Network Working Group
Request for Comments: 1002 March, 1987

PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: DETAILED SPECIFICATIONS

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 gives the detailed specifications of the NetBIOS-over-TCP packets, protocols, and defined constants and variables. A more general overview is found in a companion RFC, "Protocol Standard For a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods".

TABLE OF CONTENTS

PROTOCOL STANDARD FOR A NetBIOS SERVICE
ON A TCP/UDP TRANSPORT:
DETAILED SPECIFICATIONS

1. STATUS OF THIS MEMO

This RFC specifies a proposed standard for the DARPA 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 memorandum is unlimited.

2. ACKNOWLEDGEMENTS

This RFC has been developed under the auspices of the Internet Activities Board.

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 contains the detailed packet formats and protocol specifications for NetBIOS-over-TCP. This RFC is a companion to RFC 1001, "Protocol Standard For a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods" [1].

4. PACKET DESCRIPTIONS

Bit and byte ordering are defined by the most recent version of "Assigned Numbers" [2].

4.1. NAME FORMAT

The NetBIOS name representation in all NetBIOS packets (for NAME, SESSION, and DATAGRAM services) is defined in the Domain Name Service RFC 883[3] as "compressed" name messages. This format is called "second-level encoding" in the section entitled "Representation of NetBIOS Names" in the Concepts and Methods document.

For ease of description, the first two paragraphs from page 31, the section titled "Domain name representation and compression", of RFC 883 are replicated here:

> Domain names messages are expressed in terms of a sequence of labels. Each label is represented as a one octet length field followed by that number of octets. Since every domain name ends with the null label of the root, a compressed domain name is terminated by a length byte of zero. The high order two bits of the length field must be zero, and the remaining six bits of the length field limit the label to 63 octets or less.
>
> To simplify implementations, the total length of label octets and label length octets that make up a domain name is restricted to 255 octets or less.

The following is the uncompressed representation of the NetBIOS name "FRED ", which is the 4 ASCII characters, F, R, E, D, followed by 12 space characters (0x20). This name has the SCOPE_ID: "NETBIOS.COM"

```
        EGFCEFEECACACACACACACACACACACACA.NETBIOS.COM
```

This uncompressed representation of names is called "first-level encoding" in the section entitled "Representation of NetBIOS Names" in the Concepts and Methods document.

The following is a pictographic representation of the compressed representation of the previous uncompressed Domain Name representation.

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x20     |    E (0x45)   |    G (0x47)   |    F (0x46)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    C (0x43)   |    E (0x45)   |    F (0x46)   |    E (0x45)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    E (0x45)   |    C (0x43)   |    A (0x41)   |    C (0x43)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    A (0x41)   |    C (0x43)   |    A (0x41)   |    C (0x43)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    A (0x41)   |    C (0x43)   |    A (0x41)   |    C (0x43)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    A (0x41)   |    C (0x43)   |    A (0x41)   |    C (0x43)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    A (0x41)   |    C (0x43)   |    A (0x41)   |    C (0x43)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    A (0x41)   |    C (0x43)   |    A (0x41)   |    C (0x43)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    A (0X41)   |      0x07     |    N (0x4E)   |    E (0x45)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    T (0x54)   |    B (0x42)   |    I (0x49)   |    O (0x4F)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    S (0x53)   |      0x03     |    C (0x43)   |    O (0x4F)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    M (0x4D)   |      0x00     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Each section of a domain name is called a label [7 (page 31)]. A label can be a maximum of 63 bytes. The first byte of a label in compressed representation is the number of bytes in the label. For the above example, the first 0x20 is the number of bytes in the left-most label, EGFCEFEECACACACACACACACACACACACA, of the domain name. The bytes following the label length count are the characters of the label. The following labels are in sequence after the first label, which is the encoded NetBIOS name, until a zero (0x00) length count. The zero length count represents the root label, which is always null.

A label length count is actually a 6-bit field in the label length field. The most significant 2 bits of the field, bits 7 and 6, are flags allowing an escape from the above compressed representation. If bits 7 and 6 are both set (11), the following 14 bits are an offset pointer into the full message to the actual label string from another domain name that belongs in this name. This label pointer allows for a further compression of a domain name in a packet.

NetBIOS implementations can only use label string pointers in Name Service packets. They cannot be used in Session or Datagram Service packets.

The other two possible values for bits 7 and 6 (01 and 10) of a label length field are reserved for future use by RFC 883[2 (page 32)].

Note that the first octet of a compressed name must contain one of the following bit patterns. (An "x" indicates a bit whose value may be either 0 or 1.):

```
        00100000 -  Netbios name, length must be 32 (decimal)
        11xxxxxx -  Label string pointer
        10xxxxxx -  Reserved
        01xxxxxx -  Reserved
```

4.2. NAME SERVICE PACKETS

4.2.1. GENERAL FORMAT OF NAME SERVICE PACKETS

The NetBIOS Name Service packets follow the packet structure defined in the Domain Name Service (DNS) RFC 883 [7 (pg 26-31)]. The structures are compatible with the existing DNS packet formats, however, additional types and codes have been added to work with NetBIOS.

If Name Service packets are sent over a TCP connection they are preceded by a 16 bit unsigned integer representing the length of the Name Service packet.

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   + ------                                                ------- +
   |                            HEADER                             |
   + ------                                                ------- +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                       QUESTION ENTRIES                        /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                    ANSWER RESOURCE RECORDS                    /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                  AUTHORITY RESOURCE RECORDS                   /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                  ADDITIONAL RESOURCE RECORDS                  /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.1.1. HEADER

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           | OPCODE  |   NM_FLAGS  | RCODE |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          QDCOUNT              |           ANCOUNT             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          NSCOUNT              |           ARCOUNT             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field   Description

NAME_TRN_ID   Transaction ID for Name Service Transaction. Requestor places a unique value for each active transaction. Responder puts NAME_TRN_ID value from request packet in response packet.

OPCODE   Packet type code, see table below.

NM_FLAGS   Flags for operation, see table below.

RCODE   Result codes of request. Table of RCODE values for each response packet below.

QDCOUNT   Unsigned 16 bit integer specifying the number of entries in the question section of a Name

Service packet. Always zero (0) for responses. Must be non-zero for all NetBIOS Name requests.

ANCOUNT   Unsigned 16 bit integer specifying the number of resource records in the answer section of a Name Service packet.

NSCOUNT   Unsigned 16 bit integer specifying the number of resource records in the authority section of a Name Service packet.

ARCOUNT   Unsigned 16 bit integer specifying the number of resource records in the additional records section of a Name Service packet.

The OPCODE field is defined as:

```
     0   1   2   3   4
   +---+---+---+---+---+
   | R |    OPCODE     |
   +---+---+---+---+---+
   Symbol     Bit(s)   Description

   OPCODE        1-4   Operation specifier:
                         0 = query
                         5 = registration
                         6 = release
                         7 = WACK
                         8 = refresh

   R               0   RESPONSE flag:
                         if bit == 0 then request packet
                         if bit == 1 then response packet.
```

The NM_FLAGS field is defined as:

```
  0   1   2   3   4   5   6
+---+---+---+---+---+---+---+
|AA |TC |RD |RA | 0 | 0 | B |
+---+---+---+---+---+---+---+
```

```
Symbol     Bit(s)   Description

B               6   Broadcast Flag.
                      = 1: packet was broadcast or multicast
                      = 0: unicast

RA              3   Recursion Available Flag.

                    Only valid in responses from a NetBIOS Name
                    Server -- must be zero in all other
                    responses.

                    If one (1) then the NBNS supports recursive
                    query, registration, and release.

                    If zero (0) then the end-node must iterate
                    for query and challenge for registration.

RD              2   Recursion Desired Flag.

                    May only be set on a request to a NetBIOS
                    Name Server.

                    The NBNS will copy its state into the
                    response packet.

                    If one (1) the NBNS will iterate on the
                    query, registration, or release.

TC              1   Truncation Flag.
```

Set if this message was truncated because the datagram carrying it would be greater than 576 bytes in length. Use TCP to get the information from the NetBIOS Name Server.

AA 0 Authoritative Answer flag.

Must be zero (0) if R flag of OPCODE is zero (0).

If R flag is one (1) then if AA is one (1) then the node responding is an authority for the domain name.

End nodes responding to queries always set this bit in responses.

4.2.1.2. QUESTION SECTION

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                         QUESTION_NAME                         /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         QUESTION_TYPE         |        QUESTION_CLASS         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field Description

QUESTION_NAME The compressed name representation of the NetBIOS name for the request.

QUESTION_TYPE The type of request. The values for this field are specified for each request.

QUESTION_CLASS The class of the request. The values for this field are specified for each request.

QUESTION_TYPE is defined as:

Symbol	Value	Description:
NB	0x0020	NetBIOS general Name Service Resource Record
NBSTAT	0x0021	NetBIOS NODE STATUS Resource Record (See NODE STATUS REQUEST)

QUESTION_CLASS is defined as:

Symbol	Value	Description:
IN	0x0001	Internet class

4.2.1.3. RESOURCE RECORD

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           RR_TYPE              |          RR_CLASS            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           RDLENGTH             |                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                              |
   /                                                               /
   /                             RDATA                             /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Field	Description
RR_NAME	The compressed name representation of the NetBIOS name corresponding to this resource record.
RR_TYPE	Resource record type code
RR_CLASS	Resource record class code
TTL	The Time To Live of a the resource record's name.
RDLENGTH	Unsigned 16 bit integer that specifies the number of bytes in the RDATA field.
RDATA	RR_CLASS and RR_TYPE dependent field. Contains the resource information for the NetBIOS name.

RESOURCE RECORD RR_TYPE field definitions:

Symbol	Value	Description:
A	0x0001	IP address Resource Record (See REDIRECT NAME QUERY RESPONSE)
NS	0x0002	Name Server Resource Record (See REDIRECT

		NAME QUERY RESPONSE)
NULL	0x000A	NULL Resource Record (See WAIT FOR ACKNOWLEDGEMENT RESPONSE)
NB	0x0020	NetBIOS general Name Service Resource Record (See NB_FLAGS and NB_ADDRESS, below)
NBSTAT	0x0021	NetBIOS NODE STATUS Resource Record (See NODE STATUS RESPONSE)

RESOURCE RECORD RR_CLASS field definitions:

Symbol	Value	Description:
IN	0x0001	Internet class

NB_FLAGS field of the RESOURCE RECORD RDATA field for RR_TYPE of "NB":

```
                                         1   1   1   1   1   1
  0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| G |  ONT  |                RESERVED                           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Symbol	Bit(s)	Description:
RESERVED	3-15	Reserved for future use. Must be zero (0).
ONT	1,2	Owner Node Type: 00 = B node 01 = P node 10 = M node 11 = Reserved for future use For registration requests this is the claimant's type. For responses this is the actual owner's type.
G	0	Group Name Flag. If one (1) then the RR_NAME is a GROUP NetBIOS name. If zero (0) then the RR_NAME is a UNIQUE NetBIOS name.

The NB_ADDRESS field of the RESOURCE RECORD RDATA field for RR_TYPE of "NB" is the IP address of the name's owner.

4.2.2. NAME REGISTRATION REQUEST

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |0|  0x5  |0|0|1|0|0 0|B|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0001               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                         QUESTION_NAME                         /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |        IN (0x0001)            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Since the RR_NAME is the same name as the QUESTION_NAME, the RR_NAME representation must use pointers to the QUESTION_NAME name's labels to guarantee the length of the datagram is less than the maximum 576 bytes. See section above on name formats and also page 31 and 32 of RFC 883, Domain Names - Implementation and Specification, for a complete description of compressed name label pointers.

4.2.3. NAME OVERWRITE REQUEST & DEMAND

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |0|  0x5  |0|0|0|0|0 0|B|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0001               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                         QUESTION_NAME                         /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |        IN (0x0001)            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.4. NAME REFRESH REQUEST

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |0|  0x9  |0|0|0|0|0 0|B|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0001               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                         QUESTION_NAME                         /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |        IN (0x0001)            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.5. POSITIVE NAME REGISTRATION RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      NAME_TRN_ID              |1|  0x5  |1|0|1|1|0 0|0|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.6. NEGATIVE NAME REGISTRATION RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      NAME_TRN_ID              |1|  0x5  |1|0|1|1|0 0|0| RCODE |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

RCODE field values:

Symbol	Value	Description:
FMT_ERR	0x1	Format Error. Request was invalidly formatted.
SRV_ERR	0x2	Server failure. Problem with NBNS, cannot process name.
IMP_ERR	0x4	Unsupported request error. Allowable only for challenging NBNS when gets an Update type registration request.
RFS_ERR	0x5	Refused error. For policy reasons server will not register this name from this host.
ACT_ERR	0x6	Active error. Name is owned by another node.
CFT_ERR	0x7	Name in conflict error. A UNIQUE name is owned by more than one node.

4.2.7. END-NODE CHALLENGE REGISTRATION RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x5  |1|0|1|0|0 0|0|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.8. NAME CONFLICT DEMAND

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x5  |1|0|1|1|0 0|0|  0x7  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          0x00000000                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |0|ONT|0|     0x000             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          0x00000000                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This packet is identical to a NEGATIVE NAME REGISTRATION RESPONSE with RCODE = CFT_ERR.

4.2.9. NAME RELEASE REQUEST & DEMAND

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |0|  0x6  |0|0|0|0|0 0|B|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0001               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                         QUESTION_NAME                         /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |        IN (0x0001)            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          0x00000000                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Since the RR_NAME is the same name as the QUESTION_NAME, the RR_NAME representation must use label string pointers to the QUESTION_NAME labels to guarantee the length of the datagram is less than the maximum 576 bytes. This is the same condition as with the NAME REGISTRATION REQUEST.

4.2.10. POSITIVE NAME RELEASE RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x6  |1|0|0|0|0 0|0|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.11. NEGATIVE NAME RELEASE RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x6  |1|0|0|0|0 0|0| RCODE |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0006              |          NB_FLAGS             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          NB_ADDRESS                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

RCODE field values:

Symbol	Value	Description:
FMT_ERR	0x1	Format Error. Request was invalidly formatted.
SRV_ERR	0x2	Server failure. Problem with NBNS, cannot process name.
RFS_ERR	0x5	Refused error. For policy reasons server will not release this name from this host.
ACT_ERR	0x6	Active error. Name is owned by another node. Only that node may release it. A NetBIOS Name Server can optionally allow a node to release a name it does not own. This would facilitate detection of inactive names for nodes that went down silently.

4.2.12. NAME QUERY REQUEST

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |0|  0x0  |0|0|1|0|0 0|B|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0001               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                         QUESTION_NAME                         /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |        IN (0x0001)            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.13. POSITIVE NAME QUERY RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x0  |1|T|1|?|0 0|0|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NB (0x0020)         |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           RDLENGTH            |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   |                                                               |
   /                       ADDR_ENTRY ARRAY                        /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The ADDR_ENTRY ARRAY a sequence of zero or more ADDR_ENTRY records. Each ADDR_ENTRY record represents an owner of a name. For group names there may be multiple entries. However, the list may be incomplete due to packet size limitations. Bit 22, "T", will be set to indicate truncated data.

Each ADDR_ENTRY has the following format:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          NB_FLAGS             |          NB_ADDRESS           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   NB_ADDRESS (continued)      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.14. NEGATIVE NAME QUERY RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x0  |1|0|1|?|0 0|0| RCODE |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NULL (0x000A)       |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          0x00000000                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   RCODE field values:

   Symbol      Value   Description

   FMT_ERR       0x1   Format Error.  Request was invalidly
                       formatted.
   SRV_ERR       0x2   Server failure.  Problem with NBNS, cannot
                       process name.
   NAM_ERR       0x3   Name Error.  The name requested does not
                       exist.
   IMP_ERR       0x4   Unsupported request error.  Allowable only
                       for challenging NBNS when gets an Update type
                       registration request.
   RFS_ERR       0x5   Refused error.  For policy reasons server
                       will not register this name from this host.
```

4.2.15. REDIRECT NAME QUERY RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x0  |0|0|1|0|0 0|0|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0001               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           NS (0x0002)         |           IN (0x0001)         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           RDLENGTH            |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   /                           NSD_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           A (0x0001)          |           IN (0x0001)         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             0x0004            |          NSD_IP_ADDR          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     NSD_IP_ADDR, continued    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An end node responding to a NAME QUERY REQUEST always responds with the AA and RA bits set for both the NEGATIVE and POSITIVE NAME QUERY RESPONSE packets. An end node never sends a REDIRECT NAME QUERY RESPONSE packet.

When the requestor receives the REDIRECT NAME QUERY RESPONSE it must reiterate the NAME QUERY REQUEST to the NBNS specified by the NSD_IP_ADDR field of the A type RESOURCE RECORD in the ADDITIONAL section of the response packet. This is an optional packet for the NBNS.

The NSD_NAME and the RR_NAME in the ADDITIONAL section of the response packet are the same name. Space can be optimized if label string pointers are used in the RR_NAME which point to the labels in the NSD_NAME.

The RR_NAME in the AUTHORITY section is the name of the domain the NBNS called by NSD_NAME has authority over.

4.2.16. WAIT FOR ACKNOWLEDGEMENT (WACK) RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x7  |1|0|0|0|0 0|0|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          NULL (0x0020)        |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TTL                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           0x0002              | OPCODE  |   NM_FLAGS  |  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The NAME_TRN_ID of the WACK RESPONSE packet is the same NAME_TRN_ID of the request that the NBNS is telling the requestor to wait longer to complete. The RR_NAME is the name from the request, if any. If no name is available from the request then it is a null name, single byte of zero.

The TTL field of the ResourceRecord is the new time to wait, in seconds, for the request to complete. The RDATA field contains the OPCODE and NM_FLAGS of the request.

A TTL value of 0 means that the NBNS can not estimate the time it may take to complete a response.

4.2.17. NODE STATUS REQUEST

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |0|  0x0  |0|0|0|0|0 0|B|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0001               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                         QUESTION_NAME                         /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NBSTAT (0x0021)       |        IN (0x0001)            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2.18. NODE STATUS RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_TRN_ID           |1|  0x0  |1|0|0|0|0 0|0|  0x0  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0001              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          0x0000               |           0x0000              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                            RR_NAME                            /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        NBSTAT (0x0021)        |         IN (0x0001)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          0x00000000                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          RDLENGTH             |   NUM_NAMES   |               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               +
   |                                                               |
   +                                                               +
   /                         NODE_NAME ARRAY                       /
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   /                           STATISTICS                          /
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The NODE_NAME ARRAY is an array of zero or more NUM_NAMES entries of NODE_NAME records. Each NODE_NAME entry represents an active name in the same NetBIOS scope as the requesting name in the local name table of the responder. RR_NAME is the requesting name.

NODE_NAME Entry:

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +---                                                         ---+
   |                                                               |
   +---                    NETBIOS FORMAT NAME                  ---+
   |                                                               |
   +---                                                         ---+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         NAME_FLAGS            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The NAME_FLAGS field:

```
                                             1   1   1   1   1   1
     0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   | G |  ONT  |DRG|CNF|ACT|PRM|          RESERVED                 |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

The NAME_FLAGS field is defined as:

```
   Symbol     Bit(s)   Description:

   RESERVED     7-15   Reserved for future use.  Must be zero (0).
   PRM             6   Permanent Name Flag.  If one (1) then entry
                       is for the permanent node name.  Flag is zero
                       (0) for all other names.
   ACT             5   Active Name Flag.  All entries have this flag
                       set to one (1).
   CNF             4   Conflict Flag.  If one (1) then name on this
                       node is in conflict.
   DRG             3   Deregister Flag.  If one (1) then this name
                       is in the process of being deleted.
   ONT           1,2   Owner Node Type:
                          00 = B node
                          01 = P node
                          10 = M node
                          11 = Reserved for future use
   G               0   Group Name Flag.
                       If one (1) then the name is a GROUP NetBIOS
                       name.
                       If zero (0) then it is a UNIQUE NetBIOS name.
```

STATISTICS Field of the NODE STATUS RESPONSE:

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               UNIT_ID (Unique unit ID)                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       UNIT_ID,continued       |    JUMPERS    |  TEST_RESULT  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       VERSION_NUMBER          |      PERIOD_OF_STATISTICS     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       NUMBER_OF_CRCs          |     NUMBER_ALIGNMENT_ERRORS   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       NUMBER_OF_COLLISIONS    |        NUMBER_SEND_ABORTS     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       NUMBER_GOOD_SENDS                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      NUMBER_GOOD_RECEIVES                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       NUMBER_RETRANSMITS      | NUMBER_NO_RESOURCE_CONDITIONS |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  NUMBER_FREE_COMMAND_BLOCKS   |  TOTAL_NUMBER_COMMAND_BLOCKS  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |MAX_TOTAL_NUMBER_COMMAND_BLOCKS|    NUMBER_PENDING_SESSIONS    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  MAX_NUMBER_PENDING_SESSIONS  |  MAX_TOTAL_SESSIONS_POSSIBLE  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   SESSION_DATA_PACKET_SIZE    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.3. SESSION SERVICE PACKETS

4.3.1. GENERAL FORMAT OF SESSION PACKETS

All session service messages are sent over a TCP connection.

All session packets are of the following general structure:

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      TYPE     |     FLAGS     |            LENGTH             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /               TRAILER (Packet Type Dependent)                 /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The TYPE, FLAGS, and LENGTH fields are present in every session packet.

The LENGTH field is the number of bytes following the LENGTH field. In other words, LENGTH is the combined size of the TRAILER field(s). For example, the POSITIVE SESSION RESPONSE packet always has a LENGTH field value of zero (0000) while the RETARGET SESSION RESPONSE always has a LENGTH field value of six (0006).

One of the bits of the FLAGS field acts as an additional, high-order bit for the LENGTH field. Thus the cumulative size of the trailer field(s) may range from 0 to 128K bytes.

Session Packet Types (in hexidecimal):

00 -	SESSION MESSAGE
81 -	SESSION REQUEST
82 -	POSITIVE SESSION RESPONSE
83 -	NEGATIVE SESSION RESPONSE
84 -	RETARGET SESSION RESPONSE
85 -	SESSION KEEP ALIVE

Bit definitions of the FLAGS field:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | E |
+---+---+---+---+---+---+---+---+
```

Symbol	Bit(s)	Description
E	7	Length extension, used as an additional, high-order bit on the LENGTH field.
RESERVED	0-6	Reserved, must be zero (0)

4.3.2. SESSION REQUEST PACKET

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      TYPE     |     FLAGS     |            LENGTH             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                          CALLED NAME                          /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                          CALLING NAME                         /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.3.3. POSITIVE SESSION RESPONSE PACKET

```
                       1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |      TYPE     |     FLAGS     |            LENGTH             |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.3.4. NEGATIVE SESSION RESPONSE PACKET

```
                       1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |      TYPE     |     FLAGS     |            LENGTH             |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |   ERROR_CODE  |
  +-+-+-+-+-+-+-+-+
```

NEGATIVE SESSION RESPONSE packet error code values (in hexidecimal):

```
        80 -  Not listening on called name
        81 -  Not listening for calling name
        82 -  Called name not present
        83 -  Called name present, but insufficient resources
        8F -  Unspecified error
```

4.3.5. SESSION RETARGET RESPONSE PACKET

```
                       1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |      TYPE     |     FLAGS     |            LENGTH             |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                      RETARGET_IP_ADDRESS                      |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |           PORT                |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.3.6. SESSION MESSAGE PACKET

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      TYPE     |     FLAGS     |            LENGTH             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                                                               /
   /                           USER_DATA                           /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.3.7. SESSION KEEP ALIVE PACKET

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      TYPE     |     FLAGS     |            LENGTH             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.4. DATAGRAM SERVICE PACKETS

4.4.1. NetBIOS DATAGRAM HEADER

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   MSG_TYPE    |     FLAGS     |           DGM_ID              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           SOURCE_IP                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          SOURCE_PORT          |          DGM_LENGTH           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         PACKET_OFFSET         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MSG_TYPE values (in hexidecimal):

```
           10 -  DIRECT_UNIQUE DATAGRAM
           11 -  DIRECT_GROUP DATAGRAM
           12 -  BROADCAST DATAGRAM
           13 -  DATAGRAM ERROR
           14 -  DATAGRAM QUERY REQUEST
           15 -  DATAGRAM POSITIVE QUERY RESPONSE
           16 -  DATAGRAM NEGATIVE QUERY RESPONSE
```

Bit definitions of the FLAGS field:

```
     0   1   2   3   4   5   6   7
   +---+---+---+---+---+---+---+---+
   | 0 | 0 | 0 | 0 |  SNT  | F | M |
   +---+---+---+---+---+---+---+---+

   Symbol     Bit(s)   Description

   M               7   MORE flag, If set then more NetBIOS datagram
                       fragments follow.

   F               6   FIRST packet flag,  If set then this is first
                       (and possibly only) fragment of NetBIOS
                       datagram

   SNT           4,5   Source End-Node type:
                          00 = B node
                          01 = P node
                          10 = M node
                          11 = NBDD
   RESERVED      0-3   Reserved, must be zero (0)
```

4.4.2. DIRECT_UNIQUE, DIRECT_GROUP, & BROADCAST DATAGRAM

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   MSG_TYPE    |     FLAGS     |           DGM_ID              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           SOURCE_IP                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          SOURCE_PORT          |          DGM_LENGTH           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         PACKET_OFFSET         |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   |                                                               |
   /                          SOURCE_NAME                          /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                       DESTINATION_NAME                        /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   /                           USER_DATA                           /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.4.3. DATAGRAM ERROR PACKET

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   MSG_TYPE    |     FLAGS     |           DGM_ID              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           SOURCE_IP                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          SOURCE_PORT          |  ERROR_CODE   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ERROR_CODE values (in hexidecimal):

```
           82 -  DESTINATION NAME NOT PRESENT
           83 -  INVALID SOURCE NAME FORMAT
           84 -  INVALID DESTINATION NAME FORMAT
```

4.4.4. DATAGRAM QUERY REQUEST

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   MSG_TYPE    |     FLAGS     |           DGM_ID              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           SOURCE_IP                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          SOURCE_PORT          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   /                       DESTINATION_NAME                        /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.4.5. DATAGRAM POSITIVE AND NEGATIVE QUERY RESPONSE

```
                        1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   MSG_TYPE    |     FLAGS     |           DGM_ID              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           SOURCE_IP                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          SOURCE_PORT          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   /                       DESTINATION_NAME                        /
   /                                                               /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

5. PROTOCOL DESCRIPTIONS

5.1. NAME SERVICE PROTOCOLS

A REQUEST packet is always sent to the well known UDP port - NAME_SERVICE_UDP_PORT. The destination address is normally either the IP broadcast address or the address of the NBNS - the address of the NBNS server it set up at initialization time. In rare cases, a request packet will be sent to an end node, e.g. a NAME QUERY REQUEST sent to "challenge" a node.

A RESPONSE packet is always sent to the source UDP port and source IP address of the request packet.

A DEMAND packet must always be sent to the well known UDP port - NAME_SERVICE_UDP_PORT. There is no restriction on the target IP address.

Terms used in this section:

tid - Transaction ID. This is a value composed from the requestor's IP address and a unique 16 bit value generated by the originator of the transaction.

5.1.1. B-NODE ACTIVITY

5.1.1.1. B-NODE ADD NAME

```
PROCEDURE add_name(newname)

/*
 * Host initiated processing for a B node
 */
BEGIN

     REPEAT

          /* build name service packet */

          ONT = B_NODE; /* broadcast node */
          G = UNIQUE;   /* unique name */
          TTL = 0;

          broadcast NAME REGISTRATION REQUEST packet;

          /*
           * remote node(s) will send response packet
           * if applicable
           */
```

```
             pause(BCAST_REQ_RETRY_TIMEOUT);

        UNTIL response packet is received or
             retransmit count has been exceeded

        IF no response packet was received THEN
        BEGIN /* no response */
             /*
              * build packet
              */

             ONT = B_NODE; /* broadcast node */
             G = UNIQUE;   /* unique name */
             TTL = 0;

             /*
              * Let other nodes known you have the name
              */

             broadcast NAME UPDATE REQUEST packet;
             /* name can be added to local name table */
             return success;
        END /* no response */
        ELSE
        BEGIN /* got response */

             /*
              * Match return transaction id
              * against tid sent in request
              */

            IF NOT response tid = request tid THEN
            BEGIN
             ignore response packet;
            END
            ELSE
            CASE packet type OF

            NEGATIVE NAME REGISTRATION RESPONSE:

                 return failure; /* name cannot be added */

            POSITIVE NAME REGISTRATION RESPONSE:
            END-NODE CHALLENGE NAME REGISTRATION RESPONSE:

                 /*
                  * B nodes should normally not get this
                  * response.
                  */

                  ignore packet;
            END /* case */;
        END /* got response */
   END /* procedure */
```

5.1.1.2. B-NODE ADD_GROUP NAME

```
   PROCEDURE add_group_name(newname)

   /*
    * Host initiated processing for a B node
    */

   BEGIN
        /*
         * same as for a unique name with the
         * exception that the group bit (G) must
         * be set in the request packets.
         */

        ...
        G = GROUP;
        ...
        ...

        /*
         * broadcast request ...
         */


   END
```

5.1.1.3. B-NODE FIND_NAME

```
   PROCEDURE find_name(name)

   /*
    * Host initiated processing for a B node
    */

   BEGIN

        REPEAT
             /*
              * build packet
              */
             ONT = B;
             TTL = 0;
             G = DONT CARE;

             broadcast NAME QUERY REQUEST packet;
             /*
              * a node might send response packet
              */

             pause(BCAST_REQ_RETRY_TIMEOUT);
        UNTIL response packet received OR
             max transmit threshold exceeded

        IF no response packet received THEN
             return failure;
        ELSE
        IF NOT response tid = request tid THEN
             ignore packet;
        ELSE
        CASE packet type OF
        POSITIVE NAME QUERY RESPONSE:
             /*
              * Start a timer to detect conflict.
              *
              * Be prepared to detect conflict if
              * any more response packets are received.
              *
              */

             save response as authoritative response;
             start_timer(CONFLICT_TIMER);
             return success;

        NEGATIVE NAME QUERY RESPONSE:
        REDIRECT NAME QUERY RESPONSE:

             /*
              * B Node should normally not get either
              * response.
              */

              ignore response packet;

        END /* case */
   END /* procedure */
```

5.1.1.4. B NODE NAME RELEASE

```
   PROCEDURE delete_name (name)
   BEGIN

        REPEAT

             /*
              * build packet
              */
             ...

             /*
              * send request
              */

             broadcast NAME RELEASE REQUEST packet;

             /*
              * no response packet expected
              */

             pause(BCAST_REQ_RETRY_TIMEOUT);

        UNTIL retransmit count has been exceeded
   END /* procedure */
```

5.1.1.5. B-NODE INCOMING PACKET PROCESSING

Following processing is done when broadcast or unicast packets are received at the NAME_SERVICE_UDP_PORT.

```
   PROCEDURE process_incoming_packet(packet)

   /*
    * Processing initiated by incoming packets for a B node
    */

   BEGIN
        /*
         * Note: response packets are always sent
         * to:
         * source IP address of request packet
         * source UDP port of request packet
         */

        CASE packet type OF

        NAME REGISTRATION REQUEST (UNIQUE):
             IF name exists in local name table THEN
                  send NEGATIVE NAME REGISTRATION RESPONSE ;
        NAME REGISTRATION REQUEST (GROUP):
             IF name exists in local name table THEN
             BEGIN
                  IF local entry is a unique name THEN
                      send NEGATIVE NAME REGISTRATION RESPONSE ;
             END
        NAME QUERY REQUEST:
             IF name exists in local name table THEN
             BEGIN
                  build response packet;
                  send POSITIVE NAME QUERY RESPONSE;
        POSITIVE NAME QUERY RESPONSE:
             IF name conflict timer is not active THEN
             BEGIN
                  /*
                   * timer has expired already...  ignore this
                   * packet
                   */

                  return;
             END
             ELSE /* timer is active */
             IF a response for this name has previously been
                  received THEN
             BEGIN /* existing entry */

                  /*
                   * we sent out a request packet, and
                   * have already received (at least)
                   * one response
                   *
                   * Check if conflict exists.
                   * If so, send out a conflict packet.
                   *
                   * Note: detecting conflict does NOT
                   * affect any existing sessions.
                   *
                   */

                  /*
                   * Check for name conflict.
                   * See "Name Conflict" in Concepts and Methods
                   */
                  check saved authoritative response against
                       information in this response packet;
                  IF conflict detected THEN
                  BEGIN
                       unicast NAME CONFLICT DEMAND packet;
                       IF entry exists in cache THEN
                       BEGIN
                            remove entry from cache;
                       END
                  END
             END /* existing entry */
             ELSE
             BEGIN
                  /*
                   * Note: If this was the first response
                   * to a name query, it would have been
                   * handled in the
                   * find_name() procedure.
                   */

                  ignore packet;
             END
        NAME CONFLICT DEMAND:
             IF name exists in local name table THEN
             BEGIN
                  mark name as conflict detected;

                  /*
                   * a name in the state "conflict detected"
                   * does not "logically" exist on that node.
                   * No further session will be accepted on
                   * that name.
                   * No datagrams can be sent against that name.
                   * Such an entry will not be used for
                   * purposes of processing incoming request
                   * packets.
                   * The only valid user NetBIOS operation
                   * against such a name is DELETE NAME.
                   */
             END
        NAME RELEASE REQUEST:
             IF caching is being done THEN
             BEGIN
                  remove entry from cache;
             END
        NAME UPDATE REQUEST:
             IF caching is being done THEN
             BEGIN
                  IF entry exists in cache already,
                       update cache;
                  ELSE IF name is "interesting" THEN
                  BEGIN
                       add entry to cache;
                  END
             END

        NODE STATUS REQUEST:
             IF name exists in local name table THEN
             BEGIN
                  /*
                   * send only those names that are
                   * in the same scope as the scope
                   * field in the request packet
                   */

                  send NODE STATUS RESPONSE;
             END
   END
```

5.1.2. P-NODE ACTIVITY

All packets sent or received by P nodes are unicast UDP packets. A P node sends name service requests to the NBNS node that is specified in the P-node configuration.

5.1.2.1. P-NODE ADD_NAME

```
PROCEDURE add_name(newname)

/*
 * Host initiated processing for a P node
 */

BEGIN

     REPEAT
          /*
           * build packet
           */

          ONT = P;
          G = UNIQUE;
          ...

          /*
           * send request
           */

          unicast NAME REGISTRATION REQUEST packet;

          /*
           * NBNS will send response packet
           */

          IF receive a WACK RESPONSE THEN
               pause(time from TTL field of response);
          ELSE
               pause(UCAST_REQ_RETRY_TIMEOUT);
     UNTIL response packet is received OR
          retransmit count has been exceeded

     IF no response packet was received THEN
     BEGIN /* no response */
          /*
           * NBNS is down.  Cannot claim name.
           */

          return failure; /* name cannot be claimed */
     END /* no response */
     ELSE
```

```
        BEGIN /* response */
            IF NOT response tid = request tid THEN
            BEGIN
             /*  Packet may belong to another transaction  */
             ignore response packet;
            END
            ELSE
            CASE packet type OF

            POSITIVE NAME REGISTRATION RESPONSE:

                 /*
                  * name can be added
                  */

                 adjust refresh timeout value, TTL, for this name;
                 return success;      /* name can be added */

            NEGATIVE NAME REGISTRATION RESPONSE:
                 return failure; /* name cannot be added */

            END-NODE CHALLENGE REGISTRATION REQUEST:
            BEGIN /* end node challenge */

                 /*
                  * The response packet has in it the
                  * address of the presumed owner of the
                  * name.  Challenge that owner.
                  * If owner either does not
                  * respond or indicates that he no longer
                  * owns the name, claim the name.
                  * Otherwise, the name cannot be claimed.
                  *
                  */

                 REPEAT
                  /*
                   * build packet
                   */
                  ...

                  unicast NAME QUERY REQUEST packet to the
                       address contained in the END NODE
                       CHALLENGE RESPONSE packet;

                  /*
                   * remote node may send response packet
                   */

                  pause(UCAST_REQ_RETRY_TIMEOUT);
                 UNTIL response packet is received or
                     retransmit count has been exceeded
                 IF no response packet is received OR
                       NEGATIVE NAME QUERY RESPONSE packet
                       received THEN
                 BEGIN /* update */

                  /*
                   * name can be claimed
                   */

                  REPEAT

                      /*
                       * build packet
                       */
                       ...

                      unicast NAME UPDATE REQUEST to NBNS;

                      /*
                       * NBNS node will send response packet
                       */

                      IF receive a WACK RESPONSE THEN
                            pause(time from TTL field of response);
                      ELSE
                            pause(UCAST_REQ_RETRY_TIMEOUT);
                  UNTIL response packet is received or
                      retransmit count has been exceeded
                  IF no response packet received THEN
                  BEGIN /* no response */

                       /*
                        * name could not be claimed
                        */

                       return failure;
                  END /* no response */
                  ELSE
                  CASE packet type OF
                       POSITIVE NAME REGISTRATION RESPONSE:
                            /*
                             * add name
                             */
                            return success;
                       NEGATIVE NAME REGISTRATION RESPONSE:

                            /*
                             * you lose  ...
                             */
                            return failure;
                       END /* case */
                 END /* update */
                 ELSE

                 /*
                  * received a positive response to the "challenge"
                  * Remote node still has name
                  */

                  return failure;
            END /* end node challenge */
        END /* response */
   END /* procedure */
```

5.1.2.2. P-NODE ADD GROUP NAME

```
   PROCEDURE add_group_name(newname)

   /*
    * Host initiated processing for a P node
    */

   BEGIN
        /*
         * same as for a unique name, except that the
         * request packet must indicate that a
         * group name claim is being made.
         */

        ...
        G = GROUP;
        ...

        /*
         * send packet
         */
         ...


   END
```

5.1.2.3. P-NODE FIND NAME

```
   PROCEDURE find_name(name)

   /*
    * Host initiated processing for a P node
    */

   BEGIN
```

```
        REPEAT
             /*
              * build packet
              */

             ONT = P;
             G = DONT CARE;

             unicast NAME QUERY REQUEST packet;

             /*
              * a NBNS node might send response packet
              */

             IF receive a WACK RESPONSE THEN
                  pause(time from TTL field of response);
             ELSE
                  pause(UCAST_REQ_RETRY_TIMEOUT);
        UNTIL response packet received OR
             max transmit threshold exceeded

        IF no response packet received THEN
             return failure;
        ELSE
        IF NOT response tid = request tid THEN
             ignore packet;
        ELSE
        CASE packet type OF
        POSITIVE NAME QUERY RESPONSE:
             return success;

        REDIRECT NAME QUERY RESPONSE:

             /*
              * NBNS node wants this end node
              * to use some other NBNS node
              * to resolve the query.
              */

              repeat query with NBNS address
                  in the response packet;
        NEGATIVE NAME QUERY RESPONSE:
             return failure;

        END /* case */
   END /* procedure */
```

5.1.2.4. P-NODE DELETE_NAME

```
   PROCEDURE delete_name (name)
   /*
    * Host initiated processing for a P node
    */

   BEGIN

        REPEAT

             /*
              * build packet
              */
             ...

             /*
              * send request
              */

             unicast NAME RELEASE REQUEST packet;
             IF receive a WACK RESPONSE THEN
                  pause(time from TTL field of response);
             ELSE
                  pause(UCAST_REQ_RETRY_TIMEOUT);
        UNTIL retransmit count has been exceeded
             or response been received

        IF response has been received THEN
        CASE packet type OF
        POSITIVE NAME RELEASE RESPONSE:
             return success;
        NEGATIVE NAME RELEASE RESPONSE:

             /*
              * NBNS does want node to delete this
              * name !!!
              */

             return failure;
        END /* case */
   END /* procedure */
```

5.1.2.5. P-NODE INCOMING PACKET PROCESSING

Processing initiated by reception of packets at a P node

```
   PROCEDURE process_incoming_packet(packet)

   /*
    * Processing initiated by incoming packets at a P node
    */

   BEGIN
```

```
        /*
         * always ignore UDP broadcast packets
         */

        IF packet was sent as a broadcast THEN
        BEGIN
             ignore packet;
             return;
        END
        CASE packet type of

        NAME CONFLICT DEMAND:
             IF name exists in local name table THEN
                  mark name as in conflict;
             return;

        NAME QUERY REQUEST:
             IF name exists in local name table THEN
             BEGIN /* name exists */

                  /*
                   * build packet
                   */
                  ...

                  /*
                   * send response to the IP address and port
                   * number from which the request was received.
                   */

                  send POSITIVE NAME QUERY RESPONSE ;
                  return;
             END /* exists */
             ELSE
             BEGIN /* does not exist */

                  /*
                   * send response to the requestor
                   */

                  send NEGATIVE NAME QUERY RESPONSE ;
                  return;
             END /* does not exist */
        NODE STATUS REQUEST:
             /*
              * Name of "*" may be used for force node to
              * divulge status for administrative purposes
              */
             IF name in local name table OR name = "*" THEN
             BEGIN
                  /*
```

```
                   * Build response packet and
                   * send to requestor node
                   * Send only those names that are
                   * in the same scope as the scope
                   * in the request packet.
                   */

                  send NODE STATUS RESPONSE;
             END

        NAME RELEASE REQUEST:
             /*
              * This will be received if the NBNS wants to flush the
              * name from the local name table, or from the local
              * cache.
              */

             IF name exists in the local name table THEN
             BEGIN
                  delete name from local name table;
                  inform user that name has been deleted;
             END
             ELSE
                  IF name has been cached locally THEN
                  BEGIN
                       remove entry from cache:
                  END

        END /* case */
   END /* procedure */
```

5.1.2.6. P-NODE TIMER INITIATED PROCESSING

Processing initiated by timer expiration.

```
   PROCEDURE timer_expired()
   /*
    * Processing initiated by the expiration of a timer on a P node
    */
   BEGIN
        /*
         * Send a NAME REFRESH REQUEST for each name which the
         * TTL which has expired.
         */
        REPEAT
             build NAME REFRESH REQUEST packet;
             REPEAT
                  send packet to NBNS;

                  IF receive a WACK RESPONSE THEN
                       pause(time from TTL field of response);
                  ELSE
                       pause(UCAST_REQ_RETRY_TIMEOUT);
             UNTIL response packet is received or
                  retransmit count has been exceeded

             CASE packet type OF
               POSITIVE NAME REGISTRATION RESPONSE:
                  /* successfully refreshed */
                  reset TTL timer for this name;

               NEGATIVE NAME REGISTRATION RESPONSE:
                  /*
                   * refused, can't keep name
                   * assume in conflict
                   */
                  mark name as in conflict;
             END /* case */

        UNTIL request sent for all names for which TTL
             has expired
   END /* procedure */
```

5.1.3. M-NODE ACTIVITY

M nodes behavior is similar to that of P nodes with the addition of some B node-like broadcast actions. M node name service proceeds in two steps:

1.Use broadcast UDP based name service. Depending on the operation, goto step 2.

2.Use directed UDP name service.

The following code for M nodes is exactly the same as for a P node, with the exception that broadcast operations are done before P type operation is attempted.

5.1.3.1. M-NODE ADD NAME

```
   PROCEDURE add_name(newname)

   /*
    * Host initiated processing for a M node
    */

   BEGIN

        /*
         * check if name exists on the
         * broadcast area
         */
        REPEAT
            /* build packet */

            ....
            broadcast NAME REGISTRATION REQUEST packet;
            pause(BCAST_REQ_RETRY_TIMEOUT);

        UNTIL response packet is received or
             retransmit count has been  exceeded

        IF valid response received THEN
        BEGIN
             /* cannot claim name */

             return failure;
        END

        /*
         * No objections received within the
         * broadcast area.
         * Send request to name server.
         */

        REPEAT
             /*
              * build packet
              */

             ONT = M;
             ...

             unicast NAME REGISTRATION REQUEST packet;

             /*
              * remote NBNS will send response packet
              */

             IF receive a WACK RESPONSE THEN
                  pause(time from TTL field of response);
             ELSE
                  pause(UCAST_REQ_RETRY_TIMEOUT);

        UNTIL response packet is received or
             retransmit count has been exceeded

        IF no response packet was received THEN
        BEGIN /* no response */
             /*
              * NBNS is down.  Cannot claim name.
              */
             return failure; /* name cannot be claimed */
        END /* no response */
        ELSE
        BEGIN /* response */
            IF NOT response tid = request tid THEN
            BEGIN
             ignore response packet;
            END
            ELSE
            CASE packet type OF
            POSITIVE NAME REGISTRATION RESPONSE:

                 /*
                  * name can be added
                  */

                 adjust refresh timeout value, TTL;
                 return success;      /* name can be added */

            NEGATIVE NAME REGISTRATION RESPONSE:
                 return failure; /* name cannot be added */

            END-NODE CHALLENGE REGISTRATION REQUEST:
            BEGIN /* end node challenge */

                 /*
                  * The response packet has in it the
                  * address of the presumed owner of the
                  * name.  Challenge that owner.
                  * If owner either does not
                  * respond or indicates that he no longer
                  * owns the name, claim the name.
                  * Otherwise, the name cannot be claimed.
                  *
                  */

                 REPEAT
                  /*
                   * build packet
                   */
                  ...

                  /*
                   * send packet to address contained in the
                   * response packet
                   */

                  unicast NAME QUERY REQUEST packet;

                  /*
                   * remote node may send response packet
                   */

                  pause(UCAST_REQ_RETRY_TIMEOUT);

                 UNTIL response packet is received or
                     retransmit count has been exceeded
                 IF no response packet is received THEN
                 BEGIN /* no response */

                  /*
                   * name can be claimed
                   */
                  REPEAT

                      /*
                       * build packet
                       */
                       ...

                      unicast NAME UPDATE REQUEST to NBNS;

                      /*
                       * NBNS node will send response packet
                       */

                      IF receive a WACK RESPONSE THEN
                            pause(time from TTL field of response);
                  ELSE
                       pause(UCAST_REQ_RETRY_TIMEOUT);

                  UNTIL response packet is received or
                      retransmit count has been exceeded
                  IF no response packet received THEN
                  BEGIN /* no response */

                       /*
                        * name could not be claimed
                        */

                       return failure;
                  END /* no response */
                  ELSE
                  CASE packet type OF
                  POSITIVE NAME REGISTRATION RESPONSE:
                       /*
                        * add name
                        */

                       return success;
                  NEGATIVE NAME REGISTRATION RESPONSE:
                       /*
                        * you lose  ...
                        */

                       return failure;
                  END /* case */
                 END /* no response */
                 ELSE
                 IF NOT response tid = request tid THEN
                 BEGIN
                  ignore response packet;
                 END

                 /*
                  * received a response to the "challenge"
                  * packet
                  */

                 CASE packet type OF
                 POSITIVE NAME QUERY:

                  /*
                   * remote node still has name.
                   */

                  return failure;
                 NEGATIVE NAME QUERY:

                  /*
                   * remote node no longer has name
                   */

                  return success;
                 END /* case */
            END /* end node challenge */
            END /* case */
        END /* response */
   END /* procedure */
```

5.1.3.2. M-NODE ADD GROUP NAME

```
   PROCEDURE add_group_name(newname)

   /*
    * Host initiated processing for a P node
    */

   BEGIN
        /*
         * same as for a unique name, except that the
         * request packet must indicate that a
```

```
          * group name claim is being made.
          */

         ...
         G = GROUP;
         ...

         /*
          * send packet
          */
          ...


    END
```

5.1.3.3. M-NODE FIND NAME

```
    PROCEDURE find_name(name)

    /*
     * Host initiated processing for a M node
     */

    BEGIN
         /*
          * check if any node on the broadcast
          * area has the name
          */

         REPEAT
              /* build packet */
              ...

              broadcast NAME QUERY REQUEST packet;
              pause(BCAST_REQ_RETRY_TIMEOUT);
         UNTIL response packet received OR
              max transmit threshold exceeded

         IF valid response received THEN
         BEGIN
              save response as authoritative response;
              start_timer(CONFLICT_TIMER);
              return success;
         END

         /*
          * no valid response on the b'cast segment.
          * Try the name server.
          */

         REPEAT
```

```
             /*
              * build packet
              */

             ONT = M;
             G = DONT CARE;

             unicast NAME QUERY REQUEST packet to NBNS;

             /*
              * a NBNS node might send response packet
              */

             IF receive a WACK RESPONSE THEN
                  pause(time from TTL field of response);
             ELSE
                  pause(UCAST_REQ_RETRY_TIMEOUT);
        UNTIL response packet received OR
             max transmit threshold exceeded

        IF no response packet received THEN
             return failure;
        ELSE
        IF NOT response tid = request tid THEN
             ignore packet;
        ELSE
        CASE packet type OF
        POSITIVE NAME QUERY RESPONSE:
             return success;

        REDIRECT NAME QUERY RESPONSE:

             /*
              * NBNS node wants this end node
              * to use some other NBNS node
              * to resolve the query.
              */

              repeat query with NBNS address
                  in the response packet;
        NEGATIVE NAME QUERY RESPONSE:
             return failure;

        END /* case */
   END /* procedure */
```

5.1.3.4. M-NODE DELETE NAME

```
   PROCEDURE delete_name (name)

   /*
```

```
       * Host initiated processing for a P node
       */

      BEGIN
           /*
            * First, delete name on NBNS
            */

           REPEAT

                /*
                 * build packet
                 */
                ...

                /*
                 * send request
                 */

                unicast NAME RELEASE REQUEST packet to NBNS;

                IF receive a WACK RESPONSE THEN
                     pause(time from TTL field of response);
                ELSE
                     pause(UCAST_REQ_RETRY_TIMEOUT);
           UNTIL retransmit count has been exceeded
                or response been received

           IF response has been received THEN
           CASE packet type OF
           POSITIVE NAME RELEASE RESPONSE:
                /*
                 * Deletion of name on b'cast segment is deferred
                 * until after NBNS has deleted the name
                 */

                REPEAT
                     /* build packet */

                     ...
                     broadcast NAME RELEASE REQUEST;
                     pause(BCAST_REQ_RETRY_TIMEOUT);
                UNTIL rexmt threshold exceeded

                return success;
           NEGATIVE NAME RELEASE RESPONSE:

                /*
                 * NBNS does want node to delete this
                 * name
                 */
```

```
             return failure;
        END /* case */
   END /* procedure */
```

5.1.3.5. M-NODE INCOMING PACKET PROCESSING

Processing initiated by reception of packets at a M node

```
   PROCEDURE process_incoming_packet(packet)

   /*
    * Processing initiated by incoming packets at a M node
    */

   BEGIN
        CASE packet type of

        NAME CONFLICT DEMAND:
             IF name exists in local name table THEN
                  mark name as in conflict;
             return;

        NAME QUERY REQUEST:
             IF name exists in local name table THEN
             BEGIN /* name exists */

                  /*
                   * build packet
                   */
                  ...

                  /*
                   * send response to the IP address and port
                   * number from which the request was received.
                   */

                  send POSITIVE NAME QUERY RESPONSE ;
                  return;
             END /* exists */
             ELSE
             BEGIN /* does not exist */

                  /*
                   * send response to the requestor
                   */

                  IF request NOT broadcast THEN
                       /*
                        * Don't send negative responses to
                        * queries sent by B nodes
                        */
                       send NEGATIVE NAME QUERY RESPONSE ;
                  return;
             END /* does not exist */
        NODE STATUS REQUEST:
             BEGIN
             /*
              * Name of "*" may be used for force node to
              * divulge status for administrative purposes
              */
             IF name in local name table OR name = "*" THEN
                  /*
                   * Build response packet and
                   * send to requestor node
                   * Send only those names that are
                   * in the same scope as the scope
                   * in the request packet.
                   */

                  send NODE STATUS RESPONSE;
             END

        NAME RELEASE REQUEST:
             /*
              * This will be received if the NBNS wants to flush the
              * name from the local name table, or from the local
              * cache.
              */

             IF name exists in the local name table THEN
             BEGIN
                  delete name from local name table;
                  inform user that name has been deleted;
             END
             ELSE
                  IF name has been cached locally THEN
                  BEGIN
                       remove entry from cache:
                  END

        NAME REGISTRATION REQUEST (UNIQUE):
             IF name exists in local name table THEN
                  send NEGATIVE NAME REGISTRATION RESPONSE ;
        NAME REGISTRATION REQUEST (GROUP):
             IF name exists in local name table THEN
             BEGIN
                  IF local entry is a unique name THEN
                      send NEGATIVE NAME REGISTRATION RESPONSE ;
             END
        END /* case */
   END /* procedure */
```

5.1.3.6. M-NODE TIMER INITIATED PROCESSING

Processing initiated by timer expiration:

```
PROCEDURE timer_expired()
/*
 * Processing initiated by the expiration of a timer on a M node
 */
BEGIN
     /*
      * Send a NAME REFRESH REQUEST for each name which the
      * TTL which has expired.
      */
     REPEAT
          build NAME REFRESH REQUEST packet;
          REPEAT
               send packet to NBNS;

               IF receive a WACK RESPONSE THEN
                    pause(time from TTL field of response);
               ELSE
                    pause(UCAST_REQ_RETRY_TIMEOUT);
          UNTIL response packet is received or
               retransmit count has been exceeded

          CASE packet type OF
            POSITIVE NAME REGISTRATION RESPONSE:
               /* successfully refreshed */
               reset TTL timer for this name;

            NEGATIVE NAME REGISTRATION RESPONSE:
               /*
                * refused, can't keep name
                * assume in conflict
                */
               mark name as in conflict;
          END /* case */

     UNTIL request sent for all names for which TTL
          has expired
END /* procedure */
```

5.1.4. NBNS ACTIVITY

A NBNS node will receive directed packets from P and M nodes. Reply packets are always sent as directed packets to the source IP address and UDP port number. Received broadcast packets must be ignored.

5.1.4.1. NBNS INCOMING PACKET PROCESSING

```
   PROCEDURE process_incoming_packet(packet)

   /*
    * Incoming packet processing on a NS node
    */

   BEGIN
        IF packet was sent as a broadcast THEN
        BEGIN
             discard packet;
             return;
        END
        CASE packet type of

        NAME REGISTRATION REQUEST (UNIQUE):
             IF unique name exists in data base THEN
             BEGIN /* unique name exists */
                  /*
                   * NBNS node may be a "passive"
                   * server in that it expects the
                   * end node to do the challenge
                   * server.  Such a NBNS node is
                   * called a "non-secure" server.
                   * A "secure" server will do the
                   * challenging before it sends
                   * back a response packet.
                   */

                  IF non-secure THEN
                  BEGIN
                       /*
                        * build response packet
                        */
                       ...


                       /*
                        * let end node do the challenge
                        */

                       send END-NODE CHALLENGE NAME REGISTRATION
                            RESPONSE;
                       return;
                  END
                  ELSE
                  /*
                   * secure server - do the name
                   * challenge operation
                   */
                  REPEAT
                      send NAME QUERY REQUEST;
                      pause(UCAST_REQ_RETRY_TIMEOUT);
                  UNTIL response has been received or
                       retransmit count has been exceeded
                  IF no response was received THEN
                  BEGIN

                       /* node down */

                       update data base - remove entry;
                       update data base - add new entry;
                       send POSITIVE NAME REGISTRATION RESPONSE;
                       return;
                  END
                  ELSE
                  BEGIN /* challenged node replied */
                      /*
                       * challenged node replied with
                       * a response packet
                       */

                      CASE packet type

                      POSITIVE NAME QUERY RESPONSE:

                       /*
                        * name still owned by the
                        * challenged node
                        *
                        * build packet and send response
                        */
                        ...


                       /*
                        * Note: The NBNS will need to
                        * keep track (based on transaction id) of
                        * the IP address and port number
                        * of the original requestor.
                        */

                       send NEGATIVE NAME REGISTRATION RESPONSE;
                       return;
                      NEGATIVE NAME QUERY RESPONSE:

                       update data base - remove entry;
                       update data base - add new  entry;

                       /*
                        * build response packet and send
                        * response
                        */
                       send POSITIVE NAME REGISTRATION RESPONSE;
                       return;
                      END /* case */
                  END /* challenged node replied */
             END /* unique name exists in data base */
             ELSE
             IF group name exists in data base THEN
             BEGIN /* group names exists */

                  /*
                   * Members of a group name are NOT
                   * challenged.
                   * Make the assumption that
                   * at least some of the group members
                   * are still alive.
                   * Refresh mechanism will
                   * allow the NBNS to detect when all
                   * members of a group no longer use that
                   * name
                   */

                   send NEGATIVE NAME REGISTRATION RESPONSE;
             END /* group name exists */
             ELSE
             BEGIN /* name does not exist */

                  /*
                   * Name does not exist in data base
                   *
                   * This code applies to both non-secure
                   * and secure server.
                   */

                  update data base - add new entry;
                  send POSITIVE NAME REGISTRATION RESPONSE;
                  return;
             END

        NAME QUERY REQUEST:
             IF name exists in data base THEN
             BEGIN
                  /*
                   * build response packet and send to
                   * requestor
                   */
                   ...

                  send POSITIVE NAME QUERY RESPONSE;
                  return;
             ELSE
             BEGIN
                  /*
                   * build response packet and send to
                   * requestor
                   */
                   ...

                  send NEGATIVE NAME QUERY RESPONSE;
                  return;
             END

        NAME REGISTRATION REQUEST (GROUP):
             IF name exists in data base THEN
             BEGIN
                  IF local entry is a unique name THEN
                  BEGIN /* local is unique */

                      IF non-secure THEN
                      BEGIN
                       send  END-NODE CHALLENGE NAME
                            REGISTRATION RESPONSE;
                       return;
                      END

                      REPEAT
                       send NAME QUERY REQUEST;
                       pause(UCAST_REQ_RETRY_TIMEOUT);
                      UNTIL response received or
                           retransmit count exceeded
                      IF no response received or
                           NEGATIVE NAME QUERY RESPONSE
                            received THEN
                      BEGIN
                       update data base - remove entry;
                       update data base - add new entry;
                       send POSITIVE NAME REGISTRATION RESPONSE;
                       return;
                      END
                      ELSE
                      BEGIN
                       /*
                        * name still being held
                        * by challenged node
                        */

                        send NEGATIVE NAME REGISTRATION RESPONSE;
                      END
                  END /* local is unique */
                  ELSE
                  BEGIN /* local is group  */
```

```
                       /*
                        * existing entry is a group name
                        */

                       update data base - remove entry;
                       update data base - add new entry;
                       send POSITIVE NAME REGISTRATION RESPONSE;
                       return;
                  END /* local is group */
             END /* names exists */
             ELSE
             BEGIN /* does not exist */

                  /* name does not exist in data base */

                  update data base - add new entry;
                  send POSITIVE NAME REGISTRATION RESPONSE;
                  return;
             END /* does not exist */

        NAME RELEASE REQUEST:

             /*
              * secure server may choose to disallow
              * a node from deleting a name
              */

             update data base - remove entry;
             send POSITIVE NAME RELEASE RESPONSE;
             return;

        NAME UPDATE REQUEST:

             /*
              * End-node completed a successful challenge,
              * no update database
              */

             IF secure server THEN
                  send NEGATIVE NAME REGISTRATION RESPONSE;
             ELSE
             BEGIN /* new entry */
                  IF entry already exists THEN
                       update data base - remove entry;
                  update data base - add new entry;
                  send POSITIVE NAME REGISTRATION RESPONSE;
                  start_timer(TTL);
             END

        NAME REFRESH REQUEST:
             check for consistency;
             IF node not allowed to have name THEN
             BEGIN

                  /*
                   * tell end node that it can't have name
                   */
                  send NEGATIVE NAME REGISTRATION RESPONSE;
             END
             ELSE
             BEGIN

                  /*
                   * send confirmation response to the
                   * end node.
                   */
                  send POSITIVE NAME REGISTRATION;
                  start_timer(TTL);
             END
             return;
        END /* case */
   END /* procedure */
```

5.1.4.2. NBNS TIMER INITIATED PROCESSING

A NS node uses timers to flush out entries from the data base. Each entry in the data base is removed when its timer expires. This time value is a multiple of the refresh TTL established when the name was registered.

```
   PROCEDURE timer_expired()

   /*
    * processing initiated by expiration of TTL for a given name
    */

   BEGIN
        /*
         * NBNS can (optionally) ensure
         * that the node is actually down
         * by sending a NODE STATUS REQUEST.
         * If such a request is sent, and
         * no response is received, it can
         * be assumed that the node is down.
         */
        remove entry from data base;
   END
```

5.2. SESSION SERVICE PROTOCOLS

The following are variables and should be configurable by the NetBIOS user. The default values of these variables is found in "Defined Constants and Variables" in the Detailed Specification.):

- SSN_RETRY_COUNT - The maximum number TCP connection attempts allowable per a single NetBIOS call request.

- SSN_CLOSE_TIMEOUT is the time period to wait when closing the NetBIOS session before killing the TCP connection if session sends are outstanding.

The following are Defined Constants for the NetBIOS Session Service. (See "Defined Constants and Variables" in the Detailed Specification for the value of these constants):

- SSN_SRVC_TCP_PORT - is the globally well-known TCP port allocated for the NetBIOS Session Service. The service accepts TCP connections on this port to establish NetBIOS Sessions. The TCP connection established to this port by the caller is initially used for the exchange of NetBIOS control information. The actual NetBIOS data connection may also pass through this port or, through the retargetting facility, through another port.

5.2.1. SESSION ESTABLISHMENT PROTOCOLS

5.2.1.1. USER REQUEST PROCESSING

```
PROCEDURE listen(listening name, caller name)
/*
 * User initiated processing for B, P and M nodes
 *
 * This procedure assumes that an incoming session will be
 * retargetted here by a session server.
 */
BEGIN
     Do TCP listen; /* Returns TCP port used */
     Register listen with Session Service, give names and
          TCP port;

     Wait for TCP connection to open;   /* Incoming call */

     Read SESSION REQUEST packet from connection

     Process session request (see section on
          processing initiated by the reception of session
          service packets);
```

```
        Inform Session Service that NetBIOS listen is complete;

        IF session established THEN
             return success and session information to user;
        ELSE
             return failure;
   END /* procedure */

   PROCEDURE call(calling name, called name)
   /*
    * user initiated processing for B, P and M nodes
    */

   /*
    * This algorithm assumes that the called name is a unique name.
    * If the called name is a group name, the call() procedure
    * needs to cycle through the members of the group
    * until either (retry_count == SSN_RETRY_COUNT) or
    * the list has been exhausted.
    */
   BEGIN
        retry_count = 0;
        retarget = FALSE;   /* TRUE: caller is being retargetted */
        name_query = TRUE;  /* TRUE: caller must begin again with */
                            /*       name query. */

        REPEAT
             IF name_query THEN
             BEGIN
                  do name discovery, returns IP address;
                  TCP port = SSN_SRVC_TCP_PORT;

                  IF name discovery fails THEN
                       return failure;
                  ELSE
                       name_query = FALSE;
             END

             /*
              * now have IP address and TCP port of
              * remote party.
              */

             establish TCP connection with remote party, use an
                  ephemeral port as source TCP port;
             IF connection refused THEN
             BEGIN
                  IF retarget THEN
                  BEGIN
                       /* retry */
                       retarget = FALSE;
                       use original IP address and TCP port;
                       goto LOOP;
                  END

                  /* retry for just missed TCP listen */

                  pause(SESSION_RETRY_TIMER);
                  establish TCP connection, again use ephemeral
                       port as source TCP port;

                  IF connection refused OR
                     connection timed out THEN
                       return failure;
             END
             ELSE
             IF connection timed out THEN
             BEGIN
                  IF retarget THEN
                  BEGIN
                       /* retry */
                       retarget = FALSE;
                       use original IP address and TCP port;
                       goto LOOP;
                  END
                  ELSE
                  BEGIN
                       /*
                        * incorrect name discovery was done,
                        * try again
                        */

                       inform name discovery process of
                            possible error;
                       name_query = TRUE;
                       goto LOOP;
                  END
             END

             /*
              * TCP connection has been established
              */

             wait for session response packet;
             CASE packet type OF

                POSITIVE SESSION RESPONSE:
                  return success and session established
                       information;

                NEGATIVE SESSION RESPONSE:
                BEGIN
```

```
                  CASE error OF
                     NOT LISTENING ON CALLED NAME:
                     NOT LISTENING FOR CALLING NAME:
                     BEGIN
                       kill TCP connection;
                       return failure;
                     END

                     CALLED NAME NOT PRESENT:
                     BEGIN
                       /*
                        * called name does not exist on
                        * remote node
                        */

                       inform name discovery procedure
                            of possible error;

                       IF this is a P or M node THEN
                       BEGIN
                            /*
                             * Inform NetBIOS Name Server
                             * it has returned incorrect
                             * information.
                             */
                            send NAME RELEASE REQUEST for called
                              name and IP address to
                              NetBIOS Name Server;
                       END
                       /* retry from beginning */
                       retarget = FALSE;
                       name_query = TRUE;
                       goto LOOP;
                     END /* called name not present */
                  END /* case */
                END /* negative response */

                RETARGET SESSION RESPONSE:
                BEGIN
                  close TCP connection;
                  extract IP address and TCP port from
                       response;
                  retarget = TRUE;
                END /* retarget response */
             END /* case */

   LOOP:          retry_count = retry_count + 1;

        UNTIL (retry_count > SSN_RETRY_COUNT);
        return failure;
   END /* procedure */
```

5.2.1.2. RECEIVED PACKET PROCESSING

These are packets received on a TCP connection before a session has been established. The listen routines attached to a NetBIOS user process need not implement the RETARGET response section. The user process version, separate from a shared Session Service, need only accept (POSITIVE SESSION RESPONSE) or reject (NEGATIVE SESSION RESPONSE) a session request.

```
PROCEDURE session_packet(packet)
/*
 * processing initiated by receipt of a session service
 * packet for a session in the session establishment phase.
 * Assumes the TCP connection has been accepted.
 */
BEGIN
     CASE packet type

        SESSION REQUEST:
        BEGIN
          IF called name does not exist on node THEN
          BEGIN
               send NEGATIVE SESSION RESPONSE with CALLED
                    NAME NOT PRESENT error code;
               close TCP connection;
          END

          Search for a listen with CALLING NAME for CALLED
               NAME;
          IF matching listen is found THEN
          BEGIN
               IF port of listener process is port TCP
                  connection is on THEN
               BEGIN
                    send POSITIVE SESSION RESPONSE;

                    Hand off connection to client process
                         and/or inform user session is
                         established;
               END
               ELSE
               BEGIN
                    send RETARGET SESSION RESPONSE with
                         listener's IP address and
                         TCP port;
                    close TCP connection;
               END
          END
          ELSE
          BEGIN
               /* no matching listen pending */
```

```
                  send NEGATIVE SESSION RESPONSE with either
                       NOT LISTENING ON CALLED NAME or NOT
                       LISTENING FOR CALLING NAME error
                       code;
                  close TCP connection;
             END
           END /* session request */
        END /* case */
   END /* procedure */
```

5.2.2. SESSION DATA TRANSFER PROTOCOLS

5.2.2.1. USER REQUEST PROCESSING

```
   PROCEDURE send_message(user_message)
   BEGIN
        build SESSION MESSAGE header;
        send SESSION MESSAGE header;
        send user_message;
        reset and restart keep-alive timer;
        IF send fails THEN
        BEGIN
             /*
              * TCP connection has failed */
              */
             close NetBIOS session;
             inform user that session is lost;
             return failure;
        END
        ELSE
             return success;
   END
```

5.2.2.2. RECEIVED PACKET PROCESSING

These are packets received after a session has been established.

```
   PROCEDURE session_packet(packet)
   /*
    * processing initiated by receipt of a session service
    * packet for a session in the data transfer phase.
    */
   BEGIN
        CASE packet type OF

           SESSION  MESSAGE:
           BEGIN
             process message header;
             read in user data;
             reset and restart keep-alive timer;
             deliver data to user;
           END /* session message */

           SESSION KEEP ALIVE:
             discard packet;

        END /* case */
   END /* procedure */
```

5.2.2.3. PROCESSING INITIATED BY TIMER

```
   PROCEDURE session_ka_timer()
   /*
    * processing initiated when session keep alive timer expires
    */
   BEGIN
            send SESSION KEEP ALIVE, if configured;
            IF send fails THEN
            BEGIN
             /* remote node, or path to it, is down */

             abort TCP connection;
             close NetBIOS session;
             inform user that session is lost;
             return;
            END
   END /* procedure */
```

5.2.3. SESSION TERMINATION PROTOCOLS

5.2.3.1. USER REQUEST PROCESSING

```
   PROCEDURE close_session()

   /* initiated by a user request to close a session */

   BEGIN
        close gracefully the TCP connection;

        WAIT for the connection to close or SSN_CLOSE_TIMEOUT
             to expire;

        IF time out expired THEN
             abort TCP connection;
   END /* procedure */
```

5.2.3.2. RECEPTION INDICATION PROCESSING

```
   PROCEDURE close_indication()
   /*
    * initiated by a TCP indication of a close request from
    * the remote connection partner.
    */
   BEGIN
        close gracefully TCP connection;

        close NetBIOS session;

        inform user session closed by remote partner;
   END /* procedure */
```

5.3. NetBIOS DATAGRAM SERVICE PROTOCOLS

The following are GLOBAL variables and should be NetBIOS user configurable:

- SCOPE_ID: the non-leaf section of the domain name preceded by a '.' which represents the domain of the NetBIOS scope for the NetBIOS name. The following protocol description only supports single scope operation.

- MAX_DATAGRAM_LENGTH: the maximum length of an IP datagram. The minimal maximum length defined in for IP is 576 bytes. This value is used when determining whether to fragment a NetBIOS datagram. Implementations are expected to be capable of receiving unfragmented NetBIOS datagrams up to their maximum size.

- BROADCAST_ADDRESS: the IP address B-nodes use to send datagrams with group name destinations and broadcast datagrams. The default is the IP broadcast address for a single IP network.

The following are Defined Constants for the NetBIOS Datagram Service:

- DGM_SRVC_UDP_PORT: the globally well-known UDP port allocated where the NetBIOS Datagram Service receives UDP packets. See section 6, "Defined Constants", for its value.

5.3.1. B NODE TRANSMISSION OF NetBIOS DATAGRAMS

```
   PROCEDURE send_datagram(data, source, destination, broadcast)

   /*
    * user initiated processing on B node
    */

   BEGIN
        group = FALSE;

        do name discovery on destination name, returns name type and
             IP address;
        IF name type is group name THEN
        BEGIN
             group = TRUE;
        END

        /*
         * build datagram service UDP packet;
         */
        convert source and destination NetBIOS names into
             half-ASCII, biased encoded name;
        SOURCE_NAME = cat(source, SCOPE_ID);
        SOURCE_IP = this nodes IP address;
        SOURCE_PORT =  DGM_SRVC_UDP_PORT;

        IF NetBIOS broadcast THEN
        BEGIN
             DESTINATION_NAME = cat("*", SCOPE_ID)
        END
        ELSE
        BEGIN
             DESTINATION_NAME = cat(destination, SCOPE_ID)
        END

        MSG_TYPE = select_one_from_set
             {BROADCAST, DIRECT_UNIQUE, DIRECT_GROUP}
        DGM_ID = next transaction id for Datagrams;
        DGM_LENGTH = length of data + length of second level encoded
             source and destination names;

        IF (length of the NetBIOS Datagram, including UDP and
            IP headers, > MAX_DATAGRAM_LENGTH) THEN
        BEGIN
             /*
              * fragment NetBIOS datagram into 2 UDP packets
              */
             Put names into 1st UDP packet and any data that fits
                  after names;
             Set MORE and FIRST bits in 1st UDP packet's FLAGS;
             OFFSET in 1st UDP = 0;

             Replicate NetBIOS Datagram header from 1st UDP packet
                  into 2nd UDP packet;
             Put rest of data in 2nd UDP packet;
             Clear MORE and FIRST bits in 2nd UDP packet's FLAGS;
             OFFSET in 2nd UDP = DGM_LENGTH - number of name and
                  data bytes in 1st UDP;
        END
        BEGIN
             /*
              * Only need one UDP packet
              */
```

```
               USER_DATA = data;
               Clear MORE bit and set FIRST bit in FLAGS;
               OFFSET = 0;
          END

          IF (group == TRUE) OR (NetBIOS broadcast) THEN
          BEGIN
               send UDP packet(s) to BROADCAST_ADDRESS;
          END
          ELSE
          BEGIN
               send UDP packet(s) to IP address returned by name
                  discovery;
          END
     END /* procedure */
```

5.3.2. P AND M NODE TRANSMISSION OF NetBIOS DATAGRAMS

```
     PROCEDURE send_datagram(data, source, destination, broadcast)

     /*
      * User initiated processing on P and M node.
      *
      * This processing is the same as for B nodes except for
      * sending broadcast and multicast NetBIOS datagrams.
      */

     BEGIN
          group = FALSE;

          do name discovery on destination name, returns name type
               and IP address;
          IF name type is group name THEN
          BEGIN
               group = TRUE;
          END

          /*
           * build datagram service UDP packet;
           */
          convert source and destination NetBIOS names into
               half-ASCII, biased encoded name;
          SOURCE_NAME = cat(source, SCOPE_ID);
          SOURCE_IP = this nodes IP address;
          SOURCE_PORT =  DGM_SRVC_UDP_PORT;

          IF NetBIOS broadcast THEN
          BEGIN
               DESTINATION_NAME = cat("*", SCOPE_ID)
          END
          ELSE
```

```
        BEGIN
             DESTINATION_NAME = cat(destination, SCOPE_ID)
        END

        MSG_TYPE = select_one_from_set
             {BROADCAST, DIRECT_UNIQUE, DIRECT_GROUP}
        DGM_ID = next transaction id for Datagrams;
        DGM_LENGTH = length of data + length of second level encoded
             source and destination names;

        IF (length of the NetBIOS Datagram, including UDP and
            IP headers, > MAX_DATAGRAM_LENGTH) THEN
        BEGIN
             /*
              * fragment NetBIOS datagram into 2 UDP packets
              */
             Put names into 1st UDP packet and any data that fits
                  after names;
             Set MORE and FIRST bits in 1st UDP packet's FLAGS;

             OFFSET in 1st UDP = 0;

             Replicate NetBIOS Datagram header from 1st UDP packet
                  into 2nd UDP packet;
             Put rest of data in 2nd UDP packet;
             Clear MORE and FIRST bits in 2nd UDP packet's FLAGS;
             OFFSET in 2nd UDP = DGM_LENGTH - number of name and
                  data bytes in 1st UDP;
        END
        BEGIN
             /*
              * Only need one UDP packet
              */
             USER_DATA = data;
             Clear MORE bit and set FIRST bit in FLAGS;
             OFFSET = 0;
        END

        IF (group == TRUE) OR (NetBIOS broadcast) THEN
        BEGIN
             /*
              * Sending of following query is optional.
              * Node may send datagram to NBDD immediately
              * but NBDD may discard the datagram.
              */
             send DATAGRAM QUERY REQUEST to NBDD;
             IF response is POSITIVE QUERY RESPONSE THEN
                  send UDP packet(s) to NBDD Server IP address;
             ELSE
             BEGIN
                  get list of destination nodes from NBNS;
                  FOR EACH node in list
                  BEGIN
                       send UDP packet(s) to this node's
                            IP address;
                  END
             END
        END
        ELSE
        BEGIN
             send UDP packet(s) to IP address returned by name
                discovery;
   END /* procedure */
```

5.3.3. RECEPTION OF NetBIOS DATAGRAMS BY ALL NODES

The following algorithm discards out of order NetBIOS Datagram fragments. An implementation which reassembles out of order NetBIOS Datagram fragments conforms to this specification. The fragment discard timer is initialized to the value FRAGMENT_TO. This value should be user configurable. The default value is given in Section 6, "Defined Constants and Variables".

```
   PROCEDURE datagram_packet(packet)

   /*
    * processing initiated by datagram packet reception
    * on B, P and M nodes
    */
   BEGIN
        /*
         * if this node is a P node, ignore
         * broadcast packets.
         */

        IF this is a P node AND incoming packet is
             a broadcast packet THEN
        BEGIN
             discard packet;
        END

        CASE packet type OF

           DATAGRAM SERVICE:
           BEGIN
             IF FIRST bit in FLAGS is set THEN
             BEGIN
                  IF MORE bit in FLAGS is set THEN
                  BEGIN
                       Save 1st UDP packet of the Datagram;
                       Set this Datagram's fragment discard
                         timer to FRAGMENT_TO;
                       return;
                  END
                  ELSE
                       Datagram is composed of a single
                         UDP packet;
             END
             ELSE
             BEGIN
                  /* Have the second fragment of a Datagram */

                  Search for 1st fragment by source IP address
                     and DGM_ID;
                  IF found 1st fragment THEN
                       Process both UDP packets;
                  ELSE
                  BEGIN
                       discard 2nd fragment UDP packet;
                       return;
                  END
             END

             IF DESTINATION_NAME is '*' THEN
             BEGIN
                  /* NetBIOS broadcast */

                  deliver USER_DATA from UDP packet(s) to all
                       outstanding receive broadcast
                       datagram requests;
                  return;
             END
             ELSE
             BEGIN /* non-broadcast */
                  /* Datagram for Unique or Group Name */

                  IF DESTINATION_NAME is not present in the
                     local name table THEN
                  BEGIN
                       /* destination not present */
                       build DATAGRAM ERROR packet, clear
                            FIRST and MORE bit, put in
                            this nodes IP and PORT, set
                            ERROR_CODE;
                       send DATAGRAM ERROR packet to
                            source IP address and port
                            of UDP;
                       discard UDP packet(s);
                       return;
                  END
                  ELSE
                  BEGIN /* good */
                       /*
```

```
                        * Replicate received NetBIOS datagram for
                        * each recipient
                        */
                       FOR EACH pending NetBIOS user's receive
                            datagram operation
                       BEGIN
                            IF source name of operation
                               matches destination name
                               of packet THEN
                            BEGIN
                               deliver USER_DATA from UDP
                                 packet(s);
                            END
                       END /* for each */
                       return;
                  END /* good */
             END /* non-broadcast */
            END /* datagram service */

           DATAGRAM ERROR:
           BEGIN
                /*
                 * name service returned incorrect information
                 */

                inform local name service that incorrect
                  information was provided;

                IF this is a P or M node THEN
                BEGIN
                     /*
                      * tell NetBIOS Name Server that it may
                      * have given incorrect information
                      */

                     send NAME RELEASE REQUEST with name
                       and incorrect IP address to NetBIOS
                       Name Server;
                END
           END /* datagram error */

        END /* case */
   END
```

5.3.4. PROTOCOLS FOR THE NBDD

The key to NetBIOS Datagram forwarding service is the packet delivered to the destination end node must have the same NetBIOS header as if the source end node sent the packet directly to the destination end node. Consequently, the NBDD does not reassemble NetBIOS Datagrams. It forwards the UDP packet as is.

```
   PROCEDURE  datagram_packet(packet)

   /*
    * processing initiated by a incoming datagram service
    * packet on a NBDD node.
    */

   BEGIN
        CASE packet type OF

           DATAGRAM SERVICE:
           BEGIN
                IF packet was sent as a directed
                   NetBIOS datagram THEN
                BEGIN
                  /*
                   * provide group forwarding service
                   *
                   * Forward datagram to each member of the
                   * group.  Can forward via:
                   *   1) get list of group members and send
                   *   the DATAGRAM SERVICE packet unicast
                   *   to each
                   *   2) use Group Multicast, if available
                   *   3) combination of 1) and 2)
                   */

                  ...

                END

                ELSE
                BEGIN
                  /*
                   * provide broadcast forwarding service
                   *
                   * Forward datagram to every node in the
                   * NetBIOS scope.  Can forward via:
                   *   1) get list of group members and send
                   *   the DATAGRAM SERVICE packet unicast
                   *   to each
                   *   2) use Group Multicast, if available
                   *   3) combination of 1) and 2)
                   */

                  ...

                END
           END /* datagram service */

           DATAGRAM ERROR:
```

```
           BEGIN
             /*
              * Should never receive these because Datagrams
              * forwarded have source end node IP address and
              * port in NetBIOS header.
              */

             send DELETE NAME REQUEST with incorrect name and
                  IP address to NetBIOS Name Server;

           END /* datagram error */

           DATAGRAM QUERY REQUEST:
           BEGIN
             IF can send packet to DESTINATION_NAME THEN
             BEGIN
                  /*
                   * NBDD is able to relay Datagrams for
                   * this name
                   */

                  send POSITIVE DATAGRAM QUERY RESPONSE to
                    REQUEST source IP address and UDP port
                    with request's DGM_ID;
             END
             ELSE
             BEGIN
                  /*
                   * NBDD is NOT able to relay Datagrams for
                   * this name
                   */

                  send NEGATIVE DATAGRAM QUERY RESPONSE to
                    REQUEST source IP address and UDP port

                    with request's DGM_ID;
             END
           END /* datagram query request */

        END /* case */
   END /* procedure */
```

6. DEFINED CONSTANTS AND VARIABLES

GENERAL:

SCOPE_ID	The name of the NetBIOS scope. This is expressed as a character string meeting the requirements of the domain name system and without a leading or trailing "dot". An implementation may elect to make this a single global value for the node or allow it to be specified with each separate NetBIOS name (thus permitting cross-scope references.)
BROADCAST_ADDRESS	An IP address composed of the nodes's network and subnetwork numbers with all remaining bits set to one. I.e. "Specific subnet" broadcast addressing according to section 2.3 of RFC 950.
BCAST_REQ_RETRY_TIMEOUT	250 milliseconds. An adaptive timer may be used.
BCAST_REQ_RETRY_COUNT	3
UCAST_REQ_RETRY_TIMEOUT	5 seconds An adaptive timer may be used.
UCAST_REQ_RETRY_COUNT	3
MAX_DATAGRAM_LENGTH	576 bytes (default)

NAME SERVICE:

REFRESH_TIMER	Negotiated with NBNS for each name.
CONFLICT_TIMER	1 second Implementations may chose a longer value.
NAME_SERVICE_TCP_PORT	137 (decimal)

NAME_SERVICE_UDP_PORT	137 (decimal)
INFINITE_TTL	0

SESSION SERVICE:

SSN_SRVC_TCP_PORT	139 (decimal)
SSN_RETRY_COUNT	4 (default) Re-configurable by user.
SSN_CLOSE_TIMEOUT	30 seconds (default) Re-configurable by user.
SSN_KEEP_ALIVE_TIMEOUT	60 seconds, recommended, may be set to a higher value. (Session keep-alives are used only if configured.)

DATAGRAM SERVICE:

DGM_SRVC_UDP_PORT	138 (decimal)
FRAGMENT_TO	2 seconds (default)

REFERENCES

[1] "Protocol Standard For a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods", RFC 1001, March 1987.

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

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