BAB 5
KESIMPULAN DAN SARAN

5.1 KESIMPULAN

Berdasarkan pembahasan perangkat lunak dan uji coba yang telah diuraikan pada bab-bab sebelumnya maka dapat ditarik beberapa kesimpulan:

1. Antarmuka EN2000 mempunyai kompatibilitas yang memadai untuk beberapa protokol, tergantung dari peng endali yang ada pada disket *setup*.
2. Protokol IPX memiliki performansi yang baik dalam kecepatan transmisi, sehingga untuk transmisi yang memerlukan kecepatan tinggi, namun IPX tidak memastikan bahwa berkas sampai ke tempat tujuan dengan berhasil.
3. Aturan atau protokol berperan dominan dalam transmisi data untuk menentukan kecepatan transmisi. Karena *header* IPX kecil, maka kecepatannya juga menjadi besar, sebab aturan yang dipakai juga sederhana.
4. Bahasa pemrograman C++ cukup handal di dalam penulisan program transmisi IPX pengatur antarmuka perangkat keras, karena C++ memiliki fungsi-fungsi yang mencakup pengaturan perangkat keras dan lunak yang efisien.

5.2 SARAN

Berdasarkan hasil-hasil yang telah dicapai dan berbagai kendala-kendala yang ada maka penulis memberikan saran sebagai berikut:

1. Tugas akhir ini dapat dikembangkan menjadi lebih baik lagi untuk transmisi yang lebih kompleks dengan menggunakan lapisan OSI yang lebih tinggi atau protokol yang lebih canggih.
2. Untuk perbaikan di bidang administrasi dan keteraturan tugas akhir, misalnya jauh sebelum tugas akhir dibuat, mahasiswa/i telah dimotivasi untuk menentukan judul-judul apa saja yang menarik dan perlu dikembangkan. Dengan demikian maka tugas akhir juga akan lebih berkualitas karena mahasiswa/i telah mencari bahan saat kuliah di mana pelajaran masih segar, sehingga hasilnya juga pasti lebih memuaskan.
3. Saat praktikum merupakan saat yang tepat untuk memotivasi mahasiswa/i dalam memilih perangkat yang akan dikembangkan lebih lanjut.

DAFTAR PUSTAKA

[ARI94] Arianto Widyatmo, *Belajar Mikroprosesor-Mikrokontroler melalui komputer PC*, Elex Media Komputindo, Jakarta, © 1994.

[CAM94] Campbell, Joe, *CProgrammer's Guide to Serial Communications*, SAMS Publishing, U.S.A., © 1994.✓

[EDI93] Ediman Lukito, *Belajar Sendiri Pemrograman dengan Turbo Pascal 7.0*, Elex Media Komputindo, Jakarta, © 1993.

[FRA92] Frank J. dan Derfler, Jr, *Paduan Menggabungkan LAN*, Elex Media Komputindo, Jakarta, © 1992.

[FRE94] Freed, Les dan Frank J. Derfler, Jr, *Paduan Komunikasi Modem*, Elex Media Komputindo, Jakarta, © 1994.

[GOO90] Goodwin, Mark, *Graphical User Interface in C*, Management Information Source, Inc., Portland, Oregon, U.S.A., © 1990.✓

[HAL92] Hall, Douglas V., *Microprocessors and Interfacing*, McGraw-Hill, Inc., Singapore, International Edition © 1992. ✓

[JAM91] Jamsa, Kris, *DOS Programming: The Complete Reference*, Osborn McGraw-Hill, Inc., U.S.A., Osborn Asian Student Edition © 1991. ✓

[LUK91] Lukas Tanutama. *Mengenal Local Area Network*, Elex Media Komputindo, Jakarta, © 1991.

[LUK89] Lukas Tanutama. *Pengantar Komunikasi Data*. Elex Media Komputindo, Jakarta, © 1989.

[MCD94] McDaniel, George (Compiler and Editor), *IBM Dictionary of Computing*, McGraw-Hill, Inc., U.S.A., © 1994.✓

[PRE92] Pressman, Roger S., *Software Engineering A Practitioner's Approach, Third Edition*, McGraw-Hill, Inc., Singapore, © 1992. ✓

[SCH92] Schatt, Stan, *Understanding Local Area Networks*, SAMS Publishing, U.S.A., ©1992.✓

[SIM94] Simpson, Alan. *Up & Running with DOS 6.2*, Elex Media Komputindo, Jakarta, © 1994.

[SMI94] Smith, Norman E., *Panduan Praktis Berilustrasi Borland C++*, Elex Media Komputindo, Jakarta, © 1994.

[STO92] Storey, Neil (University of Warwick, Coventry), *Electronics A Systems Approach*, Addison-Wesley Publishing Company, Inc., Great Britain, © 1992.✓

[STR91] Stroustrup, Bjarne, *The C++ Programming Language, Second Edition*, AT&T Bell Telephone Laboratories, Inc., Addison-Wesley Publishing Company, Inc., U.S.A, © 1991.✓

[SUM89] Sumantri Slamet I.S., *Pengantar struktur Data*. Elex Media Komputindo, Jakarta, © 1989.

[SUS95] Susanto, *Belajar Sendiri Pemrograman dengan Bahasa Assembly*, Elex Media Komputindo, Jakarta, © 1995.

[SUT91] Sutiono Gunadi, FX., *Belajar Sendiri Memahami Konsep Local Area Network*, Elex Media Komputindo, Jakarta, © 1991.

[VIJ92] Vijay Mukhi, *The 'C' Odyssey Network And RDBMS*, Tech Publication Pte Ltd., Singapore, © 1992.✓

[VOS91] Voss, Greg, *Object-Oriented Programming, An Introduction*, Osborne McGraw-Hill, Inc., U.S.A., © 1991. ✓

[WAH91] Wahyu C. Wibowo, *Pemrograman Berorientasi Objek*, Elex Media Komputindo, Jakarta, © 1991.

[YUN94] Yuniarto Nurwono, Ir., MBA, *Manajemen Informasi Pendekatan Global*, Elex Media Komputindo, Jakarta, © 1994.

Tulisan ilmiah:

[BOD90] Bodine, Bill, *A Comparison of NetWare IPX, SPX and NetBIOS*, Novell, Inc., Provo, Utah, © 1990.

[CUN94] Cunnelly, Adrian M., *NetWare C Library Version 2.3 Reference Manual*, Compuserve: 100031,222, © 1994.

[TUR90] Turner, Paul, *NetWare Communications Processes*, Novell, Inc., Provo, Utah, © 1990.

[WES91] Westin, Glenn and Ken Neff, *Logging Into IBM LAN Server and NetWare from DOS Workstation*, Novell, inc., Provo, Utah, © 1991.

Majalah:

[MO495] Mikrodata Media Penggemar Komputer, Volume 4 Seri 10, Elex Media Komputindo, Jakarta, © 1995.

[M1195] Mikrodata Media Penggemar Komputer, Volome 11 Seri 10, Elex Media Komputindo, Jakarta, © 1995.

[M0396] Mikrodata Media Penggemar Komputer, Volome 3 Seri 11, Elex Media Komputindo, Jakarta, © 1996.

Manual:

[MAN95] *80486 Motherboard User's Guide*, True Green, © 1995.

LAMPIRAN A

LISTING HEADER IPX, IPX.H

```
/*************************************************************************/
/*************************************************************************/
/*         IPX/SPX PROTOCOL APPLICATIONS PROGRAMMING INTERFACE           */
/*************************************************************************/
/*************************************************************************/

#ifndef IPXUSER
#define IPXUSER

#include <time.h>

/* ************************************************************************ */
/* ************************************************************************ */

/*-----------                                                              */
/* ===========                                                              */
/* constants */
/* -----------                                                              */
/* ===========                                                              */

/* ----------------- */
/* MISCELLANEOUS */
/* ----------------- */

#define BYTE              unsigned char

#define TRUE              1
#define SEGSIZE           65536L
#define ESCAPE_KEY        0x1B
#define RSHIFT_KEY_MASK   0x01
#define LSHIFT_KEY_MASK   0x02
#define CTRL_KEY_MASK     0x04
#define ALT_KEY_MASK      0x08
#define KEYB_FLAGS        0x00400017
#define FILE_PROGRAM      -1

#define SEQLEN            1
#define MINIPXDATALEN     512
#define MAXIPXDATALEN     4096
#define ACKSTR            "ACK"
#define FIRST_SEQ_NUM     1

#define IPX_TIMEOUT       1    // # secs to wait before retry
#define IPX_TRIES         5    // # times to try RX/TX before quitting
```

```
/*                                                    */
/* IPX/SPX COMMANDS, CODES, & PARAMETERS              */
/* ---------------------                              */

/* packet types */
#define UNKNOWN_PACKET_TYPE                 0x00
#define ROUTING_INFO_PACKET_TYPE            0x01
#define ECHO_PACKET_TYPE                    0x02
#define ERROR_PACKET_TYPE                   0x03
#define IPX_PACKET_TYPE                     0x04
#define SPX_PACKET_TYPE                     0x05

/* ipx presence detection */
#define IPX_PRESENT                         0xFF
#define IPX_MULTIPLEX                       0x7A00
#define DOS_MULTIPLEX_VECTOR                0x2F

/* ipx API commands */
#define SOCKET_OPEN                         0x0000
#define SOCKET_CLOSE                        0x0001
#define GET_LOCAL_TARGET                    0x0002
#define SEND_PACKET                         0x0003
#define LISTEN_FOR_PACKET                   0x0004
#define SCHEDULE_EVENT                      0x0005
#define CANCEL_EVENT                        0x0006
#define GET_INTERVAL_MARKER                 0x0008
#define GET_INTERNETWORK_ADDR               0x0009
#define RELINQUISH_CONTROL                  0x000A
#define DISCONNECT_FROM_TARGET              0x000B

/* ---------------------  */
/* input parameters       */
/* ---------------------  */

/* socket parameters */
#define SOCKET_DYNAMIC_ASSIGN               0x00
#define SOCKET_SHORT_LIVED                  0x00
#define SOCKET_LONG_LIVED                   0xFF

/* ---------------------  */
/* IPX return codes       */
/* ---------------------  */

/* generic */
#define IPX_SUCCESS                         0x00

#define EVENT_CANCELED                      0xFC
#define FRAGMENT_SIZE_ERROR                 0xFD

/* IPXSocketOpen() */
#define SOCKET_TABLE_FULL                   0xFE
#define SOCKET_ALREADY_OPEN                 0xFF

/* IPXGetInternetworkAddress() */
#define DEST_NODE_PATH_NOT_FND              0xFA

/* IPXListenForPacket() */
#define ECB_SOCKET_NOT_OPEN                 0xFF

/* IPXSendPacket() */
#define DESTINATION_NOT_FOUND               0xFC
#define NETWORK_FAILURE                     0xFF
```

```
/* IPXCancelEvent() */
#define CANNOT_CANCEL_ECB                0xF9
#define ECB_NOT_IN_USE                   0xFF

/* ------------------------------------ */
/* "Insulation Layer" Return Codes      */
/* ------------------------------------ */

#define GOOD_IPX_RETURN                  0x00

#define USER_TERMINATED                  0xBB
#define IPX_NOT_INSTALLED                0xDD
#define BAD_MEMORY_ACCESS                0xEE

#define BAD_LOCAL_TARGET                 0x10
#define BAD_SOCKET_PARAM                 0x11
#define BAD_SOCKET_OPEN                  0x12

#define BAD_TX_TRY                       0x20
#define BAD_RETRIES                      0x21
#define BAD_RX_SEQ_NUM                   0x22
#define BAD_RX_SRCE_ADDR                 0x23

#define BAD_FILE_OPEN                    0x30
#define BAD_FILE_WRITE                   0x31
#define BAD_FILE_READ                    0x32

#define BAD_RX_BFR_LEN                   0x40

/* ------------------------------------ */
/* ------------------------------------ */
/* macros */
/* ------------------------------------ */
/* ------------------------------------ */

#define HILOSWAP( x )       ((x << 8) + (x >> 8))
#define CTRLHIT             ( *(BYTE far *)KEYB_FLAGS & CTRL_KEY_MASK )
#define ALTHIT              ( *(BYTE far *)KEYB_FLAGS & ALT_KEY_MASK )
#define RSHIFTHIT           ( *(BYTE far *)KEYB_FLAGS & RSHIFT_KEY_MASK )
#define LSHIFTHIT           ( *(BYTE far *)KEYB_FLAGS & LSHIFT_KEY_MASK )
#define USERBREAK           ( RSHIFTHIT && LSHIFTHIT && ALTHIT )
#define SAFEACOPY( x, y )   x[0] = '\0', strncat( x, y, sizeof(x)-1 )

/* ------------------------------------ */
/* ------------------------------------ */
/* structures                           */
/* ------------------------------------ */
/* ------------------------------------ */

/* ------------------ */
/* sub-structures     */
/* ------------------ */

typedef struct
{
BYTE    Network[4];
BYTE    Node[6];
} INTNETADDR;
```

```
typedef struct
{
INTNETADDR   INA;
BYTE         ImmediateAddr[6];
} LOCALTARGET;

typedef struct
{
INTNETADDR   INA;
BYTE         Socket[2];
} NETADDR;

typedef struct
{
void far  *BufAddr;
unsigned  Len;
} ECBFRAGMENT;

/* ------------------- */
/* IPX PACKET HEADER */
/* ------------------- */

typedef struct
{
unsigned  Chksum;
unsigned  Len;                /* hi/lo */
BYTE      TransportCtrl;
BYTE      PacketType;
NETADDR   DestAddr;
NETADDR   SrceAddr;
} IPXHEADER;

/* ------------------- */
/* EVENT CONTROL BLOCK */
/* ------------------- */

typedef struct
{
void far      *LinkAddr;
void far      (*ESR)(void);
BYTE          InUse;
BYTE          CompletionCode;
BYTE          Socket[2];
BYTE          IPXWorkspace[4];
BYTE          DriverWorkspace[12];
BYTE          ImmediateAddr[6];
unsigned      FragmentCnt;
ECBFRAGMENT   Fragment[2];
} EVENTCONTROLBLOCK;

/* ================== */
/* ================== */
/* function prototypes */
/* ================== */
/* ================== */

/* ------------------- */
/* low-level IPX API functions */
/* ------------------- */

unsigned IPXInstalled( void );
unsigned IPXSocketOpen( BYTE *socket, BYTE longevity );
void     IPXSocketClose( BYTE *socket );
```

```
void      IPXSendPacket( EVENTCONTROLBLOCK *ecb );
void      IPXListenForPacket( EVENTCONTROLBLOCK *ecb );
void      IPXGetInternetworkAddress( INTNETADDR *ina );
void      IPXRelinquishControl( void );
void      IPXDisconnectFromTarget( NETADDR *naddr );
unsigned  IPXCancelEvent( EVENTCONTROLBLOCK *ecb );
void      IPXScheduleEvent( EVENTCONTROLBLOCK *ecb, unsigned delay );
unsigned  IPXGetIntervalMarker( void );
unsigned  IPXGetLocalTarget( LOCALTARGET *lt );

/* ---------------------------- */
/* Internal Utility functions */
/* ---------------------------- */

int  IPXTimeOutRT( EVENTCONTROLBLOCK *ecb,
                   time_t delay,
                   void (*RTfunc)(void) );
int  xatoxb( BYTE *d, char *s );
void xantoxl( BYTE *d, char *s, unsigned len );
void ShowECB( EVENTCONTROLBLOCK *ecb );
void ShowIPXPacket( IPXHEADER *ipxhdr );

/* ---------------------------- */
/* High-Level Application IPX Interface functions */
/* ---------------------------- */

unsigned IPXReceiveRT( long *BfrLen,
                       BYTE *FileOrBfr,
                       char *SocketStr,
                       void (*RTfunc)(void) );

unsigned IPXSendRT( long     *BfrLen,
                    BYTE     *FileOrBfr,
                    char     *SocketStr,
                    NETADDR  *Destination,
                    void     (*RTfunc)(void) );

/* ---------------------------- */
/* ---------------------------- */
/* globals                      */
/* ---------------------------- */
/* ---------------------------- */

/*
NOTE: These globals default to the constants listed within IPX.H,
      but can be changed by the application that employs them.
*/

extern unsigned IPXerrno;  // latest IPX function result code
extern time_t   IPXto;     // # secs to wait for IPX command completion
extern unsigned IPXtries;  // # times to try RX/TX in IPXSend/Receive()
extern unsigned IPXdatalen;// size of data buffer in IPX packet

/* ************************************************************************ */
/* ************************************************************************ */

#endif

/* *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** */
// EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF
/* ************************************************************************ */
```

LAMPIRAN B

LISTING RUTIN PUSTAKA IPX, IPX.C

```
/*****************************************************************/
/*****************************************************************/
/* IPX LIBRARY ROUTINES                                          */
/*****************************************************************/
/*****************************************************************/

/* ------- */
/* ------- */
/* headers */
/* ------- */
/* ------- */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <dos.h>
#include <time.h>
#include <string.h>
#include <conio.h>
#include <io.h>
#include <fcntl.h>
#include <sys\stat.h>

#include "ipx.h"

/* ---------------- */
/* ---------------- */
/* global variables */
/* ---------------- */
/* ---------------- */

unsigned  IPXerrno;                    // latest IPX error
time_t    IPXLo = IPX_TIMEOUT;         // secs to wait for IPX command
unsigned  IPXtries = IPX_TRIES;        // # times to retry RX/TX
unsigned  IPXdatalen = MINIPXDATALEN; // size of IPX packet data buffer

/* ---------------- */
/* ---- I N T E R N A L  U S E ---- */
/* ---------------- */

static void (*IPXAPI)(void);
```

```
/*****************************************************************/
/*****************************************************************/
/*  IPX LOW-LEVEL FUNCTIONS */
/*****************************************************************/
/*****************************************************************/

unsigned IPXInitialize( void )
{
union REGS      regs;
struct SREGS    sregs;

regs.x.ax = IPX_MULTIPLEX;
intdosx( DOS_MULTIPLEX_VECTOR, &regs, &regs, &sregs );
(void *) *IPXAPI = MK_FP( sregs.es, regs.x.di );
return( regs.h.al );
}

/*****************************************************************/
/*****************************************************************/

unsigned IPXSocketOpen( BYTE *socket, BYTE longevity )
{
_DL = socket[0];
_DH = socket[1];
_AL = longevity;
_BX = SOCKET_OPEN;
IPXAPI();
socket[0] = _DL;
socket[1] = _DH;
_AH = 0;
return( _AX );
}

/*****************************************************************/
/*****************************************************************/

void IPXSocketClose( BYTE *socket )
{
_DL = socket[0];
_DH = socket[1];
_BX = SOCKET_CLOSE;
IPXAPI();
return;
}

/*****************************************************************/
/*****************************************************************/

unsigned IPXGetLocalTarget( LOCALTARGET *lt )
{
_ES = FP_SEG( (void far *)lt );
_SI = FP_OFF( (void far *)lt->NA.Network );
_DI = FP_OFF( (void far *)lt->ImmediateAddr );
_BX = GET_LOCAL_TARGET;
IPXAPI();
_AH = 0;

return( _AX );
}

/*****************************************************************/
/*****************************************************************/
```

```
/* returns TRUE if characters in string are legal hex digits */

int xatoxb( BYTE *d, char *s )
{
while( *s )
  {
  if ( !isxdigit( *s ) )
     {
     return( !TRUE );
     }
  *d = (isdigit(*s) ? *s++ - '0' : toupper(*s++) - '7') << 4;
  if ( !isxdigit( *s ) )
     {
     return( !TRUE );
     }
  *d++ += (isdigit(*s) ? *s++ - '0' : toupper(*s++) - '7');
  }
return( TRUE );
}

/* ================================================== */

/* allows/ignores non-hex ASCII delimiters in string */

void xantoxb( BYTE *d, char *s, unsigned len )
{
while( len-- )
  {
  while( !isxdigit( *s ) )
    s++;
  *d = (isdigit(*s) ? *s++ - '0' : toupper(*s++) - '7') << 4;
  while( !isxdigit( *s ) )
    s++;
  *d++ += (isdigit(*s) ? *s++ - '0' : toupper(*s++) - '7');
  }
return;
}

/*************************************************************************/
/*************************************************************************/

void ShowECB( EVENTCONTROLBLOCK *ecb )
{
unsigned i;

printf( "\n" );
printf( "LINK ADDR: %04X:%04X\n",
        (unsigned)((long)ecb->LinkAddr / SEGSIZE),
        (unsigned)((long)ecb->LinkAddr % SEGSIZE) );

printf( "      ESR: %04X:%04X\n",
        (unsigned)((long)ecb->ESR / SEGSIZE),
        (unsigned)((long)ecb->ESR % SEGSIZE) );

printf( "   IN USE: %02X\n",
        ecb->InUse );

printf( "COMP CODE: %02X\n",
        ecb->CompletionCode );

printf("   SOCKET: %02X %02X\n",
        ecb->Socket[0],
        ecb->Socket[1] );
```

```
/*                                                                    */
/* check that IPX API has been installed or attempt to install it */
/* ------------------------------                                     */

if (
    !(IPXAPI)
      &&
    (IPXInstalled() != IPX_PRESENT)
   )
   {
   return( IPX_NOT_PRESENT );
   }

/* ------------------------------------------ */
/* open socket for peer-to-peer communications */
/* ------------------------------------------ */

if (
     (strlen( SocketStr ) != (sizeof(Socket)*2)) // 2 ASCII chars per
byte
      ||
     !(xatoxb( Socket, SocketStr ))
   )
   {
   return( BAD_SOCKET_PARAM );
   }

IPXerrno = IPXSocketOpen( Socket, SOCKET_SHORT_LIVED );
if ( IPXerrno != IPX_SUCCESS )
   {
   if ( IPXerrno == SOCKET_ALREADY_OPEN )
      /*
         sure, we could just go ahead,
         but the socket really shouldn't have been opened at this point,
         and until this library has been thoroughly shaken out,
         (hopefully) ALL discrepancies will be caught & reported!
      */
      {
      IPXSocketClose( Socket );
      }
   return( BAD_SOCKET_OPEN );
   }

/* ------------------------ */
/* set the receive mode */
/* ------------------------ */

fileflg = TRUE;
if ( *BfrLen )
   {
   RXptr = FileOrBfr;
   fileflg = !TRUE;
   }

/* ------------------------------ */
/* attempt to open RX file */
/* ------------------------------ */

if ( fileflg )
   {
   fhandle = open( FileOrBfr,
                O_RDWR | O_CREAT | O_TRUNC | O_BINARY,
                   S_IREAD | S_IWRITE );
```

```
    if ( fhandle == FILE_PROBLEM )
       {
       printf( "Can't open %s\n" );
       exit( BAD_FILE_OPEN );
       }
    }

/* ------------------------ */
/* allocate necessary memory */
/* ------------------------ */

pIPXbfrRX = (BYTE *)calloc( 1, SEQLENIPXdatalen );
pIPXbfrTX = (BYTE *)calloc( 1, SEQLENIPXdatalen );
pIPXpktRX = (IPXHEADER *)calloc( 1, sizeof(IPXHEADER) );
pIPXpktTX = (IPXHEADER *)calloc( 1, sizeof(IPXHEADER) );
pIPXecbRX = (EVENTCONTROLBLOCK *)calloc( 1, sizeof(EVENTCONTROLBLOCK)
);
pIPXecbTX = (EVENTCONTROLBLOCK *)calloc( 1, sizeof(EVENTCONTROLBLOCK)
);
pLocalTarget = (LOCALTARGET *)calloc( 1, sizeof(LOCALTARGET) );

if (
     pIPXbfrRX == NULL ||
     pIPXbfrTX == NULL ||
     pIPXpktRX == NULL ||
     pIPXpktTX == NULL ||
     pIPXecbRX == NULL ||
     pIPXecbTX == NULL ||
     pLocalTarget == NULL
   )
   {
   IPXSocketClose( Socket );
   *Bfrlen = 0;
   if ( fileflg )
      {
      close( fhandle );
            unlink( FileOrBfr );
      }
   if ( pIPXbfrRX )
      free( pIPXbfrRX );
   if ( pIPXbfrTX )
      free( pIPXbfrTX );
   if ( pIPXpktRX )
      free( pIPXpktRX );
   if ( pIPXpktTX )
      free( pIPXpktTX );
   if ( pIPXecbRX )
      free( pIPXecbRX );
   if ( pIPXecbTX )
      free( pIPXecbTX );
   if ( pLocalTarget )
      free( pLocalTarget );
   return( BAD_MEMORY_ALLOC );
   }

/* ------------------------ */
/* initialize ECBs/IPX packets */
/* ------------------------ */

memcpy( pIPXecbRX->Socket, Socket, sizeof(pIPXecbRX->Socket) );
pIPXpktRX->PacketType          = IPX_PACKET_TYPE;
pIPXecbRX->FragmentCount       = 2;
pIPXecbRX->Fragment[0].BufAddr = (void *)pIPXpktRX,
```

```
pIPXecbRX->Fragment[0].Len      = sizeof(IPXHEADER);
pIPXecbRX->Fragment[1].BufAddr  = (void *)pIPXbfrRX;
pIPXecbRX->Fragment[1].Len      = SEQLEN+IPXdatalen;

memcpy( pIPXecbTX->Socket, Socket, sizeof(pIPXecbTX->Socket) );
pIPXpktTX->PacketType           = IPX_PACKET_TYPE;
pIPXecbTX->FragmentCnt          = 2;
pIPXecbTX->Fragment[0].BufAddr  = (void *)pIPXpktTX;
pIPXecbTX->Fragment[0].Len      = sizeof(IPXHEADER);
pIPXecbTX->Fragment[1].BufAddr  = (void *)pIPXbfrTX;
pIPXecbTX->Fragment[1].Len      = SEQLEN+strlen( ACKSTR );
strcpy( pIPXbfrTX+SEQLEN, ACKSTR );

/* ========================== */
/* DATA RX/ACK TX LOOP */
/* ========================== */

expected_seq = FIRST_SEQ_NUM;
TXflg = TRUE;
retries = 0;
totRX = 0;
while ( TRUE )
  {

  /* if user decides to break it off, then quit */
  if ( USERBREAK )
     {
     rc = USER_TERMINATED;
     break;
     }

  /* issue LISTEN for pending RX of data */
  IPXListenForPacket( pIPXecbRX );

  /* ------------------------------- */
  /* issue SEND to ACK previous RX */
  /* ------------------------------- */

  if ( TXflg )
     {

     /* RX source address = TX destination address */
     memcpy( &pIPXpktTX->DestAddr, &TXna, sizeof(NETADDR) );

     /* must also fill immediate address of ECB */
     memcpy( pLocalTarget, &TXna, sizeof(NETADDR) );

     IPXerrno = IPXGetLocalTarget( pLocalTarget );
     if ( IPXerrno != IPX_SUCCESS )
        {
        IPXCancelEvent( pIPXecbRX );
        rc = BAD_LOCAL_TARGET;
        break;
        }

     memcpy( pIPXecbTX->ImmediateAddr,
             pLocalTarget->ImmediateAddr,
             sizeof(pIPXecbTX->ImmediateAddr) );

     /* retain RX seq # for ACK */
     *pIPXbfrTX = *pIPXbfrRX;
```

```
    /* send the ACK & check result of TX */
    IPXSendPacket( pIPXecbTX );
    IPXTimeOutRT( pIPXecbTX, !RTto, RTfunc );
    IPXerrno = pIPXecbTX->CompletionCode;
    if ( IPXerrno != IPX_SUCCESS )
     {
     IPXCancelEvent( pIPXecbRX );
     rc = BAD_TX_TRY;
     break;
     }

    /* ------------------------ */
    /* TX was successfully sent */
    /* ------------------------ */

    /* if the previous RX data that we are now ACKing */
    /* did not fill the RX buffer, consider it EOT    */
    if ( lenRX < IPXdatalen )
     {
     IPXCancelEvent( pIPXecbRX );
     rc = GOOD_IPX_RETURN;
     break;
     }

    } // end of if ( TXflg )

/* ---------------- */
/* monitor LISTEN */
/* ---------------- */

/* monitor result of pending LISTEN */
IPXTimeOutRT( pIPXecbRX, IPXto, RTfunc );
IPXerrno = pIPXecbRX->CompletionCode;

/* ------------------ */
/* RX was successful */
/* ------------------ */

if ( IPXerrno == IPX_SUCCESS )
   {

   /* if first RX then save Sender's address */
   if ( !totRX )
    {
    memcpy( &TXna, &pIPXpktRX->SrceAddr, sizeof(NETADDR) );
    }

   /* ignore all but latest Sender */
   if ( memcmp( &TXna, &pIPXpktRX->SrceAddr, sizeof(NETADDR) ) )
    {
    TXflg = !TRUE;
    continue;
    }

   /* check sequence number */
   if (expected_seq != *pIPXbfrRX )
    {
    rc = BAD_RX_SEQ_NUM;
    break;
    }

   /* if any data was received, then save it */
   lenRX = HILOSWAP( pIPXpktRX->len ) - (sizeof(IPXHEADER)+SEQNUM);
```

```
    if ( lenRX )
     {

     /* data to be written to file */
     if ( fileflg )
        {
        rc = write( fhandle, pIPXbfrRX+SEQLEN, lenRX );
        if ( lenRX != rc )
           {
           rc = BAD_FILE_WRITE;
           break;
           }
        }

     /* data to be copied into buffer */
     if ( !fileflg )
        {
        totRX += lenRX;
        if ( *BfrLen < totRX )
           {
           rc = BAD_RX_BFR_LEN;
           break;
           }
        memcpy( RXptr, pIPXbfrRX+SEQLEN, lenRX );
        RXptr += lenRX;
        }

     } // end of if ( lenRX )

    /* ACK is now owed to sender; clear retry cnts & bump seq # */
    TXflg = TRUE;
    retries = 0;
    expected_seq++;

    } // end of if ( IPXerrno == IPX_SUCCESS )

  /* ------------------------ */
  /* RX was NOT successful! */
  /* ------------------------ */

  if ( IPXerrno != IPX_SUCCESS )
     {

     /* if maximum retries have been exhausted, then quit */
     if ( retries++ >= IPXtries )
      {
      rc = BAD_RETRIES;
      break;
      }

     } // end of if ( IPXerrno != IPX_SUCCESS )

  } // end of while ( TRUE )

/* ------------------ */
/* bring it on home */
/* ------------------ */

IPXSocketClose( Socket );

/* if file mode, close it & delete it if something went wrong */
if ( fileflg )
   {
```

```
   close( fhandle );
   if ( rc != GOOD_IPX_RETURN )
      {
            unlink( FileOrBfr );
      }
   }

/* if buffer mode, set number of bytes read into buffer */
if ( !fileflg )
   {
   *BfrLen = totRX;
   if ( rc != GOOD_IPX_RETURN )
      {
      *BfrLen = 0;
      }
   }

/* free that mem! */
free( pIPXbfrRX );
free( pIPXbfrTX );
free( pIPXpktRX );
free( pIPXpktTX );
free( pIPXecbRX );
free( pIPXecbTX );
free( pLocalTarget );

return( rc );
}

/*********************************************************************/
/*********************************************************************/

unsigned IPXSendRT( long      *BfrLen,
                    BYTE      *FileOrBfr,
                    char      *SocketStr,
                    NETADDR   *Destination,
                    void      (*RTfunc)(void) )

{
int       fhandle;
unsigned  rc, lenTX, retries;
BYTE      *TXptr;
int       retryTXflg, fileflg;
BYTE      Socket[2];
BYTE      expected_seq;

BYTE                  *pIPXbfrRX, *pIPXbfrTX;
IPXHEADER             *pIPXpktRX, *pIPXpktTX;
EVENTCONTROLBLOCK     *pIPXecbRX, *pIPXecbTX;
LOCALTARGET           *pLocalTarget;

/* ------------------------                                          */
/* check that IPX API has been installed or attempt to install */
/* ------------------------                                          */

if (
    !(IPXAPI)
      &&
    (IPXInstalled() != IPX_PRESENT)
   )
   {
   return( IPX_NOT_PRESENT );
   }
```

```
/* ------------------------------------------------------------ */
/* open socket for peer-to-peer communications                  */
/* ------------------------------------------------------------ */

if (
    (strlen( SocketStr ) != (sizeof(Socket)*2))
     ||
    !(xatoxb( Socket, SocketStr ))
   )
   {
   return( BAD_SOCKET_PARAM );
   }

IPXerrno = IPXSocketOpen( Socket, SOCKET_SHORT_LIVED );
if ( IPXerrno != IPX_SUCCESS )
   {
   if ( IPXerrno == SOCKET_ALREADY_OPEN )
      {
      IPXSocketClose( Socket );
      }
   return( BAD_SOCKET_OPEN );
   }

/* ------------------------------------------------------------ */
/* set the transfer mode                                        */
/* ------------------------------------------------------------ */

fileflg = TRUE;
if ( *BfrLen )
   {
   TXptr = FileOrBfr;
   fileflg = !TRUE;
   }

/* ------------------------------------------------------------ */
/* check that TX file is available                              */
/* ------------------------------------------------------------ */

if ( fileflg )
   {
      fhandle = open( FileOrBfr, O_RDONLY | O_BINARY );
   if ( fhandle == FILE_PROBLEM )
      {
      IPXSocketClose( Socket );
      return( BAD_FILE_OPEN );
      }
   }

/* ------------------------------------------------------------ */
/* allocate necessary memory                                    */
/* ------------------------------------------------------------ */

pIPXbfrRX = (BYTE *)calloc( 1, SEQNUMRXdatalen );
pIPXbfrTX = (BYTE *)calloc( 1, SEQNUMRXdatalen );
pIPXpktRX = (IPXHEADER *)calloc( 1, sizeof(IPXHEADER) );
pIPXpktTX = (IPXHEADER *)calloc( 1, sizeof(IPXHEADER) );
pIPXecbRX = (EVENTCONTROLBLOCK *)calloc( 1, sizeof(EVENTCONTROLBLOCK)
);
pIPXecbTX = (EVENTCONTROLBLOCK *)calloc( 1, sizeof(EVENTCONTROLBLOCK)
);
pLocalTarget = (LOCALTARGET *)calloc( 1, sizeof(LOCALTARGET) );
```

```
    pIPXbfrRX == NULL ||
    pIPXbfrTX == NULL ||
    pIPXpktRX == NULL ||
    pIPXpktTX == NULL ||
    pIPXecbRX == NULL ||
    pIPXecbTX == NULL ||
    pLocalTarget == NULL
  )
  {
  IPXSocketClose( Socket );
  if ( fileflg )
     close( fhandle );
  if ( pIPXbfrRX )
     free( pIPXbfrRX );
  if ( pIPXbfrTX )
     free( pIPXbfrTX );
  if ( pIPXpktRX )
     free( pIPXpktRX );
  if ( pIPXpktTX )
     free( pIPXpktTX );
  if ( pIPXecbRX )
     free( pIPXecbRX );
  if ( pIPXecbTX )
     free( pIPXecbTX );
  if ( pLocalTarget )
     free( pLocalTarget );
  return( BAD_MEMORY_ALLOC );
  }

/* ---------------------------- */
/* initialize ECBs /IPX packets */
/* ---------------------------- */

memcpy( pIPXecbRX->Socket, Socket, sizeof(pIPXecbRX->Socket) );
pIPXpktRX->PacketType          = IPX_PACKET_TYPE;
pIPXecbRX->FragmentCnt         = 2;
pIPXecbRX->Fragment[0].BufAddr = (void *)pIPXpktRX;
pIPXecbRX->Fragment[0].Len     = sizeof(IPXHEADER);
pIPXecbRX->Fragment[1].BufAddr = (void *)pIPXbfrRX;
pIPXecbRX->Fragment[1].Len     = SECTORSIZE*IPXdataLen;

memcpy( pIPXecbTX->Socket, Socket, sizeof(pIPXecbTX->Socket) );
pIPXpktTX->PacketType          = IPX_PACKET_TYPE;
pIPXecbTX->FragmentCnt         = 2;
pIPXecbTX->Fragment[0].BufAddr = (void *)pIPXpktTX;
pIPXecbTX->Fragment[0].Len     = sizeof(IPXHEADER);
pIPXecbTX->Fragment[1].BufAddr = (void *)pIPXbfrTX;

memcpy( &pIPXpktTX->DestAddr, Destination, sizeof(NETADDR) );
memcpy( pLocalTarget, Destination, sizeof(INTERADDR) );
IPXerrno = IPXGetLocalTarget( pLocalTarget );
if ( IPXerrno != IPX_SUCCESS )
  {
  IPXSocketClose( Socket );
  if ( fileflg )
     close( fhandle );
  free( pIPXbfrRX );
  free( pIPXbfrTX );
  free( pIPXpktRX );
  free( pIPXpktTX );
  free( pIPXecbRX );
  free( pIPXecbTX);
```

```
    free( pLocalTarget );
    return( BAD_LOCAL_TARGET );
    }

memcpy( pIPXecbTX->ImmediateAddr,
        pLocalTarget->ImmediateAddr,
        sizeof(pIPXecbTX->ImmediateAddr) );

/* ================== */
/* DATA TX/ACK RX LOOP */
/* ================== */

expected_seq = FIRST_SEQ_NUM;
retryTXflg = !TRUE;
retries = 0;
while( TRUE )
  {

  /* if user chose to break things off, then quit */
  if ( USERBREAK )
     {
     rc = USER_TERMINATED;
     break;
     }

  /* get ready for expected ACK of (hopefully) TX */
  IPXListenForPacket( pIPXecbRX );

  /* if TX retry is not active, prepare TX buffer */
  if ( !retryTXflg )
     {

     /* fill TX buffer with data from file */
     if ( fileflg )
        {
        lenTX = read( fhandle, pIPXbfrTX+SEQ_LEN, IPXdatalen );
        if ( lenTX == (unsigned)FILE_PROBLEM )
           {
           IPXCancelEvent( pIPXecbRX );
           rc = BAD_FILE_READ;
                break;
                }
           }

        /* fill TX buffer with data from buffer */
        if ( !fileflg )
              {
              lenTX = ((*BfrLen < IPXdatalen) ? *BfrLen : IPXdatalen);
        memcpy( pIPXbfrTX+SEQLEN, TXptr, lenTX );
        TXptr += lenTX;
        *BfrLen -= IPXdatalen;
        }

     /* set sequence number & buffer length */
     *pIPXbfrTX = expected_seq;
     pIPXecbTX->Fragment[1].Len = SEQLEN+lenTX;

     } // end of if ( !retryTXflg )

  /* send the data & check result of TX */
  IPXSendPacket( pIPXecbTX );
  IPXTimeOutRT( pIPXecbTX, IRTO, RTFuzz );
  IPXerrno = pIPXecbTX->CompletionCode;
```

```
    {
    /* ------------------------------------ */
    /* RX (ACK to previous TX) is pending */
    /* ------------------------------------ */

    /* wait for RX to complete, then check result */
    IPXTimeOutRT( pIPXecbRX, IPXto, RTfunc );
    IPXerrno = pIPXecbRX->CompletionCode;

    /* ------------------------------------ */
    /* ACK was received successfully */
    /* ------------------------------------ */

    if ( IPXerrno == IPX_SUCCESS )
       {
       /* check that RX was from expected Destination */
       if ( memcmp( Destination, &pIPXpktRX->SrcAddr, sizeof(NETWADDR) )
          {
          rc = BAD_RX_SRCE_ADDR;
          break;
          }

       /* check that TX/RX sequence is synch */
       if ( expected_seq != *pIPXbfrRX )
          {
          rc = BAD_RX_SEQ_NUM;
          break;
          }

       /* if last TX was EOT, then all done! */
       if ( lenTX < IPXdatalen )
          {
          rc = GOOD_IPX_RETURN;
          break;
          }

    /* otherwise, do next TX */
    expected_seq++;
    retries = 0;
    retryTXflg = !TRUE;

    } // end of if ( IPXerrno == IPX_SUCCESS )

/* ------------------------------------ */
/* no ACK was received */
/* ------------------------------------ */

if ( IPXerrno != IPX_SUCCESS )
   {
```

```
      /* if maximum retries have been exhausted, then quit */
      if ( retries++ >= IPXtries )
         {
         rc = BAD_RETRIES;
         break;
         }

      /* otherwise, try RX again */
      retryTXflg = TRUE;

      } // end of if ( IPXerrno != IPX_SUCCESS )

   } // end of while ( TRUE )

/* ------------------------------------ */
/* close socket and bring it on home */
/* ------------------------------------ */

IPXSocketClose( Socket );

if ( fileflg )
   close( fhandle );

free( pIPXbfrRX );
free( pIPXbfrTX );
free( pIPXpktRX );
free( pIPXpktTX );
free( pIPXecbRX );
free( pIPXecbTX );
free( pLocalTarget );

return( rc );
}

/*********************************************************************/
/*  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  */
/*********************************************************************/
```

LAMPIRAN C
LISTING PROGRAM

C.1 PROGRAM DETEKSI ALAMAT, IPXADD.C

```
/*****************************************************************/
/*                    I P X  addressing program                  */
/*****************************************************************/

#include <stdio.h>
#include "ipx.c"

/*****************************************************************/
/* main program for addressing                                   */
/*****************************************************************/

void main( void )
{
LOCALTARGET lt;

if ( IPXInstalled() != IPX_PRESENT )
  {
  printf( "\n\nIPX API NOT DETECTED!\n\n" );
  return;
  }

IPXGetInternetworkAddress( &lt.INA );
IPXGetLocalTarget( &lt );

printf( "\n" );
printf( "GET INTERNETWORK ADDRESS (IPX)\n" );
printf( "\n" );
printf( "Network Addr: %02X:%02X:%02X:%02X\n",
  lt.INA.Network[0], lt.INA.Network[1],
  lt.INA.Network[2], lt.INA.Network[3] );
printf("   Node Addr: %02X%02X%02X%02X%02X%02X\n",
  lt.INA.Node[0], lt.INA.Node[1],
  lt.INA.Node[2], lt.INA.Node[3],
  lt.INA.Node[4], lt.INA.Node[5] );
printf( "Local Target: %02X:%02X:%02X:%02X:%02X:%02X\n",
  lt.ImmediateAddr[0], lt.ImmediateAddr[1],
  lt.ImmediateAddr[2], lt.ImmediateAddr[3],
  lt.ImmediateAddr[4], lt.ImmediateAddr[5] );
return;
}
/*****************************************************************/
/* EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  */
/*****************************************************************/
```

C.2 PROGRAM KIRIM, IPXSND.C

```
/*****************************************************************/
/*****************************************************************/
/*                      I P X  file transfer program             */
/*****************************************************************/
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <mem.h>

#include "ipx.c"
#define  TXSOCKET "0000"

/*****************************************************************/
/*****************************************************************/
/* main program for sending a file                               */
/*****************************************************************/
/*****************************************************************/

int main( void )
{

unsigned rc;
NETADDR  netaddr;
long     len;
char     FileName;

// internetwork address of destination machine
xatoxb( netaddr.INA.Network, "00000000"      );
xatoxb( netaddr.INA.Node,    "FFFFFFFFFFFF" );
xatoxb( netaddr.Socket,      "0000"          );

/*****************************************************************/
/* sending a file                                                */
/*****************************************************************/

printf( "Enter file to be send: ");
scanf( "%24s", FileName );

printf( "\nATTEMPTING FILE SEND\n" );
len = 0;
rc = IPXSendRT( &len, &FileName, TXSOCKET, &netaddr, NULL );
if ( rc != GOOD_IPX_RETURN )
  {
  printf( "\n" );
  printf( "FUNCTION ERROR: %02X\n", rc );
  printf( "  IPXAPI ERROR: %02X\n", IPXerrno );
  return( -1 );
  }
printf( "\nSUCCESSFUL FILE SEND\n" );

return( 0 );
}

/*****************************************************************/
/* EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF  EOF */
/*****************************************************************/
```

C.3 PROGRAM TERIMA, IPXRCV.C

```
/* I P X  file receive program */

#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>

#include "ipx.c"
#define  RXSOCKET "0000"

/* main program for receiving a file */

int main( void )
{
BYTE      Socket[2];
unsigned  rc, i;
long      len;
char      FileName;

/* receiving a file */

printf( "Enter file name to be received: " );
scanf( "%24c", FileName );

printf( "\nATTEMPTING FILE RECEIPT\n" );

len = 0;
rc = IPXReceiveRT( &len, &FileName, RXSOCKET, NULL );
if ( rc != GOOD_IPX_RETURN )
  {
  printf( "\n" );
  printf( "FUNCTION ERROR: %02X\n", rc );
  printf( "  IPXAPI ERROR: %02X\n\n", IPXerrno );
  return( -1 );
  }

printf( "\nSUCCESSFUL FILE RECEIPT\n" );

return( 0 );
}

/* EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF */
```

LAMPIRAN D
DAFTAR DIAGRAM ALIR

D.1 Transmisi Per Bit

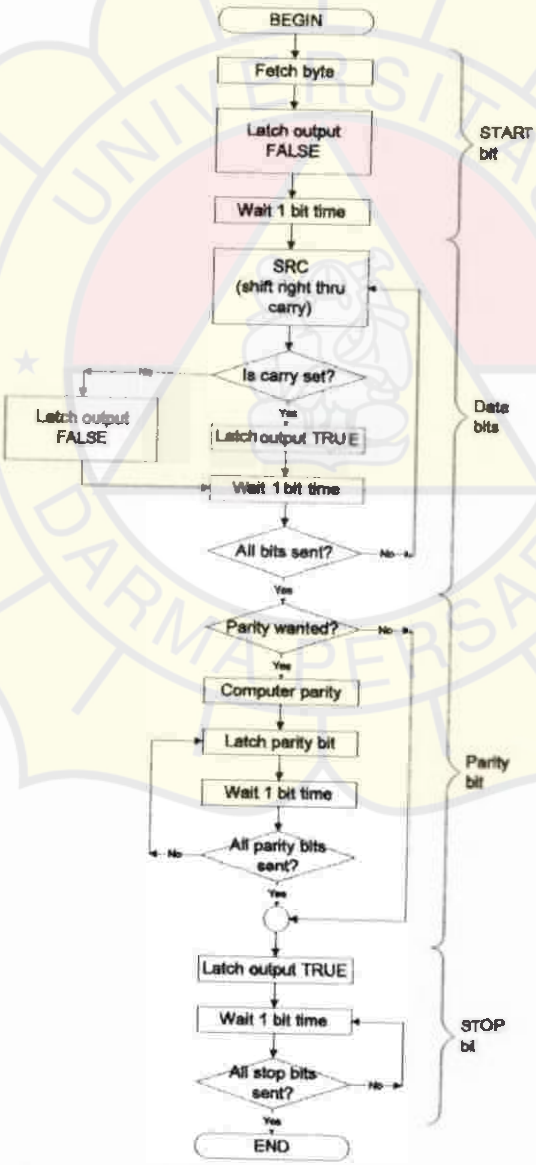


FIGURE 6.1. A hypothetical flow chart for transmitting a single byte in software. [CAM94/183]

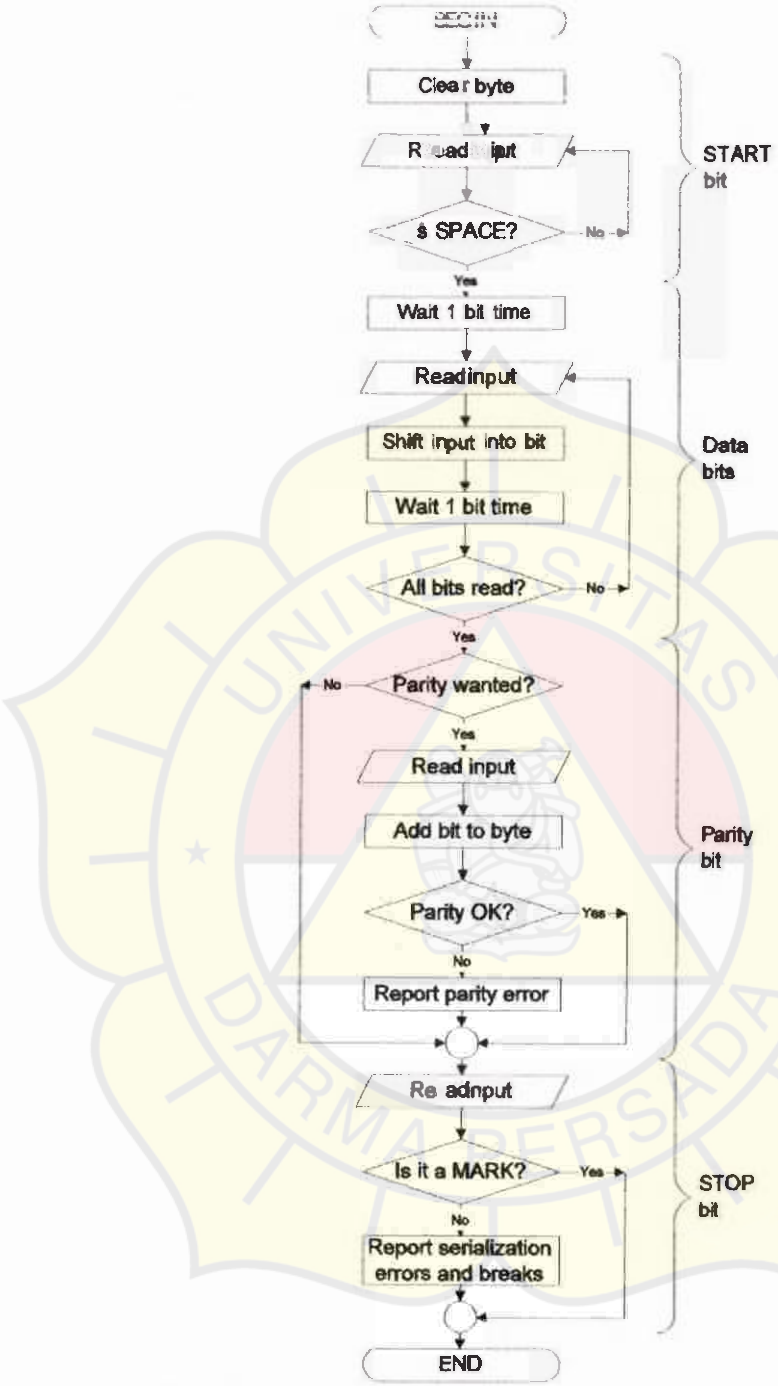
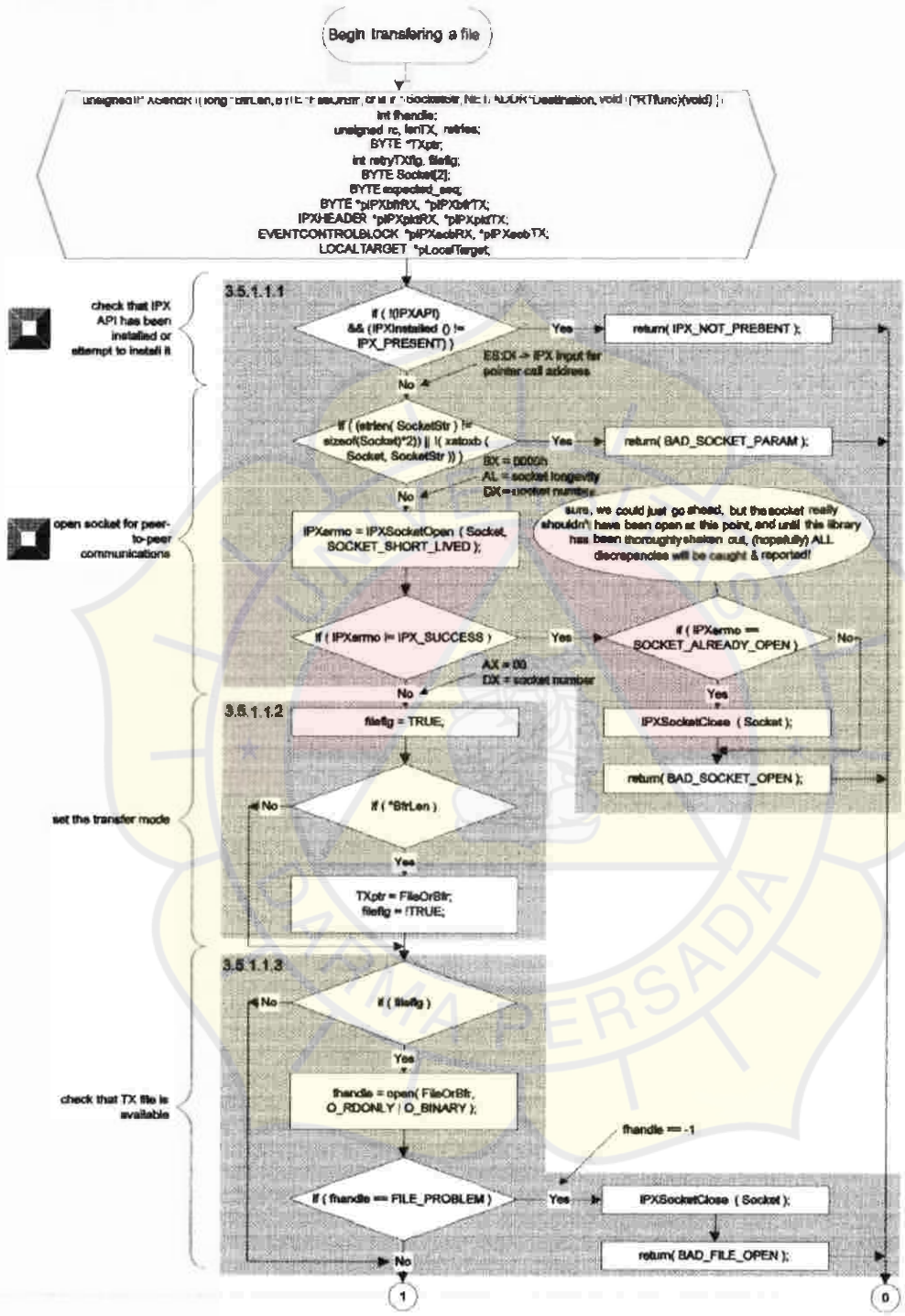
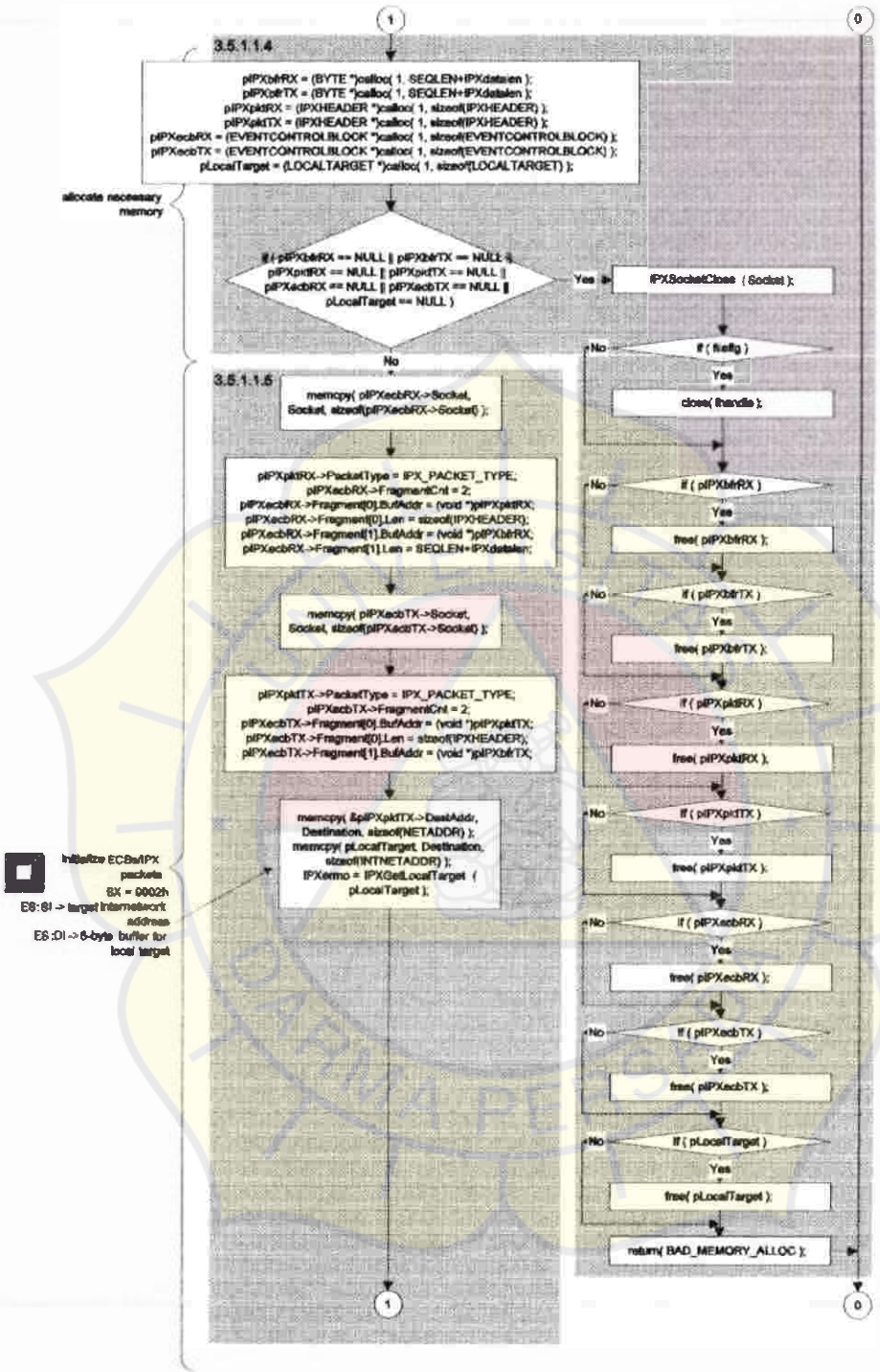
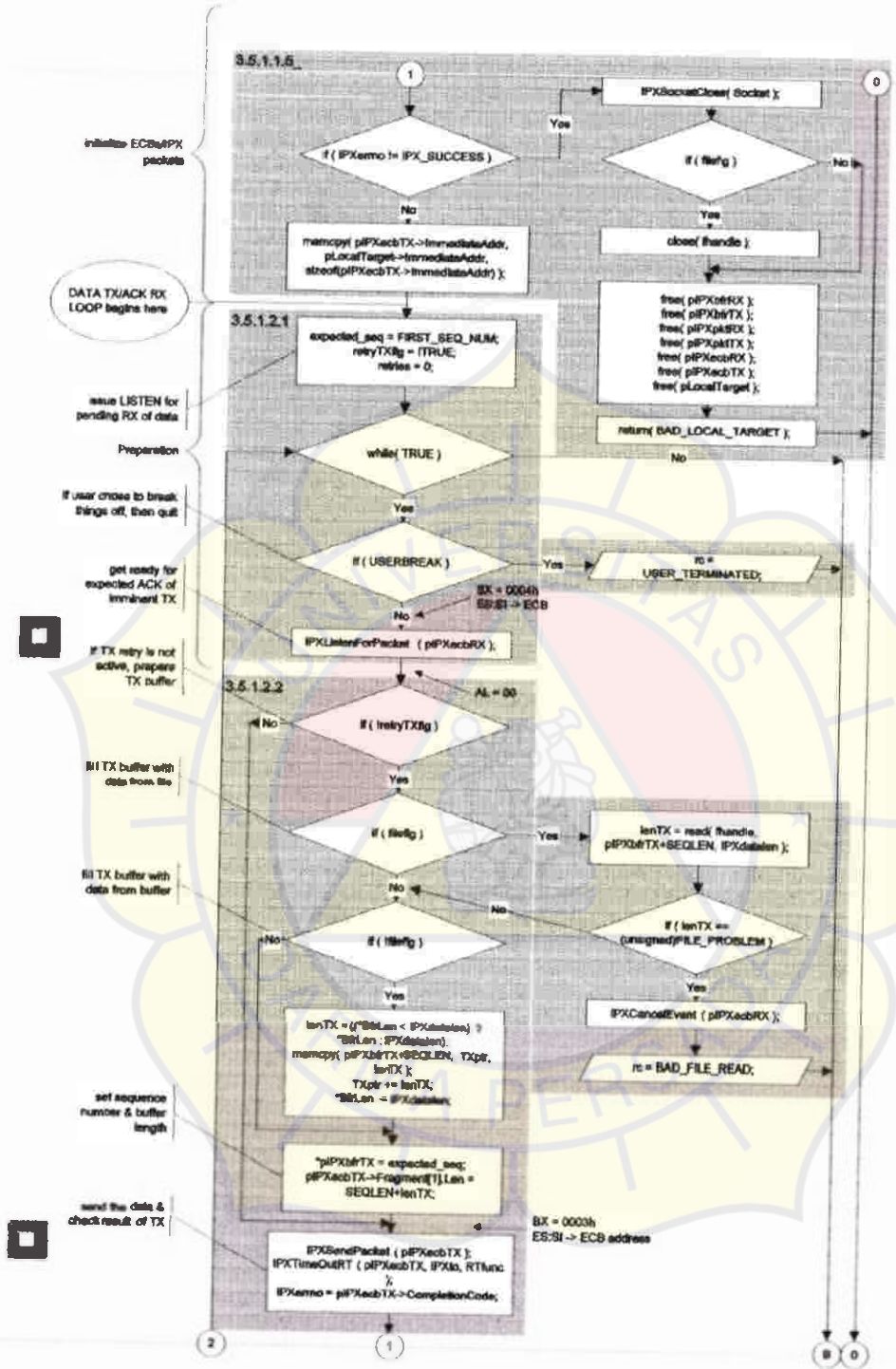


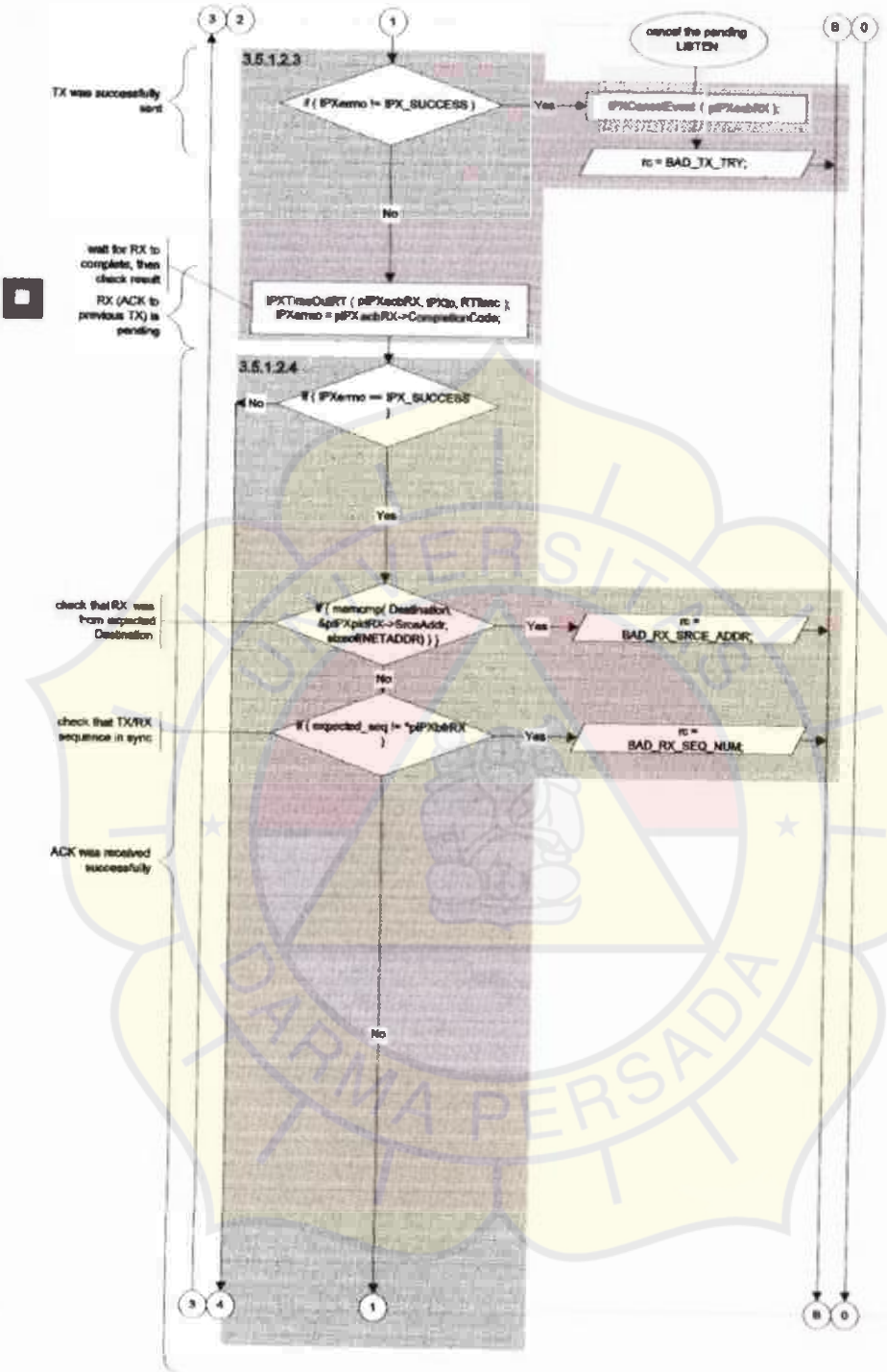
FIGURE 6.2. Flow chart for receiving a single byte in software. [CAM94/185]

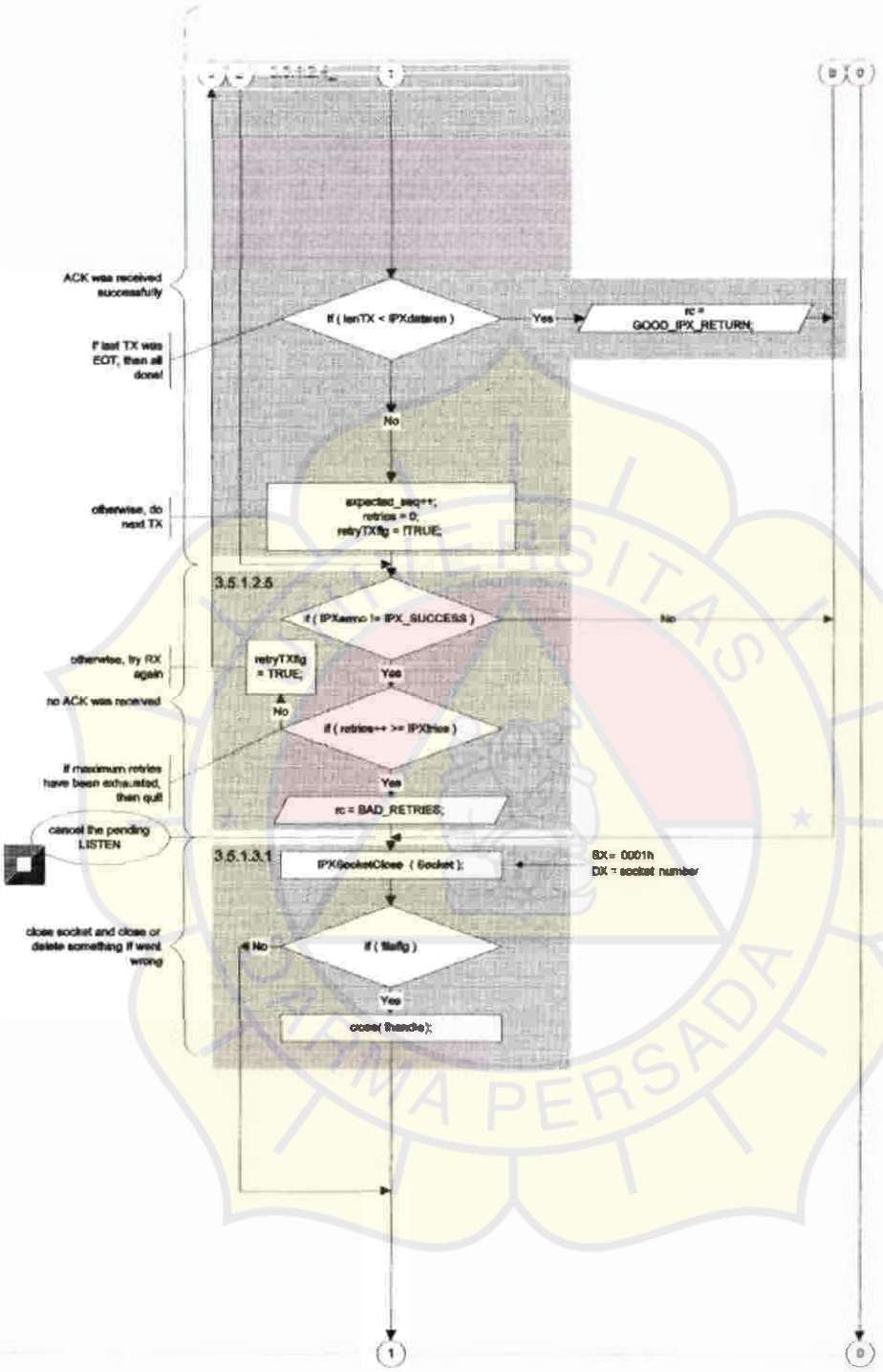
D.2 Transmisi IPX

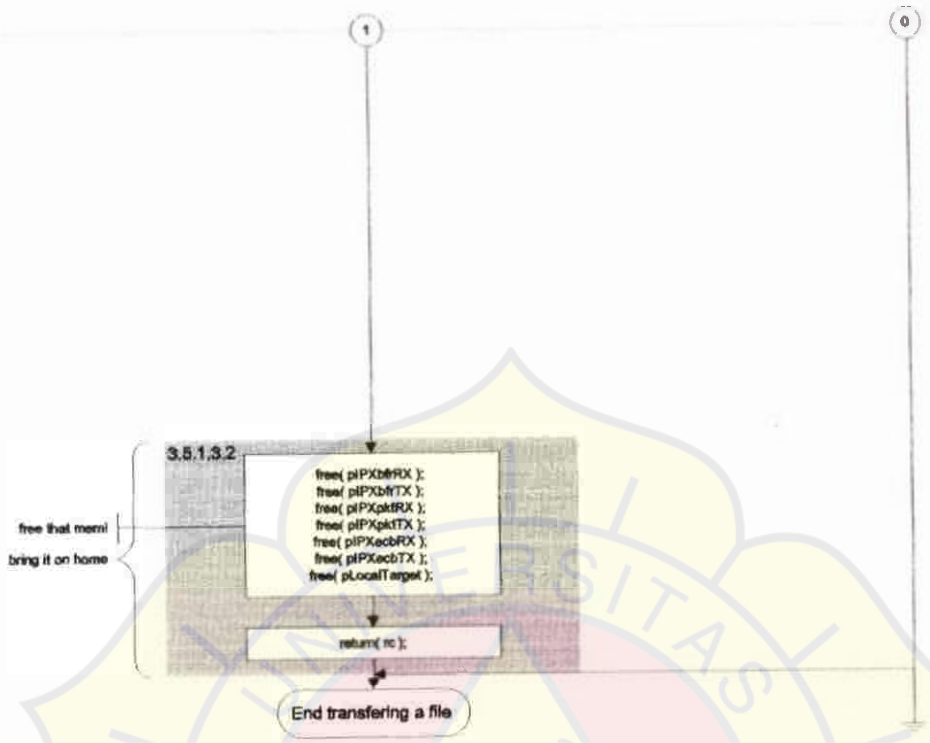


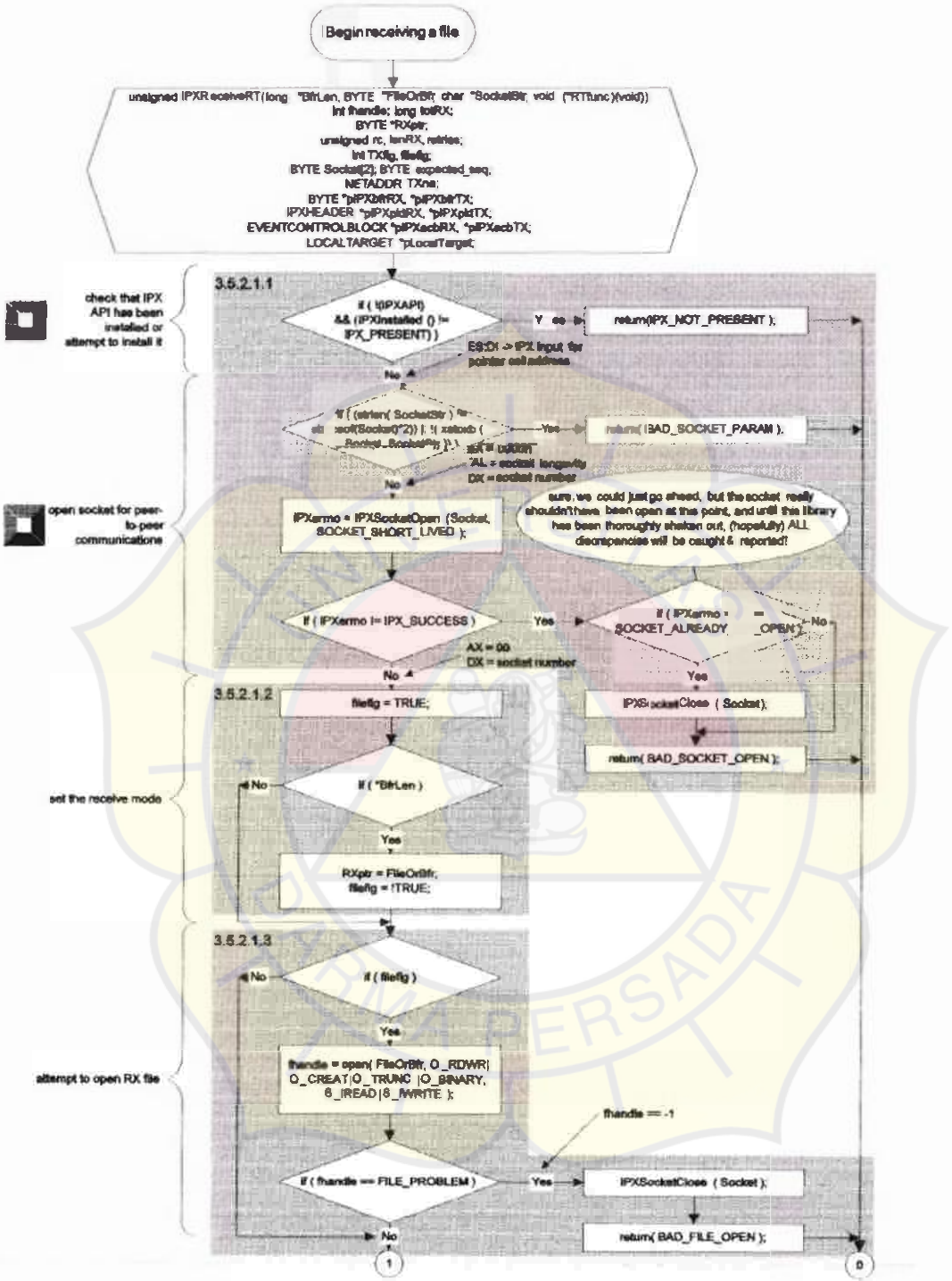

1
0
3.5.1.1.4
pIPXbfrRX = (BYTE *)calloc(1, SEQLEN+IPXdatalen);
pIPXbfrTX = (BYTE *)calloc(1, SEQLEN+IPXdatalen);
pIPXpktRX = (IPXHEADER *)calloc(1, sizeof(IPXHEADER));
pIPXpktTX = (IPXHEADER *)calloc(1, sizeof(IPXHEADER));
pIPXecbRX = (EVENTCONTROLBLOCK *)calloc(1, sizeof(EVENTCONTROLBLOCK));
pIPXecbTX = (EVENTCONTROLBLOCK *)calloc(1, sizeof(EVENTCONTROLBLOCK));
pLocalTarget = (LOCALTARGET *)calloc(1, sizeof(LOCALTARGET));
allocate necessary memory
if (pIPXbfrRX == NULL || pIPXbfrTX == NULL ||
pIPXpktRX == NULL || pIPXpktTX == NULL ||
pIPXecbRX == NULL || pIPXecbTX == NULL ||
pLocalTarget == NULL)
Yes
IPXSocketClose (Socket);
No
if (fIsBg)
Yes
close(fHandle);
No
if (pIPXbfrRX)
Yes
free(pIPXbfrRX);
No
if (pIPXbfrTX)
Yes
free(pIPXbfrTX);
No
if (pIPXpktRX)
Yes
free(pIPXpktRX);
No
if (pIPXpktTX)
Yes
free(pIPXpktTX);
No
if (pIPXecbRX)
Yes
free(pIPXecbRX);
No
if (pIPXecbTX)
Yes
free(pIPXecbTX);
No
if (pLocalTarget)
Yes
free(pLocalTarget);
return(BAD_MEMORY_ALLOC);
3.5.1.1.5
memcpy(pIPXecbRX->Socket,
Socket, sizeof(pIPXecbRX->Socket));
pIPXpktRX->PacketType = IPX_PACKET_TYPE;
pIPXecbRX->FragmentCnt = 2;
pIPXecbRX->Fragment[0].BufAddr = (void *)pIPXpktRX;
pIPXecbRX->Fragment[0].Len = sizeof(IPXHEADER);
pIPXecbRX->Fragment[1].BufAddr = (void *)pIPXbfrRX;
pIPXecbRX->Fragment[1].Len = SEQLEN+IPXdatalen;
memcpy(pIPXecbTX->Socket,
Socket, sizeof(pIPXecbTX->Socket));
pIPXpktTX->PacketType = IPX_PACKET_TYPE;
pIPXecbTX->FragmentCnt = 2;
pIPXecbTX->Fragment[0].BufAddr = (void *)pIPXpktTX;
pIPXecbTX->Fragment[0].Len = sizeof(IPXHEADER);
pIPXecbTX->Fragment[1].BufAddr = (void *)pIPXbfrTX;
memcpy(&pIPXpktTX->DestAddr,
Destination, sizeof(NETADDR));
memcpy(pLocalTarget, Destination,
sizeof(INTNETADDR));
IPXermo = IPXGetLocalTarget (
pLocalTarget);
Initialize ECB&IPX packets
BX = 0002h
ES:SI -> target Internetwork address
ES:DI -> 6-byte buffer for local target

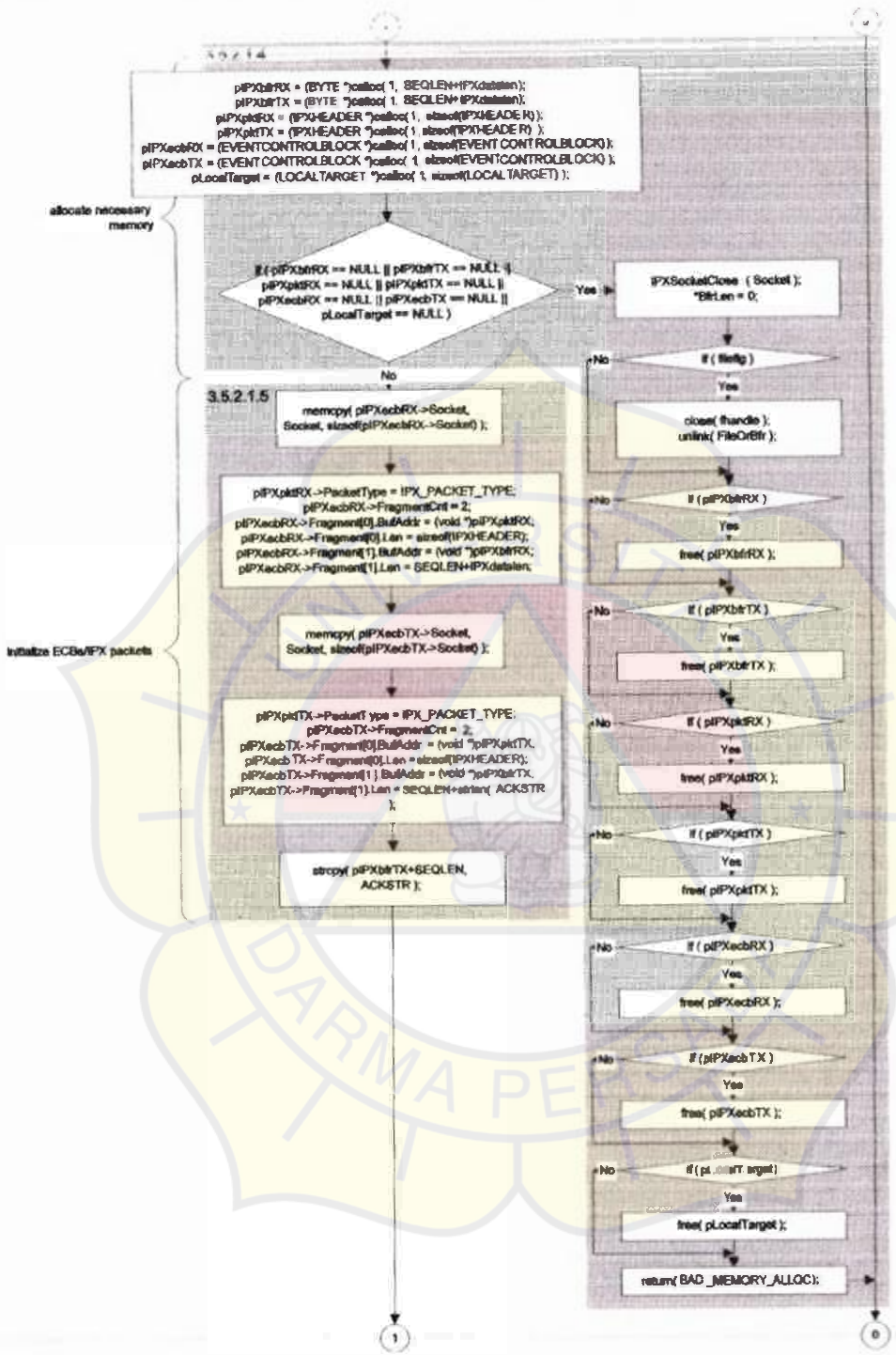

3.5.1.1.5
1
IPXSocketClose(Socket);
0
Yes
if (IPXerrno != IPX_SUCCESS)
if (fileFg)
No
No
Yes
memcpy(pIPXecbTX->ImmediateAddr, pLocalTarget->ImmediateAddr, sizeof(pIPXecbTX->ImmediateAddr));
close(fhandle);
free(pIPXbfrRX);
free(pIPXbfrTX);
free(pIPXpktRX);
free(pIPXpktTX);
free(pIPXecbRX);
free(pIPXecbTX);
free(pLocalTarget);
return(BAD_LOCAL_TARGET);
initialize ECBs/IPX packets
DATA TX/ACK RX LOOP begins here
3.5.1.2.1
expected_seq = FIRST_SEQ_NUM;
retryTXflg = !TRUE;
retries = 0;
issue LISTEN for pending RX of data
Preparation
while(TRUE)
No
If user chose to break things off, then quit
Yes
if (USERBREAK)
Yes
rc = USER_TERMINATED;
get ready for expected ACK of imminent TX
BX = 0004h
ES:SI -> ECB
No
IPXListenForPacket (pIPXecbRX);
If TX retry is not active, prepare TX buffer
3.5.1.2.2
AL = 00
No
if (!retryTXflg)
fill TX buffer with data from file
Yes
if (fileFg)
Yes
lenTX = read(fhandle, pIPXbfrTX+SEQLEN, IPXdatalen);
fill TX buffer with data from buffer
No
No
if (lenTX == (unsigned)FILE_PROBLEM)
No
if (!fileFg)
Yes
Yes
IPXCancelEvent (pIPXecbRX);
rc = BAD_FILE_READ;
lenTX = ((*BfrLen < IPXdatalen) ? *BfrLen : IPXdatalen);
memcpy(pIPXbfrTX+SEQLEN, TXptr, lenTX);
TXptr += lenTX;
*BfrLen = IPXdatalen;
set sequence number & buffer length
*pIPXbfrTX = expected_seq;
pIPXecbTX->Fragment[1].Len = SEQLEN+lenTX;
send the data & check result of TX
BX = 0003h
ES:SI -> ECB address
IPXSendPacket (pIPXecbTX);
IPXTimeOutRT (pIPXecbTX, IPXlo, RTfunc);
IPXerrno = pIPXecbTX->CompletionCode;
2
1
B
0

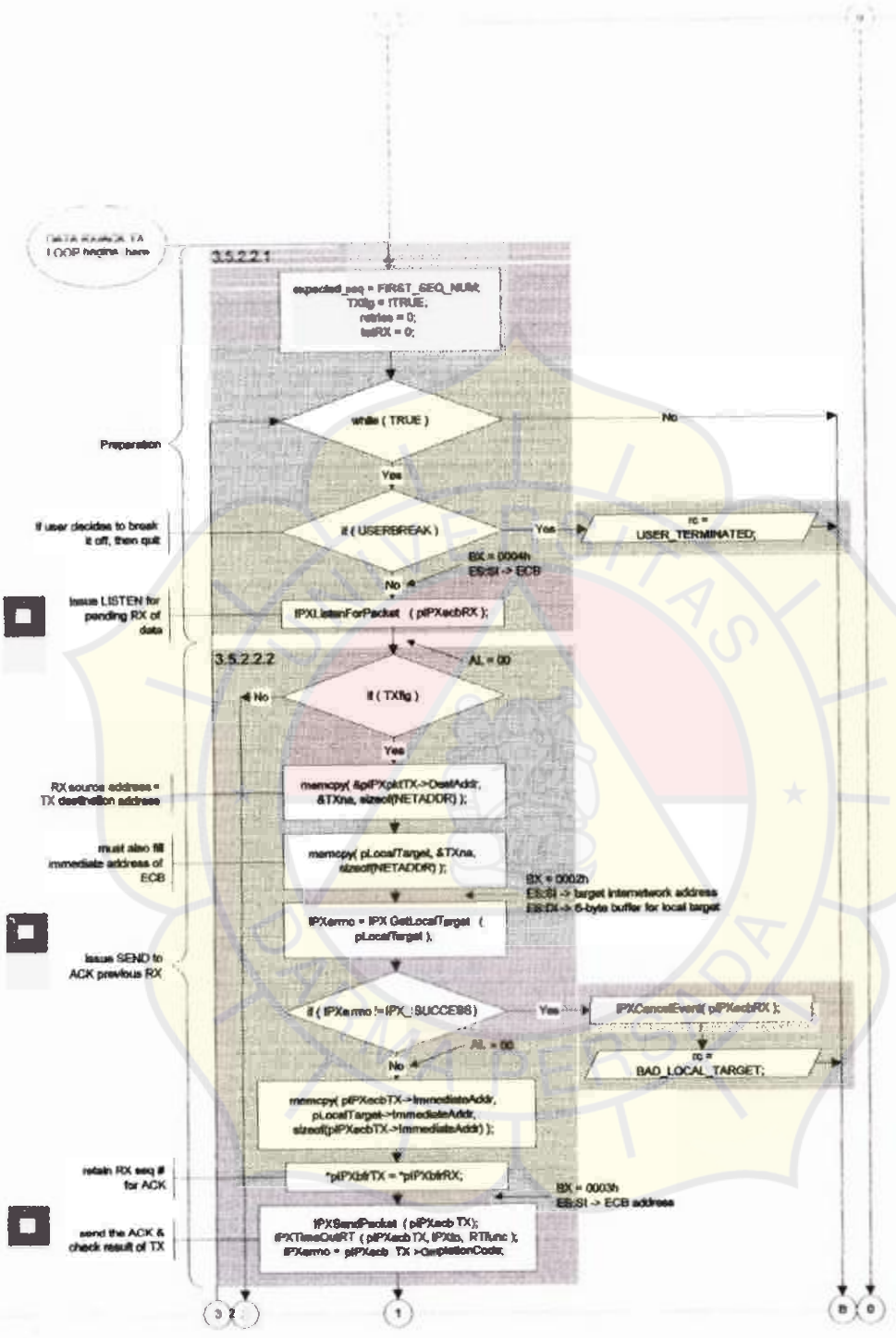

3 2
1
cancel the pending LISTEN
B 0
3.5.1.2.3
TX was successfully sent
if (IPXerrno != IPX_SUCCESS)
Yes
IPXCancelEvent (pIPXecbRX);
rc = BAD_TX_TRY;
No
wait for RX to complete, then check result
RX (ACK to previous TX) is pending
IPXTimeOutRT (pIPXecbRX, IPXto, RTIsec);
IPXerrno = pIPXecbRX->CompletionCode;
3.5.1.2.4
No
if (IPXerrno == IPX_SUCCESS)
Yes
check that RX was from expected Destination
if (memcmp(Destination, &pIPXpktRX->SrceAddr, sizeof(NETADDR)))
Yes
rc = BAD_RX_SRCE_ADDR;
No
check that TX/RX sequence in sync
if (expected_seq != *pIPXbeRX)
Yes
rc = BAD_RX_SEQ_NUM;
ACK was received successfully
No
3 4
1
B 0

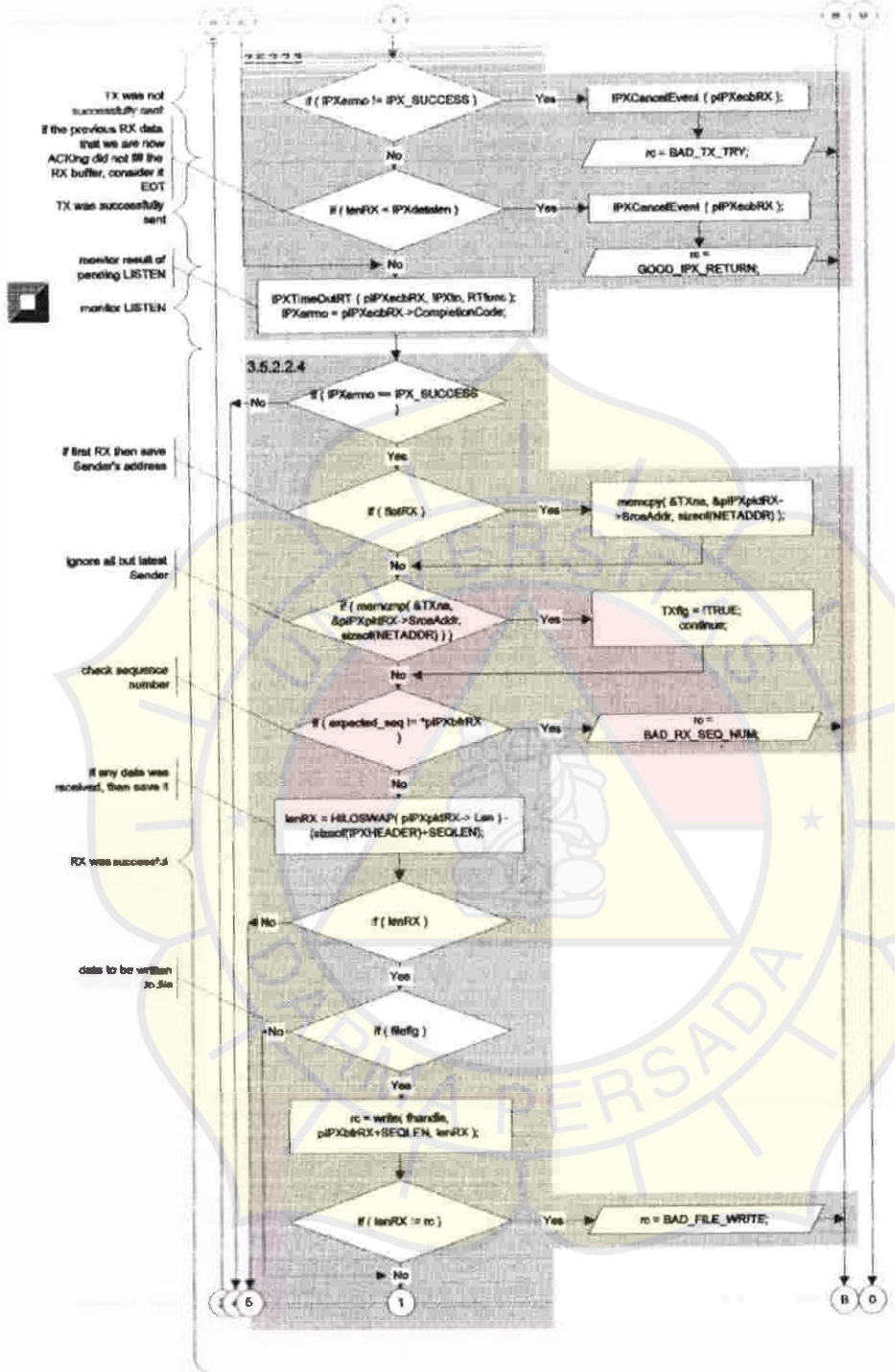

ACK was received successfully
If last TX was EOT, then all done!
otherwise, do next TX
otherwise, try RX again
no ACK was received
If maximum retries have been exhausted, then quit
cancel the pending LISTEN
close socket and close or delete something if went wrong
3.5.1.2.5
3.5.1.3.1
If (lenTX < IPXdataLen)
Yes
rc = GOOD_IPX_RETURN;
No
expected_seq++;
retries = 0;
retryTXflg = ITRUE;
if (IPXerrno != IPX_SUCCESS)
No
retryTXflg = TRUE;
Yes
No
if (retries++ >= IPXtries)
Yes
rc = BAD_RETRIES;
IPXSocketClose (Socket);
BX = 0001h
DX = socket number
No
if (fileflg)
Yes
close(handle);
1
0

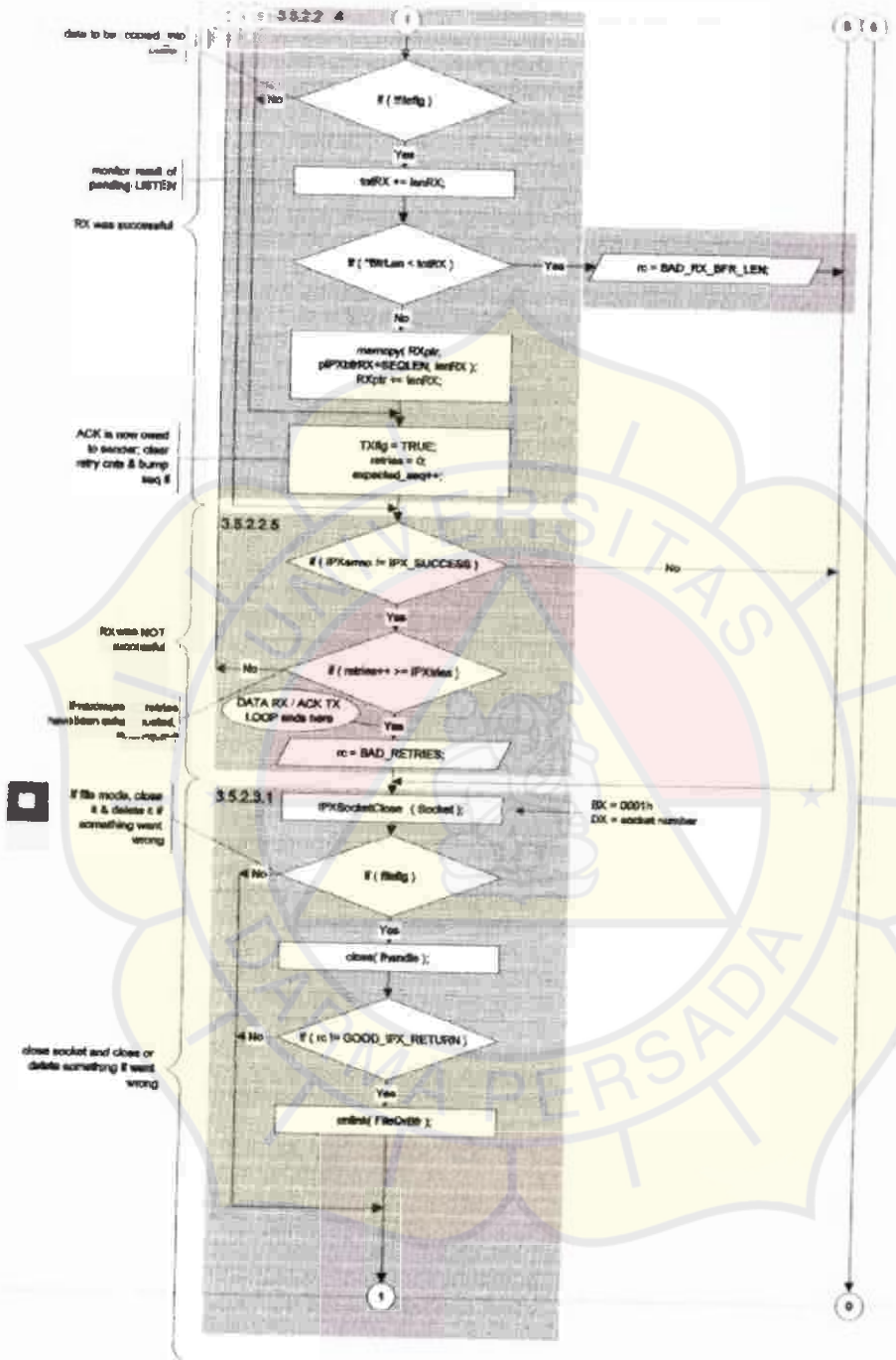

1
0
3.5.1.3.2
free(pIPXbfrRX);
free(pIPXbfrTX);
free(pIPXpktRX);
free(pIPXpktTX);
free(pIPXecbRX);
free(pIPXecbTX);
free(pLocalTarget);
free that memi
bring it on home
return(rc);
End transfering a file

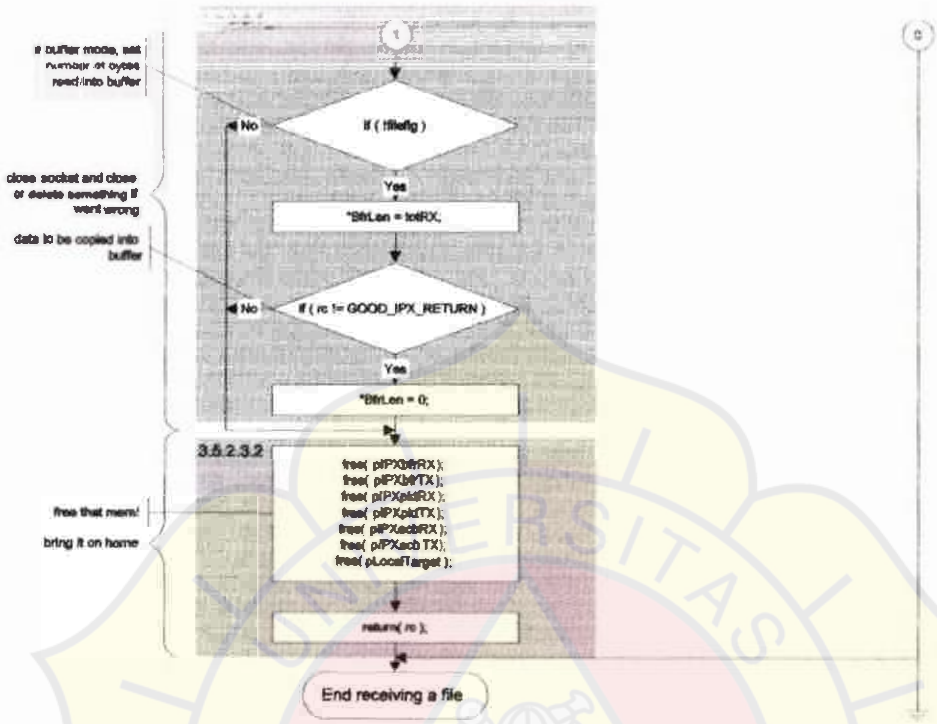

Begin receiving a file
unsigned IPXReceiveRT(long *BitLen, BYTE *FileOrBfr, char *SocketStr, void (*RTfunc)(void))
int fhandle; long totRX;
BYTE *RXptr;
unsigned rc, lenRX, retries;
int TXflg, fileflg;
BYTE Socket[2]; BYTE expected_seq;
NETADDR TXne;
BYTE *pIPXbfrRX, *pIPXbfrTX;
IPXHEADER *pIPXpktRX, *pIPXpktTX;
EVENTCONTROLBLOCK *pIPXecbRX, *pIPXecbTX;
LOCALTARGET *pLocalTarget;
check that IPX API has been installed or attempt to install it
3.5.2.1.1
if (!(IPXAPI) && (IPXInstalled () != IPX_PRESENT))
Yes
return(IPX_NOT_PRESENT);
ES:DI -> IPX input for pointer call address
No
if ((strlen(SocketStr) != sizeof(Socket)*2)) || !(xatoxb (SocketStr, Socket)*2)))
Yes
return(BAD_SOCKET_PARAM);
No
open socket for peer-to-peer communications
IPXermo = IPXSocketOpen (Socket, SOCKET_SHORT_LIVED);
sure, we could just go ahead, but the socket really shouldn't have been open at this point, and until the library has been thoroughly shaken out, (hopefully) ALL discrepancies will be caught & reported!
if (IPXermo != IPX_SUCCESS)
Yes
if (IPXermo = SOCKET_ALREADY_OPEN)
No
Yes
IPXSocketClose (Socket);
return(BAD_SOCKET_OPEN);
No
3.5.2.1.2
fileflg = TRUE;
set the receive mode
if (*BitLen)
No
Yes
RXptr = FileOrBfr;
fileflg = !TRUE;
3.5.2.1.3
if (fileflg)
No
Yes
attempt to open RX file
fhandle = open(FileOrBfr, O_RDWR| O_CREAT|O_TRUNC |O_BINARY, S_IREAD|S_IWRITE);
fhandle == -1
if (fhandle == FILE_PROBLEM)
Yes
IPXSocketClose (Socket);
return(BAD_FILE_OPEN);
No
1
0

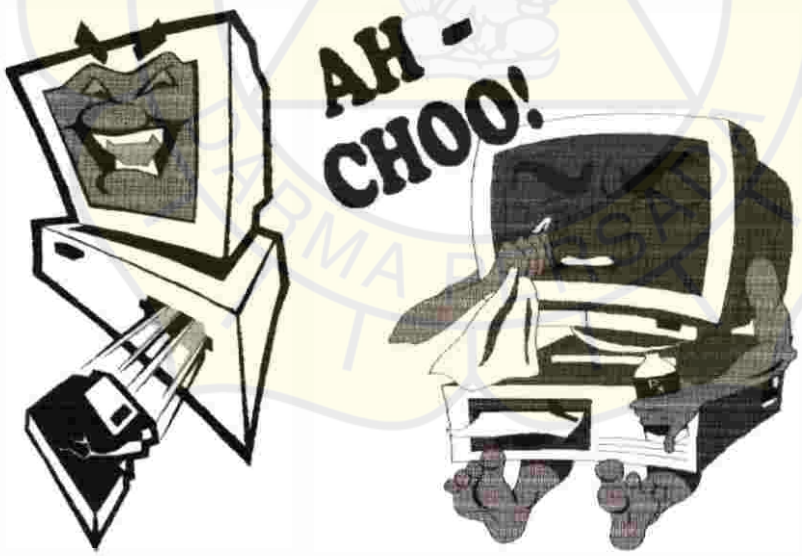

pIPXbfrRX = (BYTE *)calloc(1, SEQLEN+IPXdatalen);
pIPXbfrTX = (BYTE *)calloc(1, SEQLEN+IPXdatalen);
pIPXpktRX = (IPXHEADER *)calloc(1, sizeof(IPXHEADER));
pIPXpktTX = (IPXHEADER *)calloc(1, sizeof(IPXHEADER));
pIPXecbRX = (EVENTCONTROLBLOCK *)calloc(1, sizeof(EVENTCONTROLBLOCK));
pIPXecbTX = (EVENTCONTROLBLOCK *)calloc(1, sizeof(EVENTCONTROLBLOCK));
pLocalTarget = (LOCALTARGET *)calloc(1, sizeof(LOCALTARGET));
allocate necessary memory
if (pIPXbfrRX == NULL || pIPXbfrTX == NULL ||
pIPXpktRX == NULL || pIPXpktTX == NULL ||
pIPXecbRX == NULL || pIPXecbTX == NULL ||
pLocalTarget == NULL)
Yes
IPXSocketClose (Socket);
*BfrLen = 0;
No
3.5.2.1.5
memcpy(pIPXecbRX->Socket,
Socket, sizeof(pIPXecbRX->Socket));
pIPXpktRX->PacketType = IPX_PACKET_TYPE;
pIPXecbRX->FragmentCnt = 2;
pIPXecbRX->Fragment[0].BufAddr = (void *)pIPXpktRX;
pIPXecbRX->Fragment[0].Len = sizeof(IPXHEADER);
pIPXecbRX->Fragment[1].BufAddr = (void *)pIPXbfrRX;
pIPXecbRX->Fragment[1].Len = SEQLEN+IPXdatalen;
memcpy(pIPXecbTX->Socket,
Socket, sizeof(pIPXecbTX->Socket));
pIPXpktTX->PacketType = IPX_PACKET_TYPE;
pIPXecbTX->FragmentCnt = 2;
pIPXecbTX->Fragment[0].BufAddr = (void *)pIPXpktTX;
pIPXecbTX->Fragment[0].Len = sizeof(IPXHEADER);
pIPXecbTX->Fragment[1].BufAddr = (void *)pIPXbfrTX;
pIPXecbTX->Fragment[1].Len = SEQLEN+strlen(ACKSTR);
strcpy(pIPXbfrTX+SEQLEN,
ACKSTR);
Initialize ECBs/IPX packets
if (fileflg)
Yes
close(fhandle);
unlink(FileOrBfr);
if (pIPXbfrRX)
free(pIPXbfrRX);
if (pIPXbfrTX)
free(pIPXbfrTX);
if (pIPXpktRX)
free(pIPXpktRX);
if (pIPXpktTX)
free(pIPXpktTX);
if (pIPXecbRX)
free(pIPXecbRX);
if (pIPXecbTX)
free(pIPXecbTX);
if (pLocalTarget)
free(pLocalTarget);
return(BAD_MEMORY_ALLOC);
1
0


LOOP begins here
3.5.2.2.1
expected_seq = FIRST_SEQ_NUM;
TXflg = !TRUE;
retries = 0;
totRX = 0;
while (TRUE)
No
Yes
Preparation
if (USERBREAK)
Yes
rc = USER_TERMINATED;
If user decides to break it off, then quit
BX = 0004h
ES:SI -> ECB
No
Issue LISTEN for pending RX of data
IPXListenForPacket (pIPXecbRX);
3.5.2.2.2
AL = 00
if (TXflg)
No
Yes
RX source address = TX destination address
memcpy(&pIPXpktTX->DestAddr, &TXna, sizeof(NETADDR));
must also fill immediate address of ECB
memcpy(pLocalTarget, &TXna, sizeof(NETADDR));
BX = 0002h
ES:SI -> target internetwork address
ES:DI -> 6-byte buffer for local target
IPXerrno = IPX GetLocalTarget (pLocalTarget);
Issue SEND to ACK previous RX
if (IPXerrno != IPX_SUCCESS)
Yes
IPXCancelEvent(pIPXecbRX);
AL = 00
No
rc = BAD_LOCAL_TARGET;
memcpy(pIPXecbTX->ImmediateAddr, pLocalTarget->ImmediateAddr, sizeof(pIPXecbTX->ImmediateAddr));
retain RX seq # for ACK
*pIPXbfrTX = *pIPXbfrRX;
BX = 0003h
ES:SI -> ECB address
send the ACK & check result of TX
IPXSendPacket (pIPXecb TX);
IPXTimeOutRT (pIPXecb TX, IPXto, RTfunc);
IPXerrno = pIPXecb TX >CompletionCode;
3 2
1
B
0


TX was not successfully sent
If the previous RX data that we are now ACKing did not fill the RX buffer, consider it EOT
TX was successfully sent
monitor result of pending LISTEN
monitor LISTEN
if (IPXermo != IPX_SUCCESS)
Yes
IPXCancelEvent (pIPXecbRX);
rc = BAD_TX_TRY;
No
if (lenRX < IPXdatalen)
Yes
IPXCancelEvent (pIPXecbRX);
rc = GOOD_IPX_RETURN;
No
IPXTimeOutRT (pIPXecbRX, IPXto, RTfunc);
IPXermo = pIPXecbRX->CompletionCode;
3.5.2.2.4
No
if (IPXermo == IPX_SUCCESS)
Yes
If first RX then save Sender's address
if (flotRX)
Yes
memcpy(&TXns, &pIPXpktRX->SrceAddr, sizeof(NETADDR));
No
Ignore all but latest Sender
if (memcmp(&TXns, &pIPXpktRX->SrceAddr, sizeof(NETADDR)))
Yes
TXflg = !TRUE;
continue;
No
check sequence number
if (expected_seq != *pIPXbfrRX)
Yes
rc = BAD_RX_SEQ_NUM;
No
If any data was received, then save it
lenRX = HILOSWAP(pIPXpktRX-> Len) - (sizeof(IPXHEADER)+SEQLEN);
RX was successful
No
if (lenRX)
Yes
data to be written to file
No
if (filefig)
Yes
rc = write(thandle, pIPXbfrRX+SEQLEN, lenRX);
if (lenRX != rc)
Yes
rc = BAD_FILE_WRITE;
No
6
1
B
0


3.5.2.2.4
data to be copied into buffer
No
If (fileflg)
Yes
totRX += lenRX;
monitor result of pending LISTEN
If (*BfrLen < totRX)
Yes
rc = BAD_RX_BFR_LEN;
RX was successful
No
memcpy(RXptr, pIPXHdrRX+SEQLEN, lenRX);
RXptr += lenRX;
TXflg = TRUE;
retries = 0;
expected_seq++;
ACK is now owed to sender; clear retry cnts & bump seq #
3.5.2.2.5
If (IPXerrno != IPX_SUCCESS)
No
Yes
RX was NOT successful
No
If (retries++ >= IPXrtries)
DATA RX / ACK TX LOOP ends here
Yes
rc = BAD_RETRIES;
If maximum retries have been exhausted, re...
3.5.2.3.1
If file mode, close it & delete it if something went wrong
IPXSocketClose (Socket);
BX = 0001h
DX = socket number
No
If (fileflg)
Yes
close(fhandle);
No
If (rc != GOOD_IPX_RETURN)
Yes
unlink(FileDrBf);
close socket and close or delete something if went wrong
1
A
B
B
A


if buffer mode, set number of bytes read into buffer
close socket and close or delete something if went wrong
data to be copied into buffer
free that mem!
bring it on home
No
if (!fileflg)
Yes
*BfrLen = totRX;
No
if (rc != GOOD_IPX_RETURN)
Yes
*BfrLen = 0;
3.5.2.3.2
free(pIPXBfrRX);
free(pIPXbfrTX);
free(pIPXpktRX);
free(pIPXpktTX);
free(pIPXecbRX);
free(pIPXecb TX);
free(pLocalTarget);
return(rc);
End receiving a file


AH -
CHOO!

LAMPIRAN E

DAFTAR INTERUPROTOKOL IPX

➥ N-7A

INT 7A - Novell NetWare - LOW-LEVEL API - Notes

Note: this interrupt is used for IPX / SPX access in NetWare versions through 2.0a; in later versions, you should use INT 2F/AX=7A00h to get an entry point even though INT 7A still exists. For both INT 7A and the FAR entry point, BX contains the function number, IPX is sometimes called internally with BX bit 15 set, which causes the handler to bypass some initial checks and an optional call to the IPX Windows support handler set with INT 2F /AX=7AFFh /BX=0000h (see #1514).

SeeAlso: INT 2F/AX=7A00h, INT 64"Novell, INT 7A/BX=0000h

➥ N-7A, BX0000

INT 7A - Novell NetWare - IPX Driver - OPEN SOCKET

BX = 0000h

AL = socket longevity
- 00h open until close or terminate
- FFh open until close

DX = socket number (high byte in DL)
- 0000h dynamic allocation
- else socket to open (see #2212)

Return: AL = return code
- 00h success

DX = socket number
- FEh socket table full
- FFh socket already open

Notes:
- TSRs which need to use sockets should set AL to FFh, non-resident programs should normally use AL=00h
- IPX can be configured to support up to 150 open sockets on a workstation, and defaults to 20
- this function is supported by Advanced NetWare 1.02+

SeeAlso: INT 7A/BX=0001h,INT 7A/BX=0004h,INT 7A/BX=0023h

(Table 2212)
Values for IPX socket number:
0451h File Service (NetWare Core Protocol)
0452h Service Advertising Protocol
0453h Routing Information Packet
0455h NetBIOS Packet
0456h diagnostics
0457h server serial numbers (labeled "Copy Protection" by Lanalyzer)
4000h-7FFFh used for dynamic allocation
4444h Brightwork Development's SiteLock server
5555h Brightwork Development's SiteLock client (workstation)
8000h-FFFFh assigned by Novell
Note: SiteLock is an application metering product using IPX to communicate between the application and the license server

➥ N-7A, BX0001
INT 7A - Novell NetWare - IPX Driver - CLOSE SOCKET
BX = 0001h
DX = socket number (high byte in DL)
Notes: - also cancels events set by any Event Control Blocks for the socket
- the program must close all open sockets before terminating
- this function is supported by Advanced NetWare 1.02+
SeeAlso: BX=0000h

➥ N-7A, BX0002
INT 7A - Novell NetWare - IPX Driver - GET LOCAL TARGET
BX = 0002h
ES:SI -> target internetwork address (see INT 7A/ BX=000Bh)
ES:DI -> 6-byte buffer for local target
Return: AL = return code
00h success
CX = expected one-way transfer time (clock ticks) for a 576-byte packet
ES:DI -> local target
FAh unsuccessful (no path to destination)
Notes: - the internetwork address consists of a 4-byte network address followed by a 6-byte node address. The local target is only a 6-byte node address. If the target is in the same network, the local target is just the node address of target; otherwise, the local target is the node address of the bridge that leads to the target.
- this function may be called from inside IPX and AES Event Service Routines, but not from other interrupt handlers
- this function is supported by Advanced NetWare 1.02+
SeeAlso: BX=0009h

➥ N-7A, BX0003

INT 7A - Novell NetWare -IPX Driver - SEND PACKET
BX= 0003h
ES:SI -> Event Control Block (see #2213,#2214)

Notes:
- returns immediately; IPX attempts to send the packet in the background
- this function is supported by Advanced NetWare 1.02+
- this function is nearly identical to BX=000Fh, except that it always copies the source address into the IPX header assumed to be at the beginning of the first fragment

SeeAlso: BX=0004h,BX=000Fh,INT 21/AH=EEh"Novell"

Format of IPX Event Control Block: (Table 2213)

Offset	Size	Description
00h	DWORD	Link
04h	DWORD	-> Event Service Routine (00000000h if none)
08h	BYTE	in use flag (see #2215)
09h	BYTE	completion code (see #2216)
0Ah	WORD	(big-endian) socket number (see INT 7A/BX=0000h)
0Ch	4 BYTEs	IPX workspace
10h	12 BYTEs	driver workspace
1Ch	6BYTEs	immediate local node address
22h	WORD	fragment count
24h	var	fragment descriptors

Offset	Size	Description
00h	DWORD	-> fragment data
04h	WORD	size of fragment in bytes

Notes:
- ESR is a far procedure that is called when the ECB has been handled
- On call, the in use flag is zero if the ECB has been handled, non-zero otherwise. If the flag is zero, the completion code holds the result of the event
- the first fragment should start with an IPX header
- all fragments are concatenated and sent in one piece
- node address FFh FFh FFh FFh FFh FFh broadcasts to all nodes

Format of AES-ECB: (Table 2214)

Offset	Size	Description
00h	DWORD	Link
04h	DWORD	ESR address
08h	BYTE	in use flag (see #2215)
09h	5BYTEs	AES workspace

(Table 2215)
Values for ECB in use flag:

00h	available
E0h	AES temporary
F6h	\special IPX / SPX processing for v302+
F7h	/
F8h	IPX in critical section
F9h	SPX listening
FAh	processing
FBh	holding
FCh	AES waiting
FDh	AES couting down delay time
FEh	awaiting packet reception
FFh	sending packet

(Table 2216)
Values for ECB completion code:

00h	success
ECh	remote terminated connection without acknowledging packet
EDh	abnormal connection termination
EEh	invalid connection ID
EFh	SPX connection table full
F9h	event should not be caincelled
FAh	cannot establish connection with specified destination
FCh	cancelled
FDh	malformed packet
FEh	packet undeliverable
FFh	physical error

(Table 2217)
Values event Service Routine is called with:
AL = caller's idenfity (00h = AES, FFh = IPX)
ES:SI-> event control block
interrupts disabled

Format of IPX header: (Table 2218)

Offset	Size	Description
00h	WORD	(big-endian) checksum
02h	WORD	(big-endian) length in bytes of total packet
04h	BYTE	transport control
05h	BYTE	packet type (see #2219)
06h	10 BYTEs	destination internetwork address
10h	WORD	(big-endian) destination socket
12h	10BYTEs	source internetwork address
1Ch	WORD	(big-endian) source socket

(Table 2219)
Values for IPX packet type:

00h	unknown packet type
01h	routing information packet
02h	echo packet
03h	error packet
04h	packet exchange packet (always use this one)
05h	SPX packet
11h	NetWare Core Protocol
14h	Propagated Packet (for NetWare), NetBIOS name packet
15h-1Eh	experimental protocols
Note:	undocumented packet type 14h will cross up to 16 networks deep in all directions, as Aaron Martin of Origin Systems discovered, the first 64 bytes of the IPX data in such packets should be considered reserved, as IPX places the traversed server nodes there

Format of Service Advertising Protocol Service Query Packet: (Table 2220)

Offset	Size	Description
00h	30 BYTEs	IPX header
1Eh	WORD	(big-endian) query type
	0001h	general find service
	0003h	find nearest server
20h	WORD	(big-endian) server type (see INT 21/AH=E3h"NetWare")

Format of Service Advertising Protocol Server Identification Packet: (Table 2221)

Offset	Size	Description
00h	30 BYTEs	IPX header
1Eh	WORD	(big-endian) response type
	0002h	general service
	0004h	nearest service
20h	64N BYTEs	server entries (1-7) (see #2222)

Format of SAP server entry: (Table 2222)

Offset	Size	Description
00h	WORD	(big-endian) server type (see INT 21/AH=E3h"NetWare")
02h	48 BYTEs	ASCIZ server name
32h	2 WORDs	(big-endian) network number
34h	3 WORDs	(big-endian) node number
3Ch	WORD	(big-endian) socket number
3Eh	WORD	(big-endian) number of hops between caller and server

Format of IPX Routing Information packet: (Table 2223)

Offset	Size	Description
00h	30 BYTEs	IPX header
1Eh	WORD	operation (0001h request, 0002h response)
20h	8N BYTEs	network entries (1-50) (see #2224)

Format of RIP network entry: (Table 2224)

Offset	Size	Description
00h	DWORD	network number (FFFFFFFFh = general request)
04h	WORD	(response) number of hops
06h	WORD	(response) number of clock ticks to reach destination

➥ N-7A, BX0004

INT 7A - Novell NetWare - IPX Driver - LISTEN FOR PACKET
BX = 0004h
ES:SI -> Event Control Block (see BX=0003h)
Return: AL = status
00h successful
FFh no listening socket for packet
Desc: this function provides IPX with an ECB for receiving an IPX packet, but does not wait for a packet to arrive
Notes:
- the application must open a socket and initialize the ECB's ESR address, socket number, fragment count, and fragment descriptor fields before invoking this function
- there is no limit on the number of ECBs which may simultaneously be listening on a socket
- this function is supported by Advanced NetWare 1.02+

SeeAlso: BX=0000h,BX=0003h

➥ N-7A, BX0005

INT 7A - Novell NetWare - IPX Driver - SCHEDULE IPX EVENT
BX = 0005h
AX = delay time in clock ticks
ES:SI -> Event Control Block (see BX=0003h)
Note: this function is supported by Advanced NetWare 1.02+
SeeAlso: BX=0006h,BX=0007h,BX=0008h

➥ N-7A, BX0006

INT 7A - Novell NetWare - IPX Driver - CANCEL EVENT
BX = 0006h
ES:SI -> Event Control Block (see BX=0003h)
Return: AL = return code (see #2225)
Notes:
- cannot cancel packets which the node's driver has already sent
- this function is supported by Advanced NetWare 1.02+

SeeAlso: BX=0005h

(Table 2225)
Values for IPX event code:
00h success
F9h event in use
FCh event cancelled
FFh unsuccessful, event not in use, or unrecognized ECB flag

➥ N-7A, BX0007
INT 7A - Novell NetWare - IPX Driver - SCHEDULE SPECIAL EVENT
BX = 0007h
AX = delay time
ES:SI-> Event Control Block (see BX=0003h)
Note: this function is supported by Advanced NetWare 1.02+
Seealso: BX=0006h

➥ N-7A, BX0008
INT 7A - Novell NetWare - IPX Driver - GET INTERVAL MARKER
BX= 0008h
Return: AX = interval marker in clock ticks
Notes: - may be used to measure the time elapsed between two events, up to one hour
- this function is supported by Advanced NetWare 1.02+
Seealso: BX=0005h

➥ N-7A, BX0009
INT 7A - Novell NetWare - IPX Driver - GET INTERNETWORK ADDRESS
BX= 0009h
ES:SI-> buffer for own internetwork address (see #2226)
Return: ES:SI buffer filled
SI destroyed
Note: this function is supported by Advanced NetWare 1.02+
Seealso: BX=0002h,BX=000Bh

Format of IPX internetwork address: (Table 2226)
Offset Size Description
00h 4BYTEs (big-endian) network number
04h 6BYTEs (big-endian) node number within network

➥ N-7A, BX000A
INT 7A - Novell NetWare - IPX Driver - RELINQUISH CONTROL
BX = 000Ah
Desc: this call indicates that the application is idle and permits the IPX driver to do some work
Note: this function is supported by Advanced NetWare 1.02+
Seealso: INT 15/AX=1000h,INT 21/AH=89h,INT 2F/AX=1680h

➥ N-7A, BX000B

INT 7A - Novell NetWare - IPX Driver - DISCONNECT FROM TARGET
BX = 000Bh
ES:SI -> internetwork address (see #2227)

Notes:
- this function permits the network software on the remote machine to remove any virtual connection with the calling machine only use in point-to-point networks
- should never be called from within an Event Service Routine
- this function is supported by Advanced NetWare 102+

SeeAlso: BX=0002h,BX=0009h

Format of IPX internetwork address: (Table 2227)

Offset	Size	Description
00h	4 BYTEs	(big-endian) destination network
04h	6BYTEs	(big-endian) destination node
0Ah	2BYTEs	(big-endian) destination socket

➥ N-7A, BX000C

INT 7A U - Novell NetWare - IPX Driver - internal - INITIALIZE NETWORK ADDRESS
BX = 000Ch
CX:DX = global network address (see INT 7A/ BX=0002h)
ES:DI -> "OS!NCRITICALSECTION" flag
DS:SI -> current mode for socket

Note: the address cannot be changed once it has been initialized

Seealso: INT 7A/ BX=0024h

➥ N-7A, BX000D

INT 7A U - Novell NetWare - IPX Driver - internal - IPX GET PACKET SIZE
BX = 000Dh

Return: AX = maximum packet size
CX = retry count

Seealso: BX=001Ah

➥ N-7A, BX000E

INT 7A U - Novell NetWare - IPX Driver - internal - TERMINATE SOCKETS
BX= 000Eh

Return: nothing

Notes:
- this function terminates all sockets opened with the current mode; this may be intended for future enhancements as the socket mode never changes inv2.15
- called by the NetWare shell if a program terminates

➥N-7A, BX000F
INT 7A - Novell NetWare - IPX Driver - INTERNAL - SEND PACKET
BX = 000Fh
ES:SI-> Event Control Block (see BX=0003h)
Note: nearly identical to function 0003h, but does not copy address into the first fragment, and bypasses normal error checking
Seealso: BX=0003h

➥ N-7A, BX0010
INT 7A - Novell NetWare - SPX Driver - INSTALLATION CHECK
BX = 0010h
AL = 00h
Return: AL = FFh if SPX loaded
BH = SPX major version
BL = SPX minor version
CX = maximum SPX connections
DX = SPX connections available
Notes:
- this function is supported by Advanced NetWare 2.1+ this interrupt is used for IPX / SPX access in NetWare versions through 2.0a; in later versions, you should use INT 2F / AX=7A00h to get an entry point even though INT 7A still exists. For both INT 7A and the FAR entry point, BX contains the function number
- IPX is sometimes called internally with BX bit 15 set, which causes the entry point handler to bypass some checks and an optional call to the IPX Windows support handler set with INT 2F / AX=7AFFh / BX=0000h (see #1514)

SeeAlso: BX=0015h

➥ N-7A, BX0011
INT 7A - Novell NetWare - SPX Driver - ESTABLISH SPX CONNECTION
BX = 0011h
AL = retry count
AH = watchdog flag
ES:SI-> Event Control Block (see BX=0003h)
Return: AL = status (see #2228)
DX = assigned connection ID number
Desc: attempt to establish a connection with a listening socket
Notes:
- there should always be at least two SPX ECB's listening to a socket, so that NetWare can perform its internal packet exchanges
- the first fragment should start with a SPX header. Fill in all destination addresses.
- this function is supported by Advanced NetWare 2.1+

SeeAlso: BX=0000h,BX=0012h,BX=0013h,BX=0014h,BX=0015h

(Table 2228)
Values for SPX function status:

00h	attempting to contact destination socket
Eth	local connection table full
Fdh	buffer size not 42 or fragment count not 1
Fth	sending socket not open

Format of SPX header: (Table 2229)

Offset	Size	Description
00h	WORD	(big-endian) checksum
02h	WORD	(big-endian) length in bytes of total packet
04h	BYTE	transport control
05h	BYTE	packet type (see INT 7A / BX=0003h)
06h	10 BYTEs	destination internet address
10h	WORD	(big-endian) destination socket
12h	10BYTEs	source internet address
1Ch	WORD	(big-endian) source socket
1Eh	BYTE	connection control (see #2230)
1Fh	BYTE	datastream type
	FEh	terminate connection request packet
	FFh	terminate connection acknowledgement packet
	other	user-defined, ignored by SPX
20h	WORD	(big-endian) source connection ID
22h	WORD	(big-endian) destination connection ID
24h	WORD	(big-endian) sequence number
26h	WORD	(big-endian) acknowledge number
28h	WORD	(big-endian) allocation number

Bitfields for SPX connection control: (Table 2230)

Bit(s)	Description
3-0	unused???
4	end of message
5	reserved
6	acknowledgement required
7	system packet

➥ N-7A, BX0012

INT 7A - Novell NetWare - SPX Driver - LISTEN FOR SPX CONNECTION
BX = 0012h
AH = watchdog flag (00h disabled, 01h enabled)
AL = retry count (00h = default)
ES:SI -> Event Control Block (see BX=0003h)

Notes: - there should always be at least two SPX ECB's listening to a socket, so that NetWare can perform its internal packet exchanges
- this function is supported by Advanced NetWare 2.1+

Seealso: BX=0011h,BX=0013h,BX=0014h

➥ N-7A, BX0013

INT 7A - Novell NetWare - SPX Driver - TERMINATE SPX CONNECTION
BX = 0013h
DX = connection ID to terminate
ES:SI -> Event Control Block (see BX=0003h)

Note: - this function is supported by Advanced NetWare 2.1+

Seealso: BX=0011h,BX=0012h,BX=0014h

➥ N-7A, BX0014

INT 7A - Novell NetWare -SPX Driver - ABORT SPX CONNECTION
BX = 0014h
DX = connection ID to terminate

Notes: - this function is supported by Advanced NetWare 2.1+
- this function does not tell the other side that the connection has been terminated
- also aborts any outstanding Establish Connection, Terminate Connection, and Send Sequenced Packet commands

SeeAlso: BX=0011h,BX=0013h

➥ N-7A, BX0015

INT 7A - Novell NetWare - SPX Driver - GET SPX CONNECTION STATUS
BX = 0015h
DX = connection ID
ES:SI -> status buffer (see #2231)

Return: AL = return code
00h connection still valid
ES:SI -> status buffer filled
EEh no such connection

Note: this function is supported by Advanced NetWare 2.1+

SeeAlso: BX=0010h,BX=0011h

Format of SPX status buffer: (Table 2231)

Offset	Size	Description
00h	BYTE	connection state
	01h	waiting to establish connection
	02h	starting (attempting to create connection)
	03h	connection established
	04h	terminating

01h	BYTE	watchdog flag bit 0: used internally by SPX bit 1: SPX watchdog is monitoring connection bits 2-7 used internally by SPX
02h	WORD	(big-endian) source connection ID
04h	WORD	(big-endian) destination connection ID
06h	WORD	(big-endian) sequence number of next packet sent
08h	WORD	(big-endian) acknowledge number, expected sequence number of next received packet
0Ah	WORD	(big-endian) maximum sequence number remote SPX may send without ACK from local SPX
0Ch	WORD	(big-endian) remote acknowledge number, next sequence number remote SPX expects to receive
0Eh	WORD	(big-endian) remote allocation number, maximum sequence number local SPX may send
10h	WORD	(big-endian) connection socket
12h	6BYTEs	immediate node address--bridge on local network to destination
18h	10BYTEs	destination internetwork address (see INT 7A/BX=000Bh)
22h	WORD	(big-endian) retransmit count
24h	WORD	(big-endian) estimated roundtrip delay
26h	WORD	(big-endian) retransmitted packets
28h	WORD	(big-endian) suppressed packets
2Ah	12BYTEs	??? (v2.15)

➥ N-7A, BX0016

INT 7A - Novell NetWare - SPX Driver - SEND SPX PACKET

BX = 0016h
DX = connection ID
ES:SI-> Event Control Block (see BX=0003h)

Notes: - this function is supported by Advanced NetWare 2.1+
- CX may need to be 0001h ???

Seealso: BX=0011h,BX=0017h

➥ N-7A, BX0017

INT 7A - Novell NetWare - SPX Driver - LISTEN FOR SPX PACKET

BX= 0017h
DX = connection ID (unused in v2.15)
ES:SI-> Event Control Block (see BX=0003h)

Notes: - this function is supported by Advanced NetWare 2.1+
- CX may need to be 0001h ???

SeeAlso: BX=0011h,BX=0016h

➥ N-7A, BX0018

INT 7A U - Novell NetWare - IPX Driver - internal - ADD DIAGNOSTIC ELEMENT
BX = 0018h
ES:SI -> diagnostic element (see #2232) to be added to Diagnostic Queue
Note: this function is supported on file servers only under v2.15; v3.02 also supports it on workstations
SeeAlso: BX=0019h

Format of IPX diagnostic element: (Table 2232)

Offset	Size	Description
00h	DWORD	pointer to next diagnostic element
04h	DWORD	pointer to function for ???
08h	DWORD	pointer to function for ???

➥ N-7A, BX0019

INT 7A U - Novell NetWare - IPX Driver - internal - CANCEL DIAGNOSTIC ELEMENT
BX = 0019h
ES:SI -> diagnostic element (see BX=0018h) to be removed
Note: this function is supported on file servers only under v2.15; v3.02 also supports it on workstations
SeeAlso: BX=0018h

➥ N-7A, BX001A

INT 7A - Novell NetWare - IPX Driver - internal - GET DRIVER PACKT SIZE LIMIT
BX = 001Ah
Return: AX = packet size with preamble
CX = IPX retry count
Note: this function is has existed since November 1989; it is documented in Novell document FYI.A.3709, 03May91
SeeAlso: BX=000Dh

➥ N-7A, BX001B

INT 7A U - Novell NetWare - IPX Driver - INTERNAL
BX = 001Bh
???
Return: ???
Notes: - this function is supported on file servers only under v2.15; v3.02 also supports it on workstations
- used by NetWare Access Server

➥ N-7A, BX001C

INT 7A U - Novell NetWare - NetWare Access Server - ???
BX = 001Ch to 001Eh
???

Return: ???
AX corrupted
Notes: Procedure declared NEAR for standard IPX drivers such as IPXODIv2.12
SeeAlso: INT 2F/AX=7AF1h

➥ N-7A, BX001F

INT 7A - Novell NetWare - IPXODIv212+ - GET IPX FLAGS
BX = 001Fh
DX = 0000h
Return: AX = ???
BX:CX -> ??? entry point (see #2233)
DX = flags
bit 0:IPXODI rather than dedicated IPX
bit 1: checksumming functions 0020h-0022h supported
ES:SI -> array of words containing used socket number (undocumented);
0000h ends array
Note: IPXODI v2.12 is distributed as part of the Personal NetWare system bundled with Novell DOS 7
SeeAlso: BX=0020h,INT 2F/AX=7A2Fh

(Table 2233)
Call IPXODI entry point with:
ES:SI -> ??? data (offset 24h is far pointer to ??? data)

➥ N-7A, BX0020

INT 7A - Novell NetWare -IPXODIv2.12+ - SEND WITH CHECKSUM
BX = 0020h
ES:SI-> ECB data (see #2234)
Return: DS preserved
BX,BP corrupted
SeeAlso: BX=001Fh,BX=0021h,BX=0022h,INT 2F/AX=7A2Fh

Format of ECB data: (Table 2234)

Offset	Size	Description
00h	8 BYTEs	???
08h	BYTE	(return) ???
09h	BYTE	(return) ???
0Ah	WORD	???
0Ch	16BYTEs	???
1Ch	3WORDs	???
22h	WORD	number of records at offset 28h
24h	DWORD	-> ???
28h	6N BYTEs	??? records

➥ N-7A, BX0021

INT 7A - Novell NetWare - IPXODIv212+ - IPX GENERATE CHECKSUM
BX = 0021h
ES:SI-> ECB data (see #2234)
Return: ES,DS,SI preserved
BX,BP corrupted
Notes: - the checksum and TransportControl fields of the IPX packet are updated
- this function enables interrupts and is fully reentrant
SeeAlso: BX=001Fh,BX=0020h,BX=0022h,INT 2F / AX=7A2Fh

➥ N-7A, BX0022

INT 7A - Novell NetWare - IPXODIv2.12+ - IPX VERIFY CHECKSUM
BX = 0022h
ES:SI-> ECB data (see #2234)
Return: AX = status (0000h checksum matches)
DS,ES,SI preserved
BX,BP corrupted
Note: this function enables interrupts and is fully reentrant
SeeAlso: BX=001Fh,BX=0020h,BX=0021h,INT 2F / AX=7A2Fh

➥ N-7A, BX0023

INT 7A - Novell NetWare - IPXODI v2.12+ - OPEN LOOK-AHEAD SOCKET
BX = 0023h
AL = ???
DX = socket number
ES:SI-> Look Ahead handler (see #2235)
Return: AL = 00h if successful
BX corrupted
Notes: - the socket will always be long-lived, and must thus be explicitly closed with INT 7A/BX=0001h before the Look Ahead handler code is removed from memory (i.e. the program terminates)
- this function is only supported if INT 2F / AX=7A00h returns ES:BX pointing at an IPX version greater than 3.30
SeeAlso: INT 7A/BX=0000h,INT 7A/BX=0001h

(Table 2235)
Call IPX Look-Ahead handler with:
AX = socket
DS:SI-> look-ahead structure (see #1583 at INT 2F / AX=C000h"LSL")
DF clear
interrupt disabled (must remain disabled)
Return: AX = packet use
0000h application want packet
ES:SI-> ODI ECB (see #1584 at INT 2F / AX=C000h"LSL")

3001h application does not want packet
ZF set if AX=0000h
DS,DI,BP,SS,SP preserved

➥ N-7A, BX0024

INT 7A U - Novell NetWare - IPXODI v2.20+ - SET INTERNETWORK ADDRESS
BX = 0024h
ES:SI -> buffer containing internetwork address (see #2227)
Return: BX,CX,SI,DI,ES destroyed
Note: this function differs from INT 7A/BX=000Ch in that it unconditionally sets the address
SeeAlso: INT 7A/BX=000Ch

LAMPIRAN F

DAFTAR OPERATOR DAN SINTAKS C++

F.1 Constantsand internal representation[1]

ANSI C acknowledges that the size and numeric range of the basic data types (and their various permutations) are implementation-specific and usually derive from the architecture of the host computer. For Borland C++, the target platform is the IBM PC family (and compatibles), so the architecture of the Intel 8088 and 80x86 microprocessors governs the choices of internal representations for the various data types.

The following tables list the sizes and resulting ranges of the data types for Borland C++. Internal representation of numerical types shows how these types are represented internally.

F.1.1 16-bit data types, sizes, and ranges

Type	Size (bits)	Range	Sample applications
unsigned char	8	0 to 255	Small numbers and full PC character set
char	8	-128 to 127	Very small numbers and ASCII characters
enum	16	-32,768 to 32,767	Ordered sets of values
unsigned int	16	0 to 65,535	Larger numbers and loops
short int	16	-32,768 to 32,767	Counting, small numbers, loop control
int	16	-32,768 to 32,767	Counting, small numbers, loop control
unsigned long	32	0 to 4,294,967,295	Astronomical distances
long	32	-2,147,483,648 to 2,147,483,647	Large numbers, populations
float	32	3.4^10-38 to 3.4 ^ 1038	Scientific (7-digit precision)
double	64	1.7 E0- 30 8 to 1.7 ^ 10.308	Scientific (15-digit precision)
long double	80	3.4 E0-4 9 3 2 to 1.1 ^ 104932	Financial (18-digit precision)
near pointer	16	Not applicable	Manipulating memory addresses
far pointer	32	Not applicable	Manipulating addresses outside current segment

[1]Borland C++ 5.0 Programmer's Guide

F.1.2 32-bit data types, sizes, and ranges

Type	Size (bits)	Range	Sample applications
unsigned char	8	0 to 255	Small numbers and full PC character set
char	8	-128 to 127	Very small numbers and ASCII characters
short int	16	-32,768 to 32,767	Counting, small numbers, loop control
unsigned int	32	0 to 4,294,967,295	Large numbers and loops
int	32	-2,147,483,648 to 2,147,483,647	Counting, small numbers, loop control
unsigned long	32	0 to 4,294,967,295	Astronomical distances
enum	32	-2,147,483,648 to 2,147,483,647	Ordered sets of values
long	32	-2,147,483,648 to 2,147,483,647	Large numbers, populations
float	32	3.4*10-38 to 1.7*10^38	Scientific (7-digit) precision
double	64	1.7*10-308 to 3.4*10^10308	Scientific (15-digit precision)

F.2 Input / Output

Data type	Printf( ) format	Scanf( ) format
signed decimal integer	%d	%d
floating point		
decimal	%f	%e or %f
square	%e	%e or %f
decimal square	%g	
double Precision	%lf	%lf
single character	%c	%c
string	%s	%s
unsigned decimal integer	%u	%u
long integer	%ld	%ld
unsigned long integer	%lu	%lu
unsign hexadecimal integer	%x	%x
unsign octal integer	%o	%o

F.3 Character Constants

meaning	symbol	escape sequence
newline	NL (LF)	\n
horizontal tab	HT	\t
vertical tab	VT	\v
backspace	BS	\b
carriage return	CR	\r
form feed	FF	\f
alert	BEL	\a
backslash	\	\\
question mark	?	\?
single quote	'	\'
double quote	"	\"
integer 0	NUL	\0
octal number	ooo	*\ooo*
hex number	*hhh*	*\xhhh*

F.1 Operator[2]

Operator Summary

Operator	Meaning	Note
::	scope resolution	*class_name :: member*
::	global	*:: name*
.	member selection	*object. member*
->	member selection	*pointer -> member*
[ ]	subscripting	*pointer [expr]*
( )	function call	*expr (expr_list)*
( )	value construction	*type (expr_list)*
++	post increment	*lvalue ++*
--	post decrement	*lvalue --*
sizeof	size of object	*sizeof expr*
sizeof	size of type	*sizeof (type)*
++	pre increment	*++ lvalue*
--	pre decrement	*-- lvalue*
~	complement	*~ expr*
!	not	*!expr*
-	unary minus	*- expr*
+	unary plus	*+ expr*
&	address of	*& value*
*	dereference	** expr*
new	create (allocate)	*new type*
delete	destroy (de-allocate)	*delete pointer*
delete [ ]	destroy array	*delete[] pointer*
( )	cast (type conversion)	*(type)expr*
.*	member selection	*object. * pointer-to-member*
->*	member selection	*pointer-> * pointer-to-member*
*	multiply	*expr * expr*
/	divide	*expr/ expr*
%	modulo (remainder)	*expr% expr*
+	add (plus)	*expr + expr*
-	subtract (minus)	*expr - expr*
<<	shift left	*expr << expr*
>>	shift right	*expr >> expr*
<	less than	*expr < expr*
<=	less than or equal	*expr <= expr*
>	greater than	*expr > expr*
>=	greater than or equal	*expr >= expr*

[2][STR91/89]

Operator Summary

Operator	Meaning	Note
==	equal	*expr == expr*
!=	not equal	*expr != expr*
&	bitwise AND	*expr & expr*
^	bitwise exclusive OR	*expr^ expr*
|	bitwise inclusive OR	*expr | expr*
&&	logical AND	*expr && expr*
||	logical inclusive OR	*expr || expr*
? :	condition expression	*expr ? expr : expr*
=	simple assignment	*lvalue = expr*
*=	multiply and assign	*lvalue *= expr*
/=	divide and assign	*lvalue /= expr*
%=	modulo and assign	*lvalue %= expr*
+=	add and assign	*lvalue += expr*
-=	subtract and assign	*lvalue -= expr*
<<=	shift left and assign	*lvalue <<= expr*
>>=	shift right and assign	*lvalue >>= expr*
&=	AND and assign	*lvalue &= expr*
|=	inclusive OR and assign	*lvalue |= expr*
^=	exclusive OR and assign	*lvalue ^= expr*
throw	throw exception	*throw expr*
,	comma (sequencing)	*expr , expr*

Each box holds operators with the same precedence. An operator has higher precedence than operators in lower boxes. For example, a+b*c means a+(b*c) because * has higher precedence than +, and a+b-c means (a+b)-c because + and - have the same precedence (and associate left to right).

F.5 Syntax[3]

Statement Syntax

statement:
declaration
{ *statement-list*$_{opt}$ }
`try` {*statement-list*$_{opt}$} *handler-list*
expression$_{opt}$;

`if`(*expression*) *statement*
`if`(*expression*) *statement* `else` *statement*
`switch`(*expression*) *statement*

`while`(*expression*) *statement*
`do` *statement* `while` (*expression*);
`for` (*for-init-statement expression*$_{opt}$; *expression*$_{opt}$) *statement*

`case` *constant-expression* : *statement*
`default`:*statement*
`break` ;
`continue` ;

`return` expression$_{opt}$;

`goto` identifier;
identifier : *statement*

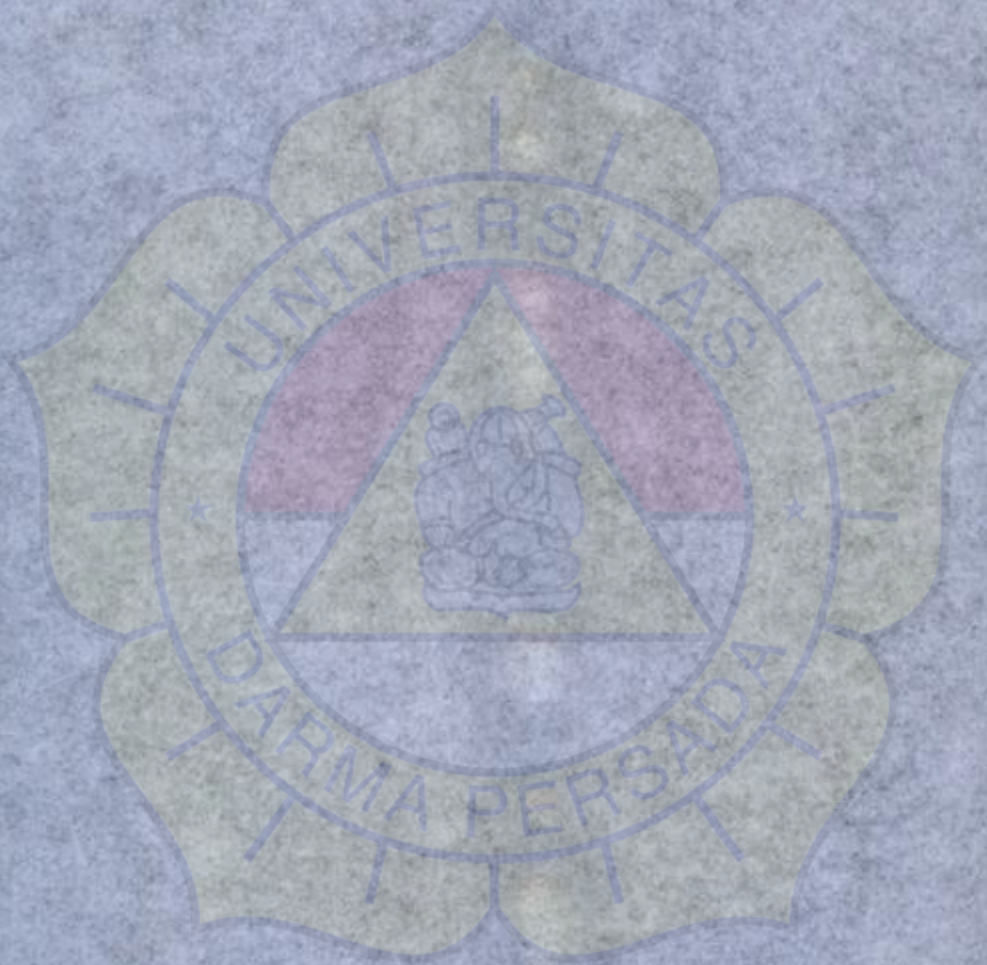
statement-list:
statement statement-list$_{opt}$

for-init-statement:
declaration
expression$_{opt}$;

handler-list:
`catch` (*exception-declaration*) { *statement-list*$_{opt}$}
handler-list handler-list$_{opt}$

Note that a declaration is a statement and that there is no assignment statement or procedure call statement; assignment and function call are handled as expression.

[3] [STR91/100]


UNIVERSITAS
DARMA PERSADA

LAMPIRAN G

INSTALASI PROGRAM PENGENDALI EN2 000 ETHERNET LAN

EN2000 Advanced Ethernet LAN

This diskette contains drive programs to allow you to run various network OS with the EN2000 Ethernet LAN. Each of the following directories contains a INSTALL.DOC file which provides you with detailed instructions to install the driver programs.

Directory Contents

README.TXT	This file.
\SETUP	Setup and diagnostic for EN2000 adapter.
\NETWARE	Novell NetWare ODI driver programs.
\WFW	The Driver (NDIS v2.0.1) is used for Windows for Workgroup.
\MSLANMAN.DOS	The Driver (NDIS v2.01) is used for Microsoft LAN Manager.
\PCTCP	Packet Driver Program.
\LANTASTI	Lantastic/AI installation guide.
\RPLBOOT	Novell Remote Boot PROM installation guide.
\WINNT	Windows NT installation guide.
\NDIS	The Driver (NDIS v2.0.1) is used for LANtastic v. 6.0, IBM LAN SERVER v4.0 (DOS LAN Services).

\SETUP Setup and diagnostic for EN2000 adapter

The following files are located in the setup directory of this Disk.

File	Contents
ESETUP.EXE	Setup program for use with software configurable Ethernet Address, Base I/O Address, Interrupt line, Boot ROM size and address. Diagnostic that your EN2000 board is working correctly.
INSTALL.DOC	This file that explains how to use the ESETUP.EXE.

Installing EN2000 SetUp

EN2000 SetUp is very easy to install. Just follow these steps:

1. Turn off the PC's power, insert the adapter into the PC.
1. Turn on the PC's power and boot DOS.
2. Insert the EN2000 SetUp Disk into a disk driver.
3. Type:
   ESETUP and strike <ENTER>.
4. Then follow the screen to appear is Main Menu screen.
   You can:
   - View the current configuration.
   - Set up new configuration.

Run diagnostics
- Save configuration to file
- Load configuration from file

Note: For help while running the program, press F1 will tell you to find information when you look for it.

\NETWARE Novell NetWare ODI driver programs.

EN2000 Novell NetWare Drivers

The description of various files are as follows:

LSLENH.NLM
- Enhanced LSL needed for 3.11 server

PATCHMAN.NLM
- A patch driver which is auto-loaded by LSLENH.NLM

EN2000LAN
- Novell Server Driver for 3.1x and 4.0x servers

ETHERTSM.NLM
- TSM module needed for 3.11 server and is auto-loaded by EN2000LAN

MSM.NLM
- MSM module needed for 3.11 server and is auto-loaded by ETHERTSM.NLM

MONITOR.NLM
- New monitor program needed for 3.11 server

EN2000.COM
- Novell DOS ODI Driver for workstation

INSTALLATION IN A 4.0x SERVER
In a Novel 4.0x system, copy EN2000LAN to F:\SYSTEM to load the driver.

LOAD EN2000

INSTALLATION IN A 3.11 SERVER
In a Novell 3.1x system, copy EN2000LAN, LSLENH.NLM, PATCHMAN.NLM, ETHERTSM.NLM, MSM.NLM and MONITOR.NLM to F:\SYSTEM to load the drivers.

LOAD LSLENH
LOAD EN2000

Note: The MONITOR program which is shipped with the original 3.11 system has a bug in it and will crash the server.

INSTALLATION DOS ODI IN A WORKSTATION
1) Refer to Novell DOS ODI installation menu.
2) Copy the NET.CFG file to the directory where the ODI driver resides.

Note: There is no need to use the NET.CFG file if the EN2000 adapter is an AT adapter configured for the default settings (i.e. Base I/O Address = 300, Interrupt = 3)

3) After booting the workstation, type the following commands:

LSL <Enter>
EN2000 <Enter>
IPXODI <Enter>
NETx <Enter>

You may wish to write all commands into a batch file and execute the batch file instead of typing all the commands each time the workstation needs to be booted up.

NOTE: The Driver (NE2K.DOS) is used for Windows For Workgroup

Follow the procedure below to run the EN2000 card with Microsoft's Windows for Workgroups(WFW). This shows how to run WFW with the NE2000 driver(or EN2000 driver).

3) Installing either the NE2000 Compatible driver or the EN2000 driver.

a. Running WFW with the NE2000 Compatible driver.

WINDOWS FOR WORKGROUPS V3.1

1. Select NETWORK program item from program group CONTROL PANEL.
2. Select ADAPTERS from the OPTIONS menu.
3. Select ADD button.
4. Select NE2000 Compatible NETWORK ADAPTER from adapter menu and choose OK button.
5. Choose OK button when asked to configured the adapter.
6. Select CLOSE button to end the ADAPTER setup.
7. Select OK to end NETWORK setup.

WINDOWS FOR WORKGROUPS V3.11

1. Select NETWORK SETUP program item from program group NETWORK.
2. Select DRIVERS button.
3. Select ADD ADAPTER button.
4. Select NE2000 Compatible NETWORK ADAPTER from adapter menu and choose OK button.
5. Select CLOSE button to end the DRIVERS setup.
6. Select OK button to end the NETWORK setup.

b. Running WFW with the EN2000 driver.

The OEMSETUP.INF file allow you to run Microsoft Windows for Workgroups with the EN2000 Ethernet LAN Adapter. The installation procedure is as follows:

WINDOWS FOR WORKGROUPS V3.1

1. Select NETWORK program item from program group CONTROL PANEL.
2. Select ADAPTERS from the OPTIONS menu.
3. Select ADD button.
4. Select UNLISTED OR UPDATED NETWORK ADAPTER from adapter menu and choose OK button.
5. Insert disk contains EN2000 DOS and OEMSETUP.INF in driver.
6. Specify the path (A:\WFW) and choose OK button.
7. Select "EN2000 Ethernet LAN" from adapter menu and choose OK button.
8. Choose OK button when asked to configured the adapter.
9. Select CLOSE button to end the ADAPTER setup.
10. Select OK to end NETWORK setup.

WINDOWS FOR WORKGROUPS V3.11

1. Select NETWORK SETUP program item from program group NETWORK.
2. Select DRIVERS button.
3. Select ADD ADAPTER button.
4. Select UNLISTED OR UPDATED NETWORK ADAPTER from adapter menu and choose OK button.
5. Insert disk contains EN2000 DOS and OEMSETUP.INF in driver.
6. Specify the path (A:\WFW) and choose OK button.
7. Select "EN2000 Ethernet LAN" from adapter menu and choose OK button.
8. Select CLOSE button to end the DRIVERS setup.
9. Select OK button to end the NETWORK setup.

4) Exit Windows for Workgroups and reboot your PC.

This completes the Windows for Workgroups installation.

Before installing the EN2000 Ethernet LAN into a station, you have to add the following lines into config.sys and autoexec.bat files or it would look like the following:

< Sample CONFIG.SYS file >

```
DEVICE=C:\WINDOWS\ IMEM.SYS
FILES=30
buffers=30
STACKS=9,256
DEVICE=C:\WINDOWS\IFSHLP.SYS
LASTDRIVE=Z
```

< Sample AUTOEXEC.BAT file >

```
@ECHO OFF
PROMPT $p$g
PATH C:\WINDOWS;
SET TEMP=C:\WINDOWS\TEMP
C:\WINDOWS\SMARTDRV.EXE
C:\WINDOWS\NET START
```

```
CALL
C:\NET\ODI\STARTNET.BAT          ; only for ODI driver
C:\WINDOWS\ODIHLP                ; only for ODI driver
```

< Sample STARTNET.BAT file >

```
CD \NET\ODI
SET NWLANGUAGE=ENGLISH
ISL
NE2000.COM
IPXODI /A
VLM
CD \
```

< Sample NET.CFG file >

```
Link driver NE2000
        PORT 300
        INT 3
        Frame ETHERNET_802.2

NetWare DOS Requester
        FIRST NETWORK DRIVE = F
```

\MSLANMAN.DOS

The Driver(NDIS v2.01) is used for Microsoft LAN Manager.

Follow the procedure below to run the EN2000 card with the Microsoft LAN Manager. The installation procedure is as follows:

1) Insert the EN2000 disk into floppy disk. Copy these files in this subdirectory to your hard disk.

```
\MS_LANMAN.DOS\DRIVERS\NIF\E_N2000.NIF
\MS_LANMAN.DOS\DRIVERS\ETHERNET\EN2000\PROTOCOL.INI
\MSLANMAN.DOS\DRIVERS\ETHERNET\EN2000\EN2000.DOS
```

To Before installing the EN2000 Ethernet LAN into workstation, you have to append the following lines into config.sys and autoexec.bat file or it would look like the following.

<CONFIG.SYS>
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\MSLANMAN.DOS\DRIVERS\PROTMAN\PROTMAN.DOS /I:C:\MSLANMAN.DOS
DEVICE=C:\MSLANMAN.DOS\DRIVERS\ETHERNET\EN2000\EN2000.DOS

<AUTOEXEC.BAT>
SET PATH=C:\MSLANMAN.DOS\NETPROG;%PATH%
NET START WORKSTATION
LOADNETNETUI
NET LOGON username

Change the path statements above which meet your own configuration.

The protocol.ini file is used by the LANMAN protocol manager and netbind program at startup time, append the protocol.ini file to the original existing protocol.ini file which contain the NDIS and transport specific parameters for the network.

\PCTCP Packet Driver Program

Follow the procedure below to run the EN2000 card with PC/TCP.
The installation procedure is as follows:

1) Insert the EN2000 disk into floppy disk. Copy EN2000.COM file to your hardisk.

2) Revise AUTOEXEC.BAT, insert the following lines:

EN2000 0x60
ETHDRV

Where, 0x60 is the software interrupt for the driver.

Now You are ready to use FTP's PC/TCP and its related functions.

[Note]. The file ETHDRV.EXE developed by FTP Software, Inc. must be installed beforehand.

\LANTASTI LantasticAI installation guide

Follow the procedure below to run the EN2000 card with Artisoft's LANtastic. This example uses LANtastic own NE2000 driver.

1) Run the LANtastic installation menu.
2) Choose NEX000 from the installation menu.
3) Follow all the operations as you are prompted on the screen.
4) After installing the Network Operating System, reboot your system.

Sample:

START.NET.BAT file for LANtastic/AI

@ECHO OFF
SET LAN_DIR=C:\LANTASTI.NET
SET LAN_CFG=C:\LANTASTI
PATH C:\LANTASTI;%PATH%
SHARE /L:200

[illegible] BROADCAST 255.255.255.255
ALL ARP
[illegible]
[illegible]
NETWARE DOS ADMIN
NET EPT TIMEOUT 10
NET POSTBOX

RPLBOOT Novell Remote Boot ROM installation guide

To use RPL BootROM. Note that the RPL.NLM files must exist in the SYSTEM directory, the RBOOT.RPL and ETHERRPL files exist in the LOGIN directory of a Netware server.

* RPL.NLM Is a Loadable Module that allows Remote Program Loading
*
REBOOTRPL For Adapters using the Novell Boot ROM Kit
*
ETHERRPL For Networks using the Ethernet Adapters

NOTE: You may not find them at the NetWare V3.11 server, copy those files from NetWare V3.12 or V4.x.

Setting up remote booting on a Netware v3.x or v4.x requires you to do the following:

1) Make a Bootable Diskette
   a. Format a floppy diskette using the DOS FORMAT /S command by typing

      FORMAT A: /S

   b. Creat or copy required files to the boot diskette.

      AUTOEXEC.BAT
      CONFIG.SYS
      COMMAND.COM
      NET.CFG
      ISL.COM
      EN2000.COM
      IPXODI.COM
      VLM.EXE and all VLM files
      .
      .

   c. Example file's

      === Sample CONFIG.SYS file ===
      .
      LASTDRIVE=Z
      .
      .

      === Sample AUTOEXEC.BAT file ===
      .
      ISL
      EN2000
      IPXODI
      VLM
      F:LOGIN
      .
      .

```
=== Example NET.CFG file ===

LINK DRIVER EN2000
    FRAME ETHERNET_802.2

NETWARE DOS REQUESTER
    FIRST NETWORK DRIVE = F
```

2) Creating Remote Boot Disk Image Files.

a. Insert you preapred bootable diskette into a floppy disk driver on the server.
b. Login NetWare as supervisor.
   Type:

```
MAP G:=SYS:SYSTEM <Enter>
MAP F:=SYS:LOGIN <Enter>
```

Change to the F drive, then type:

```
G:DOSGEN <Enter>
```

DOSGEN creat a disk images called NET$DOS.SYS in a the LOGIN directory.

c. Copy the AUTOEXEC.BAT file from the boot diskette to the LOGIN directory.

3) Load RPL.NLM (from the SYSTEM directory) and bind RPL the server's LAN card using the 802.2 frame type at the file server command prompt.

For Example:

```
LOAD NE2000 PORT=300 INT=3
  FRAME=ETHERNET_802.2 NAME=E8022
  BIND IPX TO E8022 NET=1
LOAD NE2000 PORT=300 INT=3
  FRAME=ETHERNET_802.3 NAME=E8023
  BIND IPX TO E8023 NET=2
  LOAD RPL
  BIND RPL TO E8022
```

4) Turn on the Diskless computer.

\WINNT Window.sNT installation guide

EN2000/CT use the Novell NE2000 compatible driver on a Windows NT system.

Please follow these instructions to load this driver on a WindowsNT system:

1. If Windows NT operating system is already installed,logon as Administrator.
2. Select "Network" icon from the control panel.
3. Select "Add Adapter" from the Network Settings window.
4. Select "Novell NE2000 Adapter" from the Add Network Adapter windows.
5. Also select appropriate hardware setup of the Novell NE2000 adapter.
6. Select OK from the Network window.
7. Exit Windows NT and Reboot your PC.

This completes the Windows NT installation.

How to Install EN2000.DOS on LANtastic v. 6.0

1. Insert LANTASTIC diskette labeled "DISK 1 of 4" and run INSTALL.EXE.
2. Select prefer installation options as required.
3. When asked to select your network adapter to run with your LANTASTIC software, select "NDIS Support for Network Adapter".
4. Insert your EN2000's driver diskette (EN2000.DOS, PROTOCOL.INI) in which contains the NDIS driver directory (NDIS) then select "OK".
5. Make sure your network adapter information is correct before you go on.
6. After the whole installation is completed, reboot your PC.

How to Install EN2000.DOS on IBM LAN SERVER 4.0(DOS LAN Services)

1. Insert the DOS LAN Service diskette Disk in the diskette driver (A or B)
2. Type the INSTALL command at the DOS command prompt to start the DOS LAN Services installation program.
3. Select prefer installation options as required.
4. When asked to select your network card to run with your DOS LAN Services software, select "Network card not shown in list below".
5. Insert your EN2000's driver diskette (EN2000.DOS, OEMSETUP.INF) in which contains the NDIS driver directory (NDIS) then press the Enter key to continue.
6. After you have verified you selections, select The" the listed options are correct" and press Enter.
7. Make sure your network adapter information is correct before you go on.
8. After the whole installation is completed, reboot your PC.

DAFTAR ISTILAH

Berikut adalah daftar istilah-istilah yang diambil dari buku *IBM Dictionary of Computing* [MCD94]. Angka dalam kurung superskrip adalah nomor halaman buku tersebut. Tanda (*) menunjukk an singkatan atau penjelasan yang diambil dari sumber lain.

access control Lihat kontrol akses.

adapter Bagian yang secara elektrik atau fisik menghubungkan suatu perangkat ke komputer atau perangkat lainnya. [12]

akumulasi (ACC) ① Mengumpulkan, contohnya, nilai dalam *field*. ② Memasukkan hasil operasi ke ak umulator.[6]

akumulator Register di mana satu *operand* operasi dapat ditempatkan dan selanjutnya ditimpa oleh hasil operasi tersebut.[6]

ALU Arithmetic and logic unit.

antarmuka Hardware, software, atau keduanya, yang menghubungkan sistem, program, atau perangkat. [351]

antarmuka multipleks (*multiplex interface*) Antarmuka antara DTE dan DCE yang menangani beberapa saluran dalam arti multipleksi pembagian waktu. [446]

API Application program interface.

application layer Lihat lapisan aplikasi.

***application program interface* (API)** Antarmuka fungsional yang disediakan sistem operasi atau program pesanan berlisensi terpisah yang memungkinkan program aplikasi ditulis dalam bahasa tingkat tinggi dengan memakai data atau fungsi-fungsi khusus dari sistem operasi atau program berlisensi itu. [28]

***arithmetic and logic unit* (ALU)** Bagian komputer yang melaksanakan operasi-operasi aritmetika, logika dan yang sejenisnya. [32]

array ① Susunan data dalam dimensi satu atau lebih: penyusunan bentuk daftar, tabel, atau suatu multidimensi. ② Dalam bahasa pemrograman, suatu kesatuan yang terdiri dari objek data, dengan atribut yang identik, di mana dapat direferensi secara unik dengan subsk ripsi. [32]

ASCII American National Standard Code for Information Interchange. Kode standar, memakai set karakter berkode 7-bit karakter (8 bit termasuk pengecek paritas), yang digunakan untuk pertukaran informasi di antara sistem-sistem proses, sistem-sistem komunikasi data, dan peralatan sejenisnya. Set ASCII terdiri dari karak ter-karakter kontrol dan karakter-karakter grafi k.[33]

ASCIIZ format String karakter-karakter ASCII yang diakhiri oleh karakter null. [33]

attached ① Dalam pemrograman, membuat tugas yang dapat dijalankan secara asinkron dengan eksekusi dari kode utama. ② Menghubungkan suatu perangkat secara logika ke jaringan cincin. [37]

bahasa pemrograman Bahasa artifisial untuk mengekpresikan program-program komputer. (538)

bahasa sumber ① Bahassa yang darinya pernyataan-pernyataan diterjemahkan. ② Bahasa pemrograman untuk mengekspresikan program-program sumber sehingga penerjemah tertentu dapat menerimanya. (635)

baseband transmission Lihat transmisi baseband.

Basic Input / Output System **(BIOS)** Kode pengontrol operasi-operasi perangkat keras, misalnya interaksi-interaksi degnan pengendali disk, *hard disk*, dan papan ketik. (56)

berdiri sendiri (*stand-alone*) Menyatakan operasi yang tidak terikat dengan perangkat, program, atau sistem lain. (644)

berkas (*file*) ① Serangkaian *record* bernama yang ditempatkan atau diproses sebagai suatu unit. ② Kompulan informasi yang dianggap sebagai satu unit. (268)

bingkai Struktur data yang menyatakan daerah khusus dari pengetahuan dan terdiri dari slot-slot yang dapat menerima nilai atribut spesifik dan dari mana gangguan dapat dicegah dengan penyertaan prosedur semestinya. (285)

bingkai data sinonim untuk paket. (170)

bingkai transmisi ① Dalam transmisi data, data yang dipindahkan dari satu simpul ke yang lainnya dalam format tertentu yang dapat dikenal oleh simpul penerima. Sebagai tambahan pada *field* data atau informasi, bingkai memiliki beberapa jenis delimiter yang menandai permulaan dan akhirannya dan biasanya mengontrol *field-field*, informasi alamat yang mengidentifikasi sumber dan tujuan, dan satu atau lebih bit-bit pengecek yang memungkinkan penerima mendeteksi kesalahan yang terjadi setelah pengirim telah mentransmisi bingkat. ② Dalam SDLC, pembawa perintah, respons, dan semua informasi yang ditransmisi memakai prosedur-prosedur SDLC. Tiap bingkai dimulai dan diakhiri bendera. ③ Sinonim untuk bingkai. (705)

BIOS Basic Input/ Output System.

BNC Konektor yang dipakai dengan beberapa kabel koaksial. (69)

board Lihat papan.

buffer Lihat penyangga.

bus network Lihat jaringan bus.

carrier sense Dalam LAN, aktivitas selanjutnya dari stasiun data untuk mendeteksi apakah stasiun lain sedang mengirim. (86)

carrier sense multiple access with collision detection **(CSMA/CD)** Protokol jaringan di mana stasiun kerja pengirim mengirim kembali data bila stasiun kerja penerima tidak menegaskan penerimaan data pada jangka waktu tertentu. (157)

CCITT (*International Telegraph and Telephone Consultative Committee*) Organisasi (satu dari empat organ permanen dari *International Telecommunication Union* [ITU], bermarkas di Jenewa, Swiss) yang memperhatikan persoalan-persoalan yang berkaitan dengan telepon dan telegraf internasional. CCITT Plenary Assembly bertemu secara teratur untuk mempersiapkan daftar pertanyaan teknik berkaitan dengan pelayanan telepon dan telegraf. Dewan itu memberikan pertanyaan-pertanyaan ini kepada kelompok-kelompok studi, yang kemudian menyiapkan rekomendasi untuk dipresentasikan pada pertemuan lengkap selanjutnya. Rekomendasi-rekomendasi yang disetujui dipublikasikan untuk pemakaian insinyur, ahli ilmu alam, dan ahli manufaktur di seluruh dunia. (89)

central processing unit **(CPU)** Bagian komputer yang mencakup rangkaian-rangkaian pengatur penerjemahan dan eksekusi instruksi-instruksi. (91)

checksum ① Jumlah suatu grup data yang berkaitan dengan grup itu dan memakainya untuk tujuan-tujuan pengecek an. ② Dalam deteksi kesalahan, suatu fungsi dari semua bit dalam blok. Bila hasil-

hasil yang ditulis atau dikalkulasi tidak sesuai, kesalahan ditunjukkan. ③ Pada disket, data yang ditulis pada sektor untuk tujuan-tujuan deteksi kesalahan; *checksum* hasil kalkulasi yang tidak sesuai dengan *checksum* data yang ditulis dalam sektor menunjukkan sektor buruk. Data dapat berupa numerik atau string-string karakter lain yang dianggap numerik untuk tujuan kalkulasi *checksum*. (161)

CPU Central processing unit.

CRC Cyclic redundancy check.

CSMA/CD Carrier sense multiple access with collision detection.

***cyclic redundancy check* (CRC)** Sistem pengecekan kesalahan yang dilakukan baik stasiun pengirim maupun penerima setelah karakter pengecekan blok diakumulasi. (162)

***DAK** Data AcKnowledge.

***data circuit-terminating equipment* (DCE)** Dalam stasiun data, peralatan yang menyadiakan konversi sinyal dan pengkodean di antara DTE dan saluran. (167)

datagram Dalam komunikasi paket, paket tersendiri, bebas dari paket-paket lainnya, yang membawa informasi untuk dikirim dari *data-terminal equipment* asal ke DTE tujuan tanpa bergantung pada pertukaran-pertukaran sebelumnya di antara DTE dan jaringan. (170)

data link layer Lihat lapisan sambungan data.

***data switching exchange* (DSE)** Peralatan yang diinstal pada lokasi tunggal untuk menyediakan fungsi-fungsi pensaklaran, seperti pensaklaran rangkaian (*circuit switching*), pencaklaran pesan (*message switching*), dan pensaklaran paket (*packet switching*). (177)

***data terminal equipment* (DTE)** Bagian dari stasiun data yang melayani sebagai sumber data, masukan data, atau keduanya. (117)

***DAV** Data Available.

DCE Data circuit-terminating equipment.

default Menyatakan suatu atribut, kondisi, nilai atau pilihan yang dipakai saat tidak ada spesifikasi eksplisit. (184)

delimiter ① Karakter yang dipakai mengindikasi awal dan akhir string karakter. ② Bendera yang memisahkan dan mengatur kelompok-kelompok data. Sinonim dengan simbol teratur, pemisah. ② Karakter yang mengelompokkan dan memisahkan kata-kata atau nilai-nilai dalam jalur input. (188)

demultiplekser Perangkat yang mengembalikan sebagai sinyal output, tiap kombinasi sinyal oleh multiplekser sebelumnya. (189)

detach Dalam sistem/38, mengacu pada penerima jurnal yang tidak dihubungkan pada jurnal dan tidak menerima masukan-masukan dari jurnal itu. Bertolak belakang dengan *attached*. (191)

***DFD** Data Flow Diagram.

***disk operating system* (DOS)** Sistem operasi untuk sistem-sistem komputer yang memakai *disk* atau disket untuk memori tambahan dari program-program atau data. (205)

DLL Dynamic link library.

DOS Disk operating system.

draft copy Lihat salinan draft.

driver Lihat pengendali.

DSE Data switching exchange.

DTE Data terminal equipment.

***dynamic link library* (DLL)** Berkas berisi kode dan data eksekusi berkaitan dengan program pada waktu muat atau waktu jalan, bukannya selama *linking*. Kode dan data dalam DLL dapat dibagi oleh beberapa aplikasi secara simultan. (225)

ECB Event control block.

EIA (***Electronic Industries Association***) Organisasi manufaktur elektronik yang meningkatkan pertumbuhan teknologi dari industri, mewakili pandangan-pandangan anggota-anggotanya, dan mengembangkan standar-standar industri. (231)

embedded system Sistem yang merupakan bagian dari sistem yang lebih besar di mana tujuan utamanya bukan untuk perhitungan, contohnya sistem komputer pada satelit atau sistem kontrol proses. (233)

emulator ① Kombinasi teknik program dan gambaran mesin khusus yang mengijinkan sistem komputasi menjalankan program-program yang ditulis untuk sistem yang berbeda. ② Program yang menjadikan komputer berlaku sebagai stasiun kerja yang dihubungkan pada sistem lain. (234)

entiti ① Perhatian konkrit atau abstrak, termasuk hubungan-hubungan hal satu dengan yang lainnya; contohnya, seseorang, objek, kejadian, atau proses perhatian dalam konteks di bawah pengertian, dan di mana data boleh disimpan dalam *database*. ② Dalam arsitektur OSI, elemen aktif dalam subsistem. Kerjasama antara entiti-entiti dalam sebuah lapisan dikendalikan oleh satu atau lebih protokol. ③ Dalam RACF, pemakai, grup, atau persediaan yang didefinisi untuk RACF; misalnya, rangkaian data DASD untuk *minidisk* VM. ④ Dalam FORTRAN, unit program, prosedur, operator, atau blok antarmuka, blok umu, unit input-output, fungsi pernyataan, tipe, variabel bernama, ekspresi, komponen tipe, konstanta simbolis, label pernyataan, konstruktor, huruf exponen, jangkauan *list*, atau kondisi. (239)

***ESR** Event service routine.

Ethernet LAN *baseband* 10-megabit yang memungkinkan berbagai stasiun mengakses medium transmisi semaunya tanpa koordinasi prioritas, menghindari tabrakan memakai *carrier sense* dan deferensi, dan menyelesaikan tabrakan memakai deteksi dan transmisi tumbukan. Ethernet memakai *carrier sense multiple access with collision detection* (CSMA/CD). (247)

event control block **(ECB)** Bagian pengatur yang dipakai untuk mewakili status suatu kejadian. (247)

****event service routine*** Prosedur jauh yang dipanggil ketika ECB telah ditangani.

FDM Frequency-division multiplexing.

file Lihat berkas.

file server Lihat pelayan berkas.

frame Lihat bingkai.

frequency-division multiplexing **(FDM)** Pembagian fasilitas transmisi ke dua atau lebih seluran dengan memisahkan lebar frekuensi yang dikirim saluran ke dalam lebar pita yang lebih sempit, masing-masing menangani saluran tersendiri. (288)

***FSP** File service process.

***GDI** Graphics device interface.

HDLC High-level data link control.

header ① Blok teks yang secara konsisten dicetak pada bagian atau satu atau lebih halaman dalam dokumen multihalaman. *Header* dapat terdiri dari informasi variabel, seperti nomor halaman. ② Informasi kontrol yang didefinisi sistem. ③ Bagian dari pesan yang berisi informasi kontrol untuk pesan itu seperti satu atau lebih *field-field* tujuan, nama dari stasiun asal, nomor urut input, string karakter yang menunjukkan tipe pesan, dan tingkat prioritas untuk pesan itu. (310)

high-level data link control **(HDLC)** Dalam komunikasi data, pemakaian serangkaian bit tertentu untuk mengontrol sambungan data. (314)

host Dalam TCP/IP, sistem yang mempunyai paling tidak satu alamat Internet yang dihubungkan padanya. Host dengan banyak *interface* jaringan memiliki banyak alamat Internet yang dihubungkan padanya. (318)

***IDP** Internetwork datagram protocol.

IEEE Institute of Electrical and Electronic Engineers. (324)

***IMP** Interface message processor.

interface Lihat antarmuka.

interface processor Lihat prosesor antarmuka.

***Internet Protocol* (IP)** Protokol yang dipakai untuk meruting data dari sumbernya ke tujuan dalam lingkungan Internet. (354)

***interrup request* (IR)** Dalam Sistem Informasi 8100 IBM, permintaan proses pada tingkat prioritas tertentu. IR dapat dijalankan program aktif, unit proses, atau peralatan I/O. (356)

interseptor Pencarian rute panggilan atau pesan yang ditempatkan untuk nomor telepon atau alamat terminal yang tidak terhubung atau tidak ada ke posisi operator atau ke terminal disain khusus. (350)

IP Internet Protocol.

***IPX** Internetworking packet exchange.

***IRQ** Interrupt ReQuest line.

ISO International Standards Organization. Organisasi badan-badan standar nasional dari berbagai negara yang dibentuk guna meningkatkan perkembangan standar-standar untuk mempermudah pertukaran barang-barang atau pelayanan-pelayanan internasional, dan mengembangkan kerja sama dalam intelektual, ilmu pengetahuan, teknologi dan aktivitas ekonomi. (360)

jaringan Susunan simpul-simpul dan koneksi cabang-cabang. (454)

jaringan bintang Konfigurasi simpul-simpul berbentuk radial atau menyerupai bintang yang dihubungkan ke pusat kontrol atau komputer di mana tiap simpul mempertukarkan data secara langsung dengan simpul pusat. (645)

jaringan bus Topologi LAN di mana hanya ada satu jalur di antara dua stasiun data dan di mana data yang dikirim suatu stasiun langsung terdapat pada semua stasiun lain pada medium transmisi yang sama. (78)

jaringan cincin ① Jaringan di mana tiap simpul hanya memiliki dua cabang yang dihubungkan padanya dan di mana hanya ada dua jalur di antara kedua simpul. ② Konfigurasi jaringan di mana paralatan dihubungkan oleh hubungan transmisi arah tunggal (*unidirectional*) membentuk jalur tertutup. (584)

jaringan data sinkronus Jaringan data yang memakai metode sinkronisasi antara DCE dan DSE, dan antara DSE-DSE. Rasio sinyal data dikontrol dengan perangkat pewaktu dalam jaringan. (668)

jaringan lup Konfig urasi jaringan di mana terdapat jalur tunggal di antara semua simpul dan di mana jalur tersebut merupakan rangkaian tertutup. (403)

jaringan multidrop Konfigurasi jaringan di mana ada satu atau lebih simpul penengah pada jalur di antara simpul pusat dan simpul ujung. (443)

jaringan pohon Jaringan di mana hanya ada satu jalur di antara dua simpul. (707)

kode sumber Input ke *compiler* atau asembler, yang ditulis dalam bahasa sumber. Kontras dengan kode objek. (634)

komunikasi data Transfer data di antara unit fungsional dalam arti transmisi menurut suatu protokol. (267)

konstanta ① Dalam bahasa pemrograman, objek bahasa yang memerlukan hanya sebuah nilai spesifik. ② Kontras dengan variabel. (140)

kontrol akses (*access control*) ① Dalam sekuriti komputer, memastikan sumber-sumber pada sistem komputer dapat diakses hanya oleh pemakai-pemakai berotoritas dalam caranya. ② Teknik

yang dipakai untuk mendirikan serangkaian stasiun-stasiun data yang ada di kontrol sementara dari medium transmisi, namun perlu dipindahkan ke tempat lain. (3)

LAN Local area network.

lapisan aplikasi Dalam model referensi OSI, lapisan yang menyediakan pengantara untuk proses-proses aplikasi yang ada dalam sistem-sistem terbuka untuk bertukar informasi dan yang berisi protokol-protokol berorientasi aplikasi yang olehnya proses-proses ini berkomunikasi. (27)

lapisan fisik Dalam model referensi OSI, lapisan yang menyediakan pengantara mekanik, listrik, fungsional, dan prosedural untuk membentuk, memelihara, dan melepaskan hubungan-hubungan fisik melalui mediun transmisi. (509)

lapisan jaringan Dalam model referensi OSI, lapisan yang menyediakan entiti-entiti dalam lapisan transpor pengantara-pengantara untuk pencarian rute dan pensaklaran blok-blok data melalui jaringan di antara sistem-sistem terbuka di mana entiti-entiti itu berada. (456)

lapisan presentasi Dalam model referensi OSI, lapisan yang menyediakan pilihan sintaks umum untuk mewakili informasi ddan untuk tranfformasi dari data aplikasi ke / dari sintaks umum ini. (522)

lapisan sambungan data Dalam model referensi OSI, lapisan yang menyediakan pelayanan-pelayanan untuk mentransfer data di antara entiti-entiti dalam lapisan jaringan melalui suatu sambungan komunikasi. Lapisan sambungan data mendeteksi dan bahkan mengkoreksi kesalahan yang dapat terjadi dalam lapisan fisik. (172)

lapisan sesi Dalam model referensi OSI, lapisan yang menyediakan pengantara-pengantara yang diperlukan untuk dua pemakai akhir untuk mengorganisasi dan mensinkronisasi dialog mereka dan mengatur pertukaran data mereka. Pelayanan ini membentuk, memelihara, dan menghentikan komunikasi. (616)

lapisan transfer Dalam model referensi OSI, lapisan yang menyediakan pelayanan transfer data ujung-ke-ujung (*end-to-end*) yang terpercaya. (707)

line driver Lihat pengendali saluran.

link level Lihat tingkat sambungan.

local area network **(LAN)** Jaringan komputer yang berlokasi pada perjanjian-perjanjian pemakai dalam daerah geografis yang terbatas. Komunikasi dalam LAN tidak akan diatur oleh peraturan-peraturan eksternal; namun komunikasi melewati batas LAN biasanya diatur beberapa bentuk aturan. (392)

log in ① Memulai sesi pada stasiun tampilan. ② Memulai sesi dengan sumber terpisah. ③ Tindakan mengidentifikasi diri sebagai yang berhak memakai sumber. Biasanya sistem meminta identitas pemakai dan *password* untuk mengecek hak pemakaian sumber. (401)

log out ① Mengakhiri suatu sesi. ② Permintaan supaya suatu sesi dihentikan. (402)

loop network Lihat jaringan lup.

***LSB** Least sig nificant bit.

***LSL** Link support layer.

MAC Medium-access control.

makro Sinonim untuk makroinstruksi. (409)

makroinstruksi ① Instruksi dalam bahasa sumber yang akan ditempati oleh urutan instruksi yang didefinisikan dalam bahasa sumber yang sama sehingga bisa juga menspesifikasi nilai-nilai untuk parameter-parameter dalam instruksi-instruksi yang ditempatkan. ② Dalam pemrograman asembler, pernyataan bahasa asembler yang membuat asembler memproses serangkaian pernyataan-pernyataan predefinisi yang dipanggil definisi makro. Pernyataan-pernyataan yang secara normal dihasilkan dari definisi makro menggantikan makroinstruksi dalam program. (409)

MAN Metropolitan area network.

***MAU** Multistation access unit.

***medium-access control* (MAC)** Untuk LAN, metode untuk menentukan perangkat mana yang memiliki akses ke medium transmisi pada tiap waktu. (426)

***metropolitan area network* (MAN)** Jaringan yang diebntuk dengan interkoneksi antara dua atau lebih jaringan yang dapat beroperasi pada kecepatan lebih tinggi dari jaringan-jaringan itu, bisa melewati batas-batas administratif, dan bisa memekai metode akses multipel. (431)

mikrokomputer ① Komputer digital yang unit prosesnya terdiri dari satu atau lebih mikroprosesor dan termasuk fasilitas penyimpanan dan input / output. (432)

mnemonik ① Simbol yang dipilih untuk membantu *user* mengingat simbol rumit. ② *Field* instruksi asembler yang berisi akronim atau singkatan untuk instruksi msein. Memakai mnemonik membebaskan pemrogram harus mengingat kode operator numerik komputer. (436)

***MSB** Most significant bit.

multidrop network Lihat jaringan multidrop.

multipleks Dalam transmisi data, suatu fungsi yang mengijinkan satu atau lebih sumber data untuk membagi medium transmisi umum sehingga tiap sumber data memiliki salurannya sendiri. (446)

multiplex interface Lihat antarmuka multipleks.

multipoint Mengacu pada komunikasi antara dua atau lebih stasiun melalui jalur telekomunikasi tunggal. (446)

***NCP** NetWare core protocol.

NetBIOS network basic input/ output system.

***network basic input / output system* (NetBIOS)** Antarmuka sistem operasi untuk program-program aplikasi yang dipakai pada PC IBM yang dihubungkan ke jaringan token-ring IBM. (453)

network layer Lihat lapisan jaringan.

***NIC** Network interface card.

***ODI** open data-link interface.

***OLE** Object linking and embedding.

***OME** Object module format.

open system Lihat sistem terbuka.

***open system interconnection* (OSI)** Interkoneksi sistem terbuka dalam kaitannya dengan standar ISO untuk pertukaran informasi. (478)

opto-elektronik ① Dihubungkan ke suatu alat yang berespon pada tenaga optik, mengirim atau mengubah radiasi optik, atau memakai radiasi optik untuk operasi internalnya. ② Alat yang berfungsi sebagai transduser listrik ke optik atau sebaliknya. (483)

OSI Open system interconnection.

PABX Private automatic branch exchange.

paket ① Dalam komunikasi data, sederetan digit biner, termasuk data dan sinyal kontrol, yang ditransmisikan dan disambungkan sebagai kesatuan. Data, sinyal kontrol dan mungkin juga, informasi kontrol kesalahan disusun dalam format khusus. ② Sinonim dengan bingkai data. (490)

papan (*board*) Panel yang terbuat dari material insulasi, berisi rangkaian-rangkaian pada satu atau kedua sisi. (69)

paralel Mengacu pada proses di mana semua kejadian terjadi dalam interval waktu yang sama, masing-masing ditangani oleh unit fungsional yang terpisah namun serupa; contohnya, transmisi paralel bit-bit dari *word* komputer melalui saluran-saluran bus internal. (498)

PC Personal computer.

pelayan berkas (*file server*) Perangkat *disk storage* kapasitas tinggi atau suatu komputer di mana tiap komputer pada jaringan dapat mengakses dan mencari berkas-berkas yang dapat dibagi di antara komputer terpasang; contohnya IBM 5170 PCAT yang dipakai untuk melayani berkas-berkas pada jaringan. (271)

pengendali Program (bisa juga berkas data) yang berisi informasi yang diperlukan untuk menjalankan unit tersendiri, seperti plotter, printer, port atau mouse. (218)

pengendali saluran Rangkaian yang meningkatkan arus sinyal untuk pengiriman data melalui kabel panjang atau ke banyak rangkaian lainnya. (218|3|)

penyambungan (*switching*) Proses menghubungkan koneksi yang disediakan dengan menutup saklar-saklar di antara terminal pengendali dan komputer. (666)

penyangga (*buffer*) ① Rutin atau memori yang digunakan untuk menyesuaikan perbedaan angka aliran data, atau waktu berlangsungnya kejadian, ketika transfer data dari perangkat satu ke yang lain. ② Rangkaian isolasi yang digunakan untuk menghindari rangkaian kondalian mempengaruhi rangkaian pengendali. (75)

periferal Berhubungan dengan operasi atau ke perangkat seperti *hard disk* yang berdiri sendiri atau kendali tip (*streaming tape driver*) yang dipakai mendukung operasi-operasi primer sebuah komputer. (505)

***personal computer* (PC)** Mikrokomputer yang terutama dimaksudkan untuk berdiri sendiri dengna pemakaian pribadi. (507)

physical layer Lihat lapisan fisik.

preprosesor ① Unit fungsional yang membuat persiapan komputasi atau organisasi. ② Program yang menguji program sumber untuk pernyataan-pernyataan preprosesor yang kemudian dieksekusi, menghasilkan program sumber lainnya. ③ Dalam emulasi, program pengkonversi data dari format sistem emulasi ke format yang diterima emulator. (521)

presentation layer Lihat lapisan presentasi.

***private automatic branch exchange* (PABX)** Pertukaran telepon pribadi yang otomatis yang menyediakan pelayanan untuk panggilan-panggilan transmisi ke / dari jaringan telepon umum. (529)

program sumber (*source program*) ① Program yang dapat diterima penerjemah tertentu. ② Serangkaian instruksi yang ditulis dalam bahasa pemrograman yang harus diterjemahkan ke bahasa mesin sebelum program dapat dijalankan. (635)

programming language Lihat bahasa pemrograman.

propagation delay Lihat waktu propagasi.

prosesor antarmuka (*interface processor*) Prosesor yang bertindak sebagai antaramuka antara prosesor atau terminal lain dan jaringan, atau prosesor yang mengontrol aliran data dalam jaringan. (351)

protokol Serangkaian aturan semantik dan sintaksis yang menentukan ciri dari unit-unit fungsional dalam melaksanakan komunikasi. (542)

RAM Random access memory.

***random access memory* (RAM)** Perangkat memori di mana data dapat ditulis dan dibaca. (554)

***read-only memory* (ROM)** ① Perangkat memori di mana data, pada kondisi normal, hanya dapat dibaca. ② Memori di mana data yang ditempatkan tidak dapat diubah pemakai kecuali pada kondisi khusus. (557)

real time ① Dalam arsitektur OSI, mengacu pada proses data oleh komputer dalam koneksi dengan proses lain di luar komputer menurut kebutuhan-kebutuhan waktu yang diminta proses luar. Istilah ini juga dipakai untuk menggambarkan sistem operasi dalam mode dan proses konversasional yang dapat dipengaruhi oleh intervensi manusia ketika mereka sedang berjalan. ② Dalam arsitektur OSI,

mengacu pada aplikasi seperti sistem pengontrol proses atau sistem instruksi berbantukan komputer di mana respons ke input kecepatannya memadai untuk mempengaruhi urutan input. (558)

ring network Lihat jaringan cincin.

***RIP** Routing information protocol.

ROM Read-only memory.

ruting Proses penentuan jalan yang dipakai untuk transmisi pesan lewat jaringan. (589)

salinan *draft* (*draft copy*) Versi dokumen yang dipersiapkan untuk peninjauan kembali, penyetujuan, atau berubahan. (218)

saluran ② Bagian dari rangkaian data di luar *data circuit-terminating equipment* (DCE), yang menghubungkan DCE ke *data switching exchange* (DSE), DCE ke DCE lainnya, atau DSE ke DSE lainnya. (383)

***SAP** Service advertising protocol.

SDLC Synchronous data link control.

semantik Hubungan antara karakter atau kelompok karakter yang mempunyai arti, bebas dalam cara penerjemahan dan pemakaiannya.(607)

serial Mengacu pada proses di mana semua kejadian terjadi satu setelah yang lainnya, contohnya, transmisi serial bit-bit karakter menurut protokol CCITT V24. (611)

sesi (*session*) ① Dalam arsitektur jaringan, untuk tujuan komunikasi data antar unit fungsional, semua aktivitas yang berlangsung selama penyelesaian, pemelibaraan, dan pelepasan koneksi. ② Koneksi logika antara dua *network accessible unit* (NAU) yang dapat diaktifkan, dibuat (*tailored*) untuk menyediakan berbagai protokol, dan di-nonaktif-kan, sesuai permintaan. Tiap sesi secara unik diidentifikasi dalam *transmission header* (TH) beserta pertukaran transmisi apapun selama sesi. ③ Periode waktu selama pemakai terminal dapat berkomunikasi dengan sistem interaktif, biasanya, waktu antara *logon* dan *logoff*.

session layer Lihat lapisan sesi.

shift Gerakan penyisipan beberapa atau semua karakter suatu kata dengan jumlah tempat karakter yang sama dalam arah ujung khusus kata itu. (621)

***shielded-twisted-pair* (STP)** Medium transmisi dari dua konduktor terpilin dengan lapisan gulungan bulat atau lurus. (621)

simultan Menunjukkan keberadaan dua atau lebih kejadian pada suatu waktu yang bersamaan. (626)

sistem terbuka ① Sistem yang karakteristik-karakteristiknya sesuai dengan standar-standar yang ada di seluruh industri dan karena itu dapat dihubungkan ke sistem-sistem lain yang sesuai dengan standar-standar serupa. ② Dalam sekuriti komputer, sistem di mana sumber-sumber yang tidak didefinisi ke sistem tidak diproteksi. Kontras dengan sistem tertutup. (478)

SNA System network architecture.

source code Lihat kode sumber.

source language Lihat bahasa sumber.

source program Lihat program sumber.

***SPX** Sequence packet exchange.

stack Daerah dalam memori yang menyimpan informasi register sementara dan mengembalikan alamat subrutin-subrutin. (643)

stand-alone Lihat berdiri sendiri.

star network Lihat jaringan bintang.

statiun kerja (*workstation*) ① Unit fungsional di mana pemakai bekerja. Suatu stasiun kerja biasanya memiliki beberapa kemampuan proses. (748)

STP Shielded-twisted-pair.

switching Lihat penyambungan.

***synchronous data link control* (SDLC)** Disiplin yang mengikuti subset dari *Advanced Data Communication Control Procedures* (ADCCP) dari *American National Standards Institute* (ANSI) dan HDLC dari ISO, untuk mengatur transfer informasi sinkronus, kode-transparan, serial-per-bit melalui koneksi sambungan. Pertukaran transmisi bisa *duplex* atau *half-duplex* melalui sambungan dengan / tanpa saklar. Konfigurasi koneksi sambunagn bisa titik-ke-titik, multititik (*multipoint*) atau lup. (668)

***system network architecture* (SNA)** Gambaran struktur, format, protokol, dan deeretan operasional logika untuk melewatkan transmisi unit-unit informasi, dan mengontrol konfigurasi dan operasi jaringan-jaringan. (676)

TCP/IP Transport Control Protocol/ Internet Protocol.

templet ① Pola penolong pemakai mengidentifikasi lokasi kunci-k unci pada papan kunci, fungsi-fungsi yang dihubungkan ke kunci-kunci pada papan kunci, atau saklar-saklar atau cahaya-cahaya pada panel kontrol. ② Baris yang dimasukkan dari papan kunci dan dimuat di memori yang dapat dipanggil, dipakai kembali, atau diubah. Lihat juga lapisan papan kunci. ③ Dalam sistem/38, string atau byte yang mudah menyebar yang mendefinisi atribut-atribut atau nilai-nilai dari objek antarmuka mesin. (685)

tingkat sambungan ① Bagian rekomendasi X.25 yang mendefi nisi protokol sambungan yang dipakai untuk mengambil data ke dalam atau keluar dari jaringan melalui sambungan *fi ull-duplex* yang menghubungkan mesin pemakai pada simpul jaringan. LAP dasn LAPB adalah protokol ak ses sambungan rekomendasi CCITT. Lihat tingkat sambungan data. ② Dalam SNA, kombinasi hubungan, protokol, perangkat, dan program transmisi yang menghubungkan simpul-simpul jaringan. (388)

***TLI** Transport layer interface.

transceiver (transmitter-receiver) Terminal yang dapat mengirim dan menerima lalu lintas data. (701)

transisi Pensaklaran dari status pertama (contohnya tegangan positif) ke yang lainnya (tegangan negatif) dalam transmisi serial. (703)

transmisi ① Pengiriman data dari satu tempat untuk penerimaan di tempat lain. ② Dalam ASCII dan komunikasi data, rangkaian karakter termasuk *heading* dan teks. (704)

transmisi asinkron (*asinchronout transmission*) Transmisi data di mana transmisi karakter atau kelompok karak ter dapat dimulai kapan pun namun di dalamnya, bit-bit yang mewakili karakter atau blok memiliki tenggang waktu yang sama. (37)

transmisi *baseband* (*baseband transmission*) Transmisi sinyal digital atau analog dalam bentuk asalnya, tidak diubah oleh modulasi. (54)

transmisi paralel Transmisi simultan dari suatu kelompok yang mewakili sebuah karakter atau bentuk data lain. (499)

transmisi serial Transmisi sekuensial elemen sinyal suatu kelompok mewakili karakter atau besaran data lain. (612)

transmisi sinkron Dalam komunikasi data, metode transmisi di mana pengiriman dan penerimaan karakter diatur oleh sinyal-sinyal pewaktuan. (669)

transmission frame Lihat bingkai transmisi.

***Transport Control Protocol / Internet Protocol* (TCP/IP)** Serangkaian protokol-protokol komunikasi yang mendukung rungsi-fungsi konek tivitas titik-ke-titik untuk baik LAN maupun WAN. (683)

transport layer Lihat lapisan transpor.