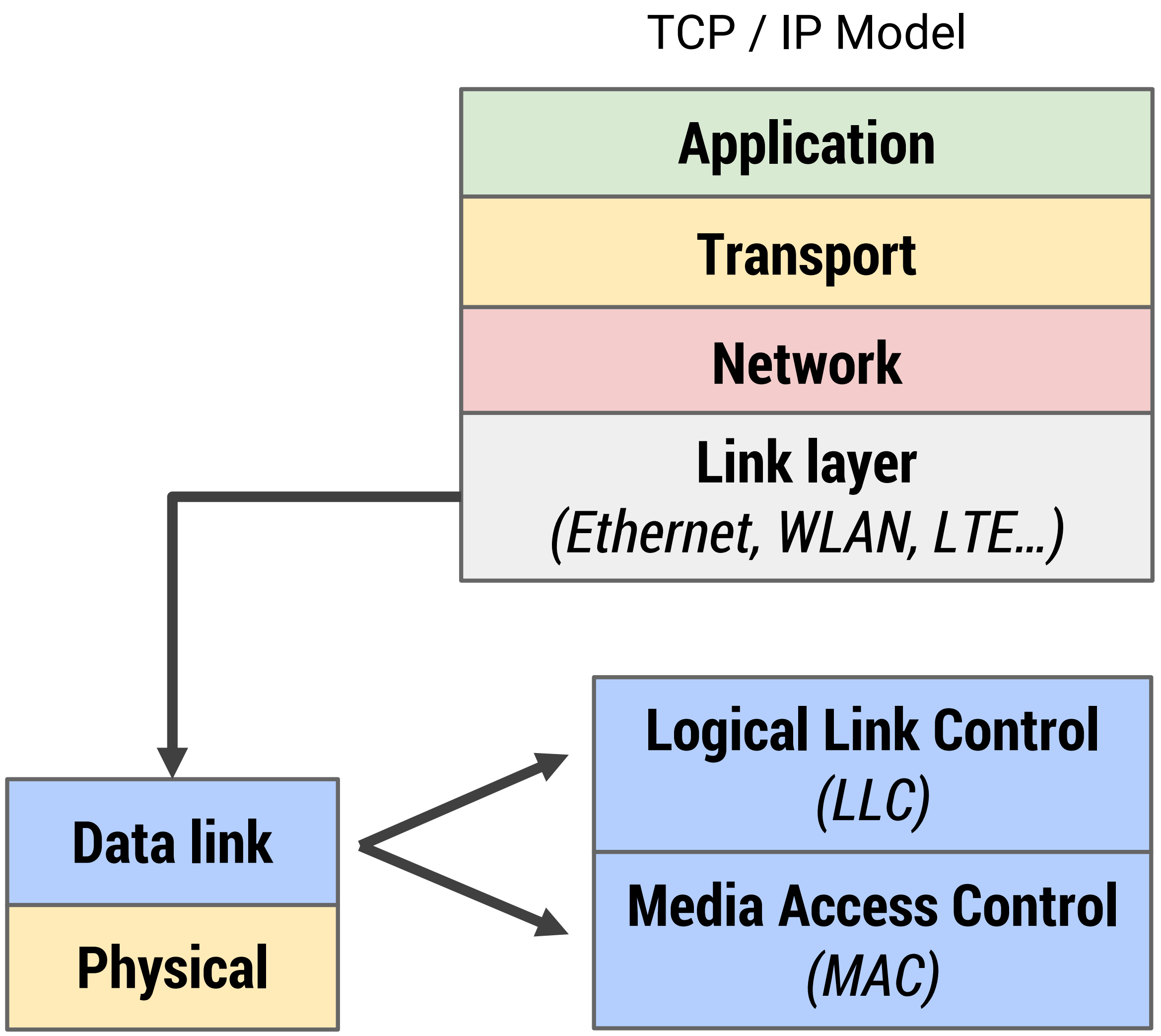
(Between) Link & Network Layer

Computer Organization and Networks 2019

Johannes Feichtner
johannes.feichtner@iaik.tugraz.at

Review: Network Basics

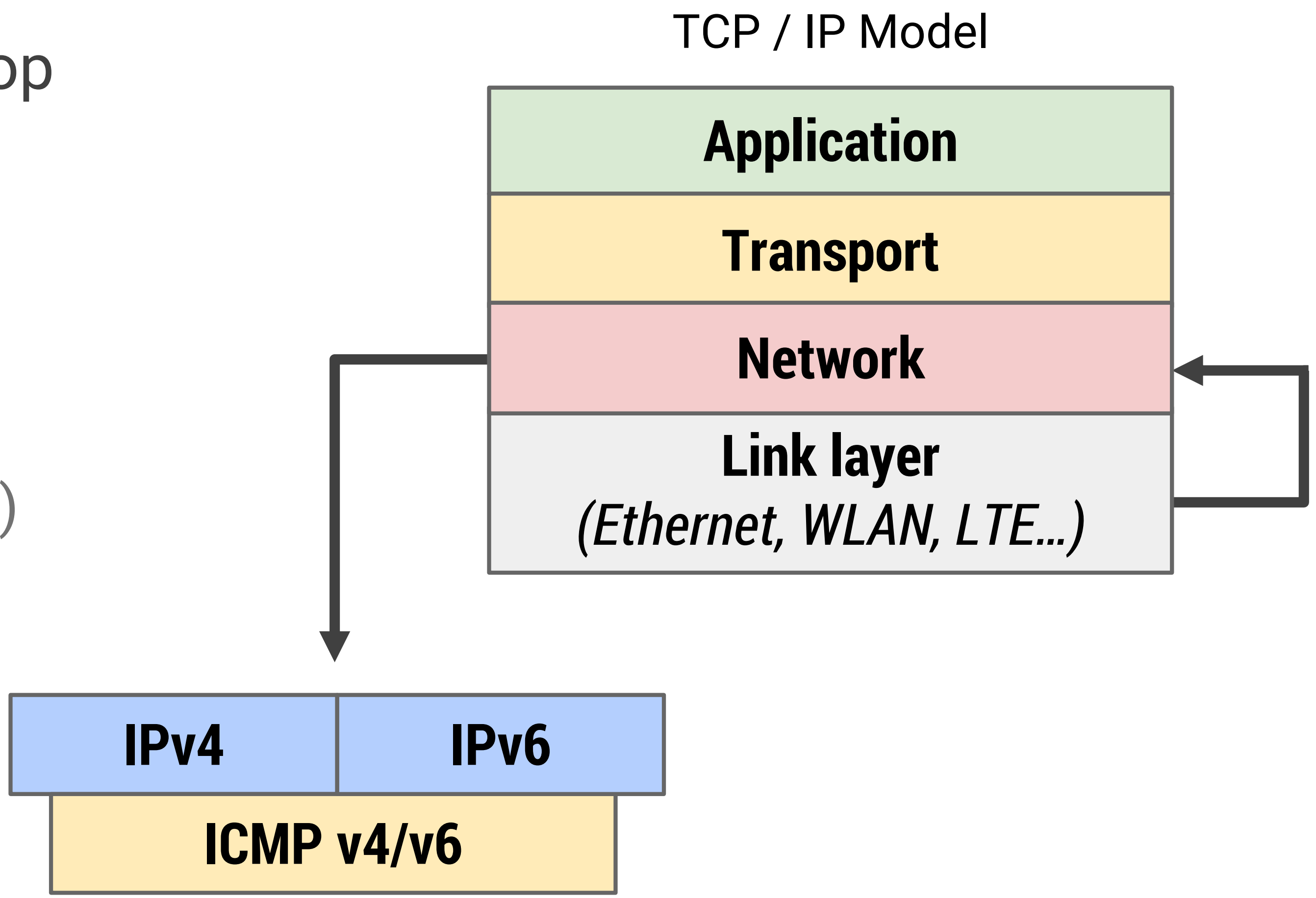
- Network Layers
- How to transfer data?
- IEEE 802
  - Logical Link Control (LLC)
  - Media Access Control (MAC)
  - Ethernet (LAN)
    - Frame Collisions
  - VLANs
- Cables, Hubs, Switches



IAIK TU Graz

Outline

- Why we need a network layer on top
  - IPv4 / IPv6
- MAC vs. IP Addresses
  - How they work together
  - Address Resolution Protocol (ARP)
- IPv4 & ICMPv4
  - Packet Structure
  - NAT & Fragmentation
- Multicasting & Routing



LANs

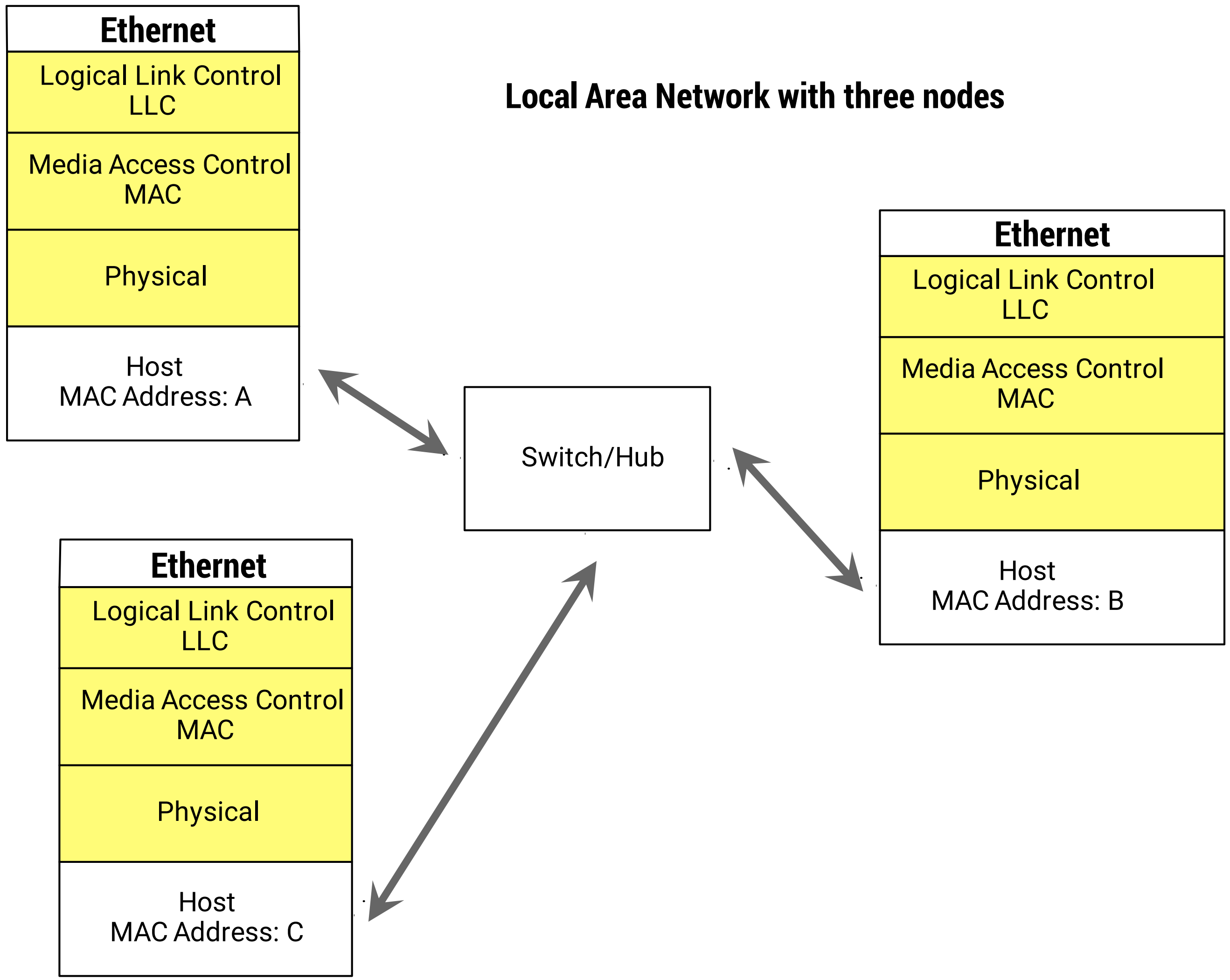
Fundamentals

We have a local network where different nodes can communicate.

So far only via the **link layer**

- Addressing via MAC addresses
  - Learned by router (SAT) via simple protocols

→ *Only Ethernet protocols needed so far!*



Media Access Control (MAC) – Addressing

IEEE 802.X

- Addressing via MAC addresses: 48 bit long → max. 2^{48} addresses
- Notation: `D4:40:F0:1B:20:80 or D4-40-F0-1B-20-80`

Network Interface Controller (NIC) specific

First 3 octets → Manufacturer
Public database: https://goo.gl/kGYaYv

- Packet / Datagram / Frame routing
  - Either via shared medium (WLAN Access Point, old Ethernet: hubs, coax cable)
  - Or via simple „routing" protocols on switches
    → „learn" new device as soon as it is sending packets

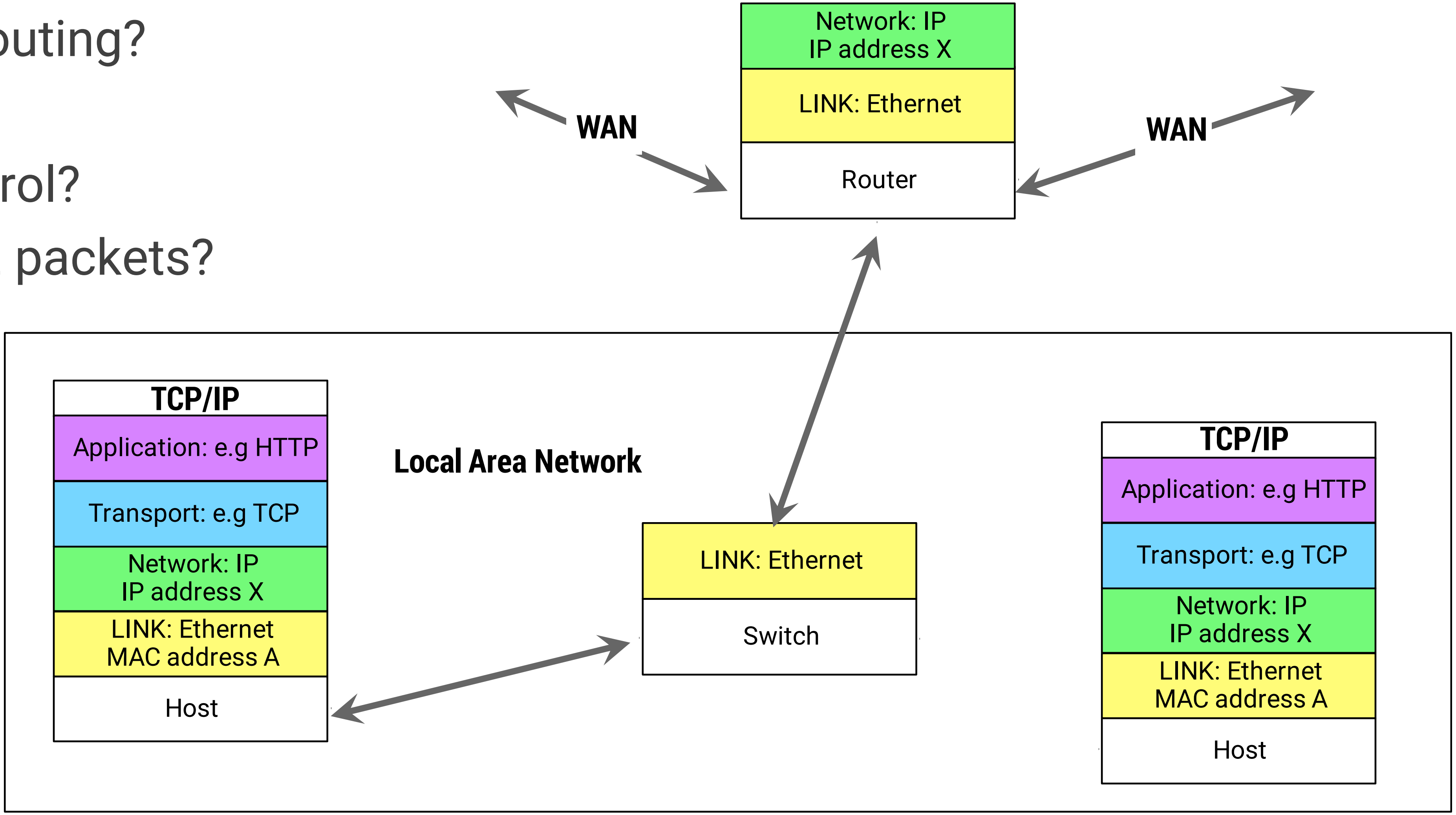
IAIK TU Graz

LANs as WANs?

Would be nice but…

- Sophisticated Routing?
- Flow control?
- Congestion control?
- Dealing with lost packets?

→ We need higher layers!



Network
(Internet) Layer

Network Layer

Purpose

Addressing across networks, routing, switching

→ Therefore, we use IPv4 / IPv6 protocols

IPv4 Topics

- Addressing via IP addresses, networks, subnet masks, …
- Data-Link layer interaction (ARP)
- Packets, fragmentation, routing, NAT, firewalls, etc.
- Routing

IPv6

- Differences / improvements since IPv4

Internet Protocol Suite

Application Layer

BGP · DHCP · DNS · FTP · HTTP · IMAP · IRC · LDAP · MGCP · NNTP · NTP · POP · RIP · RPC · RTP · SIP · SMTP · SNMP · SSH · Telnet · TLS/SSL · XMPP ·

(more)

Transport Layer

TCP · UDP · DCCP · SCTP · RSVP · ECN ·

(more)

Internet Layer

IP (IPv4, IPv6) · ICMP · ICMPv6 · IGMP · IPsec ·

(more)

Link Layer

ARP/InARP · NDP · OSPF · Tunnels (L2TP) · PPP · Media Access Control (Ethernet, DSL, ISDN, FDDI) ·
(more)

v·d·e

IAIK TU Graz

MAC – IP Interaction

Status quo – We have

- **Data link layer:** MAC addresses
- **Network layer:** IP addresses

But: *How do they work together?*

→ The data link layer does not know what to do with IP addresses

Only knows MAC addresses (including switches)

→ We need a way to map IP addresses to MAC addresses!

ARP

Address Resolution Protocol

Purpose

1. When something needs to be sent to an IP address,
   ARP is used to **ask** the local LAN for the appropriate MAC address
2. Node that has the „queried“ IP address answers with matching MAC address

→ Can then proceed to specifically deliver frames on link layer…

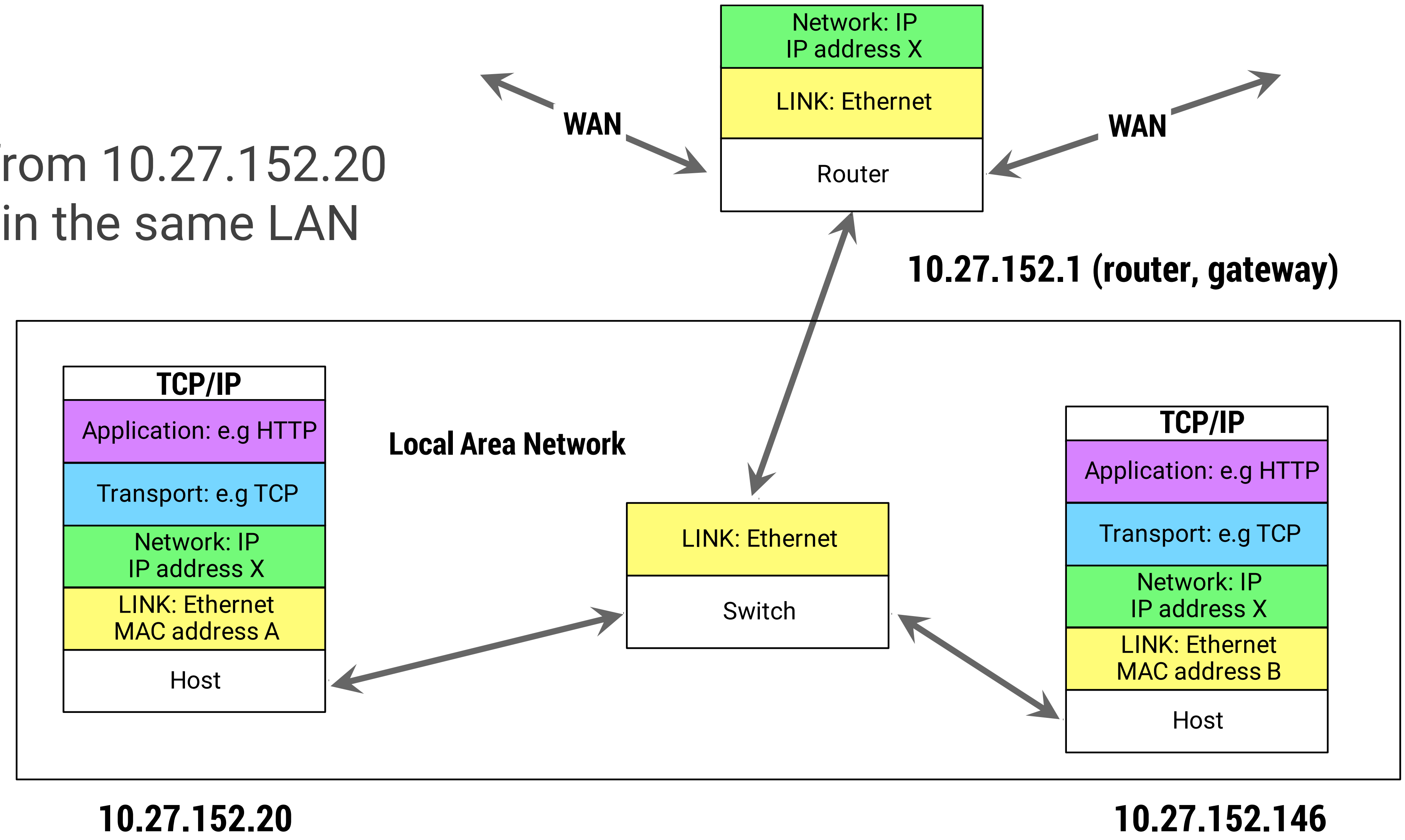
How?

Send ARP Request / Reply packets encapsulated by Ethernet frames

Within the same LAN

Scenario A

Send something from 10.27.152.20 to 10.27.152.146 in the same LAN



Within the same LAN

10.27.152.20 wants to know the MAC address of 10.27.152.146

31	29.350432000	Vmware_9b:60:02	Broadcast	ARP	60 Who has 10.27.152.146? Tell 10.27.152.20
32	29.350563000	Apple_15:ae:5a	Vmware_9b:60:02	ARP	42 10.27.152.146 is at c4:2c:03:15:ae:5a

```
Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  [Is gratuitous: False]
  Sender MAC address: Vmware_9b:60:02 (00:50:56:9b:60:02)
  Sender IP address: 10.27.152.20 (10.27.152.20)
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.27.152.146 (10.27.152.146)
```

```
Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  [Is gratuitous: False]
  Sender MAC address: Apple_15:ae:5a (c4:2c:03:15:ae:5a)
  Sender IP address: 10.27.152.146 (10.27.152.146)
  Target MAC address: Vmware_9b:60:02 (00:50:56:9b:60:02)
  Target IP address: 10.27.152.20 (10.27.152.20)
```

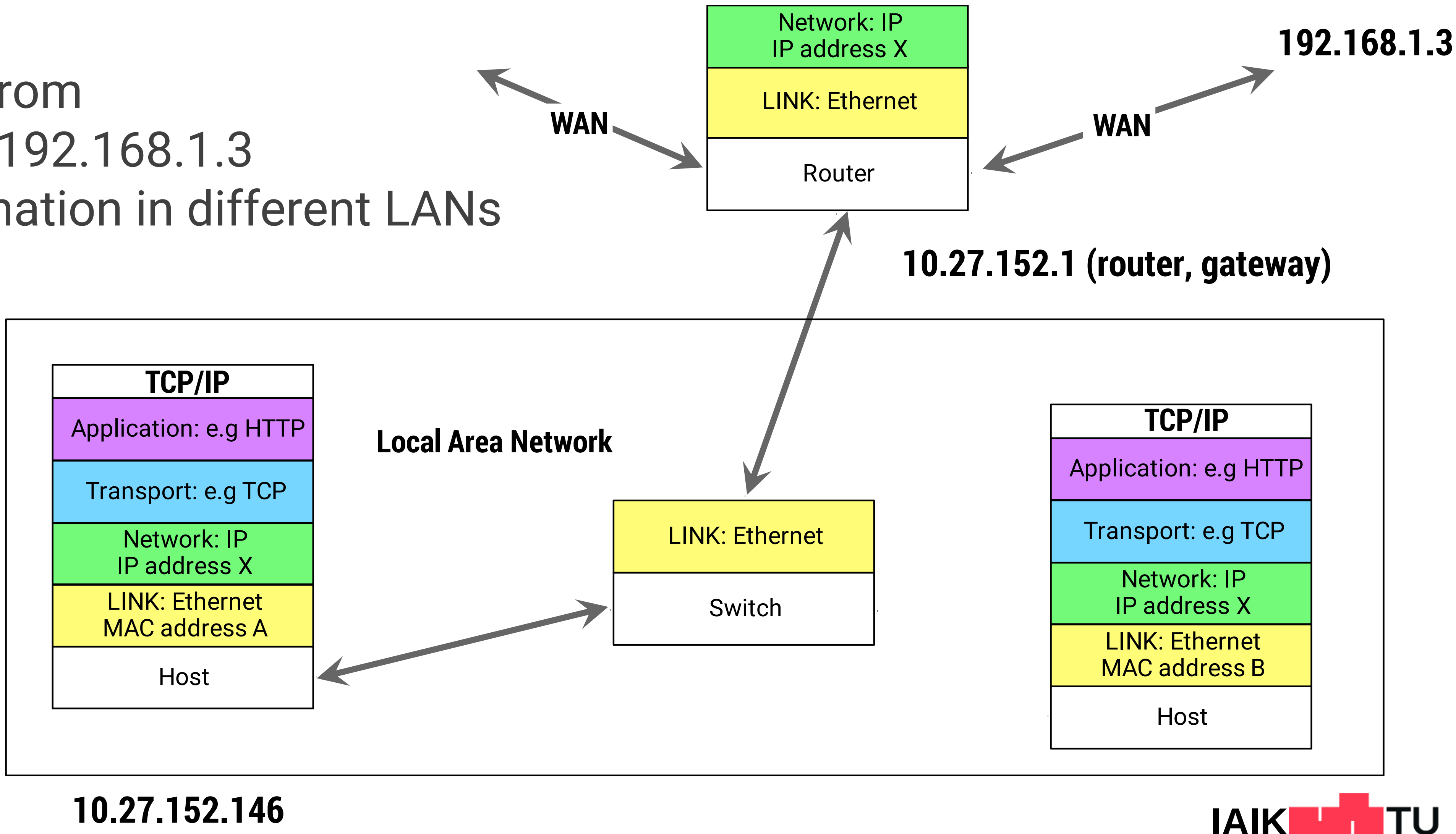
Internet Protocol (IPv4) over Ethernet ARP packet		
octet offset	**0**	**1**
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

Source: https://goo.gl/5MWZAT

Via Gateway / Router

Scenario B

Send something from
10.27.152.146 to 192.168.1.3
→ Sender & destination in different LANs



IAIK TU Graz

Via Gateway / Router

Workflow

- ARP asks for IP address 192.168.1.2 in local LAN
  → No machine with this address existing

- We have a node that acts as a gateway to other networks
  - 192.168.1.2 might still not be directly reachable but
    it is likely that the gateway knows other gateways (routers) that know…

- So even if the gateway is not representing 192.168.1.2,
  its MAC address will be used to send something to 192.168.1.2

How do we know the MAC address of the gateway?

→ Typically determined when network interface is initialized

ARP Cache

Purpose

Information gathered by ARP is stored in ARP Cache

→ Reduces communication overhead

```
arp -a
Interface: 10.27.152.146 --- 0x3
  Internet address      Physical address      Type
  10.27.152.1           f4-ac-c1-67-e4-..     dynamic
  10.27.152.9           00-50-56-9b-72-..     dynamic
  10.27.152.10          00-15-17-61-e7-..     dynamic
  10.27.152.20          00-50-56-9b-54-..     dynamic
  10.27.152.29          00-50-56-9b-34-..     dynamic
  ...
```

→ *What happens if MAC address changes but IP stays the same?*

E.g. standby machine takes over IP address of another one…

IAIK TU Graz

Gratuitous ARP Messages

= Announcements that say a MAC address belongs to an IP address

Idea

- Update other hosts‘ mapping when sender IP or MAC address changed
- Typically done using computer startup
  - Detect IP conflict: If you receive an ARP request with source IP = your own
  - Inform switch of MAC address on given switch port
  - Also to avoid problems with old MAC addresses (virtualization)
- Not intended to solicit a reply

gra•tu•i•tous |grəˈt(y)o͞oitəs|
adjective
1 uncalled for; lacking good reason; unwarranted : *gratuitous violence.*
2 given or done free of charge : *solicitors provide a form of gratuitous legal advice.*

Gratuitous ARP Messages

Method A

By broadcasting an ARP **request**

- Target IP = Sender IP address
  set to value of machine that has changed the MAC address

Standard ARP Request

```
Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  [Is gratuitous: False]
  Sender MAC address: Vmware_9b:60:02 (00:50:56:9b:60:02)
  Sender IP address: 10.27.152.20 (10.27.152.20)
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.27.152.146 (10.27.152.146)
```

Gratuitous ARP Request

```
Address Resolution Protocol (request/gratuitous ARP)
  Hardware type: Ethernet (1)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  [Is gratuitous: True]
  Sender MAC address: IntelCor_4d:34:be (00:15:17:4d:34:be)
  Sender IP address: 10.27.152.159 (10.27.152.159)
  Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
  Target IP address: 10.27.152.159 (10.27.152.159)
```

Gratuitous ARP Messages

Method B

By broadcasting an ARP **reply**

Standard ARP Reply

```
Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  [Is gratuitous: False]
  Sender MAC address: Apple_15:ae:5a (c4:2c:03:15:ae:5a)
  Sender IP address: 10.27.152.146 (10.27.152.146)
  Target MAC address: Vmware_9b:60:02 (00:50:56:9b:60:02)
  Target IP address: 10.27.152.20 (10.27.152.20)
```

Gratuitous ARP Reply

→ Same as standard but with:

```
Target IP   = Sender IP address
Target MAC  = Sender MAC address
```

Regardless of method → Receivers replace cached entries with new mapping!

Security Problem!

IAIK TU Graz

Network Layer – IPv4

Properties

- Best-effort delivery
  - Service considered unreliable by design
  - No central monitoring that could detect failures
- Dynamic routing → each packet treated independently
  - Data corruption, packet loss, out-of-order delivery etc can happen for every single packet!
- No flow and congestion control
  - Added by higher layers, e.g. TCP Sliding Window
- No security features (except IPsec extension)

Internet Protocol Suite
Application Layer
BGP · DHCP · DNS · FTP · HTTP · IMAP · IRC · LDAP · MGCP · NNTP · NTP · POP · RIP · RPC · RTP · SIP · SMTP · SNMP · SSH · Telnet · TLS/SSL · XMPP ·
(more)
Transport Layer
TCP · UDP · DCCP · SCTP · RSVP · ECN ·
(more)
Internet Layer
IP (IPv4, IPv6) · ICMP · ICMPv6 · IGMP · IPsec ·
(more)
Link Layer
ARP/InARP · NDP · OSPF · Tunnels (L2TP) · PPP · Media Access Control (Ethernet, DSL, ISDN, FDDI) · (more)
v · d · e

IAIK TU Graz

Evolution

RFC 791

Idea

Create simple network layer, move intelligence to clients (end points)

→ Extendable by creating new applications on-top layers

→ Connect different networks, technologies (radio, satellite, Ethernet) with different characteristics (loss rate, delays, transmission rates, etc.)

→ IPv4: „Internet Protocol“ (1981) See: https://goo.gl/LFCmD7

- Connection-less protocol for use on packet-switched networks
- First version that was used world-wide as it was deployed in ARPANET in 1983

Note: At that time security aspects were not considered at all!

IPv4 Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				Internet Header Length				Differentiated Services Code Point						Explicit Congestion Notification		Total Length															
4	32	Identification																Flags			Fragment Offset												
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

- <u>Version:</u> IP protocol number → 4
- <u>Internet Header Length:</u> Size of IP header in 32-bit words
- <u>Differentiated Services Code Point:</u> Used to separate traffic into classes for prioritization, e.g. Voice over IP (VoIP)

IPv4 Header

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="4">Internet Header Length</td><td colspan="6">Differentiated Services Code Point</td><td colspan="2">Explicit Congestion Notification</td><td colspan="16">Total Length</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td>8</td><td>64</td><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td>12</td><td>96</td><td colspan="32">Source IP Address</td></tr>
<tr><td>16</td><td>128</td><td colspan="32">Destination IP Address</td></tr>
<tr><td>20</td><td>160</td><td colspan="32">Options (if IHL > 5)</td></tr>
</table>

- Explicit Congestion Notification: Notification about congestion
- Total Length: Packet size of header (20-60 bytes) + data (0-65.535 bytes)
  → Size between 20-65.535 bytes

Fragmentation!

- Identification: Identify fragmented packets

IAIK TU Graz

IPv4 Header

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="4">Internet Header Length</td><td colspan="6">Differentiated Services Code Point</td><td colspan="2">Explicit Congestion Notification</td><td colspan="16">Total Length</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td>8</td><td>64</td><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td>12</td><td>96</td><td colspan="32">Source IP Address</td></tr>
<tr><td>16</td><td>128</td><td colspan="32">Destination IP Address</td></tr>
<tr><td>20</td><td>160</td><td colspan="32">Options (if IHL > 5)</td></tr>
</table>

- Flags: Bit 1 set = DF (Don‘t fragment), Bit 2 set = MF (More fragments)
- Fragment Offset: Offset of current fragment relative to unfragmented packet
- Time to Live (TTL): Hop count, if 0 → Router discards packet
- Protocol: Next layer protocol used in data portion

See: https://goo.gl/x0a8lo

IPv4 Header

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="4">Internet Header Length</td><td colspan="6">Differentiated Services Code Point</td><td colspan="2">Explicit Congestion Notification</td><td colspan="16">Total Length</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td>8</td><td>64</td><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td>12</td><td>96</td><td colspan="32">Source IP Address</td></tr>
<tr><td>16</td><td>128</td><td colspan="32">Destination IP Address</td></tr>
<tr><td>20</td><td>160</td><td colspan="32">Options (if IHL > 5)</td></tr>
</table>

- Header checksum: 16-bit checksum of IP header
  Routers verify it → on mismatch, packet is dropped without notification
- Source IP adddress: 32-bit IPv4 address of sender
- Destination IP adddress: 32-bit IPv4 address of receiver

IPv4 Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				Internet Header Length				Differentiated Services Code Point						Explicit Congestion Notification		Total Length															
4	32	Identification																Flags			Fragment Offset												
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

- Options: Rarely used, e.g. for debugging
- Data: Interpreted based on number in „Protocol“ header field
  - 1 : ICMP
  - 6 : TCP
  - 17 : UDP

→ For TCP / UDP this is the transport layer!

IPv4 Header

Wireshark Example

```
v Internet Protocol Version 4, Src: 192.168.0.13, Dst: 194.232.104.109
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x4490 (17552)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
  > Header checksum: 0xca34 [validation disabled]
    Source: 192.168.0.13
    Destination: 194.232.104.109
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
> Transmission Control Protocol, Src Port: 61451 (61451), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 0
```

IPv4 Addressing

- 32-bit addresses → max. 2^{32} addresses
- Dotted-decimal notation: 192.168.12.4 → 4 octets with values between 0-255

- Each network interface needs a unique IP address
  - Except if NAT is used
  - IPs are associated with a network interface, not the host / router
  - Cannot be assigned arbitrarily, always needs a subnet specification

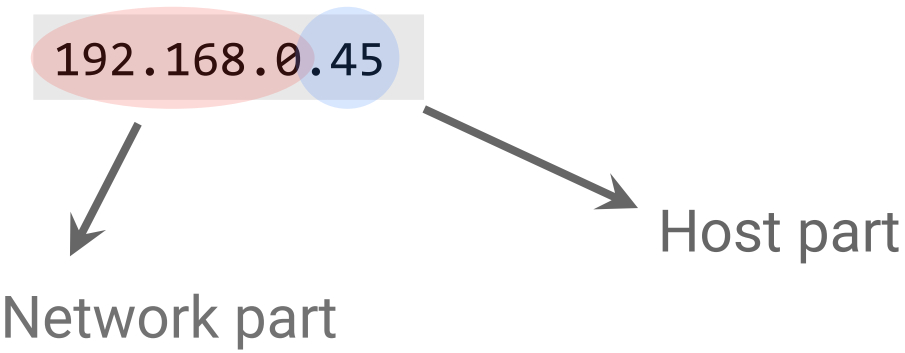
```
ip -4 addr
inet 123.243.144.204/26 brd 123.243.144.255 scope global eth0

ifconfig
inet addr:123.243.144.204  Bcast:123.243.144.255  Mask:255.255.255.192
```

IPv4 Subnets

Purpose

- Hosts sharing the same subnet do not need a router
- They can communicate via data-link layer (Ethernet, WLAN, ARP!!)

Modelling subnets



Every IP address consists of network and host part

→ This is defined via the subnet mask

IPv4 Subnets

Why subnet masks?

- Group hosts into subnets
- Route entries for each IP address on every router not feasible!

→ Organizations get assigned blocks of IP addresses

Notation via IPv4 network mask *or* Classless Inter-Domain Routing (CIDR)

Examples

∑ of set bits → CIDR

Dot-decimal notation	Binary form	CIDR	No. of addresses
255.255.255.0	11111111.11111111.11111111.00000000	/24	256
255.255.255.224	11111111.11111111.11111111.11100000	/27	32

$2^{(32-\text{ network part})}$ → No. of addresses

IPv4 Subnets

Before 1993…

Class	Leading bits	Size of *network number* bit field	Size of *rest* bit field	Number of networks	Addresses per network	Total addresses in class	Start address	End address
Class A	0	8	24	128 (2^7)	16,777,216 (2^{24})	2,147,483,648 (2^{31})	0.0.0.0	127.255.255.255
Class B	10	16	16	16,384 (2^{14})	65,536 (2^{16})	1,073,741,824 (2^{30})	128.0.0.0	191.255.255.255
Class C	110	24	8	2,097,152 (2^{21})	256 (2^8)	536,870,912 (2^{29})	192.0.0.0	223.255.255.255
Class D (multicast)	1110	not defined	not defined	not defined	not defined	268,435,456 (2^{28})	224.0.0.0	239.255.255.255
Class E (reserved)	1111	not defined	not defined	not defined	not defined	268,435,456 (2^{28})	240.0.0.0	255.255.255.255

Source: https://goo.gl/0aP0Os

- Only few network classes
  - Subnet masks with /8, /16, or /24 but nothing in between
  - Classful addressing: /8 = Class A, /16 = Class B, /24 = Class C
- Problem: Granularity of IP address distribution
  - Class C has 254 usable hosts, Class B has 65.354 → waste of resources!

Now: Classless Inter-Domain Routing (CIDR) RFC 1519

IPv4 Subnets

But although CIDR is predominant now…

Special address ranges are partially based on old classful routing!

Reserved for

- Maintenance routing tables
- Multicast traffic
- Private networks
- etc

Range	Description	Reference
0.0.0.0/8	Current network (only valid as source address)	RFC 6890
10.0.0.0/8	Private network	RFC 1918
100.64.0.0/10	Shared Address Space	RFC 6598
127.0.0.0/8	Loopback	RFC 6890
169.254.0.0/16	Link-local	RFC 3927
172.16.0.0/12	Private network	RFC 1918
192.0.0.0/24	IETF Protocol Assignments	RFC 6890
192.0.2.0/24	TEST-NET-1, documentation and examples	RFC 5737
192.88.99.0/24	IPv6 to IPv4 relay	RFC 3068
192.168.0.0/16	Private network	RFC 1918
198.18.0.0/15	Network benchmark tests	RFC 2544
198.51.100.0/24	TEST-NET-2, documentation and examples	RFC 5737
203.0.113.0/24	TEST-NET-3, documentation and examples	RFC 5737
224.0.0.0/4	IP multicast (former Class D network)	RFC 5771
240.0.0.0/4	Reserved (former Class E network)	RFC 1700
255.255.255.255	Broadcast	RFC 919

Source: https://goo.gl/e5NQLS

IPv4 Private Networks

Name	Address range	Number of addresses	*Classful* description	Largest CIDR block
24-bit block	10.0.0.0–10.255.255.255	16 777 216	Single Class A	10.0.0.0/8
20-bit block	172.16.0.0–172.31.255.255	1 048 576	Contiguous range of 16 Class B blocks	172.16.0.0/12
16-bit block	192.168.0.0–192.168.255.255	65 536	Contiguous range of 256 Class C blocks	192.168.0.0/16

Source: https://goo.gl/e5NQLS

- Advantage: These addresses are not explicitly registered to some company
  - Everybody may use them in internal networks
  - Companies often would not get enough public IPv4 addresses, especially now that officially all IPv4 blocks are assigned

- Not routed on Internet
  - Need „translation“ to Internet addresses → **How?**

IPv4 Subnets

Example

Network with max. 254 hosts, e.g. 192.168.5.1 to 192.168.5.254

Network: 192.168.5.0
Subnet mask 255.255.255.0 } **= 192.168.5.0/24**

- Now only one route entry is needed for 254 hosts
- Subnet mask tells router which bits of IP address to match to decide route

	Dot-decimal notation	Binary form
IP address	192.168.5.130	11000000.10101000.00000101.10000010
Subnet mask	255.255.255.0	11111111.11111111.11111111.00000000
Network part	192.168.5.0	11000000.10101000.00000101.00000000
Host part	0.0.0.130	00000000.00000000.00000000.10000010

IPv4 Special Addresses

- First address in network is network identifier
- Last address in network is broadcast address of network

→ In every subnet two addresses not usable!

Example

CIDR: /28

	Dot-decimal notation	Binary form	Operation
IP address	10.43.8.67	00001010.00101011.00001000.01000011	
Subnet mask	255.255.255.240	11111111.11111111.11111111.11110000	
Network part	10.43.8.64	00001010.00101011.00001000.01000000	Logical AND
Broadcast address	10.43.8.79	00001010.00101011.00001000.01001111	Logical OR on inverted subnet mask

→ Subnet range: 10.43.8.64 – 10.43.8.79
Assignable: 10.43.8.65 – 10.43.8.78

IPv4 Subnets

More Examples

Network: 192.168.1.0
Subnet mask: 255.255.255.0

CIDR: 192.168.1.0/24
Hosts: 192.168.1.1 – 192.168.1.254
Broadcast: 192.168.1.255

Network: 192.168.0.0
Subnet mask: 255.255.0.0

CIDR: 192.168.0.0/16
Hosts: 192.168.0.1 – 192.168.255.254
Broadcast: 192.168.255.255

Network: 192.168.1.4
Subnet mask: 255.255.255.252

CIDR: 192.168.1.4/30
Hosts: 192.168.1.5 – 192.168.1.6
Broadcast: 192.168.1.7

IPv4 Subnet Routing

Example

- Router: 192.168.5.0/24 via 172.20.3.5 (= next router)
- Packet arrives at router 172.20.3.5 with destination 192.168.5.130

What happens?

Subnet mask /24 or 255.255.255.0 tells router:
If first 24 bits match → foward packet to 172.20.3.5

	Dot-decimal notation	Binary form
IP address	192.168.5.130	11000000.10101000.00000101.10000010
Subnet mask	255.255.255.0	11111111.11111111.11111111.00000000
Network part	192.168.5.0	11000000.10101000.00000101.00000000

Likewise: 172.0.0.0/8 via router 172.20.3.5
→ If first 8 bits match, packet forwarded to 172.20.3.5

IPv4 Subnets

Your own routing table…

```
netstat -rn
Active routes:
  Network destination          Netmask        Gateway        Interface     Metric
              0.0.0.0          0.0.0.0    192.168.0.1     192.168.0.13         10
            127.0.0.0        255.0.0.0        On-link        127.0.0.1        306
            127.0.0.1  255.255.255.255        On-link        127.0.0.1        306
      127.255.255.255  255.255.255.255        On-link        127.0.0.1        306
          192.168.0.0    255.255.255.0        On-link     192.168.0.13        266
         192.168.0.13  255.255.255.255        On-link     192.168.0.13        266
        192.168.0.255  255.255.255.255        On-link     192.168.0.13        266
```

→ *How is the route chosen?*

Solution: Largest number of bits that match destination IP address

If multiple matching routes found → take one with lowest metric

IPv4 NAT & Fragmentation

IPv4 NAT

IP Network Address Translation (NAT)

- Important concept implemented by routers / firewalls
- Basic idea
  - Transform any IP address into another one („pure NAT“)
  - When transport layer is TCP / UDP, also translate source / destination ports

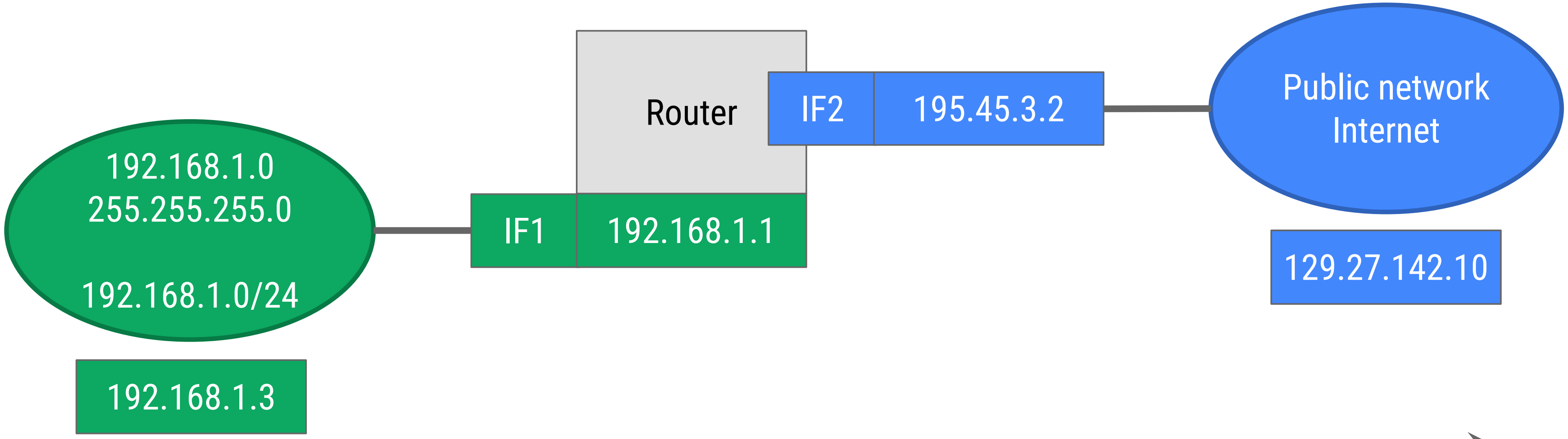
→ Router has to rewrite addresses in IP packet and re-compute checksum!

Special modes

- Destination NAT (DNAT) = „Port Forwarding“ or „Demilitarised zone“ (DMZ)
  - Transparently change destination IP (and port) of end-route packet
- Source NAT (SNAT) → counterpart of DNAT

IPv4 NAT

Network:



- Data flow wouldn‘t work:
  No route back to 192.168.1.3

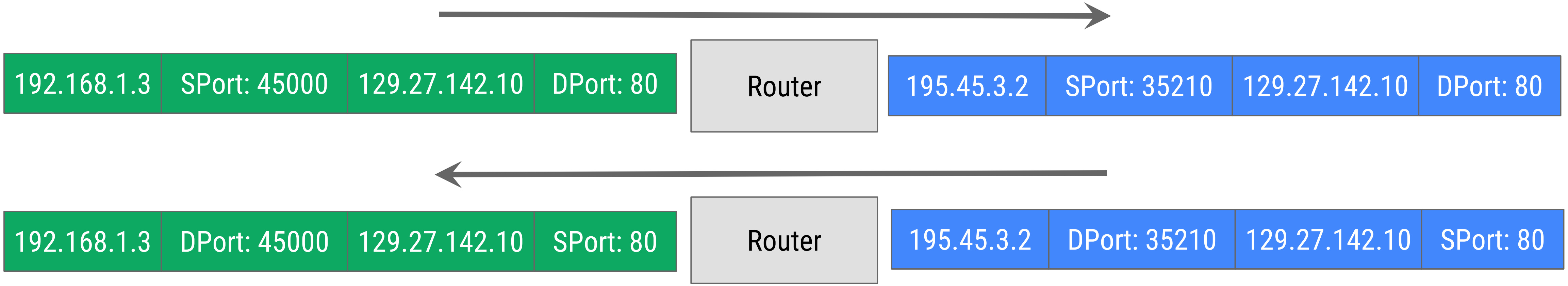
→

192.168.1.3	SPort: 45000	129.27.142.10	DPort: 80

←

192.168.1.3	DPort: 45000	129.27.142.10	SPort: 80

- But with NAT, the router translates the address:



IPv4 NAT

How does it work on the router?

- Has to keep track of translated addresses!
  - Needs to exchange them for requests: Internal IP -> External IP
  - And back when replies arrive: External IP -> Internal IP

- Transport layer
  - Translation also for source and destination ports

NAT Traversal problem

Two hosts, both behind a NAT try to connect to each other via Internet
→ Solution: „TCP Hole Punching“ See: https://goo.gl/Co4OZA
IP of other party must be known and NAT port *predictable*

IPv4 Fragmentation

Problem

- IP Packet Size: 20 – 65.536 bytes
- Lower network layers may only support smaller frames

→ Fragmentation needed

Maximum possible size?

- Maximum Transmission Unit (MTU)
  Defines max. amount of bytes the data link can pass onwards

- Headers of link layer **not** included in MTU

→ If max. MTU 1500 (Ethernet), IP packet with headers + payload may have max. 1500 bytes!

IPv4 Fragmentation

Idea

- For Ethernet we know max. MTU is 1500 bytes
- We could simply limit IP packet size to 1500

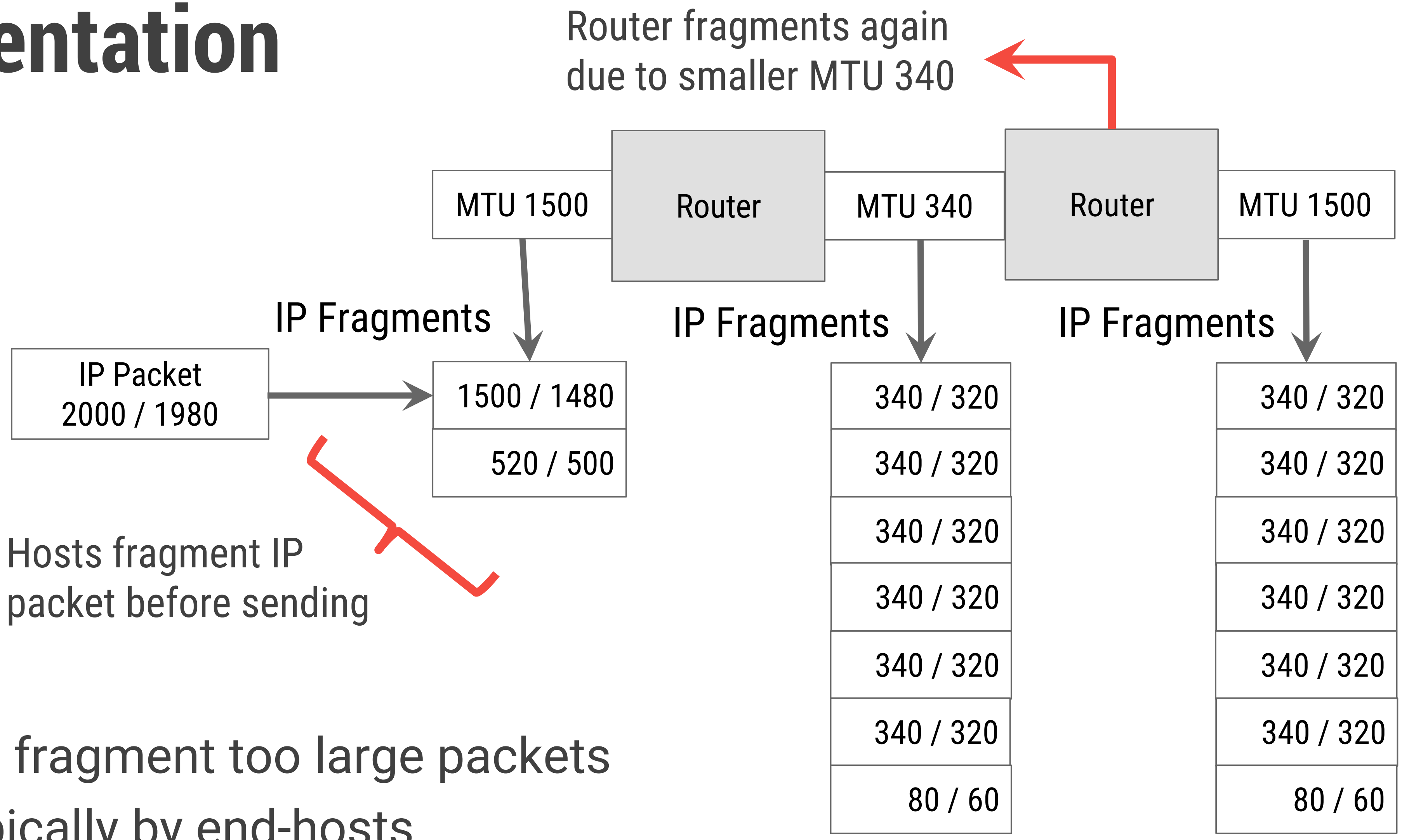
But: *Different links between routers can have different MTUs!*

1. Start with MTU 1500
2. Have one router with 340 → So fragmentation is needed anyway!
3. Finally 1500 again

Another aspect: Larger MTU = greater efficiency

But drawback: Larger packets longer occupy link → increase latency

IPv4 Fragmentation



- Hosts *or* routers fragment too large packets
- Re-assembly typically by end-hosts
  - **Not** routers because „Intelligence at end-points“
  - Exception in some cases: NAT and firewalls

IPv4 Fragmentation

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="4">Internet Header Length</td><td colspan="6">Differentiated Services Code Point</td><td colspan="2">Explicit Congestion Notification</td><td colspan="16">Total Length</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td>8</td><td>64</td><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td>12</td><td>96</td><td colspan="32">Source IP Address</td></tr>
<tr><td>16</td><td>128</td><td colspan="32">Destination IP Address</td></tr>
<tr><td>20</td><td>160</td><td colspan="32">Options (if IHL > 5)</td></tr>
</table>

- Flags: Bit 1 set = DF (Don‘t fragment), Bit 2 set = MF (More fragments)
- Fragment Offset: Offset of current fragment relative to unfragmented packet
- Identification: Identify fragmented packets

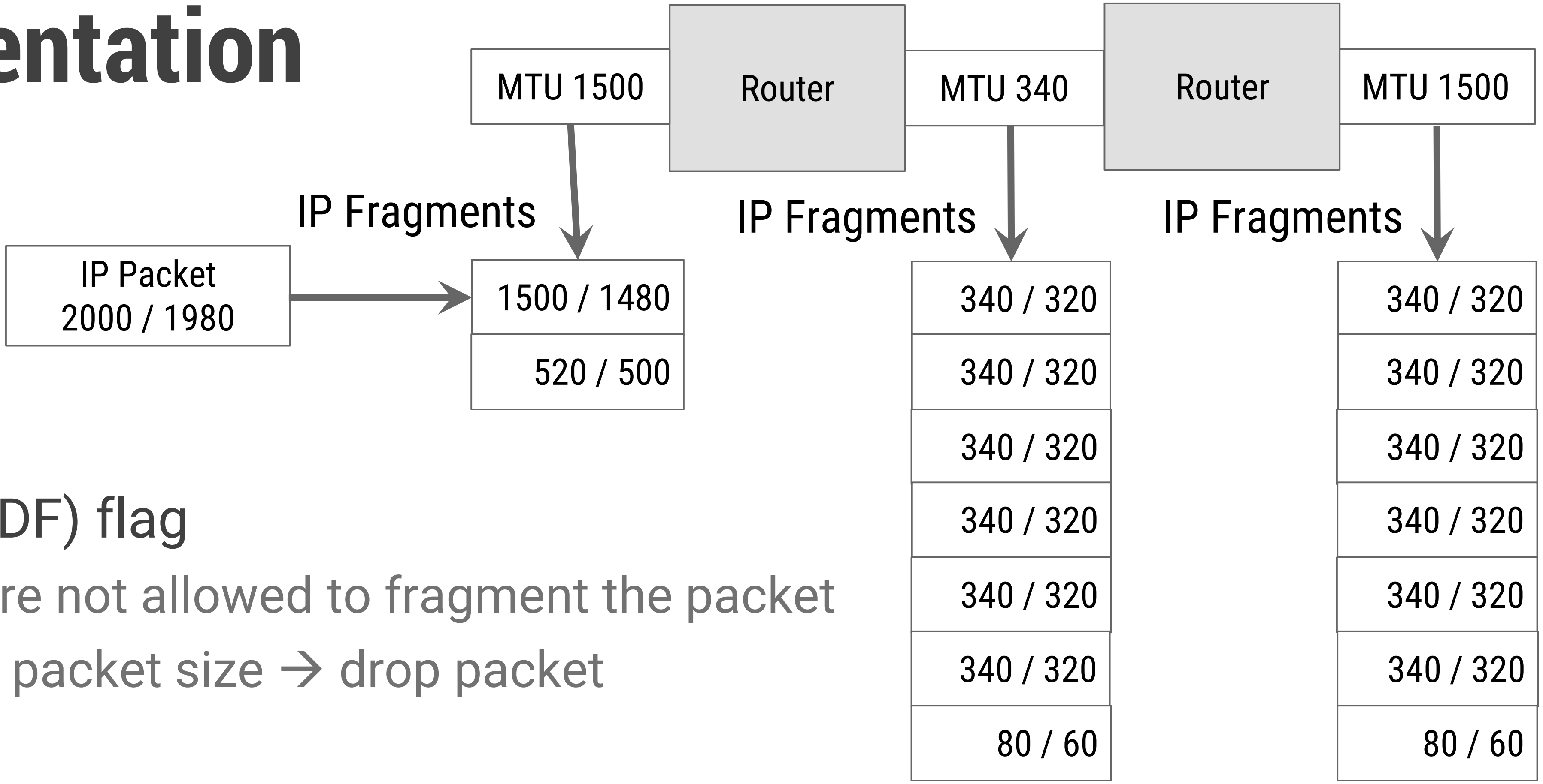
IPv4 Fragmentation

Fragmenting an IP Packet for MTU 340

IP Packet
2000 (with header) /
1980 (payload)

Fragment	Header + Payload / Payload Size	ID	Fragment Offset	MF Flag
1	340/320 bytes	788	0	1
2	340/320 bytes	788	40 (8*40 = 320)	1
3	340/320 bytes	788	80 (8*80 = 640)	1
4	340/320 bytes	788	120	1
5	340/320 bytes	788	160	1
6	340/320 bytes	788	180	1
7	80/60 bytes	788	220	0

IPv4 Fragmentation



- Don‘t fragment (DF) flag
  - Router, hosts are not allowed to fragment the packet
  - If router MTU < packet size → drop packet
- Purpose:
  Makes sense to already fragment appropriately at the beginning

→ ***But how do we find out what MTU should be used for fragmentation?***

IPv4 Fragmentation

Fragmenting at the beginning makes sense because
assuming 2000 bytes IP packet, separated into 6 fragments...

Q: Now what if 1 fragment is lost?

A: Retransmission of whole 2000 bytes IP packets and repeated process of fragmentation

Also, if firewalls or NAT require the packet to be re-assembled, it is processed faster if the appropriate MTU is used...

But still: How do we find out the maximum MTU?

IPv4 Fragmentation

RFC 1191

Solution: MTU Path Discovery

Workflow

1. Endpoints send IP packet with DF flag set
2. If router is encountered with MTU < packet size → drops packet and sends back ICMP message Type 3: *„Destination unreachable"* with code 4: *„Fragmentation required, and DF flag set"*
3. Repeated until MTU small enough to traverse path without fragmentation

Problem in practice

ICMP messages are often blocked by firewalls entirely, e.g. to prevent pings
→ Alternative using TCP: Progressively try larger packets

See: https://goo.gl/J57TwY

ICMPv4

Internet Control Message Protocol

RFC 792

= ICMPv4

- Encapsulated in IP packets
- Used to send error and information messages
  - E.g. if router drops packet as destination not reachable

- 32-bit messages
  - Protocol number 1 in IP header
  - Variable size of payload data → exploitable!

- Most popular use: ping and traceroute

	Bits 0–7	Bits 8–15	Bits 16–23	Bits 24–31
IP Header (20 bytes)	Version/IHL	~~Type of service~~ 0	Length	
	Identification		*flags* and *offset*	
	Time To Live (TTL)	~~Protocol~~ 1	Checksum	
	Source IP address			
	Destination IP address			
ICMP Header (8 bytes)	Type of message	Code	Checksum	
	Header Data			
ICMP Payload (*optional*)	Payload Data			

ICMP Codes

Mostly used...

Type	Code	Description
0 – Echo Reply	0	Echo reply (ping)
3 – Destination Unreachable	0	Destination network unreachable
	1	Destination host unreachable
	2	Destination protocol unreachable
	3	Destination port unreachable
	4	Fragmentation required, and DF flag set
8 – Echo Request	0	Echo request (ping)
9 – Router Advertisement	0	Broadcast IP address of router in local subnet
10 – Router Solicitation	0	Client requests IP addresses of routers
11 – Time Exceeded	0	TTL expired in transit (traceroute)

For more codes, see https://goo.gl/olW1ai

IAIK TU Graz

ICMPv4 Ping

```
ping online.tugraz.at
Reply by 129.27.2.210: bytes=32 time=14ms TTL=245
```

Echo Request

```
> Internet Protocol Version 4, Src: 192.168.0.13, Dst: 129.27.2.210
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4cd1 [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 138 (0x008a)
    Sequence number (LE): 35328 (0x8a00)
    [Response frame: 83]
  v Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
      [Length: 32]
```

```
0000  80 c6 ab 73 f5 64 c8 60  00 c9 e2 77 08 00 45 00   ...s.d.` ...w..E.
0010  00 3c 08 1c 00 00 80 01  ee 02 c0 a8 00 0d 81 1b   .<...... ........
0020  02 d2 08 00 4c d1 00 01  00 8a 61 62 63 64 65 66   ....L... ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

Echo Reply

```
> Internet Protocol Version 4, Src: 129.27.2.210, Dst: 192.168.0.13
v Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x54d1 [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 138 (0x008a)
    Sequence number (LE): 35328 (0x8a00)
    [Request frame: 82]
    [Response time: 14.622 ms]
  v Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
      [Length: 32]
```

Attacks using ICMP

IP Header	ICMP Header	ICMP Data
20 bytes	8 bytes	> 65.507 bytes

- Ping of Death
  - Causes buffer overflow on receiver due to flawed TCP/IP implementation
  - Happens if system cannot handle more than RFC 791 allows (65.535 bytes)
- Ping Flood
  - Send so many ping requests that normal traffic fails to reach system
- Smurf attack

DDoS attack

  - Attacker sends ping packets with spoofed source IP (= victim IP address) to broadcast address in network
  - All connected clients will answer and overwhelm victim

→ Reasons why firewalls often block ICMP (entirely!)

IPv4 Multicasting & Routing

IPv4 (Broad)casting

- Unicasting (*one-to-one*)
  - Single sender, single receiver
  - Used for all network processes where private or unique resource is requested
- Multicasting (*one-to-some*)
  - Send data to multiple „interested“ receivers
- Broadcasting (*one-to-many*)
  - Send data to all receivers
    Target: Special IP *255.255.255.255* or local broadcast addr., e.g. *192.168.1.255*
- Anycasting (*one-to-nearest*)
  - Send data with same address but only to closest → load balancing

IPv4 Multicasting

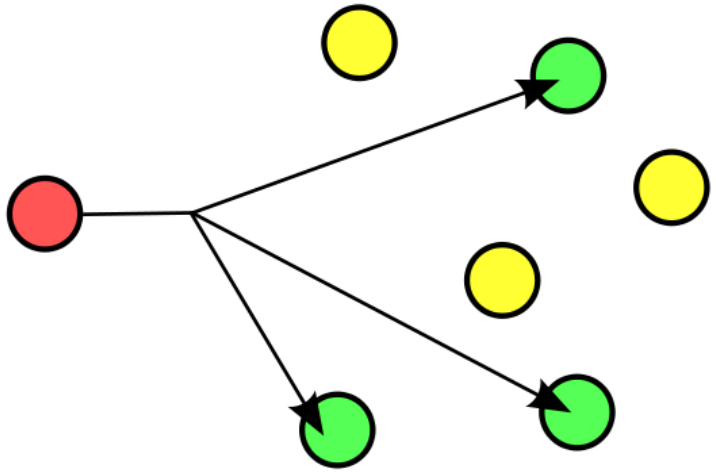
Key Facts

- Address range `224.0.0.0/4 → 224.0.0.0 – 239.255.255.255` (former Class D network)
- Protocol: *„Internet Group Management Protocol" (IGMP)*
- Usage: Streaming of audio / video (IPTV)

Workflow

- Source sends packets to multicast address with group, e.g. *239.1.1.1*
- Receiver joins group at *239.1.1.1* using IGMP protocol
  - Data usually sent connection-less way (UDP)

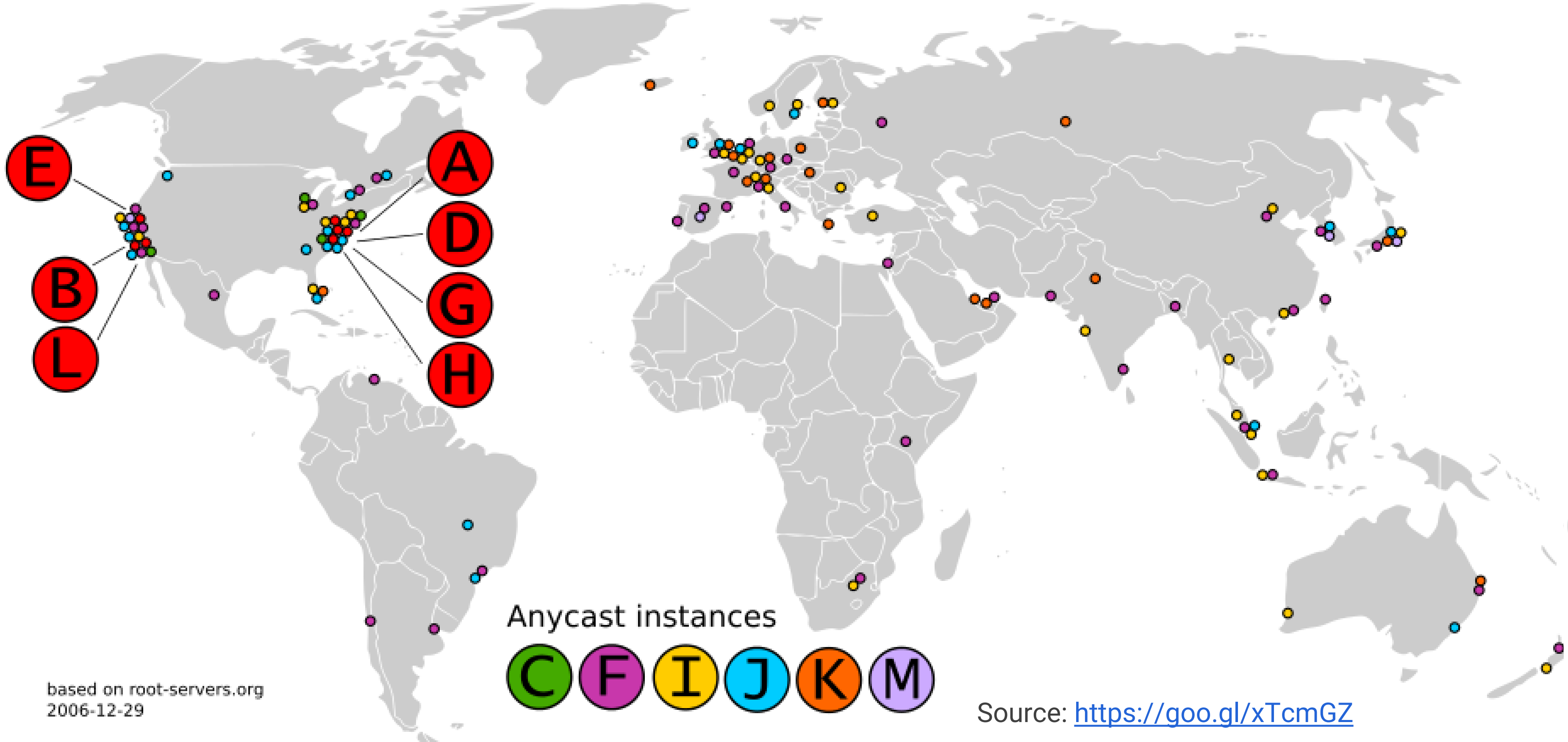
→ *Deployment typically not chosen by end-user but specific network, e.g. Telekom IPTV*



IPv4 Anycasting

Workflow

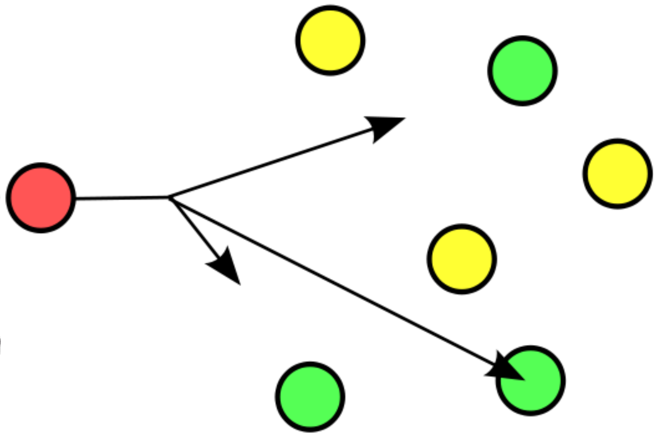
- Set same destination address for every host in a group of potential receivers
- Using Borderless Gateway Protocol (BGP) a client is routed to „nearest“ host



Source: https://goo.gl/xTcmGZ

Examples

- Domain Name System
- Content Delivery Networks (CDN)



IP Routing

In local subnets we have static entries,
assigned by admins or provided via DHCP

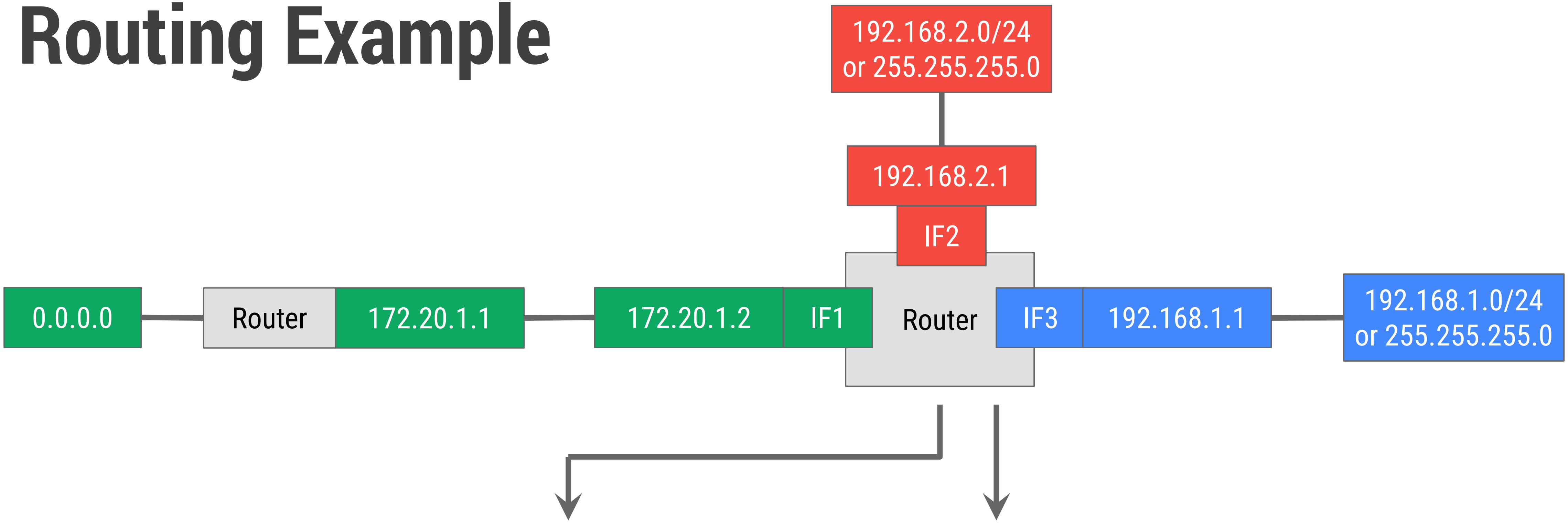
Q: But how to detect new paths through the Internet?
How to circumvent failed links or choose faster ones?

A: Using routing protocols

Two main concepts for routing

- Forwarding
  - Router must move incoming packet to appropriate output link
- Routing
  - Algorithms determine possible paths / routes for packet flow through networks

IP Routing Example




```
Routing / Forwarding table
192.168.2.0/24 via IF2
192.168.1.0/24 via IF3
172.20.1.0/24 via IF1
0.0.0.0 via 172.20.1.1
```

- Needs to understand data-link and network layer
- Routers read IP header
  - Destination address
  - Checksum
  - Fragmentation etc.

IP Routing

Autonomous System (AS) = Collection of different IP network prefixes run by one or more network operators with a clearly defined routing policy

Uses routing protocols

- Interior Gateway Protocols (IGP)
  - Routing traffic within an AS — **Metrics**: Delays, bandwidth, hop count
- Exterior Gateway Protocols (EGP)
  - Routing traffic between AS — **Metrics**: Policies, rule-sets

Why?

- Automatically determine network structure
- Provide forwarding tables for routers
- Exchange information with neighbouring routers

IP Routing

Principal routing algorithms

- **Link-state protocols (LS):** *„Tell all network nodes who are your neighbours“*
  After some time, every router knows full topology of network

- **Distance-vector protocols (DV):** *„Tell your neighbours how your world looks like“*
  Distance to other routers basis for shortest path problem
  → Improved version with better loop detection: `Path-Vector`

Protocol	Routing Algorithm	Shortest-Path Algorithm	Usage	Notice
BGP	Path-Vector	Bellman-Ford	EGP	Standard, prevents loops
RIP	DV	Bellman-Ford	IGP	Count-to-infinity (= loops!)
OSPF	LS	Dijkstra	IGP	Hierarchical routing
IS-IS	LS	Dijkstra	IGP	ISO standard, like OSPF
EIGRP	DV	DUAL	IGP	Cisco standard

Outlook

- 04.12.2019
  - Network layer: IPv6
    - Addressing, Differences to IPv4, NDP, ICMPv6
  - Transport layer: TCP / UDP
    - Flow and Congestion control

- 11.12.2019
  - Application Layer: HTTP, HTTP/2, AJAX, WebSockets
  - Application Layer: DNS

Bachelor@**IAIK Topics** + Student **Research Awards**

Friday **29 Nov 2019, 12:00–13:00**

IAIK Foyer, Inffeldgasse 16a, Ground floor

www.iaik.tugraz.at/bachelor