ODI Specification Supplement:

Source Routing

ODI Specification Version 4
Document Version 1.10
Part Number: 107–000058–001
28 July 1994

Disclaimer

Novell, Inc. makes no representations or warranties with respect to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Novell, Inc.
122 East 1700 South
Provo, Utah 84606

Trademarks

Novell has made every effort to supply trademark information about company names, products, and services mentioned in this document. Trademarks indicated below were derived from various sources.

Novell and NetWare are registered trademarks of Novell, Inc.

Internetwork Packet Exchange, ODI, Open Data-Link Interface, LSL, Link Support Layer, MLID, Multiple Link Interface Driver, MLI, Multiple Link Interface, MPI, Multiple Protocol Interface, MSM, Media Support Module, TSM, Topology Support Module, HSM, Hardware Support Module, RX-Net, NE1000, NE2000, NE/2, NE2-32, and NTR2000 are trademarks of Novell, Inc.

Table Of Contents

List of Figures

Overview

This supplement describes how to accomplish source routing in the NetWare environment. It also describes the different parameters that the send and receive packet routines require in order to implement source routing. This supplement is provided for information purposes only.

This supplement is broken into two parts. The first section, "Source Routing for the NetWare Server," explains the details of the interface between the server MLID and the source routing module. The loading options for ROUTE.NLM are described at the end of this section.

The second section, "Source Routing for the DOS and OS/2 Clients," explains source routing on DOS and OS/2 clients.

This supplement contains reference material. You should review this supplement if you are developing an MLID that must incorporate source routing.

Introduction to Source Routing

For networks comprised of more than one physical segment, IBM bridges add source routing information to the frame header at the Media Access Control (MAC) layer. When a packet that travels across one or more bridges arrives at its destination, it contains the routing information for the route that it travelled. All subsequent communications between the source and destination nodes of that packet travel the same path.

Source Routing for the NetWare Server

When an MLID supports source routing, it must call the source routing module, ROUTE.NLM, each time it transmits or receives a packet. The token-ring TSM and FDDI TSM make all necessary calls to ROUTE.NLM. These calls are transparent to the HSM. References to the "driver" in this chapter refer to transmit and receive code within the TSM.

The NetWare operating system handles packet routing between multiple physical segments above the MAC layer, so usually the packet does not require source routing information. The NetWare operating system adds the source routing information only when it must pass packets across an IBM bridge. The ROUTE.NLM module provides this capability.

Any token-ring MLID can use ROUTE.NLM to support source routing. Once it is loaded, the function of ROUTE.NLM is

transparent to the operating system and any application NLMs that are running on the file server. ROUTE.NLM automatically builds the source routing information into all frames that are transmitted by the NetWare operating system.

ROUTE.NLM requires the following:

- NetWare 3.1 or later
- At least one adapter/driver combination installed and configured to support token-ring source routing

NetWare Server Driver Source Routing Support

When an MLID supports source routing, it must initialize the *MLIDRouteHandler* field, offset 46h of the configuration table, with the offset of a routine that issues a RET instruction. When ROUTE.NLM is loaded, it replaces the pointer to the routine originally pointed to in the *MLIDRouteHandler* field with the offset of its own entry point.

Calling the Source Routing Module for Transmits

Before a send routine calls the routing module, the following registers must contain:

EAX the board number
ECX 0
ESI the destination address

On return from the source routing routine, the following conditions are in effect:

EAX is destroyed
ECX the source routing field size
ESI the source routing address
Interrupts are disabled
cld is in effect
EDX, EBP, EBX, and EDI are preserved

The server MLID uses the information in ECX and ESI to insert the source routing information into the appropriate place in the frame before it is placed on the media. If ECX equals 0, there is no source routing information for the destination address. The following example illustrates inserting the source routing information.

```
; EBP points to the configuration table
; ESI points to the Send ECB
sub  ecx,ecx                        ; ECX = default source routing size
mov eax, [esi].BoardNumber          ; EAX = board number
lea  esi, [esi].ImmediateAddress    ; ESI = destination address
call  MLIDRouteHandler[ebp]         ; offset 46h of configuration table
```

Calling the Source Routing Module for Receives

The MLID must also call the source routing module when it receives a valid frame. When the MLID calls the module, the registers contain the following information:

EAX the board number
ECX the address of the source routing field
ESI the source address of the frame

On return from the source routing module, the following conditions are in effect:

EAX is destroyed
ECX is destroyed
ESI is destroyed
Interrupts are disabled
cld is in effect
EDX, EBP, EBX and EDI are preserved

Source Routing Routines

The following pages document the source routing routines. These routines are reached through the *MLIDSrcRouting* field in the MLID configuration table. When ESI = 0, the *MLIDSrcRouting* field contains the address of one of these routines.

ChangeBroadcastResponseType

Description

Changes the type of the broadcast response.

Entry State

AL
is equal to 6, the *ChangeBroadcastResponseType* function code.

AH
determines how the server should respond to broadcast requests (see Description).

BL
the board number to change.

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

CH
contains the old broadcast response type.

EDX
is destroyed.

Interrupts
state is not changed.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

Valid response types which can be put in AH are listed below:

00h The server should respond directly to all broadcast requests. That is, the response is not a broadcast response.

82h The server should respond to broadcast requests with an all-routes broadcast response.

C2h The server should respond to broadcast requests with a single-route broadcast response.

ChangeBroadcastResponseType *continued*

Example

```
mov     al, 06    ; al = change broadcast response
                                        ; type function code
mov bl, [ebp].MLIDBoardNumber       ; bl = board number
call  SRControl
```

ChangeGeneralBroadcastRoute

Description

Changes the route of general broadcast packets.

Entry State

AL
is equal to 4, the *ChangeGeneralBroadcastRoute* function code.

AH
contains a new general broadcast route:
0C2h = single-route broadcast
082h = all-routes broadcast

BL
the board number to change.

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

CH
contains the old general broadcast route.

EDX
is destroyed.

Interrupts
state is not changed.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

This routine can be called at process or interrupt time.

Example

```
mov     al, 04          ; al = change general broadcast
                                        ; address function code
mov bl, [ebp].MLIDBoardNumber     ; bl = board number
call   SRControl
```

ChangeMulticastBroadcastRoute

Description

Changes the route of multicast broadcast packets.

Entry State

AL
is equal to 5, the *ChangeMulticastBroadcastRoute* function code.

AH
contains the new multicast broadcast route:
0C2h = single-route broadcasts
082h = all-routes broadcast

BL
the board number to change.

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

EDX
is destroyed.

CH
the old multicast broadcast route.

Interrupts
state is not changed.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

This routine can be called at process or interrupt time.

Example

```
mov     al, 05          ; al = change multicast broadcast
                                        ; address function code
mov bl, [ebp].MLIDBoardNumber     ; bl = board number
call   SRControl
```

ChangeSourceRoutingUpdateTableTimer

Description

Changes the time between source routing table updates.

Entry State

AL
is equal to 7, the *ChangeSourceRoutingUpdateTableTimer* function code.

AH
a new timer value in seconds.

BL
the board number to change.

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

CH
now contains the old timed value in seconds.

EDX
is destroyed.

Interrupts
state is not changed.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

The value in AH specifies the number of seconds that ROUTE.NLM should wait before updating its source routing table. If an entry in the source routing table has not been used for the amount of time specified in AH, ROUTE.NLM automatically updates the route as soon as a new route is available.

Example

```
mov     al, 07    ; al = change source routing
update

mov bl, [ebp].MLIDBoardNumber      ; bl = board number
call  SRControl
```

ChangeUnknownDestinationAddressRoute

Description

Changes the route for packets with unknown destinations.

Entry State

AL
is equal to 3, the *ChangeUnknownDestinationAddressRoute* function code.

AH
contains a new *UnknownDestinationAddress* control field:
0C2h = single-route broadcast
082h = all-routes broadcast

BL
the board number to change.

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

CH
now contains the *UnknownDestinationAddress* control field:
0C2h = single-route broadcast
082h = all-routes broadcast

EDX
is destroyed.

Interrupts
are disabled.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

This routine can be called at process or interrupt time.

Example

```
mov     al, 03          ; al = change unknown destination
                                        ; address function code
mov bl, [ebp].MLIDBoardNumber     ; bl = board number
call   SRControl
```

ClearSourceRoutingTable

Description

Clears the source routing table.

Entry State

AL
is equal to 2, the *ClearSourceRoutingTable* function code.

BL
the board number to change.

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

EDX
is destroyed.

Interrupts
are disabled.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

This routine can be called at process or interrupt time.

Example

```
mov     al, 02    ; al = clear table function code
mov bl, [ebp].MLIDBoardNumber      ; bl = board number being
                                     loaded
call  SRControl
```

RemoveNodefromSourceRouteTable

Description

Removes the node from the source routing table.

Entry State

AL
is equal to 8, the *RemoveNodefromSourceUpdateTableTimer* function code.

BL
the board number from which to remove the node.

EDI
the address of the 6-byte node address to be removed

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

CH
now contains the old timed value in seconds.

EDX
is destroyed.

Interrupts
state is not changed.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

The node address pointed to by EDI is removed from the source routing table for this logical board.

Example

```
mov     al, 08    ; al = remove source routing                    update
mov bl, [ebp].MLIDBoardNumber      ; bl = board number
call  SRControl
```

LoadBoardNumber

Description

Loads the source routing MLID.

Entry State

AL
is equal to 0, the *LoadBoardNumber* function code.

BL
the board number to load.

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

EDX
is destroyed.

Interrupts
are disabled.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

Do not call this routine at interrupt time.

Example

```
mov     al, 0      ; al = load board function code
mov bl, [ebp].MLIDBoardNumber       ; bl = board number being
                                      unloaded
call  SRControl
```

UnloadBoardNumber

Description

Unloads the source routing MLID.

Entry State

AL
is equal to 1, the *UnloadBoardNumber* function code.

BL
the board number to unload.

Return State

EAX
the completion code.

EBX
the input board number.

ECX
is destroyed.

EDX
is destroyed.

Interrupts
are disabled.

Flags
cld is in effect; the 80386 status flag is set according to EAX.

Preserved
EBP, EDI, and ESI

Completion Codes (EAX)

00000000h	*Successful*
FFFFFF81h	*BadCommand*
FFFFFF82h	*BadParameters*
FFFFFF85h	*ItemNotPresent*
FFFFFF89h	*OutofResources*

Remarks

Do not call this routine at interrupt time.

Example

```
mov     al, 01    ; al = load board function code
mov bl, [ebp].MLIDBoardNumber      ; bl = board number being
                                     loaded
call  SRControl
```

Loading ROUTE.NLM

ROUTE.NLM can be used to reentrantly load multiple boards or to change the configuration of a previously loaded board. To install ROUTE.NLM, enter the following command at the NetWare file server console:

```
load route   [board=nn], [clear], [def], [gbr], [mbr],
[remove=xxxxxxxxxxxx],[rsp=xx], [time=ss],
[unload]
```

The syntax is:

load The NetWare console command to load NLMs.

route ROUTE.NLM filename. This can be prefaced with the path if necessary.

The load options are:

board A decimal number that specifies the board number to be loaded. The board number must refer to an adapter that supports source routing. If this option is not specified, the default value is 1. If the board has already been loaded, ROUTE.NLM assumes that the board is being reconfigured.

clear Clears the source routing table for board=*nn*. This parameter is only valid if the specified board is already loaded.

Source routing information is built into a table as frames are received from a particular node. This parameter clears the table, forcing the ROUTE.NLM to dynamically rediscover the routes of all nodes by sending a default (def) frame when the NetWare OS addresses each specific node in the network. If an IBM bridge on the network has gone down, this option can find an alternate route, if one exists.

def Specifies that frames that have an unknown destination address (default frames) should not be sent "single route."

Single route refers to the case where the path to the destination contains parallel IBM bridges. Only bridges that are configured as single route bridges forward single route broadcast frames.

If this parameter *is not* specified and ROUTE.NLM encounters a frame that has a destination address that

is not in its table, ROUTE.NLM sends the frame single route.

If this parameter *is* specified, and ROUTE.NLM encounters a frame that has a destination address that is not in its table, ROUTE.NLM sends the frame as an all-routes broadcast. In this case, the destination could get multiple copies of the frame.

If board=*nn* is already loaded, this parameter changes all subsequent frames that have an unknown destination address from a single route to an all-routes broadcast. Sending default frames as an all-routes broadcast allows for all paths to the destination to be discovered.

gbr Specifies that general broadcast frames should be sent across all known bridges in the path to the destination.

If this parameter is not specified, all general broadcast frames are transmitted with the single route broadcast bit set to 1 in the source routing field. Note that a general broadcast frame is specified by a destination address of FFFFFFFFFFFFh or C000FFFFFFFFh.

If board=*nn* is already loaded, this parameter changes all subsequent general broadcasts from a single-route to an all-routes broadcast.

mbr Specifies that multicast broadcast frames are to be sent across all known bridges in the path of the destination.

If this parameter is not specified, all multicast broadcast frames are transmitted with the single route broadcast bit set to 1 in the source routing field. A multicast broadcast frame is specified by a destination address of C000*xxxxxxxx*h, where *xxxxxxxx* is any hexadecimal digit other than FFFFFFFFh.

If board=*nn* is already loaded, this parameter changes all subsequent multicast broadcasts from a single-route to an all-routes broadcast.

remove Specifies a NODE address that requires dynamic route discovery. Normally, all source routing information is built into a table as frames are received from that node. This parameter will remove the node from the table, forcing ROUTE.NLM to send a default frame the next time the OS sends a frame to the node.

The *xx* portion of the parameter is a 12-digit (6-byte) hexadecimal number. If you enter less than nine digits, ROUTE.NLM prefixes 4000h to the number. For example, if you specify remove=2, ROUTE.NLM changes the number to 400000000002h. If you really wanted to specify the value 2, you would specify remove=000000000002.

This parameter is only valid if the board=*nn* parameter is already loaded. It is useful if a bridge between the server and the node specified has gone down and an alternate route is available. Removing the node forces ROUTE.NLM to rediscover the route through the alternate path.

rsp Specifies how the server should respond to a request that has been broadcast. Valid settings for this parameter are:

rsp=NR The server should respond directly to all broadcast request. The response, however, is not a broadcast response. It is the default response mode.

rsp=AR The server should respond to requests with an all-routes broadcast frame.

rsp=SR The server should respond to requests with a single-route broadcast frame.

time Specifies the number of seconds (in decimal) that ROUTE.NLM should wait before updating its source routing table. If an entry in the source routing table has not been used for the amount of time specified by time=*ss*, ROUTE.NLM automatically updates the route as soon as a new route is available.

In a network configured with multiple paths to the same nodes, ROUTE.NLM automatically chooses the shortest route to the destination node. However, if the old route has not been used for the specified time period, the time parameter forces ROUTE.NLM to update, even if it is a longer route. By setting this parameter appropriately, ROUTE.NLM can perform dynamic route discovery if an IBM bridge goes down but an alternate path is still available.

The default value for time is 3 seconds. If board=*nn* has already been loaded, specifying the time parameter changes the time ROUTE.NLM waits before updating its table.

unload Specifies that board=*nn* should be unloaded. That is, source routing is no longer required, and all frames are to be sent with no source routing information.

The unload option is only valid if the board has been previously loaded. Note that unloading and then reloading a particular board clears the source routing table and causes ROUTE.NLM to rediscover all routes in the system.

Source Routing for the 16-Bit DOS and OS/2 Clients

Client source routing for the FDDI and token-ring adapters uses a memory-resident utility. This utility automatically builds the source routing information on all frames transmitted by the client workstation, and it allows the workstation to transmit and receive packets across the IBM Source Routing Bridge.

The DOS and OS/2 utilities were written to be as generic as possible. Currently, only token-ring and FDDI topologies support source routing. Once loaded, the function of the source routing utility is transparent to the NetWare shell and to any applications that might be running on the workstation.

The MSM handles all source routing for the driver and no special considerations are required.

The following sections discuss DOS and OS/2 client source routing. Both sections refer to the following figures (Figures 1 and 2) for the location of the source routing information in an 802.5 frame.

Figure 1
Token-Ring
Packet Type

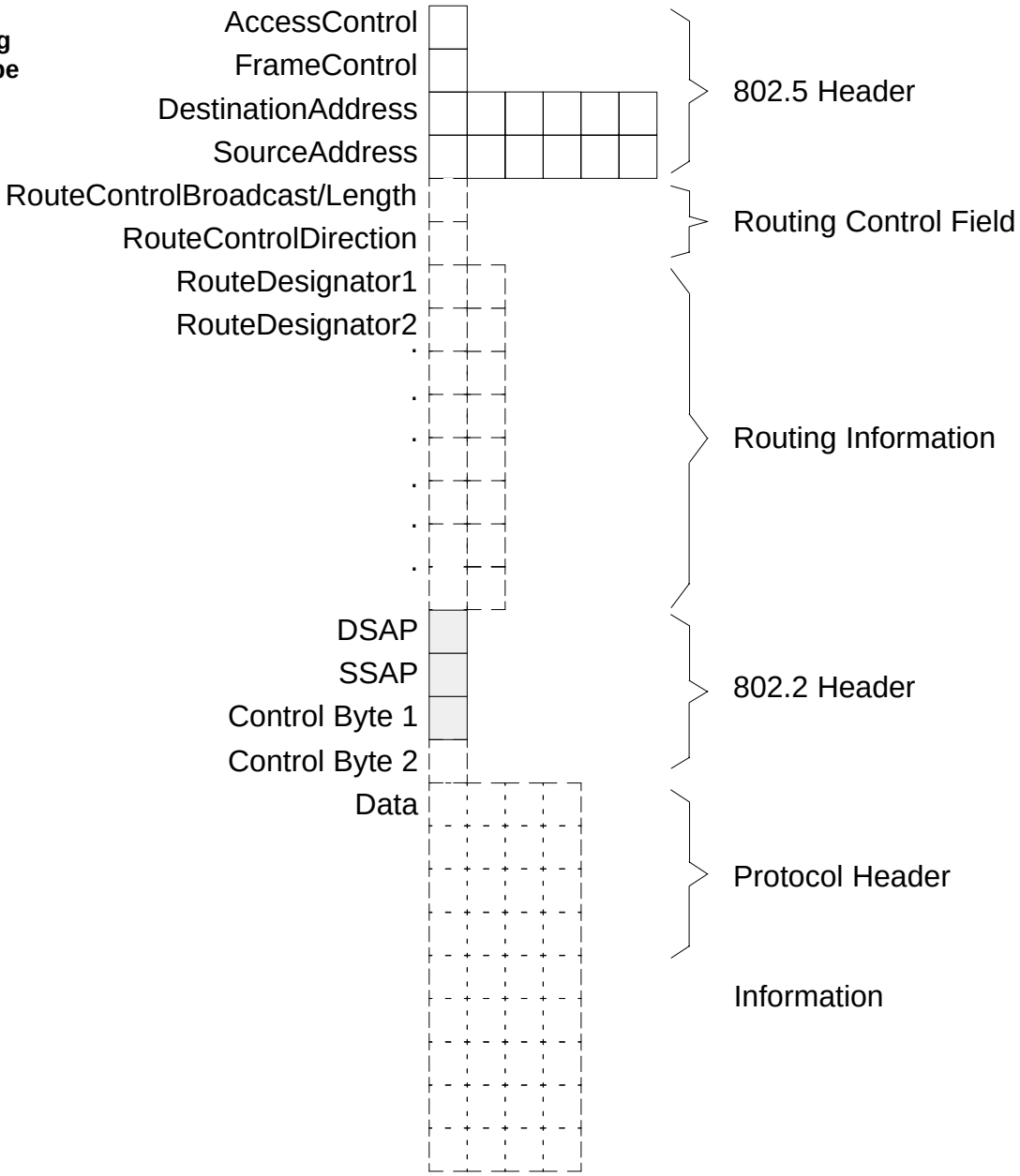
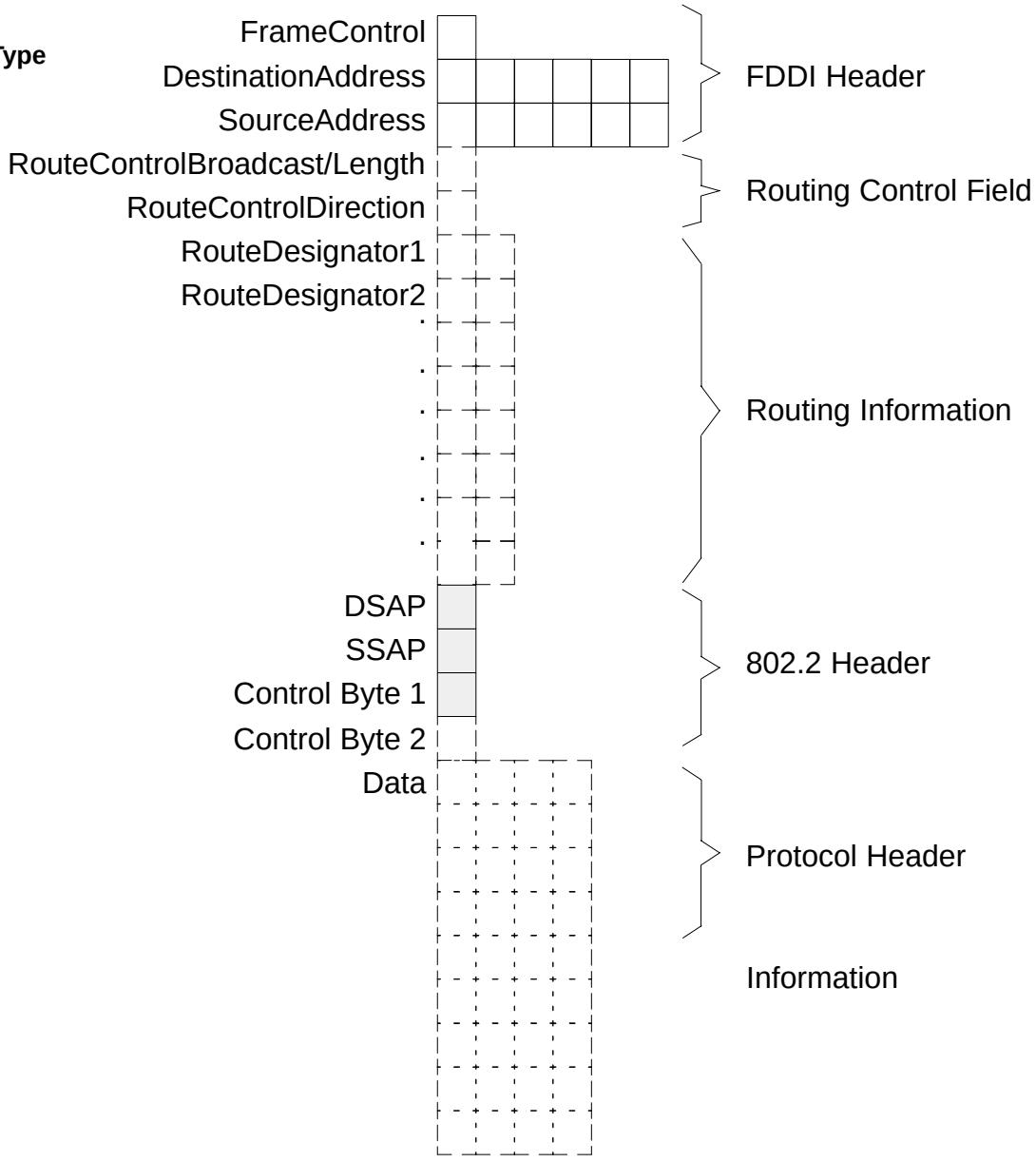


Figure 2
FDDI
Packet Type



DOS Client Source Routing

DOS Source Routing Utility Requirements

The source routing utility for the DOS client is named ROUTE.COM. ROUTE.COM requires that the LAN driver installed on the DOS workstation support source routing.

Configuration Table Changes

You must initialize the *SourceRouteHandler* field of the configuration table with the offset and segment location of a routine that issues a RET FAR instruction. (The first two bytes of the *SourceRouteHandler* field hold the offset address, and

the last two bytes hold the segment address.) When ROUTE.COM is loaded, it will replace the address of this routine with its own entry point.

Calling the DOS Source Routing Send and Receive Routines

Each time the MLID sends or receives a packet, it must call the source routing routine at the address contained in the *SourceRouteHandler* field of the configuration table.

Calling the Send Routine. Before it calls *DriverSend*, the MLID calls the source routing send routine with the following parameters:

DS:AX a pointer to destination address.
CX is equal to 0000.
Interrupts are disabled and remain disabled.

On return from the source routing send routine, the following conditions are in effect:

DX:AX a pointer to source routing field (if CX <> 0).
CX source routing field size.
Interrupts are disabled
BP, BX, DI, SI, DS, and ES are preserved
cld is in effect

Example

```
; CS:BP points to the configuration table
; DS:SI points to the send ECB
    sub    cx, cx
    lea    ax, ECB_ImmediateAddress[si]
    call   far ptr cs:SourceRouteHandler[bp]
```

The DOS workstation driver uses the information in CX and DX:AX to insert the source routing information into the appropriate place in the frame prior to placing it on the LAN medium. See Figures 1 and 2 regarding the location of the source routing information in the frame. If, on return, CX = 0000, there is no source routing information for the destination address.

Calling the Receive Routine. The MLID calls the receive routine with the following parameters:

DS:CX a pointer to source routing field (*CX* cannot be 0).
DS:SI a pointer to source address (with valid RII bit).
Interrupts are disabled, and remain disabled.

On return from the source routing receive routine, the following conditions are in effect:

CX destroyed.
Interrupts are disabled.
AX, BP, BX, DX, DI, DS:SI, and ES are preserved.
cld is in effect.

Example

```
; CS:BP points to the configuration table
; DS:SI points to the source address in the receive frame
    lea    cx, 06[si]
    call   cs:SourceRouteHandler[bp]
```

The MLID does not need to use any of the information returned.

OS/2 Client Source Routing

OS/2 Source Routing Utility Requirements

The source routing utility for the OS/2 client is named ROUTE.SYS. ROUTE.SYS requires that the LAN driver installed on the OS/2 workstation support source routing.

Configuration Table Changes

The HSM sets the *SourceRouteHandler* configuration table field initially to zero. The MSM then tests if SOURCE.SYS has been loaded (in other words, this field is not equal to zero) before it calls the far address in *SourceRouteHandler*.

Calling the OS/2 Source Routing Send and Receive Routines

Each time the MLID sends or receives a packet, it must call the source routing routine at the address contained in the *SourceRouteHandler* field of the configuration table.

Calling the Send Routine. Before it calls *DriverSend*, the MLID calls the source routing send routine with the following parameters:

AX the board number.
ES:SI a pointer to the destination address.
CX is equal to 0
Interrupts are disabled.

On return from the source routing send routine, the following conditions are in effect:

ES:SI a pointer to the beginning of the routing control field, if *CX* does *not* return zero.
CX the size (in bytes) of the *Routing Control* field and the routing information.

Interrupts are disabled.
cld is in effect.
BP, BX, CS, DI, SP, SS are preserved.

Example

```
;DS:BX points to the configuration table
;ES:SI points to the send ECB
    xor     cx, cx
    lea     si, ECB ImmediateAddress[si]
    call    [bx].SourceRouteHandler
```

The MSM uses the information in CX and ES:SI to insert the source routing information into the appropriate place in the frame prior to calling the HSM's *DriverSend*. See Figures 1 and 2 regarding the location of the source routing information in the frame. If, on return, CX = 0000, there is no source routing information for the destination address.

Calling the Receive Routine. The MLID calls the receive routine with the following parameters:

AX the board number.
ES:CX a pointer to the first byte of the *Routing Control* field of the received frame. If CX=0, the HSM must obtain the route for the send.
ES:SI a pointer to the source address of the received frame.
Interrupts are disabled.

On return from the source routing receive routine, the following conditions are in effect:

Interrupts are disabled.
cld is in effect.
BP, BX, CS, DI, SP, SS are preserved.

Example (token-ring)

```
;DS:BX points to the configuration table
;ES:SI points to the source address in the
;receive frame
    lea  cx, [si] + 6                   ;token-ring
    call [bx].SourceRouteHandler
```

The MLID does not need to use any of the information returned.

❑

Index

S

T

U