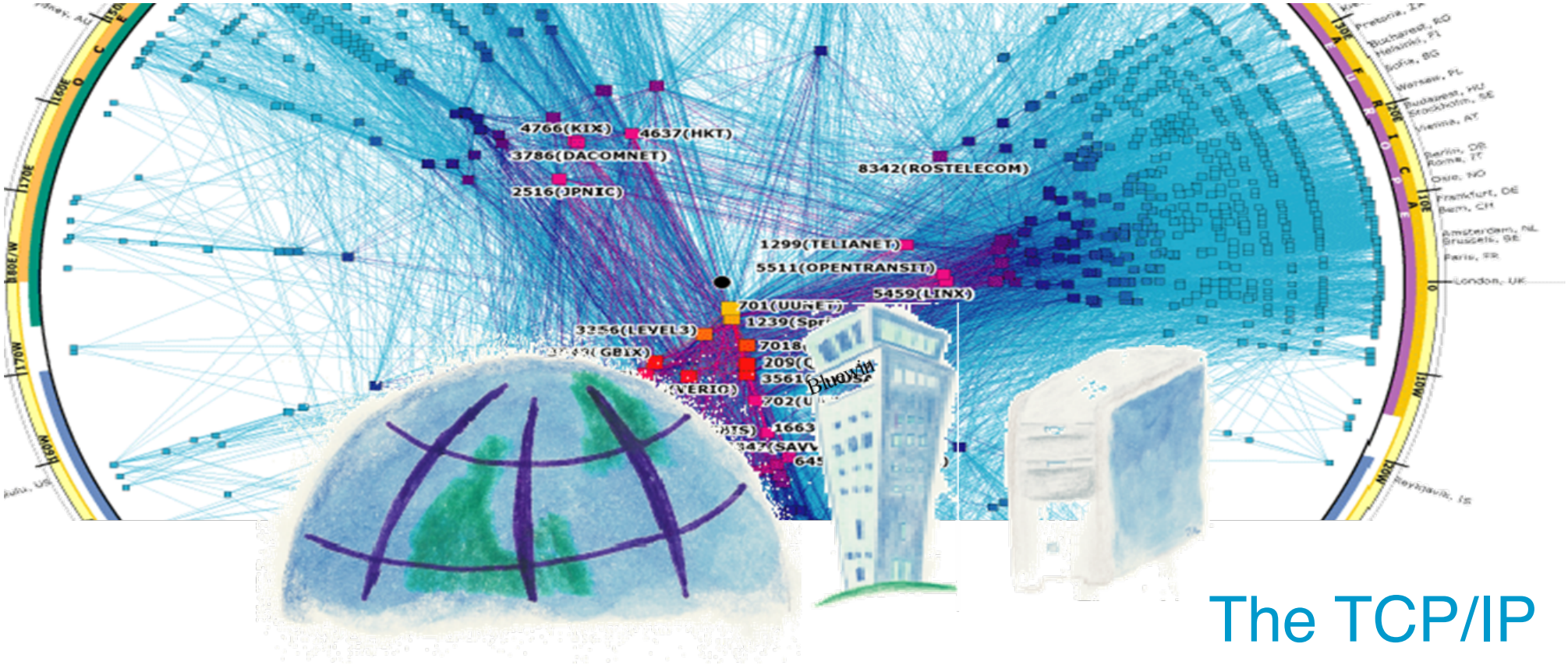


The TCP/IP Network Layer IPv4 and IPv6

2024

EPFL

Recap

communication

Application

end-host connectivity

Transport

across-LANs interconnection

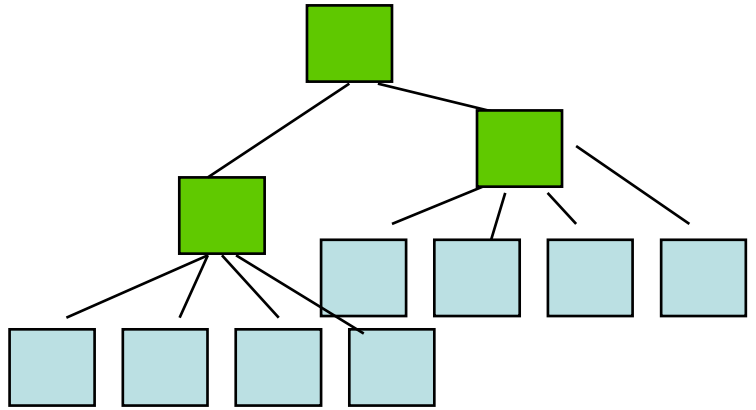
Network

within-LAN interconnection

MAC

point-to-point transmission of bits

Physical



Recap: most important protocol = IP (Internet Protocol)

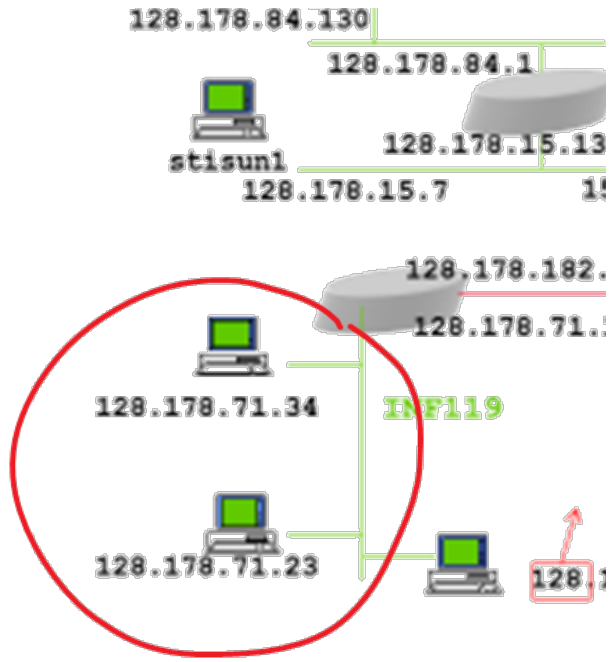
- interconnects multiple *local area networks* (LANs)
- uses packet switching
- delivers packets from a source to a destination via a series of routers
- forwards packets from router to router based on *IP addresses*
- offers *no reliable*-delivery guarantees (*best-effort* approach)
    - packets are briefly stored in routers' buffers
    - packets of the same source-destination flow may follow different routes/paths
    - so, packets may be *dropped*, *delayed* or *reordered*

Recall IP: subnets and subnet masks

- *subnet* <— a LAN, i.e. a set of devices:
  - connected at the Data-link layer
  - sharing the same IP-address *prefix*, e.g.: 128.178.71.X

- The prefix is specified using a *subnet mask*
  (= sequence of bits, where 1s indicate fixed positions of the prefix)
  - e.g. for an EPFL IPv4 LAN, the subnet mask is
    1111 1111 1111 1111 1111 1111 0000 0000
  - The size (in bits) of the prefix is not always the same, e.g.:
    ETHZ IPv4 LANs = 26 bits
    EPFL IPv6 LANs = 64 bits

- Various notations for the subnet mask:
  - dotted, decimal: e.g., address =128.178.71.34, mask = 255.255.255.0
  - *"/"* (slash): e.g. 128.178.71.34/24
    or 2001:620:618:1a6:0a00:20ff:fe78:30f9/64

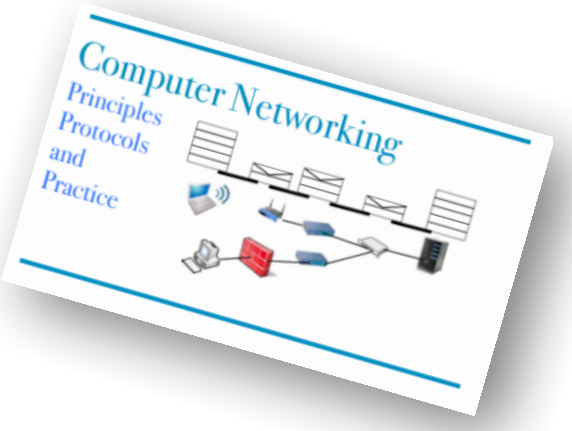


Contents - Internet Protocol (IP)

1. The 2 main rules of IP Unicast
2. IPv4 addresses
3. IPv6 addresses
4. NATs
5. Host configuration
6. Hop Limit and TTL
7. ARP (connection with MAC layer)

Textbook

Chapter 5: The Network Layer



IP Rule #1 = forward packets according to destination IP prefixes

Recall: goal of IP = interconnect all systems in the world using *IP addresses*

➡ every network interface *must* have an IP address

Rule #1 in detail:

1. assign addresses based on a structure:
   - every network interface has an IP address with *prefix* + *suffix*: e.g. 128.178.71.202
   - interfaces inside *same subnet* have *same prefix* —> *same subnet mask*

2. forward packets according to *longest prefix match*:
   - every packet contains the destination IP address in its header
   - every system (i.e. **host** = end-system or **router** = intermediate system)
     - has a **forwarding table** (= routing table) and
     - forwards each packet based on the *closest* table entry that matches the destination IP address

Longest prefix match (= *closest* matching table entry)

R1's forwarding table

to…	outgoing interface
B.*	2
A.*	0

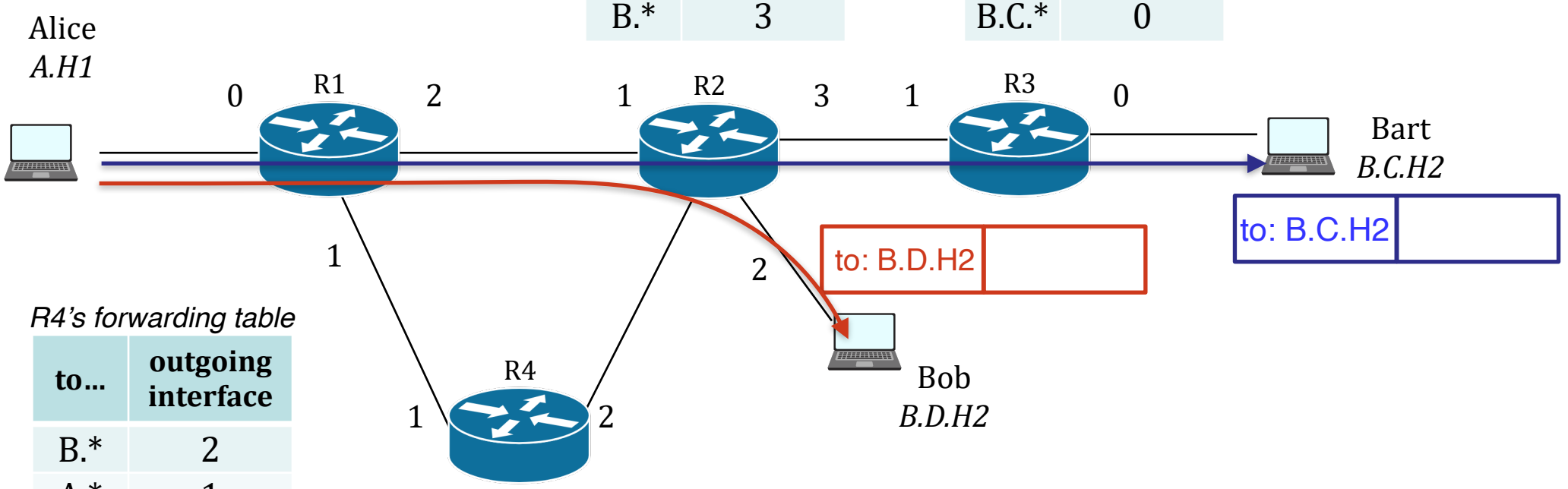
R2's forwarding table

to…	outgoing interface
A.*	1
B.D.*	2
B.*	3

R3's forwarding table

to…	outgoing interface
A.*	1
B.D.*	1
B.C.*	0

Letters in addresses denote prefixes; e.g.:
A = 2001:0620:0008::/48
B.C = 2001:0620:0618:01a5::/64
B.D = 2001:0620:0618:01a6::/64



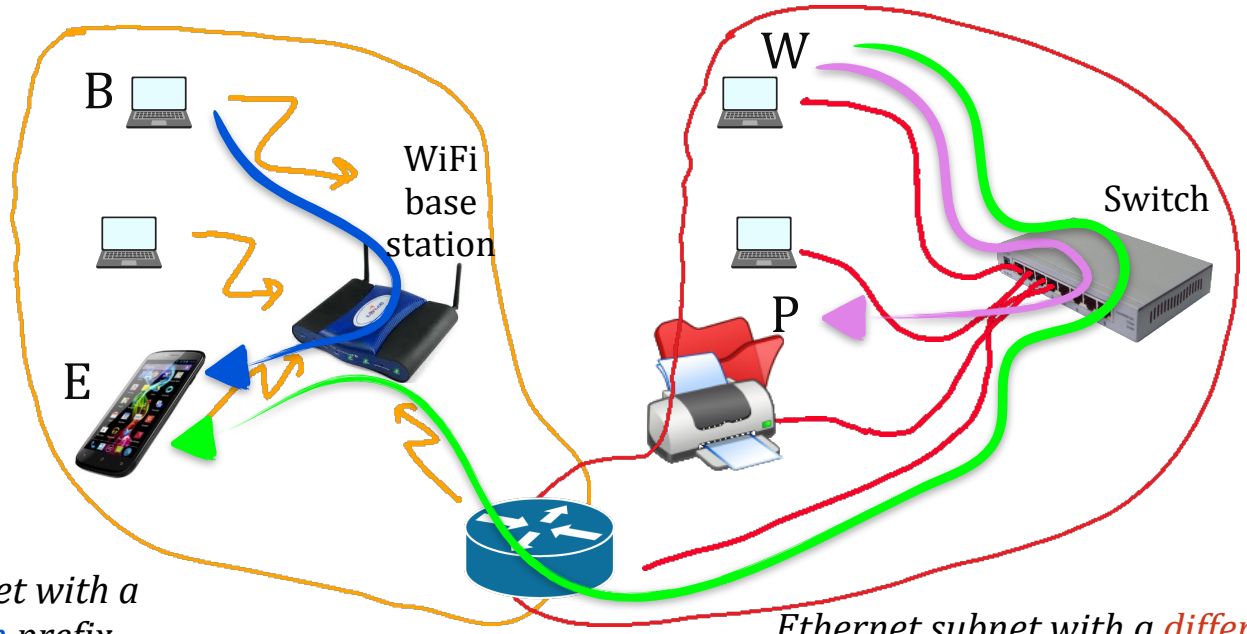
R4's forwarding table

to…	outgoing interface
B.*	2
A.*	1

- Benefit: addresses can be *aggregated*, tables can be *compressed*

IP Rule #2 = routers can only interconnect different LANs/subnets

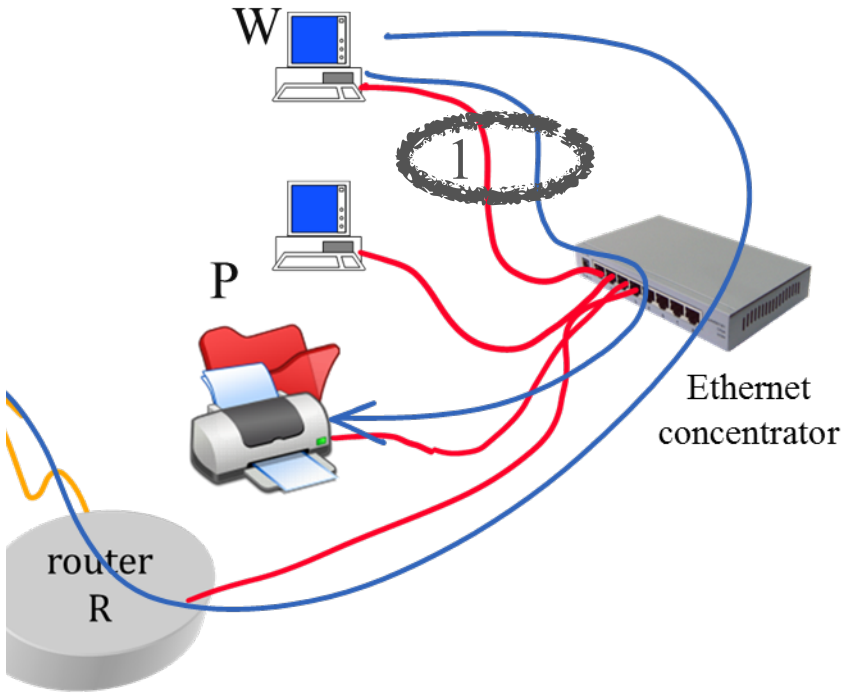
- Traffic B ↔ E and W ↔ P
  does not go through router
- Traffic W ↔ E goes through router



IP rule #2 implies:

- *between* LANs/subnets, we use *routers*
- *inside* each subnet, we do *not* —> we use data-link-layer forwarding devices
  (such as switches, wifi base stations, etc.)

We observe a packet from W to P at 1. Which IP destination address do we see ?



Go to web.speakup.info or download speakup app

Join room 46045

A. The IP address of P

B. The IP address of an Ethernet interface of the Ethernet concentrator

C. There is no destination IP address in the packet since communication is inside the subnet and does not go through a router

Solution

Answer A

The IP address is always present in the IP header if we use TCP/IP, even if communication is inside the same LAN.

2. Global/public unicast IPv4 address

- *Uniquely* identifies a network interface in the internet
- 32 bits, usually written in dotted decimal notation

  binary: 32 bits

  example 1: **b**1000 0000 1011 1111 1001 0111 0000 0001

  example 2: **b**1000 0001 1100 0000 1100 1000 0000 0010

  dotted decimal: 4 integers (one integer = 8 bits/ binary digits)

  example 1: 128.191.151.1

  example 2: 129.192.200.2

  hexadecimal: 8 hex digits (one hex digit = 4 bits/ binary digits)

  example 1: **x**80 bf 97 01

  example 2: **x**81 c0 c8 02

Binary, Decimal and Hexadecimal

Given an integer B (the basis) any integer can be represented as a string in an alphabet of B symbols, starting from 0.

	Basis	*Alphabet*	*Example*
Decimal	X	$\{0,1,2,3,4,5,6,7,8,9\}$	200
Binary	II	$\{0,1\}$	1100 1000
Hexadecimal	XVI	$\{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f\}$	$c8$

Binary <—> hex is easy: one hex digit (= nibble) is 4 binary digits

$c_{hex} = 1100_{bin}$ $8_{hex} = 1000_{bin}$ $c8_{hex} = 1100\ 1000_{bin}$

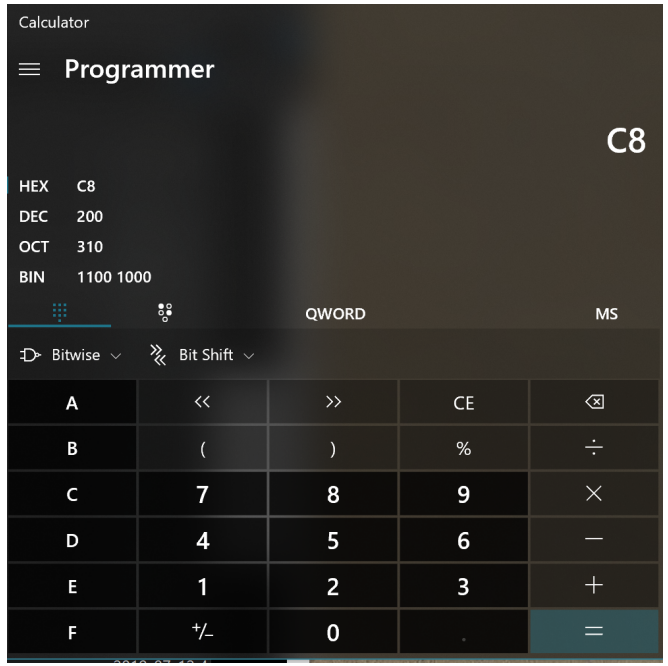
Binary/hex <—> decimal is best done by a calculator

$1100\ 1000_{bin} = 128 + 64 + 8 = 200$

Special Cases to remember

$f_{hex} = 1111_{bin} = 15_{dec}$

$ff_{hex} = 1111\ 1111_{bin} = 255_{dec}$



The mask 255.255.254.0 means that the subnet is made of the first …

Go to web.speakup.info or download speakup app

Join room 46045

A. 16 bits
B. 18 bits
C. 22 bits
D. 23 bits
E. 24 bits

Solution

Answer D

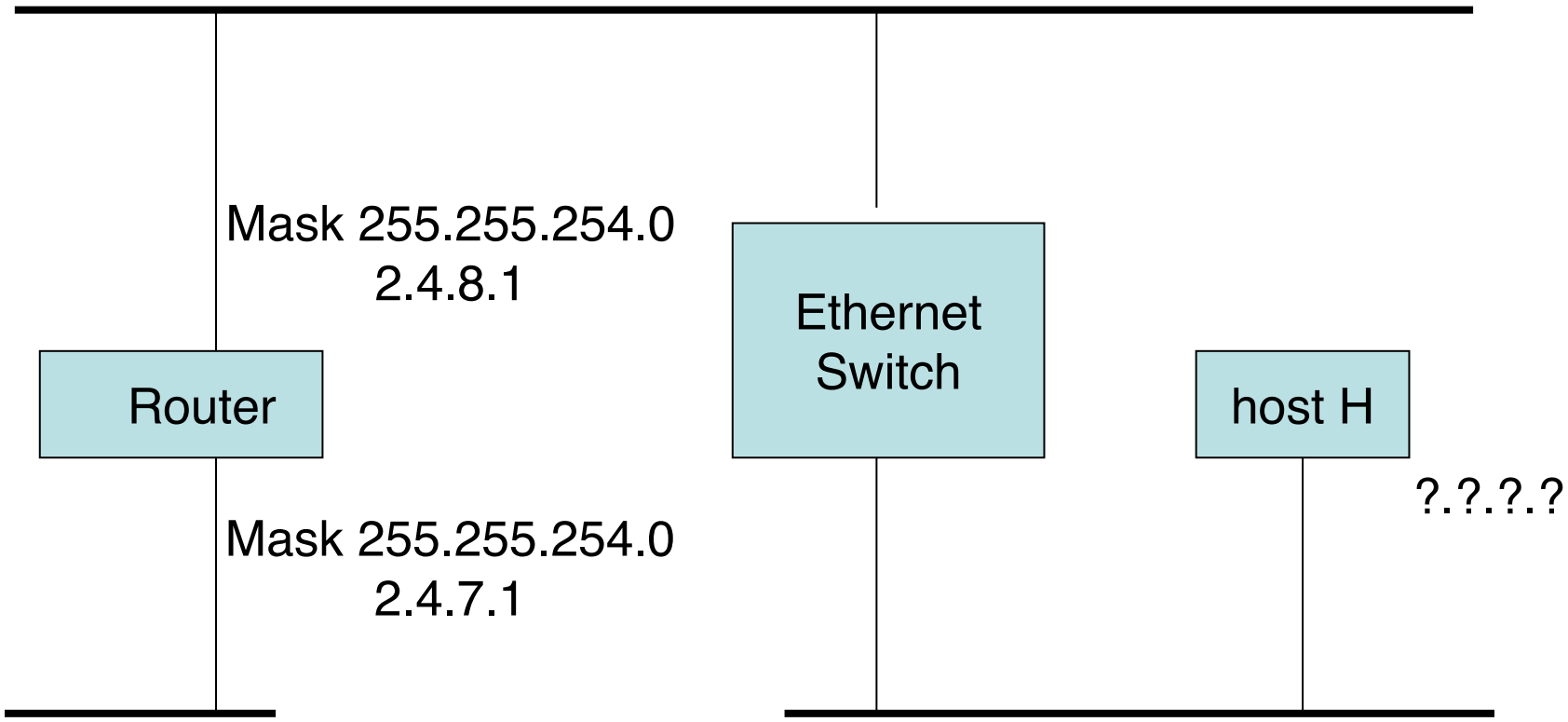
254 = 0b 1111 1110 i.e.

255.255.254.0 = 0b 1111 1111 1111 1111 1111 1110 0000 0000

23 bits equal to 1

Which address is a valid choice for H ?

A. only 2.4.8.2
B. only 2.4.9.1
C. Both A and B
D. None



Solution

Answer C

Router's north interface and H are in the same subnet

So H must have a subnet prefix of 23 bits.

We have:

- Router north's subnet prefix: 2.4.8 / 23 = 0000 0010 0000 0100 0000 100
- So A is correct because: 2.4.8.2 = 0000 0010 0000 0100 0000 1000 0000 0010

- B is also correct because: 2.4.9.1 = 0000 0010 0000 0100 0000 1001 0000 0001

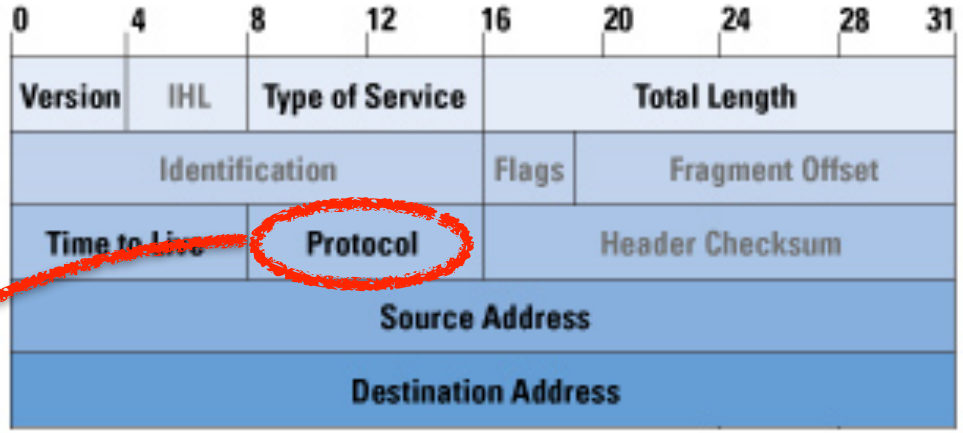
and: 2.4.9 / 23 = 0000 0010 0000 0100 0000 100

I.e.: the two prefixes are the *same*: 2.4.9/23 = 2.4.8/23 !

Reserved address blocks

0.0.0.0	**absence** of address
127/8	**loopback** addresses (this host, e.g. 127.0.0.1)
10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16	**private** addresses (e.g. at home): used by *anyone*, but *not* in the public Internet (internet routers drop packets destined to them)
100.64/10	private addresses used only by Internet Service Providers (ISPs)—**Carrier Grade NAT addresses**
192.88.99/24	**IPv6-to-IPv4** relay routers
169.254.0.0/16	**link local** addresses (can be used only by systems in same LAN)
224/4	**multicast**
240/4	reserved “for experimental/future use” until recently
255.255.255.255/32	link local (LAN) **broadcast**

IPv4 Packet Format



Higher layer protocol [1 = ICMP*, 6 = TCP, 17 = UDP]

Header 20 bytes (+ options, if any)

protocol overhead

payload

useful bits (higher-layer data)

✳ (ICMP is used to carry error messages at the network layer)

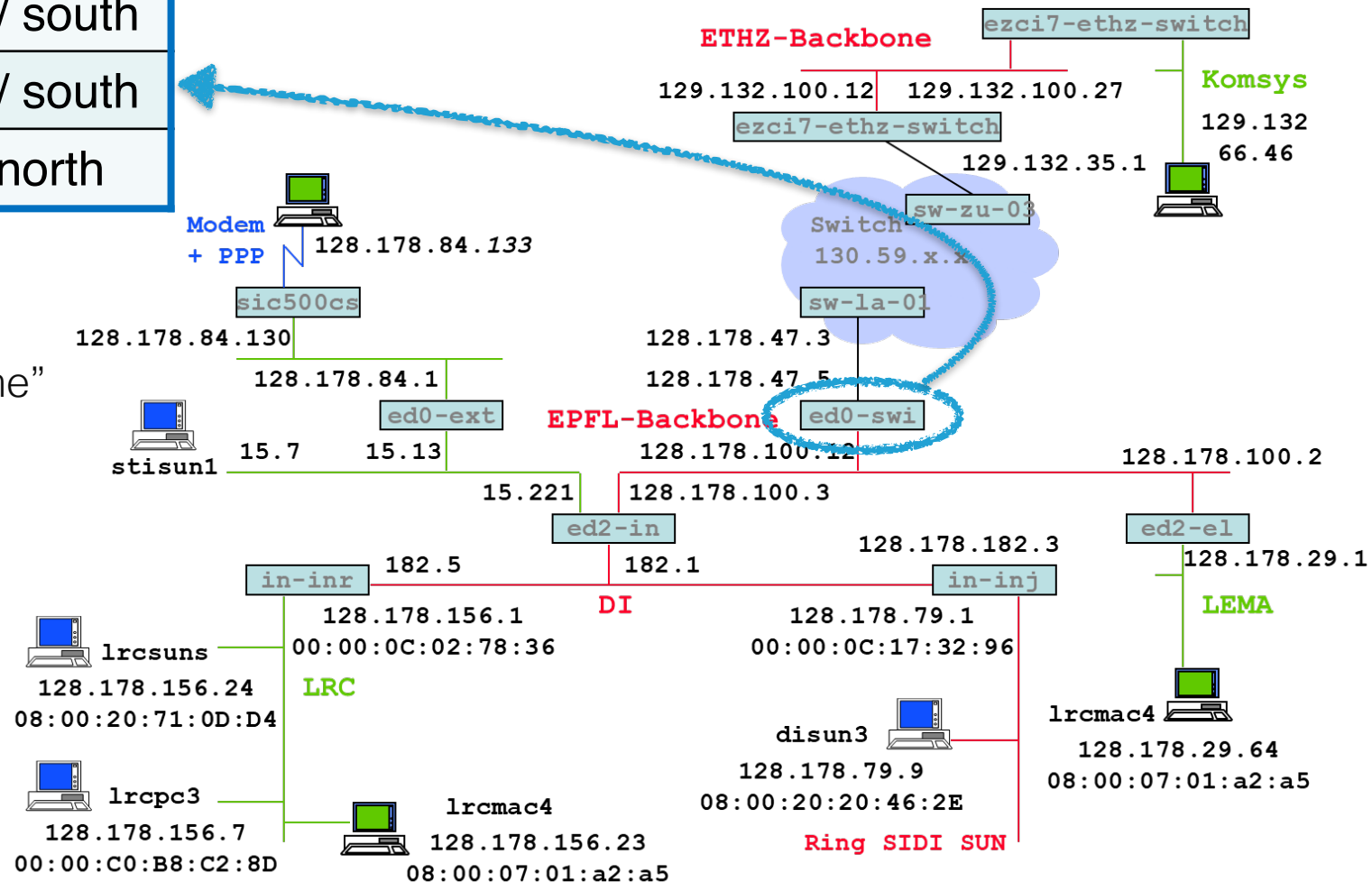
Forwarding table and longest prefix match: example from EPFL

Destination	Next-Hop / Interface
128.178.29/24	128.178.100.2 / south
128.178/16	128.178.100.3 / south
0/0	128.178.47.3 / north

- destination subnets in *aggregated* form
- next hops identified by both “IP” and “name”
- 0/0 (empty string) = default route

- Longest prefix match means:

```
if packet –> 128.178.*
    if packet –> 128.178.29*
        forward to ed2-el
    else
        forward to ed2-in
else
    forward to sw-la-01
```



3. IPv6

Why a new version ?

IPv4 address space is too small (32 bits $\rightarrow \approx 4 \cdot 10^9$ unique addresses)

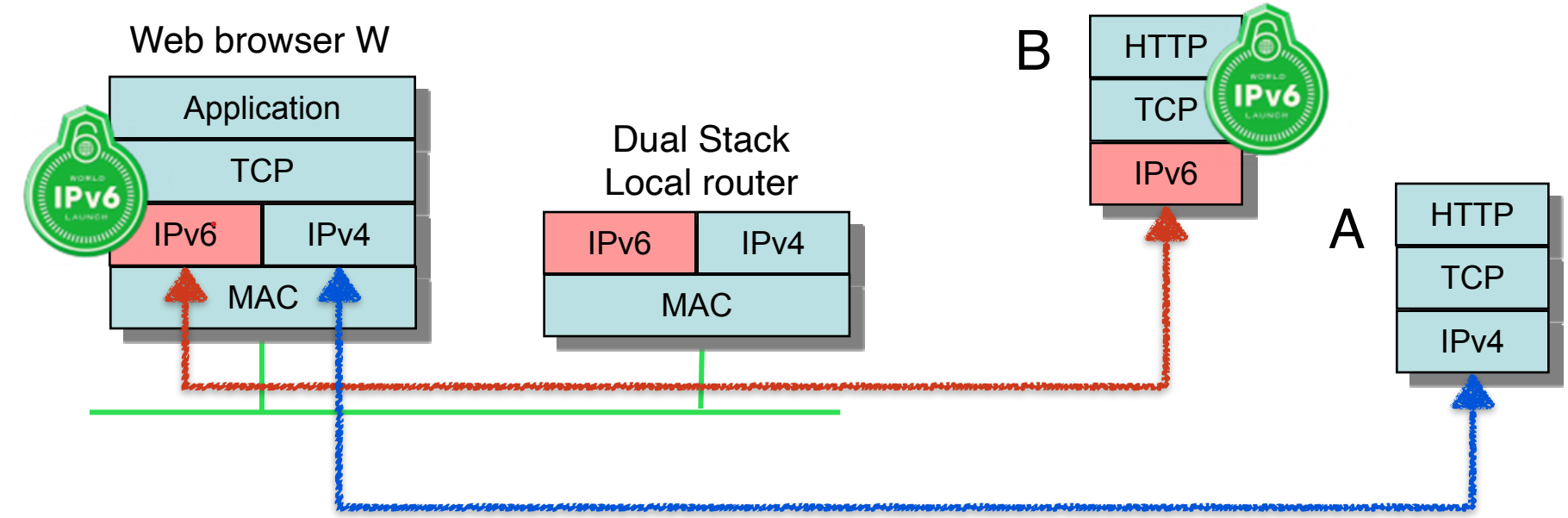
What does IPv6 do ?

Redefines packet format with larger addresses of: 128 bits ($\approx 3 \cdot 10^{38}$ unique addresses)

Otherwise, it offers essentially the same services as IPv4

But IPv6 is *incompatible* with IPv4; routers and hosts must handle separately

A can talk to W, B can talk to W, A and B cannot communicate at the network layer



IPv6 Addresses and Compression Rules

- We write them in hextets, prefer lower case letters, and separate them by “:” (colon)
  1 *hextet* = 1 *piece* = 16 bits = [0-4] hex digits;
  one IPv6 address uncompressed = 8 hextets
- leading zeroes inside a hextet can be *omitted*
- :: replaces any number of 0s in more than one hextet; can be used *at most once* in address

uncompressed form	compressed form
2002:0000:0000:0000:0000:ffff:80b2:0c26	2002::ffff:80b2:c26
2001:0620:0618:01a6:0000:20ff:fe78:30f9	2001:620:618:1a6:0:20ff:fe78:30f9

A few IPv6 global unicast addresses

The block 2000/3 (i.e. 2xxx and 3xxx) is for global/public unicast addresses

2001:620::/32	Switch
2001:620:618::/48	EPFL
2001:620:8::/48	ETHZ
2a02:1200::/27	Swisscom
2001:678::/29	provider independent address
2001::/32	Teredo (tunnels IPv6 in IPv4)
2002::/16	6to4 (tunnels IPv6 in IPv4)

Networks served by a provider use blocks that are *subsets* of the provider's address block, (e.g. check EPFL, ETHZ and SWITCH)

Reserved address blocks

::/128	**absence** of address
::1/128	**loopback** address (this host)
fc00::/7 (i.e. fcxx: and fdxx:) for example: fd24:ec43:12ca:1a6:a00:20ff:fe78:30f9	unique local addresses = **private** networks (e.g. in EPFL): *not* to be used in the public Internet
fe80::/10	**link local** addresses (used only by systems in same LAN)
ff00::/8	**multicast**
ff02::1:ff00:0/104	**solicited node multicast** (see NDP later)
ff02::1/128	**link local broadcast**
ff02::2/128	**multicast to all link-local routers** (in same LAN)

EPFL Private

IPv6 public and private prefixes are structured

an EPFL public address:

2001:620:618:1a6:a00:20ff:fe78:30f9

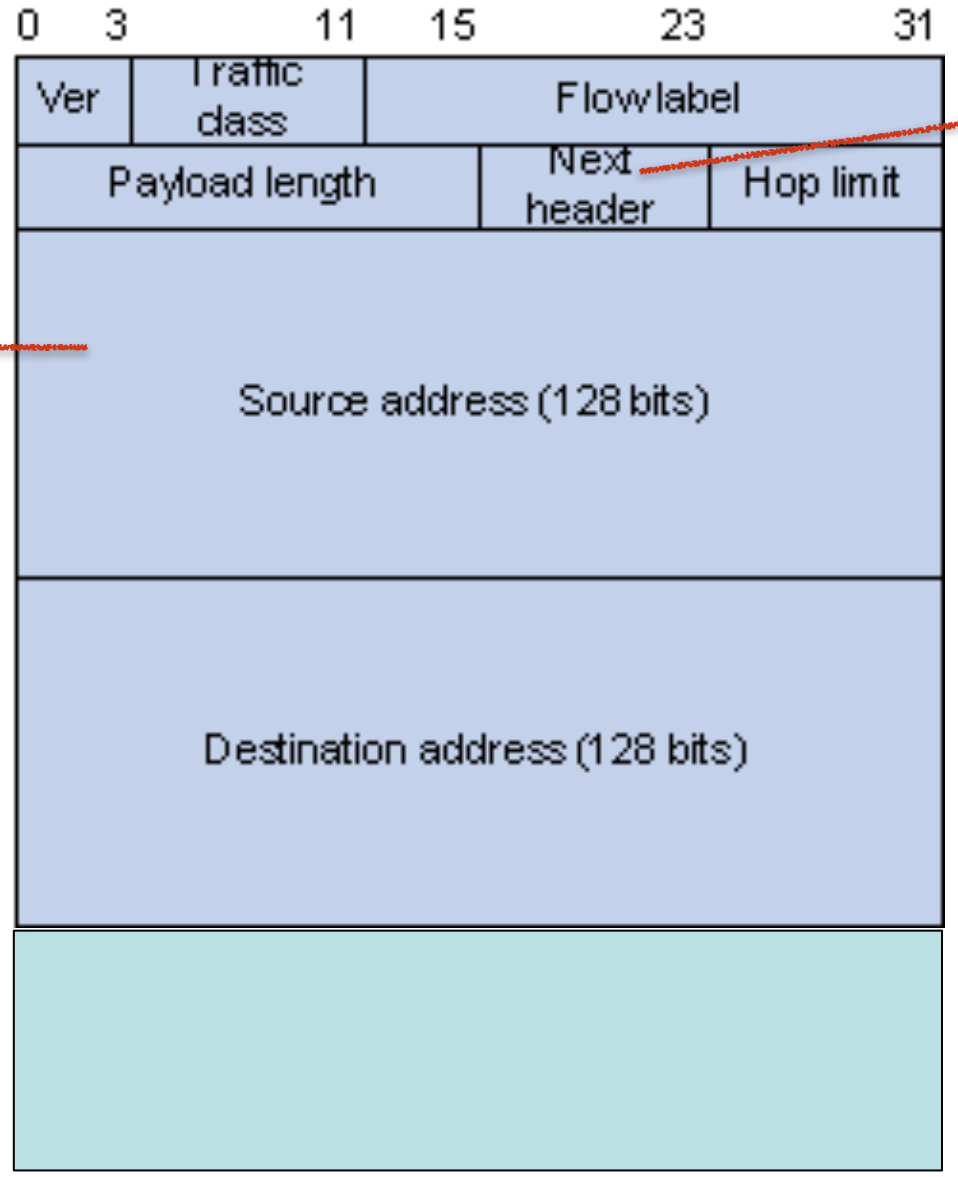
an EPFL private address:

fd24:ec43:12ca:1a6:a00:20ff:fe78:30f9

This is a private address

EPFL private prefix

IPv6 Packet Format



Higher layer protocol
[58 = IPv6-ICMP,
6 = TCP,
17 = UDP]

Address field
16 bytes

Header
40 bytes
(+ options,
if any)

Payload

***We will see the functions of the fields other than the addresses in a following lecture*

IPv6 Forwarding table: same example from EPFL at ed0-swi

IPv6 forwarding table

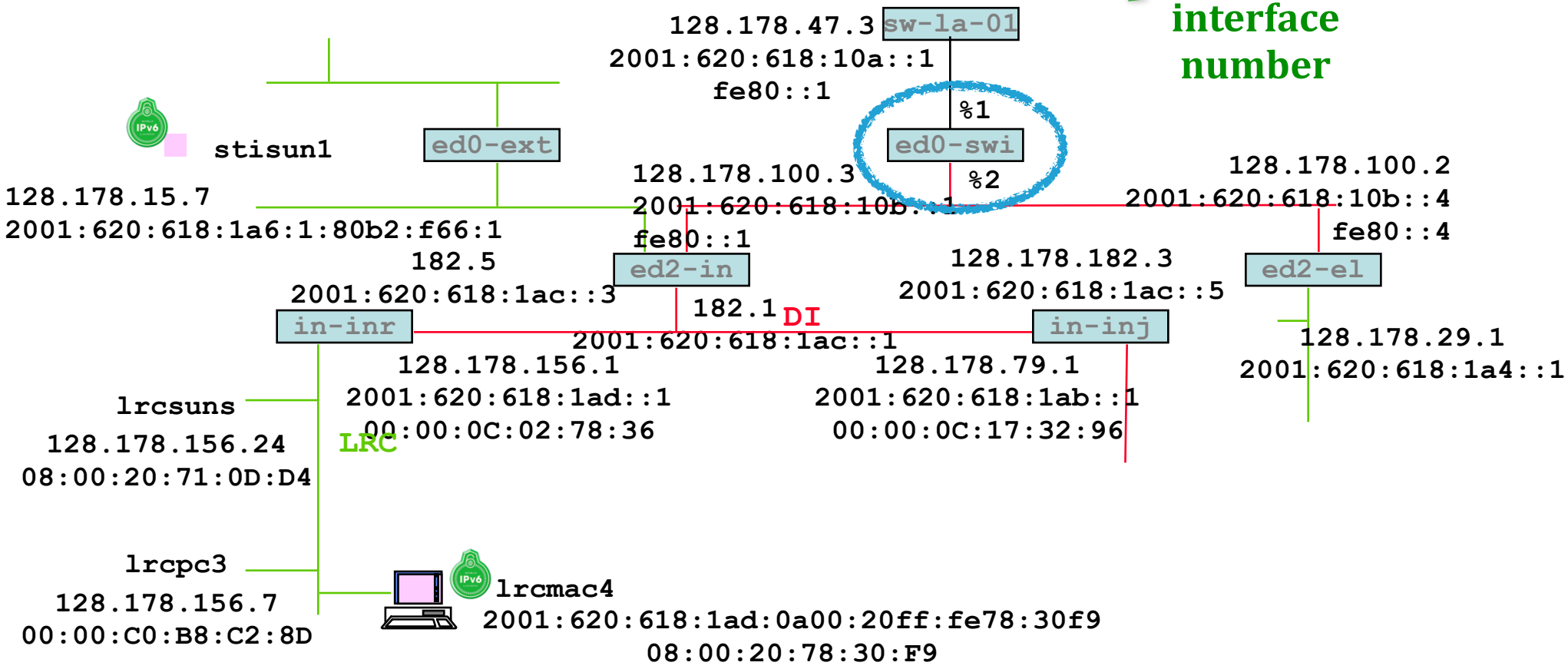
Destination	Next-Hop-IP%Interface
2001:620:618:1a4/64	fe80::4%2
2001:620:618/48	fe80::1%2
::/0	fe80::1%1

IPv4 forwarding table

Destination	Next-Hop / Interface
128.178.29/24	128.178.100.2 / south
128.178/16	128.178.100.3 / south
0/0	128.178.47.3 / north

IP address of next hop

interface number



The dotted decimal notation for $0102{:}ffff$ is …

A. 1.2.255.255
B. 16.32.255.255
C. 228.393.255.255

Go to web.speakup.info or download speakup app

Join room 46045

Solution

Answer A
Recall the mapping (hex) ff → (decimal) 255

In full, the hexadecimal notation «2001::ada:bada» means…

A. 2001:0ada:bada
B. 2001:0000:0000:0000:0000:0000:0ada:bada
C. 2001:0000:0ada:bada
D. 2001:0000:ada:bada
E. None of the above

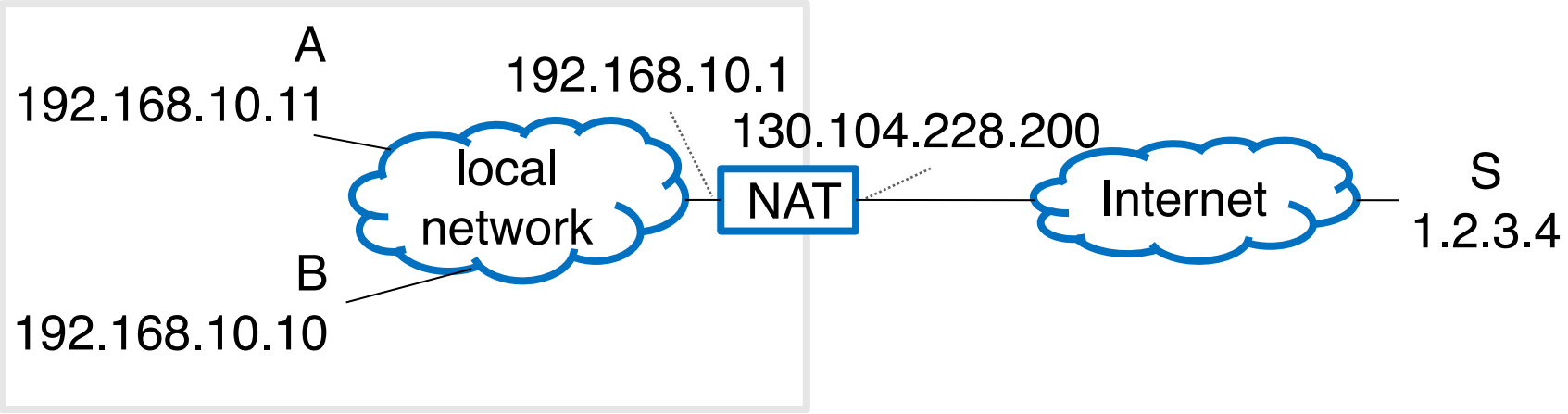
Go to web.speakup.info or download speakup app

Join room 46045

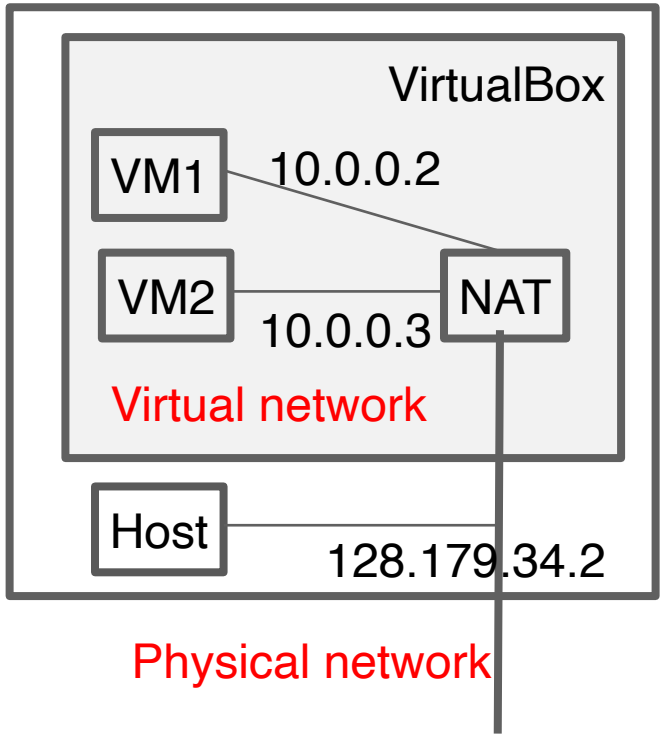
Solution

Answer B. The convention :: means as many 0's as required to make the string 128 bits.
Leading zeros are omitted, so that :bad: means :0bad:

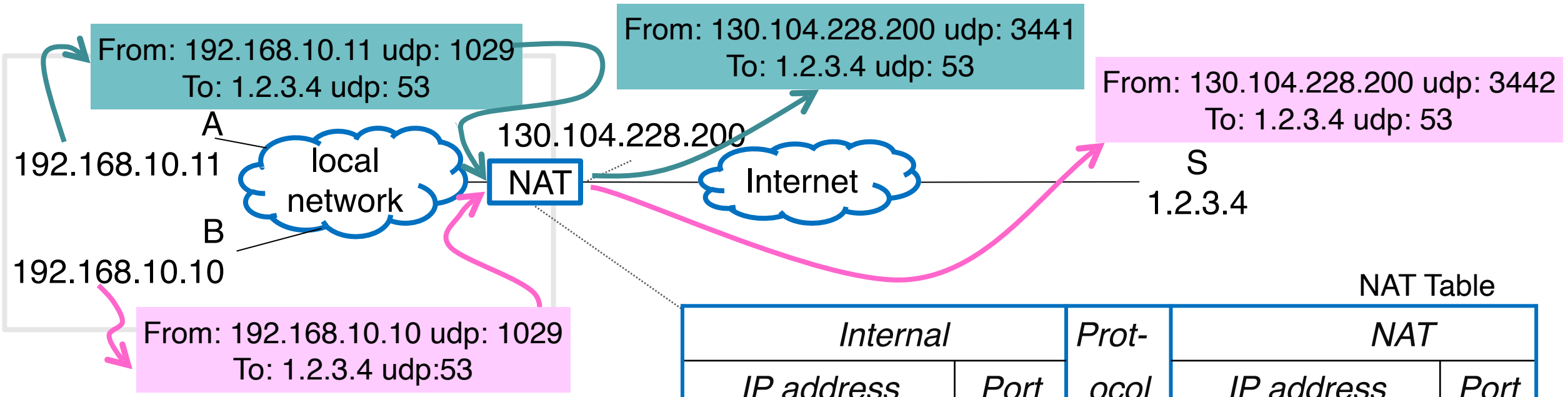
4. NAT (Network Address Translation) box



- *Why* invented? To allow $n > 1$ devices to *share* a *single* public IP address; e.g.:
  - Internet service provider gives you a single IPv4 address,
    but you have *n* devices at home and need more addresses.
  - A virtualization platform offers *n* VMs in one host and allows them to communicate with outside,
    while using the single IP address of the physical machine.
- *What* does it do?
  - NAT translates (= *masquerades*)
    an *internal IP* address and *internal port* number into *NAT IP* address and *NAT port* number
  - Internal addresses are typically ***private*** addresses, ports are either UDP or TCP
  - From outside, one sees only the (***public***) NAT IP address and a NAT port
- NAT is a network-layer middle box, but *violates*:
  - the IP principle that public addresses should identify hosts *uniquely*
  - layering —> manipulates 2 layers to work



How does NAT treat *outgoing* traffic?

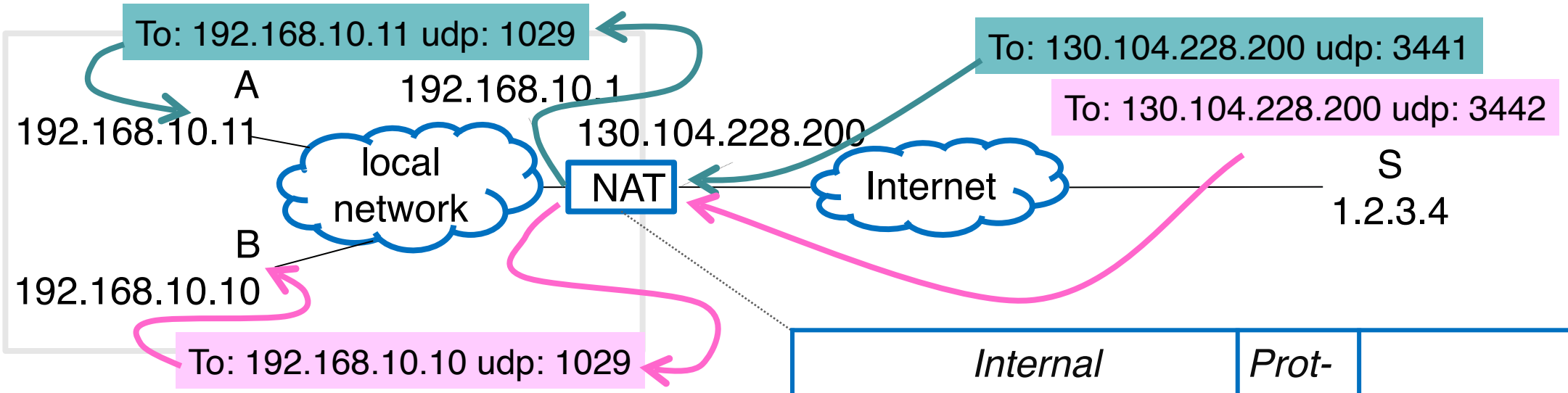


NAT Table

Internal		*Prot-*	*NAT*	
IP address	*Port*	*ocol*	*IP address*	*Port*
192.168.10.11	1029	udp	130.104.228.200	3441
192.168.10.10	1029	udp	130.104.228.200	3442

- When a packet goes from internal network (a.k.a. LAN) to the external network (a.k.a.WAN), NAT:
  - translates *source IP address + port* by *changing* the IP and UDP headers
  - stores this mapping in the *NAT table*
  - finally forwards the packet

How does NAT treat *inbound* traffic?



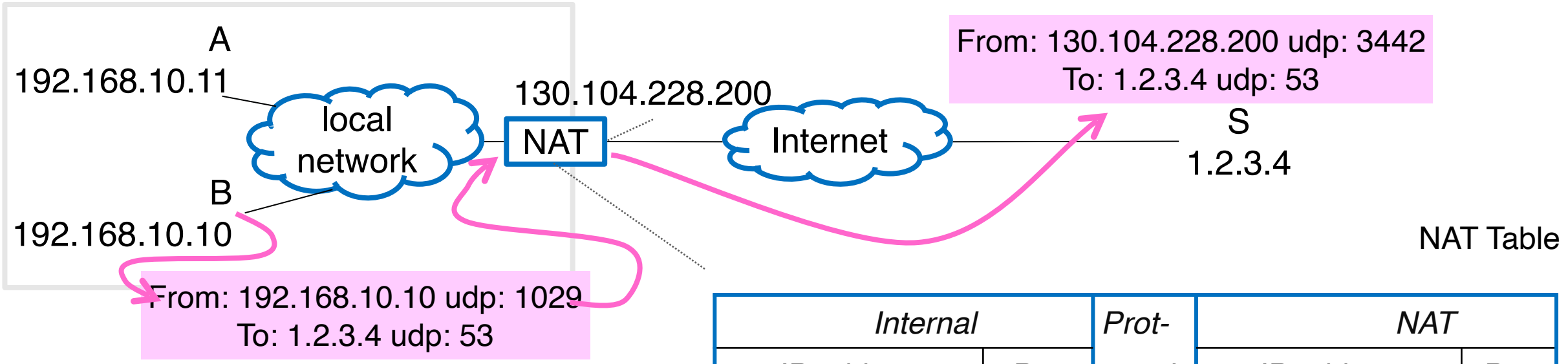
NAT Table

Internal		*Prot-ocol*	*NAT*	
IP address	*Port*		*IP address*	*Port*
192.168.10.11	1029	udp	130.104.228.200	3441
192.168.10.10	1029	udp	130.104.228.200	3442

- When packets come from external network to internal network, NAT translates *destination IP address + port*
- IP forwarding is based on *exact matching* in the NAT table

How does NAT maintain NAT table ?

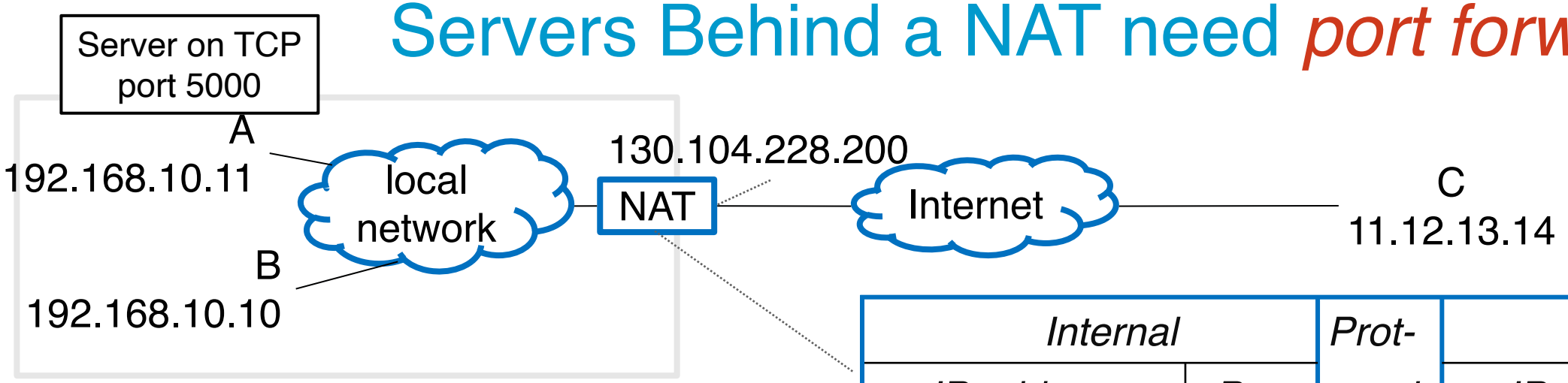
- NAT creates a NAT table entry *on-the-fly* (automatically),
  i.e. when client on internal network contacts server on external network
- NAT chooses a NAT port that does *not* create *collision* in the table
- In Linux, NAT table is implemented with `iptables`



NAT Table

Internal		*Prot-ocol*	*NAT*	
IP address	*Port*		*IP address*	*Port*
192.168.10.11	1029	udp	130.104.228.200	3441
192.168.10.10	1029	udp	130.104.228.200	3442

Servers Behind a NAT need *port forwarding*



NAT Table

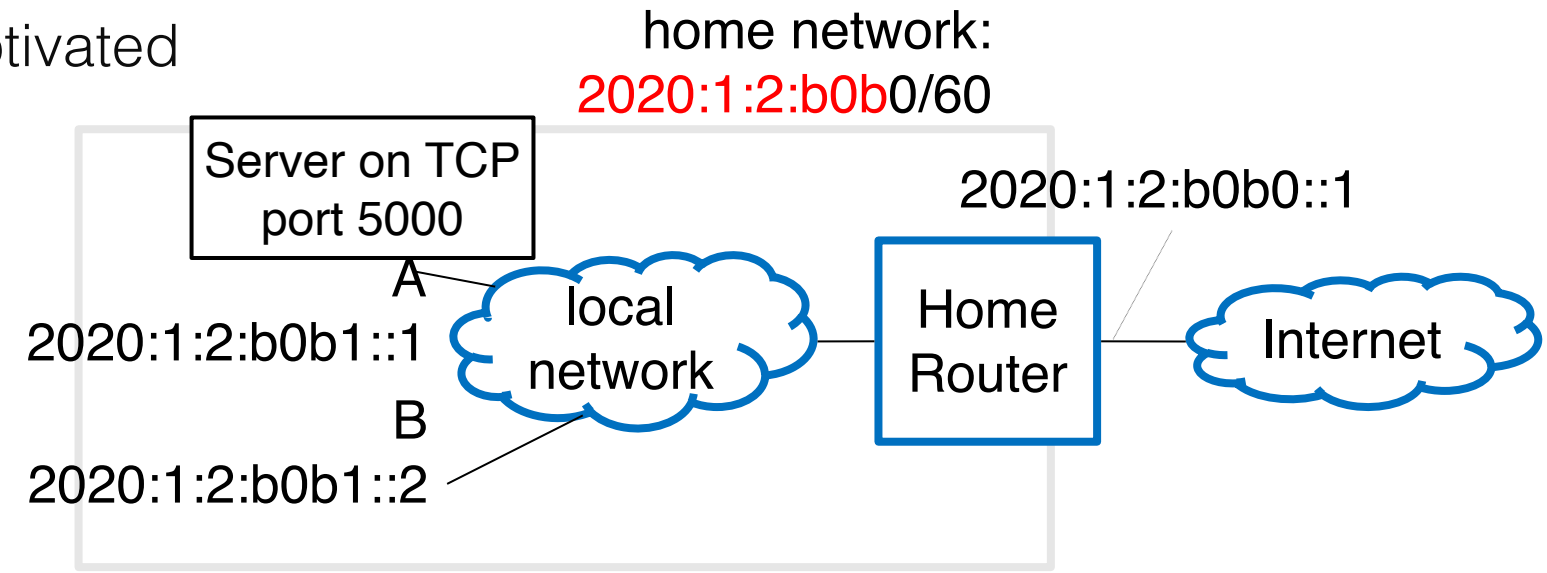
Internal		*Prot-*	*NAT*	
IP address	*Port*	*ocol*	*IP address*	*Port*
192.168.10.11	1029	udp	130.104.228.200	3441
192.168.10.10	1029	udp	130.104.228.200	3442
192.168.10.11	5000	tcp	130.104.228.200	5000

- **Problem**: Assume A has a server at tcp port 5000; automatic operation of NAT requires communication to be started by A, which is not done for a server
- **Solution**: *manual configuration* of port forwarding in NAT
  - C now connects to A at 130.104.228.200 port 5000
  - A needs to know its NAT IP address in advance and advertize it to potential clients like C
  - A can discover its NAT IP address via a STUN server or UPnP (if A and NAT use it)
- **Side benefit**: *protection* — a server port can be accessed, only if explicitly configured, while servers in the public internet are generally exposed to attacks and need to be actively protected

Manual configuration of *port forwarding* in NAT

NATs and IPv6, Version 1

NAT was developed for IPv4, motivated by lack of IPv4 addresses



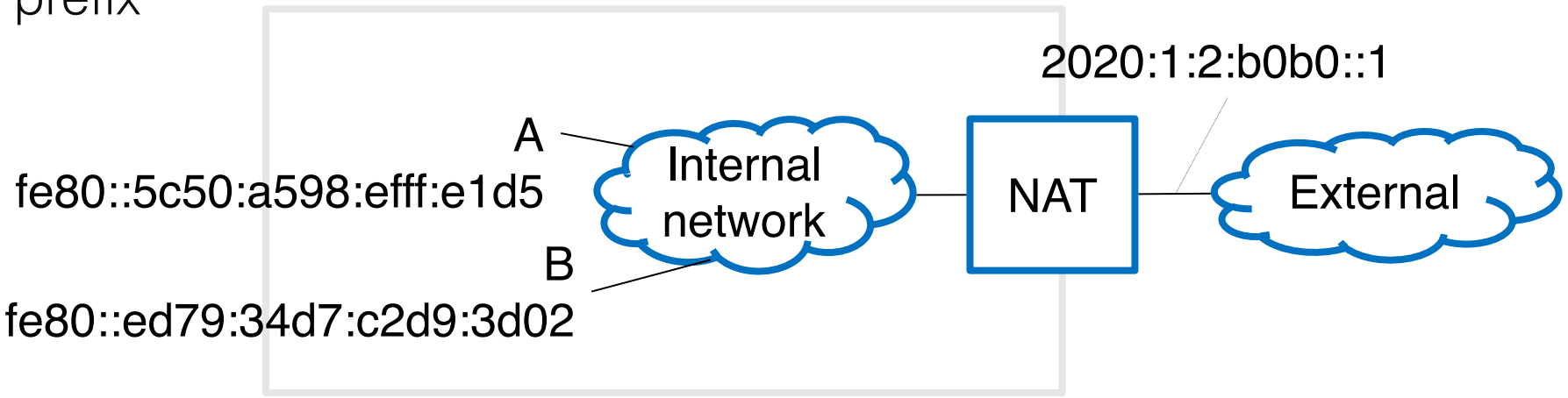
In IPv6, home routers often:

- do not use NAT,
  - their provider typically allocates a *block of IPv6 addresses*, not just one as with IPv4 [see slide "DHCP with Prefix Delegation"]

- provide protection by acting as a *filtering router*, which:
  - allows communication from outside, *only if* initiated from inside, unless manually configured

NATs and IPv6, Version 2

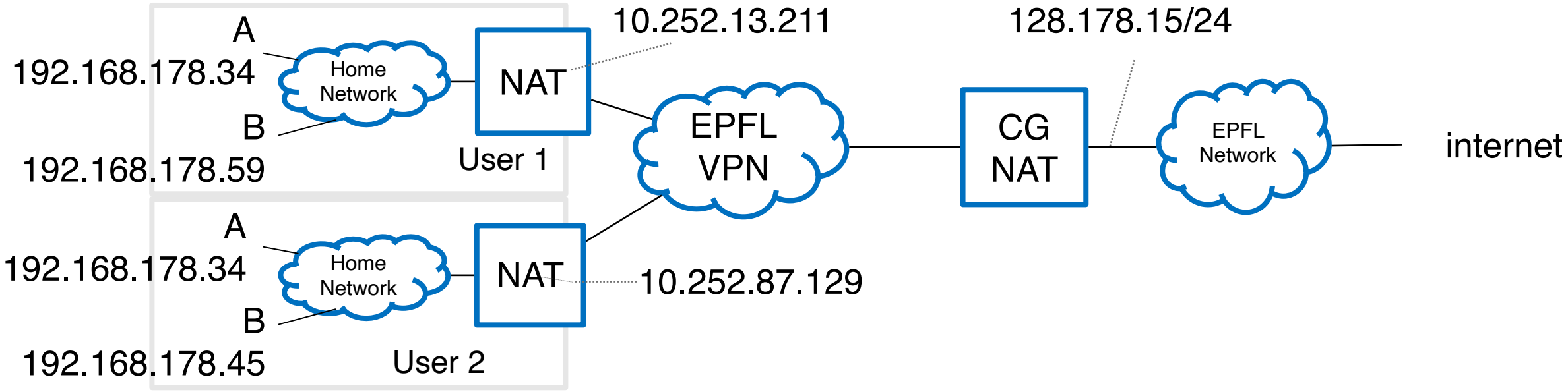
- Some systems still use NAT with IPv6, if local network receives *only one IPv6* address, not an entire prefix



- How?
  - use *NAT with link local IPv6 addresses* in internal network
  - restrict the internal network to *one LAN* (only one subnet)

- Used e.g. by Virtual Box (“NAT network”).

Carrier-Grade NAT (CG NAT)



- Shares p external/NAT addresses among $n > p$ internal hosts
  - e.g.: VPN access of EPFL uses the block 128.178.15/24 ($p = 256$);
  - VPN user 1 has IP address 10.252.13.211, may appear in the public internet with an address 128.178.15.x, while VPN user 2 may appear with the same IP address (or not)
- Allows for *"NATs behind NATs"*:
  - e.g.: A may appear as 192.168.178.34 at home, as 10.252.13.211 at EPFL's VPN and as 128.178.15.x in the public internet
- Some internet service providers do this, in order to *reduce* the number of *IPv4 addresses* allocated to end-users —> they use block 100.64/10 instead of 10/8

Few of the existing NAT variations…

- A *full cone NAT* is one where all requests from the *same internal IP address+port* are *always* mapped to the *same NAT IP address+port*.

  $$\text{(internal addr+port)} \xrightarrow{fixed} \text{(NAT addr, NAT port)}$$

  Furthermore, any external host can send a packet to the internal host, by sending a packet to the mapped external address.

- A *symmetric NAT* is one where all requests from the *same internal IP address+port*, to a *specific destination IP address+port*, are always mapped to the *same NAT IP address+port*.

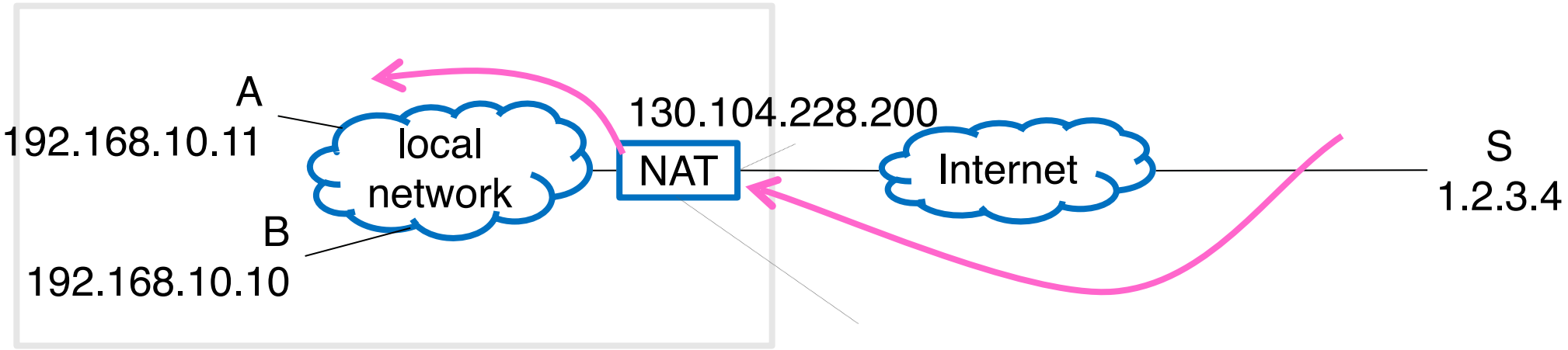
  $$\text{(internal addr+port, destination addr+port)} \xrightarrow{fixed} \text{(NAT addr, NAT port)}$$

  If the same host sends a packet with the same source address and port, but to a different destination, a different mapping is used. So, *only* the external host that receives a packet can send a packet back to the internal host.

- ICMP* packets don't have a port number. Some NATs don't support ICMP, but many do. They manipulate e.g. the *ICMP echo request identifier* (in ping messages) as a replacement of port number.

✳ (ICMP is used to carry error messages at the network layer)

From WAN to LAN, the NAT may modify…

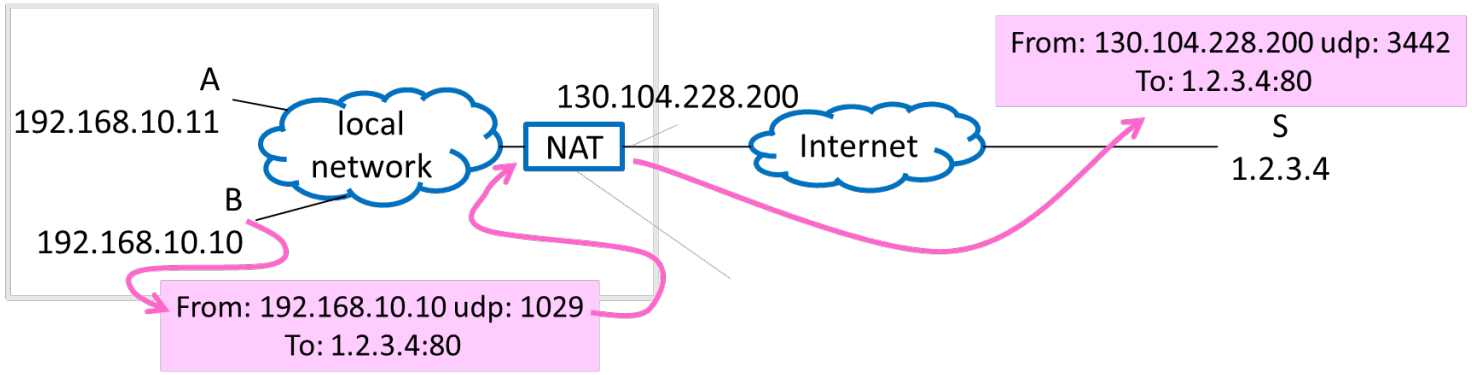


A. The source port
B. The destination port
C. None of the above
D. I do not know

Go to web.speakup.info or download speakup app

Join room
46045

When a NAT has a packet to forward and an association exists in the NAT table…



A. The NAT looks for a longest prefix match
B. The NAT looks for an exact match
C. None of the above
D. I do not know

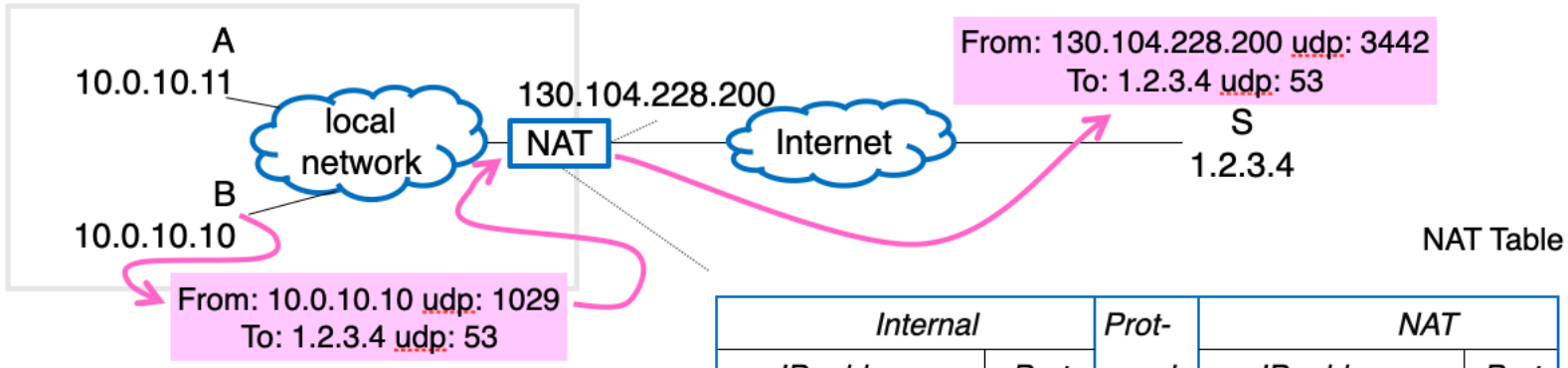
Go to web.speakup.info or download speakup app

Join room 46045

Solution

Answer B in both cases.

Suppose that the internal subnet uses a private block of addresses 10.0.0.0/8, what is the max number of internal hosts (end-systems) that a typical NAT can support



NAT Table

Internal		*Prot-*	*NAT*	
IP address	*Port*	*ocol*	*IP address*	*Port*
10.0.10.11	1029	udp	130.104.228.200	3441
10.0.10.10	1029	udp	130.104.228.200	3442

A. 2^8

B. $2^8 - 1$

C. As many as the overall number of available UDP/TCP port numbers (i.e. ≤ 2*65536)

D. None of the above

E. I do not know

Go to web.speakup.info or download speakup app

Join room 46045

Solution

Answer C.

- NAT creates table entries on the fly by also trying to avoid collisions on the port number. So, the max number of entries it can have is equal to the overall number of the TCP/UDP port numbers that it is allowed to use (i.e. that are not reserved for other purposes and not blocked by a firewall). This number is smaller than 2*65536, since each port number is 16bits long; hence for each of TCP and UDP, we have 2^16 = 65536 possible numbers.
- The latter is smaller than the overall number of private IP addresses in the allocated block (which is 2^24 addresses, because the subnet mask is /8).
- So, it is also the max number of hosts that the NAT can support—the max number of hosts is achieved by assuming that each host has only one UDP or TCP connection with some other host outside the LAN.

5. Configuration of a network interface

- A *host* IP interface is configured with:
  1. IP address of this interface
  2. Mask of this interface
  3. IP address of default gateway router
  4. IP address of DNS server

- Can be configured ***manually***, or ***automatically*** with:
  IPv4 → DHCP (Dynamic Host Configuration Protocol)
  IPv6 → DHCP stateful, SLAAC (stateless), DHCP stateless

- Same applies to *routers* connected to a provider
  IPv4 → PPP (Point to Point protocol): automatic config for telecom lines (modem, ADSL)
  IPv6 → PPP, DHCP with Prefix Delegation

DHCP (Dynamic Host Configuration Protocol)

- *Client-Server* app: configuration info is kept in DHCP server; host contacts the server when it needs an IP address
- Common in IPv4; also works with IPv6 (called stateful DHCP)

Problem: Host cannot contact the DHCP server since it does not know of *any* IP address initially

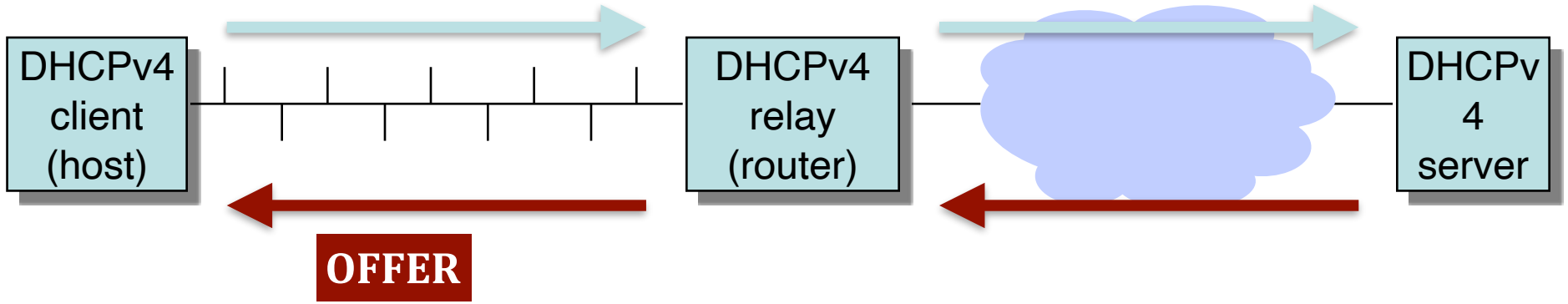
Solution: Use broadcast (in IPv4) or multicast (in IPv6) inside the LAN to *discover* DHCP servers

Details:

- Either DHCP server exists in the same LAN or gateway router implements a "*DHCP Relay*" function
- DHCP uses two phase commit with acknowledgement to avoid inconsistent reservations
- DHCP server uses limited lifetime allocations — renews lease after expiry

Example (IPv4):

DISCOVER <MAC addr of client>
UDP, dest port = 67, src port = 68
IPv4 dest addr = 255.255.255.255
IPv4 src addr = 0.0.0.0



(Self)-Autoconfiguration in IPv4

- Why? if *no DHCP* is available and *no manual* configuration is done

- What it does? Enables interconnection in unmanaged or “router-less” local networks (à la old AppleTalk), but not in a general setting

- How? When a host boots, it:
  i. picks an IPv4 link local address *at random* in block 169.254/16
  ii. performs an *address duplication test* using *broadcast* IP

- Implemented in Windows, not always supported in Linux

Stateless Address Autoconfiguration (SLAAC) in IPv6

Why invented? To configure interfaces automatically without DHCP servers in IPv6

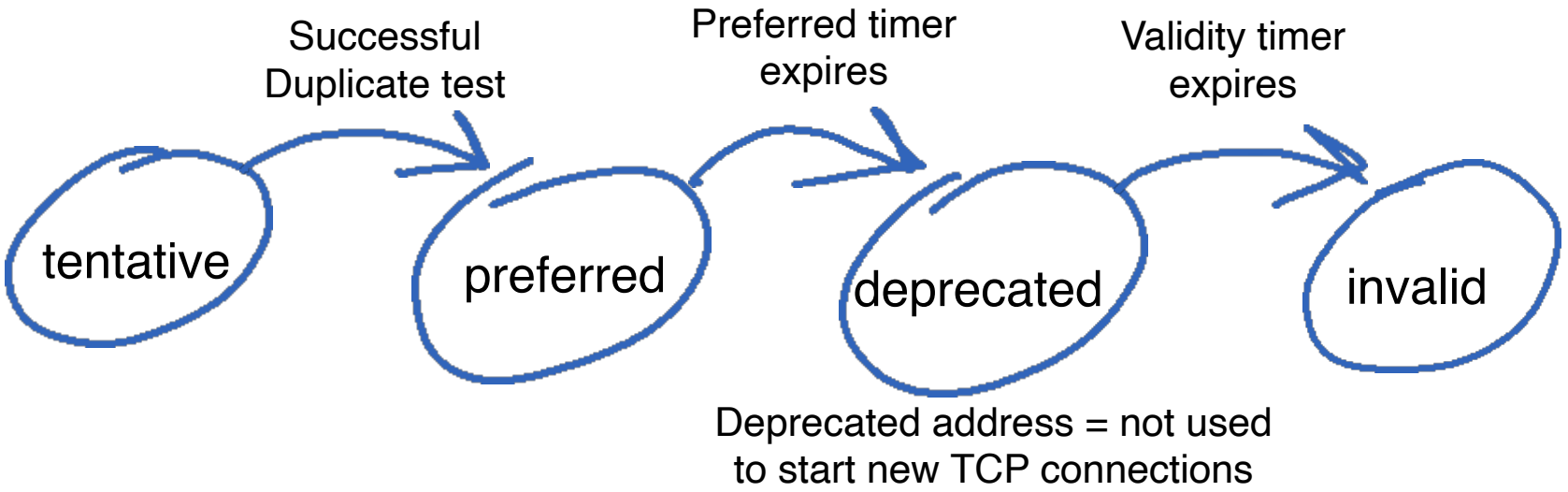
How it works?

1. host auto-configures a *link local address* with:
   a 64-bit prefix (fe80::/64) + a 64-bit host suffix that is derived by one of the following:
   (i) manually, e.g. ::1
   (ii) algorithmically from MAC address ("modified EUI 64")
   (iii) randomly (temporary validity) [see next slide]
   (iv) via "secure autoconf" (RFC7217), i.e. randomly but also related to subnet [see next slide]
   (v) cryptographically with CGA [see slides on "Secure NDP"]
2. host performs *address duplication test* using *solicited node multicast* address (which has the same 24 last bits as the address it has selected)
3. host computes a globally valid address by obtaining the network prefix from LAN's routers (via *multicasting*);
   - prefix may not be 64 bits, then new host part is derived with same methods

➡ SLAAC is fully *automatic* but does *not* provide DNS information

Random and RFC7217 Addresses avoid Host Tracking

Solution 1: temporary, random host suffix



Solution 2: "Secure autoconf" (RFC7217), not fully random but intractable host suffix:

```
Host suffix = hash (interface id, subnet prefix, secret key)
```

- Key is obtained with OS installation
- Address remains the same whenever host visits same subnet
- Address changes randomly across different subnets

How to learn IP of DNS (after SLAAC)?

- Stateless DHCP
  - gives the IP address of the DNS server to the host

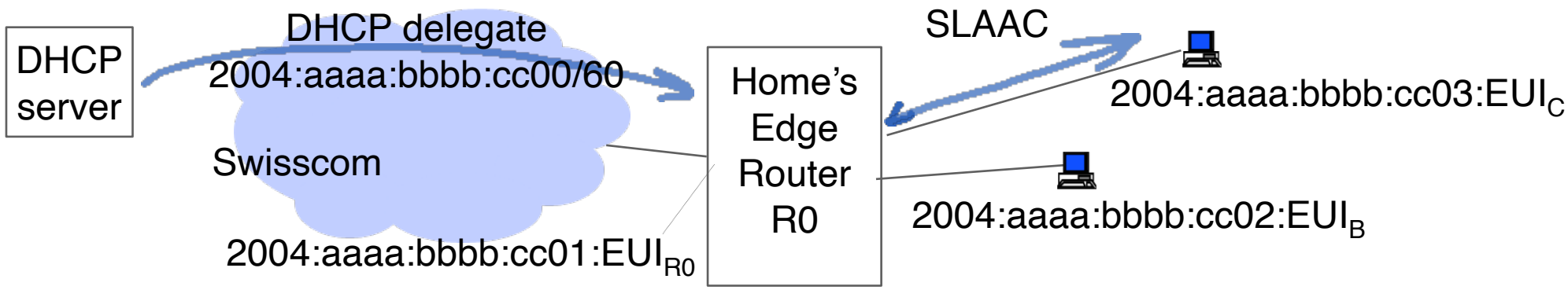
- Router Advertisements (RFC 6106)
  - Host accesses routers in the same LAN via *multicasting* asking for DNS info
  - Router returns the IP address of the DNS server to the host

DHCP with Prefix Delegation

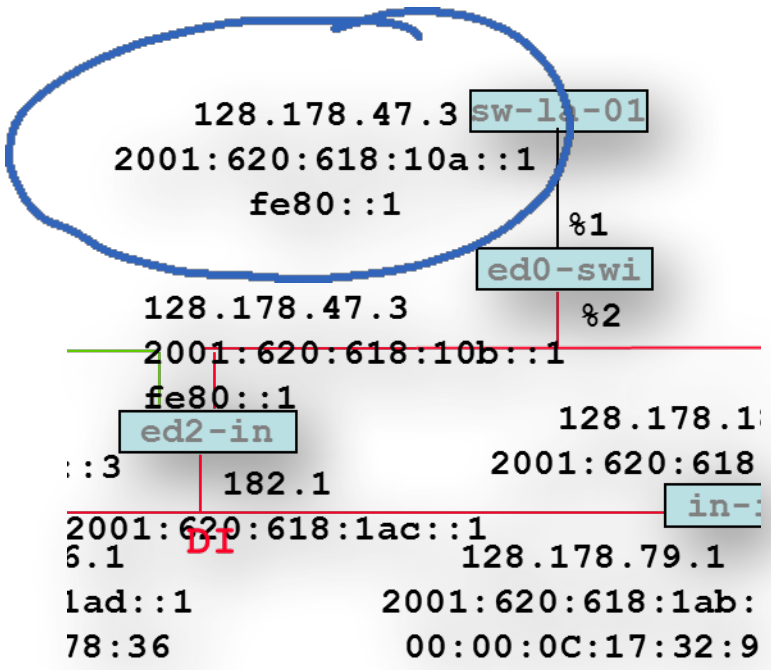
Why ? A home (or enterprise) IPv6 router is configured by ISP using DHCP; local devices are autoconfigured using e.g. SLAAC.
And home router needs to advertize an IPv6 prefix for the *entire* home network.

How ?

- ISP DHCP server (delegating router) provides the home router with not just an IPv6 address but also the network prefix that this router can *delegate* to its devices
- If the prefix used for the link from ISP to the home router is a subset of the delegated prefix, then it is *excluded* from the delegation (RFC 6603).
  E.g. 2004:aaaa:bbbb:cc01/64 is excluded from the delegated prefix 2004:aaaa:bbbb:cc01/60.



Multiple Addresses per Interface are the Rule with IPv6



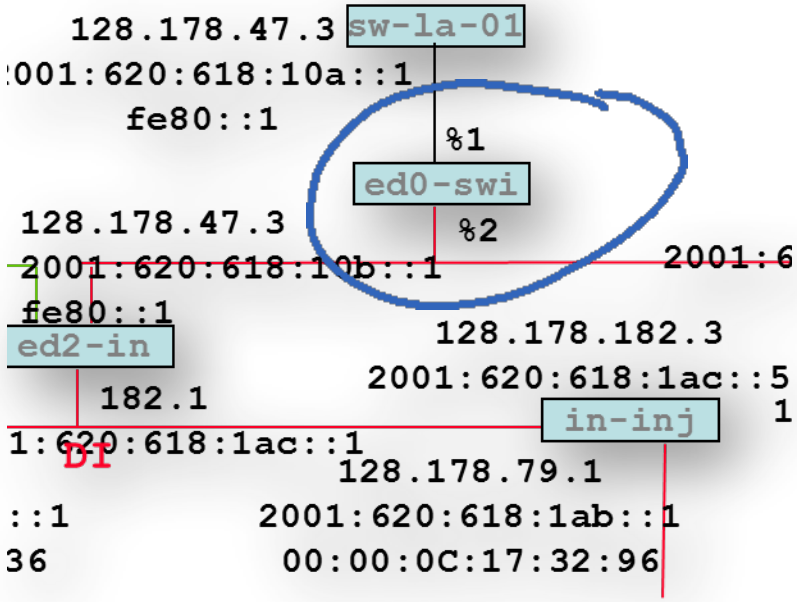
A host interface typically has

- one or several *link local* addresses
- plus one or several global *unicast* addresses (secure (CGA) address, temporary addresses)

The *preference selection algorithm*, configured by OS, says which address should be used as source address – see RFC 3484

In contrast, in IPv4:
there is usually only one IP address per interface

Zone Index



Identifies an interface inside one machine that has several interfaces (e.g. a router)

- typically visible in Windows machines
- never inside an IP packet

E.g. fe80::1%2 means: the destination IPv6 address fe80::1 on interface %2

When an IPv4 host uses DHCP, which of the following information does it acquire:

A. its IP address;
B. its subnet mask
C. its default gateway address
D. its DNS server address

A. A
B. A, B
C. A, B, C
D. A, B, C, D
E. None of the above

Go to web.speakup.info or download speakup app

Join room 46045

Solution

Answer D

With SLAAC an IPv6 host has…

A. A link local address and, if a router is present in the subnet, also a global unicast address
B. If a router is present in the subnet a global unicast address and no link-local address
C. None of the above
D. I do not know

Go to web.speakup.info or download speakup app

Join room 46045

Solution

Answer A

Private vs link local addresses (for those who may be have been confused)

Private addresses are allocated *administratively* (i.e., by a local network administrator, either statically, or automatically allocated at a single point by a suitably configured DHCP server).

Link-local addresses are allocated *automatically* (e.g. via SLAAC) when a computer has not been configured with a static IP-address and cannot find a DHCP server.

- Why do we need a separate space for link local?

So that the link-local addresses cannot possibly conflict with those allocated by a local, but temporarily unavailable, DHCP server.

- How are link local and private addresses handled by gateway routers?

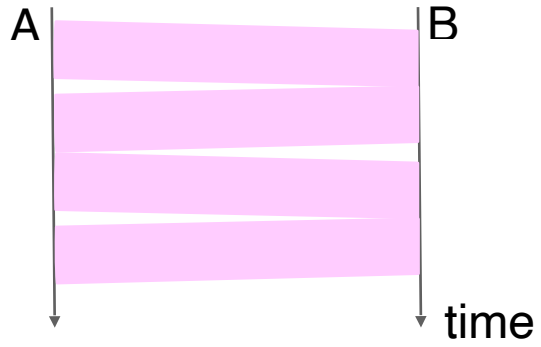
They are not routed. This is so that lots of private LANs can use the same addresses without any routing conflicts arising. Such LANs must use NAT to hide their many private addresses behind one (or a few) public addresses.

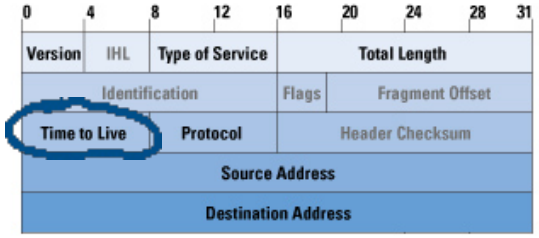
- Can both interfaces of a gateway router/NAT (internal and external) be configured with private addresses?

Yes, but they cannot belong to the same subnet. We use a different subnet at each “side” of the router/NAT.

6. Revisit IP Header: Hop Limit (HL) / Time to Live (TTL)

Why? Avoid looping packets in transient loops. Transient loops may exist due to changes to forwarding tables + propagation latency.

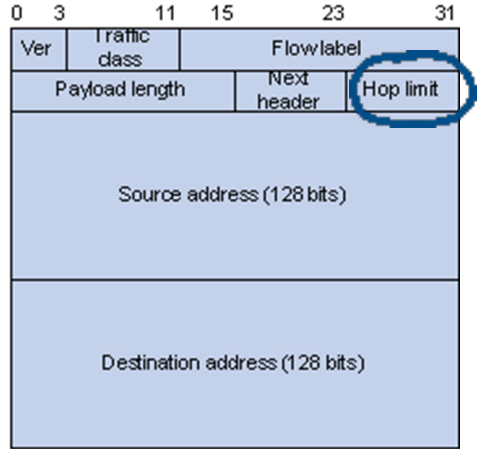




If propagation time
is small compared to transmission time, a single packet (or a few packets) caught in a loop can congest the line.

How?

- Every IP packet has a field on 8 bits (from 0 to 255) (called Hop Limit in IPv6 / Time To Live in IPv4) that is decremented at every IP hop
- Every router or NAT decreases HL/TTL, switches do *not*
- When HL/TTL reaches 0, packet is discarded
- At source, value is 64 in principle



Traceroute leverages the HLs/TTLs

- It sends a series of packets (using UDP) to a destination, with TTL = 1, 2, 3, …
- Routers on the path discard packets and send ICMP error message back to source
- Source learns address of router on the path by looking at source address of error message

- tracert (windows) is similar, but uses ICMP

```
Tracing route to www.google.com [2a00:1450:4008:800::1012]
over a maximum of 30 hops:

  1     1 ms    <1 ms    <1 ms  cv-ic-dit-v151-ro.epfl.ch [2001:620:618:197:1:80b2:9701:1]
  2    <1 ms    <1 ms    <1 ms  cv-gigado-v100.epfl.ch [2001:620:618:164:1:80b2:6412:1]
  3    <1 ms    <1 ms    <1 ms  c6-ext-v200.epfl.ch [2001:620:618:1c8:1:80b2:c801:1]
  4     1 ms    <1 ms    <1 ms  swiEL2-10GE-3-2.switch.ch [2001:620:0:ffdc::1]
  5    <1 ms    <1 ms    <1 ms  swiLS2-10GE-1-2.switch.ch [2001:620:0:c00c::2]
  6     7 ms     7 ms     7 ms  swiEZ1-10GE-2-7.switch.ch [2001:620:0:c03c::2]
  7     8 ms     8 ms     7 ms  swiEZ2-P2.switch.ch [2001:620:0:c0c3::2]
  8     8 ms     8 ms     8 ms  swiIX2-P1.switch.ch [2001:620:0:c00a::2]
  9     8 ms     8 ms     8 ms  swissix.google.com [2001:7f8:24::4a]
 10    38 ms    34 ms    15 ms  2001:4860::1:0:4ca2
 11    14 ms    14 ms    17 ms  2001:4860::8:0:5038
 12    17 ms    50 ms    17 ms  2001:4860::8:0:8f8e
 13    24 ms    24 ms    24 ms  2001:4860::8:0:6400
 14    25 ms    25 ms    25 ms  2001:4860::1:0:6e0f
 15    25 ms    24 ms    25 ms  2001:4860:0:1::4b
 16    25 ms    25 ms    25 ms  ber01s08-in-x12.1e100.net [2a00:1450:4008:800::1012]
```

Not sure whether this is the exact path to the destination but is serves as a very good approximation

Revisit IP Header: other fields

Type of service / Traffic Class

- Differentiated Services (6bits) ≈ priority field e.g. voice over IP; used *only in* networks under a *single administration entity*
- Explicit Congestion Notification (2bits) see congestion control

Total length / Payload length

- in bytes including header
- ≤ 64 Kbytes; limited in practice by link-level MTU (Maximum Transmission Unit);
  e.g. in ethernet MTU = 1500 bytes

Protocol/Next Header = identifier of higher-layer protocol

- 6 = TCP, 17 = UDP
- 1 = ICMP for IPv4, 58 = ICMP for IPv6
- 4 = IPv4; 41 = IPv6 (encapsulation = tunnels)
- 50 = ESP (encrypted payload)
  51 = AH (authentication header)

Checksum

- IPv4 only, protects header against bit errors
- *absent in IPv6* (layer 2 and router hardware assumed to have efficient error detection)

A host generates a packet with Hop Limit = 1

A. This packet is invalid
B. This packet will never be forwarded by a switch nor by a router
C. This packet will never be forwarded by a switch but may be forwarded by a router
D. This packet will never be forwarded by a router but may be forwarded by a switch
E. None of the above is true
F. I don't know

Go to web.speakup.info or download speakup app

Join room 46045

Solution

Answer D

This packet cannot be forwarded by a router because it would decrement the HL and obtain 0. It can be forwarded by a switch because a switch does not examine the IP header.

Note that such a packet is perfectly valid. Sources put HL=1 when they want to be sure that the packet remains in the LAN.

7. MAC Address Resolution (IP address → MAC address)

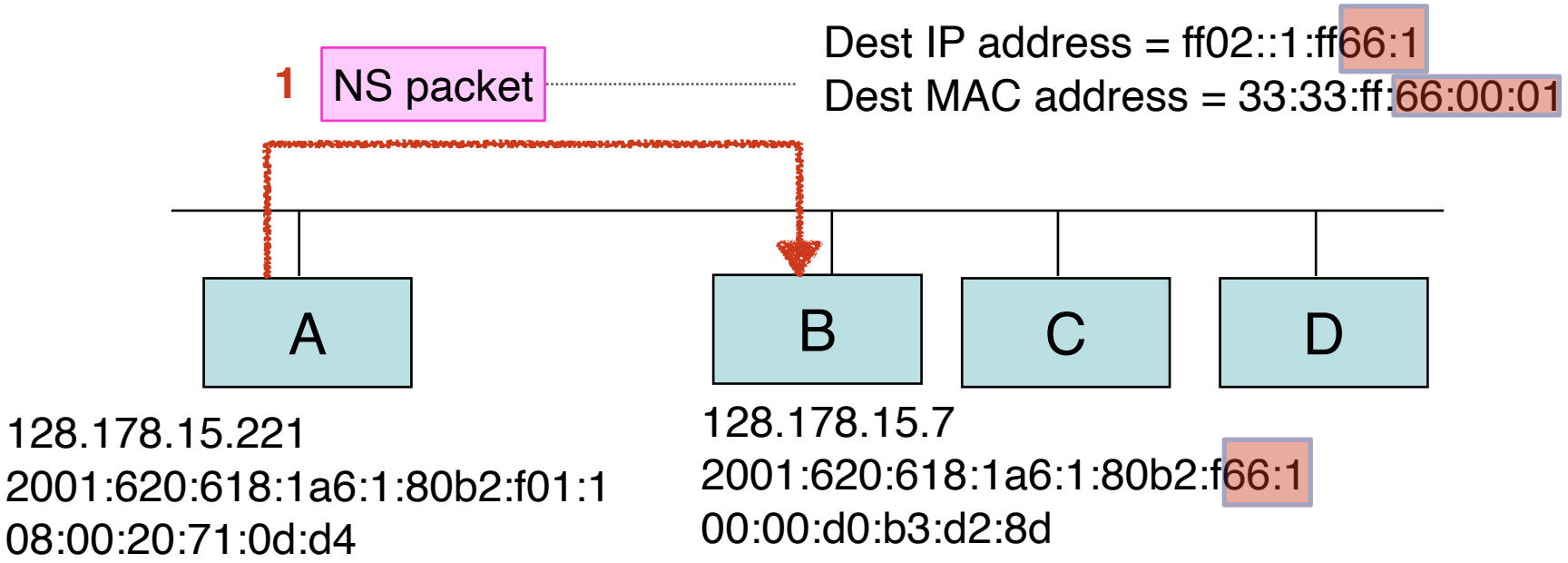
Why ?

- Say A has a packet to send to a next hop B inside the same subnet, either final destination or default gateway router;
- A knows only B's IP address, and must find B's MAC address, as within a subnet we use the MAC layer to move data

How ?

- A *broadcasts* (in IPv4) or *multicasts* (in IPv6) a packet to the LAN asking: "*who has the specific IP address of B*?"
- All hosts listening to that IP address (in principle only B) respond back with their MAC address

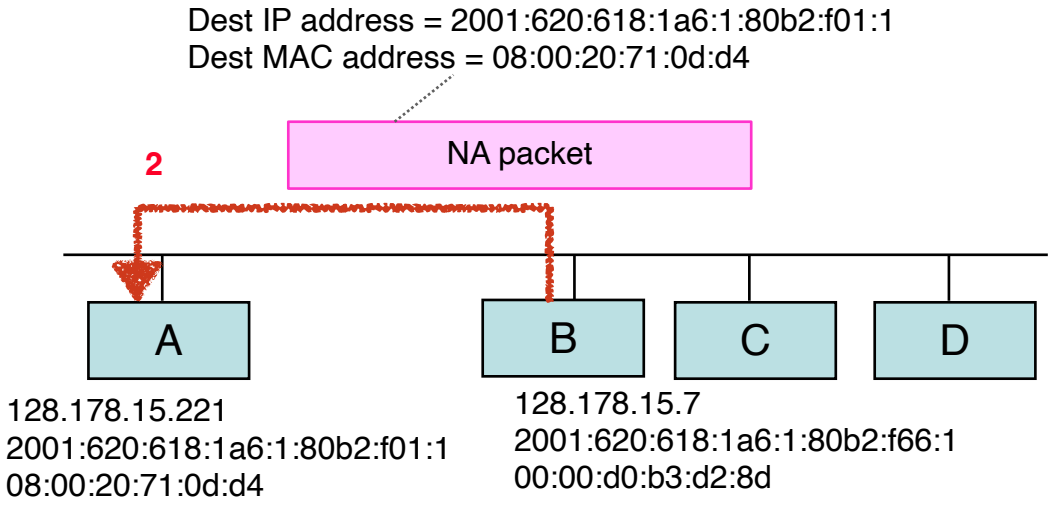
MAC address resolution in IPv6: Neighbor Discovery Protocol—NDP



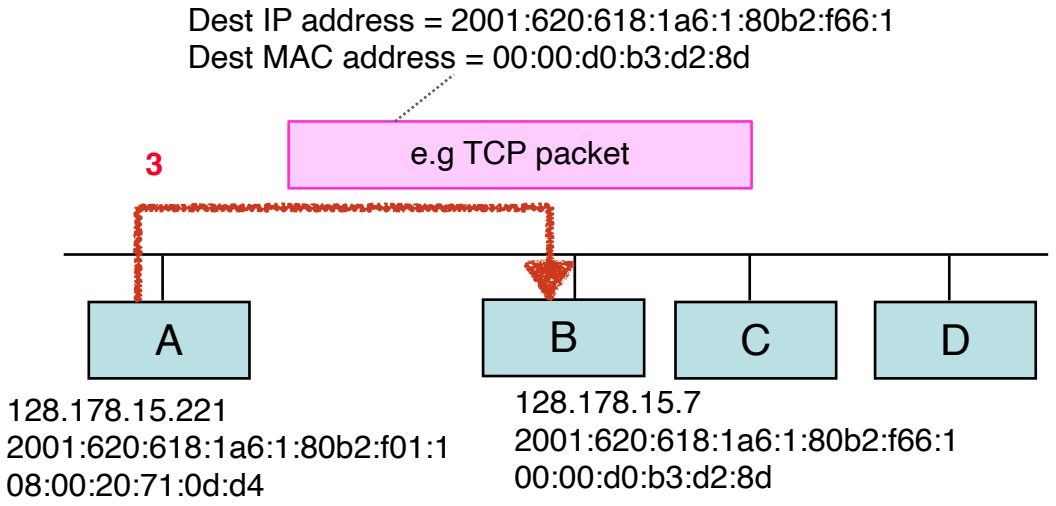
Steps:

i. A sends a *Neighbor Solicitation (NS)* packet with the question: "who has IP address B?"

- NS's dest. IP address = a *special multicast address* (Solicited Node Multicast Address)
  => *last 24 bits* are copied by B's IP address
- NS's dest. MAC address is derived from the multicast address in a similar way



ii. B responds back to A with a *Neighbor Advertisement (NA)* packet, giving its MAC address



iii. A reads B's NA, stores MAC address in its *neighbor cache* (a.k.a. *ARP table*) and can send data to B

- Cache entries expire after some configurable time of inactivity
- An entry's validity may be extended if positive feedback is received from higher-layer protocols

NA, NS packets are carried as *ICMPv6* packets
—> encapsulated inside IPv6 packets with next-header field = 58 (0x3a)

The Solicited Node Multicast Address in detail

- Obtained by *adding last 24 bits* of target IP address to the following *prefix*: ff02::1:ff00:0/104
- A packet with such a destination address is received by all nodes *all* hosts whose IP address has the *same* last 24 bits (all of them listen/"subscribe" to this multicast address)
- Used *only* in IPv6 by NDP or other protocols; IPv4 uses broadcast instead [see slides on ARP]

Target address	Compressed	2001:620:618:1a6:1:80b2:f66:1
	Uncompressed	2001:0620:0618:01a6:0001:80b2:0f66:0001
Solicited Node multicast address	Uncompressed	ff02:0000:0000:0000:0000:0001:ff66:0001
	Compressed	ff02::1:ff66:1

Look into an NDP Neighbor Solicitation Packet

```
ETHER:  Packet size =  86 bytes
ETHER:  Destination = 33:33:ff:01:00:01
ETHER:  Source = 3c:07:54:3e:ab:f2
ETHER:  Ethertype = 0x86dd
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 6
IP:   Traffic class =0x00000000
IP:      ....  0000 00.. .... .... .... .... .... = Default Differentiated Service Field
IP:      ....    ....     ..0.  .... .... .... .... = No ECN-Capable Transport (ECT)
IP:      ....    ....     ...0  .... .... .... .... = No ECN-CE
IP:   .... .... ....  0000 0000 0000  0000 0000 =  Flowlabel: 0x00000000
IP:   Payload length = 32
IP:   NextHeader= 58
IP:   Hop limit= 255
IP:   Source address = 2001:620:618:197:1:80b2:97c0:1
IP:   Destination address = ff02::1:ff01:1
IP:
ICMPv6:    ----- ICMPv6 Header -----
ICMPv6:
ICMPv6:  Type = 135
ICMPv6:  Code=0
ICMPv6:  Checksum = 0xb199 [correct]
ICMPv6:  Reserved = 00000000
ICMPv6:  Target Address=2001:620:618:197:1:80b2:9701:1
ICMPv6:
```

Reuses the last 32 bits of the destination IP address
[see also multicast and MAC lecture]

Neighbor Solicitation (~ARP Request)

Solicited Node Multicast Address corresponding to this IPv6 target address (24 last bits are the same)

MAC Address Resolution with IPv4

Similar to NDP, except:

- Terminology:
  - the protocol is called ARP (Address Resolution Protocol)
  - NS/NA pkts of NDP are called ARP Request /ARP reply
- Protocol type:
  - ARP packets are not ICMP packets (not even encapsulated in IP packets)
  - In MAC-frame headers we use Ethertype = ARP (86dd)
- Reachability:
  - ARP request is *broadcast* to all nodes in LAN (instead of multicast)

Look into an ARP request

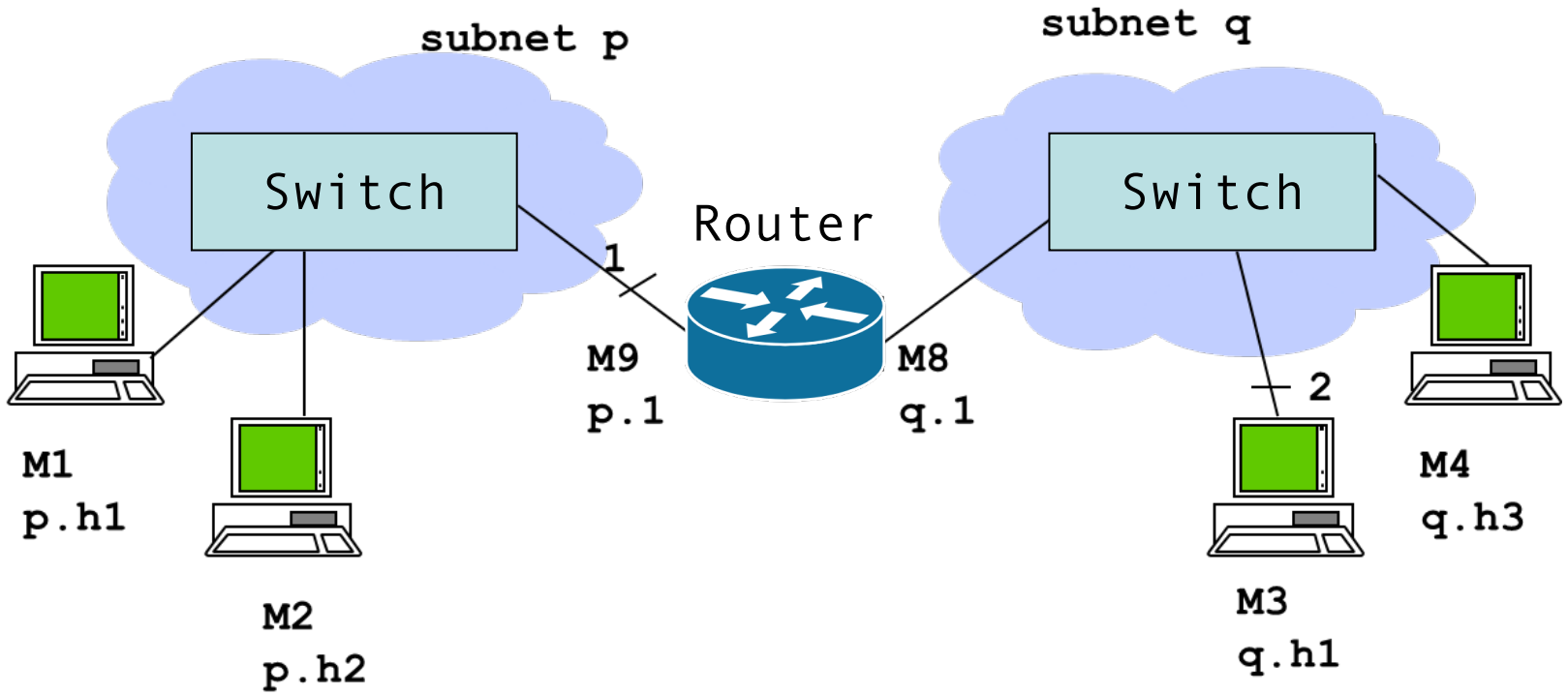
```
Ethernet II
    Destination: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source: 00:03:93:a3:83:3a (Apple_a3:83:3a)
    Type: ARP (0x0806)
    Trailer: 000000000000000000000000000000000...
Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: 00:03:93:a3:83:3a (Apple_a3:83:3a)
    Sender IP address: 129.88.38.135 (129.88.38.135)
    Target MAC address: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Target IP address: 129.88.38.254 (129.88.38.254)
```

LAN broadcast

Next-header protocol
= ARP (not IP)

This indicates which protocol has asked for MAC address resolution (IP implies IPv4)

M1 just boots up and sends a packet to M3 (whose IP address it happens to know). What happens next?



A. M1 sends an NS /ARP packet for q.h1
B. M1 sends an NS /ARP packet for p.1
C. None of the above
D. I do not know

Go to web.speakup.info or download speakup app

Join room 46045

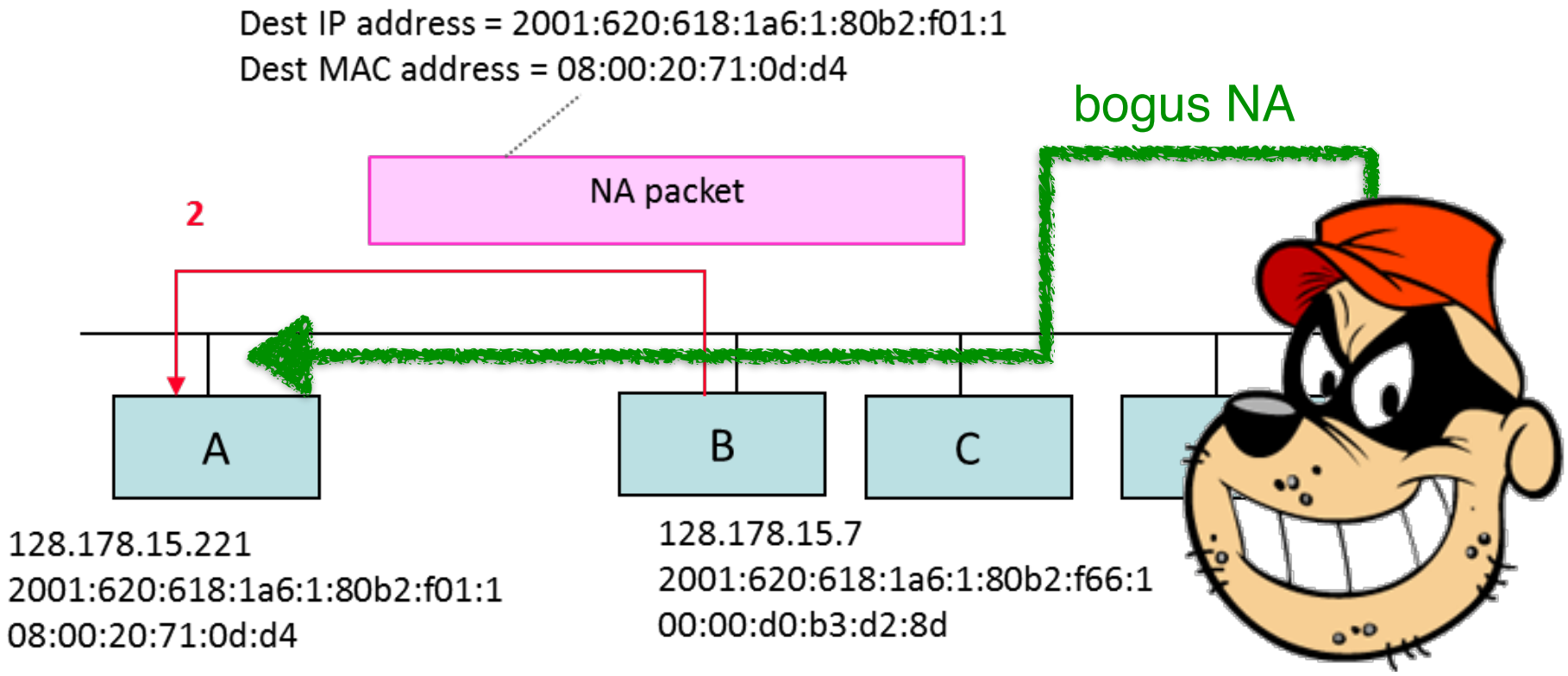
Solution

Answer B: Since M3 is not in the same subnet, M1 needs to find the MAC address of its default gateway router, namely p.1.

Note that the IP address of the default router such as p.1. is in M1's configuration, but not the MAC address of the default router.

Security Issues with ARP/ NDP

- *ARP spoofing*: ARP replies and NAs can be *forged*
  - attacker can intercepts packets destined to B (e.g. launch a man-in-the-middle attack),
    - easier in IPv4, where ARP requests are broadcast
  - sends a bogus NA/ARP reply with its own MAC address
  - may cause A to map their MAC address to B's IP address, if the bogus NA arrives *faster* at A



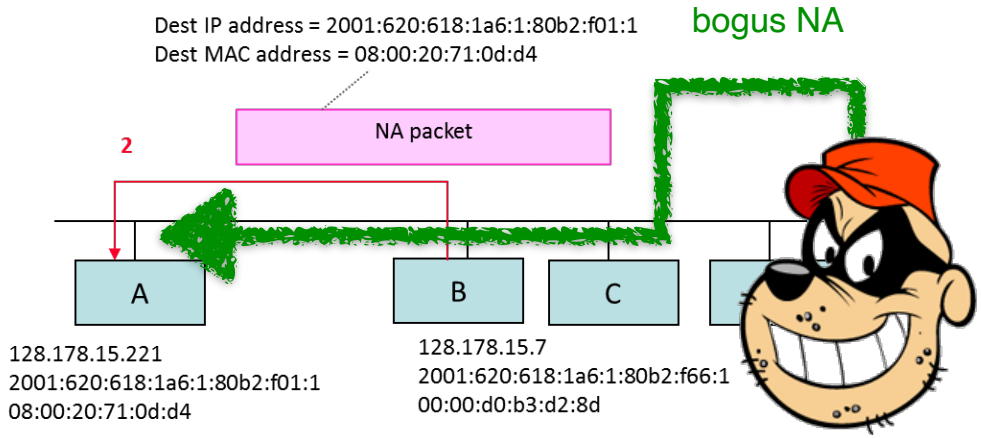
How to prevent ARP spoofing in LANs?

- *DHCP snooping*: switch/WiFi base station observes all DHCP traffic and remembers mappings IP addr <—> MAC addresses
  [recall DHCP is used to automatically configure the IP address at system startup]

- *Dynamic ARP inspection*: switch filters all ARP (or NDP) traffic and allows only valid answers – removes invalid broadcasts (IPv4) and multicasts (IPv6)

- Such solutions are deployed in enterprise networks,
  rarely at home or WiFi access points

Secure NDP (SEND)



What ?

Makes NDP spoofing impossible by using cryptographically generated addresses (CGAs).

How ? via Asymmetric (public-key) cryptography:

- B has an RSA private+public key pair {*p*,*P*}
- B uses a CGA whose `host suffix ≈ hash(P, IPv6 prefix, other CGA-parameters such as counters)`
  - ➡ this binds the B's IP address to *P* (i.e. the CGA can be verified if *P* and *CGA parameters* are known)
- B *includes* *P* + all CGA-parameters into its NA and *digitally signs* it with its own private key *p*
- A can verify the CGA using *P* + CGA parameters sent by B
- A can also verify the digital signature using *P*
  - ➡ this proves that the NA was sent indeed by someone knowing private key *p*
- NAs may also include nonces to avoid replay attacks
- Hence, anyone can pretend to have B's IP address, but only the owner of *p* (i.e. B) can issue a *valid/verifiable* NA

Solves the problem but:

requires a strong hash function (SHA-1 currently used)

may also need access to a trusted certification authority (e.g. RPKI)

→ not widespread yet

Public key cryptography in a nutshell (for those who have not heard about it)

A private/public key system (such as RSA or ECDSA) has two keys: a *public* *P* and a *private* *p*

A message can be encrypted with *p* and decrypted with *P* (and vice versa)

A *digital signature* of a message *m* is essentially `p(m)`

- ➡ can be computed only by the owner of *p*
- ➡ can be verified by whoever has *P*
- ➡ guarantees the *authenticity* of the sender of the message

Conclusion

- IP is built on two principles/rules:
  - Use a structured IP address per interface and longest prefix match; this compresses routing tables by aggregation
  - Inside subnet, don't use routers
- IPv4 and IPv6 are not compatible – interworking requires dual-stack devices
- NATs came as an after-thought and are widely deployed, primarily with IPv4, but sometimes also with IPv6
- DHCP configures IP address, network mask, DNS server's IP address, and the IP address of the gateway router to a host; SLAAC automatically assigns IPv6 addresses without DHCP, but does not provide DNS info
- TTL/HL limits the number of hops of an IP packet
- ARP/NDP provides the MAC address corresponding to an IP address; it is not secure but can be secured with public-key cryptography and CGAs