PROTOCOL STANDARD FOR A NetBIOS SERVICE
ON A TCP/UDP TRANSPORT:
CONCEPTS AND METHODS

ABSTRACT

This RFC defines a proposed standard protocol to support NetBIOS
services in a TCP/IP environment. Both local network and internet
operation are supported. Various node types are defined to accommodate
local and internet topologies and to allow operation with or without the
use of IP broadcast.

This RFC describes the NetBIOS-over-TCP protocols in a general manner,
emphasizing the underlying ideas and techniques. Detailed
specifications are found in a companion RFC, "Protocol Standard For a
NetBIOS Service on a TCP/UDP Transport: Detailed Specifications".

SUMMARY OF CONTENTS

TABLE OF CONTENTS

                    PROTOCOL STANDARD FOR A NetBIOS SERVICE
                         ON A TCP/UDP TRANSPORT:
                          CONCEPTS AND METHODS

1. STATUS OF THIS MEMO

    This RFC specifies a proposed standard for the Internet
    community. Since this topic is new to the Internet community,
    discussions and suggestions are specifically requested.

    Please send written comments to:

            Karl Auerbach
            Epilogue Technology Corporation
            P.O. Box 5432
            Redwood City, CA   94063

    Please send online comments to:

            Avnish Aggarwal
                    Internet: mtxinu!excelan!avnish@ucbvax.berkeley.edu
                    Usenet: ucbvax!mtxinu!excelan!avnish

    Distribution of this document is unlimited.

2. ACKNOWLEDGEMENTS

    This RFC has been developed under the auspices of the Internet
    Activities Board, especially the End-to-End Services Task Force.

    The following individuals have contributed to the development of
    this RFC:

Avnish Aggarwal   Arvind Agrawal   Lorenzo Aguilar
Geoffrey Arnold   Karl Auerbach   K. Ramesh Babu
Keith Ball   Amatzia Ben-Artzi   Vint Cerf
Richard Cherry   David Crocker   Steve Deering
Greg Ennis   Steve Holmgren   Jay Israel
David Kaufman   Lee LaBarre   James Lau
Dan Lynch   Gaylord Miyata   David Stevens
Steve Thomas   Ishan Wu

    The system proposed by this RFC does not reflect any existing
    Netbios-over-TCP implementation. However, the design
    incorporates considerable knowledge obtained from prior
    implementations. Special thanks goes to the following
    organizations which have provided this invaluable information:

    CMC/Syros   Excelan   Sytek   Ungermann-Bass

3. INTRODUCTION

   This RFC describes the ideas and general methods used to provide
   NetBIOS on a TCP and UDP foundation. A companion RFC, "Protocol
   Standard For a NetBIOS Service on a TCP/UDP Transport: Detailed
   Specifications"[1] contains detailed descriptions of packet
   formats, protocols, and defined constants and variables.

   The NetBIOS service has become the dominant mechanism for
   personal computer networking. NetBIOS provides a vendor
   independent interface for the IBM Personal Computer (PC) and
   compatible systems.

   NetBIOS defines a software interface not a protocol. There is no
   "official" NetBIOS service standard. In practice, however, the
   IBM PC-Network version is used as a reference. That version is
   described in the IBM document 6322916, "Technical Reference PC
   Network"[2].

   Protocols supporting NetBIOS services have been constructed on
   diverse protocol and hardware foundations. Even when the same
   foundation is used, different implementations may not be able to
   interoperate unless they use a common protocol. To allow NetBIOS
   interoperation in the Internet, this RFC defines a standard
   protocol to support NetBIOS services using TCP and UDP.

   NetBIOS has generally been confined to personal computers to
   date. However, since larger computers are often well suited to
   run certain NetBIOS applications, such as file servers, this
   specification has been designed to allow an implementation to be
   built on virtually any type of system where the TCP/IP protocol
   suite is available.

   This standard defines a set of protocols to support NetBIOS
   services.

   These protocols are more than a simple communications service
   involving two entities. Rather, this note describes a
   distributed system in which many entities play a part even if
   they are not involved as an end-point of a particular NetBIOS
   connection.

   This standard neither constrains nor determines how those
   services are presented to application programs.

   Nevertheless, it is expected that on computers operating under
   the PC-DOS and MS-DOS operating systems that the existing NetBIOS
   interface will be preserved by implementors.

   NOTE: Various symbolic values are used in this document. For
         their definitions, refer to the Detailed Specifications[1].

4. DESIGN PRINCIPLES

   In order to develop the specification the following design principles
   were adopted to guide the effort. Most are typical to any protocol
   standardization effort; however, some have been assigned priorities
   that may be considered unusual.

4.1. PRESERVE NetBIOS SERVICES

   In the absence of an "official" standard for NetBIOS services, the
   version found in the IBM PC Network Technical Reference[2] is used.

   NetBIOS is the foundation of a large body of existing applications.
   It is desirable to operate these applications on TCP networks and to
   extend them beyond personal computers into larger hosts. To support
   these applications, NetBIOS on TCP must closely conform to the
   services offered by existing NetBIOS systems.

   IBM PC-Network NetBIOS contains some implementation specific
   characteristics. This standard does not attempt to completely
   preserve these. It is certain that some existing software requires
   these characteristics and will fail to operate correctly on a NetBIOS
   service based on this RFC.

4.2. USE EXISTING STANDARDS

   Protocol development, especially with standardization, is a demanding
   process. The development of new protocols must be minimized.

   It is considered essential that an existing standard which provides
   the necessary functionality with reasonable performance always be
   chosen in preference to developing a new protocol.

   When a standard protocol is used, it must be unmodified.

4.3. MINIMIZE OPTIONS

   The standard for NetBIOS on TCP should contain few, if any, options.

   Where options are included, the options should be designed so that
   devices with different option selections should interoperate.

4.4. TOLERATE ERRORS AND DISRUPTIONS

   NetBIOS networks typically operate in an uncontrolled environment.
   Computers come on-line at arbitrary times. Computers usually go
   off-line without any notice to their peers. The software is often
   operated by users who are unfamiliar with networks and who may
   randomly perturb configuration settings.

   Despite this chaos, NetBIOS networks work. NetBIOS on TCP must also

be able to operate well in this environment.

Robust operation does not necessarily mean that the network is proof
against all disruptions. A typical NetBIOS network may be disrupted
by certain types of behavior, whether inadvertent or malicious.

4.5. DO NOT REQUIRE CENTRAL MANAGEMENT

NetBIOS on TCP should be able to operate, if desired, without
centralized management beyond that typically required by a TCP based
network.

4.6. ALLOW INTERNET OPERATION

The proposed standard recognizes the need for NetBIOS operation
across a set of networks interconnected by network (IP) level relays
(gateways.)

However, the standard assumes that this form of operation will be
less frequent than on the local MAC bridged-LAN.

4.7. MINIMIZE BROADCAST ACTIVITY

The standard pre-supposes that the only broadcast services are those
supported by UDP. Multicast capabilities are not assumed to be
available in any form.

Despite the availability of broadcast capabilities, the standard
recognizes that some administrations may wish to avoid heavy
broadcast activity. For example, an administration may wish to avoid
isolated non-participating hosts from the burden of receiving and
discarding NetBIOS broadcasts.

4.8. PERMIT IMPLEMENTATION ON EXISTING SYSTEMS

The NetBIOS on TCP protocol should be implementable on common
operating systems, such as Unix(tm) and VAX/VMS(tm), without massive
effort.

The NetBIOS protocols should not require services typically
unavailable on presently existing TCP/UDP/IP implementations.

4.9. REQUIRE ONLY THE MINIMUM NECESSARY TO OPERATE

The protocol definition should specify only the minimal set of
protocols required for interoperation. However, additional protocol
elements may be defined to enhance efficiency. These latter elements
may be generated at the option of the sender, although they must be
accepted by all receivers.

4.10. MAXIMIZE EFFICIENCY

   To be useful, a protocol must conduct its business quickly.

4.11. MINIMIZE NEW INVENTIONS

   When an existing protocol is not quite able to support a necessary
   function, but with a small amount of change, it could, that protocol
   should be used. This is felt to be easier to achieve than
   development of new protocols; further, it is likely to have more
   general utility for the Internet.

5. OVERVIEW OF NetBIOS

   This section describes the NetBIOS services. It is for background
   information only. The reader may chose to skip to the next section.

   NetBIOS was designed for use by groups of PCs, sharing a broadcast
   medium. Both connection (Session) and connectionless (Datagram)
   services are provided, and broadcast and multicast are supported.
   Participants are identified by name. Assignment of names is
   distributed and highly dynamic.

   NetBIOS applications employ NetBIOS mechanisms to locate resources,
   establish connections, send and receive data with an application
   peer, and terminate connections. For purposes of discussion, these
   mechanisms will collectively be called the NetBIOS Service.

   This service can be implemented in many different ways. One of the
   first implementations was for personal computers running the PC-DOS
   and MS-DOS operating systems. It is possible to implement NetBIOS
   within other operating systems, or as processes which are,
   themselves, simply application programs as far as the host operating
   system is concerned.

   The NetBIOS specification, published by IBM as "Technical Reference
   PC Network"[2] defines the interface and services available to the
   NetBIOS user. The protocols outlined by that document pertain only
   to the IBM PC Network and are not generally applicable to other
   networks.

5.1. INTERFACE TO APPLICATION PROGRAMS

   NetBIOS on personal computers includes both a set of services and an
   exact program interface to those services. NetBIOS on other computer
   systems may present the NetBIOS services to programs using other
   interfaces. Except on personal computers, no clear standard for a
   NetBIOS software interface has emerged.

5.2. NAME SERVICE

   NetBIOS resources are referenced by name. Lower-level address
   information is not available to NetBIOS applications. An
   application, representing a resource, registers one or more names
   that it wishes to use.

   The name space is flat and uses sixteen alphanumeric characters.
   Names may not start with an asterisk (*).

   Registration is a bid for use of a name. The bid may be for
   exclusive (unique) or shared (group) ownership. Each application
   contends with the other applications in real time. Implicit
   permission is granted to a station when it receives no objections.
   That is, a bid is made and the application waits for a period of
   time. If no objections are received, the station assumes that it has
   permission.

   A unique name should be held by only one station at a time. However,
   duplicates ("name conflicts") may arise due to errors.

   All instances of a group name are equivalent.

   An application referencing a name generally does not know (or care)
   whether the name is registered as a unique or a group name.

   An explicit name deletion function is specified, so that applications
   may remove a name. Implicit name deletion occurs when a station
   ceases operation. In the case of personal computers, implicit name
   deletion is a frequent occurrence.

   The Name Service primitives are:

      1)   Add Name

           The requesting application wants exclusive use of the name.

      2)   Add Group Name

           The requesting application is willing to share use of the
           name with other applications.

      3)   Delete Name

           The application no longer requires use of the name. It is
           important to note that typical use of NetBIOS is among
           independently-operated personal computers. A common way to
           stop using a PC is to turn it off; in this case, the
           graceful give-back mechanism, provided by the Delete Name
           function, is not used. Because this occurs frequently, the
           network service must support this behavior.

5.3. SESSION SERVICE

   A session is a reliable message exchange, conducted between a pair of
   NetBIOS applications. Sessions are full-duplex, sequenced, and
   reliable. Data is organized into messages. Each message may range
   in size from 0 to 131,071 bytes. No expedited or urgent data
   capabilities are present.

   Multiple sessions may exist between any pair of calling and called
   names.

   The parties to a connection have access to the calling and called
   names.

   The NetBIOS specification does not define how a connection request to
   a shared (group) name resolves into a session. The usual assumption
   is that a session may be established with any one owner of the called
   group name.

   An important service provided to NetBIOS applications is the
   detection of sessions failure. The loss of a session is reported to
   an application via all of the outstanding service requests for that
   session. For example, if the application has only a NetBIOS receive
   primitive pending and the session terminates, the pending receive
   will abort with a termination indication.

   Session Service primitives are:

      1)   Call

           Initiate a session with a process that is listening under
           the specified name. The calling entity must indicate both a
           calling name (properly registered to the caller) and a
           called name.

      2)   Listen

           Accept a session from a caller. The listen primitive may be
           constrained to accept an incoming call from a named caller.
           Alternatively, a call may be accepted from any caller.

      3)   Hang Up

           Gracefully terminate a session. All pending data is
           transferred before the session is terminated.

      4)   Send

           Transmit one message. A time-out can occur. A time-out of
           any session send forces the non-graceful termination of the
           session.

A "chain send" primitive is required by the PC NetBIOS
software interface to allow a single message to be gathered
from pieces in various buffers. Chain Send is an interface
detail and does not effect the protocol.

5)   Receive

Receive data. A time-out can occur. A time-out on a
session receive only terminates the receive, not the
session, although the data is lost.

The receive primitive may be implemented with variants, such
as "Receive Any", which is required by the PC NetBIOS
software interface. Receive Any is an interface detail and
does not effect the protocol.

6)   Session Status

Obtain information about all of the requestor's sessions,
under the specified name. No network activity is involved.

5.4. DATAGRAM SERVICE

The Datagram service is an unreliable, non-sequenced, connectionless
service. Datagrams are sent under cover of a name properly
registered to the sender.

Datagrams may be sent to a specific name or may be explicitly
broadcast.

Datagrams sent to an exclusive name are received, if at all, by the
holder of that name. Datagrams sent to a group name are multicast to
all holders of that name. The sending application program cannot
distinguish between group and unique names and thus must act as if
all non-broadcast datagrams are multicast.

As with the Session Service, the receiver of the datagram is told the
sending and receiving names.

Datagram Service primitives are:

1)   Send Datagram

Send an unreliable datagram to an application that is
associated with the specified name. The name may be unique
or group; the sender is not aware of the difference. If the
name belongs to a group, then each member is to receive the
datagram.

   2)   Send Broadcast Datagram

        Send an unreliable datagram to any application with a
        Receive Broadcast Datagram posted.

   3)   Receive Datagram

        Receive a datagram sent by a specified originating name to
        the specified name. If the originating name is an asterisk,
        then the datagram may have been originated under any name.

        Note: An arriving datagram will be delivered to all pending
        Receiving Datagrams that have source and destination
        specifications matching those of the datagram. In other
        words, if a program (or group of programs) issue a series of
        identical Receive Datagrams, one datagram will cause the
        entire series to complete.

   4)   Receive Broadcast Datagram

        Receive a datagram sent as a broadcast.

        If there are multiple pending Receive Broadcast Datagram
        operations pending, all will be satisfied by the same
        received datagram.

5.5. MISCELLANEOUS FUNCTIONS

   The following functions are present to control the operation of the
   hardware interface to the network. These functions are generally
   implementation dependent.

   1)   Reset

        Initialize the local network adapter.

   2)   Cancel

        Abort a pending NetBIOS request. The successful cancel of a
        Send (or Chain Send) operation will terminate the associated
        session.

   3)   Adapter Status

        Obtain information about the local network adapter or of a
        remote adapter.

   4)   Unlink

        For use with Remote Program Load (RPL). Unlink redirects
        the PC boot disk device back to the local disk. See the

NetBIOS specification for further details concerning RPL and
the Unlink operation (see page 2-35 in [2]).

5)   Remote Program Load

Remote Program Load (RPL) is not a NetBIOS function. It is
a NetBIOS application defined by IBM in their NetBIOS
specification (see pages 2-80 through 2-82 in [2]).

5.6. NON-STANDARD EXTENSIONS

The IBM Token Ring implementation of NetBIOS has added at least one
new user capability:

1)   Find Name

This function determines whether a given name has been
registered on the network.

6. NetBIOS FACILITIES SUPPORTED BY THIS STANDARD

The protocol specified by this standard permits an implementer to
provide all of the NetBIOS services as described in the IBM
"Technical Reference PC Network"[2].

The following NetBIOS facilities are outside the scope of this
specification. These are local implementation matters and do not
impact interoperability:

- RESET
- SESSION STATUS
- UNLINK
- RPL (Remote Program Load)

7. REQUIRED SUPPORTING SERVICE INTERFACES AND DEFINITIONS

The protocols described in this RFC require service interfaces to the
following:

- TCP[3,4]
- UDP[5]

Byte ordering, addressing conventions (including addresses to be
used for broadcasts and multicasts) are defined by the most
recent version of:

- Assigned Numbers[6]

Additional definitions and constraints are in:

         - IP[7]
         - Internet Subnets[8,9,10]

    8. RELATED PROTOCOLS AND SERVICES

       The design of the protocols described in this RFC allow for the
       future incorporation of the following protocols and services.
       However, before this may occur, certain extensions may be required to
       the protocols defined in this RFC or to those listed below.

         - Domain Name Service[11,12,13,14]
         - Internet Group Multicast[15,16]

    9. NetBIOS SCOPE

       A "NetBIOS Scope" is the population of computers across which a
       registered NetBIOS name is known. NetBIOS broadcast and multicast
       datagram operations must reach the entire extent of the NetBIOS
       scope.

       An internet may support multiple, non-intersecting NetBIOS Scopes.

       Each NetBIOS scope has a "scope identifier". This identifier is a
       character string meeting the requirements of the domain name system
       for domain names.

       NOTE: Each implementation of NetBIOS-over-TCP must provide
             mechanisms to manage the scope identifier(s) to be used.

       Control of scope identifiers implies a requirement for additional
       NetBIOS interface capabilities. These may be provided through
       extensions of the user service interface or other means (such as node
       configuration parameters.) The nature of these extensions is not
       part of this specification.

    10. NetBIOS END-NODES

       End-nodes support NetBIOS service interfaces and contain
       applications.

       Three types of end-nodes are part of this standard:

         - Broadcast ("B") nodes
         - Point-to-point ("P") nodes
         - Mixed mode ("M") nodes

       An IP address may be associated with only one instance of one of the
       above types.

       Without having preloaded name-to-address tables, NetBIOS participants

are faced with the task of dynamically resolving references to one
another. This can be accomplished with broadcast or mediated point-
to-point communications.

B nodes use local network broadcasting to effect a rendezvous with
one or more recipients. P and M nodes use the NetBIOS Name Server
(NBNS) and the NetBIOS Datagram Distribution Server (NBDD) for this
same purpose.

End-nodes may be combined in various topologies. No matter how
combined, the operation of the B, P, and M nodes is not altered.

NOTE: It is recommended that the administration of a NetBIOS
      scope avoid using both M and B nodes within the same scope.
      A NetBIOS scope should contain only B nodes or only P and M
      nodes.

10.1. BROADCAST (B) NODES

Broadcast (or "B") nodes communicate using a mix of UDP datagrams
(both broadcast and directed) and TCP connections. B nodes may
freely interoperate with one another within a broadcast area. A
broadcast area is a single MAC-bridged "B-LAN". (See Appendix A for
a discussion of using Internet Group Multicasting as a means to
extend a broadcast area beyond a single B-LAN.)

10.2. POINT-TO-POINT (P) NODES

Point-to-point (or "P") nodes communicate using only directed UDP
datagrams and TCP sessions. P nodes neither generate nor listen for
broadcast UDP packets. P nodes do, however, offer NetBIOS level
broadcast and multicast services using capabilities provided by the
NBNS and NBDD.

P nodes rely on NetBIOS name and datagram distribution servers.
These servers may be local or remote; P nodes operate the same in
either case.

10.3. MIXED MODE (M) NODES

Mixed mode nodes (or "M") nodes are P nodes which have been given
certain B node characteristics. M nodes use both broadcast and
unicast. Broadcast is used to improve response time using the
assumption that most resources reside on the local broadcast medium
rather than somewhere in an internet.

M nodes rely upon NBNS and NBDD servers. However, M nodes may
continue limited operation should these servers be temporarily
unavailable.

11. NetBIOS SUPPORT SERVERS

   Two types of support servers are part of this standard:

      - NetBIOS name server ("NBNS") nodes
      - Netbios datagram distribution ("NBDD") nodes

   NBNS and NBDD nodes are invisible to NetBIOS applications and are
   part of the underlying NetBIOS mechanism.

   NetBIOS name and datagram distribution servers are the focus of name
   and datagram activity for P and M nodes.

   Both the name (NBNS) and datagram distribution (NBDD) servers are
   permitted to shift part of their operation to the P or M end-node
   which is requesting a service.

   Since the assignment of responsibility is dynamic, and since P and M
   nodes must be prepared to operate should the NetBIOS server delegate
   control to the maximum extent, the system naturally accommodates
   improvements in NetBIOS server function. For example, as Internet
   Group Multicasting becomes more widespread, new NBDD implementations
   may elect to assume full responsibility for NetBIOS datagram
   distribution.

   Interoperability between different implementations is assured by
   imposing requirements on end-node implementations that they be able
   to accept the full range of legal responses from the NBNS or NBDD.

11.1. NetBIOS NAME SERVER (NBNS) NODES

   The NBNS is designed to allow considerable flexibility with its
   degree of responsibility for the accuracy and management of NetBIOS
   names. On one hand, the NBNS may elect not to accept full
   responsibility, leaving the NBNS essentially a "bulletin board" on
   which name/address information is freely posted (and removed) by P
   and M nodes without validation by the NBNS. Alternatively, the NBNS
   may elect to completely manage and validate names. The degree of
   responsibility that the NBNS assumes is asserted by the NBNS each
   time a name is claimed through a simple mechanism. Should the NBNS
   not assert full control, the NBNS returns enough information to the
   requesting node so that the node may challenge any putative holder of
   the name.

   This ability to shift responsibility for NetBIOS name management
   between the NBNS and the P and M nodes allows a network administrator
   (or vendor) to make a tradeoff between NBNS simplicity, security, and
   delay characteristics.

   A single NBNS may be implemented as a distributed entity, such as the
   Domain Name Service. However, this RFC does not attempt to define

the internal communications which would be used.

11.1.1. RELATIONSHIP OF THE NBNS TO THE DOMAIN NAME SYSTEM

The NBNS design attempts to align itself with the Domain Name System
in a number of ways.

First, the NetBIOS names are encoded in a form acceptable to the
domain name system.

Second, a scope identifier is appended to each NetBIOS name. This
identifier meets the restricted character set of the domain system
and has a leading period. This makes the NetBIOS name, in
conjunction with its scope identifier, a valid domain system name.

Third, the negotiated responsibility mechanisms permit the NBNS to be
used as a simple bulletin board on which are posted (name,address)
pairs. This parallels the existing domain sytem query service.

This RFC, however, requires the NBNS to provide services beyond those
provided by the current domain name system. An attempt has been made
to coalesce all the additional services which are required into a set
of transactions which follow domain name system styles of interaction
and packet formats.

Among the areas in which the domain name service must be extended
before it may be used as an NBNS are:

    - Dynamic addition of entries
    - Dynamic update of entry data
    - Support for multiple instance (group) entries
    - Support for entry time-to-live values and ability to accept
       refresh messages to restart the time-to-live period
    - New entry attributes

11.2. NetBIOS DATAGRAM DISTRIBUTION SERVER (NBDD) NODES

The internet does not yet support broadcasting or multicasting. The
NBDD extends NetBIOS datagram distribution service to this
environment.

The NBDD may elect to complete, partially complete, or totally refuse
to service a node's request to distribute a NetBIOS datagram. An
end-node may query an NBDD to determine whether the NBDD will deliver
a datagram to a specific NetBIOS name.

The design of NetBIOS-over-TCP lends itself to the use of Internet
Group Multicast. For details see Appendix A.

11.3. RELATIONSHIP OF NBNS AND NBDD NODES

   This RFC defines the NBNS and NBDD as distinct, separate entities.

   In the absence of NetBIOS name information, a NetBIOS datagram
   distribution server must send a copy to each end-node within a
   NetBIOS scope.

   An implementer may elect to construct NBNS and NBDD nodes which have
   a private protocol for the exchange of NetBIOS name information.
   Alternatively, an NBNS and NBDD may be implemented within the same
   device.

   NOTE: Implementations containing private NBNS-NBDD protocols or
          combined NBNS-NBDD functions must be clearly identified.

11.4. RELATIONSHIP OF NetBIOS SUPPORT SERVERS AND B NODES

   As defined in this RFC, neither NBNS nor NBDD nodes interact with B
   nodes. NetBIOS servers do not listen to broadcast traffic on any
   broadcast area to which they may be attached. Nor are the NetBIOS
   support servers even aware of B node activities or names claimed or
   used by B nodes.

   It may be possible to extend both the NBNS and NBDD so that they
   participate in B node activities and act as a bridge to P and M
   nodes. However, such extensions are beyond the scope of this
   specification.

12. TOPOLOGIES

   B, P, M, NBNS, and NBDD nodes may be combined in various ways to form
   useful NetBIOS environments. This section describes some of these
   combinations.

   There are three classes of operation:

      - Class 0: B nodes only.
      - Class 1: P nodes only.
      - Class 2: P and M nodes together.

   In the drawings which follow, any P node may be replaced by an M
   node. The effects of such replacement will be mentioned in
   conjunction with each example below.

12.1. LOCAL

   A NetBIOS scope is operating locally when all entities are within the
   same broadcast area.

12.1.1. B NODES ONLY

   Local operation with only B nodes is the most basic mode of
   operation. Name registration and discovery procedures use broadcast
   mechanisms. The NetBIOS scope is limited by the extent of the
   broadcast area. This configuration does not require NetBIOS support
   servers.

```
   ====+=========+=====BROADCAST AREA=====+=========+=========+====
       |         |                        |         |         |
       |         |                        |         |         |
    +--+--+   +--+--+                   +--+--+   +--+--+   +--+--+
    | B  |    | B  |                    | B  |    | B  |    | B  |
    +-----+   +-----+                   +-----+   +-----+   +-----+
```
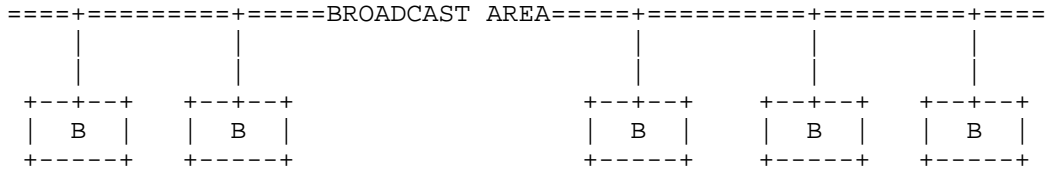
12.1.2. P NODES ONLY

   This configuration would typically be used when the network
   administrator desires to eliminate NetBIOS as a source of broadcast
   activity.

```
   ====+=========+==========+=B'CAST AREA=+=========+=========+====
       |         |          |             |         |         |
       |         |          |             |         |         |
    +--+--+   +--+--+    +--+--+       +--+--+   +--+--+   +--+--+
    | P  |    | P  |     |NBNS |       | P  |    |NBDD |    | P  |
    +-----+   +-----+    +-----+       +-----+   +-----+   +-----+
```
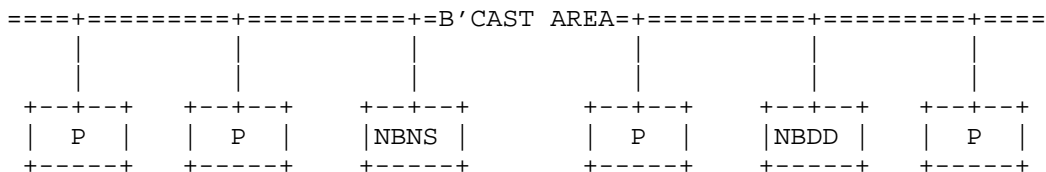
   This configuration operates the same as if it were in an internet and
   is cited here only due to its convenience as a means to reduce the
   use of broadcast.

   Replacement of one or more of the P nodes with M nodes will not
   affect the operation of the other P and M nodes. P and M nodes will
   be able to interact with one another. Because M nodes use broadcast,
   overall broadcast activity will increase.

12.1.3. MIXED B AND P NODES

   B and P nodes do not interact with one another. Replacement of P
   nodes with M nodes will allow B's and M's to interact.

   NOTE: B nodes and M nodes may be intermixed only on a local
         broadcast area. B and M nodes should not be intermixed in
         an internet environment.

12.2. INTERNET

12.2.1. P NODES ONLY

   P nodes may be scattered at various locations in an internetwork.
   They require both an NBNS and an NBDD for NetBIOS name and datagram
   support, respectively.

   The NetBIOS scope is determined by the NetBIOS scope identifier
   (domain name) used by the various P (and M) nodes. An internet may
   contain numerous NetBIOS scopes.

```
                      +-----+
                      |  P  |
                      +--+--+                   |    +-----+
                         |                      |----+  P  |
                         |                      |    +-----+
                    /-----+-----\               |
    +-----+         |           |   +------+    |    +-----+
    |  P  +------+  INTERNET +---+G'WAY |-+----+  P  |
    +-----+         |           |   +------+    |    +-----+
                    /-----+-----/               |
           /        |                      |    +-----+
          /         |                      |----+  P  |
      +-----+    +--+--+                    |    +-----+
      |NBNS +    |NBDD |                    |
      +-----+    +--+--+
```
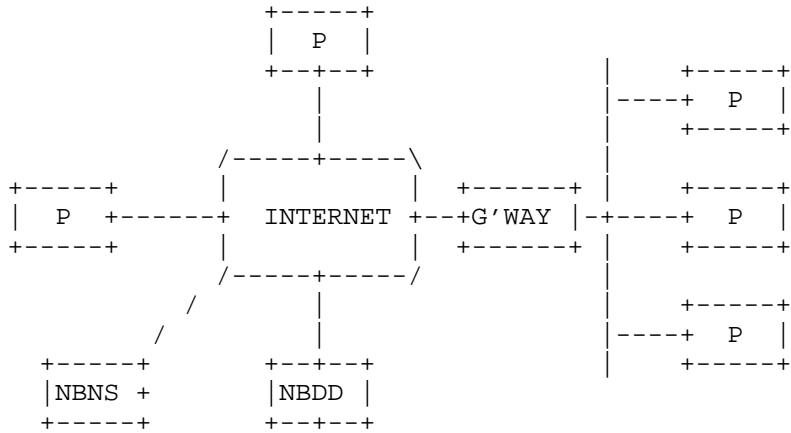
   Any P node may be replaced by an M node with no loss of function to
   any node. However, broadcast activity will be increased in the
   broadcast area to which the M node is attached.

12.2.2. MIXED M AND P NODES

   M and P nodes may be mixed. When locating NetBIOS names, M nodes
   will tend to find names held by other M nodes on the same common
   broadcast area in preference to names held by P nodes or M nodes
   elsewhere in the network.

```
                             +-----+
                             |  P  |
                             +--+--+
                                |
                                |
                                |
                          /-----+-----\
            +-----+       |           |       +-----+
            |  P  +------+  INTERNET +------+NBDD |
            +-----+       |           |       +-----+
                          /-----+-----/
                        /         |
                      /           |
             +-----+            +--+--+
             |NBNS +            |G'WAY|
             +-----+            +--+--+
                                   |
                                   |
     ====+=========+==========+=B'CAST AREA=+=========+=========+====
         |         |          |             |         |         |
         |         |          |             |         |         |
         |         |          |             |         |         |
      +--+--+   +--+--+    +--+--+       +--+--+   +--+--+   +--+--+
      |  M  |   |  P  |    |  M  |       |  P  |   |  M  |   |  P  |
      +-----+   +-----+    +--+--+       +-----+   +-----+   +-----+
```
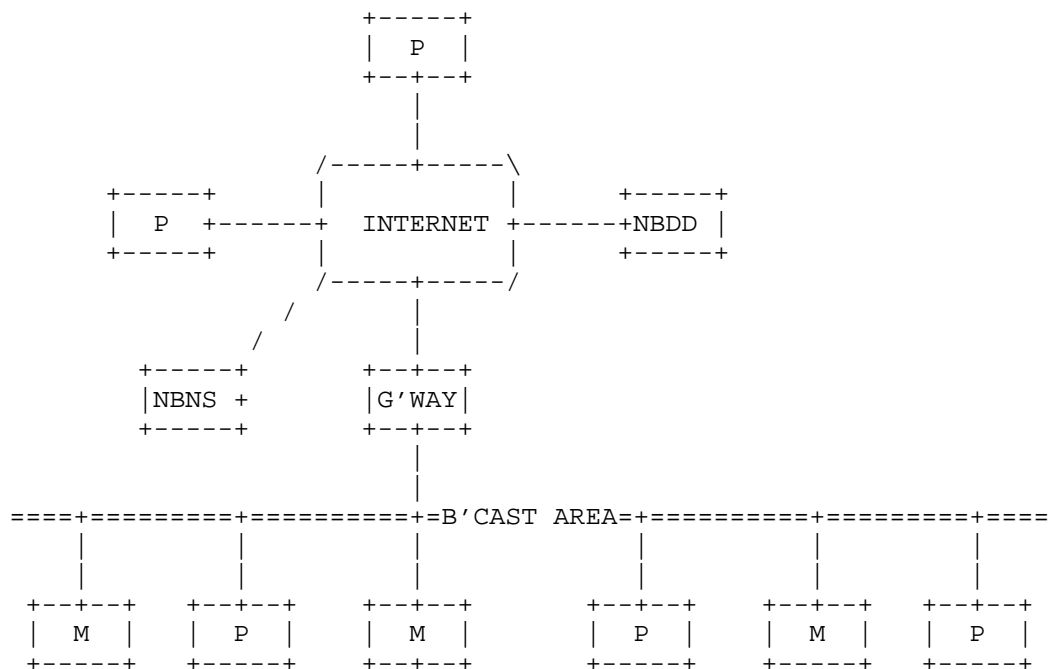
   NOTE: B and M nodes should not be intermixed in an internet
         environment. Doing so would allow undetected NetBIOS name
         conflicts to arise and cause unpredictable behavior.

13. GENERAL METHODS

   Overlying the specific protocols, described later, are a few general
   methods of interaction between entities.

13.1. REQUEST/RESPONSE INTERACTION STYLE

   Most interactions between entities consist of a request flowing in
   one direction and a subsequent response flowing in the opposite
   direction.

   In those situations where interactions occur on unreliable transports
   (i.e. UDP) or when a request is broadcast, there may not be a strict
   interlocking or one-to-one relationship between requests and
   responses.

In no case, however, is more than one response generated for a
received request. While a response is pending the responding entity
may send one or more wait acknowledgements.

13.1.1. RETRANSMISSION OF REQUESTS

UDP is an unreliable delivery mechanism where packets can be lost,
received out of transmit sequence, duplicated and delivery can be
significantly delayed. Since the NetBIOS protocols make heavy use of
UDP, they have compensated for its unreliability with extra
mechanisms.

Each NetBIOS packet contains all the necessary information to process
it. None of the protocols use multiple UDP packets to convey a
single request or response. If more information is required than
will fit in a single UDP packet, for example, when a P-type node
wants all the owners of a group name from a NetBIOS server, a TCP
connection is used. Consequently, the NetBIOS protocols will not
fail because of out of sequence delivery of UDP packets.

To overcome the loss of a request or response packet, each request
operation will retransmit the request if a response is not received
within a specified time limit.

Protocol operations sensitive to successive response packets, such as
name conflict detection, are protected from duplicated packets
because they ignore successive packets with the same NetBIOS
information. Since no state on the responder's node is associated
with a request, the responder just sends the appropriate response
whenever a request packet arrives. Consequently, duplicate or
delayed request packets have no impact.

For all requests, if a response packet is delayed too long another
request packet will be transmitted. A second response packet being
sent in response to the second request packet is equivalent to a
duplicate packet. Therefore, the protocols will ignore the second
packet received. If the delivery of a response is delayed until
after the request operation has been completed, successfully or not,
the response packet is ignored.

13.1.2. REQUESTS WITHOUT RESPONSES: DEMANDS

Some request types do not have matching responses. These requests
are known as "demands". In general a "demand" is an imperative
request; the receiving node is expected to obey. However, because
demands are unconfirmed, they are used only in situations where, at
most, limited damage would occur if the demand packet should be lost.

Demand packets are not retransmitted.

13.2. TRANSACTIONS

   Interactions between a pair of entities are grouped into
   "transactions". These transactions comprise one or more
   request/response pairs.

13.2.1. TRANSACTION ID

   Since multiple simultaneous transactions may be in progress between a
   pair of entities a "transaction id" is used.

   The originator of a transaction selects an ID unique to the
   originator. The transaction id is reflected back and forth in each
   interaction within the transaction. The transaction partners must
   match responses and requests by comparison of the transaction ID and
   the IP address of the transaction partner. If no matching request
   can be found the response must be discarded.

   A new transaction ID should be used for each transaction. A simple
   16 bit transaction counter ought to be an adequate id generator. It
   is probably not necessary to search the space of outstanding
   transaction ID to filter duplicates: it is extremely unlikely that
   any transaction will have a lifetime that is more than a small
   fraction of the typical counter cycle period. Use of the IP
   addresses in conjunction with the transaction ID further reduces the
   possibility of damage should transaction IDs be prematurely re-used.

13.3. TCP AND UDP FOUNDATIONS

   This version of the NetBIOS-over-TCP protocols uses UDP for many
   interactions. In the future this RFC may be extended to permit such
   interactions to occur over TCP connections (perhaps to increase
   efficiency when multiple interactions occur within a short time or
   when NetBIOS datagram traffic reveals that an application is using
   NetBIOS datagrams to support connection- oriented service.)

14. REPRESENTATION OF NETBIOS NAMES

   NetBIOS names as seen across the client interface to NetBIOS are
   exactly 16 bytes long. Within the NetBIOS-over-TCP protocols, a
   longer representation is used.

   There are two levels of encoding. The first level maps a NetBIOS
   name into a domain system name. The second level maps the domain
   system name into the "compressed" representation required for
   interaction with the domain name system.

   Except in one packet, the second level representation is the only
   NetBIOS name representation used in NetBIOS-over-TCP packet formats.
   The exception is the RDATA field of a NODE STATUS RESPONSE packet.

14.1. FIRST LEVEL ENCODING

   The first level representation consists of two parts:

      - NetBIOS name
      - NetBIOS scope identifier

   The 16 byte NetBIOS name is mapped into a 32 byte wide field using a
   reversible, half-ASCII, biased encoding. Each half-octet of the
   NetBIOS name is encoded into one byte of the 32 byte field. The
   first half octet is encoded into the first byte, the second half-
   octet into the second byte, etc.

   Each 4-bit, half-octet of the NetBIOS name is treated as an 8-bit,
   right-adjusted, zero-filled binary number. This number is added to
   value of the ASCII character 'A' (hexidecimal 41). The resulting 8-
   bit number is stored in the appropriate byte. The following diagram
   demonstrates this procedure:

```
                        0 1 2 3 4 5 6 7
                        +-+-+-+-+-+-+-+-+
                        |a b c d|w x y z|              ORIGINAL BYTE
                        +-+-+-+-+-+-+-+-+
                            |       |
                    +--------+      +--------+
                    |                        |   SPLIT THE NIBBLES
                    v                        v
         0 1 2 3 4 5 6 7            0 1 2 3 4 5 6 7
         +-+-+-+-+-+-+-+-+            +-+-+-+-+-+-+-+-+
         |0 0 0 0 a b c d|            |0 0 0 0 w x y z|
         +-+-+-+-+-+-+-+-+            +-+-+-+-+-+-+-+-+
                 |                            |
                 +                            +    ADD 'A'
                 |                            |
         0 1 2 3 4 5 6 7            0 1 2 3 4 5 6 7
         +-+-+-+-+-+-+-+-+            +-+-+-+-+-+-+-+-+
         |0 1 0 0 0 0 0 1|            |0 1 0 0 0 0 0 1|
         +-+-+-+-+-+-+-+-+            +-+-+-+-+-+-+-+-+
```
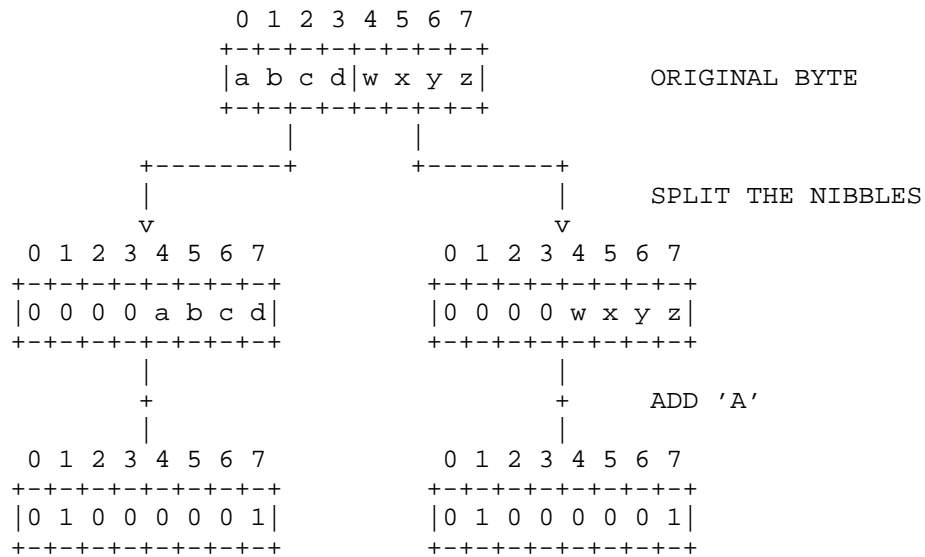
   This encoding results in a NetBIOS name being represented as a
   sequence of 32 ASCII, upper-case characters from the set
   {A,B,C...N,O,P}.

   The NetBIOS scope identifier is a valid domain name (without a
   leading dot).

   An ASCII dot (2E hexidecimal) and the scope identifier are appended
   to the encoded form of the NetBIOS name, the result forming a valid
   domain name.

For example, the NetBIOS name "The NetBIOS name" in the NetBIOS scope
"SCOPE.ID.COM" would be represented at level one by the ASCII
character string:

        FEGHGFCAEOGFHEECEJEPFDCAHEGBGNGF.SCOPE.ID.COM

14.2. SECOND LEVEL ENCODING

The first level encoding must be reduced to second level encoding.
This is performed according to the rules defined in on page 31 of RFC
883[12] in the section on "Domain name representation and
compression". Also see the section titled "Name Formats" in the
Detailed Specifications[1].

15. NetBIOS NAME SERVICE

Before a name may be used, the name must be registered by a node.
Once acquired, the name must be defended against inconsistent
registration by other nodes. Before building a NetBIOS session or
sending a NetBIOS datagram, the one or more holders of the name must
be located.

The NetBIOS name service is the collection of procedures through
which nodes acquire, defend, and locate the holders of NetBIOS names.

The name service procedures are different depending whether the end-
node is of type B, P, or M.

15.1. OVERVIEW OF NetBIOS NAME SERVICE

15.1.1. NAME REGISTRATION (CLAIM)

Each NetBIOS node can own more than one name. Names are acquired
dynamically through the registration (name claim) procedures.

Every node has a permanent unique name. This name, like any other
name, must be explicitly registered by all end-node types.

A name can be unique (exclusive) or group (non-exclusive). A unique
name may be owned by a single node; a group name may be owned by any
number of nodes. A name ceases to exist when it is not owned by at
least one node. There is no intrinsic quality of a name which
determines its characteristics: these are established at the time of
registration.

Each node maintains state information for each name it has
registered. This information includes:

   - Whether the name is a group or unique name
   - Whether the name is "in conflict"
   - Whether the name is in the process of being deleted

B nodes perform name registration by broadcasting claim requests, soliciting a defense from any node already holding the name.

P nodes perform name registration through the agency of the NBNS.

M nodes register names through an initial broadcast, like B nodes, then, in the absence of an objection, by following the same procedures as a P node. In other words, the broadcast action may terminate the attempt, but is not sufficient to confirm the registration.

15.1.2. NAME QUERY (DISCOVERY)

Name query (also known as "resolution" or "discovery") is the procedure by which the IP address(es) associated with a NetBIOS name are discovered. Name query is required during the following operations:

During session establishment, calling and called names must be specified. The calling name must exist on the node that posts the CALL. The called name must exist on a node that has previously posted a LISTEN. Either name may be a unique or group name.

When a directed datagram is sent, a source and destination name must be specified. If the destination name is a group name, a datagram is sent to all the members of that group.

Different end-node types perform name resolution using different techniques, but using the same packet formats:

   - B nodes solicit name information by broadcasting a request.

   - P nodes ask the NBNS.

   - M nodes broadcast a request. If that does not provide the
      desired information, an inquiry is sent to the NBNS.

15.1.3. NAME RELEASE

NetBIOS names may be released explicitly or silently by an end- node. Silent release typically occurs when an end-node fails or is turned-off. Most of the mechanisms described below are present to detect silent name release.

15.1.3.1. EXPLICIT RELEASE

B nodes explicitly release a name by broadcasting a notice.

P nodes send a notification to their NBNS.

M nodes both broadcast a notice and inform their supporting NBNS.

15.1.3.2. NAME LIFETIME AND REFRESH

   Names held by an NBNS are given a lifetime during name registration.
   The NBNS will consider a name to have been silently released if the
   end-node fails to send a name refresh message to the NBNS before the
   lifetime expires. A refresh restarts the lifetime clock.

   NOTE: The implementor should be aware of the tradeoff between
         accuracy of the database and the internet overhead that the
         refresh mechanism introduces. The lifetime period should
         be tuned accordingly.

   For group names, each end-node must send refresh messages. A node
   that fails to do so will be considered to have silently released the
   name and dropped from the group.

   The lifetime period is established through a simple negotiation
   mechanism during name registration: In the name registration
   request, the end-node proposes a lifetime value or requests an
   infinite lifetime. The NBNS places an actual lifetime value into the
   name registration response. The NBNS is always allowed to respond
   with an infinite actual period. If the end node proposed an infinite
   lifetime, the NBNS may respond with any definite period. If the end
   node proposed a definite period, the NBNS may respond with any
   definite period greater than or equal to that proposed.

   This negotiation of refresh times gives the NBNS means to disable or
   enable refresh activity. The end-nodes may set a minimum refresh
   cycle period.

   NBNS implementations which are completely reliable may disable
   refresh.

15.1.3.3. NAME CHALLENGE

   To detect whether a node has silently released its claim to a name,
   it is necessary on occasion to challenge that node's current
   ownership. If the node defends the name then the node is allowed to
   continue possession. Otherwise it is assumed that the node has
   released the name.

   A name challenge may be issued by an NBNS or by a P or M node. A
   challenge may be directed towards any end-node type: B, P, or M.

15.1.3.4. GROUP NAME FADE-OUT

   NetBIOS groups may contain an arbitrarily large number of members.
   The time to challenge all members could be quite large.

   To avoid long delays when names are claimed through an NBNS, an

optimistic heuristic has been adopted. It is assumed that there will
always be some node which will defend a group name. Consequently, it
is recommended that the NBNS will immediately reject a claim request
for a unique name when there already exists a group with the same
name. The NBNS will never return an IP address (in response to a
NAME REGISTRATION REQUEST) when a group name exists.

An NBNS will consider a group to have faded out of existence when the
last remaining member fails to send a timely refresh message or
explicitly releases the name.

15.1.3.5. NAME CONFLICT

Name conflict exists when a unique name has been claimed by more than
one node on a NetBIOS network. B, M, and NBNS nodes may detect a
name conflict. The detection mechanism used by B and M nodes is
active only during name discovery. The NBNS may detect conflict at
any time it verifies the consistency of its name database.

B and M nodes detect conflict by examining the responses received in
answer to a broadcast name query request. The first response is
taken as authoritative. Any subsequent, inconsistent responses
represent conflicts.

Subsequent responses are inconsistent with the authoritative response
when:

        The subsequent response has the same transaction ID as the
        NAME QUERY REQUEST.
    AND
        The subsequent response is not a duplicate of the
        authoritative response.
    AND EITHER:
            The group/unique characteristic of the authoritative
            response is "unique".
          OR
            The group/unique characteristic of the subsequent
            response is "unique".

The period in which B and M nodes examine responses is limited by a
conflict timer, CONFLICT_TIMER. The accuracy or duration of this
timer is not crucial: the NetBIOS system will continue to operate
even with persistent name conflicts.

Conflict conditions are signaled by sending a NAME CONFLICT DEMAND to
the node owning the offending name. Nothing is sent to the node
which originated the authoritative response.

Any end-node that receives NAME CONFLICT DEMAND is required to update
its "local name table" to reflect that the name is in conflict. (The
"local name table" on each node contains names that have been

successfully registered by that node.)

Notice that only those nodes that receive the name conflict message
place a conflict mark next to a name.

Logically, a marked name does not exist on that node. This means
that the node should not defend the name (for name claim purposes),
should not respond to a name discovery requests for that name, nor
should the node send name refresh messages for that name.
Furthermore, it can no longer be used by that node for any session
establishment or sending or receiving datagrams. Existing sessions
are not affected at the time a name is marked as being in conflict.

The only valid user function against a marked name is DELETE NAME.
Any other user NetBIOS function returns immediately with an error
code of "NAME CONFLICT".

15.1.4. ADAPTER STATUS

An end-node or the NBNS may ask any other end-node for a collection
of information about the NetBIOS status of that node. This status
consists of, among other things, a list of the names which the node
believes it owns. The returned status is filtered to contain only
those names which have the same NetBIOS scope identifier as the
requestor's name.

When requesting node status, the requestor identifies the target node
by NetBIOS name A name query transaction may be necessary to acquire
the IP address for the name. Locally cached name information may be
used in lieu of a query transaction. The requesting node sends a
NODE STATUS REQUEST. In response, the receiving node sends a NODE
STATUS RESPONSE containing its local name table and various
statistics.

The amount of status which may be returned is limited by the size of
a UDP packet. However, this is sufficient for the typical NODE
STATUS RESPONSE packet.

15.1.5. END-NODE NBNS INTERACTION

There are certain characteristics of end-node to NBNS interactions
which are in common and are independent of any particular transaction
type.

15.1.5.1. UDP, TCP, AND TRUNCATION

For all transactions between an end-node and an NBNS, either UDP or
TCP may be used as a transport. If the NBNS receives a UDP based
request, it will respond using UDP. If the amount of information
exceeds what fits into a UDP packet, the response will contain a
"truncation flag". In this situation, the end- node may open a TCP

connection to the NBNS, repeat the request, and receive a complete, untruncated response.

15.1.5.2. NBNS WACK

While a name service request is in progress, the NBNS may issue a WAIT FOR ACKNOWLEDGEMENT RESPONSE (WACK) to assure the client end-node that the NBNS is still operational and is working on the request.

15.1.5.3. NBNS REDIRECTION

The NBNS, because it follows Domain Name system styles of interaction, is permitted to redirect a client to another NBNS.

15.1.6. SECURED VERSUS NON-SECURED NBNS

An NBNS may be implemented in either of two general ways: The NBNS may monitor, and participate in, name activity to ensure consistency. This would be a "secured" style NBNS. Alternatively, an NBNS may be implemented to be essentially a "bulletin board" on which name information is posted and responsibility for consistency is delegated to the end-nodes. This would be a "non-secured" style NBNS.

15.1.7. CONSISTENCY OF THE NBNS DATA BASE

Even in a properly running NetBIOS scope the NBNS and its community of end-nodes may occasionally lose synchronization with respect to the true state of name registrations.

This may occur should the NBNS fail and lose all or part of its database.

More commonly, a P or M node may be turned-off (thus forgetting the names it has registered) and then be subsequently turned back on.

Finally, errors may occur or an implementation may be incorrect.

Various approaches have been incorporated into the NetBIOS-over- TCP protocols to minimize the impact of these problems.

   1.   The NBNS (or any other node) may "challenge" (using a NAME
        QUERY REQUEST) an end-node to verify that it actually owns a
        name.

        Such a challenge may occur at any time. Every end-node must
        be prepared to make a timely response.

        Failure to respond causes the NBNS to consider that the
        end-node has released the name in question.

(If UDP is being used as the underlying transport, the
challenge, like all other requests, must be retransmitted
some number of times in the absence of a response.)

2.   The NBNS (or any other node) may request (using the NODE
      STATUS REQUEST) that an end-node deliver its entire name
      table.

      This may occur at any time. Every end-node must be prepared
      to make a timely response.

      Failure to respond permits (but does not require) the NBNS
      to consider that the end-node has failed and released all
      names to which it had claims. (Like the challenge, on a UDP
      transport, the request must be retransmitted in the absence
      of a response.)

3.   The NBNS may revoke a P or M node's use of a name by sending
      either a NAME CONFLICT DEMAND or a NAME RELEASE REQUEST to
      the node.

      The receiving end-node may continue existing sessions which
      use that name, but must otherwise cease using that name. If
      the NBNS placed the name in conflict, the name may be re-
      acquired only by deletion and subsequent reclamation. If
      the NBNS requested that the name be released, the node may
      attempt to re-acquire the name without first performing a
      name release transaction.

4.   The NBNS may impose a "time-to-live" on each name it
      registers. The registering node is made aware of this time
      value during the name registration procedure.

      Simple or reliable NBNS's may impose an infinite time-to-
      live.

5.   If an end-node holds any names that have finite time-to-
      live values, then that node must periodically send a status
      report to the NBNS. Each name is reported using the NAME
      REFRESH REQUEST packet.

      These status reports restart the timers of both the NBNS and
      the reporting node. However, the only timers which are
      restarted are those associated with the name found in the
      status report. Timers on other names are not affected.

      The NBNS may consider that a node has released any name
      which has not been refreshed within some multiple of name's
      time-to-live.

      A well-behaved NBNS, would, however, issue a challenge to-,

or request a list of names from-, the non-reporting end-
node before deleting its name(s). The absence of a
response, or of the name in a response, will confirm the
NBNS decision to delete a name.

6.   The absence of reports may cause the NBNS to infer that the
end-node has failed. Similarly, receipt of information
widely divergent from what the NBNS believes about the node,
may cause the NBNS to consider that the end-node has been
restarted.

The NBNS may analyze the situation through challenges or
requests for a list of names.

7.   A very cautious NBNS is free to poll nodes (by sending NAME
QUERY REQUEST or NODE STATUS REQUEST packets) to verify that
their name status is the same as that registered in the
NBNS.

NOTE: Such polling activity, if used at all by an
implementation, should be kept at a very low level or
enabled only during periods when the NBNS has some reason to
suspect that its information base is inaccurate.

8.   P and M nodes can detect incorrect name information at
session establishment.

If incorrect information is found, NBNS is informed via a
NAME RELEASE REQUEST originated by the end-node which
detects the error.

15.1.8. NAME CACHING

An end-node may keep a local cache of NetBIOS name-to-IP address
translation entries.

All cache entries should be flushed on a periodic basis.

In addition, a node ought to flush any cache information associated
with an IP address if the node receives any information indicating
that there may be any possibility of trouble with the node at that IP
address. For example, if a NAME CONFLICT DEMAND is sent to a node,
all cached information about that node should be cleared within the
sending node.

15.2. NAME REGISTRATION TRANSACTIONS

15.2.1. NAME REGISTRATION BY B NODES

A name claim transaction initiated by a B node is broadcast
throughout the broadcast area. The NAME REGISTRATION REQUEST will be

heard by all B and M nodes in the area. Each node examines the claim
to see whether it it is consistent with the names it owns. If an
inconsistency exists, a NEGATIVE NAME REGISTRATION RESPONSE is
unicast to the requestor. The requesting node obtains ownership of
the name (or membership in the group) if, and only if, no NEGATIVE
NAME REGISTRATION RESPONSEs are received within the name claim
timeout, CONFLICT_TIMER. (See "Defined Constants and Variables" in
the Detailed Specification for the value of this timer.)

A B node proclaims its new ownership by broadcasting a NAME OVERWRITE
DEMAND.

```
                     B-NODE REGISTRATION PROCESS
<-----NAME NOT ON NETWORK------>   <----NAME ALREADY EXISTS---->

REQ. NODE                       NODE                      REQ.NODE
                                HOLDING
                                NAME

(BROADCAST) REGISTER                     (BROADCAST) REGISTER
------------------->                     <-------------------

      REGISTER                                 REGISTER
------------------->                     <-------------------

      REGISTER                        NEGATIVE RESPONSE
------------------->                  ------------------------------>

      OVERWRITE
------------------->                  (NODE DOES NOT HAVE THE NAME)

(NODE HAS THE NAME)
```
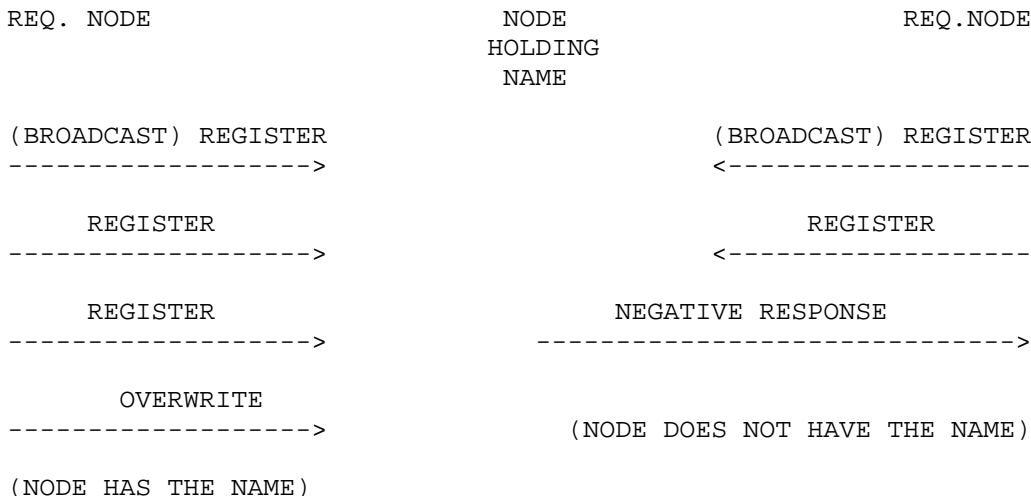
The NAME REGISTRATION REQUEST, like any request, must be repeated if
no response is received within BCAST_REQ_RETRY_TIMEOUT. Transmission
of the request is attempted BCAST_REQ_RETRY_COUNT times.

15.2.2. NAME REGISTRATION BY P NODES

A name registration may proceed in various ways depending whether
the name being registered is new to the NBNS. If the name is known
to the NBNS, then challenges may be sent to the prior holder(s).

15.2.2.1. NEW NAME, OR NEW GROUP MEMBER

The diagram, below, shows the sequence of events when an end-node
registers a name which is new to the NBNS. (The diagram omits WACKs,
NBNS redirections, and retransmission of requests.)

This same interaction will occur if the name being registered is a
group name and the group already exists. The NBNS will add the

registrant to the set of group members.

```
                    P-NODE REGISTRATION PROCESS
            (server has no previous information about the name)

       P-NODE                                  NBNS
                        REGISTER
         --------------------------------->

                        POSITIVE RESPONSE
         <--------------------------------
```
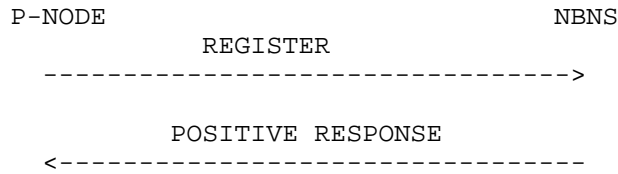
The interaction is rather simple: the end-node sends a NAME
REGISTRATION REQUEST, the NBNS responds with a POSITIVE NAME
REGISTRATION RESPONSE.

15.2.2.2. EXISTING NAME AND OWNER IS STILL ACTIVE

The following diagram shows interactions when an attempt is made to
register a unique name, the NBNS is aware of an existing owner, and
that existing owner is still active.

There are two sides to the diagram. The left side shows how a non-
secured NBNS would handle the matter. Secured NBNS activity is shown
on the right.

```
                    P-NODE REGISTRATION PROCESS
            (server HAS a previous owner that IS active)


     <------NON-SECURED STYLE-------> <---------SECURED STYLE------->

     REQ. NODE           NBNS        NODE        NBNS        REQ.NODE
                                     HOLDING
                                      NAME

           REGISTER                                REGISTER
     ------------------->                   <-------------------
                                         QUERY
      END-NODE CHALLENGE              <------------
     <-------------------                QUERY
                                     <-----------
              QUERY
     --------------------------->
                                      POSITIVE RESP
              QUERY                  ----------->
     --------------------------->              NEGATIVE RESPONSE
                                               ----------------->

           POSITIVE RESPONSE
     <---------------------------
```
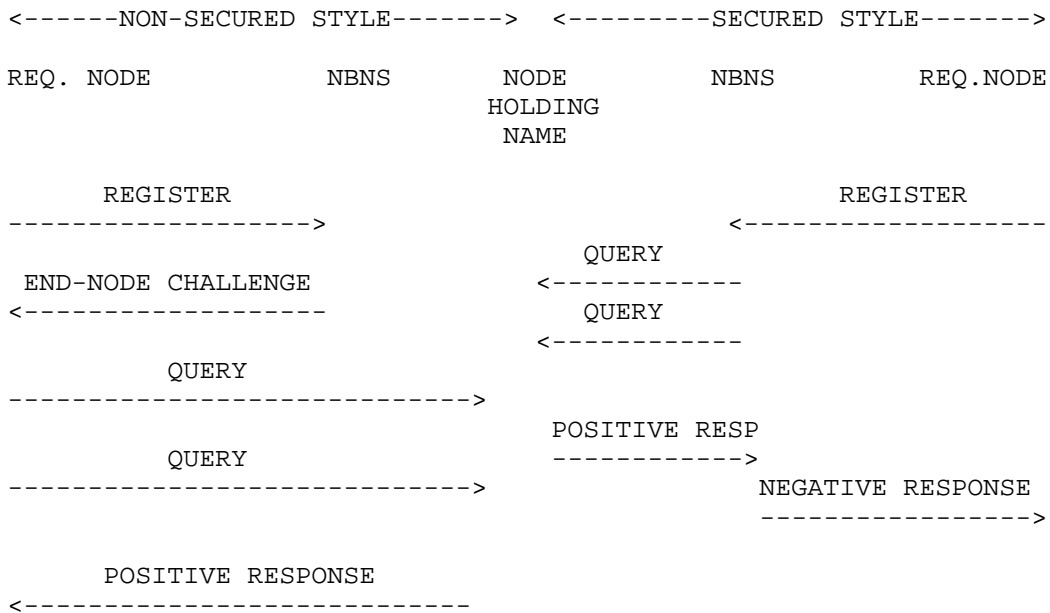
A non-secured NBNS will answer the NAME REGISTRATION REQUEST with a
END-NODE CHALLENGE REGISTRATION RESPONSE. This response asks the
end-node to issue a challenge transaction against the node indicated
in the response. In this case, the prior node will defend against
the challenge and the registering end-node will simply drop the
registration attempt without further interaction with the NBNS.

A secured NBNS will refrain from answering the NAME REGISTRATION
REQUEST until the NBNS has itself challenged the prior holder(s) of
the name. In this case, the NBNS finds that that the name is still
being defended and consequently returns a NEGATIVE NAME REGISTRATION
RESPONSE to the registrant.

Due to the potential time for the secured NBNS to make the
challenge(s), it is likely that a WACK will be sent by the NBNS to
the registrant.

Although not shown in the diagram, a non-secured NBNS will send a
NEGATIVE NAME REGISTRATION RESPONSE to a request to register a unique
name when there already exists a group of the same name. A secured
NBNS may elect to poll (or challenge) the group members to determine
whether any active members remain. This may impose a heavy load on
the network. It is recommended that group names be allowed to fade-
out through the name refresh mechanism.

15.2.2.3. EXISTING NAME AND OWNER IS INACTIVE

The following diagram shows interactions when an attempt is made to
register a unique name, the NBNS is aware of an existing owner, and
that existing owner is no longer active.

A non-secured NBNS will answer the NAME REGISTRATION REQUEST with a
END-NODE CHALLENGE REGISTRATION RESPONSE. This response asks the
end-node to issue a challenge transaction against the node indicated
in the response. In this case, the prior node will not defend
against the challenge. The registrant will inform the NBNS through a
NAME OVERWRITE REQUEST. The NBNS will replace the prior name
information in its database with the information in the overwrite
request.

A secured NBNS will refrain from answering the NAME REGISTRATION
REQUEST until the NBNS has itself challenged the prior holder(s) of
the name. In this case, the NBNS finds that that the name is not
being defended and consequently returns a POSITIVE NAME REGISTRATION
RESPONSE to the registrant.

                       P-NODE REGISTRATION PROCESS
                 (server HAS a previous owner that is NOT active)
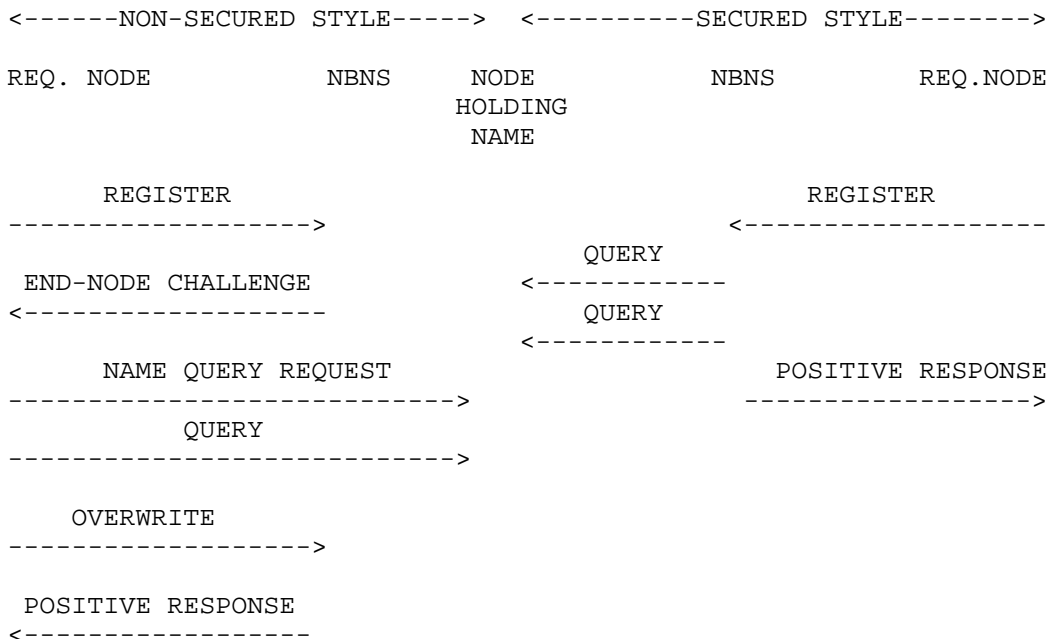
     <------NON-SECURED STYLE-----> <----------SECURED STYLE-------->

   REQ. NODE   NBNS   NODE   NBNS   REQ.NODE
                                    HOLDING
                                    NAME

         REGISTER   REGISTER
   ------------------->   <------------------
                                    QUERY
    END-NODE CHALLENGE   <------------
   <------------------   QUERY
                                    <------------
         NAME QUERY REQUEST   POSITIVE RESPONSE
   --------------------------->   ------------------->
               QUERY
   --------------------------->

       OVERWRITE
   ------------------->

    POSITIVE RESPONSE
   <------------------

   Due to the potential time for the secured NBNS to make the
   challenge(s), it is likely that a WACK will be sent by the NBNS to
   the registrant.

   A secured NBNS will immediately send a NEGATIVE NAME REGISTRATION
   RESPONSE in answer to any NAME OVERWRITE REQUESTS it may receive.

15.2.3. NAME REGISTRATION BY M NODES

   An M node begin a name claim operation as if the node were a B node:
   it broadcasts a NAME REGISTRATION REQUEST and listens for NEGATIVE
   NAME REGISTRATION RESPONSEs. Any NEGATIVE NAME REGISTRATION RESPONSE
   prevents the M node from obtaining the name and terminates the claim
   operation.

   If, however, the M node does not receive any NEGATIVE NAME
   REGISTRATION RESPONSE, the M node must continue the claim procedure
   as if the M node were a P node.

   Only if both name claims were successful does the M node acquire the
   name.

   The following diagram illustrates M node name registration:

                        M-NODE REGISTRATION PROCESS

     <---NAME NOT IN BROADCAST AREA--> <--NAME IS IN BROADCAST AREA-->

     REQ. NODE   NODE   REQ.NODE
                                       HOLDING
                                       NAME

     (BROADCAST) REGISTER   (BROADCAST) REGISTER
     ------------------->   <-------------------

          REGISTER   REGISTER
     ------------------->   <-------------------

          REGISTER   NEGATIVE RESPONSE
     ------------------->   ------------------------------->

                    !   (NODE DOES NOT HAVE THE NAME)
       INITIATE   !
       A P-NODE   !
       REGISTRATION !
                    V
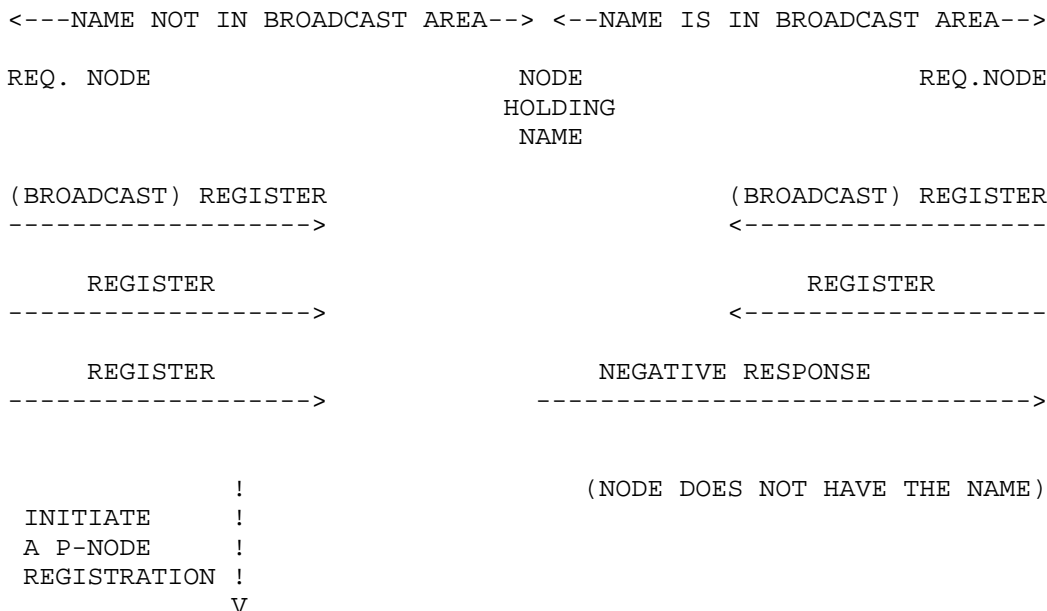
15.3. NAME QUERY TRANSACTIONS

    Name query transactions are initiated by end-nodes to obtain the IP
    address(es) and other attributes associated with a NetBIOS name.

15.3.1. QUERY BY B NODES

    The following diagram shows how B nodes go about discovering who owns
    a name.

    The left half of the diagram illustrates what happens if there are no
    holders of the name. In that case no responses are received in
    answer to the broadcast NAME QUERY REQUEST(s).

    The right half shows a POSITIVE NAME QUERY RESPONSE unicast by a name
    holder in answer to the broadcast request. A name holder will make
    this response to every NAME QUERY REQUEST that it hears. And each
    holder acts this way. Thus, the node sending the request may receive
    many responses, some duplicates, and from many nodes.

                        B-NODE DISCOVERY PROCESS

      <------NAME NOT ON NETWORK------> <---NAME PRESENT ON NETWORK-->

        REQ. NODE   NODE   REQ.NODE
                                   HOLDING
                                    NAME

          (BROADCAST) QUERY   (BROADCAST) QUERY
     ---------------------->   <----------------------

          NAME QUERY REQUEST   NAME QUERY REQUEST
     ---------------------->   <----------------------

             QUERY   POSITIVE RESPONSE
     ---------------------->   ------------------------------>
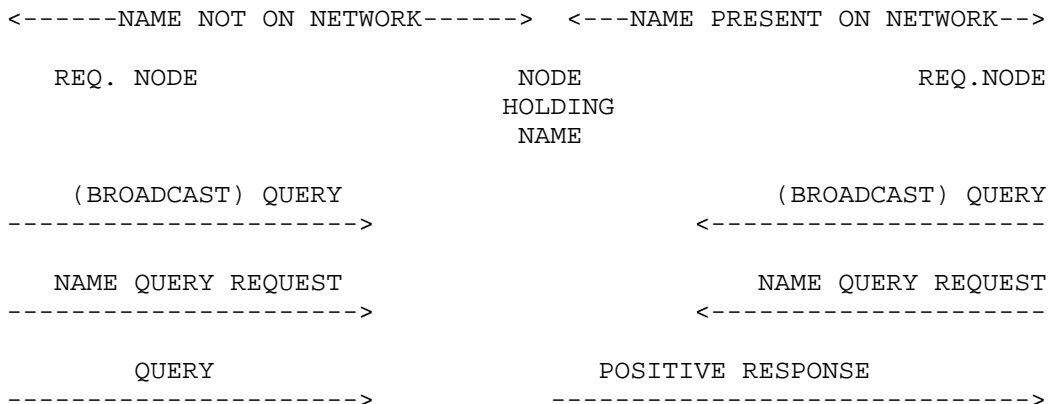
    Name query is generally, but not necessarily, a prelude to NetBIOS
    session establishment or NetBIOS datagram transmission. However,
    name query may be used for other purposes.

    A B node may elect to build a group membership list for subsequent
    use (e.g. for session establishment) by collecting and saving the
    responses.

15.3.2. QUERY BY P NODES

    An NBNS answers queries from a P node with a list of IP address and
    other information for each owner of the name. If there are multiple
    owners (i.e. if the name is a group name), the NBNS loads as many
    answers into the response as will fit into a UDP packet. A
    truncation flag indicates whether any additional owner information
    remains. All the information may be obtained by repeating the query
    over a TCP connection.

    The NBNS is not required to impose any order on its answer list.

    The following diagram shows what happens if the NBNS has no
    information about the name:

                    P-NODE DISCOVERY PROCESS
              (server has no information about the name)

          P-NODE   NBNS
                    NAME QUERY REQUEST
          --------------------------------->

                    NEGATIVE RESPONSE
          <---------------------------------
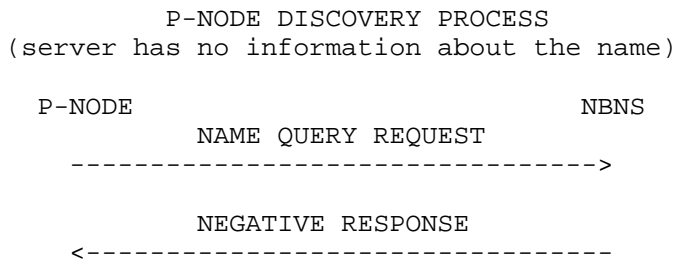
The next diagram illustrates interaction between the end-node and the
NBNS when the NBNS does have information about the name. This
diagram shows, in addition, the retransmission of the request by the
end-node in the absence of a timely response. Also shown are WACKs
(or temporary, intermediate responses) sent by the NBNS to the end-
node:

```
                     P-NODE QUERY PROCESS
           (server HAS information about the name)


      P-NODE                                           NBNS
                         NAME QUERY REQUEST
       /-------------------------------------->
      /
      !            (OPTIONAL)   WACK
      !   <- - - - - - - - - - - - - - - - - - -
      !          !
      !timer     !
      !          ! (optional timer restart)
      !          !
       \         V            QUERY
        \-------------------------------------->
                        .
                        .
                        .
                             QUERY
       /-------------------------------------->
      /
      !            (OPTIONAL)   WACK
      !   <- - - - - - - - - - - - - - - - - - -
      !          !
      !timer     !
      !          ! (optional timer restart)
      !          !
       \         V            QUERY
        \-------------------------------------->
                        .
                        .

                     POSITIVE RESPONSE
         <--------------------------------------
```
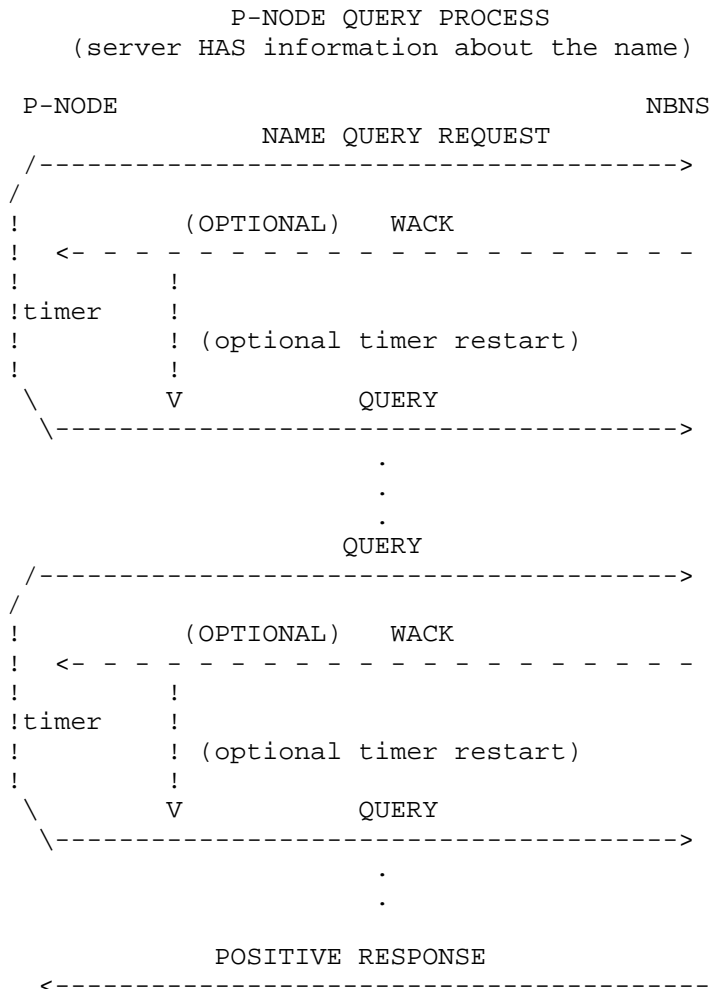
The following diagram illustrates NBNS redirection. Upon receipt of
a NAME QUERY REQUEST, the NBNS redirects the client to another NBNS.
The client repeats the request to the new NBNS and obtains a
response. The diagram shows that response as a POSITIVE NAME QUERY
RESPONSE. However any legal NBNS response may occur in actual
operation.
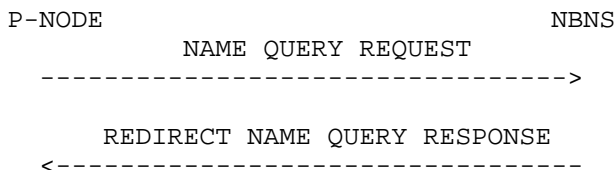
```
                        NBNS REDIRECTION

             P-NODE                              NBNS
                       NAME QUERY REQUEST
             --------------------------------->

                    REDIRECT NAME QUERY RESPONSE
                 <---------------------------------

              (START FROM THE
               VERY BEGINNING
               USING THE ADDRESS
               OF THE NEWLY
               SUPPLIED NBNS.)
                                                 NEW
             P-NODE                              NBNS
                       NAME QUERY REQUEST
             --------------------------------->

                   POSITIVE NAME QUERY RESPONSE
                 <---------------------------------
```

The next diagram shows how a P or M node tells the NBNS that the NBNS
has provided incorrect information. This procedure may begin after a
DATAGRAM ERROR packet has been received or a session set-up attempt
has discovered that the NetBIOS name does not exist at the
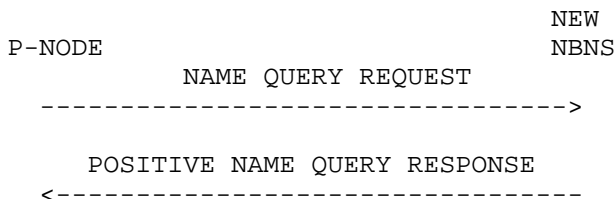destination, the IP address of which was obtained from the NBNS
during a prior name query transaction. The NBNS, in this case a
secure NBNS, issues queries to verify whether the information is, in
fact, incorrect. The NBNS closes the transaction by sending either a
POSITIVE or NEGATIVE NAME RELEASE RESPONSE, depending on the results
of the verification.

```
                   CORRECTING NBNS INFORMATION BASE

             P-NODE                              NBNS
                      NAME RELEASE REQUEST
             --------------------------------->
                                                      QUERY
                                                 --------------->

                                                      QUERY
                                                 --------------->

                                     (NAME TAKEN OFF THE DATABASE
                                      IF NBNS FINDS IT TO BE
                                      INCORRECT)

                      POSITIVE/NEGATIVE RESPONSE
                 <---------------------------------
```
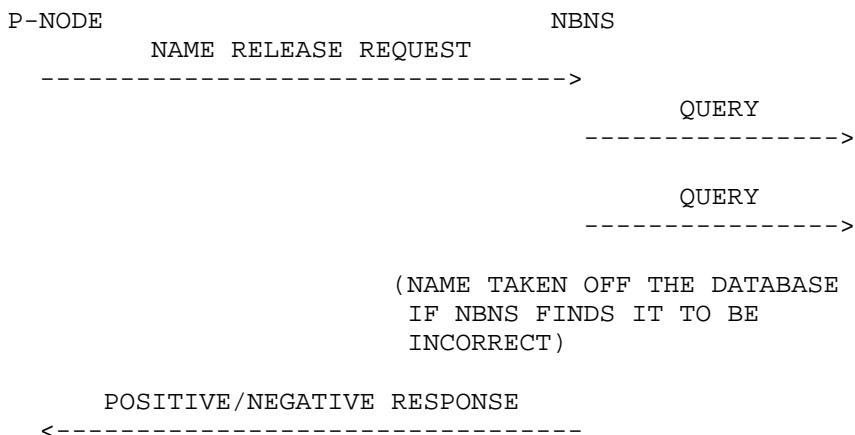
15.3.3. QUERY BY M NODES

    M node name query follows the B node pattern. In the absence of
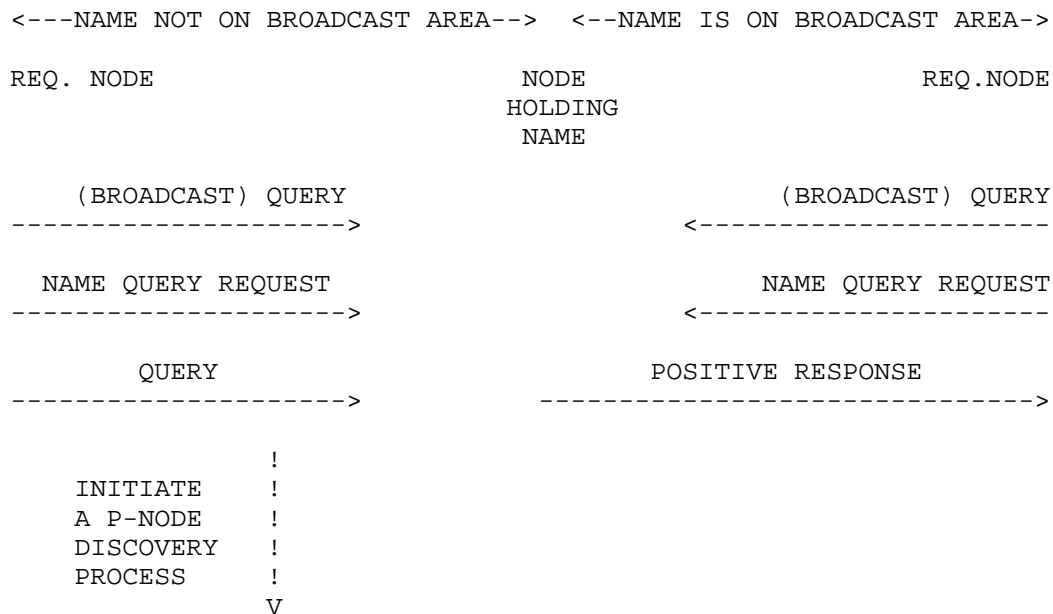   adequate results, the M node then continues by performing a P node
    type query. This is shown in the following diagram:

                        M-NODE DISCOVERY PROCESS

    <---NAME NOT ON BROADCAST AREA--> <--NAME IS ON BROADCAST AREA->

    REQ. NODE   NODE   REQ.NODE
                                       HOLDING
                                       NAME

         (BROADCAST) QUERY   (BROADCAST) QUERY
    --------------------->   <----------------------

       NAME QUERY REQUEST   NAME QUERY REQUEST
    --------------------->   <----------------------

            QUERY   POSITIVE RESPONSE
    --------------------->   ------------------------------->

                     !
          INITIATE   !
          A P-NODE   !
          DISCOVERY !
          PROCESS   !
                     V

15.3.4. ACQUIRE GROUP MEMBERSHIP LIST

    The entire membership of a group may be acquired by sending a NAME
    QUERY REQUEST to the NBNS. The NBNS will respond with a POSITIVE
    NAME QUERY RESPONSE or a NEGATIVE NAME QUERY RESPONSE. A negative
    response completes the procedure and indicates that there are no
   members in the group.

    If the positive response has the truncation bit clear, then the
    response contains the entire list of group members. If the
    truncation bit is set, then this entire procedure must be repeated,
    but using TCP as a foundation rather than UDP.

15.4. NAME RELEASE TRANSACTIONS

15.4.1. RELEASE BY B NODES

   A NAME RELEASE DEMAND contains the following information:

      - NetBIOS name
      - The scope of the NetBIOS name
      - Name type: unique or group
      - IP address of the releasing node
      - Transaction ID

   REQUESTING   OTHER
   B-NODE   B-NODES
                     NAME RELEASE DEMAND
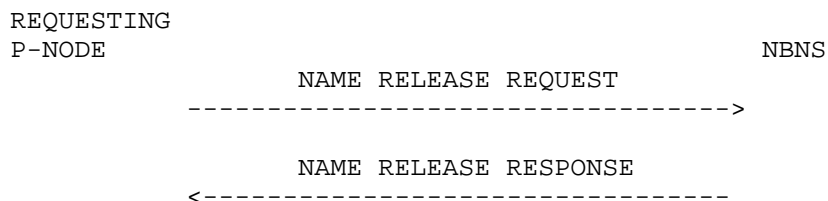   ---------------------------------->
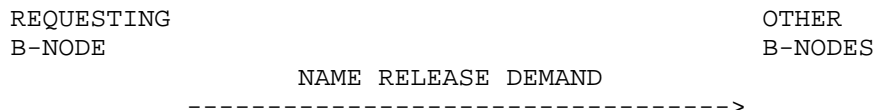
15.4.2. RELEASE BY P NODES

   A NAME RELEASE REQUEST contains the following information:

      - NetBIOS name
      - The scope of the NetBIOS name
      - Name type: unique or group
      - IP address of the releasing node
      - Transaction ID

   A NAME RELEASE RESPONSE contains the following information:

      - NetBIOS name
      - The scope of the NetBIOS name
      - Name type: unique or group
      - IP address of the releasing node
      - Transaction ID
      - Result:
         - Yes: name was released
         - No: name was not released, a reason code is provided

   REQUESTING
   P-NODE   NBNS
                     NAME RELEASE REQUEST
          ---------------------------------->

                     NAME RELEASE RESPONSE
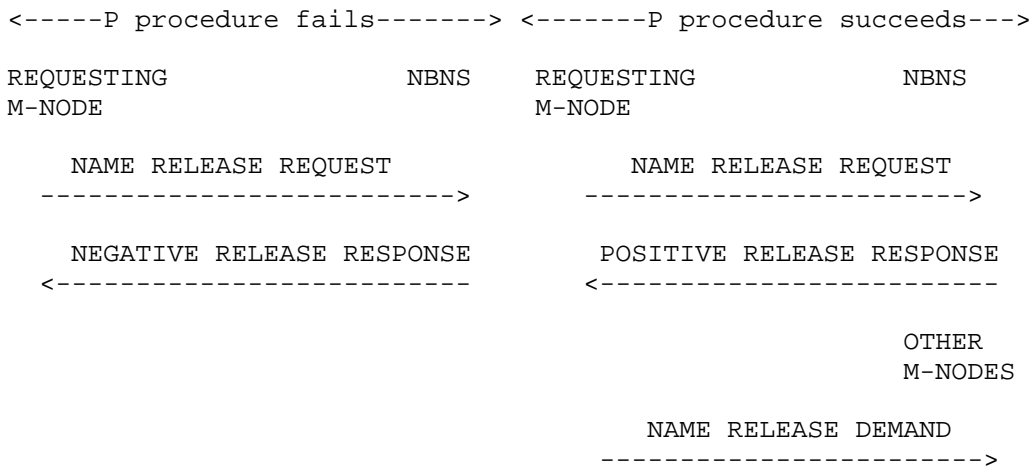   <---------------------------------

15.4.3. RELEASE BY M NODES

   The name release procedure of the M node is a combination of the P
   and B node name release procedures. The M node first performs the P

release procedure. If the P procedure fails then the release
procedure does not continue, it fails. If and only if the P
procedure succeeds then the M node broadcasts the NAME RELEASE DEMAND
to the broadcast area, the B procedure.

NOTE: An M node typically performs a B-style operation and then a
      P-style operation. In this case, however, the P-style
      operation comes first.

The following diagram illustrates the M node name release procedure:

<-----P procedure fails-------> <-------P procedure succeeds--->

REQUESTING   NBNS   REQUESTING   NBNS
M-NODE   M-NODE

     NAME RELEASE REQUEST   NAME RELEASE REQUEST
   ------------------------->   ------------------------>

     NEGATIVE RELEASE RESPONSE   POSITIVE RELEASE RESPONSE
   <------------------------   <------------------------

                                               OTHER
                                               M-NODES

                                        NAME RELEASE DEMAND
                                      ------------------------>

15.5. NAME MAINTENANCE TRANSACTIONS
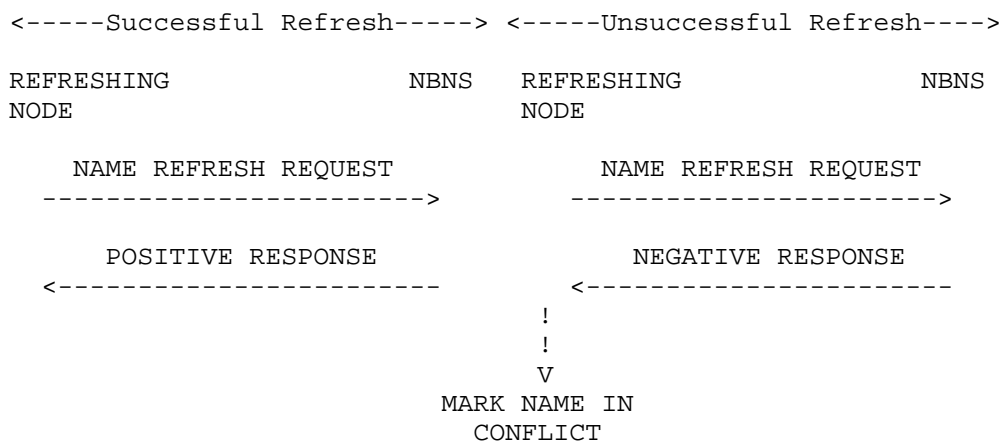
15.5.1. NAME REFRESH

   Name refresh transactions are used to handle the following
   situations:

      a)   An NBNS node needs to detect if a P or M node has "silently"
           gone down, so that names held by that node can be purged
           from the data base.

      b)   If the NBNS goes down, it needs to be able to reconstruct
           the data base when it comes back up.

      c)   If the network should be partitioned, the NBNS needs to be
           able to able to update its data base when the network
           reconnects.

   Each P or M node is responsible for sending periodic NAME REFRESH
   REQUESTs for each name that it has registered. Each refresh packet
   contains a single name that has been successfully registered by that

node. The interval between such packets is negotiated between the
end node and the NBNS server at the time that the name is initially
claimed. At name claim time, an end node will suggest a refresh
timeout value. The NBNS node can modify this value in the reply
packet. A NBNS node can also choose to tell the end node to not send
any refresh packet by using the "infinite" timeout value in the
response packet. The timeout value returned by the NBNS is the
actual refresh timeout that the end node must use.

When a node sends a NAME REFRESH REQUEST, it must be prepared to
receive a negative response. This would happen, for example, if the
the NBNS discovers that the the name had already been assigned to
some other node. If such a response is received, the end node should
mark the name as being in conflict. Such an entry should be treated
in the same way as if name conflict had been detected against the
name. The following diagram illustrates name refresh:

```
<-----Successful Refresh-----> <-----Unsuccessful Refresh---->


REFRESHING                 NBNS   REFRESHING                 NBNS
NODE                              NODE

    NAME REFRESH REQUEST             NAME REFRESH REQUEST
   ------------------------>        ------------------------>

       POSITIVE RESPONSE               NEGATIVE RESPONSE
   <------------------------        <------------------------
                                                 !
                                                 !
                                                 V
                                          MARK NAME IN
                                            CONFLICT
```

15.5.2. NAME CHALLENGE

   Name challenge is done by sending a NAME QUERY REQUEST to an end node
   of any type. If a POSITIVE NAME QUERY RESPONSE is returned, then
   that node still owns the name. If a NEGATIVE NAME QUERY RESPONSE is
   received or if no response is received, it can be assumed that the
   end node no longer owns the name.

   Name challenge can be performed either by the NBNS node, or by an end
   node. When an end-node sends a name claim packet, the NBNS node may
   do the challenge operation. The NBNS node can choose, however, to
   require the end node do the challenge. In that case, the NBNS will
   send an END-NODE CHALLENGE RESPONSE packet to the end node, which
   should then proceed to challenge the putative owner.

   Note that the name challenge procedure sends a normal NAME QUERY
   REQUEST packet to the end node. It does not require a special
   packet. The only new packet introduced is the END-NODE CHALLENGE

RESPONSE which is sent by an NBNS node when the NBNS wants the end-
node to perform the challenge operation.

15.5.3. CLEAR NAME CONFLICT

It is possible during a refresh request from a M or P node for a NBNS
to detects a name in conflict. The response to the NAME REFRESH
REQUEST must be a NEGATIVE NAME REGISTRATION RESPONSE. Optionally,
in addition, the NBNS may send a NAME CONFLICT DEMAND or a NAME
RELEASE REQUEST to the refreshing node. The NAME CONFLICT DEMAND
forces the node to place the name in the conflict state. The node
will eventually inform it's user of the conflict. The NAME RELEASE
REQUEST will force the node to flush the name from its local name
table completely. This forces the node to flush the name in
conflict. This does not cause termination of existing sessions using
this name.

The following diagram shows an NBNS detecting and correcting a
conflict:

REFRESHING NODE   NBNS

                    NAME REFRESH REQUEST
          -->

              NEGATIVE NAME REGISTRATION RESPONSE
          <--

                    NAME CONFLICT DEMAND
          <--

                            OR

                    NAME RELEASE REQUEST
          <--

                 POSITIVE/NEGATIVE RELEASE REQUEST
          -->
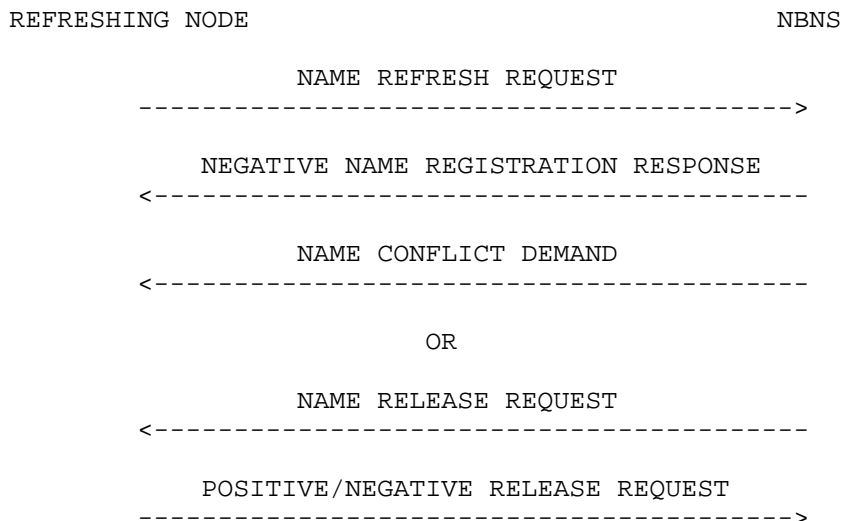
15.6. ADAPTER STATUS TRANSACTIONS

Adapter status is obtained from a node as follows:

    1.   Perform a name discovery operation to obtain the IP
         addresses of a set of end-nodes.

    2.   Repeat until all end-nodes from the set have been used:

         a.   Select one end-node from the set.

         b.   Send a NODE STATUS REQUEST to that end-node using UDP.

c.   Await a NODE STATUS RESPONSE. (If a timely response is
     not forthcoming, repeat step "b" UCAST_REQ_RETRY_COUNT
     times. After the last retry, go to step "a".)

d.   If the truncation bit is not set in the response, the
     response contains the entire node status. Return the
     status to the user and terminate this procedure.

e.   If the truncation bit is set in the response, then not
     all status was returned because it would not fit into
     the response packet. The responder will set the
     truncation bit if the IP datagram length would exceed
     MAX_DATAGRAM_LENGTH. Return the status to the user and
     terminate this procedure.

3.   Return error to user, no status obtained.

     The repetition of step 2, above, through all nodes of the set, is
     optional.

     Following is an example transaction of a successful Adapter Status
     operation:

     REQUESTING NODE   NAME OWNER

                         NAME QUERY REQUEST
             -->

                      POSITIVE NAME QUERY RESPONSE
             <--

                         NODE STATUS REQUEST
             -->

                         NODE STATUS RESPONSE
             <--

16. NetBIOS SESSION SERVICE

     The NetBIOS session service begins after one or more IP addresses
     have been found for the target name. These addresses may have been
     acquired using the NetBIOS name query transactions or by other means,
     such as a local name table or cache.

     NetBIOS session service transactions, packets, and protocols are
     identical for all end-node types. They involve only directed
     (point-to-point) communications.

16.1. OVERVIEW OF NetBIOS SESSION SERVICE

   Session service has three phases:

      Session establishment - it is during this phase that the IP
         address and TCP port of the called name is determined, and a
         TCP connection is established with the remote party.

      Steady state - it is during this phase that NetBIOS data
         messages are exchanged over the session. Keep-alive packets
         may also be exchanged if the participating nodes are so
         configured.

      Session close - a session is closed whenever either a party (in
         the session) closes the session or it is determined that one
         of the parties has gone down.

16.1.1. SESSION ESTABLISHMENT PHASE OVERVIEW

   An end-node begins establishment of a session to another node by
   somehow acquiring (perhaps using the name query transactions or a
   local cache) the IP address of the node or nodes purported to own the
   destination name.

   Every end-node awaits incoming NetBIOS session requests by listening
   for TCP calls to a well-known service port, SSN_SRVC_TCP_PORT. Each
   incoming TCP connection represents the start of a separate NetBIOS
   session initiation attempt. The NetBIOS session server, not the
   ultimate application, accepts the incoming TCP connection(s).

   Once the TCP connection is open, the calling node sends session
   service request packet. This packet contains the following
   information:

      - Calling IP address (see note)
      - Calling NetBIOS name
      - Called IP address (see note)
      - Called NetBIOS name

   NOTE: The IP addresses are obtained from the TCP service
          interface.

   When the session service request packet arrives at the NetBIOS
   server, one of the the following situations will exist:

   -     There exists a NetBIOS LISTEN compatible with the incoming
         call and there are adequate resources to permit session
         establishment to proceed.

   -     There exists a NetBIOS LISTEN compatible with the incoming
         call, but there are inadequate resources to permit

establishment of a session.

-       The called name does, in fact, exist on the called node, but
        there is no pending NetBIOS LISTEN compatible with the
        incoming call.

-       The called name does not exist on the called node.

In all but the first case, a rejection response is sent back over the
TCP connection to the caller. The TCP connection is then closed and
the session phase terminates. Any retry is the responsibility of the
caller. For retries in the case of a group name, the caller may use
the next member of the group rather than immediately retrying the
instant address. In the case of a unique name, the caller may
attempt an immediate retry using the same target IP address unless
the called name did not exist on the called node. In that one case,
the NetBIOS name should be re-resolved.

If a compatible LISTEN exists, and there are adequate resources, then
the session server may transform the existing TCP connection into the
NetBIOS data session. Alternatively, the session server may
redirect, or "retarget" the caller to another TCP port (and IP
address).

If the caller is redirected, the caller begins the session
establishment anew, but using the new IP address and TCP port given
in the retarget response. Again a TCP connection is created, and
again the calling and called node exchange credentials. The called
party may accept the call, reject the call, or make a further
redirection.

This mechanism is based on the presumption that, on hosts where it is
not possible to transfer open TCP connections between processes, the
host will have a central session server. Applications willing to
receive NetBIOS calls will obtain an ephemeral TCP port number, post
a TCP unspecified passive open on that port, and then pass that port
number and NetBIOS name information to the NetBIOS session server
using a NetBIOS LISTEN operation. When the call is placed, the
session server will "retarget" the caller to the application's TCP
socket. The caller will then place a new call, directly to the
application. The application has the responsibility to mimic the
session server at least to the extent of receiving the calling
credentials and then accepting or rejecting the call.

16.1.1.1. RETRYING AFTER BEING RETARGETTED

A calling node may find that it can not establish a session with a
node to which it was directed by the retargetting procedure. Since
retargetting may be nested, there is an issue whether the caller
should begin a retry at the initial starting point or back-up to an
intermediate retargetting point. The caller may use any method. A

discussion of two such methods is in Appendix B, "Retarget
Algorithms".

16.1.1.2. SESSION ESTABLISHMENT TO A GROUP NAME

Session establishment with a group name requires special
consideration. When a NetBIOS CALL attempt is made to a group name,
name discovery will result in a list (possibly incomplete) of the
members of that group. The calling node selects one member from the
list and attempts to build a session. If that fails, the calling
node may select another member and make another attempt.

When the session service attempts to make a connection with one of
the members of the group, there is no guarantee that that member has
a LISTEN pending against that group name, that the called node even
owns, or even that the called node is operating.

16.1.2. STEADY STATE PHASE OVERVIEW

NetBIOS data messages are exchanged in the steady state. NetBIOS
messages are sent by prepending the user data with a message header
and sending the header and the user data over the TCP connection.
The receiver removes the header and passes the data to the NetBIOS
user.

In order to detect failure of one of the nodes or of the intervening
network, "session keep alive" packets may be periodically sent in the
steady state.

Any failure of the underlying TCP connection, whether a reset, a
timeout, or other failure, implies failure of the NetBIOS session.

16.1.3. SESSION TERMINATION PHASE OVERVIEW

A NetBIOS session is terminated normally when the user requests the
session to be closed or when the session service detects the remote
partner of the session has gracefully terminated the TCP connection.
A NetBIOS session is abnormally terminated when the session service
detects a loss of the connection. Connection loss can be detected
with the keep-alive function of the session service or TCP, or on the
failure of a SESSION MESSAGE send operation.

When a user requests to close a session, the service first attempts a
graceful in-band close of the TCP connection. If the connection does
not close within the SSN_CLOSE_TIMEOUT the TCP connection is aborted.
No matter how the TCP connection is terminated, the NetBIOS session
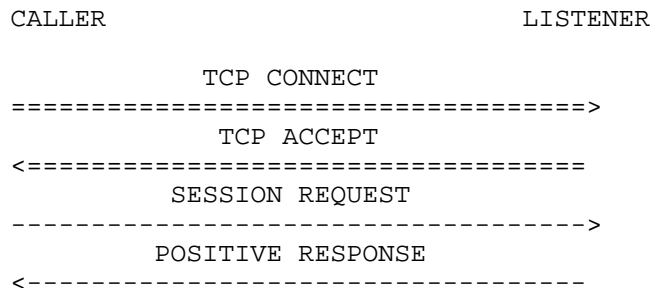service always closes the NetBIOS session.

When the session service receives an indication from TCP that a
connection close request has been received, the TCP connection and
the NetBIOS session are immediately closed and the user is informed

of the loss of the session. All data received up to the close
indication should be delivered, if possible, to the session's user.
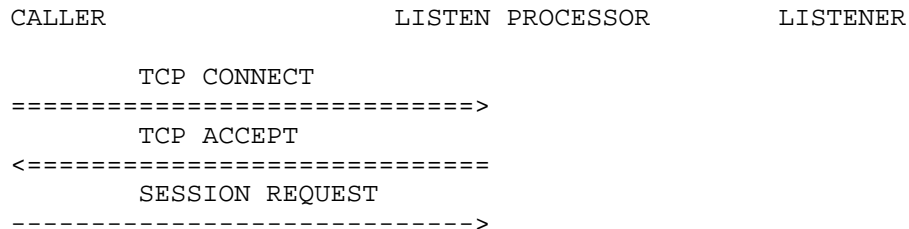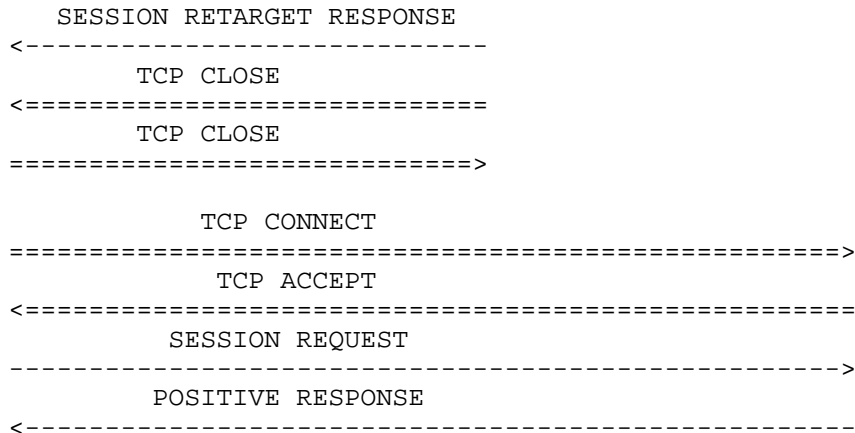
16.2. SESSION ESTABLISHMENT PHASE

All the following diagrams assume a name query operation was
successfully completed by the caller node for the listener's name.

This first diagram shows the sequence of network events used to
successfully establish a session without retargetting by the
listener. The TCP connection is first established with the well-
known NetBIOS session service TCP port, SSN_SRVC_TCP_PORT. The
caller then sends a SESSION REQUEST packet over the TCP connection
requesting a session with the listener. The SESSION REQUEST contains
the caller's name and the listener's name. The listener responds
with a POSITIVE SESSION RESPONSE informing the caller this TCP
connection is accepted as the connection for the data transfer phase
of the session.

```
         CALLER                              LISTENER

                 TCP CONNECT
         ===================================>
                  TCP ACCEPT
         <===================================
               SESSION REQUEST
         ------------------------------------>
              POSITIVE RESPONSE
         <-----------------------------------
```
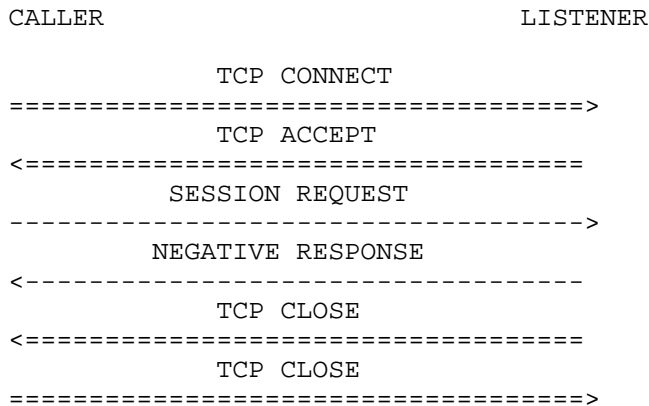
The second diagram shows the sequence of network events used to
successfully establish a session when the listener does retargetting.
The session establishment procedure is the same as with the first
diagram up to the listener's response to the SESSION REQUEST. The
listener, divided into two sections, the listen processor and the
actual listener, sends a SESSION RETARGET RESPONSE to the caller.
This response states the call is acceptable, but the data transfer
TCP connection must be at the new IP address and TCP port. The
caller then re-iterates the session establishment process anew with
the new IP address and TCP port after the initial TCP connection is
closed. The new listener then accepts this connection for the data
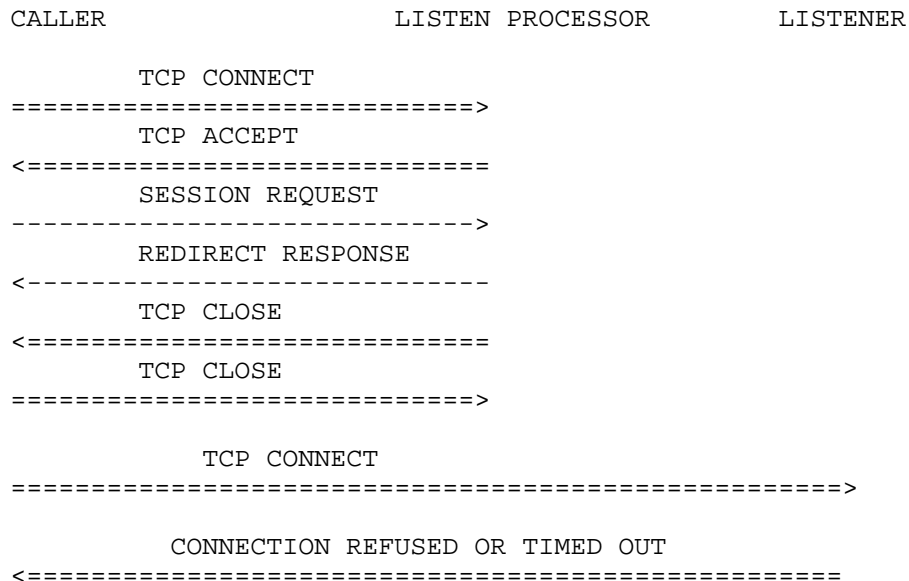transfer phase with a POSITIVE SESSION RESPONSE.

```
         CALLER                    LISTEN PROCESSOR         LISTENER

                TCP CONNECT
         ==============================>
                TCP ACCEPT
         <==============================
                SESSION REQUEST
         ------------------------------>
```

```
            SESSION RETARGET RESPONSE
       <----------------------------
                 TCP CLOSE
       <============================
                 TCP CLOSE
       ============================>


                    TCP CONNECT
       =====================================================>
                     TCP ACCEPT
       <=====================================================
                  SESSION REQUEST
       ----------------------------------------------------->
                  POSITIVE RESPONSE
       <-----------------------------------------------------
```

   The third diagram is the sequence of network events for a rejected
   session request with the listener. This type of rejection could
   occur with either a non-retargetting listener or a retargetting
   listener. After the TCP connection is established at
   SSN_SRVC_TCP_PORT, the caller sends the SESSION REQUEST over the TCP
   connection. The listener does not have either a listen pending for
   the listener's name or the pending NetBIOS listen is specific to
   another caller's name. Consequently, the listener sends a NEGATIVE
   SESSION RESPONSE and closes the TCP connection.

```
              CALLER                            LISTENER

                      TCP CONNECT
       ===================================>
                      TCP ACCEPT
       <===================================
                   SESSION REQUEST
       ----------------------------------->
                  NEGATIVE RESPONSE
       <-----------------------------------
                      TCP CLOSE
       <===================================
                      TCP CLOSE
       ===================================>
```

   The fourth diagram is the sequence of network events when session
   establishment fails with a retargetting listener. After being
   redirected, and after the initial TCP connection is closed the caller
   tries to establish a TCP connection with the new IP address and TCP
   port. The connection fails because either the port is unavailable or
   the target node is not active. The port unavailable race condition
   occurs if another caller has already acquired the TCP connection with
   the listener. For additional implementation suggestions, see
   Appendix B, "Retarget Algorithms".

```
         CALLER                 LISTEN PROCESSOR        LISTENER

              TCP CONNECT
         ============================>
              TCP ACCEPT
         <============================
              SESSION REQUEST
         ---------------------------->
              REDIRECT RESPONSE
         <----------------------------
              TCP CLOSE
         <============================
              TCP CLOSE
         ============================>


                 TCP CONNECT
         ====================================================>

                 CONNECTION REFUSED OR TIMED OUT
         <====================================================
```

16.3. SESSION DATA TRANSFER PHASE

16.3.1. DATA ENCAPSULATION

   NetBIOS messages are exchanged in the steady state. Messages are
   sent by prepending user data with message header and sending the
   header and the user data over the TCP connection. The receiver
   removes the header and delivers the NetBIOS data to the user.

16.3.2. SESSION KEEP-ALIVES

   In order to detect node failure or network partitioning, "session
   keep alive" packets are periodically sent in the steady state. A
   session keep alive packet is discarded by a peer node.

   A session keep alive timer is maintained for each session. This
   timer is reset whenever any data is sent to, or received from, the
   session peer. When the timer expires, a NetBIOS session keep-alive
   packet is sent on the TCP connection. Sending the keep-alive packet
   forces data to flow on the TCP connection, thus indirectly causing
   TCP to detect whether the connection is still active.

   Since many TCP implementations provide a parallel TCP "keep- alive"
   mechanism, the NetBIOS session keep-alive is made a configurable
   option. It is recommended that the NetBIOS keep- alive mechanism be
   used only in the absence of TCP keep-alive.

   Note that unlike TCP keep alives, NetBIOS session keep alives do not
   require a response from the NetBIOS peer -- the fact that it was

possible to send the NetBIOS session keep alive is sufficient
indication that the peer, and the connection to it, are still active.

The only requirement for interoperability is that when a session keep
alive packet is received, it should be discarded.

17. NETBIOS DATAGRAM SERVICE

17.1. OVERVIEW OF NetBIOS DATAGRAM SERVICE

Every NetBIOS datagram has a named destination and source. To
transmit a NetBIOS datagram, the datagram service must perform a name
query operation to learn the IP address and the attributes of the
destination NetBIOS name. (This information may be cached to avoid
the overhead of name query on subsequent NetBIOS datagrams.)

NetBIOS datagrams are carried within UDP packets. If a NetBIOS
datagram is larger than a single UDP packet, it may be fragmented
into several UDP packets.

End-nodes may receive NetBIOS datagrams addressed to names not held
by the receiving node. Such datagrams should be discarded. If the
name is unique then a DATAGRAM ERROR packet is sent to the source of
that NetBIOS datagram.

17.1.1. UNICAST, MULTICAST, AND BROADCAST

NetBIOS datagrams may be unicast, multicast, or broadcast. A NetBIOS
datagram addressed to a unique NetBIOS name is unicast. A NetBIOS
datatgram addressed to a group NetBIOS name, whether there are zero,
one, or more actual members, is multicast. A NetBIOS datagram sent
using the NetBIOS "Send Broadcast Datagram" primitive is broadcast.

17.1.2. FRAGMENTATION OF NetBIOS DATAGRAMS

When the header and data of a NetBIOS datagram exceeds the maximum
amount of data allowed in a UDP packet, the NetBIOS datagram must be
fragmented before transmission and reassembled upon receipt.

A NetBIOS Datagram is composed of the following protocol elements:

  - IP header of 20 bytes (minimum)
  - UDP header of 8 bytes
  - NetBIOS Datagram Header of 14 bytes
  - The NetBIOS Datagram data.

The NetBIOS Datagram data section is composed of 3 parts:

  - Source NetBIOS name (255 bytes maximum)
  - Destination NetBIOS name (255 bytes maximum)
  - The NetBIOS user's data (maximum of 512 bytes)

The two name fields are in second level encoded format (see section
14.)

A maximum size NetBIOS datagram is 1064 bytes. The minimal maximum
IP datagram size is 576 bytes. Consequently, a NetBIOS Datagram may
not fit into a single IP datagram. This makes it necessary to permit
the fragmentation of NetBIOS Datagrams.

On networks meeting or exceeding the minimum IP datagram length
requirement of 576 octets, at most two NetBIOS datagram fragments
will be generated. The protocols and packet formats accommodate
fragmentation into three or more parts.

When a NetBIOS datagram is fragmented, the IP, UDP and NetBIOS
Datagram headers are present in each fragment. The NetBIOS Datagram
data section is split among resulting UDP datagrams. The data
sections of NetBIOS datagram fragments do not overlap. The only
fields of the NetBIOS Datagram header that would vary are the FLAGS
and OFFSET fields.

The FIRST bit in the FLAGS field indicate whether the fragment is the
first in a sequence of fragments. The MORE bit in the FLAGS field
indicates whether other fragments follow.

The OFFSET field is the byte offset from the beginning of the NetBIOS
datagram data section to the first byte of the data section in a
fragment. It is 0 for the first fragment. For each subsequent
fragment, OFFSET is the sum of the bytes in the NetBIOS data sections
of all preceding fragments.

If the NetBIOS datagram was not fragmented:

  - FIRST = TRUE
  - MORE = FALSE
  - OFFSET = 0

If the NetBIOS datagram was fragmented:

  - First fragment:
      - FIRST = TRUE
      - MORE = TRUE
      - OFFSET = 0

  - Intermediate fragments:
      - FIRST = FALSE
      - MORE = TRUE
      - OFFSET = sum(NetBIOS data in prior fragments)

  - Last fragment:
      - FIRST = FALSE
      - MORE = FALSE

                - OFFSET = sum(NetBIOS data in prior fragments)

   The relative position of intermediate fragments may be ascertained
   from OFFSET.

   An NBDD must remember the destination name of the first fragment in
   order to relay the subsequent fragments of a single NetBIOS datagram.
   The name information can be associated with the subsequent fragments
   through the transaction ID, DGM_ID, and the SOURCE_IP, fields of the
   packet. This information can be purged by the NBDD after the last
   fragment has been processed or FRAGMENT_TO time has expired since the
   first fragment was received.

17.2. NetBIOS DATAGRAMS BY B NODES

   For NetBIOS datagrams with a named destination (i.e. non- broadcast),
   a B node performs a name discovery for the destination name before
   sending the datagram. (Name discovery may be bypassed if information
   from a previous discovery is held in a cache.) If the name type
   returned by name discovery is UNIQUE, the datagram is unicast to the
   sole owner of the name. If the name type is GROUP, the datagram is
   broadcast to the entire broadcast area using the destination IP
   address BROADCAST_ADDRESS.

   A receiving node always filters datagrams based on the destination
   name. If the destination name is not owned by the node or if no
   RECEIVE DATAGRAM user operations are pending for the name, then the
   datagram is discarded. For datagrams with a UNIQUE name destination,
   if the name is not owned by the node then the receiving node sends a
   DATAGRAM ERROR packet. The error packet originates from the
   DGM_SRVC_UDP_PORT and is addressed to the SOURCE_IP and SOURCE_PORT
   from the bad datagram. The receiving node quietly discards datagrams
   with a GROUP name destination if the name is not owned by the node.

   Since broadcast NetBIOS datagrams do not have a named destination,
   the B node sends the DATAGRAM SERVICE packet(s) to the entire
   broadcast area using the destination IP address BROADCAST_ADDRESS.
   In order for the receiving nodes to distinguish this datagram as a
   broadcast NetBIOS datagram, the NetBIOS name used as the destination
   name is '*' (hexadecimal 2A) followed by 15 bytes of hexidecimal 00.
   The NetBIOS scope identifier is appended to the name before it is
   converted into second-level encoding. For example, if the scope
   identifier is "NETBIOS.SCOPE" then the first-level encoded name would
   be:

        CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.NETBIOS.SCOPE

   According to [2], a user may not provide a NetBIOS name beginning
   with "*".

   For each node in the broadcast area that receives the NetBIOS

broadcast datagram, if any RECEIVE BROADCAST DATAGRAM user operations
are pending then the data from the NetBIOS datagram is replicated and
delivered to each. If no such operations are pending then the node
silently discards the datagram.

17.3. NetBIOS DATAGRAMS BY P AND M NODES

P and M nodes do not use IP broadcast to distribute NetBIOS
datagrams.

Like B nodes, P and M nodes must perform a name discovery or use
cached information to learn whether a destination name is a group or
a unique name.

Datagrams to unique names are unicast directly to the destination by
P and M nodes, exactly as they are by B nodes.

Datagrams to group names and NetBIOS broadcast datagrams are unicast
to the NBDD. The NBDD then relays the datagrams to each of the nodes
specified by the destination name.

An NBDD may not be capable of sending a NetBIOS datagram to a
particular NetBIOS name, including the broadcast NetBIOS name ("*")
defined above. A query mechanism is available to the end- node to
determine if a NBDD will be able to relay a datagram to a given name.
Before a datagram, or its fragments, are sent to the NBDD the P or M
node may send a DATAGRAM QUERY REQUEST packet to the NBDD with the
DESTINATION_NAME from the DATAGRAM SERVICE packet(s). The NBDD will
respond with a DATAGRAM POSITIVE QUERY RESPONSE if it will relay
datagrams to the specified destination name. After a positive
response the end-node unicasts the datagram to the NBDD. If the NBDD
will not be able to relay a datagram to the destination name
specified in the query, a DATAGRAM NEGATIVE QUERY RESPONSE packet is
returned. If the NBDD can not distribute a datagram, the end-node
then has the option of getting the name's owner list from the NBNS
and sending the datagram directly to each of the owners.

An NBDD must be able to respond to DATAGRAM QUERY REQUEST packets.
The response may always be positive. However, the usage or
implementation of the query mechanism by a P or M node is optional.
An implementation may always unicast the NetBIOS datagram to the NBDD
without asking if it will be relayed. Except for the datagram query
facility described above, an NBDD provides no feedback to indicate
whether it forwarded a datagram.

18. NODE CONFIGURATION PARAMETERS

      - B NODES:
            - Node's permanent unique name
            - Whether IGMP is in use
            - Broadcast IP address to use

```
           -  Whether NetBIOS session keep-alives are needed
           -  Usable UDP data field length (to control fragmentation)
        -  P NODES:
           -  Node's permanent unique name
           -  IP address of NBNS
           -  IP address of NBDD
           -  Whether NetBIOS session keep-alives are needed
           -  Usable UDP data field length (to control fragmentation)
        -  M NODES:
           -  Node's permanent unique name
           -  Whether IGMP is in use
           -  Broadcast IP address to use
           -  IP address of NBNS
           -  IP address of NBDD
           -  Whether NetBIOS session keep-alives are needed
           -  Usable UDP data field length (to control fragmentation)
```

19. MINIMAL CONFORMANCE

   To ensure multi-vendor interoperability, a minimally conforming
   implementation based on this specification must observe the following
   rules:

   a)   A node designed to work only in a broadcast area must
        conform to the B node specification.

   b)   A node designed to work only in an internet must conform to
        the P node specification.

REFERENCES

     [1]   "Protocol Standard For a NetBIOS Service on a TCP/UDP
           Transport: Detailed Specifications", RFC 1002, March 1987.

     [2]   IBM Corp., "IBM PC Network Technical Reference Manual", No.
           6322916, First Edition, September 1984.

     [3]   J. Postel (Ed.), "Transmission Control Protocol", RFC 793,
           September 1981.

     [4]   MIL-STD-1778

     [5]   J. Postel, "User Datagram Protocol", RFC 768, 28 August
           1980.

     [6]   J. Reynolds, J. Postel, "Assigned Numbers", RFC 990,
           November 1986.

     [7]   J. Postel, "Internet Protocol", RFC 791, September 1981.

     [8]   J. Mogul, "Internet Subnets", RFC 950, October 1984

     [9]   J. Mogul, "Broadcasting Internet Datagrams in the Presence
           of Subnets", RFC 922, October 1984.

     [10] J. Mogul, "Broadcasting Internet Datagrams", RFC 919,
           October 1984.

     [11] P. Mockapetris, "Domain Names - Concepts and Facilities",
           RFC 882, November 1983.

     [12] P. Mockapetris, "Domain Names - Implementation and
           Specification", RFC 883, November 1983.

     [13] P. Mockapetris, "Domain System Changes and Observations",
           RFC 973, January 1986.

     [14] C. Partridge, "Mail Routing and the Domain System", RFC 974,
           January 1986.

     [15] S. Deering, D. Cheriton, "Host Groups: A Multicast Extension
           to the Internet Protocol", RFC 966, December 1985.

     [16] S. Deering, "Host Extensions for IP Multicasting", RFC 988,
           July 1986.

APPENDIX A

   This appendix contains supporting technical discussions. It is not
   an integral part of the NetBIOS-over-TCP specification.

   INTEGRATION WITH INTERNET GROUP MULTICASTING

   The Netbios-over-TCP system described in this RFC may be easily
   integrated with the Internet Group Multicast system now being
   developed for the internet.

   In the main body of the RFC, the notion of a broadcast area was
   considered to be a single MAC-bridged "B-LAN". However, the
   protocols defined will operate over an extended broadcast area
   resulting from the creation of a permanent Internet Multicast Group.

   Each separate broadcast area would be based on a separate permanent
   Internet Multicast Group. This multicast group address would be used
   by B and M nodes as their BROADCAST_ADDRESS.

   In order to base the broadcast area on a multicast group certain
   additional procedures are required and certain constraints must be
   met.

A-1. ADDITIONAL PROTOCOL REQUIRED IN B AND M NODES

   All B or M nodes operating on an IGMP based broadcast area must have
   IGMP support in their IP layer software. These nodes must perform an
   IGMP join operation to enter the IGMP group before engaging in
   NetBIOS activity.

A-2. CONSTRAINTS

   Broadcast Areas may overlap. For this reason, end-nodes must be
   careful to examine the NetBIOS scope identifiers in all received
   broadcast packets.

   The NetBIOS broadcast protocols were designed for a network that
   exhibits a low average transit time and low rate of packet loss. An
   IGMP based broadcast area must exhibit these characteristics. In
   practice this will tend to constrain IGMP broadcast areas to a campus
   of networks interconnected by high-speed routers and inter-router
   links. It is unlikely that transcontinental broadcast areas would
   exhibit the required characteristics.

APPENDIX B

   This appendix contains supporting technical discussions. It is not
   an integral part of the NetBIOS-over-TCP specification.

IMPLEMENTATION CONSIDERATIONS

B-1. IMPLEMENTATION MODELS

   On any participating system, there must be some sort of NetBIOS
   Service to coordinate access by NetBIOS applications on that system.

   To analyze the impact of the NetBIOS-over-TCP architecture, we use
   the following three models of how a NetBIOS service might be
   implemented:

   1.   Combined Service and Application Model

        The NetBIOS service and application are both contained
        within a single process. No interprocess communication is
        assumed within the system; all communication is over the
        network. If multiple applications require concurrent access
        to the NetBIOS service, they must be folded into this
        monolithic process.

   2.   Common Kernel Element Model

        The NetBIOS Service is part of the operating system (perhaps
        as a device driver or a front-end processor). The NetBIOS
        applications are normal operating system application
        processes. The common element NetBIOS service contains all
        the information, such as the name and listen tables,
        required to co-ordinate the activities of the applications.

   3.   Non-Kernel Common Element Model

        The NetBIOS Service is implemented as an operating system
        application process. The NetBIOS applications are other
        operating system application processes. The service and the
        applications exchange data via operating system interprocess
        communication. In a multi-processor (e.g. network)
        operating system, each module may reside on a different cpu.
        The NetBIOS service process contains all the shared
        information required to coordinate the activities of the
        NetBIOS applications. The applications may still require a
        subroutine library to facilitate access to the NetBIOS
        service.

For any of the implementation models, the TCP/IP service can be
located in the operating system or split among the NetBIOS
applications and the NetBIOS service processes.

B-1.1 MODEL INDEPENDENT CONSIDERATIONS

The NetBIOS name service associates a NetBIOS name with a host. The
NetBIOS session service further binds the name to a specific TCP port
for the duration of the session.

The name service does not need to be informed of every Listen
initiation and completion. Since the names are not bound to any TCP
port in the name service, the session service may use a different tcp
port for each session established with the same local name.

The TCP port used for the data transfer phase of a NetBIOS session
can be globally well-known, locally well-known, or ephemeral. The
choice is a local implementation issue. The RETARGET mechanism
allows the binding of the NetBIOS session to a TCP connection to any
TCP port, even to another IP node.

An implementation may use the session service's globally well- known
TCP port for the data transfer phase of the session by not using the
RETARGET mechanism and, rather, accepting the session on the initial
TCP connection. This is permissible because the caller always uses
an ephemeral TCP port.

The complexity of the called end RETARGET mechanism is only required
if a particular implementation needs it. For many real system
environments, such as an in-kernel NetBIOS service implementation, it
will not be necessary to retarget incoming calls. Rather, all
NetBIOS sessions may be multiplexed through the single, well-known,
NetBIOS session service port. These implementations will not be
burdened by the complexity of the RETARGET mechanism, nor will their
callers be required to jump through the retargetting hoops.

Nevertheless, all callers must be ready to process all possible
SESSION RETARGET RESPONSEs.

B-1.2 SERVICE OPERATION FOR EACH MODEL

It is possible to construct a NetBIOS service based on this
specification for each of the above defined implementation models.

For the common kernel element model, all the NetBIOS services, name,
datagram, and session, are simple. All the information is contained
within a single entity and can therefore be accessed or modified
easily. A single port or multiple ports for the NetBIOS sessions can
be used without adding any significant complexity to the session
establishment procedure. The only penalty is the amount of overhead
incurred to get the NetBIOS messages and operation requests/responses

through the user and operating system boundary.

The combined service and application model is very similar to the
common kernel element model in terms of its requirements on the
NetBIOS service. The major difficulty is the internal coordination
of the multiple NetBIOS service and application processes existing in
a system of this type.

The NetBIOS name, datagram and session protocols assume that the
entities at the end-points have full control of the various well-
known TCP and UDP ports. If an implementation has multiple NetBIOS
service entities, as would be the case with, for example, multiple
applications each linked into a NetBIOS library, then that
implementation must impose some internal coordination.
Alternatively, use of the NetBIOS ports could be periodically
assigned to one application or another.

For the typical non-kernel common element mode implementation, three
permanent system-wide NetBIOS service processes would exist:

   - The name server
   - the datagram server
   - and session server

Each server would listen for requests from the network on a UDP or
TCP well-known port. Each application would have a small piece of
the NetBIOS service built-in, possibly a library. Each application's
NetBIOS support library would need to send a message to the
particular server to request an operation, such as add name or send a
datagram or set-up a listen.

The non-kernel common element model does not require a TCP connection
be passed between the two processes, session server and application.
The RETARGET operation for an active NetBIOS Listen could be used by
the session server to redirect the session to another TCP connection
on a port allocated and owned by the application's NetBIOS support
library. The application with either a built-in or a kernel-based
TCP/IP service could then accept the RETARGETed connection request
and process it independently of the session server.

On Unix(tm) or POSIX(tm), the NetBIOS session server could create
sub-processes for incoming connections. The open sessions would be
passed through "fork" and "exec" to the child as an open file
descriptor. This approach is very limited, however. A pre- existing
process could not receive an incoming call. And all call-ed
processes would have to be sub-processes of the session server.

B-2. CASUAL AND RESTRICTED NetBIOS APPLICATIONS

Because NetBIOS was designed to operate in the open system
environment of the typical personal computer, it does not have the

concept of privileged or unprivileged applications. In many multi-
user or multi-tasking operating systems applications are assigned
privilege capabilities. These capabilities limit the applications
ability to acquire and use system resources. For these systems it is
important to allow casual applications, those with limited system
privileges, and privileged applications, those with 'super-user'
capabilities but access to them and their required resources is
restricted, to access NetBIOS services. It is also important to
allow a systems administrator to restrict certain NetBIOS resources
to a particular NetBIOS application. For example, a file server
based on the NetBIOS services should be able to have names and TCP
ports for sessions only it can use.

A NetBIOS application needs at least two local resources to
communicate with another NetBIOS application, a NetBIOS name for
itself and, typically, a session. A NetBIOS service cannot require
that NetBIOS applications directly use privileged system resources.
For example, many systems require privilege to use TCP and UDP ports
with numbers less than 1024. This RFC requires reserved ports for
the name and session servers of a NetBIOS service implementation. It
does not require the application to have direct access these reserved
ports.

For the name service, the manager of the local name table must have
access to the NetBIOS name service's reserved UDP port. It needs to
listen for name service UDP packets to defend and define its local
names to the network. However, this manager need not be a part of a
user application in a system environment which has privilege
restrictions on reserved ports.

The internal name server can require certain privileges to add,
delete, or use a certain name, if an implementer wants the
restriction. This restriction is independent of the operation of the
NetBIOS service protocols and would not necessarily prevent the
interoperation of that implementation with another implementation.

The session server is required to own a reserved TCP port for session
establishment. However, the ultimate TCP connection used to transmit
and receive data does not have to be through that reserved port. The
RETARGET procedure the NetBIOS session to be shifted to another TCP
connection, possibly through a different port at the called end.
This port can be an unprivileged resource, with a value greater than
1023. This facilitates casual applications.

Alternately, the RETARGET mechanism allows the TCP port used for data
transmission and reception to be a reserved port. Consequently, an
application wishing to have access to its ports maintained by the
system administrator can use these restricted TCP ports. This
facilitates privileged applications.

A particular implementation may wish to require further special

privileges for session establishment, these could be associated with
internal information. It does not have to be based solely on TCP
port allocation. For example, a given NetBIOS name may only be used
for sessions by applications with a certain system privilege level.

The decision to use reserved or unreserved ports or add any
additional name registration and usage authorization is a purely
local implementation decision. It is not required by the NetBIOS
protocols specified in the RFC.

B-3. TCP VERSUS SESSION KEEP-ALIVES

The KEEP-ALIVE is a protocol element used to validate the existence
of a connection. A packet is sent to the remote connection partner
to solicit a response which shows the connection is still
functioning. TCP KEEP-ALIVES are used at the TCP level on TCP
connections while session KEEP-ALIVES are used on NetBIOS sessions.
These protocol operations are always transparent to the connection
user. The user will only find out about a KEEP-ALIVE operation if it
fails, therefore, if the connection is lost.

The NetBIOS specification[2] requires the NetBIOS service to inform
the session user if a session is lost when it is in a passive or
active state. Therefore,if the session user is only waiting for a
receive operation and the session is dropped the NetBIOS service must
inform the session user. It cannot wait for a session send operation
before it informs the user of the loss of the connection.

This requirement stems from the management of scarce or volatile
resources by a NetBIOS application. If a particular user terminates
a session with a server application by destroying the client
application or the NetBIOS service without a NetBIOS Hang Up, the
server application will want to clean-up or free allocated resources.
This server application if it only receives and then sends a response
requires the notification of the session abort in the passive state.

The standard definition of a TCP service cannot detect loss of a
connection when it is in a passive state, waiting for a packet to
arrive. Some TCP implementations have added a KEEP-ALIVE operation
which is interoperable with implementations without this feature.
These implementations send a packet with an invalid sequence number
to the connection partner. The partner, by specification, must
respond with a packet showing the correct sequence number of the
connection. If no response is received from the remote partner
within a certain time interval then the TCP service assumes the
connection is lost.

Since many TCP implementations do not have this KEEP-ALIVE function
an optional NetBIOS KEEP-ALIVE operation has been added to the
NetBIOS session protocols. The NetBIOS KEEP-ALIVE uses the
properties of TCP to solicit a response from the remote connection

   partner. A NetBIOS session message called KEEP-ALIVE is sent to the
   remote partner. Since this results in TCP sending an IP packet to
   the remote partner, the TCP connection is active. TCP will discover
   if the TCP connection is lost if the remote TCP partner does not
   acknowledge the IP packet. Therefore, the NetBIOS session service
   does not send a response to a session KEEP ALIVE message. It just
   throws it away. The NetBIOS session service that transmits the KEEP
   ALIVE is informed only of the failure of the TCP connection. It does
   not wait for a specific response message.

   A particular NetBIOS implementation should use KEEP-ALIVES if it is
   concerned with maintaining compatibility with the NetBIOS interface
   specification[2]. Compatibility is especially important if the
   implementation wishes to support existing NetBIOS applications, which
   typically require the session loss detection on their servers, or
   future applications which were developed for implementations with
   session loss detection.

B-4. RETARGET ALGORITHMS

   This section contains 2 suggestions for RETARGET algorithms. They
   are called the "straight" and "stack" methods. The algorithm in the
   body of the RFC uses the straight method. Implementation of either
   algorithm must take into account the Session establishment maximum
   retry count. The retry count is the maximum number of TCP connect
   operations allowed before a failure is reported.

   The straight method forces the session establishment procedure to
   begin a retry after a retargetting failure with the initial node
   returned from the name discovery procedure. A retargetting failure
   is when a TCP connection attempt fails because of a time- out or a
   NEGATIVE SESSION RESPONSE is received with an error code specifying
   NOT LISTENING ON CALLED NAME. If any other failure occurs the
   session establishment procedure should retry from the call to the
   name discovery procedure.

   A minimum of 2 retries, either from a retargetting or a name
   discovery failure. This will give the session service a chance to
   re-establish a NetBIOS Listen or, more importantly, allow the NetBIOS
   scope, local name service or the NBNS, to re-learn the correct IP
   address of a NetBIOS name.

   The stack method operates similarly to the straight method. However,
   instead of retrying at the initial node returned by the name
   discovery procedure, it restarts with the IP address of the last node
   which sent a SESSION RETARGET RESPONSE prior to the retargetting
   failure. To limit the stack method, any one host can only be tried a
   maximum of 2 times.

B-5. NBDD SERVICE

   If the NBDD does not forward datagrams then don't provide Group and
   Broadcast NetBIOS datagram services to the NetBIOS user. Therefore,
   ignore the implementation of the query request and, when get a
   negative response, acquiring the membership list of IP addresses and
   sending the datagram as a unicast to each member.

B-6. APPLICATION CONSIDERATIONS

B-6.1 USE OF NetBIOS DATAGRAMS

   Certain existing NetBIOS applications use NetBIOS datagrams as a
   foundation for their own connection-oriented protocols. This can
   cause excessive NetBIOS name query activity and place a substantial
   burden on the network, server nodes, and other end- nodes. It is
   recommended that this practice be avoided in new applications.