NetBIOS, NetBEUI, NBF, NBT, NBIPX, SMB, CIFS Networking

Timothy D Evans

NetBIOS, NetBEUI, NBF, NBT, NBIPX, SMB, CIFS Networking
by Timothy D Evans

> **Important:** This documentation is revised from time to time. Some of the technology described is constantly changing and being developed, especially the higher level protocols. Thus this document may not always be up to date. The reader is encouraged to ensure they have the latest version.

Table of Contents

Preface

While there is documentation readily available for protocol suits such as AppleTalk, DECnet, IPX/SPX and TCP/IP, it is difficult to find documentation for the suite or family of protocols which includes the NetBIOS Frames Protocol, NBF, (often referred to as NetBEUI or sometimes as NetBIOS), the Server Message Block protocol, SMB, and Common Internet File System, CIFS; this documentation attempts to provide some information on these protocols.

This document is primarily concerned with the networking protocols rather than specific implementations such as Samba, which are well documented elsewhere. Network programming (and discussion of the various APIs) is also outside the scope of this documentation.

Who should read this documentation

It is assumed that the reader is familiar with one or more networking protocols. Comparisons are made with other well-known protocols in order to better explain the roles of the various protocols described here and how they fit together.

Organization of this documentation

This documentation is organized in to the following chapters:

Introduction

The various protocols to be discussed are introduced and a brief history is provided.

NetBIOS, NetBEUI, NetBIOS Frames Protocol

The NetBIOS Frames Protocol (NBF) is described in terms of the various protocols that were associated with the original NetBIOS implementation.

Supporting Technology, Ethernet and Token Ring

This chapter discusses the various technologies used when NetBIOS is implemented "on the wire".

Encapsulation in TCP/IP

The most popular configuration, NetBIOS over TCP/IP is described here.

Encapsulation in various protocols and encapsulating

NetBIOS can be encapsulated in many protocols and some of the configurations are described in this chapter.

Server Message Block Protocol

The SMB protocol, used for file and print sharing is examined in this chapter.

Browser Service

Although the Browser Service is part of SMB networking (and indeed is implemented over SMB frames), the protocols are sufficiently important to merit particular discussion.

CIFS and the future

This chapter looks at the latest implementation of the SMB protocol, now called CIFS.

Appendices

Three appendices provide some additional information. The way in which the protocols discussed might be mapped to the OSI model is illustrated. Information on the original NetBIOS protocols in the IBM PC Network is provided. A brief look at Microsoft's Active Directory is also included.

A glossary is included for convenience. Following a Bibliography is a brief history of this documentation.

Acknowledgments

I would like to thank the following people for their comments and corrections.

- Ernie Cooper (bama@us.ibm.com)
- Giampaolo Tomassoni (tomassoni@ftbcc.it)

Notation

Hexadecimal numbers are shown either as 0xNNNN or NNNNh.

Language

This document has been written in UK English. My apologies for any spelling or grammatical errors.

Chapter 1. Introduction

There is a suite or family of protocols which includes the NetBIOS Frames Protocol, NBF, (often referred to as NetBEUI or sometimes as NetBIOS), the Server Message Block protocol, SMB, and Common Internet File System, CIFS. These protocols are associated with the original NetBIOS implementation with which they have a historical link.

Many systems use SMB including Microsoft's Windows for Workgroups, Windows 95 / 98 / ME, LAN Manager, Windows NT, Windows 2000 and IBM's OS/2 and LAN Server, NetWare 6 and the SAMBA implementation. SAMBA is freely available for a wide range of systems and further information can be found at the SAMBA web site. *http://www.samba.org* [1]

This document begins by describing NetBIOS (Network Basic Input / Output System) also known as NetBEUI (NetBIOS Extended User Interface) or NBF (NetBIOS Frames protocol) in terms of an original suite of protocols which includes the Name Management Protocol (NMP), Diagnostic and Monitoring Protocol (DMP), User Datagram Protocol (UDP) and the Session Management Protocol (SMP) that were used in the original implementation.

Following a brief description of supporting technologies such as Ethernet and Token Ring, encapsulation of these protocols is considered as well as using these protocols to encapsulate other protocols.

There is no formal standard that defines the protocol(s) used with NetBIOS; in practice the IBM LAN Technical Reference IEEE 802.2 and NetBIOS Application Program Interface is used as a reference (see *Bibliography*).

There are many implementations of NetBIOS networking and these implementations are generally incompatible. It is because of the diversity and lack of a formal standard that makes understanding NetBIOS networking difficult.

It is not clear whether there is only one protocol or several protocols involved in NetBIOS networking. The original implementation for the PC Network certainly seemed to have the above-mentioned protocols (NMP, DMP, UDP and SMP) however the distinction is less clear with NetBIOS on Token-Ring and other implementations. Given that at least network layer and session layer functions are involved, the various packets used will be discussed in terms of the original protocols for convenience, even if the distinctions are somewhat arbitrary.

Following descriptions of the lower level protocols and encapsulation, important higher level protocols (such as SMB, CIFS and the browser service) that run over these lower protocols are described. The situation with respect to the higher level protocols is also complicated; the protocols (SMB and CIFS) were developed as proprietary protocols and information has been difficult to obtain. Although information has been released from time to time, it is not always easy to obtain information on the latest version. Currently Microsoft continues to develop CIFS for it's range of operating systems. Teams of developers such as the SAMBA group reverse engineer the technology. This documentation presents information that is publicly available.

History

The NetBIOS interface was developed for International Business Machines Corporation (IBM) in 1983 by Sytec Inc. (which became Hughes LAN Systems, then Whittaker Communications). This operated over proprietary Sytec protocols on IBM's PC Net-

work which is a broadband local area network. The broadband PC Network is a bus-attached LAN, which can accommodate up to 72 connecting devices. The baseband PC Network is also a bus-attached LAN which can accommodate up to 80 connecting devices; *It is important to note the scale of LAN which NetBIOS was designed for.* NetBIOS was not designed for large networks.

When IBM announced the Token-Ring, an emulator for NetBIOS was produced allowing applications developed for the PC Network to operate on Token-Ring. The NetBIOS Extended User Interface (NetBEUI) was introduced in 1985. The Token-Ring network can accommodate up to 260 devices on one ring and multiple rings can be connected by Bridges.

In 1986 Novell released Advanced NetWare version 2.0. With version 2.0 and all subsequent packages a NetBIOS interface has been included; Novell implemented NetBIOS encapsulated in IPX/SPX. Later Microsoft reverse- engineered the technology to provide encapsulation of NetBIOS in IPX/SPX that is compatible with the Novell implementation.

With the Personal System /2 computer (PS/2) in 1987, IBM announced the PC LAN Support Program which included a NetBIOS driver.

In March 1987, RFC 1001 was published which described a "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport".

Prior to the IBM Lan Support Program, versions of NetBIOS were named with version numbers 1.X. With the LAN Support Program the following NetBIOS versions were used:

Table 1-1. LAN Support Program versions compared with NetBIOS versions

LAN Support Program version	NetBIOS version
1.00	2.0
1.01	2.1
1.02	2.2

Version 2.x of NetBIOS has been superseded by NetBIOS version 3.0 and version 4.0.

In 1987 Microsoft announced the LAN Manager which runs natively over NetBIOS frames.

Microsoft and Intel introduced the SMB Core Protocol in 1988 (SMB-CORE.PS). SMB has been developed during subsequent years and is widely used be many systems. The protocol began life as a proprietary protocol and documentation was very difficult to find. A Version of the protocol (version 2) was published by the Open Group X/Open 1992. However since that time subsequent versions have been developed by Microsoft which re-named the protocol “Common Internet File System” (CIFS).

The history of SMB and CIFS is further discussed in: the section called *History* in Chapter 6

Overview

The protocols considered here are mainly proprietary and documentation is often poor and hard to find. A high level view is presented here that attempts to describe

how the protocols relate to each other.

The original NetBIOS protocol was developed to become the NetBIOS Frames Protocol (NFB) often referred to as NetBEUI or just NetBIOS. This protocol is still used today, but is not popular because it is not routable or scalable. NBF or NetBEUI provides a datagram delivery and session service that can be used for a variety of network applications.

The above protocol is often encapsulated in other (routable) protocols such as IPX/SPX (which Microsoft refers to as NBIPX) or TCP/IP (which Microsoft refers to as NBT). The use of NetBIOS over TCP/IP is still one of the most popular network protocol configurations.

Although NBF (either in encapsulated form or "on the wire") can be used for a variety of applications it is often used as a foundation for the Server Message Block (SMB) protocol. One of the most widely used network configurations is SMB running over NetBIOS over TCP/IP.

SMB has been developed to become the Common Internet File System (CIFS). Recently CIFS has been implemented directly on TCP/IP without requiring the NetBIOS over TCP/IP layer.

The relationship between the various protocols with respect to the OSI model is illustrated in: Appendix A

Implementation

NetBIOS is often described as a "Session Layer" protocol and a variety of transport systems have been used in different implementations. Some of these implementations are described in Chapter 5 . The protocols used to encapsulate NetBIOS are generally well understood and well documented; what is often not well understood are implementations of NetBIOS "on the wire" in a "raw" un-encapsulated form.

Two implementations of NetBIOS "on the wire" are considered here: The original NetBIOS in IBM PC Networks (See the section called *Comparison of NetBIOS protocols in IBM PC Network* in Appendix B) and NetBIOS Frames Protocol on 802.2 networks. Although the IBM PC Network version was developed first, the current NetBIOS Frames Protocol on 802.2 networks is emphasized in this document as being the more relevant.

It should be noted that the frames in NetBIOS in IBM PC Networks are more complex and seem less consistent than frames in the NetBIOS Frames Protocol on 802.2 networks. The IBM PC Networks implementation separates in to the protocols mentioned above, where as all the frames in NetBIOS Frames Protocol on 802.2 networks are more consistent in their format.

Terminology

Because of the history of the protocols being discussed here (See the section called *History*) and lack of standards, there is often confusion in the use of some of the terms; it is not uncommon to hear statements of the form "NetBIOS is not a protocol"

or "NetBEUI is a protocol".

NetBIOS is not a protocol

As described in the history above, NetBIOS was designed as an interface. NetBIOS was designed to be an extension to the BIOS (Basic Input/Output System) of PCs to provide networking services. At the risk of being pedantic, NetBIOS was designed as an application programming interface (API). It is interesting (and the source of some confusion) that it was the API which was the standard.

NetBIOS is a protocol

The term "protocol" is often used as a shorthand reference to a suite of protocols (a well-known example is the use of the term "TCP/IP protocol" to refer to a collection of protocols). The informal use of the term "protocol" is well-understood and accepted practice. It has become standard practice to use the term "NetBIOS protocol" to refer to the original set of protocols in use with the NetBIOS API and the protocols which followed. The current official term used by IBM is "NetBIOS Frames Protocol" (NBF) and it is not unreasonable to shorten this to "NetBIOS".

NetBEUI is not a protocol

If NetBIOS is not a protocol, but is an API, then an "Extended User Interface" to this API is also not a protocol. As mentioned above, and described in the history, when IBM developed Token Ring it was continuity of the API to ensure applications would continue to function which was important. The NetBIOS API was preserved and extended in the NetBIOS Extended user Interface, NetBEUI.

NetBEUI is a protocol

With the development of NetBEUI, a set of protocols was developed, now known as the NetBIOS Frames Protocol. Since the NetBIOS Frames Protocol was used with the NetBEUI API it became accepted practice to refer to these protocols as the "NetBEUI protocol". It is still common to find documentation which refers to the "NetBEUI protocol".

Notes

1. http://www.samba.org

Chapter 2. NetBIOS, NetBEUI, NetBIOS Frames Protocol

Overview

The use of NetBIOS, (otherwise known as NetBEUI, NetBIOS Frames Protocol, NBF) is not a popular choice. Many networks use some form of encapsulation with the most popular being TCP/IP. It is important to understand that the various encapsulation implementations are designed to emulate the use of NetBIOS "on the wire"; the goal is to allow higher level protocols (such as SMB or CIFS) or applications to make use of the NetBIOS protocol irrespective of whether it is running "on the wire" or encapsulated. Thus in order to fully understand implementations that use encapsulation, it is necessary to understand "NetBIOS on the wire".

It is not clear whether there is only one protocol or several protocols involved in NetBIOS networking. The original implementation for the PC Network certainly seemed to have the following protocols: Name Management protocol (NMP), Diagnostic and Monitoring Protocol (DMP), User Datagram Protocol (UDP) and Session Management Protocol (SMP). The distinction is less clear with NetBIOS on Token-Ring and other Implementations; the official IBM documentation [IBM LAN Technical Reference IEEE 802.2 and NetBIOS Application Program Interfaces] simply describes a collection of 22 frame formats, see Table 2-1 . Given that at least network layer and session layer functions are involved, the various packets used will be discussed in terms of the original protocols for convenience, even if the distinctions are somewhat arbitrary.

Table 2-1. NBF Frames

Frame name	Command code
ADD_NAME_QUERY	0x01
ADD_GROUP_NAME	0x00
ADD_NAME_RESPONSE	0x0D
NAME_IN_CONFLICT	0x02
NAME_QUERY	0x0A
NAME_RECOGNISED	0x0E
DATAGRAM	0x08
DATAGRAM_BROADCAST	0x09
STATUS_QUERY	0x03
STATUS_RESPONSE	0x0F
TERMINATE_TRACE	0x07
TERMINATE_TRACE local and remote	0x13
SESSION_ALIVE	0x1F
SESSION_CONFIRM	0x17
SESSION_END	0x18
SESSION_INITIALIZE	0x19

Frame name	Command code
DATA_ACK	0x14
DATA_FIRST_MIDDLE	0x15
DATA_ONLY_LAST	0x16
NO_RECEIVE	0x1A
RECEIVE_OUTSTANDING	0x1B
RECEIVE_CONTINUE	0x1C

NetBIOS systems communicate in one of two manners; the protocols are often described as Session layer protocols because most of the communications occur over sessions established between two nodes on the network. The other form of communication does not involve sessions and uses a simple datagram transmission mechanism for simple communications between systems on a network. Non-session frames are used for functions such as name management or other functions that simply require a datagram delivery service. In general when one system needs to communicate with another it will contact that system and a session will be established between the two nodes; the session will be maintained as long as either node needs to communicate until one of the nodes closes the session.

Sessions are established using an exchange of packets. The station wishing to establish the session sends an Open request that should be responded to with an Open acknowledgment. The station initiating the session will then send a Session request which will elicit either a Session accept or Session reject.

Data is transmitted using an established session through the sending system sending data packets which are responded to with either acknowledgment packets (ACK) or negative acknowledgment packets (NACK) prompting re-transmission.

Sessions are closed by the system that received data sending a close request that should be responded to by the system that initiated the session sending a close response. This is followed by the system that received data sending a packet indicating that the session is closed.

Obviously in order to communicate, systems need an address scheme and this is described in the section called *Addressing - NetBIOS names*.

Addressing - NetBIOS names

To communicate, each node (computer, printer, router etc) needs to be uniquely identified on a network. Within the TCP/IP suite of protocols, under the IPv4 address scheme, a 32 bit address identifies each node and the network it is connected to. In IPX/SPX networks, a 48 bit address identifies a node on a network and a 32 bit address identifies each network. In NetBIOS networks names are used by each node.

NetBIOS names are 16 bytes (128 bits) long (padded if necessary) and there are very few restraints on the byte values that can be used. Non-alphanumeric characters can be used in NetBIOS names, although some implementations may not support this and applications using NetBIOS may impose other constraints.

Often an implementation will make use of the 16th byte, for example to indicate a particular service; thus the 16th byte may be used in a way which is broadly analogous to port numbers in TCP/IP.

It is worth noting that SMB / CIFS names map directly on to NetBIOS names and in this case the 16th byte is particularly important for identifying various services.

Microsoft has produced a Knowledge Base Article that lists the NetBIOS suffixes (i.e. the sixteenth byte) that Microsoft uses or supports.

The Knowledge Base Article is Q163409 and can be found at http://support.microsoft.com/default.aspx?scid=kb;en-us;Q163409[1]

Some examples of the 16th byte in Unique names are given below:

Table 2-2. Example Unique names

16th byte (in hex)	Description
00	Workstation service
01	Messenger service
20	File server service
2B	Lotus Notes Server Service

NetBIOS names represent a flat name space; names are non-hierarchical with no provision for subdivision. Because there is no provision for identifying networks with the NetBIOS name scheme, protocols using this name scheme can not be routed.

A NetBIOS name is often associated with one end of a session between two nodes.

A station on the network can be known by several names (aliases) originally up to 12 (See BYTE Magazine November 1989 "Two tin cans and some string" Part 2 *Bibliography*) or 17 names (See BYTE Magazine January 1989 "Understanding NetBIOS" *Bibliography*) . Modern implementations allow very many names to be used. Names can be unique names reserved for the station's exclusive use or group names shared with other stations.

Group Names

Group Names are NetBIOS names that several stations can use. Each Group Name must be unique and in many situations must be distinct from any unique (node) names. These names allow stations to be grouped together to facilitate management and browsing of the network. Stations can send broadcast messages to all stations that share a particular group name.

Within SMB / CIFS environments, collections of systems such as workgroups or domains map on to NetBIOS Group names.

Name Resolution

One name is the permanent node name, which is the physical adapter card's name; this is usually derived from the Media Access Control (MAC) address of the card i.e. the hardware address and consists of 10 bytes of zeros followed by the 6 bytes of the MAC address. This special permanent node name is often called "NETBIOS_NAME_1". It is because one of the names incorporates the physical MAC address (and because there is only one network) that there is often no name

resolution protocol (analogous to the Address Resolution Protocol ARP in TCP/IP networks).

When NetBIOS is encapsulated within other protocols such as IPX/SPX or TCP/IP there is a mechanism for mapping NetBIOS names to the address schemes of the encapsulating protocol. See Chapter 5 Encapsulation.

Name Management Protocol

The Name Management Protocol (NMP) allows systems to create unique symbolic names that are visible on the network. NMP has some similarities with the AppleTalk Name Binding Protocol: The Name Management Protocol broadcasts a system's intention to use a new name and if no other system objects, the name is registered. NetBIOS broadcasts a name claim packet several times and if no other station contests the name claim the name is added to the local name table. Typically the packets are sent at half second intervals six times, although in principal these parameters can be tuned. Often a node will require three seconds to check each name it is using.

The original Name Management Protocol is described in Appendix B in the section the section called *Name Management Frames in IBM PC Networks* in Appendix B .

In the NetBIOS Frames Protocol on 802.2 networks there are four non-session level Name Management Frames. As described in the section called *Addressing - NetBIOS names* there are two kinds of names: unique names and group names.

- The "ADD NAME QUERY" frame (0x01) is used by a node to verify that the name it wishes to add is unique within the network.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x01 identifying it as an "ADD NAME QUERY" frame.

  Five reserved octets are followed by a two byte response correlator, transmitted byte reversed, created by the sender and used to correlate any responses to the query. The next sixteen octets, used for the destination name in other frames, are reserved in this case. The following sixteen octets for the source name are used to identify the name to be added.

- The "ADD GROUP NAME" frame (0x00) is used by a node to verify that the group name does not exist as a unique name within the network.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x00 identifying it as an "ADD GROUP NAME QUERY" frame.

  Five reserved octets are followed by a two byte response correlator, transmitted byte reversed, created by the sender and used to correlate any responses to the query. The next sixteen octets, used for the destination name in other frames, are reserved in this case. The following sixteen octets for the source name are used to identify the group name to be added.

The "ADD NAME RESPONSE" frame (0x0D) is used in response to one of the above query frames to inform the node wishing to add the name that the name is already in use. The "ADD NAME RESPONSE" frame (0x0D) is used in response to one of the above query frames to inform the node wishing to add the name that the name is already in use.

The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x0D identifying it as an "ADD NAME RESPONSE" frame.

The next octet, the "DATA1" octet is set to 1 or 0; 0 = "add name not in process", 1 = "add name in process". The next two bytes, known as "DATA2", constitutes a define word set to 0 or 1; 0 = "unique name, 1 = "group name"; this is transmitted byte reversed. The next two bytes constitute a correlator, transmitted byte reversed, used to correlate the response with the original query. Two reserved octets are followed by sixteen octets holding the name to be added and a further sixteen octets which again hold the name to be added.

- The "NAME IN CONFLICT" frame (0x02) is used to indicate a problem with names on the network; it is sent if more than one adapter has the same unique name registered or a name is registered as both a unique name and a group name.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x02 identifying it as an "NAME IN CONFLICT" frame.

  Seven reserved octets are followed by sixteen octets representing the name in conflict. The final sixteen octets represent the special "NAME NUMBER 1" of the node sending the frame.

  The "ADD NAME QUERY" frame (0x01) is used by a node to verify that the name it wishes to add is unique within the network.

Table 2-3. Name Management Frames (Octets in order transmitted.)

		Management frame	Management frame	Management frame	Management frame
Field Name	**Length**	**ADD GROUP NAME QUERY**	**ADD NAME QUERY**	**ADD NAME RESPONSE**	**NAME IN CONFLICT**
Length	2	0x2C	0x2C	0x2C	0x2C
		0x00	0x00	0x00	0x00
Deliminator	2	0xFF	0xFF	0xFF	0xFF
		0xEF	0xEF	0xEF	0xEF
Command	1	*0x00*	*0x01*	*0x0D*	*0x02*

		Management frame	Management frame	Management frame	Management frame
Field Name	**Length**	**ADD GROUP NAME QUERY**	**ADD NAME QUERY**	**ADD NAME RESPONSE**	**NAME IN CONFLICT**
Data 1	1	Reserved	Reserved	0 or 1	Reserved
Data 2	2	Reserved	Reserved	0 or 1	Reserved
		Reserved	Reserved	0	Reserved
XMIT Cor	2	Reserved	Reserved	nn	Reserved
		Reserved	Reserved	nn	Reserved
RSP Cor	2	nn	nn	Reserved	Reserved
		nn	nn	Reserved	Reserved
Destination Name	16	Reserved	Reserved	Name to be added	Name in conflict
Source Name	16	Group name to add	Name to add	Name to be added	NAME NUMBER 1

In the NetBIOS Frames Protocol on 802.2 networks there are two frames used for managing names in session establishment. Although not part of name management, these frames are included here for convenience.

- The "NAME QUERY" frame (0x0A) is used to find a name on the network or to request a remote node to establish a session.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x0A identifying it as an "NAME QUERY" frame.

  Following the "DATA1" field which is reserved, the two octets of the "DATA2" field are set to "ttss" where "tt" indicates the type of name being called, 00 for a unique name and 01 for a group name; "ss" is used to indicate the session number. The "DATA2" field is transmitted byte reversed. Two reserved octets are followed by a two byte response correlator, transmitted byte reversed. Sixteen octets identify the name being called. The final sixteen octets identify the name of the node making the call.

- The "NAME RECOGNISED" frame (0x0E) is used in response to a NAME QUERY frame to indicate that a session can be established with the name or to provide the location of the name.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x0E identifying it as an "NAME RECOGNISED" frame.

Following the "DATA1" field which is reserved, the two bytes of the "DATA2" field are set to "ttss" where "tt" is set to 0x00 to indicate a unique recognized name or 0x01 to indicate a unique recognized group name. the type of name being called, 00 for a unique name and 01 for a group "ss" is used to indicate the "state" of the name: 0x00 is used when the station is not listening for this name, 0xFF is used when the station is listening for this name, but can not establish a session, 0x01 to 0xFE are used as a number which will identify this session. The "DATA2" field is transmitted byte reversed.

A two byte transmit correlator is used to correlate the response with the NAME QUERY frame. This is followed by a two byte response correlator used with SESSION INITIALIZE frames; these fields are transmitted byte reversed. Sixteen octets identify the name of the node making the NAME QUERY call. The final sixteen octets identify the name being queried.

Table 2-4. Frames for managing names in session establishment (Octets in order transmitted).

		Management frame	Management frame
Field Name	**Length**	**NAME QUERY**	**NAME RECOGNISED**
Length	2	0x2C	0x2C
		0x00	0x00
Deliminator	2	0xFF	0xFF
		0xEF	0xEF
Command	1	*0x0A*	*0x0E*
Data 1	1	Reserved	Reserved
Data 2	2	X ss	X ss
		X tt	X tt
XMIT Cor	2	Reserved	nn
		Reserved	nn
RSP Cor	2	nn	nn
		nn	nn
Destination Name	16	Name of receiver	Name of receiver
Source Name	16	Name of sender	Name of sender

User Datagram Protocol

The NetBIOS User Datagram Protocol (UDP) provides packet transmission from source to destination. UDP is an unreliable datagram delivery protocol.

NetBIOS User Datagram Protocol is comparable with the Datagram Delivery Protocol (DDP) in AppleTalk, or IP in the TCP/IP suite of protocols, or IPX in Novell's IPX/SPX protocol suite. It is important to note that the NetBIOS User Datagram Protocol differs from datagram protocols in other protocol suites; the NetBIOS User Datagram Protocol is designed specifically to provide a datagram delivery service and not necessarily to provide a foundation for higher level protocols. Where as in other protocol suites the datagram protocol supports transport and session layer protocols running over the datagram protocol, here there is a separate Session Management Protocol which does not make use of the NetBIOS User Datagram Protocol. The relationship is illustrated in the Appendix A .

UDP packets are sent between Named systems (see the section called *Addressing - NetBIOS names* above) or are broadcast from one Named system to all Names on the network.

The original User Datagram Protocol is described in Appendix B in the section the section called *Datagram Packet in IBM PC Network* in Appendix B .

NetBIOS Non-Session Frames on 802.2 networks

- The "DATAGRAM" frame (0x08) is used to send a datagram to a name.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x08 identifying it as an "DATAGRAM" frame.

  Seven reserved octets are followed by sixteen octets used to identify the destination name of the datagram. The following sixteen octets identify the source name sending the datagram. A variable number of octets contain the data or payload of the datagram.

- The "DATAGRAM BROADCAST" frame (0x09) is used to broadcast a datagram to all names on the network.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x09 identifying it as an "DATAGRAM" frame.

  Seven reserved octets are followed by a further sixteen octets which are also reserved rather than identifying the destination name as in the case of "DATAGRAM" frames. The following sixteen octets identify the source name sending the datagram. A variable number of octets contain the data or payload of the datagram.

Table 2-5. Datagram frames (Octets in order transmitted.)

		Data frame	Data frame
Field Name	**Length**	**DATAGRAM**	**DATAGRAM BROADCAST**
Length	2	0x2C	0x2C
		0x00	0x00
Deliminator	2	0xFF	0xFF
		0xEF	0xEF
Command	1	*0x08*	*0x09*
Data 1	1	Reserved	Reserved
Data 2	2	Reserved	Reserved
		Reserved	Reserved
XMIT Cor	2	Reserved	Reserved
		Reserved	Reserved
RSP Cor	2	Reserved	Reserved
		Reserved	Reserved
Destination Name	16	Name of receiver	Reserved
Source Name	16	Name of sender	Name of sender
Optional		Datagram	Datagram

NetBIOS Diagnostic and Monitoring Protocol

The NetBIOS Diagnostic and Monitoring Protocol (DMP) provides the facilities to obtain status information about local and remote nodes on the network. This protocol is broadly comparable with the basic functionality provided by the Simple Network Monitoring Protocol (SNMP) within the TCP/IP suite of protocols.

The NetBIOS Diagnostic and Monitoring Protocol (DMP) provides the facilities to obtain information including:

- Number of "Frame Reject" (FRMR) frames received.
- Number of "Frame Reject" (FRMR) frames transmitted.
- Number of I-format "Logical link control Protocol Data Units" (LPDUs) received in error.
- Number of aborted transmissions.
- Etc.

It is worth noting that this facility is part of the protocol and not an advantage of a given Operating System that happens to be using the protocol.

NetBIOS Diagnostic and Monitoring frames on 802.2 networks

- The "STATUS QUERY" frame (0x03) is used to request remote adapter status information.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x03 identifying it as an "STATUS QUERY" frame.

  The "DATA1" octet is used to indicate the status of the request, 0x0 indicates a 1.x or 2.0 type request, 0x01 indicates a NetBIOS 2.1 request and values greater than 1 indicate a NetBIOS 2.1 request. The two bytes of the "DATA2" field are used to indicate the length of the user's status buffer. The "DATA2" field is transmitted byte reversed. Two reserved octets are followed by a two octet response correlator, transmitted byte reversed. Sixteen octets identifying the name of the receiver are followed by a further sixteen octets indicating the sending node's NAME NUMBER 1.

- The "STATUS RESPONSE" frame (0x0F) is used in response to request a status request frame.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x0F identifying it as an "STATUS RESPONSE" frame.

  The "DATA1" octet is used to indicate the status of the response, 0x0 indicates a 1.x or 2.0 type response, and values greater than 0x0 indicate a NetBIOS 2.1 response. The two octets of the "DATA2" field are regarded as a 16 bit string; the first bit x is set to 1 if the length of the status data exceeds the frame size; the second bit y is set to 1 if the length exceeds the size of the user's buffer; the remaining 14 bits indicate the length of the status data sent. The "DATA2" field is transmitted byte reversed. A two octet transmit correlator, transmitted byte reversed, is followed by two reserved octets. Sixteen octets indicate the receiving node's NAME NUMBER 1 and are followed by a further sixteen octets indicating the sending node's name.

- The "TERMINATE TRACE" frame (0x07) is used to end a trace at a remote node.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x07 identifying it as an "TERMINATE TRACE" frame.

  All the remaining 39 octets are reserved.

- The "TERMINATE TRACE" frame (0x13) is used to end a local trace and a trace at a remote node.

The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x13 identifying it as an "TERMINATE TRACE" frame.

All the remaining 39 octets are reserved.

Table 2-6. Diagnostic and Monitoring frames (Octets in order transmitted.)

		Special frame	Special frame	Special frame	Special frame
Field Name	**Length**	**STATUS QUERY**	**STATUS RESPONSE**	**TERMINATE TRACE**	**Terminate local & remote trace**
Length	2	0x2C	0x2C	0x2C	0x2C
		0x00	0x00	0x00	0x00
Deliminator	2	0xFF	0xFF	0xFF	0xFF
		0xEF	0xEF	0xEF	0xEF
Command	1	*0x03*	*0x0F*	*0x07*	*0x13*
Data 1	1	nn	nn	Reserved	Reserved
Data 2	2	Length of status buf	bbbbbbbb	Reserved	Reserved
		Length of status buf	xybbbbbb	Reserved	Reserved
XMIT Cor	2	Reserved	nnnn	Reserved	Reserved
		Reserved	nnnn	Reserved	Reserved
RSP Cor	2	nnnn	Reserved	Reserved	Reserved
		nnnn	Reserved	Reserved	Reserved
Destination Name	16	Name of receiver	Receiver's NAME NUMBER 1	Reserved	Reserved
Source Name	16	Sending node NAME NUMBER 1	Name of sender	Reserved	Reserved

NetBIOS Session Management Protocol

The NetBIOS Session Management Protocol (SMP) manages sessions between

Named processes on the network. NetBIOS Sessions support full duplex operation. One Named process calls another Name on the network which answers. The Session continues until one or both systems hang up.

The original NetBIOS Session Management Protocol is described in Appendix B Appendix: NetBIOS protocols in IBM PC Network in the section called *NetBIOS Session Management Protocol in IBM PC Networks* in Appendix B NetBIOS Session Management Protocol in IBM PC Networks.

NetBIOS Session Frames - Name Query - on 802.2 networks

In the NetBIOS Frames Protocol on 802.2 networks there are two frames used for managing names in session establishment. Details of these frames are given in the section called *Name Management Protocol* "Name Management Frames in NetBIOS on 802.2 networks"

Table 2-7. Frames for managing names in session establishment (Octets in order transmitted.)

		Management frame	**Management frame**
Field Name	**Length**	**NAME QUERY**	**NAME RECOGNISED**
Length	2	0x2C	0x2C
		0x00	0x00
Deliminator	2	0xFF	0xFF
		0xEF	0xEF
Command	1	*0x0A*	*0x0E*
Data 1	1	Reserved	Reserved
Data 2	2	X ss	X ss
		X tt	X tt
XMIT Cor	2	Reserved	nn
		Reserved	nn
RSP Cor	2	nn	nn
		nn	nn
Destination Name	16	Name of receiver	Name of receiver
Source Name	16	Name of sender	Name of sender

NetBIOS Session Frames - Establishment and Termination - on 802.2

networks

- The "SESSION ALIVE" frame (0x1F) is send as the result of an inactivity timer expiring.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x1F identifying it as an "SESSION ALIVE" frame.

  All the remaining 39 octets are reserved.

- The "SESSION CONFIRM" frame (0x17) is used to request remote adapter status information.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. these fields are followed by the one octet command frame which has a value of 0x17 identifying it as an "SESSION CONFIRM" frame.

  The "DATA1" octet is an 8 bit binary string; the first bit "y" is set to 0 for versions of NetBIOS prior to version 2.20 and to 1 for higher versions. After 6 bits always set to 0, the last bit "x" is set to 0 for NetBIOS version 1.xx and set to 1 for version 2.00 or above (In practice this will now always be set to 1). The two bytes of the "DATA2" field are used to indicate the length of the user's receive buffer. The "DATA2" field is transmitted byte reversed.

  Two octets used for a transmission correlator are followed by a two octet response correlator; these fields are transmitted byte reversed. A single octet is used for the remote session number and is followed by an octet for the local session number.

- The "SESSION END" frame (0x18) is used to request remote adapter status information.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x18 identifying it as an "SESSION END" frame.

  The "DATA1" octet is reserved. The two bytes of the "DATA2" field are used to indicate the reason for termination. 0x00 indicates a normal session end, 0x01 indicates an abnormal end. The "DATA2" field is transmitted byte reversed. Four reserved octets are followed by a single octet used for the remote session number. The final octet is for the local session number.

- The "SESSION INITIALIZE" frame (0x19) is used to request remote adapter status information.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x19 identifying it as an "SESSION INITIALIZE" frame.

  The "DATA1" octet is an 8 bit binary string; the first bit "z" is set to 0 for versions of NetBIOS prior to version 2.20 and to 1 for higher versions (In practice this will now

always be set to 1). Three reserved bits "rrr", always set to 0 are followed by 3 bits "xxx" used to indicate the largest frame value as seen by the MAC layer; the last bit "z" is set to 0 for NetBIOS version 1.xx and set to 1 for version 2.00 or above. The two octets of the "DATA2" field are used to indicate the length of the user's receive buffer. The "DATA2" field is transmitted byte reversed. A single octet is used for the remote session number and is followed by an octet for the local session number.

Table 2-8. Session Establishment and Termination frames (Octets in order transmitted.)

		Session frame	Session frame	Session frame	Session frame
Field Name	**Length**	**SESSION ALIVE**	**SESSION CONFIRM**	**Session End**	**SESSION INITIALIZE**
Length	2	0x0E	0x0E	0x0E	0x0E
		0x00	0x00	0x00	0x00
Deliminator	2	0xFF	0xFF	0xFF	0xFF
		0xEF	0xEF	0xEF	0xEF
Command	1	*0x1F*	*0x17*	*0x18*	*0x19*
Data1	1	Reserved	B yrrrrrrx	Reserved	zrrrxxxy
Data2	2	Reserved	Max data rec size	Termination indicator	Max data receive size
		Reserved	Max data rec size	Termination indicator	Max data receive size
XMIT Cor	2	Reserved	nnnn	Reserved	nnnn
		Reserved	nnnn	Reserved	nnnn
RSP Cor	2	Reserved	nnnn Sess init xmit	Reserved	nnnn
		Reserved	Remote session num	Remote session num	Remote session num
Destination Num	1	Reserved	Remote session num	Remote session num	Remote session num
Source Num	1	Reserved	Local session num	Local session num	Local session num

NetBIOS Session Frames - Data Transfer - on 802.2 networks

- The "DATA ACK" frame (0x14) is sent in response to a DATA ONLY LAST frame (see below).

The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x14 identifying it as an "DATA ACK" frame.

Three reserved octets are followed by a two octet transmit correlator then a two octet receive correlator; these fields are transmitted byte reversed. A single octet is used for the remote session number and is followed by an octet for the local session number.

- The "DATA FIRST MIDDLE" frame (0x15) is used to transmit user messages across a session.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x15 identifying it as an "DATA FIRST MIDDLE" frame.

  The "DATA1" octet is an 8 bit binary string; the first four bits are reserved; the fifth bit "x" is set to 1 if an acknowledgment is included; this is followed by a reserved bit; the seventh bit "y" is set to 0 for versions of NetBIOS prior to version 2.20 and set to 1 for later versions (In practice this will now always be set to 1); the last bit "z" is set to 0 if a RECEIVE CONTINUE was not requested, otherwise it is set to 1. The next two octets are for DATA2 and is a re-synchronization indicator set to 0001 if it is the first DATA FIRST MIDDLE frame. The "DATA2" field is transmitted byte reversed.

  This is followed by a transmit correlator then a two octet receive correlator. A single octet is used for the remote session number and is followed by an octet for the local session number. Finally the user data follows.

- The "DATA ONLY LAST" frame (0x16) is used to transmit user messages across a session and is either the only frame or the last.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x16 identifying it as an "DATA ONLY LAST" frame.

  The "DATA1" octet is an 8 bit binary string; the first four bits are reserved; the fifth bit "x" is set to 1 if an acknowledgment is included; this is followed by the sixth "y" bit which indicates that an "acknowledge with data allowed" is permitted; the seventh bit "z" is a "no.ack" indicator and the final bit is reserved. The next two octets are for DATA2 and is a re-synchronization indicator set to 0001 if this frame is send following receipt of a RECEIVE OUTSTANDING. The "DATA2" field is transmitted byte reversed. This is followed by a transmit correlator then a two octet receive correlator; these fields are transmitted byte reversed. A single octet is used for the remote session number and is followed by an octet for the local session number. Finally the user data follows.

- The "NO RECEIVE" frame (0x1A) is transmitted in response to a "DATA ONLY LAST" frame or a "DATA FIRST MIDDLE" frame.

The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x1A identifying it as an "NO RECEIVE" frame.

The "DATA1" octet is an 8 bit binary string; the first six bits are reserved; the seventh bit "x" is set to 0 for versions of NetBIOS prior to 2.20, otherwise it is set to 1 (In practice this will now always be set to 1); the eighth bit is reserved. The next two bytes are for DATA2 and gives the number of data bytes accepted. The "DATA2" field is transmitted byte reversed. Four reserved octets are followed by a single octet used for the remote session number; this is followed by an octet for the local session number.

- The "RECEIVE OUTSTANDING" frame (0x1B) is transmitted in response to a "NO RECEIVE" frame.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x1C identifying it as an "RECEIVE OUTSTANDING" frame.

  The "DATA1" octet is reserved. The next two octets are for DATA2 and gives the number of data bytes accepted. The "DATA2" field is transmitted byte reversed. Four reserved octets are followed by a single octet used for the remote session number; this is followed by an octet for the local session number.

- The "RECEIVE CONTINUE" frame (0x1C) is transmitted in response to a "DATA ONLY LAST" frame which had the RECEIVE CONTINUE bit set.

  The frame begins with a two byte length field with a value of 0x002C followed by the two byte frame deliminator field 0xEFFF; these fields are transmitted byte reversed. These fields are followed by the one octet command frame which has a value of 0x1C identifying it as an "RECEIVE CONTINUE" frame.

  Three reserved octets are followed by a two octet transmit correlator, transmitted byte reversed, which is followed by two more reserved octets. A single octet is used for the remote session number and is followed by an octet for the local session number.

Table 2-9. Session Data Transfer frames (Octets in order transmitted.)

		Data frame	Data frame	Data frame	Data frame	Data frame	Data frame
Field Name	**Length**	**DATA ACK**	**DATA FIRST MIDDLE**	**DATA ONLY LAST**	**NO RECEIVE**	**RECEIVE OUTSTANDING**	**RECEIVE CONTINUE**
Length	2	0x0E	0x0E	0x0E	0x0E	0x0E	0x0E
		0x00	0x00	0x00	0x00	0x00	0x00

		Data frame	Data frame	Data frame	Data frame	Data frame	Data frame
Field Name	**Length**	**DATA ACK**	**DATA FIRST MIDDLE**	**DATA ONLY LAST**	**NO RE-CEIVE**	**RECEIVE OUT-STANDING**	**RECEIVE CON-TINUE**
Delimiter	2	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
		0xEF	0xEF	0xEF	0xEF	0xEF	0xEF
Command	1	*0x14*	*0x15*	*0x16*	*0x1A*	*0x1B*	*0x1C*
Data1	1	Reserved	Brrrxryz	Brrrxryz	Brrrrrrxr	Reserved	Reserved
Data2	2	Reserved	Re-synch indicator	Re-synch indicator	Number of data bytes accepted	Number of data bytes accepted	Reserved
		Reserved	Re-synch indicator	Re-synch indicator	Number of data bytes accepted	Number of data bytes accepted	Reserved
XMIT Cor	2	nnnn	nnnn	nnnn	Reserved	Reserved	nnnn
		nnnn	nnnn	nnnn	Reserved	Reserved	nnnn
RSP Cor	2	Reserved	nnnn	nnnn	Reserved	Reserved	Reserved
		Reserved	nnnn	nnnn	Reserved	Reserved	Reserved
Dest Num	1	Remote session num	Remote session num	Remote session num	Remote session num	Remote session num	Remote session num
Source Num	1	Local session num	Local session num	Local session num	Local session num	Local session num	Local session num
Optional data			USER DATA Message from send	USER DATA Message from send			

Notes

1. http://support.microsoft.com/default.aspx?scid=kb;en-us;Q163409

Chapter 3. Supporting Technology, 802.2, Ethernet and Token Ring

Although NetBIOS is often encapsulated, it can be implemented "on the wire". This chapter looks at the implementation of NetBIOS on two popular networking technologies, Ethernet and Token Ring as well as the 802.2 Logical Link Control layer used with these technologies. This documentation looks at the technologies in relation to NetBIOS rather than attempting to provide a full description of the protocols; there are many excellent books on 802.2, Ethernet and Token Ring that describe those subjects in detail.

IEEE 802.2 Logical Link Control

In the OSI Reference Model, the Datalink layer sits above the Physical layer and below the Network layer. When considering IEEE LAN technology the situation is a little more complex. There are a number of LAN systems such as Token Ring and Ethernet and the physical characteristics of these are defined in the Physical Layer of the OSI model. Characteristics such as the frame format for systems such as Token Ring and Ethernet are defined in the Datalink layer in standards such as 802.3, 802.5 etc. A common interface was required between these standards and the protocols in layer 3 and this is implemented in 802.2. A full description of IEEE 802.2 Logical Link Control is beyond the scope of this document; a brief overview is given below.

IEEE 802.2 Logical Link Control frames often provide the data link layer support for implementations of NetBIOS. This is the case when NetBIOS frames are being carried "on the wire" rather than encapsulated in another protocol. The relationship is illustrated in the Appendix A Open Systems Interconnection (OSI) Reference Model

802.2 supports both connection-oriented and connection-less oriented communications. The Logical Link Control offers services to the Network layer through Service Access Points (SAPs). The SAP is used to identify the process at the Network layer.

IEEE 802.2 frames have the following form:

DSAP 1 byte

Destination Service Access Point

SSAP 1 byte

source Service Access Point

Control 1 or 2 bytes

field length depends on the service

Information

This variable length field carries any data

Some examples of DSAP and SSAP values are given below.

For IPX (the network protocol traditionally used with NetWare networks), DSAP = 0xE0 (224), SSAP = 0xE0 and Control is 1 byte 0x03 which denotes the 802.2 unnumbered format.

For SNAP (Sub-Network Access Protocol), DSAP = 0xAA (170), SSAP = 0xAA

For NetBIOS, DSAP = 0xF0 (240) , SSAP = 0xF0

Some IEEE 802.2 Numbers of interest can be found at "The Internet Assigned Numbers Authority" web site, "Protocol Numbers and Assignment Services" in "IEEE 802 Numbers":

http://www.iana.org/assignments/ieee-802-numbers[1]

In 1985 IBM implemented NetBIOS over Token Ring and established the way in which NetBIOS frames would map to 802.2 frames.

When NetBIOS is implemented over Token Ring, the NetBIOS frames are mapped directly on to the 802.2 frames; the NetBIOS frame is contained in the information field of the 802.2 frame:

- DSAP 1 byte Destination Service Access Point 0xF0
- SSAP 1 byte source Service Access Point 0xF0
- Control 1 or 2 bytes field length depends on the service
- Information:
  - NetBIOS header
  - Optional data

The above scheme is general to implementations of NetBIOS over 802.2 where other underlying technologies are used such as Ethernet.

Token Ring

Token Ring is becoming less popular with many organizations moving to Ethernet. Token Ring is discussed here because of it's importance in the history of NetBIOS and understanding of NetBIOS.

When IBM introduced Token-Ring, an emulator for NetBIOS was produced. The NetBIOS Extended User Interface (NetBEUI) was introduced in 1985. NetBIOS was no longer implemented only on a set of propriety protocols, but also on 802.2 frames. The implementation on Token-Ring was the first implementation over 802.2 and provides a reference model. Detailed information can be found in the IBM manual: IBM LAN Technical Reference, see *Bibliography* IBM LAN Technical Reference IEEE 802.2 and NetBIOS Application Program Interfaces.

A full description of Token Ring is beyond the scope of this document; some basic information on Token Ring and its use with NetBIOS is given below.

There are two kinds of Token Ring frames: Media Access Control (MAC) frames and Non-MAC frames. MAC frames carry Token Ring management information between nodes, Non-MAC frames carry user data. The non-MAC frames contain IEEE 802.2 Logical Link Control frames which in turn can contain NetBIOS frames.

Non-MAC Frame Structure

Table 3-1. Non-MAC Token Ring Frame Structure

Token Ring frame	802.2 Frame detail	NBF frame
Start Delimiter (SDEL) 1 octet		
Access Control (AC) 1 octet		
Frame Control (FC) 1 octet		
Destination Address 6 octets		
Source Address 6 octets		
IEEE 802.2 Logical Link Control	DSAP 2 octets	
	SSAP 2 octets	
	Control 1 or 2 octets	
	Data 46-1500 octets	NetBIOS header
		Optional data
		Frame Check sequence (FCS) 4 octets
End Delimiter (EDEL) 1 octet		
Frame Status (FS) Check sequence 1 octet		

Further information

Many manuals and documents describe Token-Ring in detail including

Novell's Guide to NetWare LAN Analysis, see *Bibliography*

Ethernet

A full description of Ethernet is beyond the scope of this document; some basic information on Ethernet and its use with NetBIOS is given below.

Ethernet is widely used today and well documented. Four types of Ethernet frames have been in common use. For convenience the notation used by Novell is used to describe the four Ethernet frame types:

Ethernet_802.3

Known as Ethernet 802.3 raw, this frame type is used in NetWare networks and was the default type in NetWare v2.x and v3.x

Ethernet_II

Known as Ethernet DIX (Developed by Digital Intel Xerox)

Ethernet_802.2

IEEE Ethernet

Ethernet_SNAP

SNAP (Sub-Network Access Protocol) derived from the Ethernet 802.2 structure

Ethernet_802.3

Known as Ethernet 802.3 raw, this frame type is used in NetWare networks and was the default type in NetWare v2.x and v3.x Because of the nature of these frames they are unlikely to carry NBF frames, unless encapsulated in IPX.

Table 3-2. Ethernet_802.3 Frame Structure

Preamble 7 octets
Start frame deliminator 1 octet
Destination Address 6 octets
Source Address 6 octets
Length 2 octets
Data 46-1500 octets
Frame Check sequence 4 octets

Ethernet_802.2

NBF frames are found in Ethernet_802.2 frames more than in other Ethernet frames when NBF is implemented "on the wire" rather than encapsulated.

Ethernet_802.2 frames are also used with IPX/SPX and FTAM (File Transfer Access and Management) protocol.

Table 3-3. Ethernet_802.2 Frame Structure

Ethernet frame	802.2 Frame detail	NBF frame
Preamble 7 octets		
Start frame deliminator 1 octet		
Destination Address 6 octets		
Source Address 6 octets		
Length 2 octets		
IEEE 802.2 Logical Link Control	DSAP 2 octets	
	SSAP 2 octets	

Ethernet frame	802.2 Frame detail	NBF frame
	Control 1 or 2 octets	
	Data 46-1500 octets	NetBIOS header
		Optional data
Frame Check sequence 4 octets		

Ethernet_SNAP

Ethernet_SNAP frames are used by IPX/SPX, TCP/IP and AppleTalk Phase II.

Table 3-4. Ethernet_SNAP Frame Structure

Preamble 7 octets
Start frame deliminator 1 octet
Destination Address 6 octets
Source Address 6 octets
Length 2 octets
DSAP 2 octets value AA
SSAP 2 octets value AA
Control 1 octets
Organizational code 3 octets
Ethernet Type 2 octets
Data 46-1500 octets
Frame Check sequence 4 octets

Ethernet_II

Ethernet_II frames are used with IPX/SPX TCP/IP AppleTalk Phase I

Following the source address, is an Ethernet frame type. Information on Ethernet frame types can be found at: *http://www.iana.org/assignments/ethernet-numbers* [2] and at: *http://www.cavebear.com/CaveBear/Ethernet/type.html* [3]

For SNA (Systems Network Architecture) communications the value registered for the type is *0x80D5*. This value of 0x80D5 is also used for other systems using the IEEE 802.2 API *including NetBIOS*

Table 3-5. Ethernet_II Frame Structure

Preamble 8 octets

Destination Address 6 octets
Source Address 6 octets
Ethernet Type 2 octets
Data 46-1500 octets
Frame Check sequence 4 octets

Further information

Many manuals and documents describe Ethernet in detail including Novell's Guide to NetWare LAN Analysis, see *Bibliography*

Notes

1. http://www.iana.org/assignments/ieee-802-numbers
2. http://www.iana.org/assignments/ethernet-numbers
3. http://www.cavebear.com/CaveBear/Ethernet/type.html

Chapter 4. Encapsulation in TCP/IP

NetBIOS is often described as a "Session Layer" protocol and a variety of transport systems have been used in different implementations. Particularly because NetBIOS is a non-routable protocol, it has often been implemented using other routable protocols to provide the transport.

It has traditionally been the NetBIOS API that has been the "standard". In most implementations (certainly NetBIOS over TCP/IP and NetBIOS over IPX), encapsulation has been implemented to ensure that higher level protocols (such as SMB) can run over the encapsulated protocol in the same way as they would run over NetBIOS Frames Protocol, NBF (otherwise known as NetBEUI or NetBIOS). Thus it is important to understand the NetBIOS Frames Protocol, NBF in order to understand the various encapsulation implementations.

RFC1001 and RFC1002

The suite of protocols known as "TCP/IP" is perhaps the best know protocol suite. Currently most systems use IP version 4; the next generation of IP, IPv6 has not yet widely replaced IP version 4. These protocols are well documented in "Request for Comments" (RFCs) and there are many books available on the subject.

NetBIOS can be carried over TCP/IP (v4) networks. The relevant RFCs describing NetBIOS on a TCP and UDP foundation are:

RFC 1001

> Protocol standard for a NetBIOS service on a TCP/UDP transport: concepts and methods

RFC 1002

> Protocol standard for a NetBIOS service on a TCP/UDP transport: detailed specifications

The protocol standards described in the above RFCs are designed to preserve existing NetBIOS services, utilize existing standards and minimize new developments. The standards proposed also aimed to be robust and efficient while not necessarily requiring central management or many additional facilities to run.

Within this system NetBIOS names are aligned with the Domain Name Service (DNS). A "NetBIOS Scope" is defined as a population of computers through out which a NetBIOS name is known. Because many non-intersecting NetBIOS Scopes may exist on an internetwork, each NetBIOS scope has a "scope identifier"; this is a string that is in a DNS compatible format. This can be thought of as a pseudo sub-domain containing all the NetBIOS names in a given Scope.

NetBIOS names are strings of 16 bytes with few restrictions; NetBIOS names can and often do contain characters that are illegal in DNS names such as spaces, underscores and other non-alphanumeric characters. DNS names may only contain alphanumeric characters, hyphens and stops. An encoding scheme is used to represent the 16 byte NetBIOS names as a 32 character string to which a stop and then the scope identifier is appended to form a DNS name. Each name needs to be registered for use with an IP address.

There are two servers defined which may be implemented with NetBIOS on a TCP/UDP transport: The NetBIOS Name Server (NBNS) and the NetBIOS Datagram Distribution Server (NBDD).

The NBNS can be configured to work in a variety of ways either acting simply as a bulletin board where systems can register names, or completely managing names and addresses. Several NBNS system can be configured to work together to provide a distributed service.

Since multicasting and broadcasting are not widely implemented on internets, the NBDD service provides this function. Datagrams to be sent to individual nodes or broadcast, can be sent to the NBDD which will forward the datagram to the target node or nodes.

Systems implementing NetBIOS on a TCP/UDP transport, other than NBNS and NBDS servers, are known as "End-Nodes". Two distinct types of "End-Node" are defined: Broadcast nodes ("B" nodes) and Point-to-point nodes ("P" nodes). Broadcast nodes ("B" nodes) communicate using a combination of UDP datagrams and TCP connections. "B" nodes can function within a broadcast area which is a single MAC-bridged LAN. Point-to-point nodes communicate exclusively by directed UDP datagrams and TCP sessions. "P" nodes depend upon NBNS servers to register their name to IP address mappings and discover the names of other End-Nodes.

Two further kinds of End-Node are used with NetBIOS on a TCP/UDP transport: RFC 1001 defines Mixed mode nodes ("M" nodes) as "P" nodes with "B" node characteristics. "M" nodes use NBNS and NBDD servers, but may continue to function if these servers are temporarily unavailable. An "M" node typically performs functions as a "B" node and then as a "P" node if necessary. Hybrid nodes ("H" nodes) are not defined in RFC 1001 and have not been standardized; these are mixed nodes similar to "M" nodes but function broadly in the opposite manner to "M" nodes. "H" nodes function as a "P" node first and then as a "B" node.

NetBIOS on a TCP/UDP transport provides the standard NetBIOS services: Adapter Status Transactions, NetBIOS Session Service and NetBIOS Datagram Service.

Details of packet formats are given in RFC 1002.

The following UDP and TCP port numbers are used with NetBIOS on a TCP/UDP transport:

Table 4-1. UDP and TCP port numbers are used with NetBIOS

Service	UDP Port	TCP Port
Name Service	137	137
Session Service		139
Datagram Service	138	

There are several implementations of NetBIOS on a TCP/UDP transport. A free implementation is "SAMBA" which is available for various Unix platforms and non-Unix platforms. Further information about "SAMBA" can be obtained from the "SAMBA" Web page:

http://www.samba.org [1]

The product can be obtained from the above web site, which is also a useful source of information.

Name Resolution

NetBIOS over TCP/IP is probably the most common implementation and is often used in preference to NetBIOS "on the wire" (NetBIOS Frames Protocol otherwise known as NetBEUI or just NetBIOS) or in preference to NetBIOS over IPX/SPX. NetBIOS over TCP/IP is also probably most often used to carry the SMB / CIFS protocol.

When implementing NetBIOS over TCP/IP, Name resolution i.e. the mechanisms for converting NetBIOS names to IP addresses is critically important. This topic is sufficiently important (and so often the source of many problems) to merit special discussion.

It is important to note that Name resolution is separate from the Browser service. Name resolution is specific to NetBIOS over TCP/IP; the Browser service runs over SMB which runs over NetBIOS Frames Protocol, NetBIOS over IPX or NetBIOS over TCP/IP. The mapping of NetBIOS names to IP addresses is distinct from the service that makes lists of systems available (for example in "Network Neighborhood" or "My Network Places") which is provided by the Browser service. Of course services such as the Browser service (that runs over SMB) are unlikely to function correctly if the underlying facilities such as Name resolution are not working properly.

The Name resolution mechanisms discussed here do not necessarily conflict with the mechanisms discussed in the standards RFC 1001 and RFC 1002, but can be seen as alternative implementations with various enhancements.

In practice it seems that the main implementations of NetBIOS over TCP/IP utilize a local cache for Name resolution; name to IP address mappings that have recently been resolved are held in a local cache for a short time which can reduce the need to access the network to resolve names to IP addresses.

LMHOSTS

Many implementations of NetBIOS over TCP/IP make use of an LMHOSTS file. The LMHOSTS file is similar to the traditional hosts file; both are simple text files listing IPv4 addresses and host names. The LMHOSTS file consists of several lines each of which may have an IPv4 address followed by white space, a NetBIOS name and possibly additional parameters or comments. Most implementations support the use of the hash # character to indicate comments or additional parameters. While the basic structure described above is fairly universal, the use of additional information is implementation dependent.

In the SAMBA implementation, a NetBIOS hostname can be followed by a hash # and then two hexadecimal digits specifying the NetBIOS name type. In the Microsoft implementation, special characters can be included by enclosing the name in quotes and entering the hexadecimal code as \0xnn or \nn; the sixteenth byte can be specified in this manner but the name must be padded with spaces to ensure it is sixteen bytes long. In the Microsoft implementation several "directives" can be included in the file, beginning with the hash # character.

Microsoft has produced articles describing their use of LMHOSTS files including:

Article ID: Q101927

"The Lmhosts File for TCP/IP in Windows"

Article ID: Q102725

"LMHOSTS File Information and Predefined Keywords"

Article ID: Q104576

"Embedding Non-printable Characters in LMHOSTS Computer Names"

Article ID: Q108295

"TCP/IP Name Resolution"

Article ID: Q150800

"Domain Browsing with TCP/IP and LMHOSTS Files"

Article ID: Q180094

"How to Write an LMHOSTS File for Domain Validation and Other Name Resolution Issues"

NBNS

The NetBIOS Name Service is implemented in SAMBA as nmbd. This service can also support the browsing service (which is a separate service). The nmbd service can also be used as a WINS server or WINS proxy.

Microsoft has produced an implementation of the NetBIOS Name Service (NBNS) called Windows Internet Name Service (WINS). The use of WINS is described in the Microsoft article:

Article ID: Q119493

"NetBIOS over TCP/IP Name Resolution and WINS"

Hosts and DNS

Within Microsoft networks, NetBIOS Name resolution can also make use of the traditional hosts file and the Domain Name System (DNS). For this mechanism to work NetBIOS names need to be the same as the unqualified TPC/IP host name. The implication of this is that the NetBIOS names will also conform to the constraints of the DNS name space (i.e. names can only consist of letters, digits and the dash / hyphen character - and can not contain other special characters otherwise allowed in NetBIOS names such as the underscore character _). Microsoft describe the use of a hosts file and the DNS in the article:

Article ID: Q142309

"NetBIOS Name Resolution Using DNS and the HOSTS File"

Client Resolution

Systems can use a combination of the above services to resolve NetBIOS names to IP addresses for example sequentially searching cache, lmhosts file, nbns service, hosts files and the DNS.

Name management

The management of names can be a complex issue. To avoid problems it is important that multiple systems do not attempt to update the same name registration service.

In Microsoft NT 4.0 networks a typical arrangement will be as follows:

1. A DHCP server will allocate IP addresses to client systems when they boot.
2. Client systems are allocated NetBIOS names at installation time. The names conforms to the DNS rules for names and are the same as the unqualified DNS name. At boot time the client registers it's NetBIOS name and DHCP assigned address with a NBNS server (often a WINS server).
3. The Microsoft DNS server is configured to resolve host names by taking the unqualified DNS name and passing the enquiry to the WINS server.

Other implementations make use of Dynamic DNS (DDNS) with either a DHCP server or the clients themselves updating the DNS server. In this arrangement provided the NetBIOS names conform to the DNS rules for names and are the same as the unqualified DNS name, the need for a NBNS server (WINS) can be removed.

CIFS over TCP/IP

The latest versions of CIFS can run "natively" over TCP/IP without requiring the "NetBIOS over TCP/IP" layer. In this implementation the NetBIOS layer is removed completely.

With the introduction of Windows 2000 and Active Directory, the latest version of CIFS can use traditional fully qualified DNS names to represent nodes on the network.

Notes

1. http://www.samba.org

Chapter 5. Encapsulation in various protocols and encapsulating

Introduction

NetBIOS is often described as a "Session Layer" protocol and a variety of transport systems have been used in different implementations. Particularly because NetBIOS is a non-routable protocol, it has often been implemented using other routable protocols to provide the transport.

It has traditionally been the NetBIOS API that has been the "standard". In most implementations (certainly NetBIOS over TCP/IP and NetBIOS over IPX), encapsulation has been implemented to ensure that higher level protocols (such as SMB) can run over the encapsulated protocol in the same way as they would run over NetBIOS Frames Protocol, NBF (otherwise known as NetBEUI or NetBIOS). Thus it is important to understand the NetBIOS Frames Protocol, NBF in order to understand the various encapsulation implementations.

IPX/SPX

IPX/SPX are the protocols native to Novell NetWare. Details of these protocols can be found in: Novell's Guide to NetWare LAN Analysis, see *Bibliography*

Novell introduced an implementation of NetBIOS over IPX in 1986. The implementation uses IPX datagrams to carry the NetBIOS Frames protocol described above.

The IPX addressing scheme is compared with the native NetBIOS and other schemes in the section called *Addressing - NetBIOS names* in Chapter 2 above. In IPX/SPX networks, a 48 bit address (usually a MAC address) identifies a node on a network and a 32 bit address identifies each network. Thus IPX is a routable protocol requiring relatively little administration, which makes it a useful means of implementing NetBIOS.

IPX packets are broadly analogous to IP packets in the TCP/IP suite of protocols; IPX packets provide an unreliable datagram delivery service. The structure of the IPX Header is given below for reference:

The IPX Header

- Checksum (2 bytes)
- Length (2 bytes)
- Transport Control (1 byte)
- Packet Type (1 byte) 0 or 4 for IPX, 5 for SPX, 17 (0x11) for NCP, 20 (0x14) WAN broadcast
- Destination Node Address (6 bytes)
- Destination Network Address (4 bytes)
- Destination Socket (2 bytes)
- Source Node Address (6 bytes)
- Source Network Address (4 bytes)

- source Socket (2 bytes)

The Destination Socket indicates the service being carried over IPX, some examples and the identifier for NetBIOS are given below:

0x451

NetWare Core Protocol (NCP)

0x452

Service Advertising Protocol packet (SAP)

0x453

Routing Information Protocol packet (RIP)

0x455

NetBIOS packet

0x456

Diagnostic packet

0x457

Serialization packet

0x4000 to 0x8000

Dynamically assigned for use with file servers etc.

Microsoft Implementation of NetBIOS over IPX

Microsoft have implemented NetBIOS over the NWLink IPX/SPX compatible transport. (NWLink is a clone of Novell's IPX/SPX). The Microsoft implementation is compatible with Novell's NetBIOS over IPX. Microsoft sometimes refers to NetBIOS over IPX as NBIPX.

Table 5-1. IPX packets (Octets in order transmitted.)

Length	IPX Field	NBIPX
2	Checksum	
2	Length	
1	Transport Control	
1	Packet Type 0 or 4 for IPX, 20 (0x14) WAN broadcast	
6	Destination Node Address	
4	Destination Network Address	
2	Destination Socket	

Length	IPX Field	NBIPX
6	Source Node Address	
4	Source Network Address	
2	source Socket	
n	Data	NBIXP packet

Table 5-2. NBIPX session packets (Octets in order transmitted.)

Length	Field
1	NBIPX Connection Control flag
1	Data Stream type
2	Source connection id
2	Destination connection id
2	Send Sequence number
2	Total data length
2	Offset
2	Data length
2	Receive Sequence number
2	Bytes received
n	Data

NetBIOS Interface and Name Service Support by Lower Layer OSI Protocols

The MAP/TOP Users Group Technical Report Specification of NetBIOS Interface and Name Service Support by Lower Layer OSI Protocols, Version 1.0, September 27, 1989, is reproduced as an appendix in The Open Group CAE Specification "Protocols for X/Open PC Interworking: SMB, Version 2."

International Standards Organization (ISO) Protocol Suite

Communications Machinery Corporation has implemented a NetBIOS interface for ISO protocols. "Netbios for ISO Networks", see *Bibliography*

PPP (Point-to-Point Protocol)

NetBIOS can be carried over PPP (Point-to-Point Protocol). The relevant RFCs are:

RFC 2097

The PPP NetBIOS Frames Control Protocol (NBFCP)

PROPOSED STANDARD

RFC1661 STD0051

The Point-to-Point Protocol (PPP)

STANDARD

RFC 2153

PPP Vendor Extensions

INFORMATIONAL

Encapsulating

NetBIOS can be used to encapsulate other protocols by providing a virtual circuit over which other protocols can be transmitted. This is the opposite situation to those described above where other protocols provide the transport for NetBIOS.

Transmission of IP Datagrams over NetBIOS Networks

A standard method of encapsulating the Internet Protocol (IP) datagrams on NetBIOS networks is described in:

RFC 1088

A Standard for the Transmission of IP Datagrams over NetBIOS Networks

Chapter 6. Server Message Block Protocol

There are very many systems which can use the NetBIOS / NetBEUI interface or make use of the NetBIOS Frames Protocol, but perhaps one of the most important is the Server Message Block Protocol (SMB). The Server Message Block Protocol (SMB), is an application level protocol used by networking systems and operating systems such as Microsoft's Windows for Workgroups, Windows 95 / 98 / ME, LAN Manager, Windows NT, Windows 2000 and IBM's OS/2 and LAN Server, NetWare 6 and the SAMBA implementation and as such deserves special attention. The latest versions of the protocol are now known as the "Common Internet File System protocol".

An implementation of SMB is described in "Protocols for X/Open PC Interworking: SMB, Version 2", see *Bibliography*

History

According to the INTERNET-DRAFT document by christopher R Hertel draft-crhertel-smb-url-03.txt titled "SMB Filesharing URL Scheme"

" The Server Message Block protocol (SMB) was created in the 1980's by Dr. Barry Feigenbaum at IBM Corporation. It was later extended by IBM, 3Com, Intel, and Microsoft. "

In 1987 Microsoft announced the LAN Manager program and in 1988 IBM announced the OS/2 LAN Server, both use versions of the Server Message Block Protocol. Enhancements and changes to the protocol have been made and a history can be found at: http://samba.anu.edu.au/cifs/docs/smb-history.html" History of SMB[1]

<mailto:Dan.Shearer@unisa.edu.au>

Some dates in the development of the protocol are given below:

Table 6-1. History of SMB and CIFS

Date	Development
29 November 1989	SMB.TXT is the LM 2.0 protocol. Note: In the doc is calls LM 2.0 as LM 1.2 (it's original name before being renamed to LM 2.0). Microsoft Networks SMB FILE SHARING PROTOCOL EXTENSIONS SMB File Sharing Protocol Extensions Version 3.0 Document Version 1.09
October 1992	Protocols for X/Open PC Interworking: SMB, Version 2
26 March 2001	The Storage Networking Industry Association (SNIA) produced a work-in-progress document: Common Internet File System (CIFS) Version: CIFS-Spec 0.9 Draft SNIA CIFS Work Group Work-in-Progress

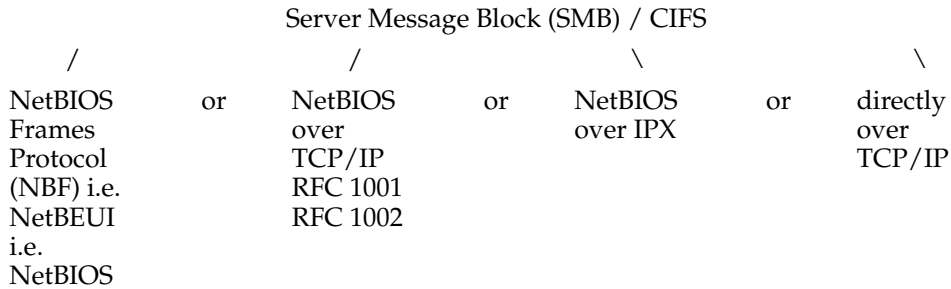
Microsoft and a number of other companies, have proposed an updated version of SMB as an internet standard The Common Internet File System (CIFS).

Overview

The Server Message Block Protocol (SMB), is an application level protocol see Appendix A

SMB is used to implement network session control, network file and print sharing and messaging. SMB is used to provide functionality that is broadly analogous to the AppleTalk Session Protocol, AppleTalk Filing Protocol and Printer Access Protocol etc in the AppleTalk suite of protocols. SMB is also broadly analogous with Novell's NetWare Core Protocol (NCP). It is difficult to find a non-proprietary protocol or protocols with in the TCP/IP suite which can be compared to SMB; file sharing via FTP or NFS and network printing via LPR are examples of similar functionality.

SMB requires a transport /session protocol and the early versions of IBM's implementation were closely linked with NetBIOS. In general SMB runs either over the NetBIOS Frames Protocol (NBF), NetBIOS over TCP/IP, or NetBIOS over IPX; the most recent versions of CIFS can run directly over TCP/IP.



See Appendix A for details of the relationship between the various protocols.

SMB has inherited some of the advantages and disadvantages of NetBIOS, in particular, prior to the latest versions of CIFS it was directly linked with the NetBIOS addressing scheme.

Addressing

Prior to the latest versions of CIFS, SMB uses network names which are strings of 16 bytes. In general these names are mapped directly on to NetBIOS names (see the section called *Addressing - NetBIOS names* in Chapter 2 above). The traditional SMB names of systems can be up to 15 characters long and are padded with blanks if necessary. The 16th byte is used to indicate whether the name refers to a server or another function.

In Microsoft networks with NT 3.x and NT 4.0 systems some names are used with NT 3.x and NT 4.0 Domains as well as for computer names. Some examples of names and use of the 16th byte are given below:

Table 6-2. SMB Names

SMB Name	Purpose
Computername[0x00]	Workstation service
Computername[0x20]	Server service
Domainname[0x00]	Register computer in domain
Domainname[0x1C]	Domain controller

Unique NetBIOS names will map to SMB individual system names, and NetBIOS group names will map to workgroup or domain names.

Like NetBIOS names, traditional SMB names are non hierarchical and constitute a flat non-routable name space which does not scale well.

SMB on NBF

The most recent version of CIFS can run directly over TCP/IP; however many implementations of SMB are designed to run over NBF frames. SMB is designed to use NBF frames as a transport. Whether NBF frames are used natively "on the wire" or encapsulated in TCP/IP, IPX or another protocol should be transparent to SMB.

SMB on NBF datagram frames

SMB uses both NBF datagram and session frames. As explained in the discussion of NBF the datagram frames are used exclusively to provide a datagram service and not a transport for higher level protocols; within this context NBF datagram frames are generally used with SMB frames that are concerned with address management.

Table 6-3. Datagram frames (Octets in order transmitted.)

		Data frame	Data frame
Field Name	**Length**	**DATAGRAM**	**SMB**
Length	2	0x2C	
		0x00	
Deliminator	2	0xFF	
		0xEF	
Command	1	0x08	
Data 1	1	Reserved	
Data 2	2	Reserved	
		Reserved	
XMIT Cor	2	Reserved	

		Data frame	Data frame
Field Name	**Length**	**DATAGRAM**	**SMB**
		Reserved	
RSP Cor	2	Reserved	
		Reserved	
Destination Name	16	Name of receiver	
Source Name	16	Name of sender	
Optional		Datagram	SMB frame

Table 6-4. Datagram frames (Octets in order transmitted.)

		Data frame	Data frame
Field Name	**Length**	**DATAGRAM BROADCAST**	**SMB**
Length	2	0x2C	
		0x00	
Deliminator	2	0xFF	
		0xEF	
Command	1	0x09	
Data 1	1	Reserved	
Data 2	2	Reserved	
		Reserved	
XMIT Cor	2	Reserved	
		Reserved	
RSP Cor	2	Reserved	
		Reserved	
Destination Name	16	Reserved	
Source Name	16	Name of sender	
Optional		Datagram	SMB frame

SMB on NBF session frames

Table 6-5. Session Data Transfer frames (Octets in order transmitted.)

		Data frame	Data frame
Field Name	**Length**	**DATA FIRST MIDDLE**	**SMB**
Length	2	0x0E	
		0x00	
Deliminator	2	0xFF	
		0xEF	
Command	1	0x15	
Data1	1	Brrrxryz	
Data2	2	Re-synch indicator	
		Re-synch indicator	
XMIT Cor	2	nnnn	
		nnnn	
RSP Cor	2	nnnn	
		nnnn	
Dest Num	1	Remote session num	
Source Num	1	Local session num	
Optional data		USER DATA Message from send	SMB frame

Table 6-6. Session Data Transfer frames (Octets in order transmitted.)

		Data frame	Data frame
Field Name	**Length**	**DATA ONLY LAST**	**SMB**
Length	2	0x0E	
		0x00	
Deliminator	2	0xFF	
		0xEF	
Command	1	0x16	
Data1	1	Brrrxryz	
Data2	2	Re-synch indicator	
		Re-synch indicator	

		Data frame	Data frame
Field Name	**Length**	**DATA ONLY LAST**	**SMB**
XMIT Cor	2	nnnn	
		nnnn	
RSP Cor	2	nnnn	
		nnnn	
Dest Num	1	Remote session num	
Source Num	1	Local session num	
Optional data		USER DATA Message from send	SMB frame

SMB frame header

Each SMB frame begins with a standard header. Following a deliminator of "0xFF", there are three bytes "0x53", "0x4d" and "0x42" corresponding to the values "S", "M", "B" which makes identifying SMB frames easier. The three ID bytes are followed by a command byte which is discussed in the section called *SMB Command Codes*

Table 6-7. SMB frames (Octets in order transmitted.)

Field Name	Length	SMB
Deliminator	1	0xFF
ID	3	0x53 "S"
		0x4d "M"
		0x42 "B"
Command	1	0xNN
Error class	1	0xNN
Reserved	1	reserved
Error code	2	0xNN
		0xNN
Flags	1	0xNN
Flags 2 / Reserved	2	0xNN
		0xNN
Reserved? 12?	12	0xNN

Field Name	Length	SMB
		0xNN
		0xNN
		0xNN
		0xNN
		0xNN
		0xNN
		0xNN
		0xNN
		0xNN
		0xNN
		0xNN
authenticated resource identifier / Tree ID	2	0xNN
		0xNN
caller's Process ID	2	0xNN
		0xNN
unathenticated User ID	2	0xNN
		0xNN
Multiplex ID	2	0xNN
		0xNN
count of 16-bit fields Word count	1	0xNN
variable no of 16-bit fields byte count	2	0xNN
		0xNN
count of 8-bit fields that follow	2	0xNN
		0xNN
variable number of 8-bit fields	2	0xNN
		0xNN

SMB is very analogous to the NetWare Core Protocol (NCF); there are numerous functions available for accomplishing various tasks. There are very many SMB frames for different functions and all share the same header format; the second field, "command", determines the function and possibly the format of the rest of the frame following the header.

SMB Command Codes

Below is a table giving some of the Core SMB commands:

Table 6-8. Core SMB Commands

Field Name	smb_com	Description
SMBmkdir	0x00	Create directory
SMBrmdir	0x01	Delete directory
SMBopen	0x02	Open file
SMBcreate	0x03	Create file
SMBclose	0x04	Close file
SMBflush	0x05	Commit all files
SMBunlink	0x06	Delete file
SMBmv	0x07	Rename file
SMBgetatr	0x08	Get file attribute
SMBsetatr	0x09	Set file attribute
SMBread	0x0a	Read byte block
SMBwrite	0x0b	Write byte block
SMBlock	0x0c	Lock byte block
SMBunlock	0x0d	Unlock byte block
SMBmknew	0x0f	Create new file
SMBchkpth	0x10	Check directory
SMBexit	0x11	End of process
SMBlseek	0x12	LSEEK
SMBtcon	0x70	Start connection
SMBtdis	0x71	End connection
SMBnegprot	0x72	Verify dialect
SMBbskattr	0x80	Get disk attributes
SMBsearch	0x81	Search multiple files
SMBsplopen	0xc0	Create spool file
SMBsplwr	0xc1	Spool byte block
SMBsplclose	0xc2	Close spool file
SMBsplretq	0xc3	Return print queue
SMBsends	0xd0	Send message

Field Name	smb_com	Description
SMBsendb	0xd1	Send broadcast
SMBfwdname	0xd2	Forward user name
SMBcancelf	0xd3	Cancel forward
SMBgetmac	0xd4	Get machine name
SMBsendstrt	0xd5	Start multi-block message
SMBsendend	0xd6	End multi-block message
SMBsendtxt	0xd7	Multi-block message text
Never valid	0xfe	Invalid
Implementation-dependent	0xff	Implementation-dependent

Below is a table giving some of the Core plus commands:

Table 6-9. Core plus Commands

Field Name	smb_com	Description
SMBlockreadr	0x13	Lock then read data
SMBwriteunlock	0x14	Write then unlock data
SMBreadBraw	0x1a	Read block raw
SMBwriteBraw	0x1d	Write block raw

Below is a table giving some of the LANMAN 1.0 SMB commands:

Table 6-10. LANMAN 1.0 SMB Commands

Field Name	smb_com	Description
SMBreadBmpx	0x1b	Read block multiplexed
SMBreadBs	0x1c	Read block (secondary response)
SMBwriteBmpx	0x1e	Write block multiplexed
SMBwriteBs	0x1f	Write block (secondary response)
SMBwriteC	0x20	Write complete response
SMBsetattrE	0x22	Set file attributes expanded
SMBgetattrE	0x23	Get file attributes expanded
SMBlockingX	0x24	Lock/unlock byte ranges and X

Field Name	smb_com	Description
SMBtrans	0x25	Transaction (name, bytes in/out)
SMBtranss	0x26	Transaction (secondary request/response)
SMBioctl	0x27	Passes the IOCTL to the server
SMBioctls	0x28	IOCTL (secondary request/response)
SMBcopy	0x29	Copy
SMBmove	0x2a	Move
SMBecho	0x2b	Echo
SMBwriteclose	0x2c	Write and Close
SMBopenX	0x2d	Open and X
SMBreadX	0x2e	Read and X
SMBwriteX	0x2f	Write and X
SMBsesssetup	0x73	Session Set Up and X (including User Logon)
SMBtconX	0x75	Tree connect and X
SMBffirst	0x82	Find first
SMBfunique	0x83	Find unique
SMBfclose	0x84	Find close
SMBinvalid	0xfe	Invalid command

SMB Error Class

Below is a table giving some of the SMB Error class values:

Table 6-11. SMB Error Class

Field Name	Value	Description
SUCCESS	0x00	The request was successful
ERRSRV	0x02	Error generated by the LMX server

SMB Return Codes for Error class 0x00

Below is a table giving some of the SMB Return Code Values when the Error class is 0x00:

Table 6-12. SMB Return Code

Field Name	Value	Description
BUFFERED	0x54	The Message was buffered
LOGGED	0x55	The Message was logged
DISPLAYED	0x56	The Message was displayed

SMB Return Codes for Error class 0x02

Below is a table giving some of the SMB Return Code Values when the Error class is 0x02:

Table 6-13. SMB Return Code

Field Name	Value	Description
ERRerror	0x01	Non-specific error code
ERRbadpw	0x02	Bad password
ERRbadtype	0x03	Reserved

SMB Dialects

The SMB protocol has been developed and enhanced since it was first introduced. The original version is known as the "core protocol" and is understood by systems implementing later versions which are supersets of the original. Systems using SMB negotiate which version i.e. dialect they will support.

The function SMBnegprot 0x72 is used at the beginning of a session to establish the dialect to be used. See the section called *SMB Command Codes*

When packets are being sent to negotiate the dialect, a string is used to indicate which dialects are supported. So just as the use of the string "SMB" within SMB packets makes identifying such packets easier, the use of readable strings makes understanding which dialects are used easier. Below is a table giving some of the strings used to identify dialects and the terms commonly used to refer to the given dialect.

Table 6-14. SMB dialects

string identifying dialect	Reference
PC NETWORK PROGRAM 1.0	core protocol
MICROSOFT NETWORKS 1.03	core plus dialect
MICROSOFT NETWORKS 3.0	extended 1.0 protocol
LANMAN1.0	extended 1.0 protocol, first version of full LANMAN 1.0 protocol
Windows for Workgroups 3.1a	

string identifying dialect	Reference
LM1.2X002	extended 2.0 protocol
LANMAN2.1	
NT LM 0.12	

SAMBA

While this documentation is primarily concerned with protocols rather than implementations; there is one implementation that deserves special mention. A project has been established to provide free implementations of the SMB protocol and file and printing sharing facilities for various platforms. More information can be found about the SAMBA project at the web site: www.samba.org[2]

SAMBA is freely available for very many platforms and has thus provided a means for file and print sharing between different platforms and Operating Systems. The SAMBA project has had to "reverse engineer" the protocols and continues to work in this manner in order to keep the software free.

Despite having released a version of SMB to the X-Open organization, Microsoft continues to develop the protocol as a proprietary protocol and details of some of the more recent versions have not been made freely available.

Further information

Further information is available on the net: Just what is SMB? V1.0 Richard Sharpe [3]

Notes

1. http://samba.anu.edu.au/cifs/docs/smb-history.html
2. http://www.samba.org
3. http://samba.anu.edu.au/cifs/docs/what-is-smb.html

Chapter 7. Browser Service

Microsoft first implemented the Browser Service as a proprietary protocol. The Browser service makes systems visible in the "Network Neighbourhood" within Windows operating systems such as Windows for Workgroups, Windows 9.x and Windows NT and NT2000. The Browser Service also operates in environments such as LAN Manager, LAN Server and OS/2 networks. The Browser Service has nothing to do with Web browsing or HTTP. The Browser Service registers SMB (NetBIOS) names dynamically and makes this dynamic list available to systems on the network. The Browser Service runs over SMB (and is described as running over a "mail slot" protocol over SMB). SMB runs over either NetBIOS Frames Protocol, NBF, often referred to as NetBEUI, or NetBIOS over TCP/IP, or over NetBIOS over IPX/SPX. Thus the Browser service is independent of the transport used to carry SMB.

Because the Browser Service is concerned with the registration of SMB (NetBIOS) names and is dynamic, it is sometimes confused with the NetBIOS Name Service (NBNS) that maps NetBIOS names to IP addresses. An example of NBNS is Microsoft WINS. The Browser Service and NBNS are two separate services.

In very broad terms, the Browser Service can be seen as providing a similar function to the Service Advertising Protocol (SAP) in NetWare environments. Lan Manager servers broadcast their presence over the network and the Browser Service was developed as a solution to the scalability problems of such an arrangement. (Novell developed Novell Directory Services, NDS, to reduce or even replace the need for SAP traffic.) It is important to note that Windows for Workgroups, Windows 9.x systems, Windows NT workstations and NT2000 workstations can share files and thus be servers; in such peer to peer networking environments every system is potentially a server.

History

With Lan Manager 1.0, servers broadcast their presence over the network and client systems listened for such broadcasts to discover servers. This is not dissimilar to the early Novell NetWare systems where servers advertised themselves over the network by broadcasting Service Advertising Protocol (SAP) packets.

When Microsoft introduced Windows for Workgroups, each PC could share it's resources acting as a server as well as acting as a client. Thus the number of servers on the network could potentially equal the number of PCs and become very large. The original broadcast system would not scale in such an environment. It was at this point that the first implementation of the browser service was introduced.

With the introduction of Windows NT, the browser service was expanded to include domains.

Following the development of the next version of SMB, CIFS, Microsoft published the latest version of the Browser protocol as a draft of an internet draft "CIFS/E Browser Protocol". This draft document specifies version 1.15 of the browsing protocol.

Overview

Browser Servers maintain lists of Servers and the services they offer. Other systems, known as Browser Clients (which may also be Browser servers) query the Browser

Servers for information. When Servers come on line they register themselves with the Browser Servers. The Servers are organized in to logical groups according to which group they belong to; these can be "Workgroups" or Domains" and correspond to SMB / NetBIOS group names.

Browser Servers, sometimes simply known as Browsers, can occupy a number of rolls, serving the needs of a particular group or sub-net:

- Domain Master Browser (is Local Master Browser for the sub-net it is on)
- Local Master Browser
- Backup Browser (maintain a copy of the information on the Local Master Browser; they get it by periodically querying the Local Master)
- Potential Browser (system that can become a Browser)

If a client system does not get a response from a Local Master Browser it can initiate an election. The Browser systems and Potential Browser systems communicate to decide which machine will become the Browser.

Client system will contact Browser servers for a list of servers within a group or for lists of groups. Typically clients will keep a list of several Browsers.

Packets

The Browser service uses "Mailslots" and is perhaps the only protocol that does. The mailslot "frames" are carried in SMB "transact" packets. The opcode within the Transact SMB packet is Mailslot Write. Within the data portion of the Transact packet is the mailslot frame. The Transact data itself begins with an opcode as shown below:

Table 7-1. Transact data opcodes

Opcode	description
1	HostAnnouncement
2	AnnouncementRequest
8	RequestElection
9	GetBackupListReq
10	GetBackupListResp
11	BecomeBackup
12	DomainAnnouncment
13	MasterAnnouncement
15	LocalMasterAnnouncement

Further information

Microsoft has published the latest version of the Browser protocol as a draft of an internet draft "CIFS/E Browser Protocol". The document is available as a Microsoft

Word document at: ftp://ftp.microsoft.com/developer/drg/cifs/cifsbrow.doc[1]

Some information is also available from the MSDN Library: The section "Windows Resource Kits" contains a section "Windows 95 Resource Kit" which contains "Chapter 11 Logon, Browsing, and Resource Sharing".

Notes

1. ftp://ftp.microsoft.com/developer/drg/cifs/cifsbrow.doc

Chapter 8. CIFS and the future

The SMB protocol has been developed and renamed CIFS. An Internet Draft dated 13 June 1996 was produced by Microsoft that described version 1.0 of CIFS: "Common Internet File System Protocol (CIFS/1.0)" by I. Heizer, P. Leach and D. Perry

Service Pack 3 for Microsoft's windows NT 4.0 (1996) includes an updated version of the Server Message Block (SMB) authentication protocol, also known as the Common Internet File System (CIFS) file sharing protocol.

More recent versions of CIFS can run "native" over IP without the "NetBIOS over TCP/IP" layer. The use of CIFS running "native" over IP has been implemented in Microsoft's Windows 2000 operating system and subsequent Microsoft Operating Systems.

Appendix A. Open Systems Interconnection (OSI) Reference Model

The Open Systems Interconnection (OSI) Reference Model is traditionally used as a general purpose reference for describing protocols and comparing protocols. It is assumed that the reader is familiar with the OSI model; there are of course numerous resources on the WEB that explain the OSI model.

The diagrams below attempt to show the components of the NetBIOS protocols and higher level protocols such as SMB in relation to the OSI Reference Model. Because the protocols were not developed specifically to comply with the OSI model any mapping is only approximate and intended as a guide. When protocols (such as NetBIOS) are encapsulated within other protocols (such as TCP/IP or IPX) it is particularly difficult to map these to a reference model, thus the diagrams below are intended to help show the relationships between the protocols rather than provide a definitive mapping to the OSI model.

NBF on 802.2 networks

NetBIOS is often described as a session layer protocol but in the IEEE 802.2 implementation there are no transport or datagram delivery protocols between the session layer and the datalink layer. While there is a datagram protocol, this is used exclusively for datagrams and not as a foundation for higher layer protocols.

Table A-1. NBF on 802.2 networks

<table>
<tr><td>7 Application</td><td colspan="2">e.g. Browser Service</td></tr>
<tr><td>6 Presentation</td><td colspan="2">Higher level protocols e.g. SMB / CIFS</td></tr>
<tr><td>5 Session</td><td rowspan="2"></td><td>Session Management Protocol</td></tr>
<tr><td>4 Transport</td></tr>
<tr><td>3 Network</td><td colspan="2">User Datagram Protocol, Name Management Protocol, NetBIOS Diagnostic and Monitoring Protocol</td></tr>
<tr><td rowspan="2">2 Datalink</td><td colspan="2">IEEE 802.2</td></tr>
<tr><td colspan="2">IEEE 802.3 / IEEE 802.5 etc</td></tr>
<tr><td>1 Physical</td><td colspan="2">Token Ring / Ethernet etc</td></tr>
</table>

NetBIOS over TCP/IP

NetBIOS over TCP/IP is described in RFC 1001 and RFC 1002. Note that when higher level protocols such as SMB or CIFS are implemented over TCP/IP they are in fact implemented over NetBIOS over TCP/IP.

Table A-2. NetBIOS over TCP/IP

<table>
<tr><td>7 Application</td><td colspan="2"></td><td>e.g. Browser Service</td></tr>
<tr><td>6 Presentation</td><td colspan="3">Higher level protocols e.g. SMB / CIFS</td></tr>
<tr><td>5 Session</td><td>Name Service</td><td>datagram service</td><td>Session Service</td></tr>
<tr><td>4 Transport</td><td colspan="2">UDP</td><td>TCP</td></tr>
<tr><td>3 Network</td><td colspan="3">IP</td></tr>
<tr><td rowspan="2">2 Datalink</td><td colspan="2">e.g. IEEE 802.2</td><td rowspan="2">e.g. Ethernet II etc</td></tr>
<tr><td colspan="2">IEEE 802.3 / IEEE 802.5 etc</td></tr>
<tr><td>1 Physical</td><td colspan="3">Token Ring / Ethernet etc</td></tr>
</table>

NetBIOS over IPX

NetBIOS over IPX uses IPX packets to provide the underlying delivery mechanism for the NetBIOS protocols.

Table A-3. NetBIOS over IPX

<table>
<tr><td>7 Application</td><td colspan="2"></td><td>e.g. Browser Service</td></tr>
<tr><td>6 Presentation</td><td colspan="3">Higher level protocols e.g. SMB / CIFS</td></tr>
<tr><td>5 Session</td><td colspan="2" rowspan="2"></td><td rowspan="2">Session Management Protocol</td></tr>
<tr><td>4 Transport</td></tr>
<tr><td rowspan="2">3 Network</td><td colspan="2">User Datagram Protocol, Name Management Protocol, NetBIOS Diagnostic and Monitoring Protocol</td><td></td></tr>
<tr><td colspan="3">IPX</td></tr>
<tr><td rowspan="2">2 Datalink</td><td colspan="2">e.g. IEEE 802.2</td><td rowspan="2">e.g. Ethernet II etc</td></tr>
<tr><td colspan="2">IEEE 802.3 / IEEE 802.5 etc</td></tr>
<tr><td>1 Physical</td><td colspan="3">Token Ring / Ethernet etc</td></tr>
</table>

CIFS over TCP/IP

The latest version of CIFS can run directly over TCP/IP.

Table A-4. CIFS over TCP/IP

<table>
<tr><td>7 Application</td><td colspan="2">CIFS</td></tr>
<tr><td>6 Presentation</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>5 Session</td></tr>
<tr><td>4 Transport</td><td>UDP</td><td>TCP</td></tr>
<tr><td>3 Network</td><td colspan="2">IP</td></tr>
<tr><td rowspan="2">2 Datalink</td><td>e.g. IEEE 802.2</td><td rowspan="2">e.g. Ethernet II etc</td></tr>
<tr><td>IEEE 802.3 / IEEE 802.5 etc</td></tr>
<tr><td>1 Physical</td><td colspan="2">Token Ring / Ethernet etc</td></tr>
</table>

Appendix B. NetBIOS protocols in IBM PC Network

The NetBIOS interface was developed by Sytec Inc. (which became Hughes LAN Systems, then Whittaker Communications) for International Business Machines Corporation (IBM) in 1983. This operated over proprietary Sytec protocols on IBM's PC Network.

Information is provided here for reference although these protocols have now been superseded by the NetBIOS Frames protocol running over Token Ring or Ethernet etc.

Name Management Frames in IBM PC Networks

In the IBM PC Network name management is performed by the Name Management Protocol which consists of the following packet types:

Name Claim / Name Cancel Packet

A single octet indicates whether the packet is used to claim a name (value 0x10) or release a name (value 0xA0)

Name Claim Response Packet

This packet is used to indicate that the name is already in use.

Details of packet structures for Name Management in IBM PC Networks are given in the the section called *Comparison of NetBIOS protocols in IBM PC Network*

Name Claim / Name Cancel Packet in IBM PC Network

From "Inside NetBIOS":

Name Claim / Name Cancel Packet:

Table B-1. Name Claim / Name Cancel Packet

Field	Length	Description
Start	1	Start Deliminator value 0x7E
Destination	6	Destination Address
Source	6	Source Address
Length	2	Length of packet
value	2	value set to 0x5000
Function	1	10h for Name Claim, 0xA0 for Cancel
Accept	1	Number of packets willing to accept
Connection id	2	Connection id

Field	Length	Description
value	2	value set to 0x0202
Undefined	2	value of two octets undefined
value	2	value set to 0x0400
Undefined	4	value of four octets undefined
value	2	value of 0x10XX
value	4	value of 0x0000
Dest Name	16	ASCII Destination NetBIOS name
Dest Node Connection id	2	Destination node connection id
CRC	4	Cyclic Redundancy Check
End of Frame	1	End of frame marker value of 0x7E

Name Claim Response Packet in IBM PC Network

Name Claim Response Packet:

Table B-2. Name Claim Response Packet

Field	Length	Description
Start	1	Start Deliminator value 0x7E
Destination	6	Destination Address
Source	1	Source Address
Length	2	Length of packet
value	2	value set to 0x4000
value	1	value set to 0x30
Accept	1	Number of packets willing to accept
Connection id	2	Connection id
Undefined	2	value of two octets undefined
Reason packet NAK	1	Reason why packet not acknowledged
Undefined	1	value of octet undefined
value	2	value set to 0x0400

Field	Length	Description
Undefined	4	value of four octets undefined
value	2	value of 0x10XX
value	4	value of 0x0000
Dest Name	16	ASCII Destination NetBIOS name
Destination node connection id	2	Destination node connection id
CRC	4	Cyclic Redundancy Check
End of Frame	1	End of frame marker value of 0x7E

Datagram Packet in IBM PC Network

Details of packet structures for User Datagram Protocol in IBM PC Networks are given in the the section called *Comparison of NetBIOS protocols in IBM PC Network*

User Datagram Protocol Packet in IBM PC Network

From "Inside NetBIOS":

User Datagram Protocol Packet:

Table B-3. User Datagram Protocol Packet

Field	Length	Description
Start	1	Start Deliminator value 0x7E
Destination	6	Destination Address
Source	6	Source Address
Length	2	Length of packet
value	2	value set to 0x5100
value	2	value set to 0x0100
value	2	value set to 0x0001
value	2	value set to 0x1010
value	2	value set to 0x0000
Source Name	16	ASCII Source NetBIOS name
Dest Name	16	ASCII Destination NetBIOS name

Field	Length	Description
Data	variable	data
Retransmit Count	2	Retransmition Count
Source Node Connection id	2	Source Node Connection id
Destination id	6	Destination id
Source id	6	Source id
Prev Node id	6	Previous Node id
CRC	4	Cyclic Redundancy Check
End of Frame	1	End of frame marker value of 0x7E

NetBIOS Session Management Protocol in IBM PC Networks

Details of packet structures for NetBIOS Session Management in IBM PC Networks are given in the the section called *Comparison of NetBIOS protocols in IBM PC Network*

Session Request Packet in IBM PC Network

From "Inside NetBIOS":

Session Request Packet :

Table B-4. Session Request Packet

Field	Length	Description
Start	1	Start Deliminator value 0x7E
Destination	6	Destination Address
Source	6	Source Address
Length	2	Length of packet
value	2	value set to 0x0040
Function	1	value 0x00-0x07=No poll 0x80-0x0F=Send Return Pkt
Accept	1	Number of packets willing to accept
Connection id	2	Connection id
Sess seq no	1	Sess seq no
ACK Seq No	1	ACK Seq No
value	2	value set to 0x0001

Field	Length	Description
Response packet size	2	Response packet size
value	4	value of 0x0000
value	4	value of 0x1010
Source Name	16	ASCII Source NetBIOS name
Dest Name	16	ASCII Destination NetBIOS name
Dest Connection id	2	Dest Connection id
CRC	4	Cyclic Redundancy Check
End of Frame	1	End of frame marker value of 0x7E

Comparison of NetBIOS protocols in IBM PC Network

Table B-5. Name Management Packets

Name Claim / Cancel	Name Response
< - 1 Octet (8 bits) ->	**< - 1 Octet (8 bits) ->**
Start Deliminator = 0x7E	Start Deliminator = 0x7E
Destination Address 6 octets	Destination Address 6 octets
Source Address 6 octets	Source Address 6 octets
Length 2 octets	Length 2 octets
Value 0x5000	Value 0x4000
Claim 0x10 Cancel 0xA0	Value 0x30
No Packets to accept N	No Packets to accept N
Connection id 2 octets	Connection id 2 octets
Value 0x02	Undefined
Value 0x02	Undefined
Undefined	Reason NAK
Undefined	Undefined
Value 0x04	Value 0x04
Value 0x00	Value 0x00
Undefined 4 octets	Undefined 4 octets
Value 0x10	Value 10h
Value XXh	Value XXh
Value 0x00	Value 0x00

Name Claim / Cancel	Name Response
< - 1 Octet (8 bits) ->	< - 1 Octet (8 bits) ->
Value 0x00	Value 0x00
ASCII Dest name 16 octets	ASCII Dest name 16 octets
Prev net con id	Dest node conn id
Prev net con id	Dest node conn id
Retransmit count	CRC 4 octets
Retransmit count	
Source node con id	
Source node con id	
Dest id 6 octets	EOF 0x7E
Source id 6 octets	
Prev node id 6 octets	
CRC 4 octets	
EOF 0x7E	

Table B-6. Session frames

Name Query	Session Request	Session Accept	Session	Acknowledgment
< - 1 Octet (8 bits) ->	< - 1 Octet (8 bits) ->	<- 1 Octet (8 bits) ->	< - 1 Octet (8 bits) ->	< - 1 Octet (8 bits) ->
Start Deliminator = 0x7E	Start Deliminator = 0x7E	Start Deliminator = 0x7E	Start Deliminator = 0x7E	Start Deliminator = 0x7E
Destination Address 6 octets	Destination Address 6 octets	Destination Address 6 octets	Destination Address 6 octets	Destination Address 6 octets
Source Address 6 octets	Source Address 6 octets	Source Address 6 octets	Source Address 6 octets	Source Address 6 octets
Length 2 octets	Length 2 octets	Length 2 octets	Length 2 octets	Length 2 octets
Value 0x5000	Value 0x0040	Value 0x0040	Value 0x4000	Value 0x4000
Value 0x10	0x00 - 0x07 No Poll 0x80 -0x0F Send return packet	0x00 - 0x07 No Poll 0x80 -0x0F Send return packet	0x00 - 0x07 No Poll 0x80 -0x0F Send return packet	0x40 - 0x47 No Poll 0x48 -0x4F Send return packet
No Packets to accept 0?h	No Packets to accept 0?h	No Packets to accept 0?h	No Packets to accept 0?h	No Packets to accept N

Name Query	**Session Request**	**Session Accept**	**Session**	**Acknowledgment**
< - 1 Octet (8 bits) ->	**< - 1 Octet (8 bits) ->**	**<- 1 Octet (8 bits) ->**	**< - 1 Octet (8 bits) ->**	**< - 1 Octet (8 bits) ->**
Connection id 2 octets	Connection id 2 octets	Connection id 2 octets	Connection id 2 octets	Connection id 2 octets
Value 0x02	Ses Seq No	Ses Seq No	Ses Seq No	Ses Seq No
Value 0x02	ACK Seq No	ACK Seq No	ACK Seq No	ACK Seq No
Undefined	Value 0x00	Value 0x00	0x80-0xF0 End message	Undefined
Undefined	Value 0x01	Value 0x02	DATA N octets	Dest Node Con
Value 0x10	Response packet size	Response packet size		Dest Node Con
Value 0x00	Response packet size	Response packet size		CRC 4 octets
Undefined	Value 0x00	Dest node conn id		
Undefined	Value 0x00	Dest node conn id		
Undefined	Value 0x10	CRC 4 octets		
Undefined	Value 0x10			EOF 0x7E
Value 0x10	ASCII Source name			
Value 0xXX	ASCII Source name			
Value 0xXX	ASCII Source name	EOF 7Eh		
Value 0x10	ASCII Source name			
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name		Dest node con id	

Name Query	Session Request	Session Accept	Session	Acknowledgment
< - 1 Octet (8 bits) ->	**< - 1 Octet (8 bits) ->**	**<- 1 Octet (8 bits) ->**	**< - 1 Octet (8 bits) ->**	**< - 1 Octet (8 bits) ->**
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name		CRC 4 octets	
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name			
ASCII Dest name	ASCII Source name		EOF 0x7E	
ASCII Dest name	ASCII Dest name			
ASCII Dest name	ASCII Dest name			
ASCII Dest name	ASCII Dest name			
ASCII Dest name	ASCII Dest name			
Prev net con id	ASCII Dest name			
Prev net con id	ASCII Dest name			
Retransmit count	ASCII Dest name			
Retransmit count	ASCII Dest name			
Source node con id	ASCII Dest name			
Source node con id	ASCII Dest name			
Dest id	ASCII Dest name			
Dest id	ASCII Dest name			
Dest id	ASCII Dest name			

Name Query	**Session Request**	**Session Accept**	**Session**	**Acknowledgment**
< - 1 Octet (8 bits) ->	**< - 1 Octet (8 bits) ->**	**<- 1 Octet (8 bits) ->**	**< - 1 Octet (8 bits) ->**	**< - 1 Octet (8 bits) ->**
Dest id	ASCII Dest name			
Dest id	ASCII Dest name			
Dest id	ASCII Dest name			
Source id	Dest node conn id			
Source id	CRC			
Source id	CRC			
Source id	CRC			
Source id	CRC			
Source id	EOF 0x7E			
Prev node id				
Prev node id				
Prev node id				
Prev node id				
Prev node id				
Prev node id				
CRC				
CRC				
CRC				
CRC				
EOF 0x7E				

Appendix C. Active Directory

With the introduction of Windows 2000, Microsoft introduced Active Directory. Active Directory can be described as a hierarchical directory system, broadly similar in concept to Novell's Directory Services or the X.500 directory system. Among the uses of Active Directory is management of resources in a CIFS network; facilities such as the traditional browser service can be "replaced" by Active Directory. Because Active Directory has a significant impact upon CIFS networking, a brief overview is presented here, however a full description is beyond the scope of this documentation; there are many excellent references on Active Directory.

As described above Active Directory (AD) implements a "tree structure" of objects that is broadly analogues to other directory systems that use "X.500" type technology. Indeed Active Directory is implemented using other standard directory technologies. In particular AD makes use of the Domain Name System (DNS) to establish an overall hierarchical tree structure. The Lightweight Directory Access Protocol (LDAP) directory system is also used to provide further granularity and provide facilities not available in the DNS. Traditional NT Domain technology is also used and provides backwards compatibility.

In order to use AD the TCP/IP v4 protocol must be configured (both DNS and LDAP run over TCP/IP). While the Browser service also ran over IPX/SPX and the NetBIOS Frames Protocol (otherwise known as NBF or NetBEUI), this is not the case with AD.

Microsoft has produced a Knowledge Base Article that provides a list of Windows 2000 Domain Controller Default Ports. This provides an insight in to the protocols used with Active Directory.

The Knowledge Base Article is Q289241 and can be found at http://support.microsoft.com/default.aspx?scid=kb;en-us;289241[1]

Domain Name System (DNS)

Active Directory requires a DNS infrastructure to be in place. AD does require that the DNS support dynamic updates, but uses the standard DNS. Thus in order to understand the impact of Active Directory on a network, it is necessary to understand the impact of DNS.

Some relevant RFCs are given below:

- RFC 1035: "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION"
- RFC 3007: "Secure Domain Name System (DNS) Dynamic Update"
- RFC 2136: "Dynamic Updates in the Domain Name System (DNS UPDATE)"
- RFC 2782: "A DNS RR for specifying the location of services (DNS SRV)"

Lightweight Directory Access Protocol (LDAP)

Active Directory uses Lightweight Directory Access Protocol (LDAP) to provide additional granularity to the "tree" structure. LDAP can be used to create Organizational Units (OUs) within the "tree" structure.

Some relevant RFCs are given below:

- RFC 2256: "A Summary of the X.500(96) User Schema for use with LDAPv3"
- RFC 2251: "Lightweight Directory Access Protocol (v3)"
- RFC 1777: "Lightweight Directory Access Protocol"
  (Original definition)

Notes

1. http://support.microsoft.com/default.aspx?scid=kb;en-us;289241

Glossary

Glossary

A

AD

Active Directory.

B

Baseband

Systems that put digital signals from the data communications device on to the cable without modulation; only one data signal can be carried.

BIOS

basic input/output system.

Broadband

Systems that have multiple independent frequency channels, usually achieved by Frequency Division Multiplexing.

Browser Server

Systems which maintain lists of Servers and the services they offer.

C

CIFS

Common Internet File System - the latest version of SMB

D

DMP

NetBIOS Diagnostic and Monitoring Protocol

Domain

A logical grouping of systems within an SMB / CIFS network used for management and authentication. Within Microsoft networks a domain might be an NT 4.0 domain or Windows 2000 domain.

DSAP

Destination Service Access Point

G

Group Name

NetBIOS (and SMB / CIFS) name shared by a number of systems on the network.

H

Hybrid node

Hybrid nodes ("H" nodes") are nodes on a network using NetBIOS over TCP/IP. Hybrid nodes are not defined in RFC 1001 and have not been standardized; these are mixed nodes (similar to "M" nodes) but function broadly in the opposite manner to "M" nodes. "H" nodes function as a "Point to Point" node first and then as a "Broadcast" node.

I

IEEE

Institute of Electrical and Electronics Engineers

IPX

Internetwork Packet Exchange

ISO

International Standards Organization

M

MAC

Media Access Control

N

NBDD

NetBIOS Datagram Distribution Server

NBF

NetBIOS Frames protocol

NBFCP

NetBIOS Frames Control Protocol

NBIPX

NetBIOS over IPX

NBNS

NetBIOS Name Server

NBT

NetBIOS over TCP/IP (term often seen in Microsoft documentation).

NetBIOS

Network Basic Input / Output System

NetBEUI

NetBIOS Extended User Interface

NetBT

NetBIOS over TCP/IP (term often seen in Microsoft documentation).

NMP

Name Management Protocol

P

PPP

Point-to-Point Protocol

S

SAP

Service Access Points

SMB

Server Message Block

SMP

System Message Block protocol

SNAP

Sub-Network Access Protocol

SPX

Sequenced Packet Exchange

SSAP

source Service Access Point

U

UDP

NetBIOS User Datagram Protocol or User Datagram Protocol in TCP/IP

W

WINS

Windows Internet Name Service - Microsoft's implementation of NBNS

Bibliography

The following texts have been used in the preparation of this documentation:

Information about NetBIOS, NetBEUI, NBF, SMB, and CIFS networking can be found "On line" at [http://www.timothydevans.me.uk/nbf.htm NetBIOS, NetBEUI, NBF, SMB, CIFS networking links page] [1]

BYTE Magazine November 1989 "Two tin cans and some string" Part 2 , Rick Grehan.

BYTE Magazine January 1989 "Understanding NetBIOS", Brett Glass.

Inside NETBIOS, 3rd Edition, J. Scott Haugdahl, Architecture Technology corporation, ISBN 0-939405-00-8.

Novell's Guide to NetWare LAN Analysis, 2nd Edition, Laura A. Chappell and Dan E. Hakes, Novell Press SYBEX, ISBN 0-7821-1362-1.

Networking Technologies, 2nd Edition, Dorothy Cady, Drew Heywood, Debra Niedermiller-Chaffins, and Cheryl Wilhite, New Riders Publishing, ISBN 1-56205-309-4.

802.2 Logical Link Control: ANSI/IEEE Std 802.2-1985, *ISO Draft*, *International Standard 8802/2*, ISBN 0-471-82748-7.

IBM LAN Technical Reference: IEEE 802.2 and NetBIOS Application Program Interfaces, Second Edition (May 1996) SC30-3587-01.

Netbios for ISO Networks, Stephen Thomas, Computer Communications Review - A Quarterly Publication of the Special Interest Group on Data Communications , July / August 1987 Volume 17, No 3.

Protocols for X/Open PC Interworking: SMB, Version 2: The Open Group, X/Open Document Number: C209, ISBN 1-87263-045-6.

Forbury Road
Reading
Berkshire
RG1 1AX
United Kingdom

Windows NT TCP/IP, Dr.. Karanjit S. Siyan, New Riders Publishing, ISBN 1-56205-887-8.

Notes

1. http://www.timothydevans.me.uk/nbf.htm

Appendix D. Document History

This document has been revised from time to time. In August 2001 a change history was added. Some of the changes to the document are listed below:

- 1998 Documentation posted on Web
- October 2001
  - Change History added.
  - Document layout changed.
  - Browser Section added.
  - Information about NetBIOS on IBM PC network moved to appendix.
  - Section on NetBIOS Addresses modified.
  - General corrections and formatting changes.
- January 2002
  - Section on Name Resolution (in TCP/IP encapsulation) added.
  - Use of CSS introduced.
  - Minor alterations to Contents section, SMB section and glossary.
- September 2002
  - Documentation overhauled. All sections edited.
  - Documentation converted to xml docbook format.

Background

How this documentation came to be written is described below:

During the 1990s there was a move towards distributed networked systems from traditional centralized systems; "down-sizing" and "right-sizing" were some of the "buzz-words" of the time. There was a considerable growth in PC Networks including those using the Server Message Block protocol.

During this time I was in the Network group of the organization I worked for. It seemed sensible to find out a little about the protocols such as NetBIOS and SMB that were being discussed at that time. As I tried to discover a little about these protocols, I found considerable confusion and very little documentation or reliable information.

During the late 1990s there were many books on other protocol suites such as TCP/IP, IPX/SPX, AppleTalk and others but there seemed to be none dedicated to NetBIOS or SMB. Eventually I found "Inside NETBIOS, 3rd Edition", by J. Scott Haugdahl, Architecture Technology corporation and later I obtained the official documentation from IBM: "IBM LAN Technical Reference: IEEE 802.2 and NetBIOS Application Program Interfaces, Second Edition (May 1996) SC30-3587-01."; these were (and as far as I know still are) the only books specifically describing the NetBIOS Frames Protocol. While there are now several books that include sections on the protocol the above mentioned books are the only ones that are dedicated to the topic. The situation with

SMB seems to be worse; apart from "Protocols for X/Open PC Interworking: SMB, Version 2: The Open Group, X/Open Document Number: C209, ISBN 1-87263-045-6.", I am not aware of any book that exclusively describes SMB.

I decided to post references to information I had found on a web site in the hopes that it might be useful to others and that this might prompt others to point me in the direction of useful information. As it became apparent how little documentation existed, I made some documentation available as a collection of web pages.

Over the years a number of people have been kind enough to let me know that they found the documentation useful and so I have continued to develop it on an ad-hoc basis. One request that I consistently received was for the documentation in another format, preferably as a single file, and typically as a PDF document. The most sensible approach seemed to be to convert the documentation to xml format as a docbook document that could then be converted to whatever format was desired for which converters were available.

Colophon

Colophon

This document has been produced as an xml docbook http://www.docbook.org[1]. Tools have been used to convert the document to other forms such as html.

- To convert to html, docbook2html was used and the resulting html was then process with htmltidy:

  tidy -asxml -q

- To convert to RTF, docbook2rtf was used.
- To convert to PDF, docbook2pdf was used.

Notes

1. http://www.docbook.org