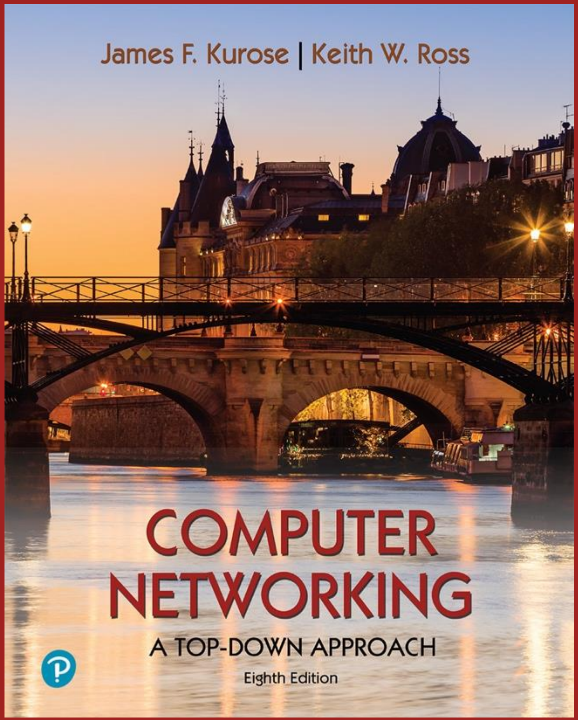
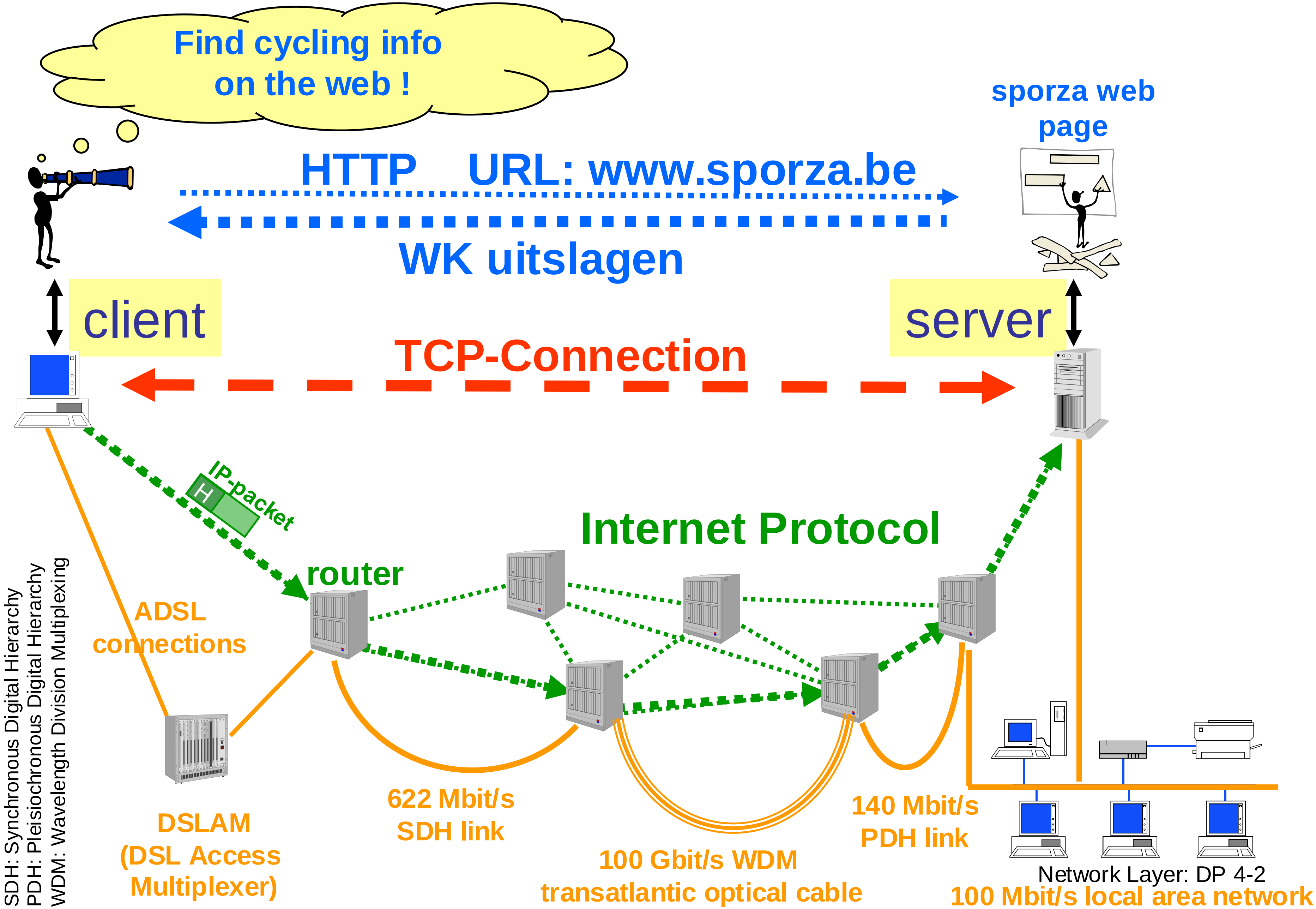
Communication Networks
W. Tavernier

Chapter 4
Network Layer : Data Plane



Computer Networking: A Top-Down Approach
8th Edition, 2020, Pearson,
James F. Kurose, Keith W. Ross

IP in the overall picture



Chapter 4: outline

4.1 Overview of Network layer

- **data plane**
- **control plane**

4.2 What's inside a router

4.3 IP: Internet Protocol

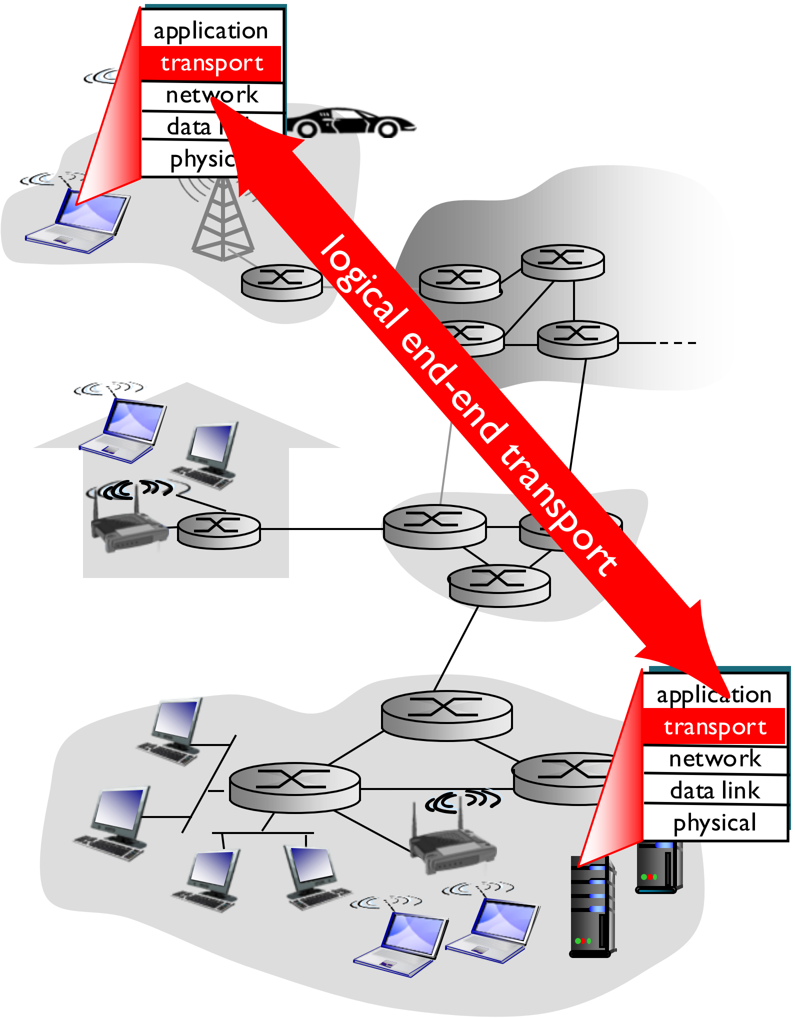
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

Transport Layer Service Recap

- provide logical end-to-end communication between app processes running on different hosts

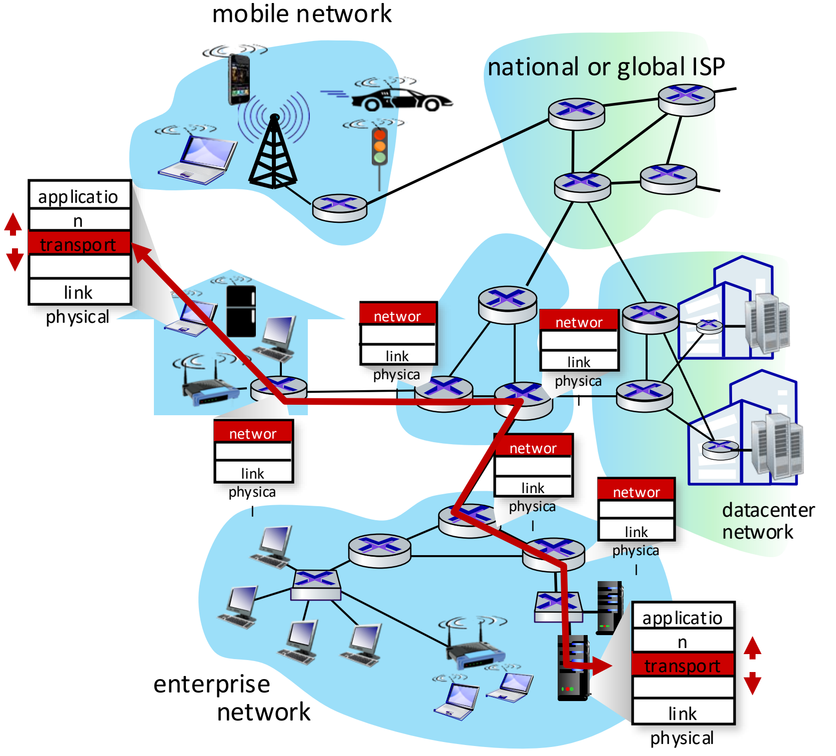


- transport protocols only run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer

Q: How do segments reach the destination end-host?

Network-layer services and protocols

- transport segment from sending to receiving host
  - sender: encapsulates segments into datagrams, passes to link layer
  - receiver: delivers segments to transport layer protocol
- network layer protocols in *every Internet device*: hosts, routers
- routers:
  - examines header fields in all IP datagrams passing through it
  - moves datagrams from input ports to output ports to transfer datagrams along end-end path



Two key network-layer functions

network-layer functions:

- *<u>forwarding</u>:* move packets from a router's input link to appropriate router output link
- *<u>routing:</u>* determine route taken by packets from source to destination
  - *routing algorithms*

analogy: taking a trip

- *<u>forwarding</u>:* process of getting through single interchange
- *<u>routing</u>:* process of planning trip from source to destination

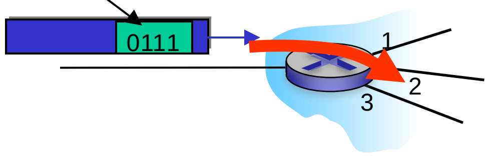
forwarding

routing

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

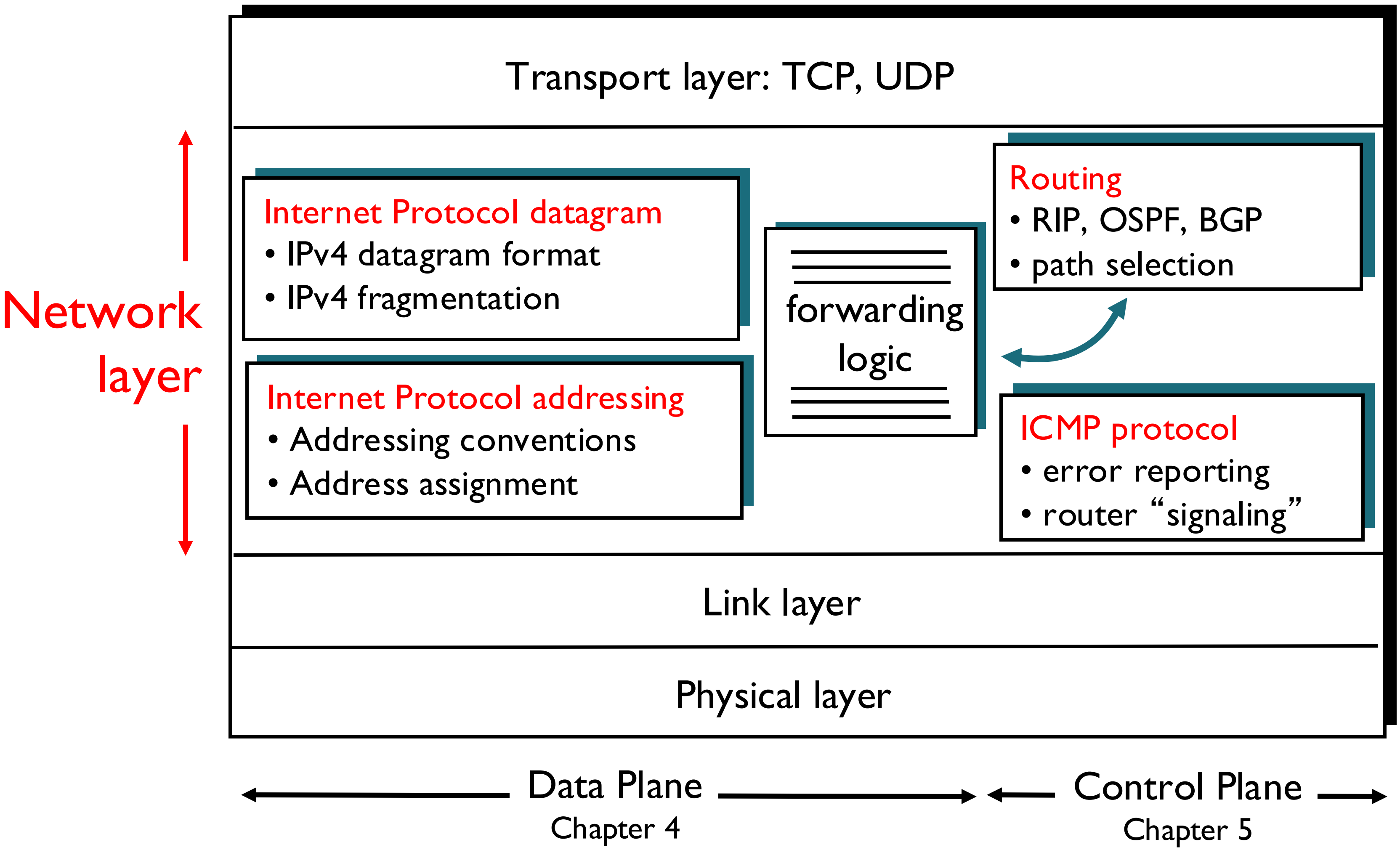


Control plane:

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host

Internet Protocol (IP) Functionality

Host & router network layer functions:



Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

example services for *individual* datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a flow of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Internet "best effort" service model

No guarantees on*:*

i. successful datagram delivery to destination
ii. timing or order of delivery
iii. bandwidth available to end-end flow

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients’ networks, allow services to be provided from multiple locations
- congestion control of “elastic” services helps

It’s hard to argue with success of best-effort service model

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

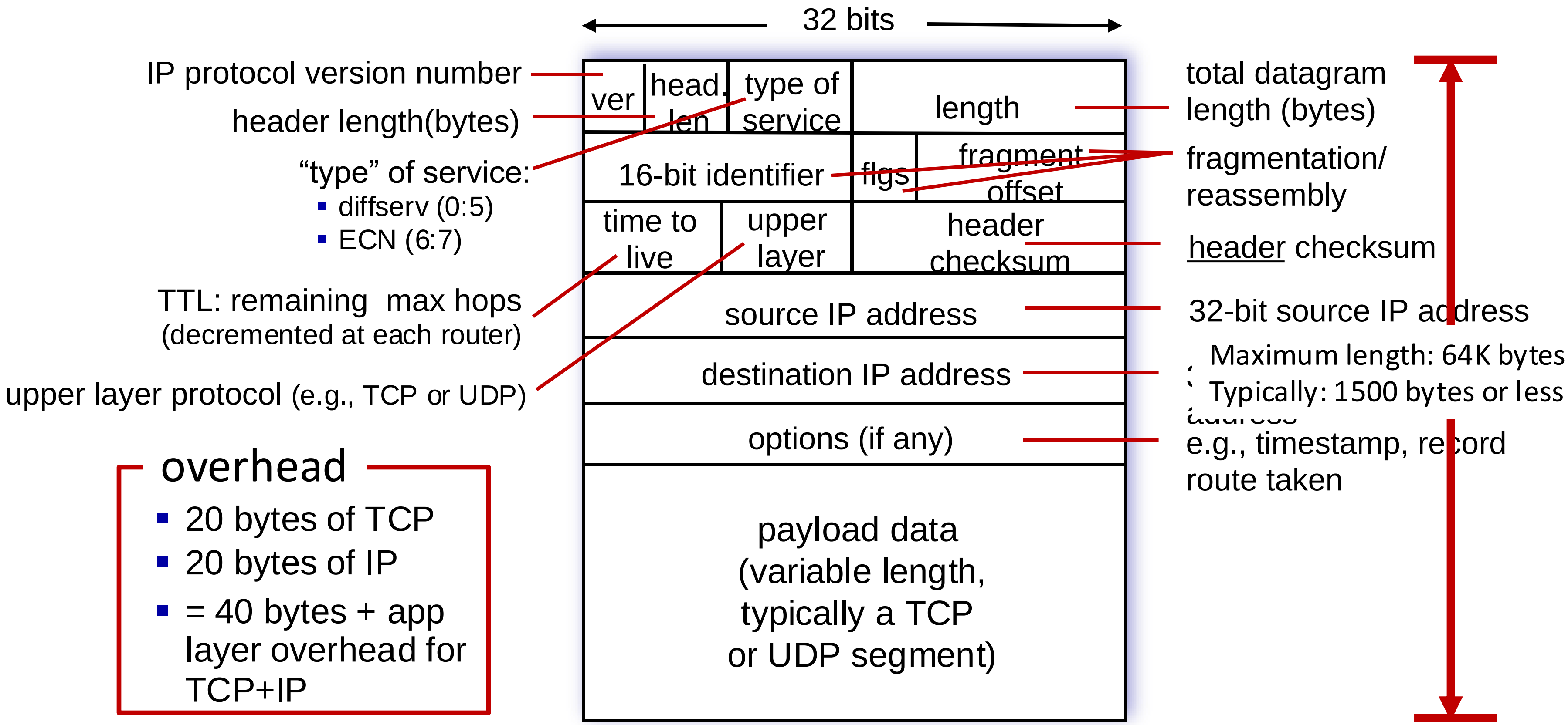
4.3 IP: Internet Protocol

- **datagram format**
- **fragmentation**
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

IP Datagram format



overhead

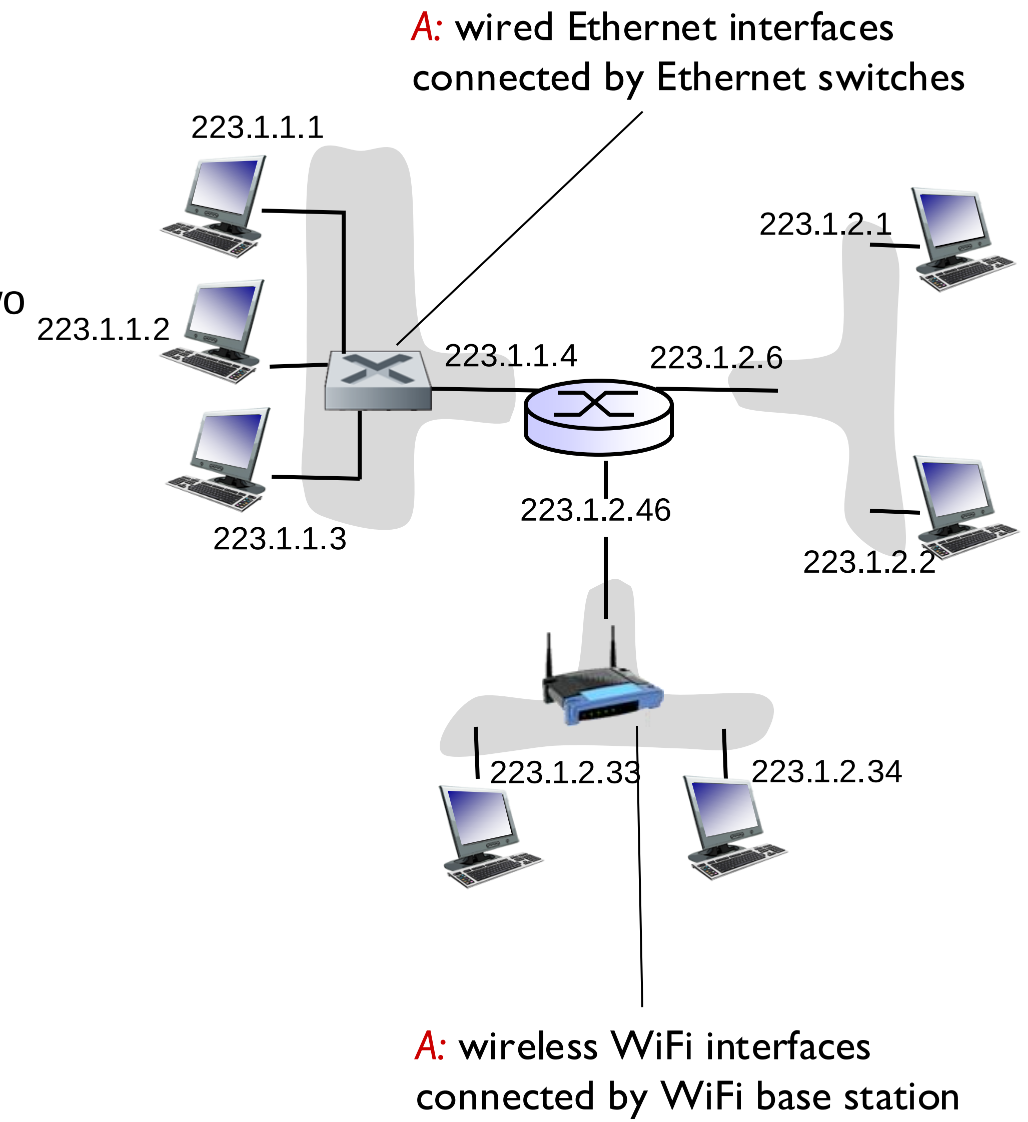
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

IPv4 addressing: interfaces

- interface: connection/port between host/router and physical link
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
  - router's typically have multiple interfaces
- Q: how are interfaces actually connected?

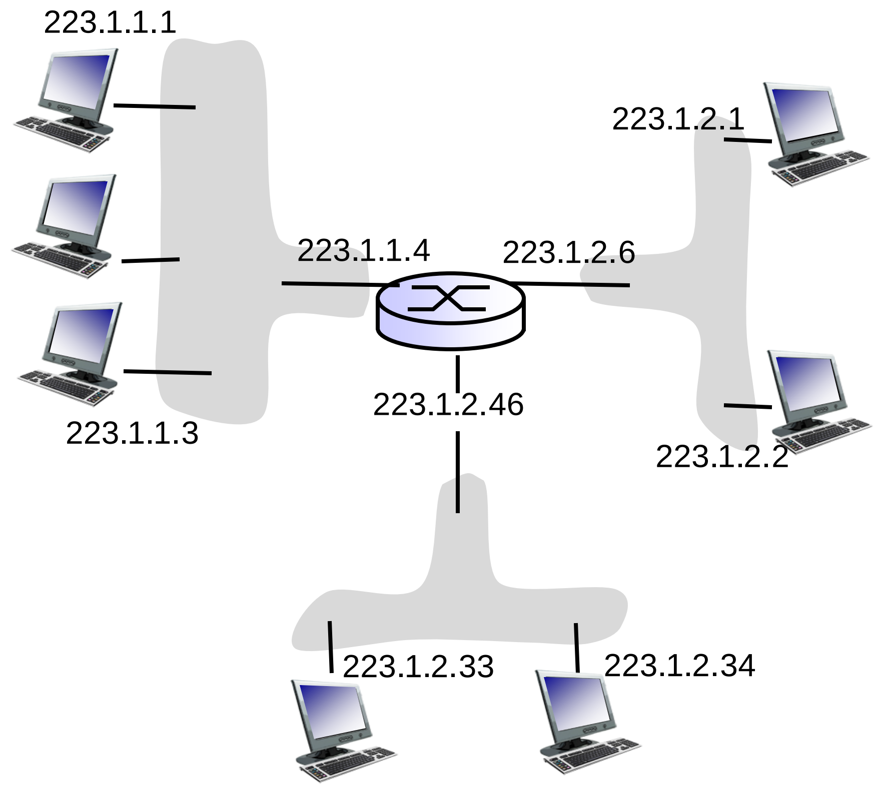
  We'll learn more about that in chapter 5, 6.

  For now: don't need to worry about how one interface is connected to another (with no intervening router)



IPv4 addressing - address format

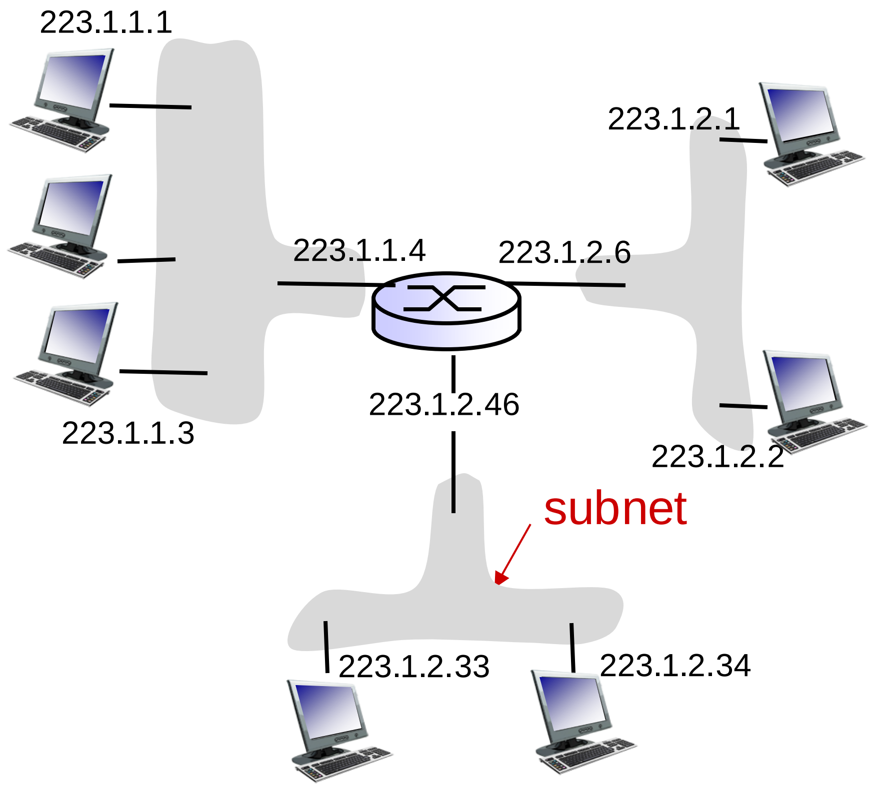
- Each interface (≠ host) receives an IP address
- IPv4 address: 32-bit identifier per host, router interface
  - 4 octets, decimal notation, separation by dot
  - 2^{32} possible addresses
- IP addresses have structure:
  - *subnet part:* devices *in same subnet* have common high order bits
  - *host part:* remaining low order bits



$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IPv4 addressing - subnets

- interfaces are grouped into subnets (networks)
- subnet: set of device interfaces that can physically reach each other without intervening router (locally connected ~ LAN)
- recipe: to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
  - each isolated network is called a subnet

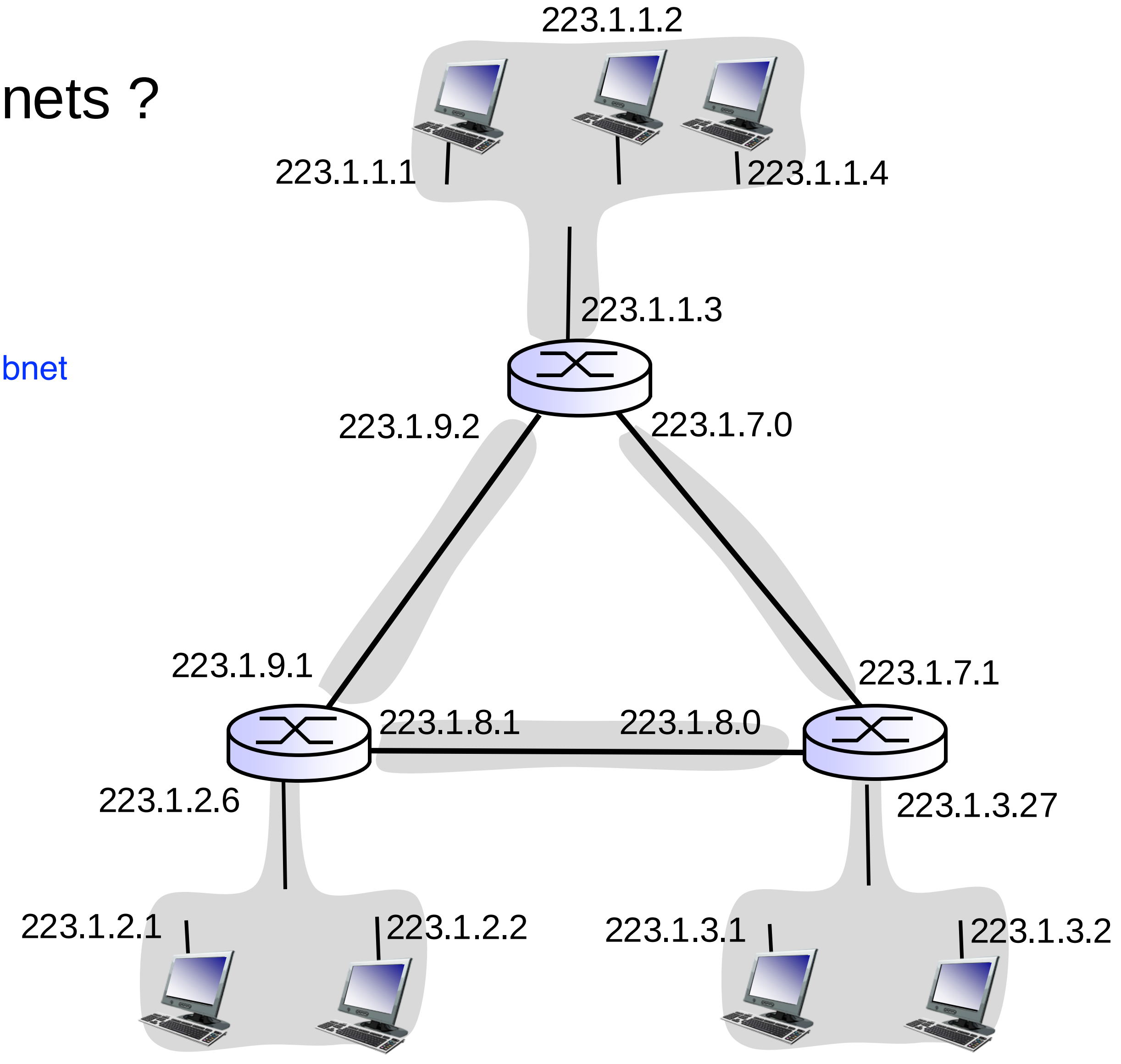


network consisting of 3 subnets

IPv4 addressing - subnets

- Q: how many subnets ?

6 xxx.x.y…
iedere y is een subnet



IP addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced "cider")

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address

subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IP address of interfaces in the same subnet

REQUIREMENT

- **Interfaces in the same subnet must have an IP address in the same address block**
- **An address block is a set of consecutive IP addresses with a common prefix (/x)**

Subnetwork : 223.1.1.0/24

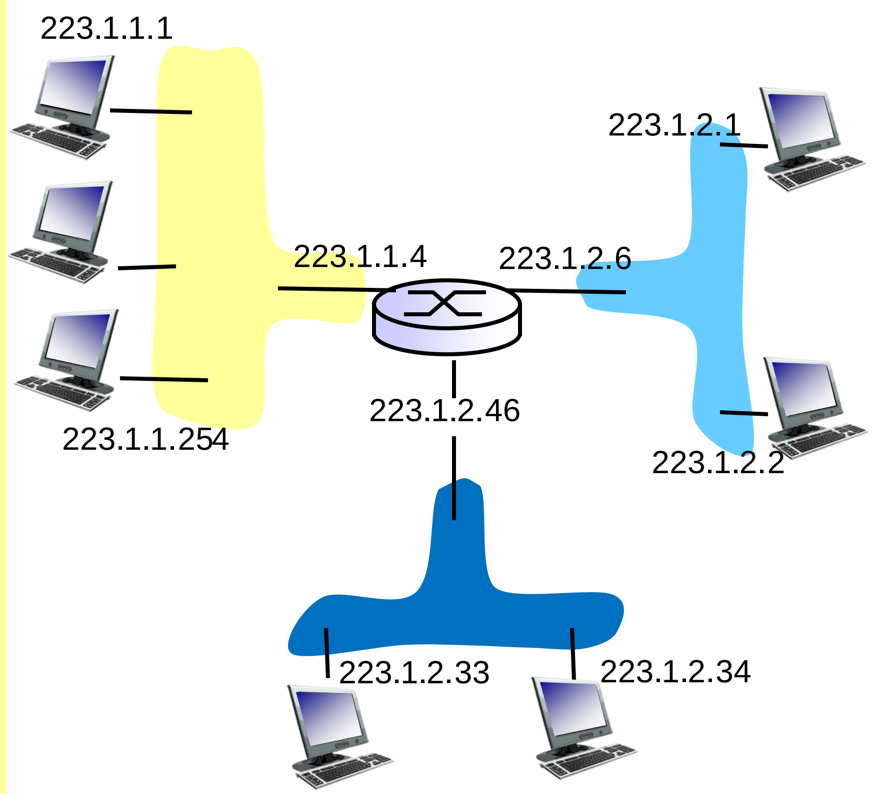
223 . 1 . 1 . 1 / 24
11011111 00000001 00000001 00000001

223 . 1 . 1 . 4 / 24
11011111 00000001 00000001 00000100

223 . 1 . 1 . 254 / 24
11011111 00000001 00000001 11111110

24 bit common prefix | 8 bit host part

First address = *network address* = 223.1.1.0/24
Last address = *broadcast address* = 223.1.1.255
host range : 223.1.1.1 - 223.1.1.254



IP address of interfaces in the same subnet

REQUIREMENT

- **Interfaces in the same subnet must have an IP address in the same address block**
- **An address block is a set of consecutive IP addresses with a common prefix**

Subnetwork : 223.1.2.0/29

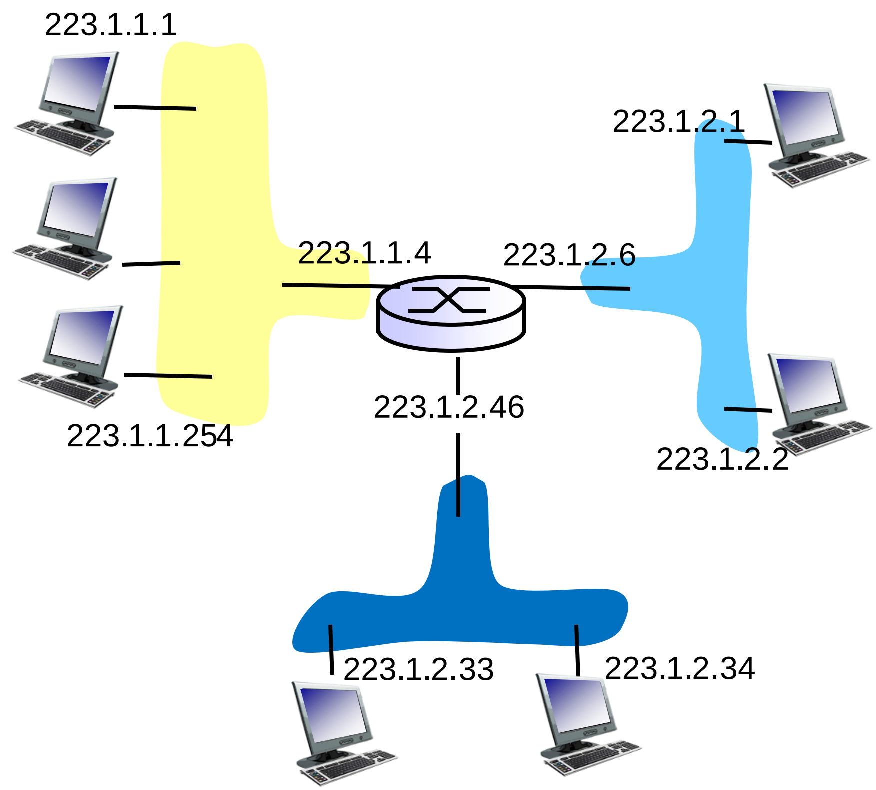
223 . 1 . 2 . 1 / 29
11011111 00000001 00000010 00000001

223 . 1 . 2 . 2 / 29
11011111 00000001 00000010 00000010

223 . 1 . 2 . 6 / 29
11011111 00000001 00000010 00000110

29 bit common prefix — 3 bit host part

First address = *network address* = 223.1.2.0
Last address = *broadcast address* = 223.1.2.7
host range : 223.1.2.1 – 223.1.2.6



IP address of interfaces in the same subnet

REQUIREMENT

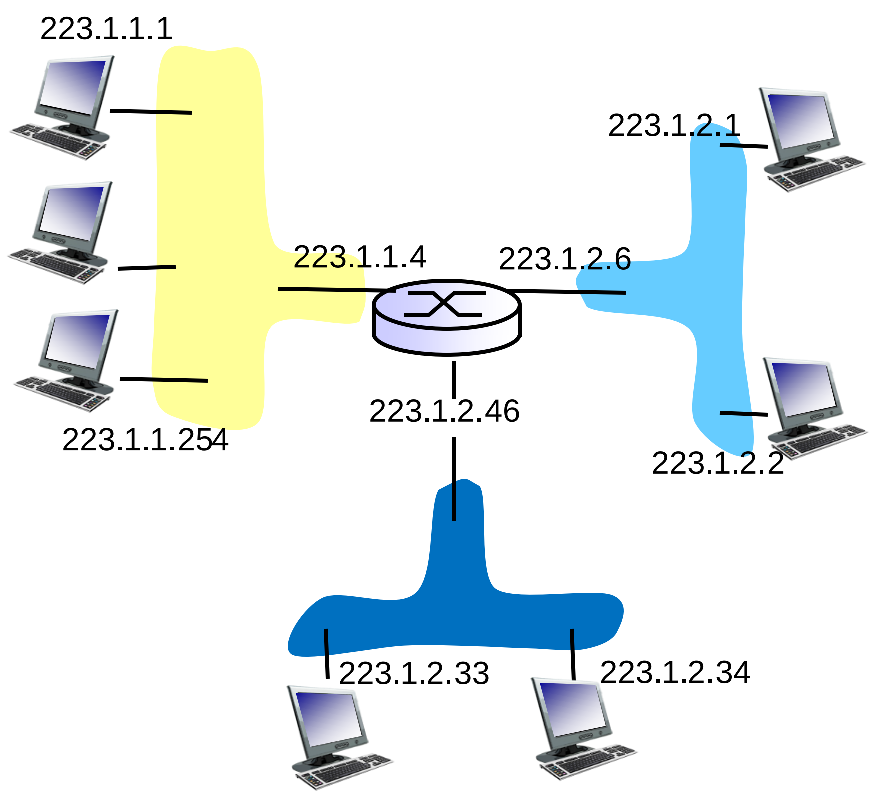
- **Interfaces in the same subnet must have an IP address in the same address block**
- **An address block is a set of consecutive IP addresses with a common prefix**

Subnetwork : 223.1.2.32/28

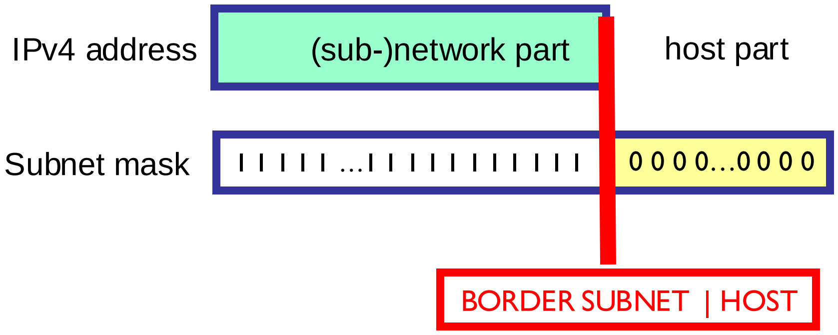
223 .	*1* .	*2* .	*33*	*/ 28*
11011111	00000001	00000010	00100001	
223 .	*1* .	*2* .	*34*	*/ 28*
11011111	00000001	00000010	00100010	
223 .	*1* .	*2* .	*44*	*/ 28*
11011111	00000001	00000010	00101100	

28 bit common prefix | 4 bit host part

First address = network address = 223.1.2.32
Last address = broadcast address = 223.1.2.47
host range : 223.1.2.33 – 223.1.2.46



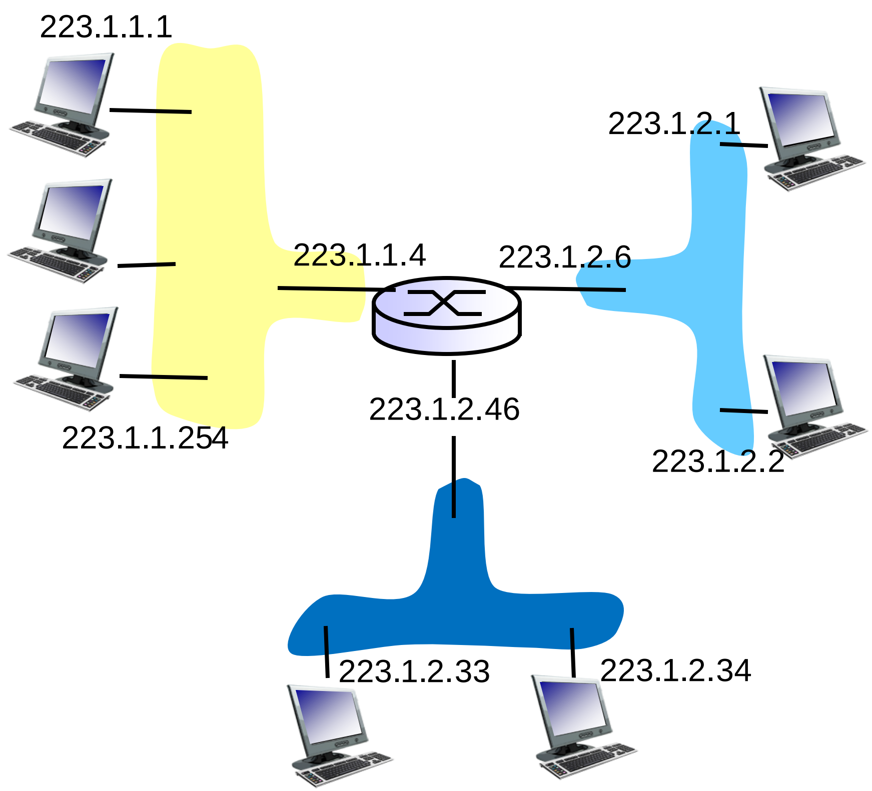
IPv4 addressing



Subnetwork : 223.1.1.0/24
(sub-)network address : 223.1.1.0 (24 bits)
mask used : 255.255.255.0
hosts : 254 (0 and 255 not allowed)
host range : 223.1.1.1 - 223.1.1.254

Subnetwork : 223.1.2.0/29
(sub-)network address : 223.1.2.0 (29 bits)
mask used : 255.255.1111 1000
hosts : 6 (000 and 111 not allowed)
host range : 223.1.2.1 – 223.1.2.6

Subnetwork : 223.1.2.32/28
(sub-)network address : 223.1.2.32 (28 bits)
mask used : 255.255.255.1111 0000
hosts : 14 (0000 and 1111 not allowed)
host range : 223.1.2.33 – 223.1.2.46



General advice/guideline:
minimize wastage of IP address space
=
Use smallest address block (largest prefix) possible

Note : binary, decimal and hexadecimal notation used where appropriate

IPv4 addressing - Special Addresses

0.X.Y.Z/8 : this host on this network (used for booting)
only allowed as source address

127.X.Y.Z : loopback Interface (for debugging)
in practice mainly 127.0.0.1 address used

169.254.0.0/16: link-local addressing (only valid on link, not routable)
only allowed as destination address, no forwarding allowed

10.0.0.0-10.255.255.255;
172.16.0.0-172.31.255.255;
192.168.0.0-192.168.255.255 :
used for private networks (can be re-used = occur multiple times)
examples: home networks, enterprise LAN

Routing protocols such as Routing Information Protocol (RIP) use broadcasts to send out "advertisements." This advertisement is used by routers to map out the topology of a network, so that data can be routed to the appropriate place accordingly.

Cfr. RFC 6890

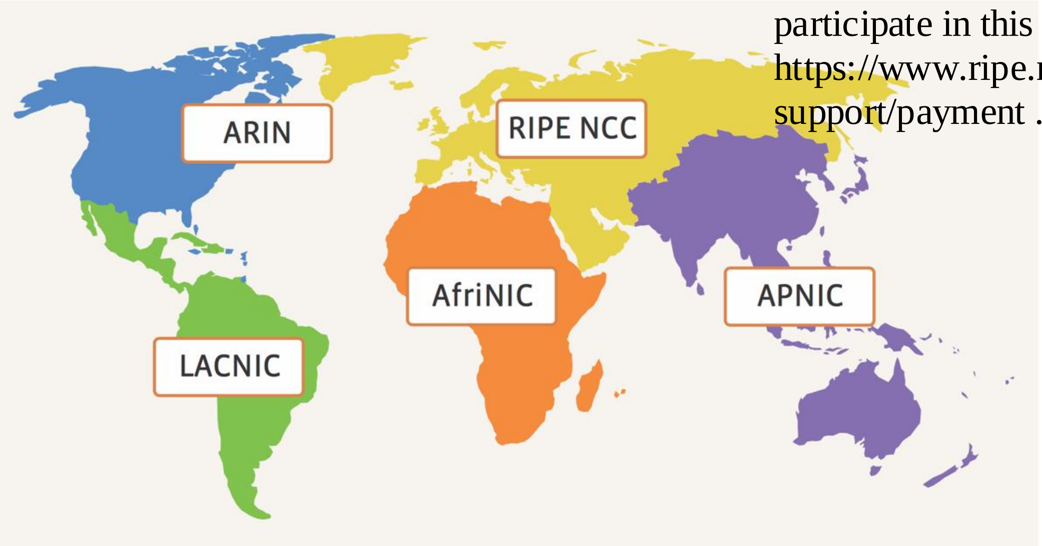
Address block assignment – Network

Q: How to get a block of addresses?

A: Internet Assigned Numbers Authority (IANA) of Internet Corporation for Assigned Names and Numbers (ICANN)

- allocates addresses, delegating to Regional Internet Registries (RIR)
- manages DNS
- assigns domain names, resolves disputes

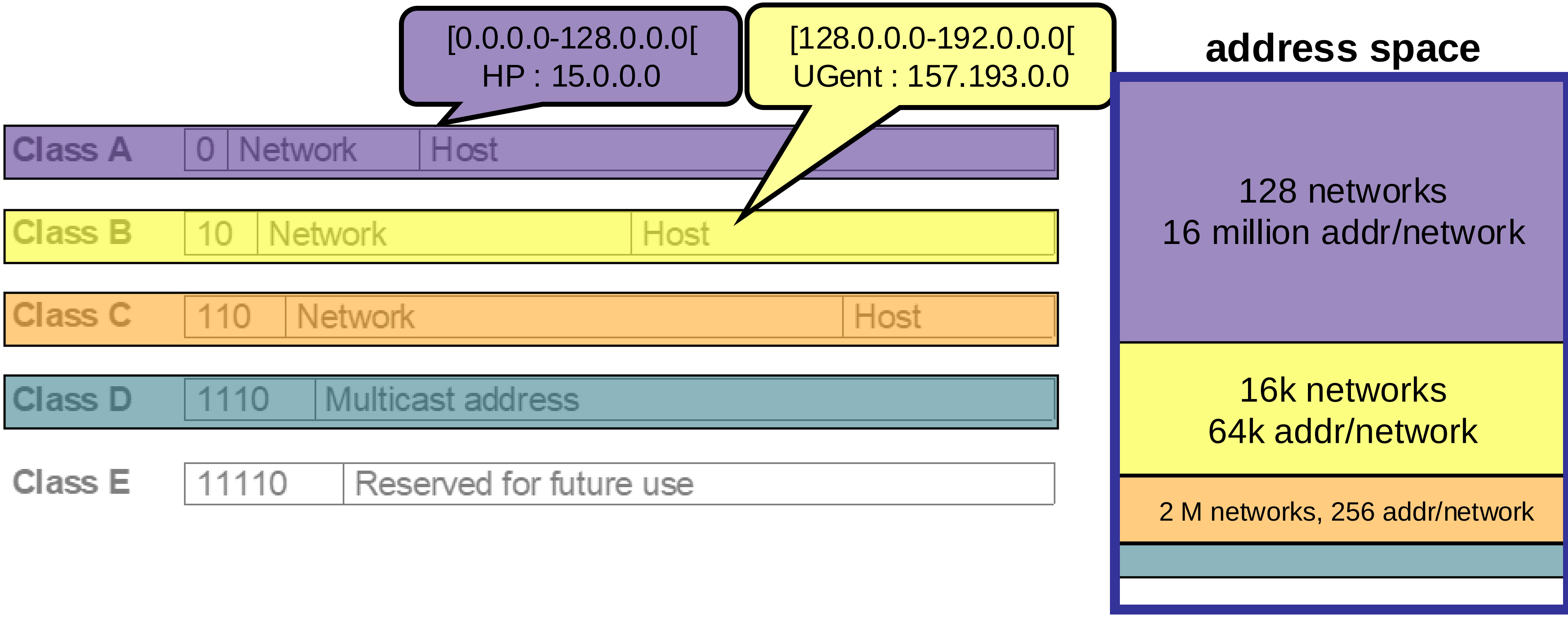
Note that IP addresses are not for sale. They are considered as global resource that is shared by the public, but are allocated by responsible Regional Internet Registratries, which usually charge fees to participate in this process. See https://www.ripe.net/participate/member-support/payment .



www.icann.org

IPv4 addressing – Classful Addressing

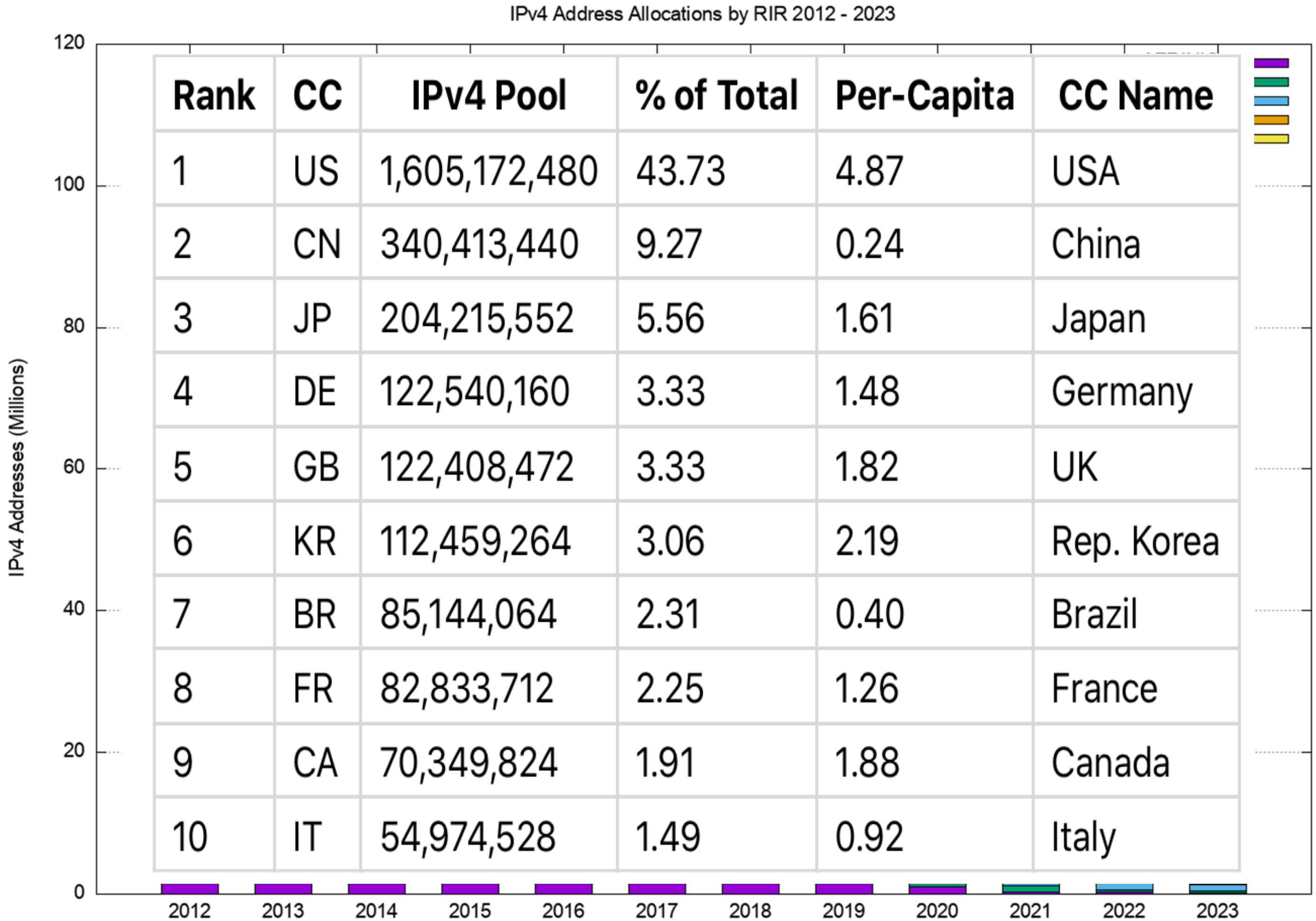
- Before the ’90s, network address ranges were constrained to /8, /16 or /24 bits (class A, B and C networks)
  - Coarse granularity with networks of 16M, 64K or 256 addresses per network
  - Waste of address ranges => classful addressing not used anymore



Since ‘90s any type of address block can be used:
Classless Interdomain Routing (CIDR)

https://blog.apnic.net/2024/01/17/ip-addresses-through-2023/
https://ipv4.potaroo.net

IPv4 Address Block Allocation Evolution



Rank	CC	IPv4 Pool	% of Total	Per-Capita	CC Name
1	US	1,605,172,480	43.73	4.87	USA
2	CN	340,413,440	9.27	0.24	China
3	JP	204,215,552	5.56	1.61	Japan
4	DE	122,540,160	3.33	1.48	Germany
5	GB	122,408,472	3.33	1.82	UK
6	KR	112,459,264	3.06	2.19	Rep. Korea
7	BR	85,144,064	2.31	0.40	Brazil
8	FR	82,833,712	2.25	1.26	France
9	CA	70,349,824	1.91	1.88	Canada
10	IT	54,974,528	1.49	0.92	Italy

- Move towards IPv6 delayed because of:
  - (CG) NAT
  - Many client/server applications

Source: APNIC

Address assignment – Network

Q: How are IP address blocks assigned to organizations?

A: Gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	…..	….	….
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

Or:	Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/24
	Organization 8	11001000 00010111 00011111	00000000	200.23.31.0/24

IPv4 addressing - /24 decomposition

/24 netwerk: de eerste 24 bits zijn vast
—> 255.255.255.0 : het netwerk de eerste 3 bytes hetzelfde blijven en het laatste byte varieert an 0 tot 255

subnetmask (CIDR)	/24		/25		/26		/27		/28		/29		/30	
subnetmask (DEC)	0		.128		.192		.224		.240		.248		.252	
subnetmask (HEX)	**.00**		**.80**		**.C0**		**.E0**		**.F0**		**.F8**		**.FC**	
	0	255	0	127	0	63	0	31	0	15	0	7	0	3
													4	7
											8	15	8	11
													12	15
									16	31	16	23	16	19
													20	23
											24	31	24	27
													28	31
							32	63	32	47	32	39	32	35
													36	39
											40	47	40	43
													44	47
									48	63	48	55	48	51
													52	55
											56	63	56	59
													60	63
					64	127	64	95	64	79	64	71	64	67
													68	71
											72	79	72	75
													76	79
									80	95	80	87	80	83
													84	87
											88	95	88	91
													92	95
							96	127	96	111	96	103	96	99
													100	103
											104	111	104	107
													108	111
									112	127	112	119	112	115
													116	119
											120	127	120	123
													124	127

		128	255	128	191	128	159	128	143	128	135	128	131
												132	135
										136	143	136	139
												140	143
								144	159	144	151	144	147
												148	151
										152	159	152	155
												156	159
						160	191	160	175	160	167	160	163
												164	167
										168	175	168	171
												172	175
								176	191	176	183	176	179
												180	183
										184	191	184	187
												188	191
				192	255	192	223	192	207	192	199	192	195
												196	199
										200	207	200	203
												204	207
								208	223	208	215	208	211
												212	215
										216	223	216	219
												220	223
						224	255	224	239	224	231	224	227
												228	231
										232	239	232	235
												236	239
								240	255	240	247	240	243
												244	247
										248	255	248	251
												252	255

/24 network example: 223.1.1.0/24

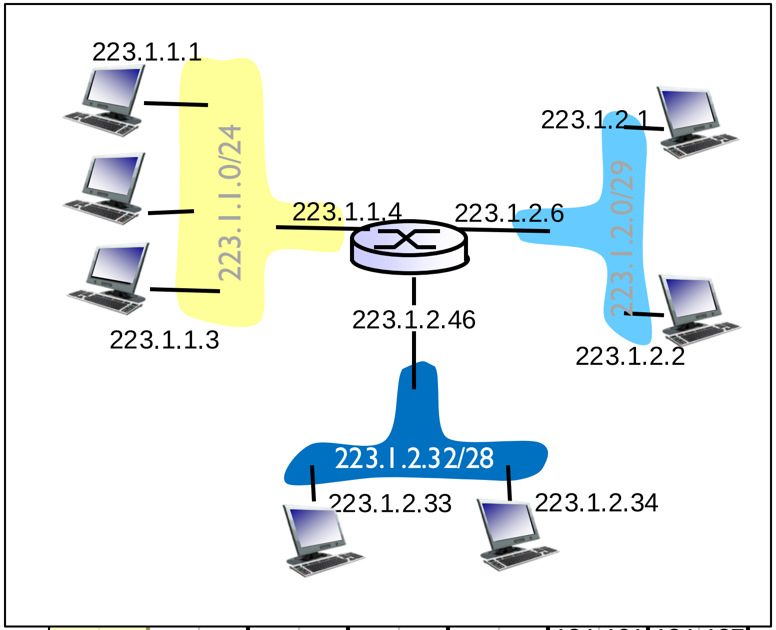
Decomposition table as reference

subnetmask (CIDR)	/24		/25		/26		/27		/28		/29		/30	
subnetmask (DEC)	0		.128		.192		.224		.240		.248		.252	
subnetmask (HEX)	**.00**		**.80**		**.C0**		**.E0**		**.F0**		**.F8**		**.FC**	
	0	255	0	127	0	63	0	31	0	15	0	7	0	3
													4	7
											8	15	8	11
													12	15
									16	31	16	23	16	19
													20	23
											24	31	24	27
													28	31
							32	63	32	47	32	39	32	35
													36	39
											40	47	40	43
													44	47
									48	63	48	55	48	51
													52	55
											56	63	56	59
													60	63
					64	127	64	95	64	79	64	71	64	67
													68	71
											72	79	72	75
													76	79
									80	95	80	87	80	83
													84	87
											88	95	88	91
													92	95
							96	127	96	111	96	103	96	99
													100	103
											104	111	104	107
													108	111
									112	127	112	119	112	115
													116	119
											120	127	120	123
													124	127

		128	255	128	191	128	159	128	143	128	135	128	131
												132	135
										136	143	136	139
												140	143
								144	159	144	151	144	147
												148	151
										152	159	152	155
												156	159
						160	191	160	175	160	167	160	163
												164	167
										168	175	168	171
												172	175
								176	191	176	183	176	179
												180	183
										184	191	184	187
												188	191
				192	255	192	223	192	207	192	199	192	195
												196	199
										200	207	200	203
												204	207
								208	223	208	215	208	211
												212	215
										216	223	216	219
												220	223
						224	255	224	239	224	231	224	227
												228	231
										232	239	232	235
												236	239
								240	255	240	247	240	243
												244	247
										248	255	248	251
												252	255

IPv4 addressing - /24 decomposition

subnetmask (CIDR)	/24		/25		/26		/27		/28		/29		/30	
subnetmask (DEC)	0		.128		.192		.224		.240		.248		.252	
subnetmask (HEX)	**.00**		**.80**		**.C0**		**.E0**		**.F0**		**.F8**		**.FC**	
	0	255	0	127	0	63	0	31	0	15	0	7	0	3
													4	7
											8	15	8	11
													12	15
									16	31	16	23	16	19
													20	23
											24	31	24	27
													28	31
							32	63	32	47	32	39	32	35
													36	39
											40	47	40	43
													44	47
									48	63	48	55	48	51
													52	55
											56	63	56	59
													60	63
					64	127	64	95	64	79	64	71	64	67
											88	95	88	91
													92	95
							96	127	96	111	96	103	96	99
													100	103
											104	111	104	107
													108	111
									112	127	112	119	112	115
													116	119
											120	127	120	123
													124	127
											184	191	184	187
													188	191
							2	223	192	207	192	199	192	195
													196	199
											200	207	200	203
													204	207
									208	223	208	215	208	211
													212	215
											216	223	216	219
													220	223
							224	255	224	239	224	231	224	227
													228	231
											232	239	232	235
													236	239
									240	255	240	247	240	243
													244	247
											248	255	248	251
													252	255



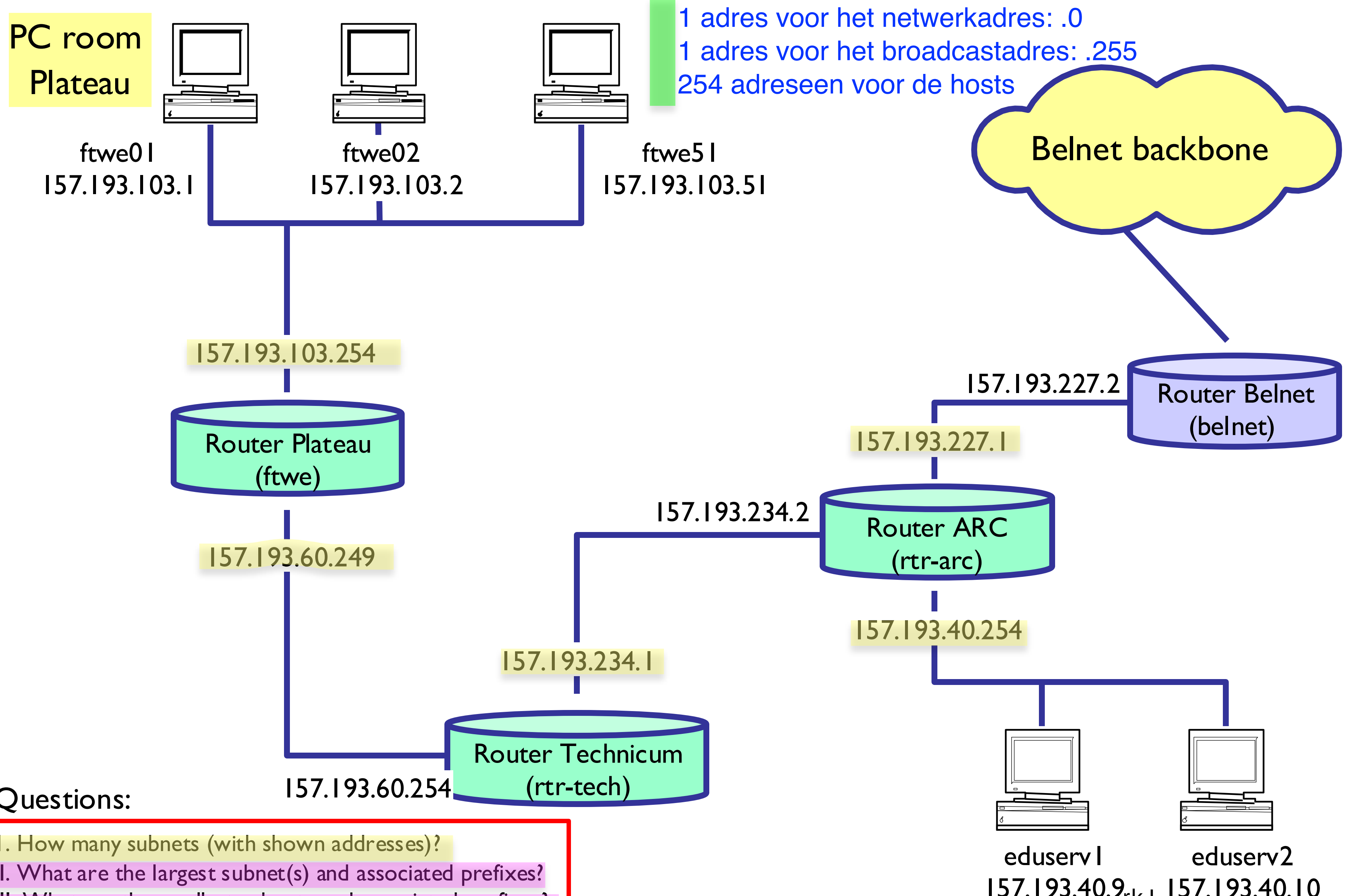
IMPOSSIBLE /28 subnet -> prefix is not the same:

0010 *0100* – 0011 *0011* : 36 - 51

Ex. 1 : Subnetting

alle subnettern hebben een /24 prefix: allemaal even groot

1 adres voor het netwerkadres: .0
1 adres voor het broadcastadres: .255
254 adreseen voor de hosts



Questions:

I. How many subnets (with shown addresses)?
II. What are the largest subnet(s) and associated prefixes?
III. What are the smallest subnets and associated prefixes?
IV. Number of addresses per subnet?

Note : assume smallest possible subnets with given addresses

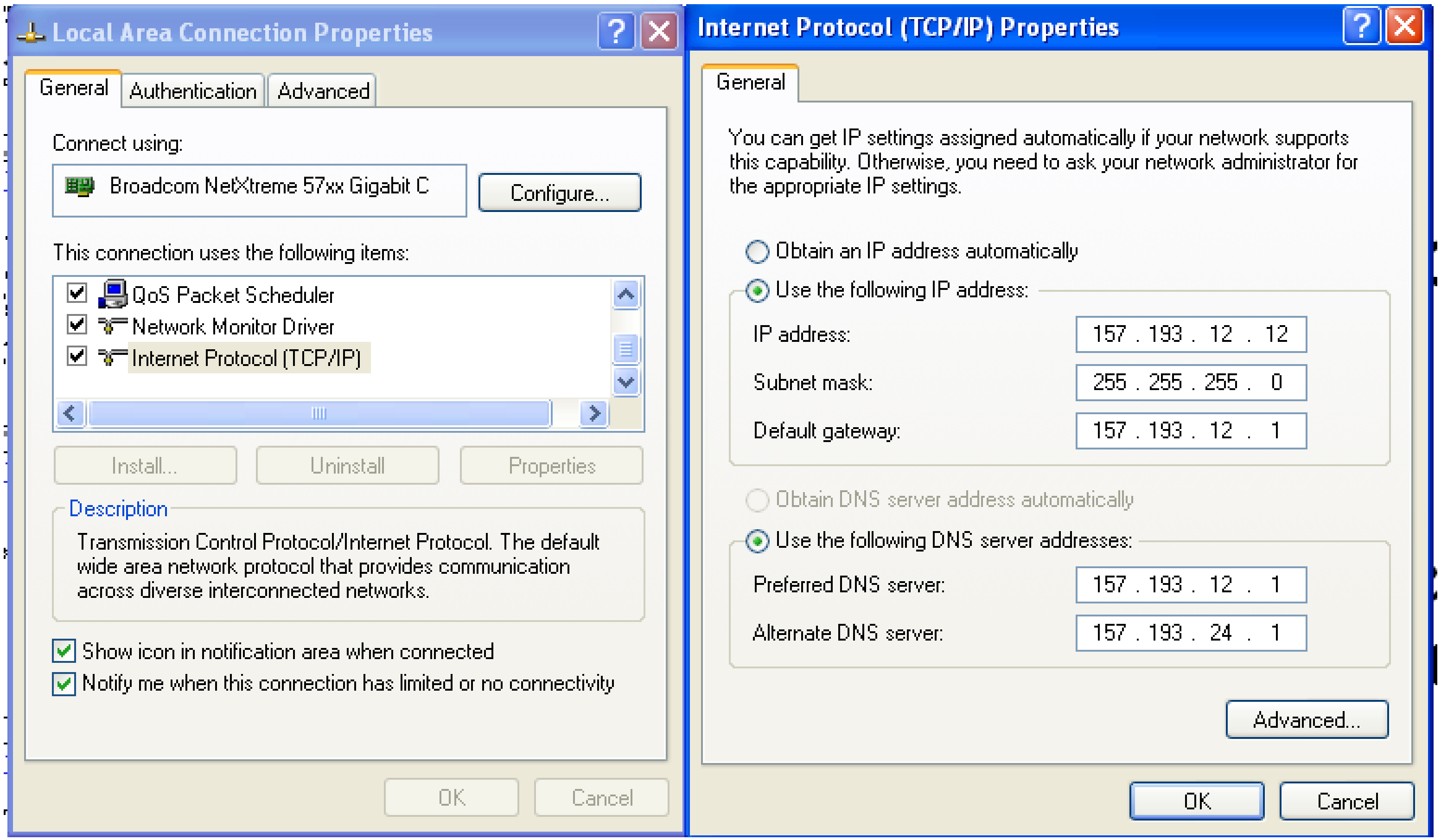
IP addresses: how to get one?

How does *host* get IP address?

- hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

Hard-coded IP address by sysadmin

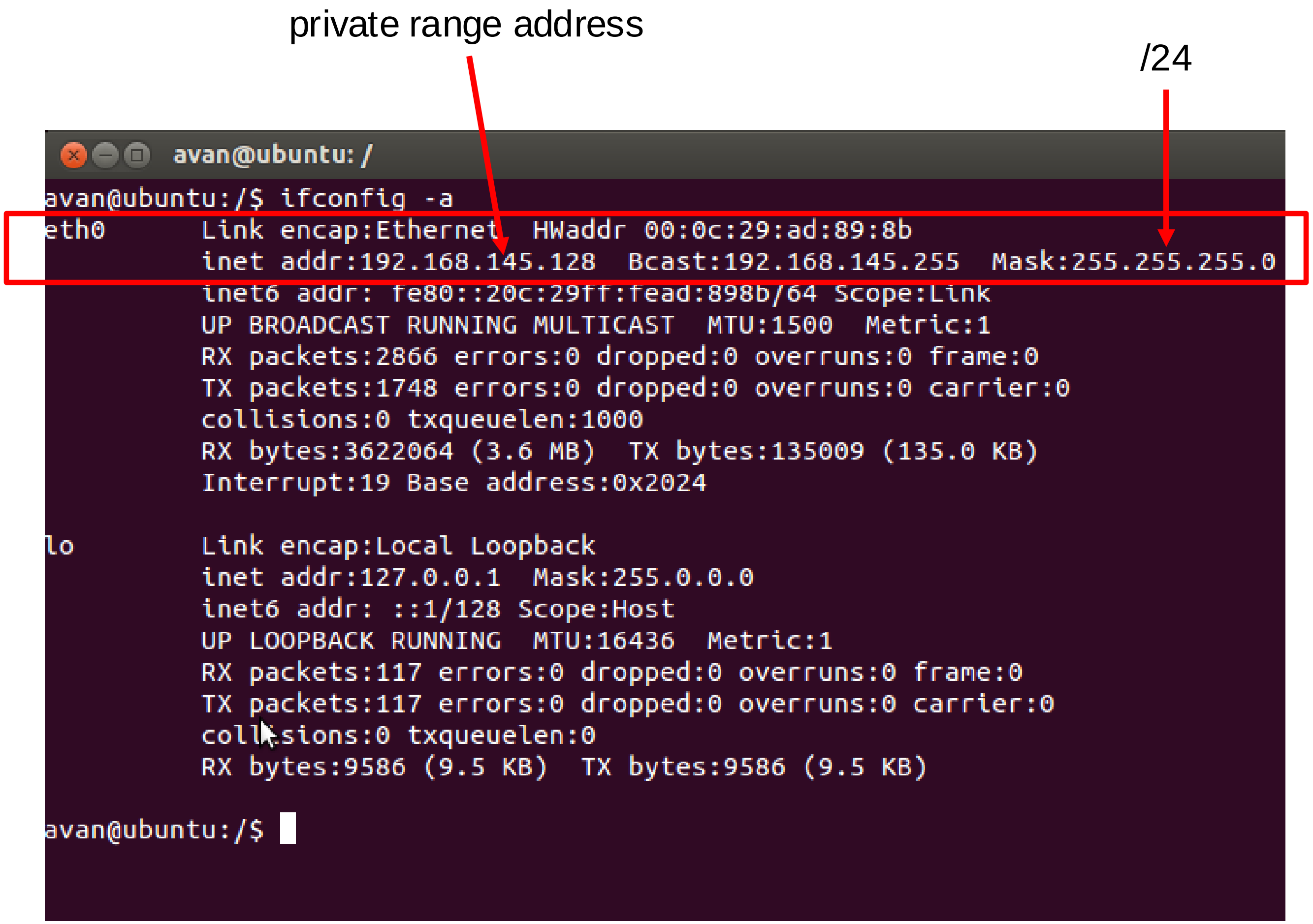
Windows: control-panel->network->configuration->tcp/ip->properties



Linux/MacOS: sudo ifconfig <ethx> <ipaddr> netmask <netmask>

persist in /etc/network/interfaces

ifconfig (Linux/macOS) or ipconfig (Windows) output



DHCP: Dynamic Host Configuration Protocol

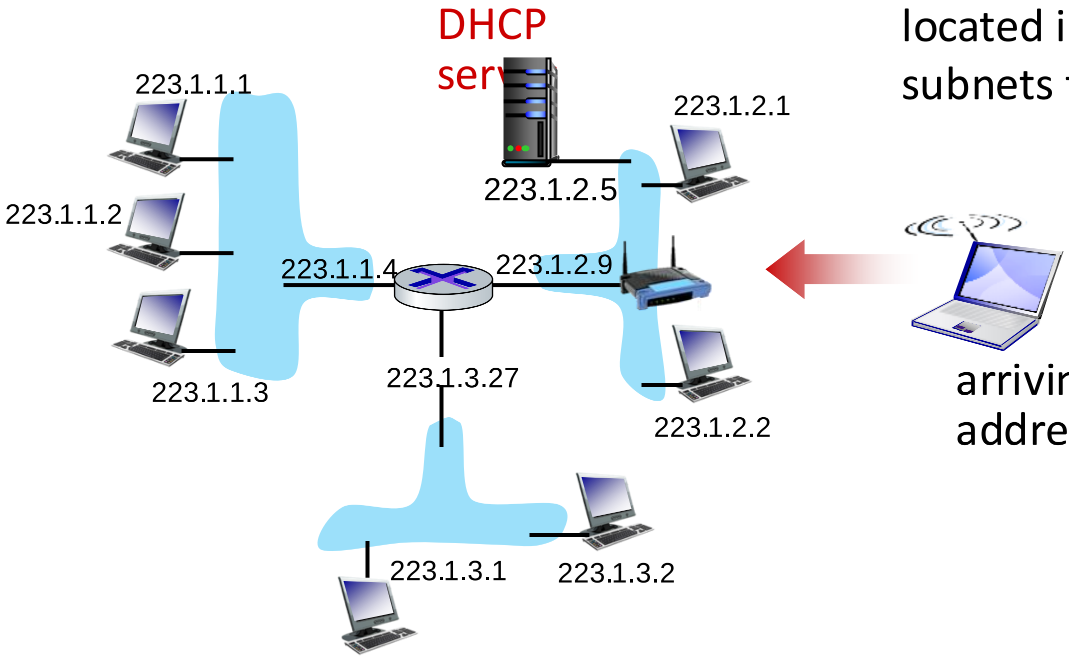
goal: host *dynamically* obtains IP address from network server when it "joins" network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

DHCP overview:

- host broadcasts DHCP discover msg [optional]
- DHCP server responds with DHCP offer msg [optional]
- host requests IP address: DHCP request msg
- DHCP server sends address: DHCP ack msg

DHCP client-server scenario

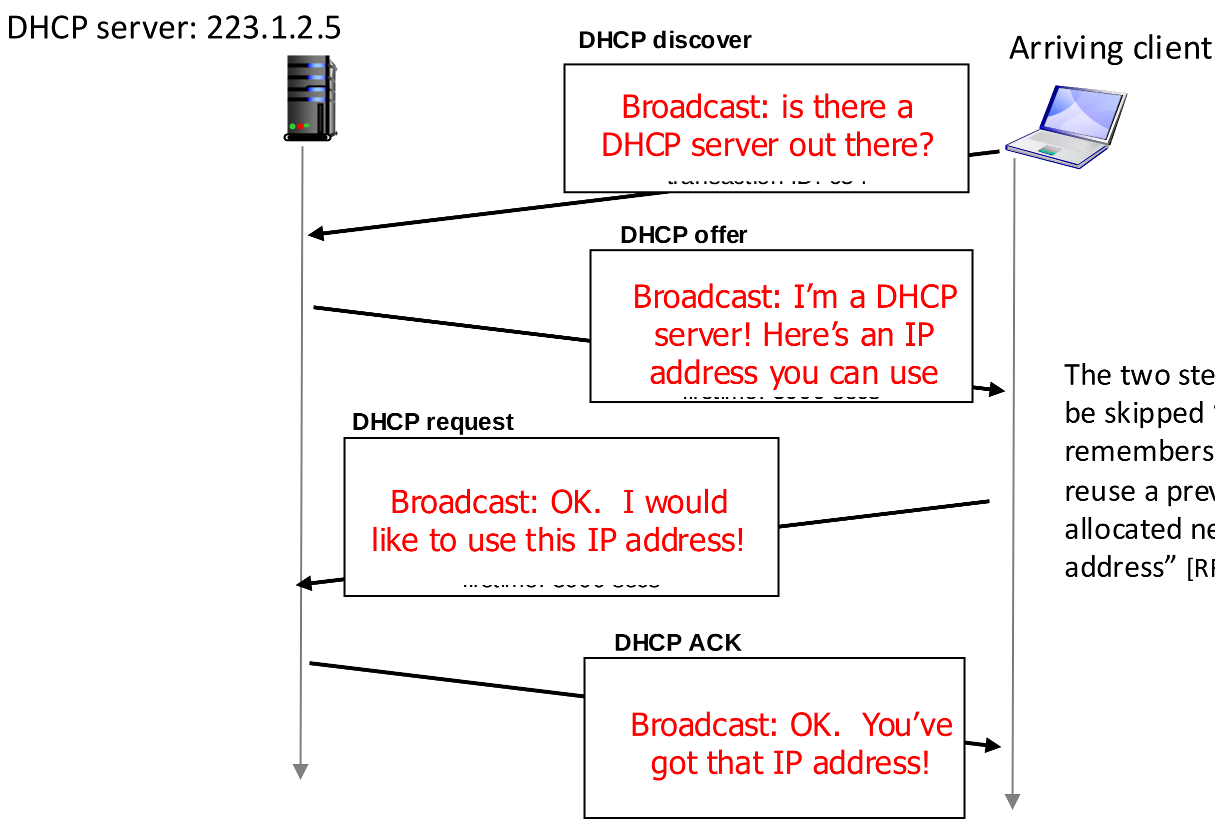


Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

arriving DHCP client needs address in this network

DHCP client-server scenario

de communicatie tussen de DHCP server en de client die een ip adres nodig heeft



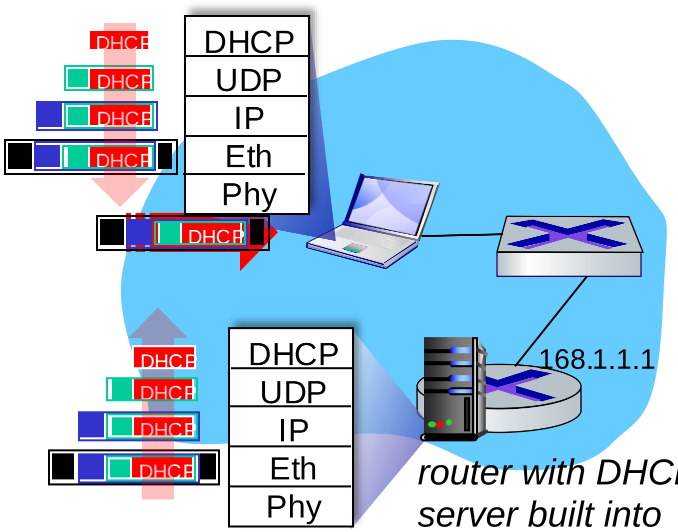
The two steps above can be skipped "if a client remembers and wishes to reuse a previously allocated network address" [RFC 2131]

DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

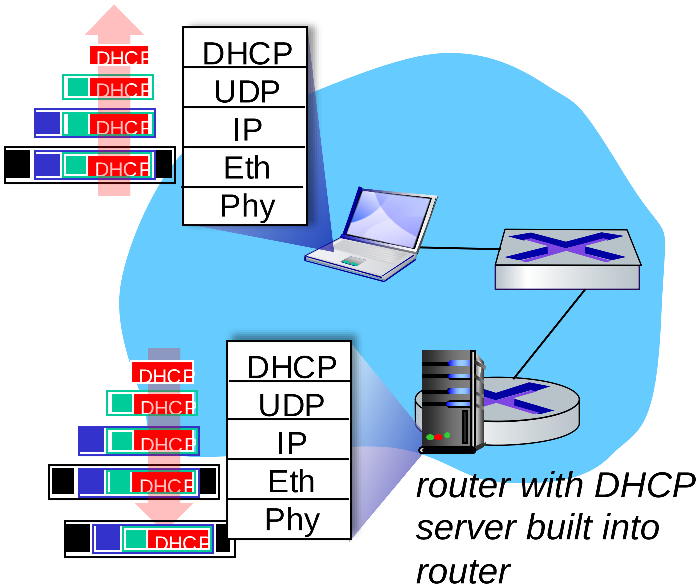
DHCP: example



router with DHCP server built into router

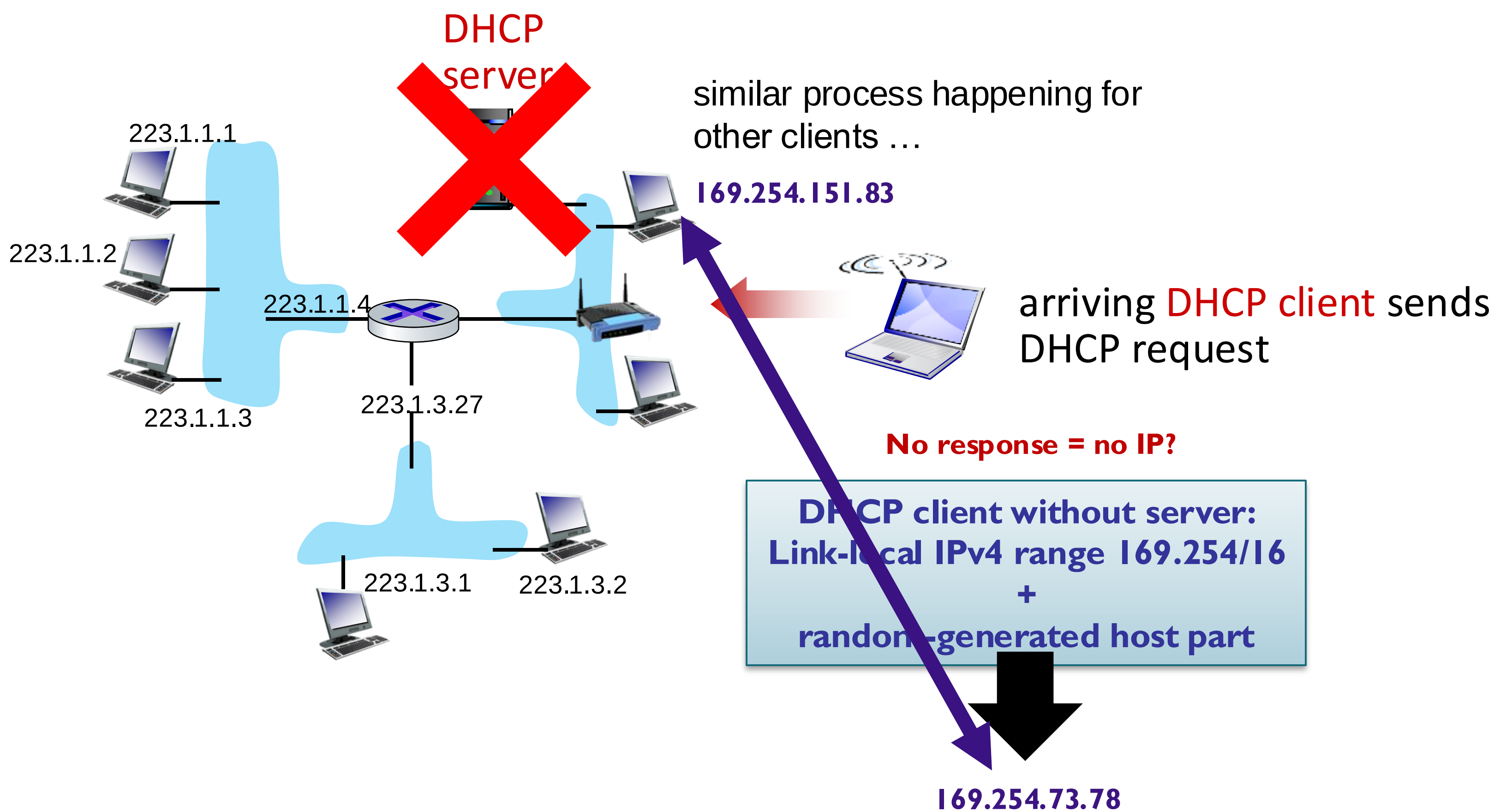
- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet de-mux'ed to IP de-mux'ed, UDP de-mux'ed to DHCP

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply forwarded to client, de-muxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

DHCP failure: Autoconfiguration Link-local



Network 169.254.0.0/16 allows communication between hosts, but not across routers !

DHCP limitations

- DHCP vs 169.254.0.0/16
  - DHCP, when again available, will configure interface with new address and priorly used link-local address won't be reachable anymore
- DHCP process
  - Single Point Of Failure in most organisations
  - DHCP relies on datalink layer broadcast mechanism
  - No IP communication possible before DHCP has finished
- IPv4 -> IPv6: update in these concepts …

The DHCP process has some limitations:

•When a host receives an address from the 169.254.0.0/16 range, it can setup communication with other hosts on the LAN. However, when a DHCP server is then enabled on the network, the 169.254.0.0/16 address will typically be disabled, and replaced with the address offered by the DHCP server. All active communication that was using this 169.254.0.0/16 address, will hence be interrupted.

•DHCP has been designed to allow multiple (synchronised) servers in the same network; the initial broadcast messages of the protocol keep all involved servers informed .

Most environments however only have one DHCP server, which creates a Single Point Of Failure. *DHCP functionality* without the need of a *DHCP server* would be something to think about when designing IPv6…

•DHCP relies on the broadcast mechanism of the underlying data-link layer. However, broadcast is a technique with drawbacks – and networks without broadcast traffic is considered a positive evolution.

•In IPv4, IP communication is not a default setting. A host node is not able to communicate until a DHCP offer has been received, or a manual IP configuration has been made.

When the IPv6 protocol was designed, these limitations of DHCP in IPv4 were considered. In the chapter IPv6 we will go into the details of the differences.

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

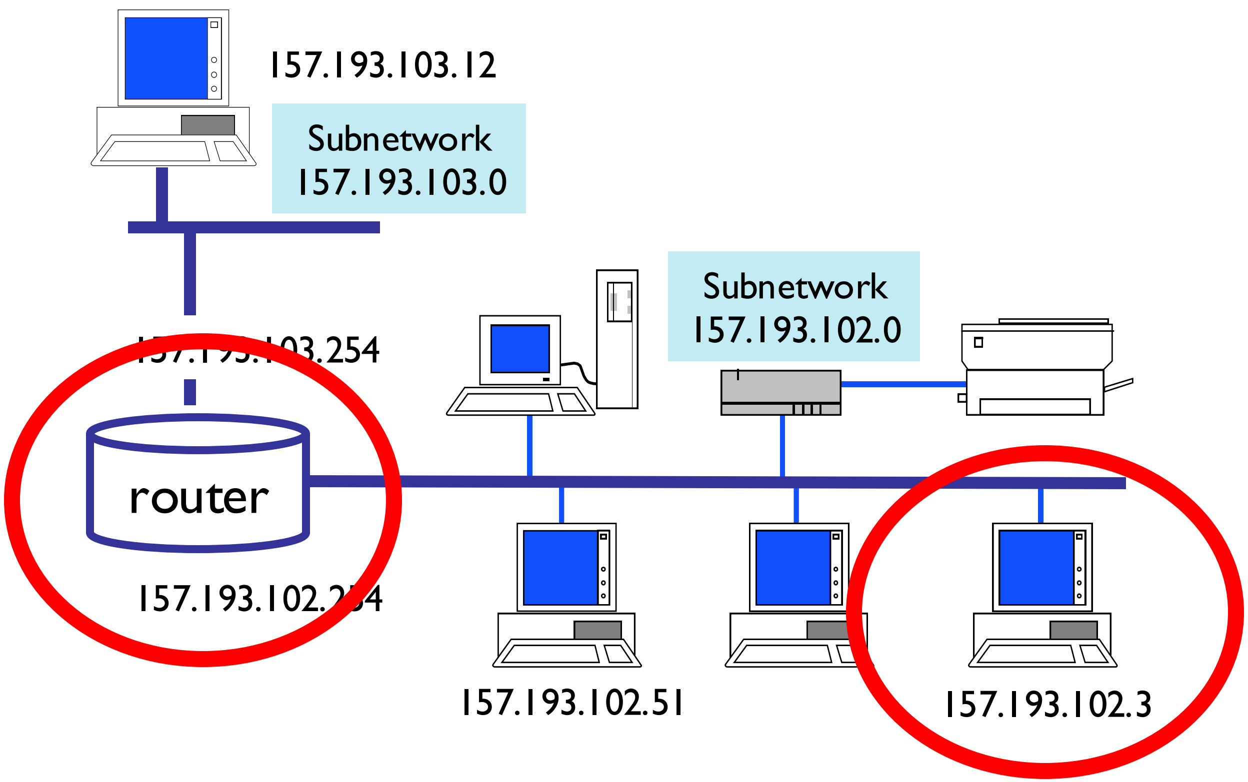
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

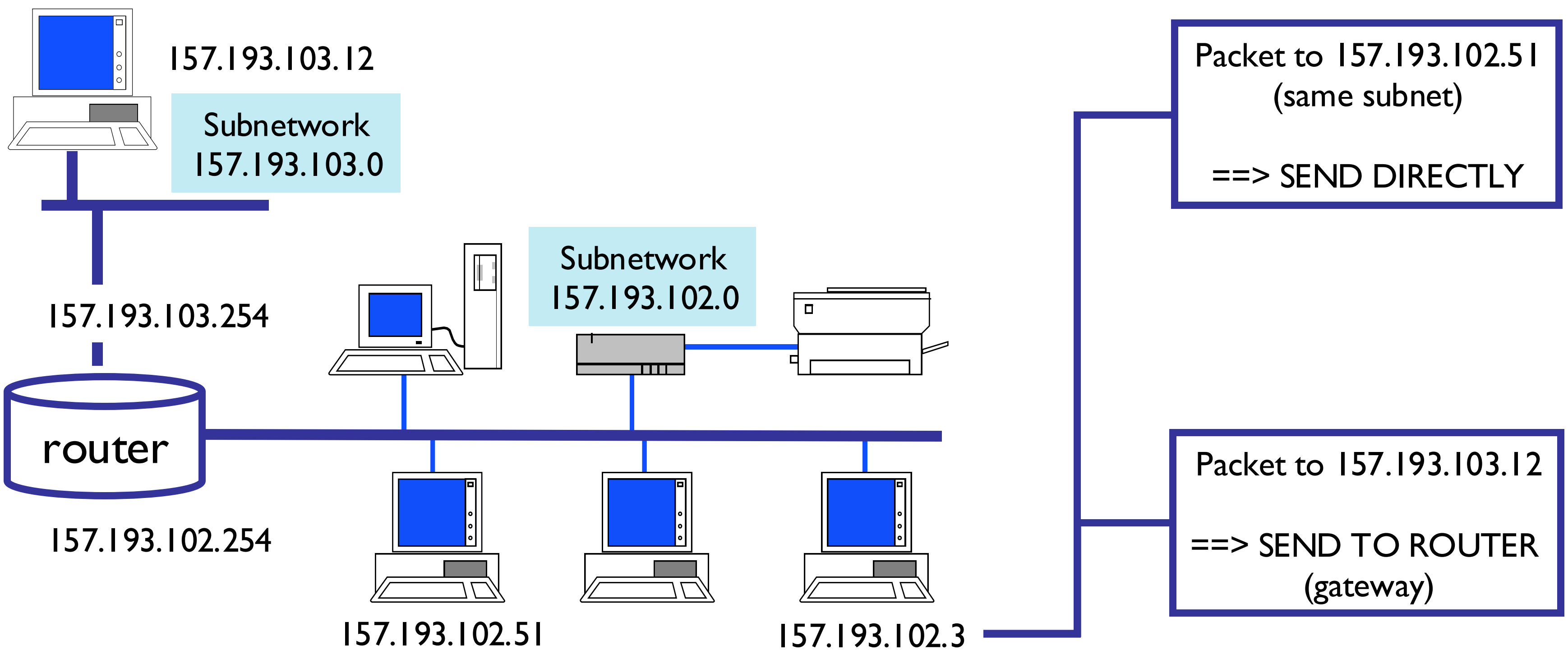
IPv4 forwarding

- Q: Which nodes do IP packet forwarding?
- A:
  - Hosts = source or destination of data packets
  - Routers = potential intermediate hop of data packets



IPv4 forwarding

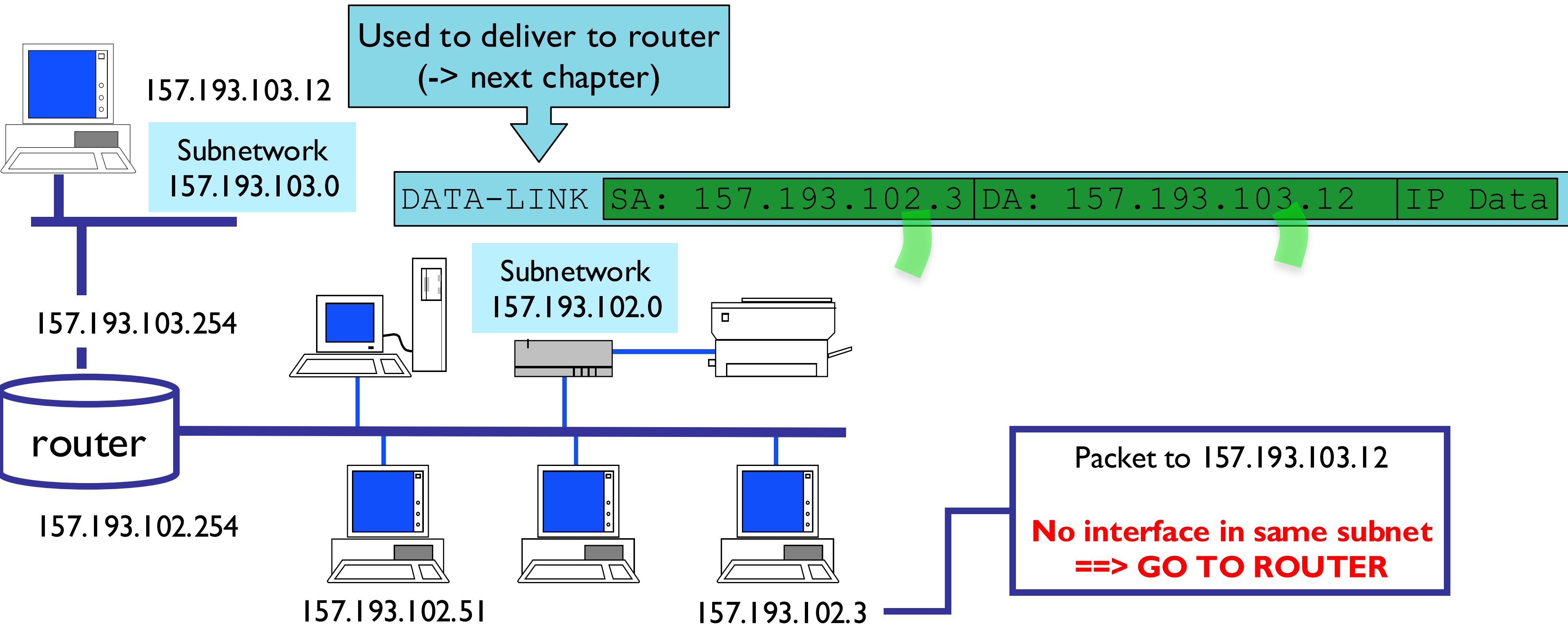
- Q: When do nodes need to forward packets to routers?
  - Subnet = collection of interfaces able to interact directly via link layer
  - Router = device/node connecting different (sub-)networks (acting at the network layer)
- A: When the packet destination is in another subnet than the sending device.
  - Gateway = router which gives access to the other subnet



subnet mask : FF.FF.FF.00 or /24

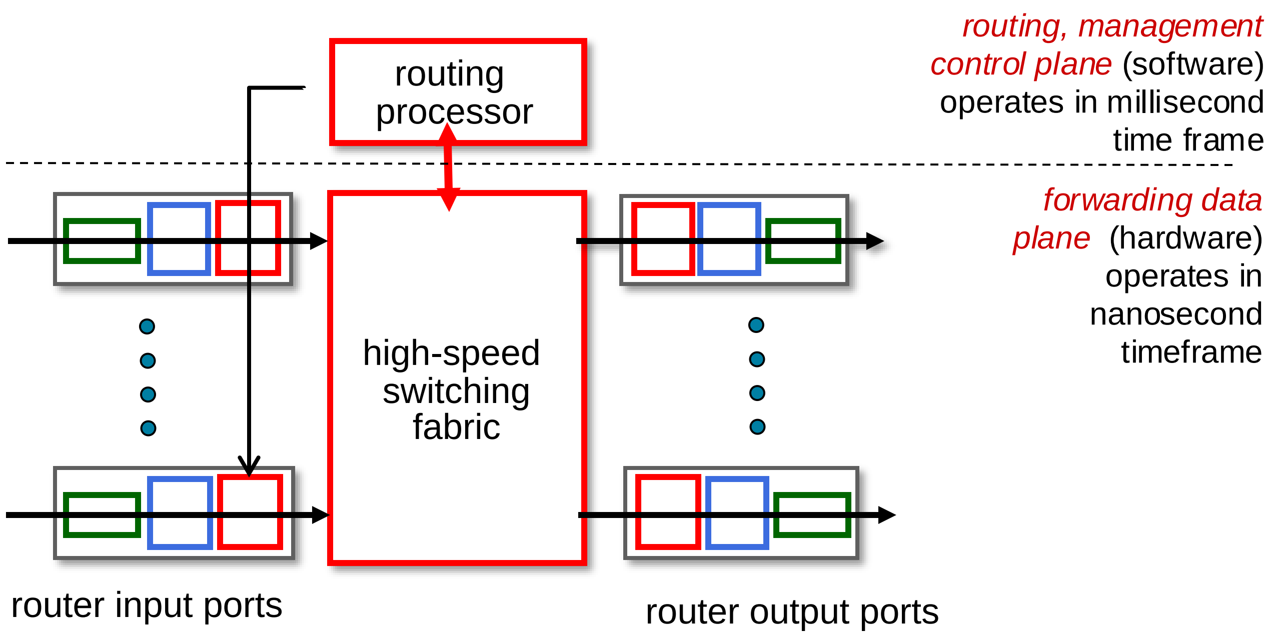
IPv4 forwarding – Role of the link layer

- Q: Sending to router: How does the packet get to the router ?
  – It can not be based on the IP header (unchanged) containing final destination

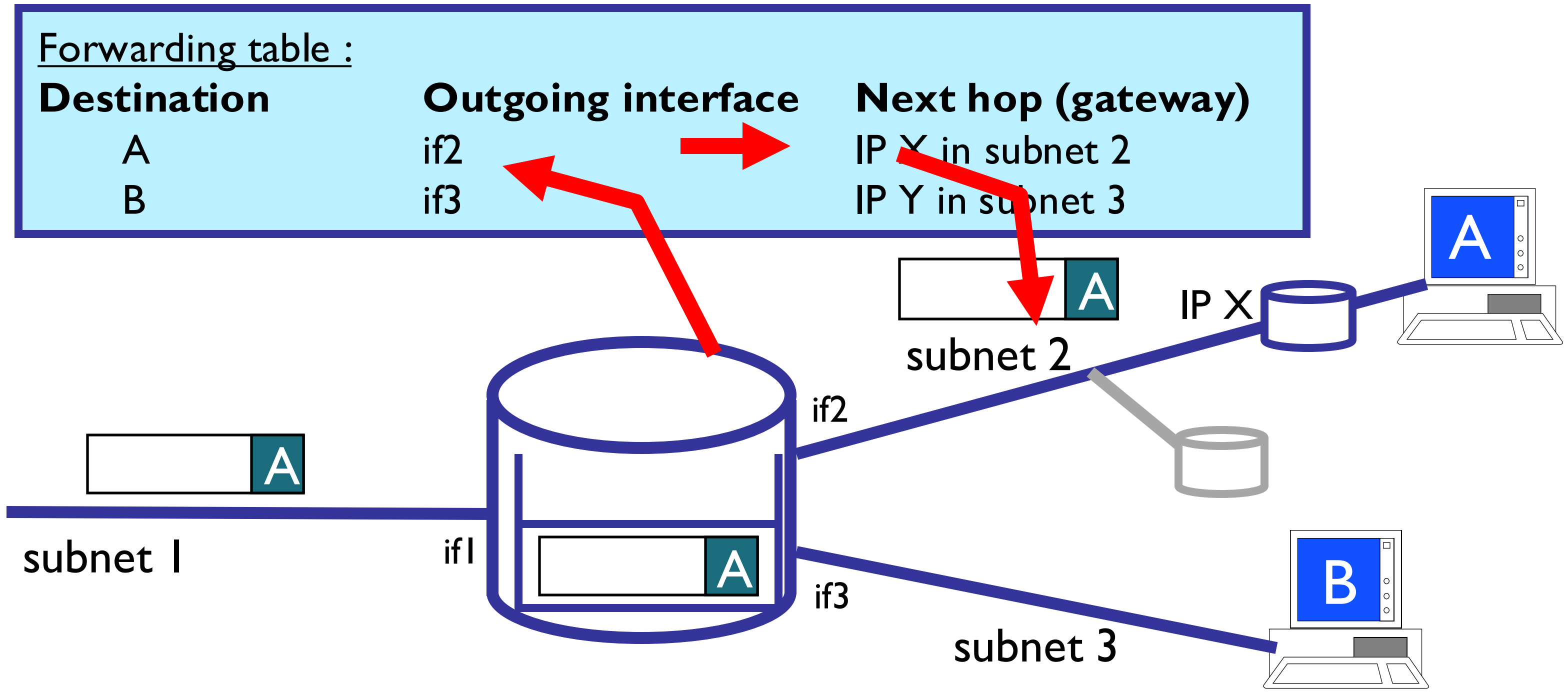


Router architecture overview

high-level view of generic router architecture:



Router forwarding



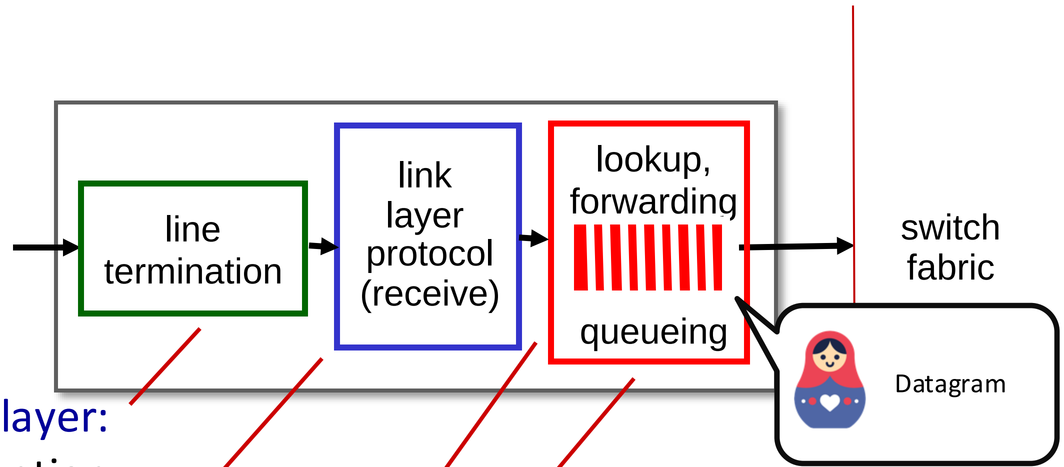
Store & forward operation in a router:

1. IP packet arrives in a router on incoming interface
   - IP header is analyzed & sanity checks
   - Use destination-based lookup for determining next hop
2. Send through switching fabric
3. Transmit on outgoing interface

Why do you need both “outgoing interface” and ”next hop”?
-> Because the gateway to take can be different for the same outgoing interface: can be router with IP X for destination A, but for another destination it could need the grey router as gateway

Router forwarding – 1. Input port functions

Green box is a physical layer responsible for receiving bit level transmission. Copper, fiber wireless.
Link layer in blue – bit assemble into Ethernet frame. We will discuss this in details in chapter 6.
But most critical one, is here. Lookup, forwarding function. Lookup and forward is “Match plus action” behavior.
GOAL – processing at “line speed”, otherwise, buffer will be full and packet will be lost



physical layer:
bit-level reception

link layer:
e.g., Ethernet
(chapter 6)

IP header sanity checks

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *(“match plus actior*
- goal: complete input port processing at ‘line speed’
- input port queuing: if datagrams arrive faster than forwarding rate into switch fabric

Router forwarding – 1a. Header analysis

- Sanity checks on the packet header
  - Checksum (RFC 791)
    - If the header checksum fails, the internet datagram is discarded at once by the entity which detects the error.
    - Why?
      - since some header fields change (e.g., TTL), this is recomputed and verified at each point that the internet header is processed.
  - Time To Live (RFC 1812)
    - When a router forwards a packet, it MUST reduce the TTL by at least one.
    - If the TTL is reduced to zero (or less), the packet MUST be discarded, and […] the router MUST send an ICMP Time Exceeded message, Code 0 (TTL Exceeded in Transit) message to the source.
    - Why?
      - the forwarding of routers might be misconfigured, causing routing loops

Router forwarding – 1b. Longest prefix match

Text for this animation:
This all works out pretty nice and looks pretty simple!
But of course the devil is in the details, as the saying goes.
What happens, for example, when a subset of addresses say in this first range should go to say interface 3, rather than interface 0. Well, of course we could split the first address range into multiple pieces, and add in this new subrange with its new destination output port. But it turns out there's a much simpler and elegant way to do this. Known as longest prefix matching.

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through	0
11001000 00010111 00010000 00000100 through 11001000 00010111 00010000 00000111	3
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

geel : de prefix: dat meerdere adressen in de tabel gemeenschappelijk hebben
roze: nauwkeurigere langere prefix aan voor de specifieke adressen binnen het getele beeiuk

Router forwarding – 1b. Longest prefix match

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

deze komt het meeste overeen met de bovenste van de 3 —> 0

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

komt het meeste overeen met de 2e uit voorbeeld -> 1

Router forwarding – 1b. Longest prefix match

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range				Link interface
11001000	00010111	00010***	********	0
11001000	00010111	00011000	********	1
11001000	00010111	00011***	********	2
otherwise				3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Router forwarding – 1b. Longest prefix match

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Router forwarding – 1b. Longest prefix match

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

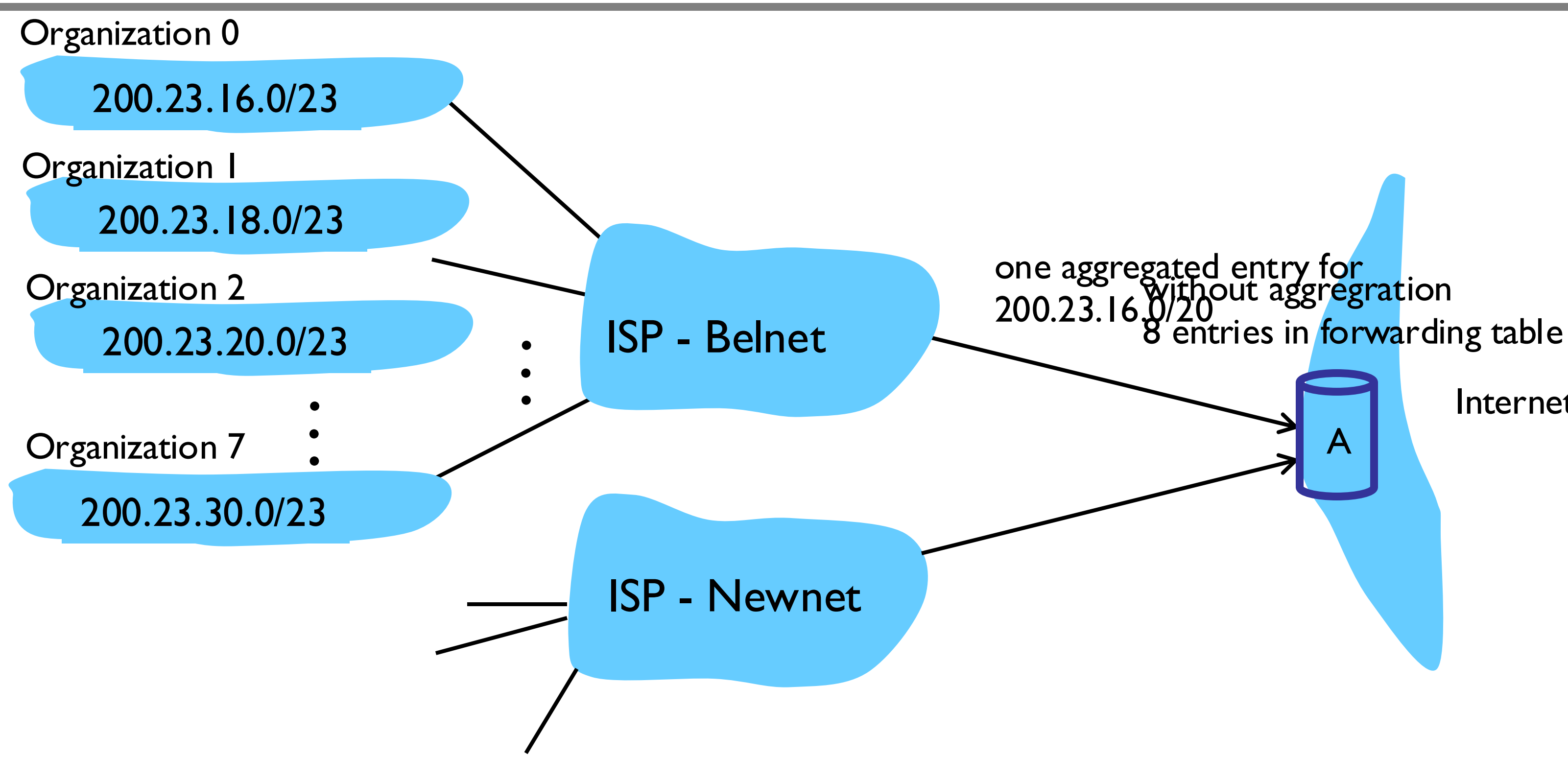
examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Router forwarding – 1b. Entry aggregation

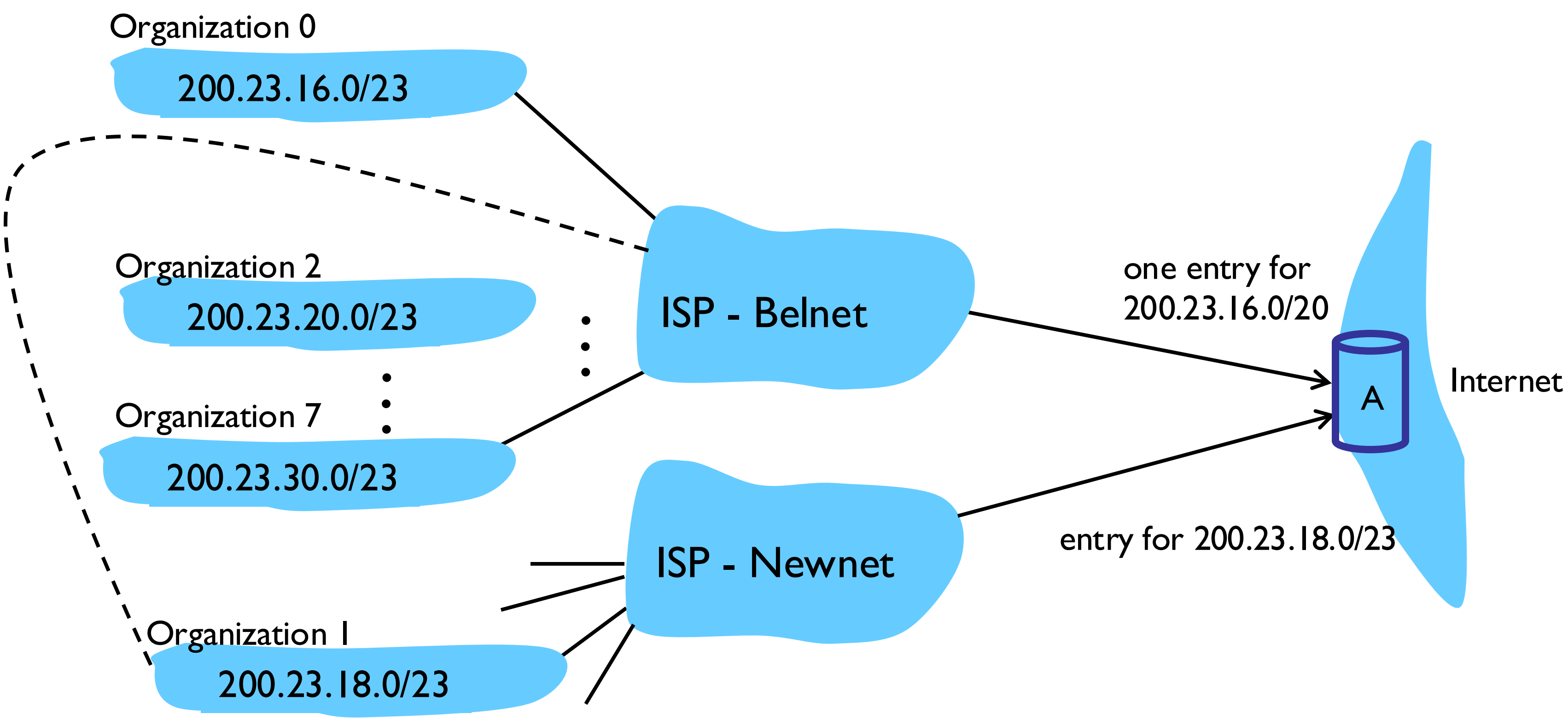
address block	11001000	00010111	00010000	00000000	200.23.16.0/20
Organization 0	11001000	00010111	00010000	00000000	200.23.16.0/23
Organization 1	11001000	00010111	00010010	00000000	200.23.18.0/23
Organization 2	11001000	00010111	00010100	00000000	200.23.20.0/23
...	…..			….	….
Organization 7	11001000	00010111	00011110	00000000	200.23.30.0/23



Note: Belnet today: no aggregation possible
UGent: 157.193.0.0; KULeuven: 134.58.0.0; VUB: 134.184.0.0; UCL: 130.104.0.0; …

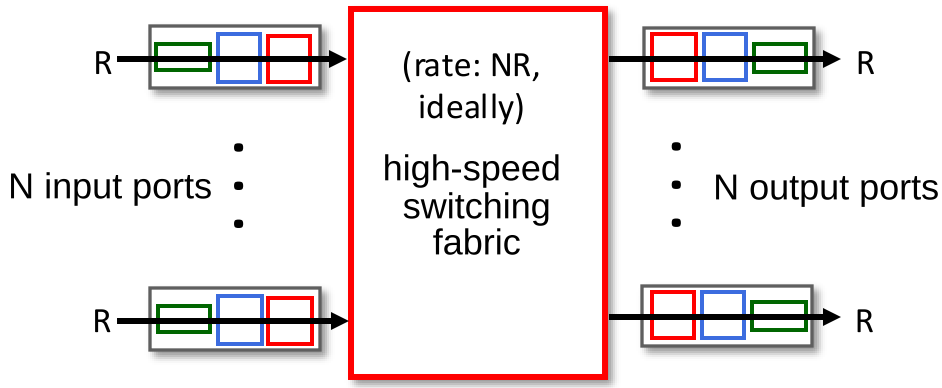
Router forwarding – 1b. Specific routes

- Shorter routes can still be used due to longest prefix matching!
  - 200.23.18.0/23 will be matching 200.23.16.0/20 (aggregated address block)
  - 200.23.18.0/23 will have a <u>longer match</u> with the specific entry 200.23.16.0/23



Router forwarding – 2. Send to switching fabric

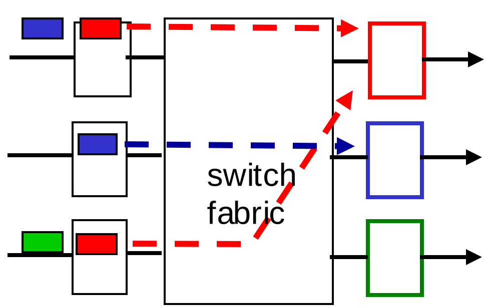
- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable



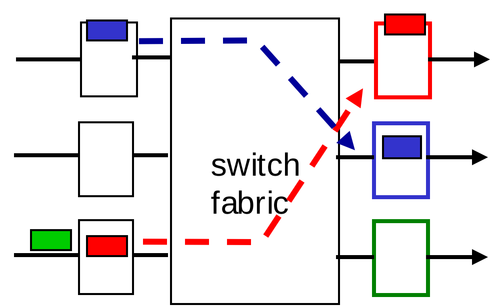
Router forwarding – 1b. Input port queueing

- If switch fabric slower than input ports combined -> queueing may occur at input queues
  - queueing delay and loss due to input buffer overflow!

- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

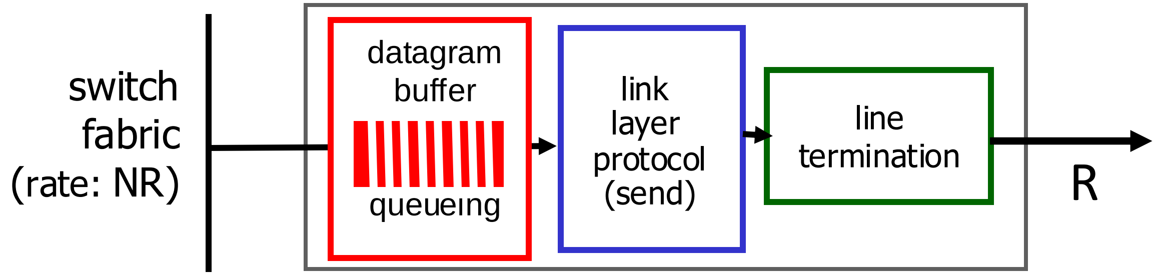


output port contention: only one red datagram can be transferred. lower red packet is *blocked*



one packet time later: green packet experiences HOL blocking

Router forwarding – 3. Transmit on outg. if.



- *Send to outgoing interface:*
  - fragment if packet size > MTU on outgoing link (Maximum Transfer Unit)
  - recalculate header checksum

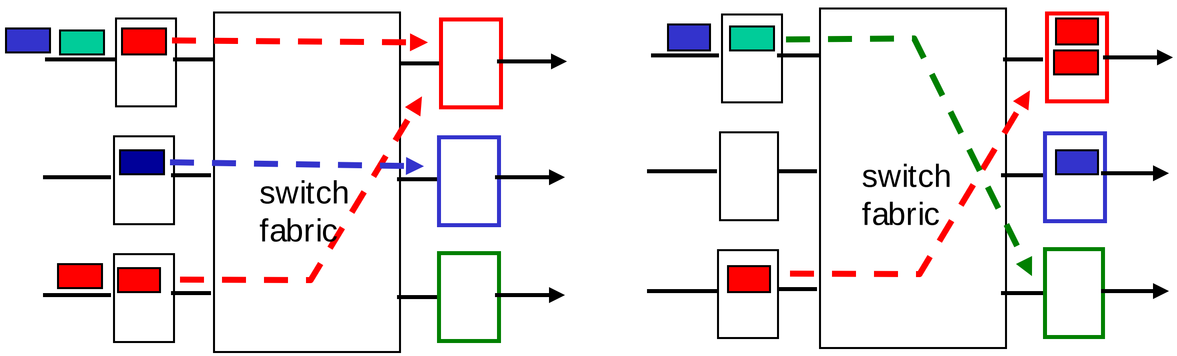
- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

Datagrams can be lost due to congestion, lack of buffers

- *Scheduling discipline* chooses among queued datagrams for transmission

Who gets best performance?

Router forwarding – 3. Output port queuing

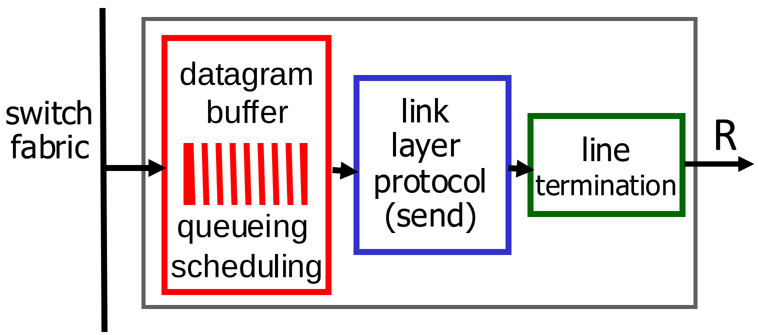


at *t,* packets more from input to output

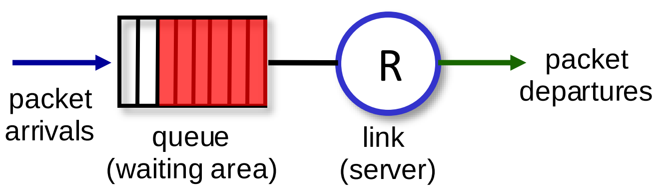
one packet time later

- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

Router forwarding – 3. Buffer management

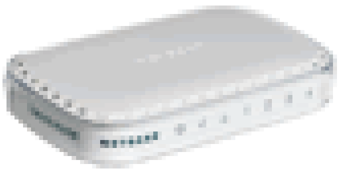


Abstraction: queue



- drop: which packet to add, drop when buffers are full
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
- marking: which packets to mark to signal congestion (ECN, RED)
- packet scheduling: deciding which packet to send next on link
  - first come, first served
  - priority
  - round robin
  - weighted fair queueing

Router Examples



Linksys, Siemens (DSL), Motorola (Cable), D-Link, US-Robotics, Netgear, …

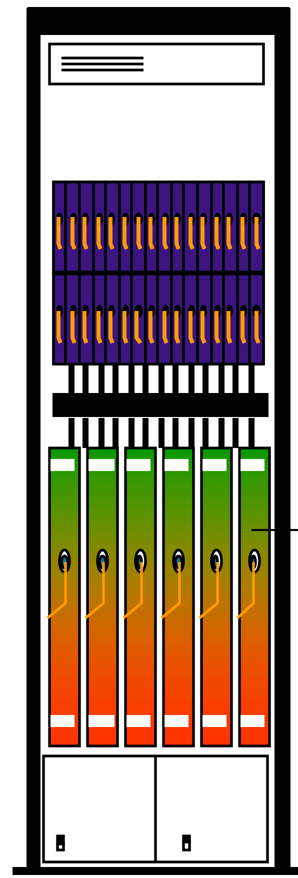
Router Examples

longest prefix matching: often performed using ternary content addressable memories (TCAMs)

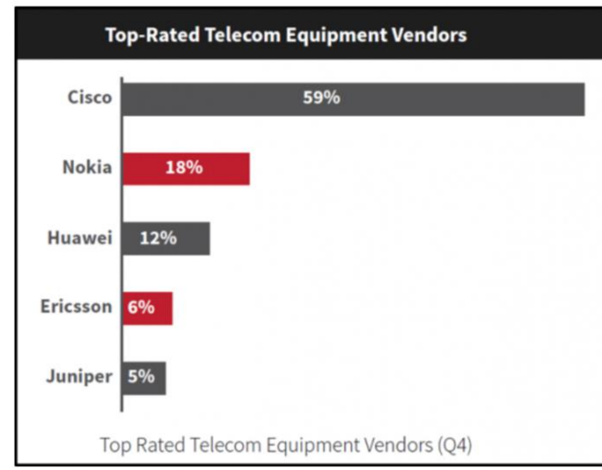
content addressable: present address to TCAM: retrieve address in one clock cycle, regardless of table size

Cisco Catalyst: ~1M routing table entries in TCAM

Cisco CRS 1 (Carrier Routing System)

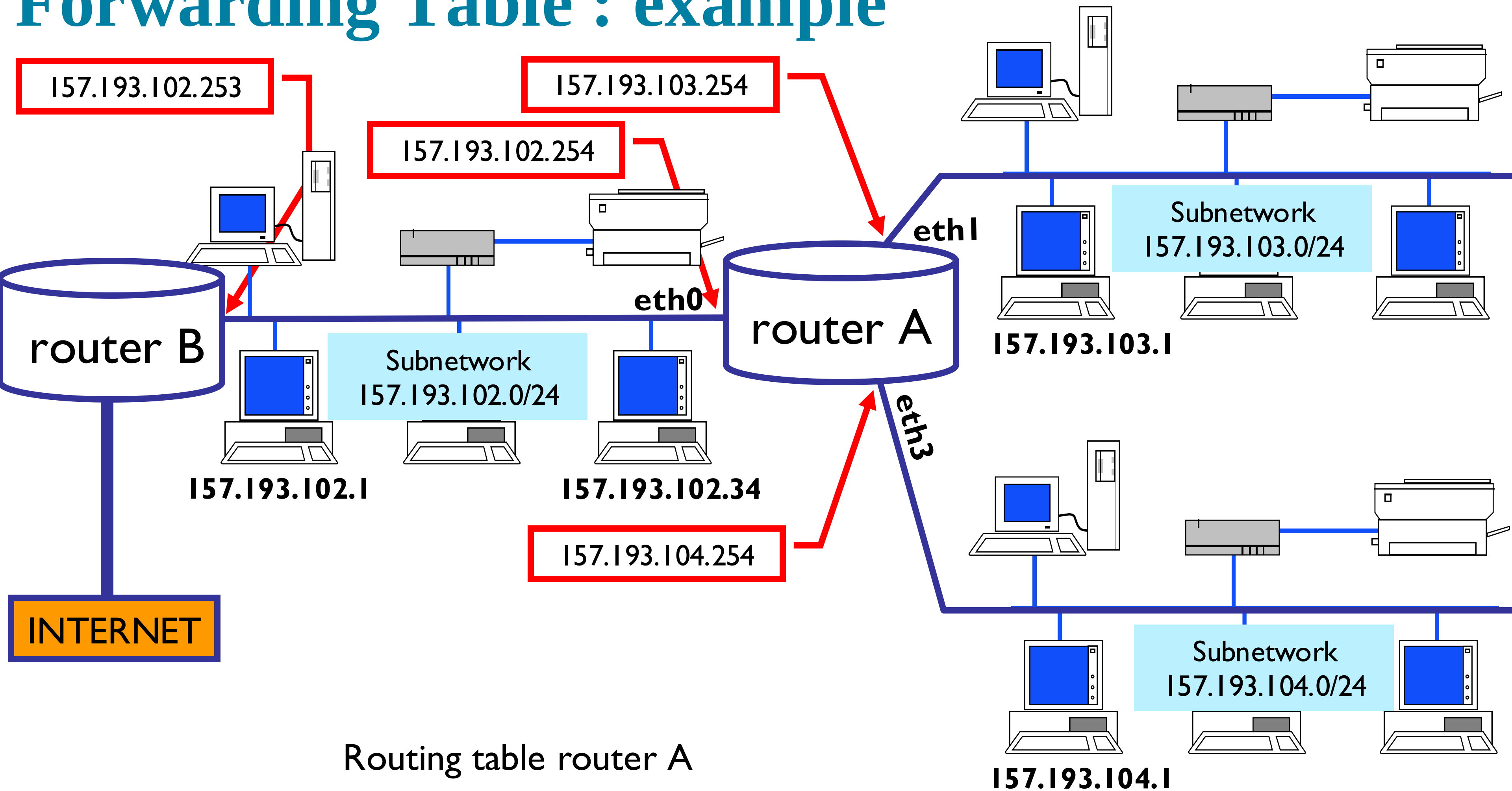


Linecards



Cisco, Juniper Networks, Ericsson, Nokia, Huawei, …

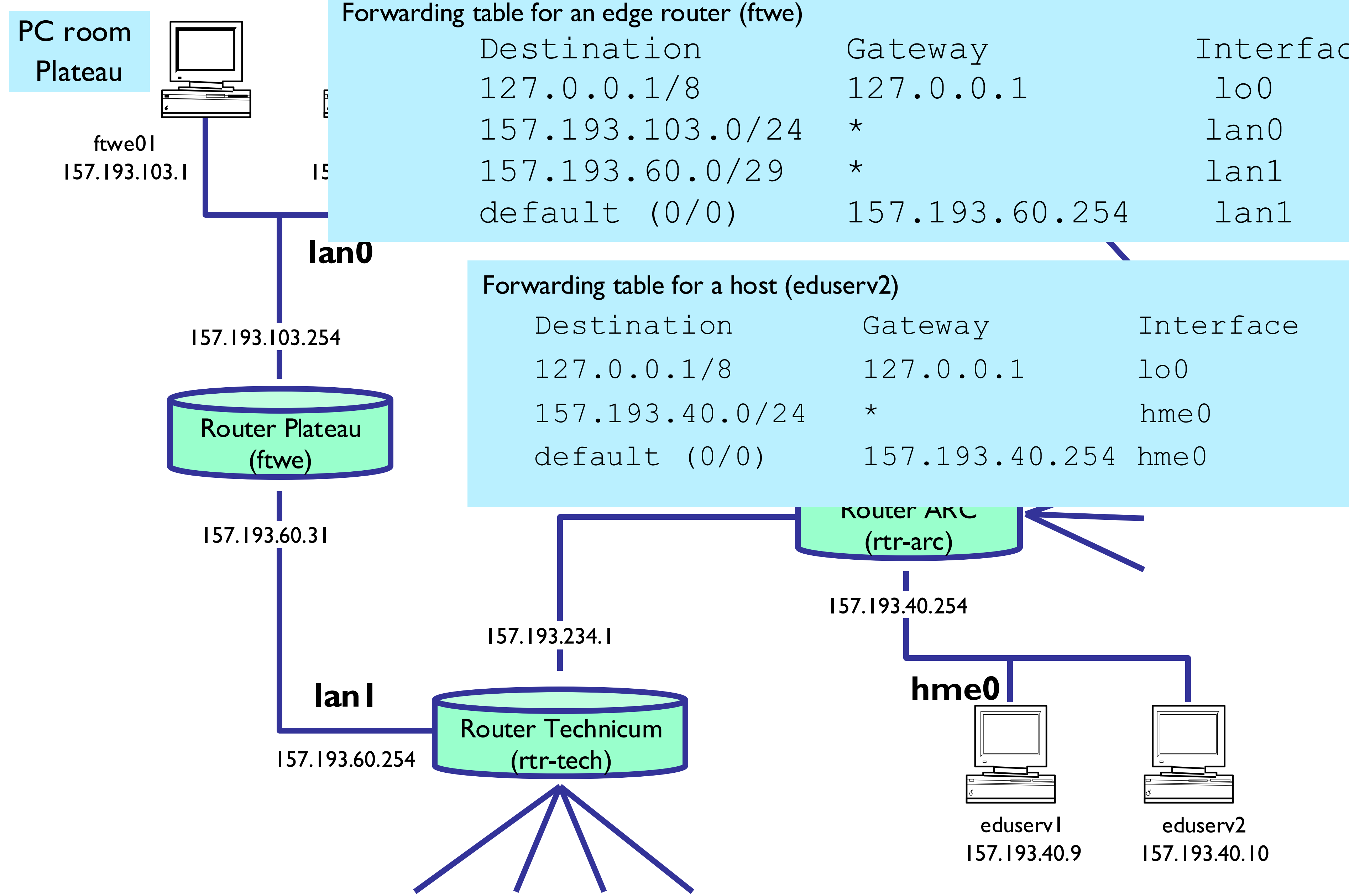
Forwarding Table : example



Routing table router A

Destination	Mask	Gateway	Interface	Interface
127.0.0.0	/8	127.0.0.1	Lo0	127.0.0.1
0.0.0.0	/0	157.193.102.253	Eth0	157.193.102.254
157.193.102.0	/24	*	eth0	157.193.102.254
157.193.103.0	/24	*	eth1	157.193.103.254
157.193.104.0	/24	*	eth3	157.193.104.254

Forwarding Table : Example UGent network



Forwarding table for an edge router (ftwe)

Destination	Gateway	Interfac
127.0.0.1/8	127.0.0.1	lo0
157.193.103.0/24	*	lan0
157.193.60.0/29	*	lan1
default (0/0)	157.193.60.254	lan1

Forwarding table for a host (eduserv2)

Destination	Gateway	Interface
127.0.0.1/8	127.0.0.1	lo0
157.193.40.0/24	*	hme0
default (0/0)	157.193.40.254	hme0

Short notation for 127.0.0.0/8

Short notation for 168.254.0.0/16

Forwarding Table : examples (macOS)

```
❯ netstat -r[illegible]
Destination        Gateway            Flags          Netif Expire
default            192.[illegible]8.50.1       UGScg            en0
127                127.0.0.1          UCS              lo0
127.0.0.1          127.0.0.1          UH               lo0
169.254            link#11            UCS              en0       !
192.168.50         link#11            UCS              en0       !
192.168.50.1/32    link#11            UCS              en0       !
192.168.50.1       c8:7f:54:c1:2:[illegible]   UHLWIir          en0    1181
192.168.50.6       0[illegible]22:6c:5c:2c:ee  UHLWI            en0     774
192.168.50.60      22:4[illegible]:73:c0:ca:5a U[illegible]WI            en0    1144
192.168.50.91/32   link#1[illegible]            UCS              en0       !
192.168.50.123     4a:87:22:[illegible]:40:89  UHLWI            en0    1102
192.168.50.156     82:8:75:9b:[illegible]4     UHLWI            en0    1180
192.168.50.255     ff:ff:ff:ff:ff[illegible]   UHLWbI           en0       !
224.0.0/4          link#11            [illegible]CS             en0       !
224.0.0.251        1:0:5e:0:0:fb      [illegible]WI            [illegible]n0
255.255.255.255    ff:ff:ff:ff:ff:ff  UH[illegible]                       !
```

- **U**: The route is up and active.
- **H**: The route is to a host (as opposed to a network).
- **G**: The route uses a gateway.
- **S**: The static route (implicitly set up by the system).
- **C**: The route is a connected route (directly connected to one of the interfaces).
- **L**: Local address (part of the routing that deals with local traffic).
- **W**: Indicates the next-hop is a proxy address.
- **b**: Broadcast address.
- **m**: Multicast route.
- **r**: Re-initialization needed.
- **i**: Route installed by the router.

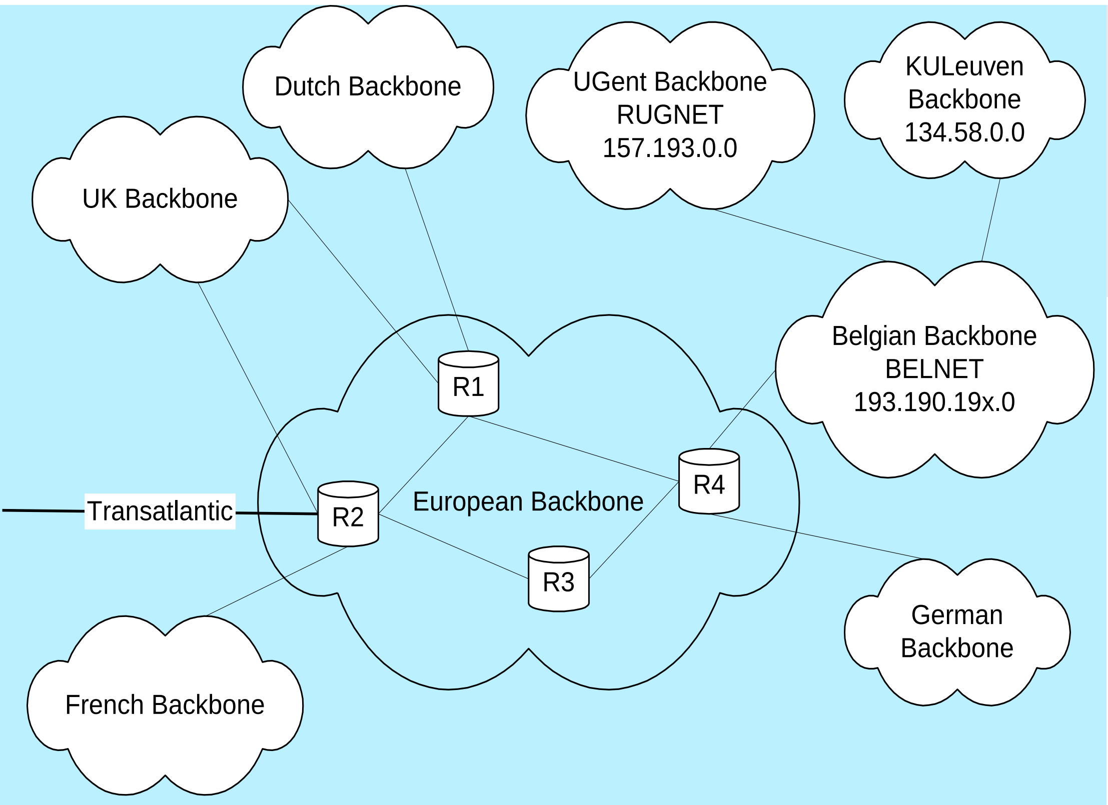
No GW needed, directly connected network (alternative name for en0)

Specific entry for directly connected device
Next line also stores MAC address (part of ARP cache, cfr. Chapter 6)

The Forwarding Table in a Large Network

European Backbone Router :

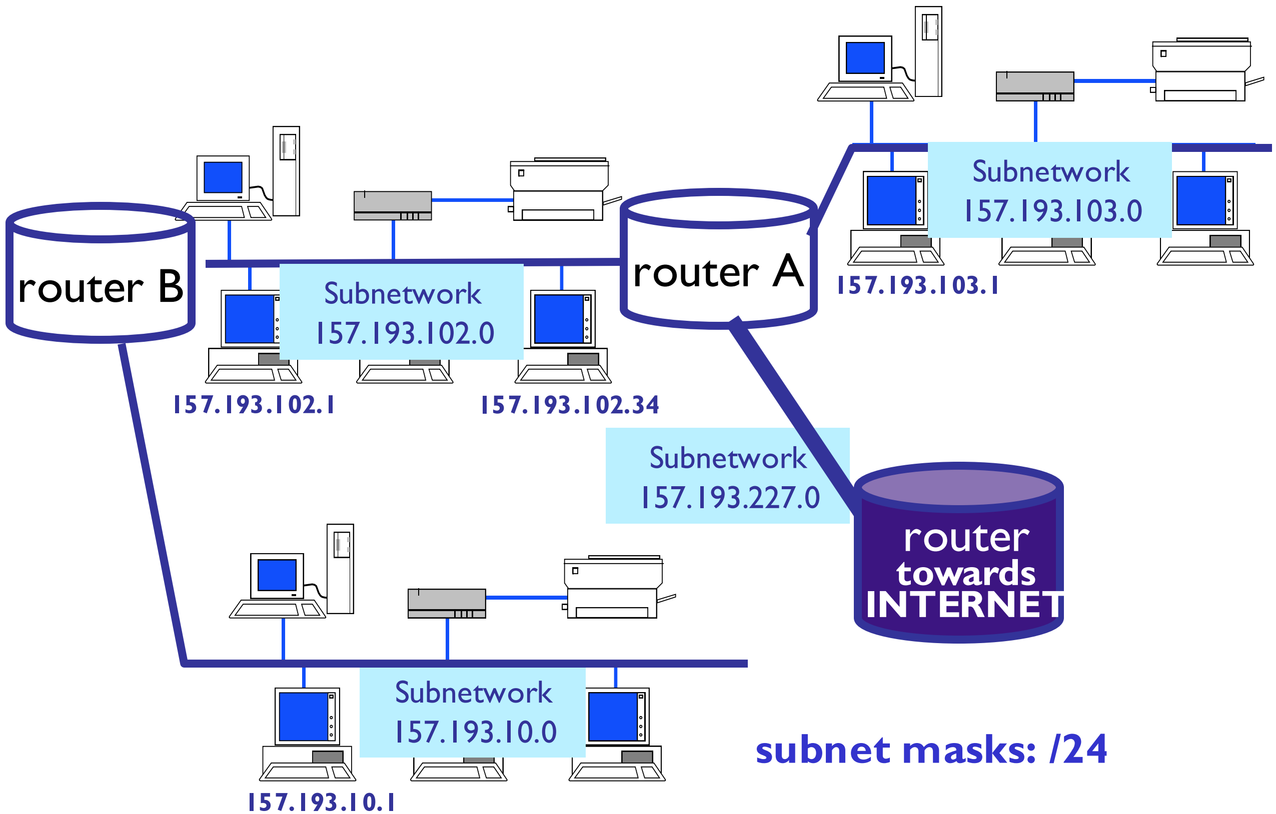
- entries to directly attached networks (e.g. BELNET)
- entries to every network attached to these directly attached networks (e.g.: BELNET : UGent, KULeuven, VUB, … backbone)
- entries to every network in the world (but e.g. US as default)



Large Router :

- up to 500.000 entries in forwarding table
- very expensive hardware needed to ensure fast lookups at line speed (100Gbps)
- stability of routing protocols (many routes to be advertised)

Ex. 2 : Forwarding Table Exercise: router A



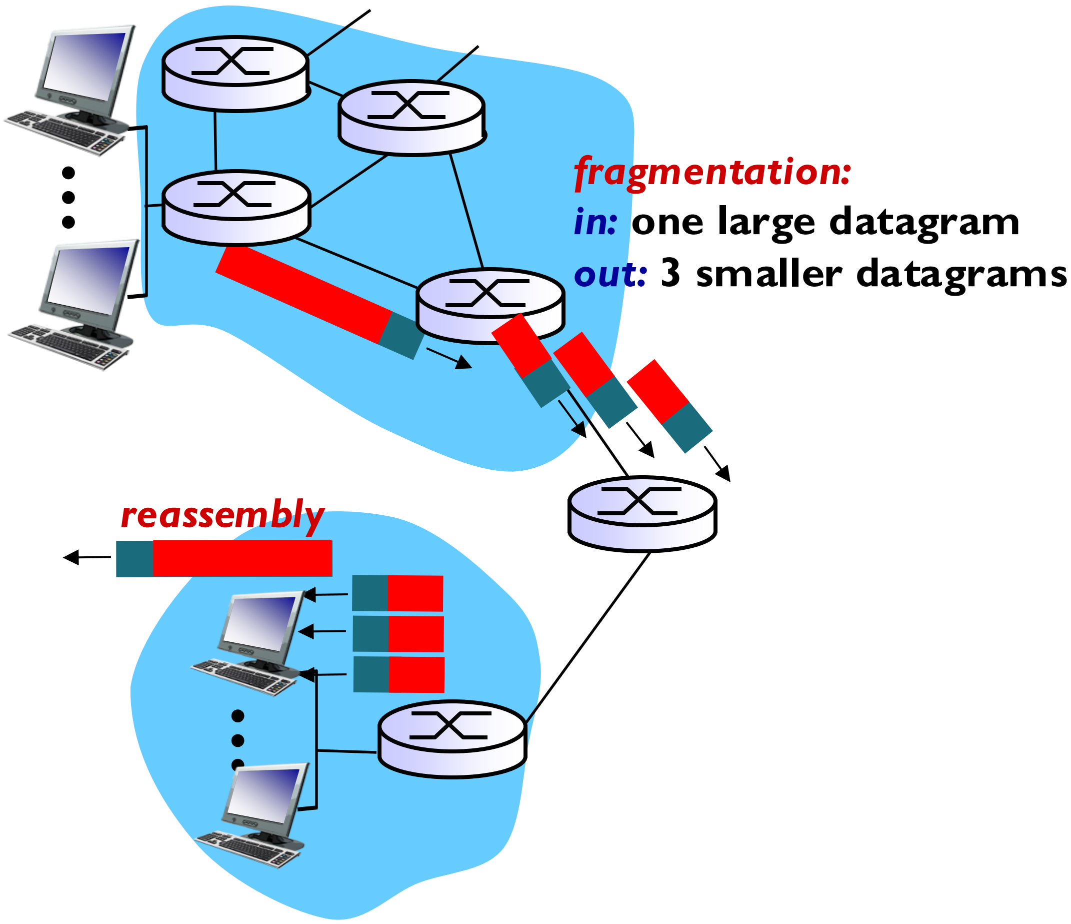
Questions:

1. Write down the forwarding table of router A

If needed, give interfaces an IP address in order to be able to provide the routing table of router A. You may assume the largest available address in the given subnet.

IP Fragmentation & Reassembly

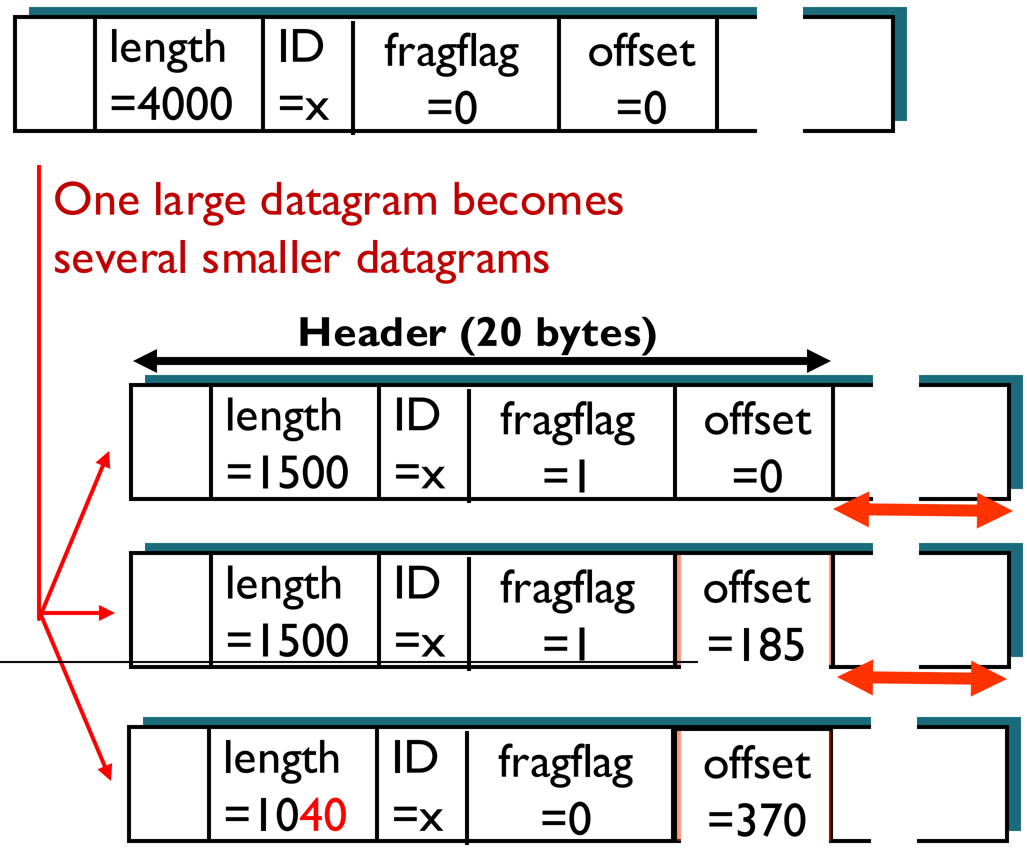
- network links have MTU (Max Transfer Unit)
  - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within network
  - one datagram split into several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes



offset = 1480/8

Note : - offset should be multiple of 8 (because encoded in 13 bits)
- for offset : do not take header into account (headers may be added when previous fragments are again fragmented)

IP Fragmentation evaluation

- Drawbacks
  - increase in CPU utilization of fragmenting router
  - TCP: one fragment dropped, resend all fragments of IP packet

- Avoid fragmentation ?
  - Fragmentation on end-nodes, not on intermediary nodes
  - Default mechanism in IPv6

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

4.2 What's inside a router

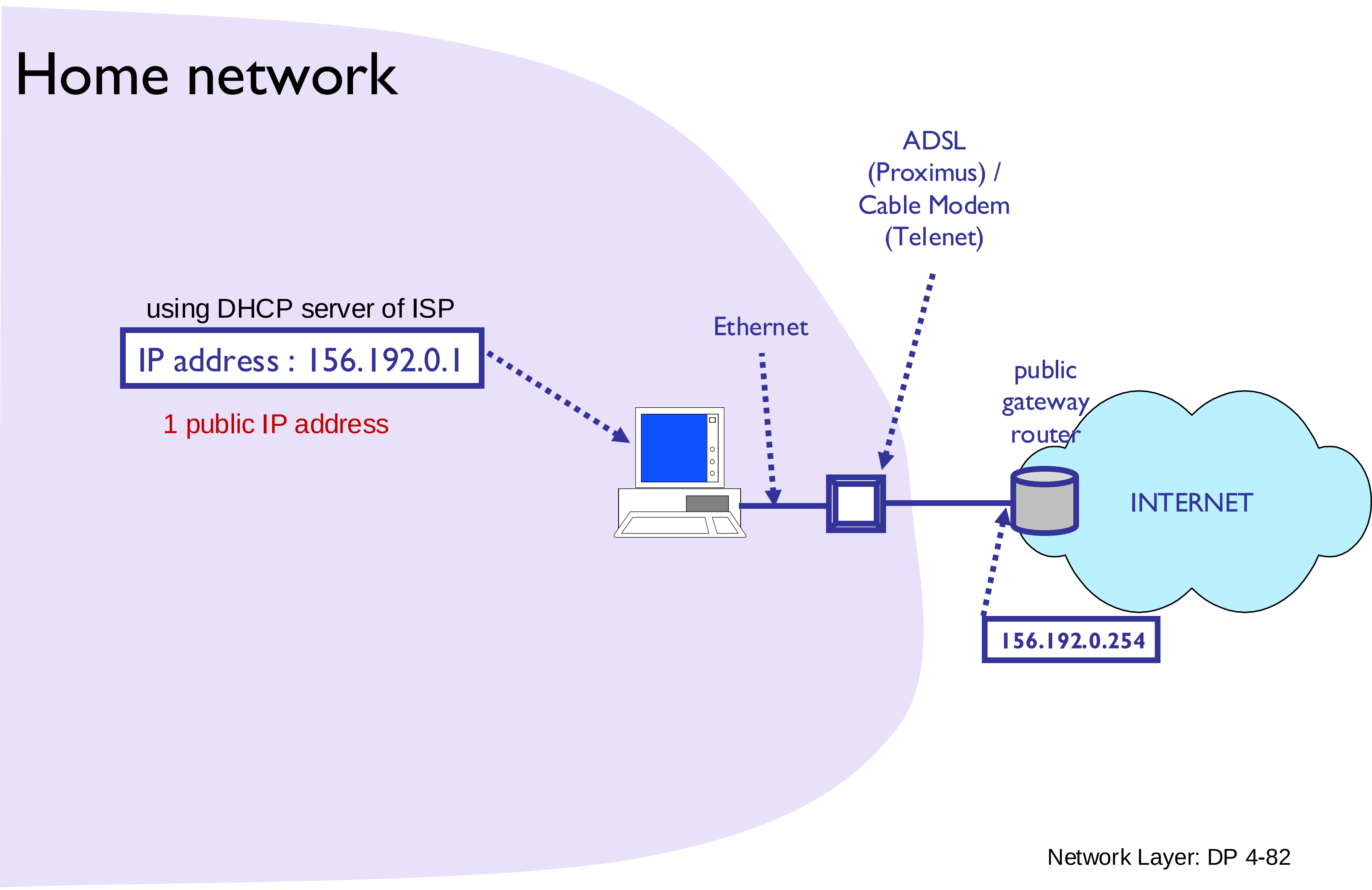
4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- **network address translation**
- IPv6

4.4 Generalized Forward and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

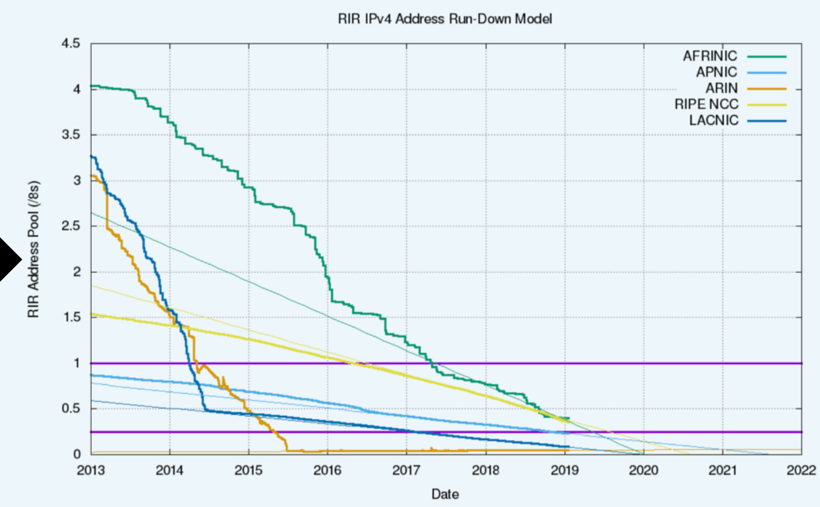
Home Network : single PC



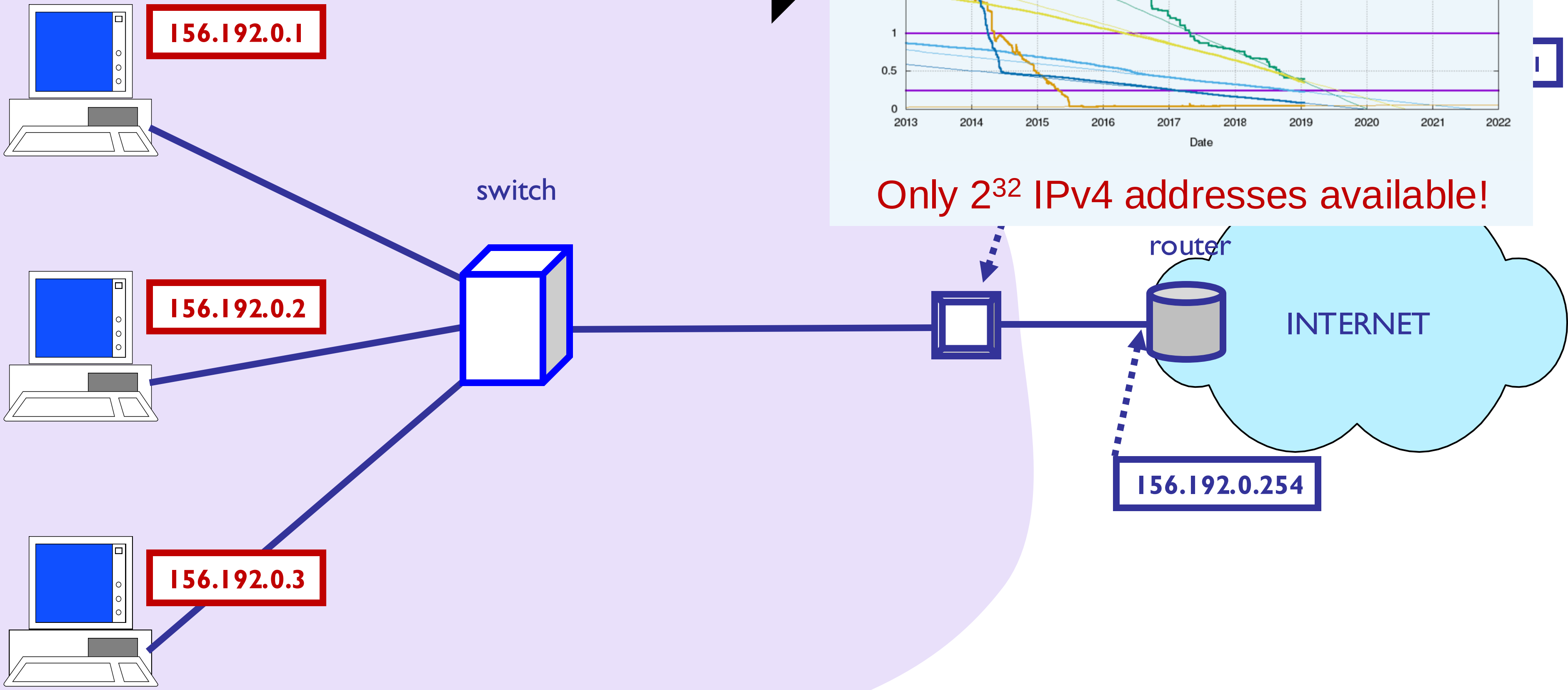
Home Network : multiple PC’s

Home network

1 public IP address per device!

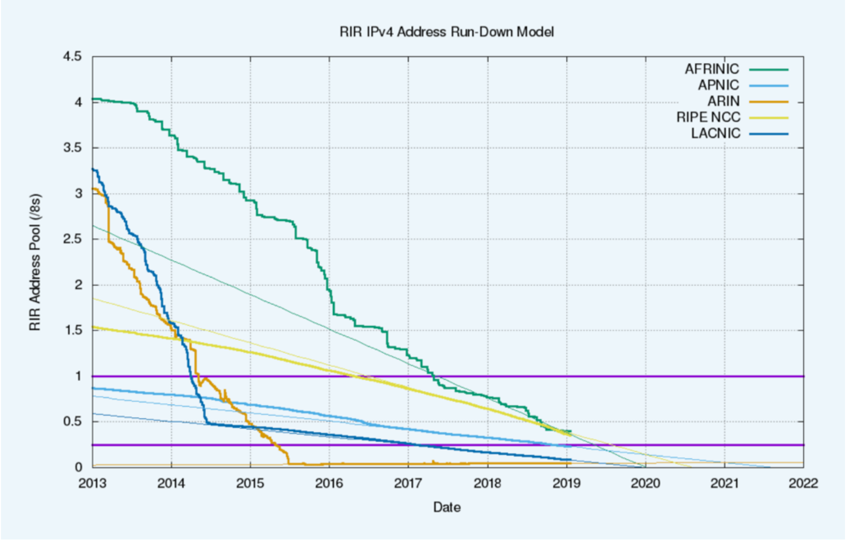


Only 2^{32} IPv4 addresses available!



NAT: Network Address Translation

- **Problem**:
  - More and more IP addresses needed due to new devices
  - IPv4 address space is exhausted



- **Facts:**
  - Many applications have a client/server architecture (client takes the initiative)
  - Many local networks are behind a single router interconnected to the Internet

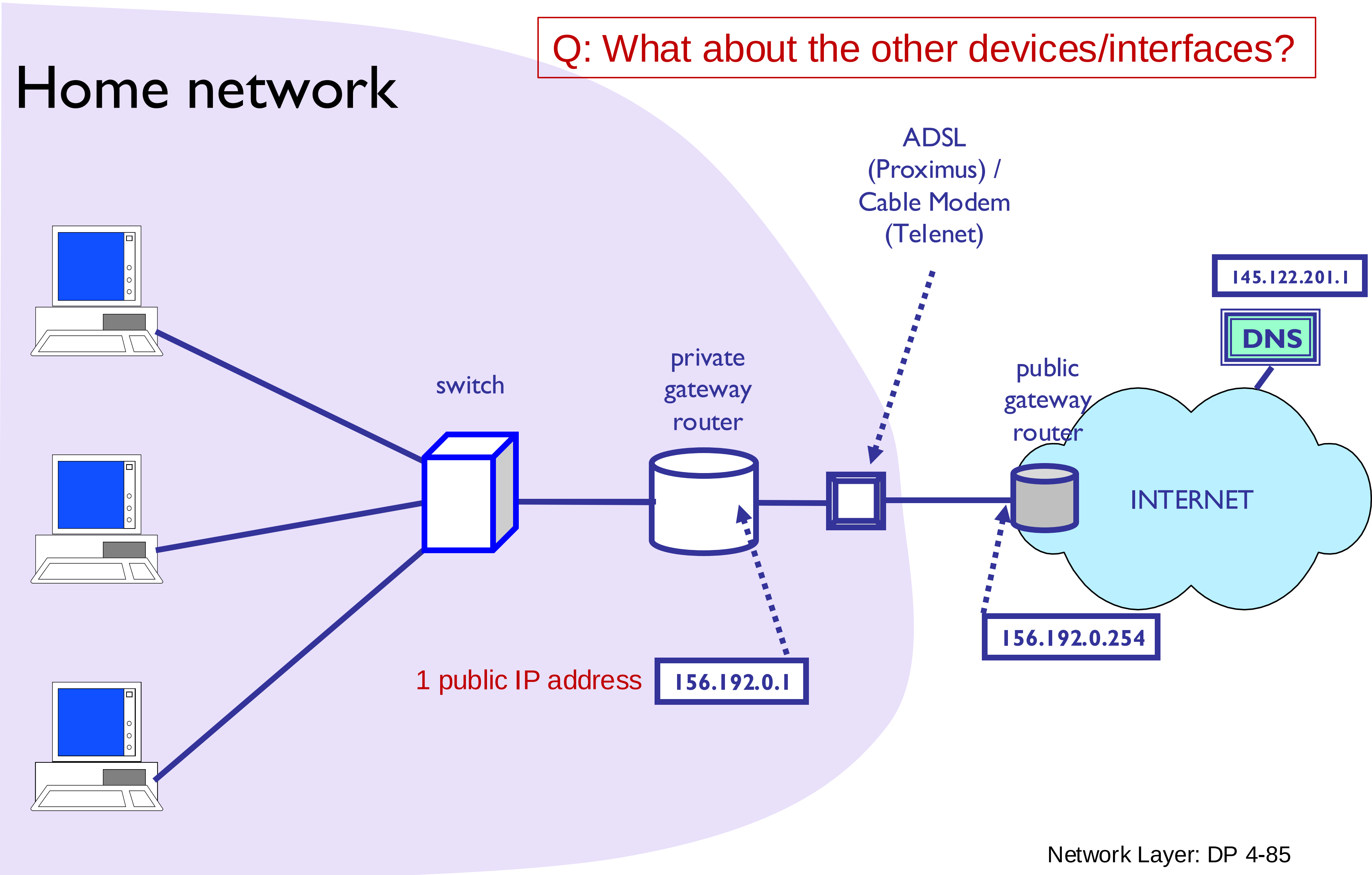
Key idea of NAT

- Use only 1 public IP address for each (home) network (access router)

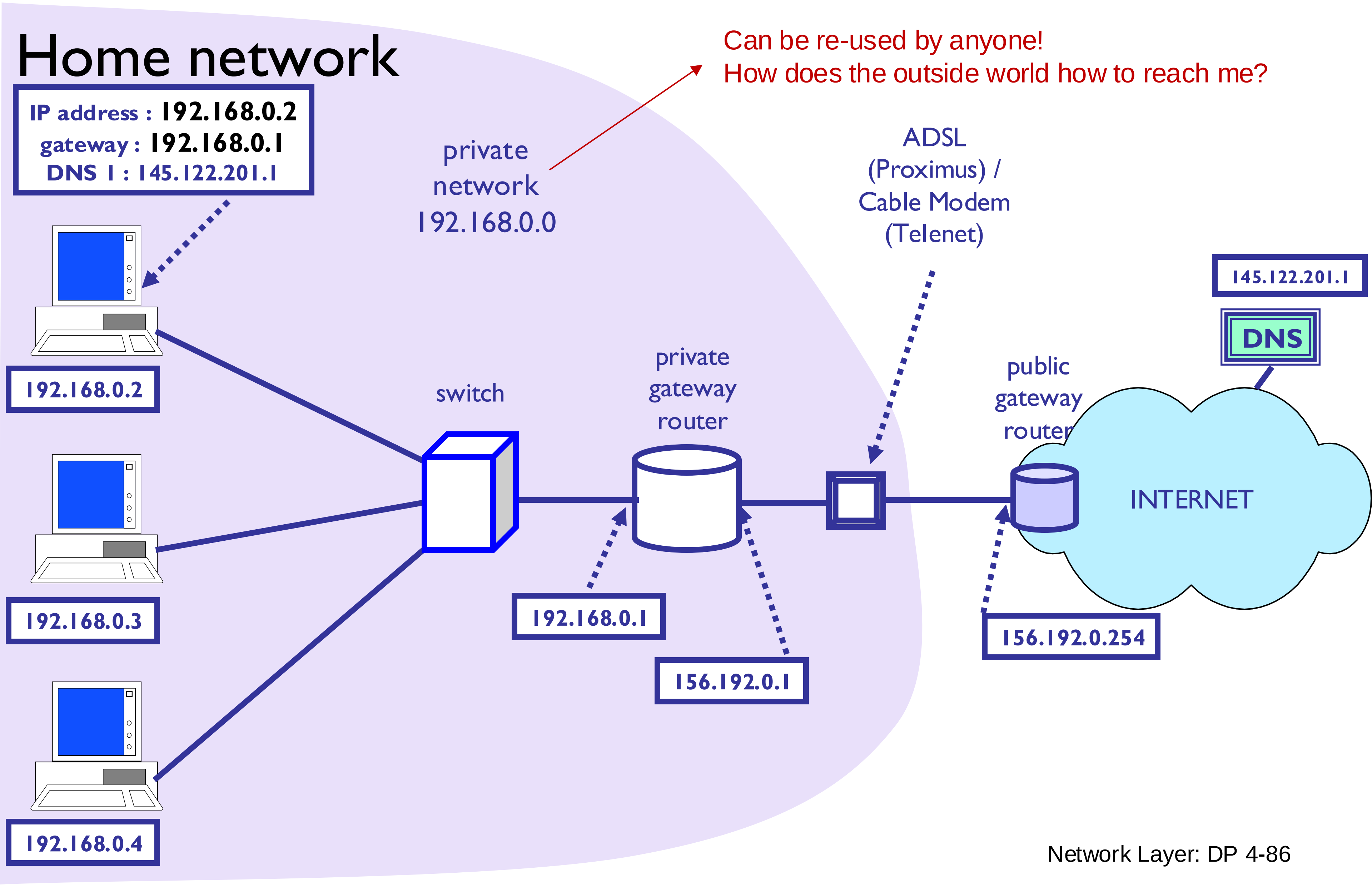
Depends on the Transport Layer used -> TCP vs. UDP port number.

NAT – Public IP address

Q: What about the other devices/interfaces?

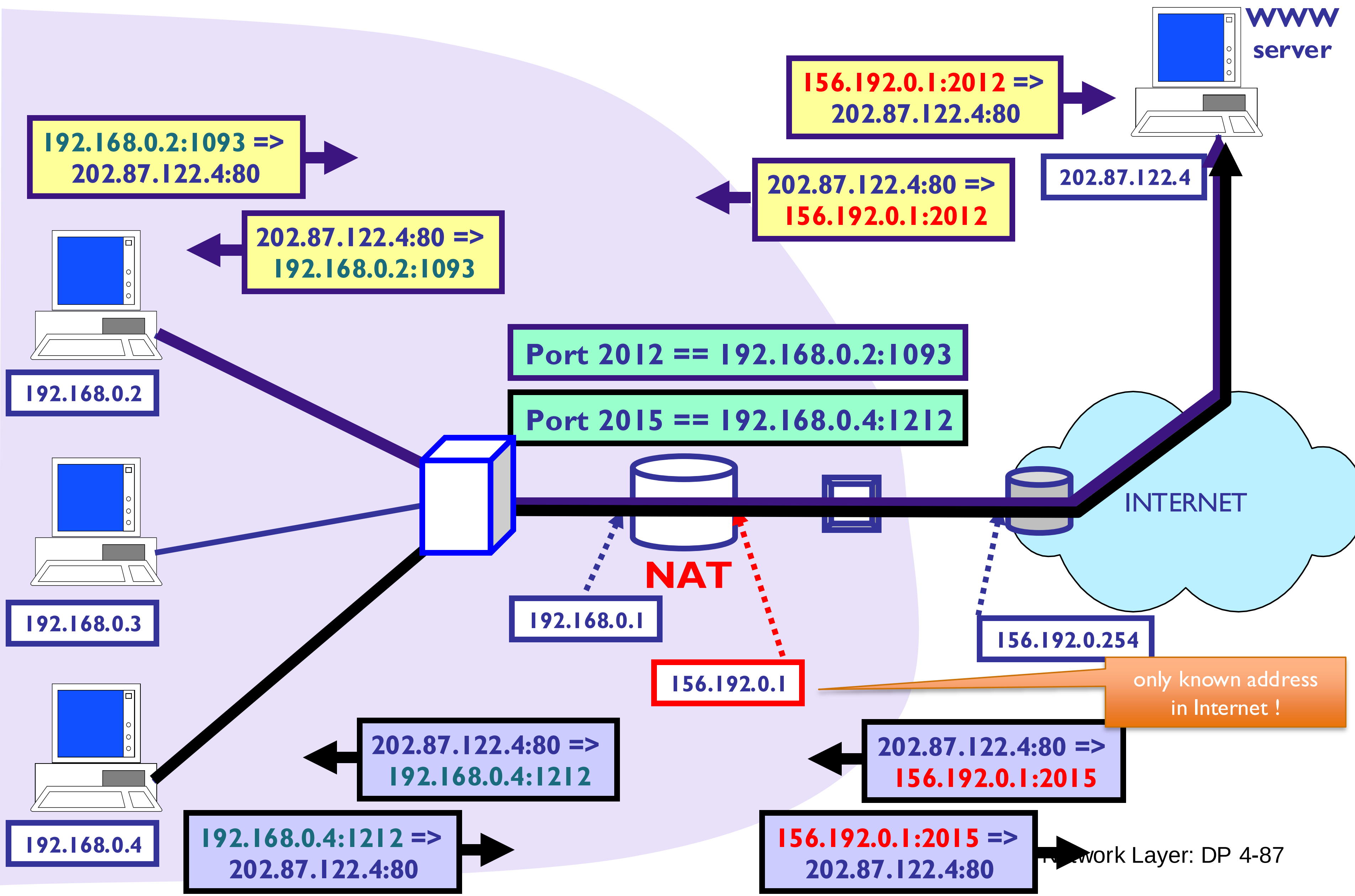


NAT - Private range IP addresses



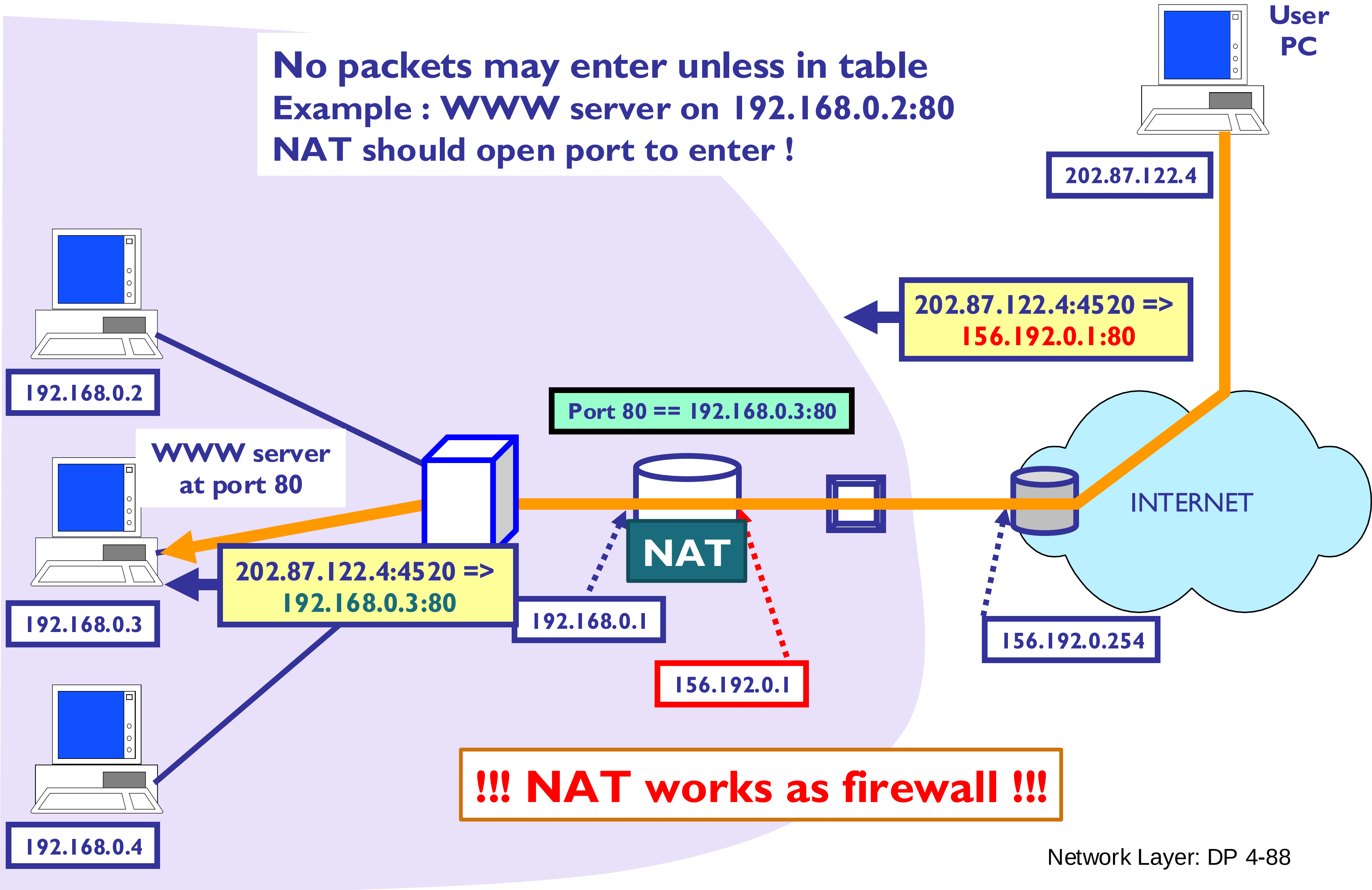
Note that NAT translates the COMBINATION of local IP address+port to port address of the external IP address (otherwise there is no way to reach the correct application/service on the targeted machine/server).

NAT - Network Address Translation



NAT : Network Address Translation

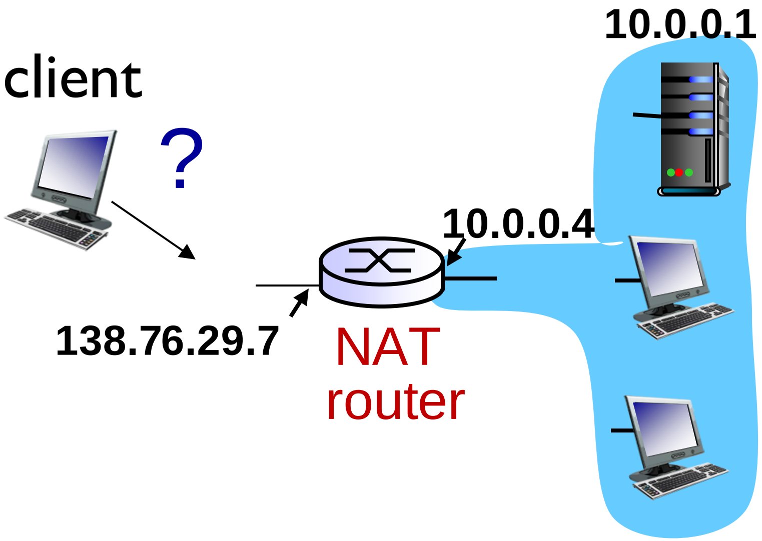
No packets may enter unless in table
Example : WWW server on 192.168.0.2:80
NAT should open port to enter !



!!! NAT works as firewall !!!

NAT traversal problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7



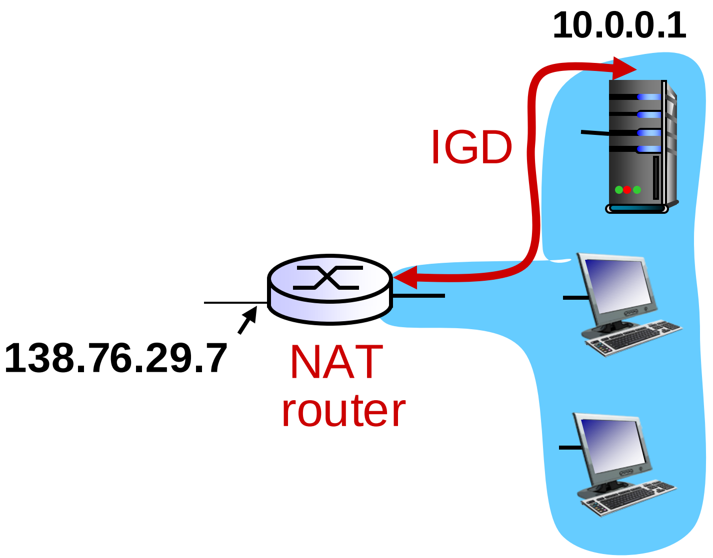
- Solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

NAT traversal problem

- Solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol.

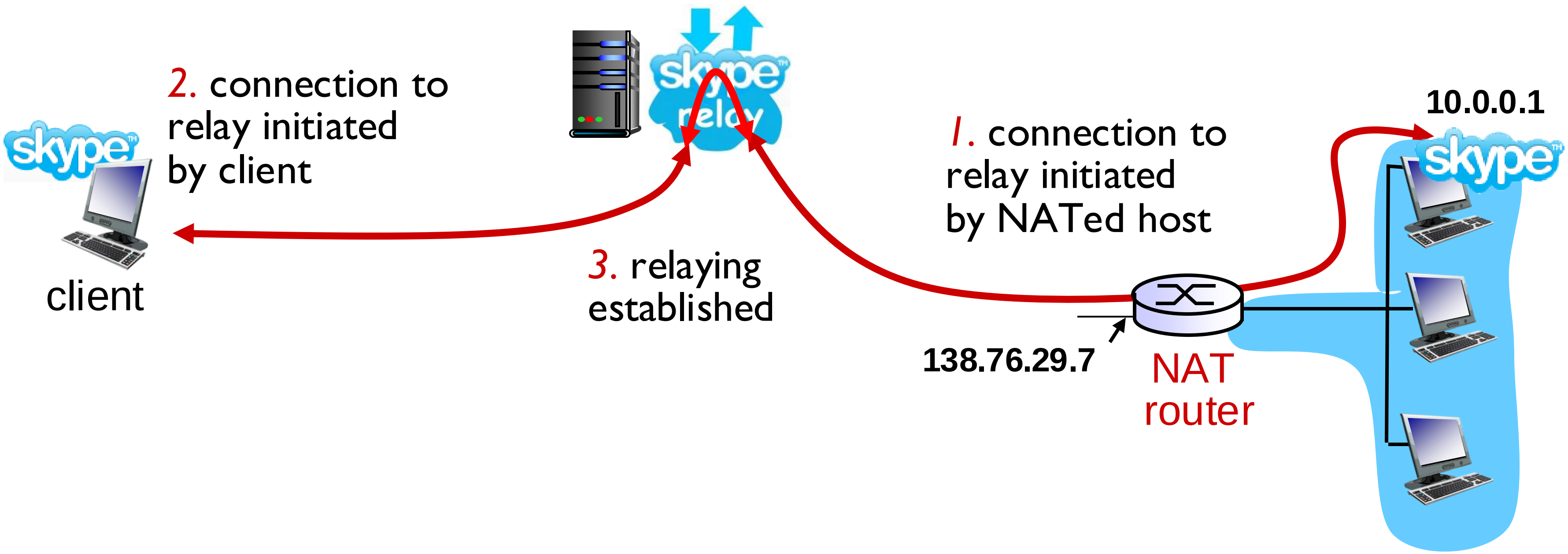
  Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  - i.e., automate static NAT port map configuration



NAT traversal problem

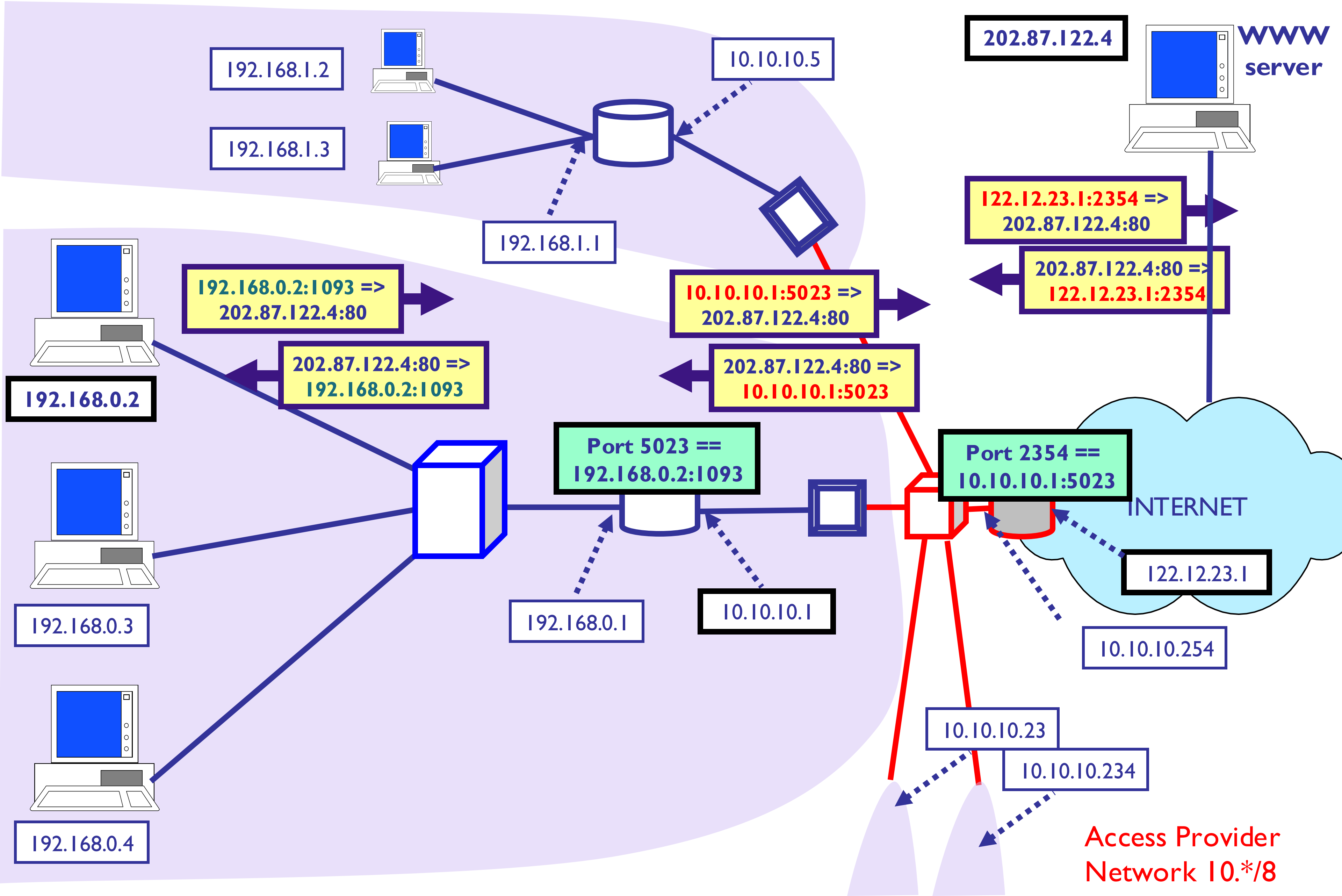
- Solution 3: relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between to connections



NAT: pro & con

- Advantages:
  - limit use of IP addresses
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).
- Drawbacks:
  - routers should only process up to layer 3 (layer violation)
  - NAT traversal problem: NAT table entries are filled based on outgoing client messages ➔ more difficult to deploy servers easily end- to-end
  - NAT possibility must be taken into account by app designers, e.g., P2P applications

Carrier-grade NAT (CGN)



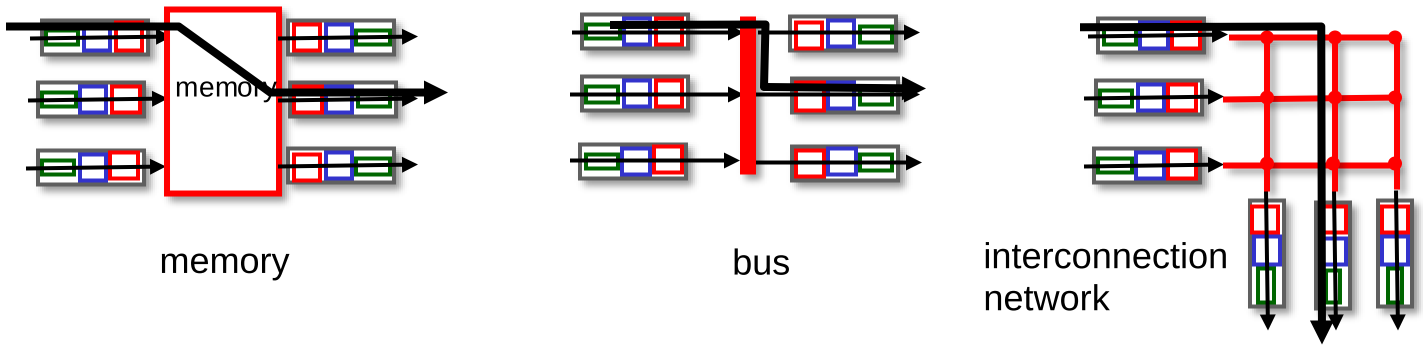
Large scale NAT

- Two levels of NAT: home and access provider
- Increase the usage of private IP addresses
- Important in countries with limited number of IP addresses (China, Russia, …)
- Possible problems:
  - ~~Identical private ranges in home and access network (conflict in home router)~~ -> IPv4 shared address space range defined 100.64/100 (RFC 6598)
  - Communication between different home networks should use access provider NAT
- Sometimes called : NAT444 or CGN

Background

Switching fabrics

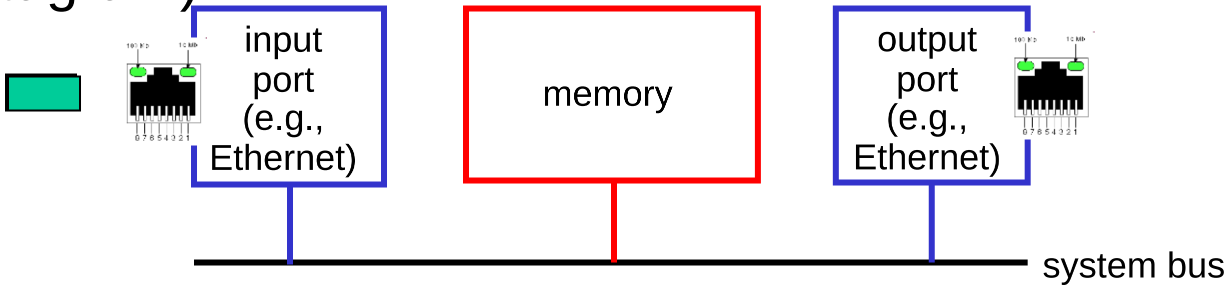
- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



Switching via memory

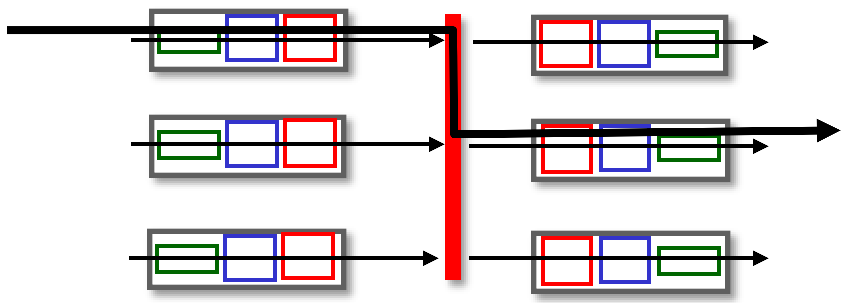
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



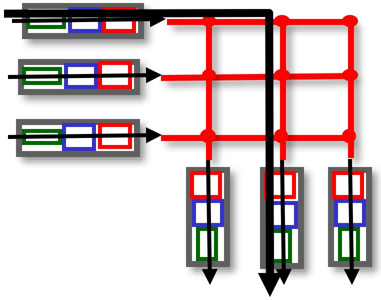
Switching via a bus

- datagram from input port memory to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access routers



Switching via interconnection network

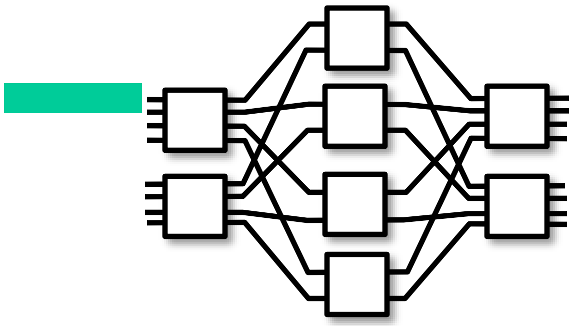
- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor



3x3 crossbar

- multistage switch: *nxn* switch from multiple stages of smaller switches

- exploiting parallelism:
  - fragment datagram into fixed length cells on entry
  - switch cells through the fabric, reassemble datagram at exit



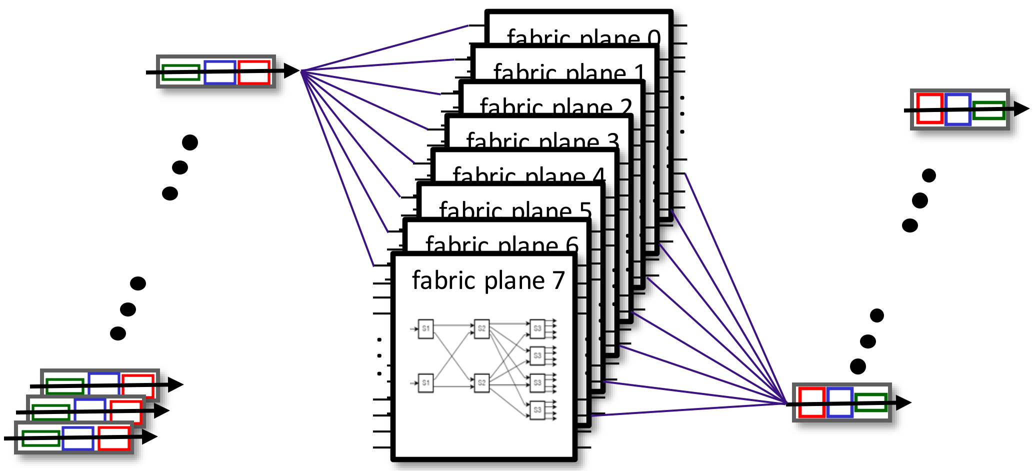
8x8 multistage switch
built from smaller-sized switches

Network Layer: 4-99

Switching via interconnection network

- scaling, using multiple switching "planes" in parallel:
  - speedup, scaleup via parallelism

- Cisco CRS router:
  - basic unit: 8 switching planes
  - each plane: 3-stage interconnection network
  - up to 100's Tbps switching capacity



Network Layer: 4-100