9

Source Routing Support

Introduction

When a driver supports source routing, the source routing module, ROUTE.NLM, must be called each time the driver 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. This chapter is provided for information purposes only.**

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. A packet that travels across a bridge or bridges arrives at its destination containing the routing information for the route it travelled. Subsequent communications between the source and destination nodes of that packet travel the same path.

The NetWare operating system handles packet routing between multiple physical segments above the MAC layer, so source routing information is not required. When NetWare must pass packets across an IBM bridge the source routing information must be added. The ROUTE.NLM loadable module provides this capability.

ROUTE.NLM can be used for any Token-Ring driver that supports source routing. Once loaded, the function of ROUTE.NLM is transparent to the operating system and any application NLMs that may be running on the file server. This NLM automatically builds the source routing information into all frames that are transmitted by the NetWare operating system.

ROUTE.NLM requires the following:

- NetWare v3.10 or later
- At least one adapter/driver combination installed and configured to support Token-Ring source routing

This chapter first explains the details of the interface between the driver and the source routing module. At the end of the chapter the loading options for ROUTE.NLM are described.

Driver Source Routing Support

When a driver supports source routing, the *MLIDRouteHandler* field, offset 46h of the configuration table, must be initialized with the offset of a routine that issues a RET instruction. When ROUTE.NLM is loaded it replaces the pointer to this routine 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	contains the board number
ECX	is 0
ESI	contains the destination address

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

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

The server driver (media module) uses the information in ECX and ESI to insert the source route into the appropriate place in the frame prior to placing it on the media. If ECX 0, there is no source routing information for the destination address.

Note the following example:

```
; 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 Source Routing module must also be called by the driver when it receives a valid frame. This interaction is defined below:

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

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

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

Source Routing Programmatic Interface

The driver may also access the source routing module to dynamically reconfigure it. This is done via *SRControl*, a public symbol exported by ROUTE.NLM. This function is used by placing a function code in AL, the board number in BL, and making the call.

The example below shows this procedure and the available functions are described in detail on the following pages.

```
mov      al, FunctionCode
mov      bl, BoardNumber
call     SRControl
```

Note: In all cases the state of the 80386 flag register is not defined on entry to *SRControl*. Also, all drivers that support source routing must notify ROUTE.NLM when they are unloaded using the Unload Board Number function described here, even if none of the other *SRControl* functions are used. The TOKENTSM.NLM takes care of this notification.

SRControl Completion Codes

On return from *SRControl* EAX contains either 0, if the function completed successfully, or one of the following:

Table 9.1 Source Routing Module Completion Codes

Completion Code	Description
FFFF FF81	BadCommand
FFFF FF82	BadParameters
FFFF FF85	ItemNotPresent
FFFF FF89	OutofResources

Load Board Number

On Entry

AL = 0	load-board-number function code
BL	contains the board number to load

On Return

EAX	contains the completion code
EBX	contains 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
Note	EBP, EDI, and ESI are preserved

Description This routine must not be called at interrupt time.

Example

```
mov   al, 0                          ; al = load board function code
mov   bl, [ebp].MLIDBoardNumber      ; bl = board number being loaded
sub   esi, esi                       ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Unload Board Number

On Entry

AL = 1	unload-board-number function code
BL	contains the board number to unload

On Return

EAX	contains the completion code
EBX	contains the input board number
ECX	is destroyed
EDX	is destroyed
Interrupts	are enabled
Flags	cld is in effect; the 80386 status flag is set according to EAX
Note	EBP, EDI and ESI are preserved

Description

This routine must not be called at interrupt time.

Example

```
mov   al, 01                        ; al = unload board function code
mov   bl, [ebp].MLIDBoardNumber     ; bl = board number being unloaded
sub   esi, esi                      ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Clear Source Routing Table

On Entry

AL = 2	clear-source-routing-table function code
BL	contains the board number to clear

On Return

EAX	contains the completion code
EBX	contains 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
Note	EBP, EDI and ESI are preserved

Description This routine may be called at process or interrupt time.

Example

```
mov   al, 02                        ; al = clear table function code
mov   bl, [ebp].MLIDBoardNumber     ; bl = board number
sub   esi, esi                      ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Change Unknown Destination Address Route

On Entry

AL = 3	change-unknown-destination-address-route function code
AH	contains a new unknown-destination-address control field: 0C2h = single route broadcast 082h = all routes broadcast
BL	contains the board number to change

On Return

EAX	contains the completion code
EBX	contains the input board number
ECX	is destroyed
EDX	is destroyed
CH	contains the old unknown-destination address
Interrupts	state is not changed
Flags	cld is in effect; the 80386 status flag is set according to EAX
Note	EBP, EDI, and ESI are preserved

Description This routine may be called at process or interrupt time.

Example

```
mov   al, 03                          ; al = change unkown destination
                                      ; address function code
mov   bl, [ebp].MLIDBoardNumber       ; bl = board number
sub   esi, esi                        ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Change General Broadcast Route

On Entry

AL = 4	change-general-broadcast-route function code
AH	contains a new general broadcast route: 0C2h = single route broadcast 082h = all routes broadcast
BL	contains the board number to change

On Return

EAX	contains the completion code
EBX	contains the input board number
ECX	is destroyed
EDX	is destroyed
CH	contains the old general broadcast route
Interrupts	state is not changed
Flags	cld is in effect; the 80386 status flag is set according to EAX
Note	EBP, EDI and ESI is preserved

Description This routine may be called at process or interrupt time.

Example

```
mov   al, 04                          ; al = change general broadcast
                                      ; route function code
mov   bl, [ebp].MLIDBoardNumber       ; bl = board number
sub   esi, esi                        ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Change Multicast Broadcast Route

On Entry

AL = 5	change-multicast-broadcast-route function code
AH	contains the new multicast broadcast route 0C2h = single route broadcasts 042h = all routes broadcast
BL	contains the board number to change

On Return

EAX	contains the completion code
EBX	contains the input board number
ECX	is destroyed
EDX	is destroyed
CH	contains the old multicast broadcast route
Interrupts	state is not changed
Flags	cld is in effect; the 80386 status flag is set according to EAX
Note	EBP, EDI and ESI are preserved

Description This routine may be called at process or interrupt time.

Example

```
mov   al, 05                          ; al = change multicast broadcast
                                      ; route function code
mov   bl, [ebp].MLIDBoardNumber       ; bl = board number
sub   esi, esi                        ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Change Broadcast Response Type

On Entry

AL = 6	change-broadcast-response-type function code
AH	determines how the server should respond to broadcast requests (see Description)
BL	contains the board number to change

On Return

EAX	contains the completion code
EBX	contains the input board number
ECX	is destroyed
EDX	is destroyed
CH	contains the old broadcast response type
Interrupts	state is not changed
Flags	cld is in effect; the 80386 status flag is set according to EAX
Note	EBP, EDI and ESI are preserved

Description

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

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

82h means that the server should respond to broadcast requests with an all routes broadcast response.

C2h means that the server should respond to broadcast requests with a single route broadcast response.

Example

```
mov   al, 06                        ; al = change broadcast response
                                    ; type function code
mov   bl, [ebp].MLIDBoardNumber     ; bl = board number
sub   esi, esi                      ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Change Source Routing Update Table Timer

On Entry

AL = 7	the change-source-routing-update-table-timer function code
AH	contains a new timer value in seconds
BL	contains the board number to change

On Return

EAX	contains the completion code
EBX	contains the input board number
ECX	is destroyed
EDX	is destroyed
CH	contains the old timed value in seconds
Interrupts	state is not changed
Flags	cld is in effect; the 80386 status flag is set according to EAX
Note	EBP, EDI and ESI are preserved

Description

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 will automatically update the route as soon as a new route is available.

Example

```
mov   al, 07                          ; al = change source routing update
                                      ; table timer function code
mov   bl, [ebp].MLIDBoardNumber       ; bl = board number
sub   esi, esi                        ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Remove Node from Source Route Table

On Entry

AL = 8	remove-node-from-source-route-table function code
BL	contains the board number from which to remove the node
EDI	contains the address of the 6 byte node address to be removed

On Return

EAX	contains the completion code
EBX	contains the input board number
ECX	is destroyed
EDX	is destroyed
Interrupts	are disabled
Flags	cld is in effect; 80386 status flag is set according to EAX
Note	EBP, EDI, and ESI are preserved

Description This routine may be called at process or interrupt time.

Example

```
mov   al, 08                          ; al = remove node from source route
                                      ; table function code
mov   bl, [ebp].MLIDBoardNumber       ; bl = board number
sub   esi, esi                        ; esi = 0
call  [ebp].MLIDRouteHandler
```

By setting ESI = 0 the driver is telling ROUTE.NLM to parse the function code in register AL.

Loading ROUTE.NLM

ROUTE.NLM may be used re-entrantly to 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]
```

where the load options are:

load is the NetWare console command to load NLMs.

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

board is 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 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 there are parallel IBM Bridges in the path to the destination. Only bridges that are configured as single route bridges will 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 will send the frame as an all routes broadcast. In this case it is possible that the destination

may get multiple copies of the frame.

If board=nn is already loaded, this parameter will change all subsequent frames that have an unknown destination address from single route to all routes broadcast. Sending default frames as 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 will be 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 will change all subsequent general broadcasts from single route to 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 will be 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 C000xxxxxxxxh, where xxxxxxxx is any hexadecimal digit other than FFFFFFFFh.

If board=nn is already loaded, this parameter will change 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 (six byte) hexadecimal number. If you enter less than nine digits, ROUTE.NLM will prefix 4000h to the number. For example, if you specify remove=2, ROUTE.NLM will change 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 will automatically update 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 will wait 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 re-loading a particular board will clear the source routing table and cause ROUTE.NLM to rediscover all routes in the system.