

Configuration and Verification of Distributed Features on Cisco 75xx and 76xx Routers

Contents

[Introduction](#)

[Prerequisites](#)

[Requirements](#)

[Components Used](#)

[Conventions](#)

[Distributed Features](#)

[Distributed MLPPP](#)

[Distributed LFI](#)

[Differences between dMLP and dLFIoLL](#)

[Distributed MLFR](#)

[Distributed DDR](#)

[Distributed Features Prerequisites and Restrictions](#)

[Number of Bundles and Links and Memory requirements](#)

[Hardware and Software MLPPP or MLFR on 7600 SIP Linecards](#)

[Life of a Packet](#)

[Rx Data Path](#)

[Tx Data Path](#)

[Reassembly](#)

[Configuring, Verifying, and Debugging Distributed Features](#)

[Configuring and Verifying dMFR](#)

[Configuring and Verifying dMLP/dLFIoLL](#)

[Configuring and Verifying dLFIoFR and dLFIoATM](#)

[Configuring and Verifying dDDR](#)

[Debugging dMLP and dDDR](#)

[Frequently Asked Questions](#)

[Debug Enhancements](#)

[NetPro Discussion Forums - Featured Conversations](#)

[Related Information](#)

Introduction

This document helps you to understand, configure, and verify these features:

- Distributed Multilink Point to Point Protocol (dMLP)
- Distributed Link Fragmentation and Interleaving (LFI)
- Distributed LFI over Leased Line (dLFIoLL)
- Distributed LFI over Frame Relay (dLFIoFR)
- Distributed LFI over ATM (dLFIoATM)

- Differences between Distributed MLP (dMLP) and dLFIoLL
- Distributed Multilink Frame Relay (dMLFR)
- Distributed Dial on Demand Routing (DDR)

Prerequisites

Requirements

Readers of this document should have knowledge of distributed features for Cisco 7500/7600/6500.

Components Used

The information in this document is based on these software and hardware versions:

- All Cisco 7500 and 7600 platforms
 - Note:** The information in this document also applies to 6500 platforms.
- Relevant Cisco IOS® software releases, which this table lists:

Distributed Features Support for Each Branch and Platform

Feature	Port Adapters (PAs) ¹ Supported	7500 Platforms		7600 Platforms	
		Major Cisco IOS Software Releases	Cisco IOS Releases (Interim)	Major Cisco IOS Software Releases	Cisco IOS Software Releases (Interim)
dMLP	Chan-PA PA-4T+ PA-8T	12.0T	12.0(3)T and later	12.2SX	
		12.0S			
		12.1			
		12.1T			
		12.2			
		12.2T			
		12.3			
		12.3T			
dLFIoLL	Chan-PA PA-4T+ PA-8T	12.2T	12.2(8)T and later	12.2SX	12.2(17)SXB and later
		12.3			
		12.3T			
		12.0S			
		12.0S			
dLFIoFR	Chan-PA PA-4T+ PA-8T	12.2T	12.2(4)T3 and later	12.2SX	12.2(17)SXB and later
		12.3			
		12.3T			

dLFIoATM	PA-A3 PA-A6	12.2T 12.3 12.3T	12.2(4)T3 and later	12.2SX	12.2(17)SXB and later
dMLFR	Chan-PA PA-4T+ PA-8T	12.0S 12.3T	12.0(24)S and later 12.3(4)T and later	12.2SX	12.2(17)SXB and later
QoS on dMLP	Chan-PA PA-4T+ PA-8T	12.0S 12.2T 12.3 12.3T	12.0(24)S and later 12.2(8)T and later	12.2SX	12.2(17)SXB and later
MPLS on dMLP MPLS on dLFIoLL	Chan-PA PA-4T+ PA-8T	12.2T 12.3	12.2(15)T11 and later 12.3(5a) and later	12.2SX	12.2(17)SXB and later
Distributed DDR	PA-MC-xT1 PA-MC-xE1 PA-MC-xTE1 PA-MCX-xTE1	12.3T	12.3(7)T and later		

Note: Be aware of this information:

1. These PAs support distributed features:

- CT3IP
- PA-MC-T3
- PA-MC-2T3+
- PA-MC-E3
- PA-MC-2E1
- PA-MC-2T1
- PA-MC-4T1
- PA-MC-8T1
- PA-MC-8E1
- PA-MC-8TE1+
- PA-MC-STM-1

2. Cisco IOS Software Release 12.1E supports these features on both 7500 and 7600 platforms.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, refer to the [Cisco Technical Tips Conventions](#).

Distributed Features

These features are explained in this document:

- Distributed MLP
- Distributed LFI
- Distributed LFI over Leased Line
- Distributed LFI over Frame Relay
- Distributed LFI over ATM
- Differences between dMLP and dLFIoLL
- Distributed MLFR
- Distributed Dialer
- Platforms and Line-cards which support distributed features

Distributed MLPPP

The Distributed Multilink Point to Point Protocol (dMLP) feature allows you to combine full or fractional T1/E1 lines in a line card (VIP, FlexWAN) on a Cisco 7500 or 7600 series router into a bundle that has the combined bandwidth of multiple links. You use a distributed MLP Bundle to do this. A user can choose the number of bundles in a router and the number of links per bundle. This allows the user to increase the bandwidth of network links beyond that of a single T1/E1 line without the need to purchase a T3 line. In non-dMLP, all of the packets are switched by the Route Processor (RP); therefore, this implementation impacts performance of the RP, with high CPU utilization for only a few T1/E1 lines running MLP. With dMLP, the total number of bundles that can be handled on the router is increased, as the data path is handled and limited by line card CPU and memory. dMLP allows you to bundle fractional T1/E1, starting from DS0 (64 Kbps) onwards.

Distributed LFI

The dLFI feature supports the transport of real-time traffic (such as voice) and non-real-time traffic (such as data) on lower-speed Frame Relay and ATM virtual circuits (VCs) and on leased lines without causing excessive delay to the real-time traffic.

This feature is implemented using multilink PPP (MLP) over Frame Relay, ATM, and leased lines. The feature fragments a large data packet into a sequence of smaller fragments, to enable delay-sensitive real-time packets and non-real-time packets to share the same link. The fragments are then interleaved with the real-time packets. On the receiving side of the link, the fragments are reassembled and the packet is reconstructed.

The dLFI feature is often useful in networks that send real-time traffic via Distributed Low Latency Queuing (such as voice) but have bandwidth problems. This delays the real-time traffic due to the transport of large, less time-sensitive data packets. You can use the dLFI feature in these networks, to disassemble the large data packets into multiple segments. The real-time traffic packets then can be sent between these segments of the data packets. In this scenario, the real-time traffic does not experience a lengthy delay while it waits for the low-priority data packets to traverse the network. The data packets are reassembled at the receiving side of the link, so the data is delivered intact.

The fragment size of the link is calculated based on the fragment delay on the multilink bundle, configured with the **ppp multilink fragment-delay *n*** command, where:

$$\text{fragment size} = \text{bandwidth} \times \text{fragment-delay} / 8$$

This fragment size represents IP payload only. It does not include the encapsulation bytes (fragment size = weight encapsulation bytes). The terms `weight` and `fragment size` are as seen in the output of the **show ppp multilink** command on the RP. If fragment delay is not configured, default fragment size is calculated for a maximum fragment-delay of 30.

Note: With links of varying bandwidth, the chosen fragment size is based on the link with the least bandwidth.

Distributed LFI over Leased Line

The dLFIoLL feature extends distributed link fragmentation and interleaving functionality to leased lines. Distributed LFI is configured with the **ppp multilink interleave** command on the multilink group interface. It is advisable that you use distributed LFI on multilink interfaces with bandwidth less than 768 Kbps. This is because the serialization delay for 1500 byte packets for bandwidth greater than 768 Kbps is within acceptable delay limits and does not need to be fragmented.

Distributed LFI over Frame Relay

The dLFIoFR feature is an extension of the Multilink PPP over Frame Relay (MLPoFR) feature. MLP is used for the fragmentation. This feature is similar to FRF.12, which supports fragmentation and can interleave high priority packets via Low Latency Queuing.

The **ppp multilink interleave** command on the Virtual-template is required to enable interleaving on the associated virtual-access interface. In addition to the enabling of distributed CEF switching on the serial interface, this command is a prerequisite for distributed LFI to work.

Note: Unless you are using Frame Relay to ATM internetworking, it is recommended that you use FRF.12 rather than dLFIoFR, because the bandwidth utilization is better with FRF.12

Distributed LFI over ATM

The dLFIoATM feature is an extension of the Multilink PPP over ATM (MLPoATM) feature. MLP is used for the fragmentation.

The **ppp multilink interleave** command on the Virtual-template is required to enable interleaving on the associated virtual-access interface. In addition to the enabling of distributed CEF switching on the serial interface, this command is a prerequisite for distributed LFI to work.

With dLFIoATM, it is very important that you select a fragment size which makes the packets to fit in ATM cells in such a way that they do not cause unnecessary padding in the ATM cells. For example, if the selected fragment size is 124 bytes, this means that an IP payload of 124 bytes would finally go as $124 + 10$ (MLP Header) + 8 (SNAP Header) = 142 bytes. It is important to note that first fragment would go out with $124 + 10 + 2$ (First Fragment PID Header size) + 8 = 144 bytes. This means that this packet will use three ATM cells to transfer the payload and, hence, would use the cell packed most efficiently.

Differences between dMLP and dLFIoLL

dMLP does not support fragmentation on the transmit side, whereas dLFIoLL does.

Note: Interleaving and fragmentation used with more than one link in the multilink bundle for voice traffic does not guarantee that the voice traffic received via multiple links in the bundle will be received in order. The correct ordering of voice is handled at the upper layers.

Distributed MLFR

The distributed MLFR feature introduces functionality based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16) to line card-enabled Cisco 7500 and 7600 series routers. The distributed MLFR feature provides a cost-effective way to increase bandwidth for particular applications because it allows multiple serial links to be aggregated into a single bundle of bandwidth. MLFR is supported on User-to-Network Interfaces (UNI) and Network-to-Network Interfaces (NNI) in Frame Relay networks.

The bundle is made up of multiple serial links, called bundle links. Each bundle link within a bundle corresponds to a physical interface. Bundle links are invisible to the Frame Relay data-link layer, so Frame Relay functionality can not be configured on these interfaces. Regular Frame Relay functionality that you want to apply to these links must be configured on the bundle interface. Bundle links are visible to peer devices.

Distributed DDR

The distributed DDR feature allows distributed switching on dialer interfaces. Without this feature, all dial-in traffic must be punted to the host for switching. With it, only control packets are sent up to the RP, whereas switching decision is done on the VIPs themselves after a connection has been established.

Both Legacy Dialer configuration and Dialer Profile configuration are supported only with PPP encapsulation. MLP on Dialer interfaces is also supported. QoS is not supported with distributed switching on Dialer interfaces.

Distributed Features Prerequisites and Restrictions

Prerequisites

These are general prerequisites for all of these distributed features:

- Distributed Cisco Express Forwarding (dCEF) switching must be enabled globally.
- dCEF switching must be enabled on the member serial interface, which are part of MLP bundle.
- dCEF switching must be enabled on physical link of dLFioFR and dLFioATM interfaces.
- Interleave configuration is required to distribute LFioFR and LFioATM.
- Configure required bandwidth on the Virtual-template interface for dLFioFR and dLFioATM interfaces.
- When PPP debugs are enabled on the RP, you might observe the MLP: Forwarded to wrong interface message on the Route Switch Processor (RSP). Because this message is confusing and unwanted especially if the message is for Cisco Discovery Protocol (CDP) packets you must configure **no cdp enable** on the member links of bundle.
- All of the member links of the bundle should have the keepalive enabled.

Restrictions

These are general restrictions for all of these distributed features:

- T1 and E1 lines can not be mixed in a bundle.
- The maximum supported differential delay is 30 ms.
- All lines in a bundle must reside on the same port adapter (PA).
- Hardware compression is not supported.
- VIP or FlexWAN CEF is limited to IP only; all other protocols are sent to the RSP.
- Fragmentation is not supported on the transmit side for dMLP and dMLFR.
- Many of the older queuing mechanisms are not supported by dLFI. These mechanisms include:
 - Fair-queuing on a virtual template interface
 - Random-detect on a virtual template interface
 - Custom queuing
 - Priority queuing
- Fair queuing, random detection (dWRED), and priority queuing can be configured in a traffic policy with the Modular QoS CLI.
- Only one link per MLP bundle is supported, when you are using dLFioFR or dLFioATM. If more than one link is used in an MLP bundle when using dLFioFR or dLFioATM, dLFI is automatically disabled. When using dLFI over leased lines, more than one link can be configured with dLFI in the MLP bundle.
- With dLFioATM, only aal5snap and aal5mux are supported. Encapsulation aal5nlpid and aal5ciscoppp are not supported.
- Only Voice over IP is supported; Voice over Frame Relay and Voice over ATM are not supported by the dLFI feature.
- Compressed Real-Time Protocol (CRTP) configuration should not be configured on the multilink interface, when you use this feature combination:
 - Multilink interface with LFI enabled
 - Multilink bundle has more than one member links

- QoS policy with priority feature is enabled on the multilink interface

With dMLP and dLFI configuration, priority packets do not carry MLP header and sequence number, and MLP will distribute the priority packets across all member links. As a result, packets that are compressed by CRTP may arrive out-of-order at the receiving router. This prohibits CRTP from decompressing the packet header and forces CRTP to drop the packets.

Recommendations

It is recommended that member links in a bundle have the same bandwidth. If you add unequal bandwidth links to the bundle, it will lead to lot of packet reordering, which will cause overall bundle throughput to diminish.

VIP2-50 (with 8 MB SRAM) or higher is recommended to be used with these distributed features.

Number of Bundles and Links and Memory requirements

Refer to [Distributed Multilink Point-to-Point Protocol for Cisco 7500 Series Routers](#).

Hardware and Software MLPPP or MLFR on 7600 SIP Linecards

MLP and MLFR can be either software- or hardware- based. In hardware based MLP or MLFR, Freedm provides the multilink functionality and the MLP headers are added by Freedm Chip. In software based MLP or MLFR, the SIP line card CPU provides the multilink functionality (which is similar to the VIP and FlexWAN implementations).

These are the limitations and conditions to run hardware based MLP or MLFR.

- There can be only a maximum of 336 bundles per line card and 168 bundles per Security Posture Assessment (SPA) (Freedm).
- There can be only a maximum of 12 DS1/E1 per bundle.
- All the links should belong to the same SPA (Freedm).
- All links in the bundle should operate at the same speed.
- The TX Fragment Size can be 128, 256, or 512. A CLI configured fragment size is mapped to the nearest supported fragment size.

```
IF (0 < cli_fragment_size 6 < 256)
  configured_fragment_size = 128
Else IF (cli_fragment_size < 512)
  configured_fragment_size = 256
Else
  configured_fragment_size = 512
```

- The RX Fragment Size can be 1 to 9.6 KB.
- Cisco proprietary format can not be supported (MLFR).

In hardware LFI, if there is only one link in the bundle and if that is DS1/E1 then Fragmentation and Interleaving will be done by Freedm.

The output of **show ppp multilink** shows whether hardware implementation is running.

```
Multilink1, bundle name is M1
Bundle up for 00:14:51
Bundle is Distributed

0 lost fragments, 0 reordered, 0 unassigned
0 discarded, 0 lost received, 1/255 load
Member links: 1 active, 0 inactive (max not set, min not set)
Se6/1/0/1:0, since 00:14:51, no frags rcvd
Distributed fragmentation on. Fragment size 512. Multilink in Hardware.
```

If multilink is software- based then the **show ppp multilink** output will not have Multilink in Hardware in the output.

Life of a Packet

Rx Data Path

1. Packet received by driver.
2. Encapsulation is checked: as follows
 - a. Basic encapsulation:
 - . In dMLP, the encapsulation type for the ingress interface is ET_PPP.
 - b. In dMLFR, the encapsulation type for the ingress interface is ET_FRAME_RELAY.
 - c. In dLFioLL, the encapsulation type for the ingress interface is ET_PPP.
 - d. In dLFioFR, the encapsulation type for the ingress interface is ET_FRAME_RELAY.
 - e. In dLFioATM, the encapsulation type for the ingress interface is ET_ATM.
 - f. In dDialer, encapsulation type is ET_PPP.
 - b. Additional encapsulation processing:

For ET_PPP, the NLPID is sniffed out.

- For dMLP, the NLPID is MULTILINK.
- For dLFioLL, there are two things to consider:
 - a. VoIP packets These do not have an MLP header and have an NLPID that indicates IP.
 - b. Data Packets The NLPID is MULTILINK.
- For dDialer, the packets will not have an MLP header and have an NLPID that indicate IP.

Note: In this case, you may configure dCRTP (Distributed Compressed Real-Time Protocol). If so, the header is decompressed before further processing.

3. For ET_FRAME_RELAY, if the link on which the packet is received is configured for dMLFR then the packet is processed for dMLFR
4. For dLFioFR and dLFioATM, the encapsulation type is ET_FRAME_RELAY and ET_ATM, respectively; but within that there is a PPP header. The PPP header, as with dLFioLL, will indicate whether the packet is a voice packet or a data packet. If dCRTP is configured, the header is decompressed before further processing. Voice packets are switched immediately.

A fragmented Data packet will have to be reassembled before it is switched.

With ET_PPP, you might come across PPP link packets; and with ET_FRAME_RELAY, you might come across MLFR control packets. All of these control packets are punted to RP for processing.

5. Depending upon the aforementioned decoding, the packet is checked for the type of switching it requires. The link type will determine whether the packet is to be IP-switched or MPLS-switched. The packets are then given to respective switching functions.
6. With bundling in conjunction with distributed features, the IP turbo fast switching vector is stolen. This is done because the packet is received on the member link; however, it must be treated such that it is received on the bundle.

You also need to check for control packets which are punted to the host. Mainly in dMLFR, there are Local Management Interface (LMI) packets which are *not* MLFR control packets. For these, a different part of the dLCI number space is used. Whenever the dLCI is decoded to fall in this space, the packet is punted up to the host, because it is recognized to be an LMI packet.

VoIP packets (queued in Low Latency Queue) are just switched out without the addition of the MLP header. The distributed features can receive and reassemble packets, when fragmented data packets are received. The reassembly process is explained in a later section.

If the packet must be tag-switched, it is then passed to the tag-switching routine, in dMLP. Otherwise, if it is to be IP-switched, it is passed to the IP-switching routine.

Note: All non-IP packets are punted to host, in dMLFR.

7. **IP:** The IP-switching function is common to all packets. It does mainly three things:
 - Do the necessary processing of the packets, in case any features are configured. Also, when Distributed Dialer is used, do idle timer updates here when an interesting packet is received. Refer to [dialer idle-timeout \(interface\)](#), [dialer fast-idle \(interface\)](#), and [Configuring a Dialer Profile](#) for details of the **idle timer** configuration parameter.

On 75xx routers, adjacency will indicate the `tx_acc_ptr` for the egress interface. If the egress interface is a virtual access interface, the `tx_acc_ptr` is NULL. In this case, fix up the encapsulation and get the `tx_acc_ptr` from the `fib_hwidb`. This lookup and encapsulation fix up is necessary in dLFIoFR and dLFIoATM. In dLFIoLL, the link is treated as part of a Multilink bundle.

Note: TTL for the packet is adjusted here, and the check for IP fragmentation is made. The `mci_status` is set to `RXTYPE_DODIP` for all packets.

8. With the switching decision made, the packet is ready to be shipped out from the interface. The interface is checked to determine if it supports local switching. If it does, it is sent out directly via fastsend. Otherwise, an attempt is made to route-cache switch it.

Note that, in case QoS is configured for the interface, the local switching vector is stolen by QoS. HQF will enqueue the packet and further process the packet, before it is finally sent out of the interface. This is the case with dLFI. For dLFI, fragmentation and interleaving is set. QoS handles the invocation of our fragmentation routine and interleaves the fragmented packets with the voice packets which will be queued in the priority queue (if LLQ is configured). This ensures that VoIP packets do not suffer from the delay required to ship huge data packets through the link.

Tx Data Path

The `vip_dtq_consumer` gets the packet and gets the interface number, from which it gets the `idb`. The fastsend routine that corresponds to the `idb` is called:

i) Fastsend

1. In dMFR, the `fr_info` structure is retrieved from the table which matches the `if_index` to the `fr_info`. Control Packets are just sent out. The frame header will give the dLCI, which will help you determine whether this is a LMI packet or a data packet. The `dldci` field in the frame header is overwritten with the dmfr sequence number. Separate sequence numbers are used for LMI and data packets.

Note: Separate sequence numbers are used for separate dLCIs.

2. In dMLP, control packets are sent with priority set to high. With data packets, if dCRTP is configured, the header is compressed. The VIP MLP header which includes the sequencing info is added and sent out of the member links.
3. In dLFI, HQF intercepts the packets to be sent through the interface. If it is a voice packet, the voice packet is placed in the priority queue (if LLQ is configured) and is sent out of the interface without the MLP encapsulation. With data packets, it calls the dLFI fragmentation code, which returns fragments to QoS code, which are then interleaved with the priority traffic so that the delay requirements of the voice traffic is met. Also, if dCRTP is configured, only the header for the voice packet is compressed. Data packet headers are left as they are.
4. In dDialer, the packet is classified in order to reset the idle-timer of the output link before the packet is sent out. This is done after the output link is chosen, in the event that several links are bound to the same dialer. No header is added to dialer packets. Thus, sequencing and re-assemble of packets are not supported on Dialer interfaces.

Note: In dMLP, dDialer, dMLFR, and dLFI with several links, the physical link on which the traffic is forwarded depends on the congestion of the link. If the link is congested, move to the next link and so on. (dMLFR, dMLP without QoS, and dDialer features also choose links based on the number of bytes being put on the link. It chooses the next link, if the current link has already transmitted its quota of bytes, on a round-robin basis. This quota is decided by the `frag_bytes` for the link. For Dialer member interfaces, `frag_bytes` is set to default value of the interface bandwidth.)

Note: In HQF configurations on the interfaces of the egress VIP, HQF steals the `dtq_consumer` vector. The packet DMA'd to the egress VIP first goes through the HQF check. If QoS is configured on the egress interface, HQF kicks in to process the packet, before

the packet is fastsent out of the interface.

Reassembly

Plain dDialer interfaces do not support reassembly and sequencing. To enable this on dialer interfaces, MLP over dialer interfaces will have to be configured. If this is done, the Rx and Tx path are identical to dMLP paths. When packets are received, the sequence number is checked against the expected sequence number.

- If the sequence numbers match:
 1. If the packet is an unfragmented packet then reassembly is not required. Proceed with further switching steps.
 2. If the packet is a fragment, then check the begin and end bits and construct the packet as and when the fragments are received.
- If the sequence numbers do not match:
 1. If the sequence number is within the expected window of sequence numbers then put it in the sorted unassigned fragment list. Later, when an expected sequence number is not received, this list is checked, in case the packet was stored here.
 2. If the sequence number is not within the window, discard it and report received lost fragment. If a time-out occurs later while waiting for this packet, the receiver is resynced, and it begins again with the next packet received.

In all of those cases, a correctly ordered packet stream is sent out of this interface. If fragments are received, a complete packet is formed and then sent out.

Configuring, Verifying, and Debugging Distributed Features

This section covers the **show** and **debug** commands that are available to verify and debug each of the distributed features.

Configuring and Verifying dMFR

MFR Sample Configuration

```
interface MFR1
 no ip address

interface MFR1.1 point-to-point
 ip address 181.0.0.2 255.255.0.0
 frame-relay interface-dlci 16
```

Note: The MFR interface is like another FR interface and hence supports most of FR configuration.

```
interface Serial5/0/0/1:0
 no ip address
 encapsulation frame-relay MFR1
 tx-queue-limit 26
```

```
interface Serial5/0/0/2:0
 no ip address
 encapsulation frame-relay MFR1
 tx-queue-limit 26
```

```
interface Serial5/0/0/3:0
 no ip address
 encapsulation frame-relay MFR1
```

Verify MFR Bundle Status on the RP

```
show frame-relay multilink
```

Bundle: MFR1, State = **up**, class = A, fragmentation disabled

BID = MFR1

Bundle links:

Serial5/0/0/3:0, HW state = **up**, link state = **Add_sent**, LID = Serial5/0/0/3:0

Serial5/0/0/2:0, HW state = **up**, link state = **Up**, LID = Serial5/0/0/2:0

Serial5/0/0/1:0, HW state = **up**, link state = **Up**, LID = Serial5/0/0/1:0

This indicates that two interfaces are added correctly, and one interface has not yet negotiated the MLFR LIP messages.

To get more information on the MFR bundle and Member links, issue this command:

show frame-relay multilink mfr1 detailed

Bundle: MFR1, State = up, class = A, fragmentation disabled

BID = MFR1

No. of bundle links = 3, Peer's bundle-id = MFR1

Rx buffer size = 36144, Lost frag timeout = 1000

Bundle links:

Serial5/0/0/3:0, HW state = up, link state = Add_sent, LID = Serial5/0/0/3:0

Cause code = none, Ack timer = 4, Hello timer = 10,

Max retry count = 2, Current count = 0,

Peer LID = , RTT = 0 ms

Statistics:

Add_link sent = 35, Add_link rcv'd = 0,

Add_link ack sent = 0, Add_link ack rcv'd = 0,

Add_link rej sent = 0, Add_link rej rcv'd = 0,

Remove_link sent = 0, Remove_link rcv'd = 0,

Remove_link_ack sent = 0, Remove_link_ack rcv'd = 0,

Hello sent = 0, Hello rcv'd = 0,

Hello_ack sent = 0, Hello_ack rcv'd = 0,

outgoing pak dropped = 0, incoming pak dropped = 0

Serial5/0/0/2:0, HW state = up, link state = Up, LID = Serial5/0/0/2:0

Cause code = none, Ack timer = 4, Hello timer = 10,

Max retry count = 2, Current count = 0,

Peer LID = Serial6/1/0/2:0, RTT = 32 ms

Statistics:

Add_link sent = 0, Add_link rcv'd = 0,

Add_link ack sent = 0, Add_link ack rcv'd = 0,

Add_link rej sent = 0, Add_link rej rcv'd = 0,

Remove_link sent = 0, Remove_link rcv'd = 0,

Remove_link_ack sent = 0, Remove_link_ack rcv'd = 0,

Hello sent = 7851, Hello rcv'd = 7856,

Hello_ack sent = 7856, Hello_ack rcv'd = 7851,

outgoing pak dropped = 0, incoming pak dropped = 0

Serial5/0/0/1:0, HW state = up, link state = Up, LID = Serial5/0/0/1:0

Cause code = none, Ack timer = 4, Hello timer = 10,

Max retry count = 2, Current count = 0,

Peer LID = Serial6/1/0/1:0, RTT = 32 ms

Statistics:

Add_link sent = 0, Add_link rcv'd = 0,

Add_link ack sent = 0, Add_link ack rcv'd = 0,

Add_link rej sent = 0, Add_link rej rcv'd = 0,

Remove_link sent = 0, Remove_link rcv'd = 0,

Remove_link_ack sent = 0, Remove_link_ack rcv'd = 0,

Hello sent = 7851, Hello rcv'd = 7856,

Hello_ack sent = 7856, Hello_ack rcv'd = 7851,

outgoing pak dropped = 0, incoming pak dropped = 0

MFR Debug Commands

These debugs are useful to troubleshoot issues where a link does not get added to the bundle.

```
debug frame-relay multilink control
```

Note: When a specific MFR interface or Serial interface is not specified, this enables debug for all MFR links. This can be overwhelming, if the router has a large number of MFR links.

To debug MFR packets that are received at the RP, as well as to debug the MFR control activities, this debug is useful:

```
debug frame-relay multilink
```

Note: Under heavy traffic, this can overwhelm the CPU.

Verify dMLFR Bundle Status on the LC

```
show frame-relay multilink
```

Note: Currently, this is not available on the LC, but it will soon be added. Until then, use **show ppp multilink**.

```
Bundle MFR1, 2 members
  bundle 0x62DBDD20, frag_mode 0
  tag vectors 0x604E8004 0x604C3628
  Bundle hwidb vector 0x6019271C
  idb MFR1, vc 0, RSP vc 0
  QoS disabled, fastsend (mlp_fastsend), visible_bandwidth 3072
  board_encap 0x60577554, hw_if_index 0, pak_to_host 0x0
  max_particles 400, mrru 1524, seq_window_size 0x200
  working_pak 0x0, working_pak_cache 0x0
  una_frag_list 0x0, una_frag_end 0x0, null_link 0
  rcved_end_bit 1, is_lost_frag 0, resync_count 0
  timeout 0, timer_start 0, timer_running 0, timer_count 0
  next_xmit_link Serial0/0:1, member 0x3, congestion 0x3
dmlp_orig_pak_to_host 0x603E7030
dmlp_orig_fastsend 0x6035DBC0
bundle_idb->lc_ip_turbo_fs 0x604A7750
  0 lost fragments, 0 reordered, 0 unassigned
  0 discarded, 0 lost received
  0x0 received sequence, 0x58E sent sequence
DLCI: 16
  769719 lost fragments, 22338227 reordered,
                                0 unassigned
  27664 discarded, 27664 lost received
  0xF58 received sequence, 0x8DE sent sequence
timer count 767176
Member Link: 2 active
  Serial0/0:0, id 0x1, fastsend 0x60191E34, lc_turbo 0x601913AC, PTH 0x603E7030, OOF
0
  Serial0/0:1, id 0x2, fastsend 0x60191E34, lc_turbo 0x601913AC, PTH 0x603E7030, OOF
0
```

Configuring and Verifying dMLP/dLFloLL

Multilink PPP Configuration

```
interface Multilink1
  ip address 151.0.0.2 255.255.0.0
  no cdp enable
  ppp chap hostname M1
  ppp multilink
```

!

Sample configuration under the Serial interface:

```
interface Serial5/0/0/4:0
no ip address
encapsulation ppp
tx-queue-limit 26
no cdp enable
ppp chap hostname M1
ppp multilink group 1
!
interface Serial5/0/0/5:0
no ip address
encapsulation ppp
tx-queue-limit 26
no cdp enable
ppp chap hostname M1
ppp multilink group 1
!
```

Note: The **ppp chap hostname M1** command does not really mean that CHAP authentication is enabled. The string **M1** in this command acts as the end-point-discriminator and is only required if there is going to be more than one multilink bundle between the same two routers. In such a case, all links that belong to a bundle should have the same endpoint discriminator, and no two links that belong to a different bundle should have the same endpoint discriminator.

Optional Configuration Parameters

[no] ppp multilink interleave

This enables interleaving on the multilink bundle. This works in conjunction with Modular QoS CLI. High priority packets will be transmitted without the addition of the MLP sequence and header, while other packets will be fragmented and transmitted with the MLP sequence and header.

Note: When interleaving is enabled with more than one link, it is possible that the high priority traffic will get re-ordered. When interleaving is enabled or disabled, a reset of the bundle is required to get it activated in the line card.

ppp multilink mrru local value

This specifies the maximum receive unit on the multilink; packets up to this size will be accepted by the multilink interface. The size here excludes the MLP header.

ppp multilink mrru remote value

This specifies the minimum MRRU that the remote end should support. If the remote end MRRU is lesser than this value, then the bundle negotiation will fail.

ppp multilink fragment delay seconds

This specifies the allowed delay in milliseconds (ms) caused by a data fragment. In other words, the value of delay is used to compute the maximum fragment size allowed. The distributed implementation differs from the Cisco IOS implementation in these ways:

1. Fragmentation is not performed unless interleaving is enabled.
2. With links of varying bandwidth, the fragment size chosen is based on the least bandwidth interface.

ppp multilink fragment disable

This command does not add any functionality in the distributed implementation. Fragmentation only occurs when interleaving is enabled; and, when interleaving is enabled, the **ppp multilink fragment disable** command is ignored.

Verify dMLP Bundle Status on the RP

```
show ppp multilink
```

```
Multilink1, bundle name is M1
Endpoint discriminator is M1
Bundle up for 00:09:09, 1/255 load
Receive buffer limit 24000 bytes, frag timeout 1000 ms
```

```
Bundle is Distributed
```

```
0/0 fragments/bytes in reassembly list
0 lost fragments, 0 reordered
0/0 discarded fragments/bytes, 0 lost received
0x9 received sequence, 0x0 sent sequence
```

```
dLFI statistics:
```

DLFI Packets	Pkts In	Chars In	Pkts Out	Chars Out
Fragmented	0	0	0	0
UnFragmented	9	3150	0	0
Reassembled	9	3150	0	0
Reassembly Drops	0			
Fragmentation Drops	0			
Out of Seq Frags	0			

```
Member links: 2 active, 0 inactive (max not set, min not set)
```

```
Se5/0/0/4:0, since 00:09:09, 768 weight, 760 frag size
```

```
Se5/0/0/5:0, since 00:09:09, 768 weight, 760 frag size
```

1. When bundle is in distributed mode, this is displayed in the output of **show ppp multilink**:

```
Bundle is Distributed
```

If not, then the bundle for some reason is not distributed.

2. When **ppp multilink interleave** is configured and enabled at the line card, the **show ppp multilink** output includes the dLFI statistics where:

- Fragmented Indicates the count of fragments that were transmitted and received.
- Unfragmented Indicates the count of packets that were transmitted or received without getting fragmented.
- Reassembled Indicates the number of full packets that were reassembled. When interleaving is not enabled, the output looks like this:

```
Multilink1, bundle name is M1
Endpoint discriminator is M1
Bundle up for 00:00:00, 0/255 load
Receive buffer limit 24000 bytes, frag timeout 1000 ms
Bundle is Distributed
0/0 fragments/bytes in reassembly list
0 lost fragments, 0 reordered
0/0 discarded fragments/bytes, 0 lost received
0x0 received sequence, 0x2 sent sequence
Member links: 2 active, 0 inactive (max not set, min not set)
Se5/0/0/5:0, since 00:00:00, 768 weight, 760 frag size
Se5/0/0/4:0, since 00:00:03, 768 weight, 760 frag size
```

The fragment size in the previous example is 760 bytes.

Verify dMLP Bundle Status on the LC

show ppp multilink

```
dmlp_ipc_config_count 24
dmlp_bundle_count 2
dmlp_ipc_fault_count 1
dmlp_il_inst 0x60EE4340, item count 0
0, store 0, hwidb 0x615960E0, bundle 0x622AA060, 0x60579290, 0x6057A29C
1, store 0, hwidb 0x615985C0, bundle 0x622AA060, 0x60579290, 0x6057A29C
2, store 0, hwidb 0x0, bundle 0x0,
3, store 0, hwidb 0x0, bundle 0x0,
4, store 0, hwidb 0x0, bundle 0x0,
5, store 0, hwidb 0x0, bundle 0x0,
6, store 0, hwidb 0x0, bundle 0x0,
7, store 0, hwidb 0x0, bundle 0x0,
8, store 0, hwidb 0x0, bundle 0x0,
9, store 0, hwidb 0x0, bundle 0x0,
```

Bundle Multilink1, 2 members

```
bundle 0x622AA060, frag_mode 0
tag vectors 0x604E8004 0x604C3628
Bundle hwidb vector 0x6057B198
idb Multilink1, vc 4, RSP vc 4
QoS disabled, fastsend (qos_fastsend), visible_bandwidth 3072
board_encap 0x60577554, hw_if_index 0, pak_to_host 0x0
max_particles 400, mrru 1524, seq_window_size 0x8000
working_pak 0x0, working_pak_cache 0x0
una_frag_list 0x0, una_frag_end 0x0, null_link 0
rcved_end_bit 1, is_lost_frag 1, resync_count 0
timeout 0, timer_start 0, timer_running 0, timer_count 1
next_xmit_link Serial0/0:3, member 0x3, congestion 0x3
dmlp_orig_pak_to_host 0x603E7030
dmlp_orig_fastsend 0x6035DBC0
bundle_idb->lc_ip_turbo_fs 0x604A7750
0 lost fragments, 0 reordered, 0 unassigned
0 discarded, 0 lost received
0xC3 received sequence, 0x0 sent sequence
Member Link: 2 active
Serial0/0:4, id 0x1, fastsend 0x60579290, lc_turbo 0x6057A29C, PTH 0x60579A18, OOF
0
Serial0/0:3, id 0x2, fastsend 0x60579290, lc_turbo 0x6057A29C, PTH 0x60579A18, OOF
0
```

With dMFR, the sequence numbers are maintained on a per-dLCI basis, with the sequence number in the bundle used for LMI dLCI.

Field	Description
dmlp_ipc_config_count	Number of IPC messages received by the LC for multilink or MLFR configuration
dmlp_bundle_count	Number of MLP and MLFR bundles in the LC
dmlp_ipc_fault_count	Number of configuration messages that resulted in failure in LC. Should be 0; if it is non-zero then there might be a problem.

tag_vectors	Indicates the idb to tag_optimum_fs and the idb to ip2tag_optimum_fs vectors used in tag switching.
board_encap	Indicates the board_encap vector that is used to add 2 bytes of board encapsulation, if there are channelized links in a 7500 platform. Should be NULL, if the link contains non-channelized interfaces.
max_particles	Maximum number of particles that can be held in the reassembly buffer
mrru	The maximum size of packet that are accepted without considering the MLP encapsulation. Not applicable for MLFR interface.
seq_window_size	The maximum window size for sequence numbers
working_pak	Indicates the current pak under reassembly. NULL, if none.
working_pak_cache	Pointer to the static pak that is used for reassembly. This is allocated when the first non-complete packet is received by the bundle.
una_frag_list	First entry in the reassembly queue. If the entry is not NULL and does not change, it indicates that the timer is not running a software issue.
una_frag_end	Last entry in the reassembly queue
rcved_end_bit	Indicates that the bundle has received an end bit, so it is hunting for a begin bit.
is_lost_frag	Is true, if a fragment is declared lost. This gets cleared when a fragment with expected sequence is received.

resync_count	Indicates the number of times that the receiver was out of sync with transmitter and had to resync by starting with the last received sequenced fragment.
timeout	Indicates that the reassembly timeout has occurred and packets are being processed from the reassembly queue.
timer_start	Number of times reassembly timer has been started
timer_running	Indicates whether or not the reassembly timer is running.
timer_count	Indicates the number of times that the reassembly timer has expired.
next_xmit_link	The link on which the next packet will be transmitted
Member	Bit field that indicates the members present.
Congestion	Field not used in all branches. Indicates which member links are not congested.
dmlp_orig_pak_to_host	The vector used to punt packets to the RP.
dmlp_orig_fastsend	The original driver fastsend before MLP or MLFR modified the driver s fastsend.
lost fragments	Number of fragments that were lost (the receiver did not receive these fragments). This is periodically cleared when an update is sent to the host.
Reordered	Number of fragments that were received out of expected order. This is periodically cleared when an update is sent to the host.

Discarded	Number of fragments discarded because a complete packet could not be made
lost received	Number of fragments received which were thought to be lost. This indicates that the inter-link delay is greater than the dMLP reassembly timeout of 30 ms.

Configuring and Verifying dLFloFR and dLFloATM

```

class-map voip
  match ip precedence 3

policy-map llq
  class voip
    priority

int virtual-templatel
  service-policy output llq
  bandwidth 78
  ppp multilink
  ppp multilink interleave
  ppp multilink fragment-delay 8

int serial5/0/0/6:0
  encapsulation frame-relay
  frame-relay interface-dlci 16 ppp virtual-templatel

```

!--- Or

```

int ATM4/0/0
  no ip address
int ATM4/0/0.1 point-to-point
  pvc 5/100
  protocol ppp virtual-template 1

```

Verify dLFloFR/ATM Bundle Status on the RP

```
show ppp multilink
```

```

Virtual-Access3, bundle name is dLFI
Endpoint discriminator is dLFI
Bundle up for 00:01:11, 1/255 load
Receive buffer limit 12192 bytes, frag timeout 1524 ms
Bundle is Distributed
  0/0 fragments/bytes in reassembly list
  0 lost fragments, 0 reordered
  0/0 discarded fragments/bytes, 0 lost received
  0x0 received sequence, 0x0 sent sequence
dLFI statistics:

```

DLFI Packets	Pkts In	Chars In	Pkts Out	Chars Out
Fragmented	0	0	0	0
UnFragmented	0	0	0	0
Reassembled	0	0	0	0
Reassembly Drops	0			
Fragmentation Drops	0			
Out of Seq Frags	0			

Member links: 1 (max not set, min not set)

Vi2, since 00:01:11, 240 weight, 230 frag size

Note: The bundle will become distributed only when **ppp multilink interleave** is configured under the virtual template; without this command, the bundle will not be distributed.

Verify dLFloFR/ATM Bundle Status on the LC

To verify the dLFI is indeed working correctly at the LC, issue this command:

```
show hqf interface
```

```
Interface Number 6 (type 22) Serial0/0:5
```

```
blt (0x62D622E8, index 0, hwidb->fast_if_number=35) layer PHYSICAL
scheduling policy: FIFO
classification policy: NONE
drop policy: TAIL
blt flags: 0x0
```

```
qsize 0 txcount 3 drops 0 qdrops 0 nobuffers 0
aggregate limit 16 individual limit 4 availbuffers 16
weight 1 perc 0.00 ready 1 shape_ready 1 wfq_clitype 0
visible_bw 64 allocated_bw 64 qlimit_tuned 0 vc_encap 2
quantum 1500 credit 0 backpressure_policy 0 nothingoncalQ 1
```

```
next layer HQFLAYER_FRAMEDLCI_IFC (max entries 1024)
```

```
blt (0x62D620E8, index 0, hwidb->fast_if_number=35) layer FRAMEDLCI_IFC
scheduling policy: FIFO
classification policy: NONE
drop policy: TAIL
blt flags: 0x0
```

```
qsize 0 txcount 1 drops 0 qdrops 0 nobuffers 0
aggregate limit 16 individual limit 4 availbuffers 16
weight 1 perc 0.00 ready 1 shape_ready 1 wfq_clitype 0
visible_bw 64 allocated_bw 64 qlimit_tuned 0 vc_encap 2
quantum 1500 credit 0 backpressure_policy 0 nothingoncalQ 1
```

```
blt (0x62D621E8, index 16, hwidb->fast_if_number=35) layer FRAMEDLCI_IFC
scheduling policy: WFQ
classification policy: PRIORITY_BASED
drop policy: TAIL
frag policy: root
blt flags: 0x0
```

```
qsize 0 txcount 2 drops 0 qdrops 0 nobuffers 0
aggregate limit 16 individual limit 4 availbuffers 16
weight 1 perc 0.00 ready 1 shape_ready 1 wfq_clitype 0
visible_bw 64 allocated_bw 64 qlimit_tuned 0 vc_encap 2
quantum 240 credit 0 backpressure_policy 0 nothingoncalQ 1
```

next layer HQFLAYER_PRIORITY (max entries 256)

blt (0x62D61FE8, index 0, hwidb->fast_if_number=35) **layer PRIORITY**
scheduling policy: FIFO
classification policy: NONE
drop policy: TAIL
frag policy: leaf
blt flags: 0x0

qsize 0 txcount 0 drops 0 qdrops 0 nobuffers 0
aggregate limit 8 individual limit 2 availbuffers 8
weight 0 perc 0.99 ready 1 shape_ready 1 wfq_clitype 0
visible_bw 32 allocated_bw 32 qlimit_tuned 0 vc_encap 2
quantum 240 credit 0 backpressure_policy 0 nothingoncalQ 1

blt (0x62D61CE8, index 1, hwidb->fast_if_number=35) **layer PRIORITY**
scheduling policy: FIFO
classification policy: NONE
drop policy: TAIL
blt flags: 0x0

Priority Conditioning enabled

qsize 0 txcount 0 drops 0 qdrops 0 nobuffers 0
aggregate limit 0 individual limit 0 availbuffers 0
weight 1 perc 0.00 ready 1 shape_ready 1 wfq_clitype 0
visible_bw 0 allocated_bw 0 qlimit_tuned 0 vc_encap 2
quantum 240 credit 0 backpressure_policy 0 nothingoncalQ 1

PRIORITY: bandwidth 32 (50%)
last 0 tokens 1500 token_limit 1500

blt (0x62D61EE8, index 255, hwidb->fast_if_number=35) **layer PRIORITY**
scheduling policy: WFQ
classification policy: CLASS_BASED
drop policy: TAIL
frag policy: MLPPP (1)
frag size: 240, vc encap: 0, handle: 0x612E1320
blt flags: 0x0

qsize 0 txcount 2 drops 0 qdrops 0 nobuffers 0
aggregate limit 8 individual limit 2 availbuffers 8
weight 1 perc 0.01 ready 1 shape_ready 1 wfq_clitype 0
visible_bw 32 allocated_bw 32 qlimit_tuned 0 vc_encap 2
quantum 1 credit 0 backpressure_policy 0 nothingoncalQ 1

next layer HQFLAYER_CLASS_HIER0 (max entries 256)

blt (0x62D61DE8, index 0, hwidb->fast_if_number=35) layer CLASS_HIER0
scheduling policy: FIFO
classification policy: NONE
drop policy: TAIL
frag policy: leaf
blt flags: 0x0

qsize 0 txcount 2 drops 0 qdrops 0 nobuffers 0
aggregate limit 8 individual limit 2 availbuffers 8
weight 1 perc 50.00 ready 1 shape_ready 1 wfq_clitype 0
visible_bw 32 allocated_bw 32 qlimit_tuned 0 vc_encap 2

```
quantum 240 credit 0 backpressure_policy 0 nothingoncalQ 1
```

There should be a priority layer and a WFQ layer. Fragmentation will be done at the WFQ leaf layer blt.

Configuring and Verifying dDDR

Distributed DDR is activated when you enable **ip cef distributed** on the global configuration and **ip route-cache distributed** on the dialer interfaces.

```
!--- Global configuration that enables distributed CEF switching.
```

```
ip cef distributed
```

```
--- Enable distributed switching on the dialer interface (on the D-channel interface).
```

```
int serial 3/1/0:23
    ip route-cache distributed
```

```
!--- Or, enable it on the dialer interface.
```

```
int Dialer1
    ip route-cache distributed
```

There are no other special configurations for distributed DDR. Further configuration follows normal DDR configuration.

Verify Distributed Dial on Demand Routing

```
BOX2002# show isdn status
```

```
Global ISDN Switchtype = primary-net5
ISDN Serial3/1/0:23 interface
```

```
--- Network side configuration.
```

```
    dsl 0, interface ISDN Switchtype = primary-net5
Layer 1 Status:
    ACTIVE
Layer 2 Status:
    TEI = 0, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
```

```
The ISDN status should be MULTIPLE_FRAME_ESTABLISHED.
```

```
This means that the physical layer is ready for ISDN connectivity.
```

```
Layer 3 Status:
    0 Active Layer 3 Call(s)
Active dsl 0 CCBs = 0
The Free Channel Mask: 0x807FFFFFFF
Number of L2 Discards = 0, L2 Session ID = 6
```

```
EDGE# show dialer
```

```
Serial6/0:0 - dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is data link layer up
```

```
Time until disconnect 119 secs
Current call connected never
Connected to 54321
```

```
Serial6/0:1 - dialer type = ISDN
Idle timer (120 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dialer state is idle
```

The dialer type tells us the type of dialer used. ISDN implies legacy dialer configuration and PROFILE implies dialer profile configuration. The dialer state indicates the present state of the dialer. The state of an unconnected dialer interface will be idle. The Idle timer is reset whenever interesting traffic is seen. If this timer ever expires, the interface will immediately disconnect. Idle timer is a configurable parameter. For further information, refer to [Configuring Peer-to-Peer DDR with Dialer Profiles](#).

```
show ppp multilink
```

```
!--- From LC for dialer profile.
```

```
dmlp_ipc_config_count 2
dmlp_bundle_count 1
dmlp_il_inst 0x60EE4340, item count 0
0, store 0, hwidb 0x0, bundle 0x0,
1, store 0, hwidb 0x0, bundle 0x0,
2, store 0, hwidb 0x0, bundle 0x0,
3, store 0, hwidb 0x0, bundle 0x0,
4, store 0, hwidb 0x0, bundle 0x0,
5, store 0, hwidb 0x0, bundle 0x0,
6, store 0, hwidb 0x0, bundle 0x0,
7, store 0, hwidb 0x0, bundle 0x0,
8, store 0, hwidb 0x0, bundle 0x0,
9, store 0, hwidb 0x0, bundle 0x0,
```

```
Bundle Dialer1, 1 member
```

```
bundle 0x62677220, frag_mode 0
tag vectors 0x604E8004 0x604C3628
Bundle hwidb vector 0x0
idb Dialer1, vc 22, RSP vc 22
QoS disabled, fastsend (mlp_fastsend), visible_bandwidth 56
board_encap 0x60577554, hw_if_index 0, pak_to_host 0x0
max_particles 200, mrru 1524, seq_window_size 0x8000
working_pak 0x0, working_pak_cache 0x0
una_frag_list 0x0, una_frag_end 0x0, null_link 0
rcved_end_bit 1, is_lost_frag 0, resync_count 0
timeout 0, timer_start 0, timer_running 0, timer_count 0
next_xmit_link Serial1/0:22, member 0x1, congestion 0x1
dmlp_orig_pak_to_host 0x603E7030
dmlp_orig_fastsend 0x60381298
bundle_idb->lc_ip_turbo_fs 0x604A7750
0 lost fragments, 0 reordered, 0 unassigned
0 discarded, 0 lost received
0x0 received sequence, 0x0 sent sequence
Member Link: 1 active
Serial1/0:22, id 0x1, fastsend 0x60579290, lc_turbo 0x6057A29C, PTH 0x60579A18,
OOF 0
```

The variables shown are the same as those for dMLP.

Debugging dMLP and dDDR

Debugs Available at the RP

dDDR

```
debug dialer [events | packets | forwarding | map]
```

Issue this command to debug control path functions like call setup and so forth. For further information, refer to [debug dialer events](#)

```
debug ip cef dialer
```

Issue this command to debug CEF-related dialer events. For further information, refer to [Dialer CEF](#).

Debugs Available at the LC

dMLP

Control Path debugging: **debug multilink event**

Data path debugging: **debug multilink fragments**

Data path and control path error debugging: **debug multilink error**

Debugging dMLP on SIP Linecards

Dumping packets based on CI: Data Packets and Control Packets can be dumped on linecards based on the control ci and sequence ci.

```
test hw-module subslot_num dump ci CI-NUM [rx|tx] num_packets_to_dump
```

The CIs can be obtained in this manner:

```
!--- Issue show controller serial interface  
for CTE1.
```

```
SIP-200-6# show controller serial 6/0/0:0
```

```
SPA 6/0 base address 0xB8000000 efc 1
```

```
Interface Serial6/0/0:0 is administratively down  
Type 0xD Map 0x7FFFFFFF, Subrate 0xFF, mapped 0x1, maxmtu 0x5DC  
Mtu 1500, max_buffer_size 1524, max_pak_size 1608 enc 84  
ROM rev: 0, FW OS rev: 0x00000000 Firmware rev: 0x00000000  
idb=0x42663A30, pa=0x427BF6E0, vip_fci_type=0, port_per_spa=0  
SPA port type is set  
Host SPI4 in sync  
SPA=0x427BF6E0 status=00010407, host=00000101, fpga=0x427EDF98  
cmd_head=113, cmd_tail=113, ev_head=184, ev_tail=184  
ev_dropped=0, cmd_dropped=0
```

```
!--- Start Link Record Information.
```

```
tag 0, id 0, anyphy 0, anyphy_flags 3, state 0  
crc 0, idle 0, subrate 0, invert 0, priority 0  
encap hdlc  
corrupt_ci 65535, transparent_ci 1
```

```
!--- End Link Record Information.
```

```
Interface Serial6/0/0:0 is administratively down
Channel Stats:
  in_throttle=0, throttled=0, unthrottled=0, started=1
  rx_packets=0, rx_bytes=0, rx_frame_aborts=0, rx_crc_errors=0
  rx_giants=0, rx_non_aligned_frames=0, rx_runts=0, rx_overruns=0
  tx_packets=0, tx_bytes=0, tx_frame_aborts=0
  is_congested=0, mapped=1, is_isdn_d=0, tx_limited=1
  fast_if_number=15, fastsend=0x403339E4
  map=0x7FFFFFFF, turbo_vector_name=Copperhead to Draco switching
  lc_ip_turbo_fs=403A9EEC, lc_ip_mdcs=403A9EEC
```

For CT3, you must obtain the `vc num`, which can be obtained from the output of **show interface serial *CT3_interface_name*** .

Now the CI information can be obtained from the SPA console. First redirect the output of SPA console commands to the RP with the **spa_redirect rp ct3_freedom336** command.

The **spa_ct3_test_freedom show linkrec vc** command shows the necessary CI information.

dMFR

Control Path debugging: **debug dmfr event**

Data path debugging: **debug dmfr packets**

Data path and control path error debugging: **debug dmfr error**

Dumping packets based on CI: See [dMLP](#).

dLFI

Control Path debugging: **debug dlfi event**

Data path debugging: **debug dlfi fragments**

Data path and control path error debugging: **debug dlfi error**


dDDR

There are no special debugging commands; you should use [dMLP debugs](#).

In case of dLFIoLL, both dMLP and dLFI debugs might have to be used. These debugs are not conditional and, hence, will trigger for all bundles.

Frequently Asked Questions

1. What is dMLP?

dMLP is short for Distributed Multilink PPP (as stated in [RFC1990](#) ). This feature is supported by distributed platforms, like the Cisco 7500 Series and 7600 Series. dMLP allows you to combine T1/E1 lines in a VIP on a Cisco 7500 series router or a FlexWAN in a 7600 Series router into a bundle that has the combined bandwidth of multiple T1/E1 lines. This allows customers to increase the bandwidth beyond T1/E1 without the need to purchase a T3/E3 line.

2. What is distributed in dMLP?

The term `distributed` implies that the packet switching is done by the VIP and not the RSP. Why? RSP switching capabilities are rather limited, and it has many more important jobs to do. The VIP being capable of switching packets offloads this activity from the RSP. RSP-based Cisco IOS still manages the links. Bundle creation and teardown are done by the RSP. Additionally, PPP control plane processing is still done by RSP, including the handling of all PPP control packets (LCP, Authentication, and the NCPs). However, once a bundle is established, the handling of MLP packets is turned over to the VIP for switching by the onboard CPU. The dMLP engine (on the VIP) handles all the MLP procedures, including fragmentation, interleaving, encapsulation, load balancing amongst multiple links, and sorting and reassembly of inbound fragments. The functions done by the VIP in a 7500 system are done by the Flexwan/Enhanced-FlexWAN in a 7600 based system.

3. How do I know if the bundle is distributed or not?

Issue the **show ppp multilink** command at the router console:

```
Router# show ppp multilink
```

```
Multilink1, bundle name is udho2  
Bundle up for 00:22:46  
Bundle is Distributed  
174466 lost fragments, 95613607 reordered, 129 unassigned  
37803885 discarded, 37803879 lost received, 208/255 load  
0x4D987C received sequence, 0x9A7504 sent sequence  
Member links: 28 active, 0 inactive (max not set, min not set)  
  Se11/1/0/27:0, since 00:22:46, no frags rcvd  
  Se11/1/0/25:0, since 00:22:46, no frags rcvd
```

!--- Output suppressed.

4. If I upgrade to RSP16 or SUP720, will my dMPLP performance be better?

No. The switching performance of dMPLP (or any distributed feature) is dependant on the VIP or FlexWAN in question. For example, the performance of a VIP6-80 will be better than the performance with VIP2-50.

5. Which PAs can I use with this feature?

- PA-MC-T3
- PA-MC-2T3+
- PA-MC-E3
- PA-MC-2E1
- PA-MC-2T1
- PA-MC-4T1
- PA-MC-8T1
- PA-MC-8E1
- PA-MC-STM-1
- PA-MC-8TE1+
- PA-4T+
- PA-8T
- CT3IP-50 (7500 only)

6. How many links can be configured in a single bundle?

There are many facets to this answer. The primary bottleneck is the CPU power of the line card (VIP/FlexWAN/Enhanced-FlexWAN2). The hard limit is 56 links per bundle, but many times you can not configure those many (and have that much traffic switching), either due to CPU power or limited buffers. These numbers are based on this guideline (based on the CPU and the memory on the VIP/FlexWAN/Enhanced-FlexWAN2):

VIP2-50 (w/ 4MB SRAM) max T1s = 12

VIP2-50 (w/ 8MB SRAM) max T1s = 16

VIP4-80 max T1s = 40

VIP6-80 max T1s = 40

FlexWAN max T1s = Will be updated shortly

Enhanced-FlexWAN max E1s = 21 E1s per bay (aggregate 42 E1s per line card)

7. Is there a change in performance if I configure 3 bundles with 3 T1s each or 1 bundle with 9 T1s?

There is no change in performance, as proven in lab tests. However, with a large number of T1s in a single bundle (say 24 or 28 T1s in a single bundle), there are issues with running out of buffers. Its highly recommended that you not have more than 8 member links (T1/E1) in a single bundle.

8. How is the bandwidth of a bundle determined?

The bandwidth of a bundle is not to be configured. Its the aggregate bandwidth of all the member links. If you have 4 T1s in the bundle, then the bandwidth of the bundle is 6.144Mbps.

9. Which is better? CEF-load balancing or dMLP?

There is no simple answer to this. Your needs decide which one is better.

PROS of MLP:

- CEF load balancing is applicable only to IP traffic. MLP balances all the traffic sent over a bundle.
- MLP maintains the ordering of packets. IP itself is tolerant of reordering, so this may not matter to you; in fact, the extra cost involved in maintaining the sequencing may be a reason to avoid MLP. IP is intended for networks which may deliver datagrams out of order, and anything using IP is supposed to be able to deal with reordering. However, despite this fact, the reality is that reordering can still pose a real problem.
- MLP provides a single logical connection to the peer system.
- QoS is supported on Multilink bundles.
- MLP provides dynamic bandwidth capabilities, as the user can add or remove member links based on current needs.
- MLP can bundle a larger numbers of links, whereas CEF load balancing is limited to 6 parallel IP paths.
- Per-flow CEF load balancing limits the maximum bandwidth of any given flow to one T1. For example, customers using voice gateways can have a lot of calls with the same source and destination and, hence, use only one path.

CONS of MLP:

- MLP adds extra overhead to each packet or frame
- MLP is CPU intensive; dMLP is line card CPU intensive.

10. How can I configure multiple bundles between two routers?

Multilink determines which bundle a link will join based on the peer s name and endpoint discriminator. To create multiple distinct bundles between two systems, the standard method is to force some of the links to identify themselves differently. Recommended method is the use of the **ppp chap hostname *name*** command.

11. Can I have member links from different PAs?

No. If you want to run dMLP, then it is not supported. However, if member links are added from different PAs, then the control is given to RSP and its not dMLP anymore. MLP is still functioning, but the benefits of dMLP are gone.

12. Can I mix member links from both bays?

No. If you want to run dMLP, then it is not supported. However, if member links are added from different PAs, then the control is given to RSP and it is not dMLP anymore. MLP is still functioning, but the benefits of dMLP are gone.

13. Can I have member links across different VIPs or FlexWANs?

No. If you want to run dMLP, then it is not supported. However, if member links are added from different PAs, then the control is given to RSP and its not dMLP anymore. MLP is still functioning, but the benefits of dMLP are gone.

14. Can I have member links across different ports from a single PA?

(For example, one member link from each CT3 port of a PA-MC-2T3+.)

Yes. As long as it is from the same PA, there are no issues.

15. **Can I bundle T3 or E3 ports?**

No. Only DS0, n*DS0, T1, and E1 speeds are allowed with dMLP for 7500/VIP, 7600/FlexWAN, and 7600/FlexWAN2.

Note: Distributed MLPPP is supported only for member links configured at T1/E1 or subrate T1/E1 speeds. Channelized STM-1/T3/T1 interfaces also support dMLPPP at T1/E1 or subrate T1/E1 speeds. Distributed MLPPP is not supported for member links configured at clear-channel T3/E3 or higher interface speeds.

16. **What are reordered fragments?**

If the received fragment or packet does not match the expected sequence number, then the `reordered` counter is incremented. For varying packet sizes, this is bound to happen. For fixed size packets, this can also happen because the PA driver processes the packets that received on one link and does not go at round-robin basis (as is done in dMLP while transmitting the packets). Reordered does not mean packet loss.

17. **What are lost fragments?**

Whenever the fragment or packet is received out of order and you find that out of order fragments or packets are received on all of the links, the `lost_fragments` counter is incremented. Another case is when the out of order fragments are being stored in the list and it reaches a limit (decided based on the SRAM on VIP and whatever is assigned for the bundle), the lost fragments counter is incremented and the next sequence number in the list is taken for processing.

18. **How does dMLP detect lost fragments?**

- . Sequence numbers: If you are waiting for a fragment with sequence number N to arrive, and all the links receive a fragment with a sequence number higher than N, you know that fragment N must be lost, because there is no way it could legally arrive behind higher numbered fragments on the same link.
- b. Timeout: If you sit too long waiting for a fragment, you will eventually declare it as lost and move on.
- c. Reassembly buffer overflow: If you are waiting for fragment N to arrive, and meanwhile other fragments (with sequence numbers higher than N) are arriving on some of the links, then you have to park those fragments in a reassembly buffer until fragment N shows up. There is a limit to how much you can buffer. If the buffer overflows, you again declare fragment N as lost, and resume processing with whatever is in the buffer.

19. **What are lost received?**

There are two possible reasons for lost received fragments or packets:

- . If the received fragment or packet is out of the expected sequence range window, the packet is dropped by marking it as lost received.
- b. If the received fragment or packet is within the expected sequence range window, but you are not able to allocate a packet header to re-parent this packet, then the packet is dropped and marked as lost received.

20. **Is encryption supported with dMLP?**

No.

21. **Do we support PFC header compression?**

No, not in the distributed path. The far end router is not recommended to configure PFC header compression because we fall back to non-distributed mode if we receive compressed header frames or packets. If you want to continue running dMLP, PFC header compression must be disabled at both ends.

22. **Is software compression supported with dMLP?**

No, because software compression will not work in the distributed path.

23. **Is fragmentation supported on the transmit side?**

Not with Vanilla dMLP. There are no issues with receiving fragments with Vanilla dMLP, but on the transmit side, fragmentation does not happen. The transmit side fragmentation is supported when **ppp multilink interleave** is configured on the dMLP interface.

24. Can we ping the member links of an MLP bundle?

No, you can not configure an IP address on the member links.

25. Is there any dependency on the link MTU and MLP fragment size?

No. The MTU size has nothing to do with the MLP fragment size, other than the obvious restriction that an MLP fragment, like any other frame, can not exceed the Serial links MTU sizes.

26. Is it possible to configure two MLP bundles between a single pair of routers?

Yes, it is possible. However, this could lead to impaired load balancing. It can be useful on testbeds, to simulate more than two routers using just two routers, but it does not have any obvious real-world value.

All links which go to a common peer *must* be put in the same bundle. By definition, a bundle is the set of links that are going to a particular peer.

A peer is identified by the Username and Endpoint Discriminator values that it offers during LCP and authentication phases. If you are trying to create multiple bundles between two routers, then it means you are trying to make each router masquerade as being more than a single peer to its counterpart. They must identify themselves appropriately.

27. Can member links have different queuing algorithms?

All of the queuing mechanisms related to a bundle need to be applied at the bundle level and not at the member link level. However, configuring a queue algorithm should not affect how the packets are switched out of the bundle.

28. Why is the tx-queue-limit set to 26 as default for member links for a multilink bundle when dMLP is enabled on a Cisco 7500?

For any Serial interface of bandwidth T1/E1, the tx-queue-limit is around 4 or 5. When you are bundling T1s/E1s together in multilink, the bandwidth would increase for the bundle. Because switching would take place based on the bandwidth of MLP interface, you need to increase the tx-queue-limit of member links. Only one of the member links, called the primary link, is used for switching, therefore, its tx-queue-limit need to be increased.

Also, this value is an empirical one chosen after testing and then tuning to this value. In general, the deployments have not more than 4 to 6 T1/E1 links in a bundle. A value of 26 can cater for 6 to 8 T1/E1 links perfectly, and hence this value was chosen.

29. What is differential delay and its value in dMLP implementation?

dMLP supports a differential delay of 30 ms. That would mean if a fragment is received at a time T and it is out of order (expecting a sequence number 100, but we received 101). If sequence number 100 is not received until T+30 ms, 100 would be declared lost and if you can start processing from 101, you would do that. In case you can not start with 101 (if it is a middle fragment), you would look for the fragment which has the begin fragment (for instance, 104) and start from there.

30. What happens when packets are fragmented at IP level with multilink on 7500?

If packets are fragmented at the IP level, then they are transported without reassembly at the intermediate hops but are reassembled at the destination router.

31. What happens when packets are fragmented at MLP level on 7500?

If the packets are fragmented at the MLP level and if the reassembled packets are greater than MRRU, then packets are dropped on multilink. Transmit-side fragmentation is supported on dMLP only with dLFI. Packets are fragmented at the MLP level only if the packet_size is greater than the frag_size and less than the MRRU. If packets more than the MRRU are sent and if it is not fragmented at the IP level, then the other end drops all the packets size they are not fragmented at the MLP level because the packets are more than MRRU.

32. How is MRRU calculated?

MRRU is calculated according to these preferences:

- a. For the new member links coming in, MRRU is again negotiated at the LCP level according to the MRRU configured on the member links.
- b. The value configured on the link interface with the **ppp multilink mrru interface** command.

- c. If not configured, the value inherited from the configuration of the **ppp multilink mrru** command on the parent interface.
- d. If both values are present, the link interface value has precedence.
- e. The default MRRU of 1524.

Debug Enhancements

These enhancements will be taken up in future. Planning is not yet to complete.

- Enable **debug frame-relay multilink** command on the LC.
- Enhance the current debug CLIs per interface and specified number of packets.
- For dDDR, QoS functionality is not yet supported. This can be taken up only with proper business case.

NetPro Discussion Forums - Featured Conversations

Networking Professionals Connection is a forum for networking professionals to share questions, suggestions, and information about networking solutions, products, and technologies. The featured links are some of the most recent conversations available in this technology.

NetPro Discussion Forums - Featured Conversations for Access
--

Network Infrastructure: Remote Access

Related Information

- [Dialer CEF](#)
- [Configuring Peer-to-Peer DDR with Dialer Profiles](#)
- [MPLS Multilink PPP Support](#)
- [Distributed Multilink Point-to-Point Protocol for Cisco 7500 Series Routers](#)
- [Distributed Multilink Frame Relay \(FRF.16\)](#)
- [Distributed Link Fragmentation and Interleaving over Leased Lines](#)
- [VoIP over PPP Links with Quality of Service \(LLQ / IP RTP Priority, LFI, cRTP\)](#)
- [Troubleshooting TechNotes - Cisco 7500 Series Router](#)
- [Router Product Support Page - Cisco Systems](#)
- [Technical Support & Documentation - Cisco Systems](#)

Home	How to Buy	Login	Profile	Feedback	Site Map	Help
----------------------	----------------------------	-----------------------	-------------------------	--------------------------	--------------------------	----------------------

All contents are Copyright © 2006-2007 Cisco Systems, Inc. All rights reserved. [Important Notices](#) and [Privacy Statement](#).