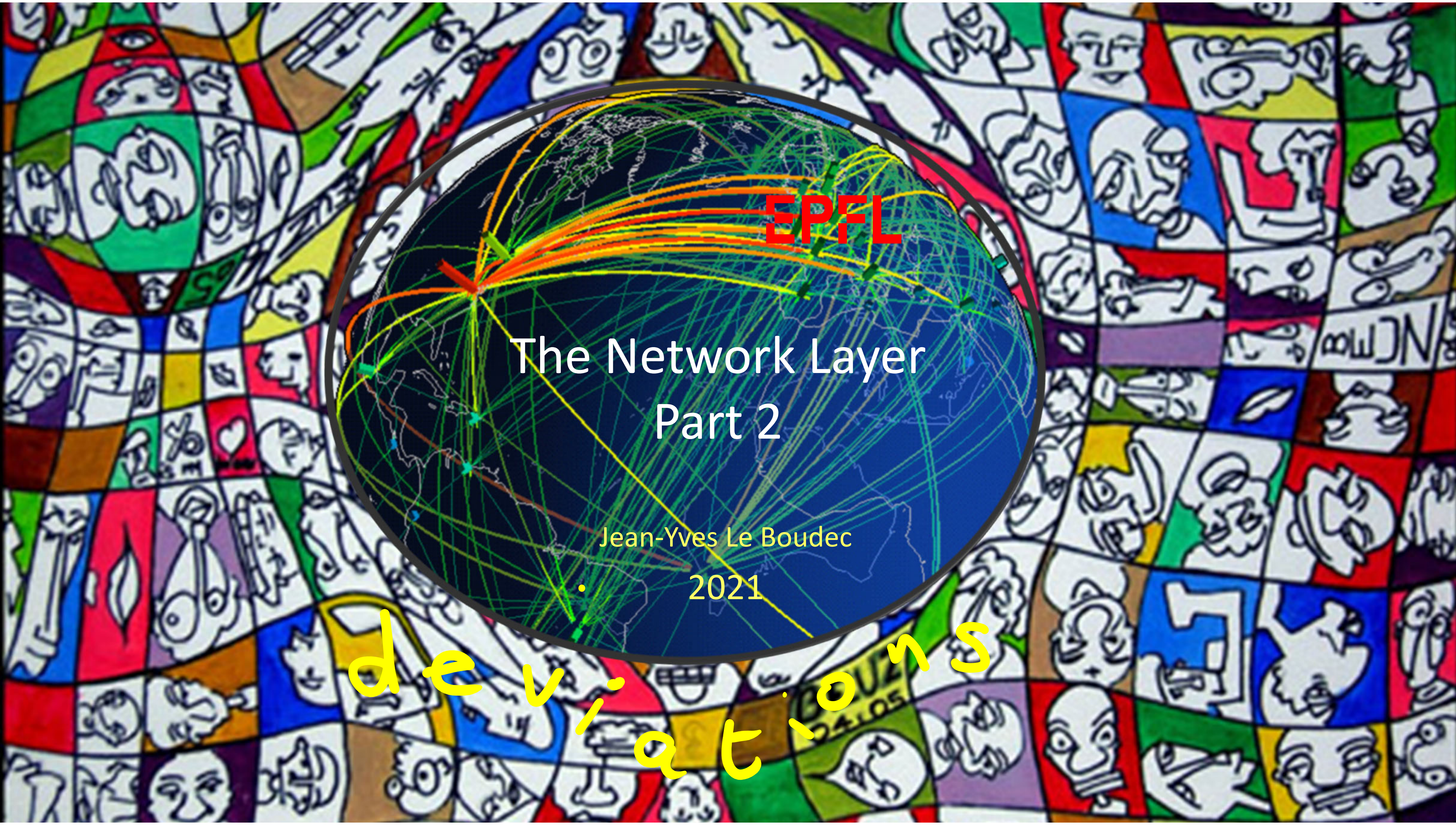


EPFL

The Network Layer Part 2

Jean-Yves Le Boudec

2021

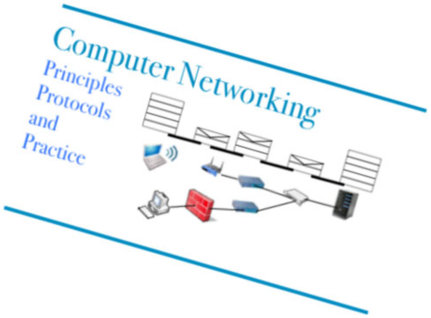
deviations

Contents

9. Proxy ARP
10. Fragmentation
11. Tunnels
12. 6to6 over 4: Tunnel Brokers and Teredo
13. 4to4 over 6: 464XLAT
14. NAT64 and DNS64
15. MPLS

Textbook

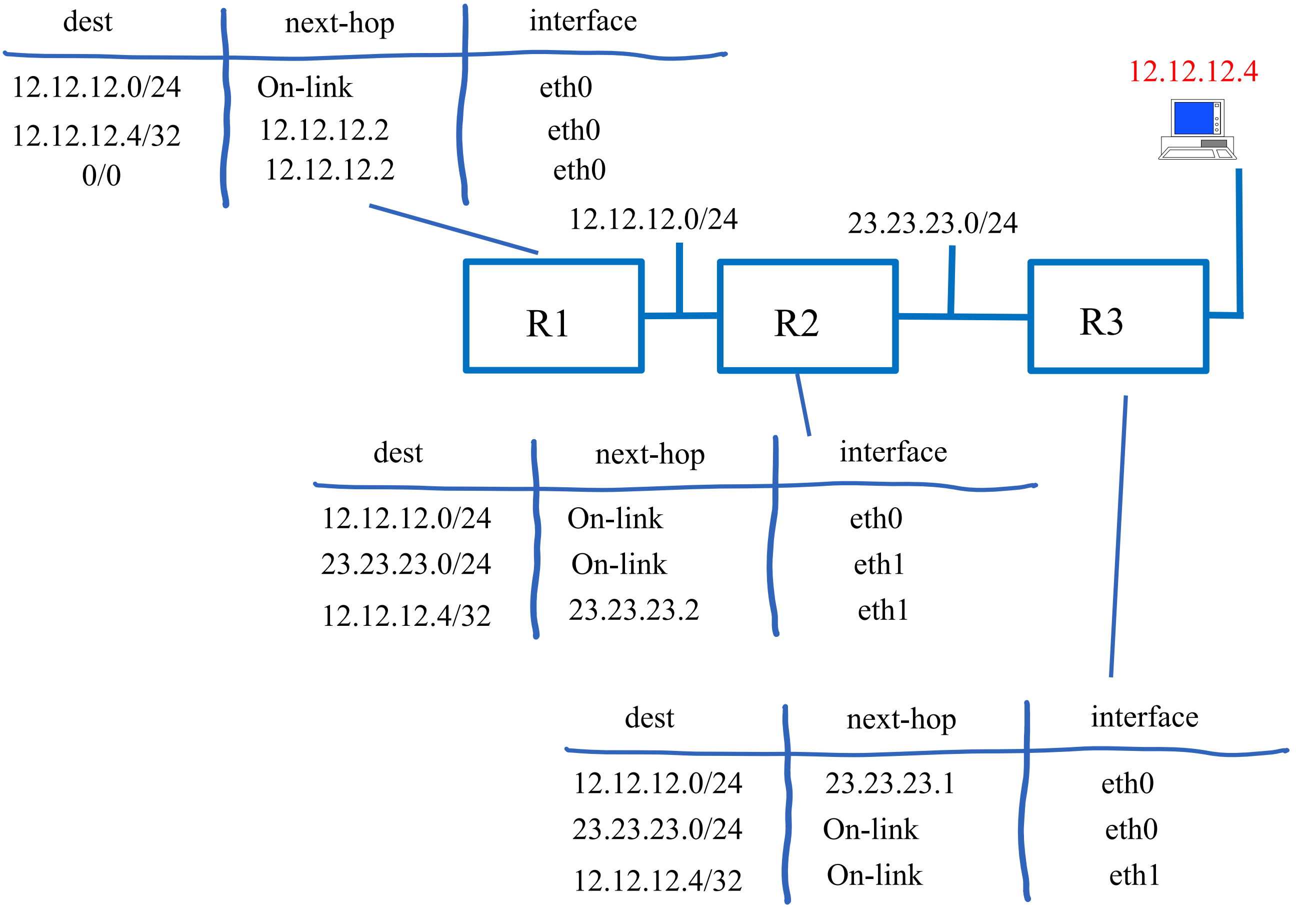
Chapter 5: The Network Layer



9. Proxy-ARP

Reminder: IP principle says one subnet = one LAN

Assume you want to cheat with this principle, e.g. 12.12.12.4 is in the wrong place, but you want to keep it that way. You can support this configuration with /32 entries (with IPv4) in forwarding tables
at R1, R2 and R3 (“Host Routes”)



dest	next-hop	interface
12.12.12.0/24	On-link	eth0
12.12.12.4/32	12.12.12.2	eth0
0/0	12.12.12.2	eth0

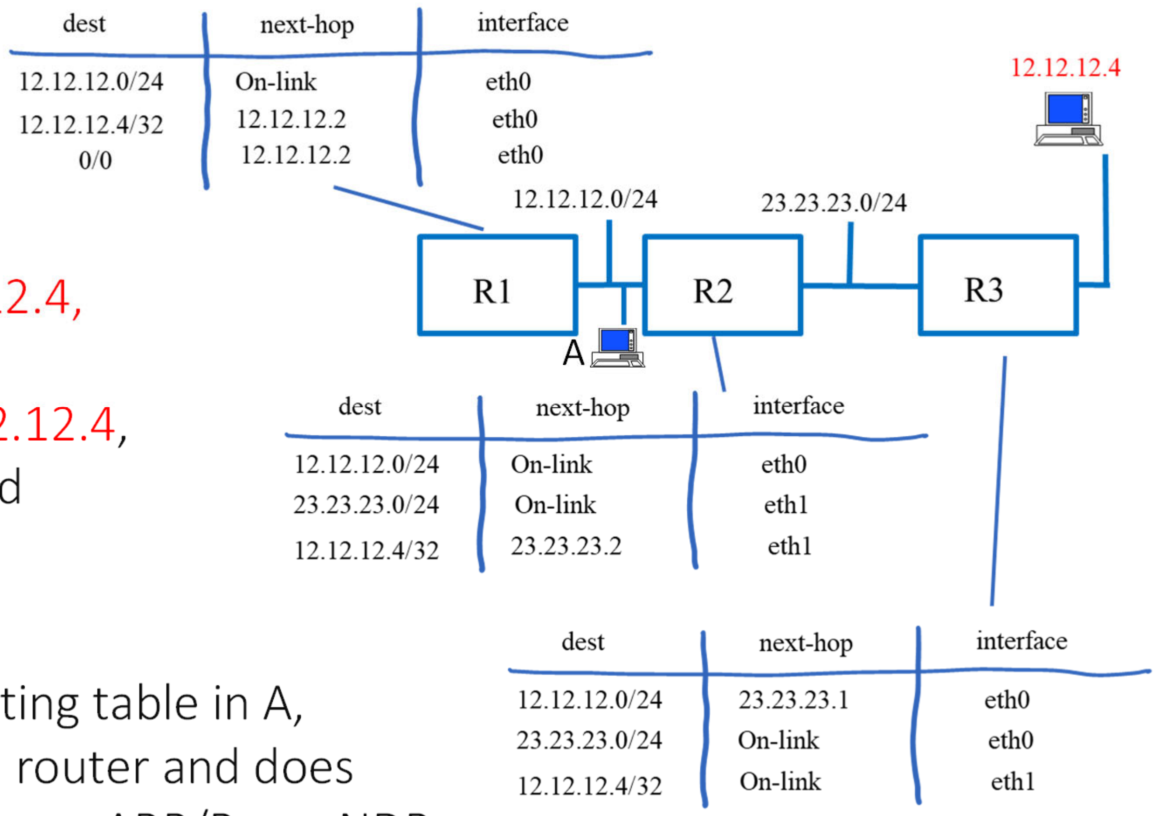
dest	next-hop	interface
12.12.12.0/24	On-link	eth0
23.23.23.0/24	On-link	eth1
12.12.12.4/32	23.23.23.2	eth1

dest	next-hop	interface
12.12.12.0/24	23.23.23.1	eth0
23.23.23.0/24	On-link	eth0
12.12.12.4/32	On-link	eth1

Does this solve the problem ?

For packets handled by R1, R2 and R3, yes.

Not for host A
(in LAN between R1 and R2):
when A has a packet to send to 12.12.12.4,
A believes 12.12.12.4 is in the same subnet and sends an ARP REQ for 12.12.12.4, which is useless. A receives no reply and declares 12.12.12.4 unreachable.



dest	next-hop	interface
12.12.12.0/24	On-link	eth0
12.12.12.4/32	12.12.12.2	eth0
0/0	12.12.12.2	eth0

dest	next-hop	interface
12.12.12.0/24	On-link	eth0
23.23.23.0/24	On-link	eth1
12.12.12.4/32	23.23.23.2	eth1

dest	next-hop	interface
12.12.12.0/24	23.23.23.1	eth0
23.23.23.0/24	On-link	eth0
12.12.12.4/32	On-link	eth1

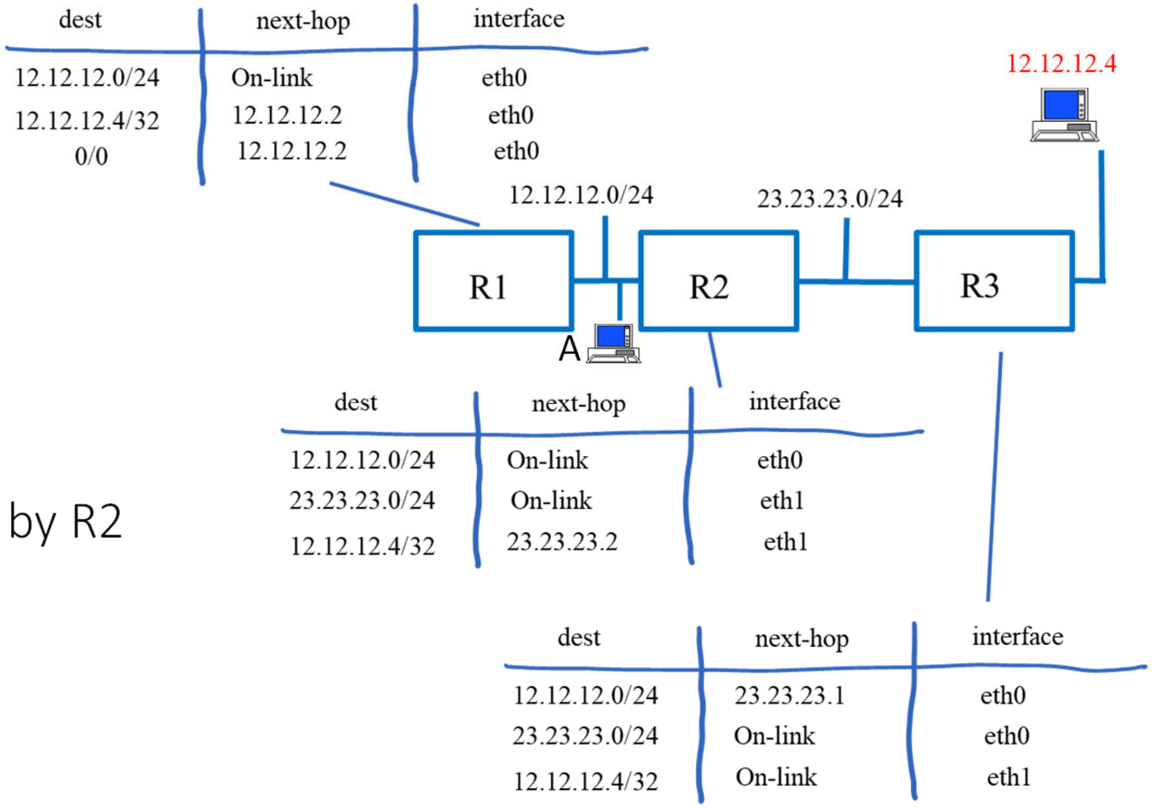
One solution is to write a complete routing table in A, but this is not usually done as A is not a router and does participate in IGP. Another solution is Proxy ARP/Proxy NDP.

Proxy ARP / ND Proxy

R2 performs Proxy ARP in lieu of 12.12.12.4: i.e. R2 answers all ARPs as if it were 12.12.12.4.

When A has an IP packet to send to 12.12.12.4, it sends an ARP REQ on its LAN, with target IP address = 12.12.12.4. R2 responds with R2’s MAC address. IP Packet sent by A is received by R2 and forwarded to R3.

R2 is a *proxy* for ARP requests sent to 12.12.12.4 on the LAN of A.

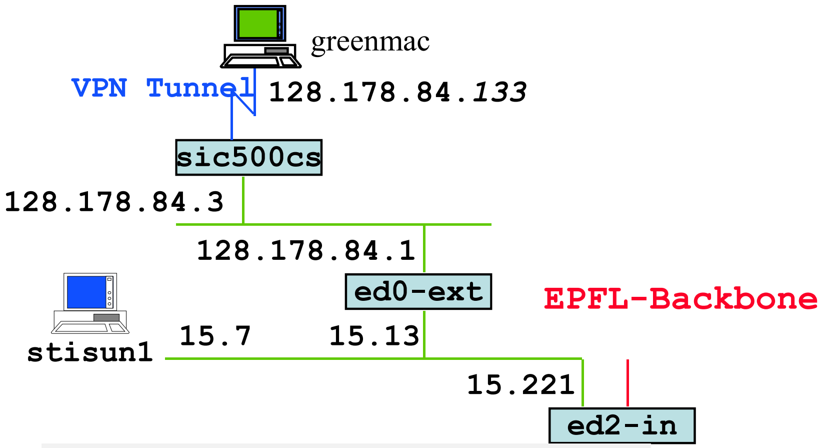


dest	next-hop	interface
12.12.12.0/24	On-link	eth0
12.12.12.4/32	12.12.12.2	eth0
0/0	12.12.12.2	eth0

dest	next-hop	interface
12.12.12.0/24	On-link	eth0
23.23.23.0/24	On-link	eth1
12.12.12.4/32	23.23.23.2	eth1

dest	next-hop	interface
12.12.12.0/24	23.23.23.1	eth0
23.23.23.0/24	On-link	eth0
12.12.12.4/32	On-link	eth1

sic500cs is a router (not a bridge); ed0-ext has a packet to 128.178.84.133 and sends an ARP REQ



1. All subnets have 255.255.255.0 netmask
2. Subnet 84 is on both sides of sic500cs
3. sic500cs does PROXY-ARP on behalf of greenmac

A. sic500cs replies with own MAC address

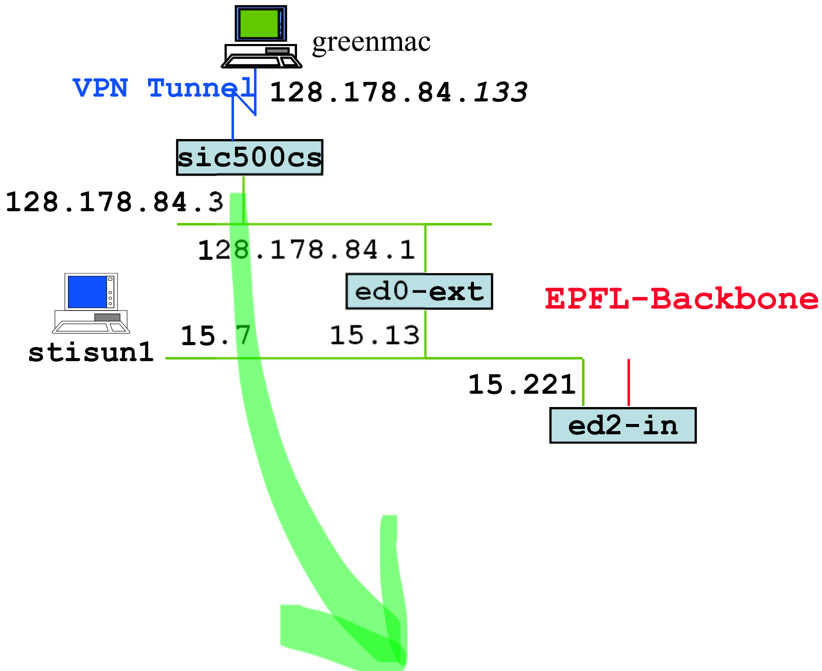
B. sic500cs replies with the MAC address of greenmac

C. Greenmac replies with his MAC address

D. Greenmac replies with the MAC address of sic500cs

E. I don’t know

Solution



dest	next-hop	interface
128.178.84.0/24	On-link	eth0
128.178.84.128/25	On-link	eth1
0.0.0.0/0	128.178.84.1	eth0

Answer A

sic500cs responds with own MAC adddress –
ed0-ext now believes that greenmac has the same MAC address as sic500cs

All traffic to greenmac is sent to sic500cs

sic500cs needs appropriate entry in routing table

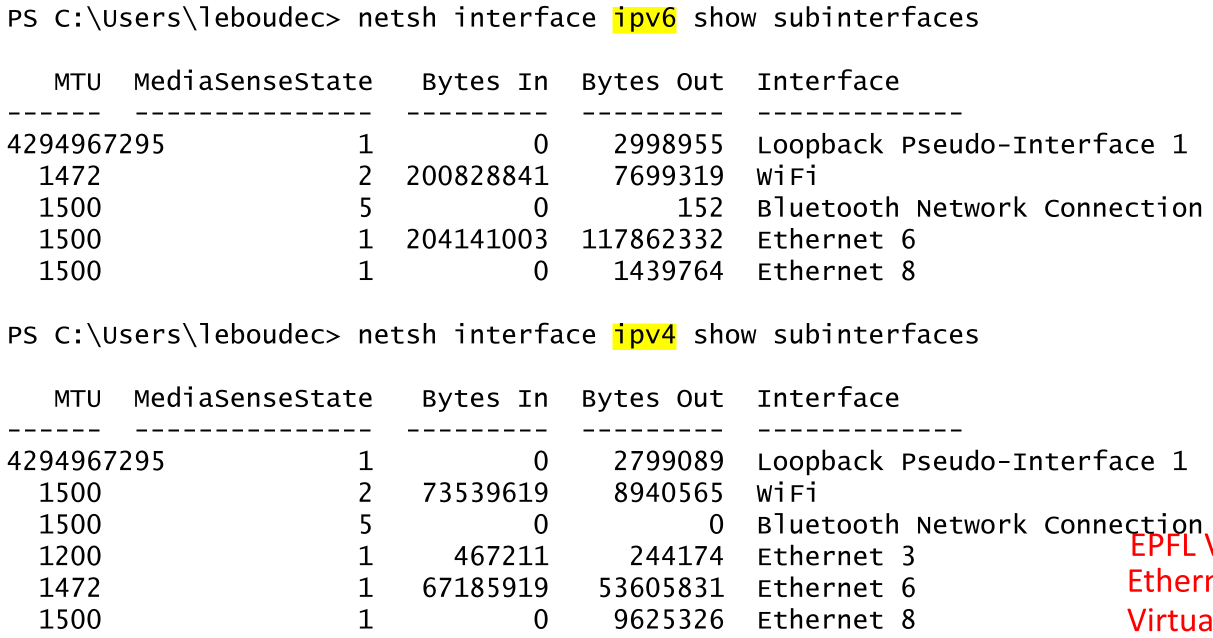
10. Fragmentation

Link-layer networks have different maximum frame lengths

MTU (maximum transmission unit) = maximum frame size usable for an IP packet (including the IP header)

MAC layer options and tunnels reduce MTU

Example	MTU
Ethernet	1500 B
Loopback	4 GB
Fibre channel	9036 B
IPv6 in tunnel	1472 B
EPFL VPN tunnel	1200 B
Zigbee with encryption	80 B



```
PS C:\Users\leboudec> netsh interface ipv6 show subinterfaces

   MTU  MediaSenseState   Bytes In  Bytes Out  Interface
------  ---------------  ---------  ---------  -------------
4294967295                1          0    2998955  Loopback Pseudo-Interface 1
  1472                2  200828841    7699319  WiFi
  1500                5          0        152  Bluetooth Network Connection
  1500                1  204141003  117862332  Ethernet 6
  1500                1          0    1439764  Ethernet 8

PS C:\Users\leboudec> netsh interface ipv4 show subinterfaces

   MTU  MediaSenseState   Bytes In  Bytes Out  Interface
------  ---------------  ---------  ---------  -------------
4294967295                1          0    2799089  Loopback Pseudo-Interface 1
  1500                2   73539619    8940565  WiFi
  1500                5          0          0  Bluetooth Network Connection
  1200                1     467211     244174  Ethernet 3
  1472                1   67185919   53605831  Ethernet 6
  1500                1          0    9625326  Ethernet 8
```

EPFL VPN
Ethernet
Virtual Box

Fragmentation

Hosts or routers may have IP datagrams larger than MTU.

Fragmentation is performed at source (IPv4 and IPv6) or at routers (IPv4 only) when IP datagram is too large.

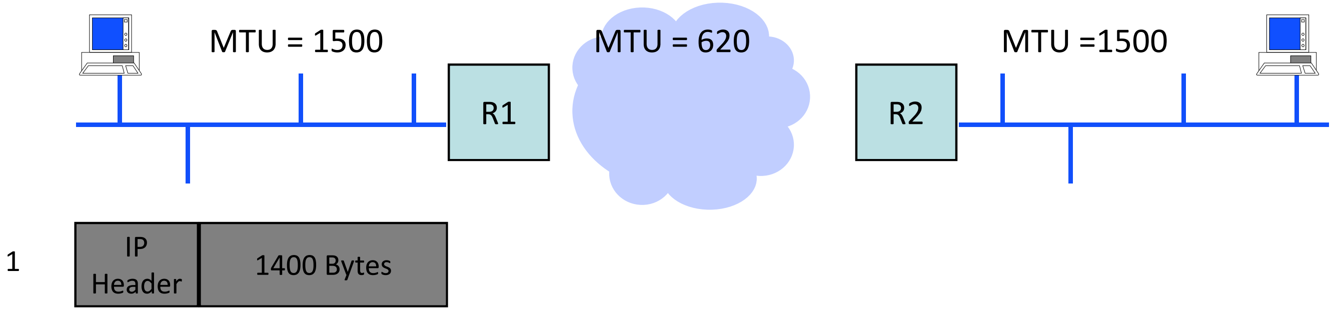
New applications should avoid fragmentation (see later).

IPv6 Routers never fragment – drop packet if too large, but IPv4 routers may.

Re-assembly is only at destination, never at routers.

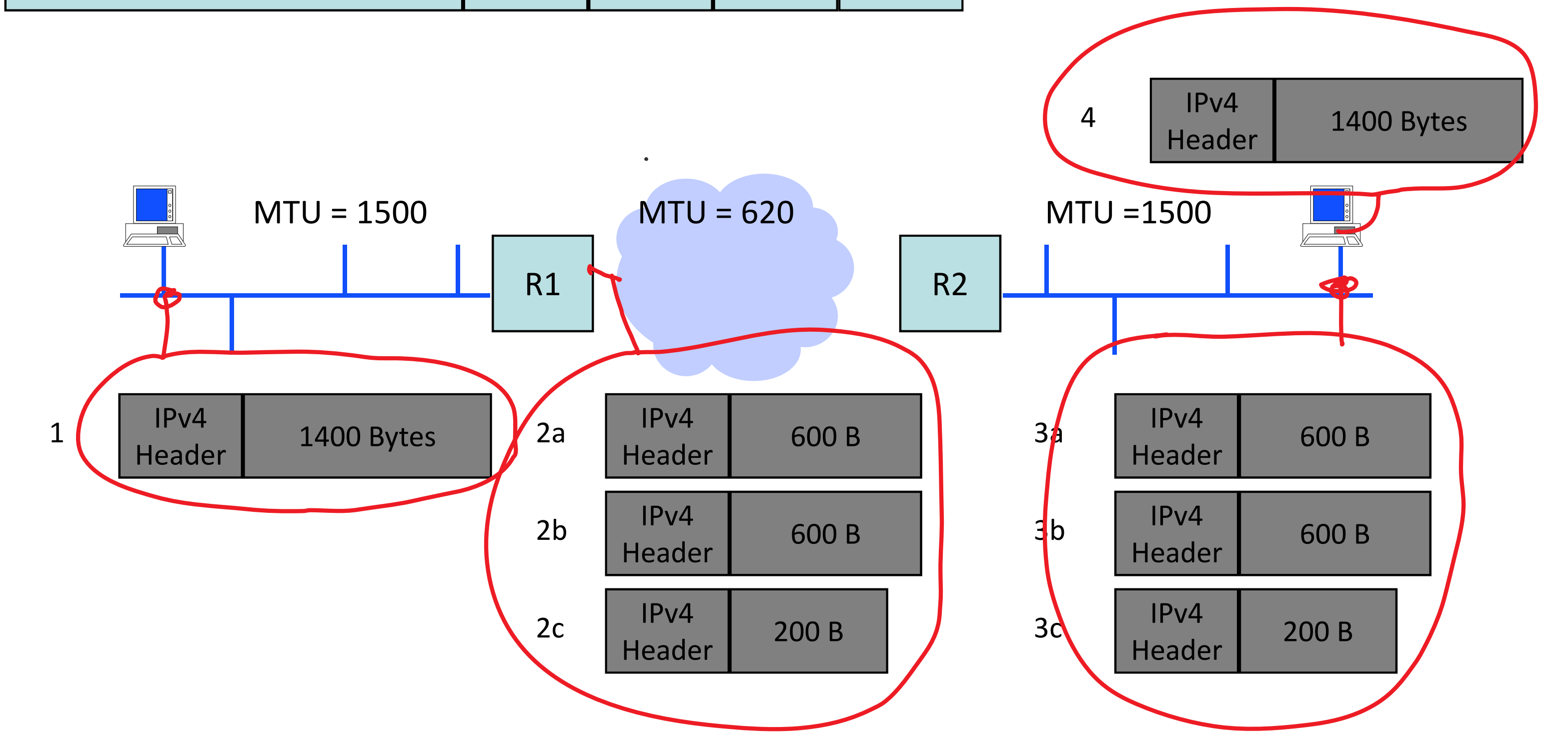
All fragments are self-contained IP packets.

IP *datagram* = original ; IP *packet* = fragment or complete datagram.



IPv4 Fragmentation Example

IPv4 header fields	1 and 4	2a, 3a	2b,3b	2c, 3c
Length	1420	620	620	220
Identification	567	567	567	567
More Fragment flag	0	1	1	0
Offset	0	0	75	150
8 * Offset	0	0	600	1200

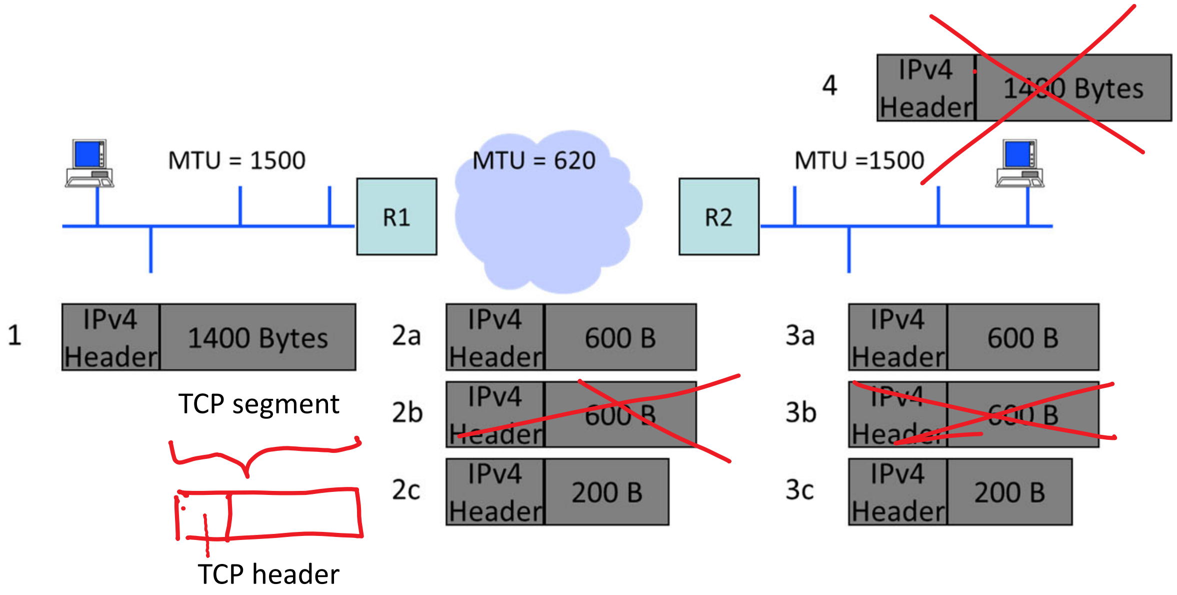


Fragment data size (here 600) is always a multiple of 8
Identification given by source

One TCP segment is contained in one IPv4 datagram that is fragmented by a router on its way from source to destination. One of the fragments is lost. What will TCP re-transmit ?

A. The bytes that were in the missing fragment
B. The bytes that were in all fragments of the datagram, missing or not
C. It depends whether the loss is detected by fast retransmit or by timeout
D. I don’t know

Solution



If the TCP segment is in several fragments and one of the fragments is lost, the whole segment is lost and will be re-transmitted

Fragmentation/re-assembly is done at IP layer, invisible to transport layer ! This is largely inefficient.

When a host generates UDP traffic, the port number is always present in all packets

A. True

B. False

C. True with IPv4, false with IPv6

D. True with IPv6, false with IPv4

E. I don’t know

Solution

Answer B: False.

If fragmentation occurs, only the first fragment contains the port number.

- This is a problem for NATs. Typically, when a NAT sees a first fragment, it caches (IP addresses, identification, port numbers).
  When a subsequent fragment arrives, the port numbers can be recovered from the cache.
  If the first received fragment is not the first fragment of the original datagram, this does not work and the fragment is discarded.
- This is also a problem for firewalls / intrusion detection systems

  Hard to do deep packet inspection.

Avoiding Fragmentation

Applications (UDP or TCP) should avoid fragmentation by estimating the Path MTU i.e. the max MTU of all links between source and destination.

One (suboptimal) method: use the default MTU:

- All IPv6 implementations must have MTU ≥ 1280 Bytes
- All IPv4 implementations must have MTU ≥ 68 Bytes but most should have MTU ≥ 576 bytes

A better method: Packetization Layer Path MTU Discovery (PLPMTUD):

- Observe the largest possible Path-MTU that works by observing packets whose reception was confirmed (acknowledged by TCP, or using the logic of the UDP app). Keep this information in the IP layer (correspondent table).
- Start from default MTU, from time to time, try a larger MTU.
- If destination is onlink, path MTU should be equal to the interface MTU.

TCP connections should negotiate an MSS (maximum segment size) such that

$$\text{MSS} \leq \text{PathMTU} - \text{IP header size} - \text{TCP header size}$$

11. Tunnels

Definition:
a *tunnel*, also called *encapsulation* occurs whenever a communication layer carries packets of a layer that is not the one above.

e.g.: IP packet in UDP
IP in TCP4
PPP(layer 2) packet in UDP
IPv4 in IPv6
IPv6 in IPv4

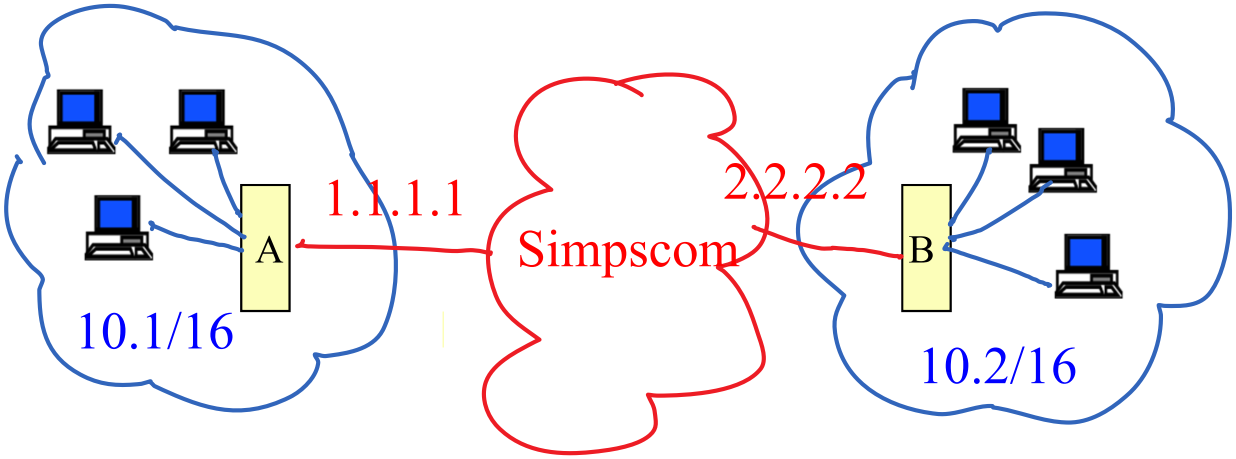
Why used ?

In theory: never

In practice: security / private networks / IPv6-IPv4 interworking

Homer’s Network

Homer deploys 10.x addresses in two sites and wants to interconnect them as one (closed) private network

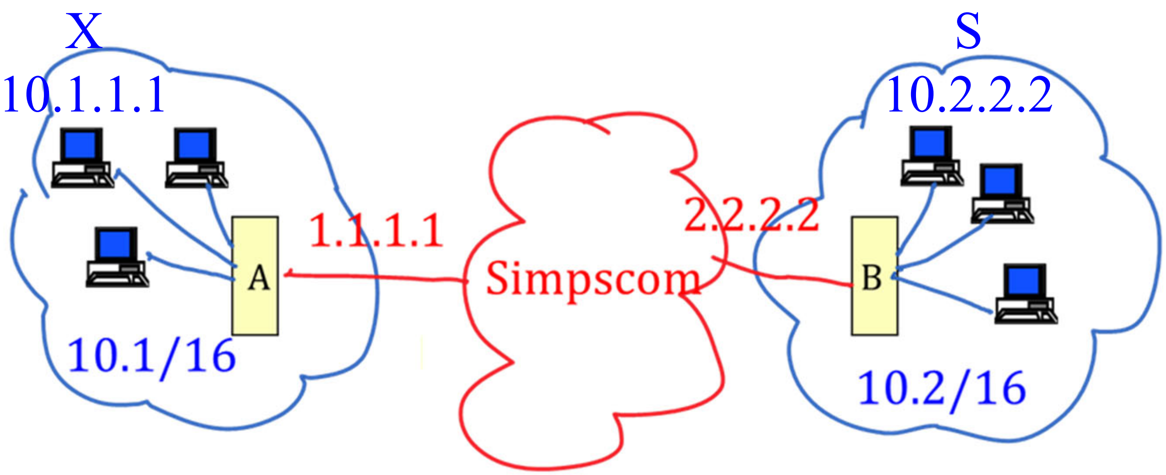


How can Homer use Simpscom’s network for that ?

IP packets with destination or source address 10.x.x.x cannot be sent to the public internet !

Wide-spread solution: Virtual Private Networks.

Example 1: Homer uses an IP over IP Tunnel



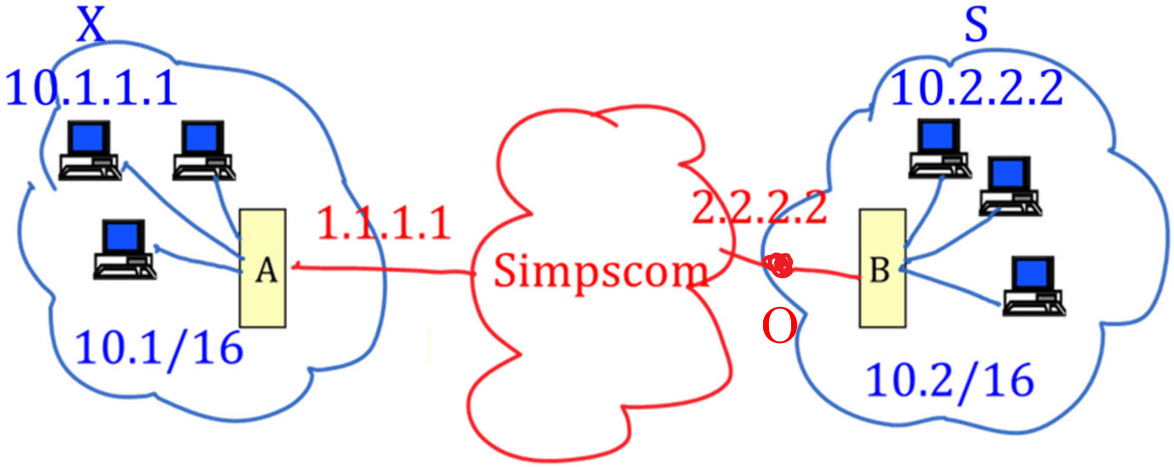
Homer configures a virtual interface in A (eth$x$); Associates this interface with an IP in IP tunnel, with endpoint 2.2.2.2

Similar stuff in B

Homer has a network with 2 routers and one virtual physical link; Homer configures routing tables at A and B (or runs a routing protocol).

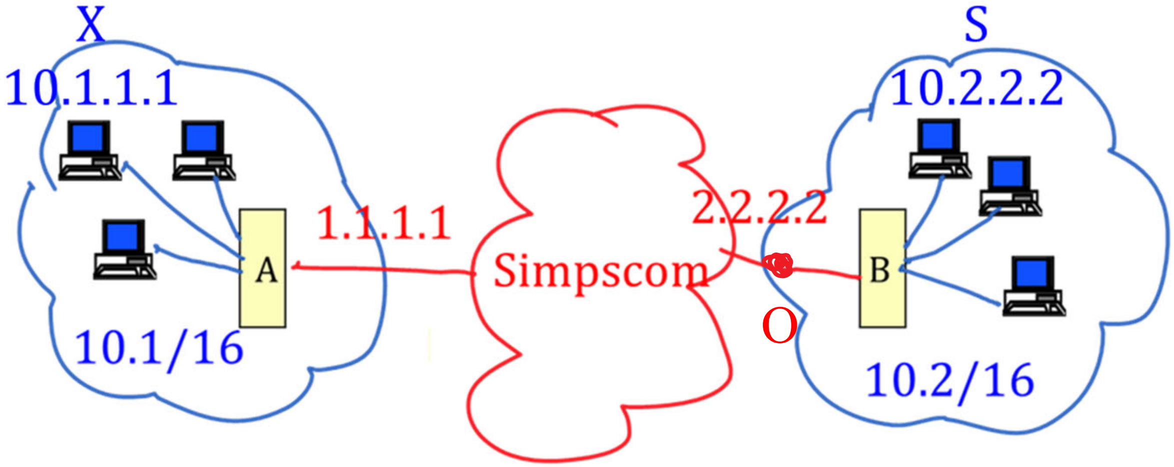
IP Packets from S to X are carried inside IP packets across Simpscom.

S sends a UDP packet to X.
What are the IP destination address and protocol at O ?



A. IP dest addr = 1.1.1.1, protocol = 17 (UDP)
B. IP dest addr = 10.1.1.1, protocol = 17 (UDP)
C. None of the above
D. I don’t know

Solution



1. The IP destination address is the tunnel endpoint 1.1.1.1
2. The protocol is not UDP but 04 (IPv4)

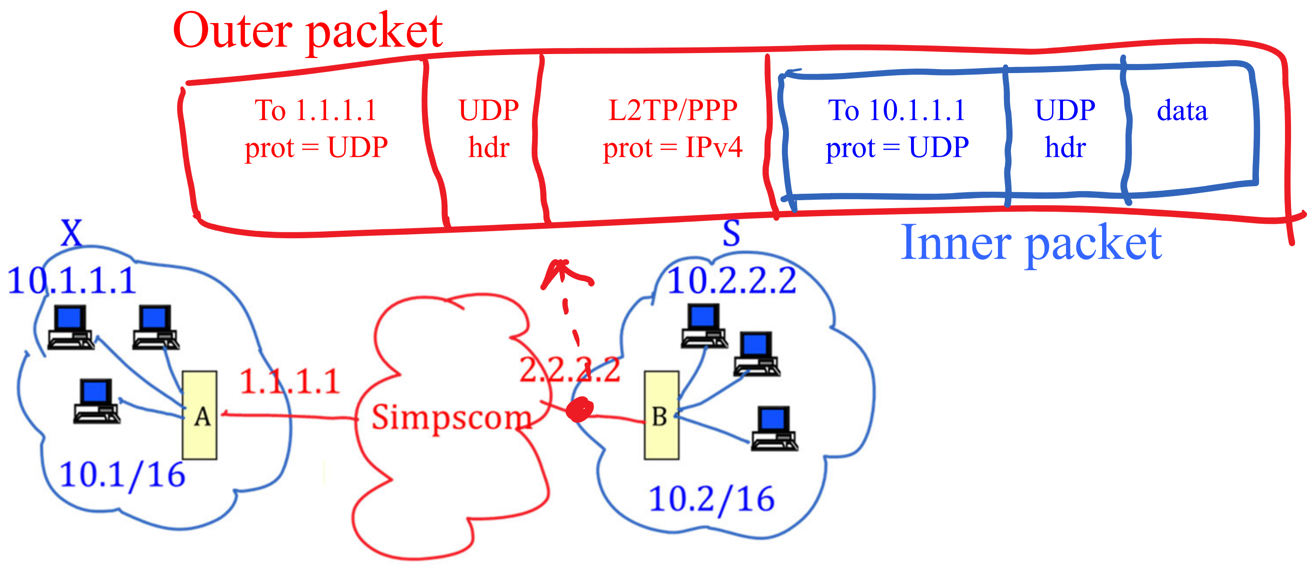
Answer C

Homer’s IP in IP solution is often replaced by IP in UDP

Some firewalls kill IP in IP packets

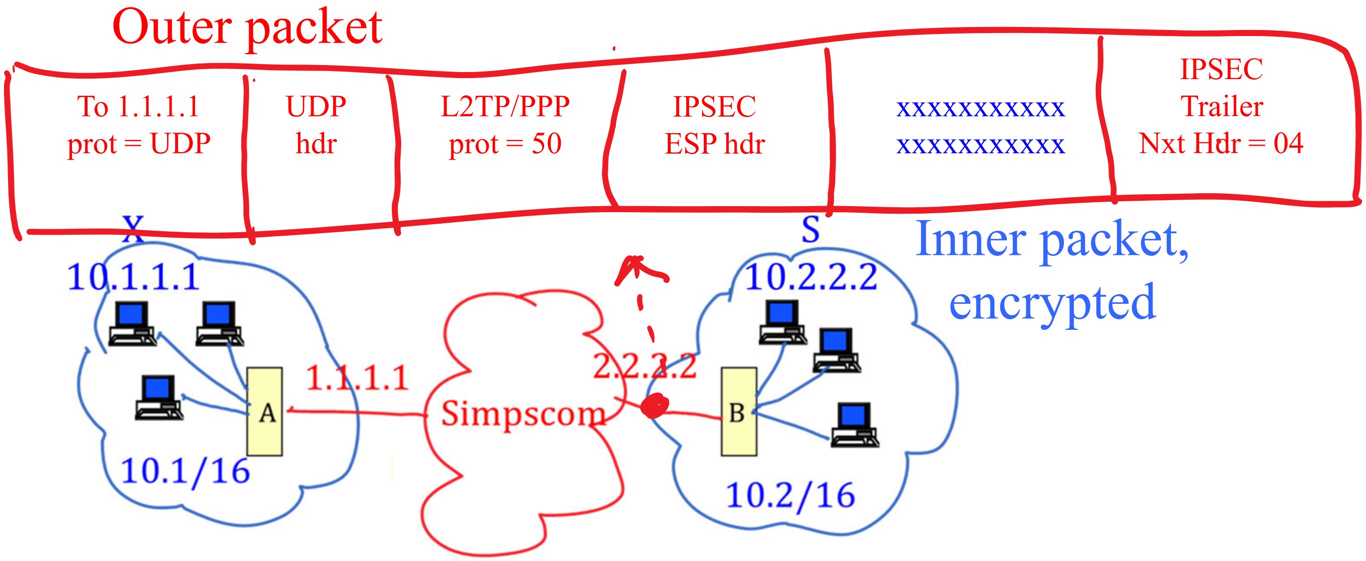
Therefore the tunnel is inside UDP

This requires a layer 2 header as well (to identify the protocol type) called L2TP / PPP



Bart does the same as Homer but wants a secure channel. He uses IPSEC.

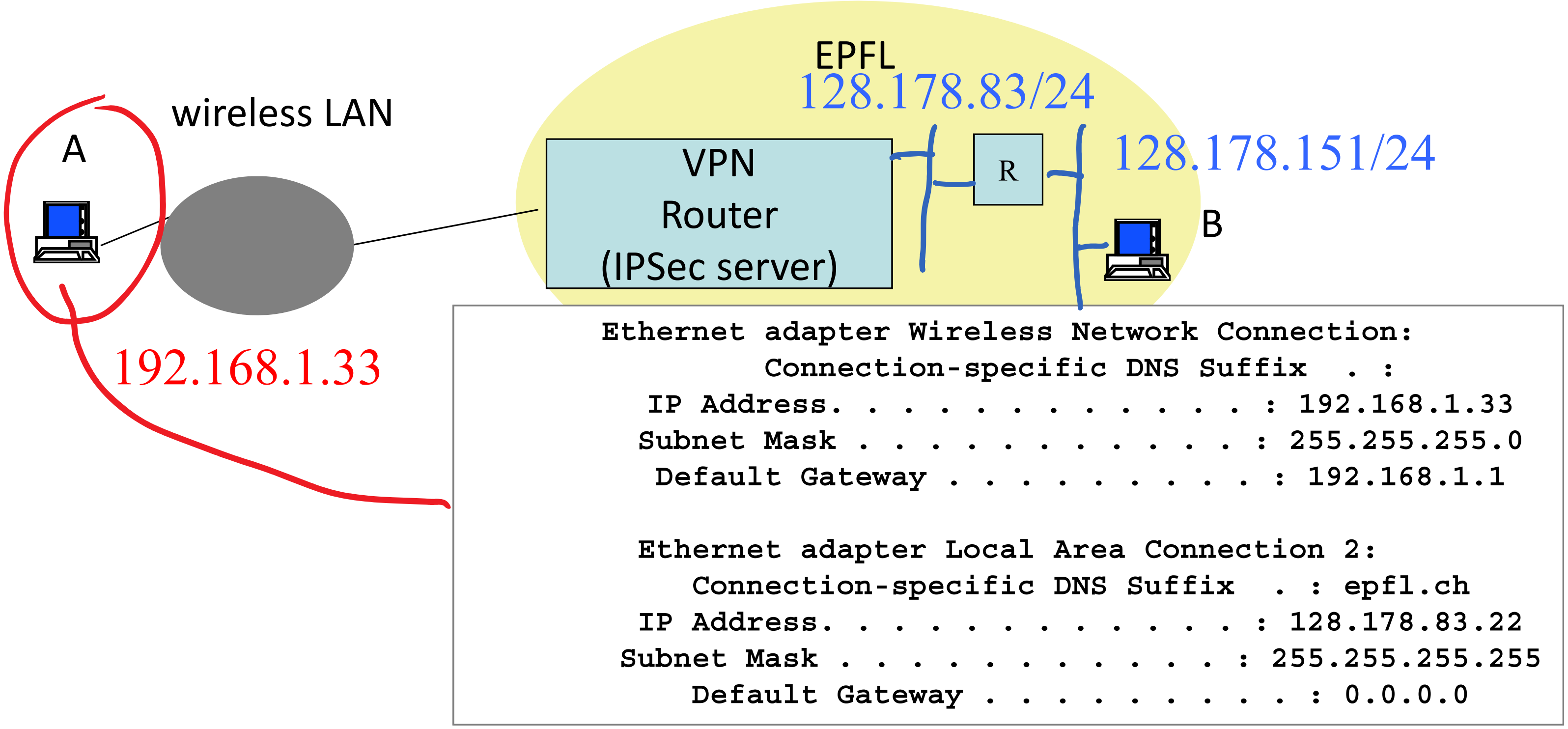
«IPSEC / ESP tunnel mode» encrypts the inner IP packet



This form of tunneling is called «L2TP/IPSEC VPN» (Virtual Private Network)

Variants (OpenVPN): IP in TLS over TCP ; IP in TLS over UDP

How does a packet from B to A find its way ?



```
                Ethernet adapter Wireless Network Connection:
                         Connection-specific DNS Suffix  . :
                   IP Address. . . . . . . . . . . . : 192.168.1.33
                  Subnet Mask . . . . . . . . . . . : 255.255.255.0
                   Default Gateway . . . . . . . . . : 192.168.1.1

                  Ethernet adapter Local Area Connection 2:
                     Connection-specific DNS Suffix  . : epfl.ch
                  IP Address. . . . . . . . . . . . : 128.178.83.22
                 Subnet Mask . . . . . . . . . . . : 255.255.255.255
                     Default Gateway . . . . . . . . . : 0.0.0.0
```

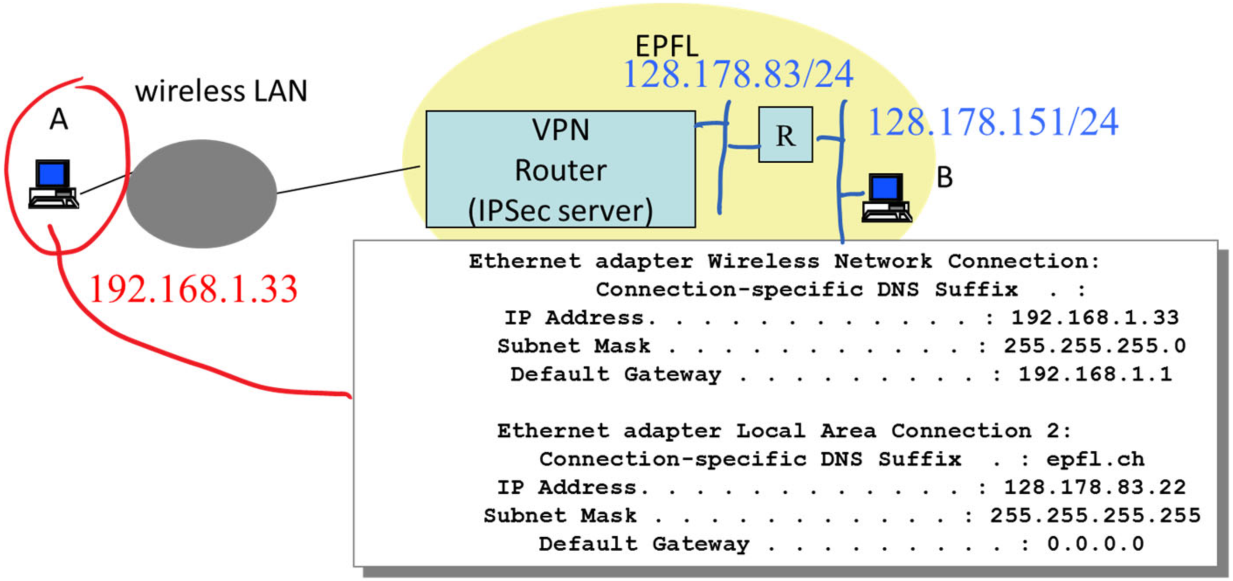
A. VPN router does proxy-ARP

B. R has a host route to A

C. Nothing special, the IGP takes care of it

D. I don’t know

Solution



```
Ethernet adapter Wireless Network Connection:
         Connection-specific DNS Suffix  . :
  IP Address. . . . . . . . . . . . : 192.168.1.33
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

  Ethernet adapter Local Area Connection 2:
     Connection-specific DNS Suffix  . : epfl.ch
  IP Address. . . . . . . . . . . . : 128.178.83.22
 Subnet Mask . . . . . . . . . . . : 255.255.255.255
     Default Gateway . . . . . . . . : 0.0.0.0
```

Answer A

A has two interfaces: one physical, with address 192.168.1.33, one virtual (tunnel) interface with address 128.178.83.22 (for example)

A appears to be on 128.178.83/24

VPN router does proxy ARP on behalf of A

R does not need a host route (but VPN Router may need one)

12. 6to6 over 4: Tunnel Brokers

IPv4 and IPv6 are incompatible

- v4 only host cannot handle IPv6 packets
- v6 only host cannot handle IPv4 packets

Pauvre petit! il a perdu la raison!
IPv6
Mais non, mais non: votre chapeau est très beau, lui aussi.
IPv4

What needs to be solved:

interworking: h6 to h4

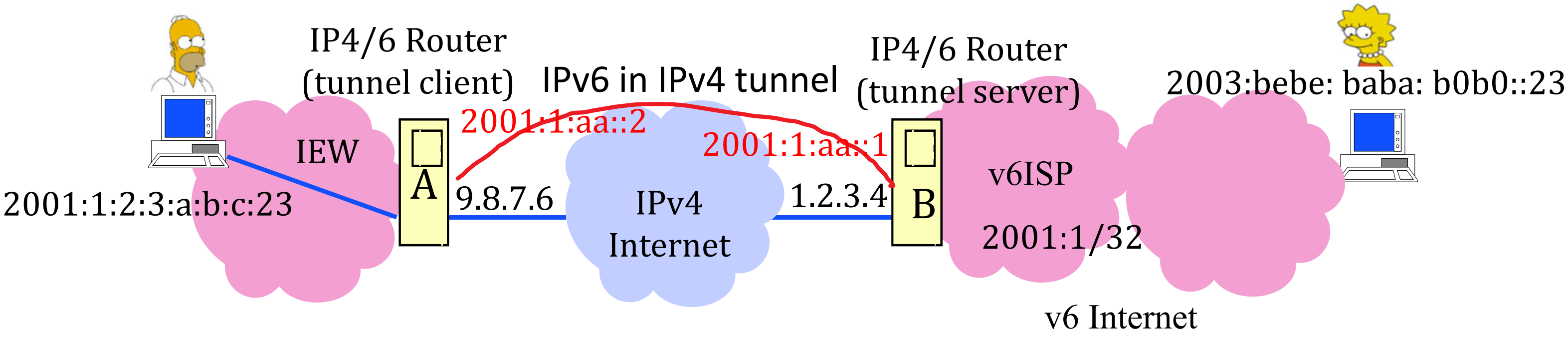
like-to-like access

- 6 to 6 over 4
- 4 to 4 over 6

In this and the next section we study mainly like-to-like

6 to 6 over 4 Solution: Tunnel Broker

Problem: 6to6 over 4 (early adopter) Homer runs IPv6 in local network, wants to connect to v6 hosts, but receives only IPv4 service from ISP



One solution: Tunnel Broker uses IPv6 in IPv4 encapsulation

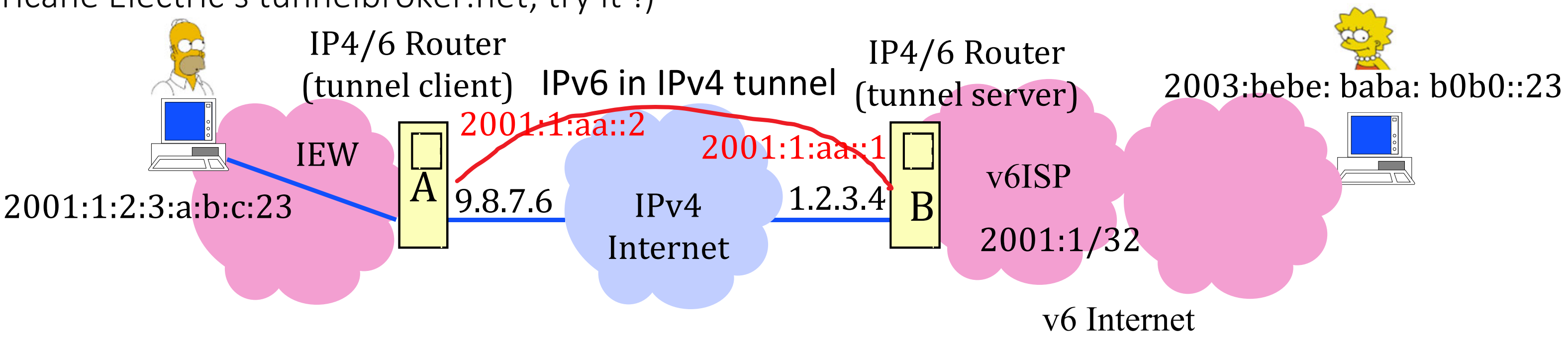
Static tunnel configured at A; provided by v6ISP

(e.g. Hurricane Electric’s tunnelbroker.net; try it !)

6 to 6 over 4 Solution: Tunnel Broker

Tunnel Broker uses IPv6 in IPv4 encapsulation

Static tunnel configured at A; provided by v6ISP
(e.g. Hurricane Electric’s tunnelbroker.net; try it !)



v6ISP delegates to IEW an IPV6 prefix e.g. 2001:1:2:3/64

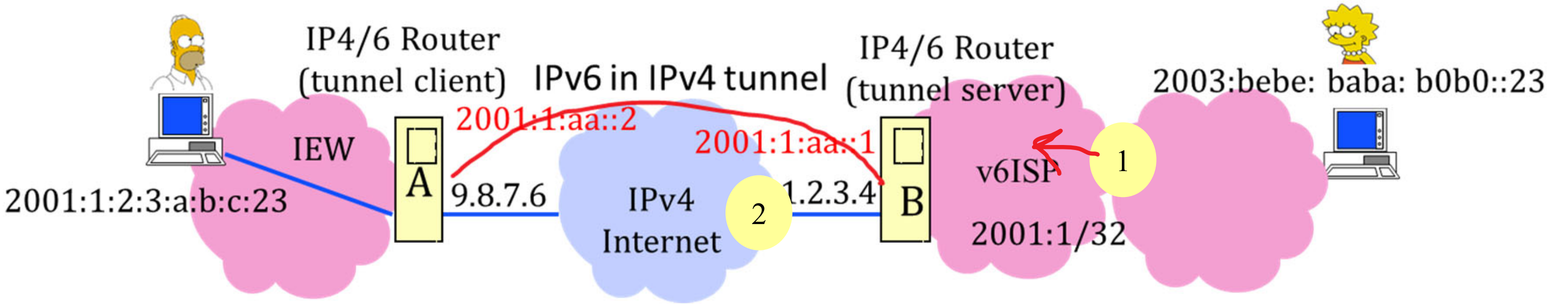
v6ISP assigns the IPv6 addresses of tunnel end-points e.g. 2001:1:aa::2 and 2001:1:aa::1

IEW can have multiple subnets

A’s IPv6 default route is 2001:1:aa::1

To Lisa, Homer appears to be a customer of v6ISP

Lisa sends one packet to Homer; what do we see at (1) ?



A. The IPv4 destination address in the encapsulating packet is 9.8.7.6

B. The IPv4 destination address in the encapsulating packet is 1.2.3.4

C. The IPv6 destination address in the packet is 2001:1:aa::2

D. The IPv6 destination address in the packet is 2001:1:2:3:a:b:c:23

E. A and C

F. B and C

G. A and D

H. B and D

I. I don't know

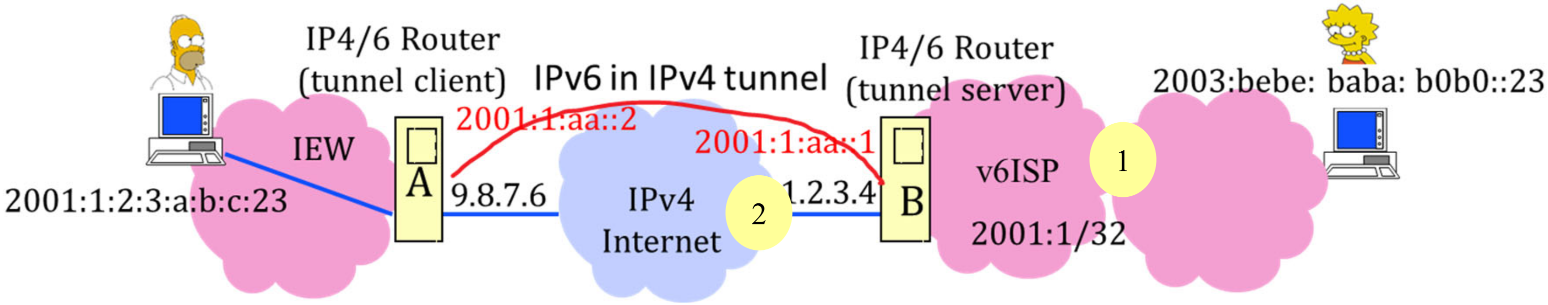
Solution

Answer D
To Lisa, Homer appears to be on the v6Internet. The packet is a regular IPv6 packet and its destination address is Homer’s IPv6 address.

There is no encapsulated packet at this point.
Encapsulation is performed by B and is removed by A.

Between A and B (e.g. at (2)) we see an IPv4 packet with destination address 9.8.7.6, source address 1.2.3.4, protocol type = 41. Inside the IPv4 packet we see an IPv6 packet with destination address = Homer’s IPv6 address.

All links are Ethernet v2 with MTU = 1500 Bytes. Assume all hosts perform Path-MTU and discover the best possible Path-MTU value. What is the value of Path-MTU between Lisa and Homer ?



A. 1500 Bytes

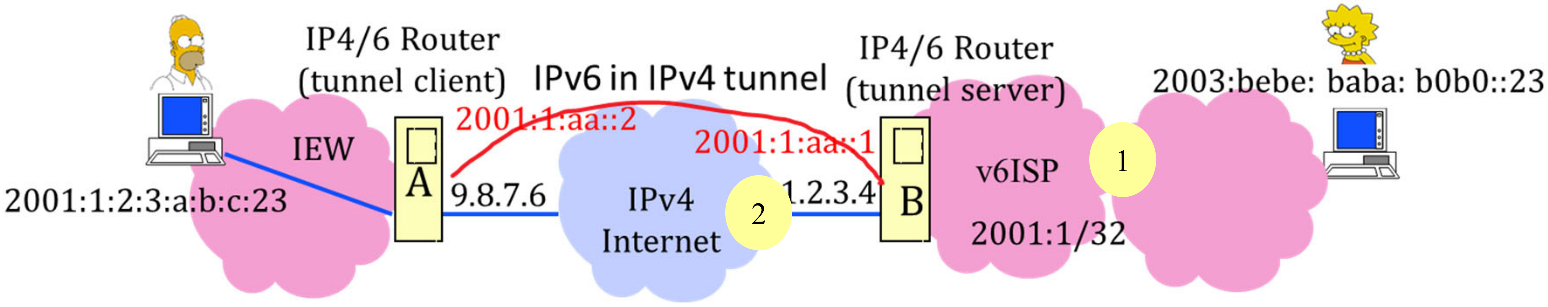
B. 1480 Bytes

C. 1460 Bytes

D. None of these

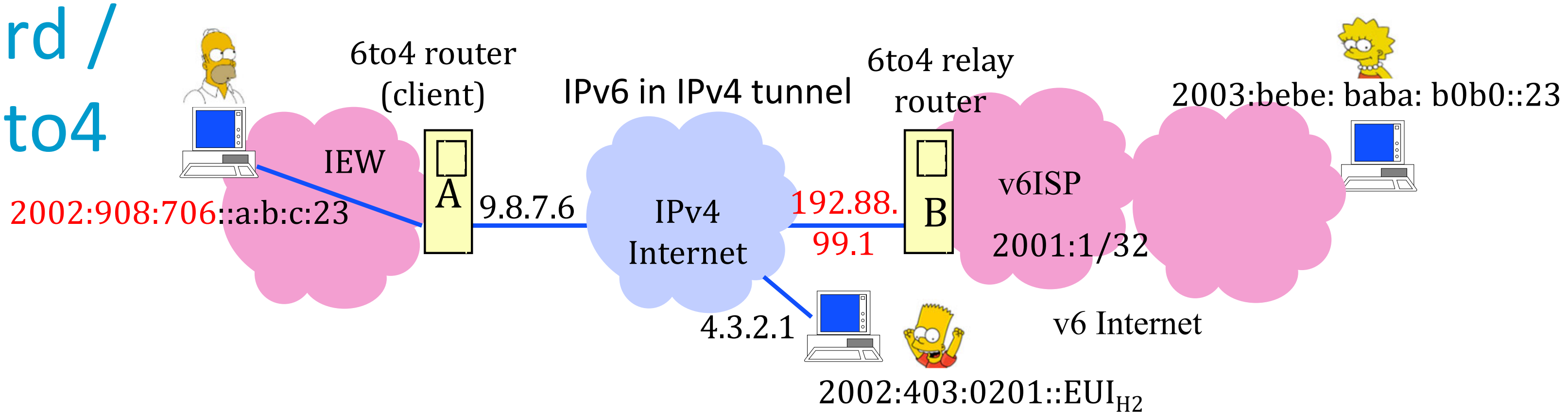
E. I don’t know

Solution



IPv6 packet of Lisa is encapsulated in an IPv4 packet by B. The encapsulation adds the IPv4 header, i.e. 20 bytes. The Path-MTU for Lisa to Homer is 1480 Bytes. Same thing in the reverse direction.

6rd / 6to4



6to4 and 6rd are similar to tunnel brokers, but the required prefixes/addresses are computed without any configuration (automatic tunnels)

e.g. with 6to4, IPv6 prefix of IEW is 2002:908:706/48

to any valid IPv4 address n, 6to4 associates the IPv6 prefix 2002:n/48

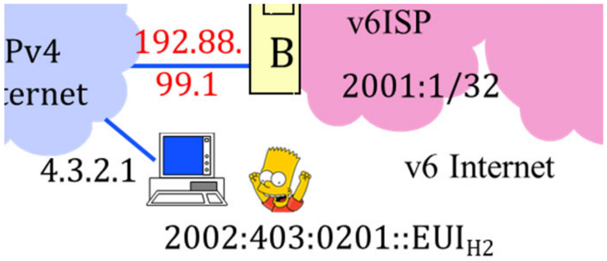
the IPv4 address 192.88.99.1 (remote end of tunnel) is associated with all 6to4 relay routers (anycast use of unicast address)

2002::/16 and 192.88.99/24 are reserved for 6to4.

6rd is similar but the blocks 2002::/16 and 192.88.99/24 are provider-dependent

6rd replaces 6to4 (deprecated) – used by operators to provide IPv6 access by change of software in telecom box.

Teredo



6to4 or 6rd can be implemented in Telecom box (NAT) or in host.
If no support in telecom box, 6to4 and 6rd require manual configuration of NATs.

Teredo is an alternative (invented by Microsoft) that works with NATs

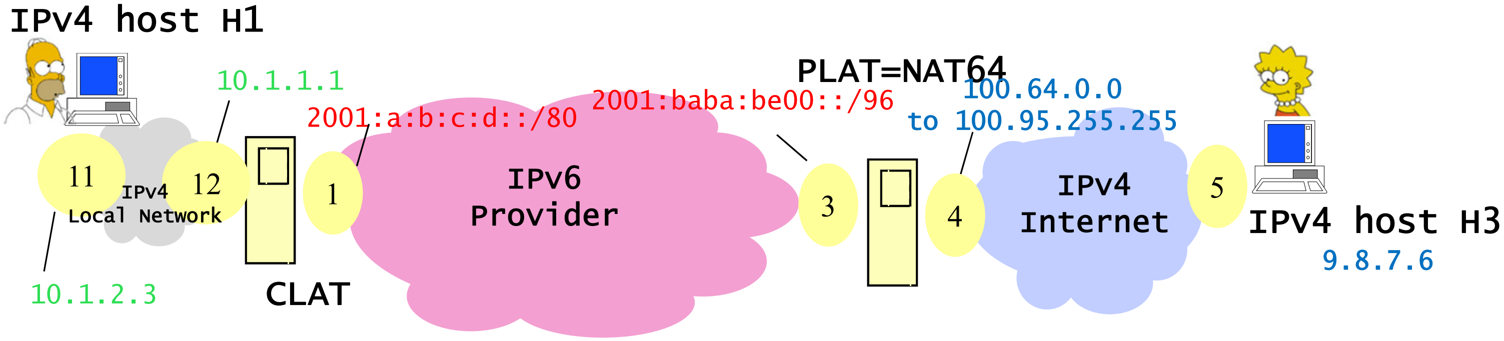
Uses :

- address block 2001:0/32
- Tunnels (IPv6 in UDP in IPv4) (UDP is used to be compatible with existing NAT and firewall filtering rules)
- relay routers (called « teredo relays »)
- teredo *servers* -> for solving the NAT mapping problem

Linux implementation is called miredo.

Experimental, no longer supported by Microsoft. But still useful !

13. 4to4 over 6: 464XLAT



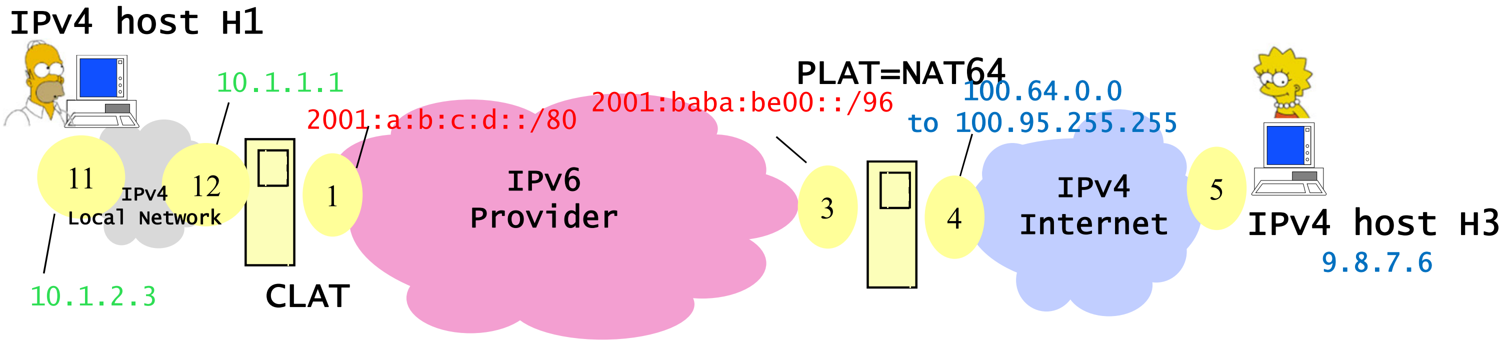
Problem: 4to4 over 6: (Legacy Problem) Homer’s device is IPv4, Homer receives only IPv6 service from ISP and still wants to communicate with v4 host H3.

Homer’s home network and device can run IPv6 but that does not solve all problems:

IPv4-only applications (skype)
IPv4-only remote correspondents (google scholar)

One Solution: 464XLAT ; uses NATs (XLAT = translation, no tunnels)

464XLAT: Customer-Side Translation



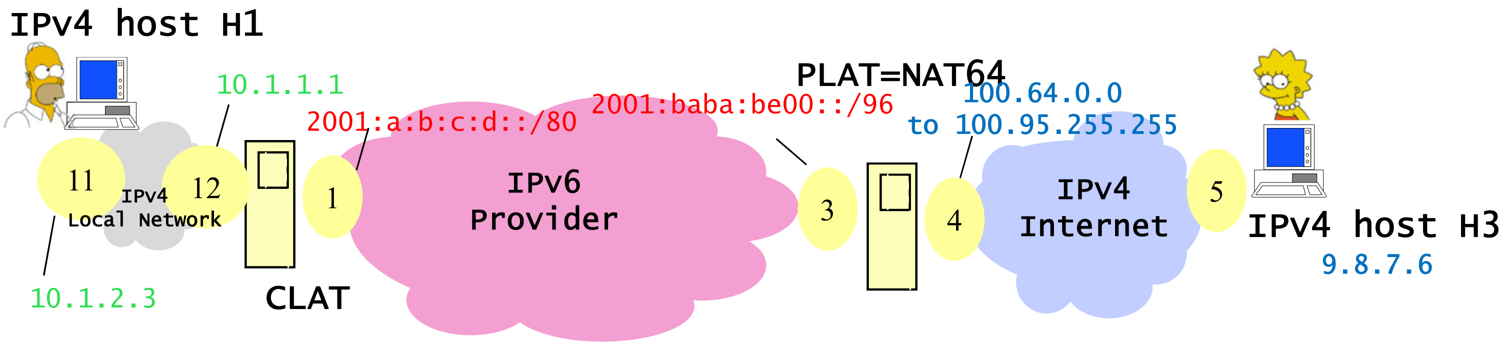
IPv6 provider reserves: one block of IPv6 addresses for the IPv4 internet (2001:baba:be00::/96), one block of IPv6 addresses per IPv4 customer (2001:a:b:c:d::/80 for Homer) and one block of IPv4 addresses for the set of all remote IPv4 customers such as Homer (e.g. 100.64/11).

CLAT (customer-side translator) performs stateless address translation IPv4 <-> IPv6 for local and remote v4 addresses. It is a NAT, but does not modify port numbers.

10.1.2.3 is mapped to 2001:a:b:c:d::a01:203

9.8.7.6 is mapped to 2001:baba:be00::908:706

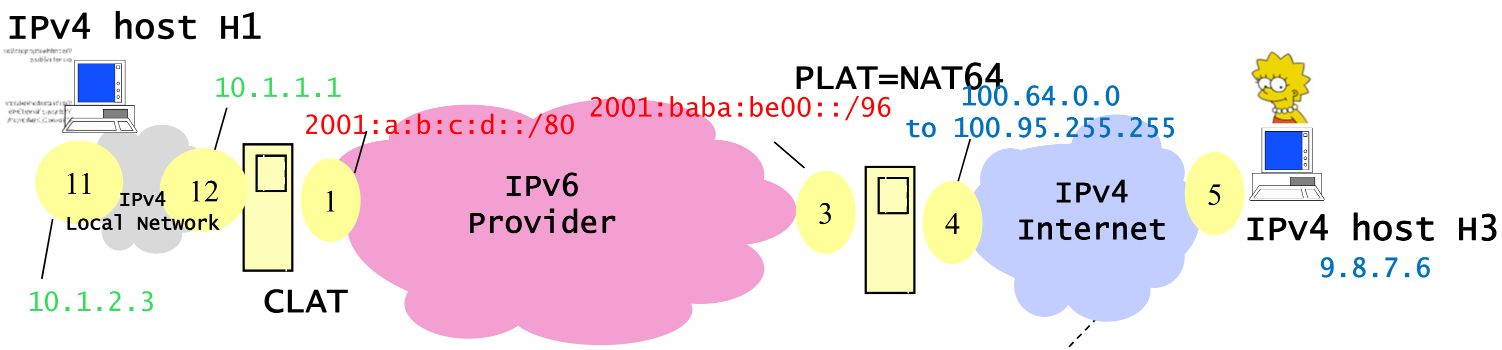
464XLAT: Provider-Side Translation



PLAT (provider-side translator), also called NAT64, performs stateful address translation IPv6 -> IPv4. Like a regular NAT, needs to modify port numbers. E.g. 2001:a:b:c:d::a01:203 port 3456 is mapped to 100.83.21.65 port 4567.

It also performs stateless address translation IPv4-> IPv6. E.g. 9.8.7.6 is mapped to 2001:baba:be00::908:706 (IPv4-embedded-IPv6 address)

PLAT (NAT64) is Stateful

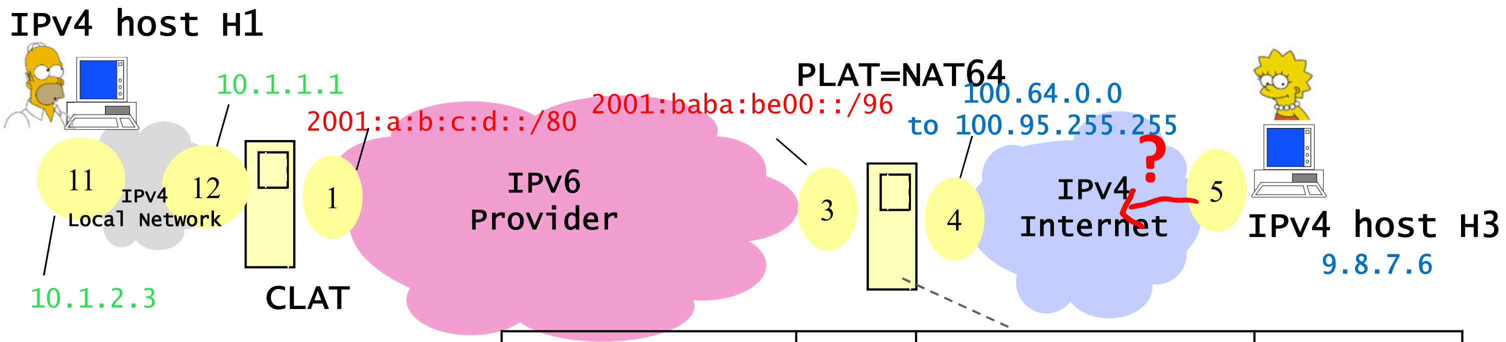


PLAT needs to remember the (v4 address, port) mapping + the IPv6 source address of Homer. In the NAT64 table we see:

IPv6 CLAT address	port	IPv4 translated address	translated port
2001:a:b:c:d::a01:203	3456	100.83.21.65	4567

PLAT does this for all customers and for every flow served by this provider. The NAT table may be very large. This is an example of “Carrier Grade NAT”. The addresse block 100.64/10 is reserved for carrier-grade NATs.

Homer sends one packet to Lisa and Lisa responds. We observe the response at 5. Say what is true.



IPv6 CLAT address	port	IPv4 translated address	translated port
2001:a:b:c:d::a01:203	3456	100.83.21.65	4567

A. The IPv4 destination address in the packet is 10.1.2.3

B. The IPv4 destination address in the packet is 100.83.21.65

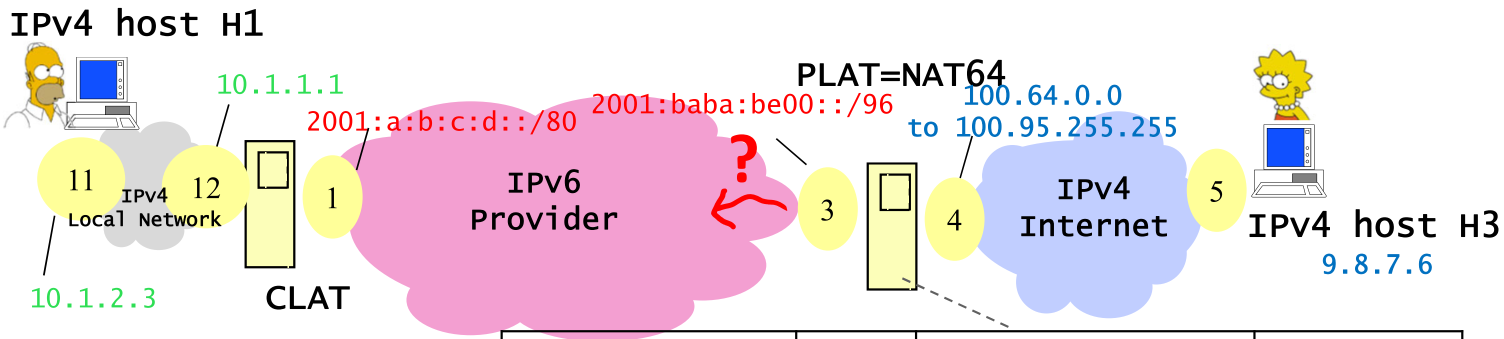
C. The IPv6 destination address in the packet is 2001:a:b:c:d::a01:203

D. A and C

E. B and C

F. I don’t know

Homer sends one packet to Lisa and Lisa responds. We observe the response at 3. Say what is true.



IPv6 CLAT address	port	IPv4 translated address	translated port
2001:a:b:c:d::a01:203	3456	100.83.21.65	4567

A. The IPv4 destination address in the packet is 10.1.2.3

B. The IPv6 destination address in the packet is 2001:a:b:c:d::a01:203

C. The IPv6 source address in the packet is 2001:baba:be00::908:706

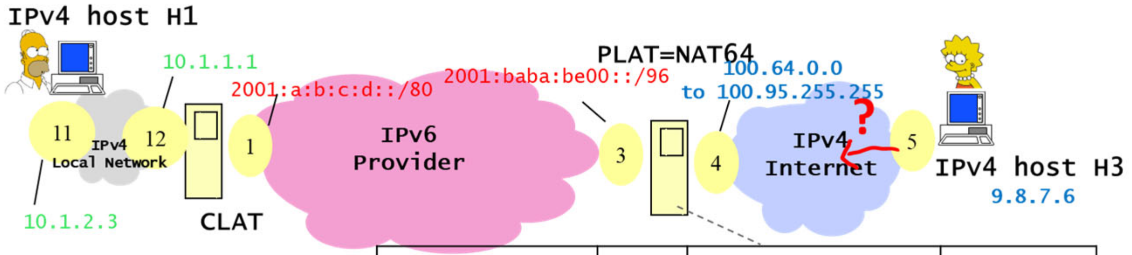
D. A and C

E. B and C

F. A, B and C

G. I don’t know

Solution



IPv6 CLAT address	port	IPv4 translated address	translated port
2001:a:b:c:d::a01:203	3456	100.83.21.65	4567

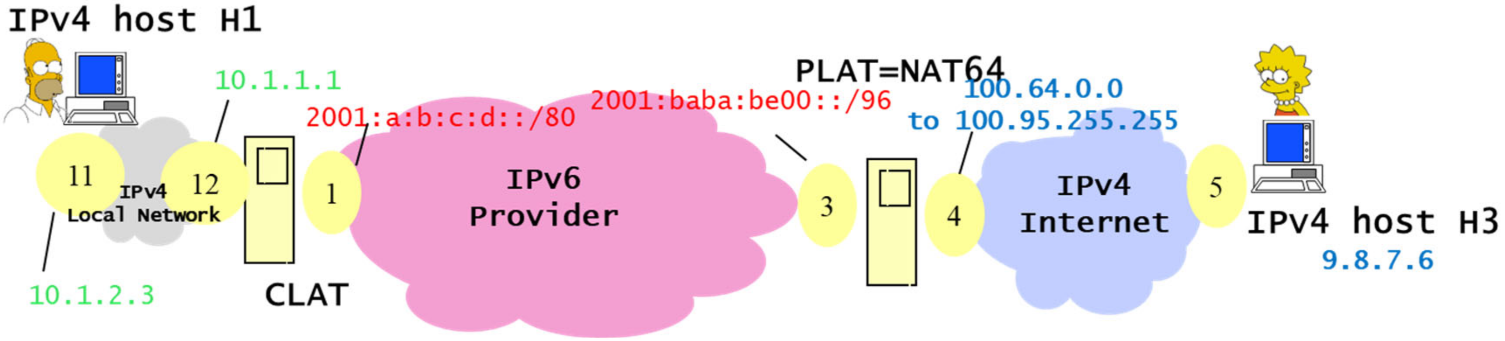
Answer B

There is no encapsulated IPv6 packet at (5), nor anywhere else in this scenario.

Answer E (B and C are true).

There is no encapsulated IPv4 packet at (3), nor anywhere else in this scenario.

All links are Ethernet with MTU = 1500 Bytes. Assume all hosts perform Path-MTU and discover the best possible Path-MTU value. What is the value of Path-MTU between Lisa and Homer ?



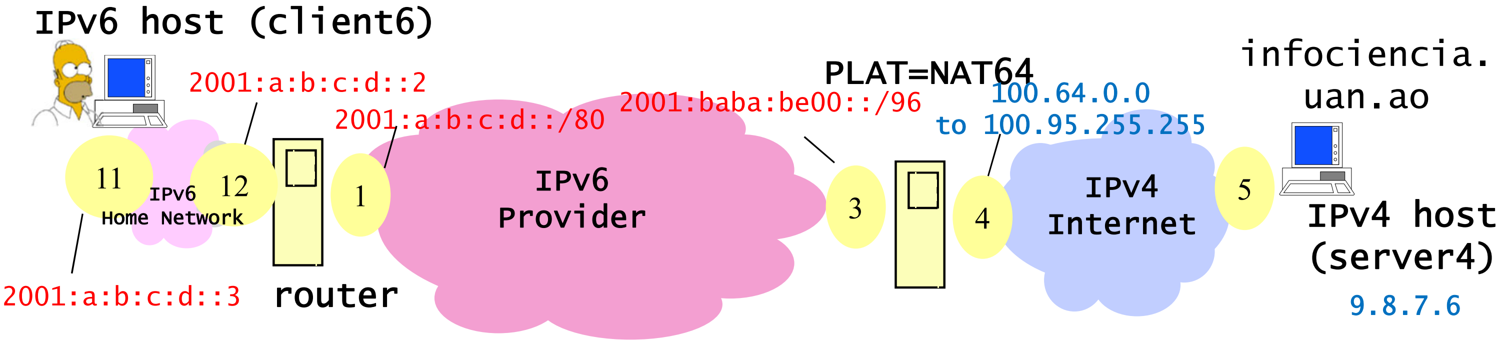
A. 1500 Bytes
B. 1480 Bytes
C. 1460 Bytes
D. None of these
E. I don’t know

Solution

The IPv4 packets at points (4) and (12) are translated into IPv6 packets at points (1) and (3). During the translation, the header is increased by 20 bytes (from 20B IPv4 header to 40B IPv6 header) – assuming there are no header options. So the packet size increases by 20 B at these points.

Therefore the Path MTU is 1480 B in this case. If the IPv6 path uses header options (e.g. for segment routing), this is further reduced.

14. 6 to 4 Interworking with NAT64 and DNS64



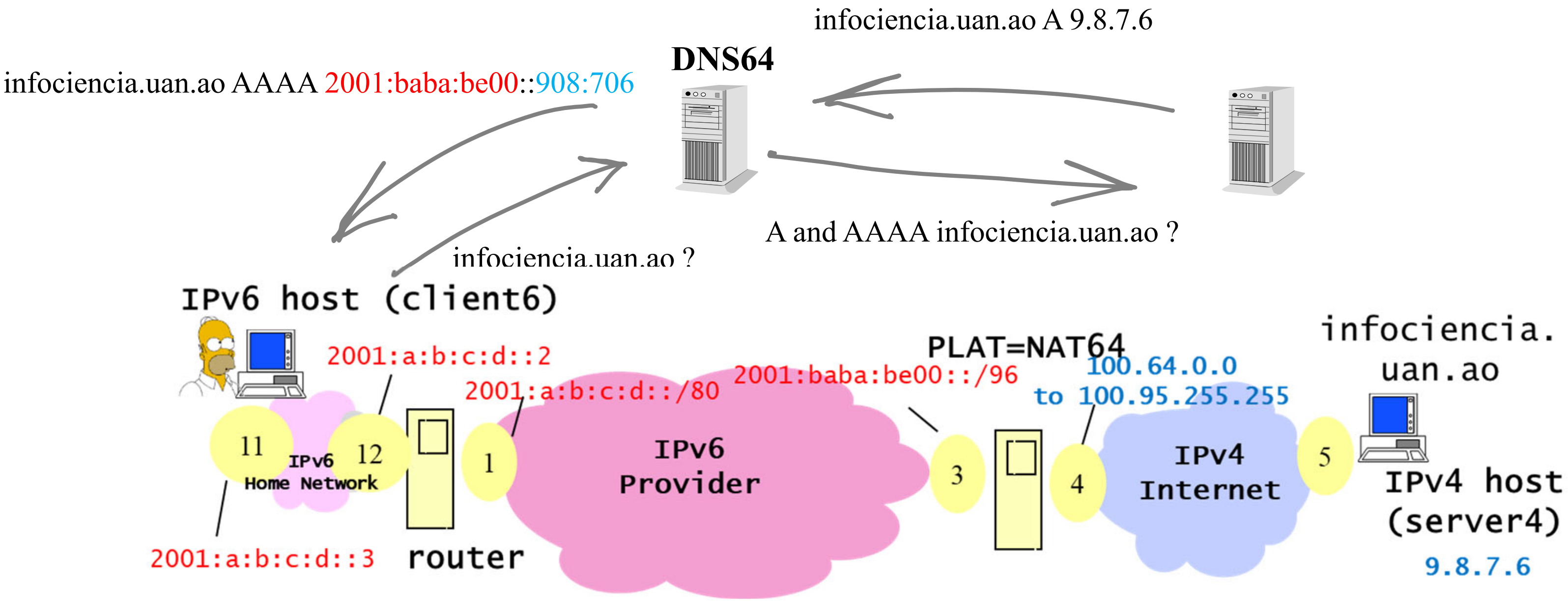
Problem: h6 to s4 Interworking: Homer has IPv6-only service and IPv6 only host and wants to communicate with v4 host

One solution: re-use elements of 464XLAT

v6ISP reserves one block of IPv6 addresses for the IPv4 internet (2001:baba:be00::/96), one block of IPv4 addresses for all remote IPv6 customers (e.g. 100.64/11) and a NAT64; NAT64 performs stateful header translation [NAT64 is the same as PLAT].

To client6, server4 appears under the IPv4-embedded-IPv6 address 2001:baba:be00::908:706

How does client6 know the IPv4-embedded-IPv6 address of server4 ?



DNS64 is used in combination with stateful NAT64.

DNS64 responds with translated IPv6 address if no AAAA record is found.

This is deployed by v6ISP and is transparent to client6.

Mechanisms for Transition to IPv6

Like-to-like access

4to4 over 6: 464XLAT,

MAP-E, 4rd: similar to 464XLAT but stateful address translation is performed on customer side (scalable)

MAP-T same as MAP-E with encapsulation instead of NATs

6to6 over 4: Tunnel brokers, 6rd, Teredo

Interworking

With NATs : NAT64, DNS64

With Application Layer Gateways

Example: mobile operator launches IPv6-only service

Android devices support 464XLAT (CLAT in device)
IOS devices do not but require that all apps work with IPv6
⇒ mobile operator deploys NAT64 (=PLAT, for Android and for IOS) and DNS64 (for IOS)

15. MPLS

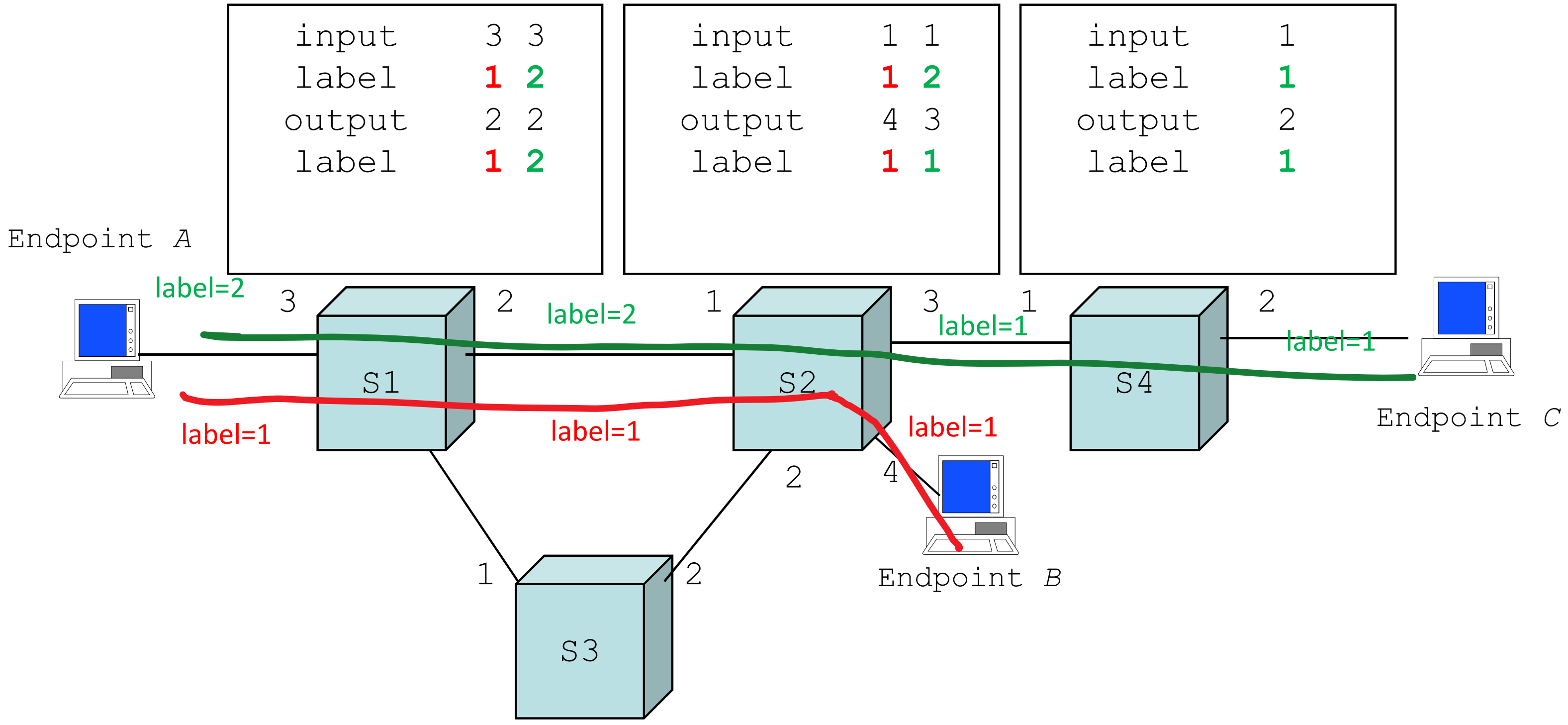
Multi-Protocol Label Switching (MPLS) complements IP

- In IP backbones of ISPs or at interconnection points;
- To interconnect private IPv4 address spaces;
- In Industrial automation networks, smart grids .

It uses the principle of “connection oriented” network layer as opposed to connectionless as IP does.

It can be used as replacement to tunnels.

Principle of Connection Oriented Networks
= Label Switching



input	3	3
label	1	2
output	2	2
label	1	2

input	1	1
label	1	2
output	4	3
label	1	1

input	1
label	1
output	2
label	1

Before transmitting data between two end-points one must establish a connection (≈ like a telephone call) using a *signaling protocol.*

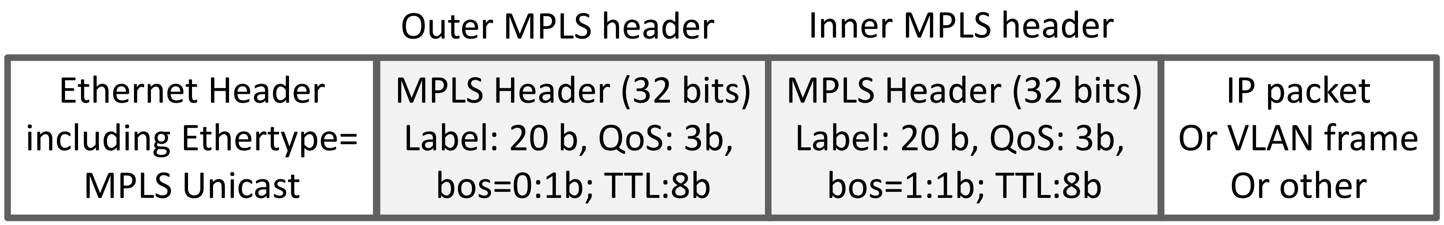
Every packet header contains a *label* which identifies the connection on this link.

The label is local to the link or to the node and can be modified by an intermediate system.

Intermediate systems, called switches (with ATM and Frame Relay) or routers (with MPLS), perform exact lookup in the label switching table. The values in the label switching table are managed by the control plane (e.g.signaling protocol LDP) or by a centralized SDN application.

The connection is called Label Switched Path (LSP) with MPLS.

MPLS Header



	Outer MPLS header	Inner MPLS header	
Ethernet Header including Ethertype= MPLS Unicast	MPLS Header (32 bits) Label: 20 b, QoS: 3b, bos=0:1b; TTL:8b	MPLS Header (32 bits) Label: 20 b, QoS: 3b, bos=1:1b; TTL:8b	IP packet Or VLAN frame Or other

MPLS header comes after the MAC layer header. There is one Ethertype for MPLS unicast (and one for MPLS multicast, not discussed here).

The MPLS header contains mainly the label, plus 3 bits for quality of service and a TTL field (similar to IPv4’s TTL).

There can be several MPLS headers on top of one another. The innermost header is recognized by the flag “Bottom of Stack”(bos).

Example: MPLS for VPNs

MPLS Routers run both MPLS and IP in parallel.

MPLS Routers A and B are Label Edge Routers (LERs). They interface between three private IP networks on one hand, and the MPLS network on the other hand. Other MPLS routers M,N,P and Q are called Label Switching Routers (LSRs). Note that all routers also run IP and a routing protocol (e.g. OSPF) with non private IP addresses (not shown).

The figures show 3 label-switched paths (LSPs) from A to B. These were setup either by an SDN application or automatically with BGP as discussed in the next section.

At router A, a packet from the blue VPN that needs to be sent to B receives an inner MPLS label (here, 78) that indicates that it belongs to blue VPN and an outer MPLS label (here, 49) that has the effect of sending the packet over the blue LSP.

The packet is then forwarded by P and N, whose only function is: look at the outer MPLS label, use the label switching table to forward the packet, replace the label by the outgoing label value and decrement the MPLS TTL.

When the packet reaches B, the outer MPLS label is removed ("popped") as instructed by B's table and the inner MPLS label (78) is examined. B's table says that this label matches the blue VPN and the IP packet is analyzed at B using the IP destination address and the routing table of the blue VPN. In B's (and A's) table, the VPN is identified by the "Virtual Routing Function" (VRF) field. Here VRF ==1 [resp. 2,3] means the green [resp. blue, red] VPN.

Example: MPLS for VPNs (continued)

In principle, the IP TTL is copied from the MPLS TTL and decremented by 1, so the MPLS hops are counted in the total hop count. Observe that the same private IPv4 addresses can be used in the blue and blue VPNs, without collision.

The first figure also shows some selected packet headers at observation points .

LSPs are unidirectional. There are other LSPs from B to A, not shown in the figures; for example, the table at B in the second figure indicates that there is an LSP from B to A for the green VPN that uses label 99 in node B.

Labels are local to a node , i.e. the same label value can be used in different non overlapping nodes (as shown here for labels 49 and 50. In some (older) MPLS implementations labels are local to a link.

The figures show IPv4 packets in MPLS, but MPLS can be used to carry IPv6 packets, or even Ethernet packets (for interconnecting virtual LANs at the MAC layer). There is no protocol field in the MPLS header. It is the innermost label and the MPLS table at a LER that identify the nature of the protocol.

An MPLS router runs both IP and MPLS. Its forwarding table contains both the IP forwarding table (plus multiple tables, one per VRF) and the label switching table.

Label Switching
Table at M (selected rows)

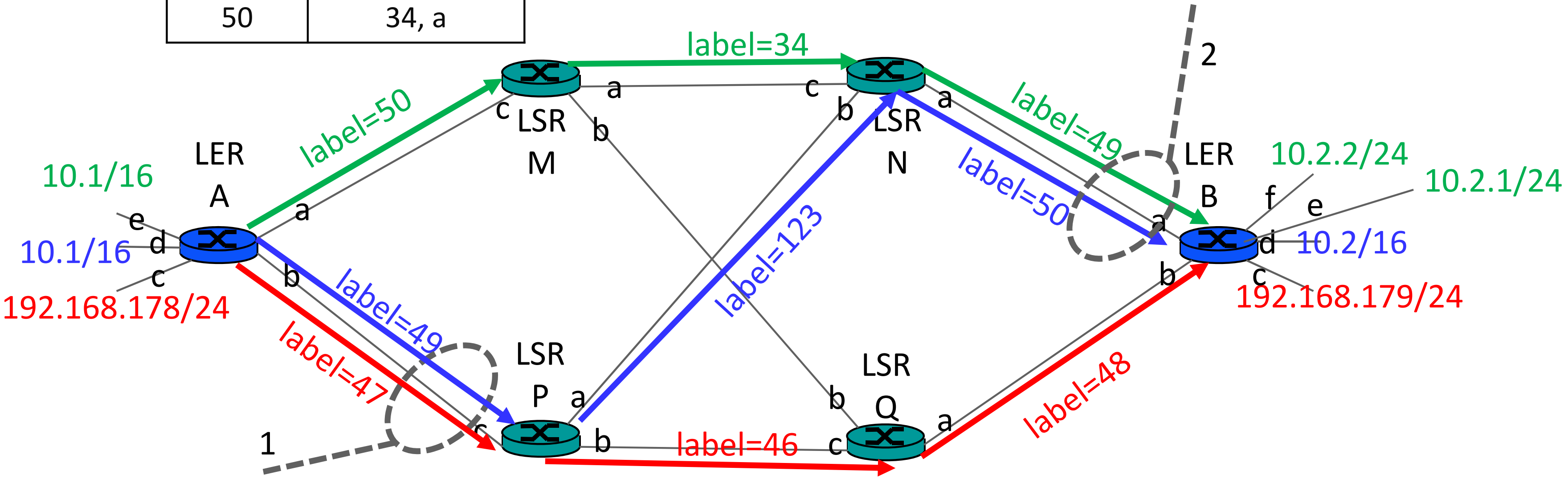
inc label	out label, i/f
50	34, a

Label Switching
Table at N (selected rows)

inc label	out label, i/f
34	49, a
123	50, a

Eth hdr	Label 49	Label 24	IP hdr

Eth hdr	Label 50	Label 78	IP hdr



Eth hdr	Label 49	Label 78	IP hdr

Eth hdr	Label 47	Label 89	IP hdr

inc label	out label, i/f
47	46, b
49	123, a

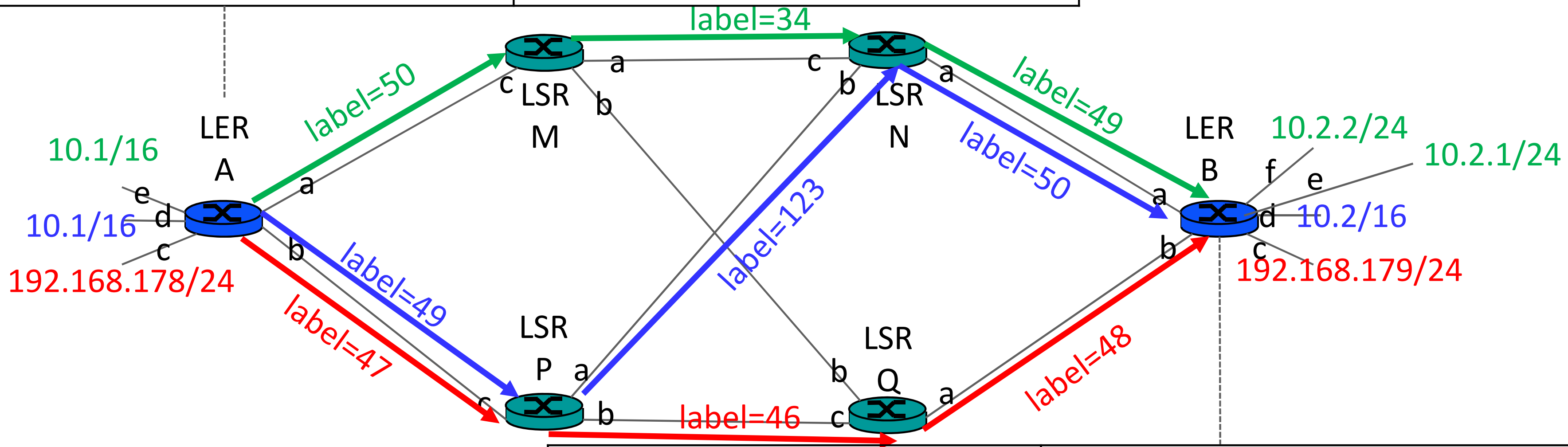
Label Switching
Table at P (selected rows)

inc label	out label, i/f
46	48, a

Label Switching
Table at Q (selected rows)

filter	action
VRF == 1, dest addr = 10.2/16	push label = 24; push label = 50; out a
VRF == 2, dest addr = 10.2/16	push label = 78; push label = 49; out b
VRF == 3, dest addr = 192.168.179/24	push label = 89; push label = 47; out b

Combined Virtual Routing and Label Switching Table at A (selected rows)



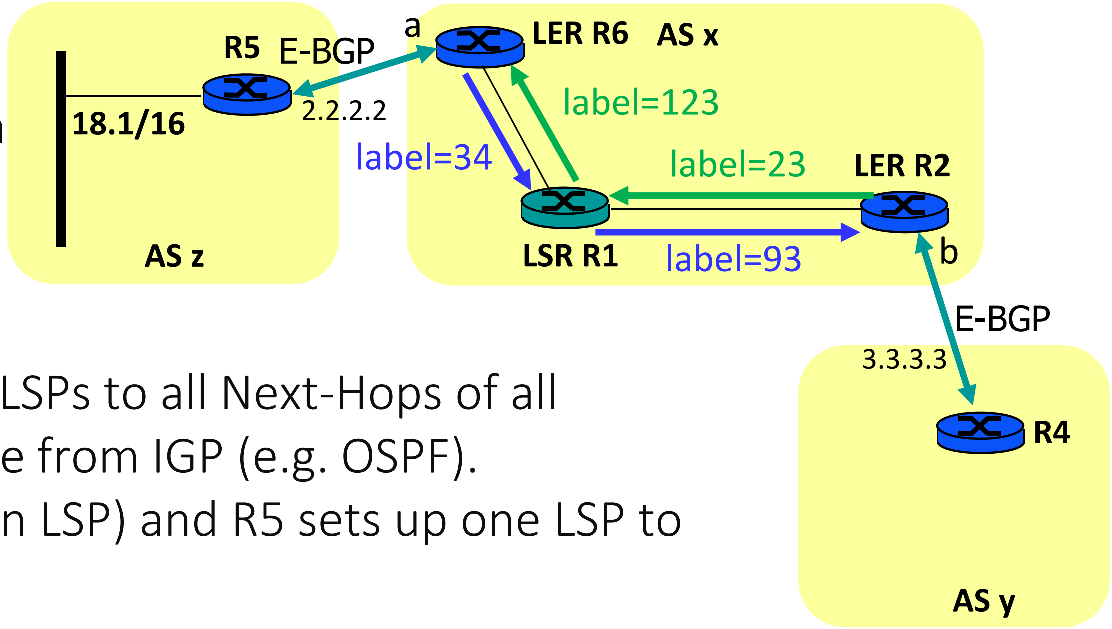
Combined Virtual Routing and Label Switching Table at B (selected rows)

filter	action
label == 49	pop;
label == 50	pop;
label == 48	pop;
label == 24	pop; use virtual routing table of VRF 1
label == 78	pop; use virtual routing table of VRF 2
label == 89	pop; use routing table of VRF 3
VRF == 1, dest addr = 10.2.1/24	out e, next-hop = 10.2.1.1
VRF == 1, dest addr = 10.2.2/24	out f, next-hop = 10.2.2.1
VRF ==1, dest addr = 10.1/16	push label = 24; push label = 99; out a

Virtual Routing Table of red VPN

Example: BGP and MPLS

MPLS can be used to avoid redistribution of BGP into IGP.
LERs analyze IP headers and use LSPs to reach egress routers.



Example: all E-BGP routers in AS-x setup LSPs to all Next-Hops of all E-BGP routers in AS-x. Using info available from IGP (e.g. OSPF).
Here: R2 sets up one LSP to 2.2.2.2 (green LSP) and R5 sets up one LSP to 3.3.3.3 (blue LSP).

R1 is a LSR and runs IGP; has no IP route to external prefixes.
R2 and R6 are LERs and run IGP +BGP; have routes to external prefixes.

Forwarding and label tables

At R6

dest addr / label	action
2.2.2.2 3.3.3.3 (rest of routing table)	on-link push 34, out se …

At R1

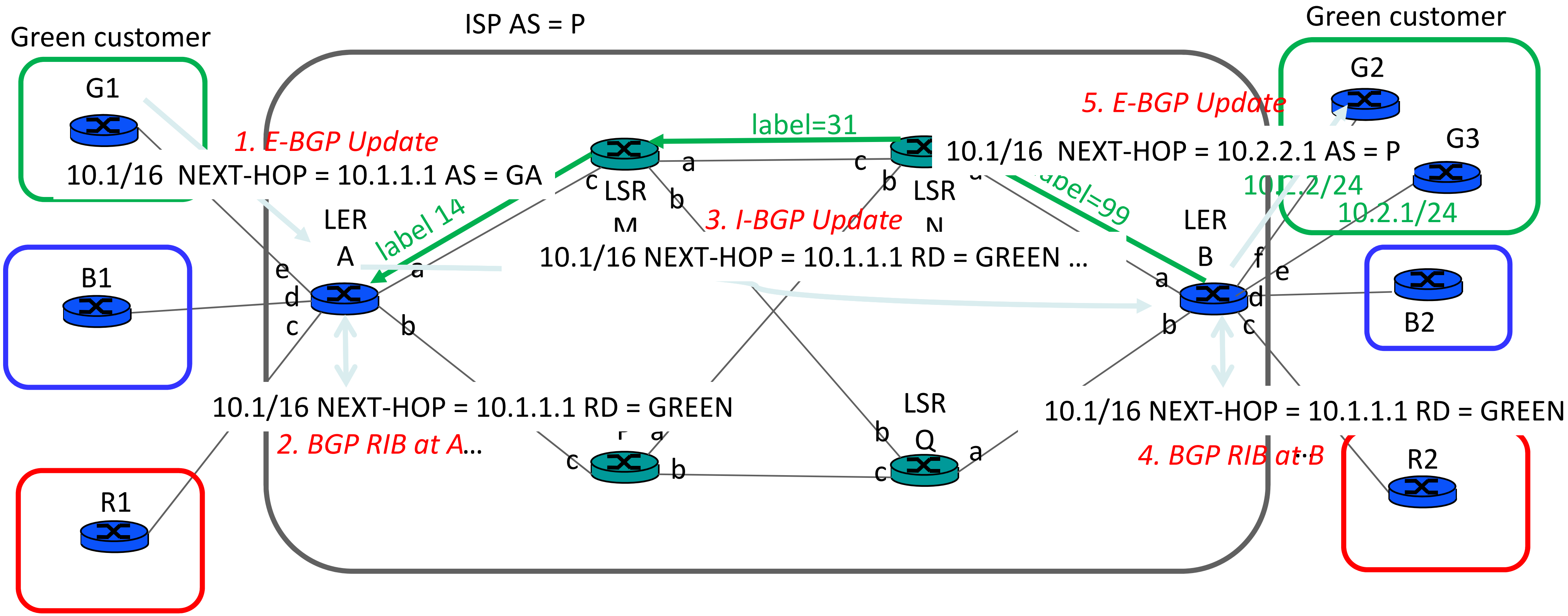
dest addr / label	action

label == 34
label == 23
(rest of routing table)
	label=93, out e
	label=123, out nw
	…

At R2

dest addr / label	action
18.1/16 2.2.2.2 (rest of routing table)	Next-Hop=2.2.2.2 push 23, out w …

BGP for MPLS VPNs



1. E-BGP Update

10.1/16 NEXT-HOP = 10.1.1.1 AS = GA

2. BGP RIB at A…

10.1/16 NEXT-HOP = 10.1.1.1 RD = GREEN

3. I-BGP Update

10.1/16 NEXT-HOP = 10.1.1.1 RD = GREEN …

4. BGP RIB at B…

10.1/16 NEXT-HOP = 10.1.1.1 RD = GREEN

5. E-BGP Update

10.1/16 NEXT-HOP = 10.2.2.1 AS = P

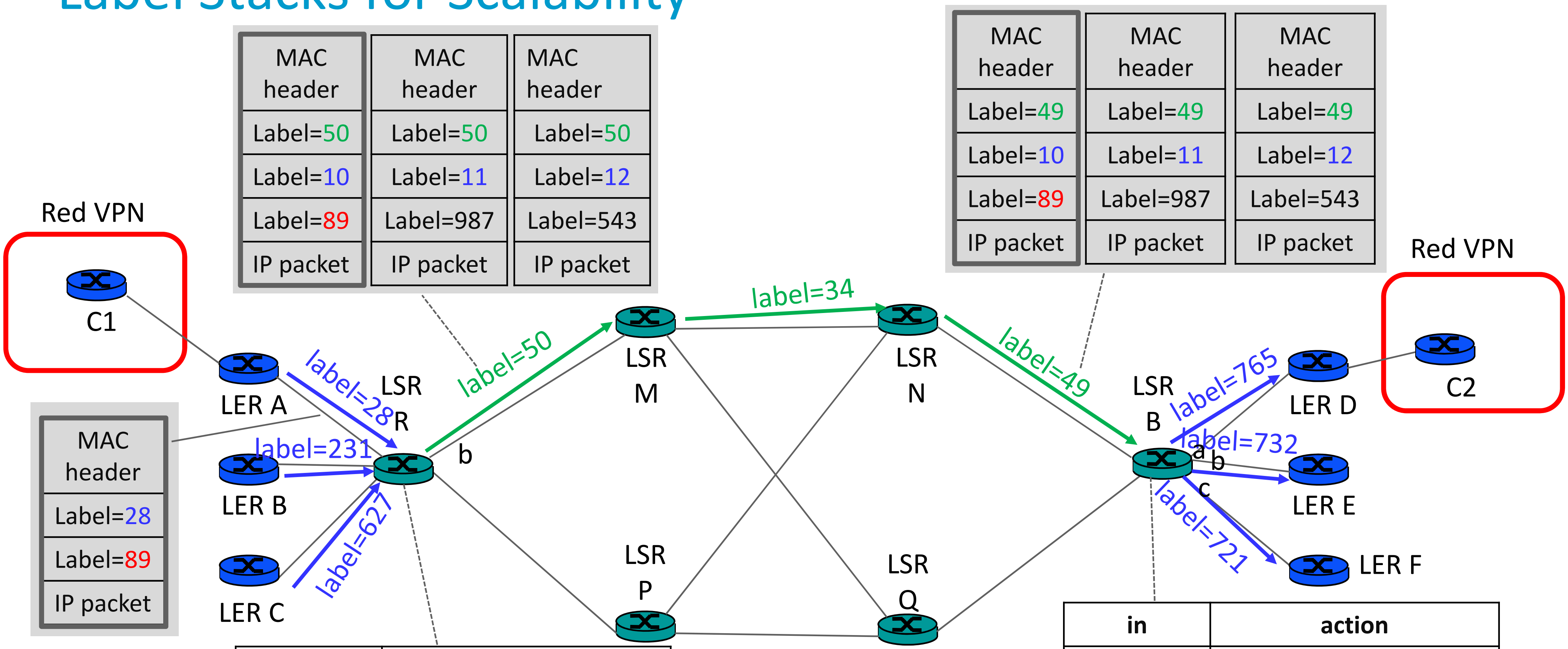
The previous figure shows how BGP can be used to automate the creation and management of MPLS-VPNs.

Customer networks use BGP to advertize their private prefixes to the ISP.

1. Router G1 sends an E-BGP Update to the ISP, advertizing the prefixes that are local to this site.
2. ISP knows that G1 is in the green customer domain. It associates the received BGP route to the “green” VPN. With BGP, this is done by using a “Route Distinguisher” (RD) attribute. The BGP record stored in the RIB-In of A contains this RD attribute. A accepts this update and selects it as best route to 10.1/16, RD==GREEN. When RD is used, the BGP decision process considers that routes with different RD values are distinct. In its forwarding table, A also injects the prefix 10.1/16, with next-hop 10.1.1.1 and VRF=1 (i.e. in the virtual routing table of the green VPN). The mapping RD=GREEN→ VRF=1 must be done in the BGP configuration of A (and of all BGP routers in the ISP domain).
3. LERs in ISP all run BGP. A sends to B an I-BGP update with the route “10.1/16 RD=GREEN”. LSRs such as M, N, P and Q do not run BGP.
4. B accepts the route “10.1/16 next-hop 10.1.1.1 RD=GREEN”. The MPLS control plane establishes an LSP from B to A; this LSP is used for the green VPN and is associated with the inner MPLS label 24. B writes “dest 10.1/16, VRF==1 → push label = 24; push label = 99; out a” in its forwarding table.
5. B sends an E-BGP update to G2 with the route “10.1/16 AS-PATH=P NEXT-HOP=10.2.2.1”. The next-hop is B’s address seen from G2. There is no RD field here. The AS-PATH contains only P as the AS number used by G1 is removed. G2 accepts this route and inject into its routing table the entry “dest 10.1/16 → 10.2.2.1”. A similar process occurs at G3.

The effect of this procedure is that (1) the ISP is able to carry all IP packets of the green customer and (2) the different sites of the green customers learn intra-VPN routes that span the ISP network.

Label Stacks for Scalability



MAC header	MAC header	MAC header
Label=50	Label=50	Label=50
Label=10	Label=11	Label=12
Label=89	Label=987	Label=543
IP packet	IP packet	IP packet

MAC header
Label=28
Label=89
IP packet

MAC header	MAC header	MAC header
Label=49	Label=49	Label=49
Label=10	Label=11	Label=12
Label=89	Label=987	Label=543
IP packet	IP packet	IP packet

in	action
label==28	label=10, push 50, out b
label==231	label=11, push 50, out b
label==627	label=12, push 50, out b

Forwarding Table at
← R and B →
(selected rows)

in	action
label==49	pop
label==10	label=765, out a
label==11	label=732, out b
label==12	label=721, out c

Label Stacks for Scalability (continued)

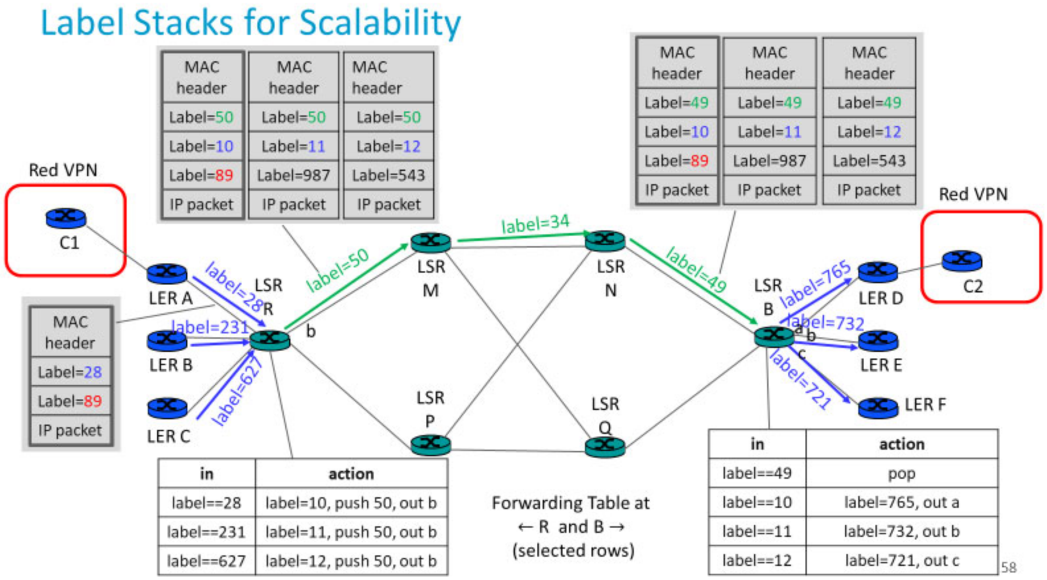
Label stacks are used to identify the protocol type and the VPN to be used at the edge (i.e. by an LER). They can also used to multiplex LSPs into LSPs (here: green). This reduces the number of LSPs and makes MPLS more scalable.

The figure shows a green LSP from R to B. Inside the green LSP are carried three blue LSPs that are merged into one at R.

1. LER A [resp. B, C] establishes one LSP to LER B (blue LSP), and the corresponding label is 28 [resp. 231, 627].
2. At LSR R, the label 28 [resp. 231, 627] is switched to label 10 [resp. 11, 12]; furthermore, an additional MPLS label (50, green) is added (pushed).
3. LSRs M and N see only one LSP and switch only the outer label.
4. LSR B’s table indicates that it is the endpoint for the LSP with label 49 and the outer label is removed. The next label (28, 231, 627) is switched and used to send the packet to D, E or F.

How does router D know that packets to be forwarded to C2 is an IPv4 packet ?

Ethernet Header including Ethertype= MPLS Unicast	MPLS Header (32 bits) Label: 20 b, QoS: 3b, bos=0:1b; TTL:8b	MPLS Header (32 bits) Label: 20 b, QoS: 3b, bos=1:1b; TTL:8b	IP packet Or VLAN frame Or other



Label Stacks for Scalability

MAC header	MAC header	MAC header
Label=50	Label=50	Label=50
Label=10	Label=11	Label=12
Label=89	Label=987	Label=543
IP packet	IP packet	IP packet

MAC header	MAC header	MAC header
Label=49	Label=49	Label=49
Label=10	Label=11	Label=12
Label=89	Label=987	Label=543
IP packet	IP packet	IP packet

MAC header
Label=28
Label=89
IP packet

in	action
label==28	label=10, push 50, out b
label==231	label=11, push 50, out b
label==627	label=12, push 50, out b

Forwarding Table at
← R and B →
(selected rows)

in	action
label==49	pop
label==10	label=765, out a
label==11	label=732, out b
label==12	label=721, out c

A. By the Ethertype contained in the received packets

B. By the version number contained in the IP packet received in an MPLS packet

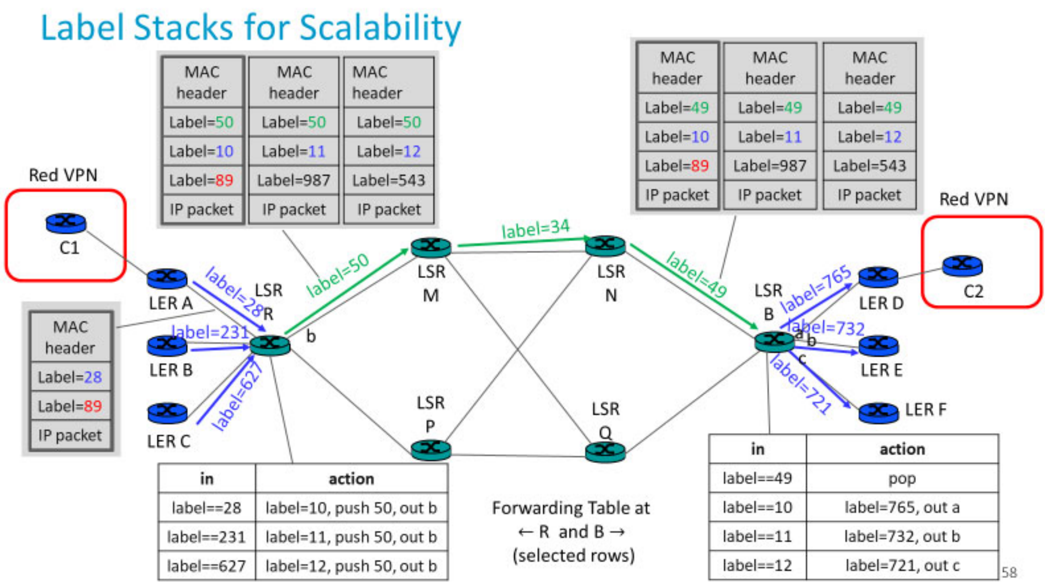
C. None of the above

D. I don’t know

Solution

Ethernet Header including Ethertype= MPLS Unicast	MPLS Header (32 bits) Label: 20 b, QoS: 3b, bos=0:1b; TTL:8b	MPLS Header (32 bits) Label: 20 b, QoS: 3b, bos=1:1b; TTL:8b	IP packet Or VLAN frame Or other

MAC header
Label=765
Label=89
IP packet



in	action
label==28	label=10, push 50, out b
label==231	label=11, push 50, out b
label==627	label=12, push 50, out b

Forwarding Table at
← R and B →
(selected rows)

in	action
label==49	pop
label==10	label=765, out a
label==11	label=732, out b
label==12	label=721, out c

Answer C

The Ethertype of the received packet indicates MPLS, but it does not allow to know what type of packet is contained in the MPLS payload.

The version number in the IPv4 packet is 4, but in order to understand this the router must know that it is an IPv4 packet, and there is no way to determine this by inspecting the IPv4 packet.

The inner label (89) indicates that it is an IPv4 packet (it also indicates the VRF value, here, red VPN). The association is performed during configuration of the LSPs.

Conclusion

Proxy ARP / ND Proxy is a trick used to solve the problems caused by a subnet present at different locations

Fragmentation is due to different MAC layers having different packet sizes. Fragmentation occurs only at IPv6 hosts, IPv4 hosts or IPv4 routers. Re-assembly is never done by routers.
Fragmentation may cause problems and should be avoided if possible.

Tunnels are used e.g. to create virtual private networks

Transition to IPv6 creates many problems that can be solved with various methods involving automatic tunnels, header translation (CLAT, NAT64 = PLAT,) and DNS manipulation (DNS64).

In operator networks, MPLS is an alternative to tunnels; it builds paths between edge routers that by-pass the usual IP forwarding.