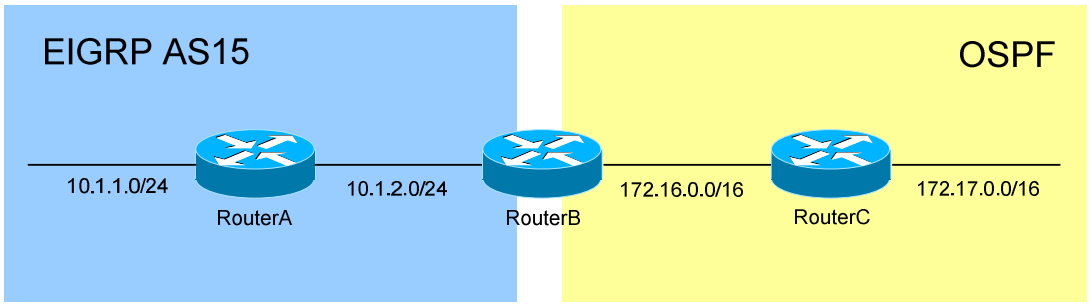
- Route Redistribution -

Route Redistribution Basics

It is preferable to employ a single routing protocol in an internetwork environment, for simplicity and ease of management. Unfortunately, this is not always possible, making multi-protocol environments common.

Route Redistribution allows routes from one routing protocol to be advertised into another routing protocol. The routing protocol receiving these redistributed routes usually marks the routes as **external.** External routes are usually *less* preferred than locally-originated routes.

At least one **redistribution point** needs to exist between the two routing domains. This device will actually run *both* routing protocols. Thus, to perform redistribution in the following example, RouterB would require at least one interface in both the EIGRP *and* the OSPF routing domains:



It is possible to redistribute from one routing protocol to the *same* routing protocol, such as between two separate OSPF domains (distinguished by unique *process ID's*). **Static routes** and **connected interfaces** can be redistributed into a routing protocol as well.

Routes will *only* be redistributed if they exist in the routing table. Routes that are simply in a topology database (for example, an EIGRP Feasible Successor), will *never* be redistributed.

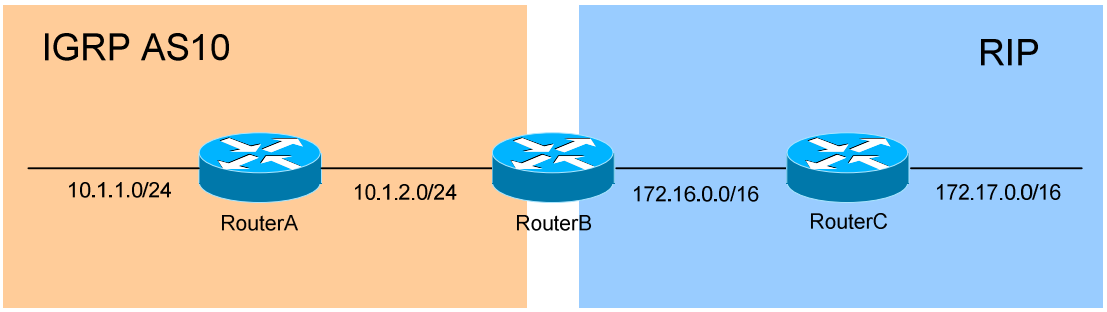
Routing **metrics** are a key consideration when performing route redistribution. With the exception of IGRP and EIGRP, each routing protocol utilizes a unique (and thus **incompatible)** metric. Routes redistributed from the *injecting* protocol must be manually (or globally) stamped with a metric that is understood by the *receiving* protocol.

(Reference: http://www.cisco.com/warp/public/105/redist.html)

* * *

Redistributing into RIP

RIP is a standardized Distance-Vector routing protocol that uses **hop-count** as its distance metric. Consider the following example:



RouterB is our redistribution point between IGRP and RIP. To redistribute all IGRP routes *into* RIP:

RouterB(config)# *router rip*
RouterB(config-router)# *network 172.16.0.0*
RouterB(config-router)# *redistribute igrp 10 metric 2*

First, the *router rip* process was enabled. Next, RIP was configured to advertise the *network* of *172.16.0.0*/16. Finally, RIP was configured to *redistribute* all *igrp* routes from Autonomous System *10*, and apply a hop-count *metric* of *2* to the redistributed routes. If a metric is *not* specified, RIP will assume a metric of **0**, and **will not advertise** the redistributed routes.

Redistributing into IGRP

IGRP is a Cisco-proprietary Distance-Vector routing protocol that, by default, uses a composite of **bandwidth** and **delay** as its distance metric. IGRP can additionally consider Reliability, Load, and MTU for its metric.

Still using the above example, to redistribute all RIP routes *into* IGRP:

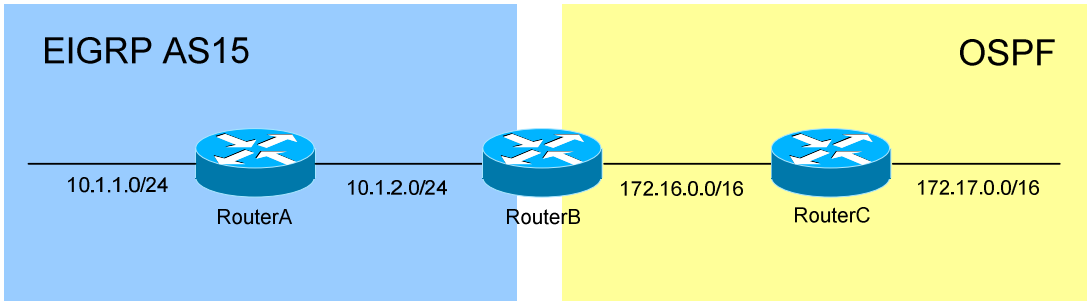
RouterB(config)# *router igrp 10*
RouterB(config-router)# *network 10.0.0.0*
RouterB(config-router)# *redistribute rip metric 10000 1000 255 1 1500*

First, the *router igrp* process was enabled for Autonomous System *10*. Next, IGRP was configured to advertise the *network* of *10.0.0.0*/8. Finally, IGRP was configured to *redistribute* all *rip* routes, and apply a *metric* of *10000* (bandwidth), *1000* (delay), *255* (reliability), *1* (load), and *1500* (MTU) to the redistributed routes.

* * *

Redistributing into EIGRP

EIGRP is a Cisco-proprietary hybrid routing protocol that, by default, uses a composite of **bandwidth** and **delay** as its distance metric. EIGRP can additionally consider Reliability, Load, and MTU for its metric.



To redistribute all OSPF routes *into* EIGRP:

```
RouterB(config)# router eigrp 15
RouterB(config-router)# network 10.1.2.0 0.0.0.255
RouterB(config-router)# redistribute ospf 20 metric 10000 1000 255 1 1500
```

First, the *router eigrp* process was enabled for Autonomous System *15*. Next, EIGRP was configured to advertise the *network* of *10.1.2.0*/24. Finally, EIGRP was configured to *redistribute* all *ospf* routes from process-ID *20*, and apply a *metric* of *10000* (bandwidth), *1000* (delay), *255* (reliability), *1* (load), and *1500* (MTU) to the redistributed routes.

It is possible to specify a *default-metric* for **all** redistributed routes:

```
RouterB(config)# router eigrp 15
RouterB(config-router)# redistribute ospf 20
RouterB(config-router)# default-metric 10000 1000 255 1 1500
```

RIP and IGRP also support the *default-metric* command. Though IGRP/EIGRP use only bandwidth and delay by default to compute the metric, it is still necessary to specify *all* five metrics when redistributing. If the *default-metric* or a manual metric is not specified, IGRP/EIGRP will assume a metric of **0**, and **will not advertise** the redistributed routes.

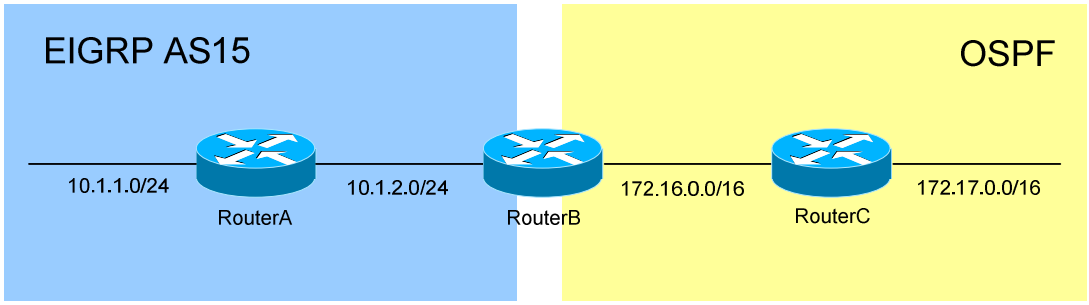
Redistribution will **occur automatically** between IGRP and EIGRP on a router, if both processes are using the same Autonomous System number.

EIGRP, by default, will auto-summarize internal routes unless the *no auto-summary* command is used. However, EIGRP ***will not auto-summarize external routes*** unless a connected or internal EIGRP route exists in the routing table from the same major network of the external routes.

* * *

Redistributing into OSPF

OSPF is a standardized Link-State routing protocol that uses **cost** (based on bandwidth) as its link-state metric. An OSPF router performing redistribution automatically becomes an **ASBR**.



To redistribute all EIGRP routes *into* OSPF:

RouterB(config)# *router ospf 20*
RouterB(config-router)# *network 172.16.0.0 0.0.255.255 area 0*
RouterB(config-router)# *redistribute eigrp 15*
RouterB(config-router)# *default-metric 30*

First, the *router ospf* process was enabled with a process-ID of *20*. Next, OSPF was configured to place any interfaces in the *network* of *172.16.0.0*/16 into *area 0*. Then, OSPF will *redistribute* all *eigrp* routes from AS *15*. Finally, a *default-metric* of *30* was applied to all redistributed routes.

If the *default-metric* or a manual metric is not specified for the redistributed routes, a **default metric of 20** will be applied to routes of all routing protocols except for BGP. Redistributed BGP routes will have a **default metric of 1** applied by OSPF.

By default, OSPF will only redistribute **classful routes** into the OSPF domain. To configure OSPF to accept **subnetted networks** during redistribution, the *subnets* parameter must be used:

RouterB(config)# *router ospf 20*
RouterB(config-router)# *redistribute eigrp 15 subnets*

Routes redistributed into OSPF are marked **external**. OSPF identifies two types of external routes, **Type-1** (which is *preferred*) and **Type-2** (which is *default*). To change the type of redistributed routes:

RouterB(config)# *router ospf 20*
RouterB(config-router)# *redistribute eigrp 15 subnets metric-type 1*

* * *

Redistributing Static and Connected Routes

Redistributing **static routes** into a routing protocol is straightforward:

RouterB(config)# *router eigrp 15*
RouterB(config-router)# *redistribute static*

Redistributing networks on **connected** interfaces into a routing protocol is equally straightforward:

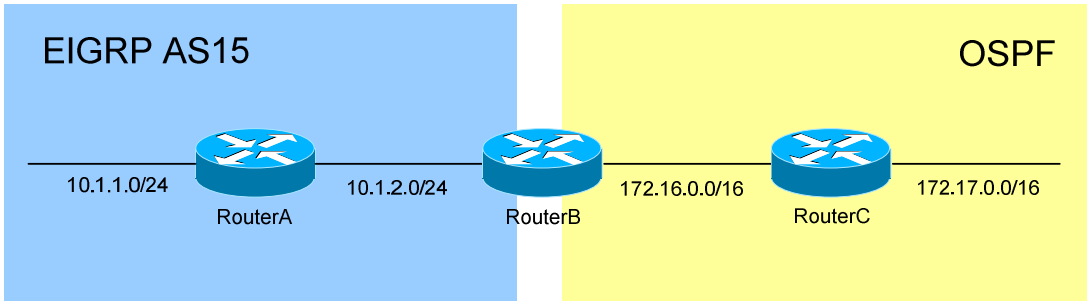
RouterB(config)# *router eigrp 15*
RouterB(config-router)# *redistribute connected*

The above commands redistribute *all* connected networks into EIGRP. **Route-maps** can be used to provide more granular control:

RouterB(config)# *route-map CONNECTED permit 10*
RouterB(config-route-map)# *match interface fa0/0, fa0/1, s0/0, s0/1*

RouterB(config)# *router eigrp 15*
RouterB(config-router)# *redistribute connected route-map CONNECTED*

Connected networks can be *indirectly* redistributed into a routing protocol. Recall that routes will *only* be redistributed if they exist in the routing table, and consider again the following example:



If RouterB is configured as follows:

RouterB(config)# *router eigrp 15*
RouterB(config-router)# *network 10.1.2.0 0.0.0.255*

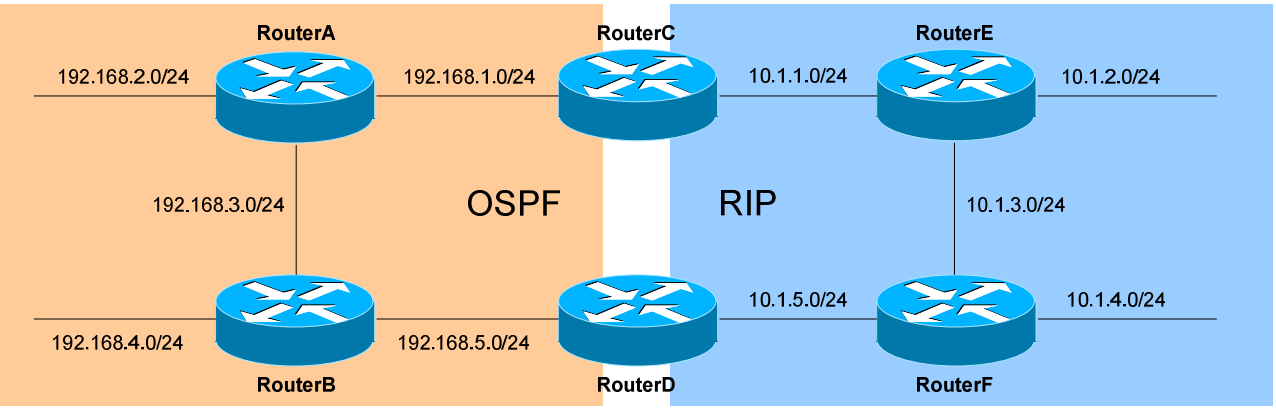
RouterB will *advertise* the 10.1.2.0/24 network to RouterA, but it *will not* have an *EIGRP route* in its routing table for that network, as the network is directly connected.

Despite this, when redistributing EIGRP into OSPF, the 10.1.2.0/24 is *still* injected into OSPF. The *network 10.1.2.0 0.0.0.255* command under the EIGRP process will *indirectly* redistribute this network into OSPF.

* * *

Pitfalls of Route Redistribution – Administrative Distance

Route redistribution introduces unique problems when there are multiple points of redistribution. Consider the following diagram:



The first issue is caused by **Administrative Distance (AD)**, which determines which routing protocol is "trusted" the most. By default, OSPF routes have an AD of 110, whereas RIP routes have an AD of 120. Lowest AD is preferred, thus making the OSPF routes the most trusted.

Assume mutual redistribution has been performed on RouterC and RouterD. The following networks will be injected *from* RIP *into* OSPF: 10.1.1.0/24, 10.1.2.0/24, 10.1.3.0/24, 10.1.4.0/24, and 10.1.5.0/24.

RouterC will eventually receive OSPF routes to the above networks from RouterD, *in addition* to the RIP routes already in its table. Likewise, RouterD will receive OSPF routes to these networks from RouterC.

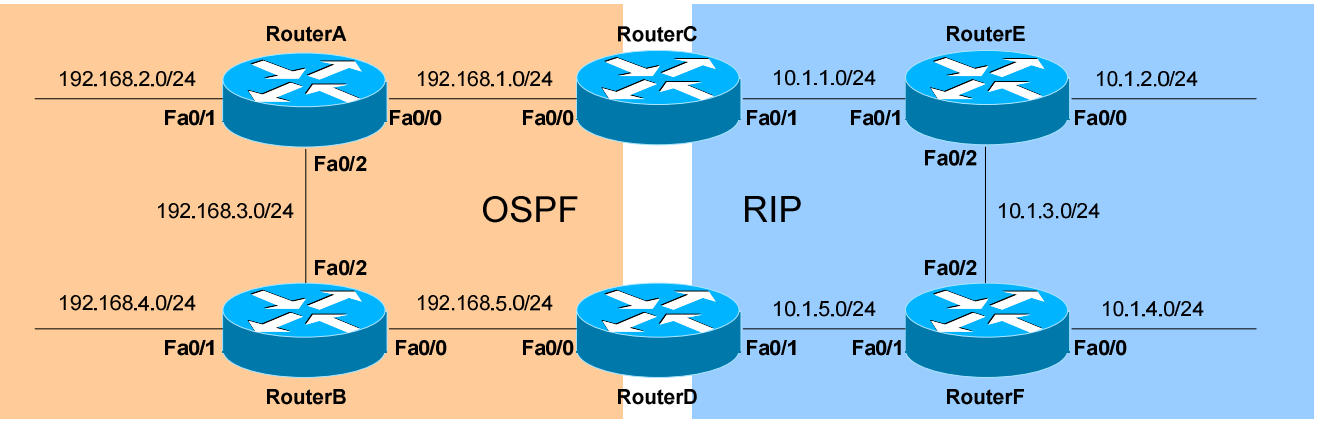
Because OSPF's AD is lower than RIP's, both RouterC and RouterD will prefer the *sub-optimal* path through OSPF to reach the *non-connected* networks. Thus, RouterC will choose the OSPF route for all the 10.x.x.x/24 networks except for 10.1.1.0/24, as it is already directly connected.

This actually creates a **routing loop.** RouterC will prefer the OSPF path through RouterA to reach the 10.x.x.x networks (except for 10.1.1.0/24), and RouterA will likely consider RouterC its shortest path to reach those same networks. Traffic will be continuously looped between these two routers.

Even if RouterC managed to send the traffic through RouterA and RouterB to RouterD, the preferred path to the 10.x.x.x networks for RouterD is *still* through OSPF. Thus, the routing loop is inevitable.

* * *

Pitfalls of Route Redistribution – Administrative Distance (continued)



There are two methods to correct this particular routing loop. The first method involves filtering incoming routes using a **distribution-list**, preventing RouterC and RouterD from accepting any routes that originated in RIP from their OSPF neighbors.

RouterC's configuration would be as follows:

RouterC(config)# *access-list 10 deny 10.1.2.0 0.0.0.255*
RouterC(config)# *access-list 10 deny 10.1.3.0 0.0.0.255*
RouterC(config)# *access-list 10 deny 10.1.4.0 0.0.0.255*
RouterC(config)# *access-list 10 deny 10.1.5.0 0.0.0.255*
RouterC(config)# *access-list 10 permit any*

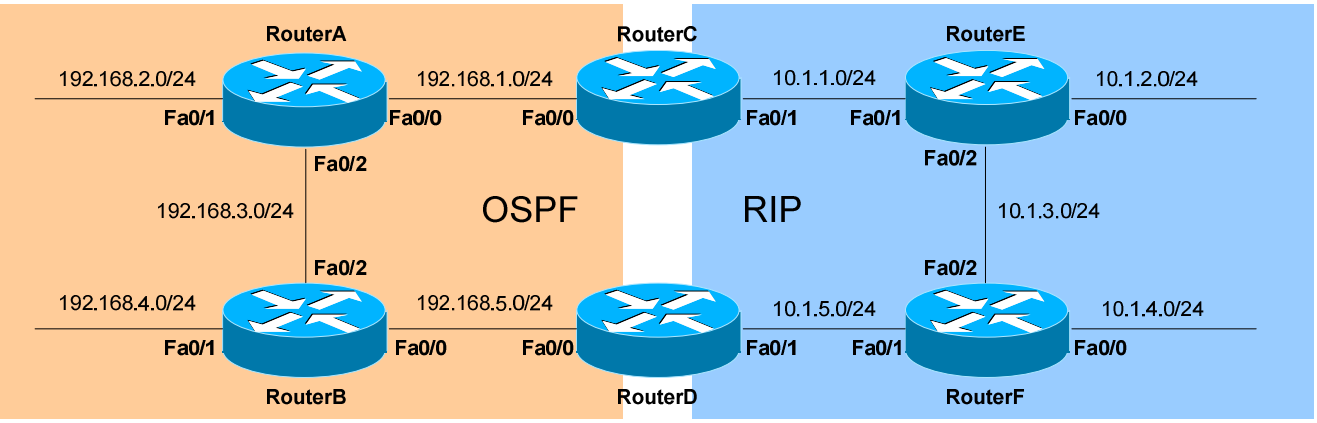
RouterC(config)# *router ospf 20*
RouterC(config-router)# *distribute-list 10 in fastethernet0/0*

An *access-list* was created that is *deny*ing the RIP networks in question, and permitting all other networks. Under the OSPF process, a *distribute-list* is created for routes coming *in*bound off of the *fastethernet0/0* interface. The *access-list* and *distribute-list* numbers must match. RouterD's configuration would be similar.

This prevents each router from building OSPF routes for the networks that originated in RIP, and thus eliminates the possibility of a loop. However, redundancy is also destroyed – if RouterC's *fa0/1* interface were to fail, it could not choose the alternate path through OSPF.

* * *

Pitfalls of Route Redistribution – Administrative Distance (continued)



The second method involves using the ***distance*** command to adjust the AD of specific routes. This can accomplished two ways:

- Lowering the AD of the local RIP-learned routes
- Raising the AD of the external OSPF-learned routes

To force the RIP routes to be preferred, RouterC's configuration would be as follows:

```
RouterC(config)# access-list 10 permit 10.1.2.0 0.0.0.255
RouterC(config)# access-list 10 permit 10.1.3.0 0.0.0.255
RouterC(config)# access-list 10 permit 10.1.4.0 0.0.0.255
RouterC(config)# access-list 10 permit 10.1.5.0 0.0.0.255
RouterC(config)# access-list 10 deny any

RouterC(config)# router rip
RouterC(config-router)# distance 70 10.1.1.0 0.0.0.255 10
```

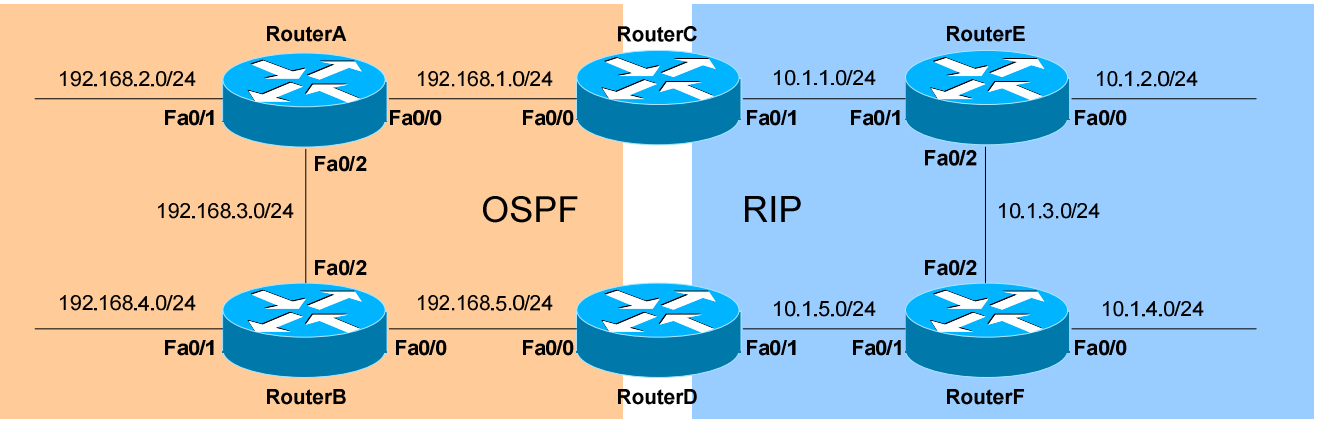
An *access-list* was created that is *permit*ting the RIP networks in question, and denying all other networks. Under the RIP process, an administrative *distance* of *70* is applied to updates from routers on the *10.1.1.0* network, for the specific networks matching access-list *10*. RouterD's configuration would be similar.

Thus, the RIP-originated networks will now have a lower AD than the redistributed routes from OSPF. The loop has again been eliminated. Another solution would be to raise the AD of the external OSPF routes. OSPF provides a simple mechanism to accomplish this:

```
RouterC(config)# router ospf 20
RouterC(config-router)# distance ospf external 240
```

* * *

Pitfalls of Route Redistribution – Route Feedback



A routing loop is only one annoying issue resulting from the above design. **Route feedback** is another problem that must be addressed.

OSPF routes redistributed into RIP on RouterC will eventually reach RouterD, and then be redistributed *again* back into OSPF. This is a basic example of route feedback.

Depending on the metrics used, this could potentially cause RouterB to prefer the route through RouterD (and through the RIP domain), to reach the 192.168.2.0/24 network. This is an obvious example of **suboptimal routing.**

Thus, routes that originated in a routing domain should not to be *re-injected* into that domain. Distribution-lists and the *distance* command can be utilized to accomplish this, but **route tags** may provide a more robust solution.

Tagging routes provides a mechanism to both identify and filter those routes further along in the routing domain. A route retains its tag as it passes from router to router. Thus, if a route is tagged when redistributed into RIP on RouterC, that same route can be selectively filtered once it is advertised to RouterD.

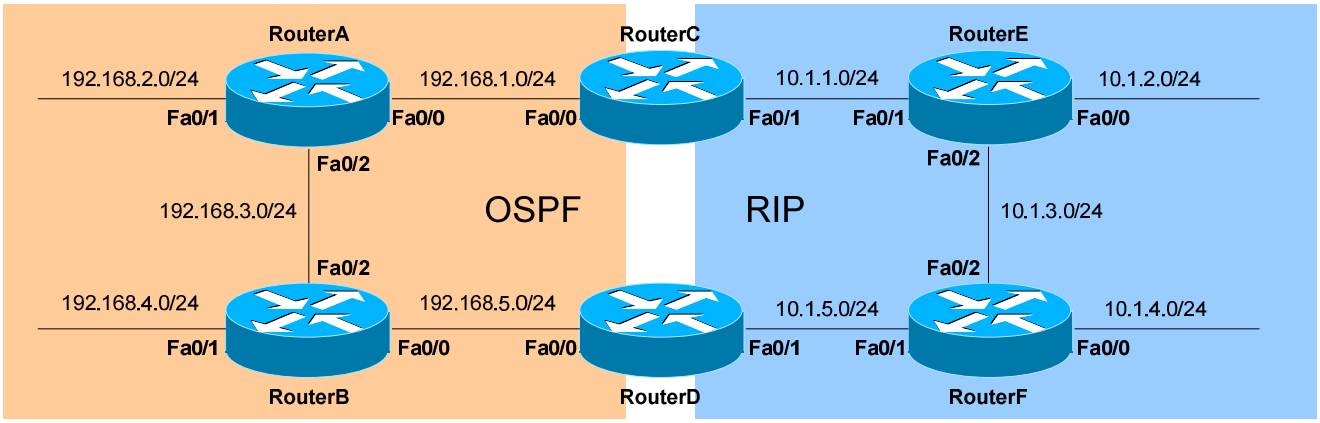
Route tags are applied using **route-maps**. Route-maps provide a sequential list of commands, each having a *permit* or *deny* result:

RouterC(config)# *route-map OSPF2RIP deny 5*
RouterC(config-route-map)# *match tag 33*
RouterC(config-route-map)# *route-map OSPF2RIP permit 15*
RouterC(config-route-map)# *set tag 44*

Route-maps are covered in great detail in a separate guide.

* * *

Pitfalls of Route Redistribution – Route Feedback (continued)



The full configuration necessary on RouterC would be as follows:

```
RouterC(config)# route-map OSPF2RIP deny 5
RouterC(config-route-map)# match tag 33
RouterC(config-route-map)# route-map OSPF2RIP permit 15
RouterC(config-route-map)# set tag 44

RouterC(config)# router rip
RouterC(config)# redistribute ospf 20 route-map OSPF2RIP

RouterC(config)# route-map RIP2OSPF deny 5
RouterC(config-route-map)# match tag 44
RouterC(config-route-map)# route-map RIP2OSPF permit 15
RouterC(config-route-map)# set tag 33

RouterC(config)# router ospf 20
RouterC(config)# redistribute rip route-map RIP2OSPF
```

Thus, OSPF routes being redistributed into RIP are *set* with a *tag* of *44*. When RIP is redistributed back into OSPF, any route with a *tag* that *match*es *44* is denied.

Similarly, RIP routes being redistributed into OSPF are *set* with a *tag* of *33*. When OSPF is redistributed back into RIP, any route with a *tag* that *match*es 33 is denied.

The net result: routes originating from a routing domain will not redistributed *back* into that domain.

* * *