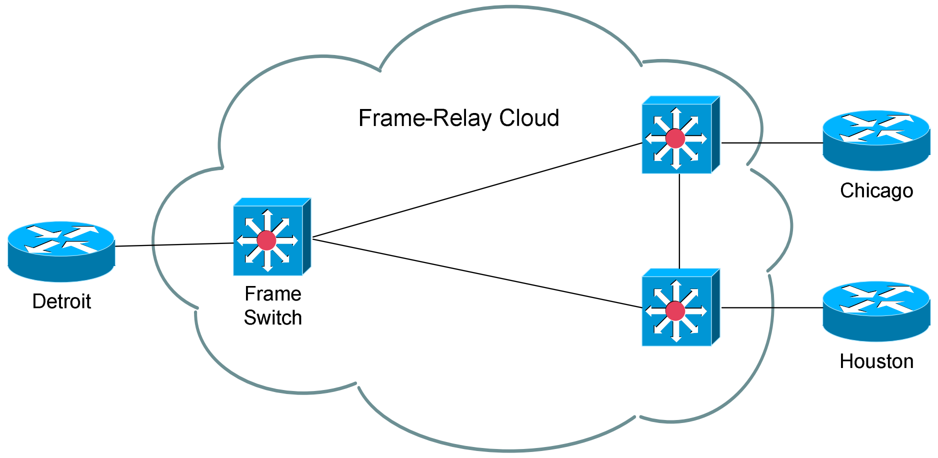
- Frame-Relay -

Frame-Relay



Frame-Relay is a packet-switched technology, which shares bandwidth between users on the switched network. Frame-relay service providers assume that all clients will *not* need the full capacity of their bandwidth at all times. Thus, in general, frame-relay is less expensive than dedicated WAN lines, but customers are not guaranteed bandwidth.

All locations plug into the frame relay "cloud," which is a conglomeration of dozens or hundreds of Frame-Relay switches and routers. The cloud is the Frame provider's network, and the customer has no control (or even knowledge) of what occurs inside that infrastructure.

For communication to occur between locations, v**irtual circuits (VC)** must be created. A VC is a one-way path through the Frame-Relay cloud.

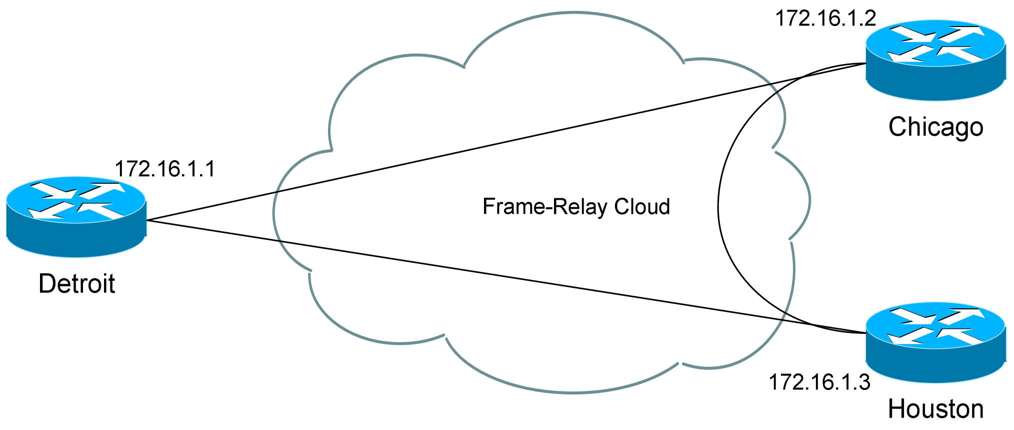
In the above example, in order to establish full communication between Detroit and Houston, we would need to create two virtual circuits:

- A virtual circuit between Detroit and Houston
- A separate virtual circuit between Houston and Detroit

Frame-relay circuits can either be **permanent (PVC)**, or **switched (SVC).** A permanent virtual circuit is always kept active, and is the most common virtual circuit. A switched virtual circuit is created only when traffic needs to be sent, and is torn down when communication is complete.

Virtual circuits are identified with **Data Link Connection Identifiers (DLCIs).** Frame-Relay switches make decisions based on DLCIs, whereas Ethernet switches make decisions based on MAC addresses.

Frame-Relay Global vs Local DLCI



The difference between a globally or locally significant DLCI is all based on perspective. Remember that a DLCI identifies a one-way virtual circuit. For example, the connection between Detroit and Chicago would be considered one virtual circuit, and Chicago to Detroit would be a separate virtual circuit.

To get this to work, we need to map a DLCI to an IP address. For example, on router Detroit, we're going to create a virtual circuit to router Chicago. We'll assign it a DLCI of "102," and point it to Chicago's IP address.

We call this *locally* significant, because it only affects the interface on the Detroit router. We could, on the Chicago router, set a DLCI of "102" and point it to the IP address of the Detroit router. Because the DLCI is set on a different router (and interface), there will be no conflict.

When we set a *globally* significant DLCI, it is really only an administrative feature. It means that an administrator has consciously decided that all virtual circuits going to Chicago will be set to DLCI 102 (or whatever DLCI number you choose), whether it is from Detroit or Houston.

In essence, you are symbolically assigning the DLCI of 102 to the Chicago *location*. Keep in mind that you are still *technically* assigning the DLCI to the virtual circuits connecting to Chicago.

Virtual circuits pointing to other locations will be configured with *different* DLCIs (Detroit could be 101; Houston could be 103, etc.). The advantage to this is that it is now easy to determine the destination of a packet, based on its DLCI.

Frame-Relay CIR

Bandwidth is provided on a best effort basis in Frame-Relay.

The Frame provider and customer agree on a **Committed Information Rate (CIR),** which is not always a guarantee of bandwidth. The provider will give a best effort to meet the CIR, which is measured in bits per second:

- 256000 bps
- 512000 bps
- 1544000 bps

The above are examples of possible CIR settings, though technically the CIR can be set to anything. At times, bandwidth speeds can **burst (Be)** above the CIR. However, speeds above the CIR are certainly not guaranteed, and if the Frame Network becomes congested, any data exceeding the CIR becomes **Discard Eligible,** and is at risk of being dropped.

Frame-Relay Encapsulation Types

On Cisco routers, two possible Frame encapsulations can be configured on the router's serial ports.

- **Cisco –** the default, and proprietary, Frame-Relay encapsulation
- **IETF –** the standardized Frame-Relay encapsulation.

Frame-Relay Local Management Interface (LMI)

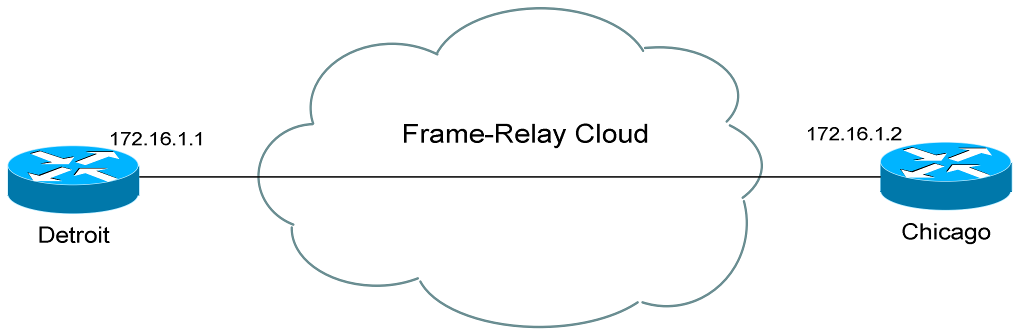
LMI is the type of signaling used between your router and your provider's Frame-Relay switch. LMI provides status updates of Virtual Circuits between the Frame switch and the router.

There are three LMI-types:

- **Cisco –** default and proprietary (naturally)
- **ANSI**
- **Q.933a**

LMI type is auto-sensed on Cisco routers, but can be manually set if desired.

Frame-Relay Point-to-Point Configuration Example



Point-to-Point is the simplest form of Frame-Relay configuration. Remember that PVCs are only one-way circuits, and thus we need to create *two* PVCs in order for full communication to occur.

Configuration on the Detroit and Chicago routers would be as follows:

Detroit Router:

```
Router(config)# int s0/0
Router(config-if)#  ip address 172.16.1.1 255.255.0.0
Router(config-if)#  encapsulation frame-relay
Router(config-if)#  frame-relay lmi-type q933a
Router(config-if)#  frame-relay interface-dlci 102
Router(config-if)#  no shut
```

Chicago Router:

```
Router(config)# int s0/0
Router(config-if)#  ip address 172.16.1.2 255.255.0.0
Router(config-if)#  encapsulation frame-relay
Router(config-if)#  frame-relay lmi-type q933a
Router(config-if)#  frame-relay interface-dlci 201
Router(config-if)#  no shut
```

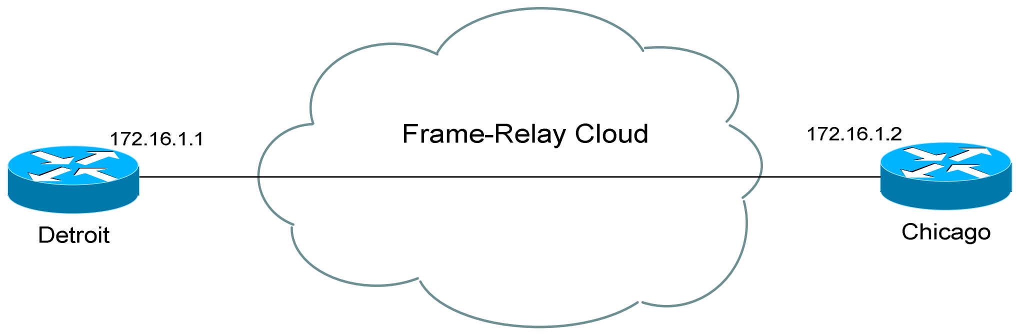
Notice that both routers are in the same IP subnet.

The *encapsulation frame-relay* command sets the frame encapsulation type to the default of *cisco*. The encapsulation must be the **same** on both routers. To change the default encapsulation type, simply append the *ietf* keyword to the *encapsulation frame-relay* command:

```
Router(config)# int s0/0
Router(config-if)#  ip address 172.16.1.1 255.255.0.0
Router(config-if)#  encapsulation frame-relay ietf
```

The *frame-relay lmi-type* command sets the signaling type. The Frame-Relay provider dictates which LMI-type to use. Remember that *cisco* is the default LMI-type, and that LMI is usually auto-sensed.

Frame-Relay Point-to-Point Configuration Example (continued)



Detroit Router:

```
Router(config)#  int s0/0
Router(config-if)#  ip address 172.16.1.1 255.255.0.0
Router(config-if)#  encapsulation frame-relay
Router(config-if)#  frame-relay lmi-type q933a
Router(config-if)#  frame-relay interface-dlci 102
Router(config-if)#  no shut
```

Chicago Router:

```
Router(config)#  int s0/0
Router(config-if)#  ip address 172.16.1.2 255.255.0.0
Router(config-if)#  encapsulation frame-relay
Router(config-if)#  frame-relay lmi-type q933a
Router(config-if)#  frame-relay interface-dlci 201
Router(config-if)#  no shut
```

The *frame-relay interface-dlci* command identifies the one-way PVC. The connection between Detroit and Chicago has been assigned DLCI 102. The connection between Chicago and Detroit has been assigned DLCI 201.

The Frame-Relay provider usually dictates which DLCI numbers to use, as the provider's Frame switch is configured with the appropriate DLCI information.

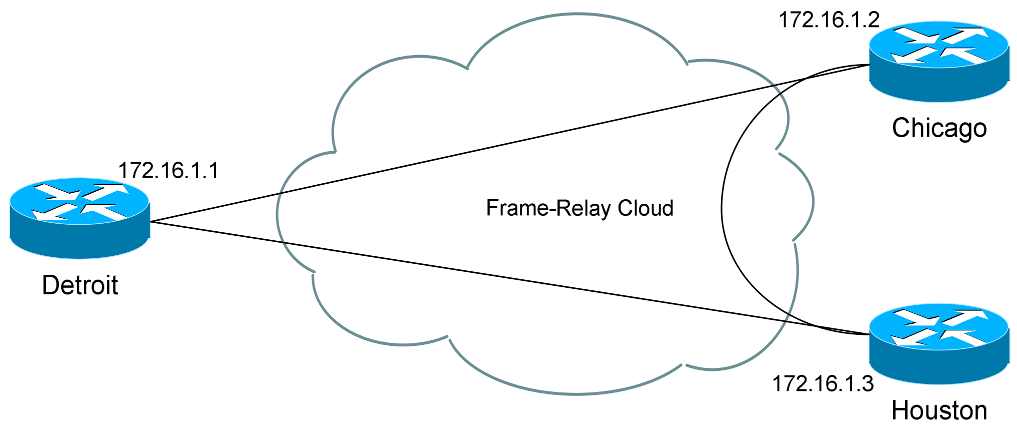
The router can actually receive all PVC and DLCI information directly from the Frame-Relay switch via LMI, using **Inverse-ARP.** Inverse-ARP is **enabled by default** on Cisco routers.

Thus, if the Frame-Relay switch is configured correctly, the *frame-relay interface-dlci* command could theoretically be removed, and the frame-relay connection will still work.

There are circumstances when DLCIs should be manually assigned. Inverse-ARP can be disabled on an interface with the following command:

```
Router(config)#  int s0/0
Router(config-if)#  no frame-relay inverse-arp
```

Frame-Relay Full Mesh Configuration Example



Consider the above example, a full mesh between three locations. All routers can still belong to the same IP subnet; however, DLCI's must now be *mapped* to IP addresses, as multiple PVCs are necessary on each interface.

This can be dynamically configured via **Inverse-Arp**, which is enabled by default (as stated earlier). Otherwise, the DLCI-to-IP mapping can be performed manually. Looking at the Detroit and Chicago router's configuration:

Detroit Router:

```
Router(config)#  int s0/0
Router(config-if)#  ip address 172.16.1.1 255.255.0.0
Router(config-if)#  encapsulation frame-relay ietf
Router(config-if)#  no frame-relay inverse-arp
Router(config-if)#  frame-relay lmi-type ansi
Router(config-if)#  frame-relay map ip 172.16.1.2 102 broadcast
Router(config-if)#  frame-relay map ip 172.16.1.3 103 broadcast
Router(config-if)#  no shut
```

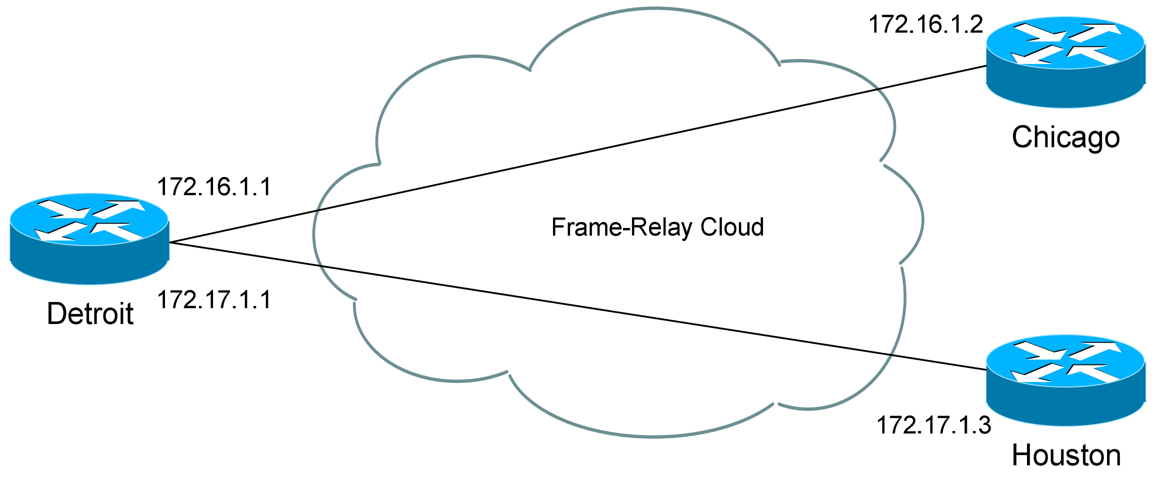
Chicago Router:

```
Router(config)#  int s0/0
Router(config-if)#  ip address 172.16.1.2 255.255.0.0
Router(config-if)#  encapsulation frame-relay ietf
Router(config-if)#  no frame-relay inverse-arp
Router(config-if)#  frame-relay lmi-type ansi
Router(config-if)#  frame-relay map ip 172.16.1.1 201 broadcast
Router(config-if)#  frame-relay map ip 172.16.1.3 203 broadcast
Router(config-if)#  no shut
```

Inverse-ARP was disabled using the *no frame-relay inverse-arp* command.

The *frame-relay map* command maps the remote router's IP address to a DLCI. On the Detroit router, a *map* was created to Chicago's IP (*172.16.1.2*), and that PVC was assigned a DLCI of *102*. The *broadcast* option allows broadcasts and multicasts to be forwarded to that address, so that routing protocols such as OSPF can form neighbor relationships.

Frame-Relay Partial Mesh Configuration Example



Full-mesh Frame-Relay environments can get quite expensive. Partial-mesh environments are often more cost-effective. A partial-mesh is essentially a hub-and-spoke design, with one **central** or **hub** location that all other locations must connect through.

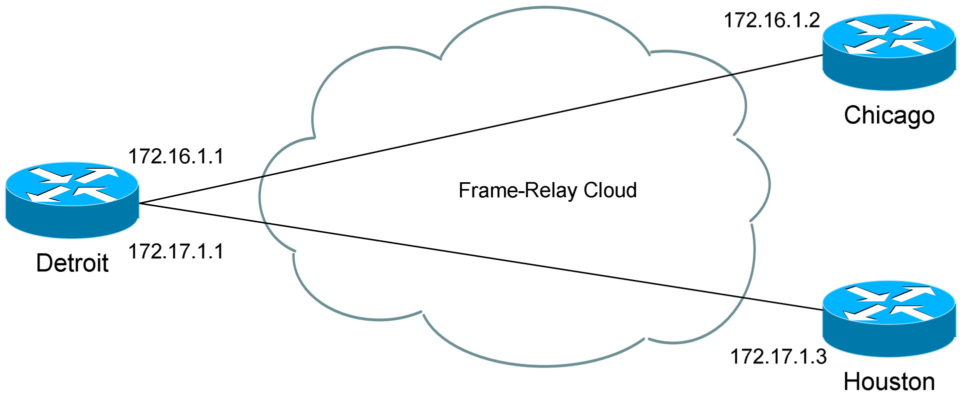
In the above example, the Detroit router serves as the hub router. In a partial-mesh environment, each **spoke** must be on a different IP subnet, which presents a special problem.

If both spokes terminate on the Detroit router's physical serial interface, **split-horizon** will prevent Chicago's routing updates from ever reaching Houston, and vice versa. Recall that split-horizon dictates that updates *received* on an interface cannot be *sent back out* the *same* interface.

Thus, on router Detroit, **sub-interfaces** must be created off of the serial interface. Sub-interfaces are **virtual** interfaces that the router treats as separate physical interfaces, providing a workaround for the split-horizon problem.

The **network type** must be specified when creating a sub-interface. A **point-to-point** sub-interface has only a single Virtual Circuit to another router. A **multipoint** sub-interface can have multiple Virtual Circuits to multiple locations.

Frame-Relay Partial Mesh Configuration Example (continued)



Configuration of the Detroit and Chicago routers would be as follows:

Detroit Router:

```
Router(config)#  int s0/0
Router(config-if)#  encapsulation frame-relay
Router(config-if)#  frame-relay lmi-type ansi

Router(config)#  int s0/0.102 point-to-point
Router(config-subif)#  no frame-relay inverse-arp
Router(config-subif)#  ip address 172.16.1.1 255.255.0.0
Router(config-subif)#  frame-relay interface-dlci 102
Router(config-subif)#  no shut

Router(config)#  int s0/0.103 point-to-point
Router(config-subif)#  no frame-relay inverse-arp
Router(config-subif)#  ip address 172.17.1.1 255.255.0.0
Router(config-subif)#  frame-relay interface-dlci 103
Router(config-subif)#  no shut
```

Chicago Router:

```
Router(config)#  int s0/0
Router(config-if)#  encapsulation frame-relay
Router(config-if)#  frame-relay lmi-type ansi


Router(config)#  int s0/0.201 point-to-point
Router(config-subif)#  no frame-relay inverse-arp
Router(config-subif)#  ip address 172.16.1.2 255.255.0.0
Router(config-subif)#  frame-relay interface-dlci 201
Router(config-subif)#  no shut
```

Notice first that the Detroit router, serving as the hub, has two sub-interfaces configured pointing to Chicago and Houston. The Chicago router only has one sub-interface pointing to Detroit.

On the Detroit router, the *int s0/0.102* command creates a sub-interface numbered *102* on the Serial0/0 interface. Using the DLCI number for the sub-interface number is an arbitrary choice, useful for documentation purposes. On the Detroit router, each sub-interface contains only one virtual circuit, thus the interface's network type was set to *point-to-point*.

Notice also that encapsulation and LMI-type information is set on the physical *interface,* but IP address and DLCI information is set on the *sub-interface.*

Frame-Relay Traffic Shaping (FRTS)

Frame-Relay's method of QoS is called **traffic-shaping**, which controls the amount of traffic sent out an interface, and dictates congestion control mechanisms.

Frame-Relay Traffic-Shaping (FRTS) is used for two purposes:

- Adhering to the Frame provider's traffic rates.
- Preventing an oversubscription of the line between hub and spoke routers.

Several terms must be understood before configuring traffic-shaping:

- **Committed Information Rate (CIR) –** the "average" traffic rate provided on a best-effort basis. By default, the CIR on a serial interface configured for traffic shaping is **56000 bits per second**.
- **Available Rate (AR) –** the maximum traffic rate, dictated either by the speed of the physical interface (using the *clock rate* command), or the restrictions of the Frame Provider.
- **Minimum CIR (MinCIR)** – the minimum traffic rate the router will "throttle" down to if congestion occurs on the Frame-Relay network (i.e., a BECN is received). This is usually the provider's guaranteed traffic rate. By default, the MinCIR is *half* that of the CIR.
- **Discard Eligible (DE) –** a bit that is set for all traffic sent above the MinCIR. Essentially, traffic that is sent above the Frame Provider's guaranteed rate can or will be dropped when congestion occurs.
- **Committed Burst (Bc)** – the amount of bits sent during a specific interval, measured as **Time Committed (Tc).** Tc is measured in milliseconds (default is 125ms, or 8 intervals a second), and determines the number of intervals per second. The CIR is derived from the Bc and Tc using the following formula:
  CIR = Bc X 1000/Tc
- **Excess Burst (Be)** – the amount of bits that can be sent exceeding the Bc (or CIR). Any bits sent at this rate will be marked as DE.

Configuring Frame-Relay (FRTS)

To configure FRTS, a **map-class** must be created:

```
Router(config)# map-class frame-relay MYCLASS
Router(config-map-class)# frame-relay cir 64000
Router(config-map-class)# frame-relay bc 8000
Router(config-map-class)# frame-relay be 0
Router(config-map-class)# frame-relay mincir 32000
Router(config-map-class)# frame-relay adaptive-shaping becn
```

A *map-class* was created for *frame-relay* called *MYCLASS*. The first three commands configure the *CIR, Bc,* and *Be* respectively.

The final commands must be used in conjunction with each other. The *adaptive-shaping* feature has been specified, indicating that the router will throttle back to the *mincir* if a *becn* is received. The router does not throttle down to the *mincir* immediately, but rather will lower the rate by 25% until either the congestion stops, or the *mincir* is reached.

A map-class applied to an interface affects *all* PVCs on that interface. Additionally, map classes can be applied to a specific PVC, providing more granular control of FRTS.

To apply a map class to an interface:

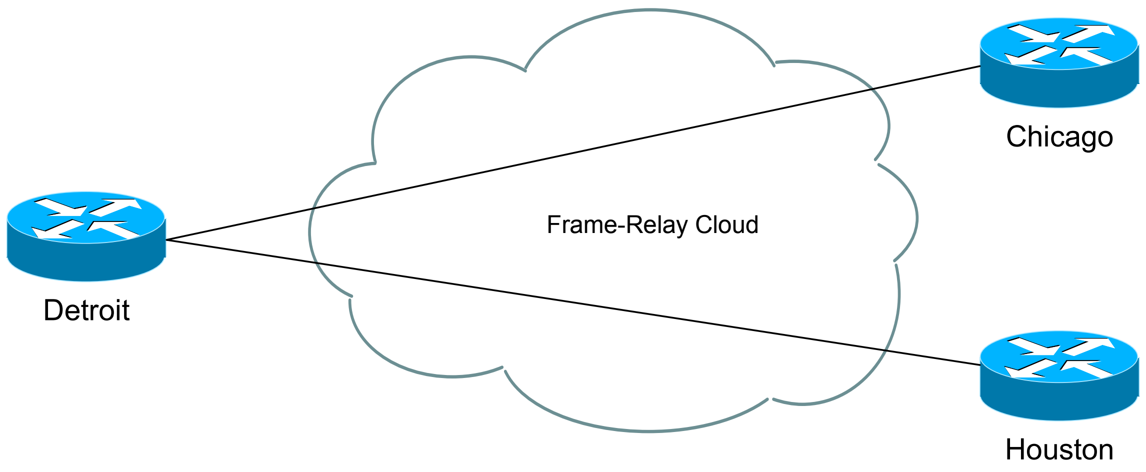
```
Router(config)# interface s0/0
Router(config-if)# encapsulation frame-relay
Router(config-if)# frame-relay traffic-shaping
Router(config-if)# frame-relay class MYCLASS
```

To apply a map class to a specific PVC:

```
Router(config)# interface s0/0
Router(config-if)# encapsulation frame-relay
Router(config-if)# frame-relay traffic-shaping
Router(config-if)# frame-relay interface-dlci 101 class MYCLASS
```

Do not forget the *frame-relay traffic-shaping* command. Once this command is configured, all PVCs are configured with the default CIR of 56,000 bps.

EIGRP and Frame-Relay



Observe the above Frame-Relay network. Two possible configuration options exist for the Detroit router:

- Configure frame-relay map statements on the physical interface
- Create separate sub-interfaces for each link, treating them as separate point-to-points.

If choosing the latter, EIGRP will treat each sub-interface as a separate link, and routing will occur with no issue.

If choosing the former, EIGRP will be faced with a split-horizon issue. Updates from Houston will not be forwarded to Chicago, and visa versa, as split horizon prevents an update from being sent out the link it was received on.

It is possible to *disable* split horizon for EIGRP:

```
Detroit(config)#  interface s0/0
Detroit(config-router)#  no ip split-horizon eigrp 10
```

Using sub-interfaces is Cisco's preferred method of circumventing the split-horizon issue, however.

Troubleshooting Frame-Relay

To view information concerning each PVC:

Router# *show frame-relay pvc*

The above command includes the following information:

- DLCI numbers
- Status of PVCs (active, inactive, deleted)
- Congestion information
- Traffic counters

To list Frame-Relay DLCI-mappings, whether manually created using the *frame-relay map* command, or created dynamically using Inverse ARP:

Router# *show frame-relay map*

To display the LMI-type configured on each interface, and LMI traffic statistics:

Router# *show frame-relay lmi*

To troubleshoot communication problems between the router and Frame-Relay switch:

Router# *debug frame-relay lmi*

To display information on packets *received* on a Frame-Relay interface:

Router# *debug frame-relay*

To display information on packets *sent* on a Frame-Relay interface:

Router# *debug frame-relay packet*