FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Practical Computer Networks and Applications

Introduction and Fundamentals

Prof. Dr. Baun, Prof. Dr. Ebinger, Prof. Dr. Hahm, Prof. Dr. Kappes, Dipl. Inf. (FH) Maurizio Petrozziello, Henry Cocos, M.Sc.
{baun,cocos,peter.ebinger,oliver.hahm,kappes,petrozziello}@fb2.fra-uas.de
Frankfurt University of Applied Sciences
Faculty of Computer Science and Engineering
Nibelungenplatz 1
60318 Frankfurt am Main

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Contents

Introduction to PCNA

The course **Practical Computer Networks and Applications** consists of two parts:

1. The theoretical lecture on computer networks
2. The practical lab exercises on the application of computer networks

Theoretical lecture

This slide set is a recap of the theoretical foundations of the course **Computer Networks** from the winter term! It is intended as a reminder of the topics discussed last semester and will give you a brief summary on the protocols and technologies neccessary for this course!

The theoretical foundation for this course

- The Lab Exercise will use technologies from all network layers and therefore the knowledge on the technologies and protocols is neccessary for the successful participation in the lab!
- You can use this slide set as a tool for the lab exercises!
- Each lab exercise will be accompanied by a corresponding slide set!

The Foundation

This slide set will give you the foundation on the lab exercises. The practical exercises will demonstrate their use in practice!

Contents

Reference Models

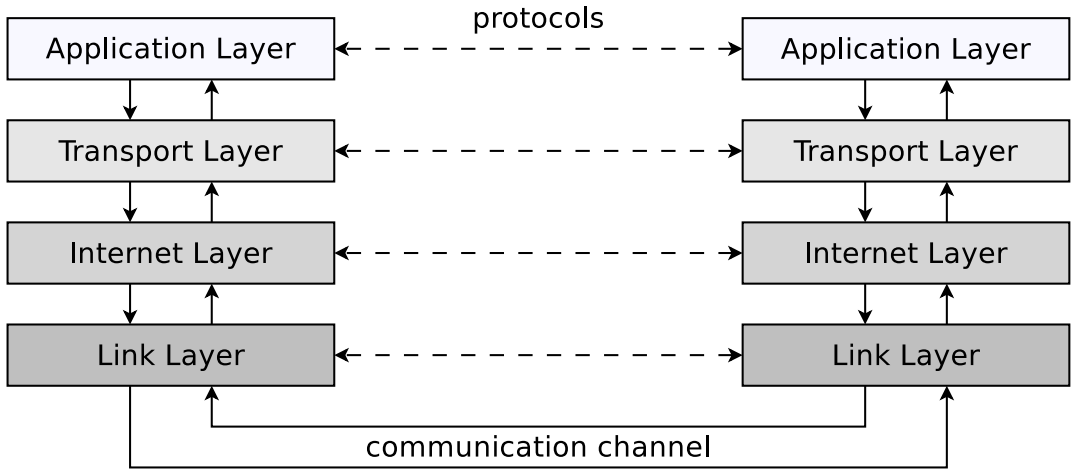
- Communication in computer networks is subdivided into **reference models**
- Each **layer** of a reference model handles a particular aspect of communication and offers **interfaces** to the overlying layer and underlying layer
- Each interface consists of a set of **operations**, which together define a **service**
- In the layers, the data is encapsulated ($\Longrightarrow$ **encapsulation**)
- Because each layer is complete in itself, single protocols can be modified or replaced without affecting all aspects of communication
- The most popular reference models are…
  - the **TCP/IP reference model**,
  - the **OSI reference model**
  - and the **hybrid reference model**

TCP/IP Reference Model or DoD Model

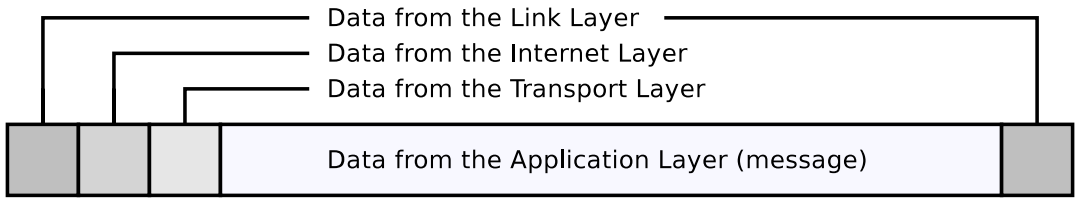
- Developed from 1970 onwards by the Department of Defense (DoD) in the Arpanet project
- Divides the required functionality to realize communication into 4 layers
- For each layer, it is specified, what functionality it provides
  - These requirements are implemented by communication protocols
    - Concrete implementation is not specified and can be implemented in different ways
    - Therefore, for each of the 4 layers, multiple protocols exist

Number	Layer	Protocols (Examples)
4	Application Layer	HTTP, FTP, SMTP, POP3, DNS, SSH, Telnet
3	Transport Layer	TCP, UDP
2	Internet Layer	IP (IPv4, IPv6), ICMP, IPsec, IPX
1	Link Layer	Ethernet, WLAN, ATM, FDDI, PPP, Token Ring

TCP/IP Reference Model – Message Structure

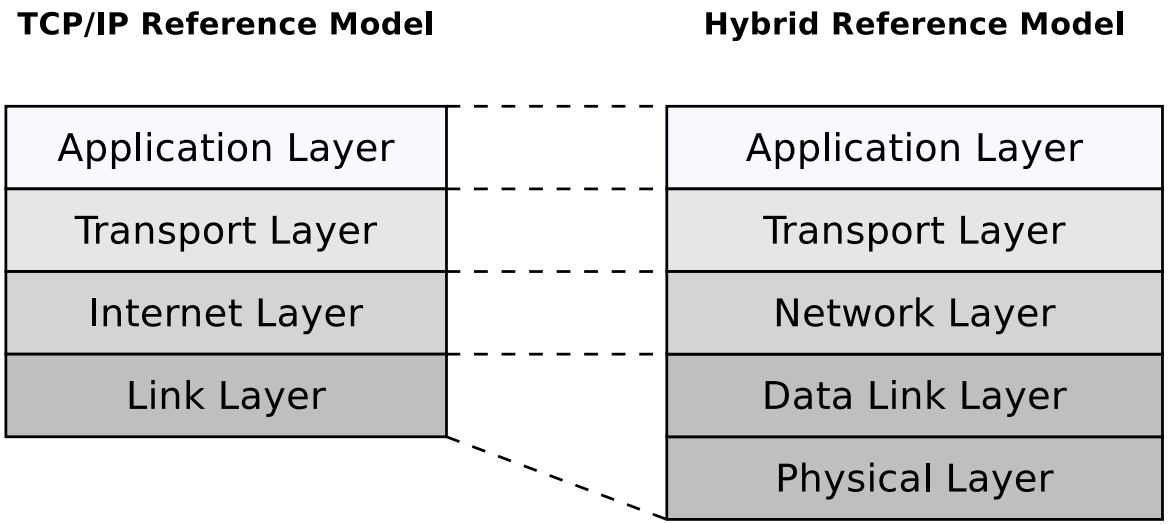


- Each layer adds additional information as **header** to the message
  - Some protocols (e.g. Ethernet) add in the link layer not only a header but also a **trailer** at the end of the message
  - The receiver analyzes the header (and trailer) on the same layer



Hybrid Reference Model

- The TCP/IP reference model is often presented in the literature (e.g. by Andrew S. Tanenbaum) as a 5-layer model
  - Reason: It makes sense to split the **Link Layer** into 2 layers, because they have different tasks
- This model is an extension of the TCP/IP model and is called **hybrid reference model**



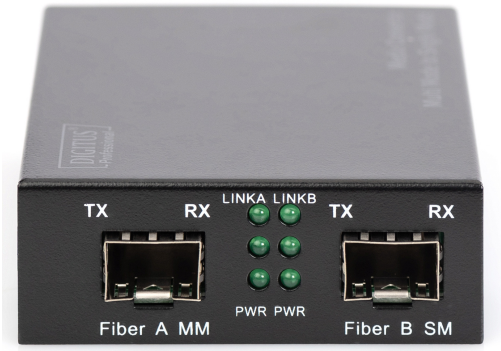
The objects of the individual layers will be discussed on the basis of the hybrid reference model

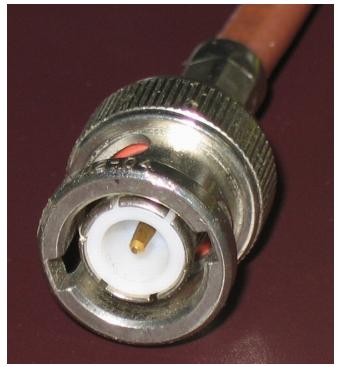
Physical Layer

- **Transmits the ones and zeros**
  - **Physical connection** to the network
  - **Conversion of data in signals**
- Protocol and transmission medium specify among others:
  - How many bits can be transmitted per second?
  - Can transmission take place simultaneously in both directions?
- Devices: **Repeater**, **Hub** (Multiport Repeater)

Hybrid Reference Model

Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer





Data Link Layer

- Ensures error-free data exchange of **frames** between devices in physical networks
  - Detects transmission errors with **checksums**
  - Controls the access to the transmission medium (e.g. via CSMA/CD or CSMA/CA)
- Specifies physical network addresses (**MAC addresses**)
  - At sender site: Packs the Network Layer packets into frames and transmits them (in a reliable way) via a physical network from one device to another
  - At receiver site: Identifies frames in the bit stream from the Physical Layer
  - Devices: **Bridges**, **Layer-2-Switches** (Multiport Bridges) and **Modems** connect physical networks

Hybrid Reference Model

Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

Network Layer

- Forwards (*routes*) **packets** between logical networks (over physical networks)
  - For this *internetworking*, the network layer defines **logical addresses** (**IP addresses**)
  - Each IP packet is *routed* independently to its destination and the path is not recorded
- At sender site: Packs the segments of the Transport Layer in packets
- At receiver site: Unpacks the packets in the frames from the Data Link Layer
  - **Routers** and **Layer-3-Switches** connect logical networks
- Usually the connectionless Internet Protocol (IP) is used
  - Other protocols (e.g. IPX) have been replaced by IP

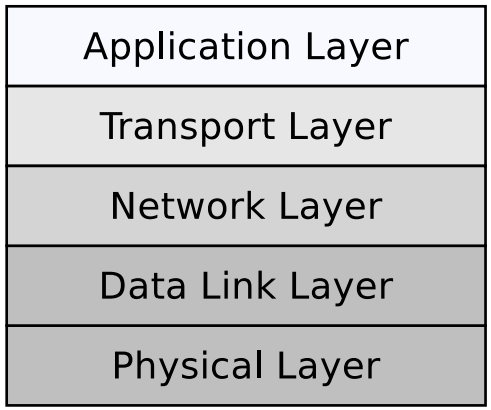
Hybrid Reference Model

Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

Transport Layer

- Transports **segments** between processes on different devices via so-called end-to-end protocols
- At sender site: Packs the data of the Application Layer into segments
- At receiver site: Unpacks the segments inside the packets from the network layer
  - Addresses processes with **port numbers**
    - Data Link Layer and Network Layer implement physical and logical addressing of the network devices
  - Transport protocols implement different forms of communication
    - UDP (User Datagram Protocol): Connectionless communication
    - TCP (Transport Control Protocol): Connection-oriented communication
      - Combination of TCP/IP = de facto standard for computer networks

Hybrid Reference Model



Application Layer

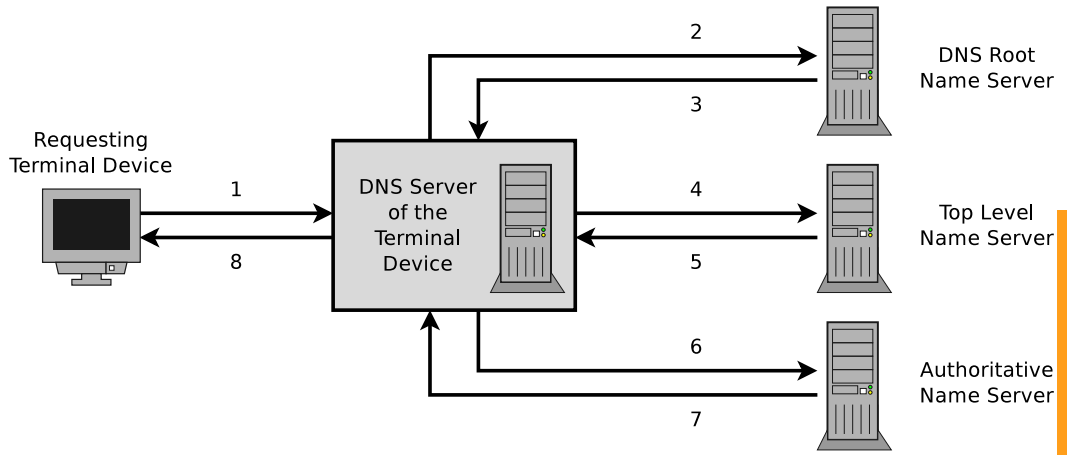
- Contains all protocols, that interact with the application programs (e.g. browser or email program)
- Here are the messages (e.g. HTML pages or emails), formated according to the used application protocol
- Some Application Layer protocols: HTTP, FTP, SMTP, POP3, DNS, SSH, Telnet

Hybrid Reference Model

Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

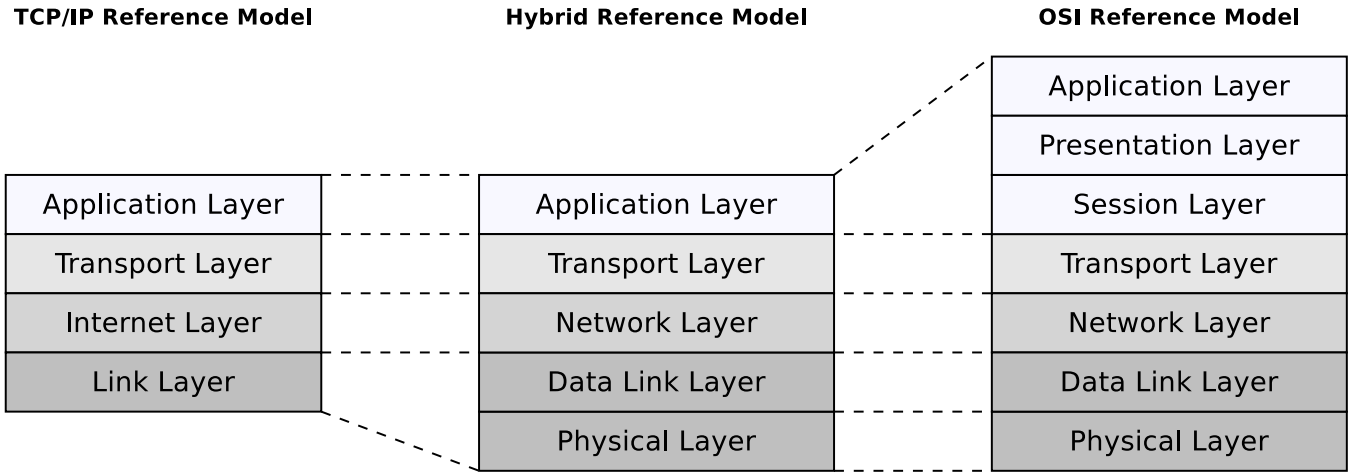
wikipedia.org (CC0)

pixabay.com (CC0)



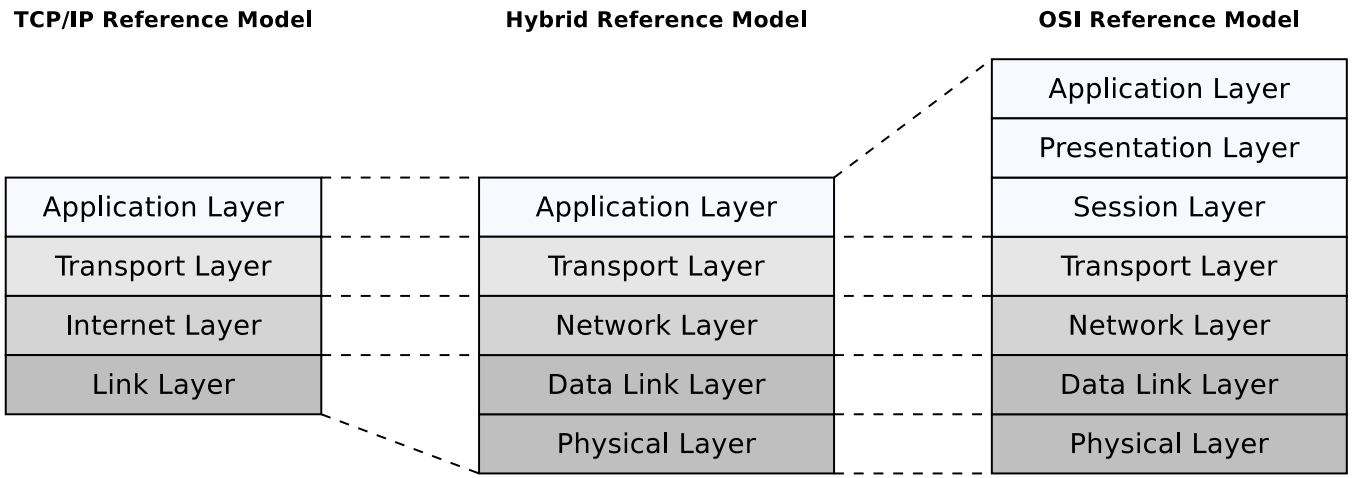
OSI Reference Model

- Some years after the TCP/IP reference model (1970s), the OSI reference model was developed from 1979 onwards
  - 1983: Standardized by the Intern. Organization for Standardization (ISO)
  - OSI = Open Systems Interconnection
- The structure is similar to the TCP/IP reference model
  - The OSI model implements 7 layers
- In contrast to the hybrid reference model, the Application Layer functionality is distributed across 3 layers in the OSI reference model



Reference Models – Summary

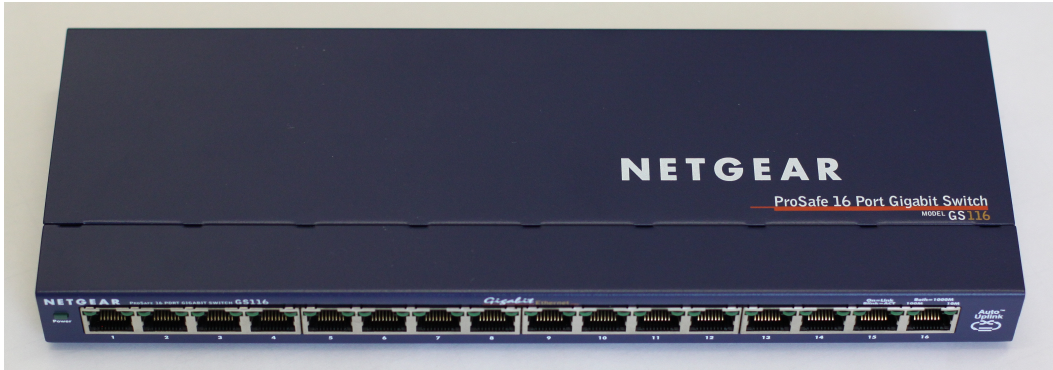
- Conclusion: The hybrid reference model illustrates the functioning of computer networks in a realistic way
  - It distinguishes between the Physical Layer and Data Link Layer
    - This is useful, because the objectives differ a lot
  - It does not subdivide the Application Layer
    - This is not useful and does not take place in practice
    - Functionalities, which are intended for Session Layer and Presentation Layer, are provided by Application Layer protocols and services
  - It combines the advantages of the TCP/IP reference model and the OSI reference model, without taking over their drawbacks

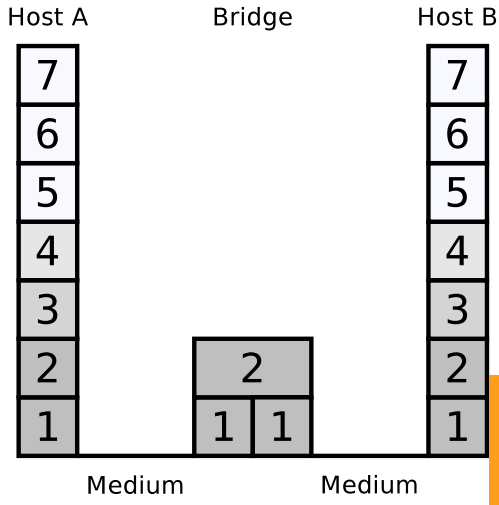


Contents

Devices of the Data Link Layer: Bridges

- Devices of the Physical Layer increase the length of physical networks
- A Bridge has only 2 ports
- Simple Bridges forward all incoming frames





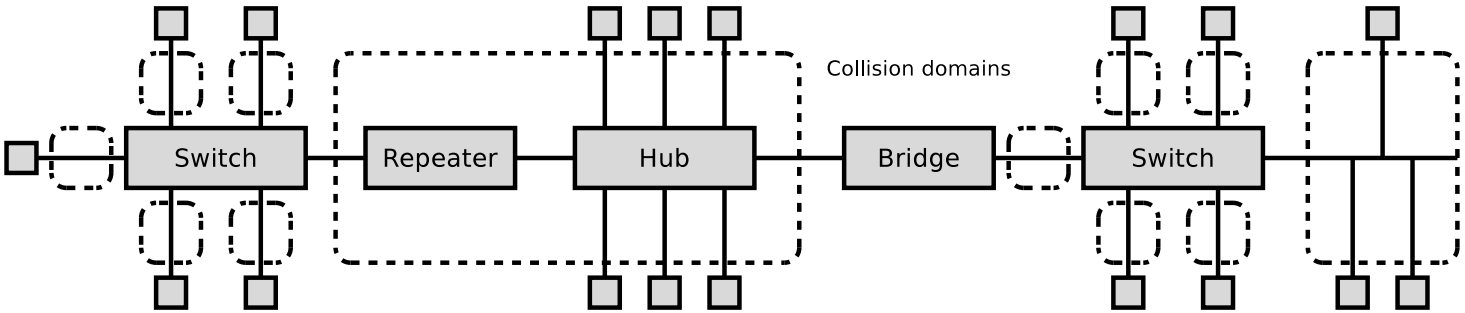
- Bridges with > 2 ports are called **Layer-2-Switch**, they typically provide 4-48 Interfaces

Functioning of Bridges and Layer-2-Switches

- Bridges and Switches check the correctness of the frames via **checksums**
- Bridges do **not need addresses** for filtering and forwarding the frames, because they do not actively participate in the communication
  - They operate transparent, just like the devices of the Physical Layer
    - Reason: They do not communicate on a higher protocol layer as the Data Link Layer

Collision Domain – Bridges and Layer-2-Switches

- Bridges and Switches operate on Data Link Layer and forward frames from one physical network to other ones
- **Each physical network is a separate collision domain**
  - If a physical network is split by a Bridge or a Switch, also the collision domain is split
    - As effect, the number of collisions drops
- For Bridges and Switches, each port forms its own collision domain



- In a *fully switched network*, each port of the Switches is connected with just a single network device
  - Such a network is collision-free and state of the art

Addressing in the Data Link Layer

- The Data Link Layer protocols specify the format of the physical network addresses
- **Terminal devices (Hosts)**, **Routers** and **Layer-3-Switches** require physical network addresses
  - Such devices must be addressable on Data Link Layer because they provide services at upper protocol layers
- **Bridges** and **Layer-2-Switches** do not actively participate in the communication
  - Therefore, they don't require physical network addresses for their basic functionality, which is the filtering and forwarding of frames
  - Bridges and Switches require physical network addresses, when they implement the STP to avoid loops, or when they offer services from an upper protocol layer
    - Examples are monitoring services or graphical web interfaces for administration tasks
- **Repeaters** and **Hubs** that operate only at the Physical Layer, have no addresses

MAC Addresses (1/2)

- The **physical network address** are called **MAC addresses** (Media Access Control)
  - They are independent from the logical addresses of the Network Layer
- Ethernet uses the **Address Resolution Protocol** (ARP) to resolute the logical addresses of the Network Layer (IPv4 addresses) to MAC addresses
  - For IPv6, the **Neighbor Discovery Protocol** (NDP) provides the identical functionality and operates in a similar way
- MAC addresses have a length of 48 bits (6 bytes)
  - Thus, the address space contains 2^{48} possible addresses
- In order to make the representation compact and human-friendly to read, MAC addresses are usually written in hexadecimal notation
  - The bytes are separated from each other with dashes (`-`) or colons (`:`)
- Example of the notation: `00-16-41-52-DF-D7`

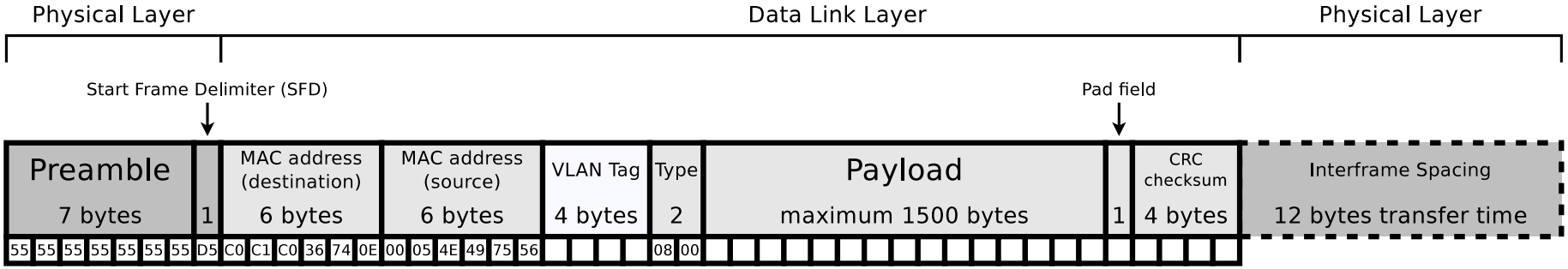
MAC Addresses (2/2)

- Each MAC address is intended to be permanently assigned to a network device and unique
  - But it is often possible to modify MAC addresses by software
    - However, this modification applies only until the next reboot of the computer
- **MAC broadcast address**
  - If a network device wants to send a frame to all other devices in the same physical network, it inserts MAC broadcast address in the destination address field of the frame
  - All 48 bits of this MAC address have the value `1`
  - Hexadecimal notation: `FF-FF-FF-FF-FF-FF`
  - Bridges and Switches do not forwarded frames to other physical networks, that contain the MAC broadcast address in the destination address field

Uniqueness of MAC Addresses

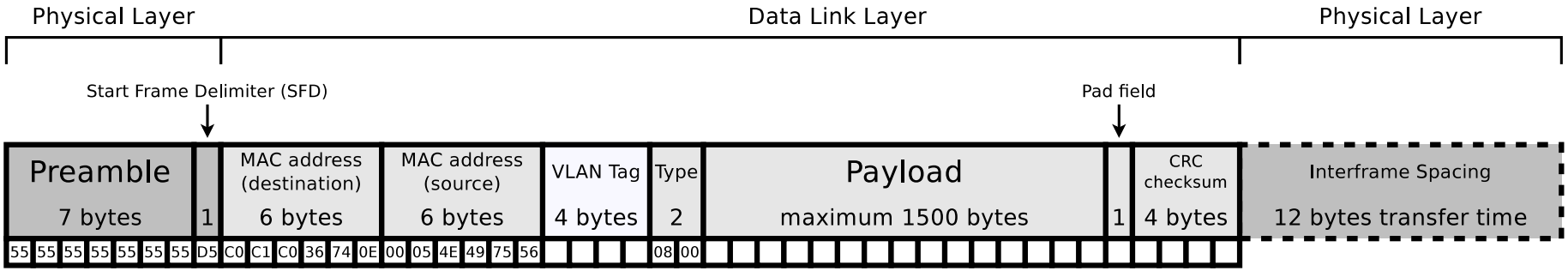
- The first 24 bits of the MAC address space are managed by the Institute of Electrical and Electronics Engineers (IEEE)
  - These 24 bits long addresses are called **MA-L** (MAC Address Block Large) or **OUI** (Organizationally Unique Identifier)
  - The OUIs can be checked in this IEEE database: `http://standards.ieee.org/develop/regauth/oui/public.html`
- The remaining 24 bits are specified by the hardware vendors independently for their network devices
  - That address space allows $2^{24} = 16,777,216$ individual device addresses per OUI
- Smaller address spaces are available too: **MA-S** (MAC Address Block Small) and **MA-M** (MAC Address Block Medium)

Framing in current Computer Networks (1/4) – Ethernet



- Up-to-date Data Link Layer protocols (e.g. Ethernet and WLAN) work bit-oriented and not byte-oriented
  - Reason: This way, every character encoding can be used
- Preamble is a 7 bytes long bit sequence `101010 ... 1010`
  - Is used in bus networks (topologies) to synchronize the receiver with the clock and to identify clearly the beginning of the frame
  - Is followed by the SFD (1 byte) with the bit sequence `10101011`

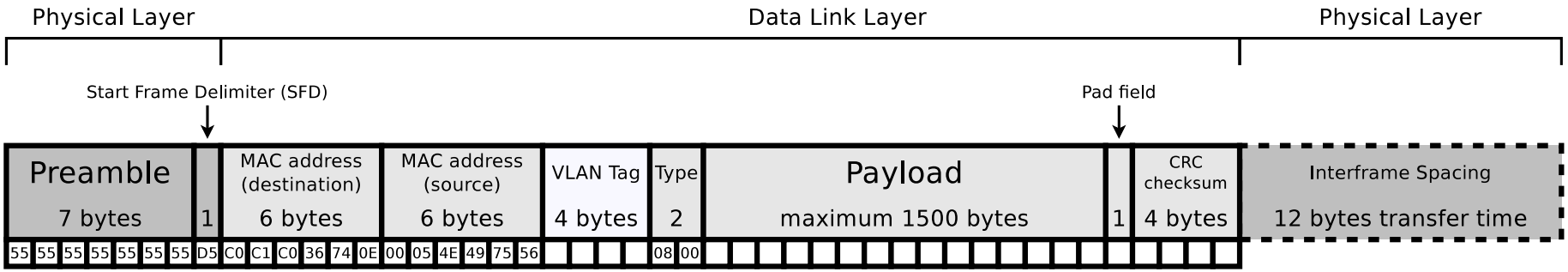
Framing in current Computer Networks (2/4) – Ethernet



- The fields for the physical addresses (MAC addresses) of sender and destination are 6 bytes long each
- The 4 bytes long optional VLAN tag contains, among others. . .
  - a 12 bits long VLAN ID
  - and a 3 bits long field for the priority information
- The field `Type` contains the information what protocol is used in the next upper protocol layer

- If IPv4 is used, the field `Type` has value `0x0800`
- If IPv6 is used, the field `Type` has value `0x86DD`
- If the payload contains an ARP message, the field `Type` has value `0x0806`

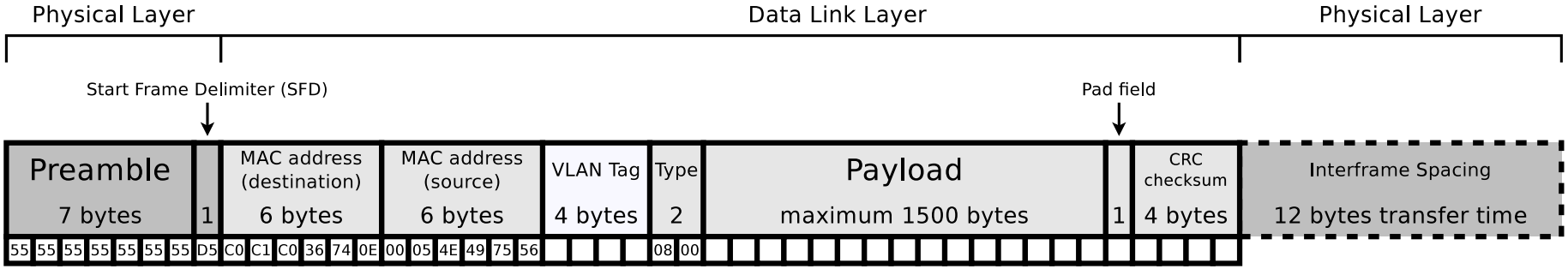
Framing in current Computer Networks (3/4) – Ethernet



- Minimum size of an Ethernet frame: 72 bytes
- Maximum size (including preamble and SFD): 1526 bytes
- The VLAN tag increases the maximum size by 4 bytes

- Each frame can contain a maximum of 1500 bytes payload
  - With the `Pad` field, the frame length can be increased to the minimum frame size (72 bytes) when needed
    - This is required to get the collision detection via CSMA/CD working
- The last field contains a checksum (32 bits) for all fields, except the preamble and SFD

Framing in current Computer Networks (4/4) – Ethernet



- The **Interframe Spacing** or **Interframe Gap** is the minimum idle period between the transmission of Ethernet frames via the transmission medium
- The minimum idle period is 96 bit times (12 bytes)
  - It is 9.6 microseconds when using 10 Mbps Ethernet
  - It is 0.96 microseconds when using 100 Mbps Ethernet
  - It is 96 nanoseconds when using 1 Gbps Ethernet
- Some network devices allow to reduce the Interframe Spacing period

Functioning of ARP (1/2)

- The **Address Resolution Protocol** (ARP) is used to resolve IP addresses of the Network Layer to MAC address of the Data Link Layer
- If a network device wants to transmit data to a receiver, it uses the receiver's IP address on the Network Layer
- But on the Data Link Layer, the MAC address is required
  - Therefore, **address resolution** must be carried out in the Data Link Layer
  - To find out the MAC address of a network device in the LAN, ARP sends a frame with the MAC broadcast address `FF-FF-FF-FF-FF-FF` as destination address
    - Each network device in the LAN receives and analyzes this frame
    - The frame contains the IP address of the searched network device
  - If a network device has this IP address, it sends an ARP response to the sender
    - The reported MAC address stores the sender in its local ARP cache

Functioning of ARP (2/2)

- The **ARP cache** is used to speed up the address resolution
  - It contains a table with these information for each entry:
    - Protocol type (IP)
    - Protocol address of the sender (IP address)
    - Hardware address of the sender (MAC address)
    - Time To Live (TTL)
  - The TTL is set by the operating system
  - If an entry in the table is used, the TTL is extended
- Modern Linux distributions discard entries after $\approx$ 5 minutes

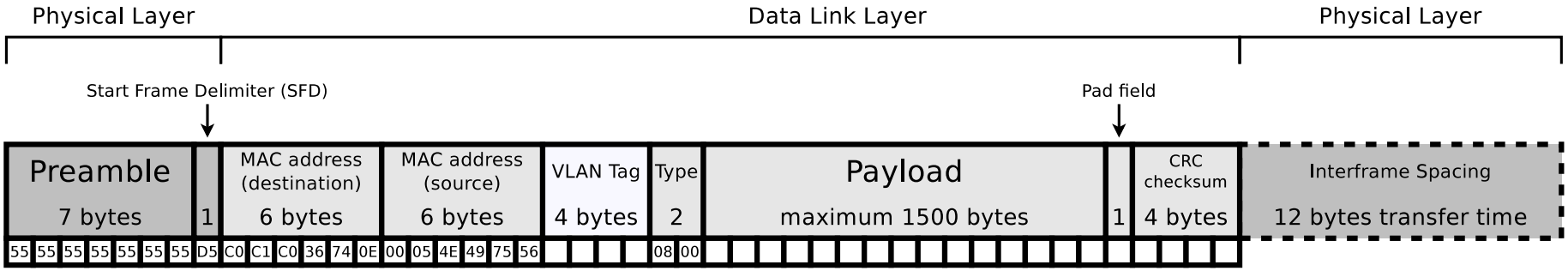
The ARP cache can be displayed via `arp -n` or `ip neighbour`

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.178.1            ether   9c:c7:a6:b9:32:aa   C                     wlan0
192.168.178.24           ether   d4:85:64:3b:9f:65   C                     wlan0
192.168.178.41           ether   ec:1f:72:70:08:25   C                     wlan0
192.168.178.25           ether   cc:3a:61:d3:b3:bc   C                     wlan0
```

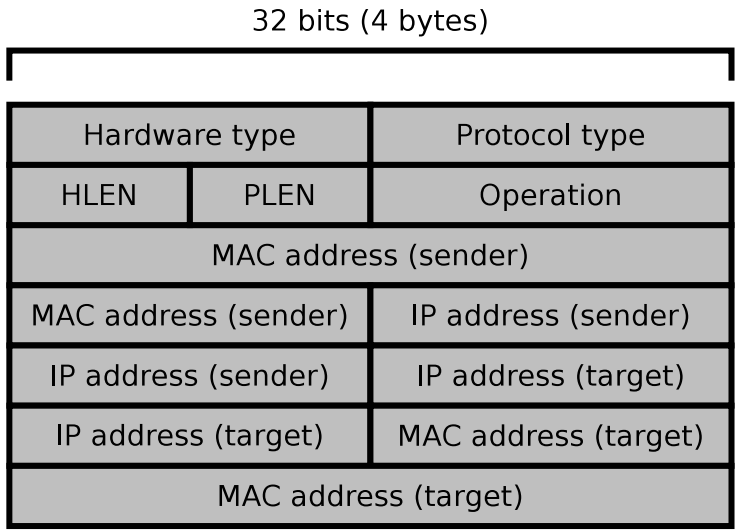
Address resolution requests can be send manually via `arping`

Structure of ARP Messages

- ARP messages are transmitted as payload via Ethernet frames
  - `type` = `0x0806` (for the ARP protocol)



- `HLEN` = hardware address (MAC address) length in bytes
  - For Ethernet: 6 bytes
- `PLEN` = IP address length in bytes
  - For IPv4: 4 bytes

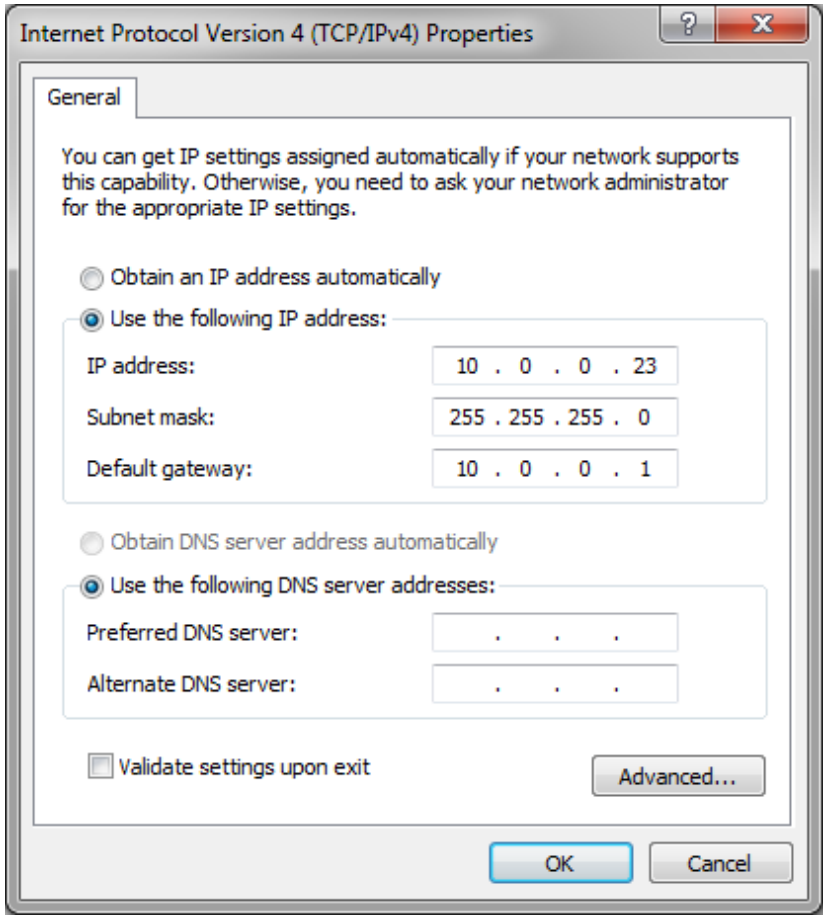


In an ARP request is the content of the field `MAC address (target)` irrelevant

Contents

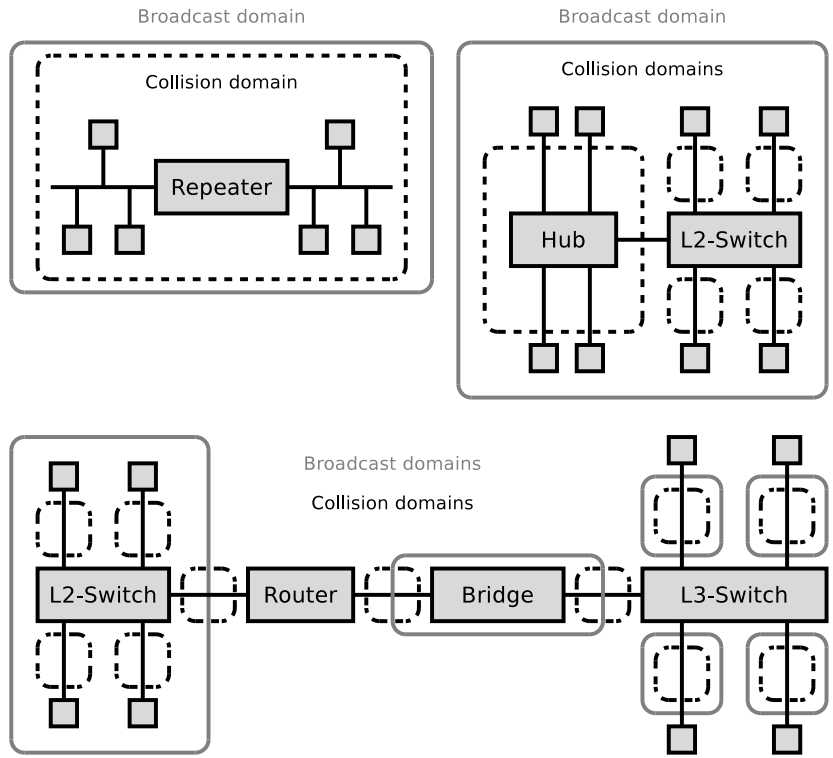
Gateways

- Modern computer networks operate almost exclusively with the Internet Protocol (IP)
  - For this reason, a protocol conversion at the Network Layer is mostly not necessary
- In the past, in the network preferences of a terminal device, the IP address of the Gateway was specified as **Default Gateway**
  - Today, this field contains the Router address, because a Gateway is usually not required any longer
    - Thus, the term **Default Router** would be suited better



Broadcast Domain (1/2)

- Logical part of a computer network, where a broadcast reaches all network devices that belong to that part
  - Devices, which operate on layer 3 (**Routers**) divide the broadcast domain
  - Devices, which operate on layer 1 and 2 (**Repeaters**, **Hubs**, **Bridges**, **Layer-2-Switches**) do not divide it
    - From the perspective of logical networks, they work transparent

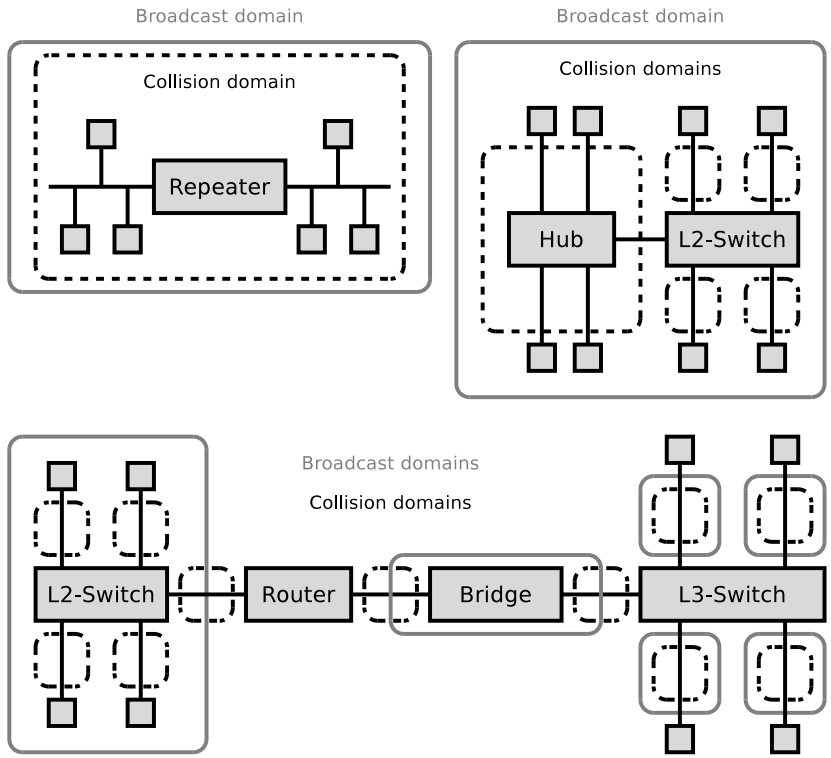


The technical term broadcast domain. . .

always applies to the network layer and never to the data link layer (although broadcasts exist also in the data link layer)

Broadcast Domain (2/2)

- Broadcast domains consist of one or multiple collision domains
- Routers operate at the Network Layer (layer 3)
  - This means, that each port of a Router is connected to a different IP network
    - This information is necessary for the calculation of the required number of subnets
  - Multiple Hubs, Switches, Repeaters or Bridges can operate in the same IP subnet
    - But it is impossible to connect an IP subnet to multiple ports of a Router

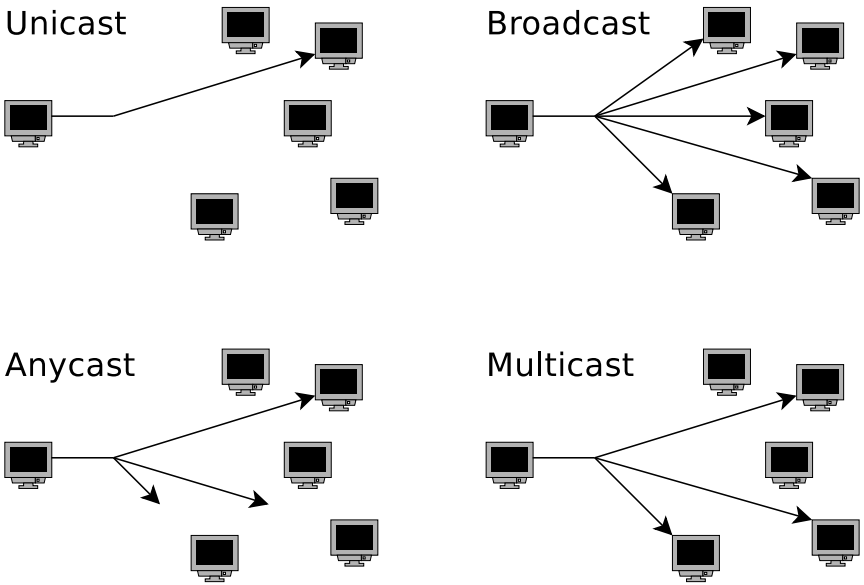


Addressing in the Network Layer (1/2)

- Using only physical addressing via MAC addresses is not useful in large-scale computer networks with possibly global proportions
  - Reason: Maintainability
- **Logical addresses** are required, which are independent from the specific hardware
  - Logical addressing separates the view of humans (logical addresses) from the internal view of computers and software (physical addresses)

Addressing in the Network Layer (2/2)

- Every Network Layer packet contains the IP address of the receiver
  - The structure of IP addresses specifies the Internet Protocol (IP)

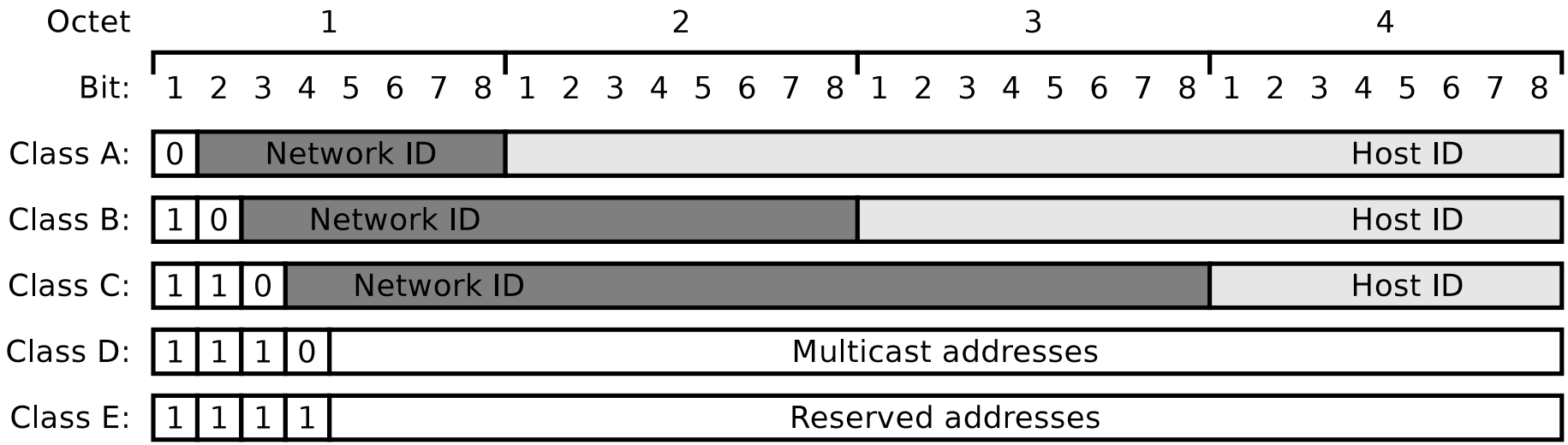


- An IP address can be assigned to a single receiver (**unicast**) or a group of receivers (**multicast** or **broadcast**)
- Multiple IP addresses can be assigned to a single network device
- If **Anycast** is used, a single device of a group of devices can be reached via a single address
  - The receiver with the shortest path responds

Multicast ist used for example by the routing protocols RIPv2 and OSPF and by Network Time Protocol (NTP) that is used for clock synchronization
Anycast is used for example by some Root Name Servers in the Domain Name System

Address Classes, Network Identifier and Host Identifier

- Originally, IPv4 addresses were categorized into classes from A to C
  - Additionally, the classes D and E for special purposes existed
- A 32 bits long IPv4 address consists of 2 fields:
  - **Network identifier** (network ID)
  - **Host identifier** (host ID)
  - Class A: 7 bits for the network ID and 24 bits for the host ID
  - Class B: 14 bits for the network ID and 16 bits for the host ID
  - Class C: 21 bits for the network ID and 8 bits for the host ID



Address Classes (1/2)

- The prefixes specify the address classes and their address ranges

Class	Prefix	Address range	Network ID	Host ID
A	0	0.0.0.0 - 127.255.255.255	7 bits	24 bits
B	10	128.0.0.0 - 191.255.255.255	14 bits	16 bits
C	110	192.0.0.0 - 223.255.255.255	21 bits	8 bits
D	1110	224.0.0.0 - 239.255.255.255	—	—
E	1111	240.0.0.0 - 255.255.255.255	—	—

- $2^7 = 128$ class A networks with a maximum of $2^{24} = 16,777,216$ host addresses each
- $2^{14} = 16,384$ class B networks with a maximum of $2^{16} = 65,536$ host addresses each
- $2^{21} = 2,097,152$ class C networks with a maximum of $2^8 = 256$ host addresses each
- Class D contains multicast addresses (e.g. for IPTV)
- Class E is reserved for future (?!) purposes and experiments

Why is the class E address space of IPv4 not used?

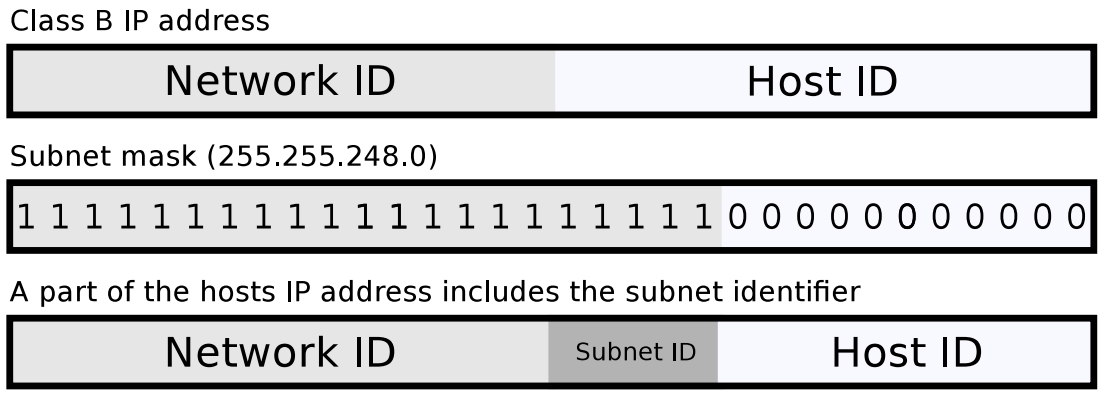
"The class E space has 268 million addresses and would give us in the order of 18 months worth of IPv4 address use. However, many TCP/IP stacks, such as the one in Windows, do not accept addresses from class E space and will not even communicate with correspondents holding those addresses. It is probably too late now to change this behavior on the installed base before the address space would be needed."

Source:
`http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_10-3/103_addr-cons.html`

Address Classes (2/2)

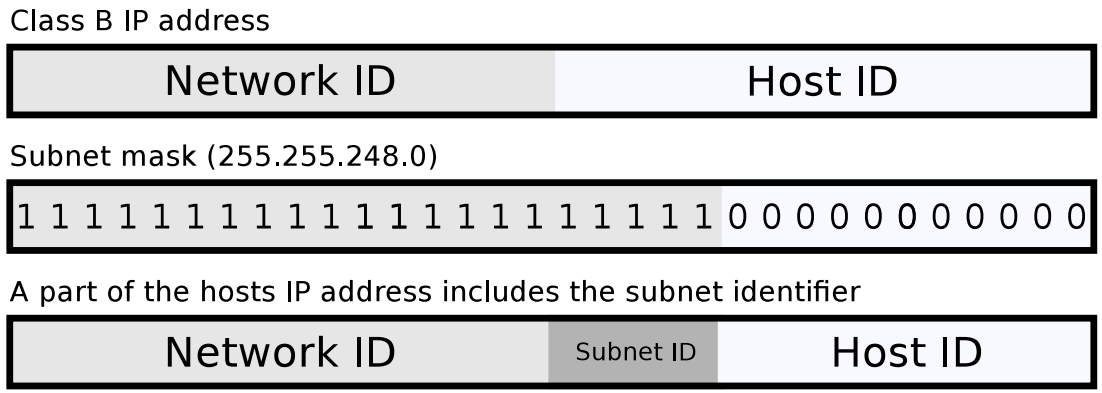
- Only the classes A, B and C are relevant in practice
- The original intention was to identify physical networks in an unique way via the network ID
  - This approach causes some drawbacks
- **Drawbacks of Address Classes:**
  - It is impossible to dynamically adjust them
  - Many addresses are wasted
    - A class C network with 2 devices wastes 253 addresses
    - The address space of class C networks is quite small
    - A class B network with 256 devices wastes $>$ 64,000 addresses
    - Only 128 class A networks exist
    - Migrating multiple devices to a different network class is complex task
- Solution: Logical networks are divided into **subnets**
  - 1993: Introduction of the **Classless Interdomain Routing** (CIDR)

Subnet Mask (1/2)



- For creating subnets, a **(sub-)netmask** is required
  - All hosts in a network have a subnet mask assigned
    - Length: 32 bits (4 bytes)
    - It is used to specify the number of subnets and hosts
- The subnet mask splits the host ID of an IP address into **subnet ID** and **host ID**
  - The network ID remains unchanged
  - The network mask adds another level of hierarchy into the IP address

Subnet Mask (2/2)



- Structure of the subnet mask:
  - 1-bits indicate, which part of the address space is used for subnet IDs
  - 0-bits indicate, which part of the address space is used for host IDs
- Example: Splitting a class B network into 20 subnets requires 5 bits
  - Each subnet requires its own subnet ID and it must be represented in binary form
  - If 5 bits are used for the representation of the subnet IDs, 11 bits remain for host IDs

Syntax of the Classless Interdomain Routing (CIDR)

- Since **CIDR** was introduced in 1993, IP address ranges are assigned in this notation: `First address/mask bits`
  - The number of mask bits indicates the number of 1-bits (prefix) in the subnet mask
- The table shows the possible splits of a class C network into subnets

Mask bits (prefix)	/24	/25	/26	/27	/28	/29	/30	/31	/32
Subnet mask	0	128	192	224	240	248	252	254	255
Subnet bits	0	1	2	3	4	5	6	7	8
Subnets IDs	1	2	4	8	16	32	64	128	256
Host bits	8	7	6	5	4	3	2	1	0
Host IDs	256	128	64	32	16	8	4	2	—
Hosts (maximum)	254	126	62	30	14	6	2	0	—

Not all Addresses can or should be used

Mask bits (prefix)	/24	/25	/26	/27	/28	/29	/30	/31	/32
Subnet mask	0	128	192	224	240	248	252	254	255
Subnet bits	0	1	2	3	4	5	6	7	8
Subnets IDs	1	2	4	8	16	32	64	128	256
Host bits	8	7	6	5	4	3	2	1	0
Host IDs	256	128	64	32	16	8	4	2	—
Hosts (maximum)	254	126	62	30	14	6	2	0	—

2 Host IDs cannot be assigned to network devices, because each (sub-)network requires. . .

- an address for the network itself (all host ID bits are 0 bits)
- a broadcast address to address all devices in network (all bits of the host ID are 1 bits)

2 subnet IDs should not be used

- The subnet IDs, consisting exclusively of 0 bits and 1 bits should not be used
  $\Longrightarrow$ This rule is obsolete, but still often followed
- Modern Routers and network software have no problem, when all possible subnet IDs are assigned to subnets

Determining the necessary Subnets Bits

Mask bits (prefix)	/24	/25	/26	/27	/28	/29	/30	/31	/32
Subnet mask	0	128	192	224	240	248	252	254	255
Subnet bits	0	1	2	3	4	5	6	7	8
Subnets IDs	1	2	4	8	16	32	64	128	256
Host bits	8	7	6	5	4	3	2	1	0
Host IDs	256	128	64	32	16	8	4	2	—
Hosts (maximum)	254	126	62	30	14	6	2	0	—

- By using the table, it is simple to determine the required bits for subnets
- Example: Subdivide a class C network into 5 subnets, each with a maximum of 25 hosts
  - Each subnet requires a subnet address
  - For representing 5 subnets, 3 subnet bits are required
  - The remaining 5 bits are used for representing the host IDs and they allow the addressing of $32 - 2 = 30$ hosts per subnet
  - Thus, the subnet mask with the prefix `/27` is well suited for this use case

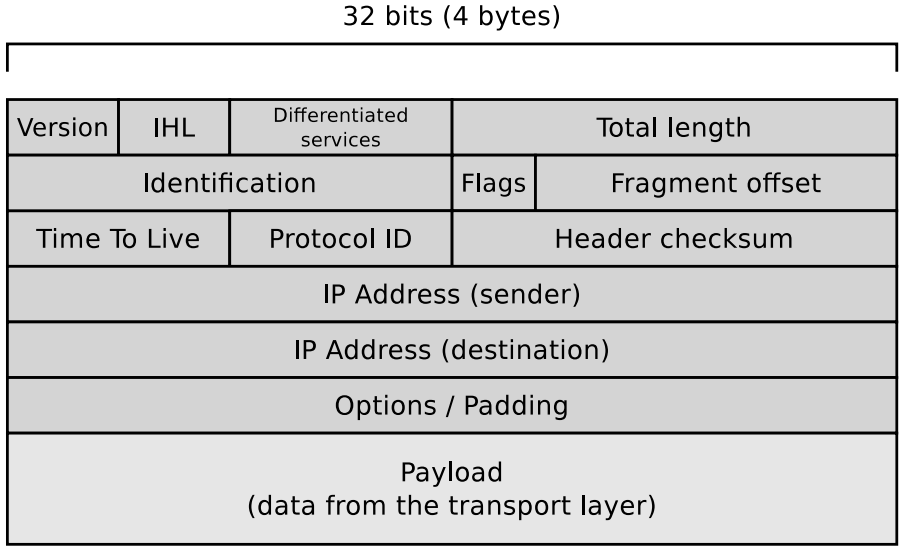
Private Networks – Private IP Address Spaces

- In private networks, it is also required to assign IPs to network devices
  - These addresses are not allowed to interfere with global accessible internet services
- Several address spaces exist, containing private IP addresses
  - These address spaces are **not routed** in the internet

Address space:	10.0.0.0 to 10.255.255.255
CIDR notation:	10.0.0.0/8
Number of addresses:	2^{24} = 16,777,216
Address class:	Class A. 1 private network with 16,777,216 addresses
Address space:	172.16.0.0 to 172.31.255.255
CIDR notation:	172.16.0.0/12
Number of addresses:	2^{20} = 1,048,576
Address class:	Class B. 16 private networks with 65,536 addresses each
Address space:	192.168.0.0 to 192.168.255.255
CIDR notation:	192.168.0.0/16
Number of addresses:	2^{16} = 65,536
Address class:	Class C. 256 private networks with 256 addresses each

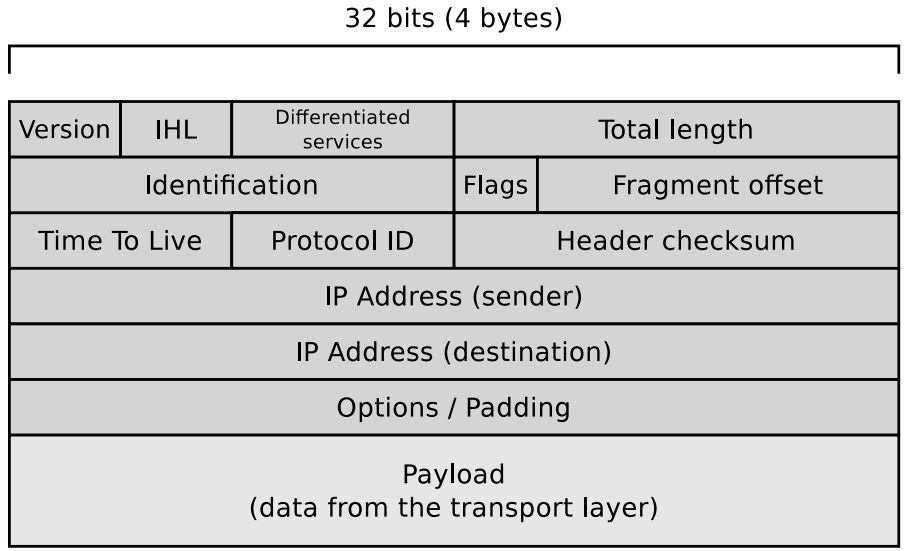
Structure of IPv4 Packets (1/6)

- **Version** (4 bits)
  - Protocol version
    - Version = 4 $\Longrightarrow$ IPv4
    - Version = 6 $\Longrightarrow$ IPv6



- **IHL** = IP Header Length (4 bits)
  - Header length, represented as the number of 4 byte words
    - Example: IHL = 5 $\Longrightarrow$ $5 * 4$ bytes = 20 bytes
  - Indicates where the payload begins
- **Differentiated services** (8 bits)
  - Prioritization of IP packets is possible with this field (Quality of Service)
  - The field slightly changed over the years (RFC 791, RFC 2474, RFC 3168)

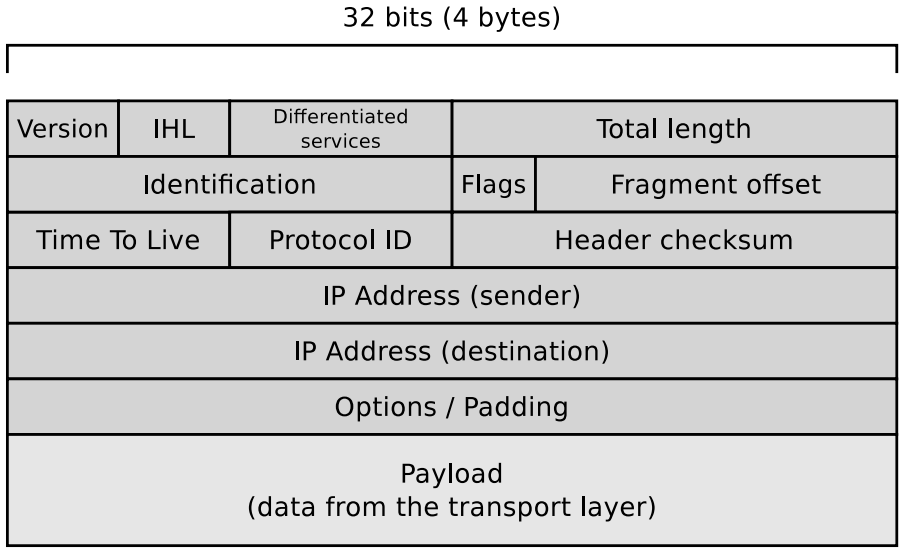
Structure of IPv4 Packets (2/6)



- **Total length** (16 bits)
  - This field defines the entire packet size (header and payload)
  - This length of the field is 16 bits and therefore the maximum possible IPv4 packet length is 65,535 bytes

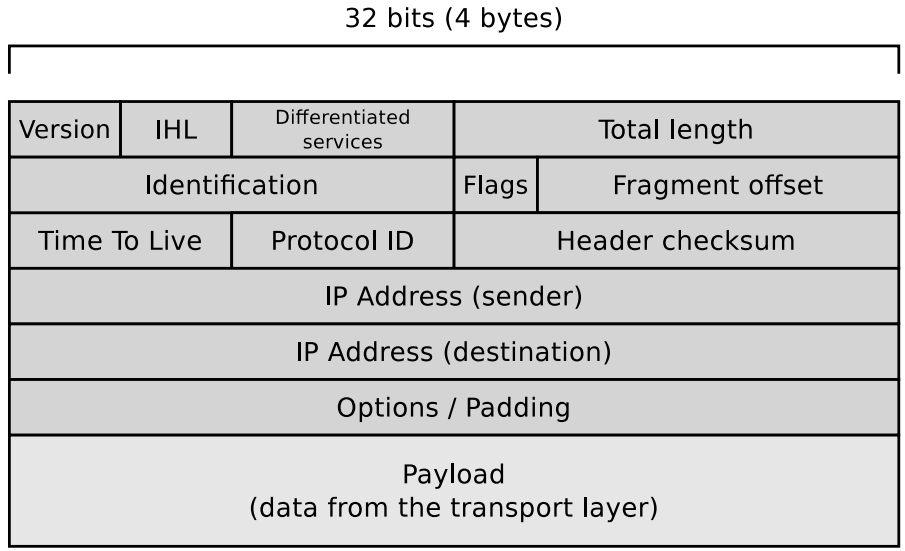
Structure of IPv4 Packets (3/6)

- The fields **Identification**, **Flags** and **Fragment offset** control the assembly of fragmented IP packets
- **Identification** (16 bits)
  - Contains a unique identifier of the IP packet
- **Flags** (3 bits)
  - Here the sender informs whether the packet can be fragmented and the receiver is informed whether more fragments follow
- **Fragment Offset** (13 bits)
  - Contains a number which states for fragmented packets, from which position of the unfragmented packet the fragment begins



More information about the fragmentation of IP packages provide the slides 53 + 54

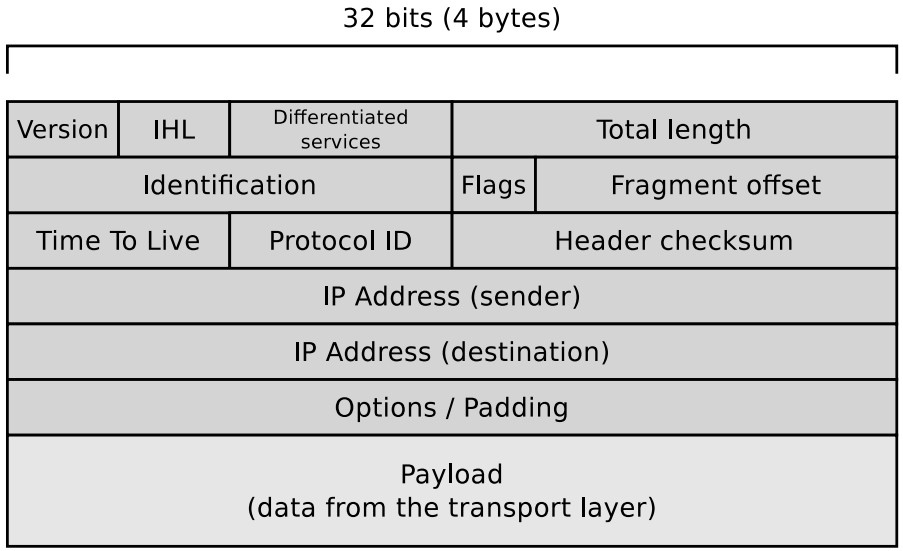
Structure of IPv4 Packets (4/6)



- **Time To Live** (8 bits)
  - Contains the maximum number of hops
    - Each Router on the route to the destination decrements the value by one
  - Prevents that undeliverable IP packets endlessly go in cycles on the network

Structure of IPv4 Packets (5/6)

- **Protokoll ID** (8 bits)
  - Contains the number of the Transport Layer protocol used
  - TCP segments $\Longrightarrow$ `6`
  - UDP segments $\Longrightarrow$ `17`
  - ICMP message $\Longrightarrow$ `1`
  - OSPF message $\Longrightarrow$ `89`

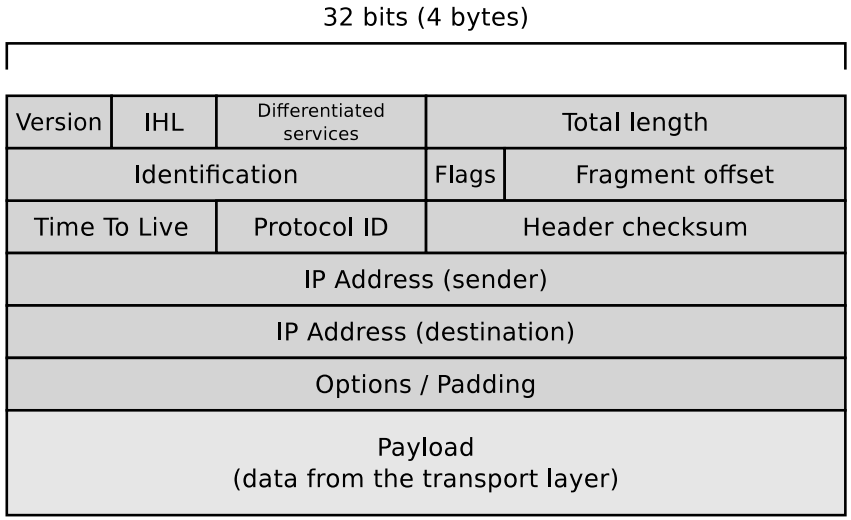


- Each IPv4 packet contains a checksum (16 bits) of the header
  - Because at each Router on the way to the destination, the content of the field **Time To Live** changes, each Router need to verify the checksum, recalculate and insert it into the header

Routers usually ignore the checksum to speedup the packet forwarding

Therefore, IPv6 packets contain no checksum field

Structure of IPv4 Packets (6/6)



- The field **IP address (sender)** (32 bits) contains the source address and **IP address (destination)** contains the destination address
- The field **Options** / **Padding** can contain additional information such as a time stamp
  - This last field before the payload area is filled with padding bits (0 bits) if necessary, to ensure that the header size is an integer number of 32 bit words
- The last field contains the data from the Transport Layer

Packet Fragmentation (1/2)

- The split up (and reassembling) of IP packets into smaller packets (**fragments**) is called **Packet fragmentation**
  - Is usually done by Routers
    - Packet fragmentation can also be carried out by the sender
- Reason for packet fragmentation:
  - The maximum packet length depends on the network technology used
- The **Maximum Transmission Unit** (MTU) specifies the maximum payload of a frame (and thus the maximum size of an IP packet too)
  - MTU of Ethernet: usually 1,500 bytes
    - For Gigabit Ethernet, *Jumboframes* exist with a size of up to 9,000 bytes
  - MTU of WLAN (IEEE 802.11): 2,312 bytes
  - MTU of Token Ring with 4 Mbit/s (IEEE 802.5): 4,464 bytes
  - MTU of Token Ring with 16 Mbit/s: 17,914 bytes
  - MTU of PPPoE (e.g. DSL): $\leq$ 1,492 bytes
  - MTU of ISDN: 576 bytes
  - MTU of FDDI: 4,352 bytes

Packet Fragmentation (2/2)

- IP packets contain a flag which can be used to prohibit fragmentation
  - If a Router needs to fragment a packet because it is too large to forward, but the fragmentation is prohibited in the packet, the Router discards the packet because he cannot forward it
- If a network device does not receive all fragments of an IP packet within a certain period of time (a few seconds), the network device discards all received fragments
- Routers can split IP packets into smaller fragments, if the MTU makes this necessary and it is not prohibited in the packets
  - **But no Router can assemble fragments of a packet to create a larger fragment**
    - Only the receiver can assemble fragments

Diagnosis and Error Messages via ICMP

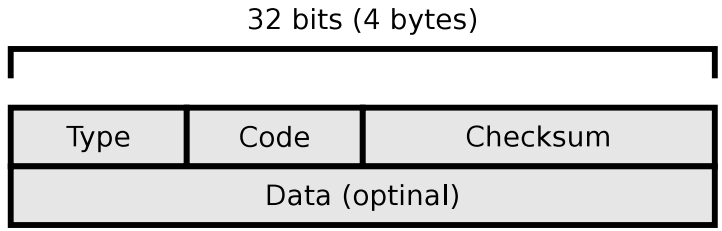
- The **Internet Control Message Protocol** (ICMP) is used for the exchange of...
  - diagnostic messages
  - control messages
  - error messages
- ICMP is a component (*sub-protocol*) of IPv4
  - but it is treated as a separate protocol

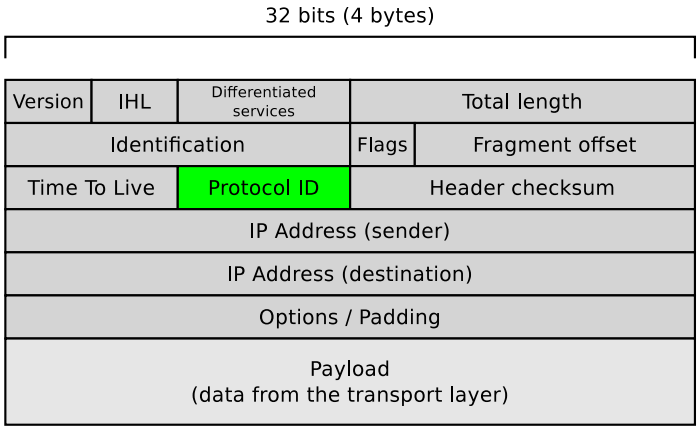
For IPv6, a similar protocol called ICMPv6 exists

- All Routers and terminal devices can handle ICMP
- Typical situations where ICMP is used:
  - A Router discards an IP packet, because it does not know how to forward it
  - Only a single fragment of an IP packet arrives at the destination
  - The destination of an IP packet cannot be reached, because the Time To Live (TTL) has expired

ICMP

- Example of an application, which sends ICMP packets: `ping`
- ICMP specifies different sorts of messages, which can be send by a Router as response to provide diagnostic information

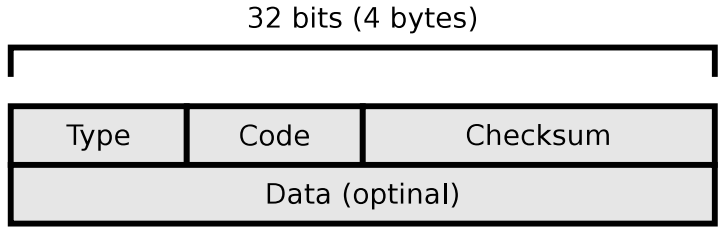




- ICMP messages are transmitted as payload inside IPv4 packets
  - In the header of the IPv4 packet the field **protocol ID** contains value `1`
  - For ICMPv6 the protocol ID is `58`
- If an ICMP packet cannot be delivered, no further action is done

ICMP Messages

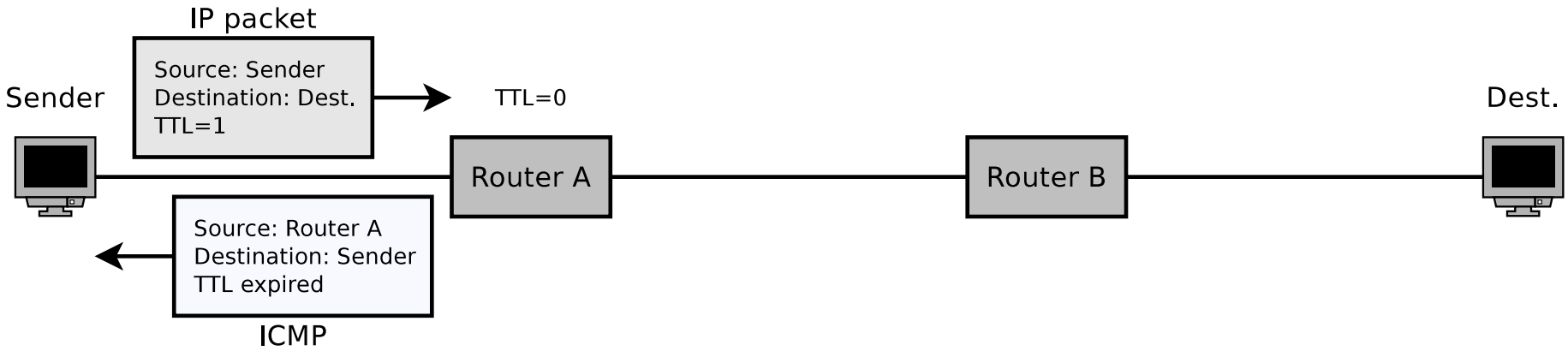
- The field **Type** in the ICMP header specifies the ICMP message type



  - The field **Code** specifies the subtype of the message type
  - The table contains some type-code combinations of ICMP messages

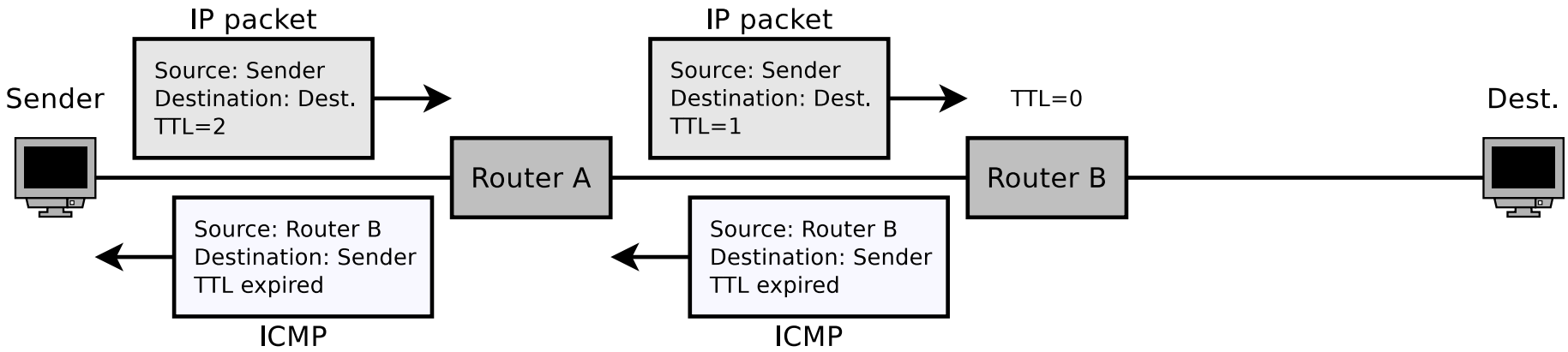
Type	Name of type	Code	Description
0	Echo reply	0	Echo reply (reply for `ping`)
3	Destination unreachable	0	Destination network unreachable
		1	Destination host unreachable
		2	Destination protocol unreachable
		3	Destination port unreachable
		4	Fragmentation required, but forbidden by the IP packet's flags
		13	Firewall at destination site rejects the IP packet
4	Source Quench	0	Receive buffer is full, IP packet discarded (congestion control)
8	Echo Request	0	Echo request (`ping`)
11	Time Exceeded	0	TTL (Time To Live) expired
17	Address Mask Request	0	Request for the number of bits in the subnet mask
18	Address Mask Reply	0	Reply to message type 17
		1	Time limit exceeded during defragmentation
30	Traceroute	0	Determine the route to the destination

Example of using ICMP: `traceroute` (1/3)



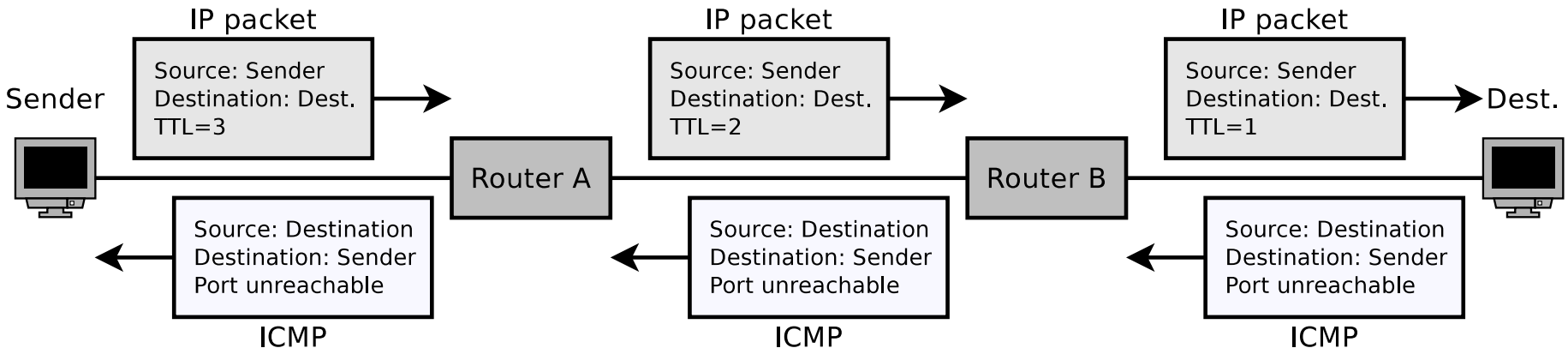
- Another application example of ICMP is the tool `traceroute`
- `traceroute` determines, which Routers are used to forward packets to the destination site
- The sender transmits an IP packet to the destination with `TTL=1`
- Router A receives the IP packet, sets `TTL=0`, discards the IP packet and transmits an ICMP message of message type `11` and code `0` to the sender

Example of using ICMP: `traceroute` (2/3)



- Next, the sender transmits an IP packet to the destination with `TTL=2`
- The IP packet is forwarded by Router A and thereby the value of `TTL` is decremented
- Router B receives the IP packet, sets `TTL=0`, discards the IP packet and transmits an ICMP message of message type `11` and code `0` to the sender

Example of using ICMP: `traceroute` (3/3)



- Once the value of `TTL` is big enough that the destination site can be reached, the receiver transmits an ICMP message of message type `3` and code `3` to the sender
- This way, the path from sender to receiver can be traced via ICMP

```
$ traceroute -q 1 wikipedia.de
traceroute to wikipedia.de (134.119.24.29), 30 hops max, 60 byte packets
1  fritz.box (10.0.0.1) 1.834 ms
2  p3e9bf6a1.dip0.t-ipconnect.de (62.155.246.161) 8.975 ms
3  217.5.109.50 (217.5.109.50) 9.804 ms
4  ae0.cr-polaris.fra1.bb.godaddy.com (80.157.204.146) 9.095 ms
5  ae0.fra10-cr-antares.bb.gdinf.net (87.230.115.1) 11.711 ms
6  ae2.cgn1-cr-nashira.bb.gdinf.net (87.230.114.4) 13.878 ms
7  ae0.100.sr-jake.cgn1.dcnet-emea.godaddy.com (87.230.114.222) 13.551 ms
8  wikipedia.de (134.119.24.29) 15.150 ms
```

Structure of IPv6 Addresses and Networks (1/5)

- IPv6 addresses have a length of 128 bits (16 bytes)
  - Therefore, $2^{128} \approx 3,4 * 10^{38}$ addresses can be represented
- The introduction is useful because of the limited address space of IPv4
- Problem: The decimal notation is confusing
  - For this reason, IPv6 addresses are represented in hexadecimal format
    - Groups of 4 bits are represented as a hexadecimal number
  - Goups of 4 hexadecimal numbers are merged into blocks
  - The blocks are separated by colons
    Example: `2001:0db8:85a3:08d3:1319:8a2e:0370:7344`

- The last 4 bytes (32 bits) of an IPv6 address may also be written in decimal notation
- This is useful to embed the IPv4 address space into the IPv6 address space
  $\Longrightarrow$ see slide 66

RFC 4291 (2006) *IP Version 6 Addressing Architecture*

Structure of IPv6 Addresses and Networks (2/5)

- Rules for simplification (RFC 5952):
  - Leading zeros within a block may be omitted
  - Successive blocks with value `0` (= `0000`), may be omitted **exactly 1 time within an IPv6 address**
    - If blocks are omitted, this is indicated by 2 consecutive colons
  - If several groups of null blocks exist, it is recommended to shorten the group with the most null blocks
- Example:
  - The IPv6 address of `j.root-servers.net` is:
    `2001:0503:0c27:0000:0000:0000:0002:0030`
    $\Longrightarrow$ `2001:503:c27::2:30`

Notation of IPv6 addresses (URLs)

- IPv6 addresses are enclosed in square brackets
- Port numbers are appended outside the brackets
  `http://[2001:500:1::803f:235]:8080/`
- This prevents the port number from being interpreted as part of the IPv6 address

Structure of IPv6 Addresses and Networks (3/5)

- IPv6 addresses consist of 2 parts

64 Bits	64 Bits
Network Prefix	Interface Identifier
2001:638:208:ef34	:0:ff:fe00:65

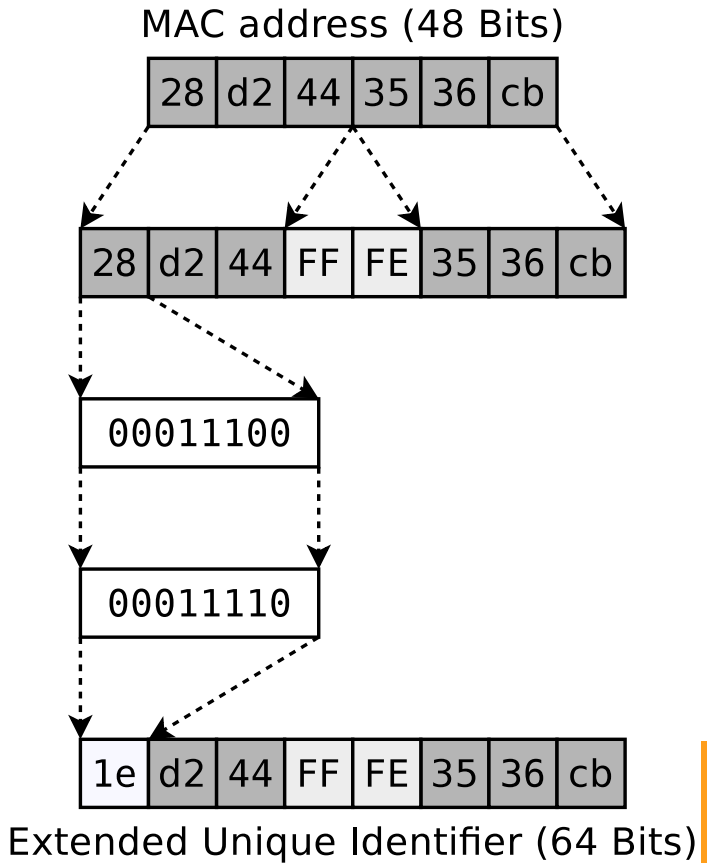
1. **Prefix** (Network Prefix)
   - Identifies the network
2. **Interface identifier** (Interface ID)
   - Identifies a network device in a network
   - Can be manually set, assigned via DHCPv6 or calculated from the MAC address of the network interface
   - If the interface identifier is calculated from the MAC address, it is called **Extended Unique Identifier** (EUI)
     - When this is done, the MAC address (48 bits) is converted into a 64-bit address $\Longrightarrow$ **modified EUI-64 address format**

Some address spaces

`fe80::/10` $\Longrightarrow$ Link local addresses. They are only valid in the local network and are therefore not forwarded by Routers
`2000::/3` $\Longrightarrow$ (`2000...` until `3fff...`) Global unicast addresses. Routers forward them
`ff00::/8` $\Longrightarrow$ All addresses `ff...` are multicast addresses. Since IPv6 has no broadcast addresses, multicast addresses implement the broadcast functionality. The addresses `ff01::1` and `ff02::1` address all nodes in the local network and the addresses `ff01::2`, `ff02::2` and `ff05::2` address all local Routers
`2001:db8::/32` $\Longrightarrow$ Addresses only for documentation purposes

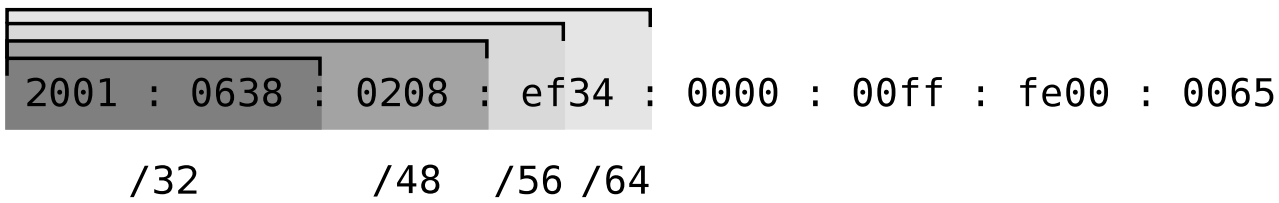
Structure of IPv6 Addresses and Networks (4/5)

- Converting a MAC address in the modified EUI-64 address format
  1. The MAC address is split into 2 parts of 24 bits
     - The 1st part becomes the first 24 bits
     - The 2nd part becomes the final 24 bits of the modified EUI-64 address
  2. The free 16 bits in the middle of the EUI-64 address have the following bit pattern:
     `1111 1111 1111 1110`
     (hex: `FFFE`)
  3. Finally, the value of the seventh bit from the left is inverted



Structure of IPv6 Addresses and Networks (5/5)

- (Sub-)netmasks do not exist in IPv6
  - The subdivision of address ranges into subnets is done by specifying the prefix length
- IPv6 networks are specified in CIDR notation
  - The address of a single device sometimes has `/128` attached
  - An example is the loopback address of IPv6: `::1/128`
    - All bits – except the last one – have value `0`
      (For IPv4, the loopback address is: `127.0.0.1`)
  - Internet Providers (ISPs) or operators of large networks get the first 32 or 48 bits assigned from a Regional Internet Registry (RIR)
    - The ISPs or network operators split this address space into subnets
    - **End users usually get a `/64` or even a `/56` network assigned**



- If a user gets a `/56` network assigned, the 8 Bits between the Prefix and the Interface Identifier are the **Subnet Prefix**

Embedd IPv4 Addresses into IPv6 (*IPv4 mapped*)

- A globally routed (unicast) IPv4 address can be represented as an IPv6 address and thus integrated into the IPv6 address space
  - In literature, this approach is called *IPv4 mapped*
- The IPv4 address gets a 96 bytes long prefix:
  `0:0:0:0:0:FFFF::/96`

80 Bits					16 Bits	32 Bits
0000	0000	0000	0000	0000	FFFF	IPv4 address

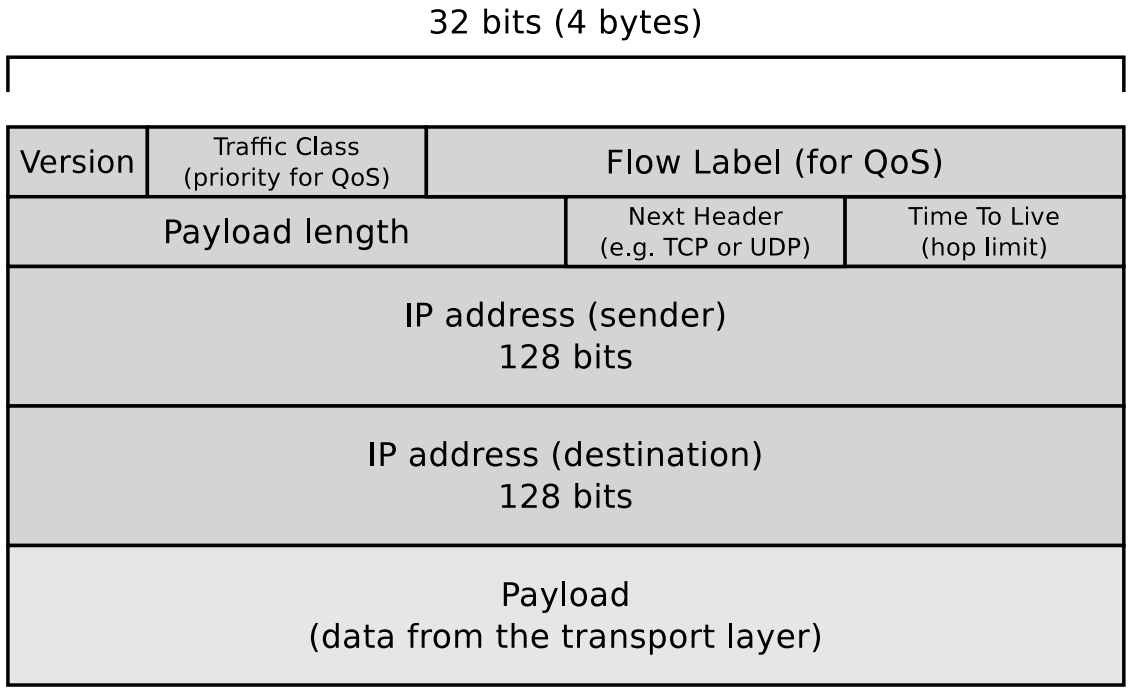
- The IPv4 address may be represented in hexadecimal or decimal notation

- Example

IPv4 address: `131.246.107.35`
IPv6 address: `0:0:0:0:0:FFFF:83F6:6B23`
Shorter notation: `::FFFF:83F6:6B23`
`::FFFF:131.246.107.35`

Structure of IPv6 Packets

- The size of the IPv6 header is fixed (320 bits $\Longrightarrow$ 40 bytes)



- The field **next header** points to an extension header field or identifies the Transport Layer protocol (e.g. TCP = type 6 or UTP = type 17) which is carried in the payload of the packet

Concept: Simplified (reduced) package structure, but simple option to add additional (new) features with a chain of extension headers

IPv6 extension headers (see RFC 2460 and RFC 4303) are not discussed in this course

Contents

Characteristics of Transport Layer Protocols

- Desired characteristics of Transport Layer protocols include...
  - guaranteed data transmission
  - ensuring the correct delivery order
  - support for data transmissions of any size
  - the sender must not overload the network
    - It must be able to adjust its own data flow (data rate) $\Longrightarrow$ flow control
  - the receiver should be able to control the transmission rate of the sender $\Longrightarrow$ congestion control
- Transport Layer protocols are required which convert the networks' negative characteristics into the (positive) characteristics that are expected of Transport Layer protocols
- The most common used Transport Layer protocols:
  - **UDP**
  - **TCP**
- The addressing is realized via **sockets**

Addressing in the Transport Layer

- Every application, which uses TCP or UDP, has a **port number** assigned
  - It specifies, which service is accessed
  - For TCP and UDP, the size of port numbers is 16 bits
    - Thus, the range of possible port numbers is from 0 to 65,535
- In principle, port numbers can be assigned as wished
  - Conventions exist, that specify, which ports are used by common used applications

Port number	Service	Description
21	FTP	File transfer
22	SSH	Encrypted terminal emulation (secure shell)
23	Telnet	Terminal emulation for remote control of computers
25	SMTP	E-mail transfer
53	DNS	Resolution of domain names into IP addresses
67	DHCP	Assignment of the network configuration to clients
80	HTTP	Webserver
110	POP3	Client access to E-mail server
143	IMAP	Client access to E-mail server
443	HTTPS	Webserver (encrypted)
993	IMAPS	Client access to E-mail server (encrypted)
995	POP3S	Client access to E-mail server (encrypted)

- The table contains only a small selection of well-known port numbers

Ports

- The port numbers are divided into 3 groups:
  - 0 to 1023 (**Well Known Ports**)
    - These are permanently assigned to applications and commonly known
  - 1024 to 49151 (**Registered Ports**)
    - Application developers can register port numbers in this range for own applications
  - 49152 to 65535 (**Private Ports**)
    - These port numbers are not registered and can be used freely

- Different applications can use identical port numbers inside an operating system at the same time, if they communicate via different transport protocols
- In addition, some applications exist, which implement communication via TCP and UDP via a single port number
- Example: Domain Name System – DNS

- Well Known Ports and Registered Ports are assigned by the Internet Assigned Numbers Authority (IANA)
- In Linux/UNIX systems, the configuration file `/etc/services` exists
  - Here, the applications (services) are mapped to specific port numbers
- In Windows systems: `%WINDIR%\system32\drivers\etc\services`

Sockets

- **Sockets** are the platform-independent, standardized **interface** between the implementation of the network protocols in the operating system and the applications
- **A socket consists of a port number and an IP address**
- Stream Sockets and datagram sockets exist
  - **Stream sockets** use the connection-oriented TCP
  - **Datagram sockets** use the connectionless UDP

Tool(s) to monitor the open ports and sockets with. . .

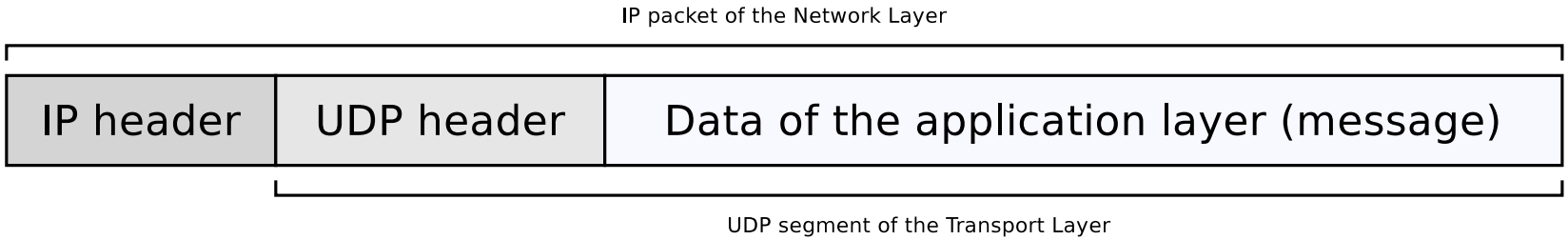
- Linux/UNIX: `netstat`, `lsof` and `nmap`
- Windows: `netstat`

Alternatives for sockets to implement inter-process communication (IPC)

Pipes, message queues and shared memory $\Longrightarrow$ see Operating Systems course

User Datagram Protocol (UDP)

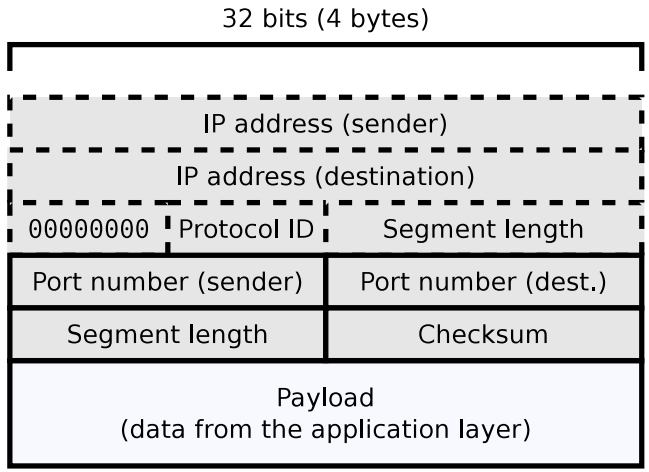
- Maximum size of an UDP segment: 65,535 Bytes
  - Reason: The size of the **length** field inside the UDP header, which contains the segment length, is 16 bits
    - The maximum representable number with 16 bits is 65,535
  - UDP segments of this size are transmitted fragmented by IP



UDP standard: RFC 768 from 1980
`http://tools.ietf.org/rfc/rfc768.txt`

Structure of UDP Segments

- The UDP header consists of 4 fields, each of 16 bits size
  - **Port number (sender)**
    - The field can stay empty (value `0`), if no response is required
  - **Port number (destination)**
  - **Length** of the complete segment (without pseudo-header)
  - **Checksum** of the complete segment (including pseudo-header)
- A pseudo-header is created, which includes the IP addresses of sender and destination, as well as some Network Layer information
  - Protocol ID of UDP = 17
- The pseudo-header is not transmitted
  - But it is used for the checksum

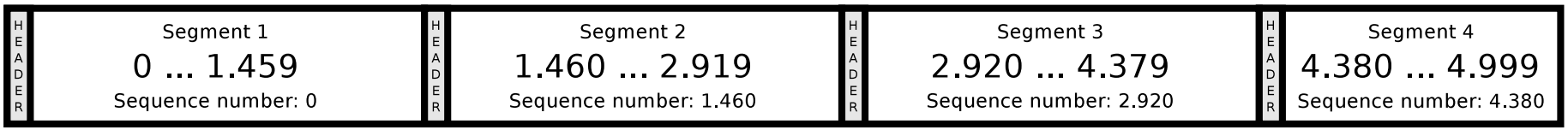


Remember NAT from slide set 8. . .

If a NAT device (Router) is used, this routing device also needs to recalculate the checksums in UDP segments when doing IP address translations

Sequence Numbers in TCP

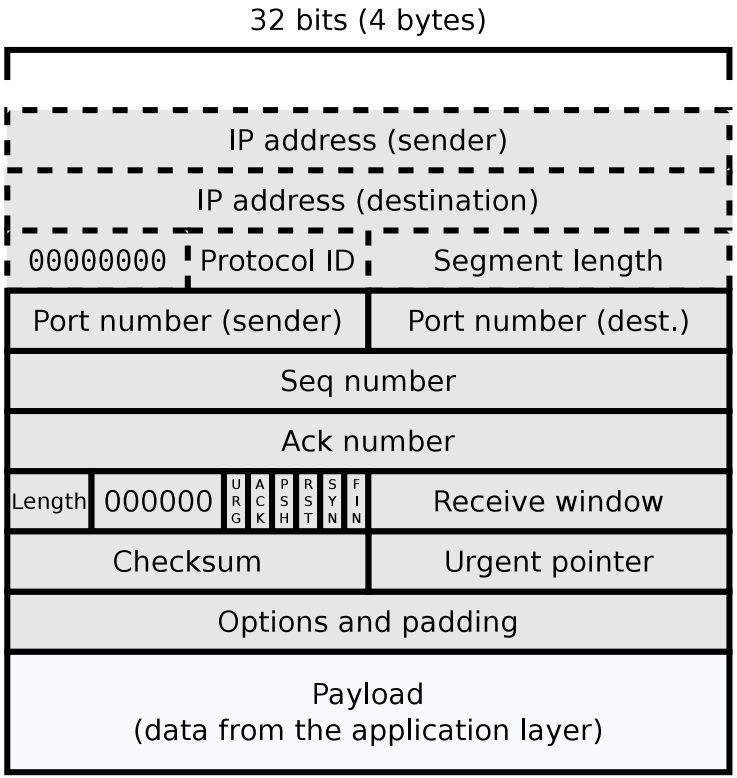
- TCP treats payload as an unstructured, but ordered data stream
- **Sequence numbers** are used for numbering the bytes in the data stream
  - The sequence number of a segment is the position of the segments first byte in the data stream
- Example
  - The sender splits the Application Layer data stream into segments
    - Length of data stream: 5,000 bytes
    - MSS: 1,460 bytes



Some key figures. . .

Maximum Transfer Unit (MTU): Maximum size of the IP packets
MTU of Ethernet = 1,500 bytes, MTU of PPPoE (e.g. DSL) = 1,492 bytes
Maximum Segement Size (MSS): Maximum segment size
MSS = MTU - 40 bytes for IPv4 header and TCP header

Structure of TCP Segments (1/5)



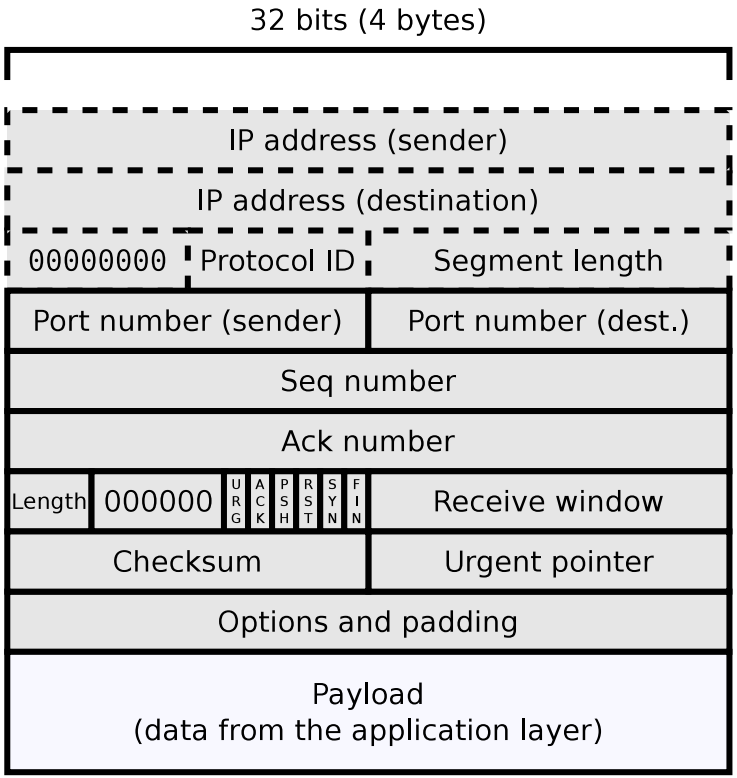
- A TCP segment can contain a maximum of 64 kB payload (data of the Application Layer)
  - Usually, segments are smaller ($\leq$ 1500 bytes for Ethernet)
- The header of TCP segments is more complex compared with UDP segments

Overhead

- Size of the TCP header (without the options field): just 20 bytes
- Size of the IP header (without the options field): also just 20 bytes

$\Longrightarrow$ The overhead, caused by the TCP and IP headers, is small for an IP packet with a size of several kB

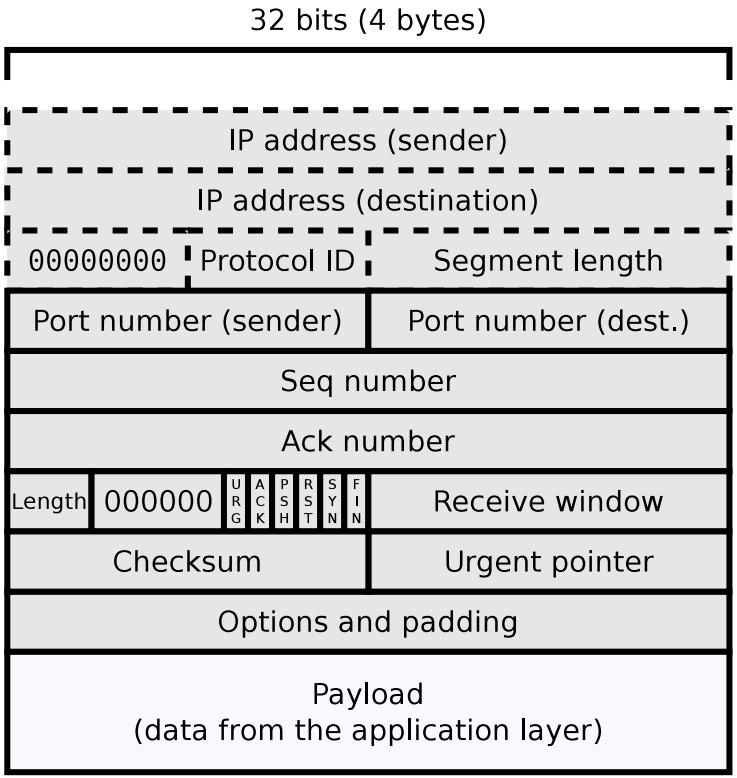
Structure of TCP Segments (2/5)



- One field contains the port number of the sender process
- Another field contains the port number of the process which is expected to receive the segment
- **Seq number** contains the sequence number of the current segment
- **Ack number** contains the sequence number of the next expected segment

- The **length** field specifies the size of the TCP header in 32-bit words to tell the receiver where the payload starts in the segment
  - The field is required, because the field **options and padding** can have a variable length (a multiple of 32 bits)

Structure of TCP Segments (3/5)

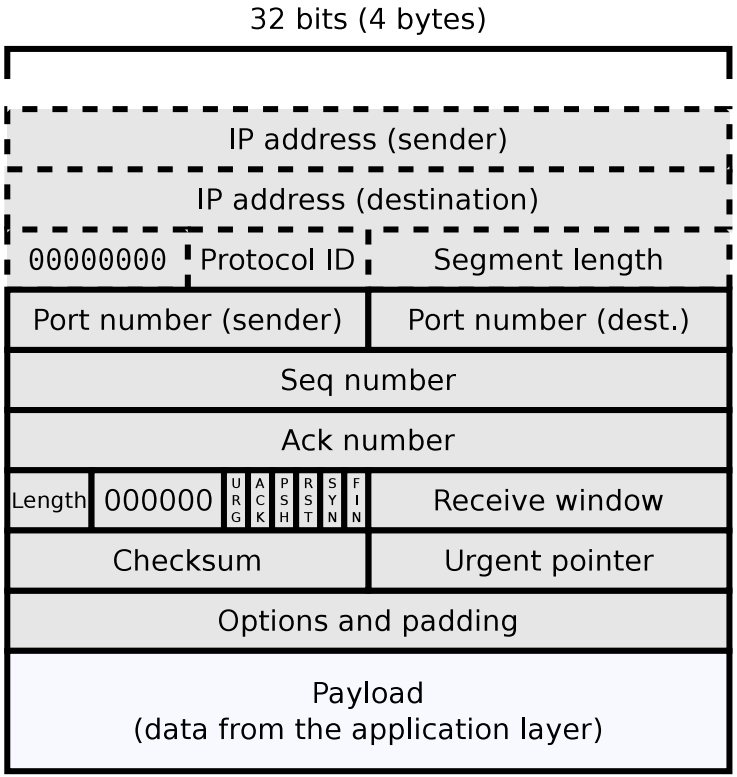


- The field `000000` is 6 bits long and not used
  - It contains per default value zero
- The 6 fields with a size of 1 bit each are required for connection establishment, data exchange and connection termination
  - The functionality of these fields is described with the assumption that the fiels contain the value `1` $\Longrightarrow$ it is *set*

URG (Urgent) is not discussed in this course

- **ACK** (Acknowledge)
  - Specifies that the acknowledgement number in **Ack number** is valid
  - It is also used to acknowledge the receive of segments

Structure of TCP Segments (4/5)



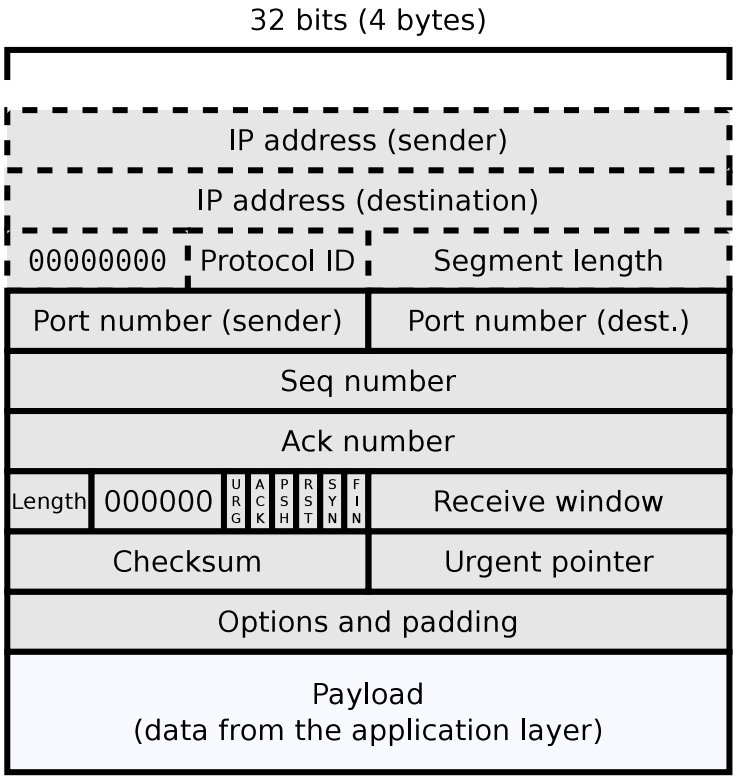
PSH (Push) is not discussed in this course

RST (Reset) is not discussed in this course

- **SYN** (Synchronize)
  - Requests the synchronization of the sequence numbers
  - That initiates the connection establishment
- **FIN** (Finish)
  - Requests the connection termination and indicates that the sender will not send any more payload

- The field **receive window** contains the number of free bytes in the sender's receive window, which is necessary for flow control

Structure of TCP Segments (5/5)



- Just as with UDP, for each TCP segment, a pseudo header exists, which is not transmitted
  - But the pseudo header fields are used together with the regular TCP header fields and the payload to calculate the **checksum**
  - **Protocol ID** of TCP = 6

The **urgent pointer** is not discussed in this course

The field **options and padding** must be a multiple of 32 bits and is not discussed in this course

Remember NAT. . .

If a NAT device (Router) is used, this routing device also needs to recalculate the checksums in TCP segments when doing IP address translations

Functioning of TCP

You already know. . .

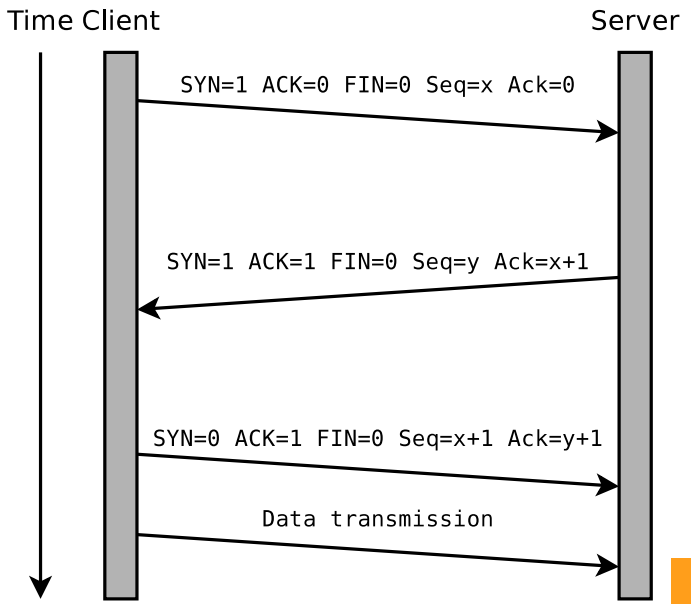
- Each segment has a unique **sequence number**
- The sequence number of a segment is the position of the segments first byte in the data stream

- The sequence number enables the receiver to. . .
  - correct the order of the segments
  - sort out segments, which arrived twice
- The length of a segment is known from the IP header
  - This way, missing bytes in the data stream are discovered and the receiver can request lost segments
- To establish a connection, TCP uses a **three-way handshake**, where both communication partners exchange control information in three steps
  - This ensures that the communication partner exists and data transmissions accepts

TCP Connection Establishment (three-way Handshake)

- The server waits passively for an incoming connection

1. Client sends a segment with SYN=1 as a request to synchronize the sequence numbers
   $\Longrightarrow$ **Synchronize**
2. Server sends as confirmation a segment with ACK=1 and requests with SYN=1 to synchronize the sequence numbers too
   $\Longrightarrow$ **Synchronize Acknowledge**
3. Client confirms with a segment with ACK=1 that the connection is established
   $\Longrightarrow$ **Acknowledge**



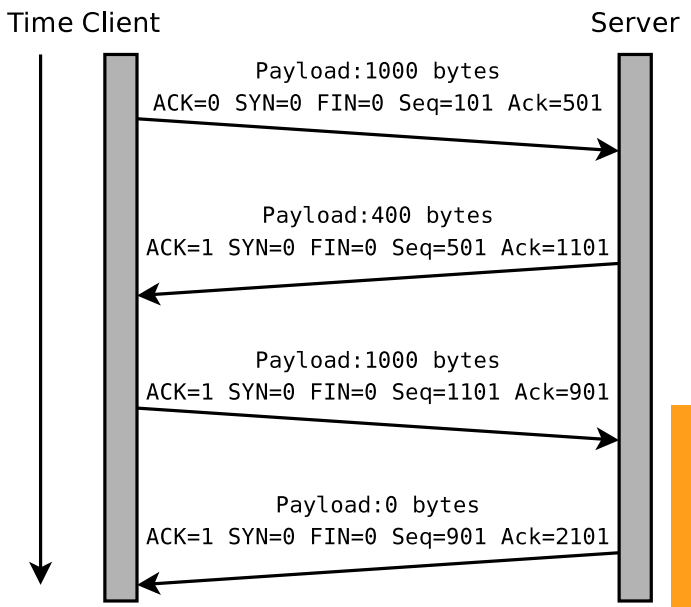
- The initial sequence numbers (x and y) are determined randomly
- No payload is exchanged during connection establishment!

TCP Data Transmission

To demonstrate a data transmission, **Seq number** (sequence number of the current segment) and **Ack number** (sequence number of the expected next segment) need particular values

- In our example at the beginning of the three-way handshake, the client's sequence number is `x`=100 and the server's sequence number is `y`=500
- After completion of the three-way handshake: `x`=101 and `y`=501

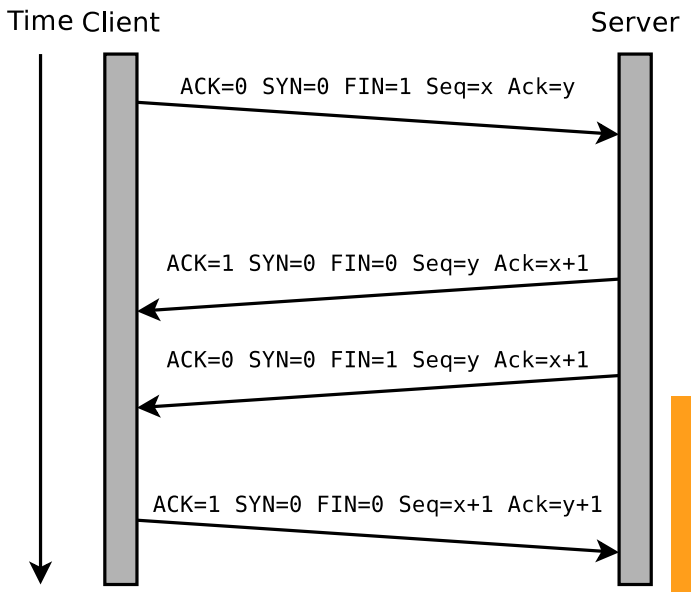
1. The client transmits 1000 bytes payload
2. Server acknowledges with ACK=1 the received payload and requests with the Ack number 1101 the next segment. In the same segment, the server transfers 400 bytes of payload
3. The client transmits another 1000 bytes payload. And it acknowledges the received payload with the ACK bit set and requests with the Ack number 901 the next segment
4. Server acknowledges with ACK=1 the received payload and requests with the Ack number 2101 the next segment



TCP Connection Termination

- Connection termination is similar to the connection establishment
- Instead of the SYN bit set, the FIN bit is used to indicate that the sender will not transmit any more payload

1. The client sends the request for connection termination with FIN=1
2. The server sends an acknowledgment with ACK=1
3. The server sends the request for connection termination with FIN=1
4. The client sends an acknowledgment with ACK=1

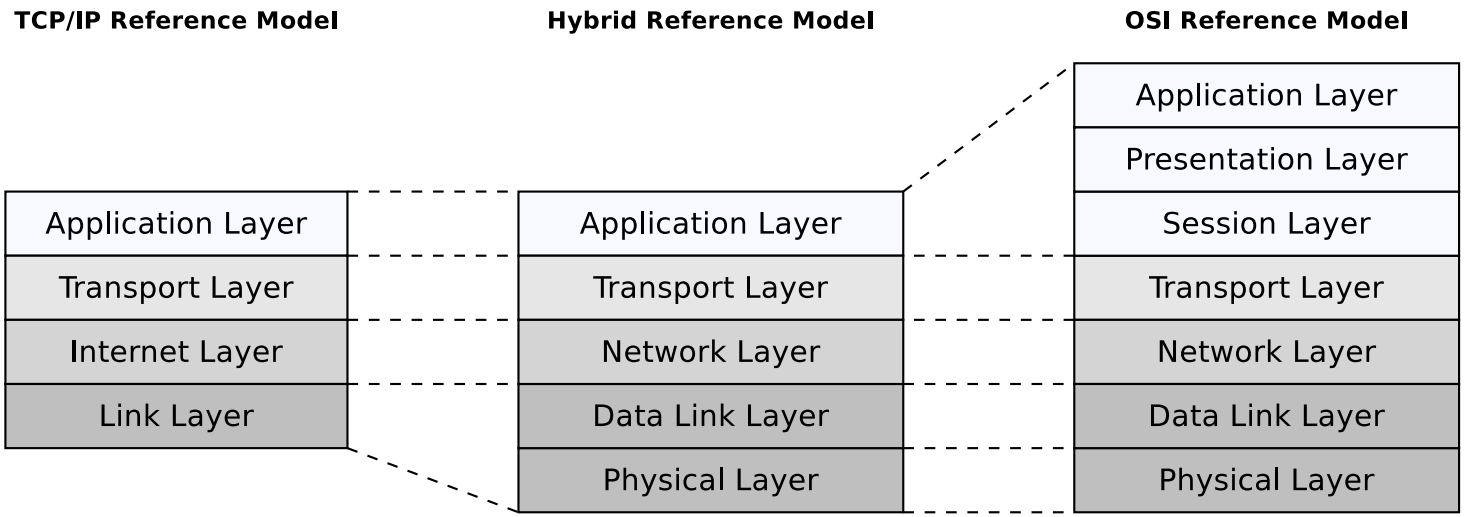


- No payload is exchanged during connection termination!

Contents

Application Layer

- Contains the protocols, which interact with applications (e.g. browser or email client)
- Contains the messages of the users and their applications (e.g. HTML pages or emails) in accordance with the Application Layer protocol used



- Devices: none
- Protocols: DNS, DHCP, NTP, Telnet, SSH, HTTP, SMTP, FTP...

Hypertext Transfer Protocol (HTTP)

- The Hypertext Transfer Protocol (HTTP) is a stateless protocol for data transmission
  - Stateless means that every HTTP message contains all the information necessary to understand the message
  - The server does not maintain any information regarding the state or session for the client, and each request is a transaction, independent of other requests

HTTP

From 1989 onwards, developed by Roy Fielding, Tim Berners-Lee and other at CERN

- Together with the concepts of URL and HTML it is the basis of the World Wide Web (WWW)
- Main purpose: Loading web pages from the World Wide Web (WWW) in a browser
- For communication, HTTP needs a reliable transport protocol
  - In almost all cases, TCP is used
- Each HTTP message consists of:
  - Message header (*HTTP header*): Includes among others Information about the encoding, desired language, browser and content type
  - Message body (*body*): Contains the payload, e.g. the HTML source code of a web page

HTTP Requests (1/2)

- If an URL is accessed via HTTP (e.g. `http://www.informatik.hs-mannheim.de/~baun/index.html`, the request for the resource `/~baun/index.html` is transmitted to the computer with hostname `www.informatik.hs-mannheim.de`
- First, via DNS, the hostname is resolved to an IP address
- Next, this HTTP `GET` request is transmitted via TCP to port 80, where the web server usually operates

```
GET /~baun/index.html HTTP/1.1
Host: www.informatik.hs-mannheim.de
User-Agent: Mozilla/5.0 (X11; U; Linux i686; de; rv:1.9.2.18) Gecko/20110628 Ubuntu/10.10
     (maverick) Firefox/3.6.18
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
...
```

HTTP Requests (2/2)

- A this large message header is not required
- The HTTP `GET` request below is sufficient

```
GET /~baun/index.html HTTP/1.1
Host: www.informatik.hs-mannheim.de
```

- The header of a HTTP message is separated from the message body with a line feed (`LF`) and a carriage return (`CR`)
  - In this example, the HTTP request has no message body

HTTP Responses (1/2)

- The HTTP response of the web server consists of a message header and the message body with the actual message
  - In this case, the message body contains the content of the requested file `index.html`

```
HTTP/1.1 200 OK
Date: Sun, 04 Sep 2011 15:19:13 GMT
Server: Apache/2.2.17 (Fedora)
Last-Modified: Mon, 22 Aug 2011 12:37:04 GMT
ETag: "101ec1-2157-4ab17561a3c00"
Accept-Ranges: bytes
Content-Length: 8535
Keep-Alive: timeout=13, max=499
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
...
</html>
```

HTTP Responses (2/2)

- Each HTTP response contains a **status code**, which consists of 3 digits, and a text string, which describes the reason for the response

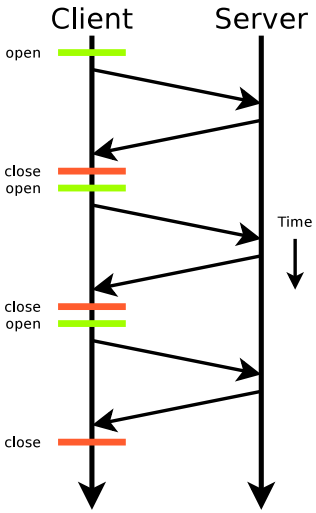
Status code	Meaning	Description
`1xx`	**Informational**	Request received, continuing process
`2xx`	**Success operation**	Action received, understood, accepted and processed successfully
`3xx`	**Redirection**	Additional action must be taken by the client to complete the request
`4xx`	**Client error**	Request of the client caused an error situation
`5xx`	**Server error**	Server failed to fulfill a valid request $\Longrightarrow$ error was caused by server

- The table contains some common status codes of HTTP

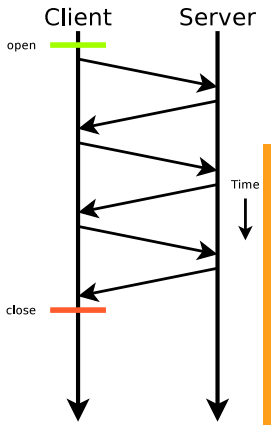
Status code	Meaning	Description
`200`	`OK`	Request processed successfully. Result is transmitted in the response
`202`	`Accepted`	Request accepted, but will be executed at a later point in time
`204`	`No Content`	Request executed successfully. Response intentionally contains no data
`301`	`Moved Permanently`	The old address is no longer valid
`307`	`Temporary Redirect`	Resource moved. The old address remains valid
`400`	`Bad Request`	Request cannot be fulfilled due to bad syntax
`401`	`Unauthorized`	Request can not be executed without a valid authentication
`403`	`Forbidden`	Request is executed because of clients lack of privileges
`404`	`Not Found`	Server could not find the requested resource
`500`	`Internal Server Error`	Unexpected server error

HTTP Protocol Versions (HTTP/1.0 and HTTP/1.1)

- 3 protocol versions exist: `HTTP/1.0`, `HTTP/1.1` and `HTTP/2`



- HTTP/1.0 (RFC 1945): Prior to any request, a new TCP connection is established and closed by default by the server after the transmission of the reply
  - If a HTML document contains e.g. 10 images, 11 TCP connections are required for the transmission to the client
- HTTP/1.1 (RFC 2616): By default, no connection termination is done
  - So the connection can be used again and again
  - Therefore, only a single TCP connection is required for the transfer of a HTML document with 10 images
    - Result: The document download finishes in a shorter time
  - Interrupted transmissions can be resumed with



HTTP Protocol Versions (HTTP/2)

- `HTTP/2` (RFC 7540) became a standard in May 2015
- Accelerates the data transfer by compressing the header with the HPACK algorithm (RFC 7541)
- Enables the aggregation (*Multiplex*) of requests and a server can send (*Server Push*) data automatically, which it expects the browser to request immediately
  - Examples of such data are CSS files (Cascading Style Sheets), which specify the layout of web pages, or script files
- `HTTP/2` is not a text-based but a binary protocol
  - Therefore it cannot be used to communicate using simple tools like `telnet` and `nc` e.g. to inspect a server
  - Tools like `curl` and `openssl -connect` can communicate via HTTP/2

Some sources about `curl` and `openssl`

`https://stackoverflow.com/questions/51278076/curl-one-liner-to-test-http-2-support`
`https://blog.cloudflare.com/tools-for-debugging-testing-and-using-http-2/`
Stephen Ludin, Javier Garza. **Learning HTTP/2: A Practical Guide for Beginners**. O'Reilly Media, Inc (2017)

HTTP Methods

- The HTTP protocol provides some methods for requests

HTTP	Description
`PUT`	Upload a new resource to the web server
`GET`	Request a resource from the web server
`POST`	Upload data to the web server in order to generate resources
`DELETE`	Erase a resource on the web server
`HEAD`	Request the header of a resource from the web server, but not the body
`TRACE`	Returns the request back, as the web server has received it. Helpful for troubleshooting purposes
`OPTIONS`	Request the list of supported HTTP methods from the web server
`CONNECT`	Establish a SSL tunnel with a proxy

HTTP is a stateless protocol. But via cookies in the header information, applications can be implemented which require state or session information because they assign user information or shopping carts to clients.