```
  File created:   19-Jan-93 10:52:26   {DSK}<python>lde>lispcore>sources>LLNS.;2

   changes to:    (RECORDS NSROUTINGINFO PACKETEXCHANGEXIP TIMEXIP)

 previous date:    5-Jan-93 00:53:11  {DSK}<python>lde>lispcore>sources>LLNS.;1

   Read Table:    INTERLISP

      Package:    INTERLISP

        Format:    XCCS
```

```
(RPAQQ LLNSCOMS
       ((COMS                                                        ; Xerox Internet Packet stuff.
              (DECLARE%: EVAL@COMPILE DONTCOPY (FILES (SOURCE)
                                                      LLNSDECLS))
              (ADDVARS * (LIST (CONS 'XIPTYPES RAWXIPTYPES)))
              (ALISTS (XIPERRORMESSAGES 1 2 3 513 514 515 516))
              (GLOBALVARS XIPTYPES XIPERRORMESSAGES))
        [COMS                                                        ; Parsing and looking up NS addresses
              (FNS PARSE-NSADDRESS COERCE-TO-NSADDRESS \COERCE.NS.SOCKET)
              (DECLARE%: DONTEVAL@LOAD DOCOPY                        ; Assign these definitions also to obsolete internal names used in
                                                                     ; earlier software
                     (P (AND (CCODEP 'PARSE-NSADDRESS)
                             (MOVD 'PARSE-NSADDRESS '\PARSE.NSADDRESSCONSTANT NIL T))
                        (AND (CCODEP 'COERCE-TO-NSADDRESS)
                             (MOVD 'COERCE-TO-NSADDRESS '\COERCE.TO.NSADDRESS NIL T]
        (COMS                                                        ; NSOCKET
              (DECLARE%: DONTCOPY (GLOBALVARS \NSOCKETS \MAX.EPKTS.ON.NSOCKET)
                     (MACROS \NSOCKET.FROM#))
              (INITRECORDS NSOCKET)
              (SYSRECORDS NSOCKET)
              (FNS OPENNSOCKET CLOSENSOCKET NSOCKETEVENT NSOCKETNUMBER NSOCKETFROMNUMBER \FLUSHNSOCQUEUE)
              (INITVARS (\NSOCKETS)
                     (\MAX.EPKTS.ON.NSOCKET 16)))
        (COMS                                                        ; assorted level 1 and 2
              (FNS \NSINIT STOPNS)
              (FNS \HANDLE.RAW.XIP \XIPERROR \FORWARD.XIP)
              (COMS (INITVARS (\NS.CHECKSUMFLG T))
                    (GLOBALVARS \NS.CHECKSUMFLG))
              (FNS GETXIP DISCARDXIPS SENDXIP SWAPXIPADDRESSES \SETXIPCHECKSUM \CLEARXIPHEADER)
              (FNS \FILLINXIP XIPAPPEND.BYTE XIPAPPEND.WORD XIPAPPEND.CELL XIPAPPEND.STRING XIPAPPEND.IFSSTRING)
              )
        (COMS                                                        ; XIP routing
              (FNS \NSGATELISTENER \HANDLE.NS.ROUTING.INFO \CANONICALIZE.NSADDRESS \ROUTE.XIP \LOCATE.NSNET
                   NSNET.DISTANCE BESTNSADDRESS SORT.NSADDRESSES.BY.DISTANCE \NSNET.CLOSERP)
              (INITVARS (\NS.ROUTING.TABLE NIL)
                     (\NS.ROUTING.TABLE.RADIUS 4)
                     (\NSROUTER.PROBECOUNT 0)
                     (\NSROUTER.PROBETIMER NIL)
                     (\NSROUTER.PROBEINTERVAL 3000)
                     (\NS.READY NIL)
                     (\NS.READY.EVENT (CREATE.EVENT "NS Ready"))
                     (\NSADDRESS.CACHE NIL))
              (ADDVARS (\SYSTEMCACHEVARS \NS.READY))
              (DECLARE%: DONTCOPY (RECORDS NSROUTINGINFO)
                     (CONSTANTS \NS.ROUTINGINFO.WORDS \XROUTINGINFO.OP.REQUEST \XROUTINGINFO.OP.RESPONSE)
                     (GLOBALVARS \NS.ROUTING.TABLE \NS.ROUTING.TABLE.RADIUS \NSROUTER.PROBECOUNT
                            \NSROUTER.PROBETIMER \NSROUTER.PROBEINTERVAL \NS.READY \NS.READY.EVENT
                            \NSADDRESS.CACHE)))
        (COMS                                                        ; Analogous to PUP stuff for tracing activity.
              (FNS XIPTRACE)
              (FNS PRINTXIP PRINTERRORXIP PRINTXIPROUTE PRINTXIPDATA)
              (INITVARS (XIPTRACEFLG)
                     (XIPTRACEFILE T)
                     (XIPTRACETIME))
              (ALISTS (XIPONLYTYPES)
                     (XIPIGNORETYPES)
                     (XIPPRINTMACROS 1 2 3 4))
              (PROP VARTYPE XIPPRINTMACROS)
              (ADDVARS (\PACKET.PRINTERS (1536 . PRINTXIP)))
              (DECLARE%: DONTCOPY (GLOBALVARS XIPTRACEFLG XIPTRACEFILE XIPIGNORETYPES XIPONLYTYPES XIPTRACETIME
                                         XIPPRINTMACROS)))
        [COMS                                                        ; Peeking
              (FNS \PEEKNS \MAYBEPEEKNS)
              (GLOBALVARS \PEEKNSNUMBER)
              (INITVARS (\PEEKNSNUMBER))
              (COMS (FNS \PROMISCUOUS.ON \PROMISCUOUS.OFF)
                    (DECLARE%: EVAL@COMPILE DONTCOPY (CONSTANTS \ETHERHOSTLOC]
        (COMS                                                        ; Simple packet exchange protocols
              (FNS \GETMISCNSOCKET CREATE.PACKET.EXCHANGE.XIP EXCHANGEXIPS RELEASE.XIP)
              [DECLARE%: DONTEVAL@LOAD DOCOPY (P (AND (CCODEP '\ALLOCATE.ETHERPACKET)
                                                      (MOVD '\ALLOCATE.ETHERPACKET 'ALLOCATE.XIP NIL T))
```

```
                                                         (AND (CCODEP '\RELEASE.ETHERPACKET)
                                                              (MOVD '\RELEASE.ETHERPACKET 'RELEASE.XIP NIL T]
              (RECORDS PACKETEXCHANGEXIP)
              (CONSTANTS (\EXTYPE.REQUEST 1)
                     (\EXTYPE.RESPONSE 2)
                     (\EXTYPE.NEGATIVE 3))
              (GLOBALVARS \MISC.NSOCKET \PACKET.EXCHANGE.CNTR)
              (INITVARS (\MISC.NSOCKET)
                     (\PACKET.EXCHANGE.CNTR 0))
              (FNS \LOOKUPPUPNUMBER))
        (COMS                                                             ; Time service
              (FNS NSNETDAYTIME0 \NS.SETTIME)
              (DECLARE%: DONTCOPY (RECORDS TIMEXIP)
                     (CONSTANTS \TIMESOCKET \TIMEOP.TIMEREQUEST \TIMEOP.TIMERESPONSE \TIMEVERSION \EXTYPE.TIME))
              )
        (COMS                                                             ; Debugging
              (FNS NS.ECHOUSER)
              (DECLARE%: DONTCOPY (CONSTANTS \XECHO.OP.REQUEST \XECHO.OP.REPLY))
              (INITVARS (\DEFAULTECHOSERVER NIL)
                     (\NS.ECHOUSERSOCKET NIL)))
        (DECLARE%: DONTEVAL@LOAD DOCOPY (P (\NSINIT)))
        (DECLARE%: EVAL@COMPILE DONTCOPY (FILES (LOADCOMP)
                                                LLETHER)
               (LOCALVARS . T))))
```

;; Xerox Internet Packet stuff.

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(FILESLOAD (SOURCE)
       LLNSDECLS)
)

(ADDTOVAR XIPTYPES (\XIPT.ROUTINGINFO 1)
                   (\XIPT.ECHO 2)
                   (\XIPT.ERROR 3)
                   (\XIPT.EXCHANGE 4)
                   (\XIPT.SPP 5)
                   (\XIPT.PUPLOOKUP 6))

(ADDTOVAR XIPERRORMESSAGES (1 "Bad checksum")
                           (2 "No socket at destination")
                           (3 "Destination congestion")
                           (513 "Gateway: Bad checksum")
                           (514 "Can't get there from here")
                           (515 "Too many hops")
                           (516 "Packet too large to forward"))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS XIPTYPES XIPERRORMESSAGES)
)
```

;; Parsing and looking up NS addresses

```
(DEFINEQ
```

(**PARSE-NSADDRESS**

```
  [LAMBDA (STR DEFAULTSOCKET)                                     ; Edited 13-Jan-88 15:36 by bvm
```

;;; If STR is a constant ether address of form net#host#socket, returns an NSADDRESS object, else NIL. DEFAULTSOCKET, if non-NIL, is socket to
;;; use if STR omits one.

```
    (if (NOT (STRINGP STR))
        then (SETQ STR (MKSTRING STR)))
    (LET* ((BASE (if (STRPOS "-" STR)
                     then 10
                   else 8))
           (MAXDIGIT (+ BASE (CHARCODE 0)
                        -1))
           NET HOST VAL NSHOST ADDR PREV10)
          (for CH instring STR do (COND
                                    [(AND (>= CH (CHARCODE 0))
                                          (<= CH MAXDIGIT))      ; Add digit into value
                                     (SETQ VAL (+ (COND
                                                    (VAL (TIMES VAL BASE))
                                                    (T 0))
                                                 (- CH (CHARCODE 0]
                                    [(EQ CH (CHARCODE %#))       ; # terminates net or host number.  Do a left shift NET _ HOST _
                                                                 ; newval
                                     (COND
                                        (NET                     ; Already have 3 parts?
                                             (RETURN NIL)))
                                     (SETQ NET HOST)
                                     (SETQ HOST (COND
                                                  (NSHOST        ; Accumulated pieces
                                                        (CONS (OR (SMALLP VAL)
                                                                  (RETURN NIL))
                                                             NSHOST))
                                                  (PREV10 (if VAL
                                                              then (+ (TIMES 1000 PREV10)
                                                                      VAL)
                                                            else
                                                                 ; Bad syntax, e.g., 0-123-#
                                                                 (RETURN NIL)))
                                                  ((NULL VAL)    ; Empty field
                                                   0)
                                                  (T VAL)))
                                      (SETQ VAL (SETQ NSHOST (SETQ PREV10 NIL]
                                     ((EQ CH (CHARCODE %.))      ; Terminates part of a 3-part host number
                                      (if (OR (NEQ BASE 8)
                                              (NOT (SMALLP VAL)))
                                          then                   ; Bad syntax
                                               (RETURN NIL)
                                        else                     ; Accumulate host pieces
                                             (push NSHOST VAL)
                                             (SETQ VAL NIL)))
                                     ((AND (EQ CH (CHARCODE -))
                                           (EQ BASE 10))         ; Decimal separator
                                      (SETQ PREV10 (if PREV10
                                                       then (+ (TIMES 1000 PREV10)
                                                               VAL)
                                                     else VAL))
                                      (SETQ VAL NIL))
                                     (T (RETURN NIL)))
              finally                                            ; Ran out of chars.  Save last value parsed, make sure we have
                                                                 ; at least a net and host
                    (RETURN (COND
                               ((AND HOST (NULL NSHOST)
                                     (NULL PREV10))              ; Must have at least a host field (at least one #), and
                                                                 ; uncompleted field (socket) must not have components
                                (SETQ ADDR (create NSADDRESS))   ; All parts start out zero
                                (replace NSSOCKET of ADDR with (OR (SMALLP (OR VAL DEFAULTSOCKET 0))
                                                                   (RETURN NIL)))
                                [COND
                                   [(LISTP HOST)                 ; Host came in a.b.c format.  Low part comes first
                                    (replace NSHNM2 of ADDR with (CAR HOST))
                                    (if (SETQ HOST (CDR HOST))
                                        then (replace NSHNM1 of ADDR with (CAR HOST))
                                             (if (SETQ HOST (CDR HOST))
                                                 then (replace NSHNM0 of ADDR with (CAR HOST))
                                                      (if (CDR HOST)
                                                          then   ; Too many pieces
                                                               (RETURN NIL]
                                   ((AND HOST (NEQ HOST 0))      ; Need to store a 48-bit number
                                    (replace NSHNM2 of ADDR with (LOGAND HOST MASKWORD1'S))
                                    (if (NEQ 0 (SETQ HOST (CL:ASH HOST -16)))
                                        then (replace NSHNM1 of ADDR with (LOGAND HOST MASKWORD1'S))
                                             (if (NEQ 0 (SETQ HOST (CL:ASH HOST -16)))
                                                 then (replace NSHNM0 of ADDR with (OR (SMALLP HOST)
                                                                                       (RETURN NIL]
                                [COND
                                   [(LISTP NET)                  ; Net in form a.b
                                    (replace NSNETLO of ADDR with (CAR NET))
                                    (if (SETQ NET (CDR NET))
                                        then (replace NSNETHI of ADDR with (CAR NET))
                                             (if (CDR NET)
                                                 then            ; Too many pieces
                                                      (RETURN NIL]
                                   ((AND NET (NEQ NET 0))        ; Store 32-bit net
                                    (replace NSNETLO of ADDR with (LOGAND NET MASKWORD1'S))
                                    (if (NEQ 0 (SETQ NET (CL:ASH NET -16)))
                                        then (replace NSNETHI of ADDR with (OR (SMALLP NET)
                                                                               (RETURN NIL]
                                ADDR])
```

(**COERCE-TO-NSADDRESS**

```
  [LAMBDA (HOST DEFAULTSOCKET)                                   ; Edited 17-Aug-92 14:21 by jds
    (CL:TYPECASE HOST
        ((OR LITATOM STRINGP) (OR (\PARSE.NSADDRESSCONSTANT (MKSTRING HOST)
                                         DEFAULTSOCKET)
                                  (\COERCE.NS.SOCKET (LOOKUP.NS.SERVER HOST)
                                         DEFAULTSOCKET)))
        (LISTP [COND
                  ((type? NSHOSTNUMBER HOST)
                   (create NSADDRESS
                          NSHOSTNUMBER _ HOST
                          NSSOCKET _ (OR DEFAULTSOCKET 0])
        (NSADDRESS (\COERCE.NS.SOCKET HOST DEFAULTSOCKET))
        (T NIL))])
```

```
(\COERCE.NS.SOCKET
  [LAMBDA (ADDR DEFAULTSOCKET)                                         (* bvm%: "15-Feb-85 01:47")

;;; If DEFAULTSOCKET is non-NIL and ADDR's socket is zero, create a new address with DEFAULTSOCKET as the socket

    (COND
       ((AND ADDR DEFAULTSOCKET (EQ (fetch NSSOCKET of ADDR)
                                    0))
        (PROG ((COPYADDR (create NSADDRESS
                                NSSOCKET _ DEFAULTSOCKET)))
              (\BLT COPYADDR ADDR (SUB1 \#WDS.NSADDRESS))
              (RETURN COPYADDR)))
       (T ADDR])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(AND (CCODEP 'PARSE-NSADDRESS)
     (MOVD 'PARSE-NSADDRESS '\PARSE.NSADDRESSCONSTANT NIL T))

(AND (CCODEP 'COERCE-TO-NSADDRESS)
     (MOVD 'COERCE-TO-NSADDRESS '\COERCE.TO.NSADDRESS NIL T))
)


;; NSOCKET

(DECLARE%: DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \NSOCKETS \MAX.EPKTS.ON.NSOCKET)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \NSOCKET.FROM# MACRO (OPENLAMBDA (SOCNUM)
                                  (for SOC in \NSOCKETS when (EQ SOCNUM (fetch ID# of SOC))
                                     do (RETURN SOC))))
)
)

(/DECLAREDATATYPE 'NSOCKET '((BITS 4)
                             POINTER WORD WORD FLAG FLAG (BITS 2)
                             POINTER WORD WORD POINTER)

       ;; ---field descriptor list elided by lister---

       '10)

(ADDTOVAR SYSTEMRECLST
          (DATATYPE NSOCKET ((NIL BITS 4)
                             (NSOCLINK POINTER)
                             (ID# WORD)
                             (NSOCHANDLE WORD)
                             (NSOC#OPENP FLAG)
                             (NSOC#CONNECTIONP FLAG)
                             (NIL BITS 2)
                             (INQUEUE POINTER)
                             (INQUEUELENGTH WORD)
                             (NSOC#ALLOCATION WORD)
                             (NSOCEVENT POINTER))))

(DEFINEQ

(OPENNSOCKET
  [LAMBDA (SKT# IFCLASH)                                               (* bvm%: " 7-Nov-85 17:53")

    ;; Creates a new local NSOCKET If SKT# is supplied, it is the identifying number (16-bit) of the socket, and an error occurs if that socket is already
    ;; in use.

    (PROG ((ID#EXPLICIT? (FIXP SKT#))
           NSOC CLASHP)
          [COND
             ((type? NSOCKET SKT#)
              (SETQ NSOC (OR (\NSOCKET.FROM# (fetch ID# of SKT#))
                             (PROGN (push \NSOCKETS SKT#)
                                    SKT#)))
              (\FLUSHNSOCQUEUE NSOC))
             (T [COND
                   ((NOT ID#EXPLICIT?)                                 ; Pick a socket that is reasonably random but won't conflict with
                                                                       ; well-known sockets
                    (SETQ SKT# (LOGOR 32768 (LOGXOR (RAND)
                                                    (fetch (FIXP LONUM) of (LOCF (fetch SECONDSCLOCK of \MISCSTATS]
                (UNINTERRUPTABLY
                    [do (COND
                           ((NOT (SETQ CLASHP (\NSOCKET.FROM# SKT#)))
                            (push \NSOCKETS (SETQ NSOC (create NSOCKET
                                                                ID# _ SKT#)))
```

```
                              (replace NSOCEVENT of NSOC with (CREATE.EVENT NSOC))
                              (RETURN))
                             [(NOT ID#EXPLICIT?)
                              (SETQ SKT# (LOGOR 32768 (ADD1 (LOGAND SKT# 32767]
                             (T (RETURN])
                 (COND
                    (CLASHP (SELECTQ IFCLASH
                                ((T ACCEPT)
                                     (\FLUSHNSOCQUEUE (SETQ NSOC CLASHP)))
                                ((DON'T FAIL)
                                     (RETURN NIL))
                                (ERROR "Socket number is already in use" SKT#]
            (RETURN NSOC])

(CLOSENSOCKET
  [LAMBDA (NSOC NOERRORFLG)                                    (* bvm%: "26-MAY-83 14:11")

    ;; Closes a local NSOCKET -- argument = T means close all sockets

    (COND
       [(EQ NSOC T)
        (while \NSOCKETS do (UNINTERRUPTABLY
                                (\FLUSHNSOCQUEUE (SETQ NSOC (pop \NSOCKETS)))
                                (replace NSOCEVENT of NSOC with NIL))]
       (T (SETQ NSOC (\DTEST NSOC 'NSOCKET))
          (UNINTERRUPTABLY
              (\FLUSHNSOCQUEUE NSOC)
              (replace NSOCEVENT of NSOC with NIL)             ; Break circular link
              (COND
                 ((FMEMB NSOC \NSOCKETS)
                  (SETQ \NSOCKETS (DREMOVE NSOC \NSOCKETS))
                  T)
                 ((NOT NOERRORFLG)
                  (ERROR NSOC "not an open NS socket"))))])

(NSOCKETEVENT
  [LAMBDA (NSOC)                                               (* bvm%: "26-MAY-83 14:14")
    (ffetch NSOCEVENT of (\DTEST NSOC 'NSOCKET])

(NSOCKETNUMBER
  [LAMBDA (NSOC)                                               (* bvm%: "10-Jun-84 16:10")
    (ffetch (NSOCKET ID#) of (\DTEST NSOC 'NSOCKET])

(NSOCKETFROMNUMBER
  [LAMBDA (SOC#)                                               (* bvm%: " 7-AUG-83 01:40")
    (\NSOCKET.FROM# SOC#])

(\FLUSHNSOCQUEUE
  [LAMBDA (NSOC)                                               (* bvm%: "11-FEB-83 12:56")
    (\FLUSH.PACKET.QUEUE (fetch (NSOCKET INQUEUE) of NSOC))
    (replace (NSOCKET INQUEUELENGTH) of NSOC with 0)
    NSOC])
)

(RPAQ? \NSOCKETS )

(RPAQ? \MAX.EPKTS.ON.NSOCKET 16)

;; assorted level 1 and 2

(DEFINEQ

(\NSINIT
  [LAMBDA (EVENT MINI)                                         ; Edited 15-Jan-88 00:30 by bvm

    ;; Enable NS network.  MINI means just enough to broadcast packets and receive answers (used by \LOOKUPPUPNUMBER)

    (for SOC in \NSOCKETS do (\FLUSHNSOCQUEUE SOC))
    (\ADD.PACKET.FILTER (FUNCTION \HANDLE.RAW.XIP))
    [PROG [(PROC (FIND.PROCESS '\NSGATELISTENER]
          (OR \LOCALNDBS (RETURN))
          (COND
             ((NULL MINI)
              [COND
                 (\3MBLOCALNDB                                 ; If we want to talk XIPs on 3mb net, we need to be able to
                                                               ; handle translations
                        (\ADD.PACKET.FILTER (FUNCTION \HANDLE.RAW.10TO3)))
                 (T (\DEL.PACKET.FILTER (FUNCTION \HANDLE.RAW.10TO3]
                                                               ; Initiate router probe to find out what our net is
              (SETQ \NS.ROUTING.TABLE (\CLEAR.ROUTING.TABLE \NS.ROUTING.TABLE))
              (SETQ \NSADDRESS.CACHE (CONS))
              (SETQ \NSROUTER.PROBECOUNT 5)
              (SETQ \NSROUTER.PROBETIMER (SETUPTIMER 0 \NSROUTER.PROBETIMER))
```

```
                                                                ; Tells gate listener to probe for routing when it gets going.
            (COND
               (\GATEWAYFLG (AND PROC (DEL.PROCESS PROC)))
               (PROC (RESTART.PROCESS PROC))
               (T (ADD.PROCESS '(\NSGATELISTENER)
                         'RESTARTABLE
                         'SYSTEM
                         'AFTEREXIT \NS.READY.EVENT)))
            (SETQ \NSFLG T]
    (SETQ \NS.READY T)
    (NOTIFY.EVENT \NS.READY.EVENT])

(STOPNS
  [LAMBDA NIL                                                   (* bvm%: "17-FEB-83 15:57")
    (\DEL.PACKET.FILTER (FUNCTION \HANDLE.RAW.XIP))
    (\DEL.PACKET.FILTER (FUNCTION \HANDLE.RAW.10TO3))
    (DEL.PROCESS '\NSGATELISTENER)
    (CLOSENSOCKET T)
    (SETQ \NSFLG NIL])
)

(DEFINEQ

(\HANDLE.RAW.XIP
  [LAMBDA (XIP TYPE)                                            ; Edited 16-Jul-90 15:54 by jds
    ;; Handles the arrival of a raw XIP.  If it is destined for a local socket that has room for it, we queue it up, else release it
    (COND
       ((EQ TYPE \EPT.XIP)
        [PROG ((MYADDR \MY.NSADDRESS)
               DESTADDR NSOC CSUM NDB DESTNET MYNET)
              [COND
                 ((NULL \NS.READY)
                  (RETURN (RELEASE.XIP XIP]
              (SETQ DESTADDR (LOCF (fetch XIPDESTNET of XIP)))  ; Treat the destination field of XIP like an NSADDRESS.  Use
                                                                ; FFETCH to avoid bogus type check
              [COND
                 ((AND (EQ (ffetch NSHNM2 of DESTADDR)
                           (ffetch NSHNM2 of MYADDR))
                       (EQ (ffetch NSHNM1 of DESTADDR)
                           (ffetch NSHNM1 of MYADDR))
                       (EQ (ffetch NSHNM0 of DESTADDR)
                           (ffetch NSHNM0 of MYADDR)))          ; Packet addressed to me

                  )
                 ((EQ (LOGAND (ffetch NSHNM0 of DESTADDR)
                              (ffetch NSHNM1 of DESTADDR)
                              (ffetch NSHNM2 of DESTADDR))
                      MASKWORD1'S)                              ; Broadcast packet--we'll have a look

                  )
                 (T                                             ; Not for us
                    (RETURN (\FORWARD.XIP XIP]
              (SETQ NDB (fetch EPNETWORK of XIP))
          ;; If it's  a packet not connected to any network, ignore it:
              (OR NDB (RETURN T))
              (SETQ MYNET (fetch NDBNSNET# of NDB))
          ;; Now check to see if the NET is reasonable.  It should always be, except when someone thinks we're a gateway
              [COND
                 ([NOT (COND
                          ((EQ (ffetch NSNETHI of DESTADDR)
                               0)                               ; Small net, easy test: take it if lo half is our number, or is zero, or
                                                                ; we are zero
                           (OR (EQ (SETQ DESTNET (ffetch NSNETLO of DESTADDR))
                                   MYNET)
                               (EQ DESTNET 0)
                               (EQ MYNET 0)))
                          [(SMALLP MYNET)                       ; Destination is small, but we're not.  Only for us if MYNET is
                                                                ; zero, in which case need to save DESTNET for later
                           (AND (EQ MYNET 0)
                                (SETQ DESTNET (ffetch NSNET of DESTADDR]
                          ((AND (EQ (ffetch NSNETLO of DESTADDR)
                                    (fetch (FIXP LONUM) of MYNET))
                                (EQ (ffetch NSNETHI of DESTADDR)
                                    (fetch (FIXP HINUM) of MYNET)))
                                                                ; Large destination is equal to us.  No need to box that
                                                                ; destination.  We are uninterested in destination otherwise
                                                                ; (except when MYNET is 0, which is handled in previous
                                                                ; clause).
                           (SETQ DESTNET MYNET]
                  ;; Packet is explicitly for a net other than us.  If we don't know our net, or packet doesn't know its, continue on.
                  (RETURN (\FORWARD.XIP XIP]
              (COND
                 [[NULL (SETQ NSOC (\NSOCKET.FROM# (fetch XIPDESTSOCKET of XIP]
                                                                  ; Packets addressed to non-active sockets are just ignored.
                  (COND
                     (XIPTRACEFLG (PRIN1 '& XIPTRACEFILE)))
                  (LET (XIPBASE)
                       (COND
                          [(AND (EQ (fetch XIPTYPE of XIP)
                                    \XIPT.ECHO)
                                (EQ (fetch XIPDESTSOCKET of XIP)
                                    \NS.WKS.Echo)
                                (EQ (\GETBASE (SETQ XIPBASE (fetch XIPCONTENTS of XIP))
                                           0)
                                    \XECHO.OP.REQUEST))           ; Play echo server
                           (COND
                              ([AND (NEQ (SETQ CSUM (fetch XIPCHECKSUM of XIP))
                                         MASKWORD1'S)
                                    (NEQ CSUM (\CHECKSUM (fetch XIPCHECKSUMBASE of XIP)
                                                     (SUB1 (FOLDHI (fetch XIPLENGTH of XIP)
                                                                   BYTESPERWORD]
                               (\XIPERROR XIP \XIPE.CHECKSUM))
                              (T (\PUTBASE XIPBASE 0 \XECHO.OP.REPLY)
                                 (SWAPXIPADDRESSES XIP)
                                 (replace EPREQUEUE of XIP with 'FREE)
                                 (SENDXIP NIL XIP]
                          (T (\XIPERROR XIP \XIPE.NOSOCKET]
                 ((IGEQ (fetch (NSOCKET INQUEUELENGTH) of NSOC)
                        (fetch (NSOCKET NSOC#ALLOCATION) of NSOC))
                                                                  ; Note that packets are just 'dropped' when the queue overflows.
                  (\XIPERROR XIP \XIPE.SOCKETFULL))
                 ([AND \NS.CHECKSUMFLG (NEQ (SETQ CSUM (fetch XIPCHECKSUM of XIP))
                                            MASKWORD1'S)
                       (NEQ CSUM (\CHECKSUM (fetch XIPCHECKSUMBASE of XIP)
                                        (SUB1 (FOLDHI (fetch XIPLENGTH of XIP)
                                                      BYTESPERWORD]
                  (\XIPERROR XIP \XIPE.CHECKSUM))
                 (T [COND
                       ((EQ DESTNET 0)                            ; Fill in unspecified destination net (possibly redundantly with
                                                                  ; zero)
                        (replace XIPDESTNET of XIP with MYNET))
                       ((EQ MYNET 0)

                        ;; Packet of specific destination net has arrived on a socket that we listen to.  If we don't know our own net number,
                        ;; assume sender is telling the truth

                        (replace NDBNSNET# of NDB with DESTNET)
                        (replace NSNET of \MY.NSADDRESS with (SETQ \MY.NSNETNUMBER DESTNET))
                        (LET [(ENTRY (OR (\LOCATE.NSNET DESTNET T)
                                         (\ADD.ROUTING.TABLE.ENTRY \NS.ROUTING.TABLE (create ROUTING
                                                                                            RTNET# _ DESTNET]
                             (replace RTHOPCOUNT of ENTRY with 0)
                             (replace RTGATEWAY# of ENTRY with NIL)
                             (replace RTNDB of ENTRY with NDB)
                             (replace RTRECENT of ENTRY with T]
                    (UNINTERRUPTABLY
                        (\ENQUEUE (fetch (NSOCKET INQUEUE) of NSOC)
                               XIP)
                        (add (fetch (NSOCKET INQUEUELENGTH) of NSOC)
                             1)
                        (NOTIFY.EVENT (fetch NSOCEVENT of NSOC)))]
        T])
```

(\XIPERROR

```
  [LAMBDA (XIP ERRCODE)                                           (* bvm%: "21-Jun-84 14:49")

;;; Turn packet around into an error packet with given error

    (COND
       ((AND (NOT (EQNSHOSTNUMBER (fetch XIPDESTHOST of XIP)
                         BROADCASTNSHOSTNUMBER))
             (NEQ (fetch XIPTYPE of XIP)
                  \XIPT.ERROR))                                   ; Don't respond to errors or to broadcasts!
        (PROG (LENGTH)
              [\BLT (LOCF (fetch ERRORXIPBODY of XIP))
                    (fetch XIPBASE of XIP)
                    (SETQ LENGTH (IMIN (fetch XIPLENGTH of XIP)
                                       (IPLUS \XIPOVLEN 12]

         ;; Copy header plus some data into data portion.  BLT is in the right direction for the overlap to work

              (replace ERRORXIPCODE of XIP with ERRCODE)
              (replace ERRORXIPARG of XIP with 0)
              (replace XIPLENGTH of XIP with (IPLUS LENGTH \XIPOVLEN (UNFOLD 2 BYTESPERWORD)))
              (replace XIPTYPE of XIP with \XIPT.ERROR)
              (SWAPXIPADDRESSES XIP)
              (replace EPREQUEUE of XIP with 'FREE)
              (SENDXIP NIL XIP)))
       (T (\RELEASE.ETHERPACKET XIP])
(\FORWARD.XIP
  [LAMBDA (XIP)                                                    (* bvm%: "12-OCT-83 15:44")
    ;; Called when we receive a XIP not addressed to us.  Unless we are a gateway, dump it
    (COND
       (\GATEWAYFLG (\GATEWAY.FORWARD.XIP XIP))
       (\PEEKNSNUMBER (\MAYBEPEEKNS XIP))
       (T (COND
             (XIPTRACEFLG (PRINTXIP XIP 'GET NIL "XIP not addressed to this host: ")))
          (\RELEASE.ETHERPACKET XIP])

)

(RPAQ? \NS.CHECKSUMFLG T)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \NS.CHECKSUMFLG)
)

(DEFINEQ

(GETXIP
  [LAMBDA (NSOC WAIT)                                              (* bvm%: "26-MAY-83 15:48")
    (PROG ([NSOCQ (ffetch (NSOCKET INQUEUE) of (\DTEST NSOC 'NSOCKET]
           EPKT TIMER)
      LP  (UNINTERRUPTABLY
              (AND (SETQ EPKT (\DEQUEUE NSOCQ))
                   (add (ffetch (NSOCKET INQUEUELENGTH) of NSOC)
                        -1)))
          (COND
             [(NULL EPKT)
              (COND
                 (WAIT (COND
                          ((EQ WAIT T))
                          [TIMER (COND
                                    ((TIMEREXPIRED? TIMER)
                                     (RETURN]
                          (T (OR (FIXP WAIT)
                                 (LISPERROR "NON-NUMERIC ARG" WAIT))
                             (SETQ TIMER (SETUPTIMER WAIT))
                             T))
                       (AWAIT.EVENT (ffetch NSOCEVENT of NSOC)
                              TIMER T)
                       (GO LP]
             [(EQ \EPT.XIP (fetch EPTYPE of EPKT))
              (AND XIPTRACEFLG (\MAYBEPRINTPACKET EPKT 'GET]
             (T (AND XIPTRACEFLG (printout XIPTRACEFILE T "Non-XIP packet " EPKT " arrived on " NSOC T))
                (SETQ EPKT)))
          (RETURN EPKT])

(DISCARDXIPS
  [LAMBDA (NSOC)                                                   (* bvm%: "11-FEB-83 12:56")
    (UNINTERRUPTABLY
        (\FLUSH.PACKET.QUEUE (fetch (NSOCKET INQUEUE) of NSOC))
        (replace (NSOCKET INQUEUELENGTH) of NSOC with 0))])

(SENDXIP
  [LAMBDA (SOCKET XIP)                                             (* bvm%: "26-OCT-83 16:31")
    ;; Returns the XIP arg iff packet can be sent;  returns a litatom explaining error otherwise
    (replace EPTYPE of XIP with \EPT.XIP)
    (SETQ XIP (\DTEST XIP 'ETHERPACKET))
    (replace XIPTCONTROL of XIP with 0)
    (until \NS.READY do (AWAIT.EVENT \NS.READY.EVENT))
    (PROG (NDB)
          (\RCLK (LOCF (fetch EPTIMESTAMP of XIP)))
          (RETURN (COND
                     ((fetch EPTRANSMITTING of XIP)
                      (AND XIPTRACEFLG (printout XIPTRACEFILE "[Put failed--packet already being transmitted]"))
                      'AlreadyQueued)
                     ((NULL (SETQ NDB (\ROUTE.XIP XIP)))
                      (AND XIPTRACEFLG (PRINTXIPROUTE XIP "[Put fails--no routing]" XIPTRACEFILE))
                      (\REQUEUE.ETHERPACKET XIP)
                      'NoRouting)
                     (T (\SETXIPCHECKSUM XIP)
                        (AND XIPTRACEFLG (\MAYBEPRINTPACKET XIP 'PUT))
                        (TRANSMIT.ETHERPACKET NDB XIP)
                        NIL])

(SWAPXIPADDRESSES
  [LAMBDA (XIP)                                                    (* bvm%: "28-Nov-83 17:59")
```

```
    (SETQ XIP (\DTEST XIP 'ETHERPACKET))
    (PROG ((NDB (\DTEST (ffetch EPNETWORK of XIP)
                      'NDB))
           (DESTSOCKET (ffetch XIPDESTSOCKET of XIP)))
          (\BLT (LOCF (ffetch XIPDESTNET of XIP))
                (LOCF (ffetch XIPSOURCENET of XIP))
                \#WDS.NSADDRESS)
          (freplace XIPSOURCESOCKET of XIP with DESTSOCKET)
          (freplace XIPSOURCENET of XIP with (ffetch NDBNSNET# of NDB))
          (freplace XIPSOURCEHOST of XIP with \MY.NSHOSTNUMBER])
```

(\SETXIPCHECKSUM

```
  [LAMBDA (XIP)                                                    (* bvm%: " 6-FEB-83 18:43")
    ;; Sets the XIPCHECKSUM field of XIP to checksum over its current contents
    (replace XIPCHECKSUM of XIP with (COND
                                       [\NS.CHECKSUMFLG (\CHECKSUM (fetch XIPCHECKSUMBASE of XIP)
                                                               (SUB1 (FOLDHI (fetch XIPLENGTH of XIP)
                                                                             BYTESPERWORD]
                                       (T \NULLCHECKSUM)))
    T])
```

(\CLEARXIPHEADER

```
  [LAMBDA (XIP)                                                    (* bvm%: "15-Feb-85 01:41")
                                                                   ; Clears the header of XIP
    (\CLEARWORDS [fetch XIPBASE of (SETQ XIP (\DTEST XIP 'ETHERPACKET]
           (FOLDHI \XIPOVLEN BYTESPERWORD])
)

(DEFINEQ
```

(\FILLINXIP

```
  [LAMBDA (TYPE SOURCENSOCKET DESTHOST DESTSOCKET# DESTNET LENGTH EPKT)
                                                                   ; Edited 13-Jan-88 15:30 by bvm
    ;; Sets indicated fields of EPKT to non-NIL args.  DESTHOST may be either an NSADDRESS or a NSHOSTNUMBER
    (PROG NIL
          (COND
             ((NULL EPKT)
              (SETQ EPKT (\ALLOCATE.ETHERPACKET))
              (replace EPTYPE of EPKT with \EPT.XIP)
              (\CLEARXIPHEADER EPKT)
              (OR LENGTH (SETQ LENGTH \XIPOVLEN)))
             (T (SETQ EPKT (\DTEST EPKT 'ETHERPACKET))
                (replace EPTYPE of EPKT with \EPT.XIP)))
          (replace XIPTCONTROL of EPKT with 0)                     ; Always zero when transmitted
          (replace XIPTYPE of EPKT with (OR TYPE 0))
          (replace XIPSOURCENSADDRESS of EPKT with (\LOCALNSADDRESS))
                                                                   ; Will put 0 in the socket field
          (AND SOURCENSOCKET (replace XIPSOURCESOCKET of EPKT with (fetch (NSOCKET ID#) of SOURCENSOCKET)))
          (replace XIPLENGTH of EPKT with (OR LENGTH \XIPOVLEN))
          [COND
             ((type? NSADDRESS DESTHOST)
              (replace XIPDESTNSADDRESS of EPKT with DESTHOST)
              (AND DESTNET (EQ (fetch NSNET of DESTHOST)
                               0)
                   (replace XIPDESTNET of EPKT with DESTNET))
              (AND DESTSOCKET# (EQ (fetch NSSOCKET of DESTHOST)
                                   0)
                   (replace XIPDESTSOCKET of EPKT with DESTSOCKET#)))
             (T [COND
                   ((type? NSHOSTNUMBER DESTHOST)                  ; Just doesn't put anything in the NET or DESTSOCKET# fields
                    (replace XIPDESTHOST of EPKT with DESTHOST)
                    (AND DESTSOCKET# (replace XIPDESTSOCKET of EPKT with DESTSOCKET#)))
                   (T (replace XIPDESTNSADDRESS of EPKT with (OR (PARSE-NSADDRESS DESTHOST DESTSOCKET#)
                                                                 (\ILLEGAL.ARG DESTHOST]
                (AND DESTNET (replace XIPDESTNET of EPKT with DESTNET]
          (RETURN EPKT])
```

(XIPAPPEND.BYTE

```
  [LAMBDA (XIP BYTE OFFSET)                                        (* bvm%: "16-FEB-83 15:09")
    ;; Make OFFSET'th byte of XIP'S data be BYTE.  OFFSET defaults to the end of the packet, in which case the length is updated
    (SETQ XIP (\DTEST XIP 'ETHERPACKET))
    (PROG [(WHERE (OR OFFSET (IDIFFERENCE (fetch XIPLENGTH of XIP)
                                    \XIPOVLEN]
          (COND
             ((IGEQ WHERE \MAX.XIPDATALENGTH)
              (RETURN)))
          (COND
             ((NOT OFFSET)
              (add (fetch XIPLENGTH of XIP)
                   1)))
          (\PUTBASEBYTE (fetch XIPCONTENTS of XIP)
                 WHERE BYTE])
```

(**XIPAPPEND.WORD**

```
  [LAMBDA (XIP WORD OFFSET)                                          (* bvm%: "16-FEB-83 15:11")

    ;; Make OFFSET'th word of XIP'S data be WORD.  OFFSET defaults to the end of the packet, in which case the length is updated

    (SETQ XIP (\DTEST XIP 'ETHERPACKET))
    (PROG (LENGTH WHERE)
          [SETQ WHERE (COND
                         (OFFSET (UNFOLD OFFSET BYTESPERWORD))
                         (T (IDIFFERENCE (SETQ LENGTH (CEIL (fetch XIPLENGTH of XIP)
                                                            BYTESPERWORD))
                                   \XIPOVLEN]
          (COND
             ((IGREATERP (IPLUS WHERE BYTESPERWORD)
                     \MAX.XIPDATALENGTH)
              (ERROR XIP "Not enough room for another word")))
          [COND
             ((NOT OFFSET)
              (replace XIPLENGTH of XIP with (IPLUS LENGTH BYTESPERWORD]
          (\PUTBASE (fetch XIPCONTENTS of XIP)
                 (FOLDLO WHERE BYTESPERWORD)
                 WORD])
```

(**XIPAPPEND.CELL**

```
  [LAMBDA (XIP CELL OFFSET)                                          (* bvm%: "16-FEB-83 15:13")

    ;; Word-aligns the beginning, and puts down two words (a 'cell' , or LONG CARDINAL).  OFFSET defaults to the end of the packet, in which case
    ;; the length is updated

    (SETQ XIP (\DTEST XIP 'ETHERPACKET))
    (PROG (LENGTH WHERE)
          [SETQ WHERE (COND
                         (OFFSET (UNFOLD OFFSET BYTESPERWORD))
                         (T (IDIFFERENCE (SETQ LENGTH (CEIL (fetch XIPLENGTH of XIP)
                                                            BYTESPERWORD))
                                   \XIPOVLEN]
          (COND
             ((IGREATERP (IPLUS WHERE BYTESPERCELL)
                     \MAX.XIPDATALENGTH)
              (ERROR XIP "Not enough room for another word")))
          [COND
             ((NOT OFFSET)
              (replace XIPLENGTH of XIP with (IPLUS LENGTH BYTESPERCELL]
          (SETQ WHERE (\ADDBASE (fetch XIPCONTENTS of XIP)
                             (FOLDLO WHERE BYTESPERWORD)))
          (\PUTBASE WHERE 0 (\HINUM CELL))
          (\PUTBASE WHERE 1 (\LONUM CELL])
```

(**XIPAPPEND.STRING**

```
  [LAMBDA (EPKT STRING OFFST IFSP)                                   (* bvm%: " 4-FEB-83 12:00")

    ;; Store STRING beginning at OFFST'th byte of XIP.  OFFST defaults to end of packet, in which case the packet's XIPLENGTH accordingly.  IFSP
    ;; means to store the string in IFS format -- the length word preceeds the string bytes.

    (OR (STRINGP STRING)
        (LITATOM STRING)
        (SETQ STRING (MKSTRING STRING)))
    (PROG ((LEN (NCHARS STRING))
           WHERE)
          (SETQ WHERE (OR OFFST (IDIFFERENCE (fetch XIPLENGTH of EPKT)
                                       \XIPOVLEN)))
          [COND
             (IFSP (SETQ WHERE (CEIL WHERE BYTESPERWORD))
                   (COND
                      ((ILESSP \MAX.XIPDATALENGTH (IPLUS WHERE LEN BYTESPERWORD))
                       (RETURN)))
                   (\PUTBASE (fetch XIPCONTENTS of EPKT)
                          (FOLDLO WHERE BYTESPERWORD)
                          LEN)
                   (add WHERE BYTESPERWORD)
                   (add LEN BYTESPERWORD))
             (T (COND
                   ((ILESSP \MAX.XIPDATALENGTH (IPLUS WHERE LEN))
                    (RETURN]
          (COND
             ((NULL OFFST)
              (add (fetch XIPLENGTH of EPKT)
                  LEN)))
          (RETURN (\PUTBASESTRING (fetch XIPCONTENTS of EPKT)
                         WHERE STRING])
```

(**XIPAPPEND.IFSSTRING**

```
  [LAMBDA (XIP STRING OFFST)                                         (* JonL "31-JUL-82 03:40")
```

```
      ;; Store STRING as an IFS string (length word followed by string) beginning at OFFST'th byte of XIP.  OFFST defaults to end of packet, in which
      ;; case the packet's XIPLENGTH is updated accordingly

      (XIPAPPEND.STRING XIP STRING OFFST T])

)

;; XIP routing

(DEFINEQ

(\NSGATELISTENER
  [LAMBDA NIL                                                    ; Edited 15-Jan-88 03:00 by bvm
    (PROG ((NSOC (OPENNSOCKET \NS.WKS.RoutingInformation T))
           (TIMER (SETUPTIMER 0))
           EVENT XIP BASE)
          (PROCESSPROP (THIS.PROCESS)
                 'INFOHOOK
                 (FUNCTION \ROUTINGTABLE.INFOHOOK))              ; For info, print our routing table
          (PROCESSPROP (THIS.PROCESS)
                 :PROTOCOL
                 'NS)
          (SETQ EVENT (fetch NSOCEVENT of NSOC))
      LP  (COND
             ((SETQ XIP (GETXIP NSOC))
              (\HANDLE.NS.ROUTING.INFO XIP)
              (BLOCK))
             ((EQ (AWAIT.EVENT EVENT (COND
                                        ((> \NSROUTER.PROBECOUNT 0)
                                         \NSROUTER.PROBETIMER)
                                        (T TIMER))
                         T)
                  EVENT)
              (GO LP)))
          (COND
             ((TIMEREXPIRED? TIMER)
              (\AGE.ROUTING.TABLE \NS.ROUTING.TABLE)
              (SETUPTIMER \RT.AGEINTERVAL TIMER)))
          [COND
             ((AND (> \NSROUTER.PROBECOUNT 0)
                   (TIMEREXPIRED? \NSROUTER.PROBETIMER))         ; Routing info desired.  Broadcast a routing request on each
                                                                 ; directly-connected net
              [SETQ XIP (\FILLINXIP \XIPT.ROUTINGINFO NSOC BROADCASTNSHOSTNUMBER \NS.WKS.RoutingInformation 0
                               (+ \XIPOVLEN BYTESPERWORD (UNFOLD \NS.ROUTINGINFO.WORDS BYTESPERWORD]
              (replace XIPFIRSTDATAWORD of XIP with \XROUTINGINFO.OP.REQUEST)
              (SETQ BASE (\ADDBASE (fetch XIPCONTENTS of XIP)
                                1))
              (replace (NSROUTINGINFO NET#) of BASE with -1)
              (replace (NSROUTINGINFO %#HOPS) of BASE with \RT.INFINITY)
              (SENDXIP NSOC XIP)
              (SETUPTIMER \NSROUTER.PROBEINTERVAL \NSROUTER.PROBETIMER)
              (SETQ \NSROUTER.PROBECOUNT (SUB1 \NSROUTER.PROBECOUNT]
          (GO LP])

(\HANDLE.NS.ROUTING.INFO
  [LAMBDA (XIP)                                                  ; Edited  1-Oct-90 09:57 by jds
                                                                 ; Processes a routing info XIP

    [COND
       ((EQ (fetch XIPFIRSTDATAWORD of XIP)
            \XROUTINGINFO.OP.RESPONSE)                           ; Unless we're a gateway, we only handle responses
        (PROG ((HOSTBASE (LOCF (fetch XIPSOURCENET of XIP)))
               (NDB (fetch EPNETWORK of XIP))
               (LENGTH (SUB1 (FOLDLO (- (fetch XIPLENGTH of XIP)
                                        \XIPOVLEN)
                                     BYTESPERWORD)))
               (BASE (\ADDBASE (fetch XIPCONTENTS of XIP)
                            1))
               (TABLE \NS.ROUTING.TABLE)
               (MASK \ROUTING.TABLE.MASK)
               (RADIUS \NS.ROUTING.TABLE.RADIUS)
               HOST ENTRY NET HOPS OLDHOPS RN BUCKET NEWTIMER)
              (COND
                 ((NOT NDB)

                  ;; Not a "real" packet -- its NDB field never got filled in, somehow.

                  (RETURN))
                 ((EQ (fetch NETTYPE of NDB)
                      10)                                        ; Host is already in about the right form.  Just canonicalize it to
                                                                 ; make the loop below faster and avoid consing
                  (SETQ HOST (\CANONICALIZE.NSADDRESS HOSTBASE)))
                 ((SETQ HOST (\TRANSLATE.10TO3 HOSTBASE NDB))    ; Host is in translation table => 3mb number

                  )
                 (T                                              ; Unknown (so far) gateway
                    (RETURN)))
              (SETQ \NSROUTER.PROBECOUNT 0)                      ; We got info from somewhere, so can stop probing
              (while (>= LENGTH \NS.ROUTINGINFO.WORDS)
                 do (SETQ HOPS (fetch (NSROUTINGINFO %#HOPS) of BASE))
                    (SETQ NET (fetch (NSROUTINGINFO NET#LO) of BASE))

                    ;; Look up this net in the routing table.  If we don't have an entry, and it's not a nearby net that we'd want to know about
                    ;; anyway, skip it.

                    ;; TABLE is a naked array containing buckets of routing entries hashed by the low bits of the net number.  Thus, we can often
                    ;; avoid dealing with non-smallp nets altogether if they happen to be uninteresting.

                    [COND
                       ((OR [AND (SETQ BUCKET (\GETBASEPTR TABLE (UNFOLD (LOGAND NET MASK)
                                                                         WORDSPERCELL)))
                                 (COND
                                    [(EQ 0 (fetch (NSROUTINGINFO NET#HI) of BASE))
                                                                  ; Easy case--specified net is smallp (NET) so can search with eq
                                     (when (EQ (fetch RTNET# of (SETQ ENTRY (CAR BUCKET)))
                                               NET)
                                        do (RETURN T) repeatwhile (SETQ BUCKET (CDR BUCKET]
                                    (T                            ; Large net--compare by low and hi to avoid boxing
                                       (when (AND (TYPENAMEP [SETQ RN (fetch RTNET# of (SETQ ENTRY (CAR BUCKET]
                                                             'FIXP)
                                                  (EQ (fetch (FIXP LONUM) of RN)
                                                      NET)
                                                  (EQ (fetch (FIXP HINUM) of RN)
                                                      (fetch (NSROUTINGINFO NET#HI) of BASE)))
                                          do (RETURN T) repeatwhile (SETQ BUCKET (CDR BUCKET]
                            (COND
                               ((<= HOPS RADIUS)
                                [\ADD.ROUTING.TABLE.ENTRY TABLE (SETQ ENTRY (create ROUTING
                                                                                   RTNET# _ (fetch (NSROUTINGINFO
                                                                                                    NET#)
                                                                                                of BASE)
                                                                                   RTTIMER _ (SETUPTIMER 0]
                                T)))

                        ;; Have an entry for this net.  Shall we accept the new info?

                        (COND
                           ((EQ (SETQ OLDHOPS (fetch RTHOPCOUNT of ENTRY))
                                0)                                ; Don't touch the directly connected net

                            )
                           ((COND
                               ((AND (EQ NDB (fetch RTNDB of ENTRY))
                                     (EQ HOST (fetch RTGATEWAY# of ENTRY)))

                                ;; Same net and gateway, so we'll want to update the hop count

                                T)
                               ((OR (NOT (fetch RTRECENT of ENTRY))
                                    (< HOPS OLDHOPS))

                                ;; Shorter route than we had, or the old route was getting out of date.  Note we only smash these fields on
                                ;; this arm of the cond, since they're unchanged on the other arm.  Smashing there would be slow, especially
                                ;; since NDB's tend to have overflowed ref counts.  Also note OLDHOPS is NIL for brand new entry, which is
                                ;; why we check RECENT first.

                                (replace RTGATEWAY# of ENTRY with HOST)
                                (replace RTNDB of ENTRY with NDB)
                                T))
                            (replace RTHOPCOUNT of ENTRY with HOPS)
                            (COND
                               ((< HOPS \RT.INFINITY)             ; Hops at infinity means inaccessible, so don't encourage this
                                                                  ; entry to stick around.
                                (replace RTRECENT of ENTRY with T)
                                (COND
                                   (NEWTIMER                      ; Save repeatedly calling the clock--everyone can get the same
                                                                  ; timer.
                                          (\BLT (fetch RTTIMER of ENTRY)
                                                NEWTIMER WORDSPERCELL))
                                   (T (SETQ NEWTIMER (SETUPTIMER \RT.TIMEOUTINTERVAL (fetch RTTIMER of ENTRY]
                    (SETQ LENGTH (- LENGTH \NS.ROUTINGINFO.WORDS))
                    (SETQ BASE (\ADDBASE BASE \NS.ROUTINGINFO.WORDS]
       (\RELEASE.ETHERPACKET XIP])
```

(\CANONICALIZE.NSADDRESS

```
  [LAMBDA (NSADDR)                                                ; Edited 15-Jan-88 01:54 by bvm

    ;; Takes an NSADDRESS or equivalent piece of storage and returns a unique NSADDRESS object for it.  Uniqueness guaranteed until the next
    ;; restart of NS.

    (for (PREVTAIL _ \NSADDRESS.CACHE)
         TAIL HOST while (SETQ TAIL (CDR PREVTAIL))
       do (SETQ HOST (CAR TAIL))
          (if (EQNSADDRESS.HOST HOST NSADDR)
              then                                                ; got it.
                   [if (NEQ PREVTAIL \NSADDRESS.CACHE)
                       then                                       ; Promote it to front to speed up next time
                            (RPLACD \NSADDRESS.CACHE (PROG1 TAIL
                                                            (RPLACD PREVTAIL (CDR TAIL))
                                                            (RPLACD TAIL (CDR \NSADDRESS.CACHE)))]
```

```
                (RETURN HOST))
          (SETQ PREVTAIL TAIL)
       finally                                                        ; Make a new entry
             (\BLT [LOCF (FETCH NSHNM0 OF (SETQ HOST (create NSADDRESS]
                   (LOCF (FETCH NSHNM0 OF NSADDR))
                   3)
             (push (CDR \NSADDRESS.CACHE)
                   HOST)
             (RETURN HOST])
```

(\ROUTE.XIP

```
  [LAMBDA (XIP READONLY)                                              ; Edited 14-Jan-88 18:46 by bvm

    ;; Encapsulates XIP, choosing the right network and immediate destination host.  Returns an NDB for the transmission.  Unless READONLY is
    ;; true, defaults source and destination nets if needed

    (PROG ((NET (fetch XIPDESTNET of XIP))
           PDH ROUTE NDB)
          (COND
             ((EQ 0 NET)
              (OR (SETQ NDB (OR \10MBLOCALNDB \3MBLOCALNDB))
                  (RETURN)))
             ((SETQ ROUTE (\LOCATE.NSNET NET))
              (SETQ NDB (fetch RTNDB of ROUTE)))
             (T (RETURN)))
          [SETQ PDH (COND
                       ((AND ROUTE (NEQ (fetch RTHOPCOUNT of ROUTE)
                                        0))                           ; Go thru this gateway
                        (fetch RTGATEWAY# of ROUTE))
                       ((EQ (fetch NETTYPE of NDB)
                            10)                                       ; Logical dest is also physical
                        (LOCF (fetch XIPDESTNET of XIP)))
                       ((EQNSHOSTNUMBER (fetch XIPDESTHOST of XIP)
                               BROADCASTNSHOSTNUMBER)                 ; On 3, broadcast goes to zero
                        0)
                       ((\TRANSLATE.10TO3 (LOCF (fetch XIPDESTNET of XIP))
                               NDB))
                       (T (RETURN]
          (replace EPNETWORK of XIP with NDB)
          (ENCAPSULATE.ETHERPACKET NDB XIP PDH (fetch XIPLENGTH of XIP)
                 \EPT.XIP)
          [COND
             ((NOT READONLY)
              [COND
                 ((EQ 0 NET)
                  (replace XIPDESTNET of XIP with (fetch NDBNSNET# of NDB]
              (replace XIPSOURCENET of XIP with (fetch NDBNSNET# of NDB]
          (RETURN NDB])
```

(\LOCATE.NSNET

```
  [LAMBDA (NET DONTPROBE)                                             ; Edited 15-Jan-88 00:23 by bvm
    (LET [(BUCKET (\GETBASEPTR \NS.ROUTING.TABLE (UNFOLD (LOGAND NET \ROUTING.TABLE.MASK)
                                                         WORDSPERCELL]
         (for DATA in BUCKET when [OR (= (fetch (ROUTING RTNET#) of DATA)
                                         NET)
                                      (AND (EQ 0 NET)
                                           (EQ 0 (fetch (ROUTING RTHOPCOUNT) of DATA]
            do (RETURN (AND (< (fetch RTHOPCOUNT of DATA)
                               \RT.INFINITY)
                            DATA))
            finally (COND
                       ((NOT DONTPROBE)                               ; Insert an entry for the net, to be purged in 30 sec if router
                                                                      ; process hasn't filled it by then
                        (\RPLPTR \NS.ROUTING.TABLE (UNFOLD (LOGAND NET \ROUTING.TABLE.MASK)
                                                          WORDSPERCELL)
                               (CONS (create ROUTING
                                            RTNET# _ NET
                                            RTHOPCOUNT _ \RT.INFINITY
                                            RTTIMER _ (SETUPTIMER 30000))
                                     BUCKET))
                        (SETQ \NSROUTER.PROBECOUNT 5)
                        (SETQ \NSROUTER.PROBETIMER (SETUPTIMER 0 \NSROUTER.PROBETIMER))
                        (WAKE.PROCESS '\NSGATELISTENER)
                        (BLOCK])
```

(NSNET.DISTANCE

```
  [LAMBDA (NET#)                                                      (* bvm%: "29-Jul-84 22:52")
    [COND
       ((type? NSADDRESS NET#)
        (SETQ NET# (fetch NSNET of NET#]
    (PROG ((ROUTE (\LOCATE.NSNET NET#)))
          [COND
             ((NULL ROUTE)
              (to 4 do (BLOCK \ETHERTIMEOUT) repeatuntil (SETQ ROUTE (\LOCATE.NSNET NET#]
          (RETURN (COND
                      (ROUTE (fetch RTHOPCOUNT of ROUTE])

(BESTNSADDRESS
  [LAMBDA (ADDRESSES ERRORSTREAM HOSTNAME)                       (* bvm%: "29-Jul-84 23:03")

    ;; Returns an NSADDRESS from the list ADDRESSES that is closest, returning NIL if there is no route.  If ERRORSTREAM = ERROR, causes
    ;; error on failure;  otherwise ERRORSTREAM is a stream to print an appropriate error message to before returning NIL.  HOSTNAME is optional
    ;; interesting name of the host being sought

    (PROG (MSG)
      RETRY
          (COND
             (ADDRESSES)
             ((SETQ ADDRESSES (LOOKUP.NS.SERVER HOSTNAME NIL T))
              (SETQ HOSTNAME (CAR ADDRESSES))
              (SETQ ADDRESSES (CDR ADDRESSES)))
             (ERRORSTREAM (SETQ MSG "Host not found")
                    (GO ERROR))
             (T (RETURN)))
          [RETURN (for TRY from 1 to 5 bind NOTLOOKEDUP HOPS BESTHOPS BESTADDR ROUTE
                      do (SETQ BESTHOPS \RT.INFINITY)
                         (SETQ NOTLOOKEDUP (SETQ BESTADDR NIL))
                         [for ADDR in ADDRESSES do (COND
                                                       ((OR [NOT (SETQ ROUTE (\LOCATE.NSNET (fetch NSNET
                                                                                                   of ADDR]
                                                            (IGEQ (SETQ HOPS (fetch RTHOPCOUNT of ROUTE))
                                                                  \RT.INFINITY))
                                                        (SETQ NOTLOOKEDUP T))
                                                       ((ILESSP HOPS BESTHOPS)
                                                        (SETQ BESTHOPS HOPS)
                                                        (SETQ BESTADDR ADDR]
                                                                     ; Enter request for routing for all hosts
                         (COND
                            ((AND BESTADDR (OR (NOT NOTLOOKEDUP)
                                               (ILEQ BESTHOPS \NS.ROUTING.TABLE.RADIUS)
                                               (IGREATERP TRY 1)))
                             (RETURN BESTADDR)))
                         (BLOCK \ETHERTIMEOUT)
                      finally (COND
                                 (ERRORSTREAM (SETQ MSG "No route to host")
                                        (GO ERROR]
      ERROR
          [OR HOSTNAME (AND ADDRESSES (SETQ HOSTNAME (fetch NSNET of (CAR ADDRESSES]
          (COND
             ((EQ ERRORSTREAM 'ERROR)
              (ERROR MSG HOSTNAME)
              (GO RETRY))
             (T (printout ERRORSTREAM T MSG ": " HOSTNAME)
                (RETURN])

(SORT.NSADDRESSES.BY.DISTANCE
  [LAMBDA (HOSTLIST)                                              (* bvm%: "22-Jun-84 18:35")
    (COND
       ((NULL (CDR (LISTP HOSTLIST)))
        HOSTLIST)
       (T                                                         ; HOSTLIST is a list each of whose elements has a
                                                                  ; NSADDRESS in its CAR and anything in its CDR.
          [for PAIR in HOSTLIST do (\LOCATE.NSNET (fetch (NSADDRESS NSNET) of (CAR PAIR]
                                                                  ; Enter request for routing for all hosts
          (BLOCK)
          (COND
             ((NOT (for PAIR in HOSTLIST always (\LOCATE.NSNET (fetch (NSADDRESS NSNET) of (CAR PAIR))
                                                        T)))
              (BLOCK \ETHERTIMEOUT)))
          (SORT HOSTLIST (FUNCTION \NSNET.CLOSERP])

(\NSNET.CLOSERP
  [LAMBDA (X Y)                                                   (* bvm%: "22-Jun-84 18:17")
    (PROG ((ROUTEX (\LOCATE.NSNET (fetch (NSADDRESS NSNET) of (CAR X))
                          T))
           ROUTEY)
          (RETURN (COND
                      ((NULL ROUTEX)
                       NIL)
                      ((SETQ ROUTEY (\LOCATE.NSNET (fetch (NSADDRESS NSNET) of (CAR Y))
                                           T))
                       (ILESSP (fetch RTHOPCOUNT of ROUTEX)
                              (fetch RTHOPCOUNT of ROUTEY)))
                      (T T])
)

(RPAQ? \NS.ROUTING.TABLE NIL)

(RPAQ? \NS.ROUTING.TABLE.RADIUS 4)
```

```
(RPAQ? \NSROUTER.PROBECOUNT 0)

(RPAQ? \NSROUTER.PROBETIMER NIL)

(RPAQ? \NSROUTER.PROBEINTERVAL 3000)

(RPAQ? \NS.READY NIL)

(RPAQ? \NS.READY.EVENT (CREATE.EVENT "NS Ready"))

(RPAQ? \NSADDRESS.CACHE NIL)

(ADDTOVAR \SYSTEMCACHEVARS \NS.READY)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

[BLOCKRECORD NSROUTINGINFO (                                    ; Format of each entry in a routing info packet
                            (NET# FIXP)
                            (%#HOPS WORD))
       (BLOCKRECORD NSROUTINGINFO ((NET#HI WORD)
                                   (NET#LO WORD]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \NS.ROUTINGINFO.WORDS 3)

(RPAQQ \XROUTINGINFO.OP.REQUEST 1)

(RPAQQ \XROUTINGINFO.OP.RESPONSE 2)

(CONSTANTS \NS.ROUTINGINFO.WORDS \XROUTINGINFO.OP.REQUEST \XROUTINGINFO.OP.RESPONSE)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \NS.ROUTING.TABLE \NS.ROUTING.TABLE.RADIUS \NSROUTER.PROBECOUNT \NSROUTER.PROBETIMER
       \NSROUTER.PROBEINTERVAL \NS.READY \NS.READY.EVENT \NSADDRESS.CACHE)
)
)
```

;; Analogous to PUP stuff for tracing activity.

```
(DEFINEQ

(XIPTRACE
  [LAMBDA (FLG REGION)                                          ; Edited 14-Jan-88 18:06 by bvm
    (MAKE-NETWORK-TRACE-WINDOW 'XIPTRACEFLG 'XIPTRACEFILE "Xerox Internet Packet Traffic" REGION FLG])

)

(DEFINEQ

(PRINTXIP
  [LAMBDA (XIP CALLER FILE PRE.NOTE DOFILTER)                   (* bvm%: "13-FEB-83 16:10")
    (OR FILE (SETQ FILE XIPTRACEFILE))
    (PROG ((TYPE (fetch XIPTYPE of XIP))
           MACRO LENGTH)
          [COND
             (DOFILTER (COND
                          ((COND
                              (XIPONLYTYPES (NOT (FMEMB TYPE XIPONLYTYPES)))
                              (XIPIGNORETYPES (FMEMB TYPE XIPIGNORETYPES)))
                           (RETURN (PRIN1 (SELECTQ CALLER
                                              ((PUT RAWPUT)
                                                   '!)
                                              ((GET RAWGET)
                                                   '+)
                                              '?)
                                         FILE]
          (AND PRE.NOTE (printout FILE T PRE.NOTE))
          (PRINTXIPROUTE XIP CALLER FILE)
          [COND
             ((SETQ MACRO (CDR (FASSOC TYPE XIPPRINTMACROS)))     ; Macro is a function to which to dispatch for the printing.
              (AND (NLISTP MACRO)
                   (RETURN (RESETFORM (OUTPUT FILE)
                                  (APPLY* MACRO XIP FILE]
          (printout FILE "Length = " .P2 (SETQ LENGTH (fetch XIPLENGTH of XIP))
                 " bytes" " (header + " .P2 (IDIFFERENCE LENGTH \XIPOVLEN)
                 ")" T "Type = ")
          (PRINTCONSTANT TYPE XIPTYPES FILE)
          (TERPRI FILE)
          (COND
             ((IGREATERP LENGTH \XIPOVLEN)                         ; MACRO tells how to print data.
              (PRIN1 "Contents: " FILE)
              (PRINTXIPDATA XIP (OR MACRO '(BYTES 12 |...|))
                     NIL FILE)))
          (TERPRI FILE)
          (RETURN XIP])
```

```
(PRINTERRORXIP
  [LAMBDA (XIP FILE)                                             (* bvm%: "15-Feb-85 01:42")
    (SETQ XIP (\DTEST XIP 'ETHERPACKET))
    (PROG ((ERRCODE (fetch ERRORXIPCODE of XIP))
           (ERRARG (fetch ERRORXIPARG of XIP)))
          [printout FILE "[Error] " (OR (CADR (ASSOC ERRCODE XIPERRORMESSAGES))
                                       (CONCAT '%# (OCTALSTRING ERRCODE]
          (COND
             ((NEQ ERRARG 0)
              (printout FILE ", Parameter " .P2 ERRARG)))
          (TERPRI FILE])
```

```
(PRINTXIPROUTE
  [LAMBDA (PACKET CALLER FILE)                                   (* bvm%: "15-Feb-85 01:42")
    (FRESHLINE FILE)
    (AND CALLER (printout FILE CALLER ":  "))
    (PROG ((CONTROL (fetch XIPTCONTROL of PACKET))
           CSECS)
          (printout FILE "From " (\PRINTNSADDRESS (LOCF (fetch (XIP XIPSOURCENET) of PACKET))
                                       FILE)
                 " to "
                 (\PRINTNSADDRESS (LOCF (fetch (XIP XIPDESTNET) of PACKET))
                        FILE))
          (COND
             ((NEQ CONTROL 0)
              (printout FILE ", Hops = " .P2 CONTROL)))
          (COND
             (XIPTRACETIME (printout FILE " [" .I4 (IQUOTIENT (SETQ CSECS (\CENTICLOCK PACKET))
                                                             100)
                                  '%. .I2..T (IREMAINDER CSECS 100)
                                  "]")))
          (TERPRI FILE])
```

```
(PRINTXIPDATA
  [LAMBDA (XIP MACRO OFFSET FILE)                                ; Edited 13-Jan-88 15:17 by bvm
```

;;; Prints DATA part of XIP starting at OFFSET (Default zero) according to MACRO. MACRO contains elements describing what format the data is in
;;; (see PRINTPACKETDATA)

```
    (PRINTPACKETDATA (fetch XIPCONTENTS of XIP)
           OFFSET MACRO (- (fetch XIPLENGTH of XIP)
                           \XIPOVLEN)
           FILE])

)

(RPAQ? XIPTRACEFLG )

(RPAQ? XIPTRACEFILE T)

(RPAQ? XIPTRACETIME )

(ADDTOVAR XIPONLYTYPES )

(ADDTOVAR XIPIGNORETYPES )

(ADDTOVAR XIPPRINTMACROS (1 "Operation = " WORDS 2 "Info: " |...|)
                         (2 "Operation: " WORDS 2 "Data: " CHARS 100 |...|)
                         (3 . PRINTERRORXIP)
                         (4 "ID = " INTEGER 4 "Type = " WORDS 6 BYTES 8))

(PUTPROPS XIPPRINTMACROS VARTYPE ALIST)

(ADDTOVAR \PACKET.PRINTERS (1536 . PRINTXIP))

(DECLARE%: DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS XIPTRACEFLG XIPTRACEFILE XIPIGNORETYPES XIPONLYTYPES XIPTRACETIME XIPPRINTMACROS)
)
)
```

;; Peeking

```
(DEFINEQ

(\PEEKNS
  [LAMBDA (HOST FILE)                                              ; Edited 13-Jan-88 15:18 by bvm
    (PROG NIL
          [COND
             ((NULL HOST)
              (\PROMISCUOUS.OFF)
              (RPTQ 20 (BLOCK))                                    ; empty the pipe
              (SETQ \PEEKNSNUMBER))
             (T (COND
                   ((EQ HOST T)
                    (SETQ \PEEKNSNUMBER T))
                   ((SETQ HOST (COERCE-TO-NSADDRESS HOST))
                    (SETQ \PEEKNSNUMBER (fetch NSHOSTNUMBER of HOST)))
                   (T (RETURN NIL)))                               ; Now make us promiscuous
                (\PROMISCUOUS.ON)
                [COND
                   (FILE (SETQ XIPTRACEFILE (OR (OPENP FILE 'OUTPUT)
                                                (OPENFILE FILE 'OUTPUT]
                (OR XIPTRACEFLG (SETQ XIPTRACEFLG T]
          (RETURN \PEEKNSNUMBER])
```

(\MAYBEPEEKNS

```
  [LAMBDA (XIP)                                                    (* bvm%: "12-OCT-83 16:25")
    [COND
       ((AND \PEEKNSNUMBER XIPTRACEFLG)
        (PROG (DIRECTION)
              (COND
                 ([OR (EQ \PEEKNSNUMBER T)
                      (AND (EQNSHOSTNUMBER (fetch XIPDESTHOST of XIP)
                                  BROADCASTNSHOSTNUMBER)
                           (NEQ \PEEKNSNUMBER 0))
                      [COND
                         ((EQNSHOSTNUMBER (fetch XIPSOURCEHOST of XIP)
                                 \PEEKNSNUMBER)
                          (SETQ DIRECTION 'PUT]
                      (COND
                         ((EQNSHOSTNUMBER (fetch XIPDESTHOST of XIP)
                                 \PEEKNSNUMBER)
                          (SETQ DIRECTION 'GET]
                  (PRINTXIP XIP DIRECTION XIPTRACEFILE NIL T]
    (\RELEASE.ETHERPACKET XIP])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \PEEKNSNUMBER)
)

(RPAQ? \PEEKNSNUMBER )

(DEFINEQ
```

(\PROMISCUOUS.ON

```
  [LAMBDA NIL                                                      (* bvm%: "12-OCT-83 15:58")
    (SELECTQ (fetch NETTYPE of \LOCALNDBS)
         (3 (\PUTBASE (EMADDRESS \ETHERHOSTLOC)
                   0 0))
         (10 (\10MB.STARTDRIVER \LOCALNDBS T BROADCASTNSHOSTNUMBER))
         NIL])
```

(\PROMISCUOUS.OFF

```
  [LAMBDA NIL                                                      (* bvm%: "12-OCT-83 15:58")
    (SELECTQ (fetch NETTYPE of \LOCALNDBS)
         (3 (\PUTBASE (EMADDRESS \ETHERHOSTLOC)
                   0
                   (fetch NDBPUPHOST# of \LOCALNDBS)))
         (10 (\10MB.STARTDRIVER \LOCALNDBS T T))
         NIL])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ \ETHERHOSTLOC 392)

(CONSTANTS \ETHERHOSTLOC)
)
)
```

;; Simple packet exchange protocols

```
(DEFINEQ
(\GETMISCNSOCKET
  [LAMBDA NIL                                                    (* bvm%: "24-FEB-83 17:51")
    ;; Opens a socket for miscellaneous services, if we don't have it open yet
    (COND
       ((AND \MISC.NSOCKET (FMEMB \MISC.NSOCKET \NSOCKETS))
        \MISC.NSOCKET)
       (T (SETQ \MISC.NSOCKET (OPENNSOCKET])

(CREATE.PACKET.EXCHANGE.XIP
  [LAMBDA (NSOCKET DESTHOST DESTSOCKET TYPE)                     (* bvm%: "15-Jun-84 12:54")
    (PROG [(XIP (\FILLINXIP \XIPT.EXCHANGE NSOCKET DESTHOST DESTSOCKET 0 (IPLUS \XIPOVLEN (UNFOLD 3 BYTESPERWORD]
          (replace (PACKETEXCHANGEXIP PACKETEXCHANGETYPE) of XIP with TYPE)
          (replace (PACKETEXCHANGEXIP PACKETEXCHANGEID0) of XIP with 0)
          [replace (PACKETEXCHANGEXIP PACKETEXCHANGEID1) of XIP with (SETQ \PACKET.EXCHANGE.CNTR
                                                                          (\LOLOC (\ADDBASE \PACKET.EXCHANGE.CNTR 1]
          (RETURN XIP])

(EXCHANGEXIPS
  [LAMBDA (SOC OUTXIP IDFILTER TIMEOUT)                          (* bvm%: "12-Jun-84 15:15")
    ;; Sends out OUTXIP on SOC and waits for a reply, which it puts in INXIP.  If IDFILTER is true, only replies with the same ID are accepted.
    ;; Returns input pup on success, or NIL on failure.  TIMEOUT overrides the default timeout.
    (OR TIMEOUT (SETQ TIMEOUT \ETHERTIMEOUT))
    (DISCARDXIPS SOC)                                            ; Flush any pups waiting on this socket
    (SENDXIP SOC OUTXIP)
    (bind INXIP (TIMER _ (SETUPTIMER TIMEOUT)) until (TIMEREXPIRED? TIMER)
       do (\BACKGROUND)
          (COND
             ([AND (SETQ INXIP (GETXIP SOC))
                   (OR (NOT IDFILTER)
                       (IEQP (fetch PACKETEXCHANGEID of INXIP)
                             (fetch PACKETEXCHANGEID of OUTXIP]
              (RETURN INXIP])

(RELEASE.XIP
  [LAMBDA (XIP)                                                  (* bvm%: "24-FEB-83 18:08")
    (\RELEASE.ETHERPACKET XIP])
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(AND (CCODEP '\ALLOCATE.ETHERPACKET)
     (MOVD '\ALLOCATE.ETHERPACKET 'ALLOCATE.XIP NIL T))

(AND (CCODEP '\RELEASE.ETHERPACKET)
     (MOVD '\RELEASE.ETHERPACKET 'RELEASE.XIP NIL T))
)

(DECLARE%: EVAL@COMPILE

[ACCESSFNS PACKETEXCHANGEXIP ((PEXBASE (fetch (XIP XIPCONTENTS) of DATUM)))
       (BLOCKRECORD PEXBASE ((PACKETEXCHANGEID FIXP)             ; Arbitrary id in packet exchange XIP
                             (PACKETEXCHANGETYPE WORD)           ; Protocol-specific type
                             (PACKETEXCHANGEBODY0 WORD)          ; Body starts here

                             ))
       (BLOCKRECORD PEXBASE ((PACKETEXCHANGEID0 WORD)
                             (PACKETEXCHANGEID1 WORD)))
       (ACCESSFNS PACKETEXCHANGEXIP ((PACKETEXCHANGEBODY (LOCF (fetch PACKETEXCHANGEBODY0 of DATUM]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \EXTYPE.REQUEST 1)

(RPAQQ \EXTYPE.RESPONSE 2)

(RPAQQ \EXTYPE.NEGATIVE 3)

(CONSTANTS (\EXTYPE.REQUEST 1)
       (\EXTYPE.RESPONSE 2)
       (\EXTYPE.NEGATIVE 3))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \MISC.NSOCKET \PACKET.EXCHANGE.CNTR)
)

(RPAQ? \MISC.NSOCKET )

(RPAQ? \PACKET.EXCHANGE.CNTR 0)
```

```
(DEFINEQ

(\LOOKUPPUPNUMBER
  [LAMBDA (NSNUMBER)                                              (* ejs%: "29-Dec-85 15:13")

;;; Looks up the pup host number for NSNUMBER.  These numbers are in gateway's database

    (PROG ((SOC (\GETMISCNSOCKET))
           OXIP RESULT)
          (SETQ OXIP (CREATE.PACKET.EXCHANGE.XIP SOC BROADCASTNSHOSTNUMBER \NS.WKS.PUPLOOKUP \EXTYPE.REQUEST))
          (replace XIPTYPE of OXIP with \XIPT.PUPLOOKUP)
          (add (fetch XIPLENGTH of OXIP)
               (UNFOLD \#WDS.NSHOSTNUMBER BYTESPERWORD))
          (\STORENSHOSTNUMBER (fetch PACKETEXCHANGEBODY of OXIP)
                 NSNUMBER)
          (DISCARDXIPS SOC)
          (to \MAXETHERTRIES bind INXIP TIMER do (SENDXIP SOC OXIP)
                                                  (SETQ TIMER (SETUPTIMER \ETHERTIMEOUT TIMER))
             repeatuntil (do (BLOCK)
                             (COND
                                [(NULL (SETQ INXIP (GETXIP SOC]
                                ((IEQP (fetch PACKETEXCHANGEID of INXIP)
                                       (fetch PACKETEXCHANGEID of OXIP))
                                 (SELECTC (fetch PACKETEXCHANGETYPE of INXIP)
                                     (\EXTYPE.RESPONSE
                                          (RETURN (PROGN (SETQ RESULT (fetch PACKETEXCHANGEBODY0 of INXIP))
                                                         (COND
                                                            ((AND (EQ 0 (LOGAND RESULT 255))
                                                                  (NOT (EQUAL NSNUMBER BROADCASTNSHOSTNUMBER)))
                                                             (COND
                                                                (XIPTRACEFLG (printout XIPTRACEFILE "Impossible NS
                                                                                    to Pup translation: " RESULT T
                                                                                    )))
                                                             (SETQ RESULT NIL)))
                                                         (RELEASE.XIP INXIP)
                                                         RESULT)))
                                     (\EXTYPE.NEGATIVE
                                          (COND
                                             (XIPTRACEFLG (printout XIPTRACEFILE
                                                                  [\GETBASESTRING (fetch PACKETEXCHANGEBODY
                                                                                     of INXIP)
                                                                         0
                                                                         (IDIFFERENCE (fetch XIPLENGTH of INXIP)
                                                                                (IPLUS \XIPOVLEN (UNFOLD 3
                                                                                                  BYTESPERWORD
                                                                                                  ]
                                                                  T)))
                                                                    ; For now, ignore negative responses.  some gateways are
                                                                    ; confused

                                          )
                                     NIL)
                                 (RELEASE.XIP INXIP))
                                (T (RELEASE.XIP INXIP)))
                           repeatuntil (TIMEREXPIRED? TIMER)))
          [COND
             (XIPTRACEFLG (COND
                             ((NULL RESULT)
                              (printout XIPTRACEFILE "NS to Pup number lookup failed" T))
                             (T (printout XIPTRACEFILE "Local pup net/host set to " (PORTSTRING RESULT)
                                       T]
          (RELEASE.XIP OXIP)
          (RETURN RESULT])
)

;; Time service

(DEFINEQ

(NSNETDAYTIME0
  [LAMBDA NIL                                                     (* bvm%: "15-Jun-84 12:41")

;;; Returns a 32-bit unsigned alto time from the network, if possible

    (PROG ((SOC (\GETMISCNSOCKET))
           OXIP RESULT IXIP)
          (SETQ OXIP (CREATE.PACKET.EXCHANGE.XIP SOC BROADCASTNSHOSTNUMBER \TIMESOCKET \EXTYPE.TIME))
          (replace TIMEOP of OXIP with \TIMEOP.TIMEREQUEST)
          (replace TIMEVERSION of OXIP with \TIMEVERSION)
          (add (fetch XIPLENGTH of OXIP)
               (UNFOLD 2 BYTESPERWORD))
          (RETURN (to \MAXETHERTRIES when (SETQ IXIP (EXCHANGEXIPS SOC OXIP T))
                     do (SELECTC (fetch TIMEOP of IXIP)
                            (\TIMEOP.TIMERESPONSE
                                 (RETURN (fetch TIMEVALUE of IXIP)))
                              NIL])

(\NS.SETTIME
  [LAMBDA (RETFLG)                                            (* bvm%: "15-Feb-85 01:42")

;;; Sets the time from an NS time server if possible.  Returns T on success

    (PROG ((SOC (\GETMISCNSOCKET))
           OXIP RESULT IXIP TIME)
          (SETQ OXIP (CREATE.PACKET.EXCHANGE.XIP SOC BROADCASTNSHOSTNUMBER \TIMESOCKET \EXTYPE.TIME))
          (replace TIMEOP of OXIP with \TIMEOP.TIMEREQUEST)
          (replace TIMEVERSION of OXIP with \TIMEVERSION)
          (add (fetch XIPLENGTH of OXIP)
               (UNFOLD 2 BYTESPERWORD))
          (RETURN (to \MAXETHERTRIES when (SETQ IXIP (EXCHANGEXIPS SOC OXIP T))
                     do (SELECTC (fetch (TIMEXIP TIMEOP) of IXIP)
                              (\TIMEOP.TIMERESPONSE
                                  (SETQ TIME (create FIXP
                                                    HINUM _ (fetch TIMEVALUEHI of IXIP)
                                                    LONUM _ (fetch TIMEVALUELO of IXIP)))
                                  (COND
                                     (RETFLG (RETURN TIME)))
                                  (SETQ \TimeZoneComp (ITIMES (COND
                                                                 ((EQ (fetch TIMEZONESIGN of IXIP)
                                                                      0)
                                                                  1)
                                                                 (T -1))
                                                             (fetch TIMEZONEHOURS of IXIP)))
                                  (SETQ \BeginDST (fetch TIMEBEGINDST of IXIP))
                                  (SETQ \EndDST (fetch TIMEENDDST of IXIP))
                                  (\SETNEWTIME0 TIME)
                                  (RETURN T))
                              NIL])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

[ACCESSFNS TIMEXIP ((TIMEBODY (fetch (PACKETEXCHANGEXIP PACKETEXCHANGEBODY) of DATUM)))
       (BLOCKRECORD TIMEBODY ((TIMEVERSION WORD)                   ; Protocol version
                              (TIMEOP WORD)                        ; What kind of request/response
                              (TIMEVALUE FIXP)
                              (TIMEZONESIGN WORD)                  ; 0 = west of prime meridian, 1 = east
                              (TIMEZONEHOURS WORD)                 ; Hours from prime meridian
                              (TIMEZONEMINUTES WORD)               ; Minutes ...
                              (TIMEBEGINDST WORD)                  ; Day of year when DST starts
                              (TIMEENDDST WORD)                    ; Day of year when DST stops

                              )
              (BLOCKRECORD TIMEBODY ((NIL 2 WORD)
                                     (TIMEVALUEHI WORD)
                                     (TIMEVALUELO WORD]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \TIMESOCKET 8)

(RPAQQ \TIMEOP.TIMEREQUEST 1)

(RPAQQ \TIMEOP.TIMERESPONSE 2)

(RPAQQ \TIMEVERSION 2)

(RPAQQ \EXTYPE.TIME 1)

(CONSTANTS \TIMESOCKET \TIMEOP.TIMEREQUEST \TIMEOP.TIMERESPONSE \TIMEVERSION \EXTYPE.TIME)
)
)

;; Debugging

(DEFINEQ

(NS.ECHOUSER
  [LAMBDA (ECHOHOST ECHOSTREAM INTERVAL NTIMES)               ; Edited 13-Jan-88 15:26 by bvm
    (RESETLST
        [PROG ((TIMER (SETUPTIMER 0))
               (ECHOADDRESS (OR (COERCE-TO-NSADDRESS ECHOHOST \NS.WKS.Echo)
                                (\ILLEGAL.ARG ECHOHOST)))
               NSOC OXIP IXIP EVENT I XIPBASE ECHOXIPLENGTH OXIPBASE)
              [RESETSAVE NIL (LIST 'CLOSENSOCKET (SETQ NSOC (OPENNSOCKET]
              (SETQ OXIP (\FILLINXIP \XIPT.ECHO NSOC ECHOADDRESS))
              (\PUTBASE (SETQ OXIPBASE (fetch XIPCONTENTS of OXIP))
                     0 \XECHO.OP.REQUEST)
              (\PUTBASE OXIPBASE 1 (SETQ I 1))
              [replace XIPLENGTH of OXIP with (SETQ ECHOXIPLENGTH (+ \XIPOVLEN (TIMES 2 BYTESPERWORD]
              (OR INTERVAL (SETQ INTERVAL 1000))
              (OR NTIMES (SETQ NTIMES 1000))
              (printout ECHOSTREAM "Echoing to " ECHOADDRESS T)
              (SETQ ECHOSTREAM (GETSTREAM (OR ECHOSTREAM T)
                                      'OUTPUT))
              (SETQ EVENT (fetch NSOCEVENT of NSOC))
          LP  (SENDXIP NSOC OXIP)
              (PRIN1 '! ECHOSTREAM)
              (SETUPTIMER INTERVAL TIMER)
              (do (COND
                     [(SETQ IXIP (GETXIP NSOC))
                      (COND
                         ((PROG1 (SELECTC (fetch XIPTYPE of IXIP)
                                      (\XIPT.ECHO (COND
                                                     ((OR (NEQ (fetch XIPLENGTH of IXIP)
                                                               ECHOXIPLENGTH)
                                                          (NEQ (\GETBASE (SETQ XIPBASE (fetch XIPCONTENTS
                                                                                          of IXIP))
                                                                      0)
                                                               \XECHO.OP.REPLY))
                                                      (PRIN1 '? ECHOSTREAM)
                                                      NIL)
                                                     ((= (\GETBASE XIPBASE 1)
                                                         I)
                                                      (PRIN1 '+ ECHOSTREAM))
                                                     (T (PRIN1 "(late)" ECHOSTREAM)
                                                        NIL)))
                                      (\XIPT.ERROR (PRINTERRORXIP IXIP ECHOSTREAM)
                                                   NIL)
                                      (PROGN (PRIN1 '? ECHOSTREAM)
                                             NIL))
                                 (RELEASE.XIP IXIP))
                          (RETURN]
                     (T (AWAIT.EVENT EVENT TIMER T)))
                 repeatuntil (TIMEREXPIRED? TIMER) finally (COND
                                                              ((fetch EPTRANSMITTING of OXIP)
                                                               (PRIN1 "[not yet transmitted; maybe transmitter is
                                                                 off]" ECHOSTREAM)))
                                                       (PRIN1 '%. ECHOSTREAM))
              (COND
                 ((> (OR (EQ NTIMES T)
                         (add NTIMES -1))
                     0)
                  (\PUTBASE OXIPBASE 1 (add I 1))
                  (GO LP])])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ \XECHO.OP.REQUEST 1)

(RPAQQ \XECHO.OP.REPLY 2)

(CONSTANTS \XECHO.OP.REQUEST \XECHO.OP.REPLY)
)
)

(RPAQ? \DEFAULTECHOSERVER NIL)

(RPAQ? \NS.ECHOUSERSOCKET NIL)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(\NSINIT)
)

(DECLARE%: EVAL@COMPILE DONTCOPY

(FILESLOAD (LOADCOMP)
       LLETHER)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)
)

(PUTPROPS LLNS COPYRIGHT ("Venue & Xerox Corporation" 1983 1984 1985 1986 1988 1989 1990 1991 1992 1993))
```

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

RECORD INDEX

MACRO INDEX

PROPERTY INDEX