

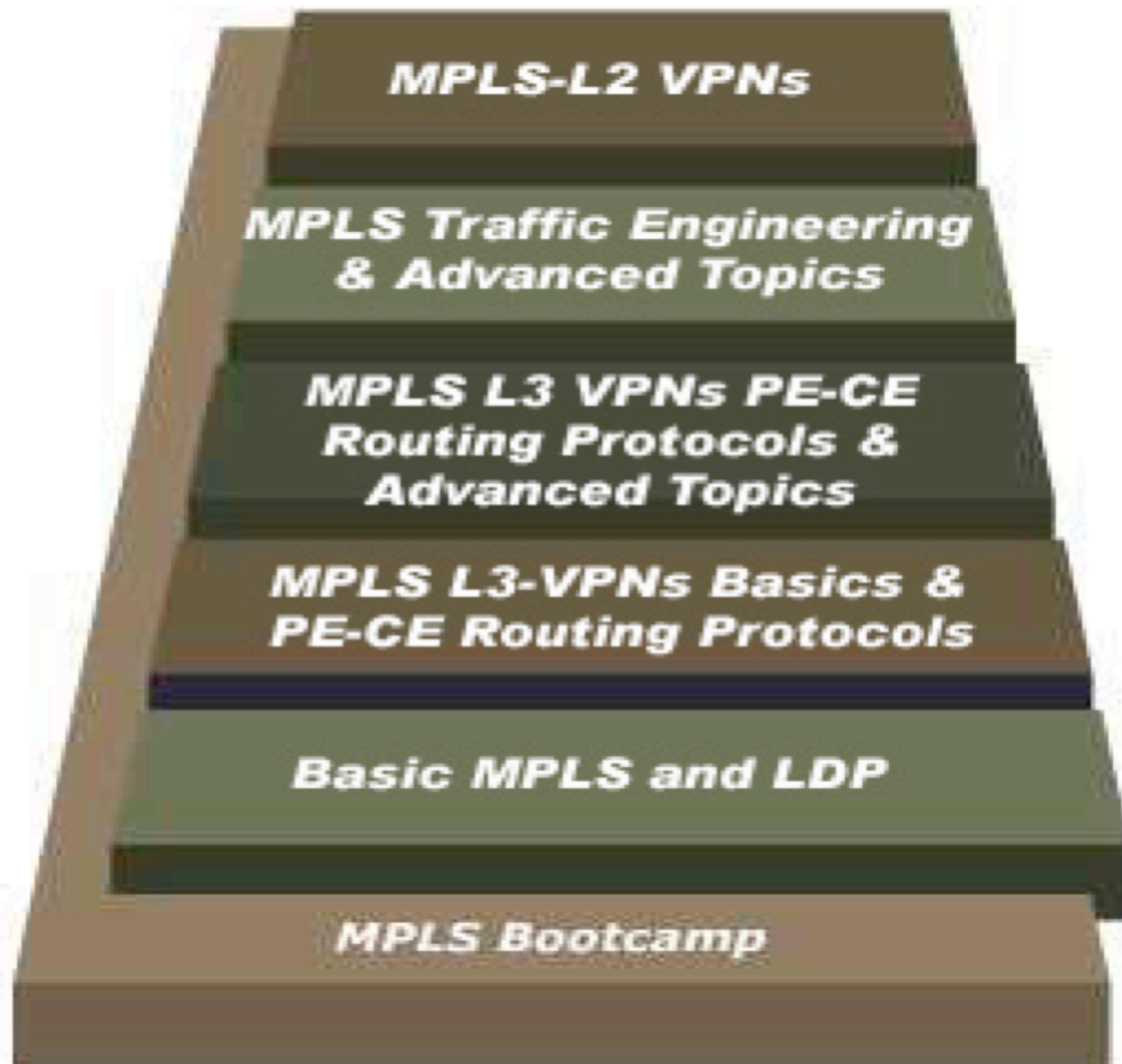


# **SANOG 14: MPLS Network Design and Deployment Workshop Agenda**

**Srini Irigi, SPG TME, Cisco Systems, CCIE 6147**

**Jonny Martin, Internet Analyst, PCH**

# Workshop Structure



# Day 1 Agenda

Day 1 Modules
Why MPLS is needed ???
How labels are advertised and stored
What protocols are used to distribute labels
Lab Overview & Initial Configuration Lab
LUNCH
LDP: LDP concepts, configuration and troubleshooting
MPLS Basics Configuration Lab

# Day 2 Agenda

Day 2 Modules
<b>Basic concepts of VPNs</b> <b>MPLS L3VPNs Basic concepts</b> <b>Route Distinguisher, VRF and Route-Target</b> <b>Why MP-BGP is used between PE routers</b> <b>L3VPN concepts and configuration</b>
<b>MPLS L3 VPNs Initial Configuration Lab</b>
LUNCH
<b>PE-CE routing protocols such as static routing</b> <b>Hub and Spoke L3VPN concepts and configuration</b> <b>BGP as a PE-CE routing protocol</b>
<b>MPLS L3 VPNs PE-CE Basics Configuration Lab</b>

# Day 4 Agenda

Day 4 Modules
<b>RIP as a PE-CE protocol</b> <b>Different ways of providing Inter-AS L3VPNs</b> <b>Scalability in Inter-AS L3VPNs</b>
<b>MPLS L3 VPNs PE-CE Configuration Lab</b>
LUNCH
<b>Multi-VRF (VRF-lite) CE</b> <b>Ways of providing Internet access with L3VPNs</b> <b>MPLS L3VPN troubleshooting</b>
<b>MPLS L3 VPNs Advanced Configuration Lab</b>

# Day 5 Agenda

Day 5 Modules
MPLS L2 VPNs
MPLS L2 VPNs Configuration Lab
LUNCH
MPLS Traffic Engineering
MPLS TE Configuration Lab



## MPLS: L2 VPNs

# Outline

- Layer 2 Transport vs. L2VPNs
- IETF standards for L2 Transport
- L2 transport example



# L2 Transport Why

- Quote from draft-ietf-pwe3-framework-xx.txt:

“ Although Internet traffic is the fastest growing traffic segment, it does not generate the highest revenue per bit. For example, Frame Relay traffic currently generates a higher revenue per bit than do native IP services. ”

# Standards: IETF Working Groups - PWE3

- Standards/Drafts:
  - Cisco's AToM:
    - draft-martini-l2circuit-trans-mpls-\*\*.txt
    - draft-martini-l2circuit-encap-mpls-\*\*.txt
  - Cisco's L2TPv3:
    - draft-ietf-l2tpext-l2tp-base-\*\*.txt

# L2VPN

- Traditional L2VPNs are built with leased lines, virtual circuits such as ATM PVCs or FR DLCIs
- L2VPN can now be built using L2 transport mechanisms standardized by IETF's PWE3 working group (aka AToM or L2TPv3)
- Similar to L3VPN service except that packet forwarding is based on L2 information rather than L3
- L2 VPN is a service model for interconnecting multiple customers sites using L2 circuits or L2 transports, taking into consideration factors such as management, QoS, security, provisioning, etc.

# Standards: IETF Working Groups - PPVPN

- L2VPNs are standardized by IETF's PPVPN working group
- PPVPN: Provider Provisioned Virtual Private Network
  - Implementation & scalability aspects of VPNs
  - Standards/Drafts:
    - L3VPNs (RFC2547bis)
    - L2VPNs leveraging the L2 transport work from PWE3
      - draft-rosen-ppvpn-l2vpn-\*.txt (VPWS)
      - draft-sajassi-vpls-architectures-\*.txt (VPLS)
      - draft-lasserre-vkompella-ppvpn-vpls-\*.txt(VPLS)

# Why Is L2VPN Needed

- It allows SP to have a single infrastructure for both IP and legacy services

Move legacy ATM/FR traffic to MPLS/IP core without interrupting current services

- ISP provide new P2P Layer 2 tunneling services

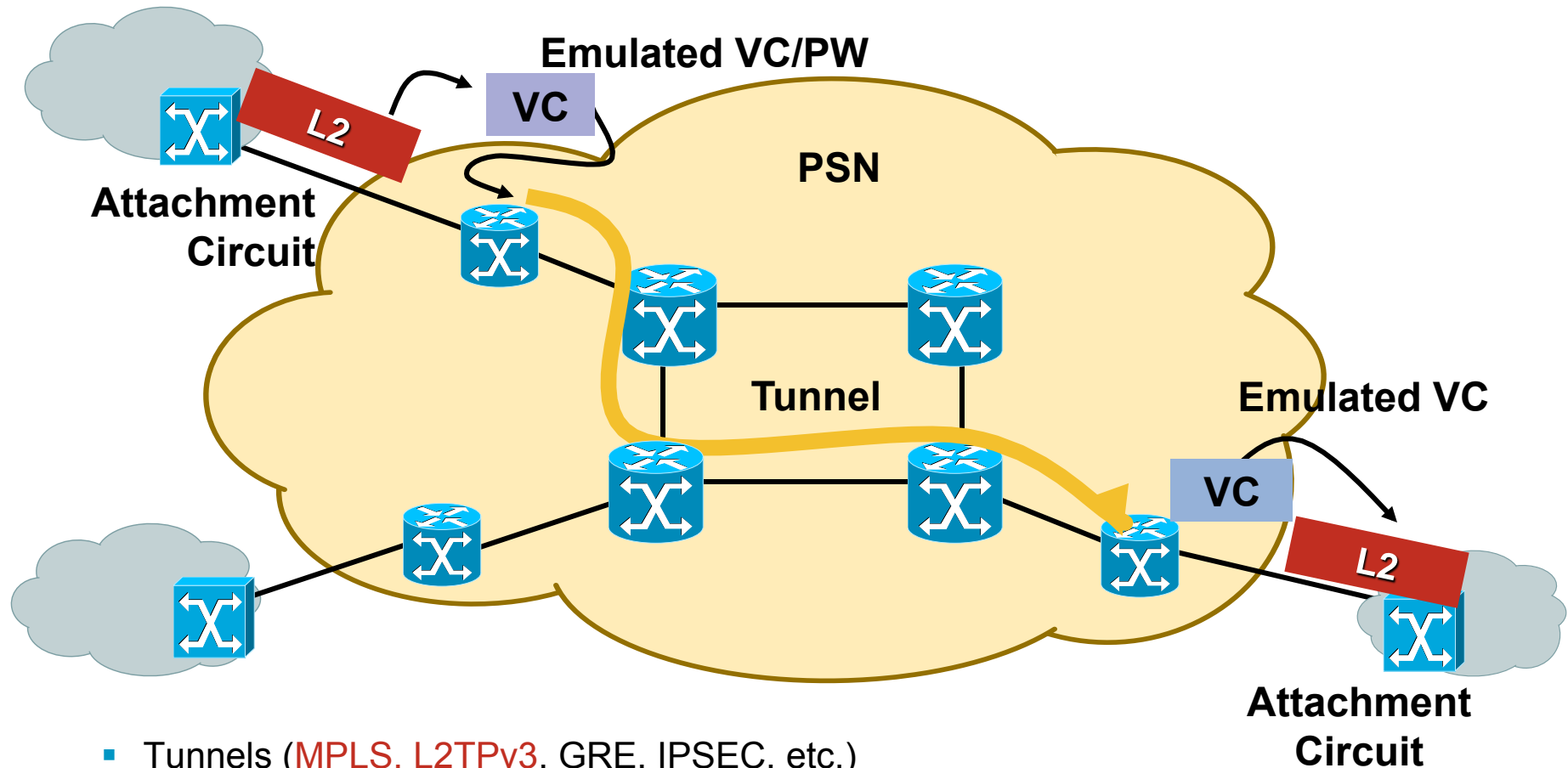
Customer can have its own routing, QOS policy, etc.

- A migration step towards IP/MPLS VPN

# Benefits for L2VPNs

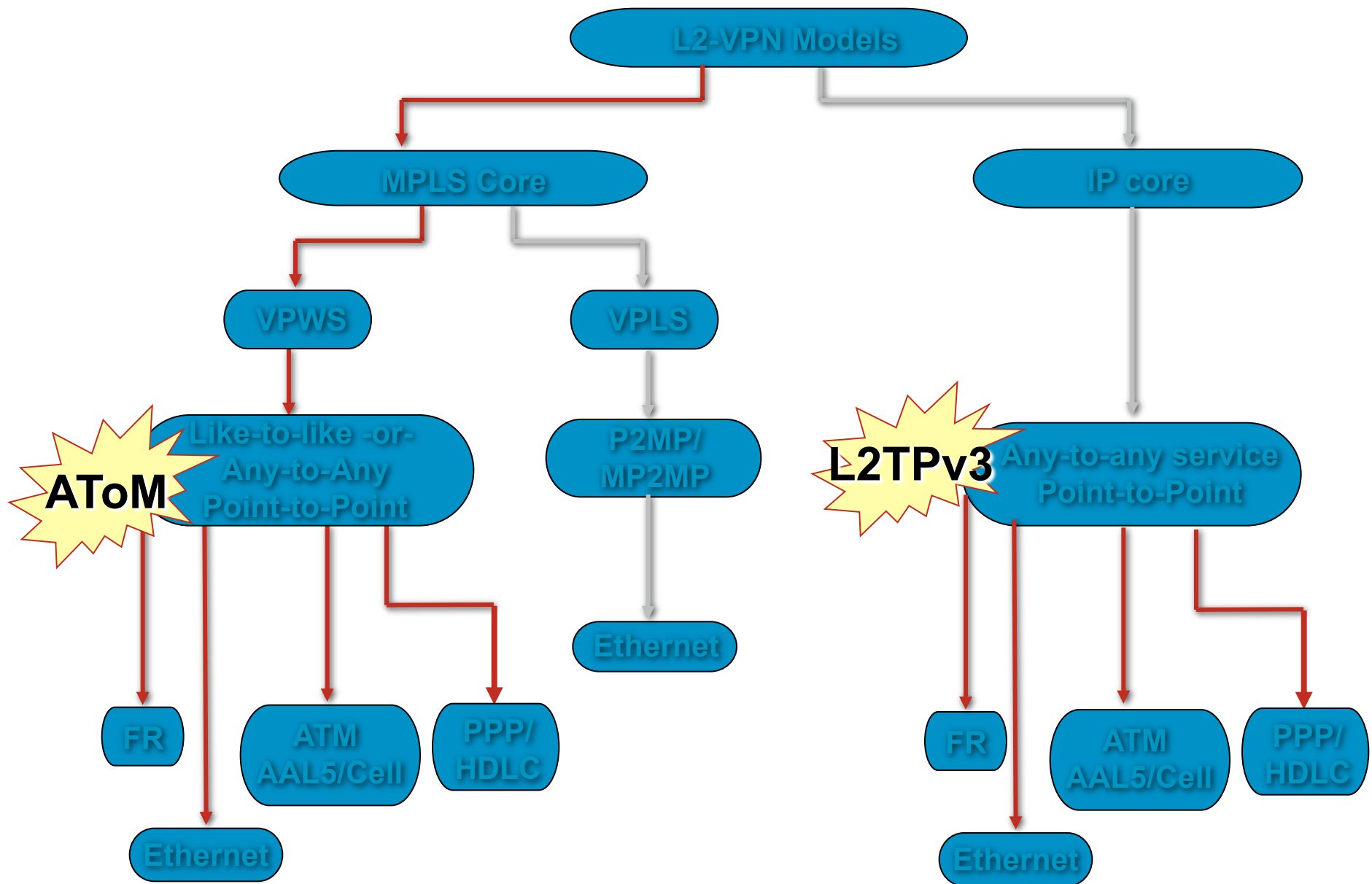
- New service opportunities:
  - Virtual leased line service
  - Offer “PVC-like” Layer 2-based service
- Reduced cost—consolidate multiple core technologies into a single packet-based network infrastructure
- Simplify services—Layer 2 transport provides options for Service Providers who need to provide L2 connectivity and maintain customer autonomy
- Protect existing investments—Greenfield networks to extend customer access to existing Layer 2 networks without deploying a new separate infrastructure
- Feature support—through the use of Cisco IOS features such as IPsec, QoS and Traffic Engineering, L2 transport can be tailored to meet customer requirements

# Generic L2VPN Architecture



- Tunnels (**MPLS**, **L2TPv3**, GRE, IPSEC, etc.)
- Emulated VCs (pseudowires) inside tunnels (many-to-one)
- Attachment VCs (e.g. FR DLCI, PPP) mapped to emulated VCs

# L2-VPN Models





## L2Transport vs. L2VPN

- L2 Transport and L2VPNs are tightly coupled so why differentiate???
- Cisco's AToM(draft-martini) is a L2 transport
- Juniper's draft-kompella is a ppvpn draft and presents a solution for L2VPNs and **not** L2 transport

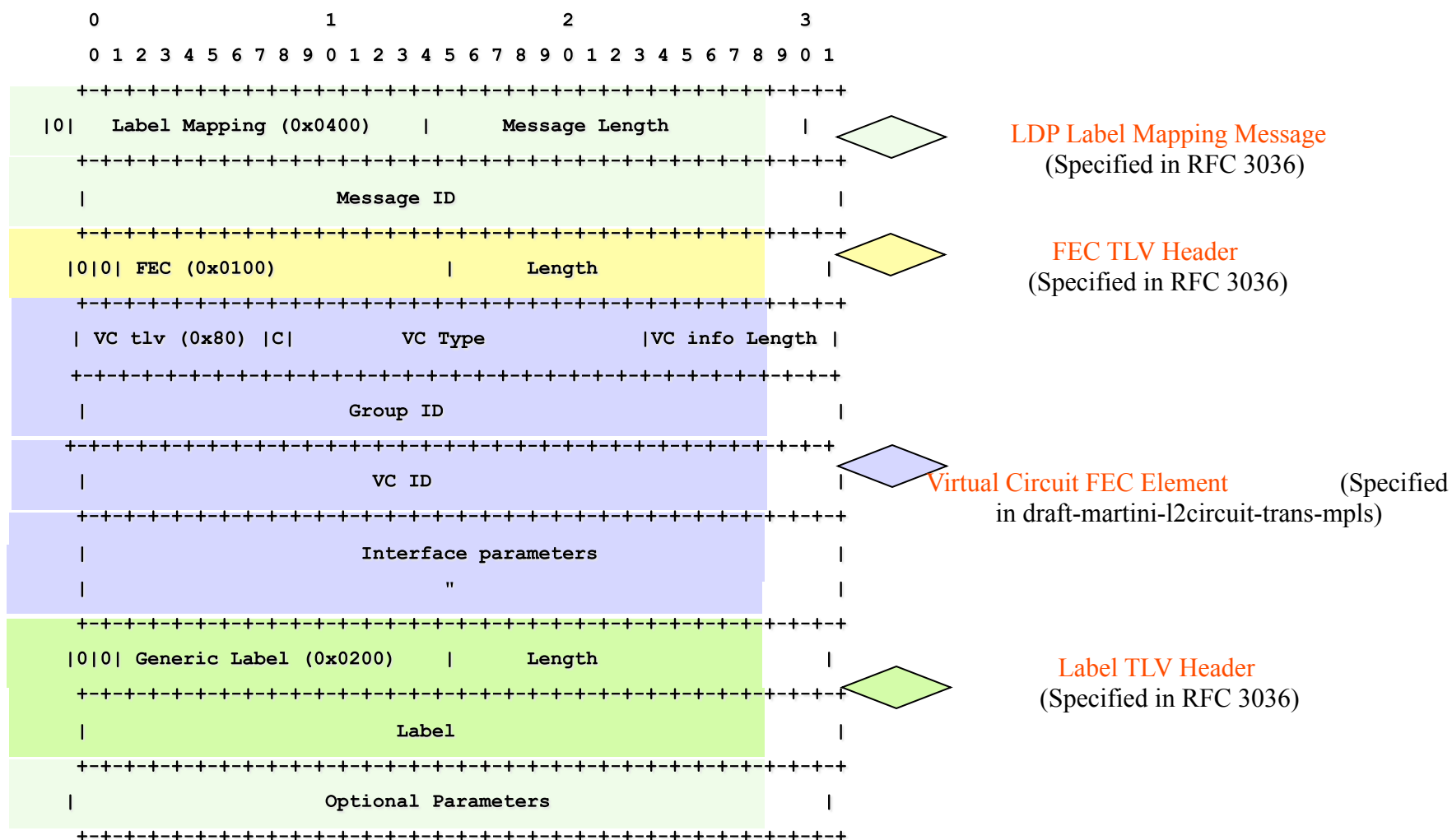
# AToM Protocol

- Protocols necessary to implement the Emulated service can be categorized as..
  - Control Plane Functions (Signaling)
    - Emulated VC signaling → *LDP draft-martini-l2circuit-trans-mpls*
    - MPLS Tunnel signaling → *LDP/TDP(LSP) or RSVP(TE)*
  - Data Plane Functions (Encapsulation)
    - Attachment VC termination → *draft-martini-l2circuit-encap-mpls*
    - Emulated VC termination → *draft-martini-l2circuit-encap-mpls*
    - Emulated VC tunneling → *draft-martini-l2circuit-encap-mpls*

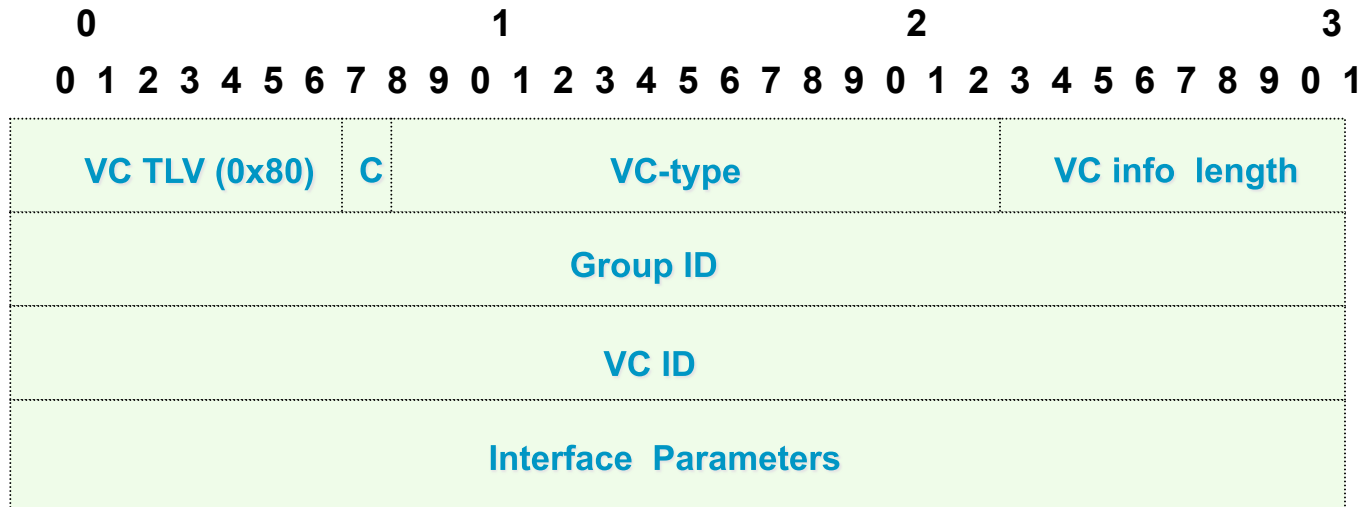
# AToM: Control Plane (Signaling)

- Need for Emulated VC and Tunnel Signaling:
  - AToM/L2 transport is implemented using two level label switching between the PEs (similar to RF2547/L3VPNs)
  - Distribution of Tunnel Labels (LDP or TDP) for Tunnel setup
  - Distribution of VC Labels (**LDP only**) for Emulated VC setup
- Emulated VC signaling **must** be done via LDP
  - Directed LDP session between PEs
  - Existing Label mapping messages used
  - New VC FEC element =128 created for distributing VC labels
- Tunnel Signaling outside the scope of draft-martini-l2circuit-trans-mpls

# AToM: LDP Label Mapping Exchange



# AToM: Virtual Circuit FEC Element



**C: Control Word** (1 bit) – Control word present if bit set

**VC-type** (15 bits) - Type of VC e.g FR, ATM, VLAN, Ethernet, PPP, HDLC

**VC info length** (8 bits) – Length of VCID field and interface parameters

**Group ID** (32 bits) – Represents a groups of VCs. Can be used for mass label withdrawal

**VC ID** (32 bits) – Connection identifier used in conjunction with the VC-type to identify a particular VC

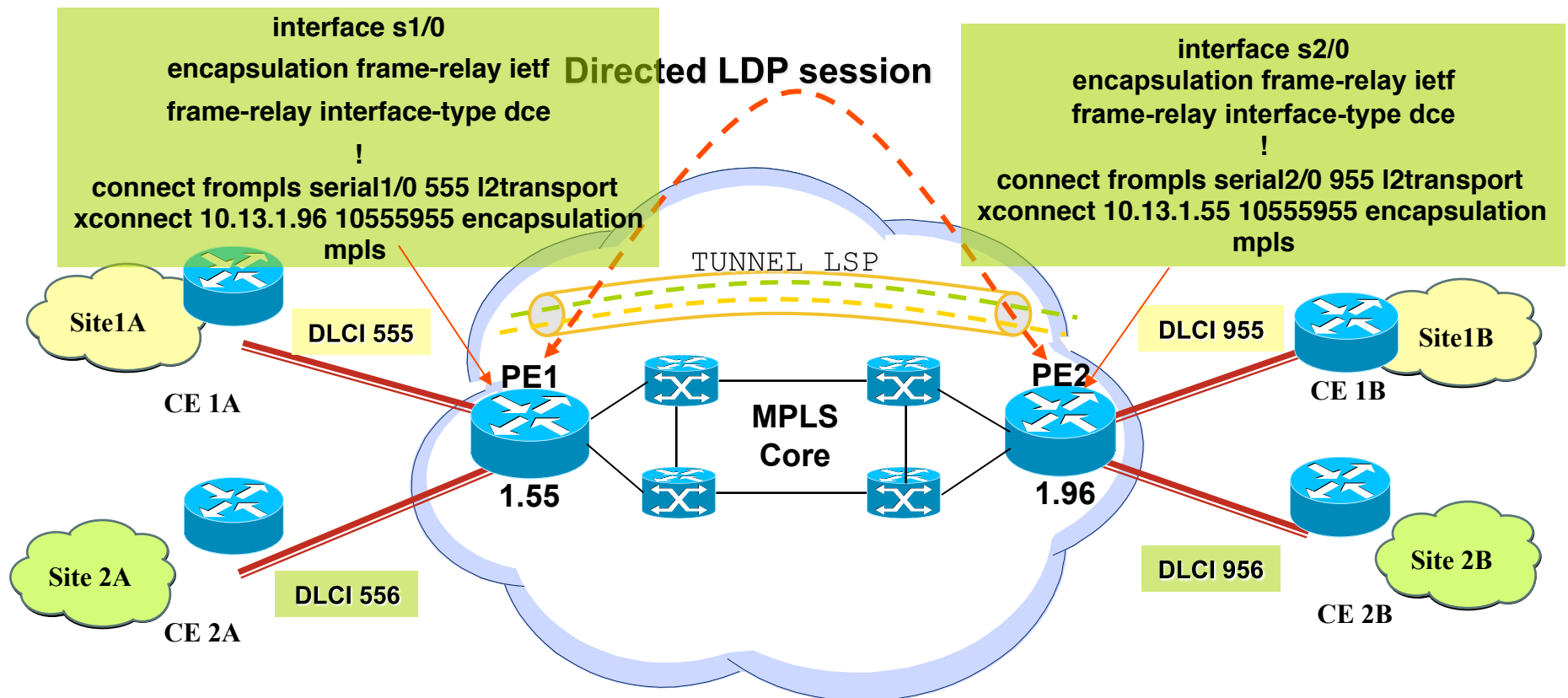
**Interface Parameters** (Variable) – Edge facing interface parameters, such as MTU

# Some Currently Defined VC-Types

	<u>PW type</u>	<u>Description</u>	
	0x0001	Frame Relay DLCI	! Frame Relay
	0x0002	ATM AAL5 SDU VCC transport	! ATM AAL5 SDU
0x0003		ATM transparent cell transport	! ATM Cell Port Mode
	0x0004	Ethernet Tagged Mode	! Ethernet VLAN
	0x0005	Ethernet	! Ethernet
	0x0006	HDLC	! HDLC
	0x0007	PPP	! PPP
0x0008		SONET/SDH Circuit Emulation Service Over MPLS (CEM) [Note1]	
	0x0009	ATM n-to-one VCC cell transport	! ATM Cell VC Mode
	0x000A	ATM n-to-one VPC cell transport	! ATM Cell VP Mode
	0x000B	IP Layer2 Transport	! Interworking IP
	0x000C	ATM one-to-one VCC Cell Mode	
	0x000D	ATM one-to-one VPC Cell Mode	
	0x000E	ATM AAL5 PDU VCC transport	
	0x000F	Frame-Relay Port mode	
	0x0010	SONET/SDH Circuit Emulation over Packet (CEP)	
	0x0011	Structure-agnostic E1 over Packet (SAToP)	
	0x0012	Structure-agnostic T1 (DS1) over Packet (SAToP)	
	0x0013	Structure-agnostic E3 over Packet (SAToP)	
	0x0014	Structure-agnostic T3 (DS3) over Packet (SAToP)	
	0x0015	CESoPSN basic mode	
	0x0016	TDMoIP basic mode	
	0x0017	CESoPSN TDM with CAS	
	0x0018	TDMoIP TDM with CAS	

**Note 1: This PW Type Is Grandfathered for a Historical Protocol; the Recommended Standards-Track Protocol to Use Is CEP (PW Type 0x0010)**

# AToM: Control Plane Example



**Step1: “xconnect 10.13.1.96 10555955 encapsulation mpls” added to PE1 with FRoMPLS enabled for serial1/0**

**Step2: Directed LDP session setup with 10.13.1.96 to exchange VC labels**

# AToM: Discovery Phase

```
PE1#sh mpls ldp discovery detail
```

```
Local LDP Identifier:
```

```
10.13.1.55:0
```

```
Discovery Sources:
```

```
Interfaces:
```

```
POS11/0/0 (tdp): xmit/recv
```

```
TDP Id: 10.13.1.58:0
```

```
Src IP addr: 10.13.5.41; Transport IP addr: 10.13.1.58
```

```
FastEthernet10/0/0.441 (tdp): xmit/recv
```

```
TDP Id: 10.13.1.59:0
```

```
Src IP addr: 10.13.5.65; Transport IP addr: 10.13.1.59
```

```
FastEthernet10/0/1.432 (tdp): xmit/recv
```

```
TDP Id: 10.13.1.58:0
```

```
Src IP addr: 10.13.5.61; Transport IP addr: 10.13.1.58
```

```
Targeted Hellos:
```

```
10.13.1.55 -> 10.13.1.96 (ldp): active/passive, xmit/recv
```

```
LDP Id: 10.13.1.96:0
```

```
Src IP addr: 10.13.1.96; Transport IP addr:
```

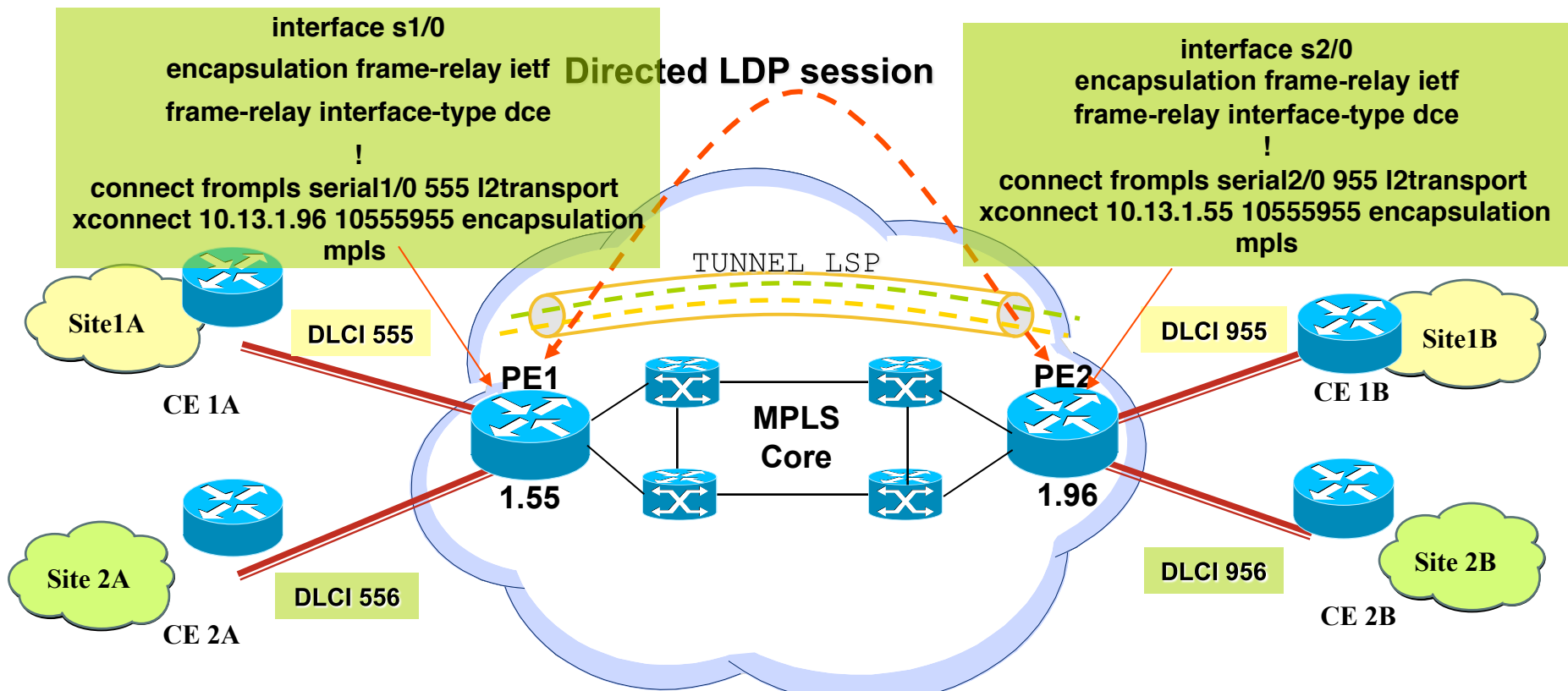
```
10.13.1.96
```



# AToM: Targeted LDP session

```
PE1#sh mpls ldp neighbor 10.13.1.96
  Peer LDP Ident: 10.13.1.96:0; Local LDP Ident 10.13.1.55:0
    TCP connection: 10.13.1.96.11014 - 10.13.1.55.646
    State: Oper; Msgs sent/rcvd: 2773/2779; Downstream
    Up time: 1d10h
    LDP discovery sources:
      Targeted Hello 10.13.1.55 -> 10.13.1.96, active,
      passive
    Addresses bound to peer LDP Ident:
      10.13.1.96      10.13.9.30      10.13.9.46
10.13.0.96
      10.13.9.66
PE1#
```

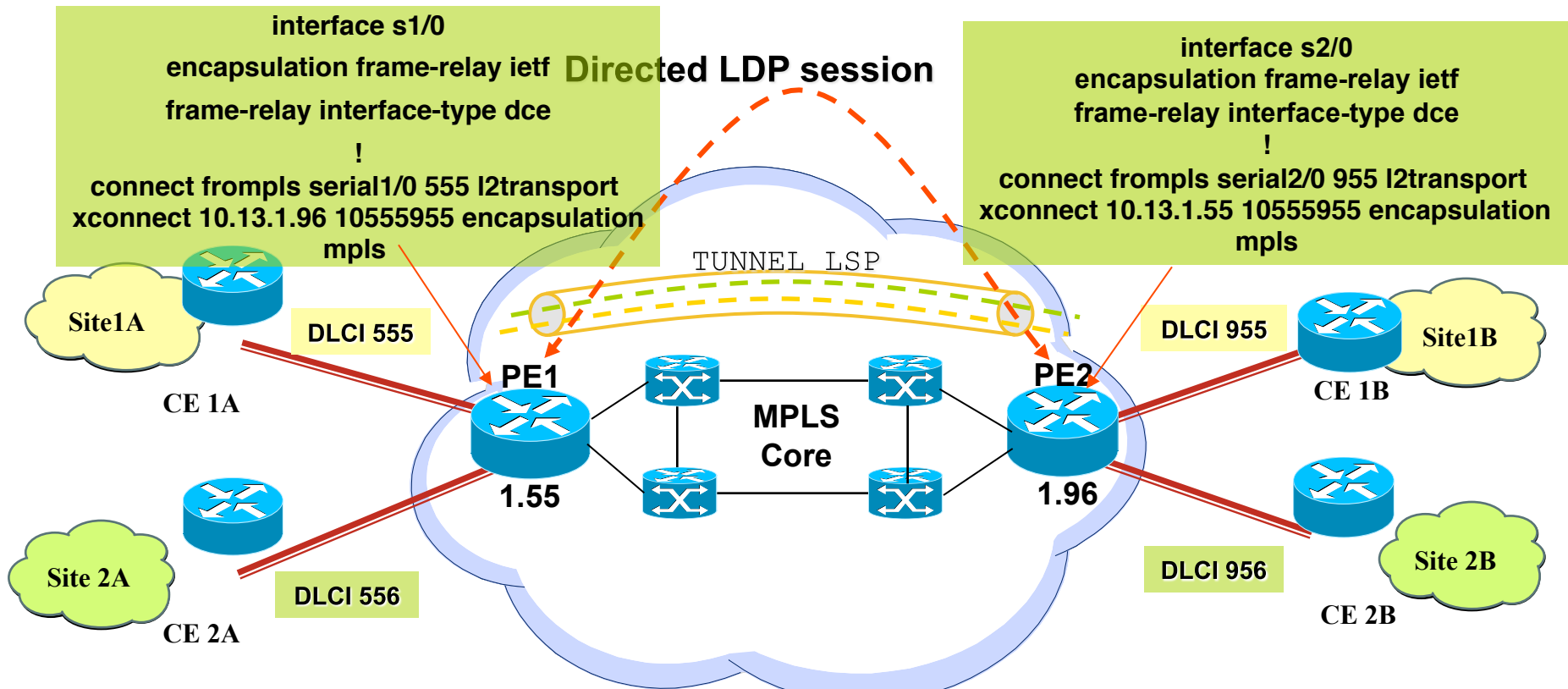
# AToM: Control Plane Example



**Step 3A: PE-CE interface on PE1 is “no shut”...**

1. PE1 will allocate a VC label for DLCI 555
2. binds it to VC ID: 10555955
3. encodes the VC Label TLV with the VC label
4. encodes the VC FEC TLV with the VC ID
5. advertises the label to 10.13.1.96

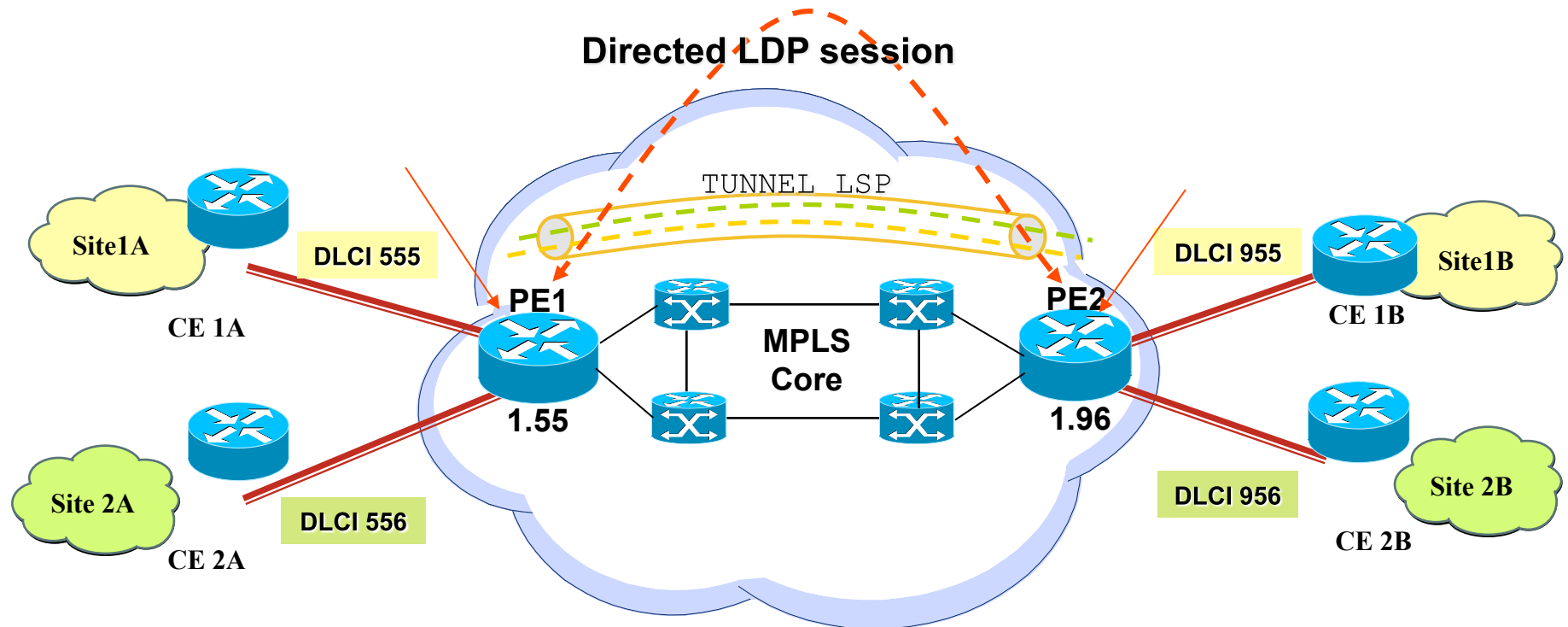
# AToM: Control Plane Example



**Step 3B: PE-CE interface on PE2 is “no shut”...**

1. PE2 will allocate a VC label for DLCI 955
2. binds it to VC ID: 10555955
3. encodes the VC Label TLV with the VC label
4. encodes the VC FEC TLV with the VC ID
5. advertises the label to 10.13.1.55

# AToM: Label Withdrawal



**PE-CE interface on PE1 is 'shut'...**

- PE1 will send a Label Withdrawal message to 10.13.1.96
- status of the VC is down

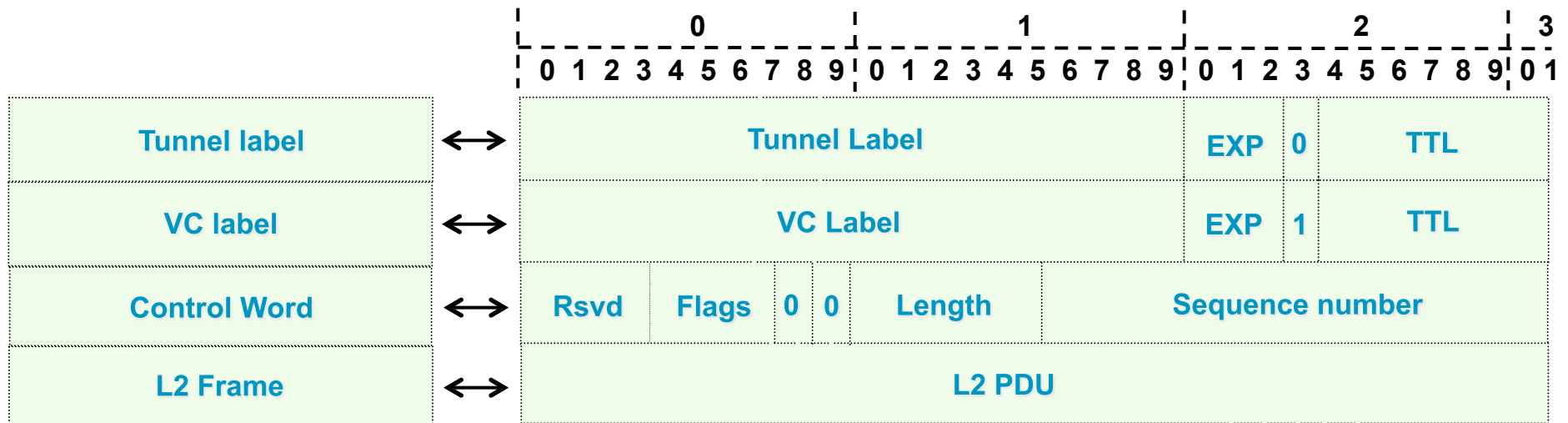
**PE-CE interface on PE2 is 'shut'...**

- PE2 will send a Label Withdrawal message to 10.13.1.55
- status of the VC is already down

## Why LDP signaling is useful between PEs

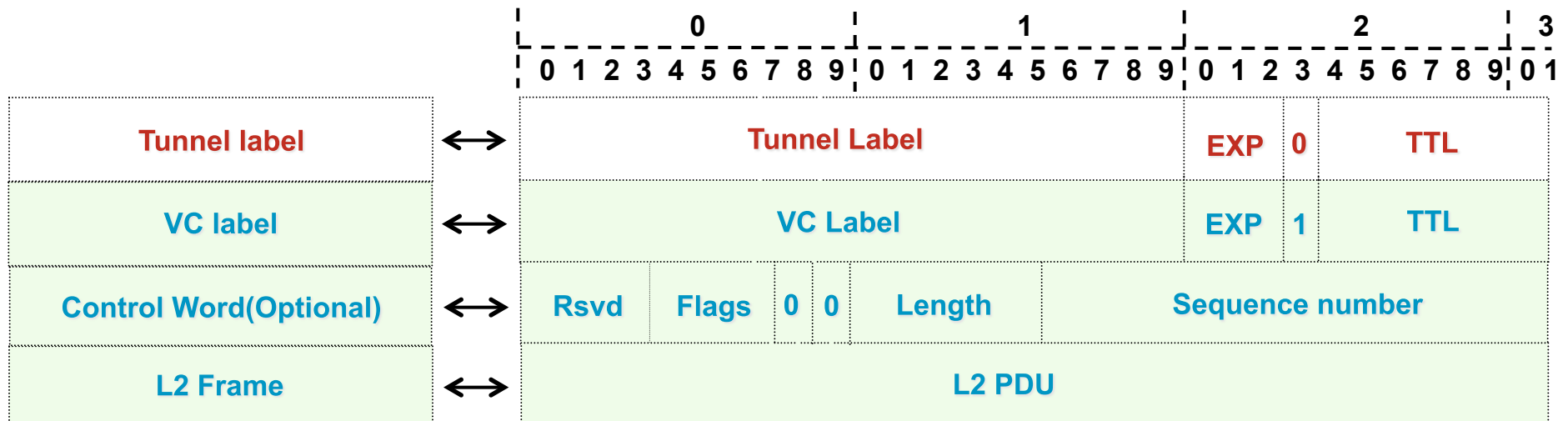
- To transport circuit status
  - eg. FR: If PE1 sees an issue with dlci 555, it withdraws the VC label so that PE2 can signal the issue on the right via LMI
  - useful for FR, ATM, HDLC, Ethernet...
- In-Sequence delivery
  - Required for ATM and FR. If Ethernet used for non-IP applications, in-sequence delivery is also required
- PE1 and PE2 can use LDP to synch their sequence numbers after reload/reboot...
- Explicit Goal for PEW3 IETF WG

# AToM: Data Plane (Martini Encapsulation)



# AToM: Data Plane (Martini Encapsulation)

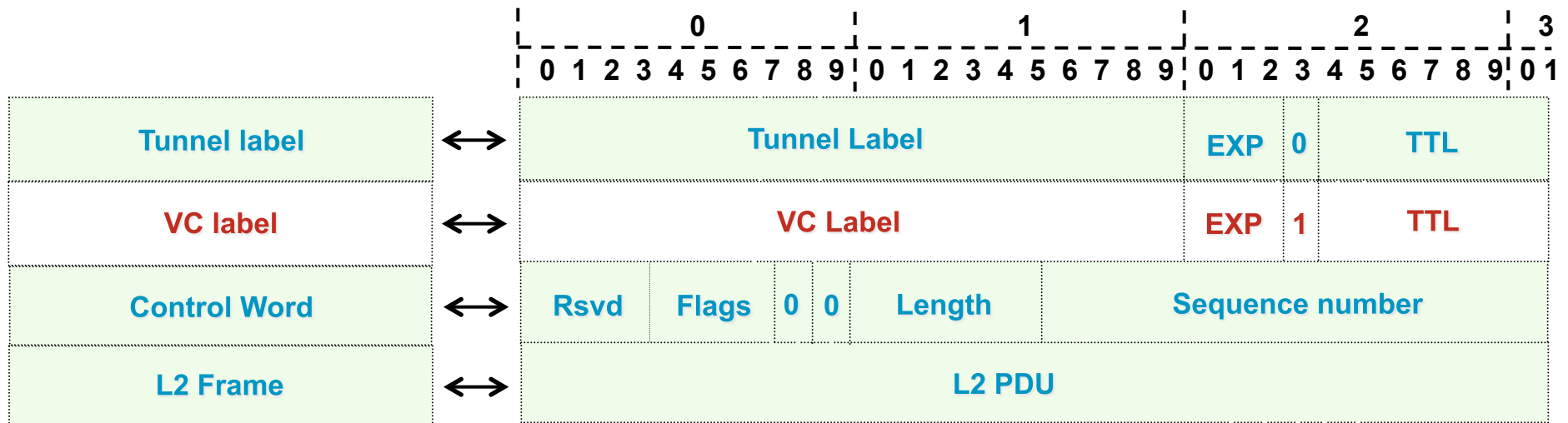
## Tunnel Label



### Tunnel Label:

- IGP or Outer label that can be distributed by any of the existing mechanisms and is outside the scope of martini draft
- label associated with the tunnel i.e. MPLS LSP or RSVP-TE used to deliver the packet from the ingress PE to egress PE

# AToM: Data Plane (Martini Encapsulation) VC Label



## VC Label:

- Inner label that is used by receiving PE to determine the following information and do disposition on the received packet...
  - egress or CE facing interface that the packet should be forwarded to
  - L2 ID such as VLAN or DLCI or PVC used on the CE facing interface

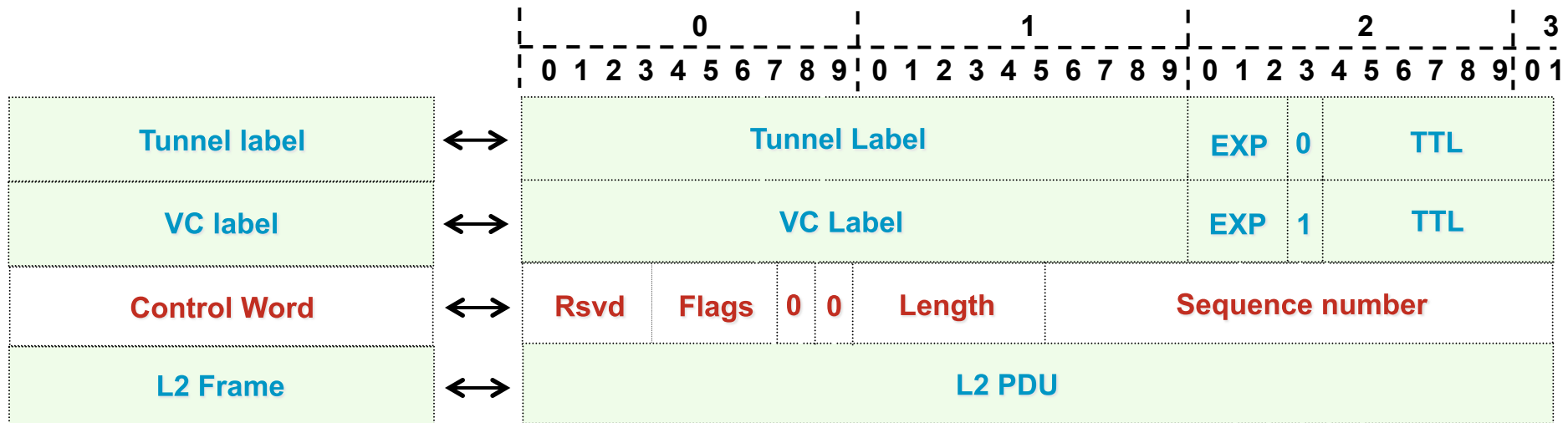
**EXP** can be set to the values received in the L2 frame, ATM CLP or FR DE bit or it can be set by the PE via CLI or as a result of some QoS policy

**TTL** is recommended to be set to '2'



# AToM: Data Plane (Martini Encapsulation)

## Control Word



### Control Word (CW):

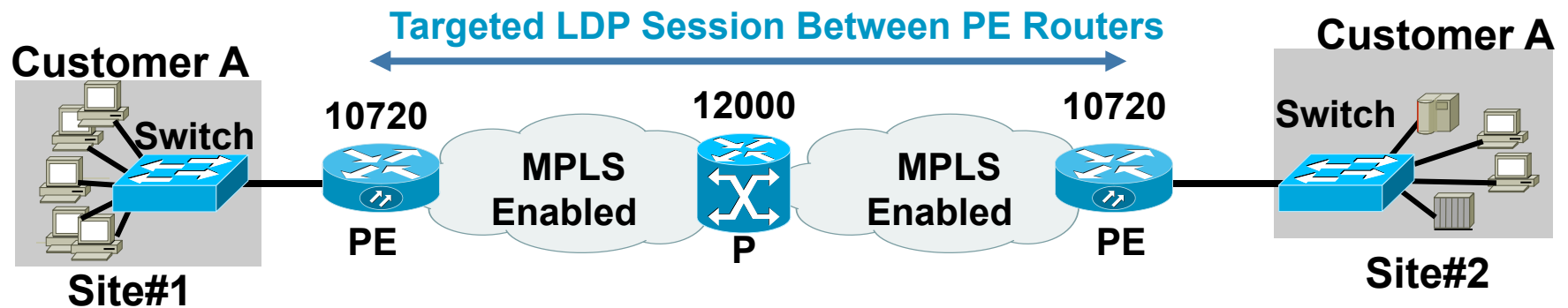
- Optional or Mandatory depending on the type of L2 transport
  - **Rsvd**: Reserved for future use
- **Flags**: to carry control bits (ATM CLP, FR DE) in the recvd. L2 frame across the MPLS network
- **Length**: used to indicate the actual packet length if any padding was done to the packet
  - **Sequence number**:
    - provides sequencing capability to detect out of order packets if needed
    - currently not in Cisco's implementation
    - Optional



# MPLS L2VPNs: EoMPLS

# EoMPLS Reference Model

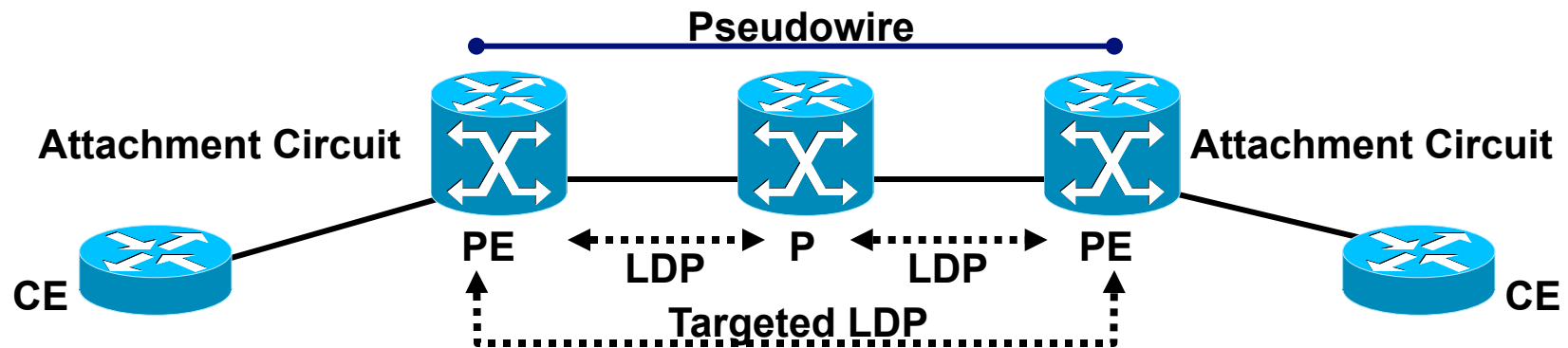
## Physical Connectivity



## Logical Connectivity

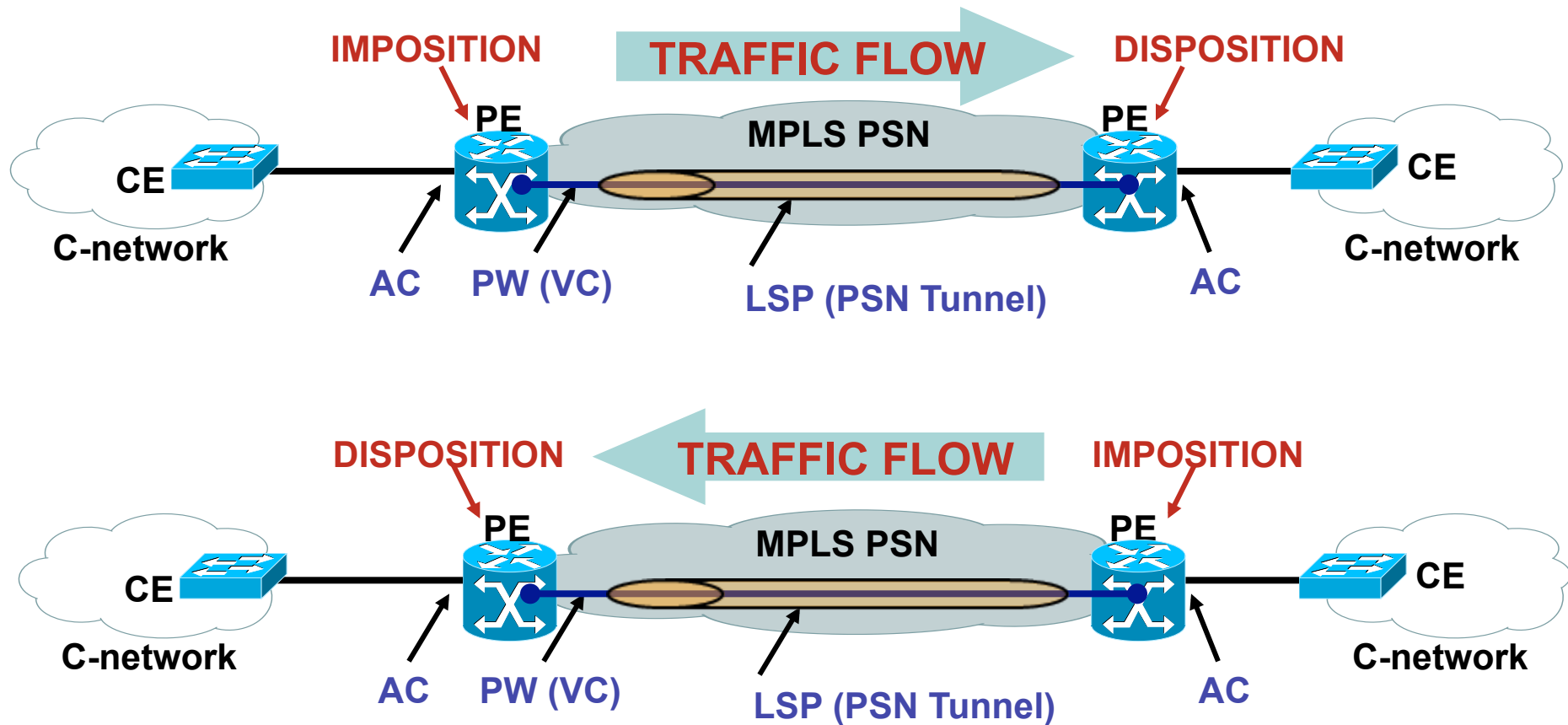


# Control Plane



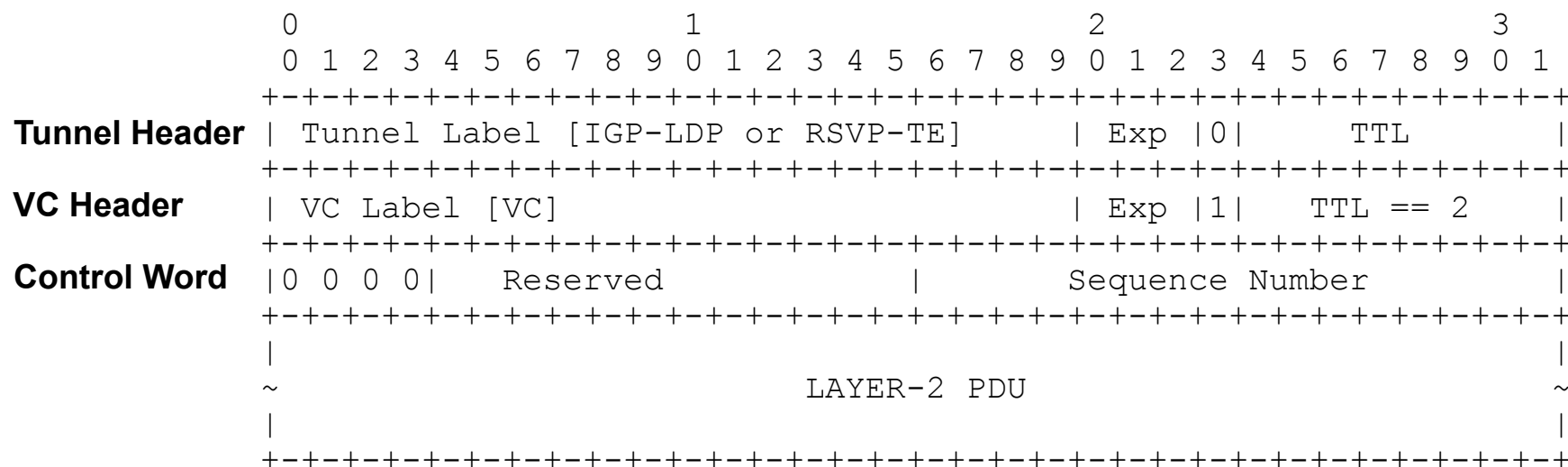
- MPLS in the core
- Targeted (AKA directed) LDP session between PEs
- Targeted LDP session distributes pseudowire (Pw AKA VC) labels
- PE uses per-platform label space (label pool) for both link and targeted LDP sessions (i.e. router\_id:0)
- Need LSPs among PEs => Use /32 loopback prefixes

# Imposition and Disposition



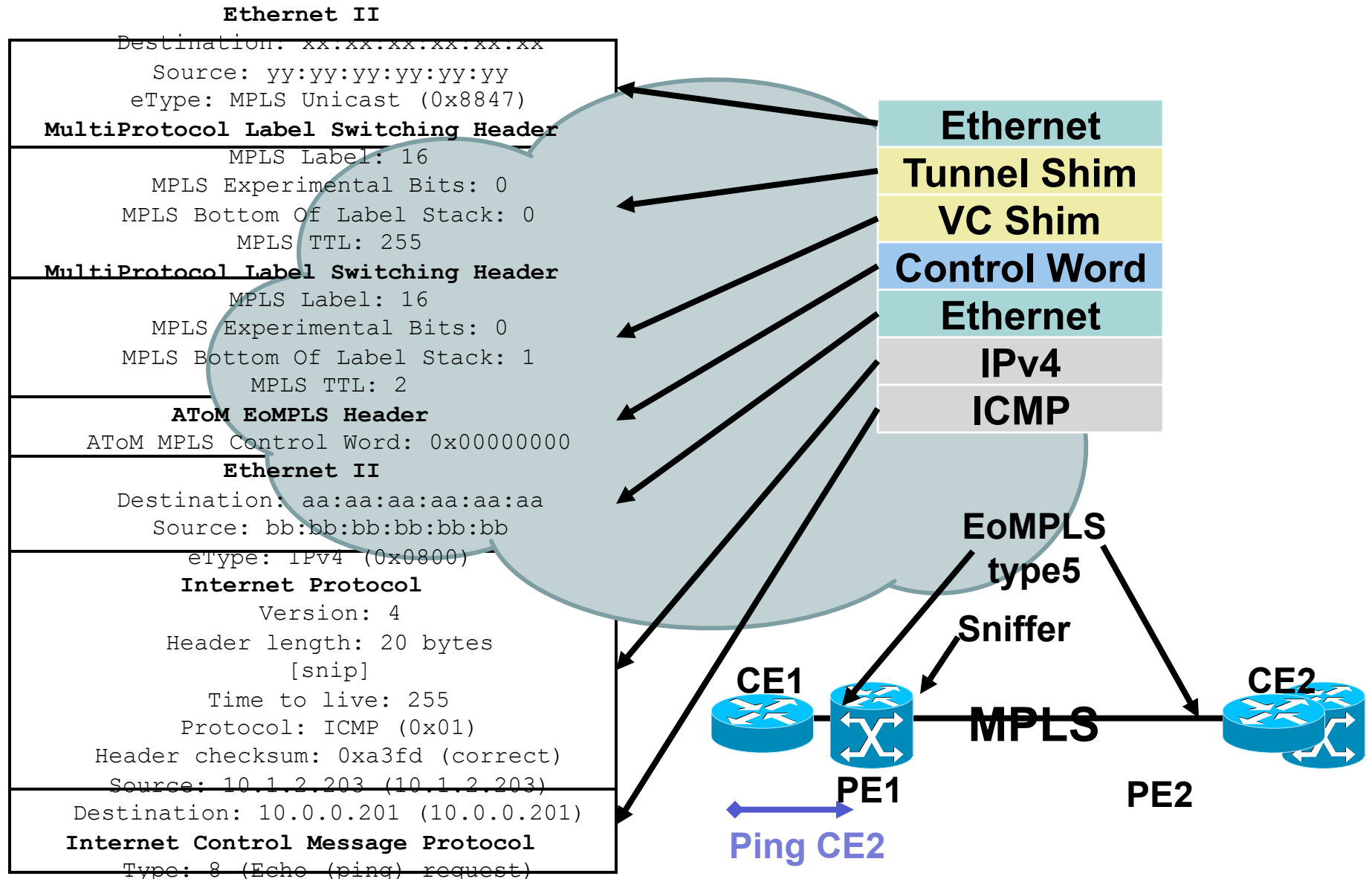
- AToM needs MPLS LSP from PE to PE
- PEs need to use /32 loopback prefixes for LSPs

# Data Plane: EoMPLS Packet Format



- The control word (AToM Header) is optional for Ethernet, PPP, HDLC, and cell relay transport types; however, the control word is required for Frame Relay, and ATM AAL5 transport types
- First nibble is 0x0 to prevent aliasing with IP Packets over MPLS
- The AToM control word is supported; however, if a peer PE does not support the control word, it is disabled; this negotiation is done by LDP label mapping

# Data Plane: A Real Packet



# AToM Prerequisite Configuration

- Enable [d]CEF globally

Router(config)#ip cef

- Enable MPLS globally

Router(config)#mpls ip

- Enable LDP globally as default label distribution protocol

Router(config)#mpls label protocol ldp

- Specify a loopback interface as LDP Router ID

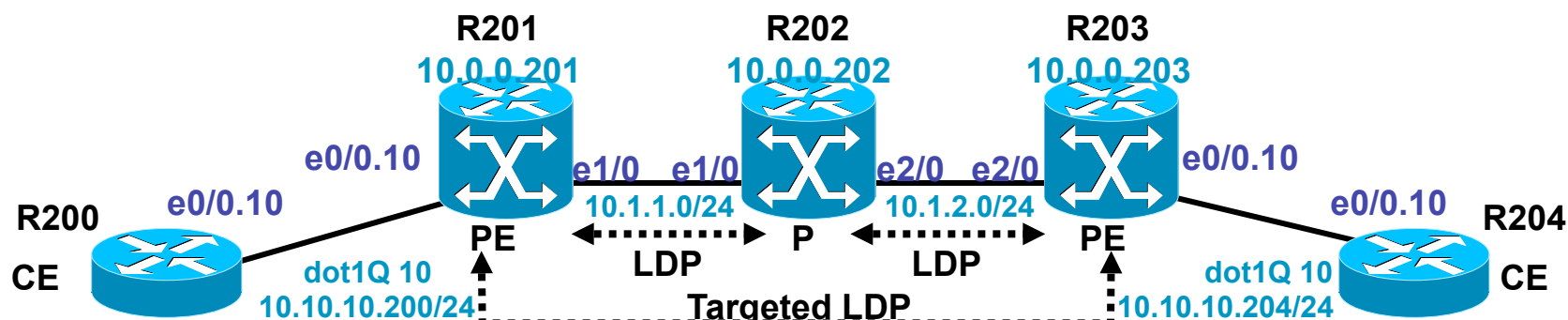
Router(config)#mpls ldp router-id loopback <#> [force]

- Enable LDP in the P-PE and P-P interfaces

Router(config-if)#mpls ip



# A Typical Configuration: EoMPLS VLAN



```
hostname R201
!
ip cef
mpls ip
mpls label protocol ldp
mpls ldp router-id Loopback0 force
!
interface Loopback0
ip address 10.0.0.201 255.255.255.255
```

```
!
interface Ethernet0/0.10
description *** To R200 ***
encapsulation dot1Q 10
no ip directed-broadcast
no cdp enable
```

```
xconnect 10.0.0.203 10 encapsulation mpls
```

```
hostname R203
!
ip cef
mpls ip
mpls label protocol ldp
mpls ldp router-id Loopback0 force
!
interface Loopback0
ip address 10.0.0.203 255.255.255.255
```

```
!
pseudowire-class eompls
encapsulation mpls
```

```
!
interface Ethernet0/0.10
description *** To R204 ***
encapsulation dot1Q 10
no ip directed-broadcast
```

```
no cdp enable
xconnect 10.0.0.201 10 pw-class eompls
```

# Verifying the Operation

## Working Example

```
R201#show mpls l2transport vc 10 detail
Local interface: Et0/0.10 up, line protocol up, Eth VLAN 10 up
Destination address: 10.0.0.203, VC ID: 10, VC status: up
Preferred path: not configured
Default path: active
Tunnel label: 17, next hop 10.1.1.202
Output interface: Et1/0, imposed label stack {17 21}
Create time: 23:06:37, last status change time: 00:30:47
Signaling protocol: LDP, peer 10.0.0.203:0 up
MPLS VC labels: local 19, remote 21
Group ID: local 0, remote 0
MTU: local 1500, remote 1500
Remote interface description: *** To R204 ***
Sequencing: receive disabled, send disabled
VC statistics:
  packet totals: receive 1683, send 1777
  byte totals:  receive 565455, send 563328
  packet drops:  receive 0, send 7
```

# VC Status Meaning

- UP—VC can carry data between the 2 endpoints; this means both imposition and disposition are programmed

The disposition interfaces is programmed if the VC has been configured and the CE interface is up

The imposition interface is programmed if the disposition interface is programmed and we have a remote VC label and an IGP label (Label Switch Path to the peer)

The IGP label can be implicit null in a back-to-back configuration

- DOWN—VC is not ready to carry traffic between the two VC endpoints
- ADMINDOWN—VC disabled by a user

# EoMPLS: Data Plane Overhead

- At imposition, PE encapsulates CE's Ethernet or VLAN packet to route across MPLS cloud
- These are the associated overheads:
  - Transport Header is 6 Bytes DA + 6 Bytes SA + 2 Bytes etype + optional 4 Bytes of VLAN Tag
  - There's (at least) 2 levels of MPLS header (Tunnel + VC) each contributing with 4 Bytes
  - There is an optional 4-Byte control word



# Calculating MTU Requirements for the Core

- Core MTU  $\geq$  Edge MTU + Transport Header + AToM Header (Control Word) + (MPLS Label Stack \* MPLS Header Size)
- Edge MTU is the MTU configured in the CE-facing PE's interface
- Examples (all in Bytes):

	Edge	Transport	AToM	MPLS Stack	MPLS Header	Total
EoMPLS Port Mode	1500	14	4 [0]	2	4	1526 [1522]
EoMPLS VLAN Mode	1500	18	4 [0]	2	4	1530 [1526]
EoMPLS Port w/ TE FRR	1500	14	4 [0]	3	4	1530 [1526]

# Changing the MTU Size in the Core

- Use the **mtu** command in PE and P routers to configured at least the calculated minimum MTU

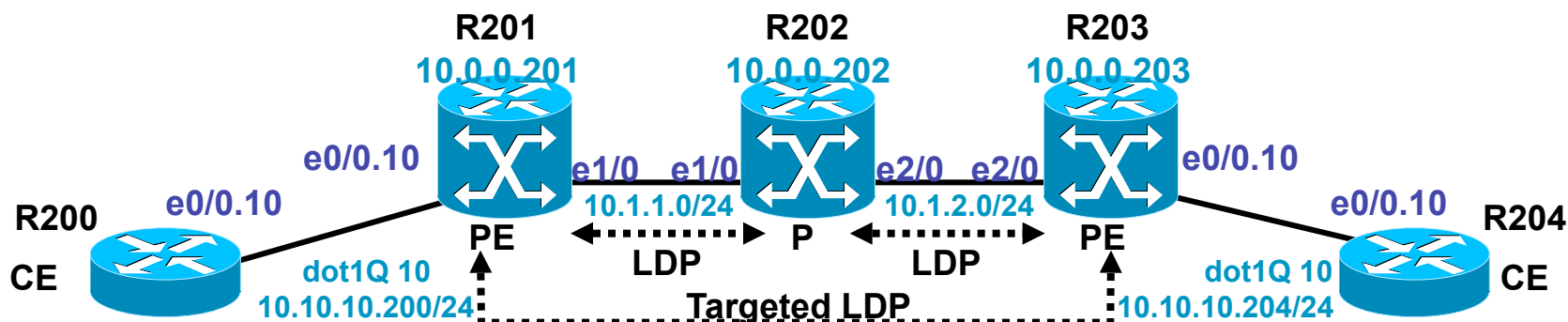
```
Router(config-if)# mtu 1526
```

- Some interfaces (such as FastEthernet interfaces) require the **mpls mtu** command to change the MTU size

# Transport Header Overhead

Transport Type	Transport Header Size
Ethernet Port	14 Bytes
Ethernet VLAN	18 Bytes
Frame Relay DLCI, Cisco Encapsulation	2 Bytes
Frame Relay DLCI, IETF Encapsulation	8 Bytes
HDLC	4 Bytes
PPP	4 Bytes
AAL5	0–32 Bytes

# MTU Problem Example



R200#ping ip 10.10.10.204 size **1470** timeout 1 df-bit

Type escape sequence to abort.

Sending 5, 1470-byte ICMP Echos to 10.10.10.204, timeout is 1 seconds:

Packet sent with the DF bit set

!!!!

Success rate is **100 percent (5/5)**, round-trip min/avg/max = 24/28/32 ms

R200#

R200#ping ip 10.10.10.204 size **1471** timeout 1 df-bit

Type escape sequence to abort.

Sending 5, 1470-byte ICMP Echos to 10.10.10.204, timeout is 1 seconds:

Packet sent with the DF bit set

.....

Success rate is **0 percent (0/5)**

R200#

	Core Overhead
Transport	18 Bytes
AToM	4 Bytes
MPLS Stack	2 Headers
MPLS Header	4 Bytes
<b>Total</b>	<b>30 Bytes</b>



# Control Plane: Displaying L2CKT Binding

```
R201#show mpls l2transport binding 10
Destination Address: 10.0.0.203, VC ID: 10
Local Label: 19
Cbit: 1, VC Type: Eth VLAN, GroupID: 0
MTU: 1500, Interface Desc: *** To R200 ***
VCCV Capabilities: Type 1, Type 2
Remote Label: 21
Cbit: 1, VC Type: Eth VLAN, GroupID: 0
MTU: 1500, Interface Desc: *** To R204 ***
VCCV Capabilities: Type 1, Type 2
R201#
```

**Advertised  
by LDP**

**Signaled as  
Interface Parameters**

- Cbit: Flags the presence of a control word
- VC Type: (PW type) A 15 bit quantity containing a value which represents the type of PW
- VCCV Capabilities: Virtual Circuit Connection Verification
  - Type 1: PWE3 control word (0x0001 as first nibble of CW)
  - Type 2: MPLS Router Alert Label
- Interface parameters: used to provide interface specific parameters, such as CE-facing interface MTU

# Decoded Label Mapping (Binding) for PW

## Label Mapping Message

```
0... .. = U bit: Unknown bit not set
Message Type: Label Mapping Message (0x400)
Message Length: 53
Message ID: 0x00000a37
Forwarding Equivalence Classes TLV
00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
TLV Type: Forwarding Equivalence Classes TLV (0x100)
TLV Length: 37
FEC Elements
  FEC Element 1 VCID: 10
    FEC Element Type: Virtual Circuit FEC (128)
    1... .... = C-bit: Control Word Present
    .000 0000 0000 0100 = VC Type: Ethernet VLAN (0x0004)
    VC Info Length: 29
    Group ID: 0
    VC ID: 10
    Interface Parameter: MTU 1500
      ID: MTU (0x01)
      Length: 4
      MTU: 1500
    Interface Parameter: Description
      ID: Interface Description (0x03)
      Length: 17
      Description: *** To R200 ***
    Interface Parameter: VCCV
      ID: VCCV (0x0a)
      Length: 4
      VCCV Capabilities: Type 1 and 2
Generic Label TLV
00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
TLV Type: Generic Label TLV (0x200)
TLV Length: 4
Generic Label: 19
```

# Checking LDP State

```
R201#show mpls ldp discovery
Local LDP Identifier:
    10.0.0.201:0
Discovery Sources:
    Interfaces:
        Ethernet1/0 (ldp): xmit/rcv
            LDP Id: 10.0.0.202:0
        Targeted Hellos:
            10.0.0.201 -> 10.0.0.203 (ldp): active/passive, xmit/rcv
                LDP Id: 10.0.0.203:0
R201#show mpls ldp neighbor
Peer LDP Ident: 10.0.0.202:0; Local LDP Ident 10.0.0.201:0
    TCP connection: 10.0.0.202.11039 - 10.0.0.201.646
    State: Oper; Msgs sent/rcvd: 54/54; Downstream
    Up time: 00:41:07
    LDP discovery sources:
        Ethernet1/0, Src IP addr: 10.1.1.202
    Addresses bound to peer LDP Ident:
        10.0.0.202      10.1.1.202      10.1.2.202
Peer LDP Ident: 10.0.0.203:0; Local LDP Ident 10.0.0.201:0
    TCP connection: 10.0.0.203.11010 - 10.0.0.201.646
    State: Oper; Msgs sent/rcvd: 53/53; Downstream
    Up time: 00:38:36
    LDP discovery sources:
        Targeted Hello 10.0.0.201 -> 10.0.0.203, active, passive
    Addresses bound to peer LDP Ident:
        10.0.0.203      10.1.2.203
R201#
```

# Verifying Core Forwarding State

- IGP Labels untagged!

```
R201#show mpls forwarding-table
```

Local	Outgoing	Prefix	Bytes	tag	Outgoing	Next Hop
	tag	tag or VC		switched	interface	
16	Untagged	10.1.2.0/24	0		Et1/0	10.1.1.202
17	Untagged	10.0.0.202/32	0		Et1/0	10.1.1.202
18	Untagged	10.0.0.203/32	0		Et1/0	10.1.1.202
19	Untagged	12ckt (10)	1984544		Et0/0.10	point2point
20	Untagged	12ckt (20)	45183		Et3/0	point2point
21	Untagged	12ckt (50)	1435873		Se5/0	point2point

R201#

**IGP Outgoing Labels Untagged;  
These Are Label Mappings We Should  
Have Received from the LDP Neighbor**

- The Link LDP session may be down

# Core LDP Problems

- Indeed, IGP LDP session is down

```
R201#show mpls ldp neighbor
Peer LDP Ident: 10.0.0.203:0; Local LDP Ident 10.0.0.201:0
TCP connection: 10.0.0.203.11027 - 10.0.0.201.646
State: Oper; Msgs sent/rcvd: 12/12; Downstream
Up time: 00:01:30
LDP discovery sources:
Targeted Hello 10.0.0.201 -> 10.0.0.203, active, passive
Addresses bound to peer LDP Ident:
10.0.0.203      10.1.2.203
R201#
R201#show mpls ldp
Local LDP Ident 10.0.0.201:0
Discovery Sources:
Interfaces:
Ethernet1/0 (ldp): xmit
Targeted Hellos:
10.0.0.201 -> 10.0.0.203 (ldp): active/passive, xmit/rcv
LDP Id: 10.0.0.203:0
R201#
```

**Only Targeted Session Is UP**

**Transmitting LDP Hellos  
but Not Receiving Any;  
Good State Is xmit/rcv**

## Note of Caution

- The operational status in `show mpls interfaces` is not an indicator of neighbor UP

```
R201#show mpls interfaces
Interface          IP          Tunnel    Operational
Ethernet1/0        Yes (ldp)   No        Yes
R201#
```

- **The operational status merely shows that some application has enabled processing of MPLS packets**
- **In fact, in Inter-AS scenarios the interface to which the MP-eBGP VPNv4 neighbor is connected will show operational Yes even without `mpls ip` configured on the interface**

# Use IP/ICMP-Based Traceroute from PE

- Traceroute with core LDP down (but targeted LDP UP)

```
R201#traceroute 10.0.0.203
Type escape sequence to abort.
Tracing the route to 10.0.0.203
 1 10.1.1.202 28 msec 24 msec 40 msec
 2 10.1.2.203 32 msec 44 msec 36 msec
```

- Traceroute with core LDP UP

**ICMP Extensions for MultiProtocol Label Switching**

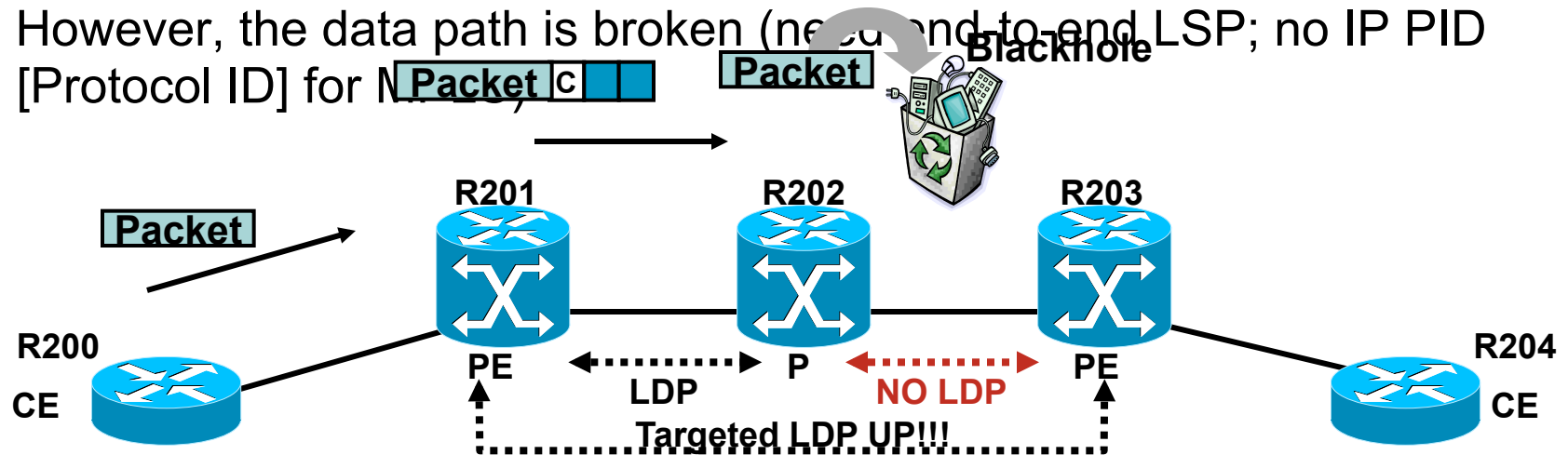
```
R201#traceroute 10.0.0.203
Type escape sequence to abort.
Tracing the route to 10.0.0.203
 1 10.1.1.202 [MPLS: Label 17 Exp 0] 36 msec 20 msec 40 msec
 2 10.1.2.203 28 msec 40 msec 28 msec
R201#
```

**No MPLS Shim Due to PHP, Would Be 0 If 'mpls ldp explicit-null'**

- Remember that PE receives Layer 2 (not IP) packet from CE; here there's no concept of 'mpls ip propagate-ttl'

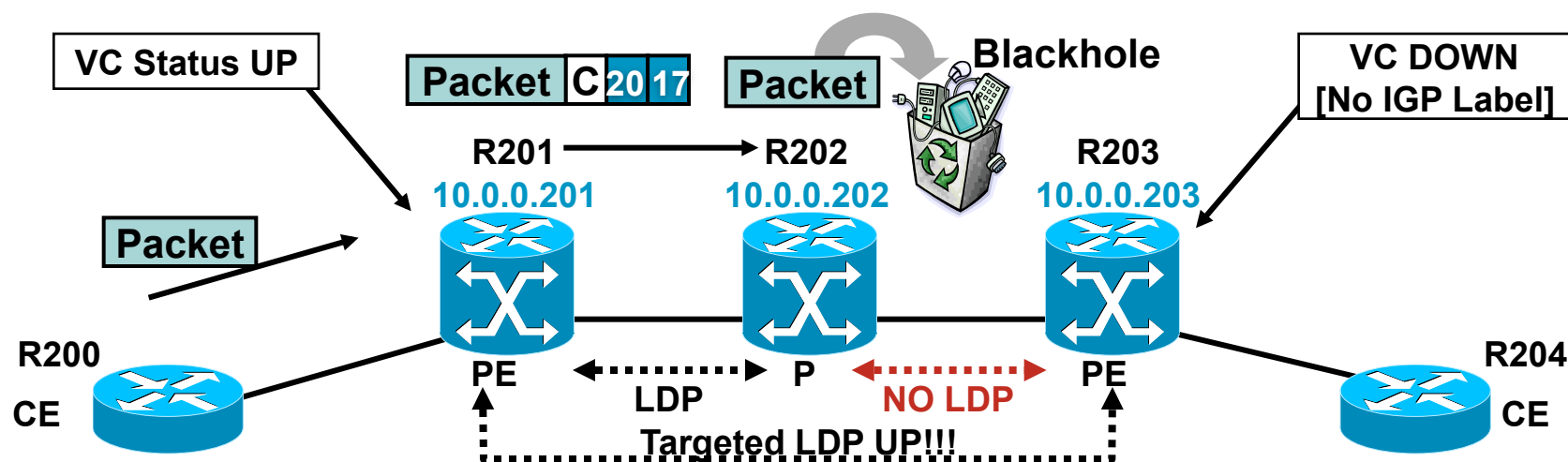
# Core LDP Important Note

- The targeted LDP session can be UP even if the Link LDP session (core LDP) is down; all VCs would be UP
- Hey, it just needs a TCP connection that can run over TCPoIP as opposed to TCPoIPoMPLS
- However, the data path is broken (need to end LSP; no IP PID [Protocol ID] for N...





# Core Network Dataplane Blackhole



```
R201#show mpls l2transport vc 10 detail | i status:|Out
Destination address: 10.0.0.203, VC ID: 10, VC status: up
Output interface: Et1/0, imposed label stack {17 20}
```

```
R201#show mpls forwarding-table 10.0.0.203
```

Local	Outgoing	Prefix	Bytes	tag	Outgoing	Next Hop
	tag	or VC			switched	interface
22	17	10.0.0.203/32	0		Et1/0	10.1.1.202

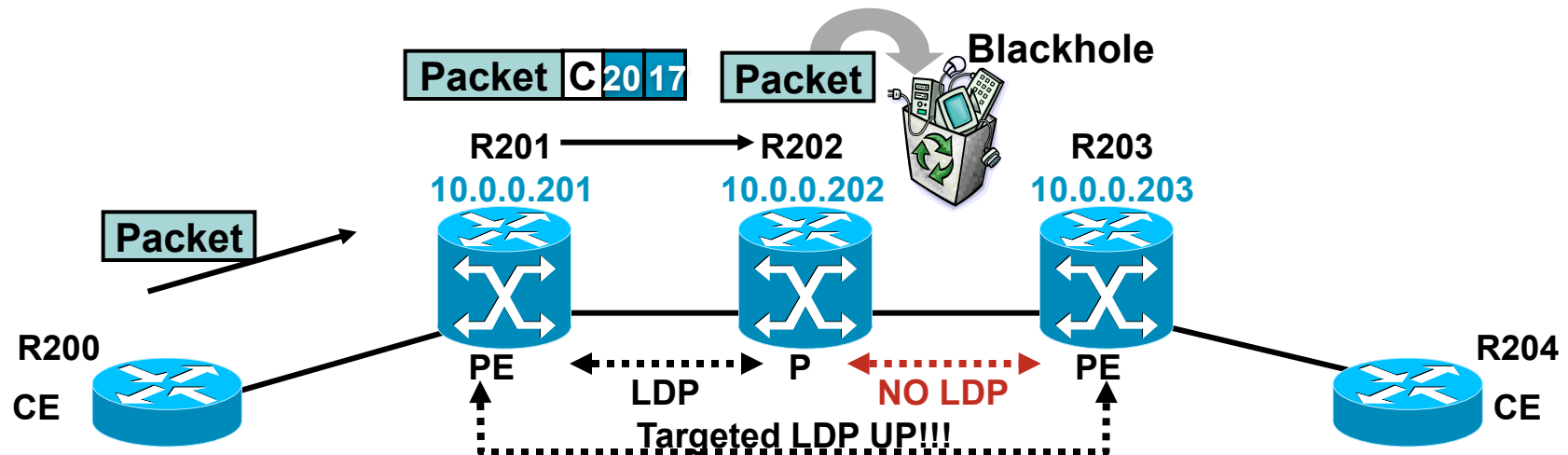
R201#

```
R202#show mpls forwarding-table
```

Local	Outgoing	Prefix	Bytes	tag	Outgoing	Next Hop
	tag	or VC			switched	interface
16	0	10.0.0.201/32	0		Et1/0	10.1.1.201
17	Untagged	10.0.0.203/32	44773		Et2/0	10.1.2.203

R202#

# Core Network Dataplane Blackhole



```
R201#traceroute 10.0.0.203

Type escape sequence to abort.
Tracing the route to 10.0.0.203

 1 10.1.1.202 [MPLS: Label 17 Exp 0] 16 msec 44 msec 20 msec
 2 10.1.2.203 28 msec 28 msec 48 msec

R201#
```

**With ICMP/IP Based Traceroute, We Cannot Tell If the Absence of MPLS Header Is Because:**

**1. PHP :-)**

**2. UNTAGGED !!! :-)**

- But data path is broken...
- How do we troubleshoot this one?

# With MPLS Embedded Management

- Virtual Circuit Connection Verification (VCCV) is smart enough to differentiate POP vs. untagged

POP—the next hop advertised an implicit NULL label for the destination and that this router pops the top label

Untagged—there is no label for the destination from the next hop; remove all labels in the stack

- “Detecting MPLS dataplane failures” has PING and traceroute modes
- MPLS echo request and echo reply are UDP packets to port 3503 using label stack to be switched **inband** the LSP

# Virtual Circuit Connection Verification

- MPLS ping has `IPv4`, `pseudowire` and `traffic-eng` modes
- MPLS traceroute has `IPv4` and `traffic-eng` modes
- VCCV consists of:

Signaling component in LDP label mapping for VC FEC with VCCV interface parameter

Switching component so that VCCV “control” packet is treated as AToM payload from switching standpoint

VCCV disposition capabilities are:

Type 1—Uses PID in AToM control word (0x1) [default]

Type 2—Uses MPLS router alert label

# MPLS Pseudowire PING

```
R201#ping 10.0.0.203
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 10.0.0.203, timeout is 2 seconds:
```

```
!!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/27/36 ms
```

```
R201#
```

```
R201#ping mpls pseudowire 10.0.0.203 10
```

```
Sending 5, 100-byte MPLS Echos to 10.0.0.203/0,  
timeout is 2 seconds, send interval is 0 msec:
```

```
Codes: '!' - success, 'Q' - request not transmitted,  
        '.' - timeout, 'U' - unreachable,  
        'R' - downstream router but not target
```

```
Type escape sequence to abort.
```

```
.....
```

```
Success rate is 0 percent (0/5)
```

```
R201#
```

# Virtual Circuit Connection Verification

- Two reply modes:

IPv4—replies with UDP/IPv4 packet; do not have control over whether the reply is forwarded as IP or IPoMPLS; if no reply, use the next option

Router alert—adds the router alert option to the IP header (RFC2113); this means that Cisco routers will be processed switched (by the RP) at each intermediate hop

- The 'router-alert' option bypasses hardware and linecard forwarding table inconsistencies by being punted to the process switching path, but it's more expensive

# MPLS Traceroute for IPv4

```
R201#traceroute mpls ipv4 10.0.0.203/32
Tracing MPLS Label Switched Path to 10.0.0.203/32, timeout is 2 seconds

Codes: '!' - success, 'Q' - request not transmitted,
        '.' - timeout, 'U' - unreachable,
        'R' - downstream router but not target

Type escape sequence to abort.
0 10.1.1.201 MRU 1500 [Labels: 17 Exp: 0] ! MRU in show mpls forw det
R 1 10.1.2.202 MRU 1504 [No Label] 60 ms ! Untagged
! 2 10.1.2.203 80 ms
```

```
R201#
R201#traceroute mpls ipv4 10.0.0.203/32
Tracing MPLS Label Switched Path to 10.0.0.203/32, timeout is 2 seconds

Codes: '!' - success, 'Q' - request not transmitted,
        '.' - timeout, 'U' - unreachable,
        'R' - downstream router but not target

Type escape sequence to abort.
0 10.1.1.201 MRU 1500 [Labels: 17 Exp: 0]
R 1 10.1.2.202 MRU 1500 [Labels: 0 Exp: 0] 40 ms ! If Explicit NULL
! 2 10.1.2.203 80 ms

R201#
```

# MPLS Traceroute for IPv4

```
R201#traceroute mpls ipv4 10.0.0.203/32
Tracing MPLS Label Switched Path to 10.0.0.203/32, timeout is 2 seconds

Codes: '!' - success, 'Q' - request not transmitted,
        '.' - timeout, 'U' - unreachable,
        'R' - downstream router but not target

Type escape sequence to abort.
 0 10.1.1.201 MRU 1500 [Labels: 17 Exp: 0]
R 1 10.1.2.202 MRU 1504 [implicit-null] 64 ms ! Implicit NULL
    ! 2 10.1.2.203 92 ms
```

```
R201#
R201#traceroute mpls ipv4 10.0.0.203/32
Tracing MPLS Label Switched Path to 10.0.0.203/32, timeout is 2 seconds

Codes: '!' - success, 'Q' - request not transmitted,
        '.' - timeout, 'U' - unreachable,
        'R' - downstream router but not target

Type escape sequence to abort.
 0 10.1.1.201 MRU 1500 [Labels: 17 Exp: 0]
R 1 10.1.2.202 MRU 1504 [No Label] 76 ms
    . 2 *
    [snip]
    . 6 *

R201#
```



# Verifying Forwarding State for L2CKTs

- No local labels for PW; targeted LDP down?

```
R201#show mpls forwarding-table | include Prefix|l2ckt
Local  Outgoing  Prefix          Bytes tag  Outgoing  Next Hop
R201#
```

- Local labels assigned to PW (VC)

```
R201#show mpls forwarding-table | include Prefix|l2ckt
Local  Outgoing  Prefix          Bytes tag  Outgoing  Next Hop
19     Untagged  12ckt (10)      1952531   Et0/0.10  point2point
20     Untagged  12ckt (20)      3024      Et3/0     point2point
21     Untagged  12ckt (50)      1404829   Se5/0     point2point
R201#
```

**Labels Advertised**      **VC ID (Pw ID)**      **P2p Adjacency**

# EoMPLS VC Status DOWN...

- ...but everything seems fine on the PEs

- VC Type (PW Type) mismatch

PW Type—A 15 bit quantity containing a value which represents the type of PW

VC type—0x0004 is used for IEEE 802.1Q VLAN over MPLS application; ISL **is not supported**

VC type—0x0005 is used for Ethernet port tunneling application (**port transparency**)

- MTU mismatch

MTU is carried as interface parameter in label mapping message for VC (PW) FEC

# VC Type Mismatch

- Show AToM bindings

```
R201#show mpls l2transport binding 10
Destination Address: 10.0.0.203, VC ID: 10
Local Label: 19
Cbit: 1, VC Type: Ethernet, GroupID: 0
MTU: 1500, Interface Desc: n/a
VCCV Capabilities: Type 1, Type 2
Remote Label: 21
Cbit: 1, VC Type: Eth VLAN, GroupID: 0
MTU: 1500, Interface Desc: *** To R204 ***
VCCV Capabilities: Type 1, Type 2
```

VC Type Mismatch !!!

- AToM VC is not enough to find the problem

```
R201#show mpls l2transport vc 10 detail
Local interface: Et3/0 up, line protocol up, Ethernet up
Destination address: 10.0.0.203, VC ID: 10, VC status: down
Tunnel label: not ready
Output interface: unknown, imposed label stack {}
Create time: 00:47:45, last status change time: never
Signaling protocol: LDP, peer 10.0.0.203:0 up
MPLS VC labels: local 19, remote 21
Group ID: local 0, remote 0
MTU: local 1500, remote 1500
Remote interface description: *** To R204 ***
Sequencing: receive disabled, send disabled
VC statistics:
packet totals: receive 0, send 0
byte totals: receive 0, send 0
packet drops: receive 0, send 0
```

VC Down  
But All Seems Fine

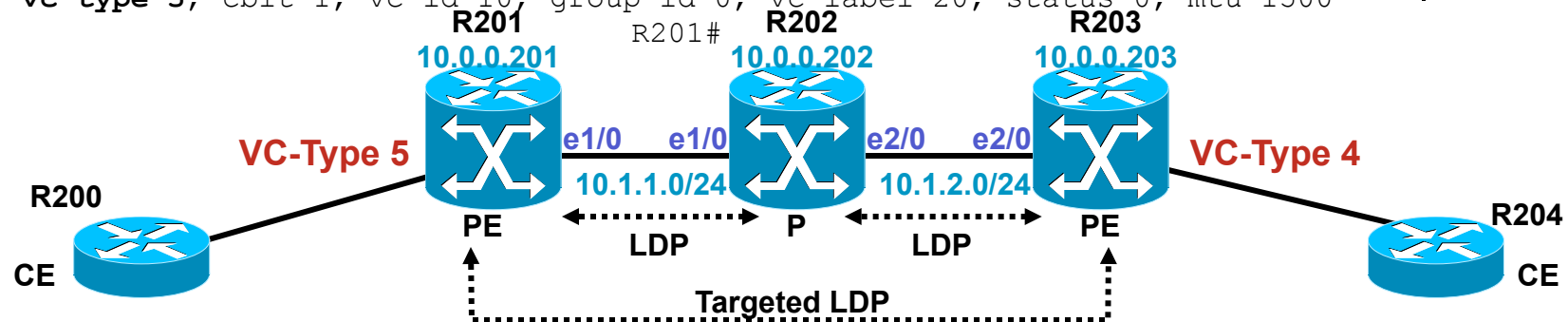
# VC Type Mismatch

- Enable `debug mpls l2transport signaling message`
- Need to shut/no shut to trigger sending new LDP msgs

```

R201#
*Apr 21 19:17:44.308: ATOM LDP [10.0.0.203]: Received label withdraw msg, id 3964
  vc type 4, cbit 1, vc id 10, group id 0, vc label 22, status 0, mtu 0
*Apr 21 19:17:44.400: ATOM LDP [10.0.0.203]: Sending label release msg
  vc type 4, cbit 1, vc id 10, group id 0, vc label 22, status 0, mtu 0
*Apr 21 19:17:58.188: ATOM LDP [10.0.0.203]: Received label mapping msg, id 3965
  vc type 4, cbit 1, vc id 10, group id 0, vc label 21, status 0, mtu 1500
*Apr 21 19:18:42.100: ATOM LDP [10.0.0.203]: Sending label withdraw msg
  vc type 5, cbit 1, vc id 10, group id 0, vc label 19, status 0, mtu 1500
*Apr 21 19:18:42.512: ATOM LDP [10.0.0.203]: Received label release msg, id 3968
  vc type 5, cbit 1, vc id 10, group id 0, vc label 19, status 0, mtu 0
*Apr 21 19:18:43.500: ATOM LDP [10.0.0.203]: Sending label mapping msg
  vc type 5, cbit 1, vc id 10, group id 0, vc label 20, status 0, mtu 1500
  
```

R201	R203
<u>Withdraw</u>	
<u>Release</u>	
<u>Mapping</u>	
<u>Withdraw</u>	
<u>Release</u>	
<u>Mapping</u>	



# VC Type Mismatch

- debug mpls l2transport vc event

```
R203# ! Remote shut
1d04h: AToM MGR [10.0.0.201, 10]: Delete remote vc label binding
R203# ! Remote no shut
1d04h: AToM MGR [10.0.0.201, 10]: Remote end up
1d04h: AToM MGR [10.0.0.201, 10]: Mismatch vc type in remote label binding, local 4, remote 5
R203#
```

**Remote PE**      **VC Id**

- debug mpls l2transport vc fsm

```
R203# ! Remote shut
1d04h: AToM MGR [10.0.0.201, 10]: Event remote down, state changed from establishing to local ready
R203# ! Remote no shut
1d04h: AToM MGR [10.0.0.201, 10]: Event remote up, state changed from local ready to establishing
1d04h: AToM MGR [10.0.0.201, 10]: Event remote invalidated, state changed from establishing to establishing
1d04h: AToM MGR [10.0.0.201, 10]: Take no action
R203#
```

# MTU Mismatch

- debug mpls l2transport vc event

```
R203#  
1d05h: ATOM MGR [10.0.0.201, 50]: Remote end up  
1d05h: ATOM MGR [10.0.0.201, 50]: Mismatch MTU in remote label binding, local 1500, remote 4400  
R203#
```

- debug mpls l2transport vc fsm

```
R203#  
1d05h: ATOM MGR [10.0.0.201, 50]: Event remote up, state changed from local ready to establishing  
1d05h: ATOM MGR [10.0.0.201, 50]: Event remote invalidated, state changed from establishing to establishing  
1d05h: ATOM MGR [10.0.0.201, 50]: Take no action  
R203#
```

- Note: Changing `MTU` will cause a new LDP Mapping message to be sent for the VC (PW) FEC
- Changing the `description` will not, and shut/no shut is needed to generate the signaling message

# A Successful Pseudowire Establishment

- debug mpls l2transport vc event

```
R203#
1d05h: AToM MGR [10.0.0.201, 50]: Remote end up
1d05h: AToM MGR [10.0.0.201, 50]: Validate vc, activating data plane
1d05h: AToM SMGR: Submit Imposition Update
1d05h: AToM SMGR: Submit Disposition Update
1d05h: AToM SMGR: Submit SSM event
1d05h: AToM SMGR [10.0.0.201, 50]: Event Imposition Enable, imp-ctrlflag 83, remote vc label 20
1d05h: AToM SMGR [10.0.0.201, 50]: Imposition Programmed, Output Interface: Et1/0
1d05h: AToM SMGR [10.0.0.201, 50]: State [Provisioned->Imposition Rdy]
1d05h: AToM SMGR [10.0.0.201, 50]: Event Disposition Enable, disp-ctrlflag 3, local vc label 16
1d05h: AToM SMGR [10.0.0.201, 50]: State [Imposition Rdy->Imposition/Disposition Rdy]
1d05h: AToM SMGR: Event SSM event
1d05h: AToM SMGR [10.0.0.201, 50]: sucessfully processed ssm provision request pwid 300001F
1d05h: AToM SMGR [10.0.0.201, 50]: Send COMPLETE signal to SSM
1d05h: AToM SMGR [10.0.0.201, 50]: sucessfully setup sss switch for pwid 300001F
1d05h: AToM SMGR: Submit SSM event
1d05h: AToM SMGR: Event SSM event
1d05h: AToM SMGR [10.0.0.201, 50]: sucessfully processed ssm bind for pwid 300001F
1d05h: AToM MGR [10.0.0.201, 50]: Receive SSM dataplane up notification
1d05h: AToM MGR [10.0.0.201, 50]: Dataplane activated
R203#
```

**Pseudowire Dataplane Is Up  
Only if Imposition/Disposition  
Are Successfully Programmed**

# What If the Core Uses Traffic Engineering?

- Need to use the command ``preferred-path {interface | peer}`` under the ``pseudowire-class``; have in mind that:

The selected path must be a label switched path (LSP) destined to the peer PE router

If you specify a tunnel (selecting interface):

The tunnel must be an MPLS traffic engineering tunnel

The tunnel tailend must be on the remote PE router

If you specify an IP address (selecting peer):

The address must be the IP address of a *loopback* interface on the *remote PE router*, not necessarily the LDP router-id address; peer means *targeted LDP peer*

The address must have a /32 mask

There must be an LSP destined to that selected address

The LSP does not have to be a TE tunnel



# Tunnel Selection Details

- Tunnel selection configuration

```
pseudowire-class AToM_FRoMPLS
    encapsulation mpls
    preferred-path interface Tunnell
    !
    interface Tunnell
        description AToM_Tunnel
        ip unnumbered Loopback1
        mpls ip
        tunnel destination 192.168.200.2
        tunnel mode mpls traffic-eng
        tunnel mpls traffic-eng autoroute announce
        tunnel mpls traffic-eng path-option 1 dynamic
        connect VPN200 POS3/0 200 l2transport
```

- Tunnel selection verification

```
xconnect 192.168.200.2 200 encapsulation mpls pw-class AToM_FRoMPLS
PE1#show mpls l2transport vc 200 de
Local interface: PO3/0 up, line protocol up, FR DLCI 200 up
Destination address: 192.168.200.2, VC ID: 200, VC status: up
Data plane status: imposition OK, disposition OK
Output interface: Tu1, imposed label stack {24 35}
Preferred path: Tunnell, active
Default path: not supported
Create time: 00:28:33, last status change time: 00:27:09
Signaling protocol: LDP, peer 192.168.200.2:0 up
MPLS VC labels: local 39, remote 24
Group ID: local 0, remote 0
MTU: local 4470, remote 4470
Remote interface description: to CE2
Sequencing: receive disabled, send disabled
VC statistics:
packet totals: receive 30, send 30
byte totals: receive 11295, send 11745
packet drops: receive 0, send 0
```

# AToM and Label Stack

- If only LDP is used in the MPLS network, the label stack size is 2 {LDP label; VC label}
- If only RSVP-TE is used in the MPLS network (a TE tunnel between PE routers), the imposition PE uses a label stack size of 2 {TE label; VC label}
- If RSVP-TE and LDP are both used in the MPLS network (a P-P or PE-P TE tunnel, with LDP on the tunnel), the label stack is 3 {TE label; LDP label; VC label}
- If using MPLS Fast Reroute (FRR) anywhere in the MPLS network add one more label to the stack for the above cases, max label stack in this case is 4 {FRR label; TE label; LDP label; VC label}

# AToM and Label Stack

- If AToM is used in a MPLS-VPN CSC environment, the maximum MPLS label stack size in the provider carrier network is 5 {FRR label; TE label; LDP label; VPN label; VC label}
- If AToM tunnel spans different service providers and the providers exchange MPLS labels using IPv4 BGP (RFC3107), then the max label stack size is 5 {FRR label; TE label; BGP label; LDP label; VC label}
- Add a Router Alert label here or there, and label stack size is now 6
- ...and there is no mechanism to discover the MTU of an LSP dynamically...

# Troubleshooting Attachment Circuits

- Using `debug acircuit {error | event}`

```
R203#debug acircuit ?
error  Attachment Circuit errors
event  Attachment Circuit events

R203#
R203(config-if)#no shut
R203(config-if)#
3d20h: ACLIB [10.0.0.201, 20]: SW AC interface UP for Ethernet interface Et3/0
3d20h: ACLIB [10.0.0.201, 20]: pthru_intf_handle_circuit_up() calling acmgr_circuit_up
3d20h: ACLIB [10.0.0.201, 20]: Setting new AC state to Ac-Connecting
3d20h: ACLIB: Update switching plane with circuit UP status
3d20h: ACLIB [10.0.0.201, 20]: SW AC interface UP for Ethernet interface Et3/0
3d20h: ACLIB [10.0.0.201, 20]: pthru_intf_handle_circuit_up() ignoring up event. Already connected or connecting.
3d20h: Et3/0 ACMGR: Receive <Circuit Up> msg
3d20h: Et3/0 ACMGR: circuit up event, FSP state chg sip up to connected, action is send connected msg
3d20h: ACLIB: pthru_intf_response hdl is D8000019, response is 2
3d20h: ACLIB [10.0.0.201, 20]: Setting new AC state to Ac-Connected
3d20h: Et3/0 ACMGR: Receive <Remote Up Notification> msg
3d20h: Et3/0 ACMGR: remote up event, FSP connected state no chg, action is ignore
3d20h: %LINK-3-UPDOWN: Interface Ethernet3/0, changed state to up
3d20h: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet3/0, changed state to up
```

- Also `debug xconnect {error | event}`

# Hardware Platforms: 7600

## From the RP

```
cat13#sh mpls 12 vc det
vcid: 200, local groupid: 24, remote groupid: 102 (vc is up)
client: V1200 is up, destination: 1.2.2.1, Peer LDP Ident: 1.2.1.1:0
local label: 215, remote label: 215, tunnel label: implc-null
outgoing interface: s0/0, next hop: point2point
Local MTU: 1500, Remote MTU: 1500
Remote interface description: Vlan200
imposition: LC Programmed ← Imposition rewrite resolved
current imposition/last disposition slot: 2/32
Packet totals(in/out): 6246/6159
byte totals(in/out): 536999/444722
```

# Hardware Platforms: 7600

## From the LC: Imposition

```
CWTLC-Slot2#show mpls 12 imposition detail
label:0 vc_id:200 label:215 vlan:200    ! 0 is Tunnel label, 215 is VC label
output pkts: 82  output bytes: 5917  priority: 0  active: yes
MPLS Detailed info:
  Impose Rewrite:
    215          Enet/MPLS impose      0          PO2/2          point2point
    MAC/Encaps=14/18, MTU=4470, Tag Stack{215}
    00000210000000005DC57880A0800 000D7100    ! Actual imposition rewrite
                                Ethernet II      MPLS shim
```

```
CWTLC detailed info: if_number 28 oper:PUSH 1(1) func_tbl:0x1 flags:0x0
      t VLAN          tun lbl  vc lbl  MAC Address      cos2exp
- x---(d---) d----- d----- ----- x-----    ! X: hex; d: dec;
  0 00C8(0200) 00000000 00000215 0000.0000.0000 76543210
```

```
VLAN          V S  p en VPN  intf trnk ltl          ! Tx VLAN Table
x---(d---) - - - - -
00C8(0200) v 03 0 01 0000 0401 0000 0061
```

```
txvc lkhd l2hd tx_q hi_q          ! Tx VC Table
```

```
-----
0401 0100 FF03 00D2 0000
```

Used by the Toaster

# Hardware Platforms: 7600

## From the LC: Imposition

```
CWTLC-Slot2#show mpls l2 disposition detail
Vlan200 vcid: 200 vc label: 215
  input pkts: 83  input bytes: 7137
  CWTLC detailed info: if_number 28
t vclbl VLAN      Type      impidx intfidx rx_q
- d---- x---(d---) ----- x----- x----- x---
0 00215 00C8(0200) vlan      000061 000401 00D3
1 00215 00C8(0200) vlan      000061 000801 0054
```

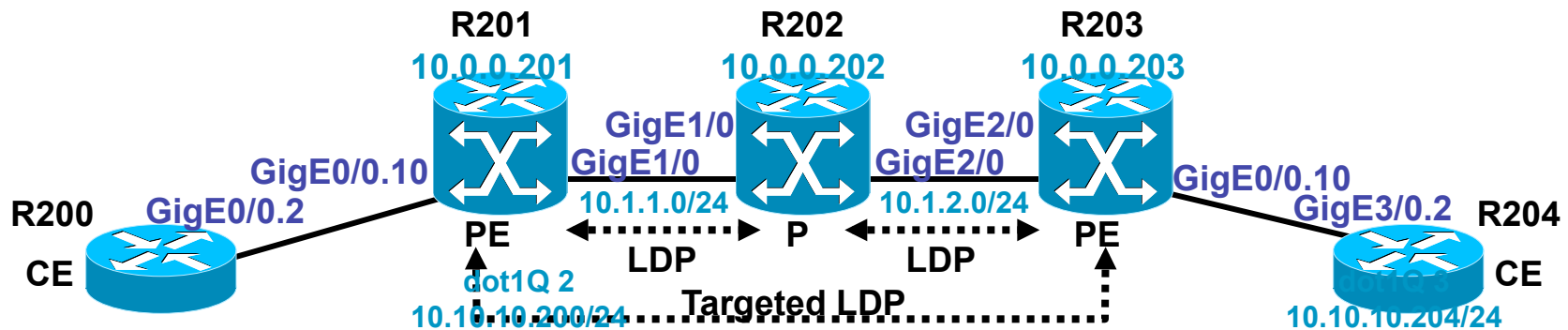
- debug mpls l2transport vlan control

Ethernet VLAN transport over MPLS, control interactions

- debug mpls l2transport vlan distributed

Ethernet VLAN transport over MPLS, distributed switching

# VLAN ID Rewrite Configuration



**R201**

```
interface GigabitEthernet0/0.2
  encapsulation dot1q 2
  no ip directed-broadcast
  no cdp enable
  xconnect 10.0.0.203 2 encapsulation mpls
  remote circuit id 3
```

XConnect submode

**R203**

```
interface GigabitEthernet3/0.2
  encapsulation dot1q 3
  no ip directed-broadcast
  no cdp enable
  xconnect 10.0.0.201 2 encapsulation mpls
  remote circuit id 2
```



# Checking VLAN ID Rewrite in a C12K

In R201 in Slot 0 [``using attach`` or ``execute-on``]:

Interface      VLAN-Id

```
LC-Slot0#show controllers eompls forwarding-table 0 2
Port # 0, VLAN-ID # 2, Table-index 2
EoMPLS Forwarding Table
tag_rewrite_ptr = D0018B58
Leaf entry? = 1
FCR index = 20
**tag_rewrite_psa_addr = 0006ED60
**tag_rewrite_vir_addr = 7006ED60
**tag_rewrite_phy_addr = F006ED60
[0-7] loq 8800 mtu 4458 oq 4000 ai 3 oi 04019110 (encaps size 4)
cw-size 4 vlanid-rew 3
gather A30 (bufhdr size 32 EoMPLS (Control Word) Imposition profile 81)
2 tag: 18 18
counters 1182, 10 reported 1182, 10.
Local OutputQ (Unicast): Slot:2 Port:0 RED queue:0 COS queue:0
Output Q (Unicast): Port:0 RED queue:0 COS queue:0
```

**1 Means Port is Configured to Transport Layer 2 VLAN Packets**

**Shows Which Slot and Port the Tunneled Packets Exit from to Enter the MPLS Backbone**

# A Couple of CEF Details

## Note That We Are Creating CEF RAW Adjacencies

```

R203#show cef idb | include Int|AC
    Interfaces      FIndex  IIndex Subblocks
Ethernet0/0.10     15       15 AC Xconnect vlan
Ethernet2/0.20     16       16 AC Xconnect vlan
    Ethernet3/0      5         5 AC Xconnect
Ethernet4/0.1     17       17 AC Xconnect vlan
    Serial5/0        7         7 AC Xconnect
    Serial6/0        8         8 AC Xconnect
    Serial9/0       11       11 AC Xconnect

R203#show adjacency
    Protocol Interface      Address
RAW      Ethernet3/0      point2point(4)
RAW      Serial5/0        point2point(4)
RAW      Serial6/0        point2point(4)
RAW      Serial7/0        point2point(3) ! FR doesn't appear above
RAW      Serial9/0        point2point(3)
RAW      Ethernet0/0.10   point2point(4)
RAW      Ethernet2/0.20   point2point(4)
RAW      Ethernet4/0.1    point2point(4)
TAG      Ethernet1/0      10.1.2.202(14)
IP       Ethernet1/0      10.1.2.202(22)
R203#
  
```

# A Couple of CEF Details

- In 12.2S the following commands are also available:

`show cef non-ip`

`show mpls infrastructure { lfd | lsd } rewrite detail`

LFD: Label Forwarding Database

LSD: Label Switch Database



# MPLS L2VPNs: FRoMPLS, ATMoMPLS, Local Switching, PPP/HDLC over MPLS

# Agenda

- Frame Relay over MPLS
- ATM over MPLS
- Local Switching
- PPP/HDLC over MPLS



# MPLS L2VPNs: FRoMPLS

# Configuring FRoMPLS PVC Mode

- Typical PE configuration

```
R201#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R201(config)#frame-relay switching      ! To enable FR DCE/NNI LMI
R201(config)#interface serial7/0
R201(config-if)#encapsulation frame-relay ietf
R201(config-if)#frame-relay intf-type dce
R201(config-if)#no shut
R201(config-if)#exit
R201(config)#
*Apr 26 16:12:35.344: %LINK-3-UPDOWN: Interface Serial7/0, changed state to up
*Apr 26 16:12:51.416: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial7/0,
changed state to up
R201(config)#connect frompls serial7/0 100 12transport
R201(config-fr-pw-switching)#xconnect 10.0.0.203 70 encapsulation mpls
R201(config-fr-pw-switching)#end
```

- debug mpls l2transport signaling

```
Oct 31 22:21:26.924 PST: ATOM LDP [192.168.0.7]: Sending label mapping msg
vc type 1, cbit 1, vc id 70, group id 0, vc label 69, status 0, mtu 1500
```

# Configuring FRoMPLS

## Typical CE Configuration (Same as If Connected to FR Switch)

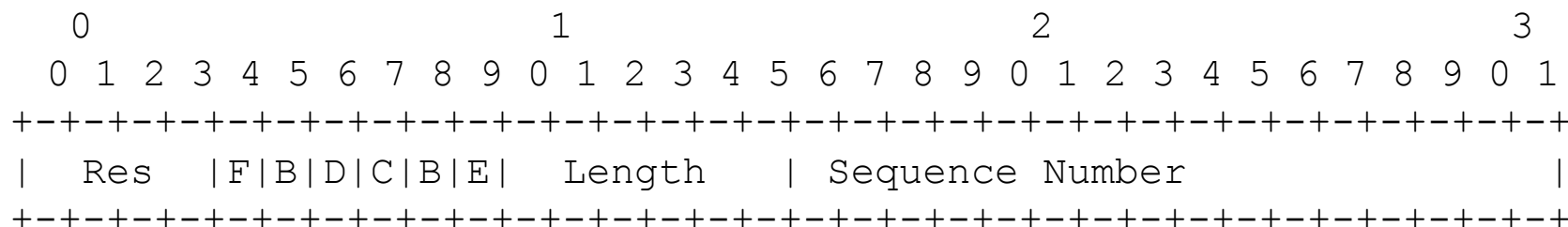
```
!
interface Serial7/0
  no ip address
  no ip directed-broadcast
  encapsulation frame-relay IETF
!
interface Serial7/0.1 point-to-point
  ip address 10.10.7.204 255.255.255.0
  no ip directed-broadcast
  frame-relay interface-dlci 100 IETF
```



# Frame Relay: Terms

- **DLCI**—Data Link Connection Identifier  
10 bit address for a Frame Relay connection; it only has local significance; DLCI's 16-1007 are definable on Cisco routers
- **FECN**—Forward Explicit Congestion Notification  
This bit may be flipped by nodes in the cloud when it experiences congestion in its path to the destination remote FRAD
- **BECN**—Backward Explicit Congestion Notification  
This bit may be flipped by nodes in the cloud when packets in the opposite direction of this packet experience congestion
- **DE**—Discard Eligible  
This bit may be flipped by nodes in the cloud to indicate that packet exceeded the rate the carrier has committed to transport; packets with the DE bit set may be intentionally dropped by Frame Relay nodes in the face of congestion

## F<sub>RoPW</sub> Header (AKA Control Word)



Res (bits 0 to 3):

Reserved bits. They are set to zero on transmission and ignored on reception.

**F (bit 4):** FR FECN (Forward Explicit Congestion Notification) bit.

**B (bit 5):** FR BECN (Backward Explicit Congestion Notification) bit.

**D (bit 6):** FR DE bit indicates the discard eligibility.

**C (bit 7):** FR frame C/R (command/response) bit.

B, E (bits 8 and 9) :

B and E are fragmentation bits and their functionality is specified in [draft-ietf-pwe3-fragmentation-04.txt].

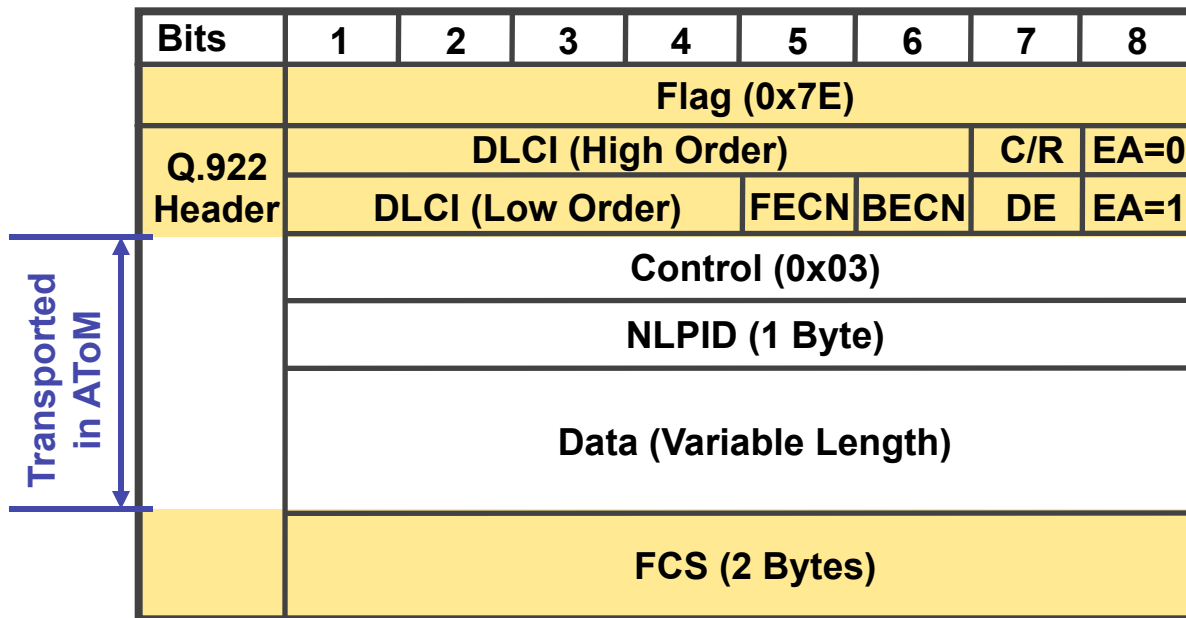
Length (bits 10 to 15):

The length field is used in conjunction with the padding of short FRoPW packets when the link layer protocol requires a minimum frame length.

Sequence number (Bit 16 to 31):

If it is not used, it is set to zero by the sender and ignored by the receiver. Otherwise it specifies the sequence number of a FROPW packet.

# Frame Relay IETF (RFC2427/1490) Packet



NLPID	Protocol
0x08	Q.933
0x80	SNAP
0x81	ISO CLNP
0xCC	Internet IP
0xB1	FRF.12

**NOTE:** Cisco Encap Uses 2 Byte Ethertype Instead of the Control and NLPID Bytes

SNAP Format—3 Byte OUI and 2 Byte PID

**OUI** \_\_\_\_\_  
 0x00 00 00 routed  
 0x00 80 C2 bridged

**PID**  
 Ethertype  
**With Preserved FCS**  
 0x00 01  
 0x00 02  
 0x00 03  
 0x00 04

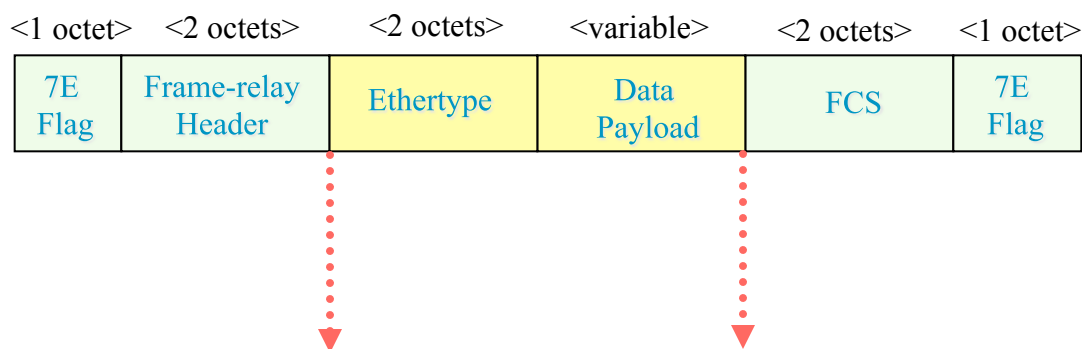
**W/O FCS**  
 0x00 07  
 0x00 08  
 0x00 09  
 0x00 0A  
 0x00 0D  
 0x00 0E  
 0x00 0F

**Media**  
 802.3/Ethernet  
 802.4  
 802.5  
 FDDI  
 Fragments  
 BPDU's  
 Source-routing  
 BPDU's

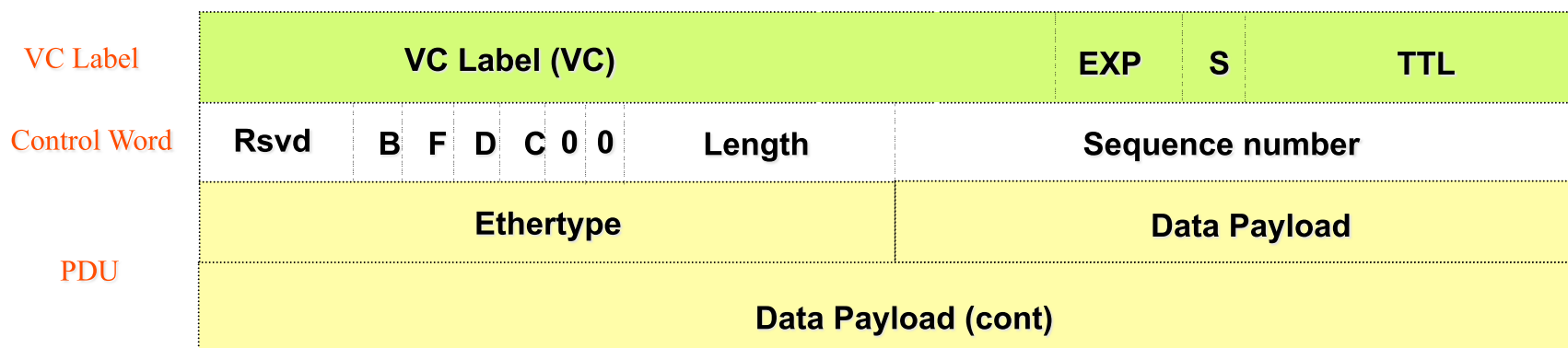
**Frame Relay PDUs are transported WITHOUT the FR header or FCS**

# Cisco Proprietary Encapsulation

## Cisco Frame Relay Encapsulation



## Transported using AToM



# Decoding an FRoMPLS Packet

- Captured using `debug mpls l2transport packet data`

```
*Apr 26 16:39:09.092: ATOM imposition: out Et1/0, size 128, EXP 0x0, seq 0, control word 0x0
*Apr 26 16:39:09.092: XX XX XX XX XX XX YY YY YY YY YY YY 88 47 00 01
                        ^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^^ ^^^^^ ^^^^^
                        SA MAC                DA MAC                etype top_shim-->
                                                MPLS
                                                Unic

*Apr 26 16:39:09.092: 00 FF 00 01 01 02 00 00 00 00 03 CC 45 00 00 64
                        ^^^^^ ^^^^^^^^^^^^^ ^^^^^^^^^^^^^ ^ ^ ^ ^^^^^...
                        <--top_shim VC_Label   Ctrl-word   |   |   Begins IP Packet
                        Label=17 Label=16           |   IP NLPID
                        S=0      S=1               Control
                        TTL=255  TTL=2

*Apr 26 16:39:09.092: 04 7A 00 00 FF 01 93 77 0A 0A 07 CC 0A 0A 07 C8
*Apr 26 16:39:09.092: 08 00 80 46 00 09 00 04 00 00 00 00 14 4E E9 A8
*Apr 26 16:39:09.092: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Apr 26 16:39:09.092: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Apr 26 16:39:09.092: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Apr 26 16:39:09.092: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Apr 26 16:39:09.092: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
```

- Capture between PE and P doing FRoMPLS

# Decoding an FRoMPLS Packet

- Captured using `debug mpls l2transport packet data`

```
*Apr 26 16:39:09.108: ATOM disposition: in Et1/0, size 102, seq 0, control word 0x0
*Apr 26 16:39:09.108: 03 CC 45 00 00 64 04 7A 00 00 FF 01 93 77 0A 0A
                        ^ ^  ^ ^  ^ ^ ^ ^ ...
                        |  |  Begins IP Packet
                        |  IP NLPID
                        Control

*Apr 26 16:39:09.108: 07 C8 0A 0A 07 CC 00 00 88 46 00 09 00 04 00 00
*Apr 26 16:39:09.108: 00 00 14 4E E9 A8 AB CD AB CD AB CD AB CD AB CD
*Apr 26 16:39:09.108: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Apr 26 16:39:09.108: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Apr 26 16:39:09.108: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
*Apr 26 16:39:09.108: AB CD AB CD AB CD
```

- At disposition, only AToM PDU is displayed

# Verifying the FRoMPLS VC

```
R203#show connection all
```

ID	Name	Segment 1	Segment 2	State
1	frompls	Se7/0 100	10.0.0.201 70	UP

```
R203#show connection id 1
```

```
FR/Pseudo-Wire Connection: 1 - frompls
```

```
Status - UP
```

```
Segment 1 - Serial7/0 DLCI 100
```

```
Segment status: UP
```

```
Line status: UP
```

```
PVC status: ACTIVE
```

```
NNI PVC status: ACTIVE
```

```
Segment 2 - 10.0.0.201 70
```

```
Segment status: UP
```

```
Requested AC state: UP
```

```
PVC status: ACTIVE
```

```
NNI PVC status: ACTIVE
```

```
R203#
```

# Verifying the FRoMPLS VC

```
R203#show mpls l2transport vc | i Local|---|70
```

Local intf	Local circuit	Dest address	VC ID	Status
Se7/0	FR DLCI 100	10.0.0.201	70	UP

```
R203#
R203#show mpls l2transport vc 70 detail
Local interface: Se7/0 up, line protocol up, FR DLCI 100 up
Destination address: 10.0.0.201, VC ID: 70, VC status: up
  Preferred path: not configured
    Default path: active
      Tunnel label: 16, next hop 10.1.2.202
        Output interface: Et1/0, imposed label stack {16 16}
          Create time: 00:17:17, last status change time: 00:06:54
            Signaling protocol: LDP, peer 10.0.0.201:0 up
              MPLS VC labels: local 16, remote 16
                Group ID: local 0, remote 0
                  MTU: local 2000, remote 2000
                    Remote interface description:
                      Sequencing: receive disabled, send disabled
                        VC statistics:
                          packet totals: receive 106, send 103
                          byte totals:   receive 11862, send 12420
                          packet drops:  receive 0, send 0
```



# Frame Relay: Show Frame Relay PVC on CE

```
R204#show frame-relay pvc interface serial 7/0 100
```

PVC Statistics for interface Serial7/0 (Frame Relay DTE)

DLCI = 100, **DLCI USAGE = LOCAL, PVC STATUS = ACTIVE**, INTERFACE = Serial7/0.1

```
input pkts 783          output pkts 788          in bytes 84510
out bytes 86254         dropped pkts 0          in FECN pkts 0
in BECN pkts 0         out FECN pkts 0        out BECN pkts 0
                        in DE pkts 0          out DE pkts 0
                        out bcast pkts 604    out bcast bytes 73636
pvc create time 01:30:14, last time pvc status changed 01:22:27
R204#
```

USAGE	LOCAL	If DLCI Is Configured on the Router
	UNUSED	If DLCI Is Not Configured on the Router But Switch Is Reporting the DLCI
STATUS	STATIC	If keepalives Are Disabled
	DELETED	If DLCI Is Defined on the Router But Not Switch
	INACTIVE	If DLCI Is Defined on Switch But the PVC Is Not up (I.E. Network, Atom, VC Failure)
	ACTIVE	If DLCI Is Defined on the Switch and Is Enabled

# FRoPW Debugging on the PE

- Captured using `debug frame-relay pseudowire`

```
R203(config-if)#no shut
R203(config-if)#
*Apr 27 14:16:51.247: %LINK-3-UPDOWN: Interface Serial7/0, changed state to up
*Apr 27 14:16:51.247: FRoPW [10.0.0.201, 70]: Local up, sending acmgr_circuit_up
*Apr 27 14:16:51.247: FRoPW [10.0.0.201, 70]: Setting pw segment UP
*Apr 27 14:16:51.263: Se7/0 ACMGR: Receive <Circuit Up> msg
*Apr 27 14:16:51.263: Se7/0 ACMGR: circuit up event, SIP state chg fsp up to
connected, action is p2p up forwarded
*Apr 27 14:16:51.263: FRoPW [10.0.0.201, 70]: PW nni_pvc_status set ACTIVE
*Apr 27 14:16:51.607: Se7/0 ACMGR: Rcv SIP msg: resp peer-to-peer msg,
hdl 78000004, sss_hdl B000006
*Apr 27 14:16:51.607: Se7/0 ACMGR: remote up event, SIP connected state no chg,
action is ignore
*Apr 27 14:16:52.267: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial7/0,
changed state to up
*Apr 27 14:17:01.271: FRoPW [10.0.0.201, 70]: SW AC update circuit state to up
*Apr 27 14:17:01.271: ACLIB: Update switching plane with circuit UP status
```



# MPLS L2VPNs: ATMoMPLS

# ATMoMPLS

## Two Different Modes of Operation of ATMoMPLS:

- ATM AAL5 (RFC2684/1483) over MPLS:  
Configured using ``encapsulation aal5'`
- ATM Cell Relay over MPLS:  
Configured using ``encapsulation aal0'`
  - VC Mode
  - VP Mode
  - Port Mode
  - Single Cell Relay (Default)
  - Packed Cell Relay

# AAL5oMPLS and AAL5 Encapsulation Details

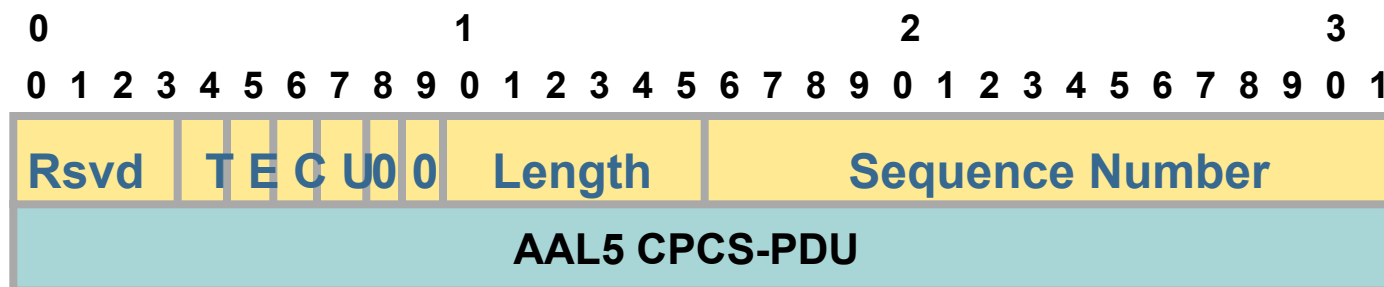
- ATM AAL5 PDUs are re-assembled at ingress PE router

Each CPCS-PDU is transported as a single packet

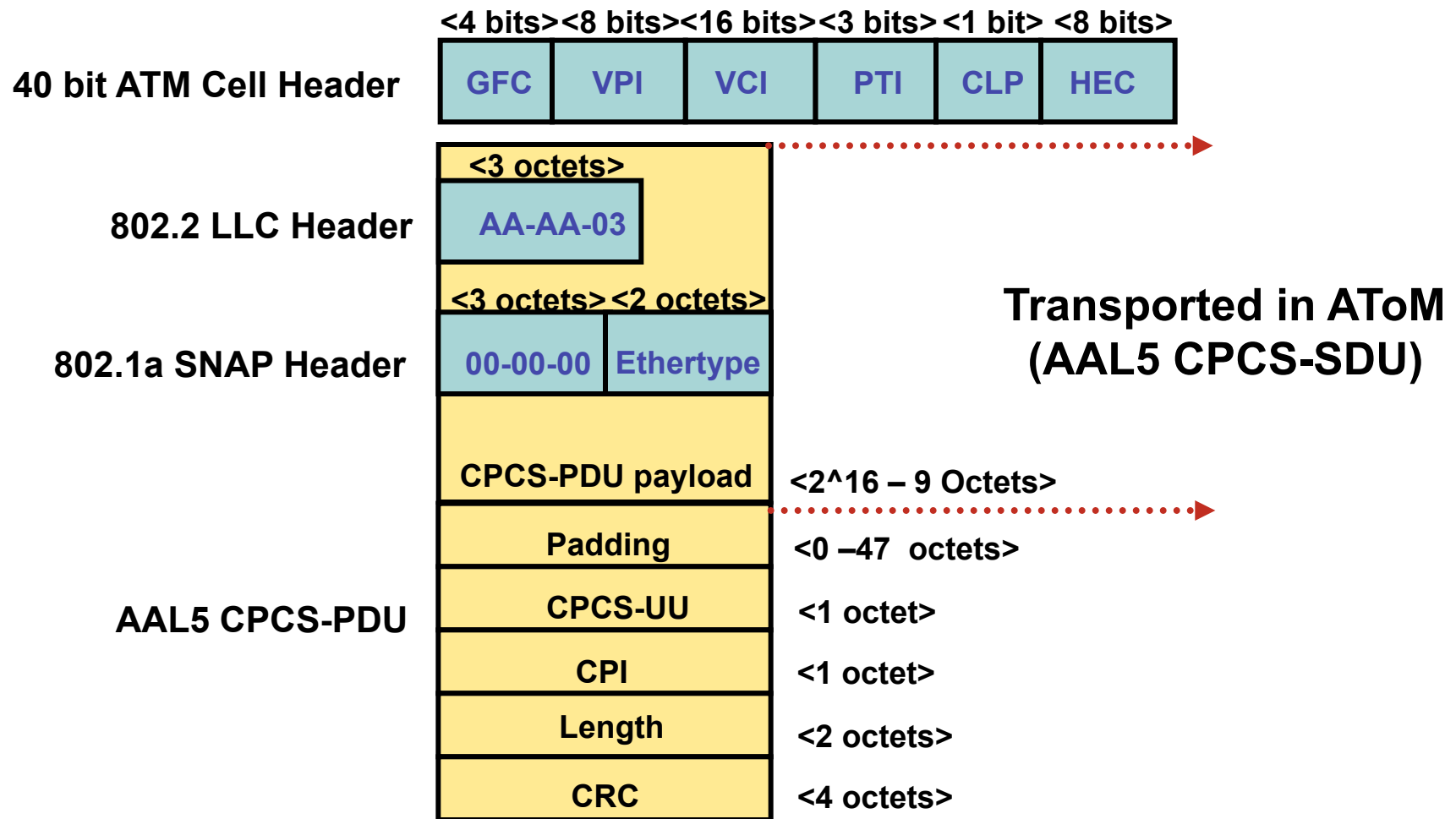
- The control word is **REQUIRED**

But its use is optional

- EFCI and CLP bits are carried within the control word



# ATM AAL5 Frame Format (Routed)



# ATMoMPLS Configuration

```

!
interface ATM3/0.1 point-to-point
 pvc 1/100 l2transport
  encapsulation aal5
  xconnect 192.168.0.6 10 encapsulation mpls
!
interface ATM3/0.3 point-to-point
 pvc 1/300 l2transport
  encapsulation aal0
  xconnect 192.168.0.6 20 encapsulation mpls

```

**AAL5oMPLS**

**ATM\_CelloMPLS**

Local-PE#show mpls l2transport vc

Local intf	Local circuit	Dest address	VC ID	Status
AT3/0.1	<b>ATM AAL5 1/100</b>	192.168.0.6	10	UP
AT3/0.3	<b>ATM VCC CELL 1/300</b>	192.168.0.6	20	UP

**ATM Cell  
VC Mode**

Local-PE#show mpls l2transport vc

Local intf	Local circuit	Dest address	VC ID	Status
AT1/0	<b>ATM CELL ATM1/0</b>	192.168.0.6	100	UP

**ATM Cell  
Port Mode**

# Debug ATMoMPLS AAL5 and AAL0

## Debug signaling [AAL5]:

Nov 1 07:02:50.113 UTC: AToM LDP [192.168.0.23]: Sending label withdraw msg  
**vc type 2**, cbit 1, vc id 10, group id 10, vc label 72, status 0, mtu 4470

## Debug signaling [AAL0]:

Nov 1 07:04:16.321 UTC: AToM LDP [192.168.0.23]: Sending label withdraw msg  
**vc type 9**, cbit 1, vc id 300, group id 10, vc label 74, status 0, mtu 4470

```
Remote-PE#show mpls l2transport vc 10 detail
Local interface: AT3/0.1 up, line protocol up, ATM AAL5 1/100 up
Destination address: 192.168.0.23, VC ID: 10, VC status: up
  Preferred path: not configured
  Default path: active
  Tunnel label: imp-null, next hop point2point
  Output interface: Tu0, imposed label stack {64 16}
  Create time: 1d21h, last status change time: 00:00:17
  Signaling protocol: LDP, peer 192.168.0.23:0 up
  MPLS VC labels: local 16, remote 16
  Group ID: local 10, remote 7
MTU: local 4470, remote 4470
  Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 6, send 7
    byte totals:   receive 672, send 772
    packet drops:  receive 0, send 0
```



# OAM Cell Emulation for AAL5oMPLS

- Locally terminates or loops OAM Cells
- Need to configure in both PEs

```
Router#interface ATM 1/0/0
Router(config-if)#pvc 1/100 l2transport
Router(config-atm-vc)#oam-ac emulation-enable 30
Router(config-atm-vc)#oam-pvc manage
```

```
Router#show atm pvc 1/100
ATM4/1/0.200: VCD: 6, VPI: 5, VCI: 500
UBR, PeakRate: 1
AAL5-LLC/SNAP, etype:0x0, Flags: 0x34000C20, VCmode: 0x0
OAM Cell Emulation: enabled, F5 End2end AIS Xmit frequency: 1 second(s)
OAM frequency: 0 second(s), OAM retry frequency: 1 second(s)
OAM up retry count: 3, OAM down retry count: 5
[snip]
Router#
```

# ATM Cell over MPLS Modes

- Q: How do we select VC vs. VP vs. port mode?
- A: Depending on the configuration mode the `xconnect` command is applied

## VC MODE:

```
interface ATM4/0.300 point-to-point
    pvc 0/300 l2transport
        encapsulation aal0
xconnect 10.0.0.200 300 encapsulation mpls
```

## VP MODE:

```
interface atm 5/0
    atm pvp 1 l2transport
xconnect 10.0.0.200 100 encapsulation mpls
```

## PORT MODE:

```
interface atm 5/0
xconnect 10.0.0.200 100 encapsulation mpls
```

# Packed Cell Relay

- Use the command **`cell-packing`**
- The **`atm mcpt-timers`** indicates the time the PE should wait for cells to be packed into an MPLS packet; three timers are configured
- The command **`cell-packing`** indicates how many cells should be packet per MPLS packet, and which timer to use

```
int atm 1/0
  atm mcpt-timer 500 800 1000
  pvc 1/100 12transport
    encapsulation aal0
  xconnect 10.0.0.1 123 encapsulation mpls
  cell-packing 5 mcpt-timer 1
```

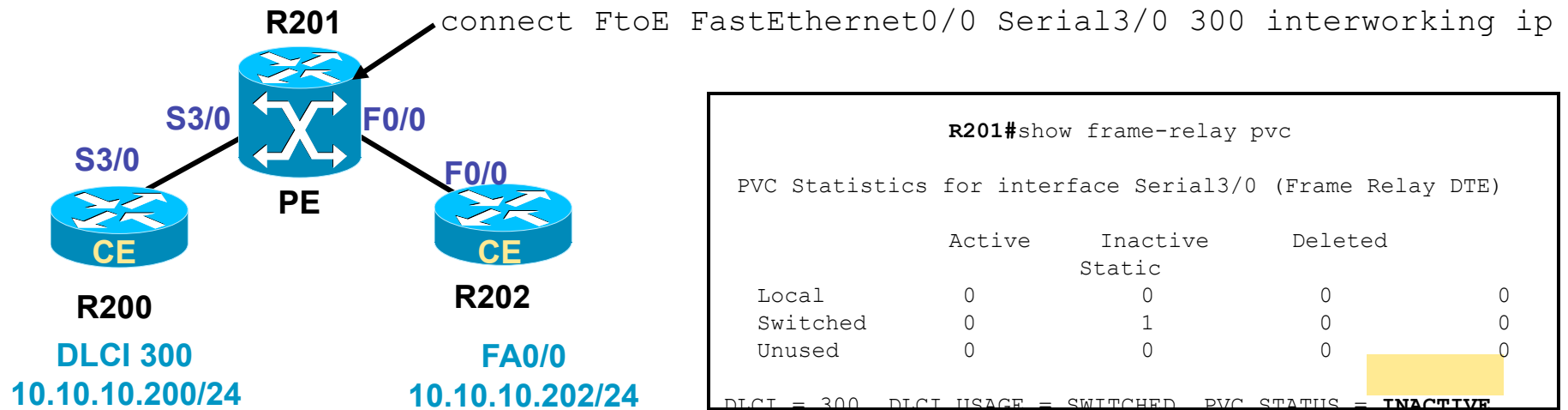


# MPLS L2VPNs: Local Switching

# Local Switching Support

L2 Attachment Type	L2 Attachment Type
Ethernet Port/VLAN	Frame
Ethernet Port/VLAN	ATM
Ethernet Port/VLAN	Ethernet Port/VLAN
ATM	ATM
ATM	Frame

# FR to Ethernet Local Switching Troubleshooting



```

R201#show connection all
INTERFACE = Serial3/0
    
```

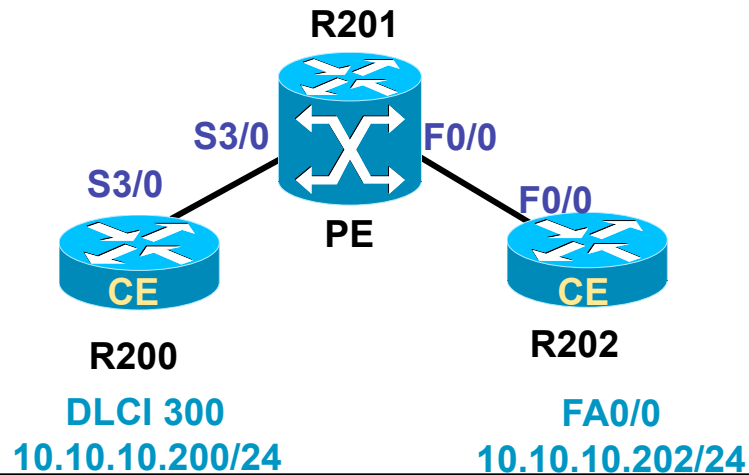
ID	Name	Segment 1	Segment 2	State
2	FtoE	Fa0/0.10	Se3/0 300	<b>DOWN</b>

```

R201#show connection id 1
  Connection: 1 - Frame
  Current State: ADMIN UP
  Segment 1: FastEthernet0/0 up
  Segment 2: Serial3/0 301 down <-- Frame-relay circuit down.
  Interworking Type: ip
    
```

**Frame Relay Link Possible Down**

# Troubleshooting FR Problems for Local Switching



```
R201(config)interface serial 3/0
R201(config)#frame-relay intf-type dce
```

```
R201#show interfaces ser 3/0
Serial3/0 is up, line protocol is up
```

```
R201#show frame-relay pvc
```

```
PVC Statistics for interface Serial3/0 (Frame
Relay DCE)
```

```
DLCI = 300, DLCI USAGE = SWITCHED, PVC STATUS =
INACTIVE, INTERFACE = Serial3/0
```

```
R201#show connection all
```

ID	Name	Segment 1	Segment 2	State
2	FtoE	Fa0/0	Se3/0 300	OPER DOWN

```
R201#debug frame-relay pseudowire
```

```
bad condition when possible Frame-Relay interface down on ce or PE
00:06:09: FRoPW [Se3/0, 100]: acmgr_circuit_down
00:06:09: FRoPW [Se3/0, 100]: Setting pw segment DOWN
00:06:09: FRoPW [Se3/0, 100]: SW AC update circuit state to down
00:07:04: FRoPW [Se3/0, 100]: PW nni_pvc_status set INACTIVE
00:07:04: FRoPW [Se3/0, 100]: Local up, sending acmgr_circuit_up
00:07:04: FRoPW [Se3/0, 100]: Setting pw segment DOWN
00:07:04: FRoPW [Se3/0, 100]: SW AC update circuit state to down
00:07:04: FRoPW [Se3/0, 100]: PW nni_pvc_status set INACTIVE
```

**Frame Relay Link Is Up but OPER DOWN**  
**NOTE: Possible Problem on Ethernet; if Ethernet Is Not Operational, DLCI Status Will Be INACTIVE**

# debug acircuit event

## When You Disable an Interface

**Zero Because Local Switching!!!**

```
R201#debug acircuit event
22:07:30: ACLIB [0.0.0.0, 0]: SW AC interface DOWN for Ethernet interface Fa0/0
22:07:30: ACLIB [0.0.0.0, 0]: Setting new AC state to Ac-Idle
22:07:30: ACLIB: Update switching plane with circuit DOWN status
22:07:30: ACLIB [0.0.0.0, 0]: SW AC interface DOWN for Ethernet interface Fa0/0
22:07:30: Fa0/0 ACMGR: Receive <Circuit Down> msg
22:07:30: Fa0/0 ACMGR: circuit down event, FSP state chg connected to sip up, action is p2p down
22:07:30: Se3/0 ACMGR: Rcv SIP msg: resp peer-to-peer msg, hdl A1000031, sss_hdlA7000032
22:07:30: Se3/0 ACMGR: remote down event, SIP state chg connected to both down, action is
circuit disconnect req
22:07:30: ACLIB: Update switching plane with circuit DOWN status
22:07:30: Se3/0 ACMGR: Receive <Circuit Up> msg
22:07:30: Se3/0 ACMGR: circuit up delayed event, SIP state chg both down to sip up, action is
p2p up forwarding
22:07:30: Fa0/0 ACMGR: Receive <Remote Up Notification> msg
22:07:30: Fa0/0 ACMGR: remote up event, FSP sip up state no chg, action is ignore
22:07:38: ACLIB [0.0.0.0, 0]: Sent ICMP RDP solicitation message
22:07:38: ACLIB: Added circuit to retry queue, type 5, id 3, idb Fa0/0
```

- To display errors and events that occur on the attachment circuits (the circuits between the (PE) and (CE) routers), use the **debug acircuit**



# debug acircuit event

## When You Enable an Interface

**Zero Because Local Switching!!!**

Interface FastEthernet 0/0  
no shutdown

```
22:10:13: ACLIB [0.0.0.0, 0]: SW AC interface UP for Ethernet interface Fa0/0
22:10:13: ACLIB: Added circuit to retry queue, type 5, id 3, idb Fa0/0
22:10:13: ACLIB [0.0.0.0, 0]: pthru_intf_handle_circuit_up() calling acmgr_circuit_up
22:10:13: ACLIB [0.0.0.0, 0]: Setting new AC state to Ac-Connecting
22:10:13: ACLIB: Update switching plane with circuit UP status
22:10:13: ACLIB [0.0.0.0, 0]: SW AC interface UP for Ethernet interface Fa0/0
22:10:13: ACLIB [0.0.0.0, 0]: pthru_intf_handle_circuit_up() ignoring up event.
Already connected or connecting.
22:10:13: ACLIB [0.0.0.0, 0]: pthru_intf_handle_circuit_up() ignoring up event.
Already connected or connecting.
22:10:13: Fa0/0 ACMGR: Receive <Circuit Up> msg
22:10:13: Fa0/0 ACMGR: circuit up event, FSP state chg sip up to connected, action is send
connected msg
22:10:13: ACLIB: pthru_intf_response hdl is 57000035, response is 2
22:10:13: ACLIB [0.0.0.0, 0]: Setting new AC state to Ac-Connected
22:10:13: Se3/0 ACMGR: Rcv SIP msg: resp peer-to-peer msg, hdl A1000031, sss_hdl A7000032
22:10:13: Se3/0 ACMGR: remote up event, SIP state chg sip up to connected, action is
respond forwarded
22:10:13: ACLIB: Update switching plane with circuit UP status
22:10:15: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
22:10:16: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
22:10:23: ACLIB [0.0.0.0, 0]: Sent ICMP RDP solicitation message
22:10:23: ACLIB: Added circuit to retry queue, type 5, id 3, idb Fa0/0
```

- Problem on FE administratively down

## Let's Do a Poll...

- Would MTU mismatches prevent a circuit to come up in the “local switching” case?

## Let's Do a Poll...

- Would MTU mismatches prevent a circuit to come up in the “local switching” case?
- NO, because in the “local switching” case there is no signaling

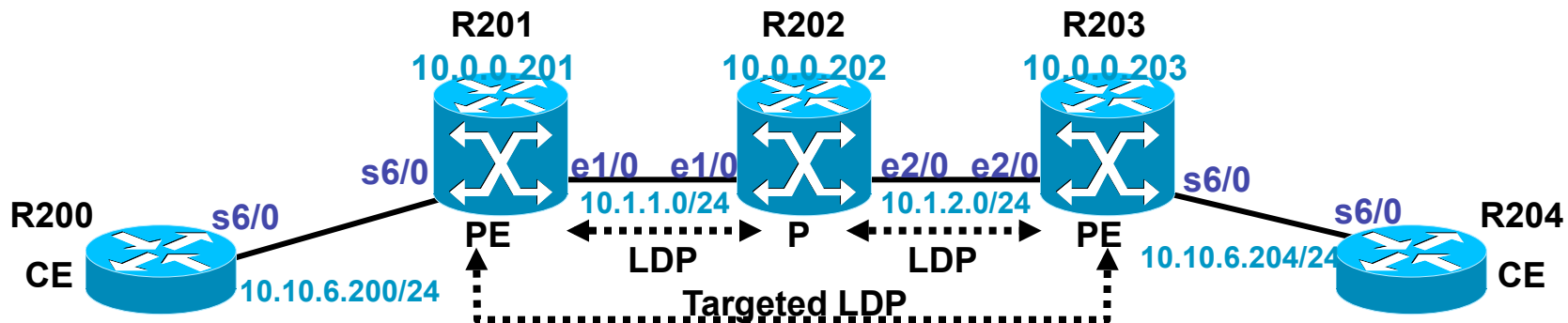
# Summary of Useful Debugs

- debug connection
- show connection all
- debug xconnect event
- debug xconnect error
- debug frame-relay pseudowire
- debug acircuit event
- debug acircuit error
- show frame-relay pvc
- show frame-relay lmi



# MPLS L2VPNs: HDLC and PPP over MPLS

# Configuration: PPPoMPLS



- Configuration is the same as EoMPLS changing the `encapsulation`

## R201

```
interface Serial6/0
  no ip address
  no ip directed-broadcast
  encapsulation ppp
  no cdp enable
  xconnect 10.0.0.203 60 encapsulation mpls
```

## R203

```
interface Serial6/0
  no ip address
  no ip directed-broadcast
  encapsulation ppp
  no cdp enable
  xconnect 10.0.0.201 60 encapsulation mpls
```

# Cisco's HDLC

- It's proprietary because:

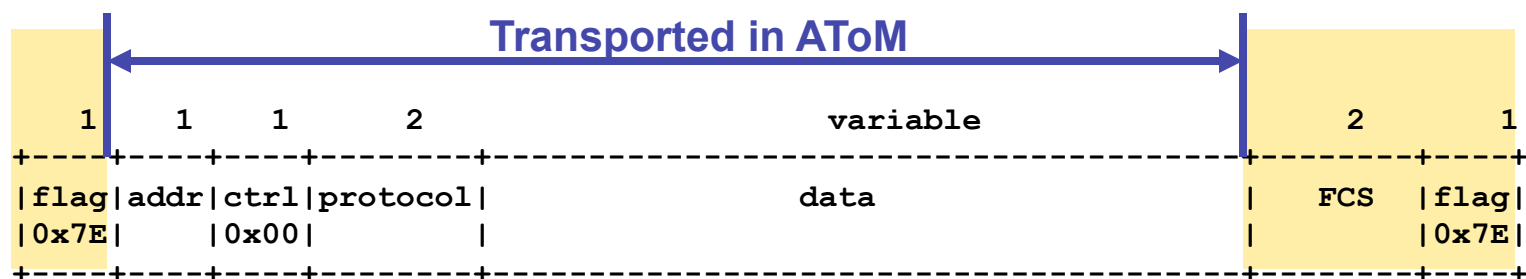
- It does not perform windowing or retransmission

- Higher layer protocol identification method is not standardized

- Its frame format and bit stuffing technique are per the ANSI T1.618 standard

- It doesn't matter for AToM, since AToM only checks flag 0x7E and FCS!!!

# Cisco's HDLC: Frame Format



flag = start/end of frame = 0x7E  
 (Other special characters: Idle = 0xFF, Abort = 0x7F)

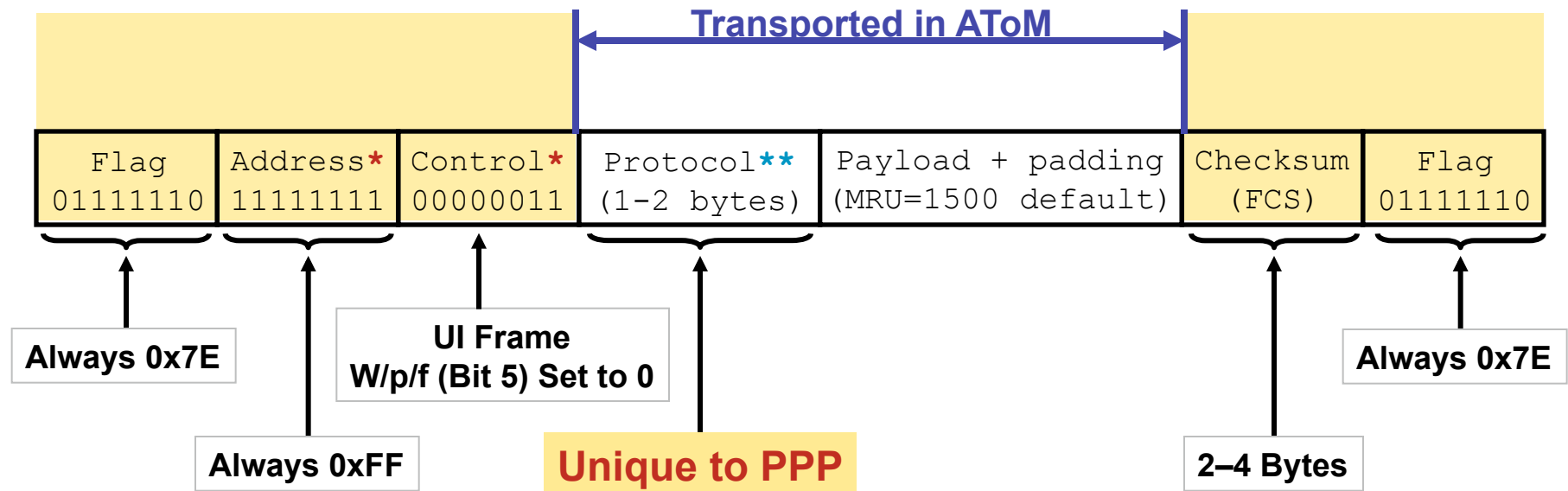
address = this is really a frame type field  
 0x0F = Unicast Frame  
 0x80 = Broadcast Frame (used if upper layer packet is a broadcast)  
 0x40 = Padded Frame  
 0x20 = Compressed Frame

Protocol = the Ethernet type of the encapsulated data:

0x0800 IP  
 0x4242 DSAP/SSAP for IEEE bridge spanning protocol  
 0x6003 DECnet phase IV  
 0x6558 Bridged Ethernet/802.3 packet  
 0x8035 cisco SLARP  
 0x8038 DEC bridge spanning tree protocol  
 0x809b Apple EtherTalk  
 0x80f3 Appletalk ARP  
 0x8137 Novell IPX  
 0xFEFE ISO CLNP/ISO ES-IS DSAP/SSAP



# PPP: Frame Format



- Standard method for transporting **multiprotocol** datagrams over point-to-point links
- Modeled after HDLC frame w/addition of protocol field
- PPP frame size ranges from 10 bytes to 4 bytes (when PFC and ACFC compression are enabled, and w/o ending flag)
- Ending flag only needed on single frame or final frame of a sequence

**\*Omitted When Address and Control Field Compression (ACFC) Is Used**

**\*\*Only 1 Byte When Protocol Field Compression (PFC) Is Used**

# HDLC and PPP over MPLS

- HDLC

- The whole packet is transported

- HDLC flags and Frame Check Sequence (FCS) bits are removed at ingress

- Point-to-Point Protocol

- Exclude media-specific framing information: The ingress PE router removes the flags, address, control field, and the FCS

- PPP PDU is transported in its entirety, including the protocol field (whether compressed using PFC or not)

- Consequences: The following will not work (unless using HDLCoMPLS ;^):

- Frame Check Sequence (FCS) Alternatives

- Address-and-Control-Field-Compression (ACFC)

- Asynchronous-Control-Character-Map (ACCM)

- Captured using `debug mpls l2transport packet data`

```
*Apr 22 15:16:09.703: XX XX XX XX XX XX YY YY YY YY YY YY 88 47 00 01
                        ^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^^ ^^^^^ ^^^^^
                        SA MAC                DA MAC                etype top_shim-->
                                                MPLS
                                                Unic
```

```
*Apr 22 15:16:09.703: 10 FF 00 01 11 02 00 00 00 0F 00 08 00 45 C0
                        ^^^^^^ ^^^^^^^^^^^^^^^^^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^...
<--top_shim VC_Label      Ctrl-word      |   |   |           Begins IP Packet
          Label=17 Label=17                |   |   eetype = IPv4
              S=0       S=1                 |   Control
        TTL=255  TTL=2                    Address = Unicast Frame
```

```
*Apr 22 15:16:09.703: 00 50 CD 2C 00 00 01 59 FB 91 0A 0A 05 C8 E0 00
*Apr 22 15:16:09.703: 00 05 02 01 00 30 0A 0A 0B C8 00 00 00 00 C0 F2
*Apr 22 15:16:09.703: 00 00 00 00 00 00 00 00 00 FF FF FF 00 00 0A
*Apr 22 15:16:09.703: 12 01 00 00 00 28 00 00 00 00 00 00 00 00 0A 0A
    *Apr 22 15:16:09.703: 0B CC FF F6 00 03 00 01 00 04 00 00 00 01
```

# Decoding an HDLCoMPLS Packet

- Captured using `debug mpls l2transport packet data`

```
*Apr 22 15:16:10.179: ATOM disposition: in Et1/0, size 136, seq 0, control word 0x0
  *Apr 22 15:16:10.179: 0F 00 08 00 45 C0 00 84 CC 3D 00 00 01 59 FC 48
                        ^^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ...
                        | | |           Begins IP Packet
                        | | etype = IPv4
                        | Control
                        Address = Unicast Frame

*Apr 22 15:16:10.179: 0A 0A 05 CC E0 00 00 05 02 04 00 70 0A 0A 0B CC
*Apr 22 15:16:10.179: 00 00 00 00 C1 D9 00 00 00 00 00 00 00 00 00 00
*Apr 22 15:16:10.179: 00 00 00 01 00 01 22 01 0A 0A 0B CC 0A 0A 0B CC
*Apr 22 15:16:10.179: 80 00 00 C6 CF D8 00 54 00 00 00 05 0A 0A 0B CC
*Apr 22 15:16:10.179: FF FF FF FF 03 00 00 01 0A 0A 0B C8 0A 0A 05 CC
*Apr 22 15:16:10.179: 01 00 00 40 0A 0A 05 00 FF FF FF 00 03 00 00 40
*Apr 22 15:16:10.179: 0A 0A 14 00 FF FF FF 00 03 00 00 0A 0A 0A 0A 00
  *Apr 22 15:16:10.179: FF FF FF 00 03 00 00 0A
```

- At disposition, only AToM PDU is displayed

# PPPoMPLS Notes

- Q: Does PPP run on the PE? Let's `debug ppp negotiation`
- A: No—once the `xconnect` command is entered, PPP is closed; no LCP or NCP negotiation with CE

```
R201(config-if)# xconnect 10.0.0.203 60 encapsulation mpls
*Apr 22 17:16:35.583: Se6/0 LCP: State is Closed
*Apr 22 17:16:35.583: Se6/0 PPP: Phase is DOWN
*Apr 22 17:16:35.583: Se6/0 PPP: Phase is ESTABLISHING, Passive Open
*Apr 22 17:16:35.583: Se6/0 LCP: State is Listen
*Apr 22 17:16:35.583: Se6/0 LCP: State is Closed
*Apr 22 17:16:35.583: Se6/0 PPP: Phase is DOWN
*Apr 22 17:16:36.631: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Serial6/0,
changed state to up
```

```
R201(config-if)#do sh int ser 6/0
Serial6/0 is up, line protocol is up
Hardware is HD64570
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
Encapsulation PPP, loopback not set
Keepalive set (10 sec)
Last input 00:03:10, output 00:00:06, output hang never
```

# Show Interface from the CE

- Note that PPP is up and running

```
R200#show interfaces serial 6/0
Serial6/0 is up, line protocol is up
  Hardware is HD64570
  Internet address is 10.10.6.200/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation PPP, loopback not set
  Keepalive set (10 sec)
  LCP Open
  Open: OSICP, IPCP, CDPCP
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters never
```

# Decoding an PPPoMPLS Packet

- Captured using `debug mpls l2transport packet data`

```
*Apr 22 17:31:13.163: ATOM imposition: out Et1/0, size 108, EXP 0x0, seq 0, control word 0x0
*Apr 22 17:31:13.163: XX XX XX XX XX XX YY YY YY YY YY YY 88 47 00 01
                        ^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^^ ^^^^^ ^^^^^
                        SA MAC              DA MAC              |      top_shim-->
                                                etype = MPLS Unicast

*Apr 22 17:31:13.163: 10 FF 00 01 41 02 00 00 00 00 00 21 45 C0 00 50
                        ^^^^^ ^^^^^^^^^^^^^ ^^^^^^^^^^^^^ ^^^^^ ^^^^^...
                        <--top_shim VC_Label   Ctrl-word   |      Begins IP Packet
                        Label=17 Label=20         PPP DLL Protocol# = IPv4
                                                S=0          S=1
                                                TTL=255     TTL=2

*Apr 22 17:31:13.163: 09 6E 00 00 01 59 BE 50 0A 0A 06 C8 E0 00 00 05
*Apr 22 17:31:13.163: 02 01 00 30 0A 0A 0B C8 00 00 00 00 C0 F2 00 00
*Apr 22 17:31:13.163: 00 00 00 00 00 00 00 00 FF FF FF 00 00 0A 12 01
*Apr 22 17:31:13.163: 00 00 00 28 00 00 00 00 00 00 00 00 00 0A 0A 0B CC
*Apr 22 17:31:13.163: FF F6 00 03 00 01 00 04 00 00 00 00 01
```

# Decoding an PPPoMPLS Packet

- Captured using `debug mpls l2transport packet data`

```
*Apr 22 17:31:13.163: ATOM disposition: in Et1/0, size 82, seq 0, control word 0x0
  *Apr 22 17:31:13.163: 00 21 45 C0 00 50 09 4C 00 00 01 59 BE 6E 0A 0A
                        ^^^^^ ^^^^^...
                        |      Begins IP Packet
                        PPP DLL Protocol # = IPv4

  *Apr 22 17:31:13.163: 06 CC E0 00 00 05 02 01 00 30 0A 0A 0B CC 00 00
  *Apr 22 17:31:13.163: 00 00 C0 F2 00 00 00 00 00 00 00 00 00 00 FF FF
  *Apr 22 17:31:13.163: FF 00 00 0A 12 01 00 00 00 28 00 00 00 00 00 00
  *Apr 22 17:31:13.163: 00 00 0A 0A 0B C8 FF F6 00 03 00 01 00 04 00 00
                        *Apr 22 17:31:13.163: 00 01
```

- At disposition, only AToM PDU is displayed





# MPLS: Traffic Engineering

# Agenda

- How MPLS-TE Works
- Basic Configuration
- Knobs! Knobs! Knobs!
- Configuring and Monitoring

# How MPLS-TE Works

- How MPLS-TE Works

- What good is MPLS-TE?

- Information Distribution

- Path Calculation

- Path Setup

- Forwarding Traffic Down A Tunnel

# What Good Is MPLS-TE?

- There are two kinds of networks
  1. Those that have plenty of bandwidth everywhere
  2. Those with congestion in some places, but not in others
  3. Those with constant congestion everywhere
- The first kind *always* evolves into the second kind!

# What Good Is MPLS-TE?

- MPLS-TE introduces a 4<sup>th</sup> kind:

Those that have plenty of bandwidth everywhere

Those with congestion in some places, but not in others

Those with constant congestion everywhere

Those that use all of their bandwidth to its maximum efficiency, regardless of shortest-path routing!

- MPLS-TE can help turn #2 into #4.

If you have #1, you probably don't need MPLS-TE  
– *yet*.

If you have #3, you're stuck – you either need more bandwidth (or less traffic).

# Information Distribution

- You need a link-state protocol as your IGP  
IS-IS or OSPF
- Link-state requirement is *only* for MPLS-TE!  
*Not* a requirement for VPNs, etc!

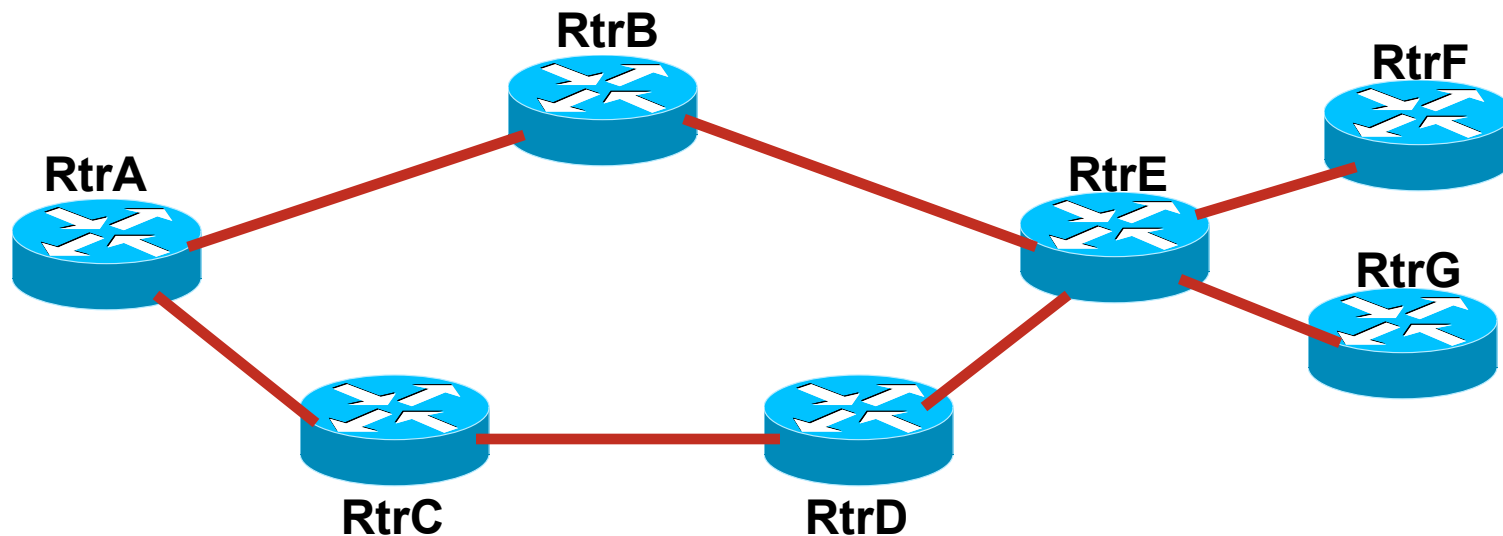
# Need for a Link-State Protocol

- Why do I need a link-state protocol?
  1. To make sure info gets flooded
  2. To build a picture of the entire network

# Need for a Link-State Protocol

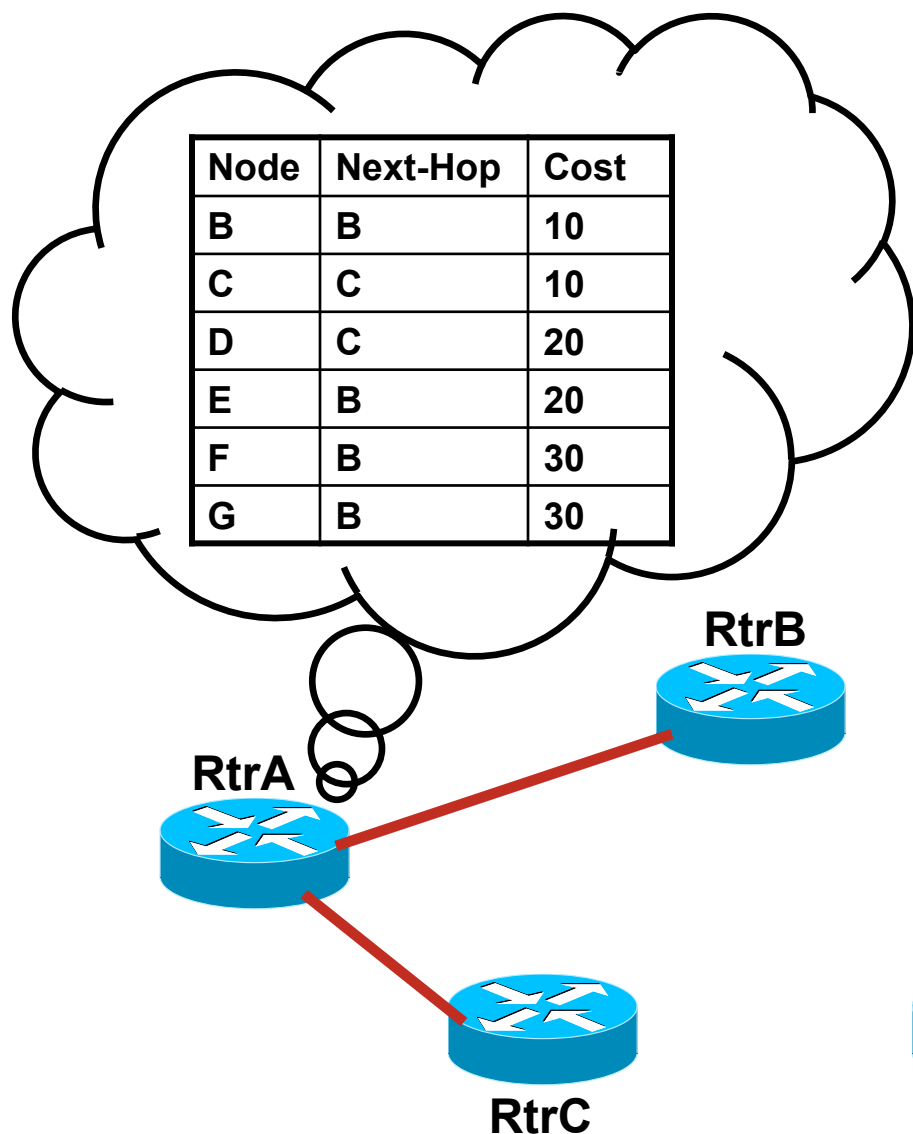
Consider the following network:

- All links have a cost of 10
- RtrA's path to RtrE is A->B->E, cost 20
- All traffic from A to {E,F,G} goes A->B->E



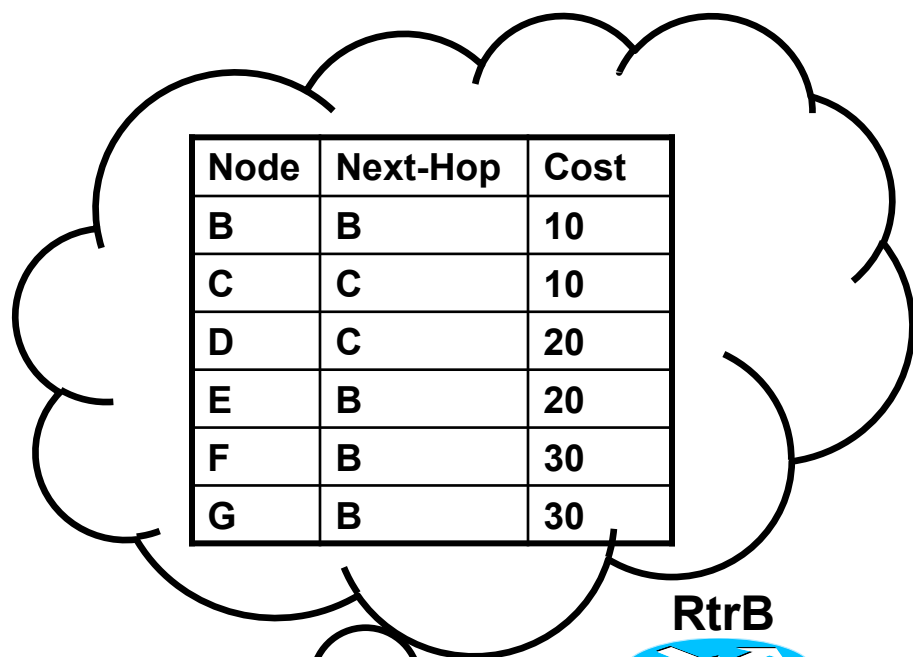


## What a DV Protocol Sees



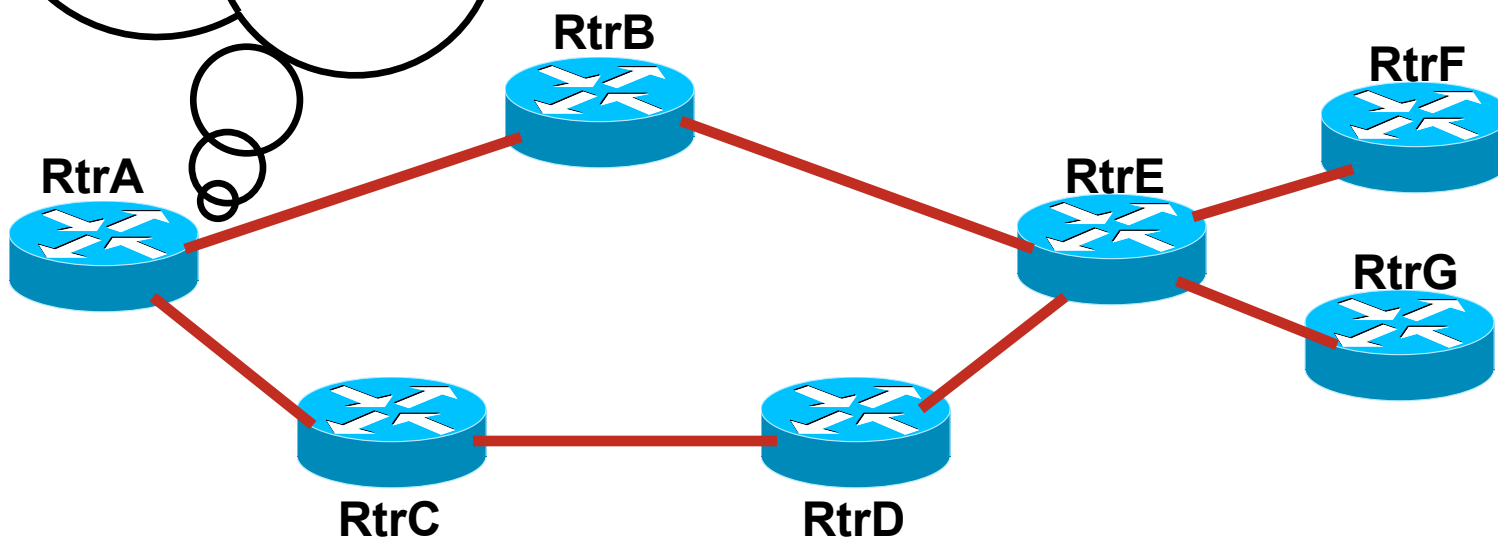
- RtrA doesn't see all the links
- RtrA only knows about the shortest path
- This is by design

## What a LS Protocol Sees

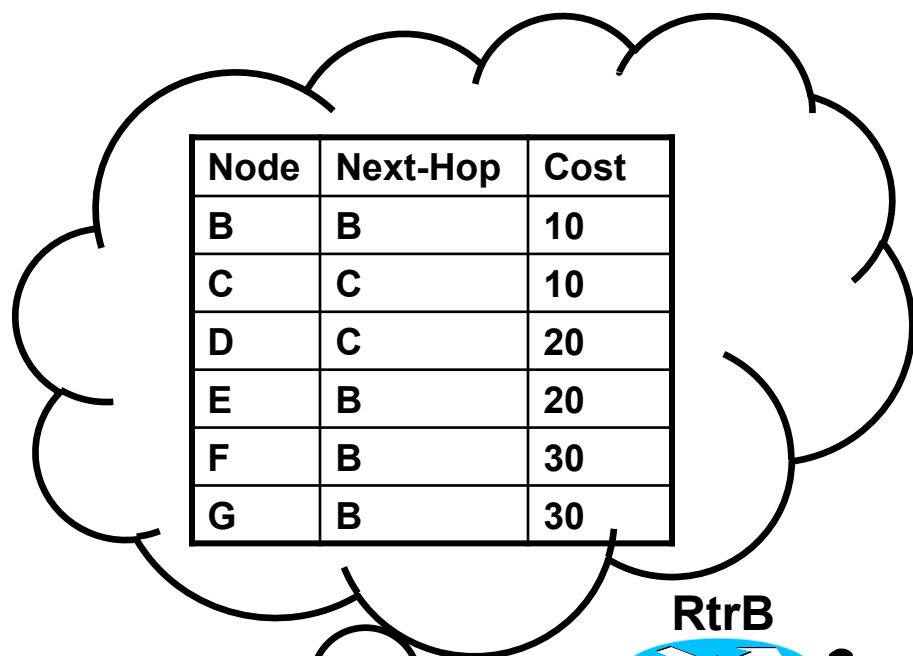


Node	Next-Hop	Cost
B	B	10
C	C	10
D	C	20
E	B	20
F	B	30
G	B	30

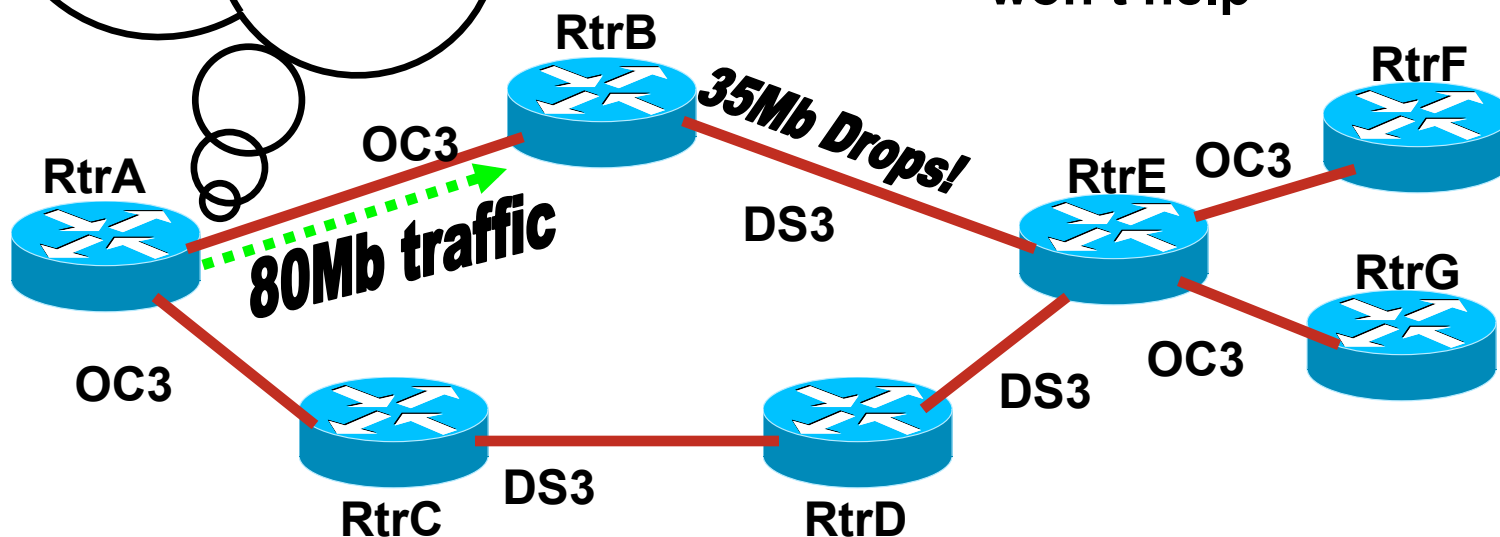
- RtrA sees all links
- RtrA only *computes* the shortest path
- Routing table doesn't change



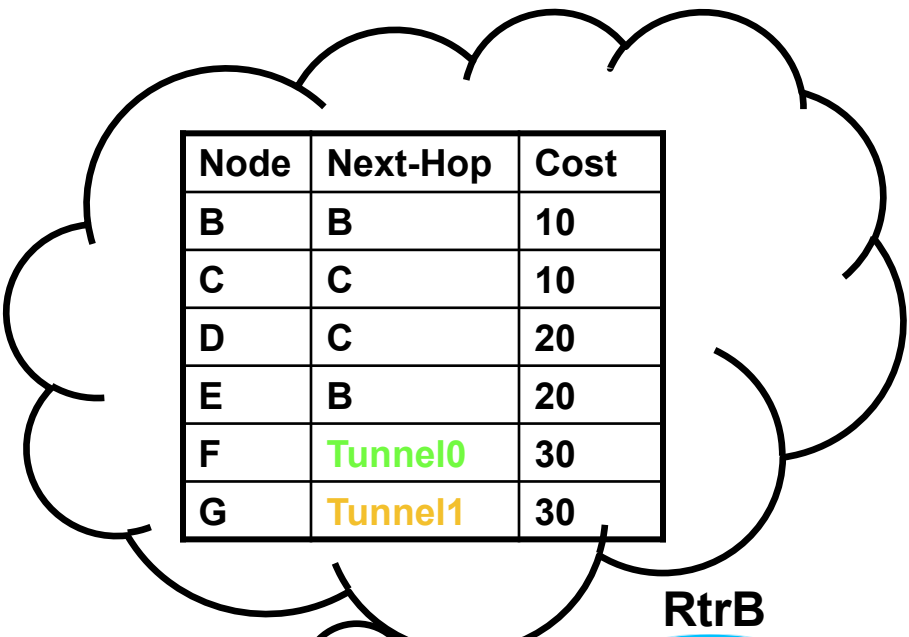
# The Problem With Shortest-Path



- Some links are DS3, some are OC3
  - RtrA has 40Mb of traffic for RtrF, 40Mb of traffic for RtrG
  - Massive (44%) packet loss at RtrB->RtrE!
  - Changing to A->C->D->E won't help

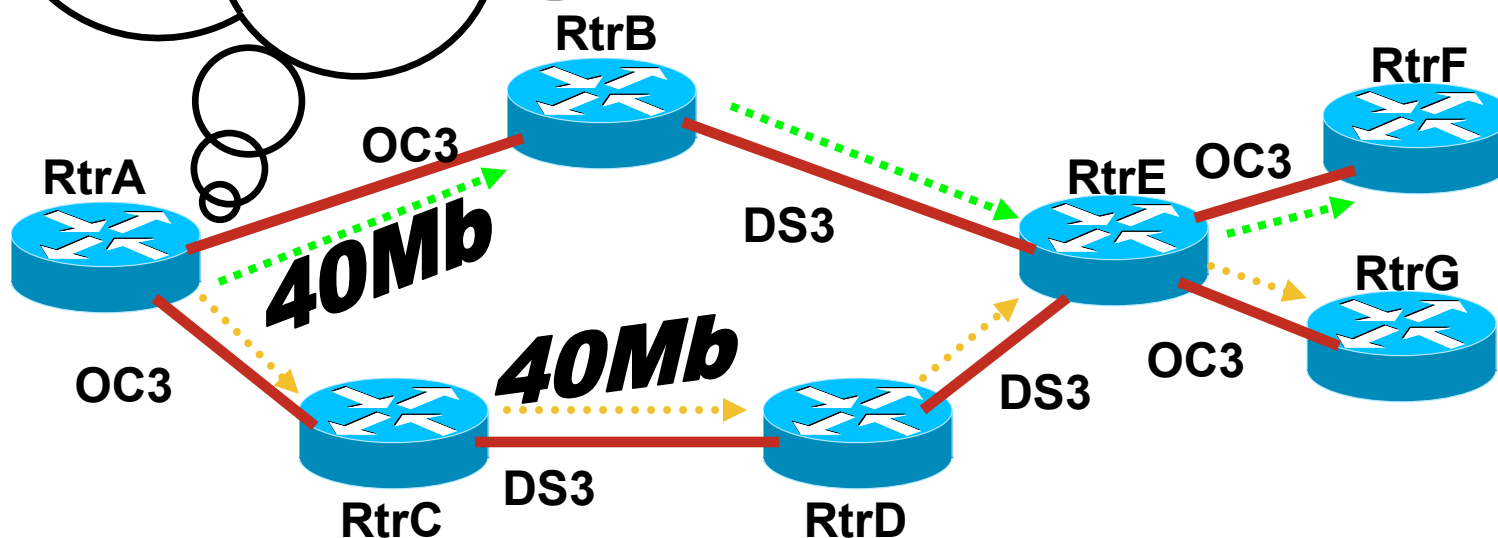


# What MPLS-TE Addr

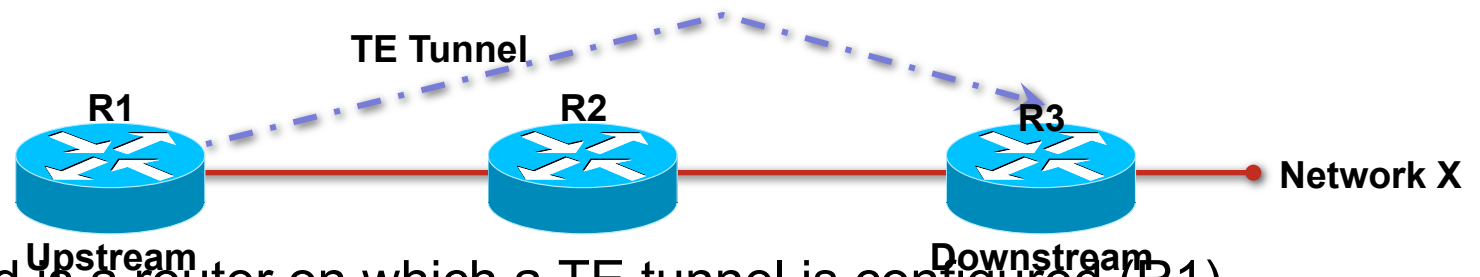


Node	Next-Hop	Cost
B	B	10
C	C	10
D	C	20
E	B	20
F	Tunnel0	30
G	Tunnel1	30

- RtrA sees all links
- RtrA computes paths on properties other than just shortest cost
- No link oversubscribed!



## A Terminology Slide—Head, Tail, LSP, etc.



- Head-End is a router on which a TE tunnel is configured (R1)
- Tail-End is the router on which TE tunnel terminates (R3)
- Mid-point is a router thru which the TE tunnel passes (R2)
- LSP is the Label Switched Path taken by the TE tunnel, here R1-R2-R3
- Downstream router is a router closer to the tunnel tail
- Upstream router is farther from the tunnel tail (so R2 is upstream to R3's downstream, R1 is upstream from R2's downstream)

# How MPLS-TE Works

- How MPLS-TE Works

- What good is MPLS-TE?

- Information Distribution

- Path Calculation

- Path Setup

- Forwarding Traffic Down A Tunnel

# Information Distribution

- OSPF
  - Uses Type 10 (Opaque Area-Local) LSAs
  - See draft-katz-yeung-ospf-traffic

# Information Distribution

- IS-IS
  - Uses Type 22 TLVs
  - See draft-ietf-isis-traffic



# Information Distribution

- IS-IS and OSPF propagate the same information!
  - Link identification
  - TE Metric
  - Bandwidth info (max physical, max reservable, available per-class)
  - Attribute flags

# Information Distribution

- TE flooding is local to a single {area|level}
- Inter-{area|level} TE harder, but possible (think PNNI)

# Information Distribution

- Dynamics of ISIS and OSPF are unchanged
  - Periodic flooding
  - Hold-down timer to constrain the frequency of advertisements
- Current constraint information sent when IGP decides to re-flood
- TE admission control requests re-flooding on significant changes
  - *significant* is determined by a configurable set of thresholds
  - On link configuration changes
  - On link state changes
  - On LSP Setup failure
  - TE refresh timer expires (180 seconds default)

# How MPLS-TE Works

- How MPLS-TE Works

- What good is MPLS-TE?

- Information Distribution

- Path Calculation

- Path Setup

- Forwarding Traffic Down A Tunnel

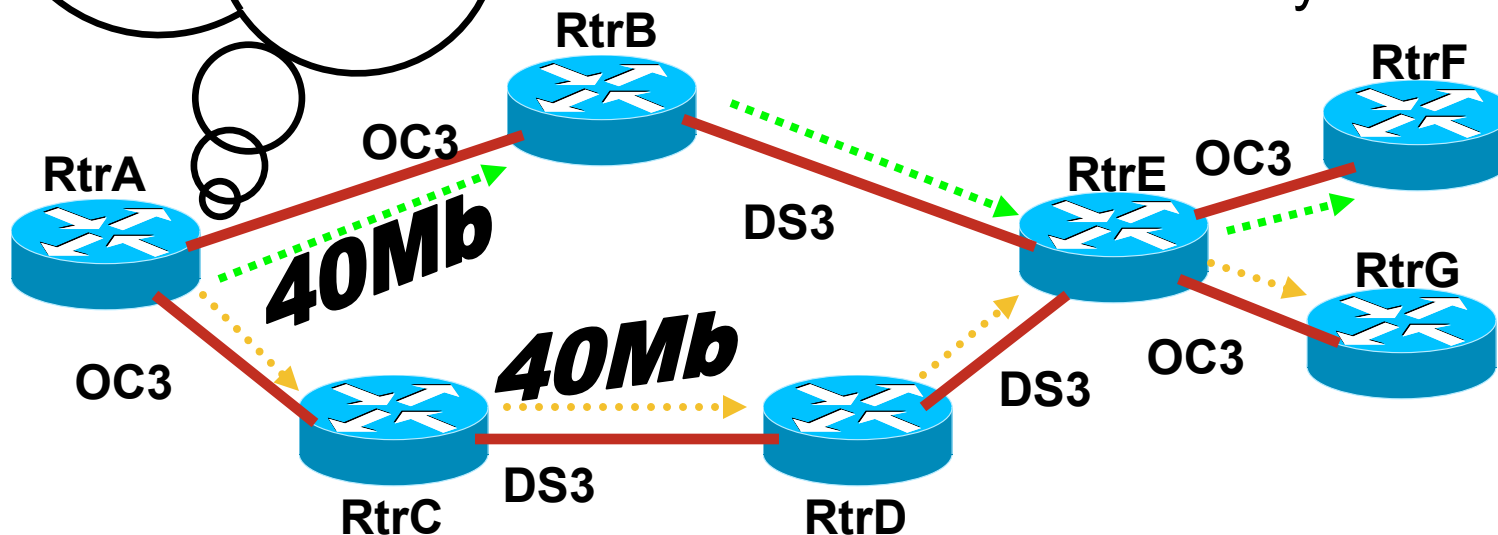
# Path Calculation

- Modified Dijkstra at tunnel head-end
- Often referred to as CSPF  
Constrained SPF
- ...or PCALC (path calculation)

# Path Calculation

Node	Next-Hop	Cost
B	B	10
C	C	10
D	C	20
E	B	20
F	Tunnel0	30
G	Tunnel1	30

- PCALC takes bandwidth, other constraints into account
- Paths calculated, resources reserved if necessary
- End result: bandwidth used more efficiently!



# Path Calculation (C-SPF)

- Shortest-cost path is found that meets administrative constraints
- These constraints can be
  - bandwidth
  - link attribute (aka color, resource group)
  - priority
- The addition of constraints is what allows MPLS-TE to use paths other than *just* the shortest one

# Path Computation

“On demand” by the trunk’s head-end:

- for a new trunk

- for an existing trunk whose (current) LSP failed

- for an existing trunk when doing re-optimization



# Path Computation

## Input:

- configured attributes of traffic trunks originated at this router

- attributes associated with resources

  - available from IS-IS or OSPF

- topology state information

  - available from IS-IS or OSPF

# Path Computation

- Prune links if:
  - insufficient resources (e.g., bandwidth)
  - violates policy constraints
- Compute shortest distance path
  - TE uses its own metric
- Tie-break:
  1. Path with the highest available bandwidth
  2. Path with the smallest hop-count
  3. Path found first in TE topology database

# Path Computation

## Output:

- explicit route - expressed as a sequence of router IP addresses

- interface addresses for numbered links

- loopback address for unnumbered links

- used as an input to the path setup component

# Path Calculation

- What if there's more than one path that meets the minimum requirements (BW, etc)?
- PCALC algorithm:
  1. find all paths with the lowest IGP cost
  2. then pick the path with the highest minimum bandwidth along the path
  3. then pick the path with the lowest hop count (not IGP cost, but hop count)
  4. then just pick one path at random

# How MPLS-TE Works

- How MPLS-TE Works

- What good is MPLS-TE?

- Information Distribution

- Path Calculation

- Path Setup

- Forwarding Traffic Down A Tunnel

# Path Setup

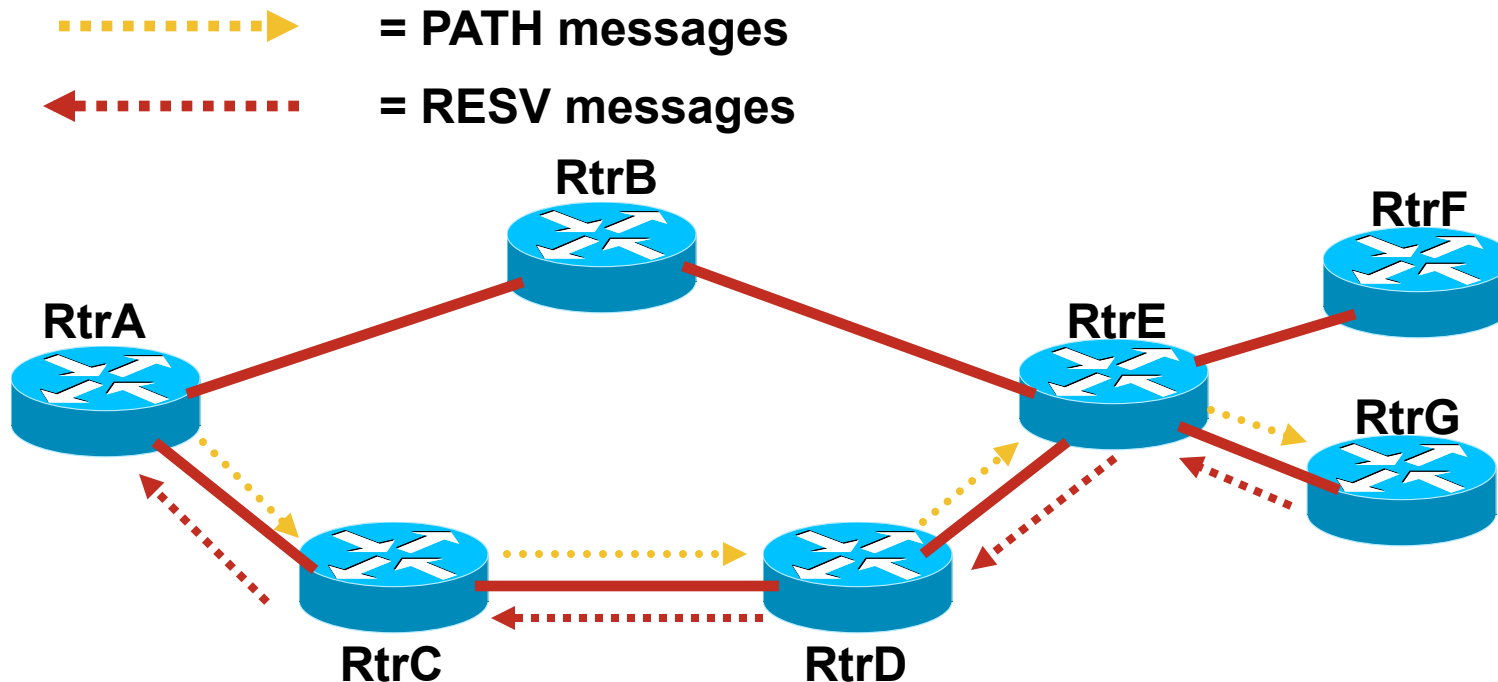
- Cisco MPLS-TE uses RSVP
- RFC2205, plus draft-ietf-mpls-rsvp-lsp-tunnel

# Path Setup

- Once the path is calculated, it is handed to RSVP
- RSVP uses PATH and RESV messages to request an LSP along the calculated path

## Path Setup

- PATH message: “Can I have 40Mb along this path?”
- RESV message: “Yes, and here’s the label to use.”
- LFIB is set up along each hop

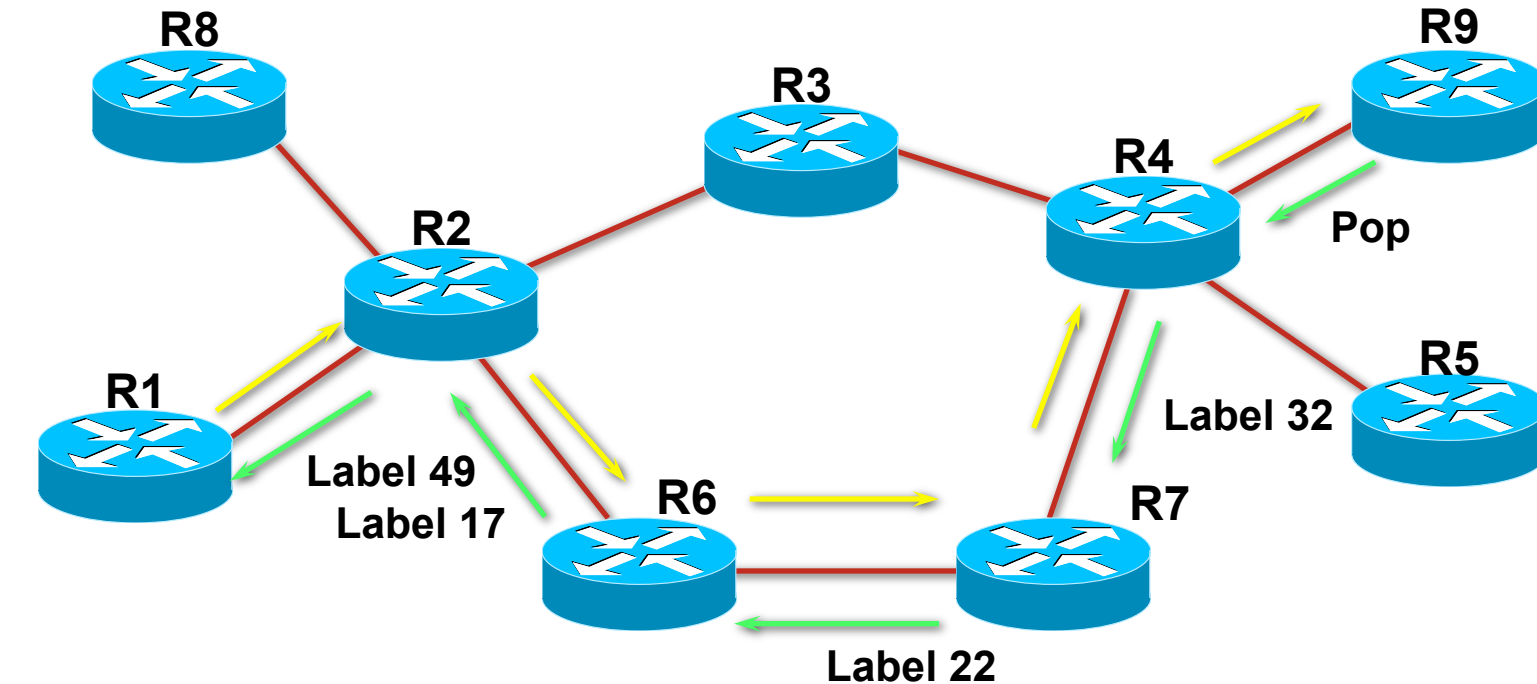




# Path Setup

- Errors along the way will trigger RSVP errors
- May also trigger re-flooding of TE info if appropriate

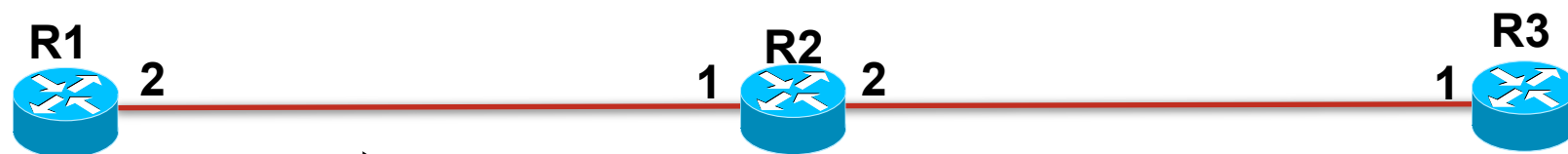
# Path Setup - Example



**Setup: Path (ERO = R1->R2->R6->R7->R4->R9)**

**Reply: Resv communicates labels and reserves bandwidth on each link**

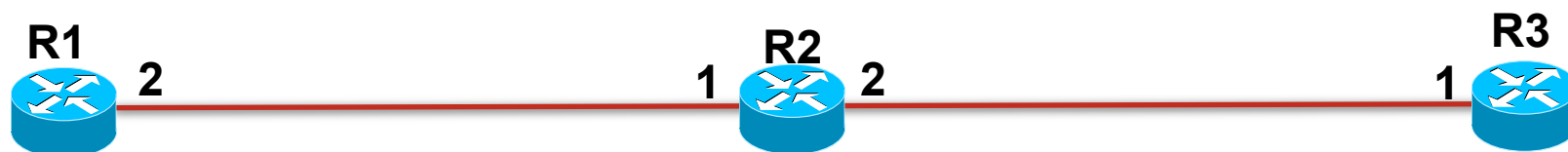
## Path Setup - more details



Path:

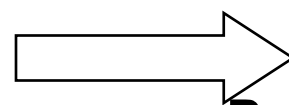
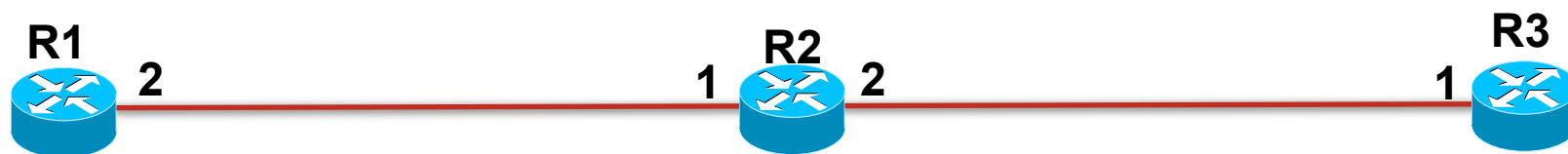
Common\_Header  
Session(**R3-1o0**, 0, **R1-1o0**)  
PHOP(**R1-2**)  
Label\_Request(**IP**)  
ERO (**R2-1**, **R3-1**)  
Session\_Attribute (**S(3)**, **H(3)**, **0x04**)  
Sender\_Template(**R1-1o0**, **00**)  
Sender\_Tspec(**2Mbps**)  
Record\_Route(**R1-2**)

## Path Setup - more details



**Path State:**  
**Session(R3-Io0, 0, R1-Io0)**  
**PHOP(R1-2)**  
**Label\_Request(IP)**  
**ERO (R2-1, R3-1)**  
**Session\_Attribute (S(3), H(3), 0x04)**  
**Sender\_Template(R1-Io0, 00)**  
**Sender\_Tspec(2Mbps)**  
**Record\_Route (R1-2)**

## Path Setup - more details



Path:

Common\_Header

Session(**R3-Lo0**, 0, **R1-Lo0**)

PHOP(**R2-2**)

Label\_Request(**IP**)

ERO (**R3-1**)

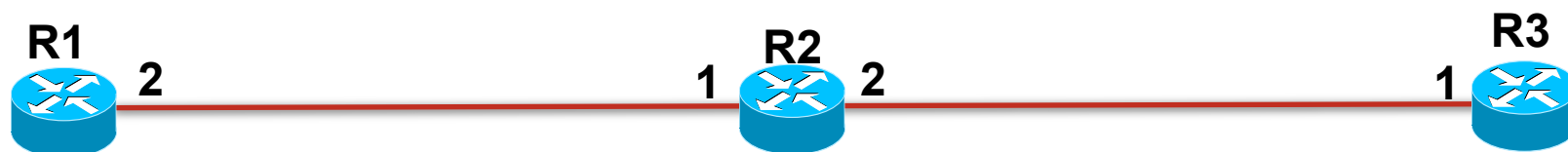
Session\_Attribute (**S(3)**, **H(3)**, **0x04**)

Sender\_Template(**R1-Lo0**, **00**)

Sender\_Tspec(**2Mbps**)

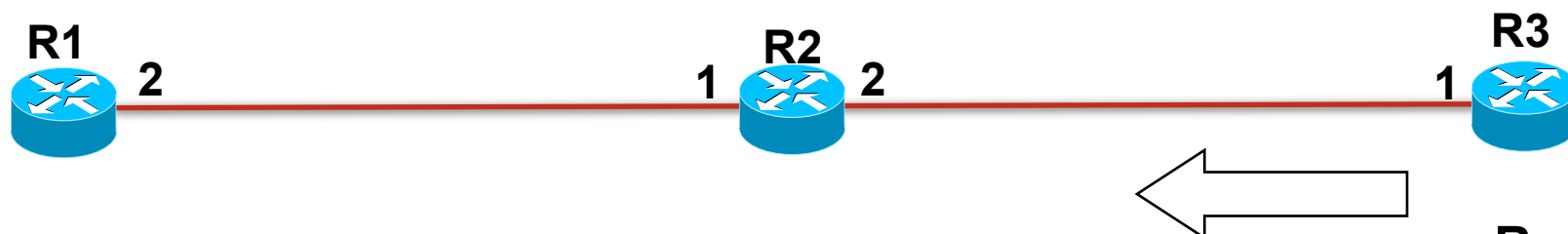
Record\_Route (**R1-2**, **R2-2**)

## Path Setup - more details



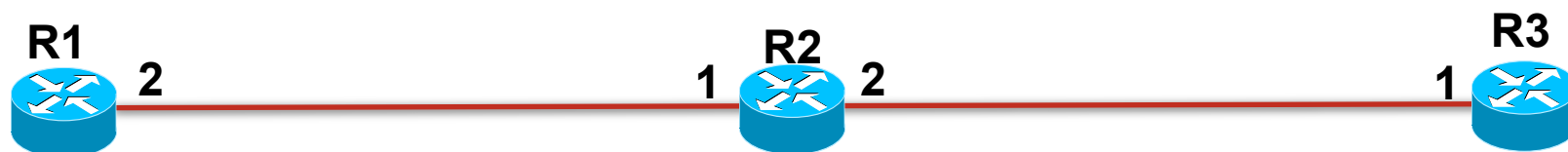
**Path State:**  
**Session(R3-Io0, 0, R1-Io0)**  
**PHOP(R2-2)**  
**Label\_Request(IP)**  
**ERO ()**  
**Session\_Attribute (S(3), H(3), 0x04)**  
**Sender\_Template(R1-Io0, 00)**  
**Sender\_Tspec(2Mbps)**  
**Record\_Route (R1-2, R2-2, R3-1)**

## Path Setup - more details



**Resv:**  
**Common\_Header**  
**Session(R3-Io0, 0, R1-Io0)**  
**PHOP(R3-1)**  
**Style=SE**  
**FlowSpec(2Mbps)**  
**Sender\_Template(R1-Io0, 00)**  
**Label=POP**  
**Record\_Route(R3-1)**

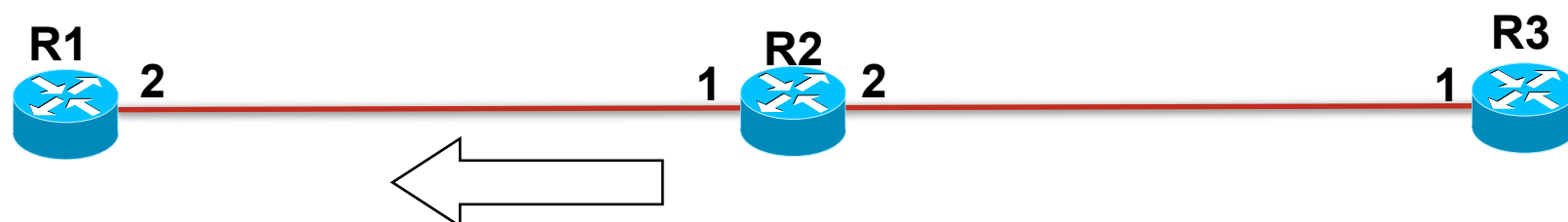
## Path Setup - more details



```
Resv State
Session(R3-Io0, 0, R1-Io0)
PHOP(R3-1)
Style=SE
FlowSpec (2Mbps)
Sender_Template(R1-Io0, 00)
OutLabel=POP
IntLabel=5
Record_Route(R3-1)
```

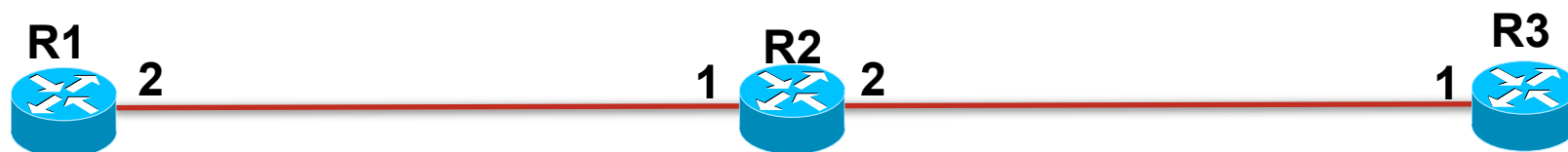


## Path Setup - more details



**Resv:**  
**Common\_Header**  
**Session(R3-Io0, 0, R1-Io0)**  
**PHOP(R2-1)**  
**Style=SE**  
**FlowSpec (2Mbps)**  
**Sender\_Template(R1-Io0, 00)**  
**Label=5**  
**Record\_Route(R2-1, R3-1)**

## Path Setup - more details



**Resv state:**  
**Session(R3-Io0, 0, R1-Io0)**  
**PHOP(R2-1)**  
**Style=SE**  
**FlowSpec (2Mbps)**  
**Sender\_Template(R1-Io0, 00)**  
**Label=5**  
**Record\_Route(R1-2, R2-1, R3-1)**

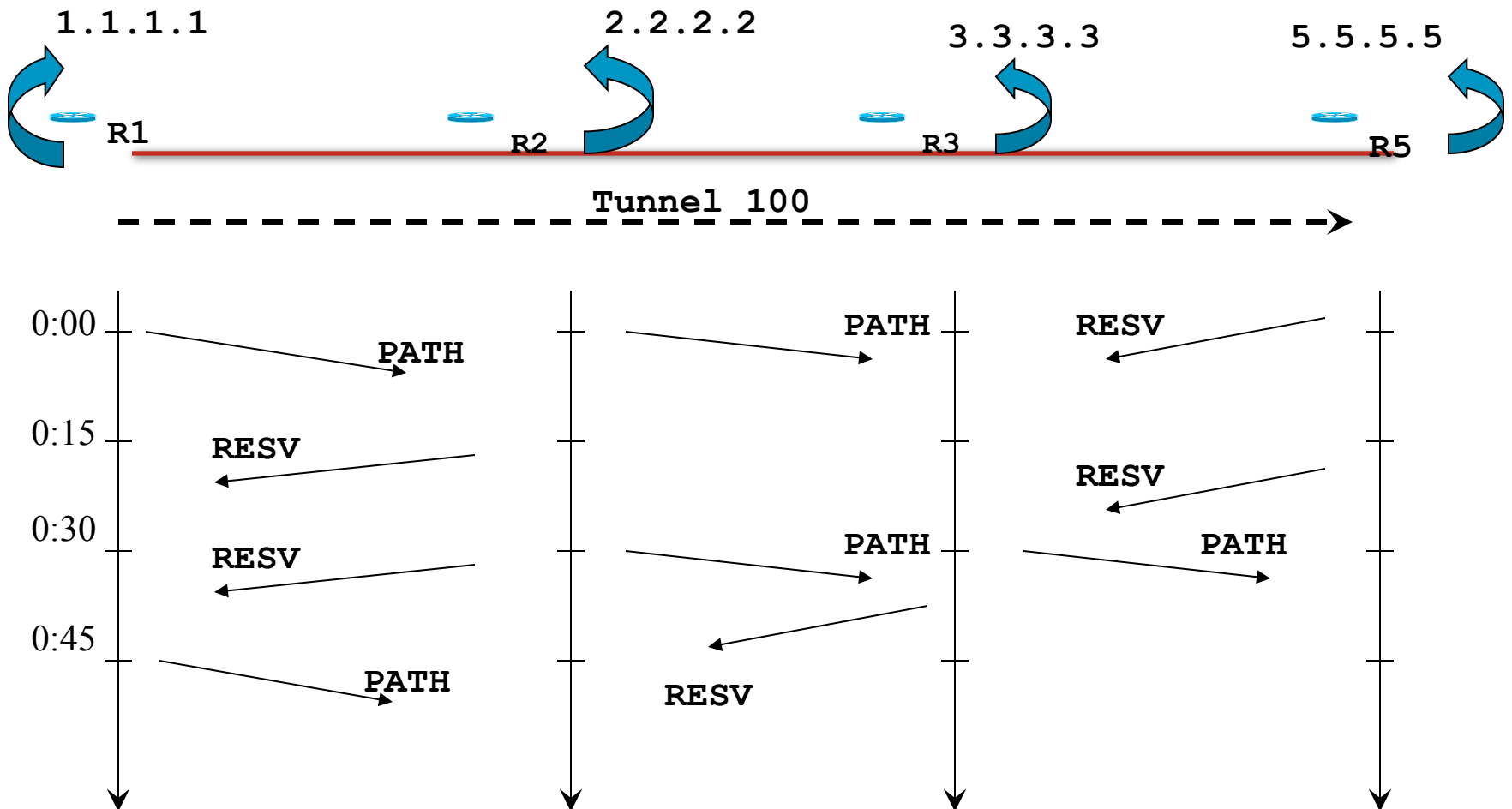
# Trunk Admission Control

- Performed by routers along a Label Switched Path (LSP)
- Determines if resources are available
- May tear down (existing) LSPs with a lower priority
- Does the local accounting
- Triggers IGP information distribution when resource thresholds are crossed
- Since TE tunnels are unidirectional, we do admission control on outbound interfaces only

# Path Maintenance

- Once the TE tunnel is setup, PATH and RESV messages are used to maintain the tunnel state
- RSVP is a soft-state protocol, relying on PATH & RESV messages for state refresh
- PATH & RESV messages are sent out on average, every 30 seconds
- If we miss 4 consecutive PATH or RESV messages, we consider the RSVP reservation dead

# Path Maintenance in action



# How MPLS-TE Works

- How MPLS-TE Works

- What good is MPLS-TE?

- Information Distribution

- Path Calculation

- Path Setup

- Forwarding Traffic Down A Tunnel

# Forwarding Traffic Down a Tunnel

- There are three ways traffic can be forwarded down a TE tunnel
  - Autoroute
  - Static routes
  - Forwarding Adjacency
  - Policy routing
- With the first two, MPLS-TE gets you **unequal**-cost load-balancing.

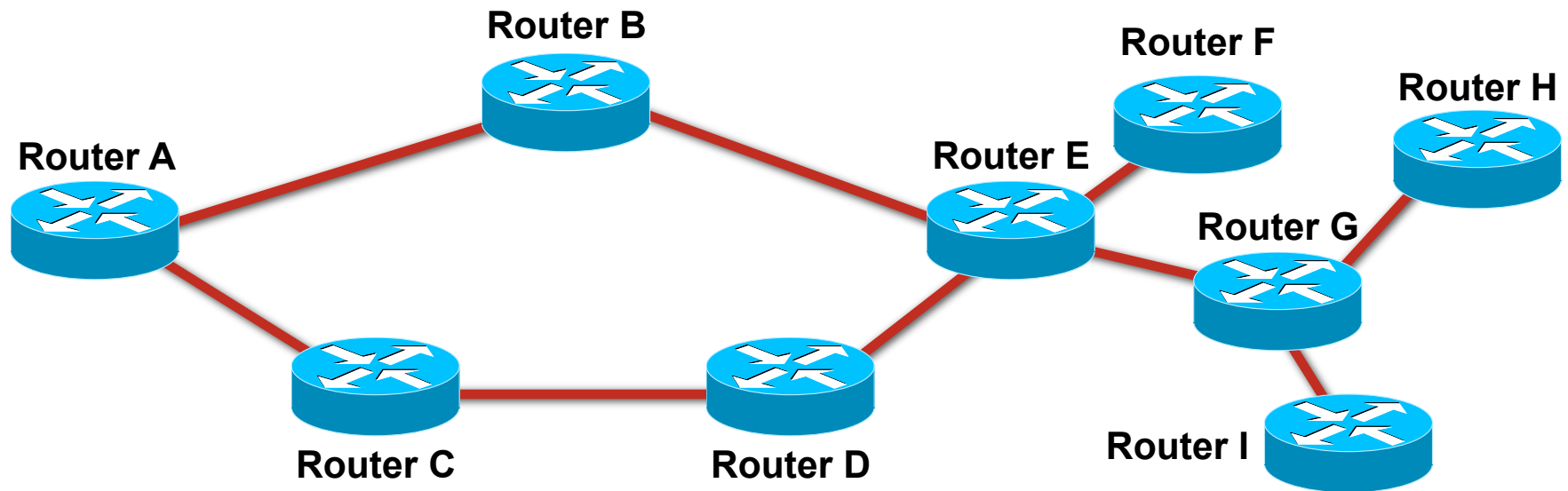
# Autoroute

- Autoroute = “use the tunnel as a directly connected link for SPF purposes”
- This is *not* the CSPF (for path determination), but the regular IGP SPF (route determination)
- IGP adjacency is **NOT** run over the tunnel!  
Unlike an ATM/FR VC
- Autoroute limited to single area/level only
- Behavior is intuitive, operation can be confusing



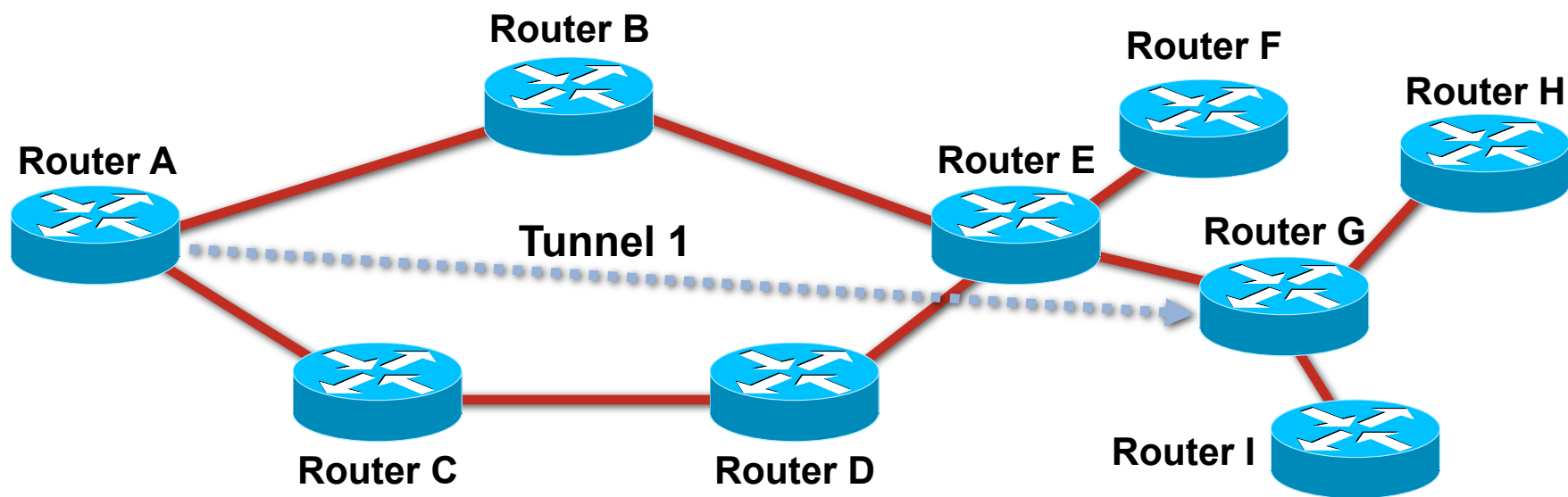
# Autorange

This Is the Physical Topology



# Autorange

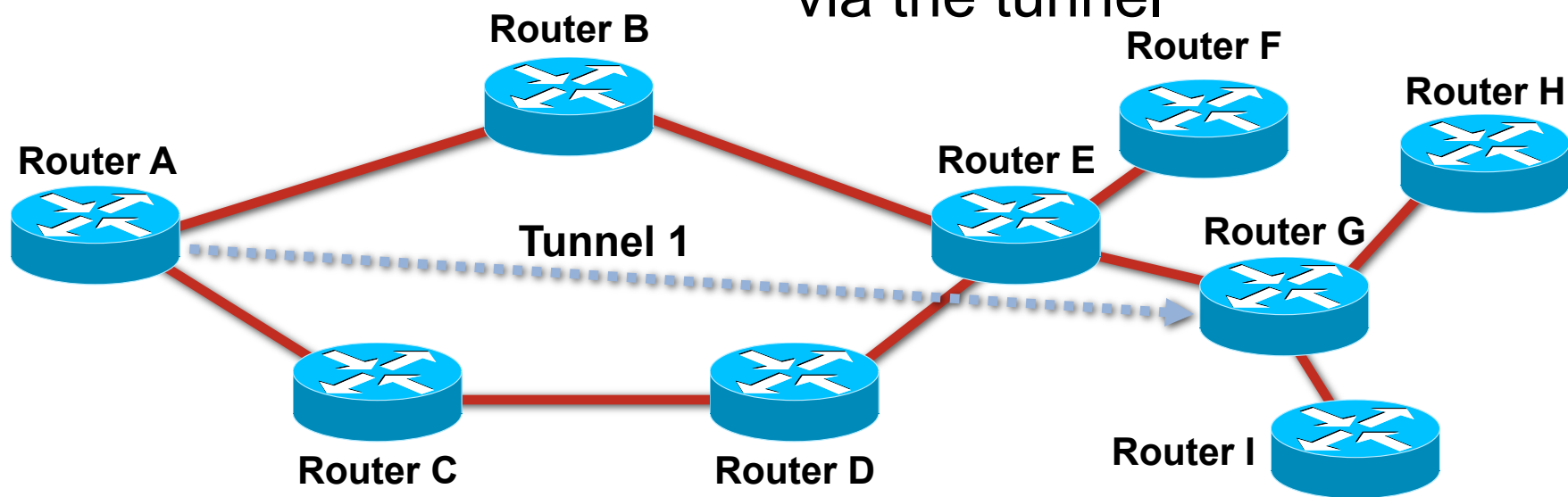
- This is Router A's logical topology
- By default, other routers don't see the tunnel!



# Autoroute

Node	Next-Hop	Cost
B	B	10
C	C	10
D	C	20
E	B	20
F	B	30
G	Tunnel 1	30
H	Tunnel 1	40
I	Tunnel 1	40

- Router A's routing table, built via auto-route
- Everything "behind" the tunnel is routed via the tunnel

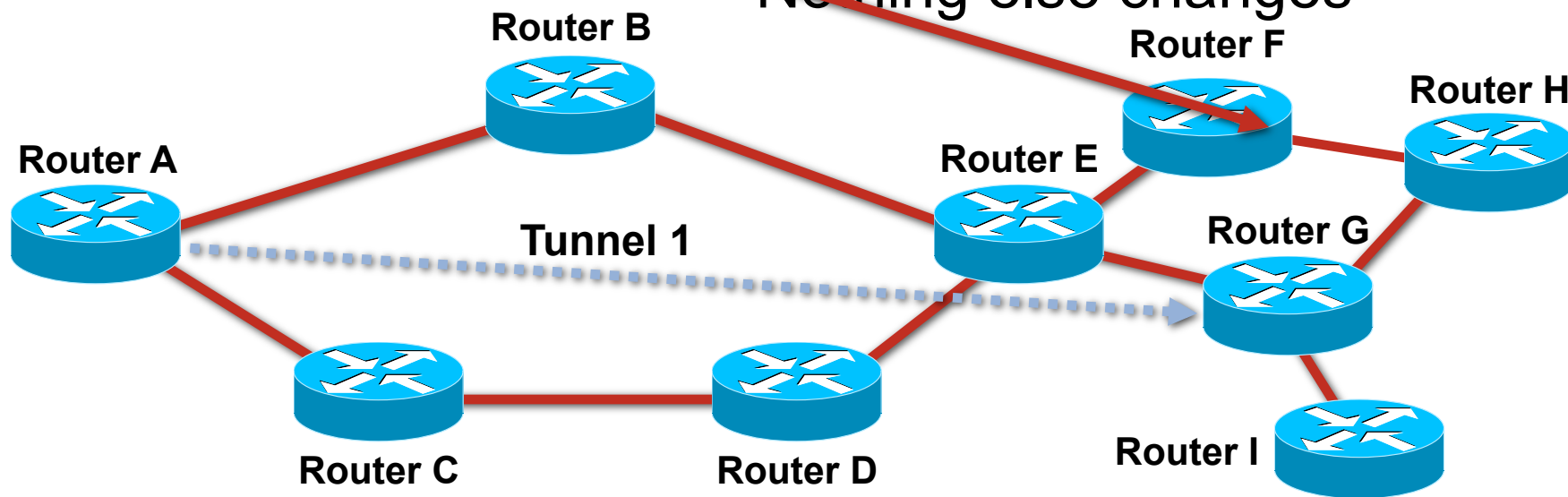


# Autoroute

Node	Next-Hop	Cost
B	B	10
C	C	10
D	C	20
E	B	20
F	B	30
G	Tunnel 1	30
H	Tunnel 1 & B	40
I	Tunnel 1	40

- If there was a link from F to H, Router A would have 2 paths to H (A->G->H and A->B->E->F->H)

- Nothing else changes

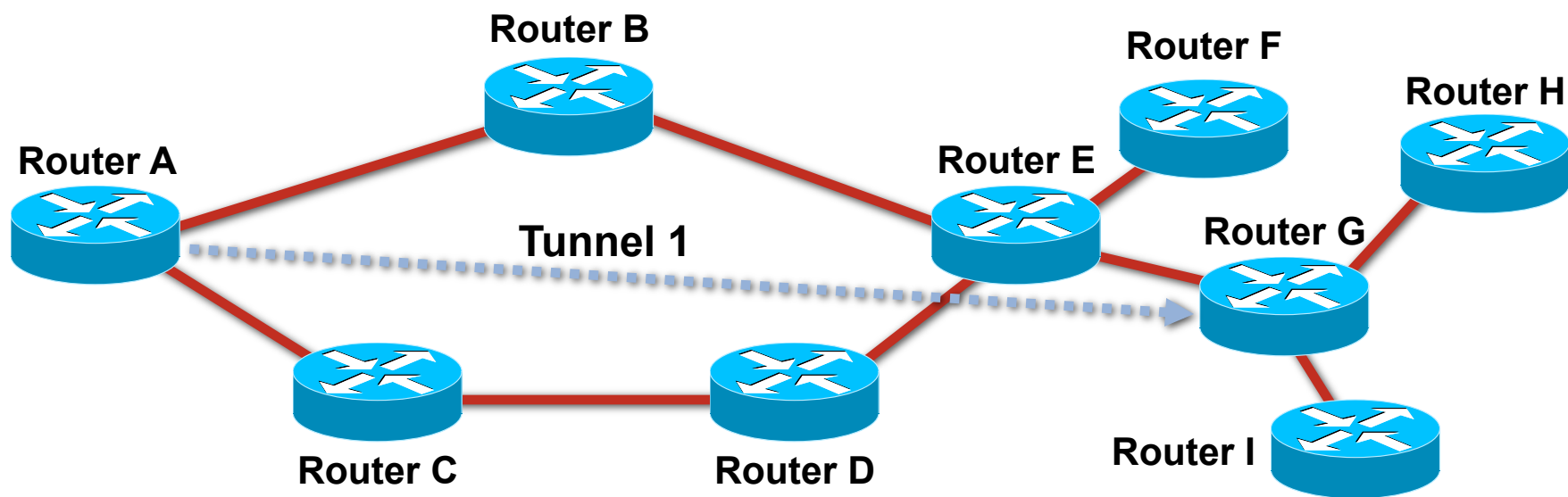


# Autoroute

Node	Next-Hop	Cost
B	B	10
C	C	10
D	C	20
E	B	20
F	B	30
G	Tunnel 1	30
H	Tunnel 1	40
I	Tunnel 1	40

```
interface Tunnel1
```

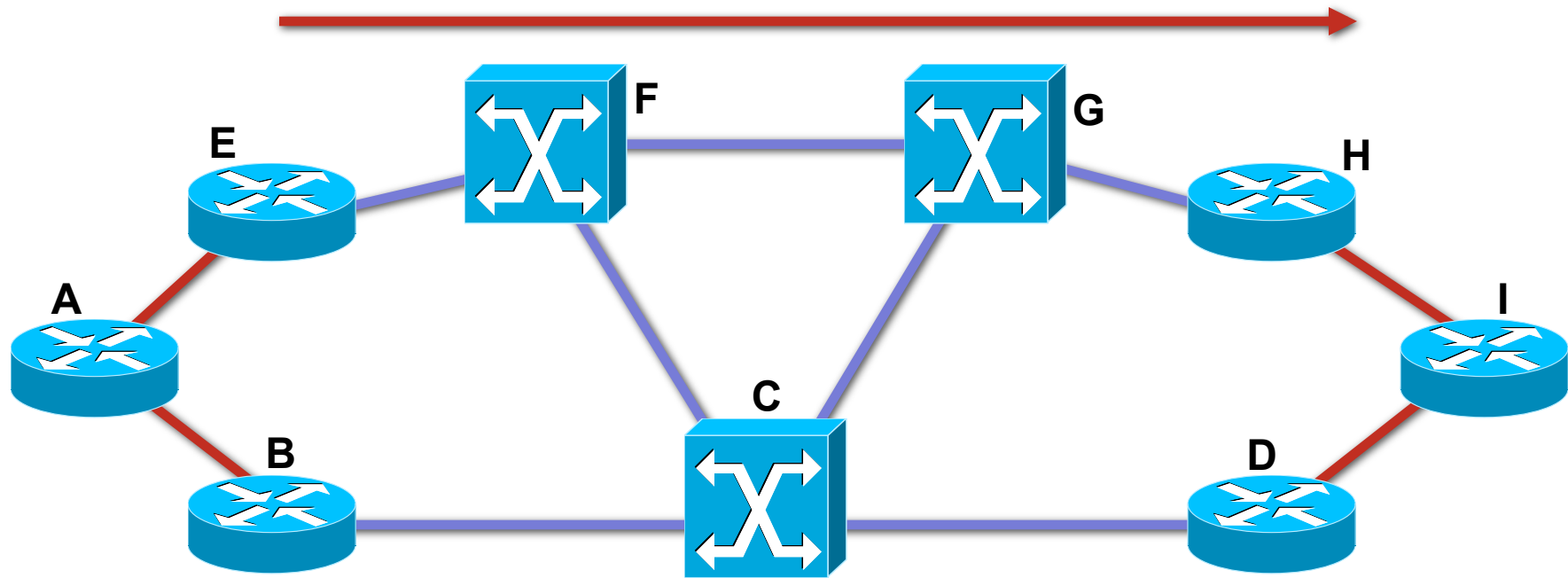
```
tunnel mpls traffic-eng autoroute announce
```



# Forwarding Adjacency

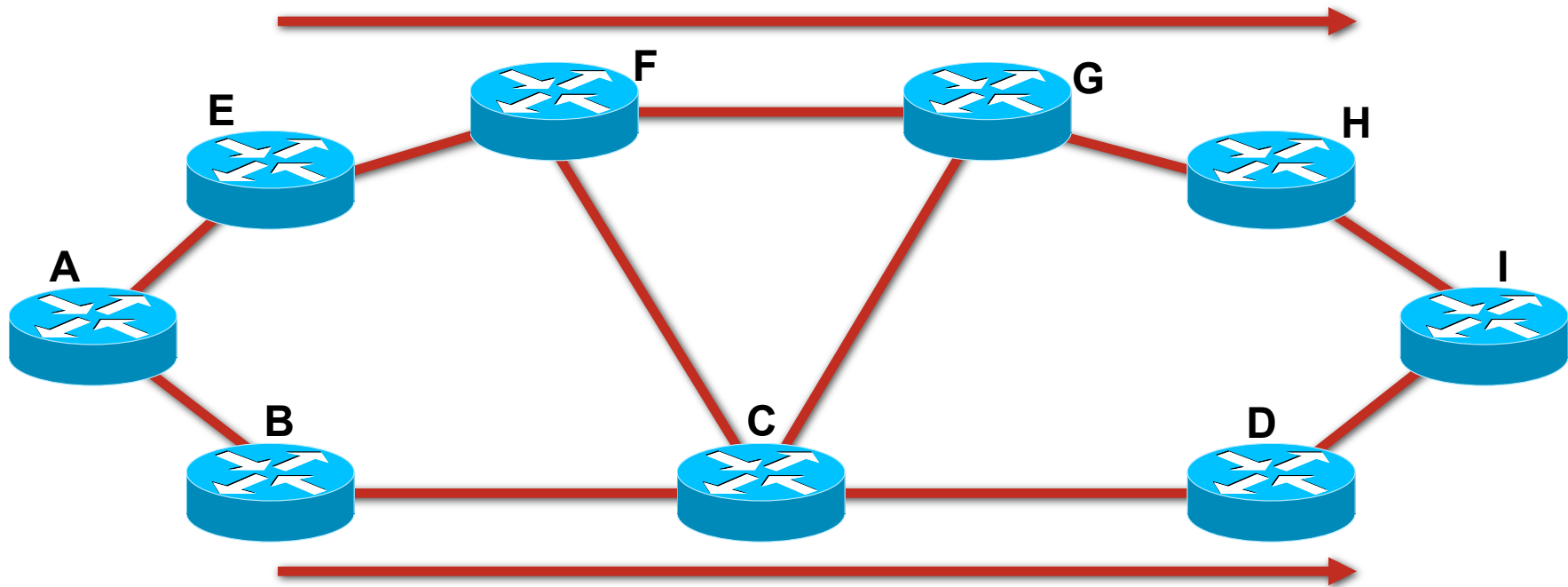
- With autoroute, the LSP is not advertised into the IGP
- This is the right behavior if you're adding TE to an IP network, but maybe not if you're migrating from ATM/FR to TE
- Sometimes advertising the LSP into the IGP as a link is necessary to preserve the routing outside the ATM/FR cloud

# ATM Model



- Cost of ATM links (blue) is unknown to routers
- A sees two links in IGP—E->H and B->D
- A can load-share between B and E

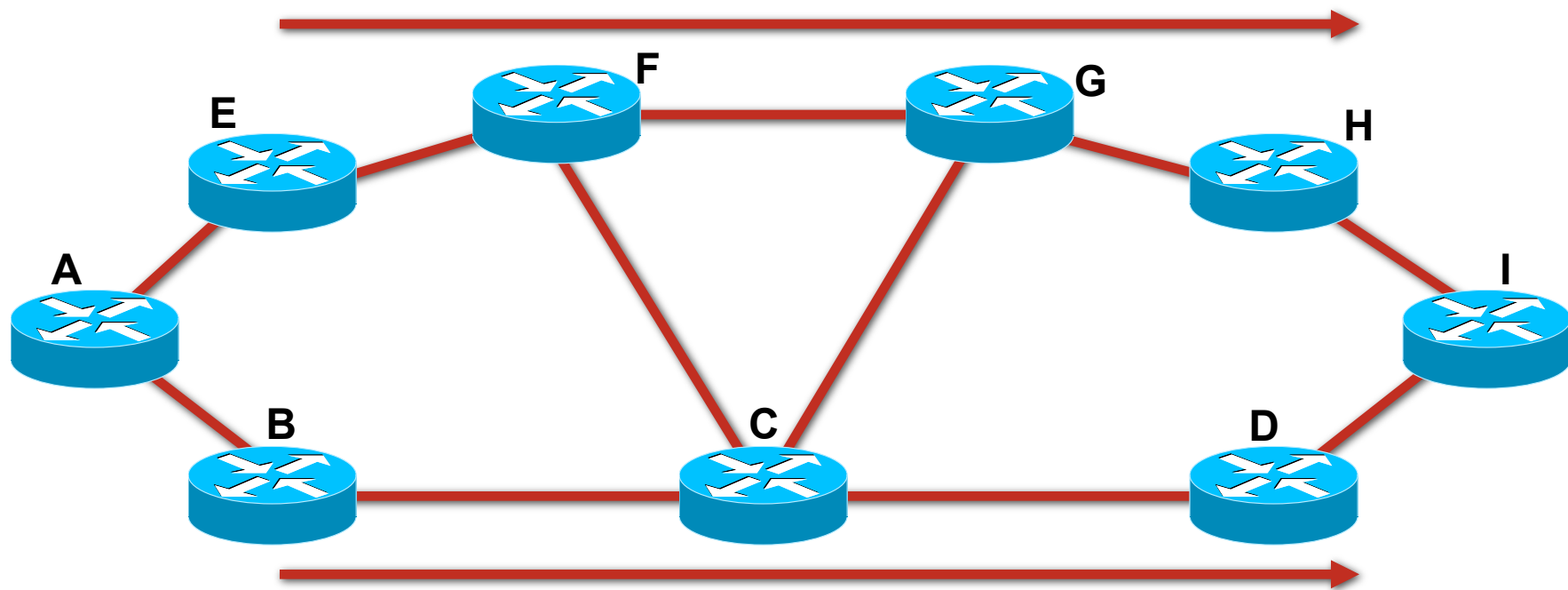
\_\_\_\_\_



- All links have cost of 10
- A's shortest path to I is A->B->C->D->I
- A doesn't see TE tunnels on {E,B}, alternate path never gets used!
- Changing link costs is undesirable, can have strange adverse effects

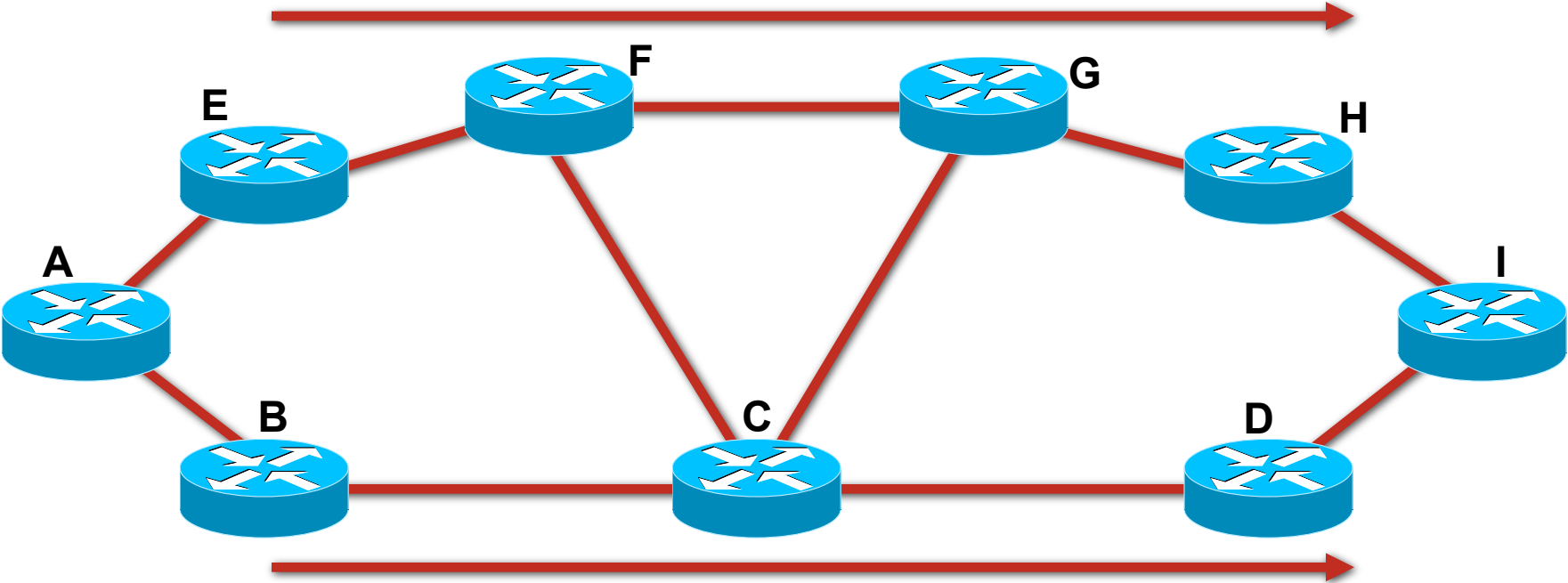


# FA Advertises TE Tunnels in the IGP



- With forwarding-adjacency, A can see the TE tunnels as links
- A can then send traffic across both paths
- This is desirable in some topologies (looks just like ATM did, same methodologies can be applied)

© 2013 Pearson Education, Inc. or its affiliate(s). All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without prior written permission from Pearson Education, Inc.



tunnel mpls traffic-eng forwarding-adjacency

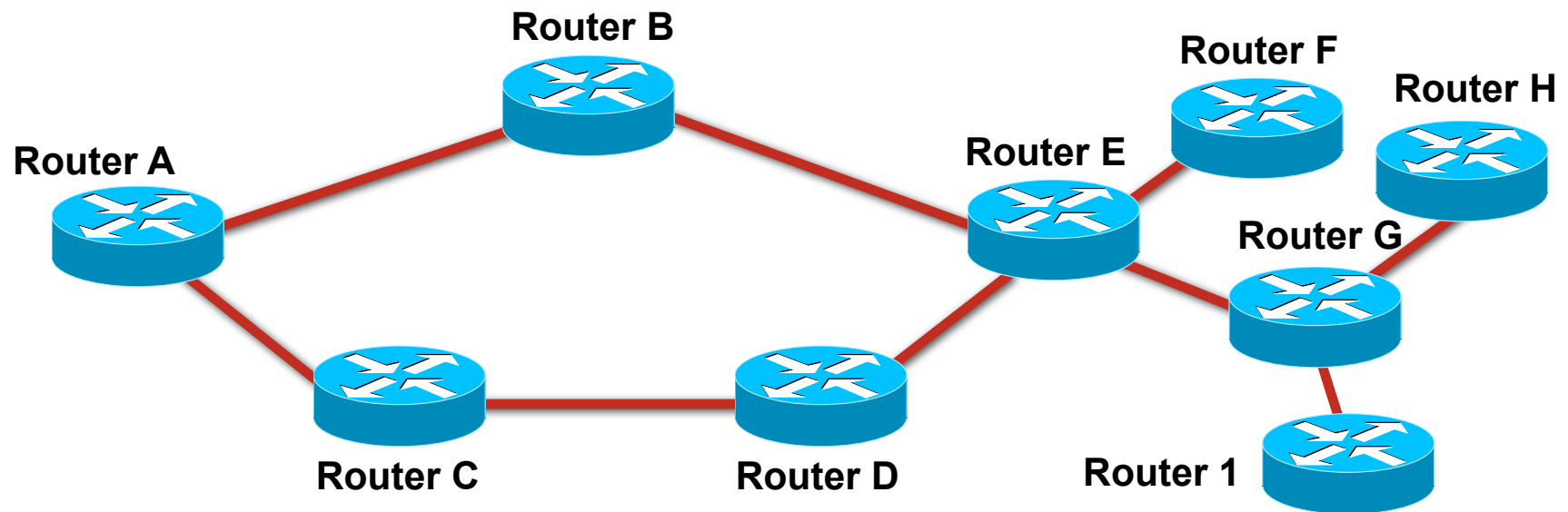
isis metric &lt;x&gt; level-&lt;y&gt;

OR

```
ip ospf cost <x>
```

# Static Routing

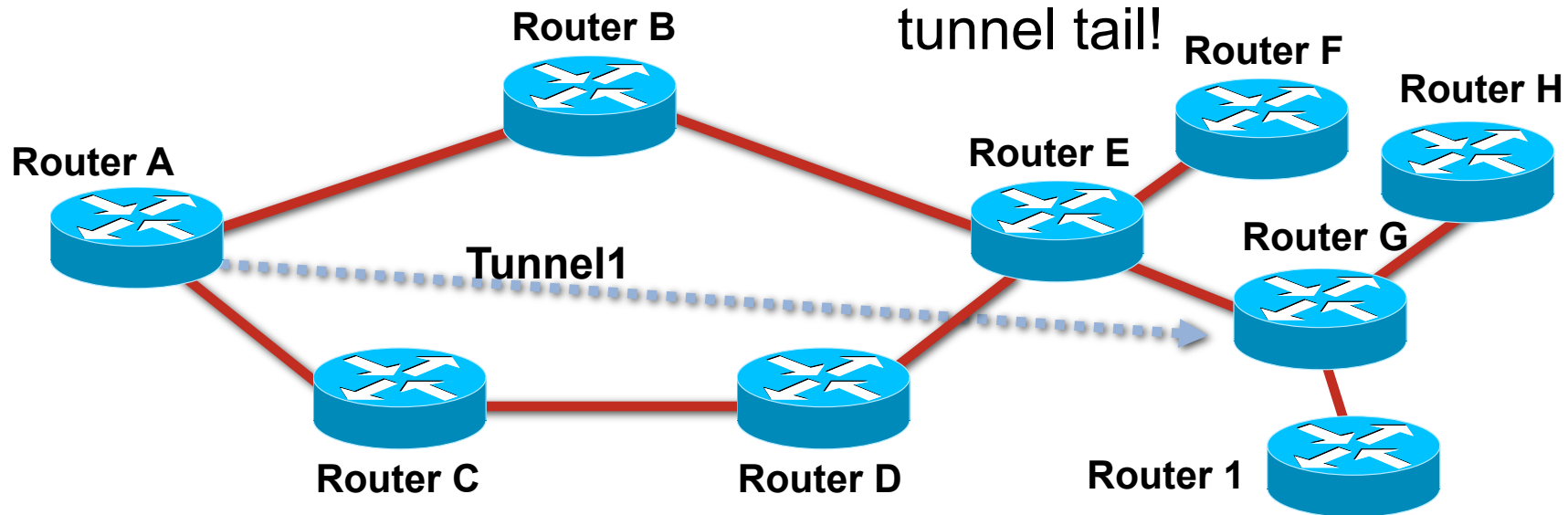
```
RtrA(config)#ip route H.H.H.H 255.255.255.255 Tunnel1
```



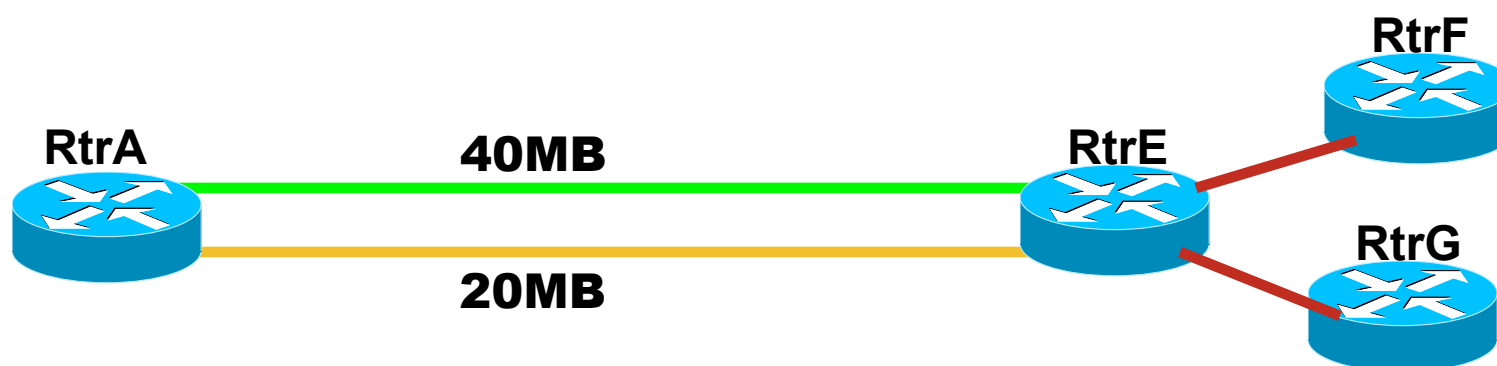
# Static Routing

Node	Next-Hop	Cost
B	B	10
C	C	10
D	C	20
E	B	20
F	B	30
G	B	30
H	Tunnel 1	40
I	B	40

- Router H is known via the tunnel
- Router G is **not** routed to over the tunnel, even though it's the tunnel tail!



# Static routing



```
gsr1(config)#ip route 1.2.3.4 255.255.255.255 192.168.1.11
```

```
gsr1#sh ip cef 1.2.3.4
```

.....

```
Load distribution: 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 (refcount 1)
```

Hash	OK	Interface	Address	Packets	Tags imposed
1	Y	Tunnel0	point2point	0	{23}
2	Y	Tunnel1	point2point	0	{34}

.....

**Static routes inherit unequal-cost load-sharing when recursing through a tunnel.**

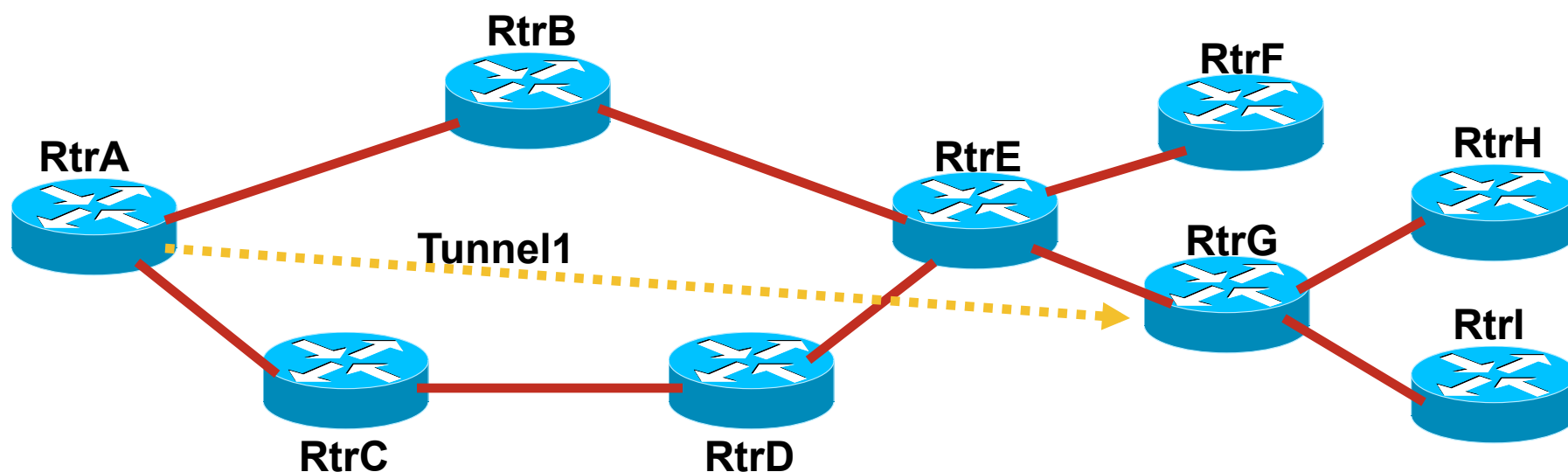
# Policy routing

```
RtrA(config-if)#ip policy route-map set-tunnel
```

```
RtrA(config)#route-map set-tunnel
```

```
RtrA(config-route-map)#match ip address 101
```

```
RtrA(config-route-map)#set interface Tunnel1
```



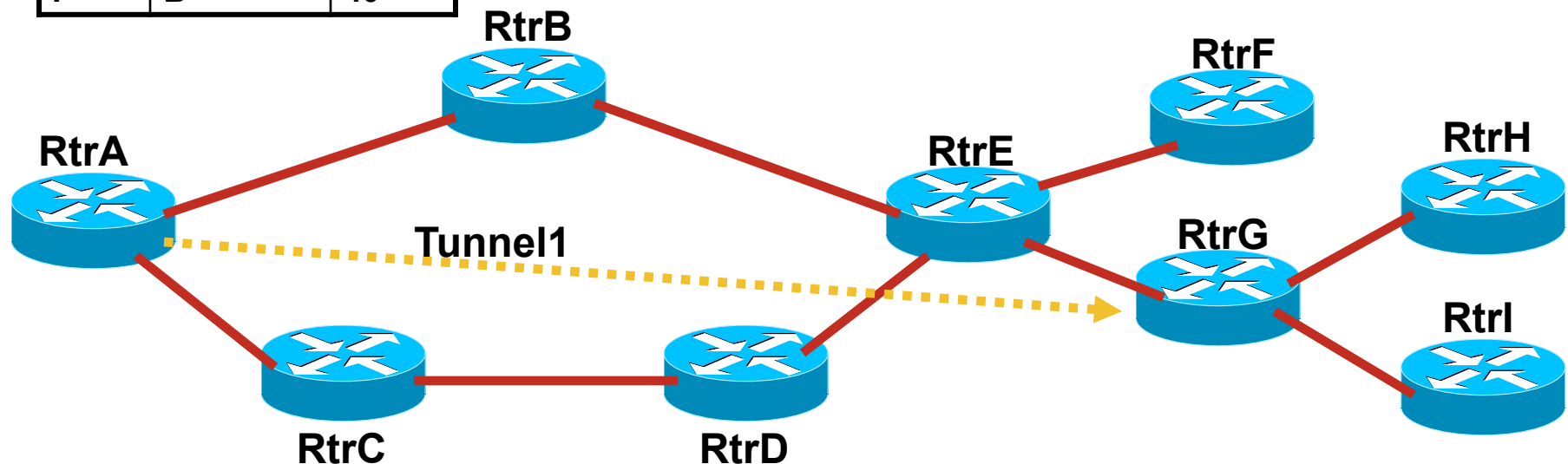
# Policy routing

Node	Next-Hop	Cost
B	B	10
C	C	10
D	C	20
E	B	20
F	B	30
G	B	30
H	B	40
I	B	40

Routing table isn't affected by policy routing.



Need (12.0(16)ST or 12.2T) or higher for 'set interface Tunnel1' to work



## Forwarding Traffic Down a Tunnel

- You can use any combination of autoroute, static routes, FA or PBR.
- ...but simple is better unless you have a good reason.
- Recommendation: either autoroute or statics to BGP next-hops, depending on your needs.





## Load Sharing with TE tunnels

# Unequal Cost Load Balancing

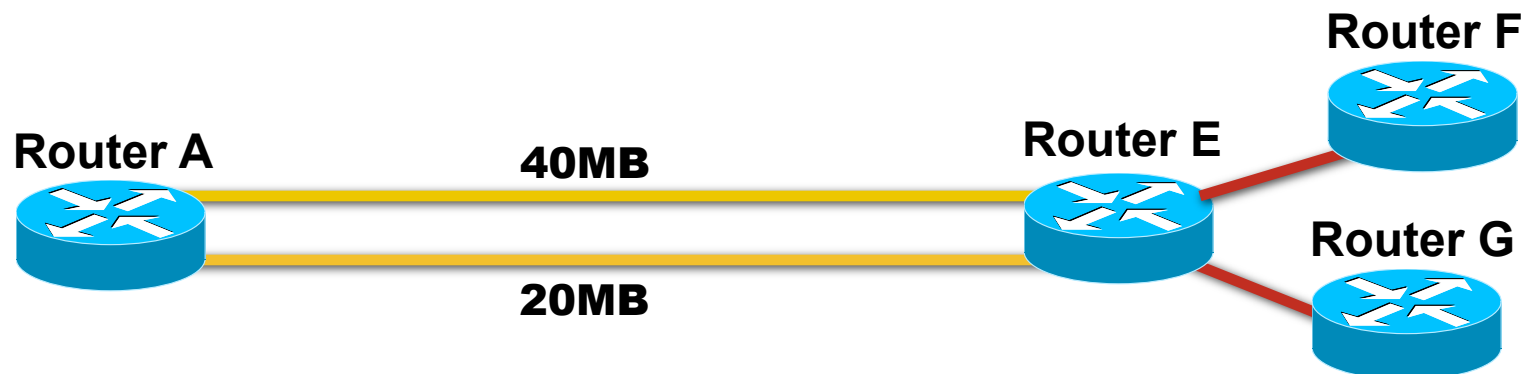
- IP routing has equal-cost load balancing, but not unequal cost\*
- Unequal cost load balancing difficult to do while guaranteeing a loop-free topology
- 16 hash buckets for next-hop, shared in rough proportion to configured tunnel bandwidth or load-share value

\*EIGRP Has 'Variance', but That's Not as Flexible

# Unequal Cost Load Balancing

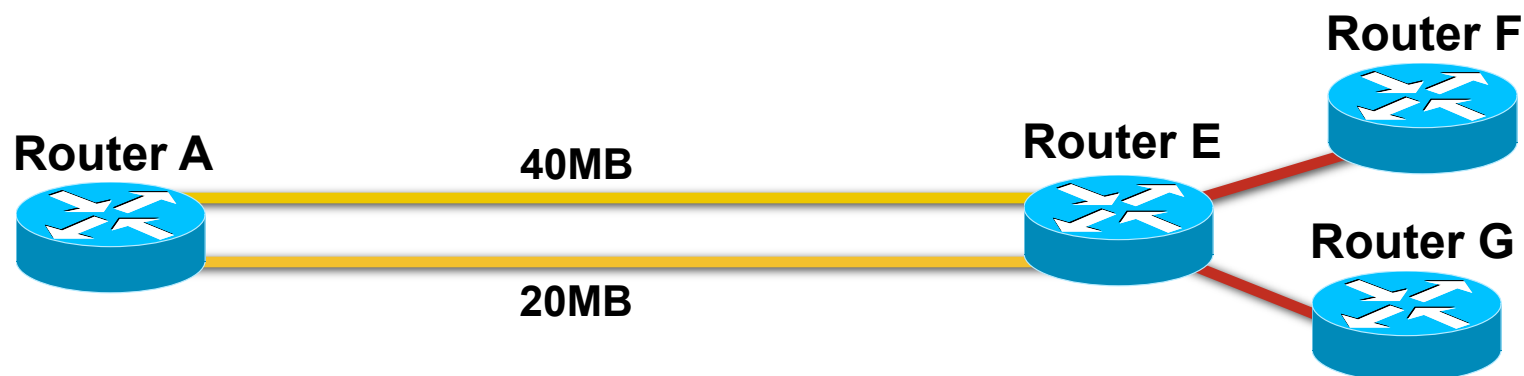
- A TE tunnel does not load share traffic between itself and the native IP path it takes
- Multiple parallel tunnels can load share traffic based on bandwidth. This can be equal or unequal cost load balancing
- TE tunnels and native IP links can load share traffic, provided the destination is downstream to the tunnel destination. In this case load sharing is equal cost

# Unequal Cost: Example 1



```
gsr1#show ip route 192.168.1.8
Routing entry for 192.168.1.8/32
Known via "isis", distance 115, metric 83, type level-2
  Redistributing via isis
  Last update from 192.168.1.8 on Tunnel0, 00:00:21 ago
    Routing Descriptor Blocks:
  * 192.168.1.8, from 192.168.1.8, via Tunnel0
    Route metric is 83, traffic share count is 2
  192.168.1.8, from 192.168.1.8, via Tunnel1
    Route metric is 83, traffic share count is 1
```

# Unequal Cost: Example 1

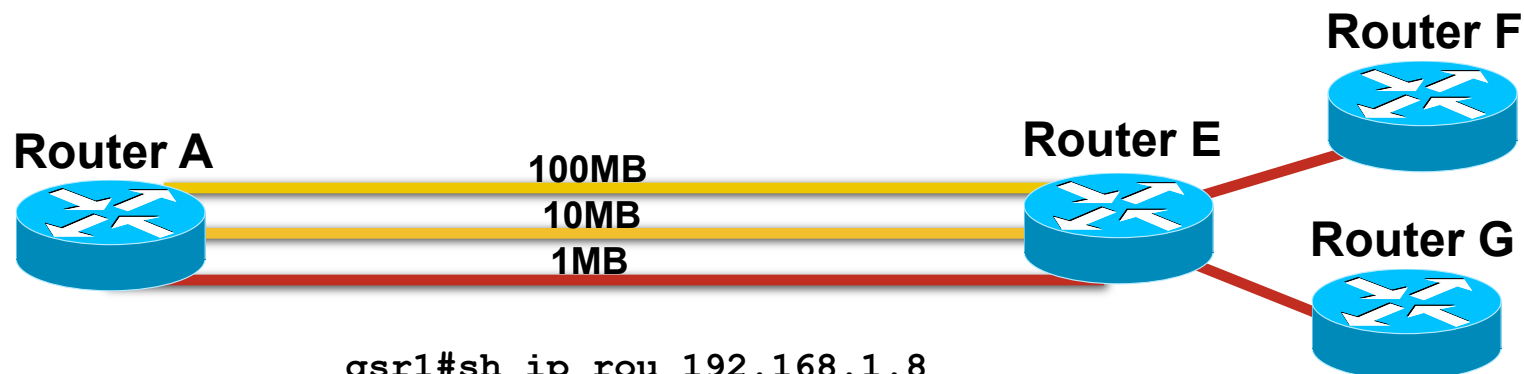


```
gsr1#sh ip cef 192.168.1.8 internal
```

```
Load distribution: 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 (refcount 1)
Hash  OK  Interface      Address      Packets  Tags imposed
  1    Y   Tunnel0         point2point      0      {23}
  2    Y   Tunnel1         point2point      0      {34}
```

**Note That the Load Distribution  
Is 11:5—Very Close to 2:1, but Not Quite!**

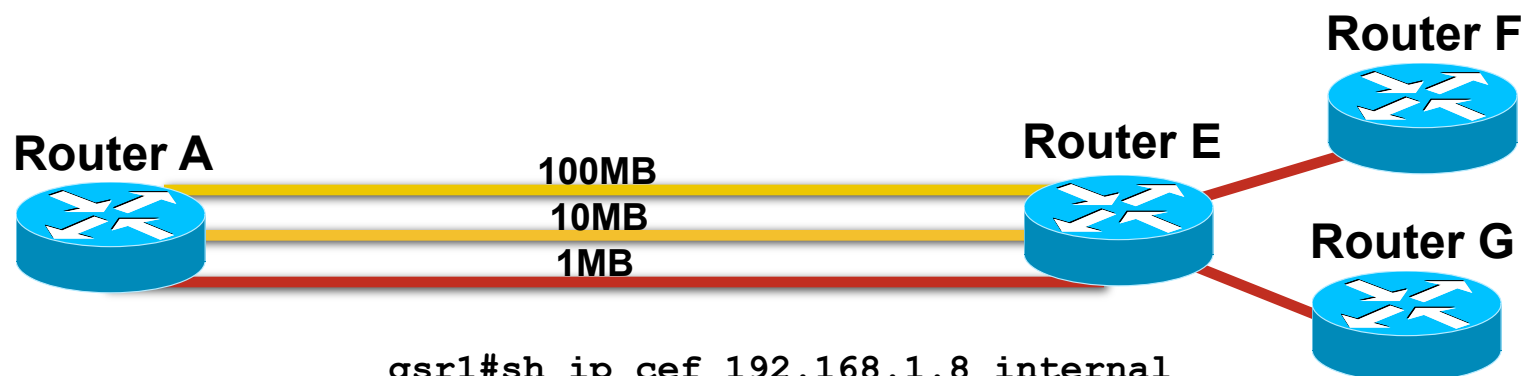
## Unequal Cost: Example 2



```
gsr1#sh ip rou 192.168.1.8
Routing entry for 192.168.1.8/32
Known via "isis", distance 115, metric 83, type level-2
  Redistributing via isis
  Last update from 192.168.1.8 on Tunnel2, 00:00:08 ago
  Routing Descriptor Blocks:
    * 192.168.1.8, from 192.168.1.8, via Tunnel0
      Route metric is 83, traffic share count is 100
    192.168.1.8, from 192.168.1.8, via Tunnel1
      Route metric is 83, traffic share count is 10
    192.168.1.8, from 192.168.1.8, via Tunnel2
      Route metric is 83, traffic share count is 1
```

**Q: How Does 100:10:1 Fit Into a 16-Deep Hash?**

## Unequal Cost: Example 2



```
gsr1#sh ip cef 192.168.1.8 internal
```

```
.....  
Load distribution: 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (refcount 1)
```

Hash	OK	Interface	Address	Packets	Tags imposed
1	Y	Tunnel0	point2point	0	{36}
2	Y	Tunnel1	point2point	0	{37}

**A: Any Way It Wants to! 15:1, 14:2, 13:2:1, it depends on the order the tunnels come up**  
**Deployment Guideline: Don't use tunnel metrics that don't reduce to 16 buckets!**

# Agenda

- Prerequisites
- How MPLS-TE Works
- Basic Configuration
- Knobs! Knobs! Knobs!
- Configuring and Monitoring





# MPLS-TE: Configuring and monitoring

# Basic Configuration

- Basic Configuration
  - Basic Midpoint/Tail Config
  - Basic Headend Config
  - Path-option
  - Bandwidth

# Basic Midpoint/Tail Config

(globally)

```
ip cef {distributed}  
mpls traffic-eng tunnels
```

(per interface)

```
mpls traffic-eng tunnels
```

# Basic Midpoint/Tail Config

(if IGP == OSPF)

```
router ospf <x>
```

```
    mpls traffic-eng router-id Loopback0
```

```
    mpls traffic-eng area <y>
```

## Basic Midpoint/Tail Config

(if IGP == IS-IS)

```
router isis <x>
```

```
    mpls traffic-eng router-id  
    Loopback0
```

```
    mpls traffic-eng level-{1,2}
```

```
    metric-style wide
```

# Information Distribution

- On each physical interface

```
interface pos0/0  
  mpls traffic-eng tunnels  
  ip rsvp bandwidth Kbps (Optional)  
  mpls traffic-eng attribute-flags  
  attributes (Opt)
```

# Basic Headend Config

- Create the tunnel interface

```
interface Tunnel0
  ip unnumbered Loopback0
  tunnel mode mpls traffic-eng
  tunnel source Loopback0
  tunnel destination < tunnel endpoint >
  tunnel mpls traffic-eng autoroute
  tunnel mpls traffic-eng path-option 10 dynamic
```

**unnumbered to Loop0**

**...instead of GRE, etc**

**autoroute is not strictly necessary, but is useful**

**path-option tells the tunnel how to get to tail**  
**'10' is the priority of the path-option**  
**there are other options besides dynamic**

# Basic Headend Config

- Total config tally:
    - 1 line globally
    - 1 line per interface
    - 2 lines if OSPF
    - 3 lines if IS-IS
    - + 7 lines per tunnel at headend
- Not really much to the basic config.



# Agenda

- Prerequisites
- How MPLS-TE Works
- Basic Configuration
- Knobs! Knobs! Knobs!
- Deploying and Designing

# Knobs! Knobs! Knobs!

- Influencing the Path Selection
- Auto-Bandwidth
- Fast Reroute
- DiffServ-Aware Traffic Engineering

# Knobs! Knobs! Knobs!

- Influencing the Path Selection

Bandwidth

Priority

Administrative Weight

Attributes & Affinity

# Bandwidth

```
tunnel mpls traffic-eng  
bandwidth <Kb>
```

- Per-tunnel command
- Tunnel default: 0 Kb

# Bandwidth

```
ip rsvp bandwidth <x> <y>
```

- Per-physical-interface command
- X = amount of reservable BW, in K
- Y = not used by MPLS-TE
- default: X==Y==75% of link bandwidth

# Priority

```
tunnel mpls traffic-eng <S> {H}
```

- Configured on tunnel interface
- S = setup priority (0-7)
- H = holding priority (0-7)
- *lower* number is *more* important, or *better*.

# Priority

- New tunnel with better setup priority will force teardown of already-established tunnel with worse holding priority
- Configuring  $S < H$  is illegal
- Default is  $S=7, H=7$

# Priority

“Should I ever set  $S \neq H$ ?”

Not unless you know you have a good reason to.

“Can you think of a good reason to?”

No.



# Administrative Weight

```
mpls traffic-eng administrative-weight <X>
```

- Per-physical-interface command
- X = 0-4,294,967,295
- gives a metric that be considered for use instead of the IGP metric
- this can be used as a per-tunnel delay-sensitive metric for doing VoIP TE

# Delay-Sensitive Metric with Administrative Weight

```
tunnel mpls traffic-eng path-selection  
metric {te|igp}
```

- configure admin weight == interface delay
- configure VoIP tunnels to use TE metric to calculate the path cost (see the PCALC algorithm earlier in these slides)
- delay-sensitive metric – just add water!

# Attributes & Affinity

- Link attribute – 32 separate link properties
- Tunnel affinity – desire for links to have certain properties set
- Invent your own property meanings

## Attributes & Affinity

```
tunnel mpls traffic-eng affinity  
<0x0-0xFFFFFFFF> {mask <0x0-0xFFFFFFFF>}
```

- Per-tunnel command
- Mask is a collection of *do-care* bits
- 'affinity 0x2 mask 0xA' means 'I care about bits 1 and 3 (with the values 2 and 8); bit 1 must be set, bit 3 must be 0'

## Attributes & Affinity

```
mpls traffic-eng attribute-  
flags <0x0-0xFFFFFFFF>
```

- Per-physical-interface command

# Attributes & Affinity

- Q: How should I use link attributes?
- A: To exclude some links from consideration by some tunnels
- ...so give a satellite link an attribute of 0x2, and any VoIP tunnels can be configured with 'affinity 0x0 mask 0x2'

# Knobs! Knobs! Knobs!

- Influencing the Path Selection
- Auto-Bandwidth
- Fast Reroute
- DiffServ-Aware Traffic Engineering

# Auto-Bandwidth

```
tunnel mpls traffic-eng auto-bw ?  
  collect-bw Just collect Bandwidth info on this tunnel  
  frequency  Frequency to change tunnel BW  
  max-bw     Set the Maximum Bandwidth for auto-bw on this tunnel  
  min-bw     Set the Minimum Bandwidth for auto-bw on this tunnel  
  <cr>
```

- Per-tunnel command
- Periodically changes tunnel BW reservation based on traffic out tunnel
- Timers are tunable to make auto-bw more or less sensitive

tradeoff: quicker reaction vs. more churn



# Knobs! Knobs! Knobs!

- Influencing the Path Selection
- Auto-Bandwidth
- Fast Reroute
- DiffServ-Aware Traffic Engineering

# Fast Reroute

- In an IP network, a link failure causes several seconds of outage

Thing	Dependency	Time
Link failure detection	Media- and platform-specific	~usecs (POS + APS)
Info propagation	IGP timers, network size, collective router load	~5-30sec
Route recalculation	LSDB size, CPU load	~1-2sec

## Fast Reroute

- In an MPLS network, there's more work to be done, so a (slightly) longer outage happens

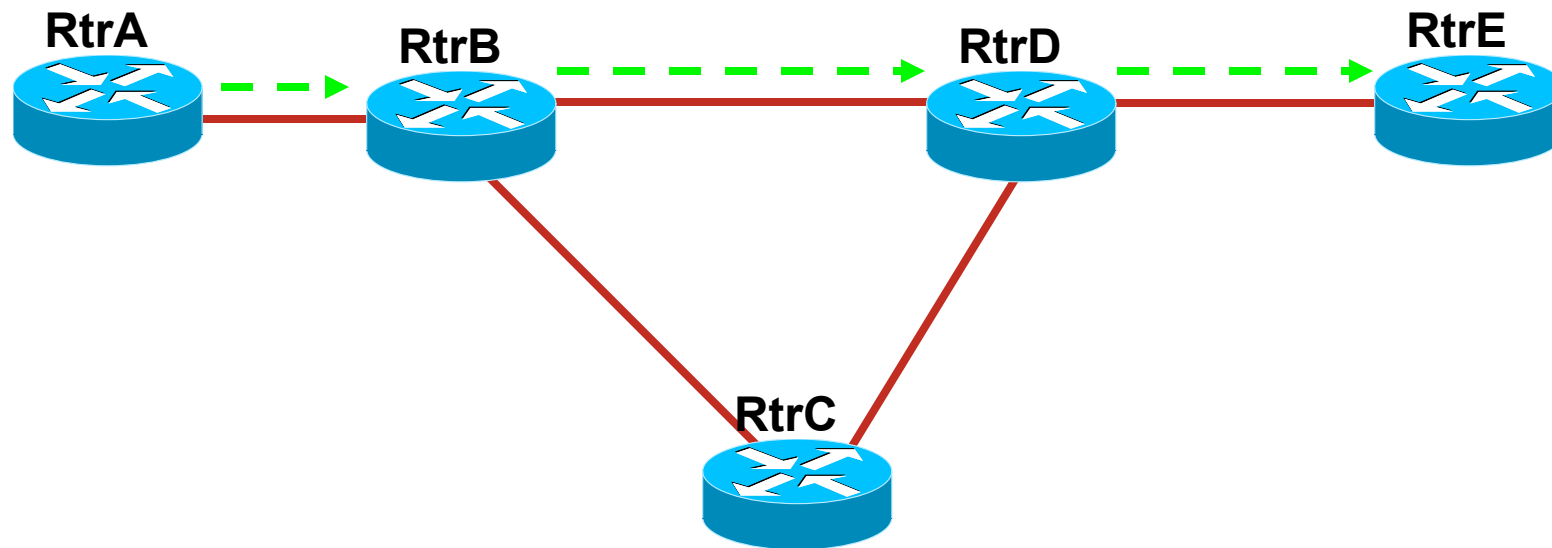
Thing	Dependency	Time
Link failure detection	Media- and platform-specific	~usecs (POS + APS)
Info propagation	IGP timers, network size, collective router load	~5-30sec
Route recalculation	LSDB size, CPU load	~1-2sec
<b>New LSP setup</b>	<b>network size, CPU load</b>	<b>~5-10sec</b>

# Three Kinds of FRR

- Link Protection  
the only scheme implemented today
- Node Protection  
on the way
- Path Protection  
on development radar

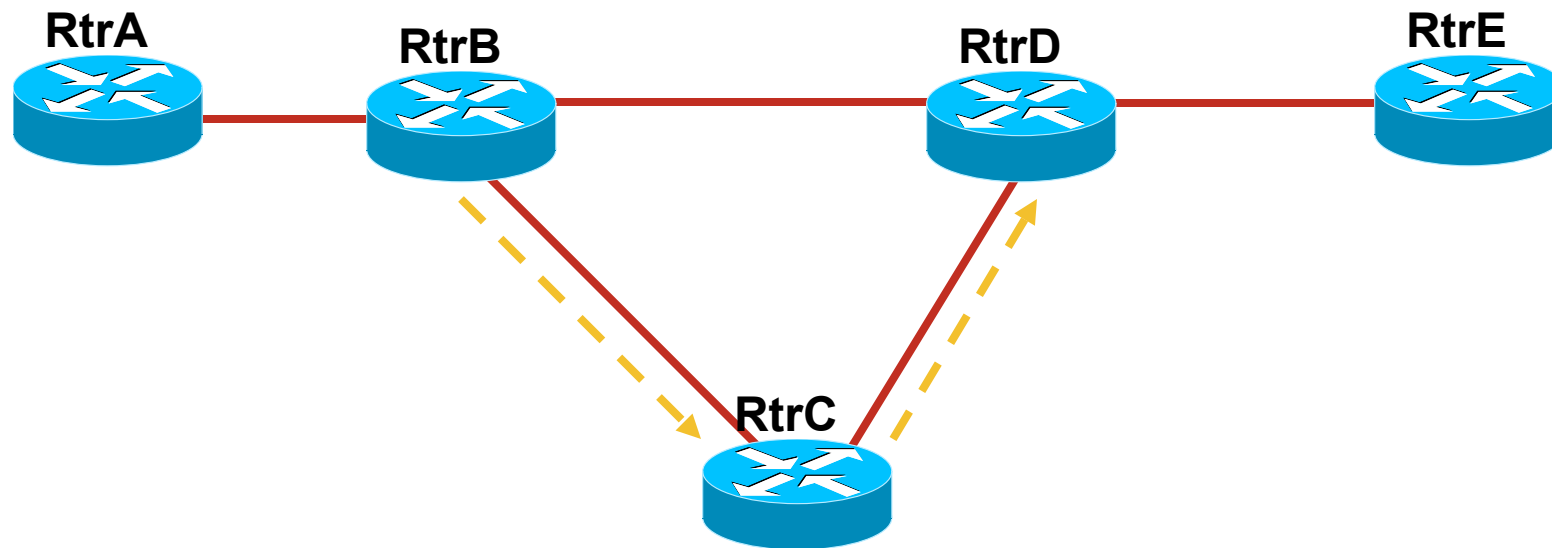
# Link Protection

- TE tunnel A->B->D->E



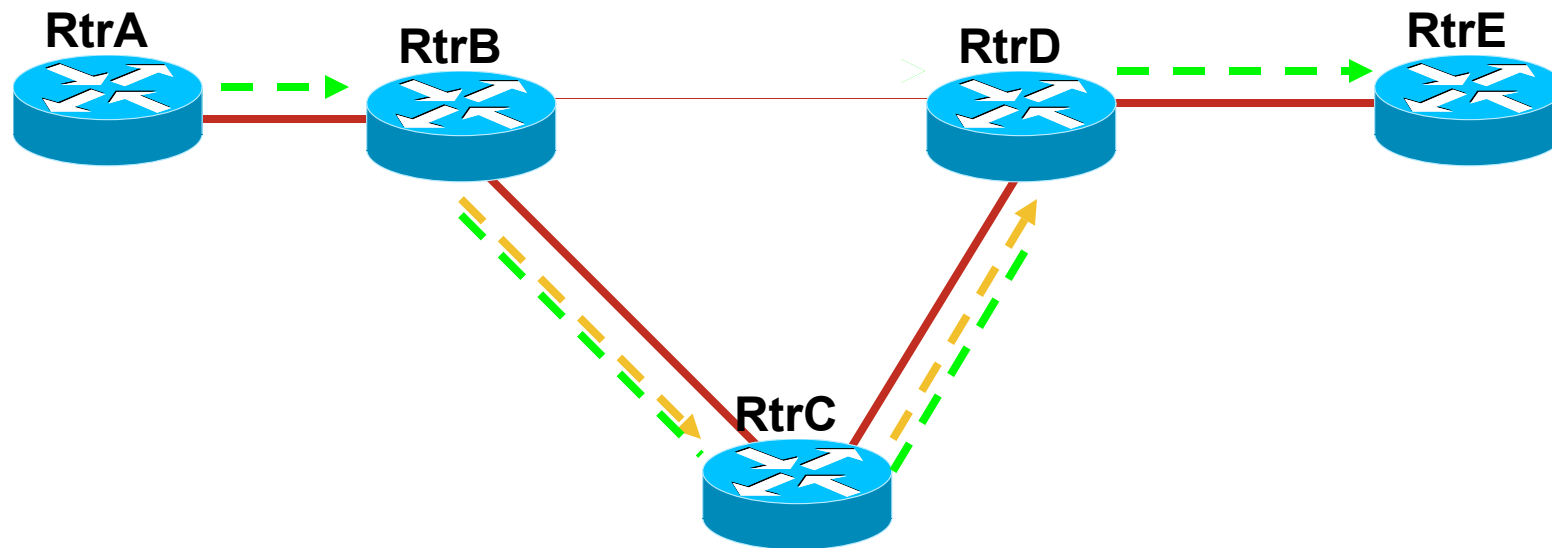
# Link Protection

- B has a pre-provisioned backup tunnel to the other end of the protected link (RtrD)
- B relies on the fact that D is using global label space



# Link Protection

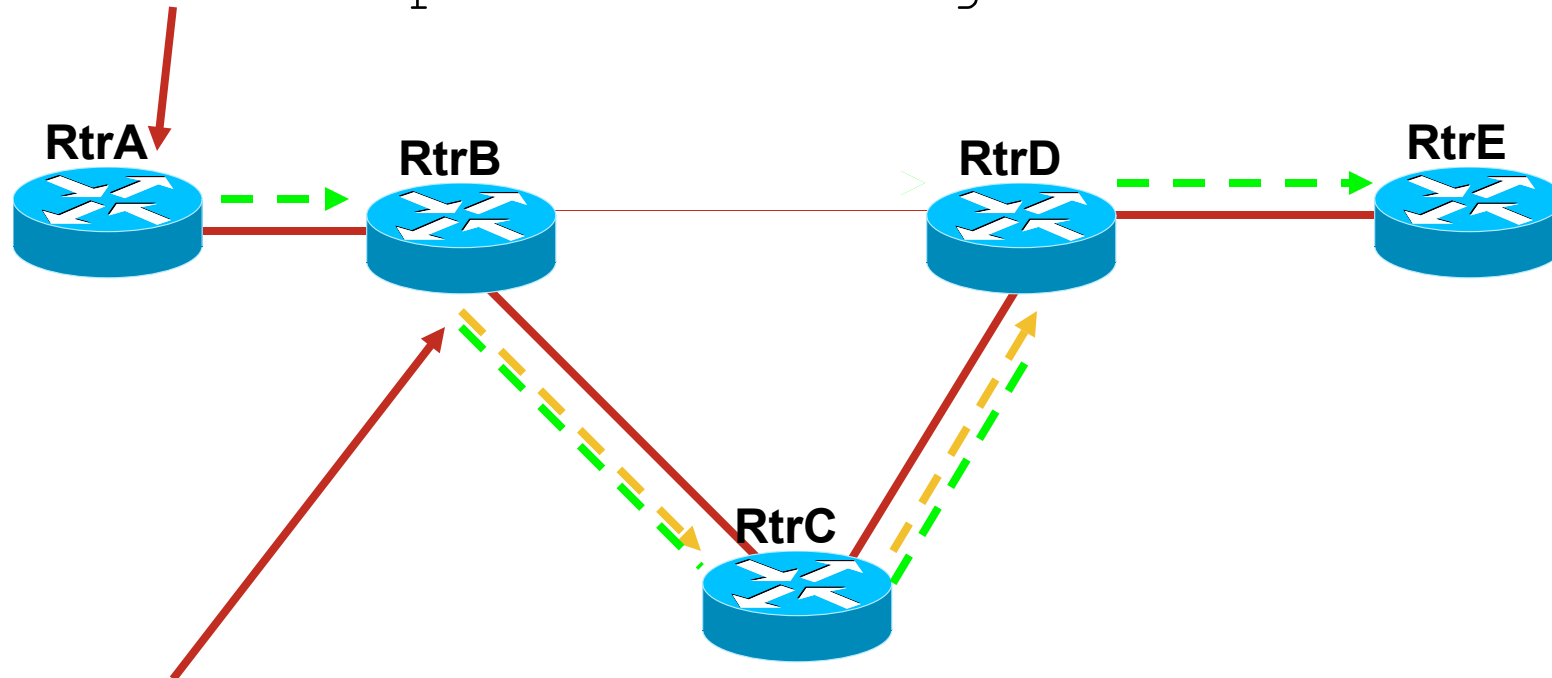
- B->D link fails, A->E tunnel is encapsulated in B->D tunnel
- Backup tunnel is used until A can recompute tunnel path as A->B->C->D->E (so 10-30sec or so)



# Link Protection

- On tunnel headend:

```
tunnel mpls traffic-eng fast-reroute
```



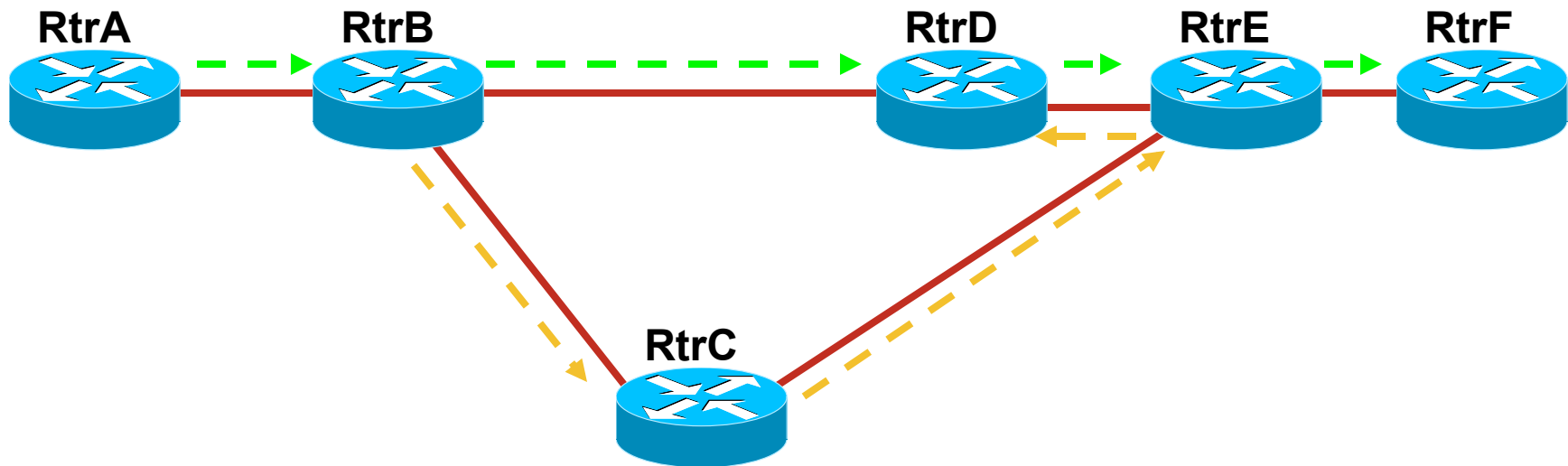
- On protected link:

```
mpls traffic-eng backup-path <backup-tunnel>
```



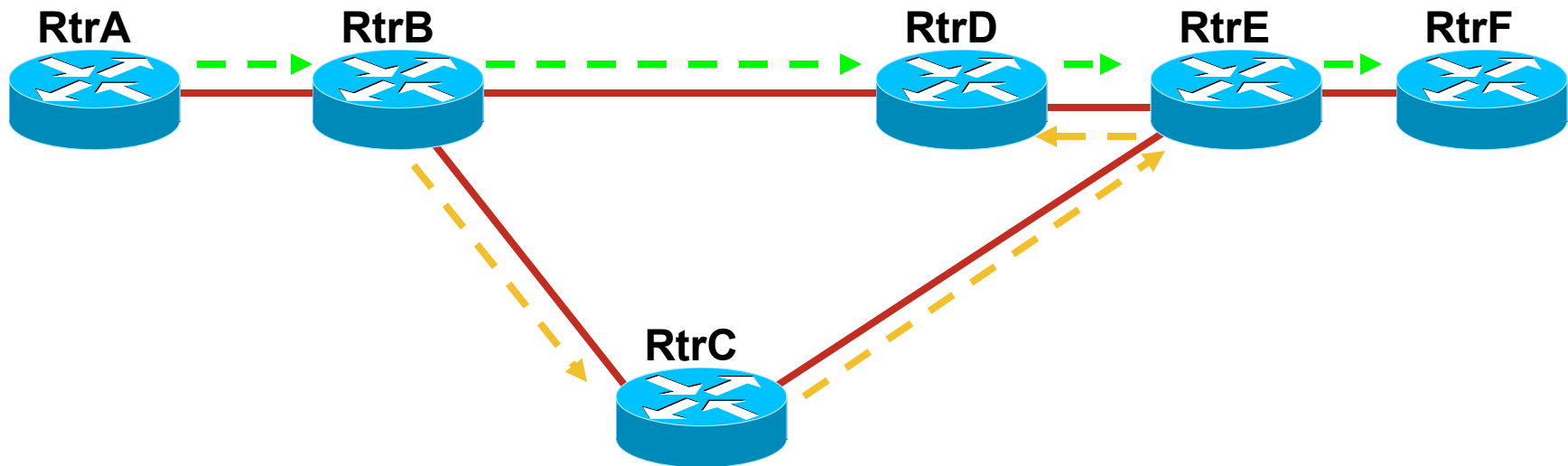
# Node Protection

- RtrA has a tunnel A->B->D->E->F
- RtrB has a protect tunnel B->C->E->D



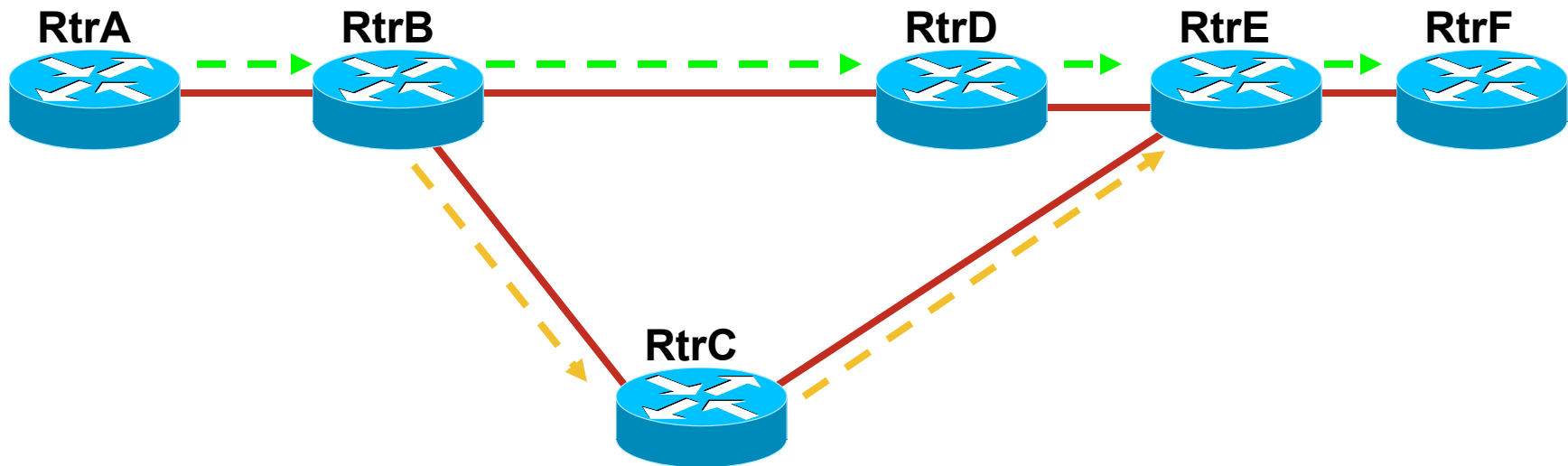
# Node Protection

- Link protection is OK if the B->D link goes down
- What if Router D goes away?



# Node Protection

- Solution: protect tunnel to the hop *past* the protected link

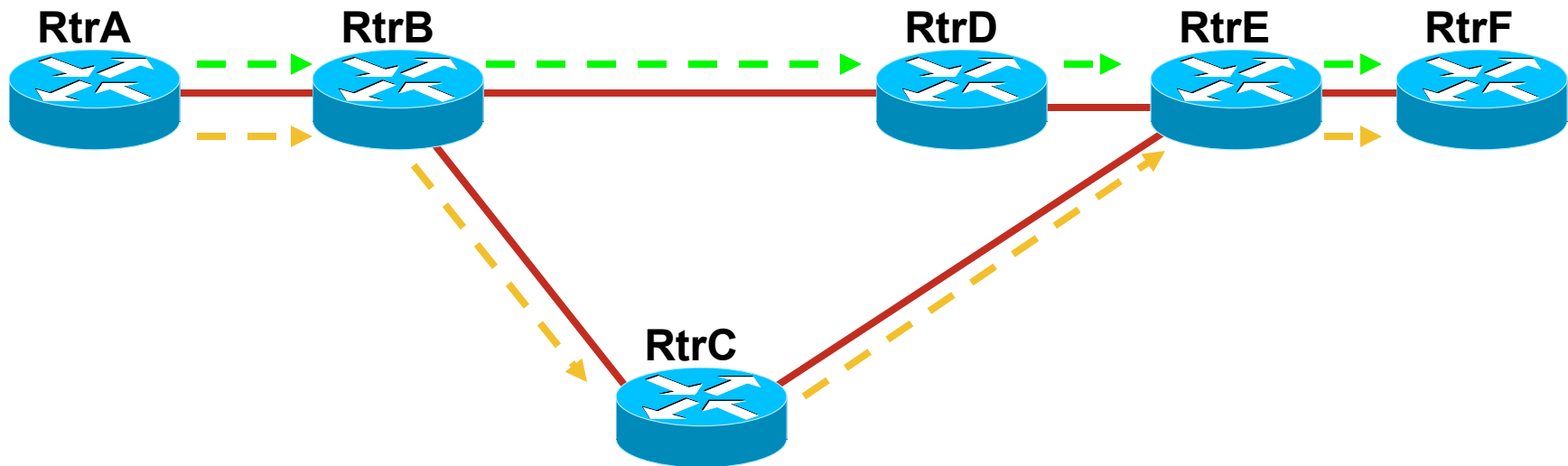


# Node Protection

- Node protection still has the same convergence properties as link protection
- Deciding where to place your backup tunnels is a much harder problem to solve large-scale  
...turns out it's an NP-complete problem.
- For small-scale protection, link may be better
- Cisco is developing tools to solve these hard problems for you (see TunnelVision, later)

# Path Protection

- Path Protection: multiple tunnels from TE head to tail, across diverse paths



# Path Protection

- Path Protection: least scalable, most resource-consuming, slowest convergence of all 3 protection schemes
- Path protection is useful in two places:
  - 1) when you have more links than tunnels
  - 2) when you need to protect links not using global label space

# Path vs. Local Protection

## Local (link/node) Protection

Thing	Dependency	Time
Link failure detection	Media- and platform-specific	~usecs (POS + APS)
Local switchover to protect tunnel	RP->IPC communication time	~few msec or less

## Path Protection

Thing	Dependency	Time
Link failure detection	Media- and platform-specific	~usecs (POS + APS)
Info propagation	IGP timers, network size, collective router load	~5-30sec
Headend switchover to protect LSP	network size, CPU load	~msec



# Verifying MPLS-TE



# TE show Commands

```
show mpls traffic-eng link-management admission-cont
show mpls traffic-eng link-management advertisements
show mpls traffic-eng link-management bandwidth-alloc
show mpls traffic-eng link-management igp-neighbors
show mpls traffic-eng link-management interfaces
show mpls traffic-eng link-management summary
show mpls traffic-eng forwarding-adjacency
show mpls traffic tunnel backup
show mpls traffic-eng fast-reroute database
show mpls traffic-eng tunnels
show mpls traffic-eng tunnels summary
```

# Verifying Tunnel Setup

- `sh int tun<x>`  
up/up == tunnel is up/LSP is successfully built
- `sh ip route <tunnel dest>`  
should be via the configured tunnel(s)
- `sh mpls traffic-eng tunnels`  
lots of options under here

# Verifying Tunnel Setup

- `sh int tun<x>`

up/up == tunnel is up/LSP is successfully built

```
VXR-6#sh int tun0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel0
  ...
  Encapsulation TUNNEL, loopback not set
  Keepalive set (10 sec)
  Tunnel source 192.168.1.6, destination 192.168.1.4
```

# Verifying Tunnel Setup

- `sh ip route <tunnel dest>`  
should be directly connected via tunnel

```
VXR-6#sh ip rou 192.168.1.4
Routing entry for 192.168.1.4/32
  Known via "isis", distance 115, metric 30, type level-2
  ...
Routing Descriptor Blocks:
* 192.168.1.4, from 192.168.1.4, via Tunnel0
  Route metric is 30, traffic share count is 1
```

# Verifying Tunnel Setup

- **sh mpls traffic-eng tunnels**

lots of options here

```
VXR-6#sh mpls traffic-eng tunnels ?
```

Tunnel	Tunnel interface
brief	Brief summary of tunnel status and configuration
destination	Restrict display to tunnels with this destination
name	Restrict display to tunnels with this name
role	Restrict display to tunnels with specified role
source-id	Tunnel identifier address/id
summary	Show summary information
up	Restrict display to tunnels in up state
<cr>	

# Verifying Tunnel Setup

- `sh mpls traffic-eng tunnels <intf>`

```
VXR-6#sh mpls traffic-eng tunnels Tunnel0
```

```
Name:VXR-6_t0 (Tunnel0) Destination: 192.168.1.4
```

```
Status:
```

```
Admin: up Oper: up Path: valid Signalling: connected
```

```
path option 10, type dynamic (Basis for Setup, path weight 20)
```

```
Config Paramters:
```

```
Bandwidth: 6 Priority: 7 7 Affinity: 0x0/0xFFFF
```

```
AutoRoute: enabled LockDown: disabled
```

```
.....
```

```
RSVP Path Info:
```

```
My Address: 192.168.4.6
```

```
Explicit Route: 192.168.4.5 192.168.6.5 192.168.6.4 192.168.1.4
```

```
Record Route: NONE
```

```
Tspec: ave rate=6 kbits, burst=1000 bytes, peak rate=6 kbits
```

# Verifying Tunnel Setup

- sh mpls traffic-eng tunnels brief

```
VXR-6#sh mpls traffic-eng tunnels brief
```

```
Signalling Summary:
```

```
    LSP Tunnels Process:          running
```

```
    RSVP Process:                running
```

```
    Forwarding:                  enabled
```

```
    Periodic reoptimization:      every 10 seconds, next in 5 seconds
```

TUNNEL NAME	DESTINATION	STATUS	STATE
VXR-6_t0	192.168.1.4	up	up
VXR-6_t1	192.168.1.4	up	up
VXR-4_t0	192.168.1.6	signalled	up

```
Displayed 2 (of 2) heads, 0 (of 0) midpoints, 1 (of 1) tails
```

# Verifying Tunnel Setup

- sh mpls traffic-eng tunnels summary

```
VXR-6#sh mpls tr tu summary
```

Signalling Summary:

LSP Tunnels Process:	running
RSVP Process:	running
Forwarding:	enabled
Head:	2 interfaces, 2 active signalling attempts, 2 established 24 activations, 22 deactivations
Midpoints:	0, Tails: 1
Periodic reoptimization:	every 10 seconds, next in 8 seconds



# MPLS TE Topology Database

```

PE3(7500-B)#sh mpls traffic-eng topology
My_System_id: 0000.0000.0001.00 (isis level-2)
Signalling error holddown: 10 sec Global Link Generation 29
IGP Id: 0000.0000.0001.00, MPLS TE Id:1.1.1.1 Router Node (isis level-2)
link[0]: Point-to-Point, Nbr IGP Id: 0000.0000.0002.00, nbr_node_id:3, gen:29
    frag_id 0, Intf Address:10.1.35.5, Nbr Intf Address:10.1.35.3
    TE metric:10, IGP metric:10, attribute_flags:0x0
    physical_bw: 155000 (kbps), max_reservable_bw_global: 155000 (kbps)
    max_reservable_bw_sub: 0 (kbps)
  
```

		Global Pool	Sub Pool
	Total Allocated	Reservable	Reservable
	BW (kbps)	BW (kbps)	BW (kbps)
	-----	-----	-----
bw[0]:	0	155000	0
bw[1]:	0	155000	0
bw[2]:	0	155000	0
bw[3]:	0	155000	0
bw[4]:	0	155000	0
bw[5]:	110000	45000	0
bw[6]:	0	45000	0
bw[7]:	0	45000	0

# MPLS-TE link-management

```
router#show mpls traffic-eng link-management ?
```

admission-control	Link Management admission-control
advertisements	Link Management advertisements
bandwidth-allocation	Link Management bandwidth-allocation
igp-neighbors	Link Management igp-neighbors
interfaces	Link Management Traffic Engineering interfaces
summary	Link Management summary

# MPLS-TE link-management (Cont)

```
router#sh mpls tra link-management admission-control
```

```
PO0/0          1/1          Resv Admitted   100000      R System
Information::
```

```
Tunnels Count:      62
```

```
Tunnels Selected:   62
```

TUNNEL ID	UP IF	DOWN IF	PRIORITY	STATE	BANDWIDTH
106.166.110.67 21431_	PO4/3	-	1/1	Resv Admitted	0
106.166.110.67 22431_	PO4/3	PO0/0	1/1	Resv Admitted	100000 R
106.166.110.68 21431_	PO4/3	-	1/1	Resv Admitted	0
106.166.110.68 22431_	PO4/3	PO0/0	1/1	Resv Admitted	100000 R
106.166.111.67 21431_	PO0/0	-	1/1	Resv Admitted	0

Head-end  
TE rtr-id

Tunnel-ID

## Note:

1. For the first entry we are the tail end because we have an UP IF & no DOWN IF. That's why we just admit the tunnel without any bandwidth requirement
2. For the second entry, we're the mid-point because we have an UP IF and a DOWN IF. We are reserving 1000000 kbps of BW & R => bandwidth reserved

# MPLS-TE link-management (Cont)

```
router#sh mpls tr link advertisements
```

```
Flooding Status:      ready
```

```
Configured Areas:     1
```

```
IGP Area[1] ID::  isis level-2
```

```
System Information::
```

```
Flooding Protocol:    ISIS
```

```
Header Information::
```

```
IGP System ID:  1061.6611.0067.00
```

```
MPLS TE Router ID: 106.166.110.67
```

```
Flooded Links:      1
```

```
Link ID::  0
```

```
Link IP Address:  106.166.110.122
```

```
IGP Neighbor:ID 1061.6611.0070.00,  
                IP 106.166.110.121
```

```
Admin. Weight:      2
```

```
Physical BW:        622000000 bits/sec
```

```
Reservable BW:      622000000 bits/sec
```

```
Output Bandwidth::
```

```
BW Unreserved[0]:   622000000 bits/sec
```

```
BW Unreserved[1]:   622000000 bits/sec
```

```
BW Unreserved[2]:   622000000 bits/sec
```

```
BW Unreserved[3]:   622000000 bits/sec
```

```
BW Unreserved[4]:   621669952 bits/sec
```

```
BW Unreserved[5]:   615589952 bits/sec
```

```
BW Unreserved[6]:   571040000 bits/sec
```

```
BW Unreserved[7]:   571040000 bits/sec
```

```
Affinity Bits       0x00000000
```

# MPLS-TE link-management (Cont)

```
router#show mpls traffic-eng link bandwidth-allocation pos1/0
```

```
System Information::
```

```
Links Count:          12
```

```
Bandwidth Hold Time:  max. 15 seconds
```

```
Link ID::  PO1/0 (106.166.254.197)
```

```
Link Status:
```

```
Physical Bandwidth:    2488000000 bits/sec
```

```
MPLS TE Bandwidth:     2488000000 bits/sec (reserved: 0% in, 28% out)
```

```
BW Descriptors:        398
```

```
MPLS TE Link State:    MPLS TE on, RSVP on, admin-up, flooded
```

```
Inbound Admission:     allow-all
```

```
Outbound Admission:    allow-if-room
```

```
Admin. Weight:         600 (IGP)
```

(contd...)

# MPLS-TE link-management (Cont)

IGP Neighbor Count: 1

Up Thresholds: 15 30 45 60 75 80 85 90 95 96 97 98 99 100 (default)

Down Thresholds: 100 99 98 97 96 95 90 85 80 75 60 45 30 15 (default)

Downstream Bandwidth Information (bits/second):

KEEP	PRIORITY	BW HELD	BW TOTAL	HELD	BW LOCKED	BW TOTAL	LOCKED
	0	0		0	0		0
	1	0		0	510000		510000
	2	0		0	20000		530000
	3	0		0	10660000		11190000
	4	0		0	144410000		155600000
	5	0		0	16470000		172070000
	6	0		0	517090000		689160000
	7	0		0	0		689160000

# MPLS-TE link-management (Cont)

```
router#sh mpls tr link igp-neighbors
      Link ID::  PO0/0
Neighbor ID:  1061.6611.0070.00 (area: isis level-2, IP: 106.166.110.74)
      Link ID::  PO1/0
Neighbor ID:  1061.6604.1065.00 (area: isis level-2, IP: 106.166.254.198)
      Link ID::  PO2/0
Neighbor ID:  1061.6611.4069.00 (area: isis level-2, IP: 106.166.110.109)
      Link ID::  PO3/0
Neighbor ID:  0642.1114.7028.00 (area: isis level-2, IP: 64.211.147.5)
      ..... . .
```

# MPLS-TE link-management (Cont)

```
router#show mpls traffic-eng link-management interfaces

      System Information::
      Links Count:          12
      Link ID::  PO0/0 (106.166.110.73)
      Link Status:
      Physical Bandwidth:    2488000000 bits/sec
MPLS TE Bandwidth:    2488000000 bits/sec (reserved: 0% in, 10% out)
      MPLS TE Link State:    MPLS TE on, RSVP on, admin-up, flooded
      Inbound Admission:     allow-all
      Outbound Admission:    allow-if-room
      Admin. Weight:         1 (IGP)
IGP Neighbor Count:    1  #of IGP neighbors reachable over this link
IGP Neighbor:          ID 106.166.110.74, IP 106.166.110.74 (Up)
      Flooding Status for each configured area [1]:
      IGP Area[1]:  isis level-2:  flooded
      .....

```



# MPLS-TE link-management (Cont)

```
wr1.sfo1#sh mpls tr link summ
```

```
System Information::
```

```
Links Count:          12
```

```
Flooding System:      enabled
```

```
IGP Area ID::  isis level-2
```

```
Flooding Protocol:    ISIS
```

```
Flooding Status:      data flooded
```

```
Periodic Flooding:    enabled (every 180 seconds)
```

```
Flooded Links:        9
```

```
IGP System ID:         1061.6611.0069.00
```

```
MPLS TE Router ID:     106.166.110.69
```

```
IGP Neighbors:         9
```

```
Link ID::  PO0/0 (106.166.110.73)
```

```
Link Status:
```

```
Physical Bandwidth:    2488000000 bits/sec
```

```
MPLS TE Bandwidth:     2488000000 bits/sec (reserved: 0% in, 10% out)
```

```
MPLS TE Link State:    MPLS TE on, RSVP on, admin-up, flooded
```

```
Inbound Admission:     allow-all
```

```
Outbound Admission:    allow-if-room
```

```
Admin. Weight:         1 (IGP)
```

## RSVP related

```
router#sh ip rsvp ?
```

host	RSVP Endpoint Senders and Receivers
installed	RSVP installed reservations
interface	RSVP interface information
neighbor	RSVP neighbor information
request	RSVP Reservations Upstream
reservation	RSVP Reservation Requests from Downstream
sender	RSVP Path State information
temp-psb decisions	RSVP PATH Requests awaiting Policy
temp-rsb decisions	RSVP Reservation Requests awaiting Policy

# RSVP commands

```
router#sh ip rsvp inst
```

```
RSVP: POS0/0
```

BPS	To	From	Protoc	DPort	Sport
100K	106.166.118.70	106.251.0.68	0	22431	10001
100K	106.166.118.70	106.166.112.68	0	22431	10001
100K	106.166.118.70	106.251.0.12	0	22431	10001
100K	106.166.118.70	106.166.151.8	0	22431	10001

```
RSVP: POS1/0 has no installed reservations
```

```
RSVP: POS4/3
```

BPS	To	From	Protoc	DPort	Sport
100K	106.166.119.162	106.166.118.70	0	21771	1
100K	106.166.119.163	106.166.118.70	0	22771	1

# At the headend: CEF lookup

```
router#sh ip cef 106.166.115.67 255.255.255.255  
106.166.115.67/32, version 975848  
0 packets, 0 bytes  
  via 106.166.115.67, Tunnel21921, 1 dependency  
    next hop 106.166.115.67, Tunnel21921  
    valid adjacency
```

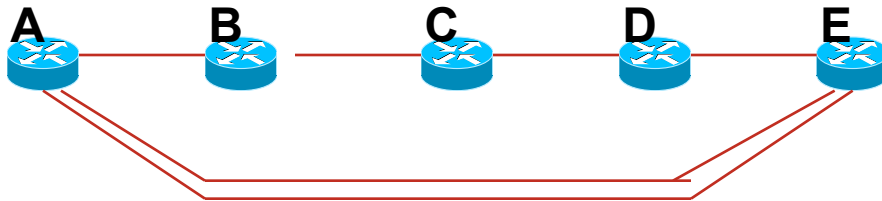
```
router#sh ip cef 106.166.115.67 255.255.255.255 int  
106.166.115.67/32, version 975848  
0 packets, 0 bytes  
has tag information: local tag tunnel head, shared tag information  
fast tag rewrite: Tu21921, point2point, tags imposed 28  
  via 106.166.115.67, Tunnel21921, 1 dependency  
    next hop 106.166.115.67, Tunnel21921  
    valid adjacency
```

# At midpoint: LFIB lookup

```
router#sh tag for det
```

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
26	Pop tag point2point	108.108.254.50	10232 [1896] 0	PO3/2	
MAC/Encaps=4/4, MTU=4474, Tag Stack{}					
0F008847					
27	727 point2point	103.172.134.96	10051 [1883] 3214340	PO2/0	
MAC/Encaps=4/8, MTU=4470, Tag Stack{727}					
0F008847 002D7000					

# Packet forwarding via tunnel



**A has a tunnel to E**

**A's lo0 - 106.166.110.67**

**E's lo0 - 106.166.115.67**

```
A#sh ip cef 106.166.115.67 255.255.255.255 int
106.166.115.67/32, version 975848
0 packets, 0 bytes
has tag information: local tag tunnel head, shared tag information
fast tag rewrite: Tu21921, point2point, tags imposed 28
via 106.166.115.67, Tunnel21921, 1 dependency
next hop 106.166.115.67, Tunnel21921
valid adjacency
```

# Packet forwarding via tunnel

A is the origin of the packet - So there is no incoming tag - Puts an outgoing tag of 28

For B - it is an incoming tag (28) - with an outgoing tag (182)

For C - the incoming tag (182) - swaps with an outgoing tag (105)

For D - the incoming tag (105) - is the penultimate hop - so POPS the tag and sends as an IP packet.

# Packet forwarding via tunnel

```
A#sh mpls tra tunn tun21921
```

```
Explicit Route: 106.166.110.121      106.166.110.109      106.166.253.141  
                  B                      C                      D
```

```
B#sh tag for | include 106.166.110.67
```

```
28      182      106.166.110.67 21921 [1] 7987      PO8/0  
point2point
```

```
C#sh tag for | include 106.166.110.67
```

```
182      105      106.166.110.67 21921 [1] 7987      PO3/0  
point2point
```

```
D#sh tag for | include 106.166.110.67
```

```
105      Pop tag      106.166.110.67 21921 [1] 7855      PO3/0  
point2point
```



# Packet forwarding via tunnel: ping

```
A#ping
```

```
Target IP address: 106.166.115.67
```

```
Repeat count [5]: 1
```

```
Datagram size [100]:
```

```
Sending 1, 100-byte ICMP Echos to 106.166.115.67, timeout is 2  
seconds:
```

```
!
```

```
Success rate is 100 percent (1/1), round-trip min/avg/max =  
64/64/64 ms
```

# Packet forwarding via tunnel: ping

```
B# sh mpls for | include 106.166.110.67
```

Before the ping

```
28    182    106.166.110.67 21921 [1] 7987      PO8/0
    point2point
```

After the ping

```
28    182    106.166.110.67 21921 [1] 8095      PO8/0
    point2point
```

$8095 - 7987 = 108$

100 byte DATA

8 byte HEADER

# Packet forwarding via tunnel: ping

```
C#sh tag for | include 106.166.110.67
```

Before the ping

```
182    105    106.166.110.67 21921 [1] 7987      PO3/0
point2point
```

After the ping

```
182    105    106.166.110.67 21921 [1] 8095      PO3/0
point2point
```

$8095 - 7987 = 108$

100 byte DATA

8 byte HEADER

# Packet forwarding via tunnel: ping

```
D#sh tag for | include 106.166.110.67
```

Before the ping

```
105  Pop tag    106.166.110.67 21921 [1] 7855    PO3/0  
    point2point
```

After the ping

```
105  Pop tag    106.166.110.67 21921 [1] 7959    PO3/0  
    point2point
```

$7959 - 7855 = 104$

100 byte DATA

4 byte HEADER

