

# *MPLS VPN*

Peer to Peer VPNs

# Agenda

---

- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing

- **BGP-4 (RFC 1771) is capable of carrying routing information only for IPv4**
- **The only three pieces of information carried by BGP-4 that are IPv4 specific are**
  - the NEXT\_HOP attribute (expressed as an IPv4 address),
  - the AGGREGATOR (contains an IPv4 address)
  - the NLRI (expressed as IPv4 address prefixes)
- **Multiprotocol Extensions to BGP-4**
  - RFC 2858
  - enable it to carry routing information for multiple network layer protocols (e.g., IPv6, IPX, etc...).

- **To enable BGP-4 to support routing for multiple network layer protocols two things have to be added**
  - the ability to associate a particular network layer protocol with the next hop information
  - the ability to associate a particular network layer protocol with a NLRI
- **To identify individual network layer protocols**
  - Address Family Identifiers (AFI) are used
  - values defined in RFC 1700
    - RFC 1700 is historic, obsoleted by RFC 3232
    - RFC 3232 specifies a Online Database for ASSIGNED NUMBERS
      - [www.iana.org](http://www.iana.org)

# Address Family Numbers (RFC 1700)

Number	Description
-----	-----
0	Reserved
1	IP (IP version 4)
2	IP6 (IP version 6)
3	NSAP
4	HDLC (8-bit multidrop)
5	BBN 1822
6	802 (includes all 802 media plus Ethernet "canonical format")
7	E.163
8	E.164 (SMDS, Frame Relay, ATM)
9	F.69 (Telex)
10	X.121 (X.25, Frame Relay)
11	IPX
12	AppleTalk
13	Decnet IV
14	Banyan Vines
65535	Reserved

- **Address Family Identifier (AFI) in MP-BGP**
  - this parameter is used to differentiate routing updates of different protocols carried across the same BGP session
  - it is a 16-bit value
- **MP-BGP uses an additional Sub-Address Family Identifier (SAFI)**
  - it is a 8-bit value
    - 1 NLRI used for unicast forwarding
    - 2 NLRI used for multicast forwarding
    - 3 NLRI used for both unicast and multicast forwarding
- **Usual notation AFI/SAFI (i.e. x/y)**
  - 1/1            IP version 4 unicast
  - 1/2            IP version 4 multicast
  - 1/128         VPN-IPv4 unicast (used for MPLS-VPN)

- **Capability Advertisement Procedures are used**

- by a BGP speaker that to determine whether the speaker could use multiprotocol extensions with a particular peer or not -> RFC 3392
- done during BGP Open with Capabilities Optional Parameter (Parameter Type 2)

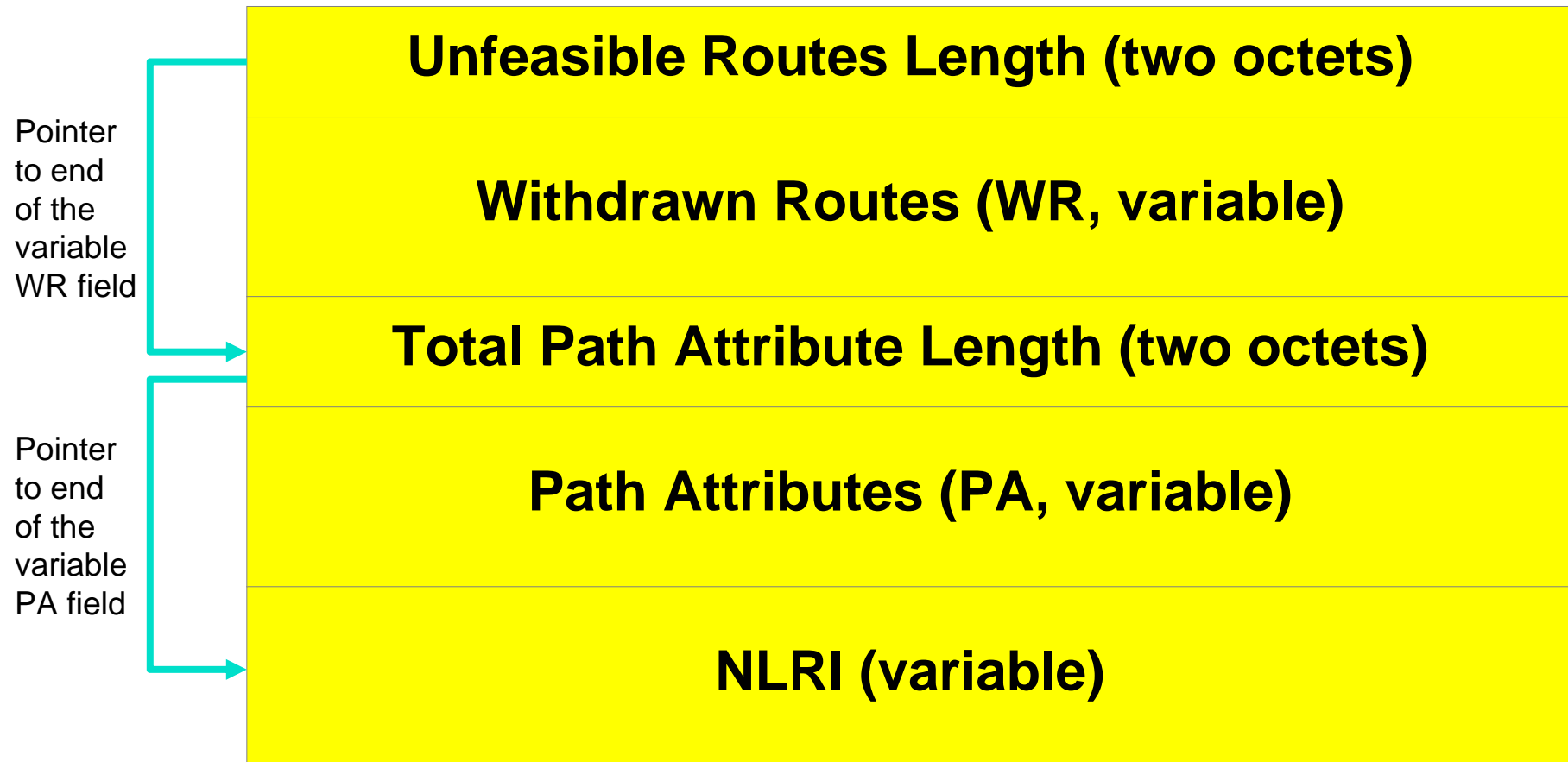
```
+-----+
| Capability Code (1 octet) |
+-----+
| Capability Length (1 octet) |
+-----+
| Capability Value (variable) |
+-----+
```

- Capability Code is unambiguously identifies individual capabilities. Capability Value is interpreted according to the value of the Capability Code field.

- **Two new attributes**
  - Multiprotocol Reachable NLRI (MP\_REACH\_NLRI)
  - Multiprotocol Unreachable NLRI (MP\_UNREACH\_NLRI)
- **MP\_REACH\_NLRI is used**
  - to carry the set of reachable destinations together with the next hop information to be used for forwarding to these destinations
- **MP\_UNREACH\_NLRI is used**
  - to carry the set of unreachable destinations
- **Both of these attributes**
  - are optional and non-transitive



# BGP Update Message Format for IPv4



# BGP Update Message Details for IPv4

- **NLRI**

- 2-tuples of (length, prefix)

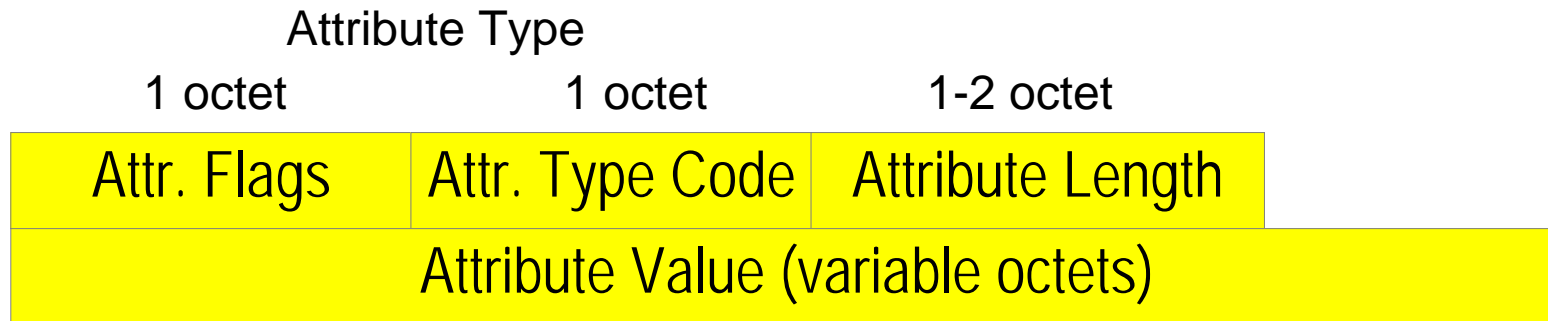
- length = number of masking bits (1 octet)
    - prefix = IP address prefix (1 - 4 octets)
    - note: prefix field contains only necessary bits to completely specify the IP address followed by enough trailing bits to make the end of the field fall on an octet boundary

- **path attributes are composed of**

- triples of (type, length, value) -> TLV notation

- attribute type (two octets)
      - 8 bit attribute flags, 8 bit attribute type code
    - attribute length (one or two octets)
      - signaled by attribute flag-bit nr.4
    - attribute value (variable length)
      - content depends on meaning signaled by attribute type code

# IPv4 Path Attribute Format / NLRI Format



## Path Attribute Format



## NLRI

# VPN-IPv4 BGP Update with MP\_Reach\_NLRI

1 octet	1 octet	1 octet
Attr. Flags	Type Code = <u>14</u>	Attribute Length
AFI = 1		SAFI = 128
Length of NHA		
Next Hop Address (NHA, 1- 4 octets)		

## Path Attribute MP\_Reach\_NLRI

1 octet
Length = 120
Label (3 octets = 24 bits)
Route Distinguisher (8 octets = 64 bits)
IPv4 address (4 octets = 32 bits)

## NLRI for VPN-IPv4

# Format of Attribute-Type

- **8 bit attribute flags**

- 1. bit (MSB)
  - optional (1) or well-known (0)
- 2. bit
  - transitive (1) or non-transitive (0)
  - only for optional; set to 1 for well-known
- 3. bit
  - partial (1) or complete (0)
  - set to 0 for well-known and optional non-transitive
- 4. bit
  - two octet (1) or one octet (0) attribute length field

- **8 bit attribute type code**

- values 1 - 16 currently defined

- **well-known**
  - must be recognized by all BGP implementations
- **well-known mandatory**
  - must be included in every Update message
    - Origin, AS\_Path, Next\_Hop
- **well-known discretionary**
  - may or may not be included in every Update message
    - Local\_Preference, Atomic\_Aggregate
- **all well-known attributes must be passed along to other BGP peers**
  - some will be updated properly first, if necessary

- **optional**

- it is not required or expected that all BGP implementation support all optional attributes
- may be added by the originator or any AS along the path
- paths are accepted regardless whether the BGP peer understands an optional attribute or not

- **handling of recognized optional attributes**

- propagation of attribute depends on meaning of the attribute
- propagation of attribute is not constrained by transitive bit of attribute flags
  - but depends on the meaning of the attribute

- **handling of unrecognized optional attribute**
  - propagation of attribute depends on transitive bit of attribute flags
  - transitive
    - paths are accepted (attribute is ignored) and attribute remains unchanged when path is passed along to other peers
    - attribute is marked as partial (bit 3 of attribute flags)
    - example: Community
  - non-transitive
    - paths are accepted, attribute is quietly ignored and discarded when path is passed along to other peers
    - example: Multi\_Exit\_Discriminator



- **Basic attributes**

- defined in RFC 1771 (Draft Standard)
- Origin
  - well-known mandatory; type 1
- AS\_Path
  - well-known mandatory; type 2
- Next\_Hop
  - well-known mandatory; type 3
- Multi\_Exit\_Discriminator MED
  - optional non-transitive; type 4
- Local\_Preference
  - well-known discretionary; type 5

- **Basic attributes (cont.)**
  - Atomic\_Aggregate
    - well-known discretionary; type 6
  - Aggregator
    - optional transitive; type 7
  
- these are the attributes that you can rely on in a multi-vendor environment

- **Advanced attributes**

- **Community**

- optional transitive; type 8
- defined in RFC 1997 (Proposed Standard)

- **Originator\_ID**

- optional non-transitive; type 9
- defined in RFC 1966 (Experimental) and RFC 2796 (Proposed Standard) -> Route Reflector

- **Cluster\_List**

- optional non-transitive; type 10
- defined in RFC 1966 (Experimental) and RFC 2796 (Proposed Standard) -> Route Reflector

- **Advanced attributes (cont.)**

- Multiprotocol Reachable NLRI

- MP\_REACH\_NLRI
- optional non-transitive; type 14
- defined in RFC 2858 (Proposed Standard) -> Multiprotocol Extensions

- Multiprotocol Unreachable NLRI

- MP\_UNREACH\_NLRI
- optional non-transitive; type 15
- defined in RFC 2858 (Proposed Standard) -> Multiprotocol Extensions

- in a multi-vendor environment carefully check implementation details

- **optional transitive attribute**
- **community is a group of destinations that share a common property**
  - group of networks which should be handled by a foreign AS in a certain way
  - community is not restricted to one network or one AS
- **community attributes are used**
  - to simplify routing policy based on logical properties rather than IP prefix or AS number (= physical location)
  - to tag routes to ensure consistent filtering or route-selection policy

- **32 bit values (range 0 - 4.294.967.200)**
- **well-known communities**
  - 0xFFFFFFFF01 ... No\_Export
  - 0xFFFFFFFF02 ... No\_Advertise
- **private communities**
  - value range 0x00010000 to 0xFFFFEFFF
  - common practice for using private communities:
    - high order 16 bit: number of AS
      - which is responsible for defining the meaning of the community
    - low order 16 bit: definition of meaning
      - might have only local significance within the defining AS

- **BGP Extended Communities Attribute**

- consists of a set of "extended communities"
  - optional transitive; type 16
  - defined in draft-ietf-idr-bgp-ext-communities-07.txt
- two important enhancements over the existing BGP Community Attribute:
  - it provides an extended range, ensuring that communities can be assigned for a plethora of uses, without fear of overlap.
  - the addition of a type field provides structure for the community space.
- Important for MPLS\_VPN
  - Route Target Community
  - Route Origin Community

- **Route Target:**

- The Route Target Community identifies one or more routers that may receive a set of routes (that carry this Community) carried by BGP. This is transitive across the Autonomous system boundary.
- It really identifies only a set of sites which will be able to use the route, without prejudice to whether those sites constitute what might intuitively be called a VPN.

- **Route Origin:**

- The Route Origin Community identifies one or more routers that inject a set of routes (that carry this Community) into BGP. This is transitive across the Autonomous system boundary.



- **Route Target and Router Origin**

- type: 2 octets (extended form of this attribute)
  - high octet -> 00, 01, 02 -> defines the structure of the value field
  - low octet -> defines the actual type
- value: 6 octets

- **Route Target:**

- high octet type: 0x00 or 0x01 or 0x02
- low octet type: 0x02

- **Route Origin:**

- high octet type: 0x00 or 0x01 or 0x02
- low octet type: 0x03

- **Structure of value field based on high octet part of type**
  - 0x00:
    - 2 octets Global Administrator Field (IANA assigned AS #)
    - 4 octets Local Administrator Field (actual value of given type contained in low octet part of type)
  - 0x01:
    - 4 octets Global Administrator Field (IP address assigned by IANA)
    - 2 octets Local Administrator Field
  - 0x02:
    - 4 octets Global Administrator Field (IANA assigned 4 octet AS #)
    - 2 octets Local Administrator Field

# Agenda

---

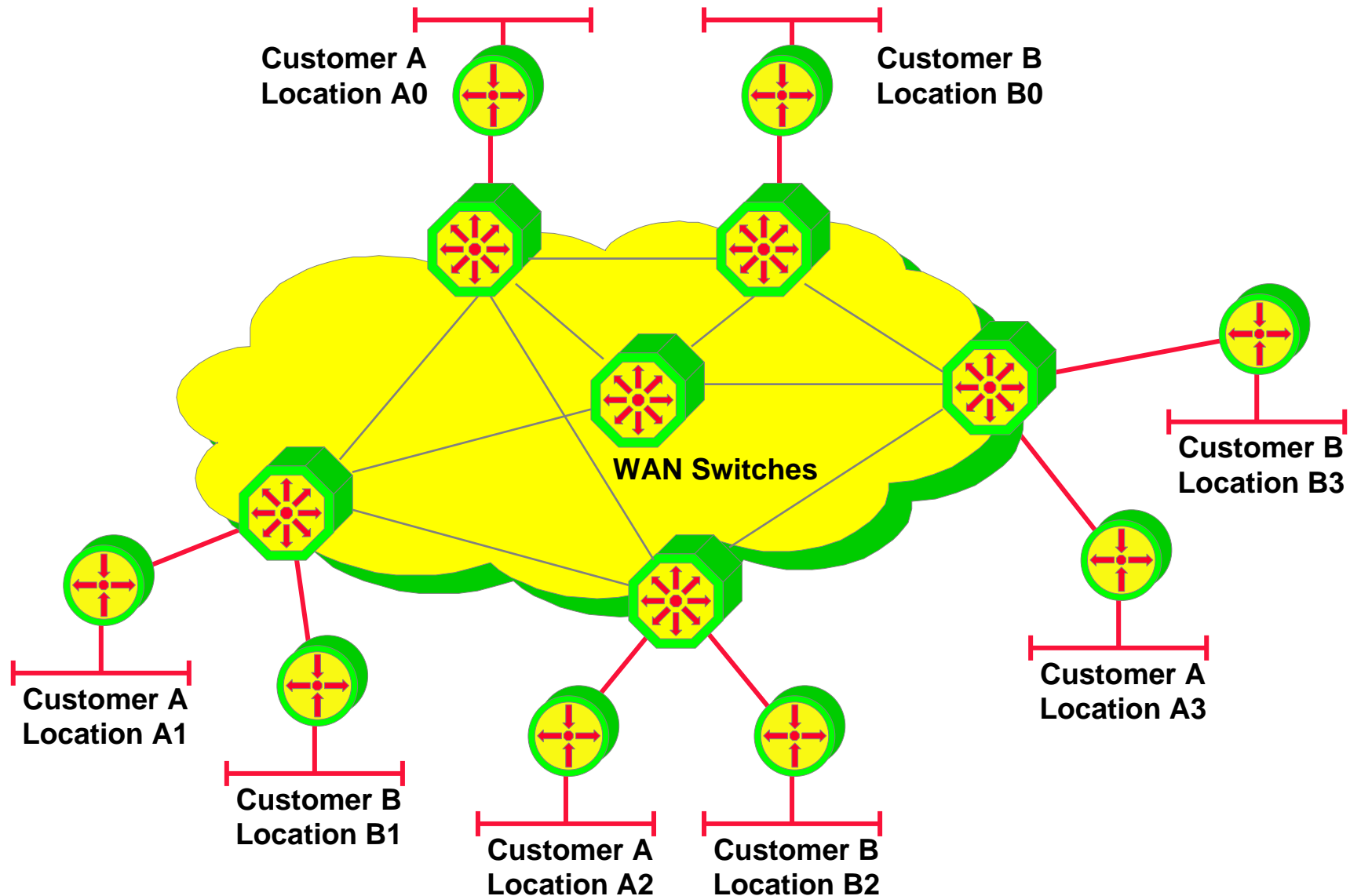
- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing

# Classical VPNs

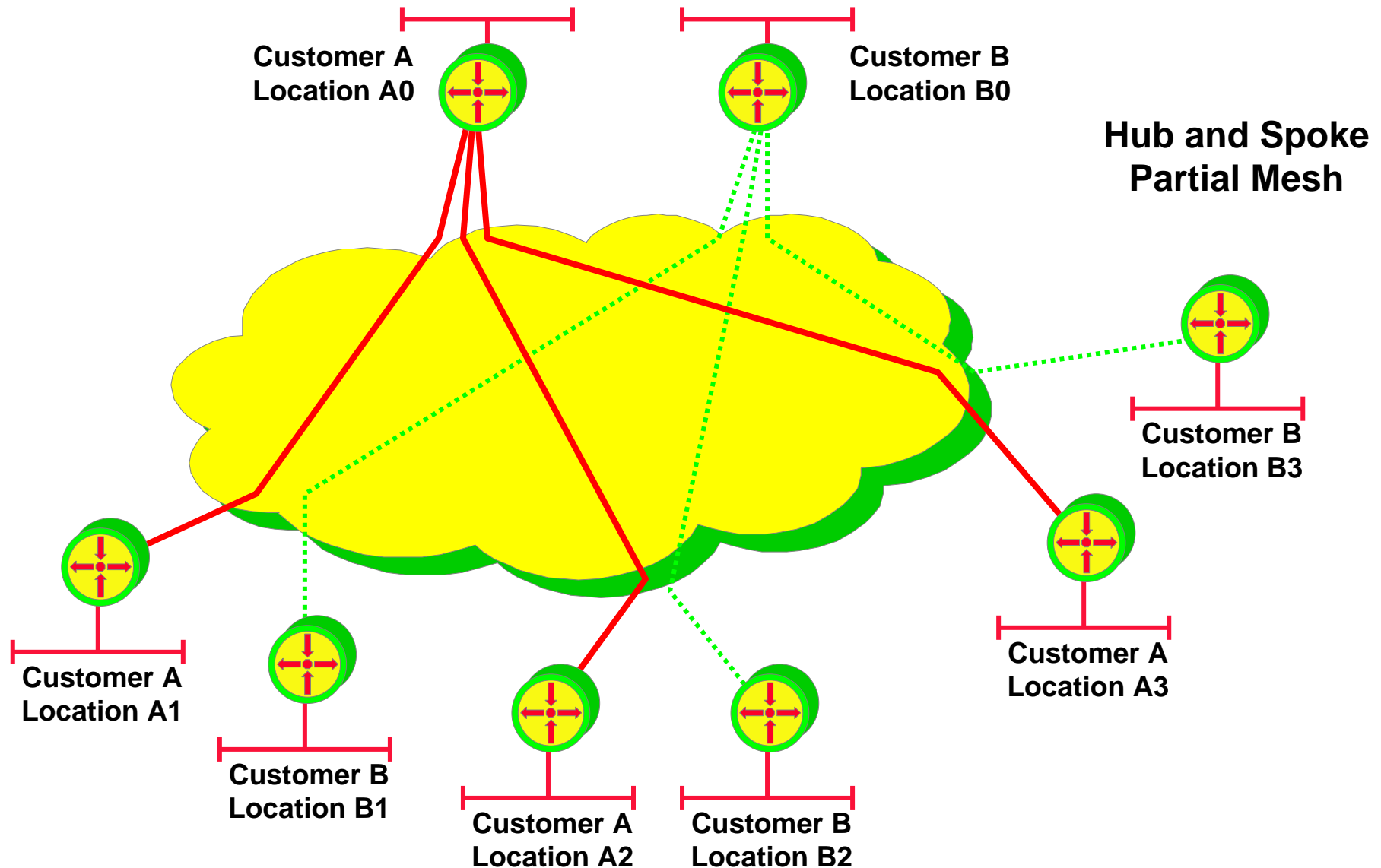
- X.25, Frame Relay or ATM in the core
- dedicated physical switch ports for every customers CPE
  - router, bridge, computer
- customer traffic separation in the core done by concept of virtual circuit
  - PVC service
    - management overhead
  - SVC service with closed user group feature
    - signaling overhead
- separation of customers inherent to virtual circuit technique
- privacy is aspect of customer
  - in most cases overlooked

## VPNs based on Overlay Model

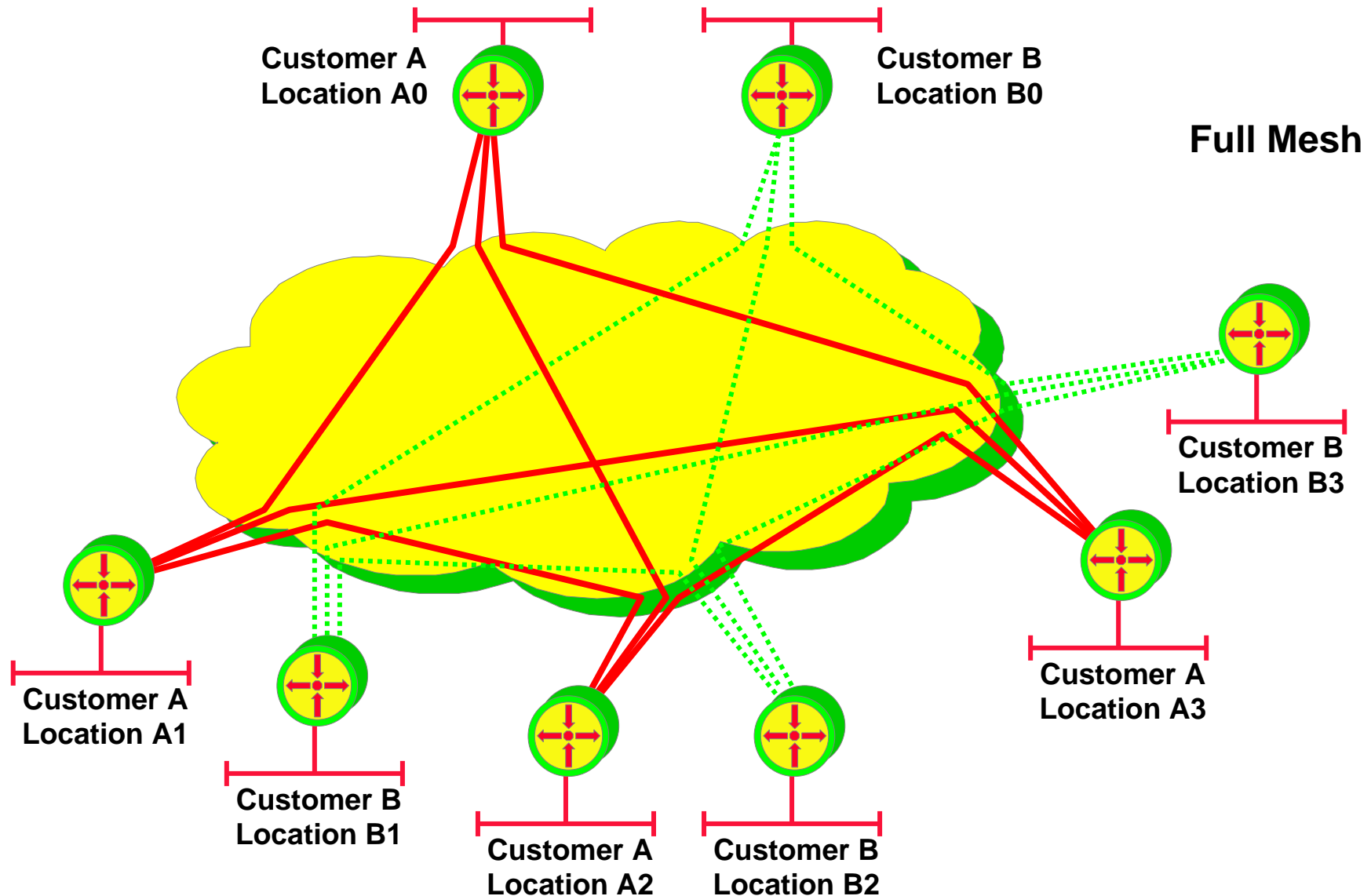
# Physical Topology of Classical VPN



# Logical Topology Classic VPN (1)



# Logical Topology Classic VPN (2)



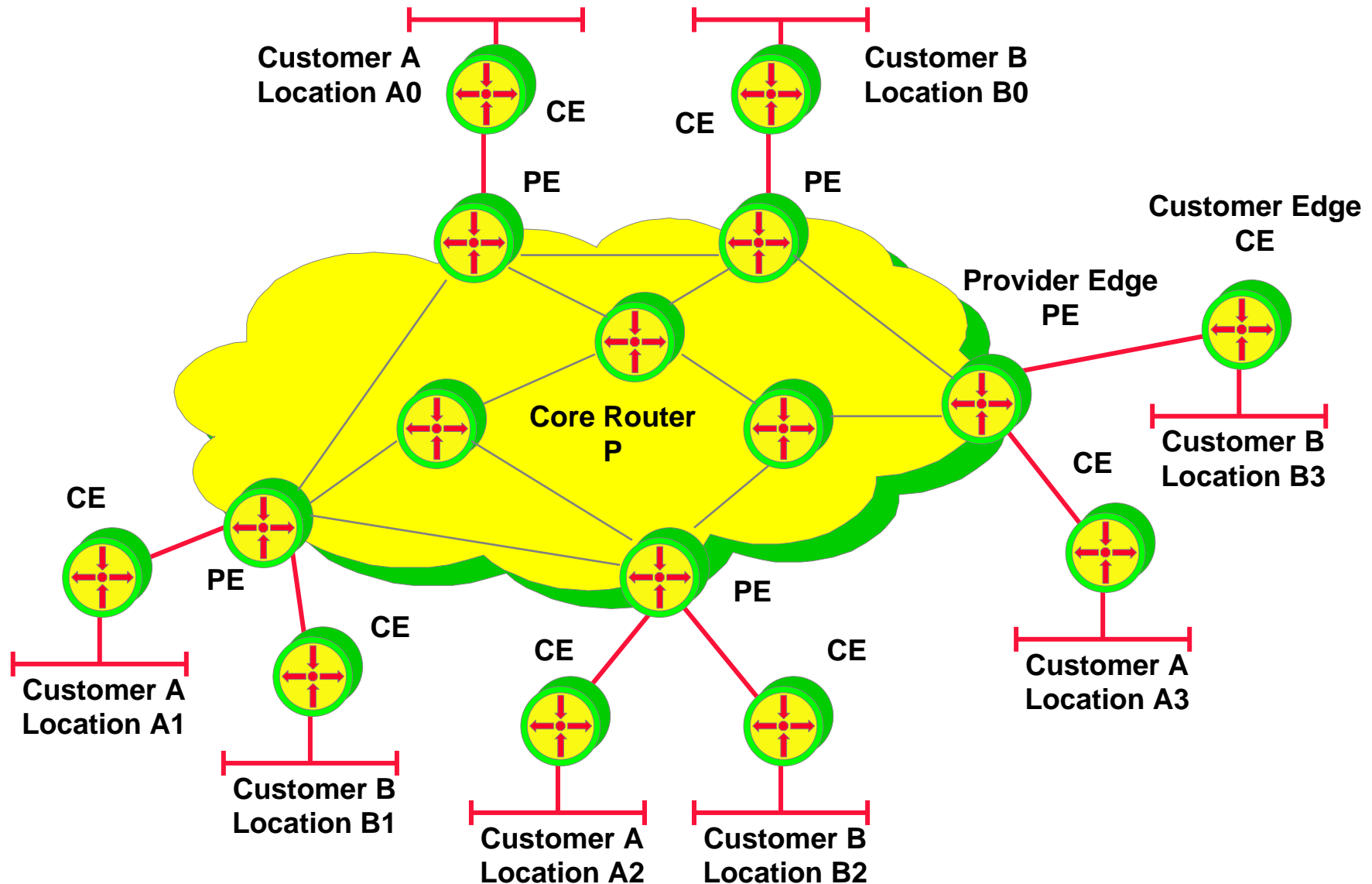
# Virtual Private Networks based on IP

- single technology end-to-end
  - IP forwarding and IP routing
- no WAN switches in the core
  - based on different technology (X.25, FR or ATM)
  - administered by different management techniques
- but accounting and quality of service just coming in the IP world
  - X.25, FR and ATM have it already
- often private means cases control over separation but not privacy
  - data are seen in clear-text in the core
  - encryption techniques can solve this problem
  - but encryption means must be in the hand of the customer

## VPNs based on Peer Model



# Physical Topology IP VPN



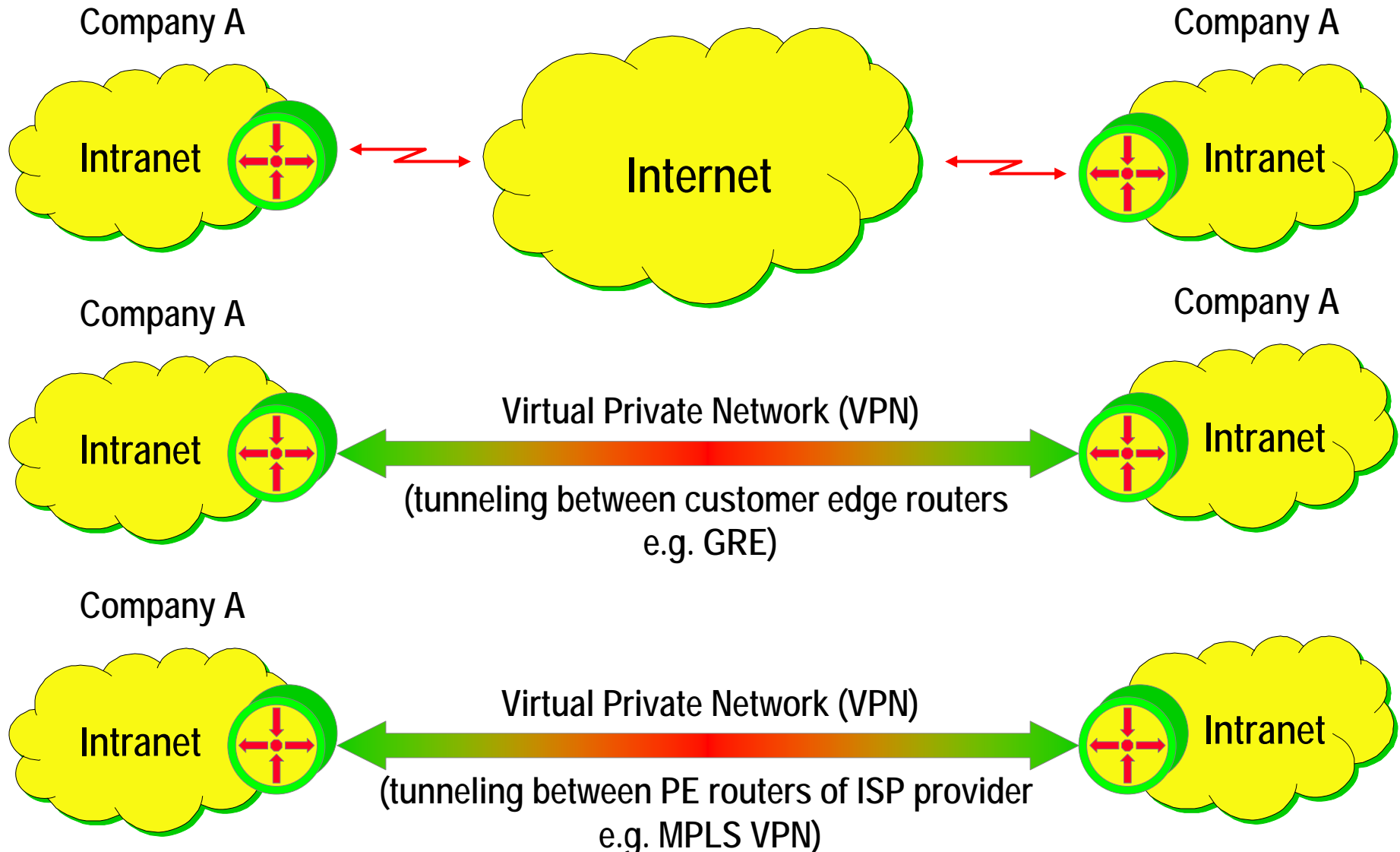
# Possible Solutions for IP VPNs

- **IP addresses of customers non overlapping**
  - filtering and policy routing techniques can be used in order to guarantee separation of IP traffic
    - exact technique depends on who manages routes at the customer site
- **IP addresses of customers overlapping**
  - tunneling techniques must be used in order to guarantee separation of IP traffic
    - GRE
    - L2F, PPTP, L2TP
    - MPLS-VPN
- **If privacy is a topic**
  - encryption techniques must be used
    - SSL/TLS, IPsec

# Tunneling Solutions for IP VPNs

- **Tunneling techniques are used in order to guarantee separation of IP traffic**
  - IP in IP Tunneling or GRE (Generic Routing Encapsulations)
    - Bad performance on PE router
  - PPTP or L2TP for LAN to LAN interconnection
    - Originally designed for PPP Dial-up connections
    - LAN – LAN is just a special case
  - MPLS-VPN
    - Best performance on PE router
- **In all these cases**
  - Privacy still an aspect of the customer

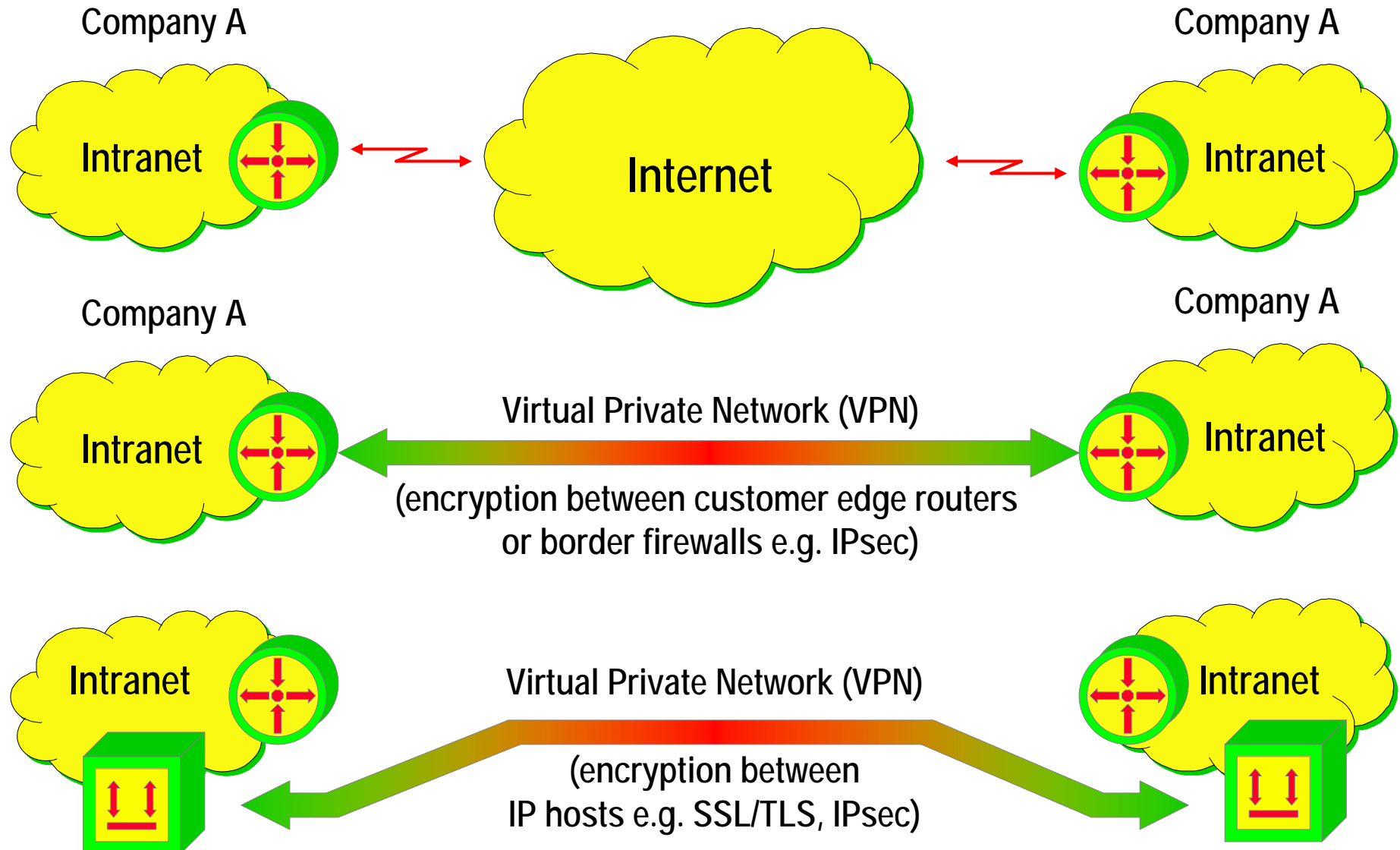
# Tunneling IP VPNs without Encryption



# Encryption Solutions for IP VPNs

- **If privacy is a topic tunneling techniques with encryption are used in order to hide IP traffic**
  - SSL (secure socket layer)
    - Usually end-to-end
    - Between TCP and Application Layer
  - IPsec
    - Could be end-to-end
    - Could be between special network components (e.g. firewalls, VPN concentrators) only
    - Between IP and TCP/UDP Layer
  - PPTP and L2TP Tunnels
    - With encryption turned on via PPP option

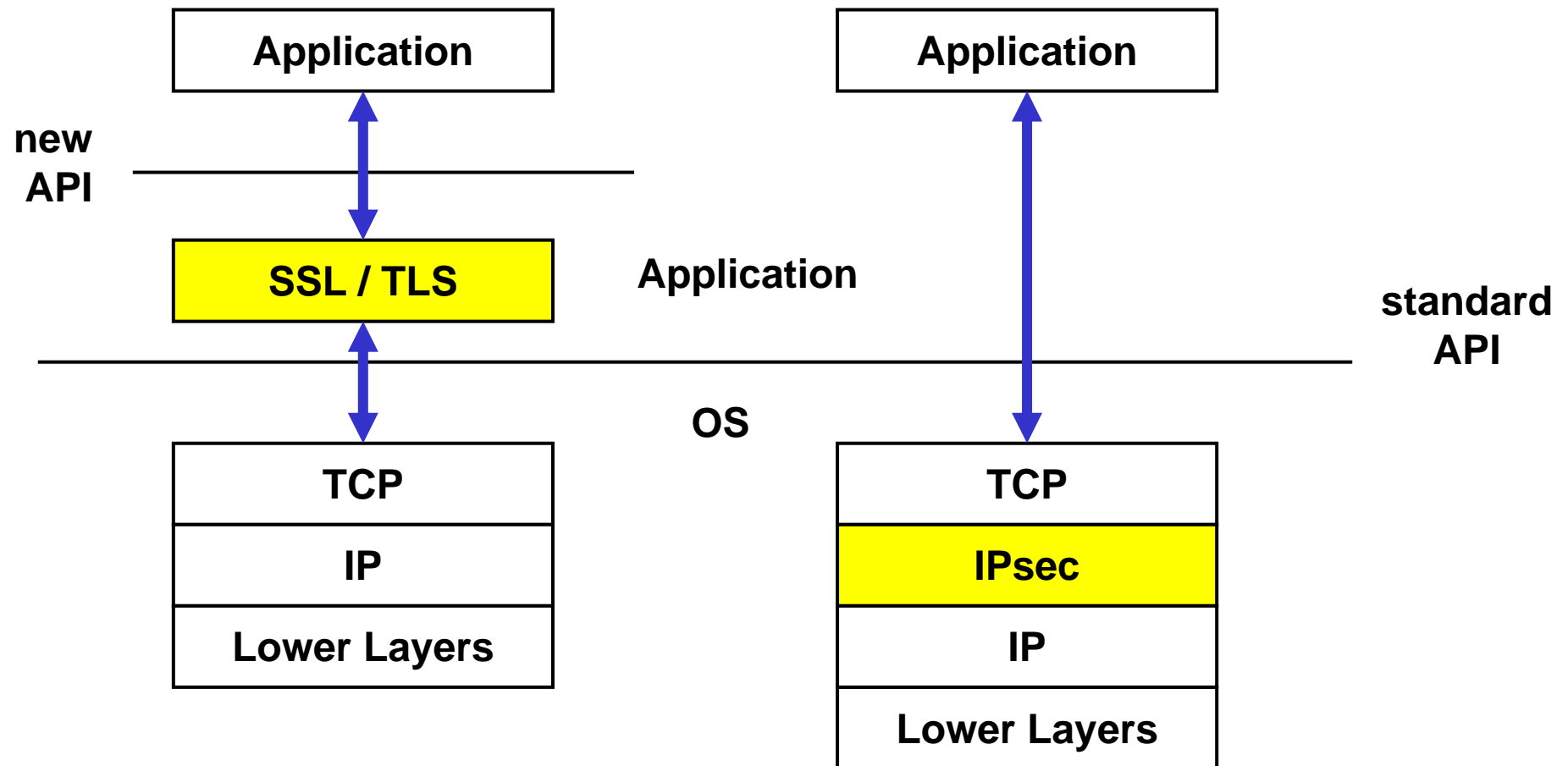
# Tunneling IP VPNs without Encryption



# SSL/TLS versus IPsec

Application must be aware of new application programming interface

Application can use standard application programming interface



# Two Major VPN Paradigms

- **Overlay VPNs: Transparent P2P links**
  - Well-known technology
  - Provider does not care about customer routing
  - Best customer isolation
  
- **Peer VPNs: Participation in Provider-routing**
  - Optimum routing
  - Simple provision of additional VPN
  - Problems with address space



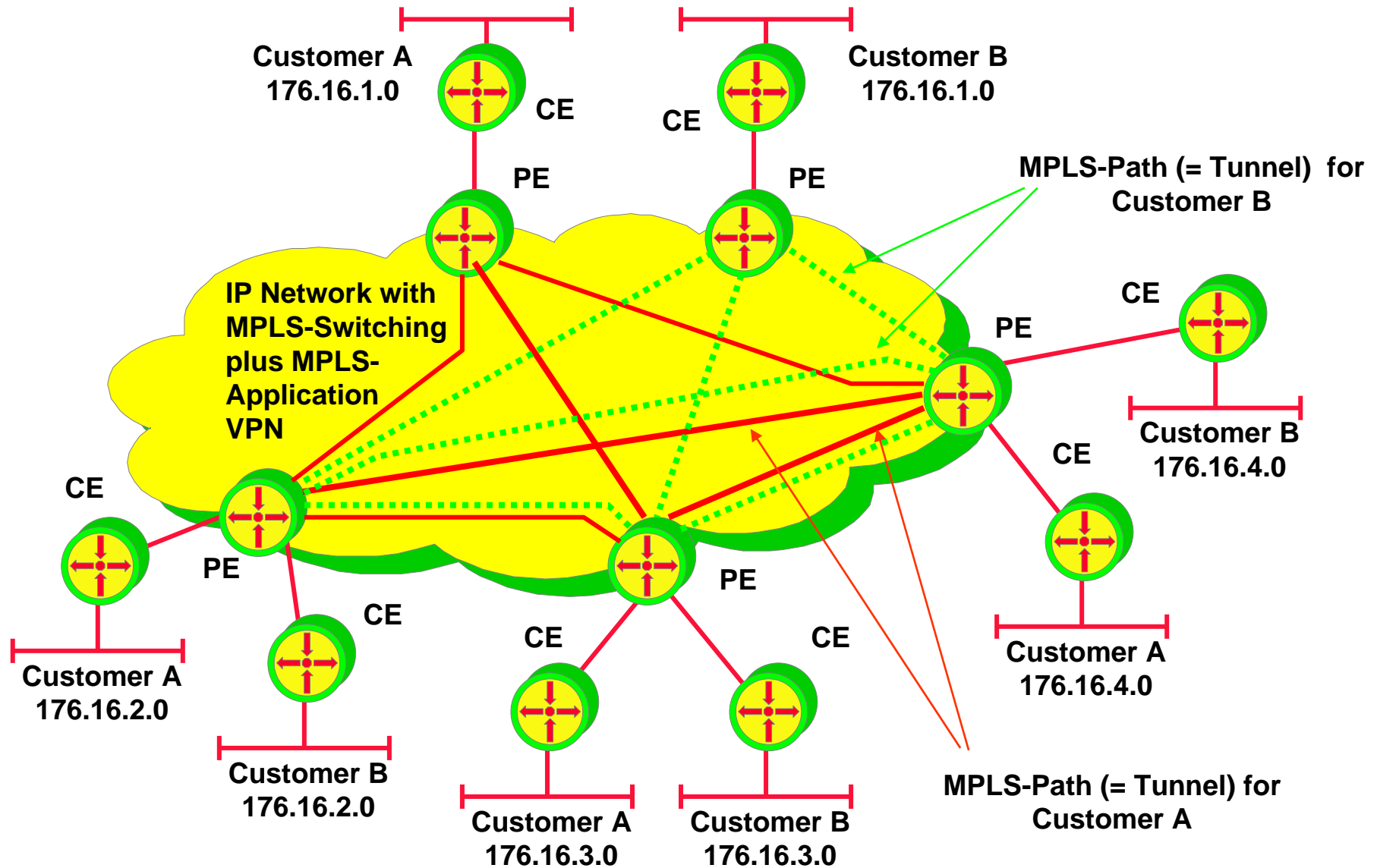
# MPLS VPN – Best of Both Worlds

- **Combines VPN Overlay model with VPN Peer model**
- **PE routers allow route isolation**
  - By using Virtual Routing and Forwarding Tables (VRF) for differentiating routes from the customers
  - Allows overlapping address spaces
- **PE routers participate in P-routing**
  - Hence optimum routing between sites
  - Label Switched Paths are used within the core network
  - Easy provisioning (sites only)
- **Overlapping VPNs possible**
  - By a simple (?) attribute syntax

# MPLS VPN – Principles

- **Requires MPLS Transport within the core**
  - Using the label stack feature of MPLS
- **Requires MP-BGP among PE routers**
  - Supports IPv4/v6, VPN-IPv4, multicast
  - Default behavior: BGP-4
- **Requires VPN-IPv4 96 bit addresses**
  - 64 bit Route Distinguisher (RD)
  - 32 bit IP address
- **Every PE router uses one VRF for each VPN**
  - Virtual Routing and Forwarding Table (VRF)

# MPLS-VPN

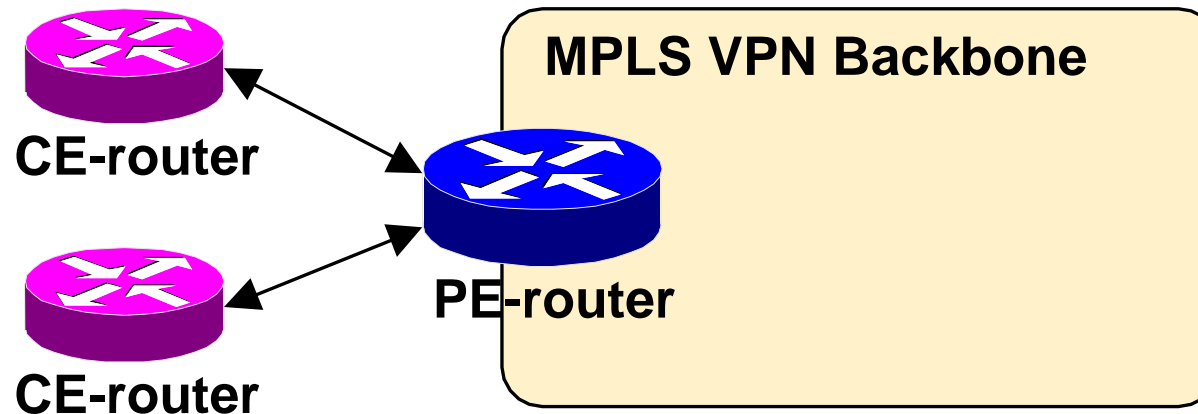


# Agenda

---

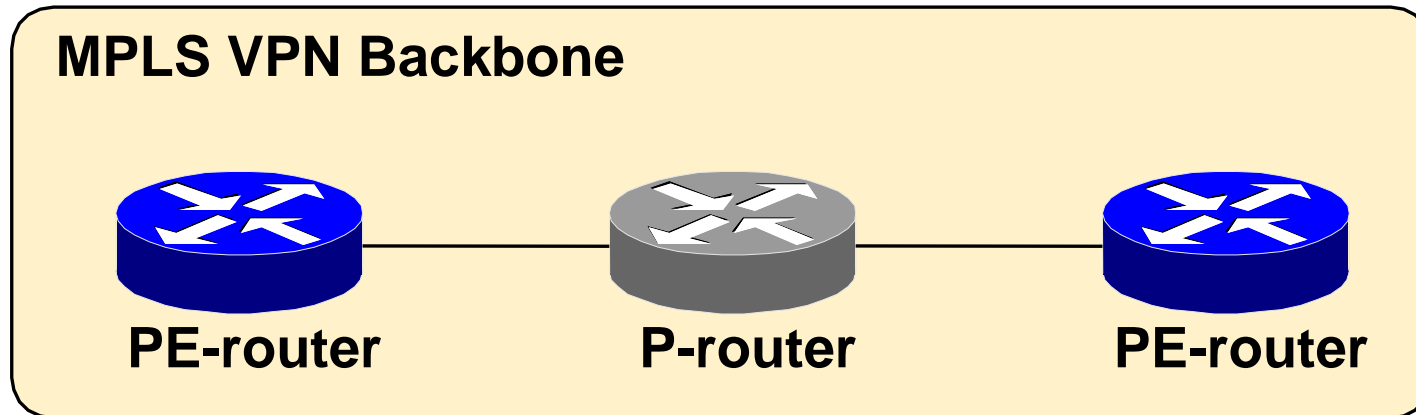
- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing

# CE-Router Perspective



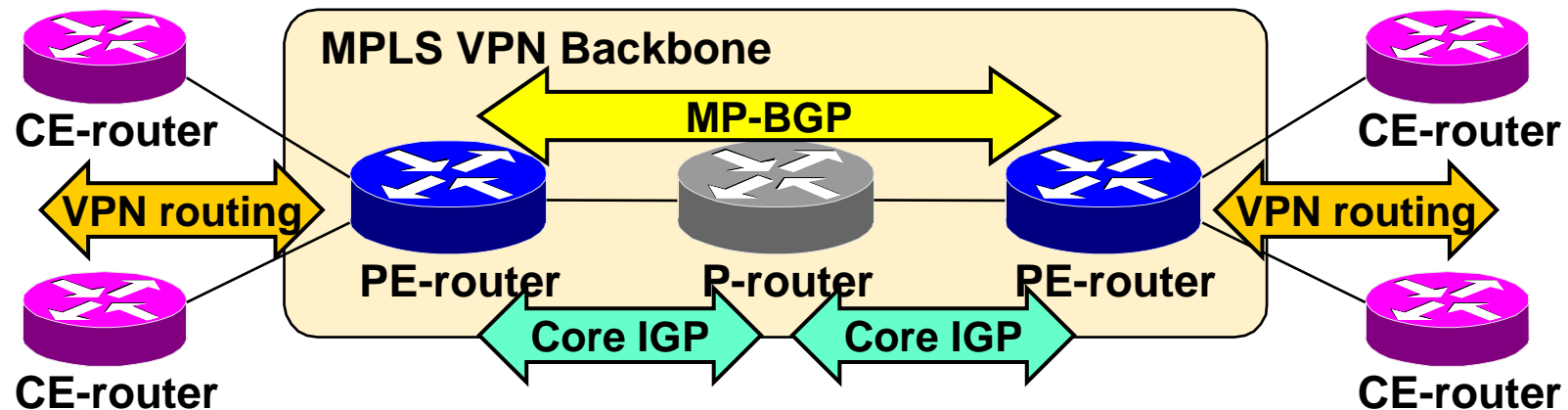
- **CE (Customer Edge) - routers run standard IP routing software and exchange routing updates with the PE-router**
  - EBGP, OSPF, RIPv2 or static routes are supported
- **PE (Provider Edge) - router appears as just another router in the customer's network**

# P-Router Perspective



- **P (Provider) - routers do not participate in MPLS VPN routing and do not carry VPN (customer) routes**
- **P - routers run backbone IGP like OSPF or IS-IS with the PE-routers**

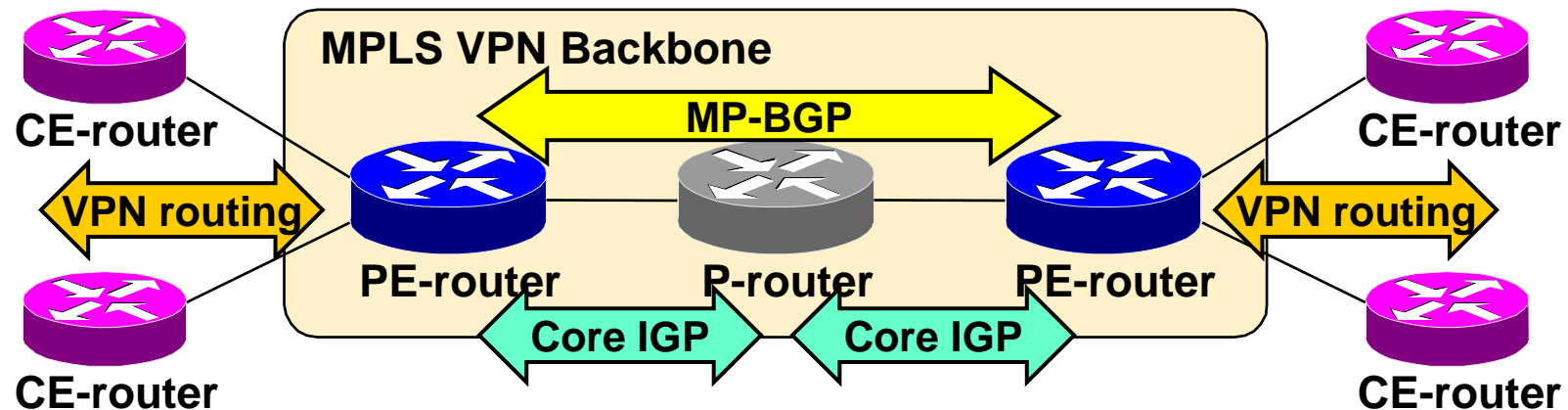
# PE-Router Perspective



## PE-routers contain a number of routing tables:

- [Global routing table](#) that contains core routes (filled with core IGP)
- [Virtual Routing and Forwarding \(VRF\)](#) tables for sets of sites with identical routing requirements
- VRF's are filled with information from CE-routers and MP-BGP information from other PE-routers

# PE-Router Perspective

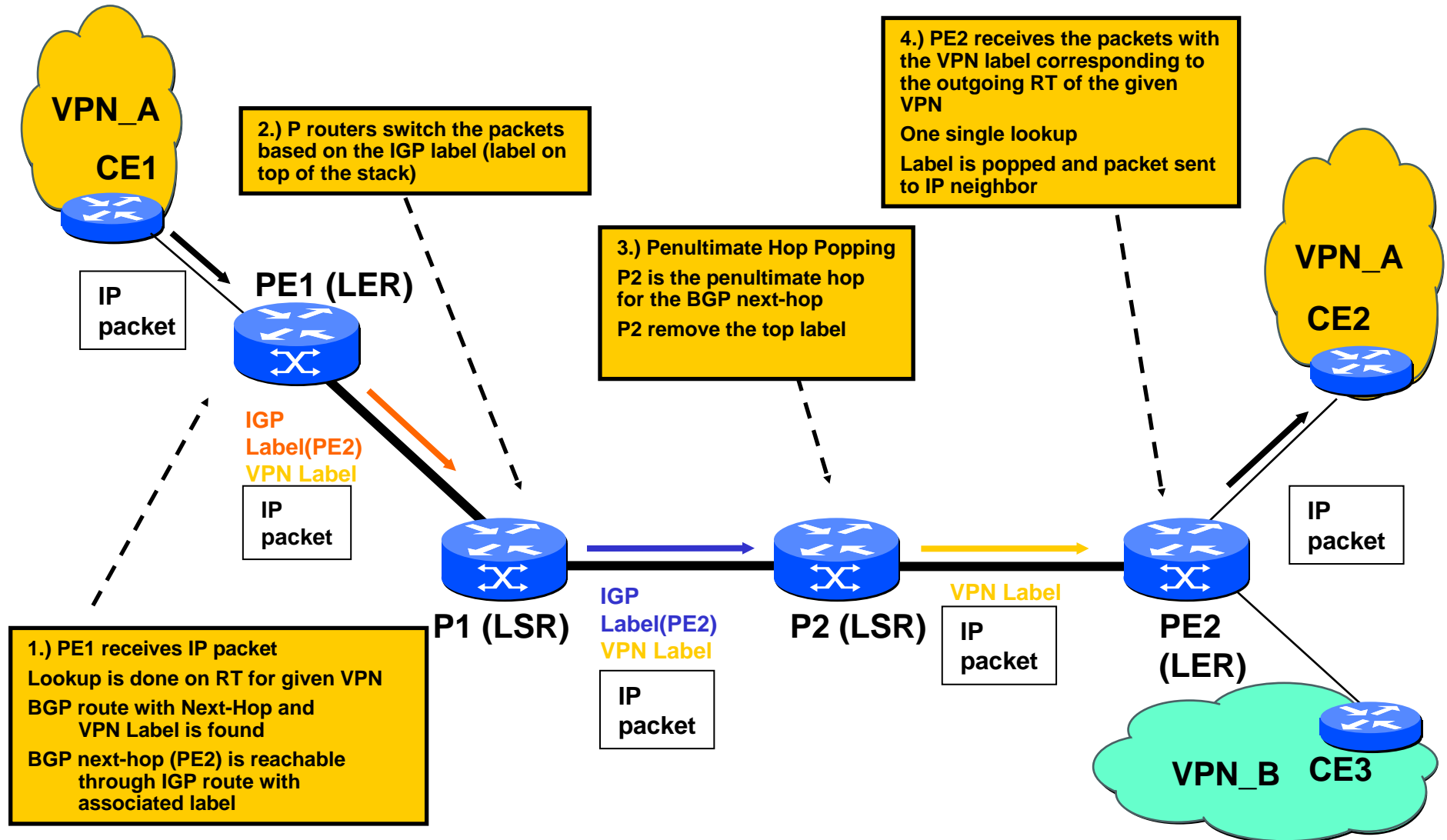


## PE-routers:

- Exchange VPN routes with CE-routers via per-VPN routing protocols
- Exchange core routes with P-routers and PE-routers via core IGP
- Exchange VPN-IPv4 routes with other PE-routers via Internal MP-BGP sessions



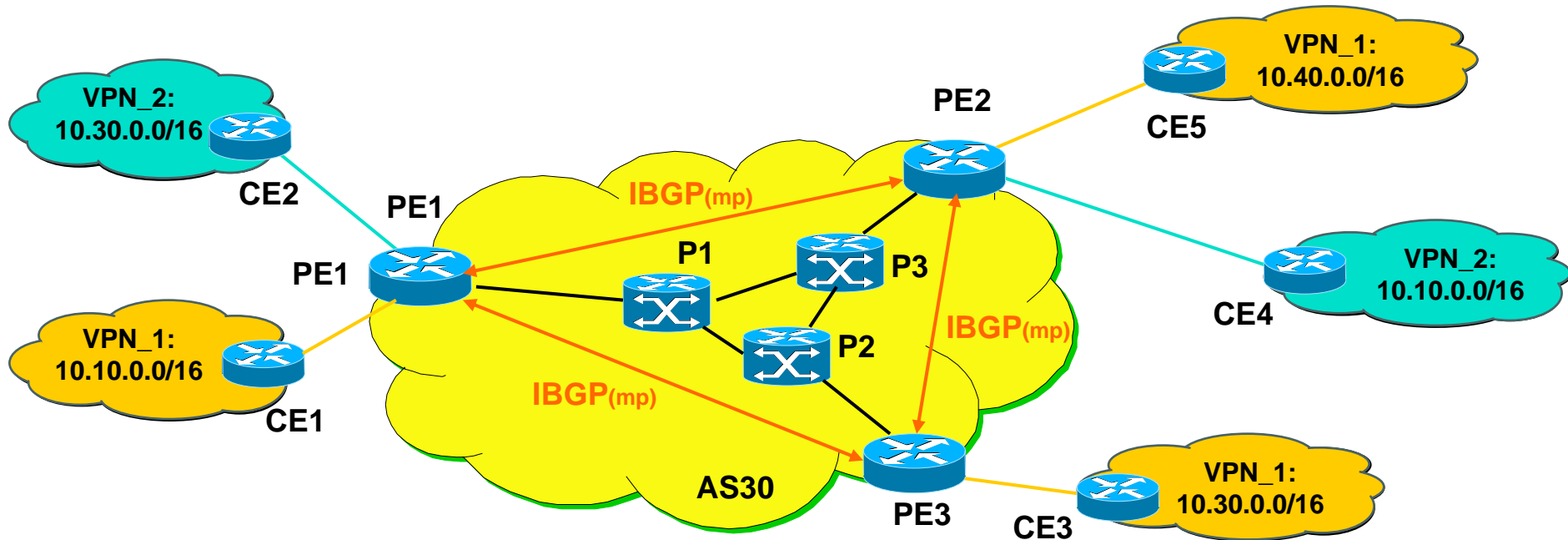
# MPLS VPN using MPLS Label Stack



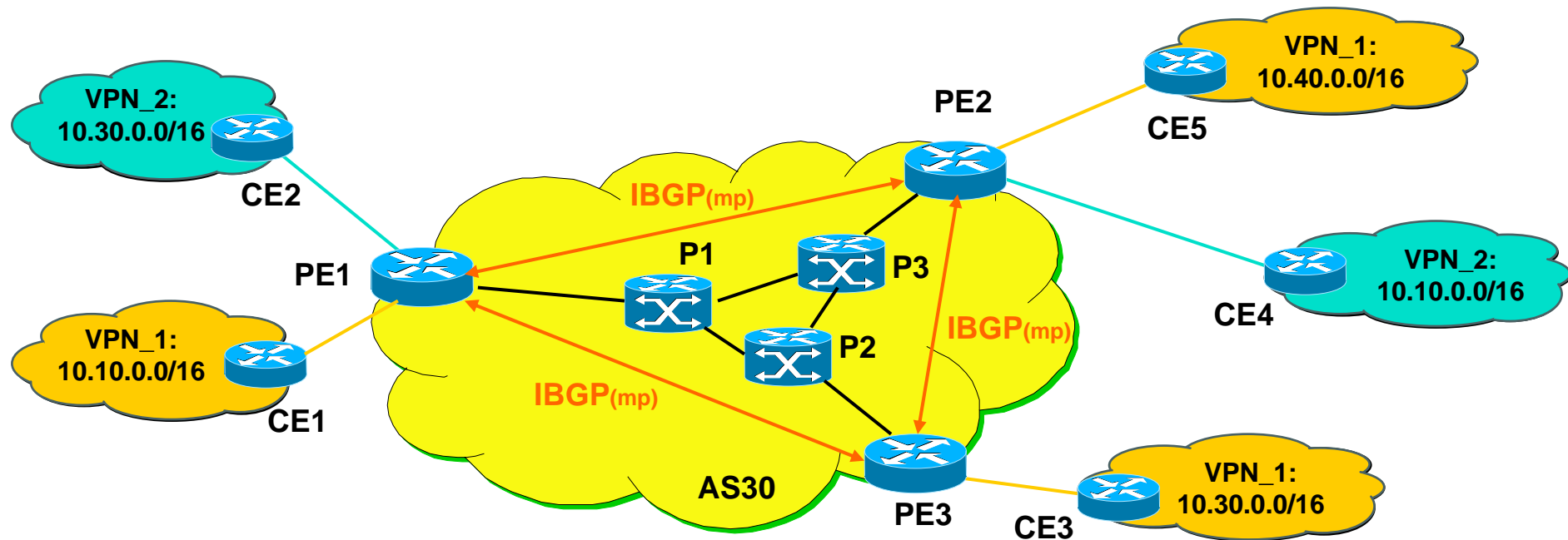
# Agenda

---

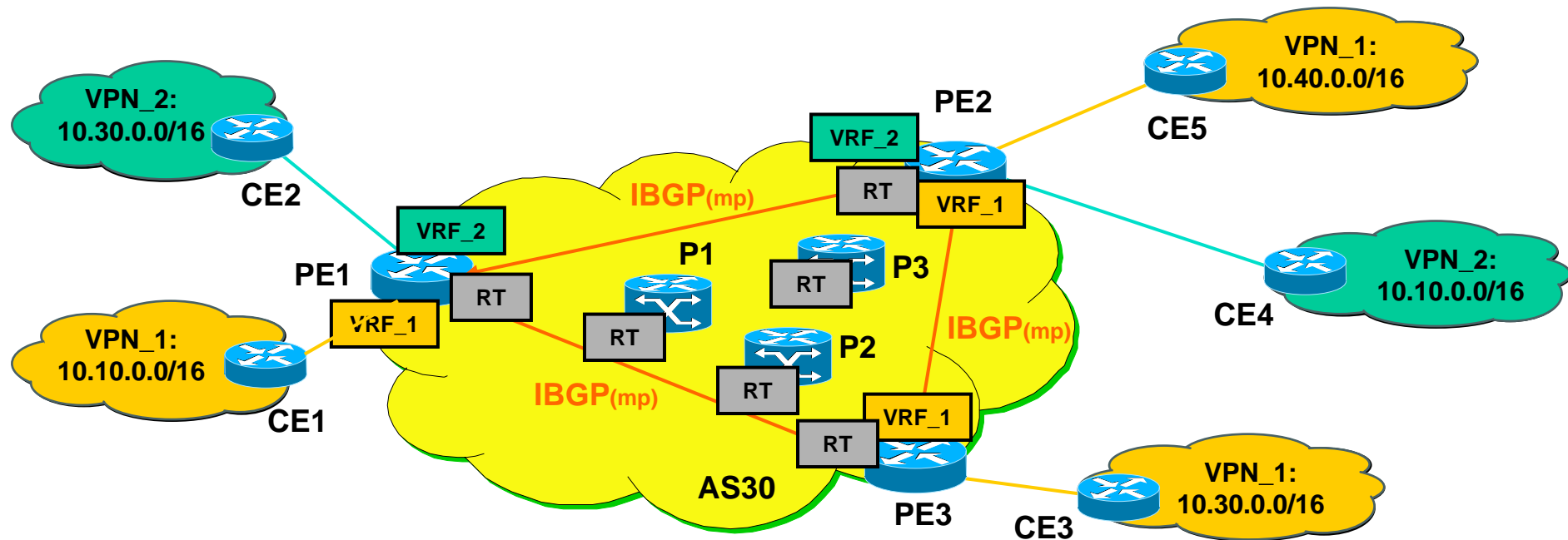
- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing



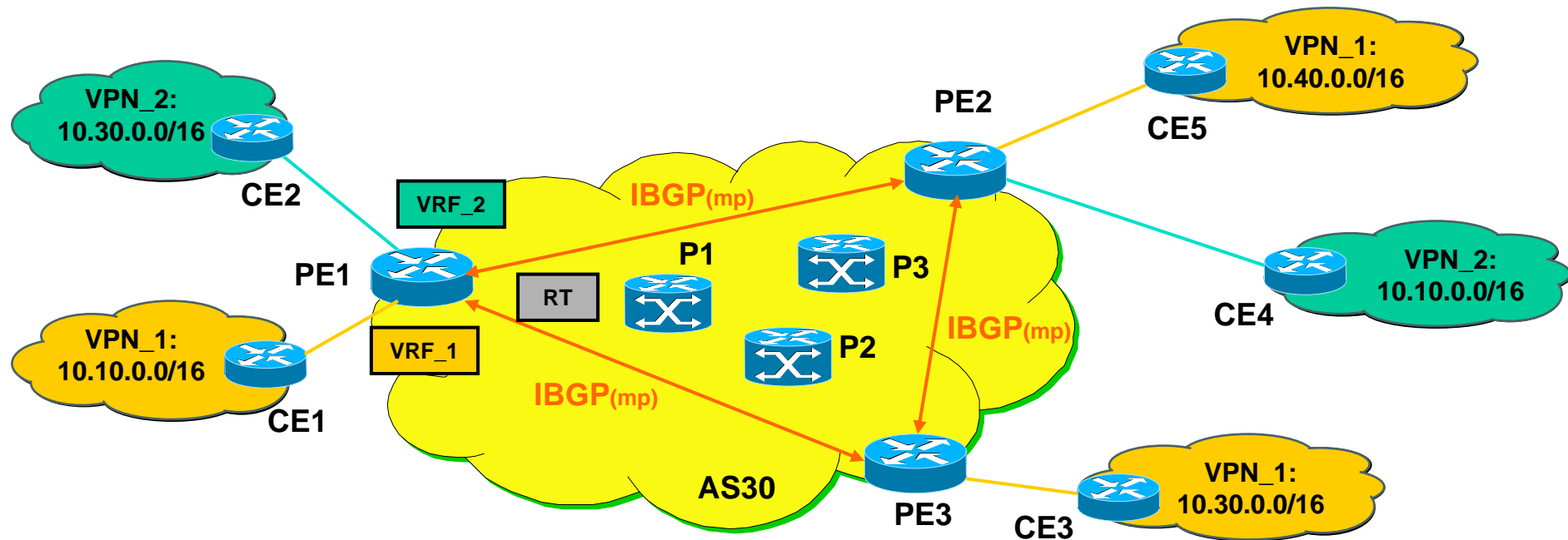
- Customer Networks are connected to MPLS VPN service provider
- Basic scenario:
  - VPNs are not overlapping -> that means they are totally separated from each other
  - VPN\_1: Orange Customer
  - VPN\_2: Green Customer



- Provider routers P (LSRs) are in the core of the MPLS cloud
- Provider Edge routers PE (LER) use MPLS within the core and plain IP with CE routers
- PE routers are fully meshed concerning Internal MP-BGP Sessions
- P and PE routers share a common IGP (e.g. OSPF or IS-IS)
- Customer Edge CE routers connect customer sites to provider



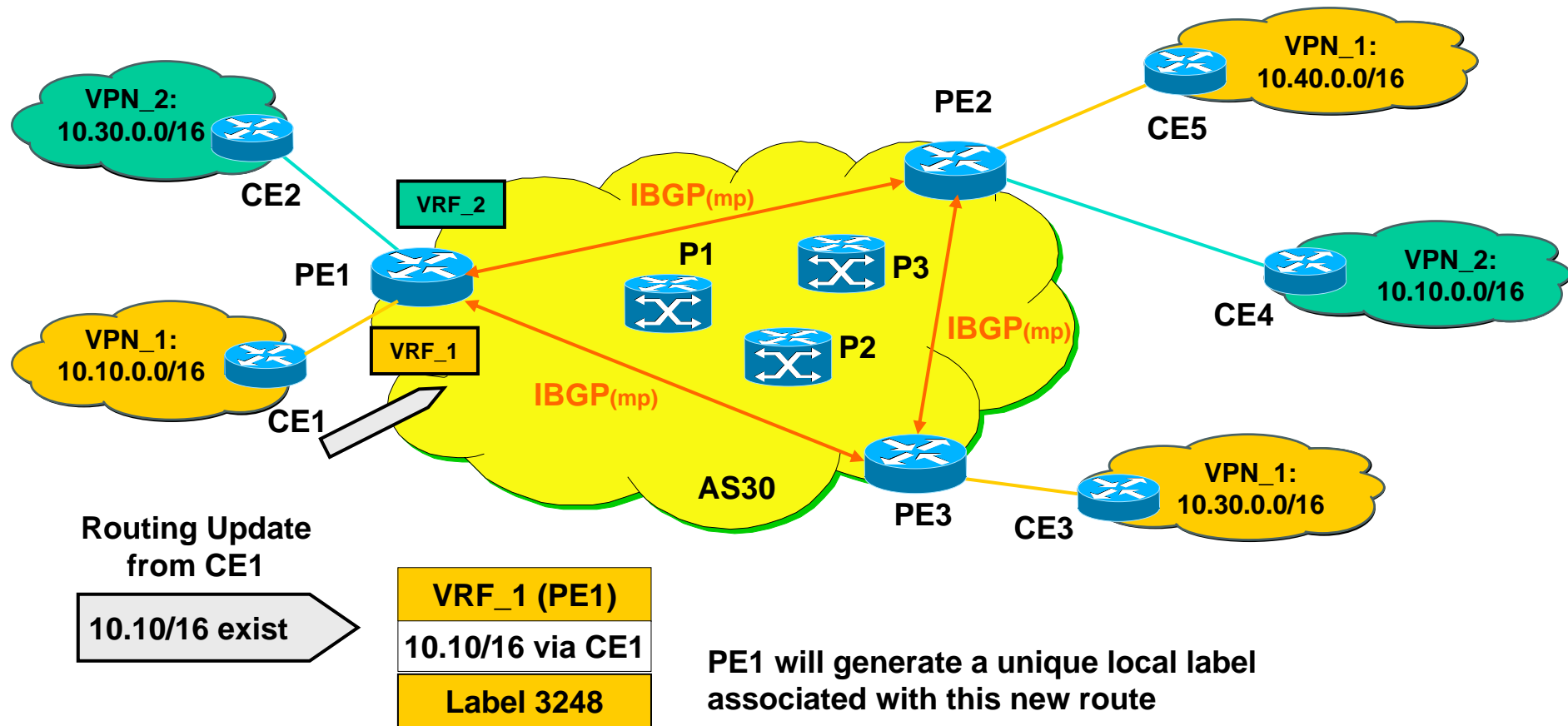
- PE router
  - maintains a separate routing table VRF per customer site
    - VRF (VPN Routing and Forwarding) Table
  - holds global routing table RT for communication within MPLS cloud
    - maintained by IGP
  - forwarding within MPLS cloud is based on labels
    - distributed by LDP



- VRF table
  - contains Net-IDs received from corresponding CE site
    - via RIPv2, OSPF, External BGP session or static routes
  - contains NET-IDs received from other PE routers
    - via Internal MP-BGP Sessions received as VPN-IPv4 addresses
  - hence overlapping addresses are no problem

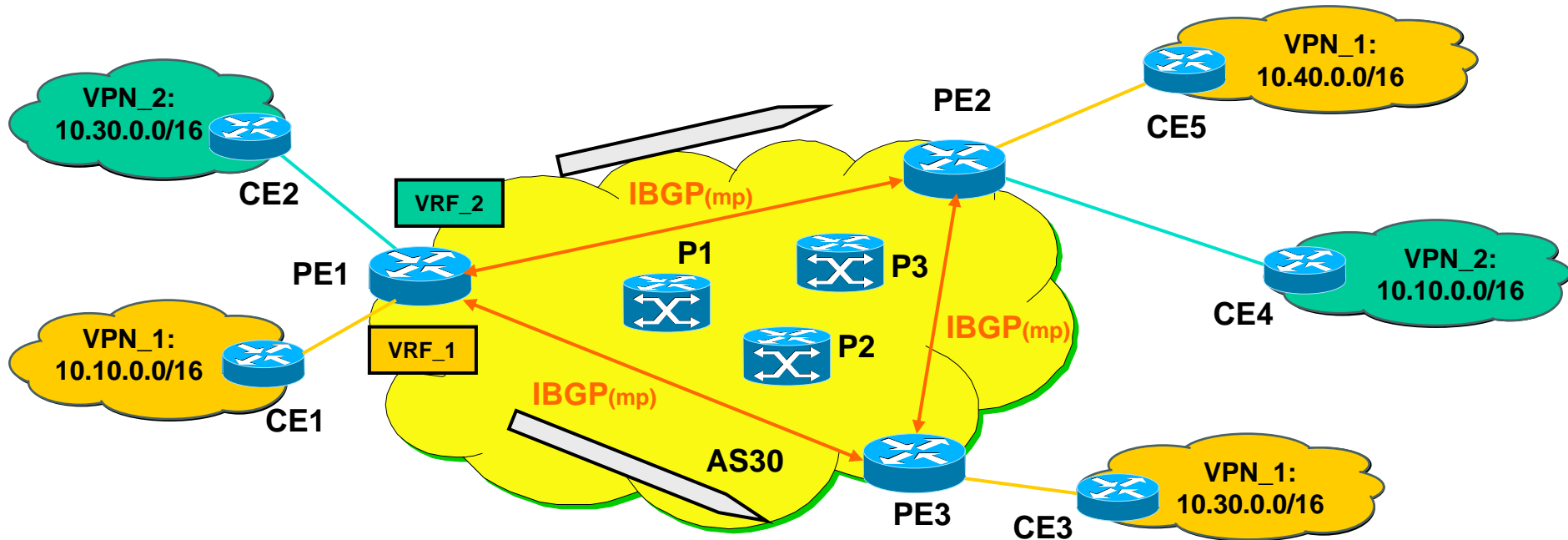
# New Network 10.10.0.0/16 at CE1

1



- Routing Update will install a new route in the corresponding VRF table of PE1 and hence the new route must be advertised to all other PEs via Internal MP-BGP
  - as VPN-IPv4 address

# Advertise Network 10.10.0.0/16 to PE's 2



MP-BGP uses:

- MP\_Reach\_NLRI attribute
- Next-Hop
- VPN-IPv4\_NLRI
- RD=Route Distinguisher
- Net
- Label
- Extended Community attr.
- RT = Route Target

Routing Update from PE1 via  
Internal MP-BGP to all other PE's

VPN-IPv4 update:

RD (ID to uniquely distinguished Net from other nets) = 30:1

Net = 10.10/16, Next-Hop = PE1

Label that should be used to reach this Net = 3248

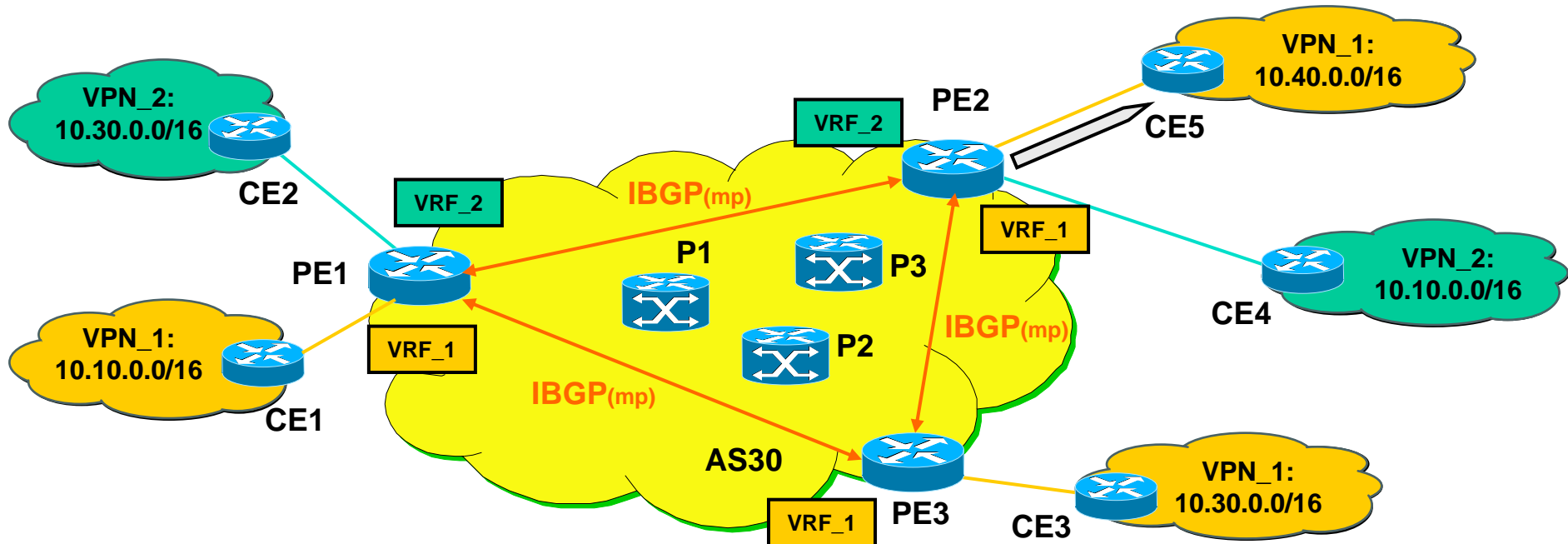
RT (Hint to which VRF's this Net should be imported) = Orange

AS30 VPN #1



# New Network 10.10.0.0/16 at PE2/CE5

3



Routing Update from PE1  
received at PE 2

VPN-IPv4 update:  
RD = 30:1  
Net = 10.10/16, Next-Hop = PE1  
Label = 3248  
RT = Orange

New Route put into  
VRF\_1 based on RT=Orange

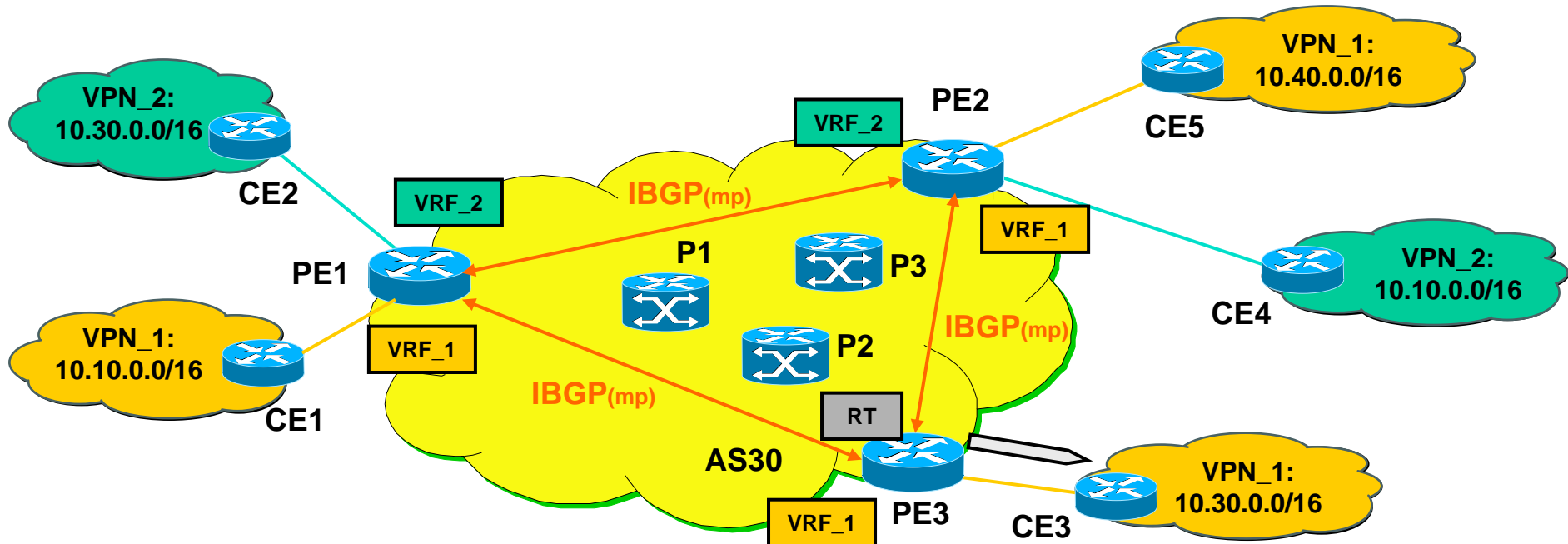
VRF_1 (PE2)
10.10/16 via PE1 use 3248
VRF_2 (PE2)

Routing Update  
to CE5

10.10/16 exist

# New Network 10.10.0.0/16 at PE3/CE3

4



Routing Update from PE1  
received at PE 3

VPN-IPv4 update:

RD = 30:1

Net = 10.10/16, Next-Hop = PE1

Label = 3248

RT = Orange

New Route put into  
VRF\_1 based on RT=Orange

VRF\_1 (PE3)

10.10/16 via PE1 use 3248

FIB (RT PE3)

PE1 via P2 use 89

Routing Update  
to CE3

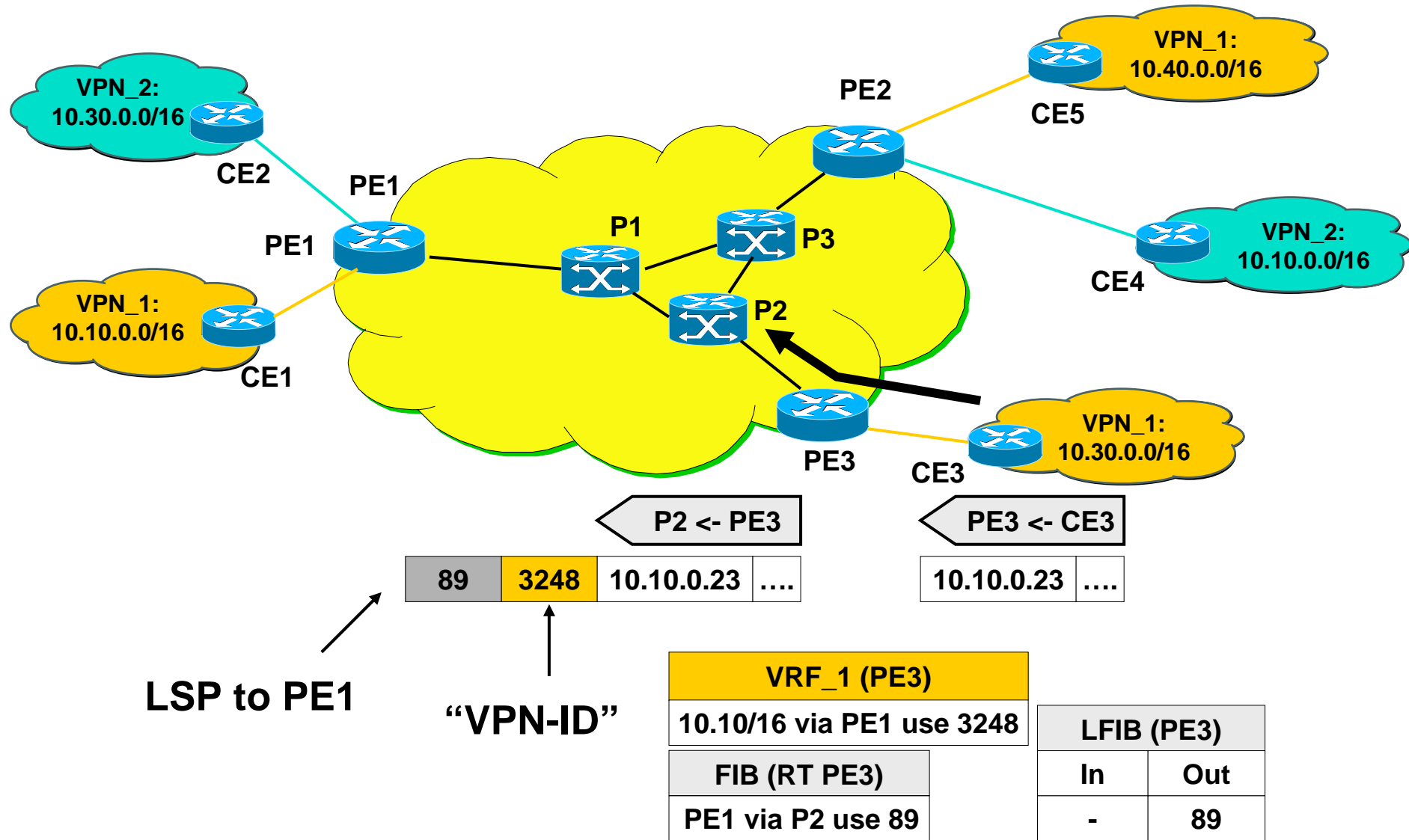
10.10/16 exist

RT (CE3)

10.10/16 via PE3

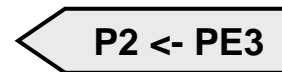
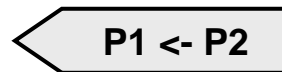
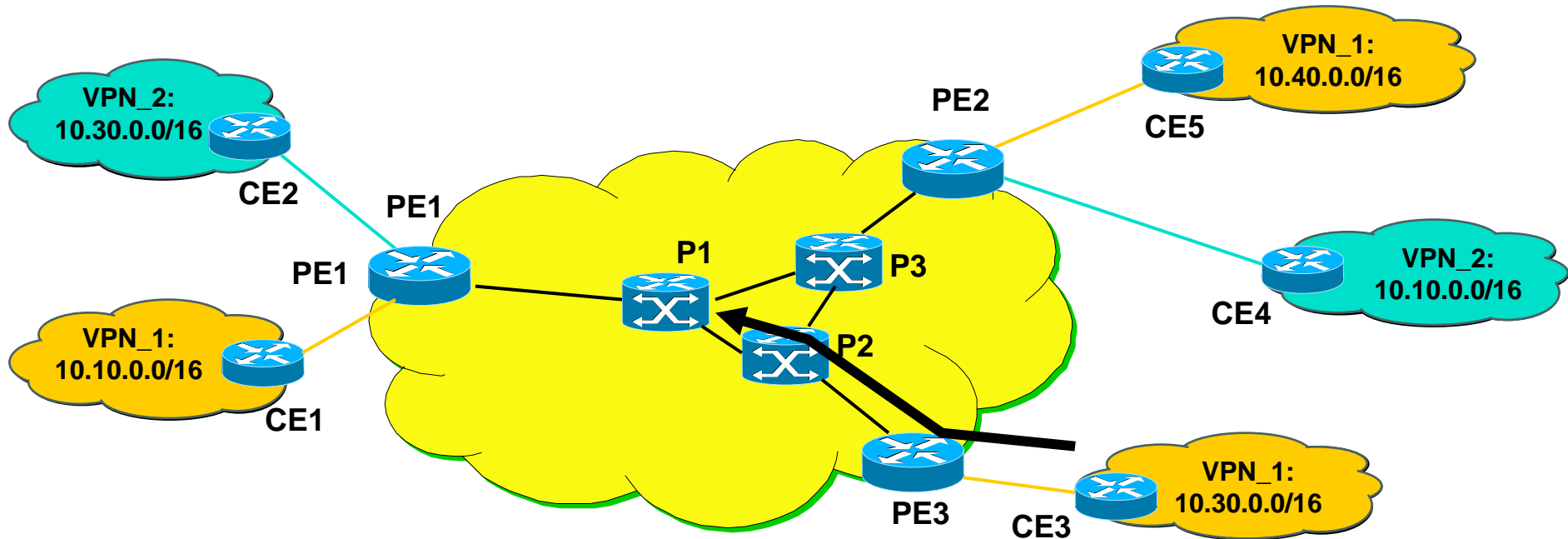
# MPLS\_VPN in Action

1



# MPLS\_VPN in Action

2



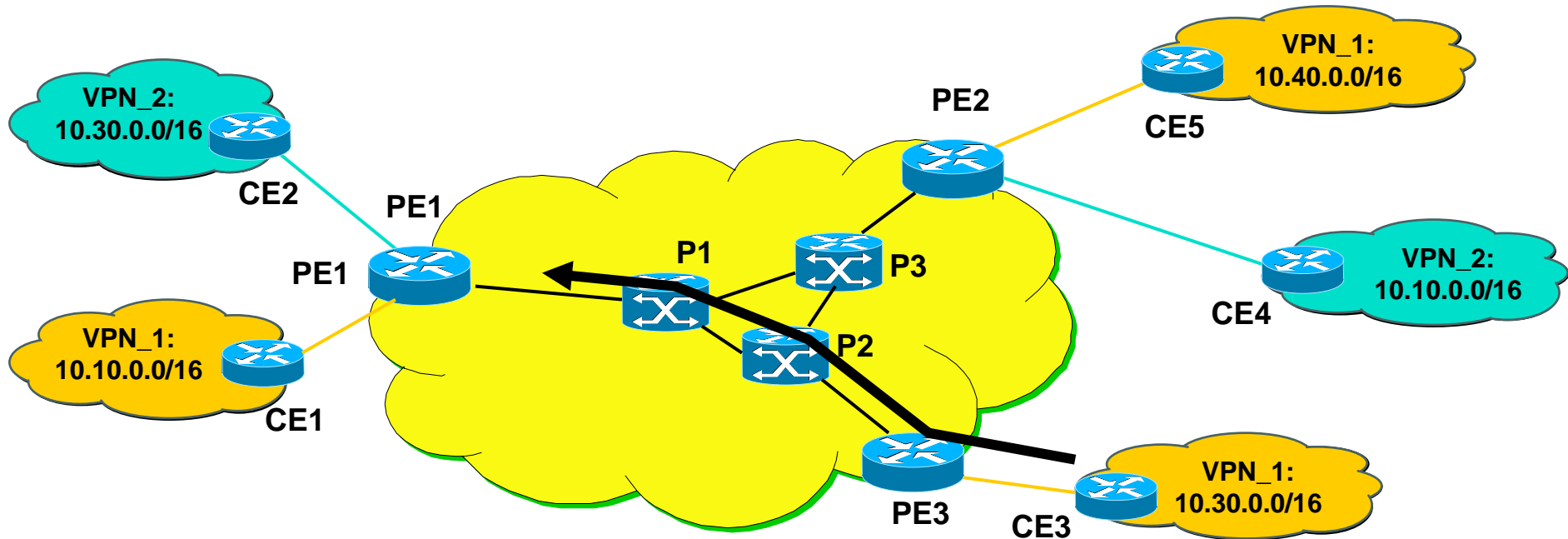
77	3248	10.10.0.23	....
----	------	------------	------

89	3248	10.10.0.23	....
----	------	------------	------

LFIB (P2)		
FIB (RT P2)	In	Out
PE1 via P1 use 77	89	77

# MPLS\_VPN in Action

3



PE1 <- P1

P1 <- P2

3248 10.10.0.23 ....

77 3248 10.10.0.23 ....

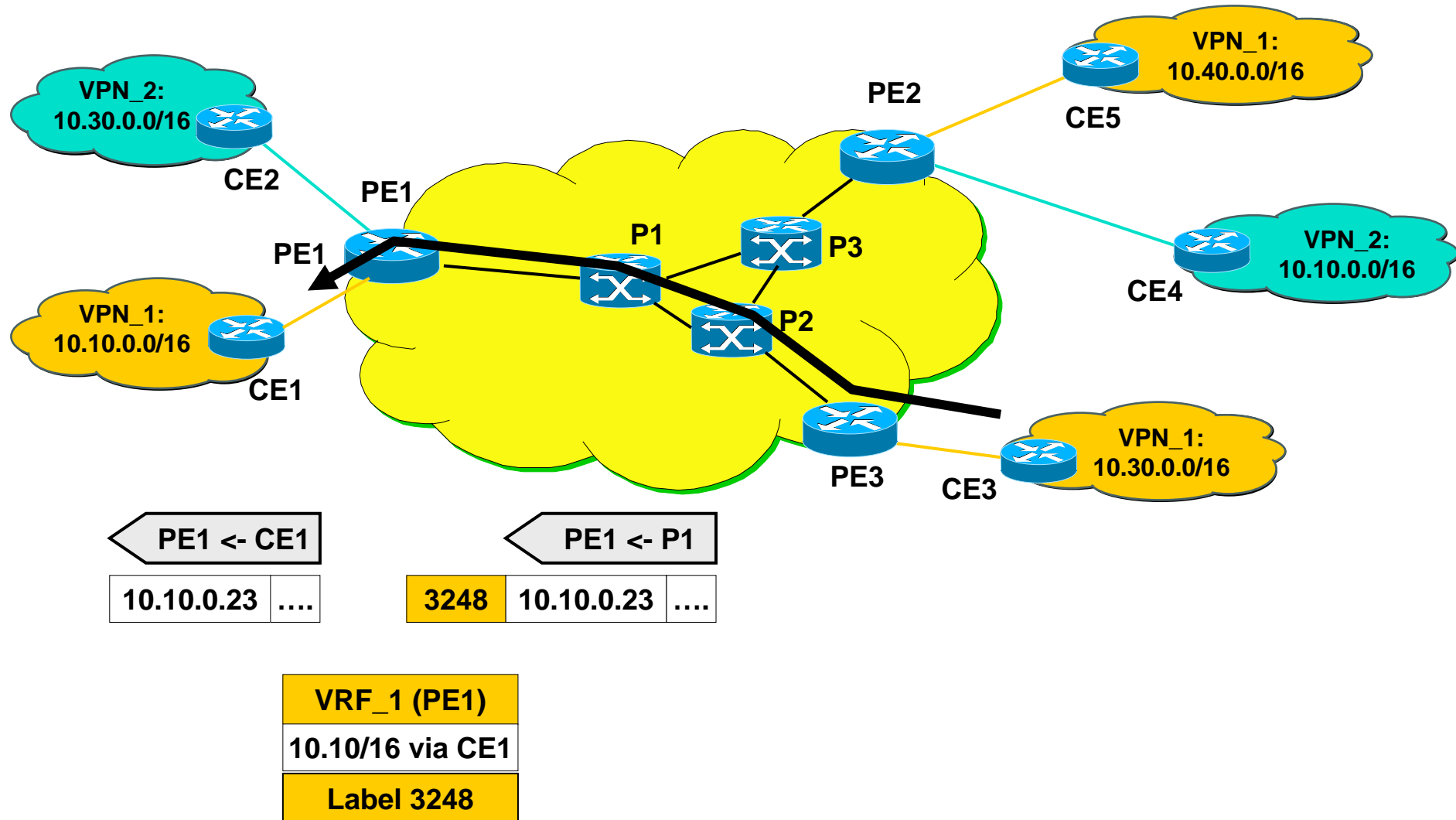
FIB (RT P1)		
PE1 via PE1 use null	77	POP

LFIB (P1)	
In	Out
77	POP

# MPLS\_VPN in Action

4



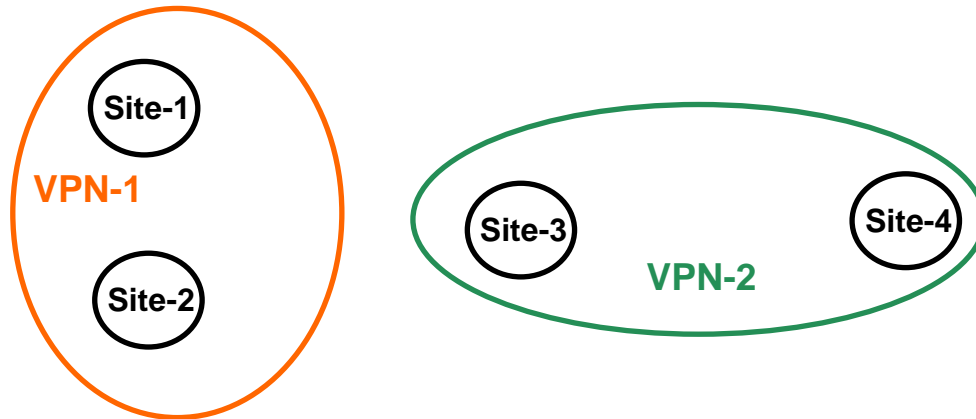
# Agenda

---

- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing

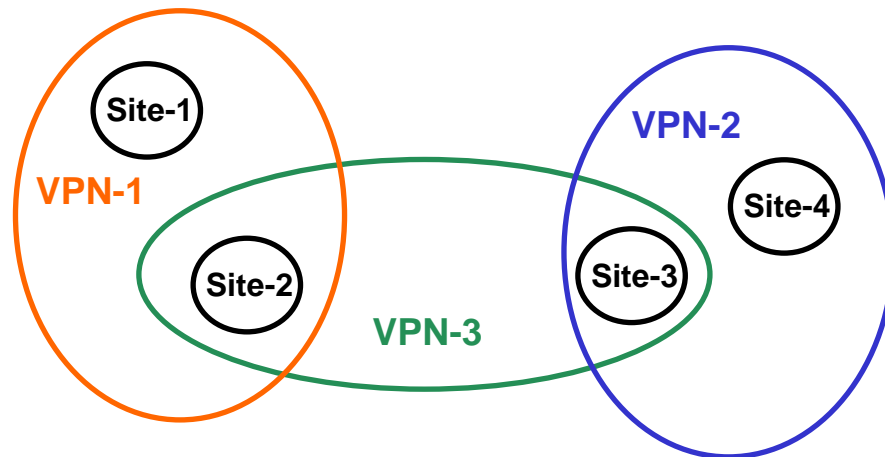
# MPLS VPN Types

## VPNs not overlapping (Intranet)



site-2 networks can reach site-1 networks and vice versa, site-3 networks can reach site-4 networks and vice versa.

## VPNs overlapping (Intranet/Extranet)



site-2 networks can reach site-1 and site-3 networks, site-3 networks can reach site-4 and site-3 networks, site-1 networks can reach site-2 networks only, site-4 networks can reach site-3 networks only.



# A New Sight of VPN

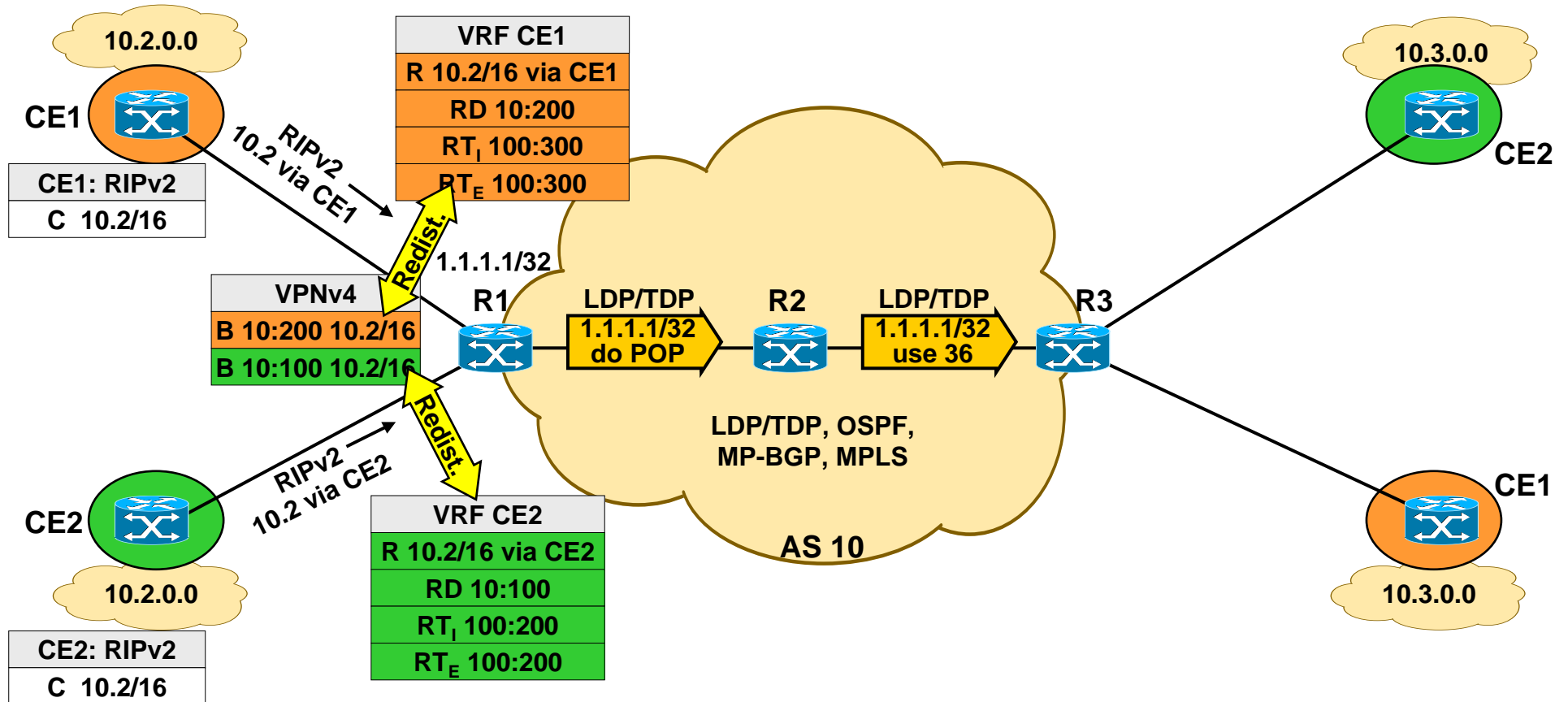
- **For non-overlapping VPNs**
  - The Route Distinguisher would be sufficient
- **For overlapping VPNs**
  - The Route Distinguisher is not sufficient to achieve the new sight (the Extranet policy) of VPNs
- **In order to implement this new sight of VPNs in case of overlapping VPNs**
  - the Route Target was introduced in the MPLS\_VPN Architecture

# The real Role of the Route Target

- **PE router which announces a VPNv4 route**
  - uses the Route Target community to specify in which foreign VRFs the announced route should be installed
  - Route Target has export meaning
  
- **PE router which receives a VPNv4 route**
  - uses the received Route Target community to decide in which local VRFs the announced route should be installed
  - Route Target has import meaning

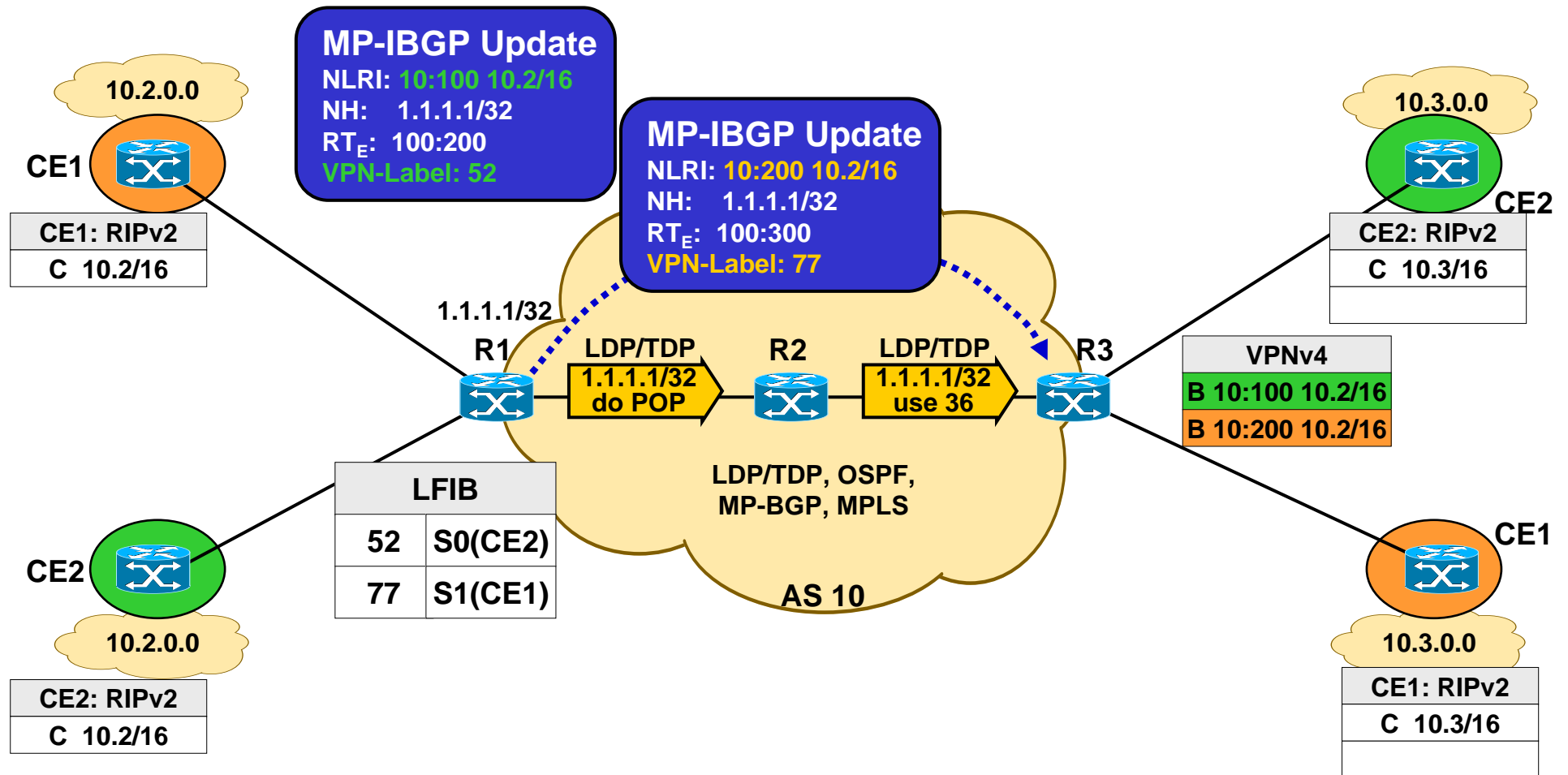
# MPLS VPN (The Complete Story)

1



# MPLS VPN (The Complete Story)

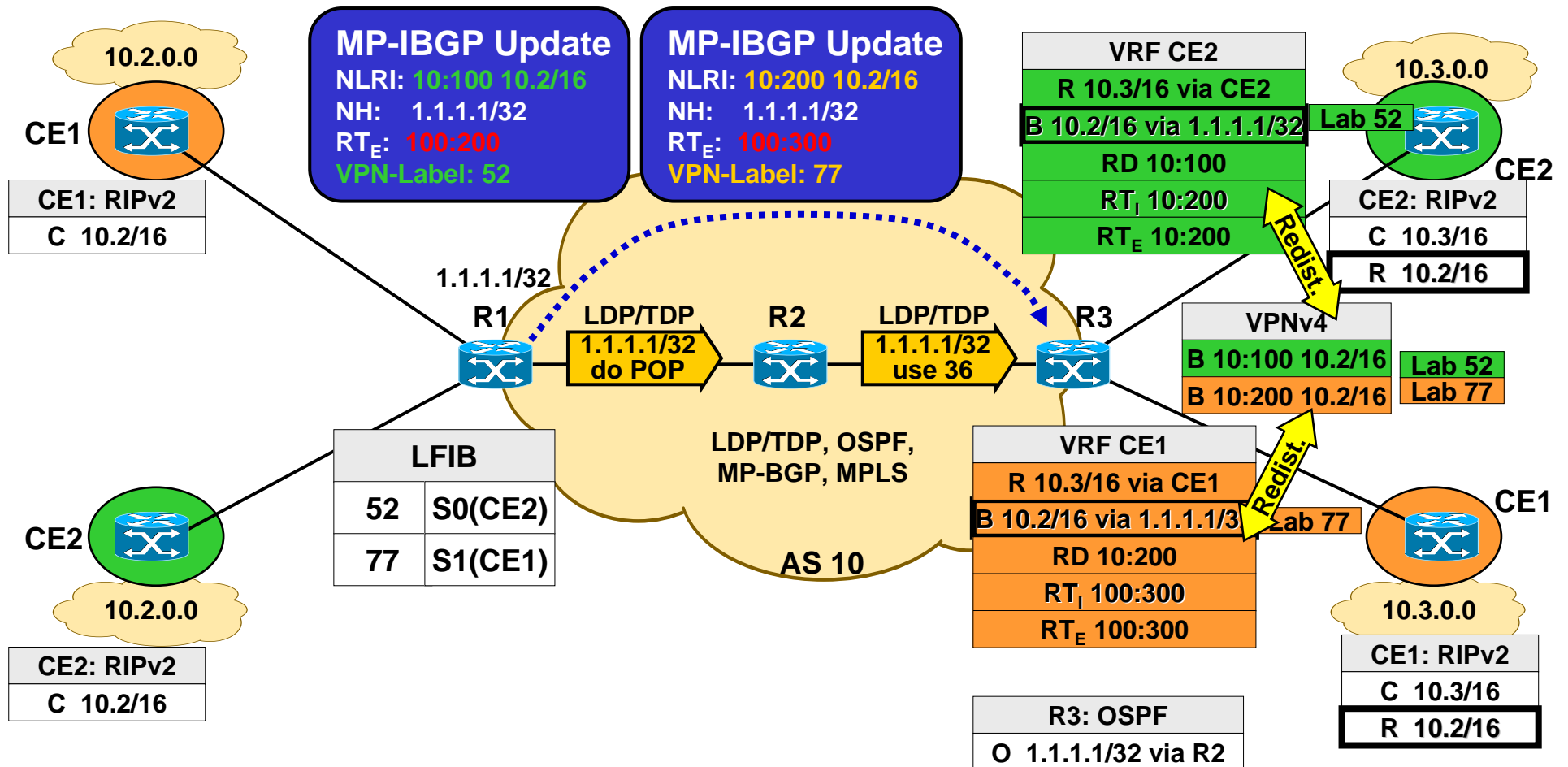
2



IGP Metric → MED

# MPLS VPN (The Complete Story)

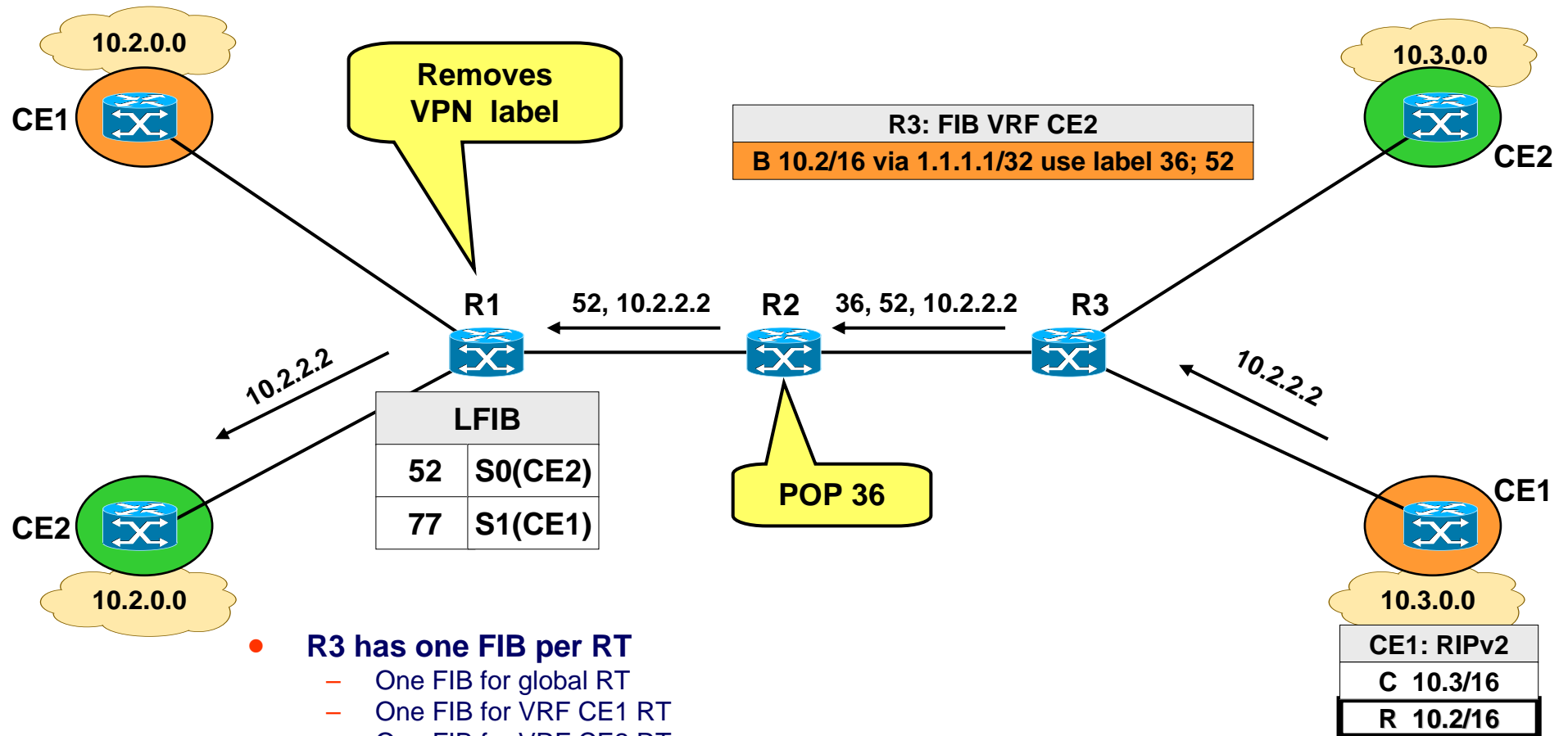
3



MED → IGP Metric

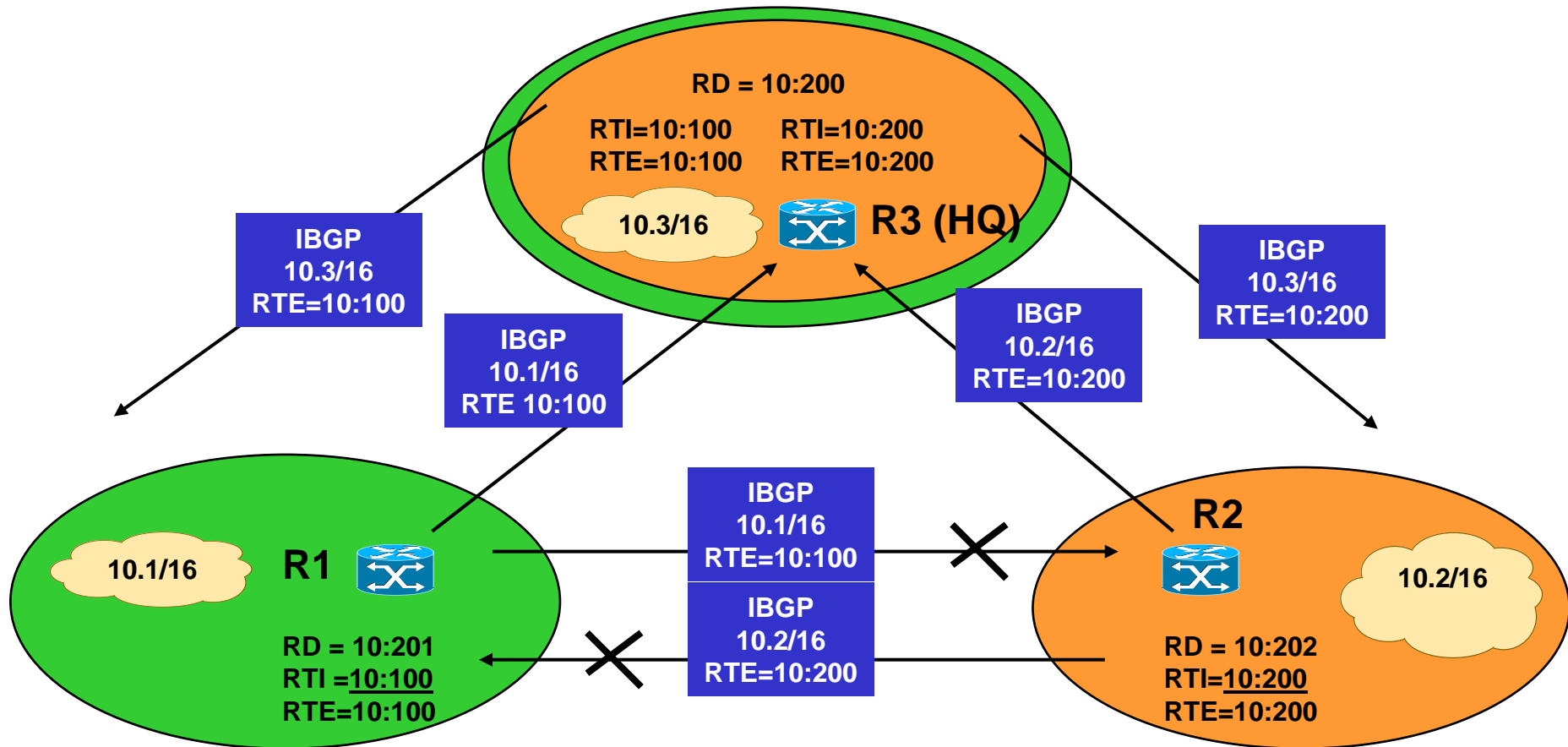
# MPLS VPN (The Complete Story)

4



- **R3 has one FIB per RT**
  - One FIB for global RT
  - One FIB for VRF CE1 RT
  - One FIB for VRF CE2 RT
- **Each MPLS-Router has exactly one LFIB**
  - PE routers must be connected with CE routers via (sub) interfaces

# Example for Overlapping VPNs using Different Route Targets



- IBGP Split Horizon Rule assures that R3 (HQ) does not forward routes learned by peers
- IP addresses must be unique in overlapping situations!

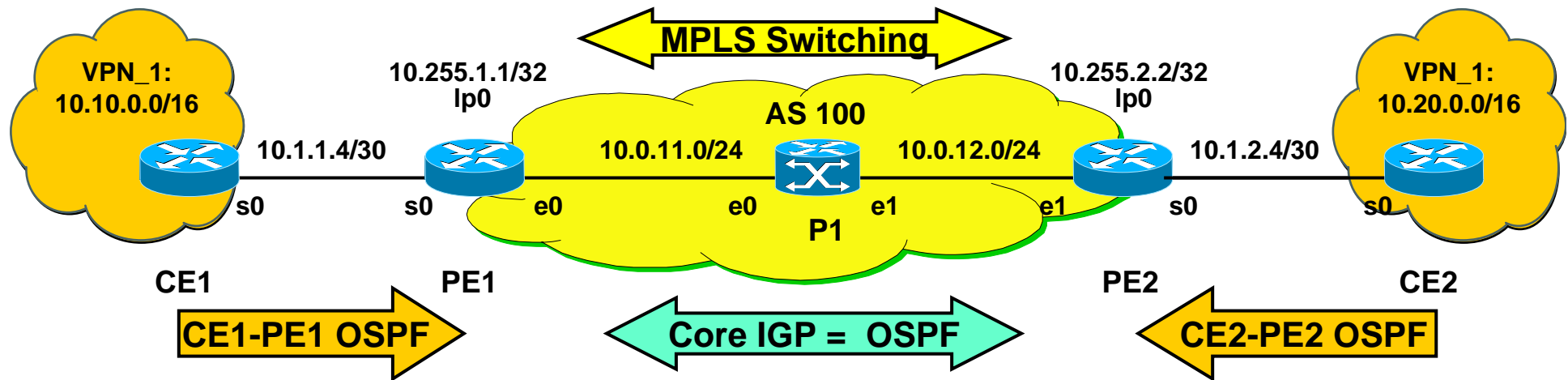
# Agenda

---

- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing



# IP Addressing, OSPF Routing in VPN\_1, Basic OSPF Routing and MPLS in AS 100



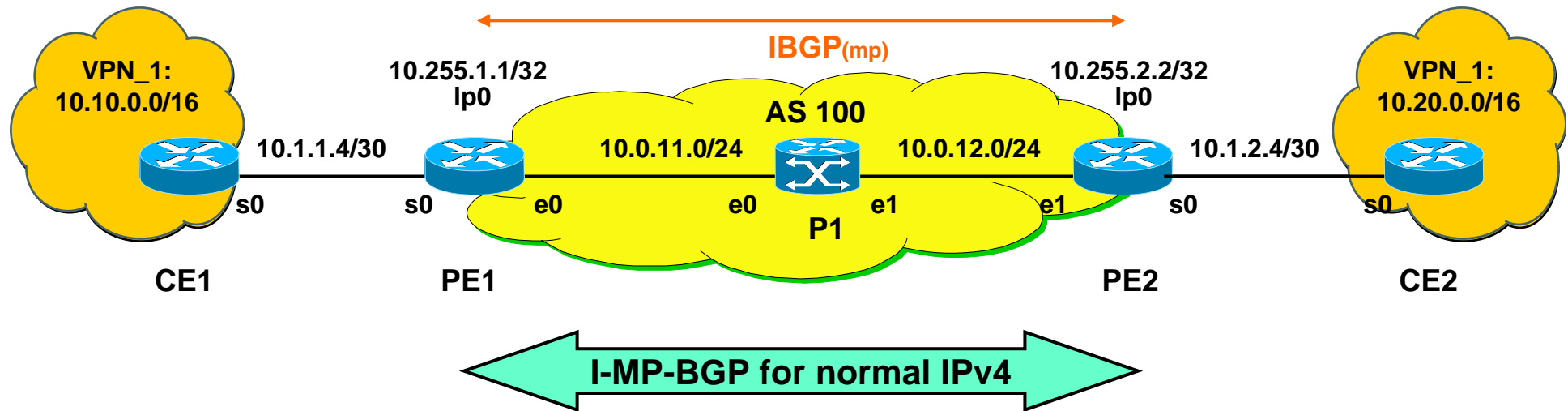
CE1:  
 int s0  
 ip address 10.1.1.5 255.255.255.252  
router ospf 10  
network 10.1.1.5 0.0.0.0 area 0

PE1:  
 ip cef  
 int loopback 0  
 ip address 10.255.1.1 255.255.255.255  
 int s0  
 ip address 10.1.1.6 255.255.255.252  
 int e0  
 ip address 10.0.11.1 255.255.255.0  
mpls ip  
router ospf 100  
network 10.0.11.1 0.0.0.0 area 0

CE2:  
 int s0  
 ip address 10.1.2.5 255.255.255.252  
router ospf 10  
network 10.1.2.5 0.0.0.0 area 0

PE2:  
 ip cef  
 int loopback 0  
 ip address 10.255.2.2 255.255.255.255  
 int s0  
 ip address 10.1.2.6 255.255.255.252  
 int e0  
 ip address 10.0.12.2 255.255.255.0  
mpls ip  
router ospf 100  
network 10.0.12.1 0.0.0.0 area 0

# Start Normal I-BGP in AS 100



```

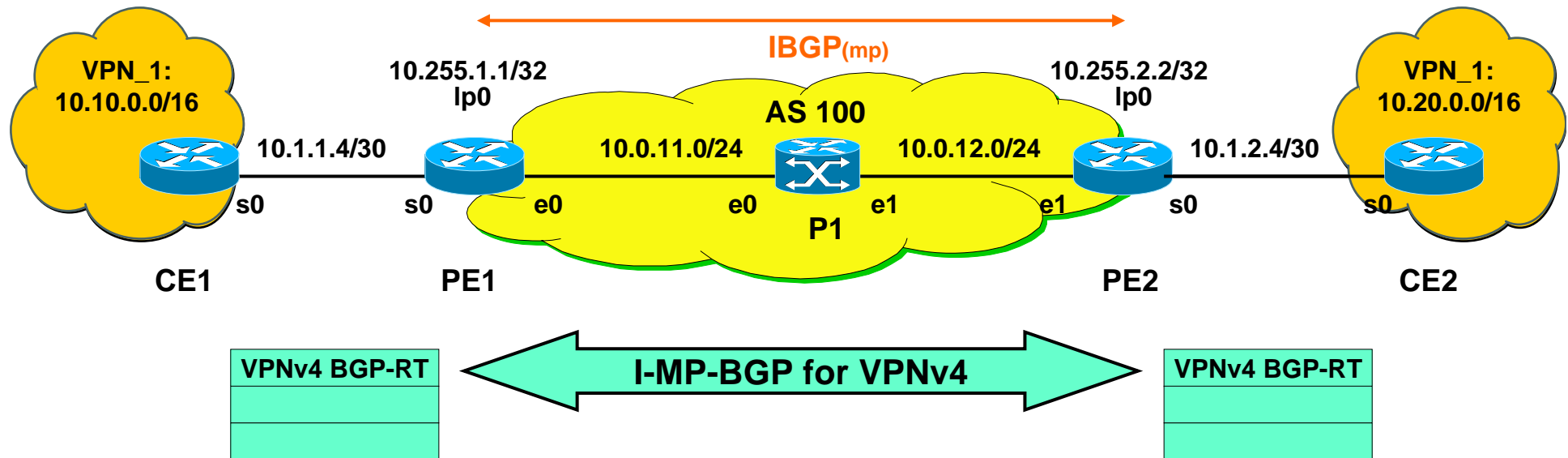
PE1:
int loopback 0
ip address 10.255.1.1 255.255.255.255

router bgp 100
  no bgp default ipv4-unicast
  bgp router-id 10.255.1.1
  neighbor 10.255.2.2 remote-as 100
  neighbor 10.255.2.2 update-source loop 0
  address-family ipv4
    neighbor 10.255.2.2 next-hop-self
    neighbor 10.255.2.2 activate
    no auto-summary (default)
    no synchronization (default)
  exit address-family
  
```

```

PE2:
int loopback 0
ip address 10.255.2.2 255.255.255.255
router bgp 100
  no bgp default ipv4-unicast
  bgp router-id 10.255.2.2
  neighbor 10.255.1.1 remote-as 100
  neighbor 10.255.1.1 update-source loop 0
  address-family ipv4
    neighbor 10.255.1.1 next-hop-self
    neighbor 10.255.1.1 activate
    no auto-summary (default)
    no synchronization (default)
  exit address-family
  
```

# Start MP-BGP in AS 100



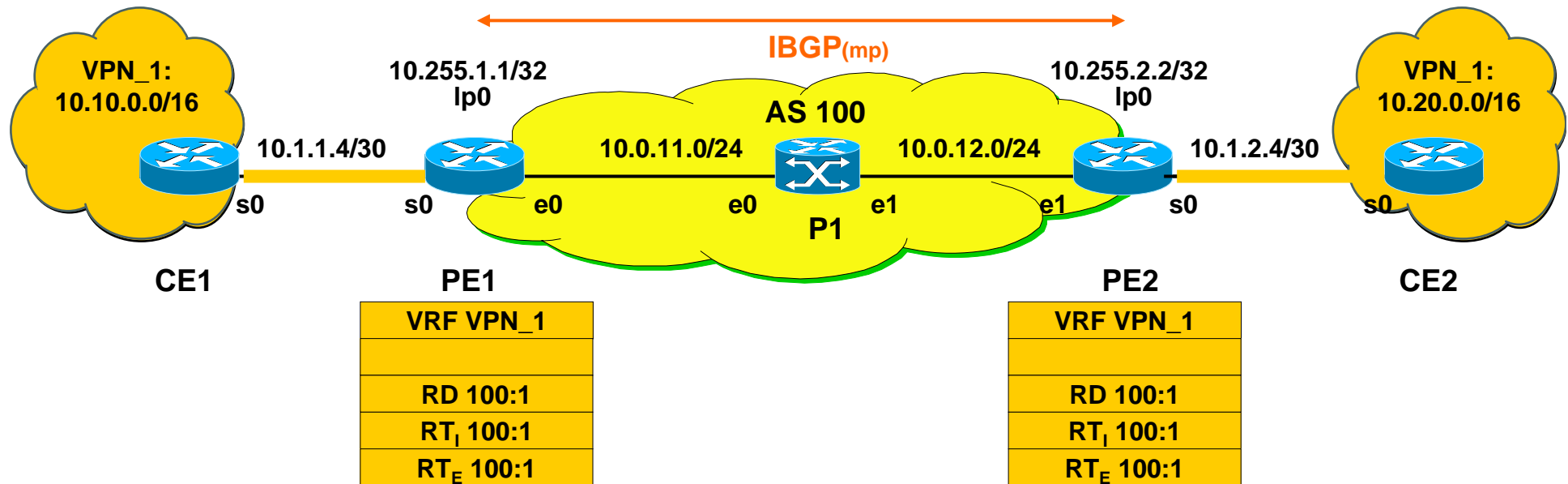
```

PE1:
int loopback 0
 ip address 10.255.1.1 255.255.255.255
router bgp 100
 no bgp default ipv4-unicast
 bgp router-id 10.255.1.1
 neighbor 10.255.2.2 remote-as 100
 neighbor 10.255.2.2 update-source loop 0
address-family vpnv4
 neighbor 10.255.2.2 activate
 neighbor 10.255.2.2 next-hop-self
 neighbor 10.255.2.2 send-community extended (default)
exit-address-family
  
```

```

PE2:
int loopback 0
 ip address 10.255.2.2 255.255.255.255
router bgp 100
 no bgp default ipv4-unicast
 bgp router-id 10.255.2.2
 neighbor 10.255.1.1 remote-as 100
 neighbor 10.255.1.1 update-source loop 0
address-family vpnv4
 neighbor 10.255.1.1 activate
 neighbor 10.255.1.1 next-hop-self
 neighbor 10.255.1.1 send-community extended
exit-address-family
  
```

# Create VRF and Bring Interface into VRF (PE router)



PE1:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

int s0

```
ip vrf forwarding VPN_1
ip address 10.1.1.6 255.255.255.252
```

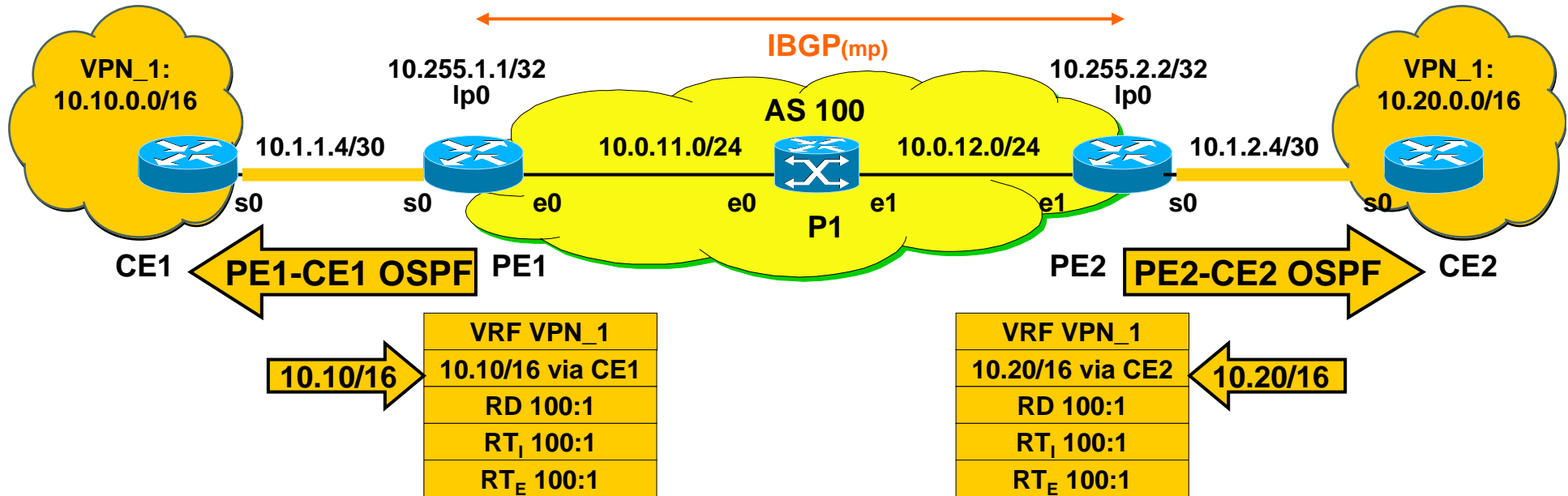
PE2:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

int s0

```
ip vrf forwarding VPN_1
ip address 10.1.2.6 255.255.255.252
```

# Start Dynamic Routing (OSPF) towards CE (PE router)



PE1:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

```
int s0
ip vrf forwarding VPN_1
ip address 10.1.1.6 255.255.255.252
```

```
router ospf 110 vrf VPN_1
network 10.1.1.6 0.0.0.0 area 0
```

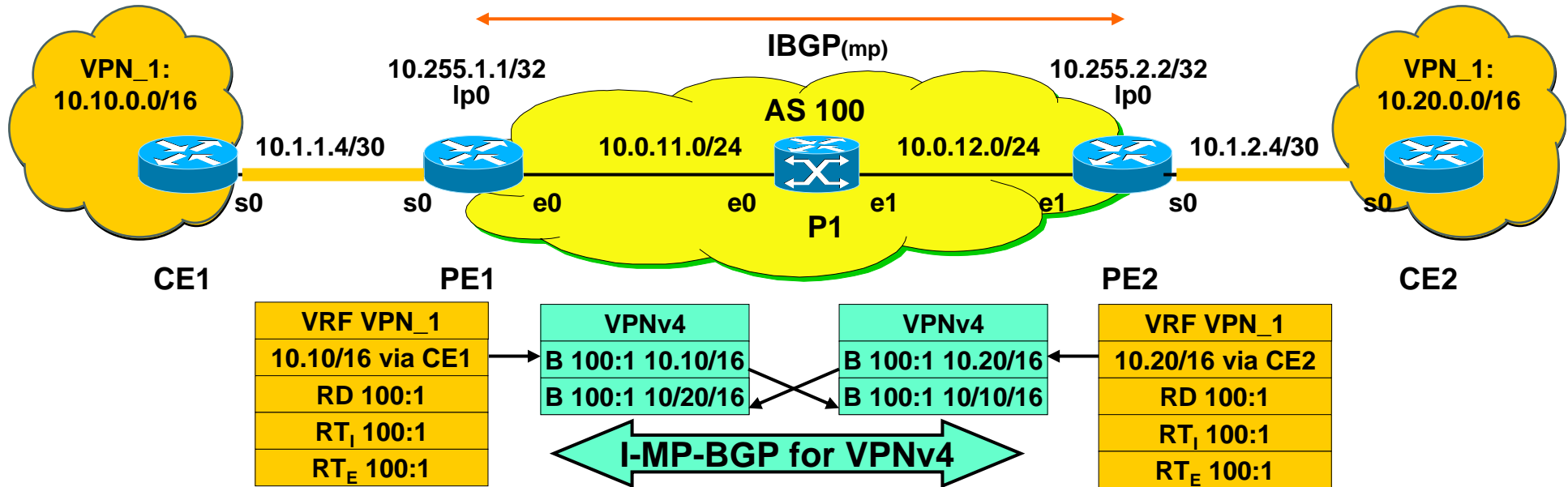
PE2:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

```
int s0
ip vrf forwarding VPN_1
ip address 10.1.2.6 255.255.255.252
```

```
router ospf 120 vrf VPN_1
network 10.1.2.6 0.0.0.0 area 0
```

# Redistributing VRF OSPF into MP-BGP and Transport of VPNv4 routes via I-MP-BGP (PE router)



PE1:  
 ip vrf **VPN\_1**  
 rd 100:1

router ospf 110 vrf **VPN\_1**  
 network 10.1.1.6 0.0.0.0 area 0

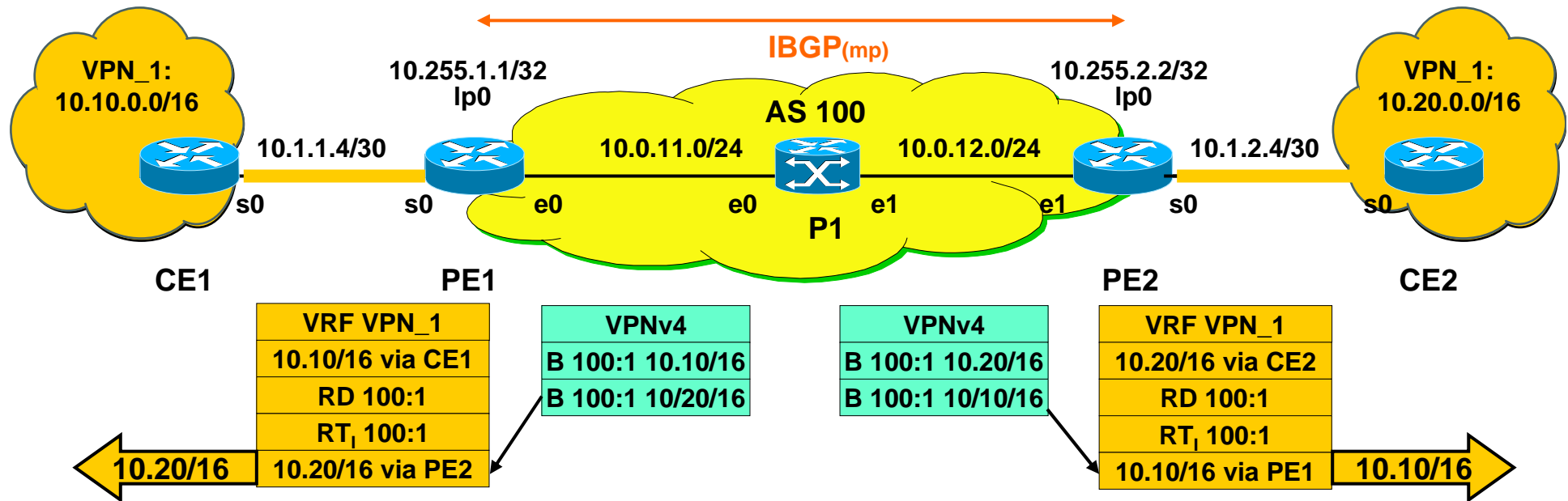
router bgp 100  
 no bgp default ipv4-unicast  
 address-family ipv4 vrf **VPN\_1**  
redistribute ospf 110 match internal  
 no auto-summary  
 no synchronization  
 exit-address-family

PE2:  
 ip vrf **VPN\_1**  
 rd 100:1

router ospf 120 vrf **VPN\_1**  
 network 10.1.2.6 0.0.0.0 area 0

router bgp 100  
 no bgp default ipv4-unicast  
 address-family ipv4 vrf **VPN\_1**  
redistribute ospf 120 match internal  
 no auto-summary  
 no synchronization  
 exit-address-family

# Redistribution of VPNv4 routes into VRF OSPF (PE router)



PE1:  
 router ospf 110 vrf **VPN\_1**  
 network 10.1.1.6 0.0.0.0 area 0  
redistribute bgp 100 metric <20> subnet

router bgp 100  
 no bgp default ipv4-unicast  
 address-family ipv4 vrf **VPN\_1**  
 redistribute ospf 100 match internal  
 no auto-summary  
 exit-address-family

PE2:  
 router ospf 120 vrf **VPN\_1**  
 network 10.1.2.6 0.0.0.0 area 0  
redistribute bgp 100 metric <20> subnets

router bgp 100  
 no bgp default ipv4-unicast  
 address-family ipv4 vrf **VPN\_1**  
 redistribute ospf 100 match internal  
 no auto-summary  
 exit-address-family

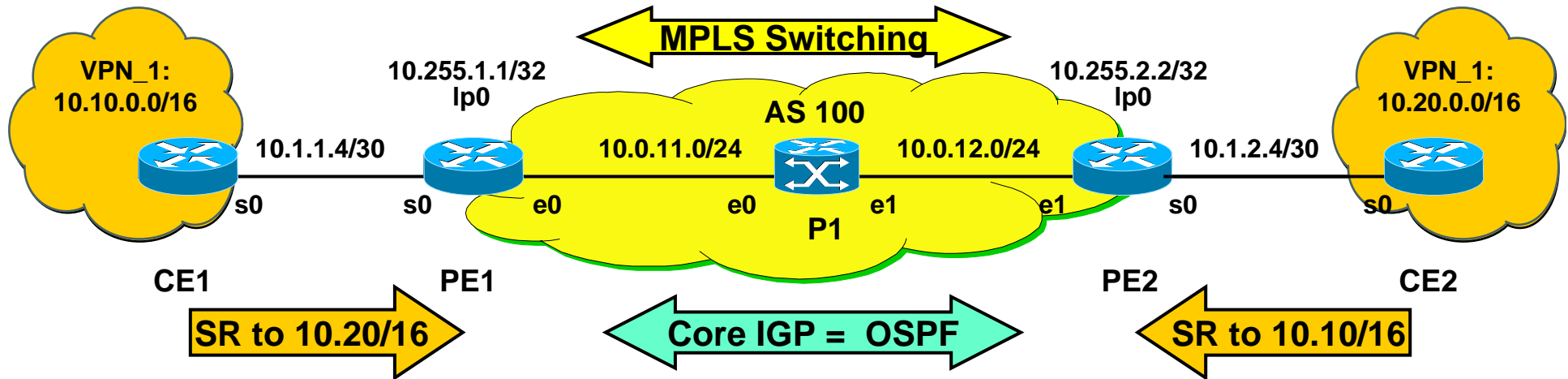
# Agenda

---

- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing



# IP Addressing, Static Routing in VPN\_1, Basic OSPF Routing and MPLS in AS 100



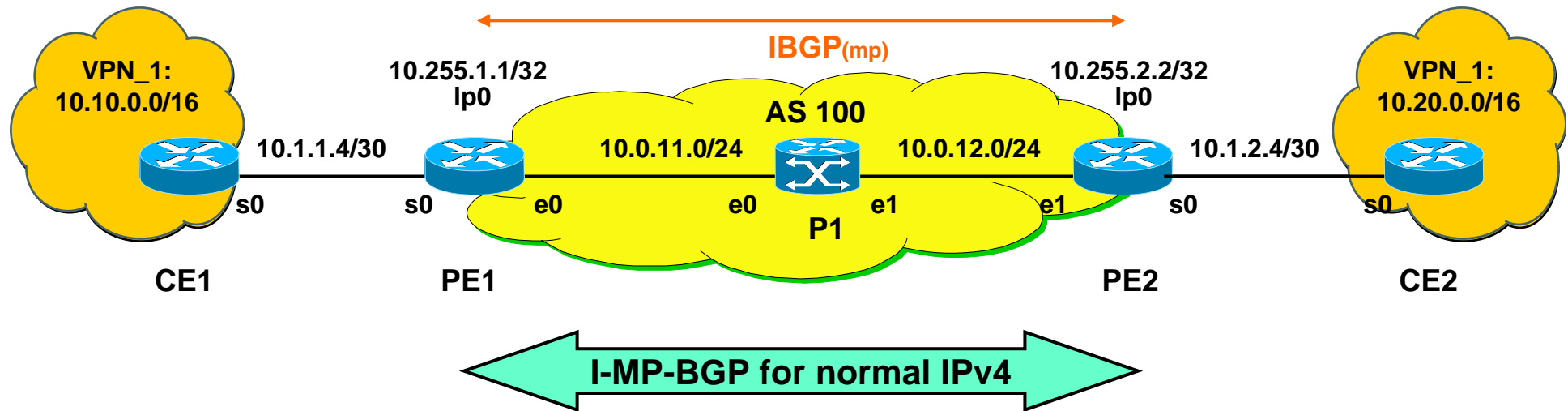
CE1:  
 int s0  
 ip address 10.1.1.5 255.255.255.252  
ip route 10.20.0.0 255.255.0.0 10.1.1.6

PE1 (OSPF and MPLS in Backbone):  
 ip cef  
 int loopback 0  
 ip address 10.255.1.1 255.255.255.255  
 int s0  
 ip address 10.1.1.6 255.255.255.252  
 int e0  
 ip address 10.0.11.1 255.255.255.0  
mpls ip  
router ospf 100  
network 10.0.11.1 0.0.0.0 area 0

CE2:  
 int s0  
 ip address 10.1.2.5 255.255.255.252  
ip route 10.10.0.0 255.255.0.0 10.1.2.6

PE2 (OSPF and MPLS in Backbone):  
 ip cef  
 int loopback 0  
 ip address 10.255.2.2 255.255.255.255  
 int s0  
 ip address 10.1.2.6 255.255.255.252  
 int e0  
 ip address 10.0.12.2 255.255.255.0  
mpls ip  
router ospf 100  
network 10.0.12.1 0.0.0.0 area 0

# Start Normal I-BGP in AS 100



```

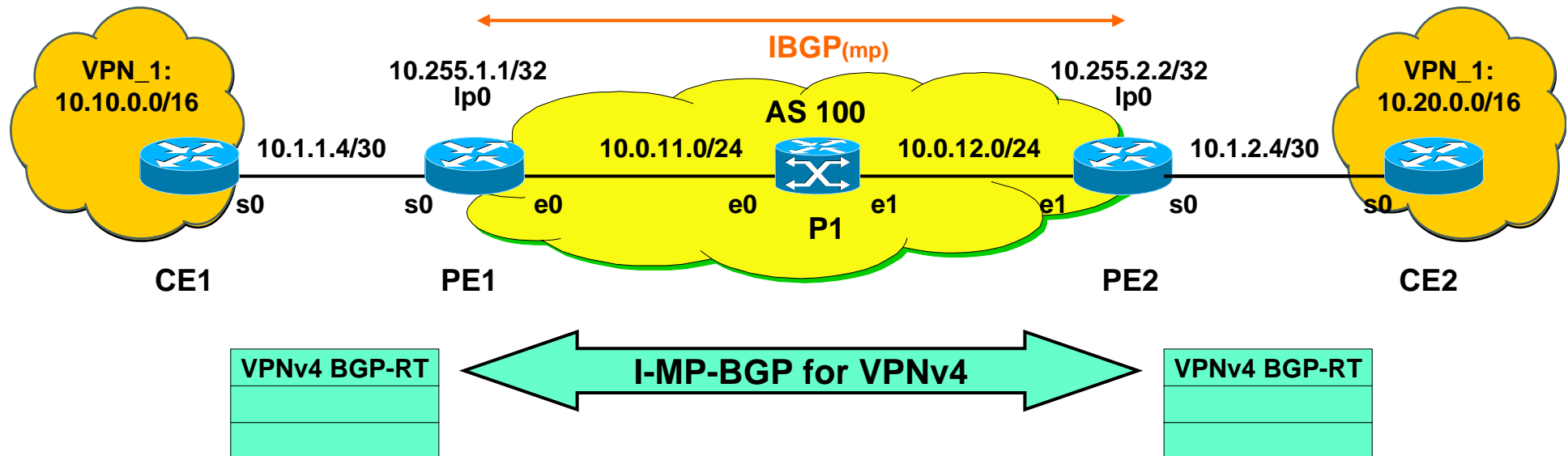
PE1:
int loopback 0
ip address 10.255.1.1 255.255.255.255

router bgp 100
  no bgp default ipv4-unicast
  bgp router-id 10.255.1.1
  neighbor 10.255.2.2 remote-as 100
  neighbor 10.255.2.2 update-source loop 0
  address-family ipv4
    neighbor 10.255.2.2 next-hop-self
    neighbor 10.255.2.2 activate
    no auto-summary (default)
    no synchronization (default)
  exit address-family
  
```

```

PE2:
int loopback 0
ip address 10.255.2.2 255.255.255.255
router bgp 100
  no bgp default ipv4-unicast
  bgp router-id 10.255.2.2
  neighbor 10.255.1.1 remote-as 100
  neighbor 10.255.1.1 update-source loop 0
  address-family ipv4
    neighbor 10.255.1.1 next-hop-self
    neighbor 10.255.1.1 activate
    no auto-summary (default)
    no synchronization (default)
  exit address-family
  
```

# Start MP-BGP in AS 100



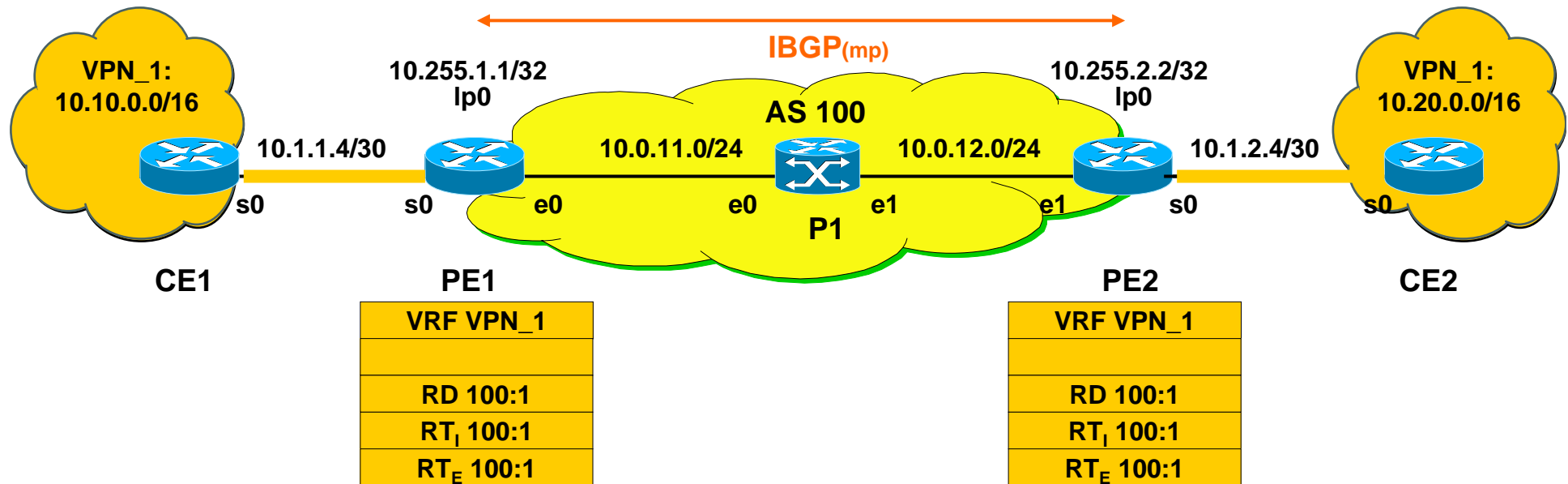
```

PE1:
int loopback 0
 ip address 10.255.1.1 255.255.255.255
router bgp 100
 no bgp default ipv4-unicast
 bgp router-id 10.255.1.1
 neighbor 10.255.2.2 remote-as 100
 neighbor 10.255.2.2 update-source loop 0
address-family vpnv4
 neighbor 10.255.2.2 activate
 neighbor 10.255.2.2 next-hop-self
 neighbor 10.255.2.2 send-community extended (default)
exit-address-family
  
```

```

PE2:
int loopback 0
 ip address 10.255.2.2 255.255.255.255
router bgp 100
 no bgp default ipv4-unicast
 bgp router-id 10.255.2.2
 neighbor 10.255.1.1 remote-as 100
 neighbor 10.255.1.1 update-source loop 0
address-family vpnv4
 neighbor 10.255.1.1 activate
 neighbor 10.255.1.1 next-hop-self
 neighbor 10.255.1.1 send-community extended
exit-address-family
  
```

# Create VRF and Bring Interface into VRF (PE router)



PE1:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

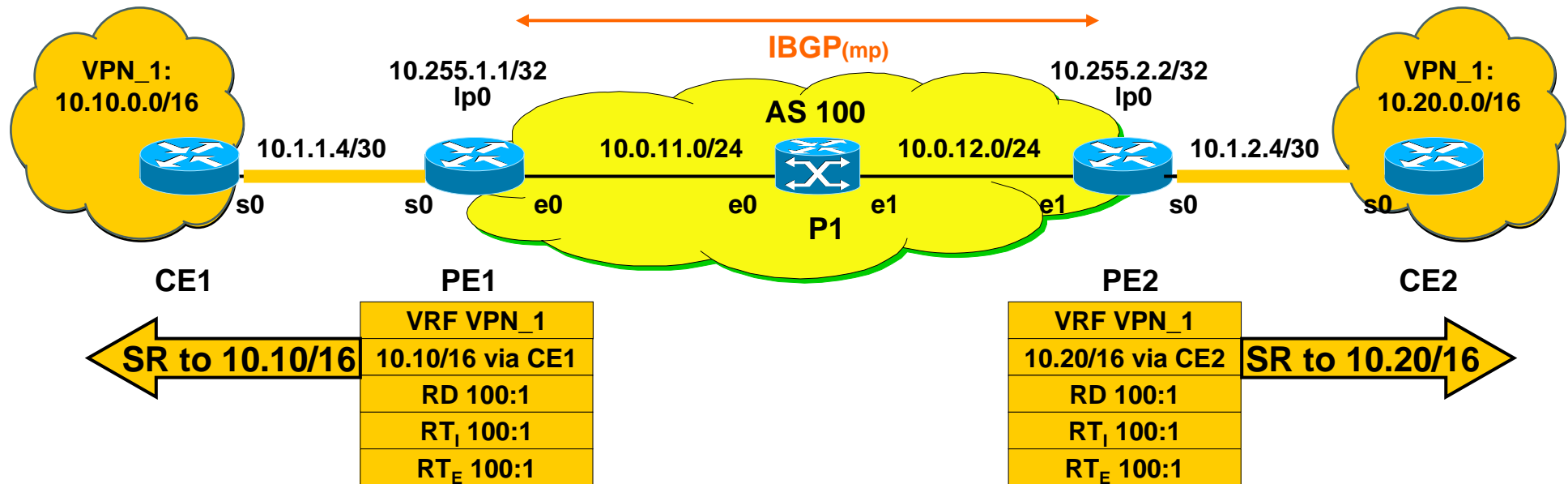
```
int s0
ip vrf forwarding VPN_1
ip address 10.1.1.6 255.255.255.252
```

PE2:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

```
int s0
ip vrf forwarding VPN_1
ip address 10.1.2.6 255.255.255.252
```

# Static Routing (SR) towards CE (PE router)



PE1:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

```
int s0
ip vrf forwarding VPN_1
ip address 10.1.1.6 255.255.255.252
```

```
ip route vrf VPN_1 10.10.0.0 255.255.0.0 serial 0 10.1.1.5
```

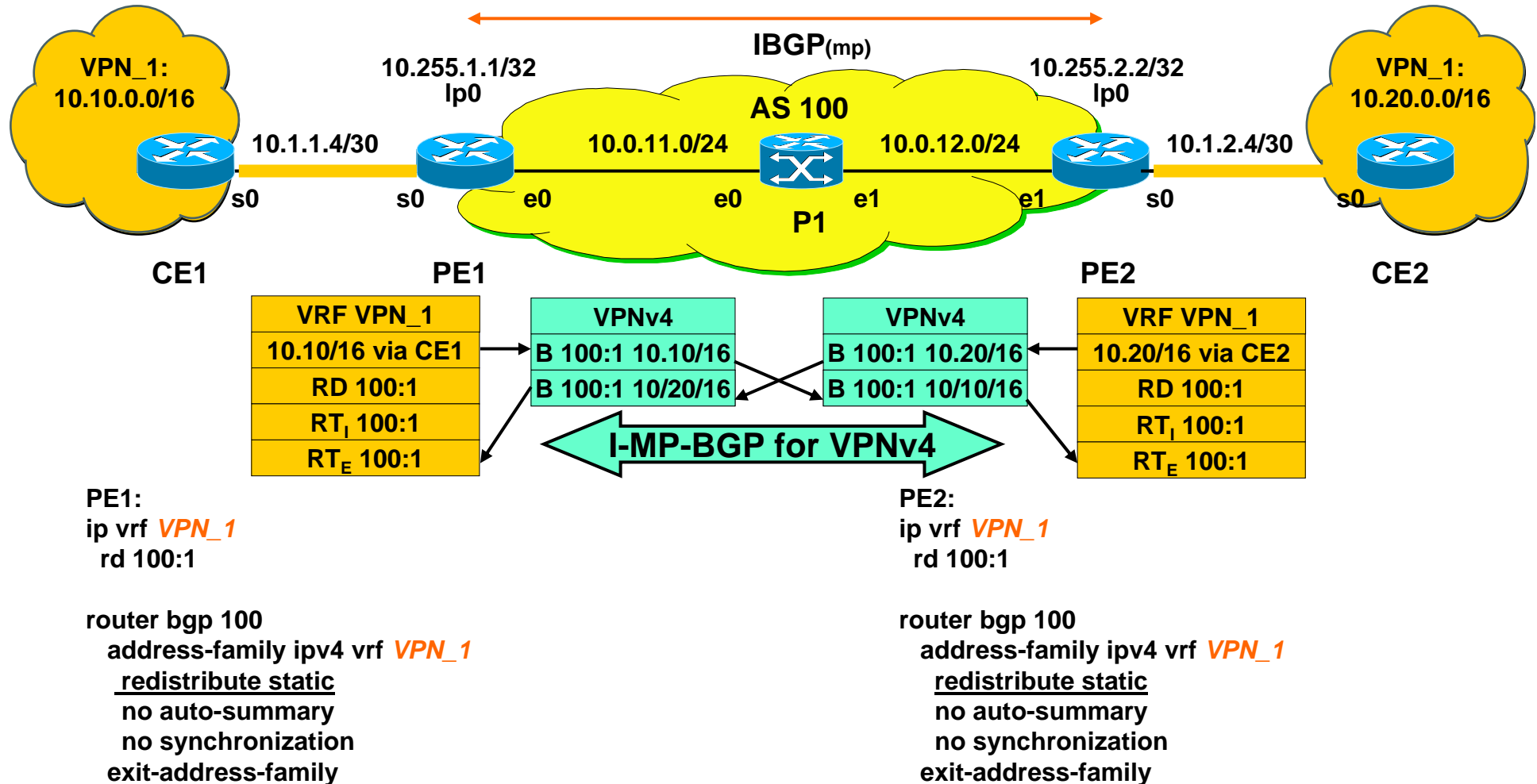
PE2:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

```
int s0
ip vrf forwarding VPN_1
ip address 10.1.2.6 255.255.255.252
```

```
ip route vrf VPN_1 10.20.0.0 255.255.0.0 serial 0 10.1.2.5
```

# Redistributing Static into MP-BGP and Transport of Static routes via I-MP-BGP (PE router)

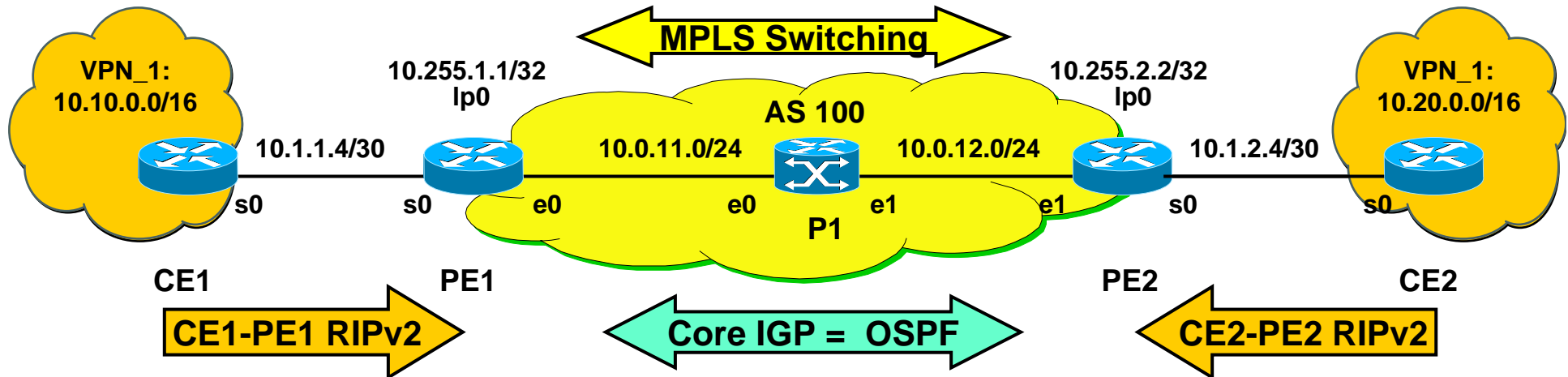


# Agenda

---

- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing

# IP Addressing, RIPv2 Routing in VPN\_1, Basic OSPF Routing and MPLS in AS 100



CE1:  
 int s0  
 ip address 10.1.1.5 255.255.255.252  
router rip  
version 2  
network 10.0.0.0  
no auto-summary

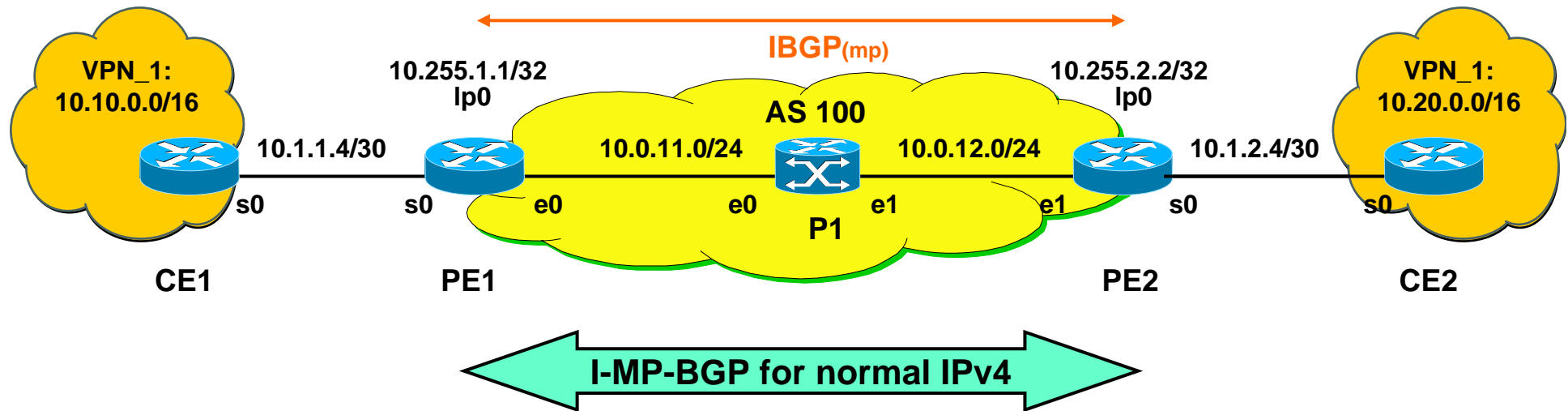
PE1:  
 ip cef  
 int loopback 0  
 ip address 10.255.1.1 255.255.255.255  
 int s0  
 ip address 10.1.1.6 255.255.255.252  
 int e0  
 ip address 10.0.11.1 255.255.255.0  
mpls ip  
router ospf 100  
network 10.0.11.1 0.0.0.0 area 0

CE2:  
 int s0  
 ip address 10.1.2.5 255.255.255.252  
router rip  
version 2  
network 10.0.0.0  
no auto-summary

PE2:  
 ip cef  
 int loopback 0  
 ip address 10.255.2.2 255.255.255.255  
 int s0  
 ip address 10.1.2.6 255.255.255.252  
 int e0  
 ip address 10.0.12.2 255.255.255.0  
mpls ip  
router ospf 100  
network 10.0.12.1 0.0.0.0 area 0



# Start Normal I-BGP in AS 100



```

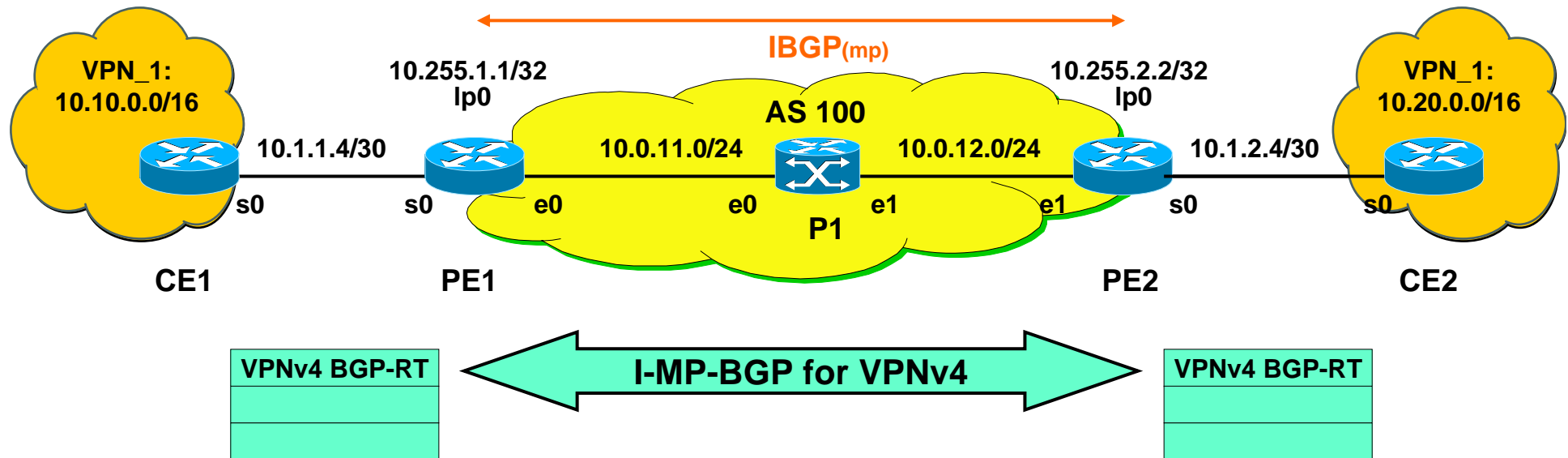
PE1:
int loopback 0
 ip address 10.255.1.1 255.255.255.255

router bgp 100
  no bgp default ipv4-unicast
  bgp router-id 10.255.1.1
  neighbor 10.255.2.2 remote-as 100
  neighbor 10.255.2.2 update-source loop 0
  address-family ipv4
    neighbor 10.255.2.2 next-hop-self
    neighbor 10.255.2.2 activate
    no auto-summary (default)
    no synchronization (default)
  exit address-family
  
```

```

PE2:
int loopback 0
 ip address 10.255.2.2 255.255.255.255
router bgp 100
  no bgp default ipv4-unicast
  bgp router-id 10.255.2.2
  neighbor 10.255.1.1 remote-as 100
  neighbor 10.255.1.1 update-source loop 0
  address-family ipv4
    neighbor 10.255.1.1 next-hop-self
    neighbor 10.255.1.1 activate
    no auto-summary (default)
    no synchronization (default)
  exit address-family
  
```

# Start MP-BGP in AS 100



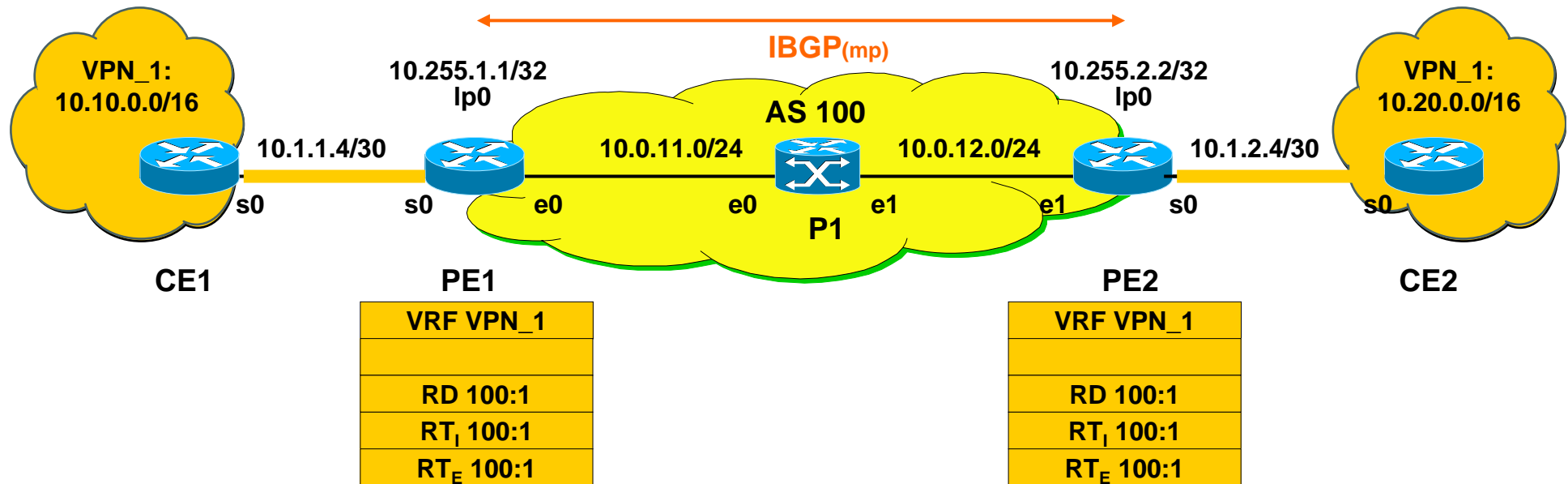
```

PE1:
int loopback 0
 ip address 10.255.1.1 255.255.255.255
router bgp 100
 no bgp default ipv4-unicast
 bgp router-id 10.255.1.1
 neighbor 10.255.2.2 remote-as 100
 neighbor 10.255.2.2 update-source loop 0
 address-family vpnv4
  neighbor 10.255.2.2 activate
  neighbor 10.255.2.2 next-hop-self
  neighbor 10.255.2.2 send-community extended (default)
 exit-address-family
  
```

```

PE2:
int loopback 0
 ip address 10.255.2.2 255.255.255.255
router bgp 100
 no bgp default ipv4-unicast
 bgp router-id 10.255.2.2
 neighbor 10.255.1.1 remote-as 100
 neighbor 10.255.1.1 update-source loop 0
 address-family vpnv4
  neighbor 10.255.1.1 activate
  neighbor 10.255.1.1 next-hop-self
  neighbor 10.255.1.1 send-community extended
 exit-address-family
  
```

# Create VRF and Bring Interface into VRF (PE router)



PE1:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

int s0

```
ip vrf forwarding VPN_1
ip address 10.1.1.6 255.255.255.252
```

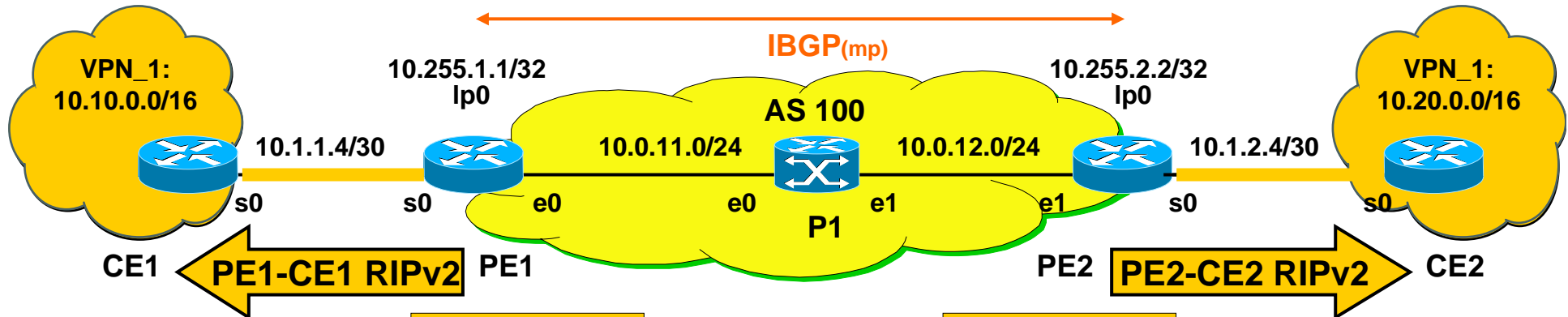
PE2:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

int s0

```
ip vrf forwarding VPN_1
ip address 10.1.2.6 255.255.255.252
```

# Start Dynamic Routing (RIPv2) towards CE (PE router)



10.10/16	VRF VPN_1
	10.10/16 via CE1
	RD 100:1
	RT <sub>I</sub> 100:1
	RT <sub>E</sub> 100:1

10.20/16	VRF VPN_1
	10.20/16 via CE2
	RD 100:1
	RT <sub>I</sub> 100:1
	RT <sub>E</sub> 100:1

PE1:

```
ip vrf VPN_1
rd 100:1
```

```
int s0
```

```
ip vrf forwarding VPN_1
ip address 10.1.1.6 255.255.255.252
```

```
router rip
version 2
```

```
address-family ipv4 vrf VPN_1
network 10.0.0.0
no auto-summary
exit-address-family
```

PE2:

```
ip vrf VPN_1
rd 100:1
```

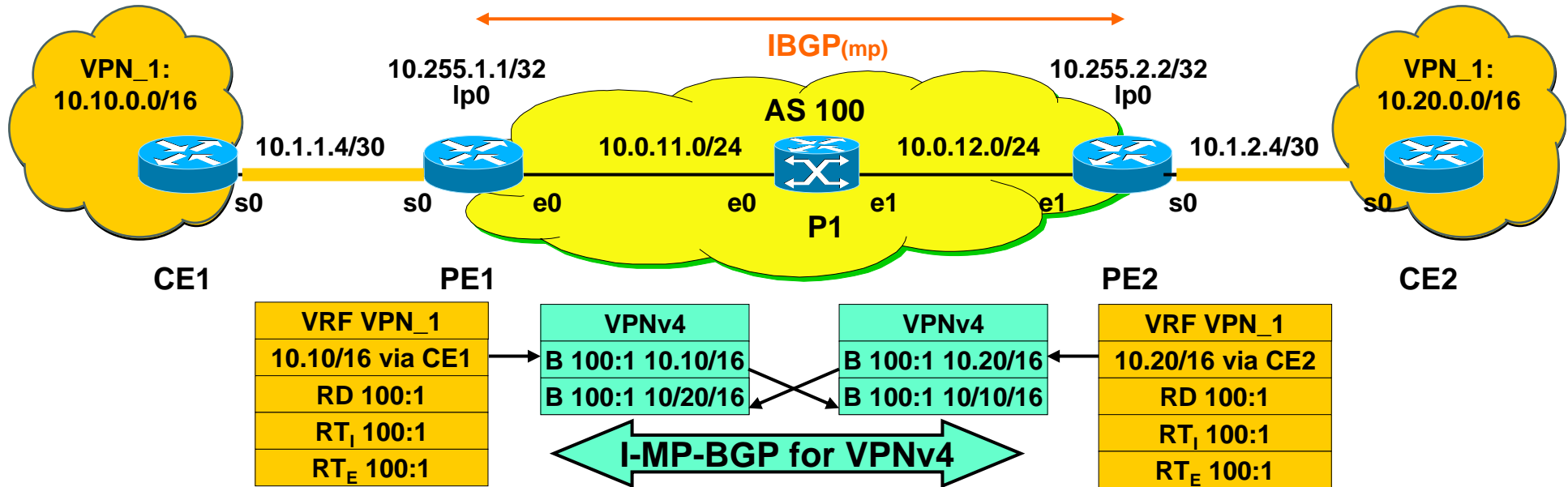
```
int s0
```

```
ip vrf forwarding VPN_1
ip address 10.1.2.6 255.255.255.252
```

```
router rip
version 2
```

```
address-family ipv4 vrf VPN_1
network 10.0.0.0
no auto-summary
exit-address-family
```

# Redistributing VRF RIPv2 into MP-BGP and Transport of VPNv4 routes via I-MP-BGP (PE router)



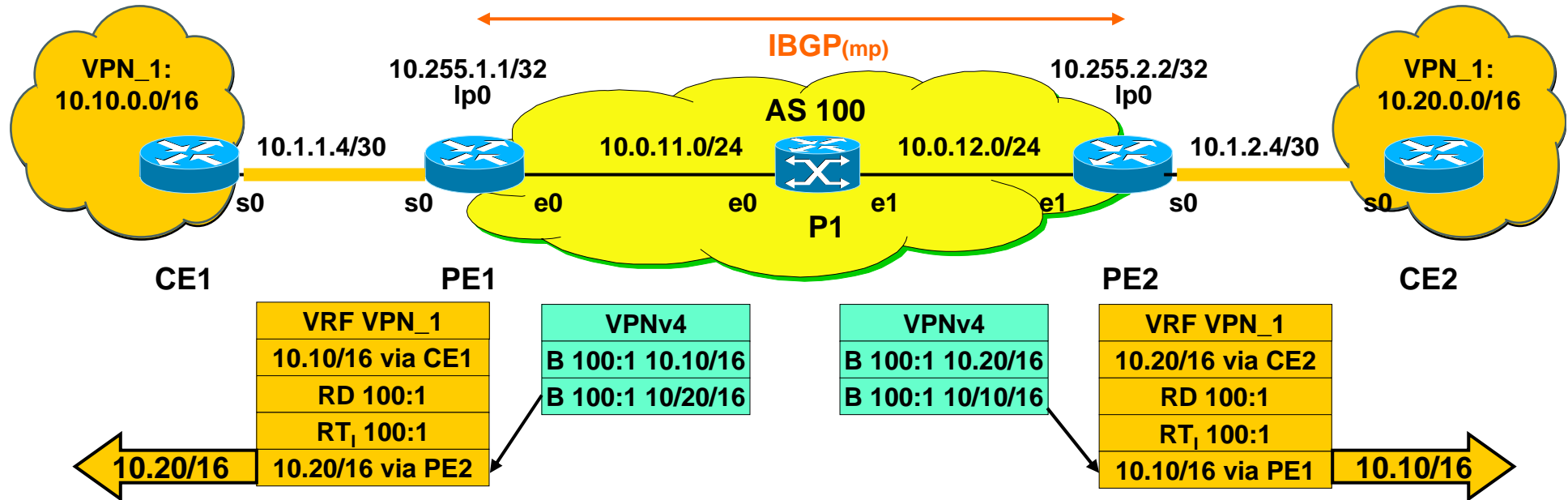
PE1:  
 ip vrf **VPN\_1**  
 rd 100:1

PE2:  
 ip vrf **VPN\_1**  
 rd 100:1

```
router bgp 100
no bgp default ipv4-unicast
address-family ipv4 vrf VPN_1
  redistribute rip
no auto-summary
no synchronization
exit-address-family
```

```
router bgp 100
no bgp default ipv4-unicast
address-family ipv4 vrf VPN_1
  redistribute rip
no auto-summary
no synchronization
exit-address-family
```

# Redistribution of VPNv4 routes into VRF RIPv2 (PE router)



PE1:  
 router rip  
 version 2  
 address-family ipv4 vrf **VPN\_1**  
redistribute bgp 100 metric transparent  
 exit-address-family

router bgp 100  
 no bgp default ipv4-unicast  
 address-family ipv4 vrf **VPN\_1**  
 redistribute rip  
 exit-address-family

PE2:  
 router rip  
 version 2  
 address-family ipv4 vrf **VPN\_1**  
redistribute bgp 100 metric transparent  
 exit-address-family

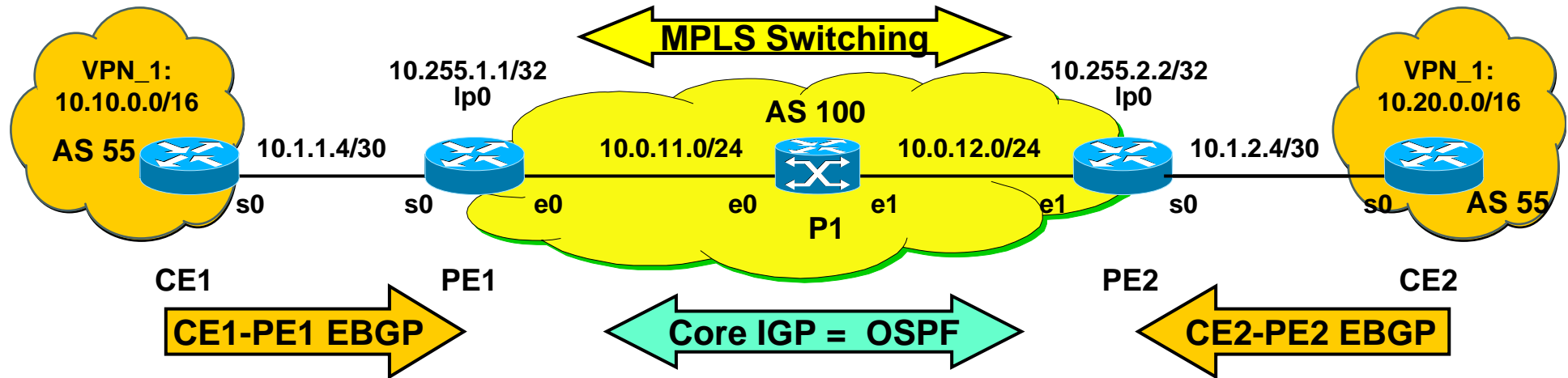
router bgp 100  
 no bgp default ipv4-unicast  
 address-family ipv4 vrf **VPN\_1**  
 redistribute rip  
 exit-address-family

# Agenda

---

- **MP-BGP**
- **VPN Overview**
- **MPLS VPN Architecture**
- **MPLS VPN Basic VPNs**
- **MPLS VPN Complex VPNs**
- **MPLS VPN Configuration (Cisco)**
  - CE-PE OSPF Routing
  - CE-PE Static Routing
  - CE-PE RIP Routing
  - CE-PE External BGP Routing

# IP Addressing, EBGP Routing in VPN\_1, Basic OSPF Routing and MPLS in AS 100



CE1:  
 int s0  
 ip address 10.1.1.5 255.255.255.252  
 router bgp 55  
 neighbor 10.1.1.6 remote-as 100  
 network 10.10.0.0 mask 255.255.0.0

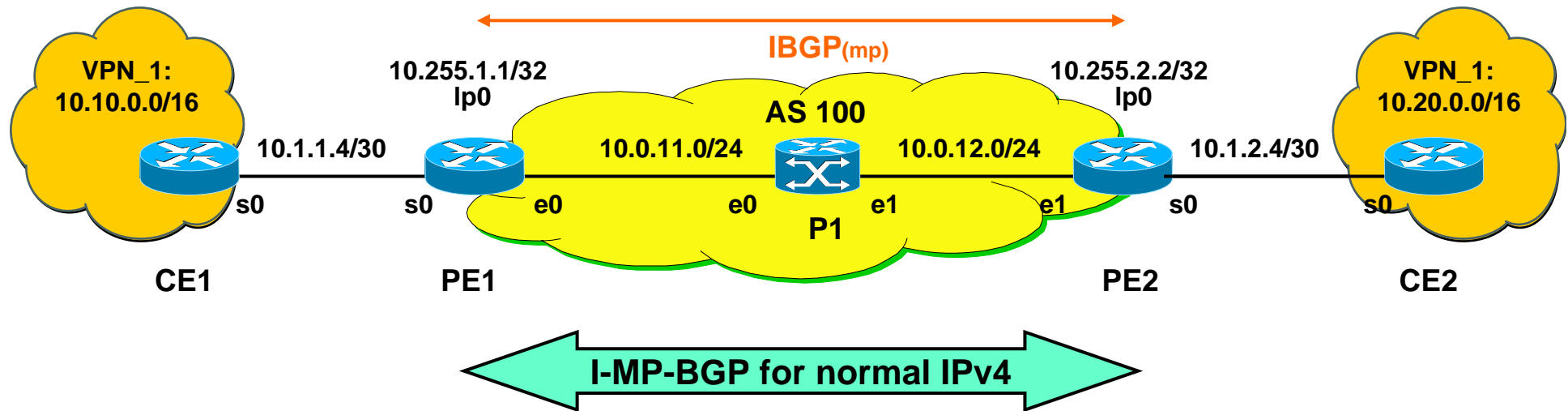
PE1:  
 ip cef  
 int loopback 0  
 ip address 10.255.1.1 255.255.255.255  
 int s0  
 ip address 10.1.1.6 255.255.255.252  
 int e0  
 ip address 10.0.11.1 255.255.255.0  
 mpls ip  
 router ospf 100  
 network 10.0.11.1 0.0.0.0 area 0

CE2:  
 int s0  
 ip address 10.1.2.5 255.255.255.252  
 router bgp 55  
 neighbor 10.1.2.6 remote-as 100  
 network 10.20.0.0 mask 255.255.0.0

PE2:  
 ip cef  
 int loopback 0  
 ip address 10.255.2.2 255.255.255.255  
 int s0  
 ip address 10.1.2.6 255.255.255.252  
 int e0  
 ip address 10.0.12.2 255.255.255.0  
 mpls ip  
 router ospf 100  
 network 10.0.12.1 0.0.0.0 area 0



# Start Normal I-BGP in AS 100



```

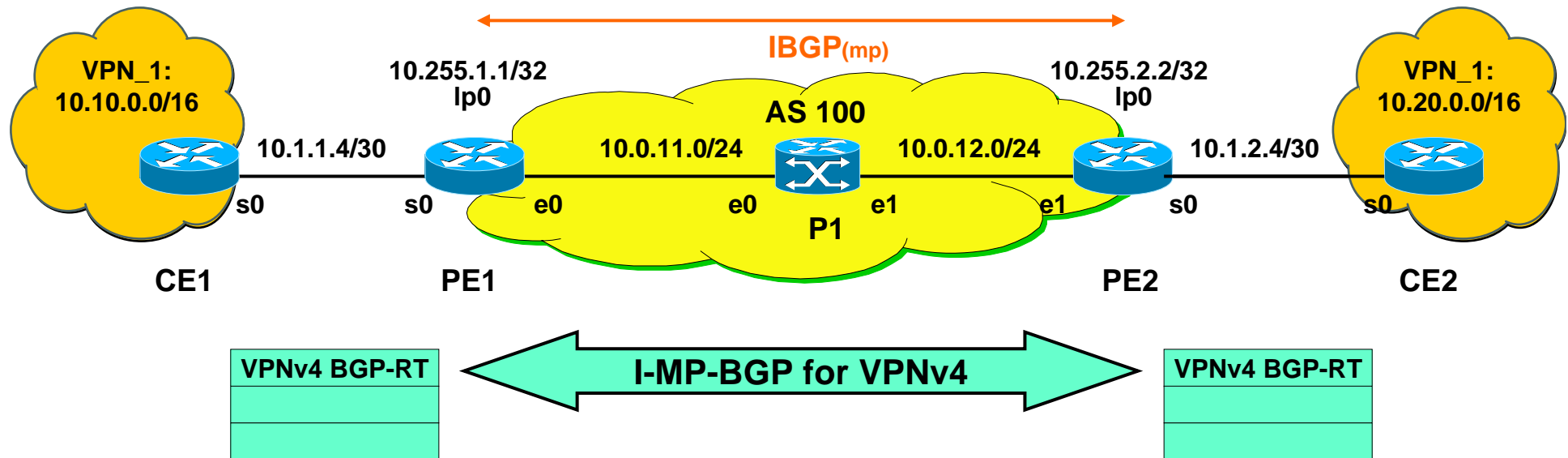
PE1:
int loopback 0
ip address 10.255.1.1 255.255.255.255

router bgp 100
  no bgp default ipv4-unicast
  bgp router-id 10.255.1.1
  neighbor 10.255.2.2 remote-as 100
  neighbor 10.255.2.2 update-source loop 0
  address-family ipv4
    neighbor 10.255.2.2 next-hop-self
    neighbor 10.255.2.2 activate
    no auto-summary (default)
    no synchronization (default)
  exit address-family
  
```

```

PE2:
int loopback 0
ip address 10.255.2.2 255.255.255.255
router bgp 100
  no bgp default ipv4-unicast
  bgp router-id 10.255.2.2
  neighbor 10.255.1.1 remote-as 100
  neighbor 10.255.1.1 update-source loop 0
  address-family ipv4
    neighbor 10.255.1.1 next-hop-self
    neighbor 10.255.1.1 activate
    no auto-summary (default)
    no synchronization (default)
  exit address-family
  
```

# Start MP-BGP in AS 100



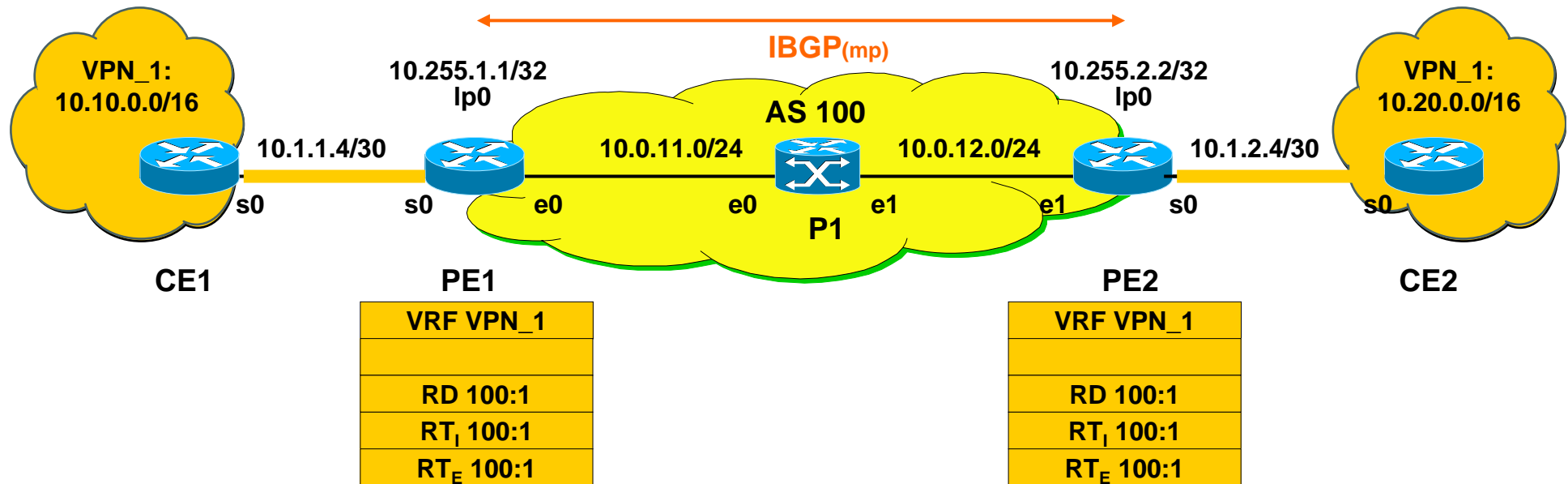
```

PE1:
int loopback 0
 ip address 10.255.1.1 255.255.255.255
router bgp 100
 no bgp default ipv4-unicast
 bgp router-id 10.255.1.1
 neighbor 10.255.2.2 remote-as 100
 neighbor 10.255.2.2 update-source loop 0
 address-family vpnv4
  neighbor 10.255.2.2 activate
  neighbor 10.255.2.2 next-hop-self
  neighbor 10.255.2.2 send-community extended (default)
 exit-address-family
  
```

```

PE2:
int loopback 0
 ip address 10.255.2.2 255.255.255.255
router bgp 100
 no bgp default ipv4-unicast
 bgp router-id 10.255.2.2
 neighbor 10.255.1.1 remote-as 100
 neighbor 10.255.1.1 update-source loop 0
 address-family vpnv4
  neighbor 10.255.1.1 activate
  neighbor 10.255.1.1 next-hop-self
  neighbor 10.255.1.1 send-community extended
 exit-address-family
  
```

# Create VRF and Bring Interface into VRF (PE router)



PE1:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

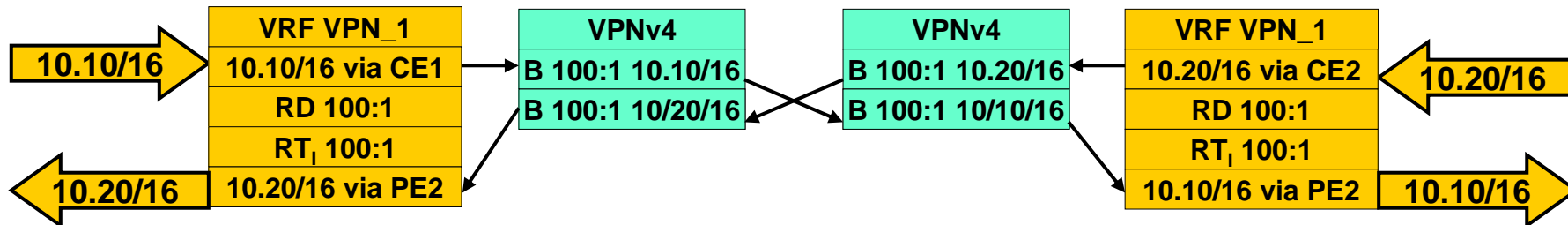
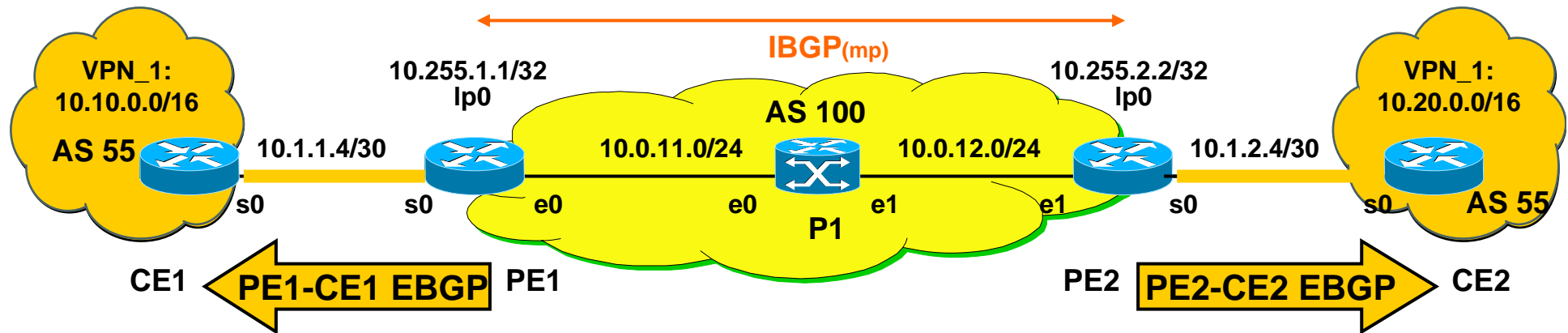
```
int s0
ip vrf forwarding VPN_1
ip address 10.1.1.6 255.255.255.252
```

PE2:

```
ip vrf VPN_1
rd 100:1
route-target import 100:1
route-target export 100:1
```

```
int s0
ip vrf forwarding VPN_1
ip address 10.1.2.6 255.255.255.252
```

# Start Dynamic Routing towards CE using EBGP Redistribute into MP-BGP and vice versa



PE1:

```
ip vrf VPN_1
rd 100:1
```

```
router bgp 100
address-family ipv4 vrf VPN_1
neighbor 10.1.1.5 remote-as 55
neighbor 10.1.1.5 activate
no auto-summary
no synchronization
exit-address-family
```

PE2:

```
ip vrf VPN_1
rd 100:1
```

```
router bgp 100
address-family ipv4 vrf VPN_1
neighbor 10.1.2.5 remote-as 55
neighbor 10.1.2.5 activate
no auto-summary
no synchronization
exit-address-family
```