layer operation. Some variables are passed by the System Management during startup time; some are statically created by the Transport Layer.

```
Layer instance data block
BEGIN
        MBID    iold_fc_dir[16]     my iold function mailbox id
        MBID    rcv_mailbox;        my receive process mailbox
        MBID    xmt_maiblox;        my transmit process mailbox
        MBID    dma_mailbox;        IO SW dma mailbox id
        MBID    io_mailbox;         IO SW iold dispatcher mb id
        MBID    sm_event;           System Management event mb id
        MBID    net_sm_id;          network layer management mb id
        MBID    net_xmt_id;         network transmit process mb id
        MBID    net_rcv_id;         network receive process mb id

        unsigned l_tsap_dir_sz;     max size of local tsap directory
        L_TSAP_DIR *l_tsap_dir      local tsap directory pointer

        unsigned r_tsap_dir_sz;     max size of remote tsap
                                    directory
        unsigned max_tran_cxt       max.connections of this layer
        R_TSAP_DIR *r_tsap_dir;     remote tsap directory pointer
        unsigned major_state;       major state
        unsigned substate;          sub state
        long     nsdu_size          network service data unit size
                                    layer instance statistics and
                                    attributes

        ulong
END
```

2.1.2 Local TSAP Table

This table is created and initialized to null upon the entry of the initializtion section of the Transport Layer Management Process. The size of the table is a parameter passed by the System Management when this process is created. A create primitive from the System Management will cause a TSAP entry in the table defined as in the following "C" notation:

```
struct l_tsap
     BEGIN
          short class;           DSA class; not use
          char  tsap_name[16];   16 char symbolic names *
          char  type[4];         type
          short venue;           Venue
          char  majorstate;      Major administrate state
          char  substate;        Sub Administrate state
          struct r_tsap *rtsap   remote tsap pointer
          short net_inst;        network layer instance #
          TSAP  tsap             transport selector 2 bytes
          NSAP  nsap;            newtork address
          char  null;
          char  tsap_class;      always class 4
          short log_addr;        logical local TSAP address
          T_EVENT *tsap_evnt     tsap event indication msg ptr
          ushort max_connect     max.# of connect in this tsap
          ushort max_actv;       number of activiated remote
                                 tsaps in this local tsap
          CONN *connect          connection directory pointer
          C_IND  *c_indicate     connection indicate structure
          unsigned dynamic_config dynamic configuration option
          int   timer_value[5]   transport timers values
                                 local tsap statistics
          ulong  ndtoctsent      octets of normal prio data sent
          ulong  ndtoctrecd      octets of normal prio data recd
          ulong  edtoctsent      octets of expedited data sent
          ulong  edtoctrecd      octets of expedited data recd
          ulong  tpdusent        number of TPDUs sent
          ulong  tpdurecd        number of TPDUs received
          ulong  tpduresent      number of TPDUs resent
          ulong  dtpdresent      number of DATA TPDUs resent
          ushort atpdresent      number of ack tpdus resent
          ushort disconnect      number of disconnect request
          ushort opnconnect      current open connections
          ushort rfconnect1      refused;all connections in use
          ushort rfconnect2      refused;all other reason
          ushort iconnectok      inbound sucessful connections
          ushort iconnectno      inbound unsuceddful connections
          ushort oconnectok      outbound sucessful connections
          ushort oconnectno      outbound unsucessful connections
          ushort conntimout      timeout connections
          ushort creqresent      connect request resent
          ushort erprotocol      transport protocol errors
          ushort erinvtpdus      invalid TPDUs
     END
```

2.1.3 Remote TSAP Table

This table is created and initialized to null upon the entry of the initializtion section of the Transport Layer Management Process. The size of the table is a parameter passed by the System Management when this process is created. A create primitive from the System Management will cause a TSAP entry in the table defined as in the following "C" notation:

```
struct r_tsap
     BEGIN
            short class;          DSA class; not use
            char  tsap_name[16];  16 char symbolic names
            char  type[4];        type
            short venue;          Venue
            char  majorstate;     Major administrate state
            char  substate;       Sub Administrate state
            short net_inst;       network layer instance #
            NSAP  nsap;           newtork address
            TSAP  tsap            transport selector 2 bytes
            char  null;
            char  tsap_class;     always class 4
            short log_addr;       logical local TSAP address
     END
 define TSAP struct tsap
        struct tsap
           BEGIN
            int   len
            unsigned char TSAP[2]
           END
 define NSAP struct nsap
        struct nsap
           BEGIN
             unsigned len
             unsigned char afi
             unsigned char subnet[2]
             unsigned char subnetap[7]
             unsigned char NSAP
             unsigned char filler[21]
           END
```

2.1.4 Activated Remote Tsap directory

This directory contains an array of pointers to remote tsaps which represent the link between this local tsap and its remote peer tsap. When a subsequent operation occurs, such as connect request, this local tsap and the remote tsap will be used as source and destination address respectively. The index to the activated remote tsap directory will be returned to the user who issues this activate remote tsap primitive, as the logical remote tsap address. The depth of this directory is set up according the passed parameter in the create tsap message.

2.1.5 Transport Connection Directory

This directory contains an array of pointers to transport connection control block ,TCCB which is dynamically created or deleted each time when a connect request or disconnect primitive is received. The size of the directory is configurable and is created on a create local tsap primitive. DDI transport has an equivalent connection table but it is for the entire transport layer instance and is unique. The index to the DDI transport connection table is the transport connection identifier that must be kept in the TCCB for subsequent read or write operations use. The index to the connection directory referred to as the connection identifier, and along with the local tsap number must be returned to the user who issues the connect request or accepts the connection indication. Any subsequent operation primitive such as read CO or write CO data must have these two items along to be used to identify the DDI transport connection identifier. DDI connection table is defined as:

```
struct TCXT *trans_ctx[MAXTRAN]
```

2.1.6 Transport Connection Control Block

This TCCB is dynamically created and deleted by Receive process when it receives a connection request or a disconnect request primitive locally or remotely. This block contains vital information for subsequent connection operation. This control block is considered an extension of the connection block TCTX in the DDI transport layer.

```
struct ctx
   BEGIN
        unsigned l_local_ta        logical local tsap address
        unsigned l_remote_ta       logical remote tsap address
        long     connection_id     connection directory id
        long     transport_id      connection id from DDI
        struct   TSAP  l_tsap      local tsap
        struct   TSAP  r_tsap      remote tsap
        struct   NSAP  l_nsap      local network
        struct   NSAP  r_nsap      remote network
        XPT_TRANS *connect_rqt     connect request transaction ptr
        XPT_TRANS *connect_ind     connection indication trans ptr
        WR_BUFD  *bufdes           L6 write buffer descriptor
        RD_BUFD  *rbufdes          L6 read buffer descriptor
        WR_EBUFD *ewbufdes         L6 write expedited buffer des
        RD_EBUFD *erbufdes         L6 read expedited buffer des
        unsigned tpdusize          tpdu size
        unsigned init_wrcdt        initial write credits
        unsigned real_wrcdt        actual write credit used
        unsigned writecount        write CO data in hand
        unsigned init_rdcdt        initial read credits
        unsigned real_cdt          actual read credit used
        unsigned readcount         read CO data in hand
        unsigned write_exp         # of write expedited data
        unsigned read_exp          # of read expedited data
        unsigned t_flow:1          DDI flow control flag
        unsigned discn_pend:1      disconnection waiting flag
        unsigned reason            disconnect reason code
        struct XMT_HDR *send_head  first to send
        struct XMT_HDR *send_tail  next to send
   END

        struct XMT_HDR
          BEGIN
              unsigned eot
              BD       *buffer
          END
```

DDI has the following transport connection control block defined. It shows here as for reference.

```
struct tctx
       BEGIN
            unsigned  state
            unsigned  class4:1
            unsigned  expedited:1
            unsigned  chksum:1
            unsigned  extended:1
            unsigned  flow:1
            unsigned  session_flow:1
            unsigned  flow_controlled:1
            unsigned  prio:3
            unsigned  dst_ref
            unsigned  src_ref
            long      pid
            struct tsap rsuffix
            struct tsap ssuffix
            long      session_id
            long      recv_lwe
            long      recv_uwe
            long      recv_next
            long      recv_expd
            struct frag_hdr *recv_head
            struct frag_hdr *recv_ehead
            long      send_lwe
            long      send_uwe
            long      send_next
            long      send_expd
            short     send_subseq
            struct frag_hdr *send_head
            struct frag_hdr *send_tail
            struct frag_hdr *send_notsent
            struct frag_hdr *send_ehead
            struct frag_hdr *send_etail
            struct frag_hdr *send_enotsent
            unsigned   send_retry_count
            unsigned   tpdusz
            unsigned   tpdusize_parm:8
            unsigned   vers:8
            long       maxseq
            unsigned   reason
            unsigned   cdt
            struct     ctb timer[5]
            struct     nsap source_address
            struct     nsap destination_address
            unsigned   tsap_id           see note
            unsigned   connection_id     see note
       END

       note: these are added to speed up the searching
             local tsap # and its local connection id
```

2.2 EXTERNAL INTERFACES

The Transport interfaces with the System Management, ISO network services and Megabus interface software in the LACS and the ISO user in Level 6. All users and services provider interfaces to this Transport are via the Bridge Communication Inc. Kernel messages call. The following describes the interfaces and the message format will be shown at the later sections.

2.2.1 System Mangement interfaces

Startup Parameters Data structure

These paramaters, passed by the System Management to Tranpsort Layer Management in the process creating phase, are used by the Transport Layer Management Process for it initial setup.

```
Startup parameter
BEGIN
        char     layer             layer number
        char     layer_inst_num    layer instance number
        char     my_priority       this process priority
        unsigned max_local_sdir    size of local tsap directory
        unsigned max_remote_sdir   size of remote tsap directory
        unsigned max_connect       size of transport connections
END
```

The Transport Layer Management provides the following services to the Systeem Management in the LACS:

Action:
- create tsap
  - create local tsap
  - create remote tsap
- update state
- list all

Get Request
supporting read statistics and get attributes only

The Transport layer provides the following indication to the System Management in the LACS.

Event_indication

2.2.2 Users (Session layer) Inferfaces

The Transport provides the following services to the user via LCB through Lacs Driver interface:

Connect Request
Connect Response
Write CO data(Data Request)
Write Expedited Data
Read CO data
Read Expedited Data
Disconnect Request
Activate Local TSAP
Activate Remote TSAP
Deactivate Local TSAP
Deactivate Remote TSAP

The Transport provides the following indication to the user

TSAP Event Indication
- TSAP deactivated
- Connection Indication

Connection Event Indication
- data arrivals
- additional data write credit available
- disconnect request

2.2.3 Network Layer Inferfaces

The Transport uses the following service of the Null Network Layer:

N_Data Request

The Transport expects the following indication from the Null Network Layer

N_Data Indicate

2.2.4 Megabus Interface Software Interfaces

IOLD registration for LCB image copy

Data Transfer between L6 and Lacs buffer requests

2.2.5 Messages Format

The communication among processes and between layer is typically through Kernel sendmsg call. All interfaces to and from Transport are via messages to mailboxes. The following describes each message format that formulate the above mentioned requests and indications.

2.2.5.1 Messages from System Management

2.2.5.1.1 Action

```
struct action
  BEGIN
    MSG     m;                  Kernel message header
    MBID    ret_mbid;           return message mailbox id
    short   sm_id;              SM identifier
    short   exchangeid;         exchange identification
                                internal layer selector
    char    name[16];           symbolic tsap name
    short   class;              transport class 4
    short   type[4];            type
    short   venue;              who knows
    char    majorstate;         set to locked state
    char    substate;           set to reset state
    short   access_control;     ignored by Transport
                                status return by Transport
    char    source;             layer number = 4
    char    statusid;           status id
    short   statuslngth;        status length = 2
    short   statusdata;         not used
                                SM request operation code
    short   operation_code;     set to action
    short   operation_info;     start of operation info
                                action request
    short   operation;          action opcode
    short   length;             set to tsap size
  END
```

2.2.5.1.2 Create local tsap

```
struct creat_l_tsap
  BEGIN
    MSG     m                   Kernel message header
    MBID    ret_mbid            return message mailbox id
    short   sm_id               SM identifier
    short   exchangeid          exchange identification
                                layer internal selector
    char    name[16]            ignored by Transport
    char    class               ignored by Transport
    char    type[4]             ignored by Transport
    char    venue               ignored by Transport
    char    majorstate          ignored by Transport
    char    substate            ignored by Transport
    short   access_control      ignored by Transport
                                status return by Transport
    char    source              layer number = 4
    char    statusid            status id
    short   statuslngth         status length = 2
    short   statusdata          not used
                                SM request operation code
    short   operation_code      equal action
    short   operation_info      start of operation info
                                action create request
    short   operation           creat local tsap opcode
    short   length              tsap size
                                this tsap parameters
    char    name[16]            symbolic tsap name
    short   class;              transport class 4
    short   type[4];            type
    short   venue;              who knows
    char    majorstate;         set to locked state
    char    substate;           set to reset state
                                flow control info
    char    addr[2];            tsap address
    char    rfu;                not used
    char    net_li_mapping;     network layer instance
    short   max_xmit_bytes;
    short   max_rcv_bytes;
    short   max_xmit_credit;
    short   max_rcv_credit;
    short   max_activate;       max. activation remote tsap on
                                this local tsap
    short   current_act;        current number of activation
    ACT_REMOTE *act_remote;     point to activated remote tsap
                                table
    TSAP_EVENT *tsap_event;     tsap event lcb pointer
  END
```

2.2.5.1.3 Create remote tsap

```
struct creat_r_tsap
  BEGIN
    MSG     m                   Kernel message header
    MBID    ret_mbid            return message mailbox id
    short   sm_id               SM identifier
    short   exchangeid          exchange identification
                                layer internal selector
    char    name[16]            ignored by Transport
    char    class               ignored by Transport
    char    type[4]             ignored by Transport
    char    venue               ignored by Transport
    char    majorstate          ignored by Transport
    char    substate            ignored by Transport
    short   access_control      ignored by Transport
                                status return by Transport
    char    source              layer number = 4
    char    statusid            status id
    short   statuslngth         status length = 2
    short   statusdata          not used
                                SM request operation code
    short   operation_code      equal action
    short   operation_info      start of operation info
                                action create request
    short   operation           creat remote tsap opcode
    short   length              tsap size
                                this tsap parameters
    char    name[16]            symbolic tsap name
    short   class               transport class 4
    short   type[4]             type
    short   venue               who knows
    char    majorstate          set to locked state
    char    substate            set to reset state
                                flow control info
    char    addr[2]             tsap address
    char    rfu                 not used
    char    net_li_mapping      network layer instance
    short   max_xmit_bytes
    short   max_rcv_bytes
    short   max_xmit_credit
    short   max_rcv_credit
  END
```

2.2.5.1.4 Update_state

```
struct update_state
  BEGIN
    MSG     m                   Kernel message header
    MBID    ret_mbid            return message mailbox id
    short   sm_id               SM identifier
    short   exchangeid          exchange identification
                                internal layer selector
    char    name[16]            symbolic tsap name
    short   class               transport class 4
    short   type[4]             type
    short   venue               who knows
    char    majorstate          set to locked state
    char    substate            set to reset state
    short   access_control      ignored by Transport
                                status return by Transport
    char    source              layer number = 4
    char    statusid            status id
    short   statuslngth         status length = 2
    short   statusdata          not used
                                SM request operation code
    short   operation_code      set to udpate_state
    short   operation_info      start of operation info
                                action create request
    short   operation           creat remote tsap opcode
    short   length              tsap size
  END
```

2.2.5.1.5 list_all

```
struct list_all
  BEGIN
    MSG     m;                  Kernel message header
    MBID    ret_mbid;           return message mailbox id
    short   sm_id;              SM identifier
    short   exchangeid;         exchange identification
                                internal layer selector
    char    name[16];           symbolic tsap name
    short   class;              transport class 4
    short   type[4];            type
    short   venue;              who knows
    char    majorstate;         set to locked state
    char    substate;           set to reset state
    short   access_control;     ignored by Transport
                                status return by Transport
    char    source;             layer number = 4
    char    statusid;           status id
    short   statuslngth;        status length = 2
    short   statusdata;         not used
                                SM request operation code
    short   operation_code;     set to list all
    short   operation_info;     start of operation info
                                list all request
    short   operation;          list all opcode
    short   length;             set to tsap size
  END
```

2.2.5.1.6.Get Request

```
struct get request
  BEGIN
    MSG     m                   Kernel message header
    MBID    ret_mbid            return message mailbox id
    short   sm_id               SM identifier
    short   exchangeid          exchange identification
                                internal layer selector
    char    name[16]            symbolic tsap name
    short   class               transport class 4
    short   type[4]             type
    short   venue               who knows
    char    majorstate          set to locked state
    char    substate            set to reset state
    short   access_control      ignored by Transport
                                status return by Transport
    char    source              layer number = 4
    char    statusid            status id
    short   statuslngth         status length = 2
    short   statusdata          not used
                                SM request operation code
    short   operation_code      set to get
    short   operation_info      start of operation info
                                action create request
    short   operation           creat remote tsap opcode
  END
```

2.2.5.1.7 System Management Event

```
struct event_info
  BEGIN
   char   source;          transport layer magm message
   char   code;            event status code
   short  info_length;     length of status which follow
   char   info[0x400];     information data
  END
```

2.2.6 LCB Data Strutures used between User and Transport

The Session and other users in the L6 communicate with the Transport via LACS Driver which issues IOLDs point to a LCB in L6 main memory to pass information to/from Transport The following describes the various LCBs.

2.2.6.1 Activate Local TSAP

```
struct L_activate
  BEGIN
    short  cb_icw;          interrupt control word
    short  cb_fnc;          functio = activate local tsap
    short  cb_ind;          buffer indicator
    long   cb_rng;          total range in bytes
    short  cb_bct;          number of buffers
    L6BD   16_bd[3];        16 buffer descriptors

    char   cb_sym[16];      symbolic name
    long   cb_lsa;          logical address - tsap selector
    long   cb_pms;          proposed read SDU size
    short  cb_prc;          proposed read max. credits


    short  fns[12];         null fields

    long   cb_mss;          maximum SDU size
    long   cb_iss;          ideal SDU size
    short  cb_mpr;          maximum pending read count
    short  cb_wcc;          write credit count
    short  cb_mcc;          maximum number of connections

    short  cb_cts;          controller status
    short  cb_sts;          this command status
    short  cb_cbs;          completion word
  END
```

2.2.6.2 Activate remote TSAP

```
struct r_activate
  BEGIN
     short   cb_icw;          interrupt control word
     short   cb_fnc;          functio = activate local tsap
     short   cb_ind;          buffer indicator
     long    cb_rng;          total range in bytes
     short   cb_bct;          number of buffers
     L6BD    l6_bd[3];        16 buffer descriptors

     char    cb_sym[16];      symbolic name
     long    cb_lsa;          logical address - tsap selector
     long    null;            not used

     short   fns[18];         null fields

     long    cb_rla;          remote logical address

     short   cb_cts;          controller status
     short   cb_sts;          this command status
     short   cb_cbs;          completion word
   END
```

2.2.6.3 Deactivate remote TSAP

```
struct r_dactivate
  BEGIN
     short   cb_icw;          interrupt control word
     short   cb_fnc;          functio = activate local tsap
     short   cb_ind;          buffer indicator
     long    cb_rng;          total range in bytes
     short   cb_bct;          number of buffers
     L6BD    l6_bd[3];        16 buffer descriptors

     short   fns[30];         null fields

     long    cb_lla;          logical local tsap address
     long    cb_rla;          logical remote tsap address

     short   cb_cts;          controller status
     short   cb_sts;          this command status
     short   cb_cbs;          completion word
   END
```

2.2.6.4 Deactivate local TSAP

```
struct l_dactivate
  BEGIN
      short   cb_icw;          interrupt control word
      short   cb_fnc;          functio = activate local tsap
      short   cb_ind;          buffer indicator
      long    cb_rng;          total range in bytes
      short   cb_bct;          number of buffers
      L6BD    l6_bd[3];        16 buffer descriptors

      short   fns[30];         null fields

      long    cb_lla;          logical local tsap address

      short   cb_cts;          controller status
      short   cb_sts;          this command status
      short   cb_cbs;          completion word
  END
```

2.2.6.5 Connect Request

```
struct connect Request
  BEGIN
   unsigned  cb_icw            interrupt control word
   unsigned  cb_fsf            connection request function

                               next field sets by user or xport
   unsigned  cb_ind            user data present indication
   long      size              size of user data in this lcb
   short     null              not used
   char      data[32]          user data field

   long      lr_tsap           logical remote tsap address
   long      ll_tsap           logical local tsap address
   unsigned  qos               quality of services
   unsigned  expedited         expedited data option
   unsigned  p_sdu_size        proposed Max SDU size
   unsigned  p_rd_credit       proposed read credits

                               return parameters to user
   long     connection_id      connection identifier
   long     rspnd_add          responding address ?
   unsigned expedited          expedited data option
   unsigned qos                quality of services
   unsigned sdu_size           max. SDU size
   unsigned ideal_size         ideal max. sdu size
   unsigned read_credit        read order credits
   unsigned wr_credit          write credits

                               return status
   short    cb_cts             controller status
   short    cb_sts             this command status
   short    cb_cbs             completion word
  END
```

2.2.6.6 Connect Response

```
struct connect_rsponse
  BEGIN
  unsigned  cb_icw;          interrupt control word
  unsigned  cb_fnc;          connection response function
  unsigned  cb_ind;          user data present indicator
  long      size;            size of user data in this lcb
  short     null;            not used
  char      data[32];        user data field

  long      lr_tsap;         logical remote tsap address
  long      l_tsap;          logical local tsap address
  unsigned  qos;             quality of services
  unsigned  expedited;       expedited data option
  unsigned  p_sdu_size;      proposed Max SDU size
  unsigned  p_rd_credit;     proposed read credits

                             return parameters to user
  long     connection_id;    connection identifier
  long     rspnd_add;        responding address ?
  unsigned expedited;        expedited data option
  unsigned qos;              quality of services
  unsigned sdu_size;         max. SDU size
  unsigned ideal_size;       ideal max. sdu size
  unsigned read_credit;      read order credits
  unsigned wr_credit;        write credits

                             return status
  short    cb_cts;           controller status
  short    cb_sts;           this command status
  short    cb_cbs;           completion word
  END
```

2.2.6.7 Disconnect Request

```
struct disconnect request
 BEGIN
  unsigned  cb_icw;            interrupt control word
  unsigned  cb_fnc;            connection request function

                               next field sets by user
  unsigned  cb_ind;            user data present indication
  long      size;              size of user data in this lcb
  short     null;              not used
  char      data[64];          user data field

  long     connection_id;      connection identifier
  unsigned reason;             disconnect reason code

                               return status
  short    cb_cts;             controller status
  short    cb_sts;             this command status
  short    cb_cbs;             completion word
 END
```

2.2.6.8 TSAP Event Lcb

```
struct Tsap_event
  BEGIN
  unsigned  cb_icw;            interrupt control word
  unsigned  cb_fnc;            TSAP event function

                               next field sets by transport
  long      ll_addr;           logical local sap address
  long      data_size;         data size in bytes
  unsigned evnt_mask;          event mask code

                               return status
  short    cb_cts;             controller status
  short    cb_sts;             this command status
  short    cb_cbs;             completion word
  END
```

2.2.6.9 Connect Indication TSAP Event Lcb

```
struct Connection_indication
  BEGIN
  unsigned  cb_icw;            interrupt control word
  unsigned  cb_fnc;            connection indication function
  unsigned  cb_ind;            remote user data indicator
  long      size;              size of user data in this lcb
  short     null;              not used
  char      data[32];          user data field

                               return parameters to user
  long     connection_id;      connection identifier
  unsigned expedited;          expedited data option
  unsigned qos;                quality of services
  unsigned sdu_size;           max. SDU size
  unsigned ideal_size;         ideal max. sdu size
  unsigned read_credit;        read order credits
  unsigned wr_credit;          write credits -
  long     ll_addr;            logical local tsap address
  unsigned evnt_mask;          connection indicate mask code

                               return status
  short    cb_cts;             controller status
  short    cb_sts;             this command status
  short    cb_cbs;             completion word
 END
```

2.2.6.10 Write Connection Oriented LCB

```
struct write_CO_data
  BEGIN
     unsigned  cb_icw;         interrupt control word
     unsigned  cb_fnc;         function = write data
     unsigned  cb_ind;         buffer indicator
     long      cb_rng;         total range in bytes
     unsigned  cb_bct;         number of buffers
     L6BD      16_bd[3];       16 buffer descriptors

     long      connect_id;     connection identifier
     unsigned  wr_credit;      not used

     short  fns[xx];           null fields


     short  cb_cts;            controller status
     short  cb_sts;            this command status
     short  cb_cbs;            completion word
  END
```

2.2.6.11 Write Expedited LCB

```
struct write_Edata
   BEGIN
      unsigned  cb_icw;          interrupt control word
      unsigned  cb_fnc;          function = write data
      unsigned  cb_ind;          buffer indicator
      long      cb_rng;          total range in bytes
      unsigned  cb_bct;          number of buffers
      L6BD      l6_bd[3];        16 buffer descriptors

      long      connect_id;      connection identifier
      unsigned  wr_credit;       not used

      short  fns[xx];            null fields


      short  cb_cts;             controller status
      short  cb_sts;             this command status
      short  cb_cbs;             completion word
   END
```

2.2.6.12 Read Connection Oriented LCB

```
struct read_CO_data
  BEGIN
    unsigned  cb_icw;          interrupt control word
    unsigned  cb_fnc;          function = write data
    unsigned  cb_ind;          buffer indicator
    long      cb_rng;          total range in bytes
    unsigned  cb_bct;          number of buffers
    L6BD      l6_bd[3];        16 buffer descriptors

    long      residue[3];      buffer residue ranges


    long      connect_id;      connection identifier
    unsigned  rd_credit;       read credits

    short  fns[xx];            null fields


    short  cb_cts;             controller status
    short  cb_sts;             this command status
    short  cb_cbs;             completion word
  END
```

2.2.6.13 Read Expedited LCB

```
struct read_edata
  BEGIN
    unsigned  cb_icw;          interrupt control word
    unsigned  cb_fnc;          function = write data
    unsigned  cb_ind;          buffer indicator
    long      cb_rng;          total range in bytes
    unsigned  cb_bct;          number of buffers
    L6BD      l6_bd[1];        16 buffer descriptors

    long      residue[1];      buffer residue ranges


    long      connect_id;      connection identifier
    unsigned  rd_exp_cr;       read expedited credits
    unsigned  act_size;        buffer actual size

    short  fns[xx];            null fields

    short  cb_cts;             controller status
    short  cb_sts;             this command status
    short  cb_cbs;             completion word
  END
```

2.2.7 Messages used between Transport and Megabus Interface Software

2.2.7.1 Mailbox Registration for IOLDs

```
struct mbid_ptr
  BEGIN
      MSG     m;                  kernel message header
      ushort  chan_nmb;           channel number
      MBID    *mbid;              mailbox pointer for this chann
      MBID    return_id;          return mailbox id
      short   status;             return from IO dispatcher
  END
```

2.2.7.2 IOLD indication

```
struct IOLDMSG
  BEGIN
      MSG     m;                  kernel message header
      ushort  chanfc;             channel number and opcode
      long    *l6_addr;           lcb address in bytes
      ushort  lcb_info;           lcb size in byte
  END
```

2.2.7.3 LCB to/from L6 memory

```
struct lcbio
  BEGIN
      MSGX    mx;                 expanded kernel header
      long    *l6_addr;           L6 address
      ushort  range;              lac ram range
      long    *ram_addr;          lac ram address
  END
```

2.2.7.4 Data Request to/from L6 memory

```
struct bufio
  BEGIN

     MSGX    mx;              expanded kernel message header
     ushort  16_buf_cnt;      buffer counts in this message
     L6_DES  16[1-9];         16 buffer descriptors
  END


struct bufiox
   BEGIN
     MSGX    mx;              expanded kernel message header
     L6_LIST *16ptr;          16 buffer descriptors pointer
   END


struct buflcbio
   BEGIN
     MSGX    mx;              expanded kernel message header
     L6_LIST *16ptr;          16 buffer descriptors pointer
     long    *16_addr;        16 address in bytes
     ushort  range;           lac ram range in bytes
     long    *ram_addr;       lac ram address
   END
```

2.2.7.5 Transport transaction structure

```
struct xpt_trans
      BEGIN
           union
                BEGIN
                     LCBIO  lcbio  DMA requests for LCBIs
                     BUFIO  bufio  DMA requests for buffer data
                     BUFIOX bufiox DMA requests for buffer data
                                   with a list
                END type
           BD      *data_bd       data buffer descriptor
           ushort  lcbi_leng      the length of LCBI
           ushort  lcb_chan       the channel involved
           LCBI    *lcbi_blk      pointer to LCBI block
           caddr_t L6_mem_ptr     L6 memory pointer to LCB
           BDI     *rd_bdi_blk    ptr to read BD block
           BDI     *wr_bdi_blk    ptr to write BD blockk
           caddr_t L6_rdbdi_ptr   L6 memory ptr to read BD
           caddr_t L6_wrbdi_ptr   L6 memory ptr to write BD
           L_TSAP  *l_tsap_table  ptr to local tsap table
           R_TSAP  *r_tsap_table  ptr to remote tsap table
           ulong   l_log_addr     local logical address
           ulong   r_log_addr     remote logical address
           ulong   connection_id  logical connection id
           ushort  transaction_id type of transaction
      END
```

2.2.8 Messages used between Transport and Network

2.2.8.1 N_Data_indication

This message is used by the network layer when it indicates the arrival of a NSDU which may contain one or several TPDU.

```
struct n_data_indicate
  BEGIN
      MSG     m                    kernel message header
      int     function             not used
      int     qos                  not used
      struct NSAP *source          NSAP source address
      struct NSAP *destination     NSAP destination address
      int     datasize             data size
      char    *data                data pointer
  END
```

2.2.8.2 N_Data_Request

This message is used by the Transport to request the Network layer to transmit a NSDU.

```
struct n_data_request
  BEGIN
      MSG     m                    kernel message header
      int     function             not used
      int     qos                  not used
      struct NSAP *source          source address
      struct NSAP *destination     destination address
      int     datasize             data size
      char    *data                data pointer
  END
```

2.3. Initialization Requirements

The initialization of the Transport is part of the entire LAN software initialization sequences. The System Management is responsible to spawn the Transport Layer Management process which then allocates memory for common data structure used by all three processes; create mailboxes for interprocess communication;create Transport Transmit and Recieve Processes. Then the Transport Layer Management waits for creat TSAP messages to create local and remote tsap table directories. The state of the TSAP is set to inactive until activate message is received from user. Initialization is done once only at the startup time.

2.4 TERMINATION REQURIREMENTS

The Transport Layer will be active as long as the LAN software is active. No termination is required.

2.5 ENVIRONMENT

The Transport is operating under the Bridge Communication Inc. kernel environment It must be part of the LAN software bound unit that resides in the Lacs hardware subsystem.

2.6 TIMING AND SIZE REQUIREMENTS

Sizes and memory usage are not an issue at this point. However, the code must efficient enough to produce high performance product.

2.7 ASSEMBLY AND LINKING

The Transprot module will be written in C language for 68000 machine code. Assembly and linking is accomplished through the makefile in the Honeywell Unix Operation Development System.

2.8 TESTING CONSIDERATION

All functions must be tested throughoutly. Testing with the NBS scenerio is a must. A test routine may be considered to replace the Network Layer initially for initial checkout with NBS testing before integrating with rest of the software modules. This test routine will be a turnaround routine that it behaves as if it were the remote peer entity.

2.9 DOCUMENTATION CONSIDERATIONS

Documentation of this product should follow the Honeywell Software Documentations Guidelines. A procedure design language should be accompanied in this component specification

2.10 OPERATING PROCUDURES

None is required to operate this module.

2.11 ERROR MESSAGES

There are several types of error messages this Transport can handle. They are described as follows:

Non_fatal operation error
This kind of errors are usually detected on the interfaces messages. The message will be returned with appropriate status to inform the message sender about the conditon.

Fatal operational error
TBD

Transport Protocol error

These errors are associated with the transport protocol machine and are handled according to protocol specification Its statistics counters may be read via the System Mangement interface.

3. INTERNAL SPECIFICATION

3.1 Overview

The Transport Layer is an implementation of the ISO transport layer class 4, connection oriented protocol. This module communicates with user in L6 (session layer),the network layer module and the system management module. The heart of this module is the Transport machine which is adapted from DDI GM_MAP transport layer. Three separate modules and many routines are added to interface to DDI transport machine to provide the necessary interface conversion to use the DDI transport module Modification are kept in minimal to speedup the development efforts. The primarily changes to the DDI transport is the buffer memory management which is essential to run under the Lacs environment.The sections below will describe the external requirement that requires to interface the DDI transport layer. No attempts to describe the DDI transport module is done at this time. DDI does not have any documentations at all.

3.1.1 DDI Interfaces and Data Structure Requirement

The following describes the interfaces and parameters requirement of the DDI transport function. The two structures shown below are are used to pass information between the service user and service provider. Note that certain parameters are not used as it depends on the function.

The DDI provides the following services to user:

T_Initialize Request
T_Connect Request
T_Connect Response
T_Data Request
T_Expedited Data Request
T_Disconnect Request
T_Statistic Request

The DDI provides the following indication to user:

T_Disconnect
T_Connection Response
T_Data
T_Expedited Data
T_Connect Request
T_Flow
T_Stop_Flow

The following data structure are used to call for service request and provider indication

```
struct fpt_tran
  BEGIN
       unsigned  function          connect request,T_data,etc
       unsigned  qos               quality of Service
       unsigned  expedited:1       expediated data option
       unsigned  chksum:1          checksum option
       unsigned  eot:1             end of frame
       unsigned  reason            reason code
       unsigned  transport_id      identifier for DDI tctx
       unsigned  tsap_id           local tsap id see note 1
       unsigned  connection_id     local connection id see note 1
       long      session_id        session identifier
       struct    address source    source address
       struct    address destn     destination address
       int       datasize          data size
       char      *data             data address
  END
       note 1: These are added to speed up searching tsap and
               its connection id.
```

The network provides a Network Data Request service and N_data Indicate to DDI transport

Data structure used to passed information for service request and indication:

```
struct fpt_netw
  BEGIN
       int       function          data_indicate,data request
       int       qos
       struct    NSAP *source
       struct    NSAP *destination
       int       datasize
       char      *data
  END
```

3.2 Subcomponent Description

3.2.1 Transport Layer Management Process

This process provides the function of task lead of the Transport layer. Its responsibility is to initialize and set certain data structure which will be used by all processes. It creates both local and remote TSAP directories tables and sets up necessary functions before ready to accept messages from System and others. The Layer Management Process primary consists of an initalization routine, a main routine which responds to messages and some support routines. The initialization routine allocates memory for data structure that all processes will be operating on, sets various tables, and finally spawns the Transport Process before ready for messages.
The main routine responds, decodes,and executes messages delivered to this mailbox. The only requests from the user to this module is the activate local/remote tsap and delete tsap requests. Any other requests from the user will be returned with appropriated status. The reception of the activate call causes the Transport majorstate into in_use_state and therefore Transport process is ready to receive requests from the users.

3.2.2 Transport Process

This transport function mainly proceses messages from its users and network layer for incoming data. It decodes and validates user primitive LCBs and converts them into DDI inteface data structure before calling DDI transport function. In case of network data indicate this transport builds DDI required data structure before calling DDI to handle this connection.

It manages locally or remotely initiated connection establishment and connection release primitives. This may involve memory allocation and deallocation for connection table and transport connection control block; transferring user data on connection request; informing the user of the disconnect request.

The transmit function manages the data transfer across the level 6 and the DDI. In case of large TSDU the function will segment data into multiple TPDUs. Releasing the LCB and its 16 buffer descriptor is done at the completion of data transfer.

The receive function of this process is to receive data from its peer entity and transfer the data to user buffer in 16 memory.

3.3 Future Development and Maintenance Considerations

TBD

4.0 PROCEDURE DESIGN LANGUAGE (PDL)

4.1 Transport Layer Management Process

4.1.1 Transport Layer Management Initialization routine.

```
Tran_lm_init(startup_parm)
STARTUP_PARM *startup_parm

BEGIN
     Allocate memory for layer common data structure table
     Initialize all tables entries to a known state.
     Save this pointer to this process's PCB

     Allocate memory for local TSAP directory table.
     Initialize the local tsap directory entries to null.

     Allocate memory for remote TSAP directory table.
     Initialize the remote tsap directory entries to null.

     Allocate memory for event indication to System Management

     Create a second mailbox id for Transport LME

     Registrate the well known mailbox for transport process

     Create transport process.
     Move the transport process to ready list.

     Resolve IO software IO and DMA mailbox id.
     Call request_io_mb(return_id,common_ptr);

     set transport major state to none existence
     set transport substate to reset


     wait for messages to arrive to this mailbox.
END
```

4.1.1.1 Request IO Software mailboxes Id function

```
Req_io_mb(parameters)
BEGIN
     Allocate memory for request message
     Setup message parameters
     call sendmsg kernel call
END
```

4.1.2 Transport Layer Mangement Main Function

```
Tran_main(msgptr,mboxid)
MSG *msgptr
MBID mboxid

BEGIN
     retrieve TLIDB pointer

     switch(message type)
         case iold from IO software
              call common iold handler function
              break
         case lcb arrival
              call lcb handling function
              break
         case lcb to level 6 confirmation
              call lcb cleanup function
              break
         case system management message arrival
              call system management message function
              break
         case IO software delivers IO and DMA mailbox id
              call IO software mailbox id arrival function
              break
         case IOLD FC mailbox directory confirmation message
              call iold sign in return function
              break
         case network sign in confirmation message
              call network sign in return function
              break
         default
              return message memory to memory pool
              break
    switchend
END
```

4.1.2.1 System Management message processor

This routine determines the request validity. The message is forwarded to the function which then executes the request.

```
Sm_request(parameters)
BEGIN
     switch on system message operation code
          case Get Request
               call get function
               break
          case Set Request
               send message to SM with INVALID status
               break
          case Compare and Set Request
               send message back to SM with INVALID status
               break
          case Action
               call action function
               break
          default
               return message to memory pool
               break
      switchend
END
```

4.1.2.1.1 Get function

```
Get(parametes)
BEGIN
   TBD
END
```

4.1.2.1 4 Action function

```
Action(parameters)
BEGIN
     switch on action_identifier
         case list all
              call list all function
              break
         case update state
              call udpate state function
         case create tsap
              if create local tsap
                 call create l_tsap function
              else call create remote tsap function
              break
         default
              return(status)
              break
      switchend
END
```

4.1.2.1.4.1 Create Local Tsap Function

```
Create_l_tsap(parameters)
BEGIN
     set major state to locked state

     if local directory has no room
        set status and return
     if local directory is not empty
     call comparing symbolic names function
     if match
        set duplicated status and return
     allocate memory for remote tsap
     add entry into the directory table
     increment next directory entry pointer
     initialze the tsap
     copy all parameters from messges to this tsap
     reset all statistical counters
END
```

4.1.2.1.4.2 Create remote Tsap function

```
Create_r_tsap(parameters)
BEGIN
    set majorstate to locked state

    if remote directory has no room
      set status and return
    if remote directory is not empty
      call comparing symbolic names function
      if match
         set duplicated status and return
    allocate memory for tsap
    add entry to remote directory table
    increment next directory entry pointer
    initialze the tsap
    copy all parameters from messges to this tsap
    reset all statistical counters

END
```

3.1.2.1.4.3 List all function

```
list_all(parameters)
BEGIN
    TBD
END
```

4.1.2.1.4.4 Update state function

```
update(parameters)

BEGIN
     TBD
END
```

4.1.2.2 LCB to level 6 Confirmation

This message returned by Megabus Interface Software indicates that the lcb has been returned to level 6.

```
lcb_cleanup_function(parameter)
BEGIN
     if status not ok
        call statistics update function
        call send event to System Management
     return message meory to memory pool
END
```

4.1.2.3 IO Software mailbox id arrival Function

This message sent by IO megabus software responding to mailbox identifiers request message that initiated by the initialization section of this module.

```
IO_id_arrival(parameters)
BEGIN
    Store the two mailbox ids in the common data structure
    Release this message to memory pool
    Register with IO Dispatcher the IOLD mailbox directory
    to deliver activate/deactivate TSAP iolds
END
```

4.1.2.4 Lcb arrival handler

This message returned by the Megabus Interface Software DMA module indicating transfer lcb from level 6 has been completed

```
lcb_handler(parameters)
BEGIN
    Combine channel number,cpu number and interrupt level
    and place this into message

    if return status not ok
       call return_lcb_to_16 and return
    switch on lcb specific function code
          case activate local tsap
               call activate local tsap function
               break
          case activate remote tsap
               call activate remote tsap function
               break
          case deactivate local tsap
               call deactivate local tsap
               break
          case deactivate remote tsap
               call deactivate remote tsap
               break
          default
               set invalid function status and return lcb to 16
               break
    switchend
END
```

```
4.1.2.4.1 Activate local tsap
          activate_local(parameters)
          BEGIN
              switch on layer majorstate
                    case locked state
                         Call search routine if local tsap exists
                         if local tsap not found
                            set tsap not found status
                            return lcb to l6 and exit
                         Turn off IOLD mailbox
                         Put this message back to mailbox used later
                         Create a maibox
                         Allocate memory activate network message
                         Set registration message type
                         Set all other message parameters
                         Send message
                         break
                    case in_use_state
                         Call search routine if local tsap exists
                         if local tsap not found
                            set tsap not found status
                            return lcb to l6 and exit
                         switch on this tsap majorstate
                                case null
                                     set tsap majorstate to in_use
                                     set tsap substate to operational
                                     get output parameters into lcb
                                     set successful status
                                     send lcb to l6 message
                                     break
                                case in use state
                                     get output parameters into lcb
                                     set sap_already activated status
                                     send lcb to l6 message
                                     break
                                case down
                                case test
                                default
                                     set sap not available status
                                     send lcb to l6 message
                                     break
                         switchend
                     default
                          set bad local tsap status
                          send lcb to l6 message
                          break
               switchend
          END
```

4.1.2.4.2 Activate remote tsap

```
activate_remote(parameters)
BEGIN
    switch on layer majorstate
           case in_use_state
               Call search routine if remote tsap exists
               if remote tsap not found
                  set tsap not found status
                  return lcb to 16 and exit
               Search local tsap existence with this logical
                 local address input parameter
               if local tsap non existence or not operational
                  set bad logical local tsap address status
                  return lcb to 16
               if current activate count > max. allowed
                  set exceeded limit status
                  return lcb to 16
               if duplicated entry in remote activated table
                  set duplicated activated status
                  return lcb to 16
               put entry into activated remote table
               increment current activated count
               put logical remote tsap address into message
               set successful status
               return lcb to 16
               break
           case down
           case locked
           default
                set bad local tsap status
                send lcb to 16 message
                break
        switchend
END
```

4.1.2.4.3 Deactivate local tsap

```
Deactivate_local
BEGIN
     TBD
END
```

4.1.2.4.4 Deactivate remote tsap

```
Deactivate_remote

BEGIN
     TBD
END
```

4.1.2.6 Network Sign In Confirmation

```
network_activated(parameters)
BEGIN
    if return status not ok or nsdu size not defined
       set layer majorstate to non existence
       get emergency event message from common data area
       set event to network_not_operational
       send event message to System Management
       delete network activated mailbox
       turn on this layer mailbox
       return message to mempry pool and exit
    store network data mailbox id in common data area
    store the max. PDU size into common data area
    turn on the this layer management mailbox
    delete network activated mailbox
    set layer majorstate to in use state
    return this message to memory pool
END
```

4.2 Transport Process

4.2.1 Transport Initializtion Routine

```
Transport_init(transport layer instance data block pointer)
BEGIN
     save TLIDB pointer into PCB
     allocate memory for emergency message to System Management
     Fill in FC table pointers for IOLDs dispatching mailbox
     call initialize DDI transport function
END
```

4.2.2 Transport main function

```
Transport_main(msgptr,mbid)
BEGIN
     retrieve TLIDB pointer from PCB
     switch on message type
          case iold from IO software
               if layer majorstate not equal in-use-state
                  call iold return function with status
                  exit
               call common iold handler
               break
          case lcb arrival
               call lcb arrival function
               break
          case network data indicate
               call network data indicate handler
               break
          case data BUFIO arrival
               call data BUFIO arrival function
               break
          case data BUFIOX arrival
               call data BUFIOX arrival function
               break
          case lcb to l6 return confirmation
               call lcb to l6 clean up function
               break
          case L6 buffer descriptor arrival
               call L6 buffer descriptor function
               break
          case buflcbio confirmation
               call buflcbio confirmation function
               break
          case DDI resume data
               call resume write data function
               break
          default
               return memory to memory pool
               break
     switchend
END
```

4.2.2.1 LCB arrival function

```
lcb_arrival(parameters)
BEGIN
     if DMA return status not ok
        to be defined
     switch on lcb specific function code
          case write connection data
               call write connection data function
               break
          case write expedited data
               call write expedited data function
               break
          case write connectionless data function
               not supported at this time
               break
          case read connection data
               call read connection data function
               break
          case read expedited data
               call read expedited data function
               break
          case read connectionless data
               not supported at this time
               break
          case connect request
               call connect request function
               break
          case connect response
               call connect response function
               break
          case disconnect request
               call disconnect request Function
               break
          case connection indication event
               call connection indication function
               break
          case tsap event indication
               call tsap event indication function
               break
          default
               return memory to memory pool
               break
     switchend
END
```

4.2.2.1.1 Write Connection Data function

```
Write_connect_data(parameters)
BEGIN
    if logical local tsap address invalid
        return lcb with INVALID ADDRESS status and exit
    if connection identifier not in the connection directory
       return lcb with INVALID CONN ID and exit
    if tsap write credit is equal zero
       return lcb with CREDIT EXCEEDED status and exit
    decrement tsap write credit count
    switch on buffer indicator
         case buffer pointer in lcb
              if total range is > sdu size
                 return lcb with BUFFER_EXCEEDS_SDU and exit
              allocate memory to contain the buffer descriptor
              convert it into 'buffer descriptor type'
              if write data buffer ptr is not empty
                 link transaction block into TCCB and exit
              put this transaction block into TCCB
              initialize 'write_buf_info'
              call L6 write data buffer management routine
              break
         case data in lcb
              if write data buffer ptr is not empty
                 link transaction block into TCCB and exit
              put this WRITE CO transaction block into TCCB
              allocate memory for DDI parameters block
              allocate data buffer
              move data from lcb into data buffer
              set DDI function to T_DATA
              set EOT flag
              call DDI transport function
              break
         case buffer descritpor in L6
              allocate memory for WRITE CO transaction block
              allocate memory for buffer descriptors
              set message parameters
              set messate type to write data buffer descritpor
              call DMA module
              break
    switchend
END
```

4.2.2.1.2 Write Expedited Data function

The function assumes that the LCB contains an expedited TSDU, 1 to 16 bytes data in the LCB. Any data pointers in the LCB is invalid.

```
Write_exp_data(parameters)
BEGIN
    if logical local tsap addess invalid
        return lcb with INVALID ADDRESS status and exit
    if connection identifier not in the connection directory
       return lcb with INVALID_CONN_ID and exit
    if expedited option not supported
       return lcb with EXPEDITED NOT SUPPORTED and exit
    if tsap write credit count equal zero
       return lcb with WRITE CREDIT EXCEEDED status and exit
    decrement tsap write credit count
    switch on buffer indicator
         case buffer pointer in lcb
              return lcb with INVALID status
              break
         case data in lcb
              if datasize is zero or greater than 16
                 return lcb with INVALID status and exit
              get DDI parameter block
              get data buffer
              move data from lcb into data buffer
              set data size
              set function to T_EXPEDITED_DATA
              call DDI transport function
              return lcb to L6
              release memory
              break
         case buffer descritpor in L6
              return lcb with INVALID status
              break
    switchend
END
```

4.2.2.1.3 Read Connection Data function

```
Read_connection(parameters)
BEGIN
    if logical local address invalid
       return lcb with INVALID ADDRESS status and exit
    if connection identifier not in the connection directory
       return lcb with INVALID CONN ID and exit
    if tsap read credit count is equal zero
       return lcb with READ CREDIT EXCEEDED status and exit
    decrement tsap credit count
    switch on buffer indicator
        case buffer pointer in lcb
             if read buffer pointer is not null
                link this buffer pointer and exit
             allocate memory for this buffer descriptor
             put this buffer descriptor to TCCB
             if read data pending
                if total range is < buffer size
                  return lcb with BUFTOOSMALL status and exit
                call write data to l6 management function
             break
        case buffer descriptor in l6
             allocate memory for transaction block
             allocate memory for buffer descriptor
             set message parameters
             set message type to read buffer_descriptor
             send message to DMA module
             break
        default
             return lcb with NO BUFFER to L6
             break
     switchend
END
```

4.2.2.1.4 Read expediated Data function

```
read_exp_data(parameters)
BEGIN
    if logical local tsap address invalid
       return lcb with INVALID ADDRESS status and exit
    if connection identifier not in the connection directory
       return lcb with INVALID CONN ID and exit
    if tsap read credit count is equal zero
       return lcb with READ CREDIT EXCEEDED status and exit
    decrement tsap read credit count
    switch on buffer indicator
        case buffer pointer in lcb
             if read expedited buffer ptr is not null
                link this buffer pointer in TCCB and exit
             allocate memory for this buffer descriptor
             put this buffer descriptor into TCCB
             if expedited read data pending
                if total range is < 16
                  return lcb with BUFTOOSMALL status and exit
                call L6 read data buffer management routine
             break
        case buffer descriptor in L6
             allocate memory for transaction block
             allocate memory for buffer descriptor
             set message parameters
             set message type to read expedited buffer desc.
             send message to DMA module
             break
        case buffer in lcb
             if buffer size < 16
                return lcb with BUFTOOSMALL status and exit
             if expedited read data pending
                move data into lcb
                set range residue if necessary
                return lcb to L6
                clean up and release memory
             else put this transaction into TCCB
             break
        default
             return lcb with NO BUFFER to L6
             break
     switchend
END
```

4.2.2.1.5 Connect Request function

```
connect request(parameters)
BEGIN
    if logical local tsap address not within LTD
       return lcb with BADLCOAL tsap status and exit
    if logical remote tsap address not within RTD
       return lcb with BADREMTPE tsap status and exit
    if no room in connection directory
       return lcb with NO ROOM status and exit
    if there is user data
       switch on data buffer indicator in LCB
              case data in LCB
                   not supported
                   return LCB with INVALID status and exit
              case data buffer pointer in LCB
                   if read buffer is not available
                      return lcb with INVALID status and exit
                   get memory for CR transaction block
                   move read buffer to LCB trans.block
                   allocate buffer for data
                   set bufio request type
                   send message to DMA
                   break
              case data buffer descriptor in L6
                   get memory for cr transaction block
                   allocate memory for both buffer descriptor
                   move buffer pointer to transaction block
                   set transaction block type to CR_TRANS_BDI
                   send message to DMA for buffer descriptor
                   break
       switchend
    else
        if not read buffer for user data
           return lcb with INVALID status and exit
        increment current connection count in this local tsap
        allocate memory for TCCB
        initialize TCCB
        allocation memory for CR transaction block
        put this CR lcb into TCCB
        Write ID (index to connection directory) into TCCB
        put local local tsap address in this TCCB
        put logical remote tsap address in this TCCB
        build parameters to pass to DDI for connect request
        set expedited option
        set tpdu size from TLIDB
        set source tsap from local tsap
        set desination tsap from remote tsap
        set function to t_connnect_request
        set data size to zero
    call DDI transport function
END
```

4.2.2.1.6 Connect Response function

```
Connect_response(parameters)
BEGIN
    If there is no user data
       setup parameters to pass to DDI Transport
       set expedited optin
       set tpdu size from TLIDB
       set function to T_CONNECT_RESP
       set data size and data pointer
       call DDI transport function
       return lcb to l6
    else switch on buffer indicator
           case on data in buffer pointer in LCB
                if buffer range > 32 bytes
                   return lcb with DATAINVALID status
                   exit
                allocate memory for transaction block
                allocate memory for data buffer
                set transaction and message parameters
                set transaction type to CC BUFIO
                send message to DMA
                break
           case data in LCB
                if datasize > 32 bytes
                   return lcb with DATATINVALID status and
                   exit
                setup parameters to pass to DDI Transport
                set expedited optin
                set tpdu size from TLIDB
                set function to T_CONNECT_RESP
                set data size and data pointer
                call DDI transport function
                return lcb to l6
                break
           case data in buffer descriptor in L6
                allocate memory for transaction block
                allocate memory for buffer descriptor
                set transaction and message parameters
                set transaction type to CC buffer descr.
                send message to DMA
                break
       switchend
END
```

4.2.2.1.7 Disconnect request function

```
Disconnect request(parameters)
BEGIN
     If there is no user data
        set DDI transport paramters
        set function to T_DISCONNECT
        set data size to zero
        call DDI transport function
        return lcb to L6
        exit
     switch on data indicator
           case on data buffer pointer in LCB
                allocate memory for transaction block
                allocate memory for buffer data
                set transaction and message parameters
                set transaction type to DISC BUFIO
                send message to DMA
                break
           case data in LCB
                if data size exceeds 64 bytes
                set DDI transport parameters
                set function to T_DISCONNECT
                set data size
                call DDI transport function
                return lcb to L6
                break
           case data in buffer descriptor in L6
                get memory for transaction block
                get memory for buffer descriptor
                set transaction and message parameters
                set transaction type to DIC buffer desc
                send message to DMA
                break
     switchend
END
```

4.2.2.1.8 Connection Indication Event

```
Connection_indication_event(parameters)
BEGIN
    if connection identifier is not valid
       return INVALID_CONN_ID and exit
    if event lcb pointer not null in TCCB
       return old event lcb with new mask to 16 and exit
    switch on connection event mask
        case normal data arrival
             get TCCB with the connection id
             if data arrival pending flag is on
                set data length to SDU size in LCB
                return lcb to L6 and exit
             put the connection event LCB into TCCB
             break
        case normal write credit available
             call credit control function
             if return value positive
                move available credits to lcb
                set event mask to amount of addition credit
                return lcb to 16 and exit
             put this connection event lcb into TCCB
             break
        case disconnect indication
             if no user data buffer available
                return lcb with INVALID status and exit
             switch on data indicator
                  case on data in LCB
                       not supported
                       return lcb with INVALID status
                       break
                  case data buffer in LCB
                       get TCCB with connection identifier
                       if disconnect flag is pending
                          move reason code from TCCB
                          set event mask to reason code
                          clear disconnect pending flag
                          if user data available
                             allocate transaction block
                             move data pointer to tr block
                             set parameters for BUFLCBIO
                             send message to DMA
                             break
                          return lcb to 16 and exit
                       put connection event LCB in TCCB
                       break
        default
             return this lcb with INVALID MASK status
             clear connection event lcb pointer in TCCB
             break
    switchend
END
```

4.2.2.1.9 Tsap Event Indication

```
tsap_event(parameters)
BEGIN
    switch on tsap event mask
        case connection indication
            if not user data available
               return lcb with INVALID and exit
            switch on data indicator
               case data in lcb
                    not supported
                    return lcb with INVALID and exit
               case data buffer pointer in LCB
                    if connection indication queue not null
                       get the TCCB with connection id
                       put connection identifier in lcb
                       put logical remote tsap addr in lcb
                       set expedited option
                       put qos in lcb
                       if user data in this connect request
                          allocation memory for BUFLCBIO
                          set message parameters
                          move data pointer into lcb
                          move write credit to lcb
                          unlink event indication queue
                          set message to BUFLCBIO
                          send message to DMA
                          break
                       set message parameters
                       move write credit to lcb
                       unlink event indication queue
                       return lcb to 16 and exit
                    put this lcb into tsap event pointer
                    break
              case  on data buffer descriptor in L6
                    allocate memory for transaction block
                    allocate memory buffer descriptor
                    set transaction and messge parameters
                    set transaction type to tsap event bd
                    send message to DMA
                    break
        case tsap deactivated
            if deactivated tsap queue is not empty
               move TSAP Deactivated Reason into lcb
               return lcb to 16
            break
        default
            return lcb with UNKNOWN MASK status
            break
    switchend
END
```

4.2.2.2 Network Data Indicate

```
Network_data_indicate(parameters)
BEGIN
     build parameters for DDI to understand
     return message to memory pool
     call DDI T_network()
END
```

4.2.2.3 Data BUFIO arrival function

```
Data BUFIO arrival (parameters)
BEGIN
     switch on transaction type
          case connect request
               retrieve all logical tsap address from LCB
               increment current connection count
               allocate memory for TCCB
               initialize TCCB
               put CR transaction block to TCCB
               add TCCB to connection directory
               release data buffer arrival message
               build parameters to pass to DDI for CR
               call DDI
               break
          case read CO data confirmation
               call normal data confirmation function
               break
          case eot read CO data confirmation
               call eot normal read data confirmation
               break
          case write CO data
               call normal tpdu data arrival function
               break
          case eot write CO data
               call eot normal tpdu data arrival function
               break
          case write expedited data
               call expedited data arrival function
               break
          case eot write expedited data
               call eot expedited data arrival function
               break
          case disconnect request
               call disconnect user data arrival function
               break
          case connect response
               call connect response data arrival function
               break
          default
               break
     switchend
 END
```

4.2.2.3.1 Normal tpdu data arrival function

```
Normal tpdu(parameters)
BEGIN
     if DDI transport flow control flag is on
        queue this tpdu for later and exit
    set transport id
    move data pointer into DDI parameter block
    set function = T_DATA
    call DDI transport function
    call L6 write data buffer mangement routine
END
```

4.2.2.3.2 Eot normal tpdu data arrival function

```
eot normal tpdu(parameters)
BEGIN
     if DDI transport flow control flag is on
        queue this tpdu for later and exit
    set transport id
    move data pointer into DDI parameter block
    set function = T_DATA
    set eot flag on
    call DDI transport function
    call return write CO lcb to L6
    unlink write CO from queue
    if normal write CO data queue is not empty
       call L6 write data buffer mangement routine
END
```

4.2.2.3.3 Expedited tpdu data arrival function

```
Expedited tpdu(parameters)
BEGIN
    if DDI transport flow control flag is on
       queue this tpdu for later
    else
       set transport id
       move data pointer into DDI parameter block
       set function = T_EXPEDITED_DATA
       call DDI transport function
       set expedited flag on
       call L6 write data buffer mangement routine
END
```

4.2.2.3.4 Eot expedited write tpdu data arrival function

```
eot expedited tpdu(parameters)
BEGIN
     if DDI transport flow control flag is on
        queue this tpdu for later
     else
        set transport id
        move data pointer into DDI parameter block
        set function = T_EXPEDITED_DATA
        set eot flag on
        call DDI transport function
        call return expedited write CO lcb to L6
        unlink expedited write CO from queue
        if expedited write CO data queue is not empty
           set expedited flag
        call L6 write data buffer mangement routine
END
```

4.2.2.3.5 Read tpdu data confirmation function

```
Read tpdu conf(parameters)
BEGIN
    release the transaction block
    release data buffer to memory
END
```

4.2.2.3.6 EOT normal tpdu data confirmation function

```
EOT tpdu data conf(parameters)
BEGIN
    return Read CO lcb to L6
    unlink Read CO from queue
    clean up and release all memory
END
```

4.2.2.3.7 Disconnect user data arrival function

```
Discnect data(parameters)
BEGIN
     set DDI transport parameters
     set function to T_DISCONNECT
     set data size
     call DDI transport function
     return disconnect request lcb to L6
END
```

4.2.2.3.9 Connect Response data arrival function

```
 Connect response data(parameters)
 BEGIN
       setup parameters to pass to DDI Transport
       set expedited optin
       set tpdu size from TLIDB
       set function to T_CONNECT_RESP
       set data size and data pointer
       call DDI transport function
       return lcb to l6
 END
```

4.2.2.4 L6 buffer descriptor arrival function

```
L6 bfdes(parameters)
BEGIN
     switch on transaction type
          case read CO data
               call read CO data bd arrival function
               break
          case read expedited CO data
               call read expedited data bd arrival function
               break
          case write CO data
               call write CO data bd arrival function
               break
          case write expedited Co data
               call write expedited bd arrival function
               break
          case connect request
               call connect request bd arrival function
               break
          case disconnect request
               call disconnect request bd arrival function
               break
          case tsap event indicator
               call tsap event bd arrival function
               break
          case connect response
               call connect response bd arrival function
               break
          default
               break
     switchend
END
```

4.2.2.4.1 Read CO data buffer descriptror arrival function

```
Read CO bfdes(parameters)
BEGIN
     initialize 'read_buf_info'
     if read buffer pointer is not null
        link this buffer pointer and exit
     put this buffer descriptor into TCCB
     if read data pending
        if total range is < buffer size
           return lcb with BUFTOOSMALL status and exit
        call L6 read data buffer management routine
        with normal data flag on
END
```

4.2.2.4.2 Read Expedited data buffer descriptor arrival function

```
Read Exped bfdes(parameters)
BEGIN
     if read expedited buffer ptr is not null
        link this buffer pointer in TCCB and exit
     put this buffer descriptor into TCCB
     if expedited read data pending
        if total range is < buffer size
           return lcb with BUFTOOSMALL status and exit
        get memory for BUFIO transactio block
        move data into buffer
        set transaction type to EXPED_READ_CO
        if range residue if necessary
        send message to DMA
END
```

4.2.2.4.3 Write CO data buffer descriptor arrival function

```
Write CO bfdes(parameters)
BEGIN
     if total range is > sdu size
        return lcb with BUFFER_EXCEEDS_SDU and exit
     initialize 'write_buf_info'
     if write CO buffer ptr is not null
        link this buffer ptr into queue and exit
     put this buffer pointer into TCCB queue
     call L6 write data buffer management routine
     with normal data flag on
END
```

4.2.2.4.4 Write Expedited CO data buffer descriptor arrival function

```
Write Exped bufdes(parameters)
BEGIN
     if total range is > sdu size
        return lcb with BUFFER_EXCEEDS_SDU and exit
     if write data buffer ptr is not empty
        link this buffer pointer in TCCB and exit
     get memory for BUFIO transaction block
     get memory for data buffer
     set transaction type to EXPED_WRITE
     send message to DMA
END
```

4.2.2.4.5 Connect request data buffer descriptor arrival function

```
Connect request bfdes(parameters)
BEGIN
     get memory for CR transaction block
     move write/read buffers into CR trans.block
     allocate buffer for data
     set bufio request type
     send message to DMA
END
```

4.2.2.4.6 Disconnect Request data buffer descriptor arrival function

```
Disconnect bfdes(parameters)
BEGIN
     allocate memory for discon transaction block
     allocate memory for buffer data
     set transaction and message parameters
     set transaction type to DISC BUFIO
     send message to DMA
END
```

4.2.2.4.7 TSAP Event data buffer descriptor arrival function

```
TSAP bfdes(parameters)
BEGIN
     scan each TCCB for this tsap
     if connection indication queue not null
        get the TCCB with connection id
        put connection identifier in lcb
        put logical remote tsap addr in lcb
        set expedited option
        put qos in lcb
        if user data in in this connect request
           allocation memory for BUFLCBIO
           set message parameters
           move data pointer into lcb
           move write credit to lcb
           unlink event indication queue
           set message to BUFLCBIO
           send message to DMA
           exit
        else
           set message parameters
           move write credits into lcb
           unlink event indication queue
           set message lcb to 16
           send message to DMA
     else put this buffer descriptor into tsap event queue
 END
```

4.2.2.4.8 Connect Response data buffer descriptor arrival function

```
Connect response bfdes(parameters)
BEGIN
     if buffer range > 32 bytes
        return lcb with DATAINVALID status
        exit
     allocate memory for transaction block
     allocate memory for data buffer
     set transaction and message parameters
     set transaction type to CC BUFIO
     send message to DMA
 END
```

4.2.2.5 Data BUFIOX arrival function

```
Data BUFIOX arrival (parameters)
BEGIN
     switch on transaction type
            case Write CO data
                 call normal tpdu data arrival function
                 break
            case Expedited Write Co data
                 call expedited tpdu data arrival function
                 break
            case read CO data confirmation
                 call read CO data confirmation function
                 break
            case eot write CO data
                 call eot normal tpdu data arrival function
                 break
            case eot expedited write data
                 call eot expedited write data arrival function
                 break
            case disconnect request
                 call disconnect user data arrival function
                 break
            case connect response
                 call connect response data arrival function
                 break
            case read expedited CO data
                 call expedited tpdu data arrival function
                 break
            default
                 break
     switchend
END
```

4.2.2.5 Lcb to L6 return confirmation function

```
Lcb to L6 confirm(parameters)
BEGIN
     clean up and release all memory
END
```

4.2.2.6 BUFLCBIO to L6 return confirmation function

```
buflcbio to l6(parameters)
BEGIN
     clean up and release all memory
END
```

4.2.2.7 DDI resume write data function

```
Resume(parameters)
BEGIN
     If there is data queue up in the TCCB
        get DDI parameters block
        set parameters
        set function to T_DATA
        call DDI transport function
     else call L6 write data buffer mangement routine
END
```

4.3 DDI Transport Network Request

```
N_Network(netw)
struct fpt_netw *netw
BEGIN
     allocate memory to send message to network layer
     format message
     enter parameters to message
     send message
END
```

4.4 DDI Transport Indication Function

```
DDI_Transport(fpt)
   struct fpt_tran *fpt
BEGIN
     switch on fpt->function
         case T_CONNECT_REQ
              get local tsap number from LTD with pid
              get activated tsap directory from local tsap
              call search remote tsap selector name from RTD
              if name not found
                 if dymanic configuration not allowed
                    call disconnect request to DDI and exit
              if remote directory has no room
                 call disconnect request to DDI and exit
              if current connection count equal to max
                 call disconnect request to DDI and exit
              allocate memory for dynamic remote tsap
              add entry into the remote directory
              increment next remote directory entry pointer
              increment activated remote tsap count
              initialize this remote tsap
              allocate transport connection control block(TCCB)
              attach TCCB to connection directory
              increment current connection count
              initialize TCCB(see connect request)
              if tsap event pointer is null in this tsap
                  set tsap event pending flag = connection
                  set logical local tsap address in connect ind
                  set connection id in connect indicate
                  and exit
              if tsap event mask in not connection indicate
                  same as above
              put connection id in lcb
              put logical remote tsap address in lcb
              set expedited option  tp expedited flag
              put qos in lcb
              if user data in this connect reqeust
                 move data in lcb
                 set data indicator flag on in lcb
              copy write credit to lcb
              return lcb to l6
              break
```

```
case T_CONNECT_RESP
     call DDI with connection identifier to obtain
          local tsap #,connection directory index
     get connection TCCB
     get connect request lcb from TCCB
     move in all parameters from connection parameters
     to lcb output parameters
         . connection identifier = DDI connection id
         . expedited option
         . quality of service
         . max.SDU size
         . CO read credit = the smaller of cc tdpu or
                            cr tpdu
         . CO write credit
         if remote user data in this cc tpdu
            if output buffer is not available
               set lcb status with NOROOM for DATA
               return cr lcb to 16
               set cr lcb in TCCB to null and exit
            setup message to transfer remote user data
              to output buffer.
            set up message to return lcb to 16
            set cr lcb in TCCB to null
     break
```

```
case T_DISCONNECT
     call DDI with connection id to obtain local
          tsap and connection directory index
     if connect request lcb still outstanding
        move in all parameters into
        set status to disconnection reason
        return lcb to l6
        release TCCB
        remove connection id from connection directory
        decrement local tsap connection count
        exit
     if there is no connection event available
        set disconnect pending flag on
        copy disconnect reason into TCCB and exit
     if connection event mask is not disconnect
        set disconnection pending flag on
        copy disconnect reason into TCCB and exit
     move parmeters into connect event lcb
     send message to return lcb to l6
     release TCCB memory to memory pool
     break
case T_DATA
     get connection TCCB
     if read data buffer descriptor is null
        if connection event is null
           return to DDI with status to queue tpdu
        if connection event is not normal data arrival
           return to DDI with status to queue tpdu
     if total range is < data size
        return to DDI with status to queue tpdu
     call L6 write data buffer mangement routine
     return to DDI with good status
     break
```

```
                case T_EXPEDITED
                     get connection TCCB
                     if expedited buffer is null
                        break
                     switch on buffer indicator
                        case buffer pointer in lcb
                             allocation bufiolcb transaction block
                             move data pointer into transaction block
                             clean up and release memory
                             send message to DMA return data and lcb
                             break
                        case buffer in lcb
                             move data into lcb
                             return lcb to L6
                             clean up and relase memory
                             break
                        case buffer point in L6
                             allocate bufiox transaction block
                             move data pointer inot trans. block
                             clean up and release memory
                             send message to DMA
                             break
                case T_FLOW
                     get connection TCCB
                     set transmit data flow control flag on
                     break
                case T_FLOW_STOP
                     allocate memory for message to send to itself
                     to wait up to continue to perform transfer across
                     Level 6 memory
                     set message to resume send data
                     set local tsap address and connection id
                     break
                default
                     break
          END
```

4.5 Common Supporting Routines

4.5.1 IOLD Handler(parameters)

This message sent by IO Dispatcher indicating an iold arrival. This is common routine for all processes.

```
IOLD_handler(parameters)
 BEGIN
      Allocate memory for transport transaction block
      allocate memory for LCBI
      move level 6 address and range into LCBIO
      save level address and range for return
      move 6 bit channel number into LCBIO
      save channel number for return
      clear interrupt level to zero
      setup the other message parameters
      release iold message to memory pool
      send message to DMA to bring in LCB
 END
```

4.5.2 L6 transmit data buffer management

This function will copy the host data into the Lacs buffer one TPDU at a time. The following structure is needed to operate the read Level 6 data and it must be in the TCCB. The caller of this function must setup the first buffer address, range and the total range when it calls for the first time. After that it is the responsibility of this function to update the current buffer information. The main function is the segmentation of data from a list of L6 buffer descriptors; build a L6 buffer descriptor list that DMA module can understand; keep track of each L6 buffer descriptor being used;

```
struct write_buf_info
       BEGIN
            ushort   curbuf       current working buffer number
            ushort   bufleft      number of outstanding buffers
            ulong    total_range  total ranges in all buffers
            ulong    cur_address  current buffer working address
            ulong    cur_range    current buffer working range
            BDI      *wr_bdi_blk  ptr to write buffer descriptor
       END

struct bdi
       BEGIN
           ushort    count        number of buffers
           struct bufdes
                BEGIN
                     ulong     buf_addres  buffer address
                     ulong     buf_ind     buffer indicator
                     ulong     buf_range   buffer range
                     ulong     buf_rsr     buffer residual range
                END bdides[count]
       END
```

```
L6_write_data(tctx,flag)
BEGIN
     if flag is NORMAL
        get normal write data ptr from TCCB
     else get expedited write data ptr from TCCB
     get transaction block from TCCB
     move in local tsap number into transaction block
     move in connection id into transaction block
     if total_range > tpdusize
     BEGIN
          ptr = allocate buffer memory sizeof tpdusize
          move ptr to message bufdes pointer
          total_range = total_range - tpdusize
          if cur_range > tpdusize
          BEGIN
               move buffer descriptor information into BUFIO
               transaction block
               update current working address and range
               set transaction type = NORMAL_WRITE
               send message to DMA
          END
          else if current range == tpdusize
          BEGIN
               move buffer descriptor information into BUFIO
               transaction block
               increment the current buffer count by 1
               copy next buffer descriptor information into
               current working address and range
               set transaction type to EOT_WRITE_CO
               send message to DMA
          END
```

```
else current range is < tpudsize
BEGIN
     n = find the size of the buffer descriptor to be
     build for DMA
     if n =< 9
        move buffer descriptor information into BUFIO
        transaction block
        move buffer descriptor information into
        BUFIO transaction block
        set message type to BUFIO
     else
        get sizeof (16_DES * n + 2) memory to build a
        list of buffer descriptor for BUFIOX
        transaction block
        set message type to BUFIOX
        move buffer descriptor information into the
        list
     update current working buffer range and address
     update current working buffer count if necessary
     set transaction type to NORMAL WRITE CO
     send message to DMA module
END
```

```
        else total range =< tpdusize
        BEGIN
             ptr = allocate buffer memory sizeof total range
             move ptr to message bufdes pointer
             if bufleft =< 9
                move buffer descriptor information into BUFIO
                transaction block
                set message type to BUFIO
             else
                get sizeof(L6_DES * bufleft + 2) memory to build a
                list of buffer descriptor for BUFIOX transaction
                block
                move buffer descriptor information into the list
                set message type to BUFIOX
             set transaction type to EOT_WRITE_CO
             send message to DMA
             endif
        END

    END
```

4.5.3 L6 Receive data buffer management

This function will write LACS buffer data into L6 host memory one TPDU at a time. The following structure is needed to operate the write data into L6 and it must be in the TCCB. The caller of this function must setup the first buffer address, range and the total range when it calls for the first time. After that it is the responsibility of this function to update the current buffer information. Alos, it is the caller 's responsibility to make sure there is room in L6 memory to hold the user's data. This function assumed there is always room. The main function is the reassembly of data to a list of L6 buffer descriptors; build a L6 buffer descriptor list that DMA module used; keep track of each L6 buffer descriptor being used;

```
struct read_buf_info
       BEGIN
            ushort   curbuf       current working buffer number
            ushort   bufleft      number of outstanding buffers
            ulong    total_range  total ranges in all buffers
            ulong    cur_address  current buffer working address
            ulong    cur_range    current buffer working range
            BDI      *rd_bdi_blk  ptr to read buffer descriptor
       END

struct bdi
       BEGIN
           ushort    count        number of buffers
           struct bufdes
              - BEGIN
                     ulong     buf_addres  buffer address
                     ulong     buf_ind     buffer indicator
                     ulong     buf_range   buffer range
                     ulong     buf_rsr     buffer residual range
                END bdides[count]
       END
```

```
L6_read_data(tctx,datasize,flag)
BEGIN
     if flag is NORMAL
        get normal write data ptr from TCCB
     else get expedited write data ptr from TCCB
     get transaction block from TCCB
     move in local tsap number into transaction block
     move in connection id into transaction block
     if total_range > datasize
     BEGIN
          ptr = allocate buffer memory sizeof datasize
          move ptr to message bufdes pointer
          total_range = total_range - datasize
          if cur_range > datasize
          BEGIN
               move buffer descriptor information into BUFIO
               transaction block
               update current working address and range
               if flag is equal read CO
                  set transaction type to READ_CO
               else
                  set transaction type to EOT_READ_CO
                  udpate buffer residue range
               send message to DMA
          END
```

```
else if current range == datasize
BEGIN
     move buffer descriptor information into BUFIO
     transaction block
     increment the current buffer count by 1
     copy next buffer descriptor information into
     current working address and range
     if flag equals to read CO
        set transaction type to READ_CO
     else
        set transaction type to EOT_READ_CO
        udpate buffer residue range
     send message to DMA
END
```

```
else current range is < datasize
BEGIN
     n = find the size of the buffer descriptor to be
     build for DMA
     if n =< 9
        move buffer descriptor information into BUFIO
        transaction block
        move buffer descriptor information into
        BUFIO transaction block
        set message type to BUFIO
     else
        get sizeof (16_DES * n + 2) memory to build a
        list of buffer descriptor for BUFIOX
        transaction block
        set message type to BUFIOX
        move buffer descriptor information into the
        list
     update current working buffer range and address
     update current working buffer count if necessary
     if flag equals to read CO
        set transaction type to READ_CO
     else
        set transaction type to EOT_READ_CO
        udpate buffer residue range
     send message to DMA module
END
```

```
        else total range is equal to datasize
        BEGIN
             ptr = allocate buffer memory sizeof total range
             move ptr to message bufdes pointer
             if bufleft =< 9
                move buffer descriptor information into BUFIO
                transaction block
                set message type to BUFIO
             else
                get sizeof(L6_DES * bufleft + 2) memory to build a
                list of buffer descriptor for BUFIOX transaction
                block
                move buffer descriptor information into the list
                set message type to BUFIOX
              set total range = 0
              set transaction type to EOT_READ_CO
              send message to DMA
        END
     END
```

4.5.6 Update statistics ,bump state by one. The statistics is kept on a tsap basis.

```
T_note(parameters)
BEGIN
     get the local tsap table with the parameter
     switch on parameter
            case  NDTOCTSENT
                  ndtoctent++
                  break
            case  NDTOCTRECD
                  ndtoctrecd++
                  break
            case  EDTOCTSENT
                  edtoctsent++
                  break
            case  EDTOCTRECD
                  edtoctrecd++
                  break
            case  TPDUSENT
                  tpdusent++
                  break
            case  TPDURESENT
                  tpduresent++
                  break
            case  TPDURECD
                  tpdurecd++
                  break
            case  DTPDRESENT
                  dtpdresent++
                  break
            case  ATPDRESENT
                  atpdresent++
                  break
            case  DISCONNREQ
                  disconnreq++
                  break
            case  OPNCONNECT
                  opnconnect++
                  break
            case  RFCONNECT1
                  rfconnect1++
                  break
            case  RFCONNECT2
                  rfconnect2++
                  break
            case  ICONNECTOK
                  iconnectok++
                  break
            case  OCONNECTOK
                  oconnectok++
                  break
```

```
                    case  OCONNECTNO
                          oconnectno++
                          break
                    case  CONNTIMOUT
                          conntimout++
                          break
                    case  CREQRESENT
                          creqresent++
                          break
                    case  ERTPROTCOL
                          ertprotcol++
                          T_event(TEPROTOCOL)
                          break
                    case  ERINVTPDUS
                          erinvtpdus++
                          break
                    case  TEBADABORT
                          tebadabort++
                          T_event(TEBADABORT)
                          break
                    default
                          break
          END
```

4.5.7 Event Notification Routine
Report an event to System Management

```
T_event(parameters)
BEGIN
END
```

4.6 Modification to DDI Transport Function

4.6.1 Tsap selector Initializatioin

The DDI has an array of local tsap each contains a tsap selector (suffix),a PID and a routine entry pointer for DDI to call.

```
struct convsuffix consuff[]
struct convsuffix
       short suffix
       long  pid
       int   (*sentry())
```

The suffix is administratively setup and PID is entered by invoking T_Init_req primitive call to DDI Transport.This structure is used for remote initiated connect request acceptance purpose. If the called tsap id is not found or the PID is null in the array then the remote initiated connect request will not be accepted. They are imcompatable with the Lacs setup environment. It must be modified to accommondate the Lacs requrirment.The modification are as follow:
The size of the array is passed in startup parmeters, by the System Management.Each entry to this array is the local tsap number which is and index to the local tsap directory. Therefore the suffix will become the local tsap selector of the local tsap,and the PID will be the lcoal tsap directory index number. This will quicken the serach for the remote tsap selector and identify the local connection directory easy. Also,there is no need sentry entry requirement since the DDI call is known within the layer.

4.6.2 DDI buffer Management

When the session layer or the network layer hands the DDI a pdu the DDI transport will return the pdu for the called layer to release the associated pdu buffer. In case the DDI transport has to holdback the pdu for retransmission or the data cannot send to session layer it copies the data before return to caller This is a costly performance penalty. Therefore the copy business must be modified. In the transmit data case a copy of the buffer descriptor is passed to the network layer so that it can be released by the network layer. In case of the receive data it is the responsiblity of the transport to release the buffer.No the original buffer will not be returned to the network on the return call.

4.6.3 DDI SDU Segmentation

The T_SAVE_DT function performs segmenting a single data TSDU into multiple TPDUs before actually sending them to remote entity. This function assumes that the user passes a complete TSDU and therefore it inserts eot on the last TPDU. This causes problem with buffer resource mangement. In case of large file transfer it will take away our entire buffer resources. This routine must be modified to accept one TPDU data and set eot according to the user,s wish.