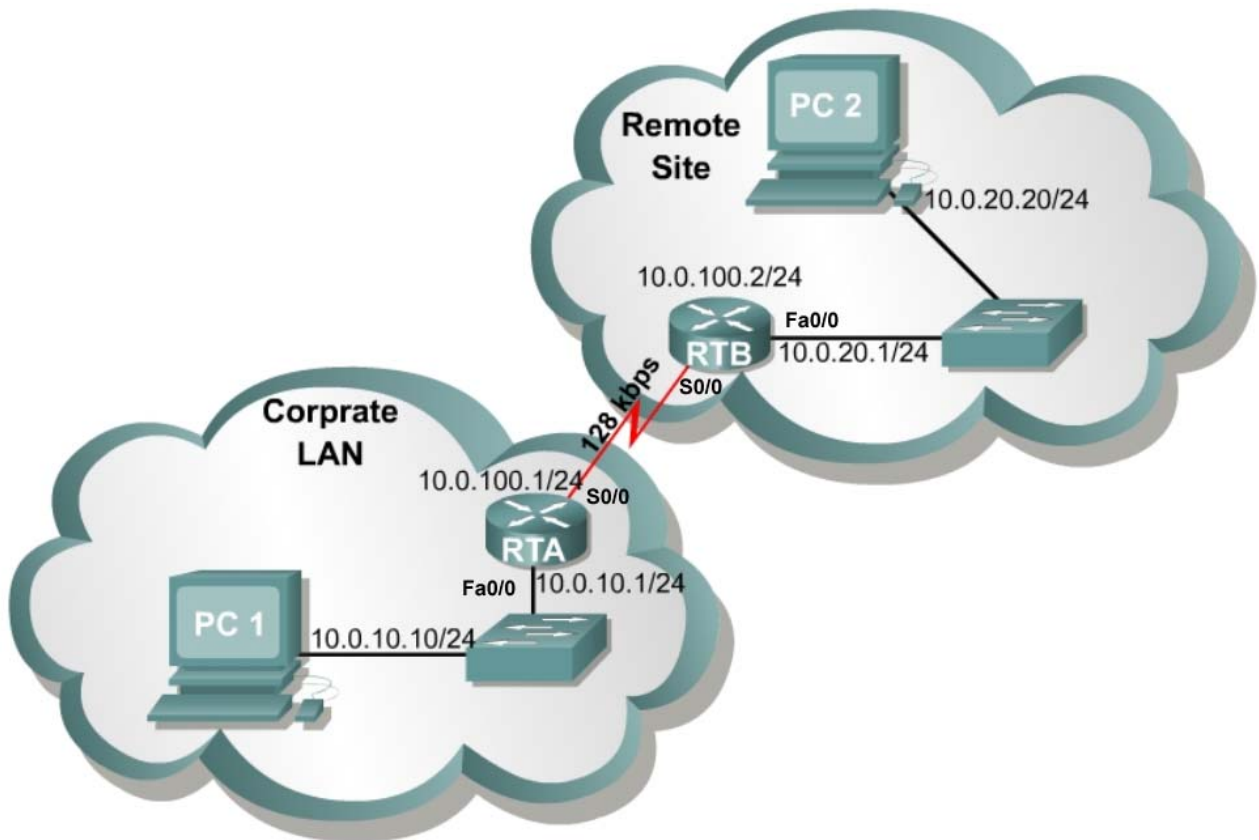


## Lab 8.1.10.11 Configuring Link Fragmentation and Interleaving



### Objective

In this lab, configure Link Fragmentation and Interleave (LFI) to control latency over a low speed WAN link.

### Scenario

The International Travel Agency has a low speed, 128-kbps WAN link to a remote office. In order to save costs they would like to send their long distance voice traffic over this link. The system works correctly when voice traffic travels across the WAN link on its own. However, even a small simultaneous transfer of data packets results in a severely degraded or unusable voice call. Configure Link Fragmentation and Interleaving to ensure that small delay sensitive voice packets do not get stuck behind large data packets traveling across the WAN link.

---

**Note** Proceed to step 10 and complete the assignment of the multilink group to the interface if during the configuration of the lab the serial line continuously flaps, such as:

---

```

06:46:24: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to
down
06:46:24: %LINEPROTO-5-UPDOWN: Line protocol on Interface Multilink1, changed state
to down
06:46:25: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
06:46:25: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
06:46:27: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
06:46:27: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up

```

This will prevent the interface flapping and allow completion of the lab without the annoying up-down messages.

## Step 1

Build and configure the network according to the diagram. Before beginning a lab, the configurations on all the routers should be cleared and then reloaded or power cycled to reset their default configurations. Delete the **vlan.dat** and startup configuration files on the switches before reloading them.

Configure the hostnames and interfaces on the routers. Use the Enhanced Interior Gateway Routing Protocol (EIGRP) as the routing protocol and be sure to set the clock rate to 128 kbps.

Use the **ping** and **show ip route** commands to test the connectivity between all interfaces. The two PCs should be able to ping each other.

```

Router(config)#hostname RTA
RTA(config)#interface fastethernet 0/0
RTA(config-if)#ip address 10.0.10.1 255.255.255.0
RTA(config-if)#no shutdown
RTA(config-if)#interface serial 0/0
RTA(config-if)#clock rate 128000
RTA(config-if)#ip address 10.0.100.1 255.255.255.0
RTA(config-if)#no shutdown
RTA(config-if)#router eigrp 100
RTA(config-router)#network 10.0.10.0 0.0.0.255
RTA(config-router)#network 10.0.100.0 0.0.0.255

```

```

Router(config)#hostname RTB
RTB(config)#interface fastethernet 0/0
RTB(config-if)#ip address 10.0.20.1 255.255.255.0
RTB(config-if)#no shutdown
RTB(config-if)#interface serial 0/0
RTB(config-if)#clock rate 128000
RTB(config-if)#ip address 10.0.100.2 255.255.255.0
RTB(config-if)#no shutdown
RTB(config-if)#router eigrp 100
RTB(config-router)#network 10.0.100.0 0.0.0.255
RTB(config-router)#network 10.0.20.0 0.0.0.255

```

## Step 2

Configure file sharing on PC1 and verify files that can be transferred from PC1 to PC2. The transfer of files from PC1 to PC2 will be used to create data traffic to compete with the simulated voice traffic.

---

<b>Note</b>	For this lab, file sharing was implemented by configuring PC1 as a Web server hosting a 20MB file.
-------------	--

---

The file needs to be large enough to generate a continuous stream of background traffic.

## Step 3

To simulate voice traffic traveling across the WAN link, **ping** PC2 from the command prompt on PC1. Use the following **ping** parameters to generate a continuous stream of small packets.

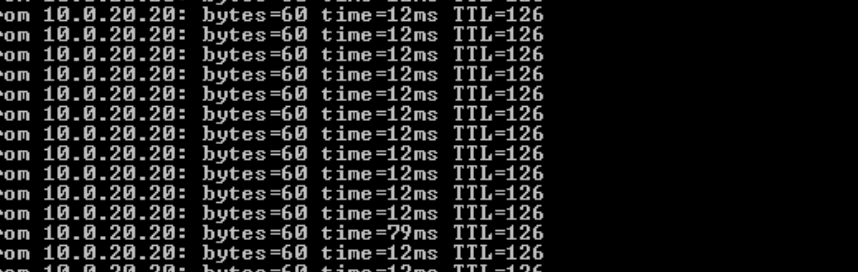
```
Ping -t -l 60 -w 5000 10.0.20.20
```

The `-w 5000` instructs `ping` to wait up to 5 seconds before declaring that a timeout has occurred. By default, `ping` will only wait 2 seconds.

1. What is the average time for a packet to cross the WAN link?
2. What is the acceptable latency for voice traffic?

### Step 4

Use PC2 to copy a large file from PC1. Now examine the continuous **ping** that is running on PC1.



The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\System32\cmd.exe". The window contains a series of 20 lines of network output, each starting with "Reply from 10.0.20.20:". Each line displays three metrics: "bytes=60", "time", and "TTL=126". The "time" values are mostly 12ms, with some variations: 79ms, 415ms, 480ms, 536ms, 588ms, 102ms, 251ms, 118ms, 355ms, 244ms, 487ms, 548ms, 225ms, 299ms, 117ms, 189ms, 247ms, and 122ms. The "TTL" value is consistently 126 for all responses.

Line	Reply from	bytes	time	TTL
1	10.0.20.20	60	12ms	126
2	10.0.20.20	60	12ms	126
3	10.0.20.20	60	12ms	126
4	10.0.20.20	60	12ms	126
5	10.0.20.20	60	12ms	126
6	10.0.20.20	60	12ms	126
7	10.0.20.20	60	12ms	126
8	10.0.20.20	60	12ms	126
9	10.0.20.20	60	12ms	126
10	10.0.20.20	60	12ms	126
11	10.0.20.20	60	12ms	126
12	10.0.20.20	60	12ms	126
13	10.0.20.20	60	12ms	126
14	10.0.20.20	60	12ms	126
15	10.0.20.20	60	12ms	126
16	10.0.20.20	60	12ms	126
17	10.0.20.20	60	12ms	126
18	10.0.20.20	60	12ms	126
19	10.0.20.20	60	12ms	126
20	10.0.20.20	60	12ms	126

Notice that the average time is no longer 12ms.

1. What is the average time for a packet to cross the WAN link?
2. Would this be acceptable for voice traffic?

**Note** The **ping** command actually measures the round-trip time, whereas the latency requirements for voice are stated in terms of a one-way trip. However, as the link is heavily congested in one direction only, most of the latency is experienced in the PC1 to PC2 direction.

## Step 5

The amount of latency experienced is the result of voice packets waiting in the router queue for the data packets to cross the WAN link. Problem resolution could be attempted by implementing some kind of priority queuing scheme. However, the situation will arise where a data packet has just started its journey across the WAN link, when a voice packet arrives. Therefore, the voice packet will have to wait for the data packet to be sent before it can be forwarded across the WAN link. If the data packet is 1500 bytes in length, the wait could be up to 94 ms.

$$\frac{1500 \text{ (bytes)} \times 8 \text{ (bits)}}{128000 \text{ (Bandwidth)}} = 93 \text{ ms}$$

Clearly, even giving priority to voice packets cannot guarantee low latency when the link is shared with large packets.

In order to guarantee a low latency, any large packets will need to be fragmented or broken up into smaller pieces. The IOS feature that allows this to occur is Link Fragmentation and Interleaving (LFI).

LFI makes use of PPP Multilink ability to break up and reassemble traffic across multiple physical links. LFI also has the ability to breakup and reassemble fragments across a single physical link. In order to achieve this, it is necessary to create a PPP Multilink virtual interface and link this to the physical interface.

Begin by removing the IP address from the physical interface and configure PPP multilink:

```
RTA(config)#interface serial 0/0
RTA(config-if)#no ip address
RTA(config-if)#encapsulation ppp
RTA(config-if)#ppp multilink
RTA(config-if)#no shutdown

RTB(config)#interface serial 0/0
RTB(config-if)#no ip address
RTB(config-if)#encapsulation ppp
RTB(config-if)#ppp multilink
RTB(config-if)#no shutdown
```

---

**Note** Do not remove the clock rate from the DCE end of the serial link.

---

## Step 6

Configure a PPP multilink virtual interface called multilink 1 and set the IP address.

```
RTA(config)#interface multilink 1
RTA(config-if)#ip address 10.0.100.1 255.255.255.0

RTB(config)#interface multilink 1
RTB(config-if)#ip address 10.0.100.2 255.255.255.0
```

## Step 7

Use the `ppp multilink fragment-delay` command to instruct the routers to break up any large packets into fragments that will not take longer than 10 ms to cross the WAN link.

```
RTA(config)#interface multilink 1
RTA(config-if)#ppp multilink fragment-delay 10
RTA(config-if)#bandwidth 128

RTB(config)#interface multilink 1
```

```
RTB(config-if)#ppp multilink fragment-delay 10
RTB(config-if)#bandwidth 128
```

The **bandwidth** command is an essential element as the router uses this value in conjunction with the **fragment-delay** command to determine the size of the fragments.

## Step 8

The **ppp multilink fragment-delay** command used in Step 7 will break up larger packets. PPP will still deliver all the fragments belonging to one packet before forwarding any new packets. This behavior can be changed by using the **ppp multilink interleave** command.

```
RTA(config)#interface multilink 1
RTA(config-if)#ppp multilink interleave

RTB(config)#interface multilink 1
RTB(config-if)#ppp multilink interleave
```

## Step 9

At this point large packets have been broken into smaller fragments and PPP will interleave new packets subject to whatever queuing strategy is in place. However, if the queuing strategy is first-in, first-out (FIFO), there is a good chance the voice packets will get caught behind a stream of fragmented data packets. By turning on weighted fair queuing (WFQ), intermittent traffic is given a better chance of accessing the media.

```
RTA(config)#interface multilink 1
RTA(config-if)#fair-queue

RTB(config)#interface multilink 1
RTB(config-if)#fair-queue
```

## Step 10

It is now necessary to tell the router that the virtual interface **multilink 1** will use physical interface **S0/0**.

```
RTA(config)#interface serial 0/0
RTA(config-if)#ppp multilink group 1

RTB(config)#interface serial 0/0
RTB(config-if)#ppp multilink group 1
```

## Step 11

Verify the operation of the PPP multilink bundle using the **show ppp multilink** command.

```
RTA#show ppp multilink

Multilink1, bundle name is RTB
Bundle up for 00:00:26
0 lost fragments, 0 reordered, 0 unassigned
0 discarded, 0 lost received, 1/255 load
0x1F received sequence, 0x21 sent sequence
Member links: 1 active, 0 inactive (max not set, min not set)
Se0/0, since 00:00:26, last rcvd seq 00001E 160 weight, 152 frag size
RTA#
```

## Step 12

Verify the operation of the Link Fragmentation and Interleave feature using the `debug ppp multilink fragments` command.

**Note:** This `debug` command generates a large volume of console information. It may be difficult to turn off the debug if a file is being copied across the WAN link. This command should be used with caution in a live network environment.

```
RTA#debug ppp multilink fragments
Multilink fragments debugging is on
RTA#
*Mar 1 00:55:58.063: Se0/0 MLP: O frag 00004FD1 size 160
*Mar 1 00:55:58.067: Se0/0 MLP: I frag C00031FA size 50 direct
*Mar 1 00:55:58.083: Se0/0 MLP: O frag 00004FD2 size 160
*Mar 1 00:55:58.083: Se0/0 MLP: O frag 00004FD3 size 160
*Mar 1 00:55:58.103: Se0/0 MLP: O frag 00004FD4 size 160
*Mar 1 00:55:58.103: Se0/0 MLP: O frag 00004FD5 size 160
*Mar 1 00:55:58.123: Se0/0 MLP: O frag 00004FD6 size 160
*Mar 1 00:55:58.123: Se0/0 MLP: O frag 00004FD7 size 160
*Mar 1 00:55:58.143: Se0/0 MLP: O frag 40004FD8 size 94
*Mar 1 00:55:58.143: Se0/0 MLP: O frag 80004FD9 size 160
*Mar 1 00:55:58.151: Se0/0 MLP: O frag 00004FDA size 160
*Mar 1 00:55:58.171: Se0/0 MLP: O frag 00004FDB size 160
*Mar 1 00:55:58.171: Se0/0 MLP: O frag 00004FDC size 160
*Mar 1 00:55:58.191: Se0/0 MLP: O frag 00004FDD size 160
```

Note that the large packets are broken down into 160 byte fragments.

How many milliseconds does it take to transmit 160 bytes at 128 kbps?

---

<b>Note</b>	Small packets that are less than 160 bytes after encapsulation, are sent without fragmentation. These packets are labeled "direct". Larger packets will be broken down into 160 byte fragments.
-------------	---

---

## Step 13

Repeat the file and copy, and `ping` test of Steps 2 - 3. Notice the reduced `ping` times are well within the latency requirements of voice traffic. If everything is configured correctly, the `ping` times should be reduced to approximately 30 ms.

## Step 14

Although the configuration performed so far is capable of providing the low latency that is required for voice traffic, the limitations of WFQ will become apparent. As more streams of traffic are added, WFQ will provide fair access to each of the streams. This eventually results in insufficient bandwidth to support a voice call. What is really required is guaranteed bandwidth for the voice traffic.

To provide guaranteed bandwidth, begin by identifying the voice traffic with access-lists:

```
RTA(config)#access-list 102 permit udp any any range 16384 32767
RTA(config)#access-list 103 permit tcp any eq 1720 any
RTA(config)#access-list 103 permit tcp any any eq 1720

RTB(config)#access-list 102 permit udp any any range 16384 32767
RTB(config)#access-list 103 permit tcp any eq 1720 any
RTB(config)#access-list 103 permit tcp any any eq 1720
```

The User Datagram Protocol (UDP) represents voice and the TCP represents call management. They are defined using separate access-lists as the quality of service (QoS) requirements differs.

```

RTA#show access-lists
Extended IP access list 102
    permit udp any any range 16384 32767
Extended IP access list 103
    permit tcp any eq 1720 any
    permit tcp any any eq 1720

RTB(config)#access-list 102 permit udp any any range 16384 32767
RTB(config)#access-list 103 permit tcp any eq 1720 any
RTB(config)#access-list 103 permit tcp any any eq 1720

RTB#show access-lists
Extended IP access list 102
    permit udp any any range 16384 32767
Extended IP access list 103
    permit tcp any eq 1720 any
    permit tcp any any eq 1720

```

## Step 15

Now create **class-maps** that define the classes of traffic using the Access Control Lists (ACLs).

```

RTA(config)#class-map match-all VOICE-SIGNALING
RTA(config-cmap)#match access-group 103
RTA(config-cmap)#class-map match-all VOICE-TRAFFIC
RTA(config-cmap)#match access-group 102

RTB(config)#class-map match-all VOICE-SIGNALING
RTB(config-cmap)#match access-group 103
RTB(config-cmap)#class-map match-all VOICE-TRAFFIC
RTB(config-cmap)#match access-group 102

RTA#show class-map
Class Map match-any class-default (id 0)
    Match any

Class Map match-all VOICE-TRAFFIC (id 2)
    Match access-group 102

Class Map match-all VOICE-SIGNALING (id 1)
    Match access-group 103

RTB#show class-map
Class Map match-any class-default (id 0)
    Match any

Class Map match-all VOICE-TRAFFIC (id 2)
    Match access-group 102

Class Map match-all VOICE-SIGNALING (id 1)
    Match access-group 103

```

## Step 16

Create a **policy-map** that defines the QoS requirements for the classes of traffic. Ensure that 8 kbps of bandwidth is available to support voice signaling. Voice traffic is priority queued and all other traffic is subject to a weighted fair queue.

```

RTA(config)#policy-map VOICE-POLICY
RTA(config-pmap)#class VOICE-SIGNALING

```



```

RTA(config-pmap-c)#bandwidth 8
RTA(config-pmap-c)#class VOICE-TRAFFIC
RTA(config-pmap-c)#priority 48
RTA(config-pmap-c)#class class-default
RTA(config-pmap-c)#fair-queue

RTB(config)#policy-map VOICE-POLICY
RTB(config-pmap)#class VOICE-SIGNALING
RTB(config-pmap-c)#bandwidth 8
RTB(config-pmap-c)#class VOICE-TRAFFIC
RTB(config-pmap-c)#priority 48
RTB(config-pmap-c)#class class-default
RTB(config-pmap-c)#fair-queue

RTA#show policy-map
  Policy Map VOICE-POLICY
    Class VOICE-SIGNALING
      Bandwidth 8 (kbps) Max Threshold 64 (packets)
    Class VOICE-TRAFFIC
      Strict Priority
      Bandwidth 48 (kbps) Burst 1200 (Bytes)
    Class class-default
      Flow based Fair Queueing Max Threshold 64 (packets)

RTB#show policy-map
  Policy Map VOICE-POLICY
    Class VOICE-SIGNALING
      Bandwidth 8 (kbps) Max Threshold 64 (packets)
    Class VOICE-TRAFFIC
      Strict Priority
      Bandwidth 48 (kbps) Burst 1200 (Bytes)
    Class class-default
      Flow based Fair Queueing Max Threshold 64 (packets)

```

## Step 17

Apply the QoS policy to the outbound WAN interfaces.

```

RTA(config)#interface multilink 1
RTA(config-if)#no fair-queue
RTA(config-if)#service-policy output VOICE-POLICY

RTB(config)#interface multilink 1
RTB(config-if)#no fair-queue
RTB(config-if)#service-policy output VOICE-POLICY

```

---

<b>Note</b>	It may be necessary to remove fair-queueing with the "no fair-queue" command before the router will accept the service-policy statement.
-------------	--

---

Verify that the QoS policy has been applied correctly to the interface as follows:

```

RTA#show policy-map interface
Multilink1

Service-policy output: VOICE-POLICY

Class-map: VOICE-SIGNALING (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: access-group 103

```



```

Queueing
  Output Queue: Conversation 41
  Bandwidth 8 (kbps) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 0/0
  (depth/total drops/no-buffer drops) 0/0/0

Class-map: VOICE-TRAFFIC (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: access-group 102
  Queueing
    Strict Priority
    Output Queue: Conversation 40
    Bandwidth 48 (kbps) Burst 1200 (Bytes)
    (pkts matched/bytes matched) 0/0
    (total drops/bytes drops) 0/0

Class-map: class-default (match-any)
  1723 packets, 2256767 bytes
  5 minute offered rate 64000 bps, drop rate 0 bps
  Match: any
  Queueing
    Flow Based Fair Queueing
    Maximum Number of Hashed Queues 32
    (total queued/total drops/no-buffer drops) 5/0/0
RTA#

```

```

RTB#show policy-map interface
Multilink1

```

Service-policy output: VOICE-POLICY

```

Class-map: VOICE-SIGNALING (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: access-group 103
  Queueing
    Output Queue: Conversation 41
    Bandwidth 8 (kbps) Max Threshold 64 (packets)
    (pkts matched/bytes matched) 0/0
    (depth/total drops/no-buffer drops) 0/0/0

Class-map: VOICE-TRAFFIC (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: access-group 102
  Queueing
    Strict Priority
    Output Queue: Conversation 40
    Bandwidth 48 (kbps) Burst 1200 (Bytes)
    (pkts matched/bytes matched) 0/0
    (total drops/bytes drops) 0/0

Class-map: class-default (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: any
  Queueing
    Flow Based Fair Queueing
    Maximum Number of Hashed Queues 32
    (total queued/total drops/no-buffer drops) 0/0/0

```