# Introduction to Networking (NSWI141)

**Libor Forst, SISAL MFF UK**

- Essential facts concerning communications
- Layered Network model (OSI vs. TCP/IP, addressing, multiplexing, ...)
- Application layer (DNS, FTP, email, web, VoIP, ...)
- Transport layer
- Network layer (IPv4, IPv6, routing, firewalls, ...)
- Data link and physical layer (switch vs. repeater, Ethernet, Wi-Fi, cabeling, ...)

# Literature

- D. E. Comer, D. L. Stevens: Internetworking With TCP/IP; Prentice Hall 1991
- A. S. Tanenbaum: Computer Networks; Prentice Hall 2003
- C. Hunt: TCP/IP Network Administration; O'Reilly & Associates 1992

- internet resources

- Request For Comment (RFC)
- **`http://www.warriorsofthe.net`**

# General attributes of communication

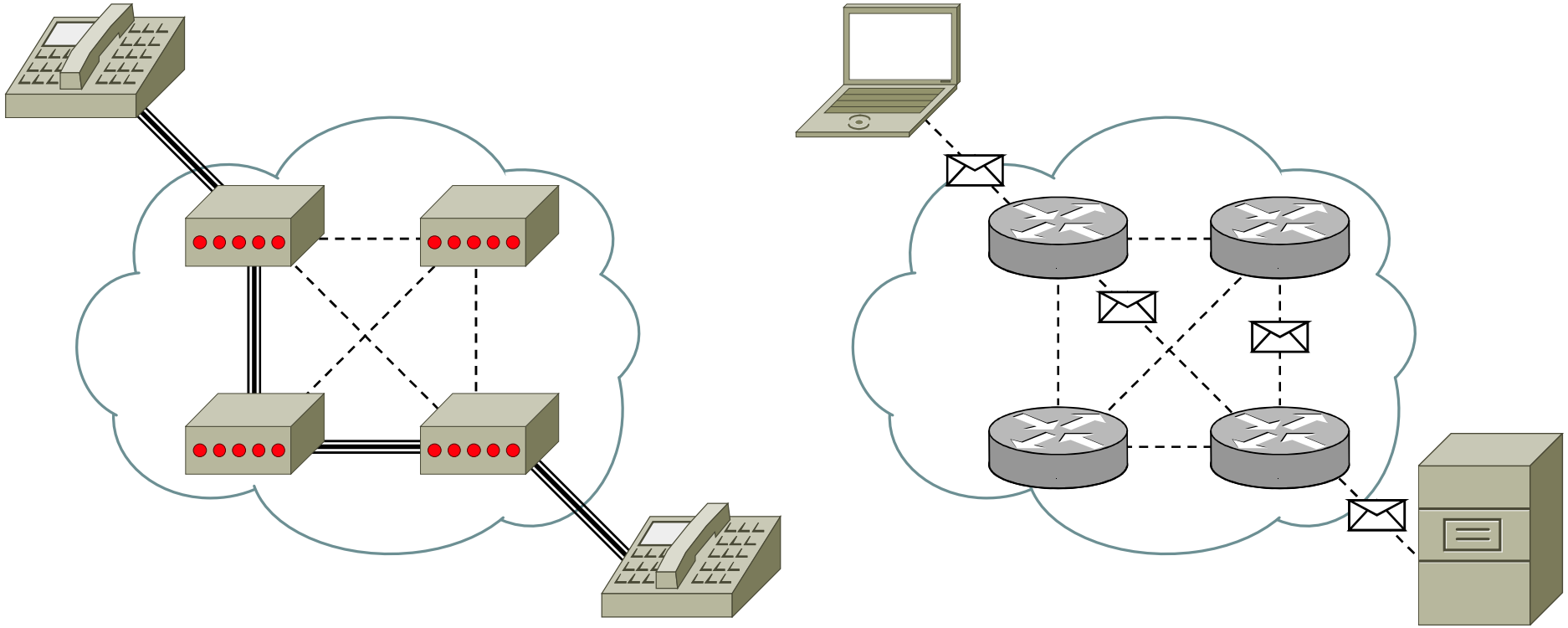
- Identification
  - actors must „find“ themselves (phone numbers) and introduce each other
- Method
  - e.g.: a deaf man at a counter he tries sign language, the officer doesn’t understand and suggests written form of communication
- Language
  - both sides must agree on a common language
- Speed
  - both sides must agree on a communication speed
- Process
  - requests, answers, confirmations

# Types of communication

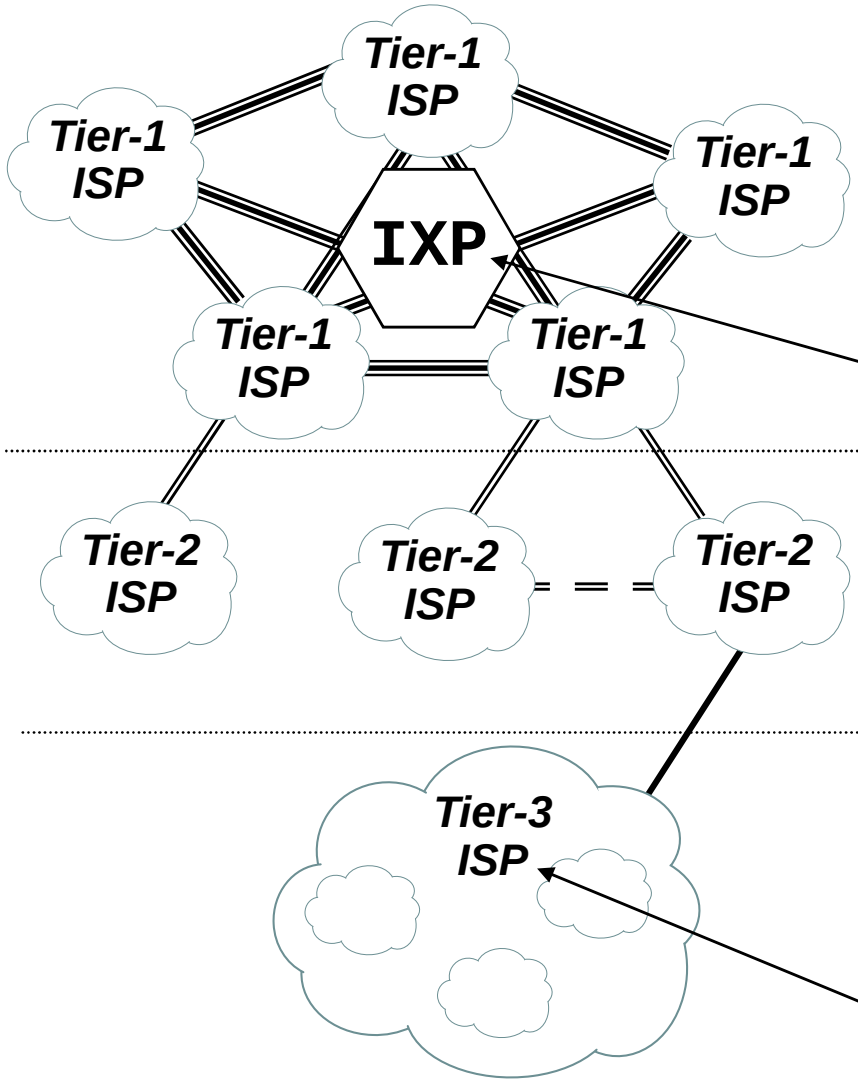
- Ordinary human communications
  - voice, signals, writing
  - loose intuitive rules
- Telecommunication
  - complex technology with embedded rules
  - entire network (incl. end devices) is under centralized control
- Computer network
  - rules are open and freely accessible
  - most of logic is moved to end devices
  - network controls just the transmission
- Converged network
  - joins telecomm. and computer worlds (price, effectiveness...)
  - better is convergence based on computer network

# Requirements - fault tolerance

- circuit switching: faster, fluent, however failures disrupt whole connection
- packet switching: packets use various ways => transport time may differ, but network can overcome node failures



# Requirements - scalability (WAN)



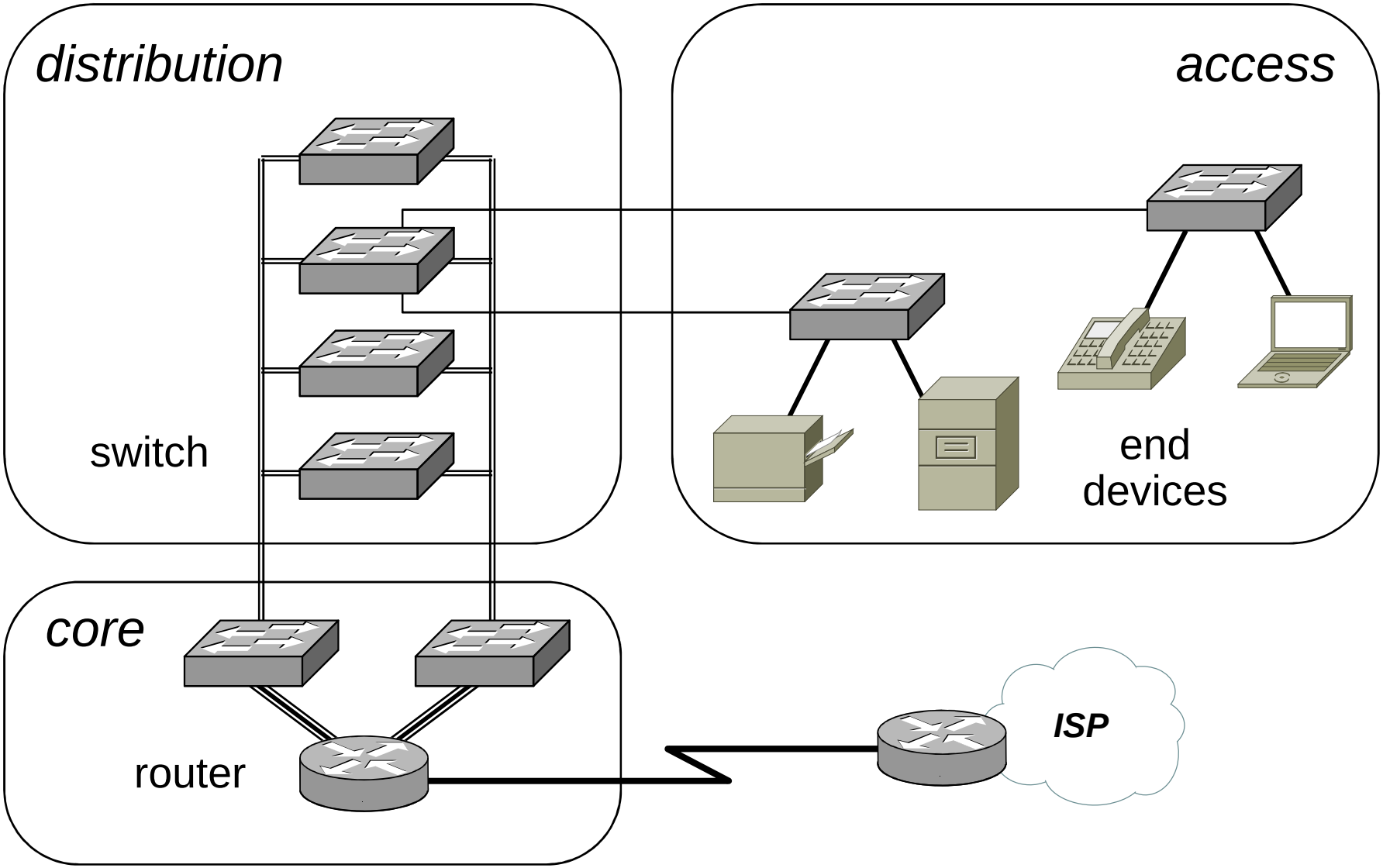

- Tier 1: key players (having direct access to backbone)

*Internet Exchange Point*

- Tier 2: depend on ISP Tier 1, partial peering, usually without end customers
- Tier 3: depend on ISP Tier 2, connect end customers

*Internet Service Provider*

# Requirements - scalability (LAN)



# Requirements - security

- Quite new requirement, old technologies were naive:
  - open communication (tapping possible)
  - full confidence in partner identity
  - content trust
- Security viewpoints:
  - infrastructure (physical) security
  - data (logical) security
- Current methods:
  - user authentication, access rights control
  - host authentication (servers, clients)
  - data inspection (application proxy, antivirus, antispam, ...)
  - cryptography (ciphers, encryption, subscription)

# Requirements - quality of service

- Various applications have various requirements
  - latency, delay
  - evenness of delivery (*jitter*, variance of delay)
    - both parameters crucial for multimedia applications
  - data loss
    - crucial for data (files) transport (WWW, e-mail)
  - bandwidth („speed“)

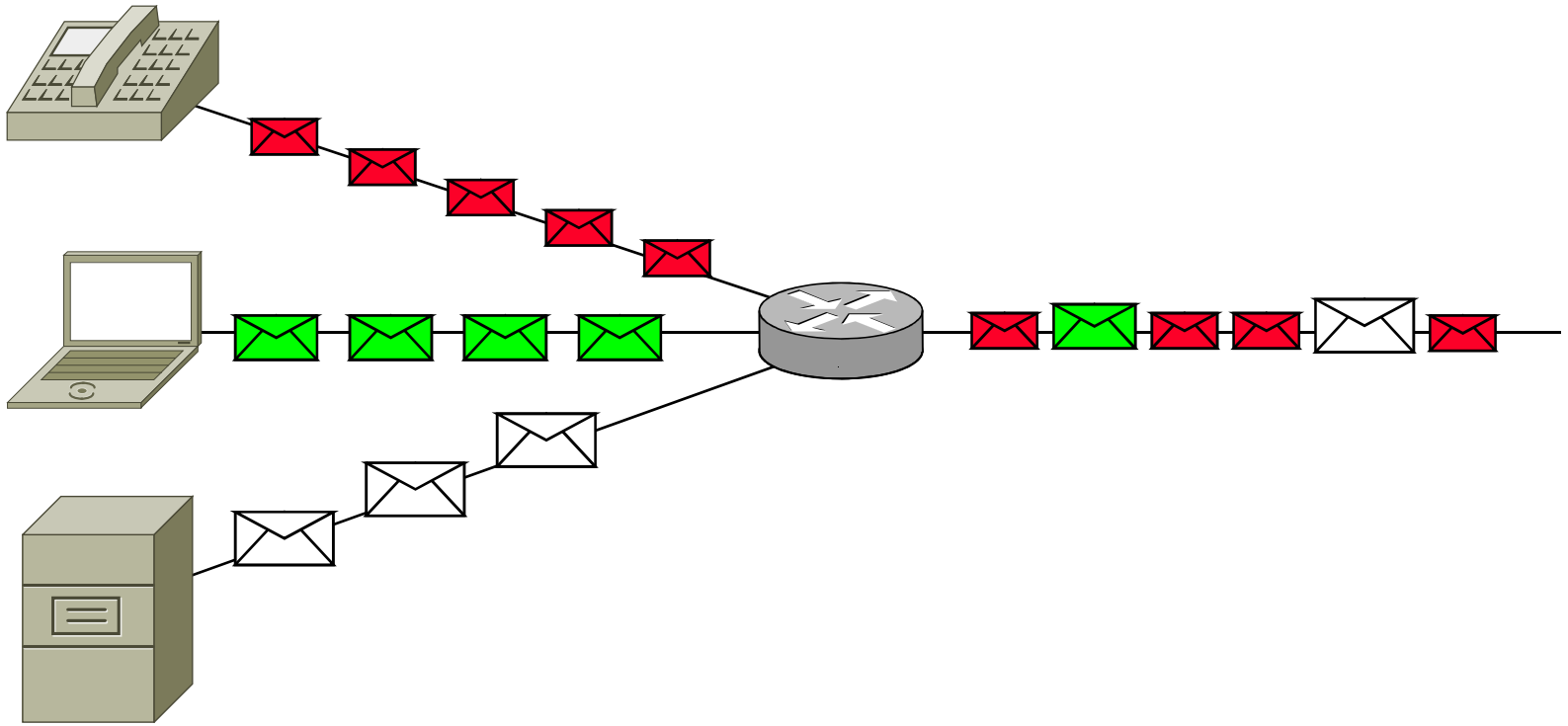
- Goal:
  - guarantee dedicated throughput for particular traffic types
  - guarantee faster delivery of priority messages

# Quality of Service

- External factors:
  - quality and saturation of communication channel
  - form changes (voice $\Rightarrow$ text $\Rightarrow$ image)
  - forwarding (address changes)
  - amount of time allotted for successful communication
- Internal factors:
  - size, complexity, importance of message
- Implementation:
  - data is classified by QoS tag
  - *guaranteed service* strategy: dedicated part of channel
    - quality guaranteed, wasting of channel capacity
  - *best effort* strategy: priority queues
    - effective media usage, quality not guaranteed

# Segmentation, multiplexing

- Channel is not dedicated for a single transmission
- Nodes segments data into smaller units
- All transmissions share the same media



# Current network communications

Examples:

- traffic lines
- weather forecast
- news
- recipes
- internetbanking
- e-mail
- publishing, blog
- chat, IP telephony
- co-operation
- education

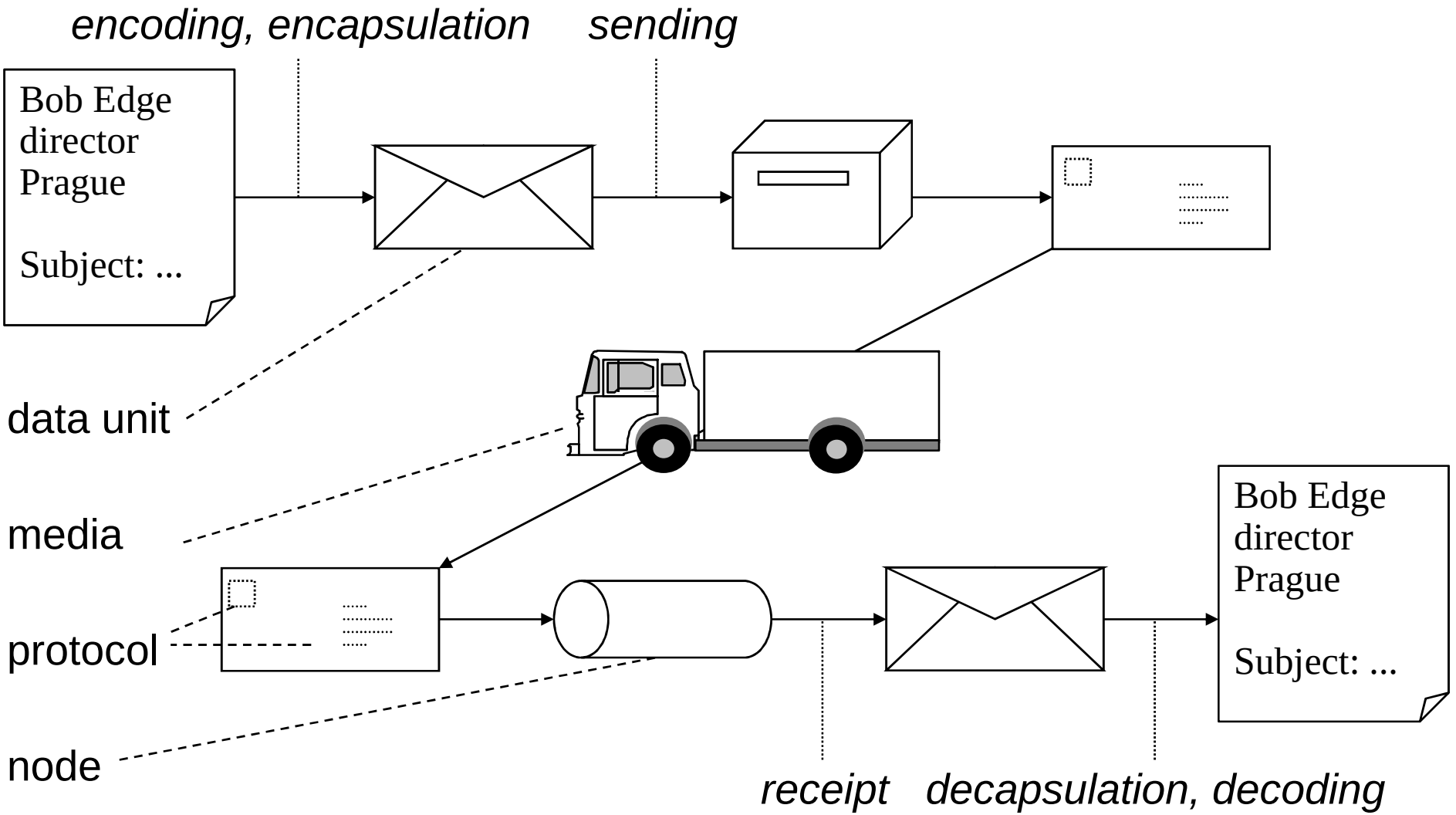
Advantages:

- timeliness
- accessibility
- price
- network integration

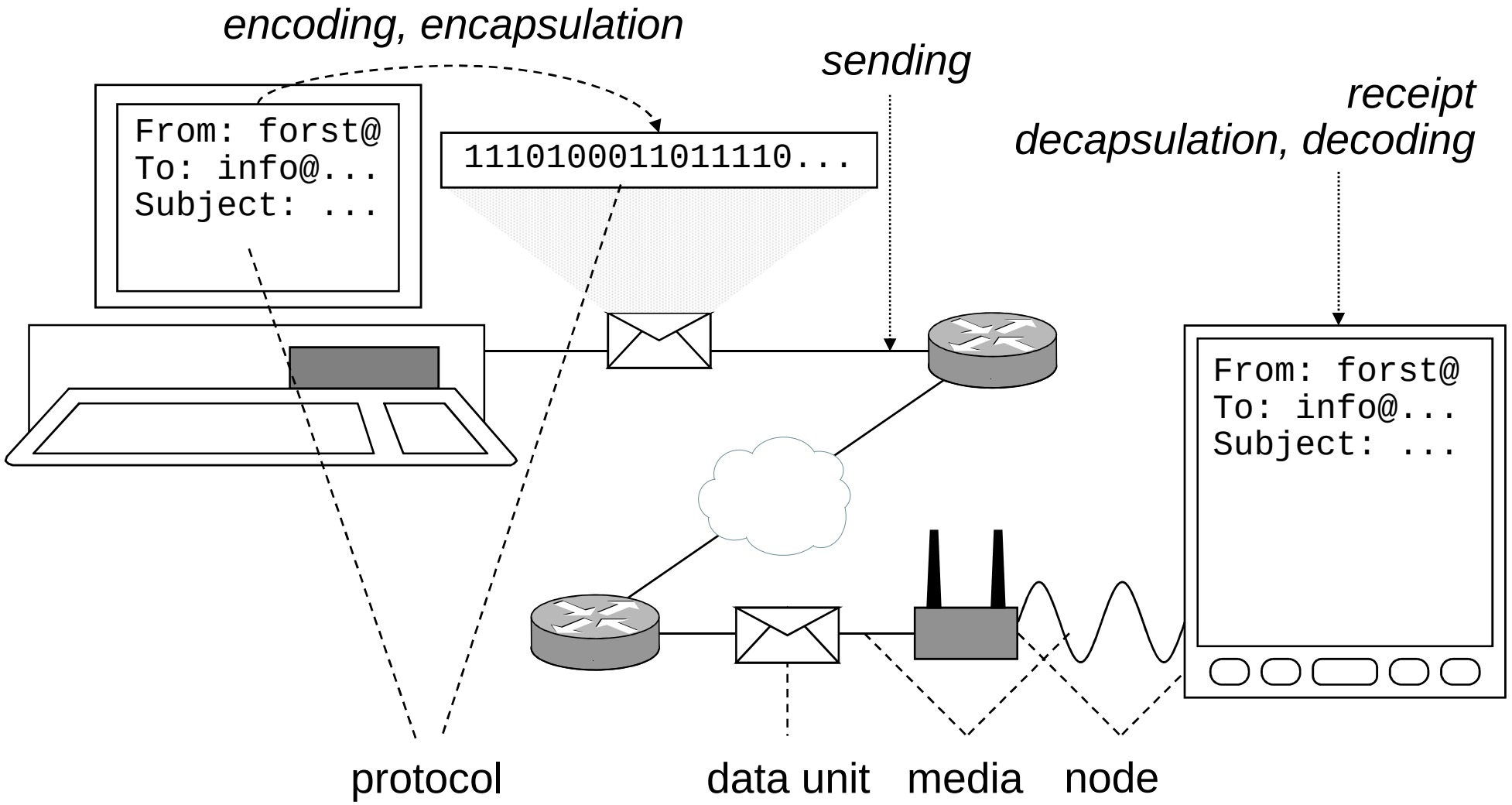
# Components of networking

- Protocols (rules)
  - norms
  - standards
  - recommendations
- Data units
  - message
  - packet
  - bit
- Media
  - wire
  - optical fiber
  - “air” (waves)
- Nodes
  - end nodes (hosts)
  - intermediary nodes (network devices)

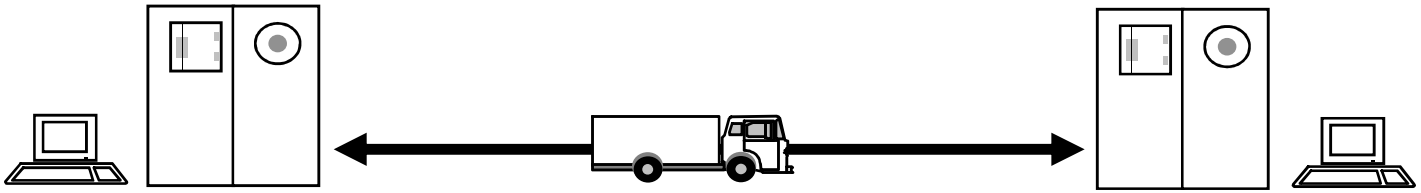
# Message transmission (mail)



# Message transmission (e-mail)

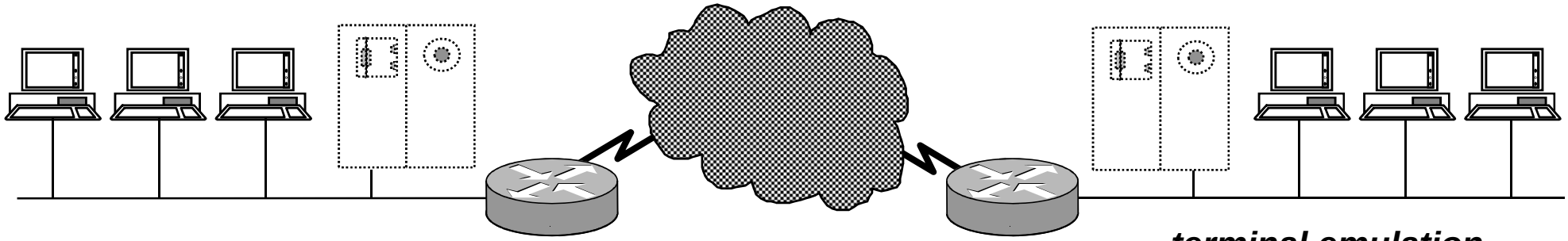


# Computer network history



**isolated computer systems**

**terminals** **point-to-point connections** ***compact applications***



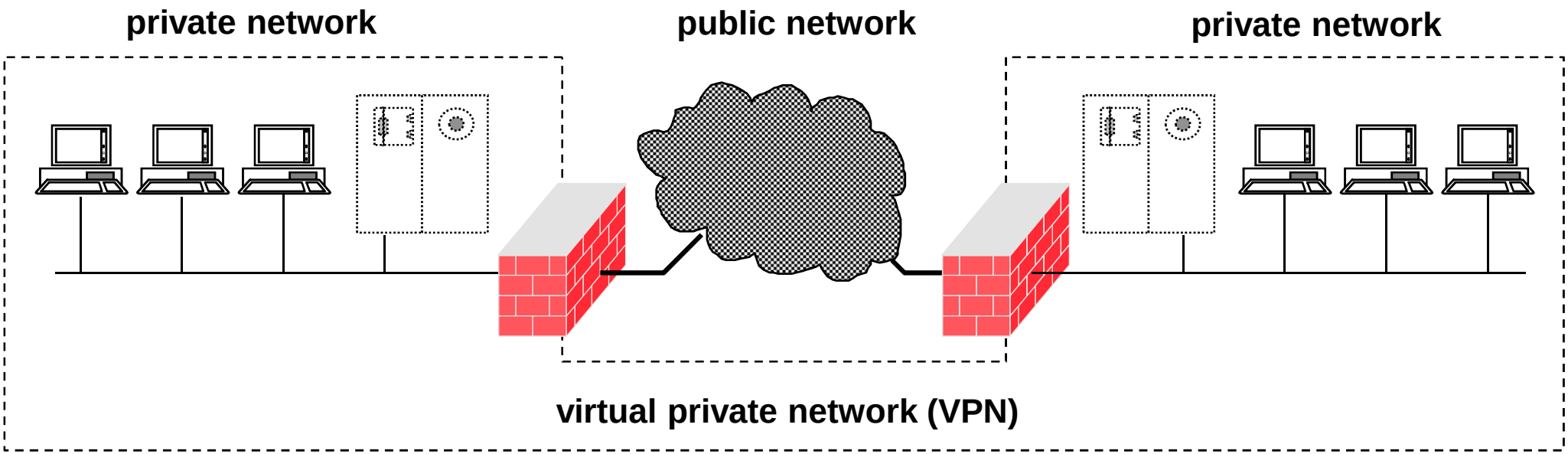
**local area networks (LAN)** **wide area networks (WAN)** ***terminal emulation + client-server applications***

# Basic network types

- Local Area Network (LAN)
  - resources sharing (file- and database-servers, printers,...)
  - shorter distances, minor delay
  - proprietary networks, centralized control
- Wide Area Network (WAN)
  - remote access, end-to-end communication
  - large distances, notable delay
  - multiple owners, distributed control
- Nowadays:
  - differences fade away (most important is possession)
  - interlayers occur (MAN)
- Classification is not technical (no definition), but logical

# Public and private networks

- Most LAN are private (user is the owner)
- Most non-LAN are public (user is not owner)

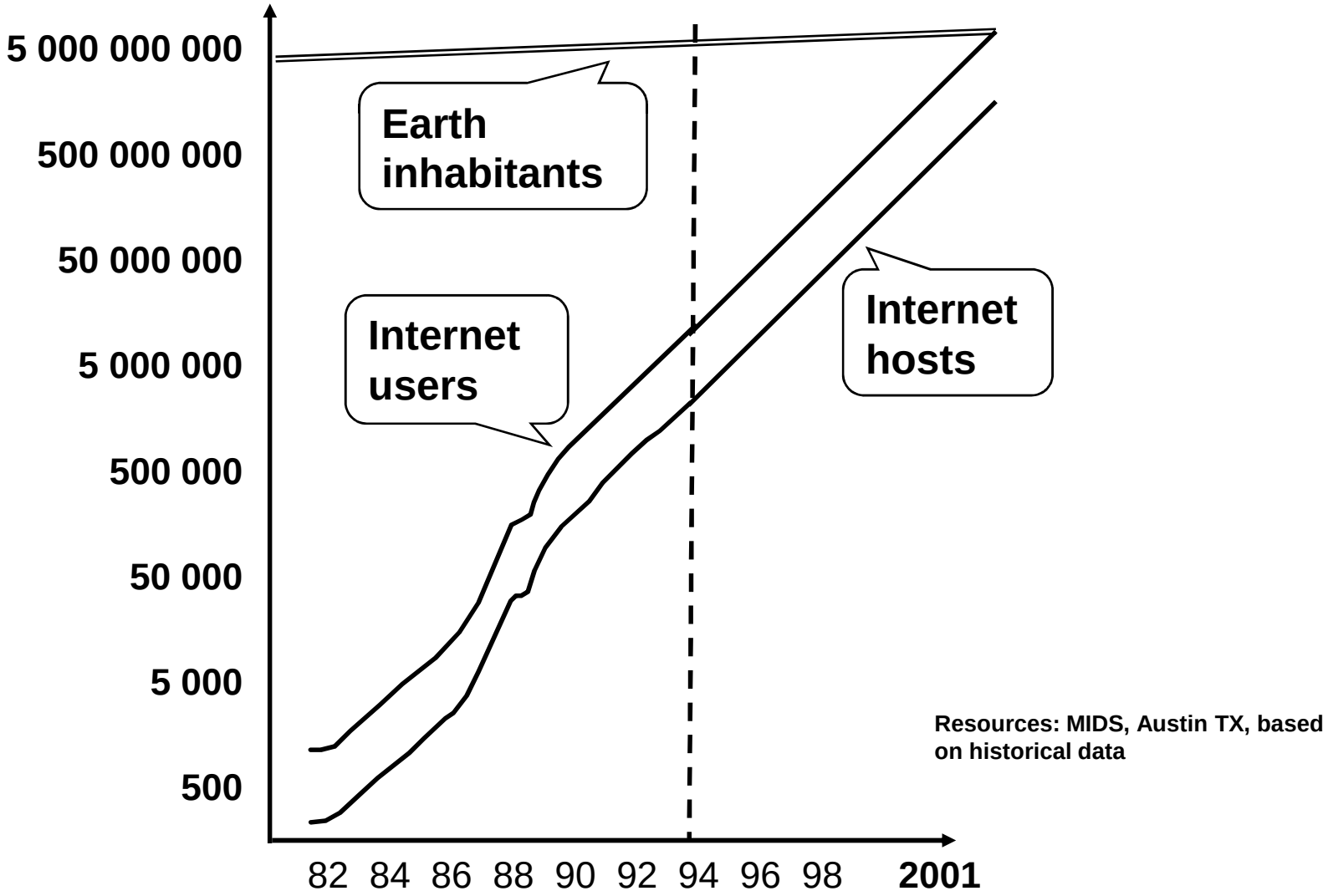



- VPN motivation: security, expenses
- Typical use of VPN: affiliates interconnections, connection of (mobile) end users

# Internet history

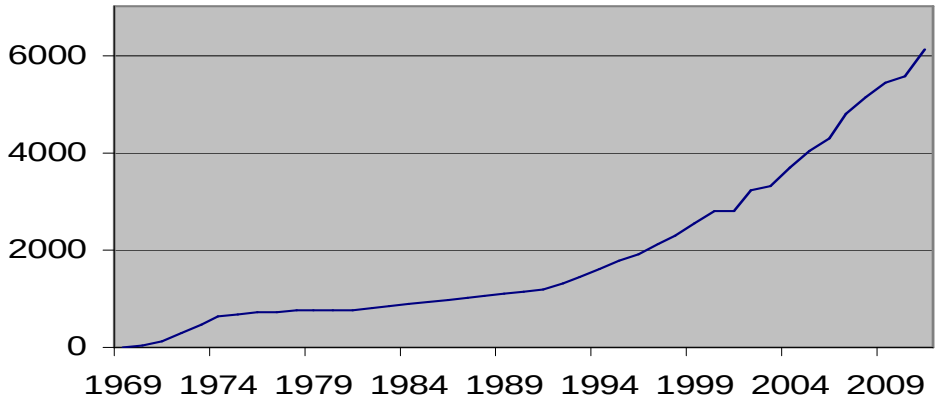
- beg. of the 60s - „packet switching“ concept
- the 60s - US DoD supports „packet switching“ concept for its resistance against a physical attack
- 1969 - ARPANET - paid by Defense Advanced Research Project Agency, managed by academic institutions, point-to-point leased lines
- 1974 - term „Internet“ (abbr. of „internetworking“) used in RFC 675 defining TCP
- 1977 - first network bound to ARPANET backbone
- 1983 - TCP/IP replacing NCP in ARPANET
- half of the 80s - TCP/IP included into BSD UNIX

# Internet progress



# Request for Comments (RFC)

- Mean of Internet „standardization”
- RFC 1 published Apr 7,1969




- Freely accessible (`http://www.ietf.org/rfc.html`)
- Various nature: standards, information, best-practice
- Drafts are sent to and judged by IAB $\Rightarrow$ IETF,IRTF $\Rightarrow$ WG
- Document texts are fixed, upgrades obtain new number (SMTP: *772*, *780*, *788*, 821, 2821, 5321)
- Current status can be found in the index file
- Recommendations are widely violated

# Layered architecture

- Example: sending of minutes from a meeting
  - layer Recorder
    - writes minutes
    - rules: minutes outline
    - request to Secretary: send a letter [minutes;person]
  - layer Secretary
    - searches for addresses, adds a header, footer
    - inserts into an envelope
    - rules: commercial letter form
    - request to Registry: send by mail [letter;address]
  - layer Registry
    - stamps the envelope, places it into a packet sent to the post office
    - rules: post office rules
- Benefits:
  - simpler decomposition and description of the entire process
  - easy technology change (mail/e-mail, post/messenger)

# Network model, network architecture

- Network (reference) model:
  - number of layers and their structure
  - distribution of tasks among layers
  - e.g.: ISO/OSI

- Network (protocol) architecture (suite):
  - network model
  - communication technologies
  - services and protocols
  - e.g.: TCP/IP

# OSI model

- Built from the top, megalomaniac, inconvenient
- Suitable as theoretical model

| Num. | Layer | Task |
|---|---|---|
| 7 | application | end application communication |
| 6 | presentation | data conversions for applications |
| 5 | session | end nodes dialog control |
| 4 | transport | end-to-end data block transmission |
| 3 | network | searching path to target network |
| 2 | data link | data transmission over the media |
| 1 | physical | physical data transmission |

# X.400, X.500

- OSI services implementation was based on a set of complex standards designed “from the top”
  - X.400: Message Handling System (e-mail), some time played the core role in Microsoft Exchange Server
  - X.500: Directory Access Protocol (directory services like phone book), *note: default attribute of a person record is a favorite drink*
- Principle survives in key owner (persons, organizations) identification used in X.509 (apart from X.500):

```
G=Libor; S=Forst;
O=Charles University;
OU=Faculty of Mathematics and Physics;
OU=SISAL;
C=cz
```

# TCP/IP stack

- Grown according to needs, developed gradually

<table>
<tr><th>OSI</th><th>Layer</th><th colspan="5">Protocol examples</th></tr>
<tr><td>7</td><td rowspan="3">application</td><td rowspan="3">FTP,<br>HTTP</td><td rowspan="3">DNS</td><td>NFS</td><td></td></tr>
<tr><td>6</td><td>XDR</td><td></td></tr>
<tr><td>5</td><td>RPC</td><td></td></tr>
<tr><td>4</td><td>transport</td><td colspan="2">TCP</td><td>UDP</td><td>ICMP</td></tr>
<tr><td>3</td><td>network</td><td colspan="3">IP</td><td>ARP</td></tr>
<tr><td>2</td><td rowspan="2">network interface</td><td colspan="4" rowspan="2">Ethernet, FDDI, ATM, WiFi,<br>SLIP, PPP, ...</td></tr>
<tr><td>1</td></tr>
</table>

# Connection-oriented/connectionless services

- Connection-oriented services
  - real-life example: phone call
  - proper order of packet delivery guaranteed (*stream*)
  - simpler application logic, lack of communication control
  - non-fluent but lossless delivery
  - big overhead, TCP itself is complicated
- Connectionless services
  - real-life example: mail
  - neither order nor delivery of packets guaranteed
  - control logic must be included in the application
  - fluent data flow, but data loss may occur
  - low overhead, IP and UDP are simpler

# Application models

- Client-server model
  - client knows the fixed server address
  - client connects to the server, or sends requests
  - server usually handles more clients
  - data flow server ⇒ client: download
  - data flow client ⇒ server: upload
  - examples: DNS, WWW, SMTP
- Peer-to-peer (P2P) model
  - partners do not know data resource addresses
  - no clear roles
  - each partner plays the role of both the client and server (=offer data!)
  - Napster, Gnutella, BitTorrent

# Services addressing

- Uniform Resource Identifier (URI, RFC 3986)
  - unified resource reference system
  - one client can approach various sources (FTP in WWW)
  - former classification: URL (location), URN (name)

***URI*** = ***schema*****://*****authority*** [***path***] [**?*****query***] [**#*****fragment***]

***authority*** = [*login*[**:***password*]@]***address***[**:***port*]

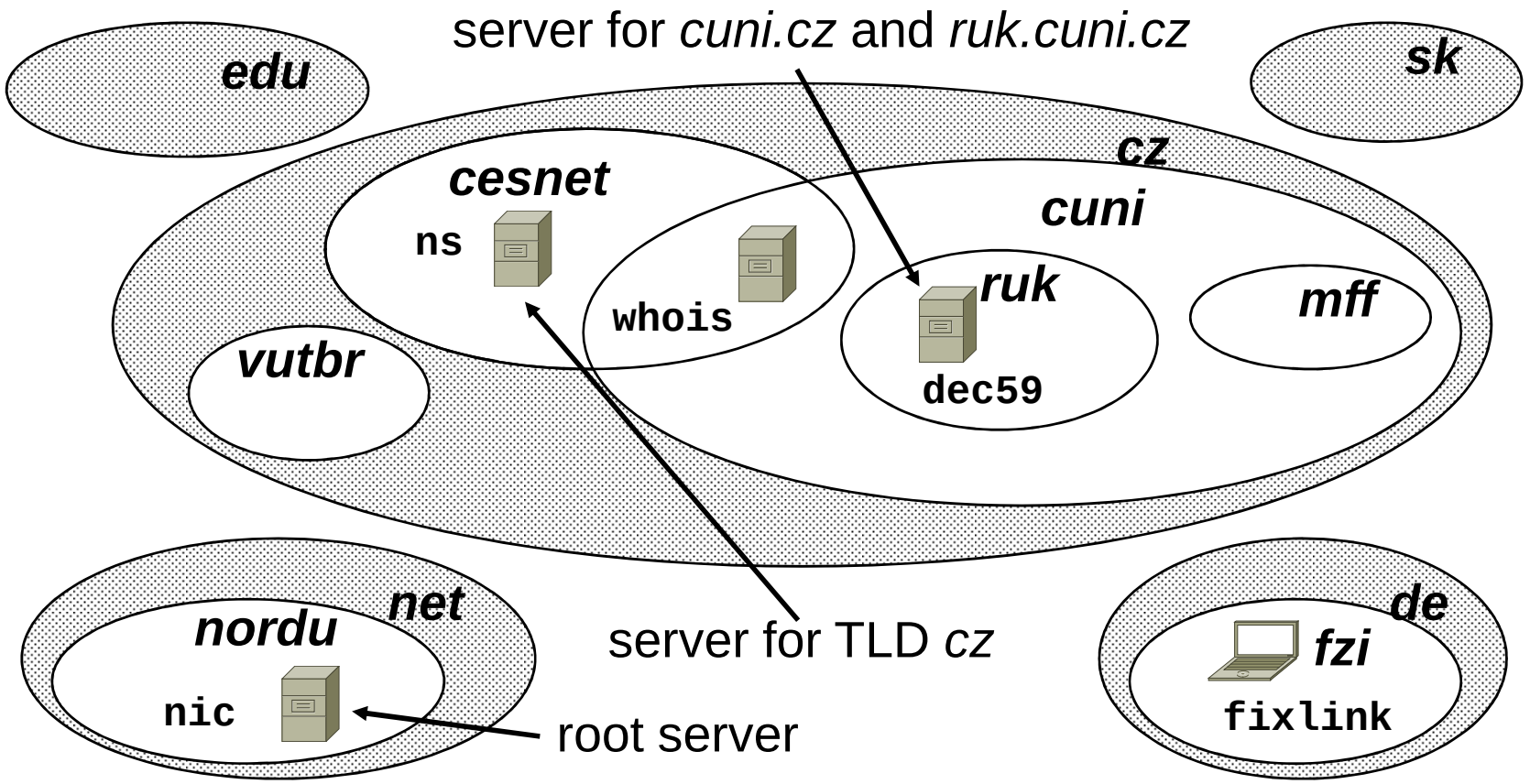
e.g.: **`ftp://sunsite.mff.cuni.cz/Net/RFC`**
**`http://www.cuni.cz:8080/q?ID=123#Local`**
**`mailto:forst@cuni.cz`**

# Hosts addressing

- **HW**
  (data link layer)
- **SW**
  (network layer)
- **People**
  (application layer)

- **physical, MAC address**
  (e.g. Ethernet: `8:0:20:ae:6:1f`)
  - factory given (formerly), programmed (now)
  - does not respect topology
- **IP address**
  (e.g.: `194.50.16.71`, `::1`)
  - assigned according to network topology
  - given border between net and host part
- **domain address**
  (např.: `whois.cuni.cz`)
  - assigned according to the organization structure
  - easy to remember

# Domain names system

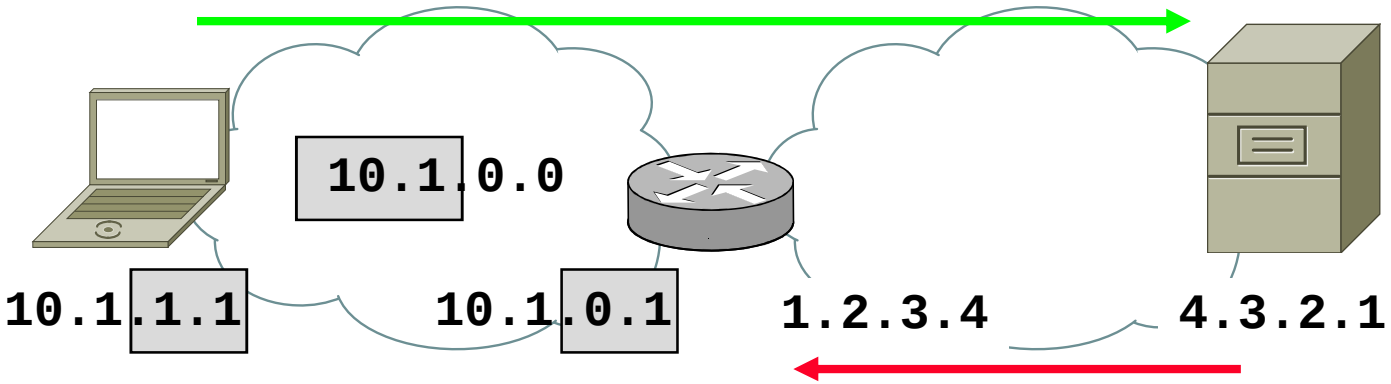


# Domain administration

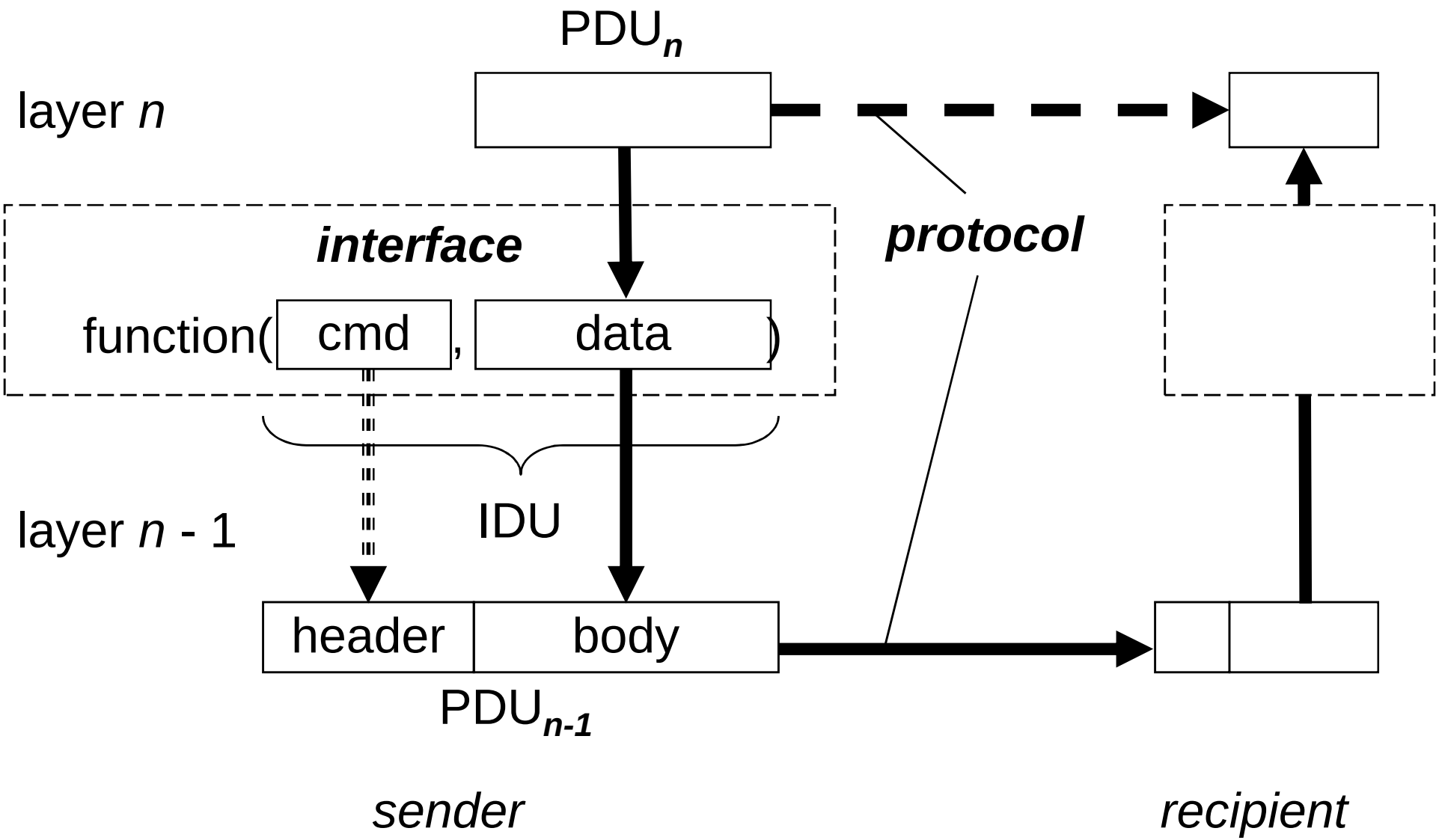
- Top level domains (administered by ICANN):
  - administrative (**`arpa`**)
  - originally pure U.S. resort domains (**`com`**, **`net`**, **`org`**, **`edu`**, **`mil`**, **`gov`**); later released and new ones added (**`info`**, **`biz`**, **`aero`**, ...); nowadays private subjects can request
  - ISO country codes (**`cz`**, **`sk`**, ...) and some exceptions (**`uk`**, **`eu`**); some small countries sell interesting names (**`nu`**, **`to`**)
  - internationalized codes (`.中国 = .xn--fiqs8s`, `.рф`)
- TLD **`.cz`**:
  - administered by CZ.NIC (ISP corporation)
  - no structure, circa 3/4 mil. names under `.cz`
  - no support for localized names (IDN)
- Lower level domain:
  - administered by owner (`ms.[mff.[cuni.cz]]`)

# IP addresses

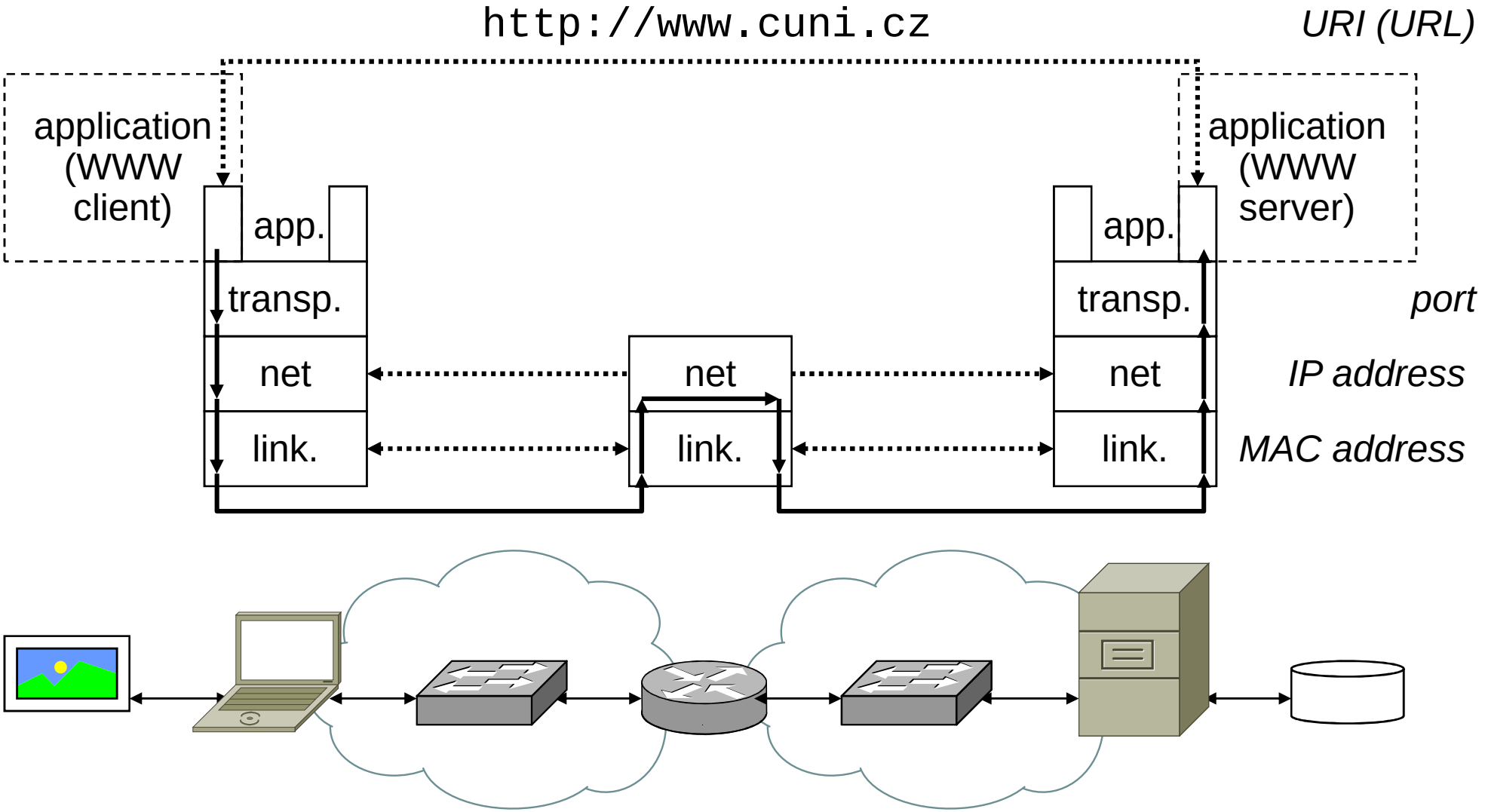
- Every end node in TCP/IP network must have an IP address
- Current versions:
  - IPv4: 4 bytes (e.g. **`195.113.19.71`**)
  - IPv6: 16 bytes (e.g. **`2001:718:1e03:a01::1`**)
- Address block assignment:
  - public addresses assigned by an ISP
  - LAN can use private addresses, unreachable from the internet (security vs. interoperability), *address translation* (NAT)



# Protocol vs. interface



# Addressing in TCP/IP



# Port, socket

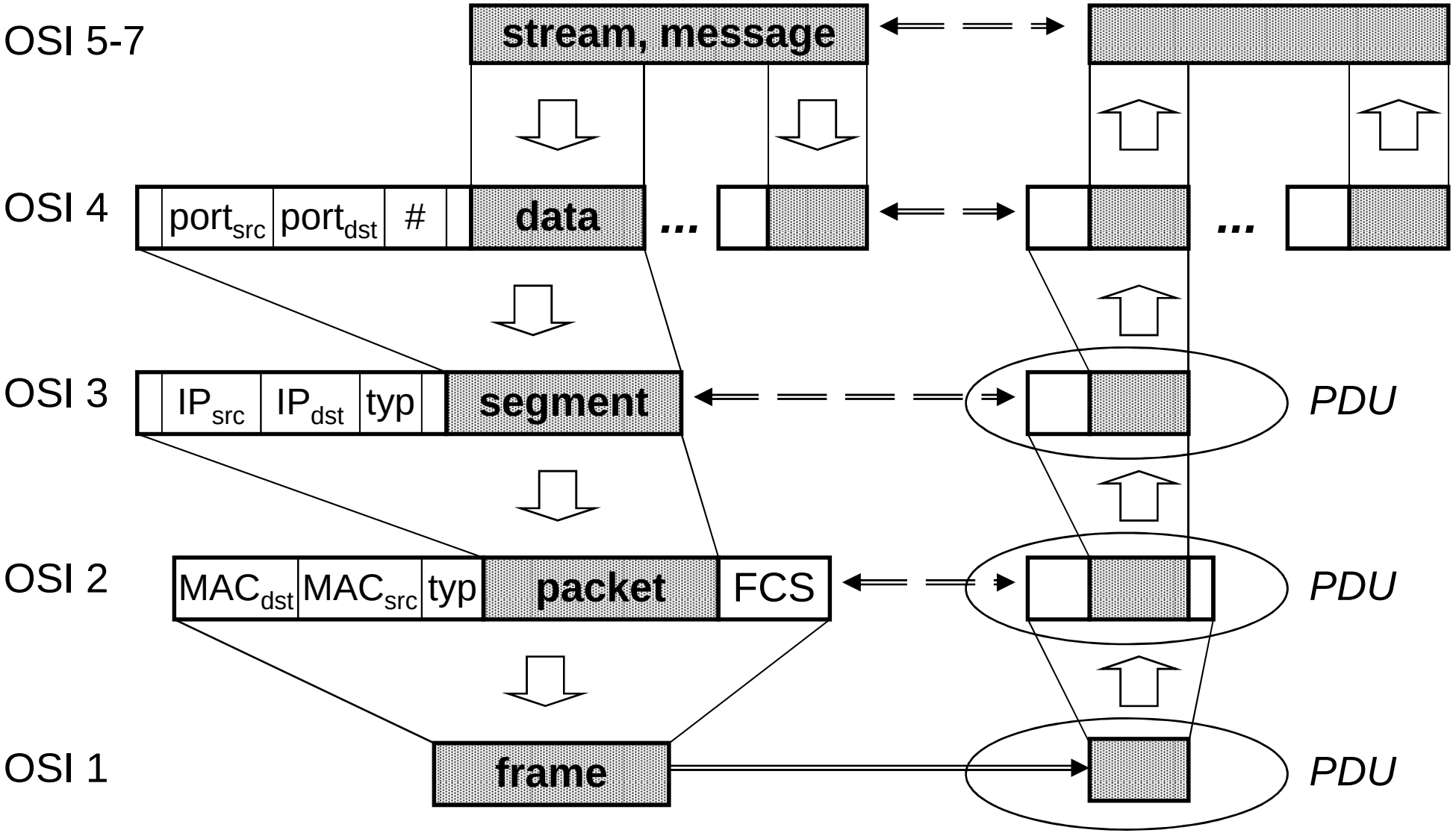
- **Port**
  - ... 16bit integer identifying one end of the communication channel - an application, or a process responsible for incoming data processing
    - destination-port must be known by clients, usually it is one of so-called *well-known services*
    - source-port (>1024) is assigned by the originator OS

- **Socket**
  - ... one end of the communication channel between the client and the server
  - … identification (“address”) of the channel end <*IP address*, *port*>

# Well-known services examples

- **21/TCP: FTP - File Transfer Protocol**
  (file transfer)
- **22/TCP: SSH - Secure Shell**
  (remote logging and file transfer)
- **23/TCP: telnet - Telecommunication network**
  (remote logging)
- **25/TCP: SMTP - Simple Mail Transfer Protocol**
  (electronic mail transfer)
- **53/*: DNS – Domain Name System**
  (conversion of IP addresses to names and vice versa)
- **67,68/UDP: DHCP - Dynamic Host Configuration Protocol**
  (remote configuration)
- **80,443/TCP: HTTP - HyperText Transfer Protocol**
  (transfer of WWW information system pages)
- **5060,5061/*: SIP - Session Initiation Protocol**
  (VoIP, IP telephony)

# Multiplexing, encapsulation



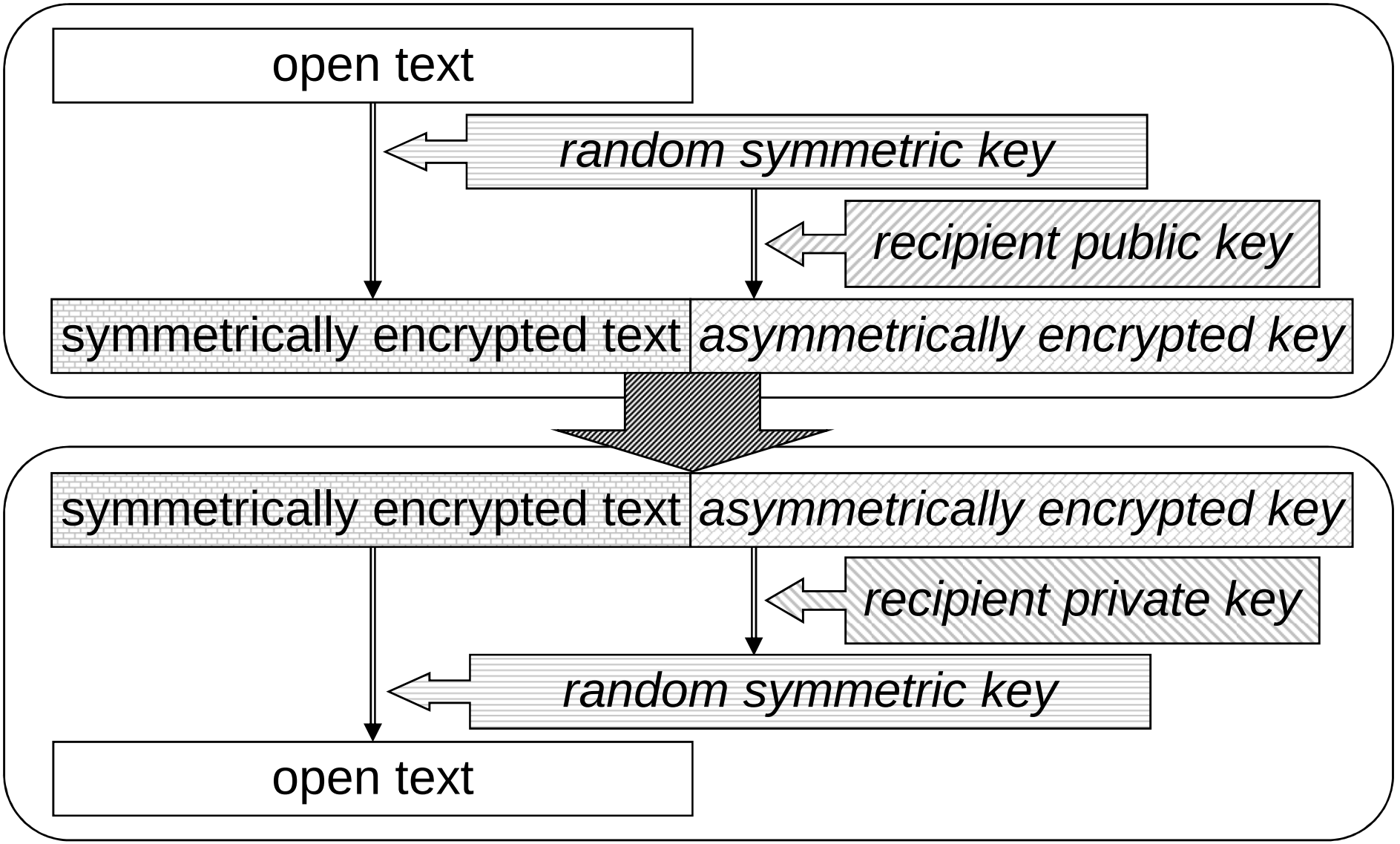
# Data flow in TCP/IP

- Multiplexing:
  - sharing of a communication channel by multiple services
- Demultiplexing:
  - the receiver must distribute data properly according to control information contained in the PDU
- Encapsulation:
  - data and control information of one layer stored into the PDU of another layer (usually *n*+1 => *n*, other modes possible)
- Segmentation:
  - dividing of application data on the transport layer
- Fragmentation:
  - dividing of segments on the network layer due to small MTU (maximum transmission unit) of the data link layer

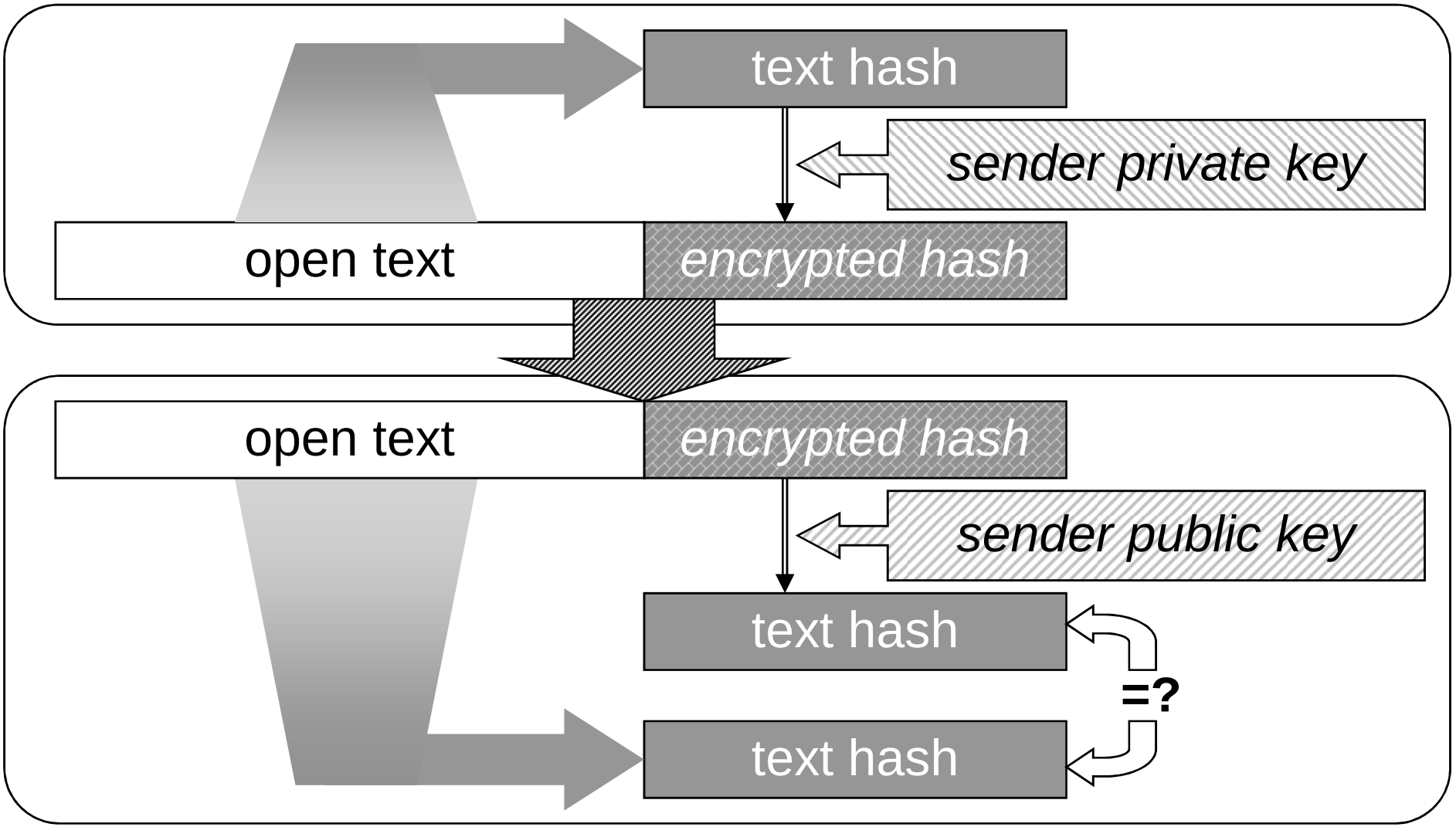
# Basic cryptographic methods

- Symmetric cryptography
  - the same key for encryption and decryption
  - pros: fast, suitable for large data
  - cons: partners must safely exchange a common key
- Asymmetric cryptography
  - two keys (private and public) for encryption and decryption
  - pros: the public key is free, just the private one must be secured
  - cons: slow, suitable only for small data
  - the public key must be carefully verified
- Hash
  - calculation of a small piece of code from the given text
  - minor text change = major change of hash, „almost unique“
  - one-way function, finding the text from a hash is „hard“

# Data encryption



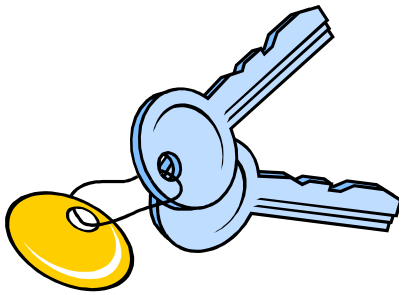
# Digital signature

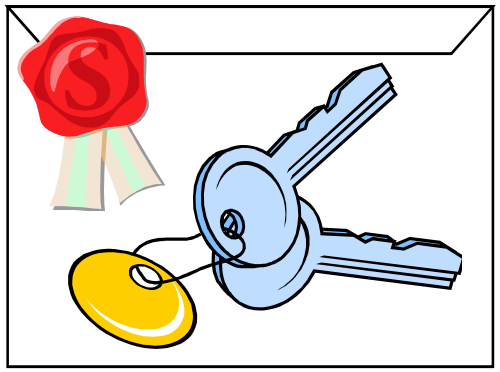


# Diffie-Hellman algorithm

- Method of information exchange between two partners via an open channel to get a common secret (e.g. a symmetric key)
- Used in many protocols exploring the symmetric cryptography
- Procedure description:
  1. Alice picks a secret number $a$ and public primes $p$ and $g$.
  2. She computes $A = g^a \bmod p$ and sends $p$, $g$, and $A$ to Bob.
  3. Bob picks a secret number $b$, computes $B = g^b \bmod p$ and sends $B$ to Alice.
  4. Alice computes $s = B^a \bmod p$ and Bob the same $s = A^b \bmod p$.
- Principle:
  - $A^b = ( g^a )^b = g^{ab} = g^{ba} = ( g^b )^a = B^a$
  - Without knowing of secret numbers $a$ and $b$ and when choosing large prime numbers $p$ and $g$, computing the $s$ from $A$ and $B$ is considered to be “hard”.

# Public key authenticity





- Does the identity tag belong to the key?
  - people usually can make sure that they communicate with the proper partner prior to disclosing some secret
  - the key may be and should be verified from independent sources
  - applications must use some automated way
- Authenticity is verified by a third party and its signature is appended; this can be
  - someone who is known for me and I have his or her key verified
  - well-known public certification authority; listed e.g. in browsers, however such lists credibility is not absolutely sure

# Certificate

- Certificate is key plus an identification tag, signed by a certification authority (CA)
- Trusting a CA, we can trust the key owner (verifying the CA credibility needed!)
- Certificate structure by X.509 (RFC 3280; SSL, not SSH):
  - certificate
    - certificate version
    - certificate serial number
    - issuer
    - validity dates
    - public key owner
    - public key information (algorithm and key)
  - certificate signature algorithm
  - certificate signature

# Application layer in TCP/IP

- Covers functions of OSI layers 5, 6 and 7
  - communication rules between client and server
  - dialog status
  - data interpretation

- Application layer protocol defines
  - operation control flow on both sides
  - message format (textual/binary data, structure,...)
  - message types (requests and responses)
  - message and information fields semantics
  - dialog control
  - transport layer interaction

# Domain Name System

- Client-sever application for names to addresses resolving
- Binary protocol over UDP & TCP, port 53, RFC 1034, 1035
  - Common requests (up to 512B) use UDP
  - Larger data transfers use TCP
- Client contacts servers defined in its configuration, getting information about other servers until he finds the answer
- Serious security issues, DNS with signed data (DNSSEC) is complicated and slowly spreading
- Data unit is called resource record (RR), e.g.:

```
ns.cuni.cz. 3600 IN A 195.113.19.78
```

  - RR name
  - validity time (TTL)
  - type and data

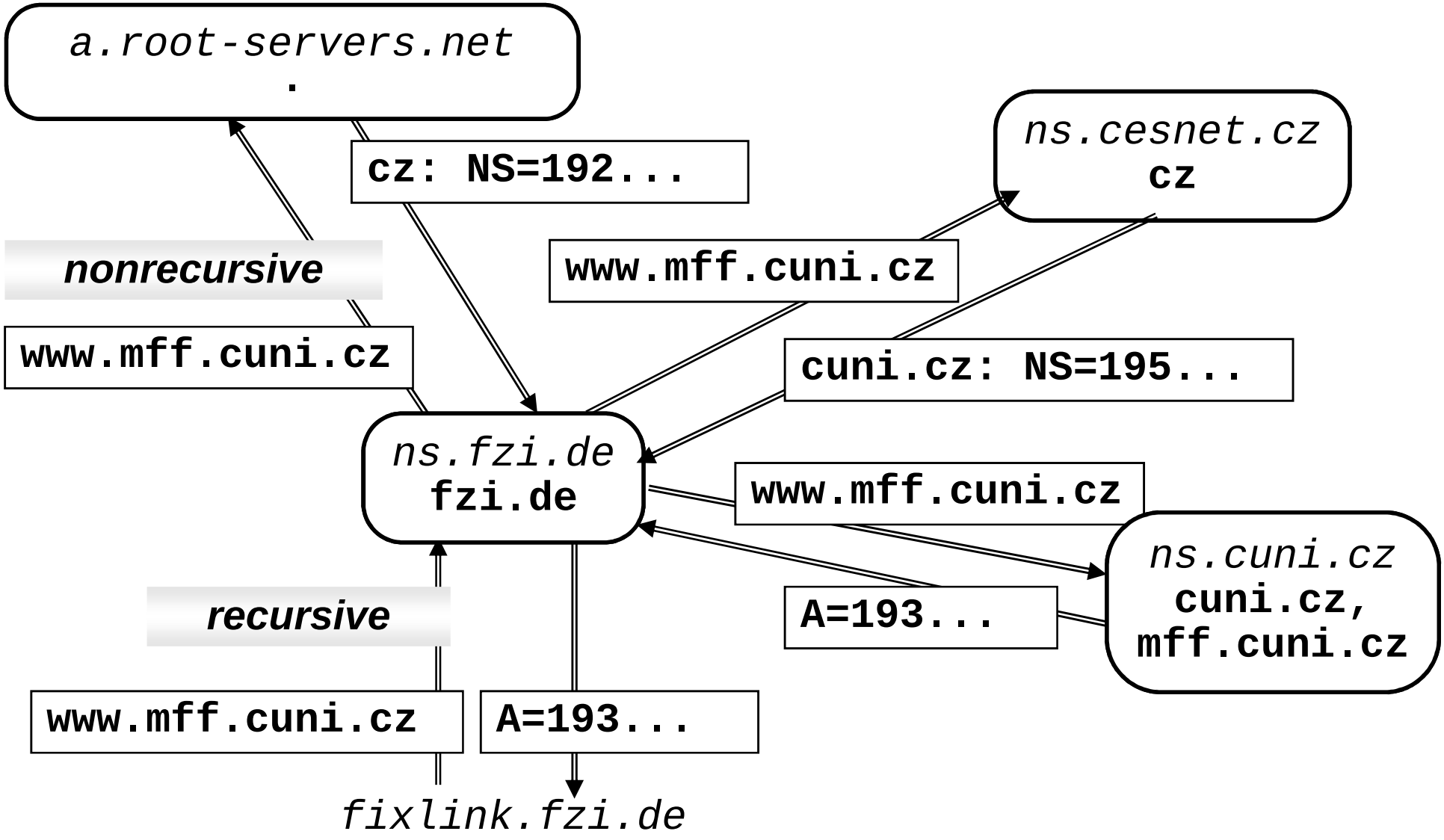
# DNS resource records

| Type | RR name semantics | Data semantics |
|---|---|---|
| **SOA** | domain name | general domain attributes |
| **NS** | domain name | domain nameserver name |
| **A** | host name | host IPv4 address |
| **AAAA** | host name | host IPv6 address |
| **PTR** | reverse name (e.g. IP address 1.2.3.4 is represented by `4.3.2.1.in-addr.arpa`, ::1 by `1.0…0.ip6.arpa`) | host domain name |
| **CNAME** | alias name | canonical (real) host name |
| **MX** | domain/host name | mailserver name and priority |

# DNS servers

- Server types:
  - primary: manages the domain RR database
  - secondary: downloads and keeps a copy of the RR database
  - caching-only: keeps just (un)resolved requests within their validity time
- Every domain (zone) must have at least one (better more) *authoritative* (primary or secondary) nameservers
- Data exchanges run over TCP with regular query/answer form (data is sent as DNS RRs)
- Zone database actualization is started by the secondary server, but the primary one can signal the need of updates

# DNS request processing



# DNS query and answer

- Request:

```
QUERY:      www.cuni.cz.   IN A
```

- Response:

```
FLAGS:      Authoritative, Recursive
QUERY:      www.cuni.cz.   IN A
ANSWER:     www.cuni.cz.   IN CNAME tarantula
            tarantula      IN A     195.113.89.35
AUTHORITY:  cuni.cz.       IN NS    golias
ADDITIONAL: golias         IN A     195.113.0.2
```

- Problem: `Authoritative` flag is not applied to the `AUTHORITY` and `ADDITIONAL` sections, a legal server for some domain can apparently legally add there false data about other domains.

# DNS configuration

## UNIX

local domain and nameserver: **`/etc/resolv.conf`**

```
domain domainname
nameserver nameserver_IP_address
```

## Windows

**Control Panel ⇨ Network and Internet**
**⇨ Network Connections**
**⇨ Local Area Connection ⇨ Properties**
**⇨ TCP/IPv4 ⇨ Properties**
**⇨ General ⇨ Advanced ⇨ DNS**

# DNS diagnostics

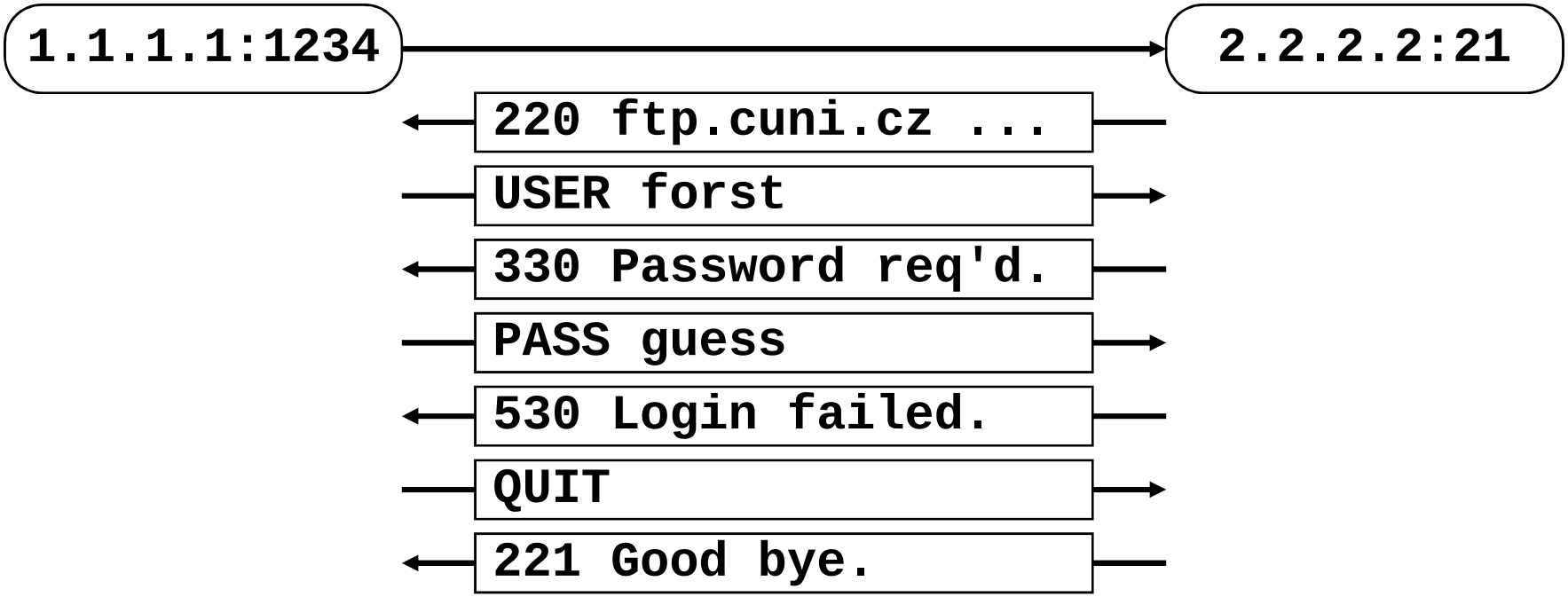
- Program **`nslookup`**
  - commands: **`set type`**, **`server`**, *name*, *IPadr*, **`ls`**, **`exit`**

```
> set type=ns
> cuni.cz
Server:         195.113.19.71
Address:        195.113.19.71#53

Non-authoritative answer:
cuni.cz nameserver = golias.ruk.cuni.cz.
cuni.cz nameserver = ns.ces.net.

Authoritative answers can be found from:
```

- Program **`dig`**
  - **`dig`** [*@server*] *name* [*RR_type*]

# File Transfer Protocol

- One of the oldest protocols (RFC 959, valid till now!)
- Originally: own account data access, open password sent!
- Today: mostly anonymous access to public domain data (user **`anonymous`** or **`ftp`**, email instead of password)
- Control connection example (commands and responses):

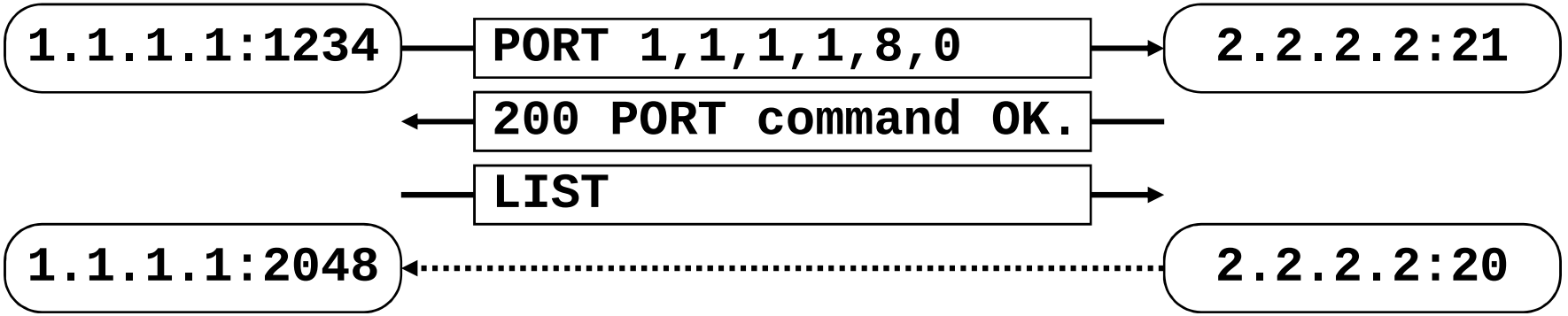


# Response codes

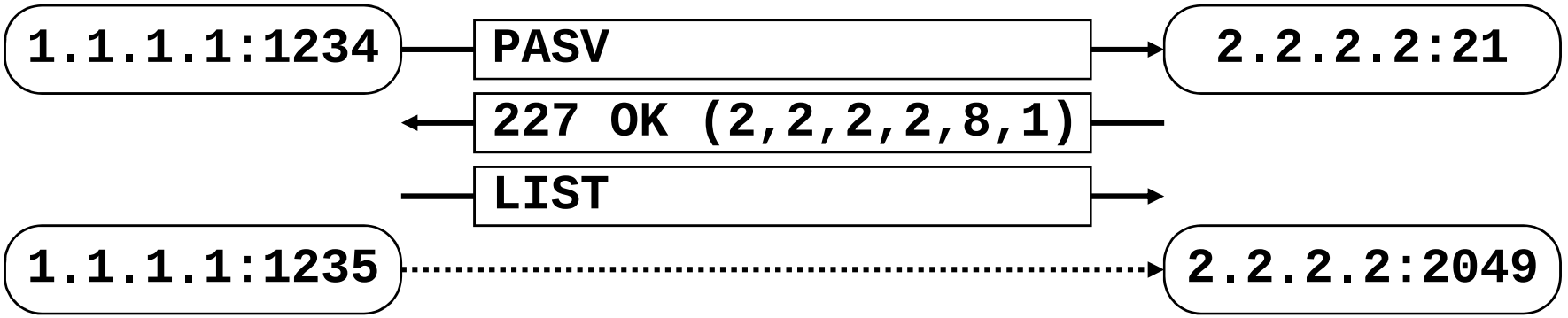
- For simpler automated processing of responses, they start with 3-digit  number
- The first digit expresses response severity:
  - 1xx **positive preliminary reply** (action was started, further responses expected)
  - 2xx **positive completion reply**
  - 3xx **positive intermediate reply** (further commands necessary)
  - 4xx **transient negative completion reply** (action failed, however repeating later makes sense)
  - 5xx **permanent negative completion reply** (action failed and will fail later, too)
- A similar schema adopted by many protocols

# Active/passive data connection

- Every data transfer uses new (data) connection
- Active data connection establishment:



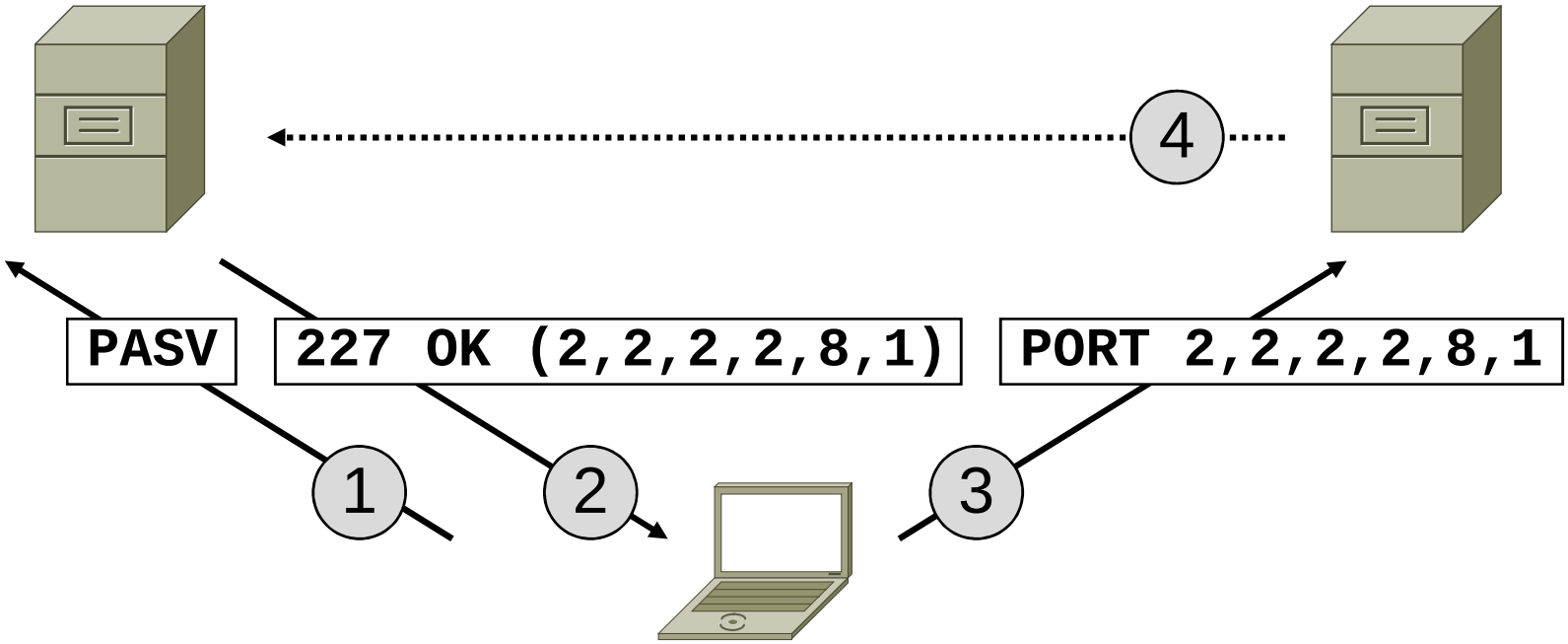

- Passive data connection establishment:




- After data transfer closing, the connection is terminated

# Third Party Transfer

- Direct data transfer between two servers (for performance, capacity or security reasons)




- Security risk: attacker can fake own address and port

# Applications for FTP

- WWW browsers
- file managers (Total Commander)
- command-line **`ftp`** client
  - session opening: **`open`**, **`user`**
  - session closing: **`close`**, **`quit`**, **`bye`**
  - local commands: **`lcd`**, **`!`***command*
    (`!cd` generally does not work!)
  - remote commands: **`cd`**, **`pwd`**, **`ls`**, **`dir`**
  - file transfer: **`get`**, **`put`**, **`mget`**, **`mput`**
  - file transfer mode: **`ascii`**, **`binary`**
    (mind text/binary file transfers between different OS!)
  - file management: **`delete`**, **`rename`**, **`mkdir`**, **`rmdir`**
  - miscellaneous: **`prompt`**, **`hash`**, **`status`**, **`help`**,...

# Electronic mail

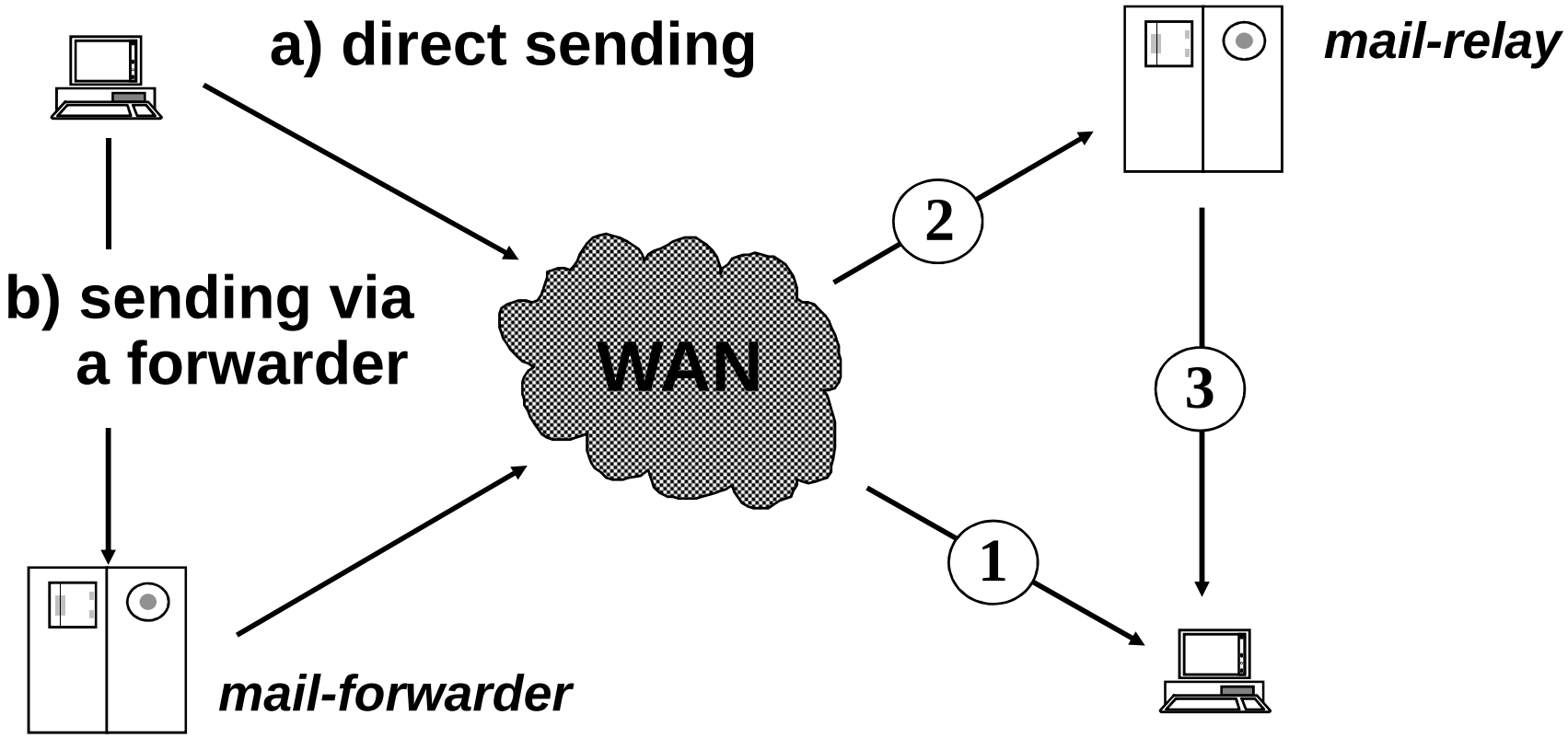
- General service, exists also out of the Internet
  - off-line sending of messages or files
  - off-line usage of information services
  - mailing-lists, conferences
  - communication outside the Internet
- In TCP/IP based on RFC 821, 2821 resp. 5321 (SMTP) and RFC 822, 2822 resp. 5322 (message format) on the port 25
- General form of e-mail address in the Internet:

  *login@host* or *alias@domain*

  e.g.:

  **`forst@ms.ms.mff.cuni.cz`** or **`Libor.Forst@cuni.cz`**

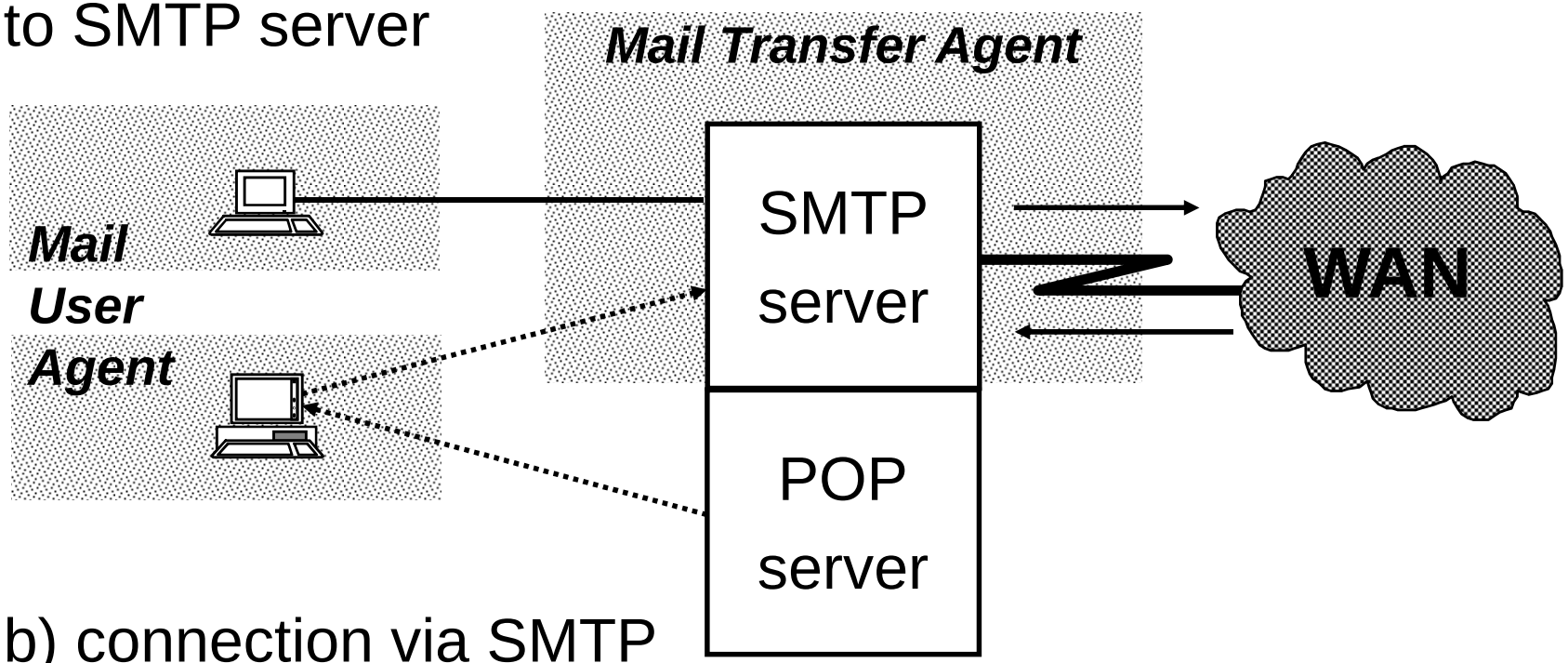
# E-mail delivery in SMTP




```
mbox IN MX  0 mbox
     IN MX 20 relay
```

# User approach to e-mail

a) direct approach
(via terminal or web)
to SMTP server




b) connection via SMTP
and POP or IMAP

# SMTP protocol example

```
⇐ 220 alfik.ms.mff.cuni.cz ESMTP Sendmail ...
⇒ HELO betynka
⇐ 250 alfik Hello betynka, pleased to meet you
⇒ MAIL FROM: <forst@cuni.cz>
⇐ 250 2.1.0 <forst@cuni.cz>... Sender ok
⇒ RCPT TO: <libor@forst.cz>
⇐ 250 2.1.5 <libor@forst.cz>... Recipient ok
⇒ DATA
⇐ 354 Enter mail, end with "." on a line by itself
⇒ From: <forst@cuni.cz>
⇒ To: <libor@forst.cz>
⇒ ...
⇒ .
⇐ 250 2.0.0 h98G9FxT Message accepted for delivery
⇒ QUIT
⇐ 221 2.0.0 alfik closing connection
```

*envelope*

*letter*

# Electronic letter

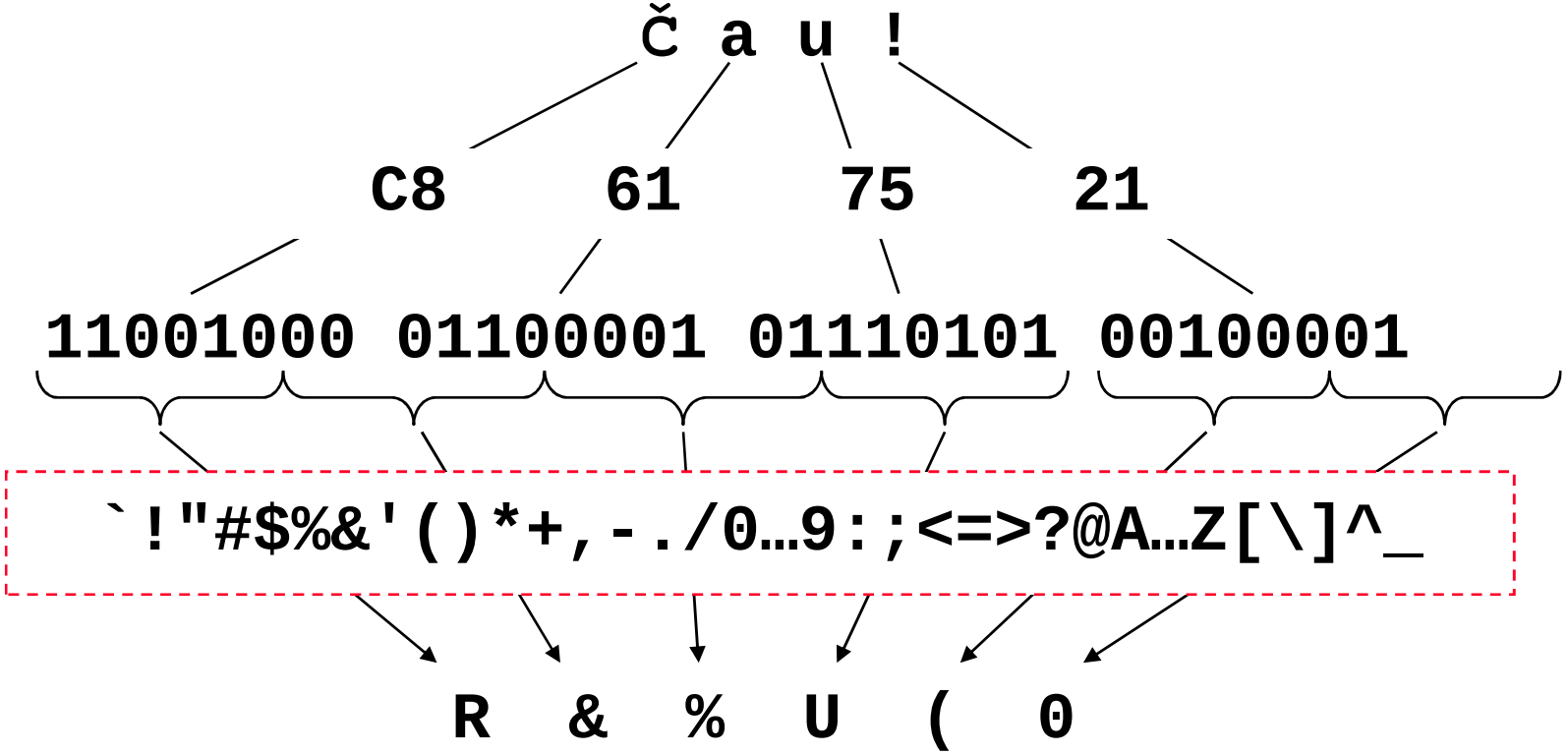
```
Received: from alfik.ms.mff.cuni.cz
     by betynka.ms.mff.cuni.cz...
Date: Thu, 16 Nov 1995 00:54:31 +0100
To: student1@ms.mff.cuni.cz
From: Libor Forst <forst@cuni.cz>
Subject: Mail test
Cc: student2@ms.mff.cuni.cz
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="=_XXX_="

--=_XXX_=
Content-Type: text/plain; charset=Windows-1250
Content-Transfer-Encoding: 8bit

Čau Petře!
...
--=_XXX_=--
```

# Mail headers

| | |
|---|---|
| `Date:` | mail creation date |
| `From:` | mail author(s) |
| `Sender:` | mail sender |
| `Reply-To:` | response address |
| `To:` | mail recipient(s) |
| `Cc:` | (carbon copy) additional mail copy recipient(s) |
| `Bcc:` | (blind cc) hidden recipient(s) |
| `Message-ID:` | mail identification code |
| `Subject:` | mail subject |
| `Received:` | recording of mail transfer |

# Files and diacritics in mail

- Originally pure 7-bit ASCII, files encoding using UUENCODE (coming from UUCP, unix-to-unix-copy)




- Encoding itself is OK, but lack of methodical incorporation

# Multipurpose Internet Mail Extension

- RFC 2045-2049, it enables:
  - Creating structured document
  - For each part:
    - Describing content type (e.g. `text/html`) and format
    - Defining character set and document encoding
    - Joining additional document processing info
  - Using diacritics in (some) headers:

```
Subject: =?utf-8?b?SVRBVCAyMDEyIC0gcG96?=
```

- Encodings:
  - **`Base64`**: based on uuencode, table and line form changed
  - **`Quoted-Printable`**: non-ASCII chars encoded as string „=*HH*“, where *HH* is character hexadecimal value
- Nowadays widely used in other protocols, too

# Netiquette Guidelines

- RFC 1855
  - read all mails before answering
  - consider taking part in the discussion if you are only Cc
  - let recipient some time to answer (checking delivery is OK)
  - answer promptly, at least as an acknowledgement
  - choose Subject carefully, check list of recipients
  - select properly language, charset, means of expression
  - leave relevant parts of the original text when answering
  - respect ©, ask original author when forwarding
  - use effective file transfer
  - check what your mailer sends (duplicate HTML version!)
  - don’t bother people, don’t overload network
  - sign

# Mail security

- A mail is always **an open letter post** (for various reasons it can be delivered to unexpected people)

  *Solution*: letter encryption (e.g. PGP - Pretty Good Privacy)

- The **sender** is never obvious, neither compliance of data from the envelope and the letter

  *Partial solution*: Sender Policy Framework, call-back attempt

  *Solution*: challenge/response system, signatures

- Don’t open files from unknown source!

# Post Office Protocol

- Protocol for remote access to user mailboxes
- Current version 3, RFC 1939, port 110
- Main disadvantages:
  - Sending passwords in plaintext; there is an extension for sending them encrypted (APOP command), but many clients have it unimplemented
  - Letter must be withdrawn in its entirety; there is also an extension (TOP command) for withdrawal of the letter beginning, again rarely implemented
  - No support for attachment structure handling
- Nowadays supported mainly for backward compatibility and gradually replaced by IMAP

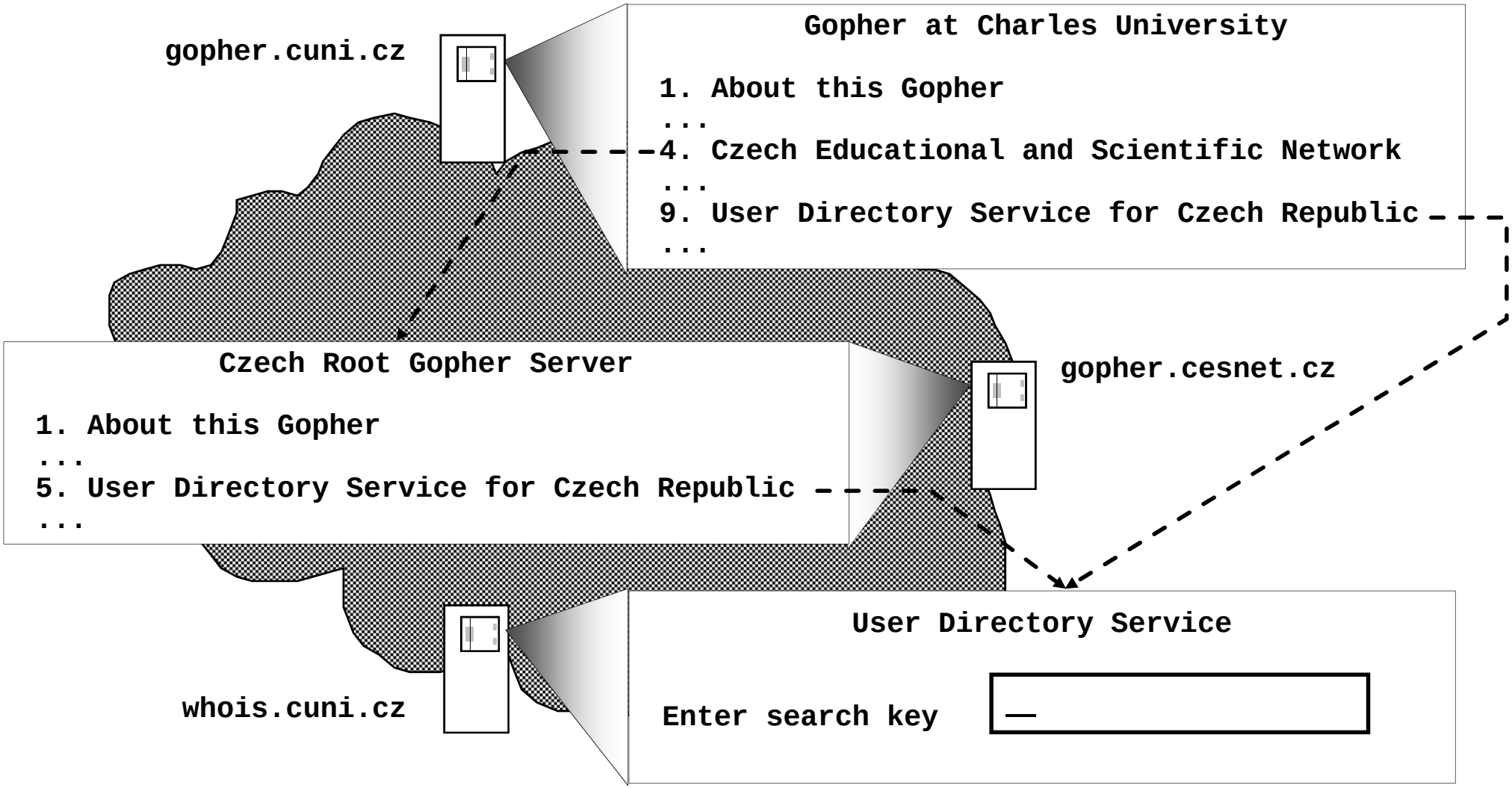
# POP3 example

```
⇐ +OK POP3 server ready ...
⇒ USER forst
⇐ +OK User accepted
⇒ PASS heslo
⇐ +OK Pass accepted
⇒ LIST
⇐ +OK 2 messages (1234 octets)
⇐ 1 1111
⇐ 2 123
⇐ .
⇒ RETR 1
⇐ +OK 1111 octets
⇐ From: …
⇐ .
⇒ DELE 1
⇐ +OK message 1 deleted
```

# Internet Message Access Protocol

- More powerful and more complicated successor to POP
- Current version 4rev1, RFC 3501, port 143
- Main advantages:
  - Embedded support for cryptography
  - Server keeps information about mails (status)
  - More mailboxes (folders) support
  - Commands for withdrawal of part of a mail
  - Server-side searching in mailboxes
  - Protocol contains parallel commands
- Encryption:
  a) connection to port 993
  b) requested by STARTTLS command
- IMAP is implemented in most current MUA

# Distributed database principle



# Hypertext

- The first idea (1945):

  non-linear hierarchical text containing references that allow to continue reading of more detailed information, or similar topic

- The later extension (1965):

  adding some non-textual information (images, sound, video...); sometimes called *hypermedial text*

- Practical implementation (1989):

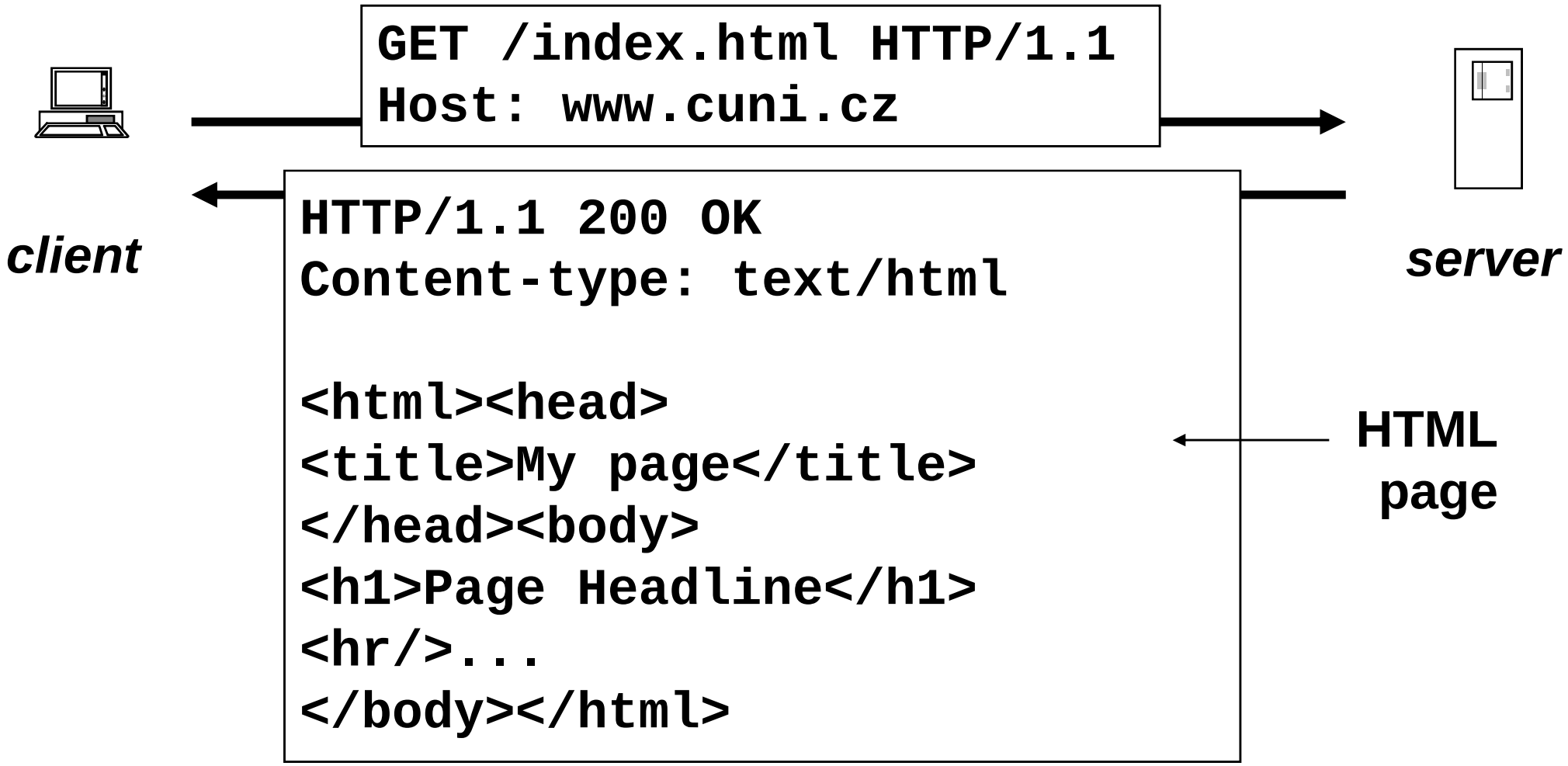
  World Wide Web system developed in CERN

# World Wide Web

- WWW is a distributed hypertext database
- The basic information unit is called hypertext *page* (document) sent by a server to clients on demand
- Documents are written in textual form using HTML (Hypertext Markup Language)
  - describes both content and form
  - final view depends on the client SW and configuration
  - pages are static (saved in files), or dynamic (generated)
- Page transfer is driven by the Hypertext Transfer Protocol (HTTP)

# HTTP example

**URL: http://www.cuni.cz/index.html**



# Hypertext Transfer Protocol

- Current version 1.1, RFC 2616, port 80
- General message form:
  - the first line (request/response)
  - additional header lines
    - request: language, charset, page age, authentication,...
    - response: document type, encoding, expiration,...
  - (optional) document body
- Status (response) codes:

  1xx **informational** (provisional response, processing continues)

  2xx **success** (final response)

  3xx **redirection** (some additional client request expected)

  4xx **client error** (incorrect request)

  5xx **server error** (transient or permanent error on server)

# HTTP methods

| Method | Request body | Response body |
|---|---|---|
| **GET** | --- | requested page |
| **HEAD** | --- | --- |
| **POST** | page parameters | requested page |
| **PUT** | uploaded file | --- |
| **CONNECT** | ⇦ ..... *tunnel* ..... ⇨ | |

# HTTP properties

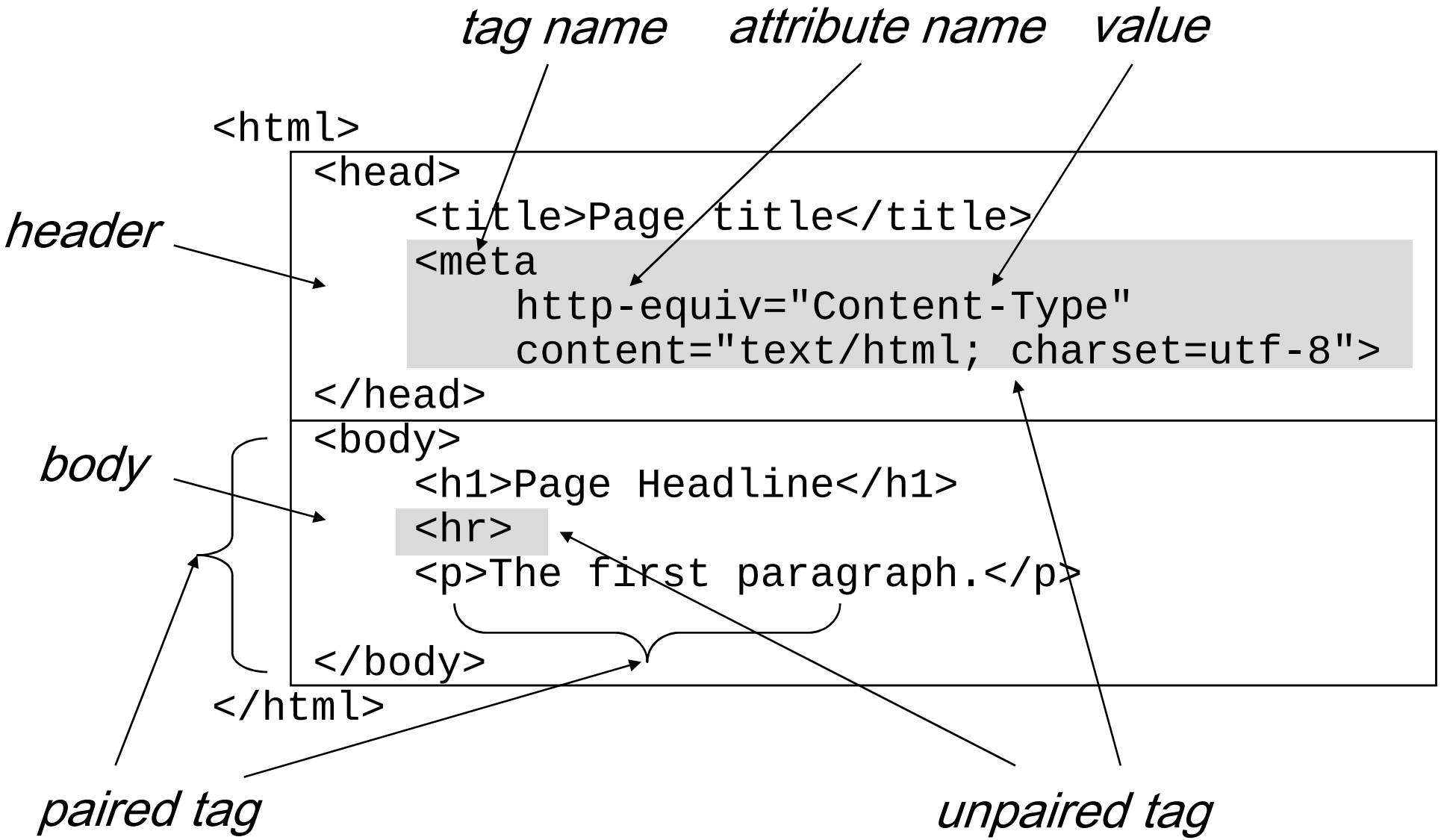
- One request typically leads to a single document (page, picture,...)
- One (persistent) connection can serve for more requests, clients usually open more connections at the same time in parallel
- Individual requests are independent, the communication is stateless; state must be carried via additional data, so called *cookies*:
  - the server generates cookies with connection data and sends them to the client in HTTP headers
  - the client stores them and sends them in HTTP request header when contacting the same server

# Hypertext Markup Language

- Current version 4.01, further progress not very clear (XHTML vs. HTML 5.0)
- Textual page content is supplemented by meta-tags: structural (e.g. paragraph), semantic (e.g. post address), formatting (e.g. bold)
- Application of older SGML (Standard Generalized ML) and predecessor to XML (Extensible ML)

- Tag form: **<***tag* [*attributes*]**>**
- Whitespaces not significant
- Special chars - entities (**`<`**, **`>`**, **`&`**, **` `**...)
- Comments (**`<!-- ... -->`**)

# HTML - document structure



# HTML - hypertext

- References - tag *anchor*:
  - another page reference: `<a href="`*target URI*`">...</a>`
  - fragment positioning: `<a name="`*name*`"></a>`
  - fragment reference: `<a href="#`*name*`">...</a>`

- Images - tag *image* (`img`), attributes:

  `src` image source URI

  `alt` alternative text for text-only clients

  `width`, `height` display size of the picture

  `border` border line

# HTML - formatting

- Basic formatting:
  - paragraph ( ... )
  - headline (<h1> to <h6>)
  - line break (<br>)
  - horizontal line (<hr>)
  - centering ( )

- Font specification:
  - explicit: <font size=... color=... face=...>...
  - physical: bold (<b>), italic (<i>), underscore (<u>), fixed size (<tt>), index (<sub>)...
  - logical: emphasis (, ), source code example (<code>)...

# HTML - lists

```
<ul>
<li>item A</li>
<li>item B</li>
</ul>

<ol>
<li>item A</li>
<li>item B</li>
</ol>

<dl>
<dt>term A</dt>
<dd>explanation</dd>
<dt>term B</dt>
<dd>explanation</dd>
</dl>
```

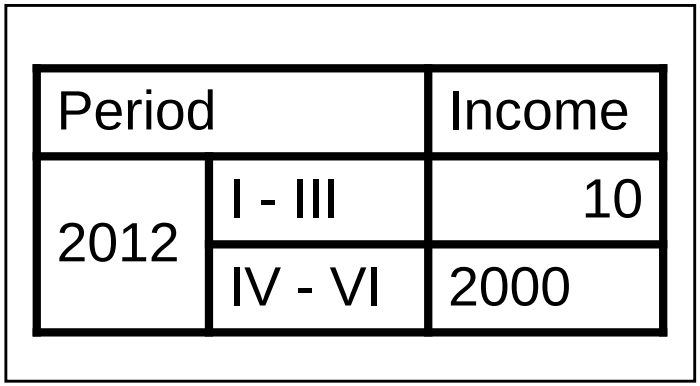
- item A
- item B

1. item A
2. item B

term A
    explanation
term B
    explanation

# HTML - tables

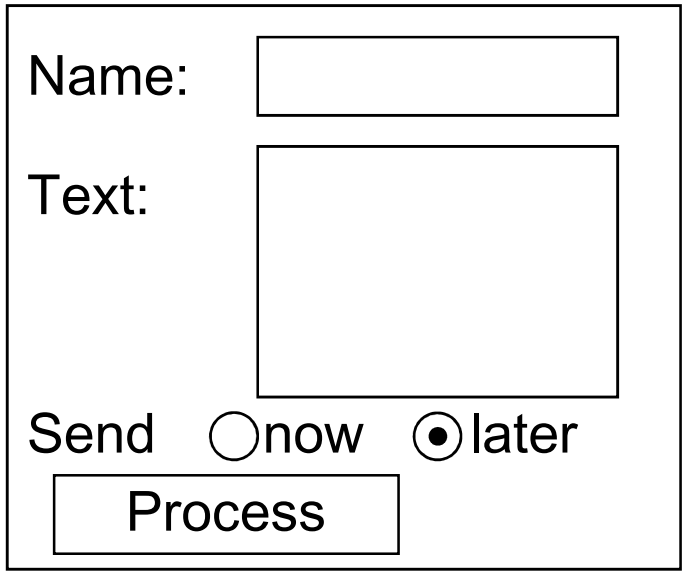
```
<table border="1">
  <tr>
    <td colspan="2">Period</td>
    <td>Income</td>
  </tr><tr>
    <td rowspan="2">2012</td>
    <td>I - III</td>
    <td align="right">10</td>
  </tr><tr>
    <td>IV - VI</td>
    <td>2000</td>
  </tr>
</table>
```



| Period | | Income |
|---|---|---|
| 2012 | I - III | 10 |
| | IV - VI | 2000 |

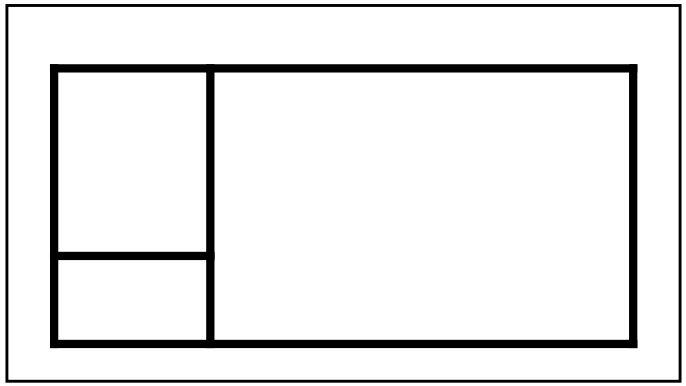
# HTML - forms

```
<form action="mailto.cgi" method="post">
Name: <input name="name">
Text: <textarea name=„text"
          rows="3" cols="40"></textarea>
Send
<input type="radio"
  name="when" value="n">
now
<input type="radio"
  name="when" value="l">
later
<input type="submit"
        value="Process">
</form>
```



# HTML - frames

```
<html>
<head><title>Title</title></head>
<frameset cols="20%,*">
  <frameset rows="*,120">
    <frame name="cont" src="cont.html">
    <frame name="logo" src="logo.html">
  </frameset>
  <frame name="main" src="main.html">
  <noframes><body>
  Document for clients w/o frames support.
  </body></noframes>
</frameset>
</html>
```



# Cascading Style Sheets

- Formatting directly in HTML is complicated
- CSS introduce way how to
  - define properties for whole areas of page
  - create own formatting styles
  - inherit and modify properties of other styles
- CSS ease maintenance of large page sets conforming given formatting rules
- Example:

```
<style type="text/css">
h1 {color: blue; font-style: italic;}
</style>
```

# Page view responsibility

1. Page author
   - introduces basic idea to the page
   - depth of the idea depends on him/her

2. Type and version of browser
   - different (versions of) browsers can interpret identical source code different ways
   - checking the view on various browsers is recommended

3. Browser configuration
   - user usually can affect some attributes of page view by the configuration (e.g. to choose strategy of fonts and colors using)

# Dynamic pages (server-side)

Dynamics driven by the server, no code runs on the client.

- Sending an HTML form leads to running so called *cgi-script* on the server which generates dynamic page text using parameters entered by the user into the form and transferred in the URI, or in the request body
- Page author can request the server SW to include some pieces of text to the page (so called *server-side includes*)
- Page can contain a code for *the HTML preprocessor* (PHP), the client get just the result (date and time here)

```
<?php
    echo date(DATE_RFC822);
?>
```

PHP has wide range of libraries, e.g. for database handling

# Dynamic pages (client-side)

Moving the dynamics (running the code) to the client.

- Java - a language based on C++ ideas, with emphasis on security issues, with libraries for simple creation of the user interface

  Java programs (*applets*) are transferred to clients in form of platform independent **bytecode**, clients interpret and execute it using local libraries

- Javascript - a similar principle, however, the **source code** is being transferred to clients and interpreted there, e.g.:

```
<script>
    document.write("<b>WARNING</b>");
</script>
```

  Today libraries can even communicate with the server.

# WWW security

- User security
  - **plain-text** communication, risk of transferring of sensitive data (passwords, form values)
  - page content can be faked
  - malware in Java(script) code
  - ➢ authentication and encryption (HTTPS: HTTP+SSL)
  - cookies are stored in browsers in readable form, they can be unintentionally sent to a foreign server
- Server security
  - WWW servers are holes for many attacks
  - ➢ carefully maintained system, minimum rights
- Network security
  - if a client and a server negotiate, any traffic can be tunneled via HTTP channel

# Telnet

- Remote host access protocol, port 23
- Abbreviation from *Telecommunication Network*
- One of the oldest protocols, first definition RFC 97 (1971!)
- The user has a network virtual terminal (NVT), the protocol carries chars and NVT control commands in both directions (weakness: e.g. no difference between request/response)
- Main problem: clear-text data transfer (solved in extension in RFC 2946, too late)
- Today:
  - network devices access within separate LAN segment
  - other protocol debugging:

```
> telnet alfik 25
220 alfik.ms.mff.cuni.cz ESMTP Sendmail ...
HELO betynka
250 alfik Hello betynka, pleased to meet you
```

# One Time Password

- General term for mechanisms allowing non-repeatable plain-text user authentication
- Original method:

  Printed list of single-shot passwords.
- Older on-line method:

  Server sends single-shot randomized key, user uses defined procedure how to create the answer (e.g. by a special HW or SW calculator combining the received key and the user password) and sends it back.
- Current method:

  User gets a special authentication item (*token*), which is exactly synchronized with the server and generates a time-limited single-shot authentication code.

# Secure Shell (SSH)

- Secured replacement of older protocols for remote access and file transfer
- Current version 2, RFC 4250-4254, port 22
- SSHv2 extends the possibilities by:
  - opening more secured channels at the same time
  - tunneling different protocols through secured channels
  - accessing the filesystem (SSHFS)
  - ...
- Clients (Windows): putty, winscp
- Commands (Unix):

```
ssh [user@]host [command]
scp [-pr] [user@[host:]]file1 [user@[host:]]file2
```

# Security in SSH

- Clients verify servers
  - according to a key (user confirmed)
  - certificate (signed by trusted CA)
- Servers authenticate users
  - using the password
  - using a challenge/response system (OTP)
  - using public keys (the server sends a challenge encrypted by the user key, the client sends the plain-text response)
- Key usage strategy
  - carefully verify the server key, beware namely when key change is announced („*man-in-the-middle*“ attack danger)
  - permit passwordless login just for private key with password
  - less-important accounts could have passwordless login, but never mutually (A→B & B→A) - protection against *worms*

# Voice over IP

- General name for many technologies for voice transfer over IP network
- Various methods:
  - H.323 standard
  - SIP standard
  - proprietary (Skype)
- Many problems to solve:
  - voice digitalization
  - devices capability negotiation
  - finding the target device
  - binding to regular telephony network

# H.323

- Complex solution of multimedial communication (ITU)
- Based on ASN.1 (binary, even bit-oriented protocols)
- Includes a lot of special protocols, e.g.:
  - H.225/RAS (Registration/Admission/Status) for partner searching by so called *gatekeeper* nodes
  - Q.931 (network layer ISDN) for circuit connecting
  - H.245 for dialog control (negotiation about used properties of available devices)
  - RTP channels (Realtime Transport Protocol, RFC 3550) are used for the multimedia data transfer
  - RTCP (RTP Control Protocol) controls the RTP transfer
- Today gradually replaced by SIP

# Abstract Syntax Notation 1

- Formal definition of data structures, e.g.:

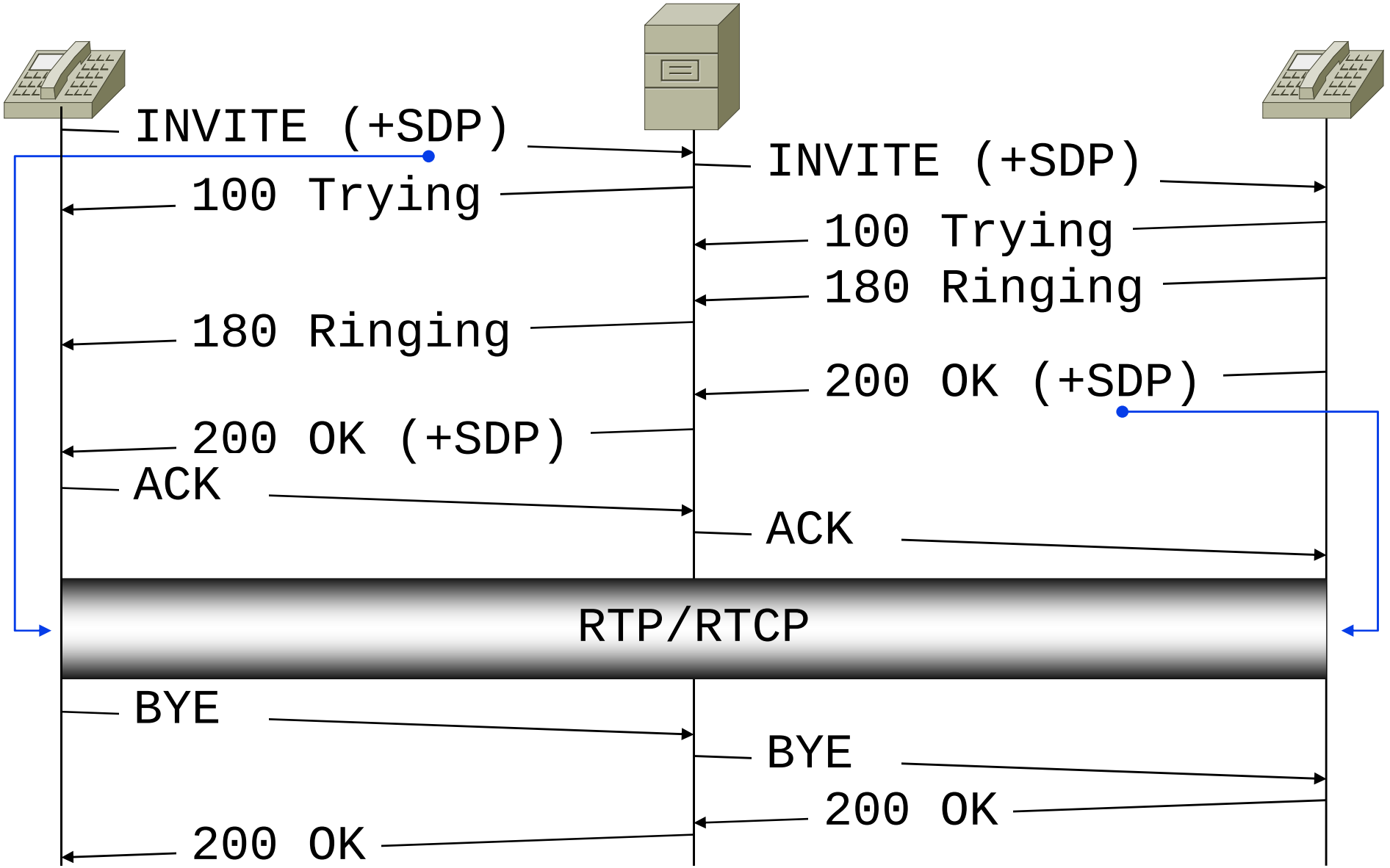
```
Answer ::= CHOICE {
   word PrintableString,
   flag BOOLEAN }

SignedData ::= SEQUENCE {
   version Version,
   digestAlgorithms DigestAlgorithmIdentifiers,
```

- Comes from the 80's (and looks like it)
  - e.g.: enumeration value is stored into as many **bits** as needed, in front of it, the 0 bit is added, however, the value of 1 in this bit means that the type is stored in **different** number of bits
- Automatic generation of protocol parser possible
- Data could be shorter, but opaque
- Usage examples: H.323, X.509

# Session Initiation Protocol

- Replacement of complex H.323 by simpler one
- RFC 3261, port TCP & UDP 5060
- Protocol architecture is similar to HTTP, most information carried in headers
- Does not handle the multimedia transfer itself (this is often done by RTP/RTCP channels)
- Handles only the signalization (finding the partner and contacting it)
- Data channel properties negotiation usually controlled by SDP (Session Description Protocol, RFC 4566), its data is carried encapsulated into SIP message bodies
- End node can register at some registrator and thus bind itself to regular public telephony network

# SIP session example



# Filesystem sharing

- Connecting a foreign filesystem transparently into local one
- Network File System (NFS)
  - originally developed at Sun Microsystems, today IETF
  - current version 4, RFC 3530, port 2049 (UDP, TCP)
  - source identification: server:path
  - authentication: Kerberos
  - note: relation (RPC) and presentation (XDR) layer
- Server Message Block (SMB)
  - originally developed at IBM, later adopted by Microsoft
  - open implementation Samba (UNIX)
  - source identification: UNC (\\server_name\source_name)
  - authentication: usually username and password

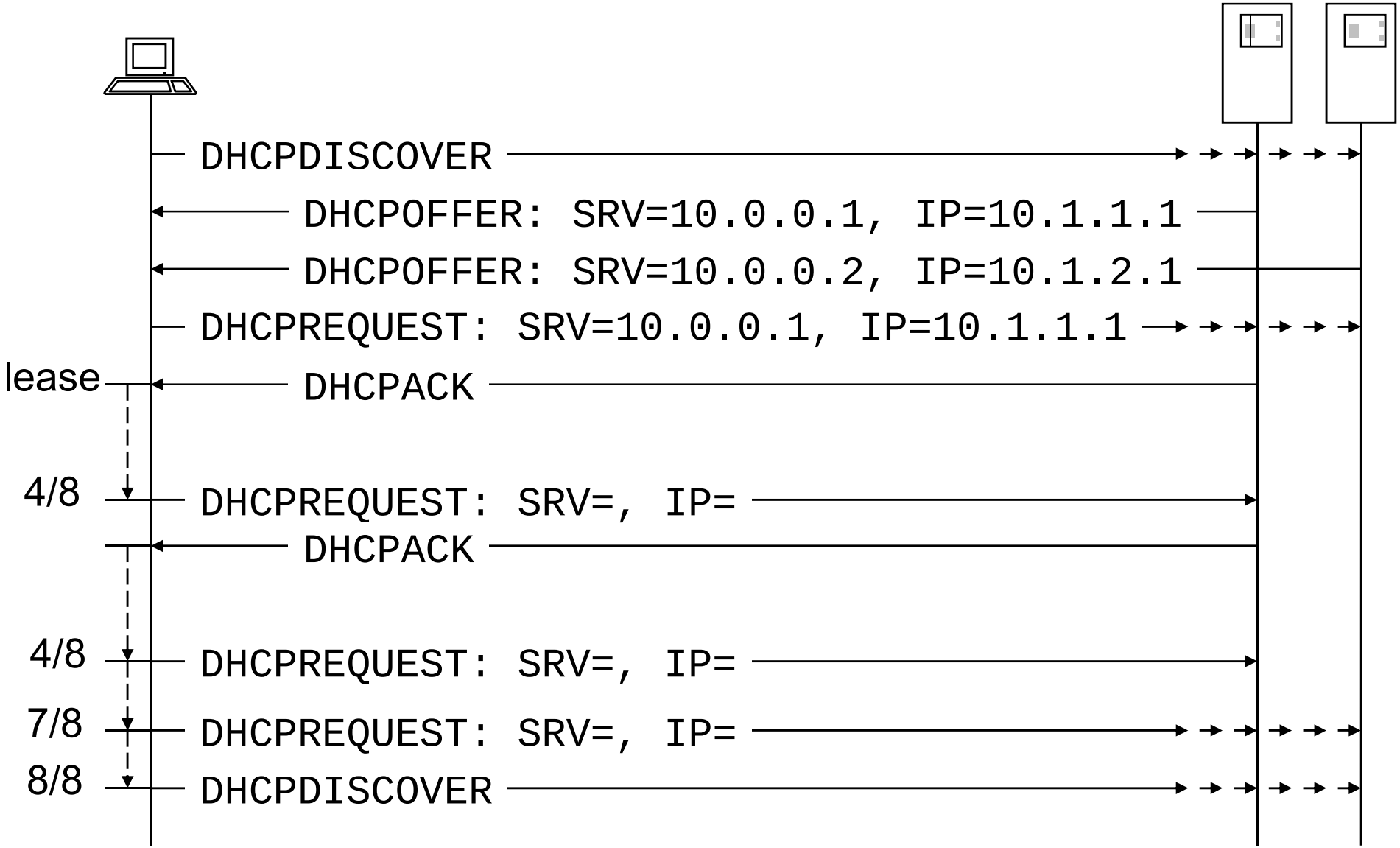
# Network Time Protocol

- Time synchronization among network nodes
  - file timestamp consistency
  - comparing logs from different machines
- Current version 4, RFC 5905, port 123 (UDP)
- Client contacts servers listed in its configuration
- Servers can have different accuracy - *stratum*:
  - 0: atomic clock, GPS clock
  - 1: synchronized with stratum 0 source, ...
- Problem: server responses have (different) delay
  - using timestamps, the most probable interval for the time response from every server is computed
  - Marzullo's algorithm is used for choosing the best intersection of intervals

# BOOTP and DHCP

- Bootstrap Protocol, RFC 951, was developed for automatic configuration of diskless stations
  - client sends (to all) a request with MAC address
  - server finds proper answer and sends IP address, name...
- Replaced by DHCP (Dynamic Host Configuration Protocol)
  - compatible message form
  - besides the static address allocation, also dynamic one
  - limited lease time
  - more servers may co-operate
- RFC 2131, UDP ports 67 (server) a 68 (client)
- Client chooses the best offer (by address, lease time,...)

# DHCP process



# SSL, TLS

- Secure Socket Layer (3.0), Transport Layer Security (1.0)
- Interlayer between the transport and application layer allowing authentication and encryption
- Many protocols uses it (e.g. HTTPS on port 443)
- Principle:
  1. A client sends a request for an SSL connection + parameters.
  2. The server responses with parameters and own certificate.
  3. The client verifies the server, generates a common key basis, encrypts it by the server public key and sends it back.
  4. The server decrypts the key basis. Using this basis, both the client and the server generate common encryption key.
  5. The client and the server mutually negotiate, that from this moment both will encrypt their communication by this key.

# Presentation layer (OSI 6)

- Idea of a general model describing all encoding
  - data types: integers, strings,...
  - data structures: arrays, records, pointers,...
- Very complicated in general: who and when en/decrypts
- Implementation attempt: ASN.1
- TCP/IP suppressed the need of a general model:
  the format definitions are included into the application protocols, conversions must be done by every application
- Practical problems:
  - textual line endings: CRLF (`0x0D`, `0x0A`)
  - byte order: *big endian* (1 = `0x00`, `0x00`, `0x00`, `0x01`),
    e.g. Intel has *little endian* (1 = `0x01`, `0x00`, `0x00`, `0x00`)

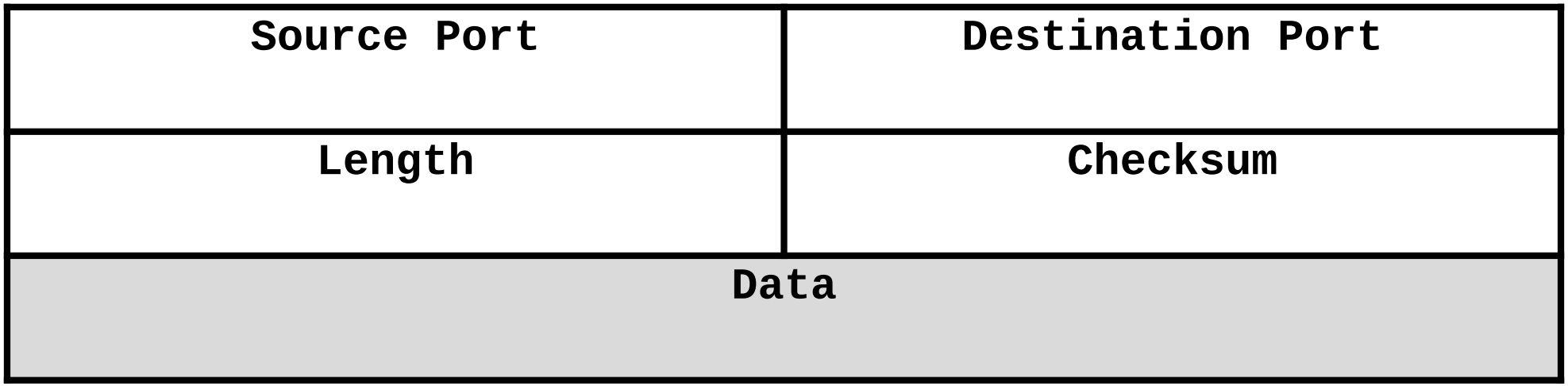
# Relation layer (OSI 5)

- Idea of a general dialog model
  - one dialog can consist of more connections
  - one connection can carry more dialogs
- TCP/IP suppressed the need of a general model: the dialog principle has been included directly into the application protocols, e.g.:
  - within one SMTP connection a client can send several mails to the server
  - SIP (could be considered as OSI 5 layer protocol) initializes dialog using more partial media data channels

# Transport layer (OSI 4)

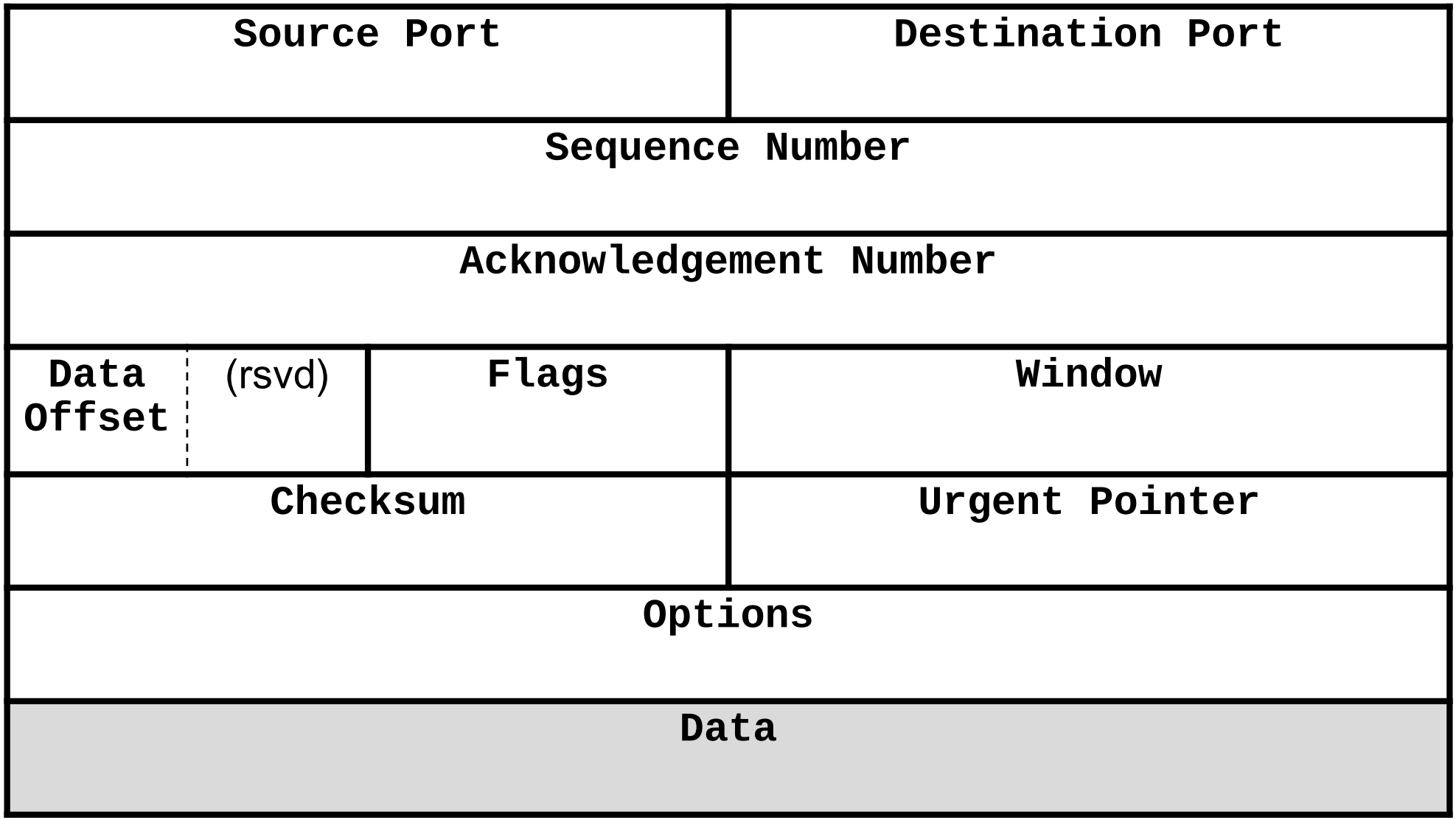
- Layer functions:
  - is responsible for end-to-end data transfer
  - mediates network services for application protocols having various requirements to the transfer channel
  - allows running of multiple applications (both clients and servers) on the same network node
  - (optionally) guarantees data transfer reliability
  - (optionally) segments data for smoother transfer and puts them back together in proper order for applications
  - (optionally) provides data flow control (e.g. “egress speed”)
- On the TCP/IP transport layer used:
  - TCP (Transmission Control Protocol, connection-oriented)
  - UDP (User Datagram Protocol, connectionless)

# UDP datagram structure



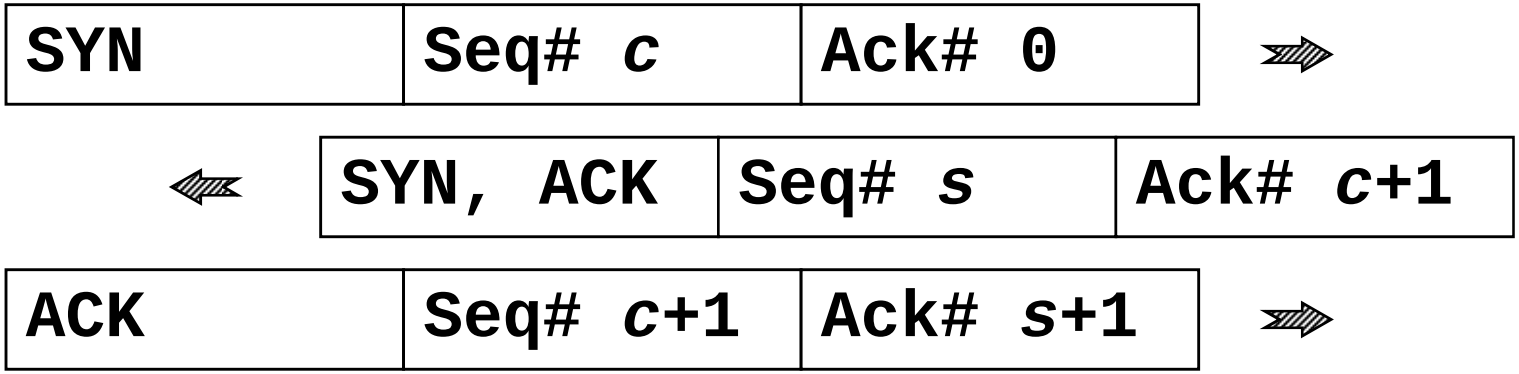

- Application data in UDP are sent as separate messages
- Applications must adjust the PDU size and guarantee control logic
- No “connection” exists in UDP

# TCP packet structure

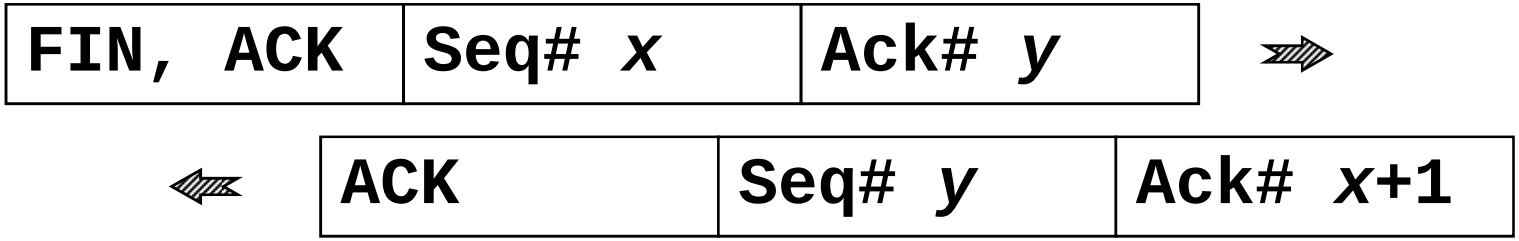


# Connection initialization and termination

- TCP connection initialization (“three-way handshake”):

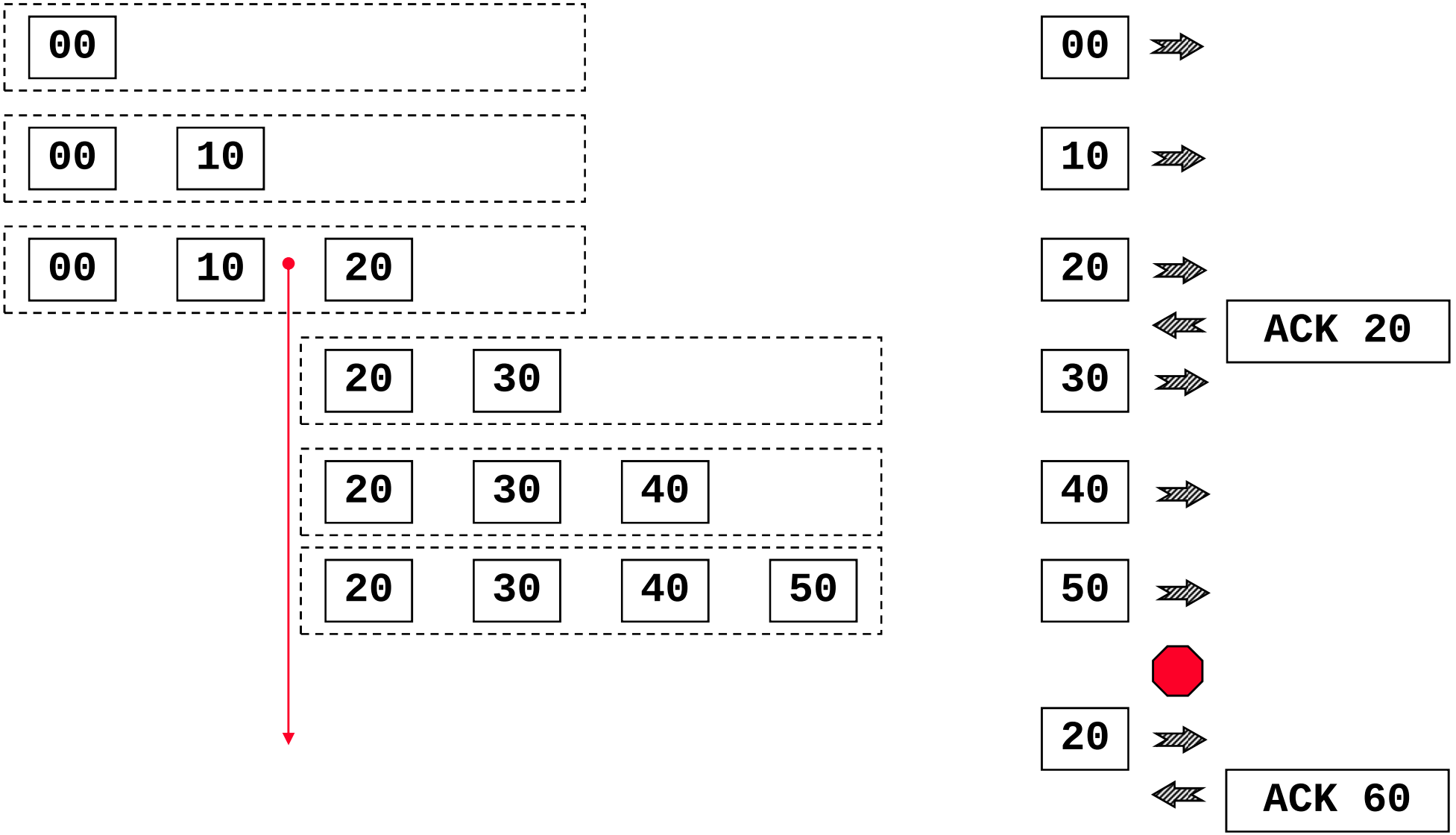



- Connection closure (one-way closure):




The partner (immediately, or later) closes his side, too.

# TCP windows

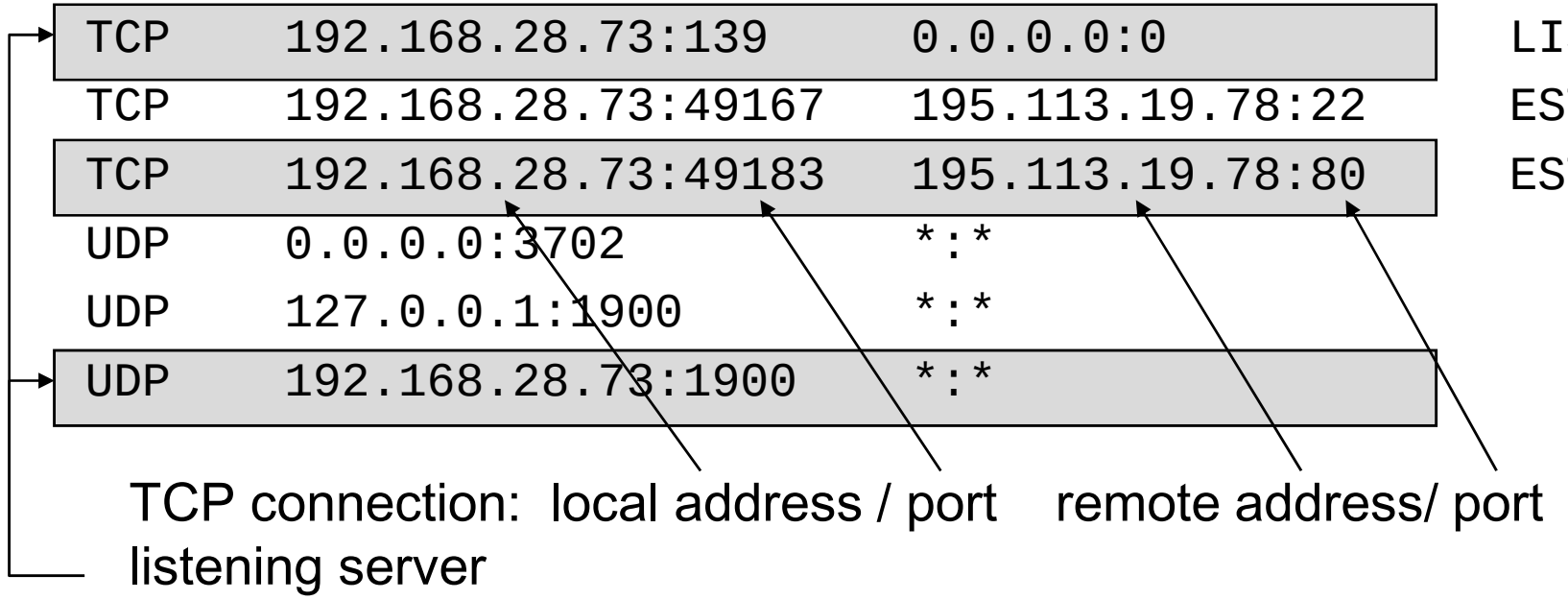

00
00
10
00
10
20
20
30
20
30
40
20
30
40
50
00
10
20
ACK 20
30
40
50
20
ACK 60

# TCP flags

- **SYN** - packet for segment numbers synchronization („`Sequence number`“ initialization)
- **ACK** - packet acknowledges delivery of all packets up to „`Acknowledgement number`“ (not inclusive); packet can but need not to contain also data
- **PSH** - informs that the delivered block is completed and can be passed to the application („`push`“)
- **FIN** - sender closes own side of the connection, no more data will be sent
- **RST** - sender refuses to accept the connection, or immediately terminates the connection („`reset`“)
- **URG** - packet contains urgent (*out-of-band*) data, the address is in „`Urgent pointer`“

# Existing sockets listing

```
C:\Users\forst>netstat -an
Active Connections
  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:623            0.0.0.0:0              LISTENING
  TCP    127.0.0.1:49209        127.0.0.1:49210        ESTABLISHED
  TCP    127.0.0.1:49210        127.0.0.1:49209        ESTABLISHED
  TCP    192.168.28.73:139      0.0.0.0:0              LISTENING
  TCP    192.168.28.73:49167    195.113.19.78:22       ESTABLISHED
  TCP    192.168.28.73:49183    195.113.19.78:80       ESTABLISHED
  UDP    0.0.0.0:3702           *:*
  UDP    127.0.0.1:1900         *:*
  UDP    192.168.28.73:1900     *:*
```



# Network layer (OSI 3)

- Main function: the transport of data passed down by the transport layer to the target host
- Essential operations:

  *addressing* - network layer protocols define the format and structure of communicating partners' addresses

  *encapsulation* - control data needed for the transfer (namely addresses) must be included into PDU

  *routing* - searching the best way to the target through intermediate networks

  *forwarding* - passing the data from the input network interface to the output one

  *decapsulation* - unpacking the data and passing to the transport layer
- Protocol examples: **IPv4**, **IPv6**, IPX, AppleTalk

# Internet Protocol (IP)

- Properties:
  - connectionless (all datagrams run by individual paths)
  - best effort (unreliable, logic delegated to higher layers)
  - media independent (higher layers need not to bother with it)
- Addresses:
  - contain network address part and node address part
  - IPv4: 4 bytes
  - IPv6: 16 bytes
- Assignment:
  - central: IANA (Internet Assigned Numbers Authority)
  - regions: RIR (5x, Europe: RIPE NCC)
  - further: ISP
  - local network: network management (manually/automatically)

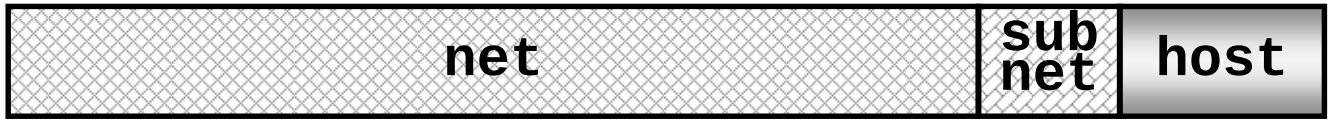
# IPv4 addresses

- Originally: one byte
- 1975 (RFC 687): three bytes („*This expansion is adequate for any forseeable ARPA Network growth.*“)
- 1976 (RFC 717): one byte (network) + three bytes (host)
- 1981 (RFC 791): classes A, B and C

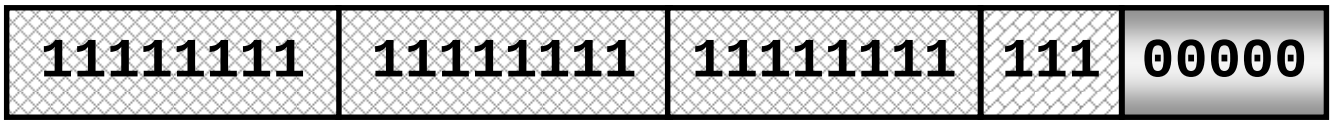
| Class | byte 1 | byte 2 | byte 3 | byte 4 | 1st byte | Nets | Hosts |
|---|---|---|---|---|---|---|---|
| A | 0 | net | host | | 1-126 | 126 | ~16 M |
| B | 10 | net | host | | 128-191 | ~16 k | ~64 k |
| C | 110 | net | host | | 192-223 | ~2 M | 254 |
| D | 1110 | net | | | 224-239 | multicast | |
| E | 1111 | | | | 240-255 | experimental | |

# Subnetting

- Network splitting by expanding network part of address:




  using so called network mask (*netmask*),
  in this case 255.255.255.224:

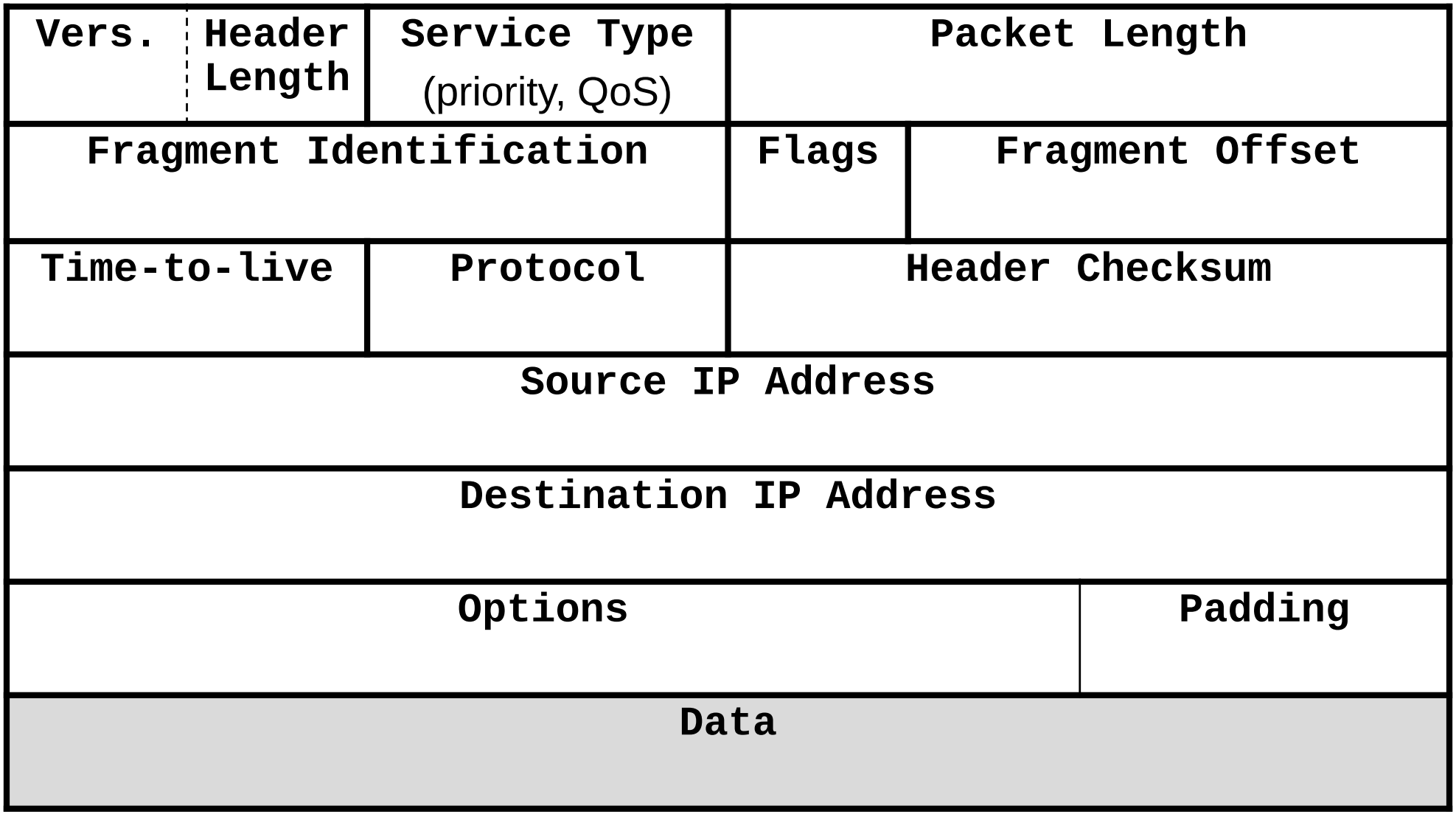



- Subnets "all-zeros" and "all-ones“ are not recommended, so we have here just 6 x 30 addresses (70%)
- Non contiguous mask is possible, but usually not used
- Nowadays, the classes are often ignored (*classless* mode), using only the prefix length in bytes (e.g. 193.84.56.71/27)
- The term *variable length subnet mask* (VLSM) describes situation when various masks are used in a network
- Moving the border in opposite direction: *supernetting*

# Special IPv4 addresses (RFC 5735)

- Special addresses „by design“
  - **this host** (used only as source one): `0.0.0.0/8`
    - an interface with so far unassigned address
  - **loopback** (RFC 1122): `127.0.0.1/8`
    - the address of local host, enables loop creation
  - **network broadcast** (RFC 919) <*network address*>.<*all 1s*>
    - „to all within the net“, normally delivered to the target network
  - **limited broadcast** (RFC 919): `255.255.255.255`
    - „to all within this net“, not allowed to leave the network
- Special addresses „by definition“ (not allowed to leave the network)
  - **private addresses** (RFC 1918):
    `10.0.0.0/8, 172.16-31.0.0/16, 192.168.*.0/24`
    - for the local network traffic only, assigned by a network administrator
  - **link-local addresses** (RFC 3927): `169.254.1-254.0/16`
    - for connections within local network segment only,
      a host can choose it by itself

# IPv4 datagram structure



# Internet crisis

- Address space exhausting
  - Nature of the problem: due to rough space fragmentation large wasting occurs
  - Partial solution: classless address blocks assigning, recycling of unused blocks, private addresses + NAT
  - IANA has no more blocks for regions, APNIC 2011/04, RIPE 2012/09, others will follow soon

- Routing tables growth
  - Nature of the problem: large number of not contiguously assigned blocks overfills routing tables
  - Partial solutions: address reallocation, CIDR (Classless InterDomain Routing) aggregation

# IPv6 addresses

- Long development, the final format: 128 bits (16 bytes)
- Notation: `fec0::1:800:5a12:3456`
- Address types:
  - unicast - address of one node; special ones (RFC 5156):
    - *Loopback* (`::1/128`)
    - *Link-Scope* (`fe80::/10`), formerly *link-local*
    - *Unique-Local* (`fc00::/7`), formerly *site-local*, analogy of private addresses in IPv4
  - multicast (`ff00::/8`) - address of group of nodes (interfaces)
  - anycast - formally this is a unicast address, assigned to more nodes (interfaces); intention: server farms
  - no analogy of broadcast
- Migration from IPv4 will be facilitated by various form of tunneling IPv4 into IPv6 and vice versa

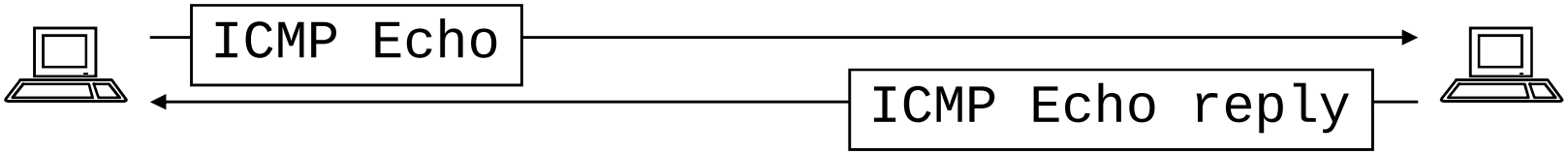
# Internet Control Message Protocol

- Used for sending control messages concerning IP:
  - **`Echo, Echo Reply`** ... host reachability test (program **`ping`**)
  - **`Destination Unreachable`** ... host, service, or network unreachable
  - **`Redirect`** ... routing table change appeal
  - **`Time Exceeded`** ... null TTL (routing error)
  - **`Source Quench`** ... request to slow data flow
  - **`Parameter Problem`** ... datagram header error
  - **`Timestamp Request, Reply`** ... transfer time estimation
  - **`Information Request, Reply`** ... network address request
  - **`Address Mask Request, Reply`** ... network mask request
- Uses IP datagrams; however, no transport protocol

# Ping

- Basic tool for network diagnostics

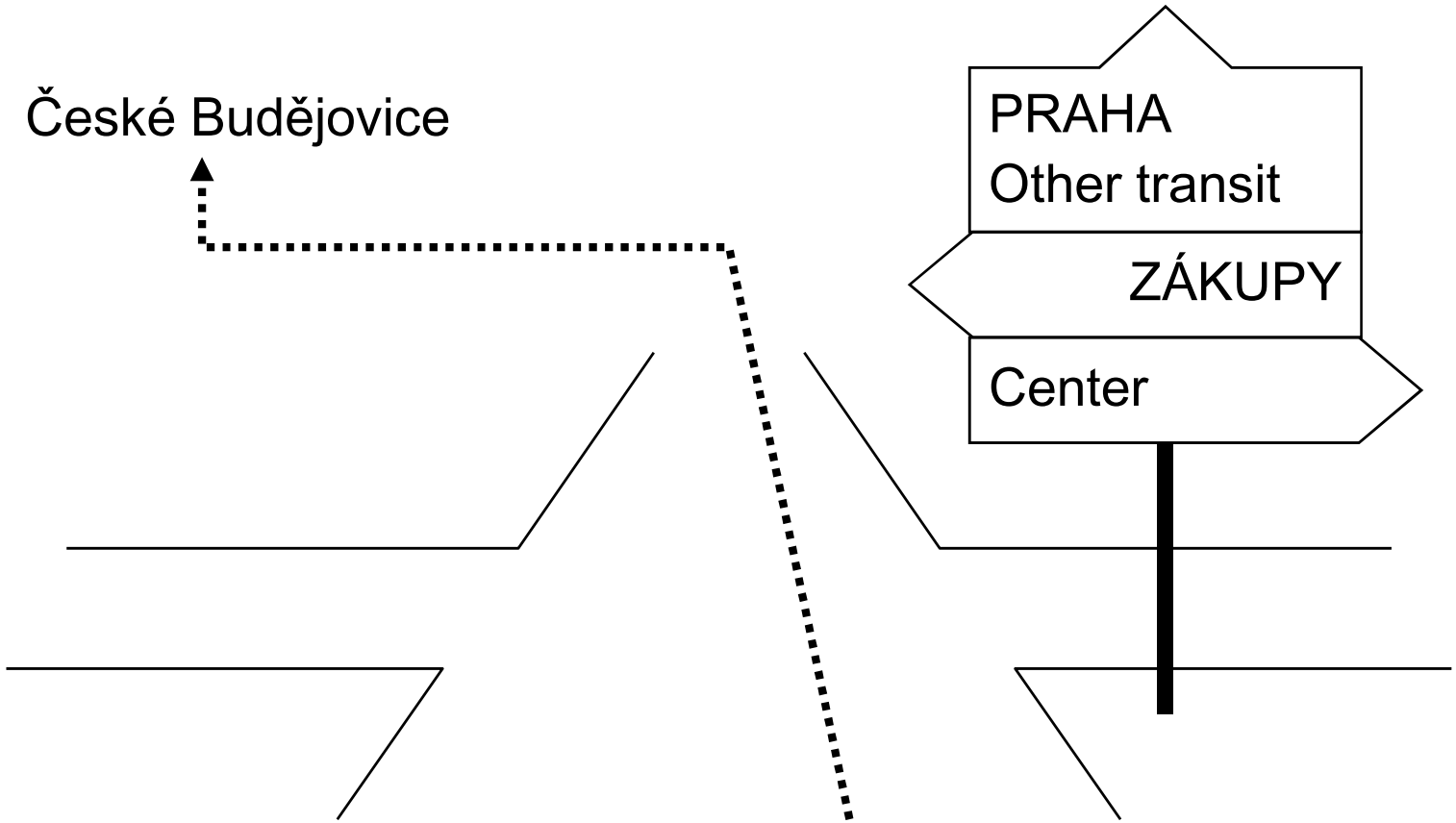
```
betynka:~> ping alfik
```




```
PING alfik.ms.mff.cuni.cz (195.113.19.71): 56 data bytes
64 bytes from 195.113.19.71: icmp_seq=0 ttl=64 time=0.214 ms
64 bytes from 195.113.19.71: icmp_seq=1 ttl=64 time=0.323 ms
64 bytes from 195.113.19.71: icmp_seq=2 ttl=64 time=0.334 ms
^C
--- alfik.ms.mff.cuni.cz ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.214/0.290/0.334/0.054 ms
```

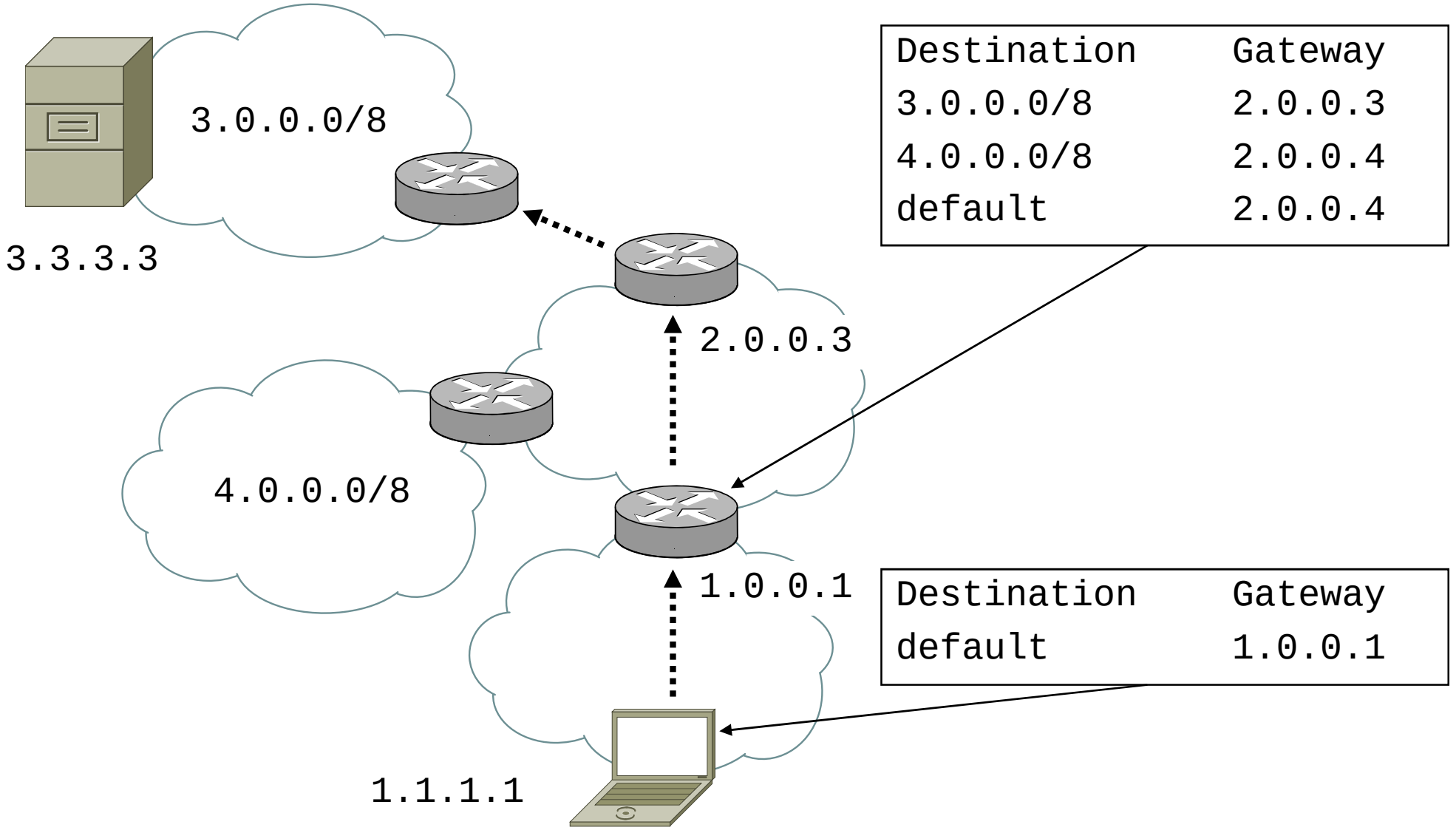
  - no need of any special SW on the target machine
  - pure network layer reachability, tells nothing about services

# Routing (road)



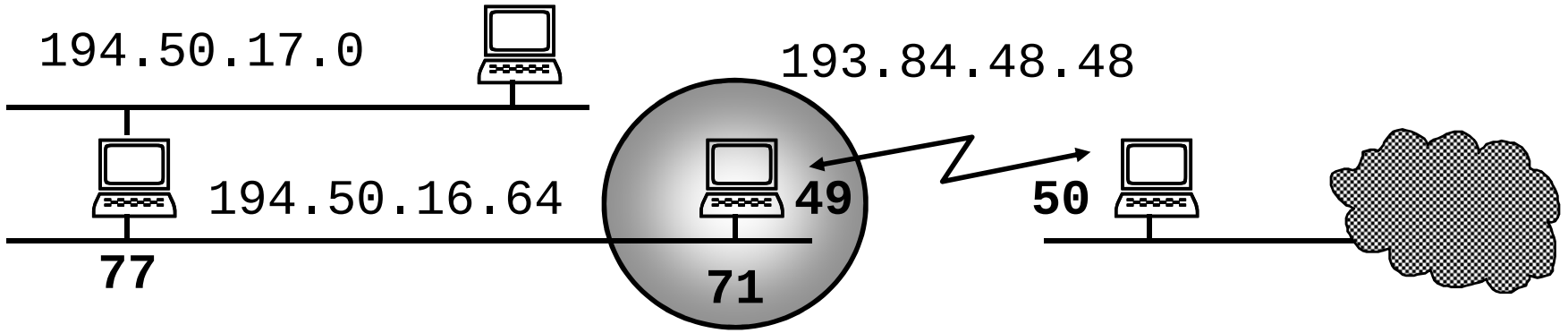

- On every crossing we decide according to signs
- For proper interpretation we need linguistic and geographic knowledge

# Routing (network)



# Routing table example

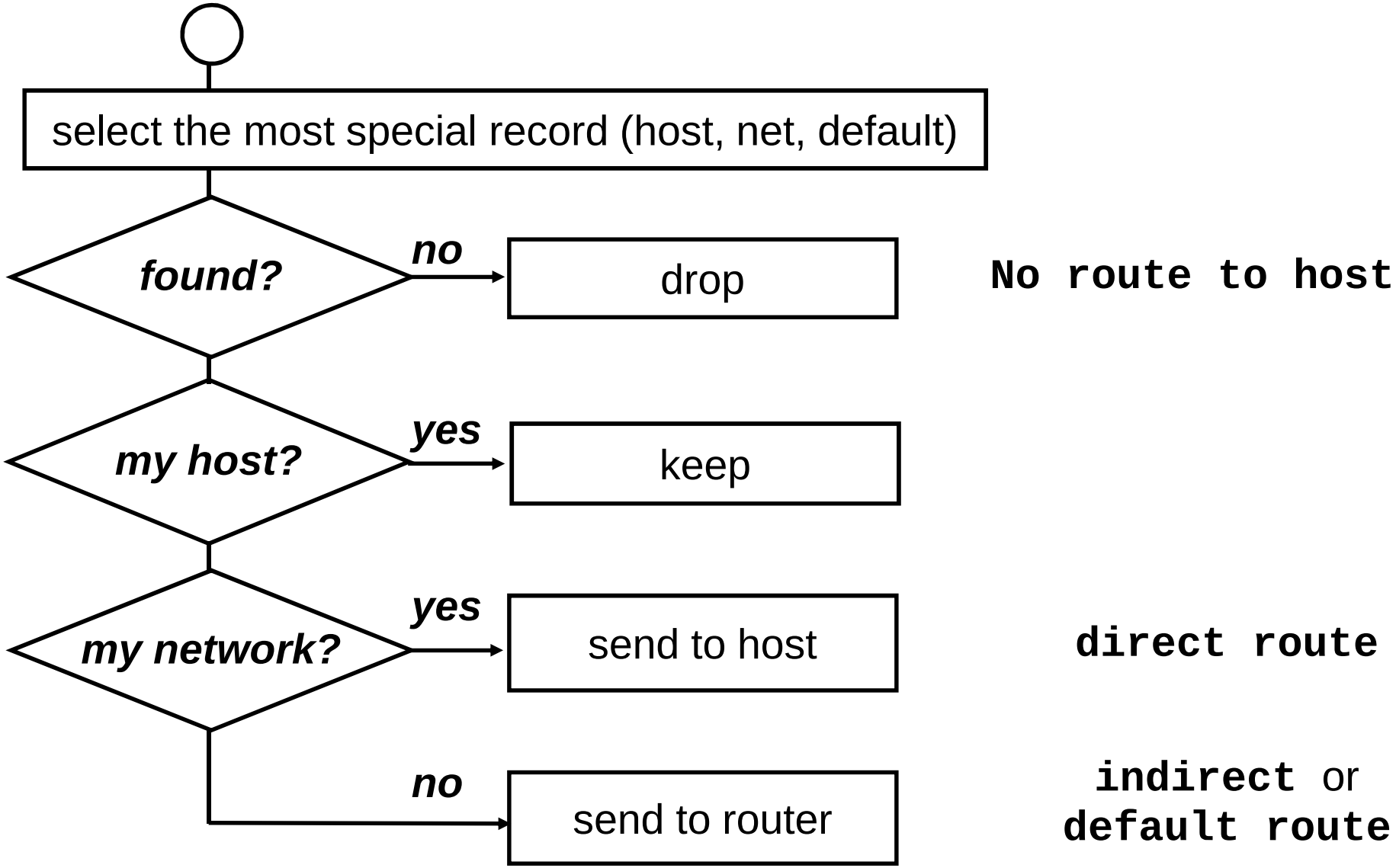
| Destination | Gateway | Mask | |
|---|---|---|---|
| 193.84.48.50 | 193.84.48.49 | 255.255.255.255 | ***direct, host*** |
| 194.50.16.64 | 194.50.16.71 | 255.255.255.224 | ***direct, subnet*** |
| 194.50.17.0 | 194.50.16.77 | 255.255.255.0 | ***indirect, net*** |
| default | 193.84.48.50 | 0.0.0.0 | ***default*** |



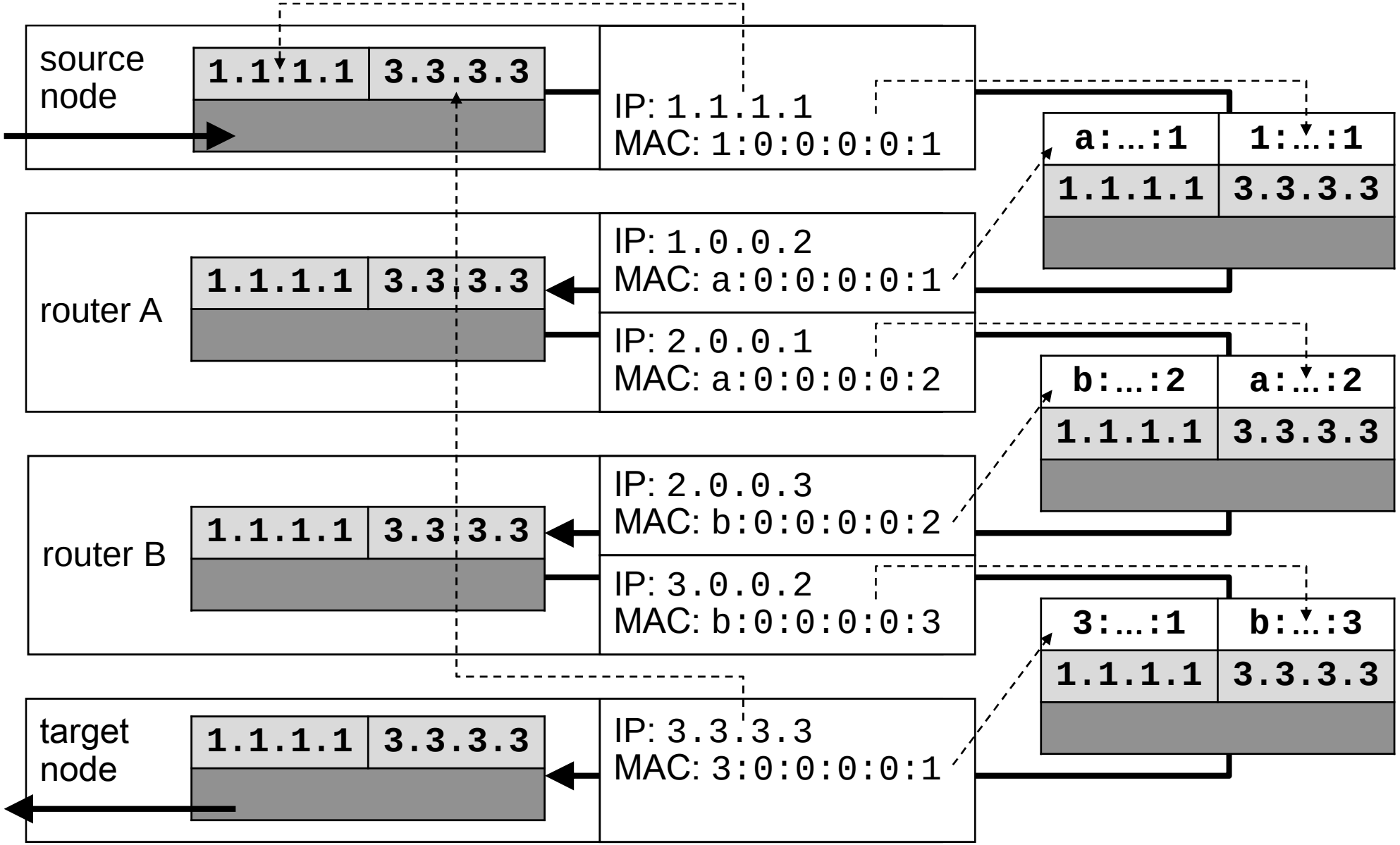
# Routing principles

- Every node in the TCP/IP network should use routing
- Routing table record contains following columns:
  *destination*, *mask*, *gateway*
- Mask tells „considered part“ of the destination address
- Former destination categories: host (/32), net, default (/0)
- Record types:
  - *direct* (directly connected net, “gateway” is “my” address)
  - *indirect*, *default*
- Record origin:
  - *implicit* (added by default after configuring an interface)
  - *explicit* (added “manually” by entering the command)
  - *dynamic* (added during the work using information sent by partners on the network)

# Routing algorithm



# Forwarding


source node
1.1.1.1
3.3.3.3
IP: 1.1.1.1
MAC: 1:0:0:0:0:1
a:...:1
1:...:1
1.1.1.1
3.3.3.3
router A
1.1.1.1
3.3.3.3
IP: 1.0.0.2
MAC: a:0:0:0:0:1
IP: 2.0.0.1
MAC: a:0:0:0:0:2
b:...:2
a:...:2
1.1.1.1
3.3.3.3
router B
1.1.1.1
3.3.3.3
IP: 2.0.0.3
MAC: b:0:0:0:0:2
IP: 3.0.0.2
MAC: b:0:0:0:0:3
3:...:1
b:...:3
1.1.1.1
3.3.3.3
target node
1.1.1.1
3.3.3.3
IP: 3.3.3.3
MAC: 3:0:0:0:0:1

# Address Resolution Protocol

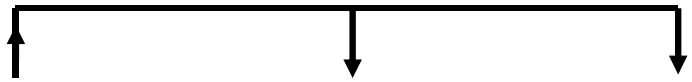
- MAC (Ethernet) addresses to network (IP) ones conversion
- Unknown addresses learned using ARP broadcast requests:

| Ethernet=1 | IP=0x0800 | | ARPreq=1 |
|---|---|---|---|
| Sender MAC | | | Sender IP |
| FF:FF:FF:FF:FF:FF | | | Target IP |

- Results stored into ARP *cache*
- Unicast response (the responder must also add proper requestor data into own ARP cache)
- No proof of the authenticity of the answer
- Gratuitous ARP: unsolicited ARP (faster changes, risk)
- ARP cache listing: **`arp -a`**
- Limited to a link segment, OSI 3 operates between networks

# Proxy ARP

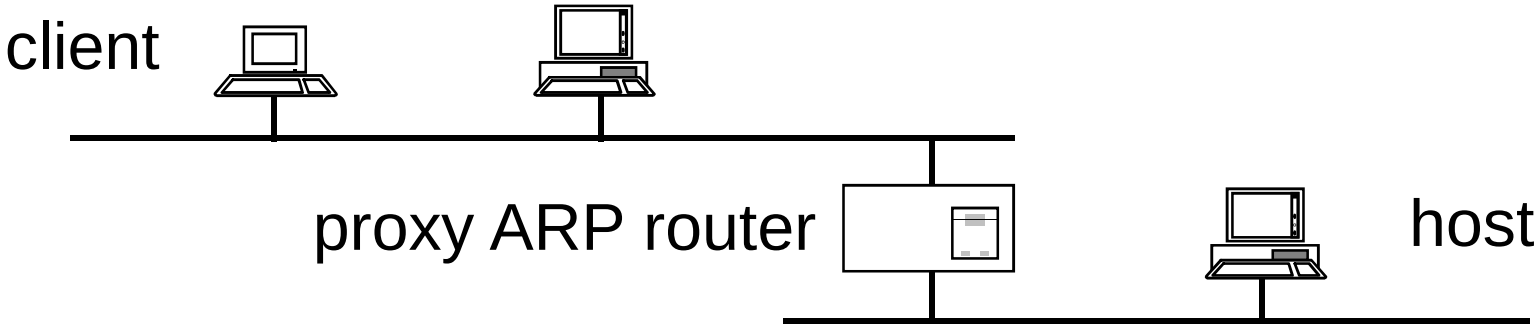
① **client** sends **ARP `request`** with the **host** IP address



② **router** knows that ARP won't succeed, thus it sends **ARP `reply`** with MAC address of the **router**

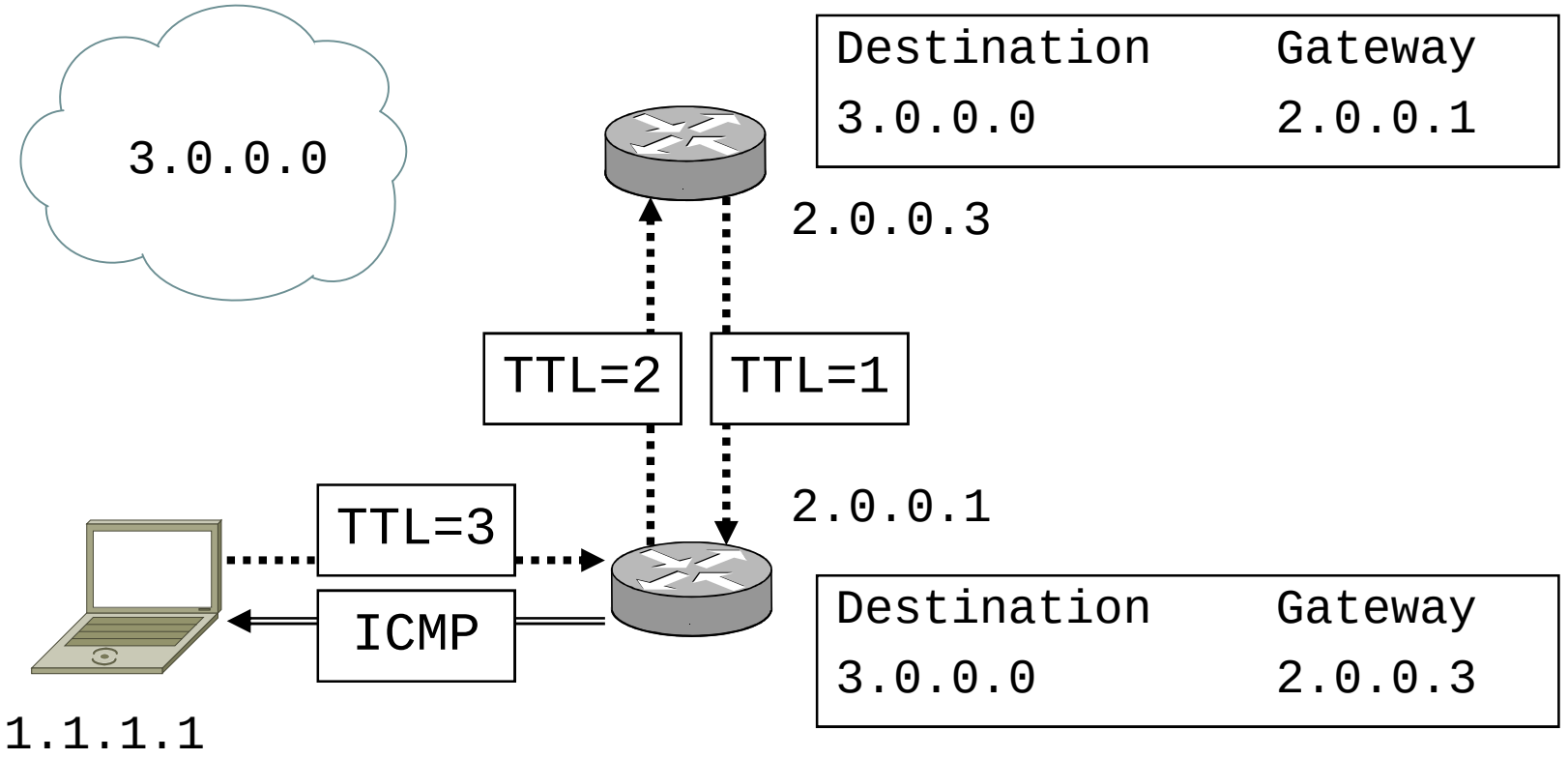
③ **router** MAC assigned to the **host** IP in **client**'s ARP cache

④ **client** sends data to the **host** with the **router** MAC address



# Time To Live (IP)

- Mechanism defending against infinite loop in case of routing loop (e.g. errorneous router configuration)
- Number of hops the packet can still traverse
- When reaching zero, the ICMP Time Exceeded is sent



# Routing diagnostics

- Routing table listing: **`netstat -r`**[**`n`**]
  or: **`route print`**

```
Destination Gateway Flags Ipkts ... Colls Interface
194.50.16.0 this     U    15943 ... 0     tu0
127.0.0.1   loopback UH                   lo0
default     gw       UG                   tu0
193.84.57.0 gate     UGD                  tu0
```

- Path check: **`traceroute`**, **`tracert`**

```
1 gw.thisdomain (194.50.16.222)  2 ms  1 ms  1 ms
2 gw.otherdomain (193.84.48.49) 12 ms 15 ms 15 ms
3 target.net (195.113.0.2)          *     *     *
```

# Network configuration

UNIX

- IP address: **`ifconfig`** *interface IP_adr* [ **`netmask`** *mask* ]
- default router: **`route add default`** *router*
- usually stored in a configuration file, details vary according to the OS

Windows

**`Control Panel ⇨ Network and Internet`**
**`⇨ Network Connections`**
**`⇨ Local Area Connection ⇨ Properties`**
**`⇨ TCP/IPv4 ⇨ Properties`**
**`⇨ General`**

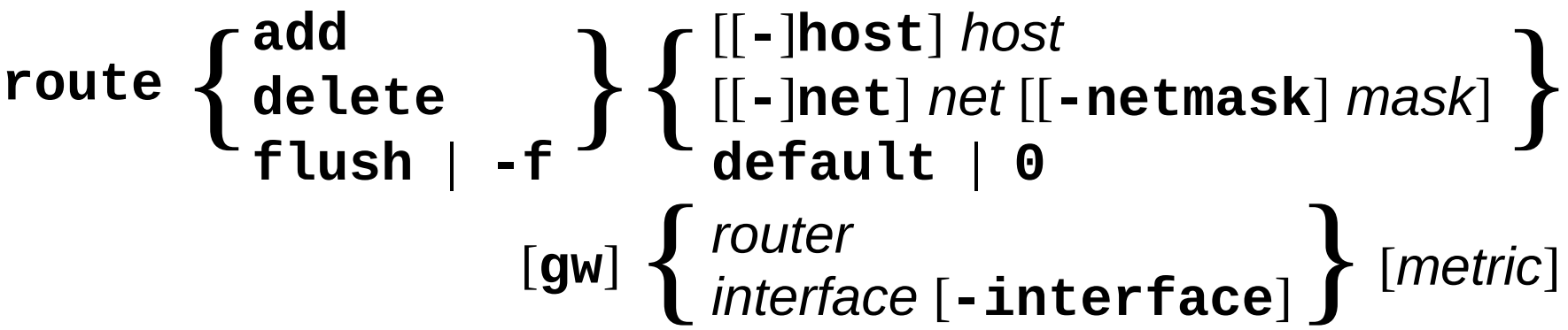
# Static management of routing tables

Routes installed during startup by configuration

- rigid, not flexible
- problems with subnetting
- complicated solution of backup routes

+ less sensitive, more robust
+ working in totally heterogeneous environment

⇨ suitable for smaller, stable networks



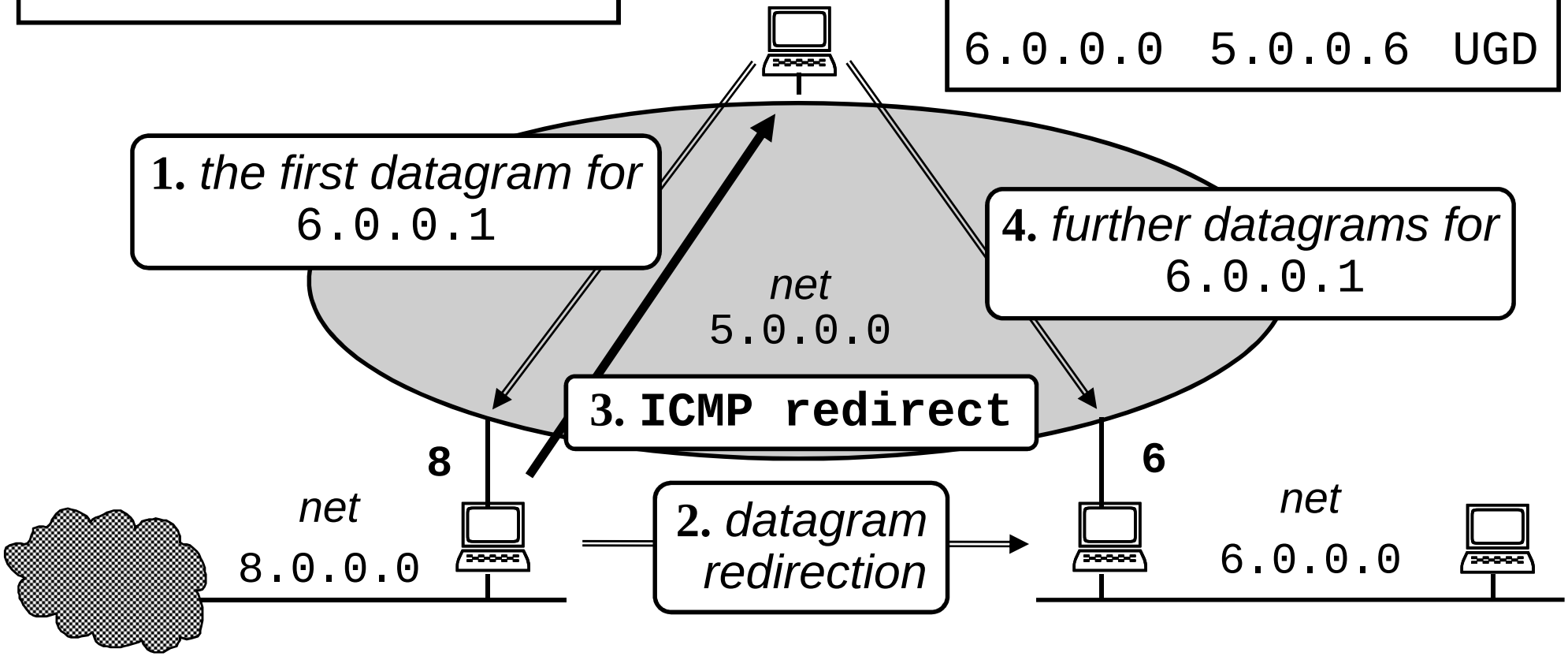
# Redirection

*original* table content:

```
default 5.0.0.8 UG
```

*new* table content:

```
default  5.0.0.8  UG
6.0.0.0  5.0.0.6  UGD
```



# Dynamic management of routing tables

Routers exchange information about the network using some *routing protocol*; end nodes can listen to it, too

+ simpler configuration changes
+ network can react to failures
+ no manual administration of routing tables needed
- more sensitive to errors and attacks

- host must run an application handling the protocol
  - e.g. routed, gated, BIRD (developed at MFF)
  - RIP and OSPF are two of the most popular protocols for local networks *(internal routers)*

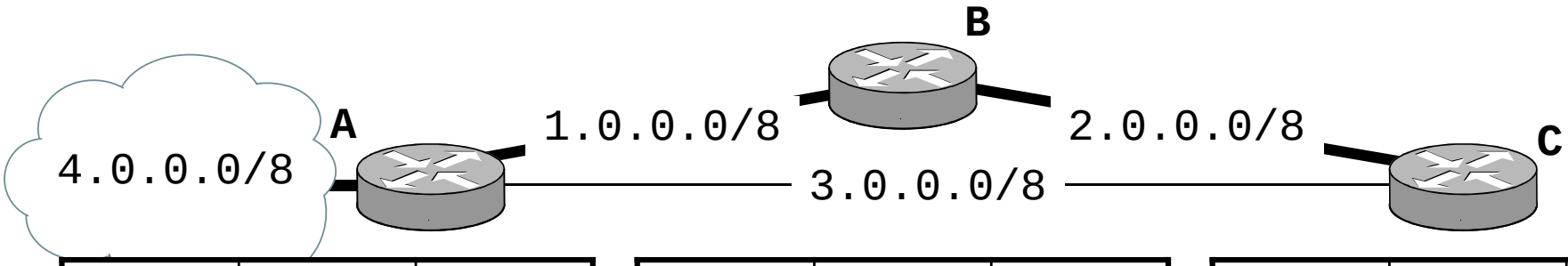
# Distance vector protocols

- Basic idea:
  - routing table records contain also “distance” (*metrics*)
  - each router sends periodically its table to all neighbors, they modify own tables and send them further
- Advantages:
  - simple, easy to implement
- Disadvantages:
  - slow reaction to failures
  - metrics poorly reflects lines properties (bandwidth, reliability, price...)
  - limited network diameter
  - one router calculation mistake affects the whole network (routing loops possible)

# Routing Information Protocol

- The oldest routing protocol, RFC 1058
- Properties:
  - metrics: path length (number of routers, *hop count*)
  - network diameter: limited to 15 hops, 16 is „infinity“
  - algorithm for getting the shortest paths: Bellman-Ford
- Current version 2
  - uses UDP port 520, multicast address 224.0.0.9
  - support for subnetting incl. VLSM
  - network convergence speedup mechanisms (triggered updates, split horizon, poison reverse)
- Available on almost all systems
- Not suitable for large, complex, or rapidly changing nets

# Metrics and lines properties




| | | |
|---|---|---|
| 1../8 | - | 1 |
| 3../8 | - | 3 |
| 4../8 | - | 1 |

| | | |
|---|---|---|
| 1../8 | - | 1 |
| 2../8 | - | 1 |

| | | |
|---|---|---|
| 2../8 | - | 1 |
| 3../8 | - | 3 |

A sends update:

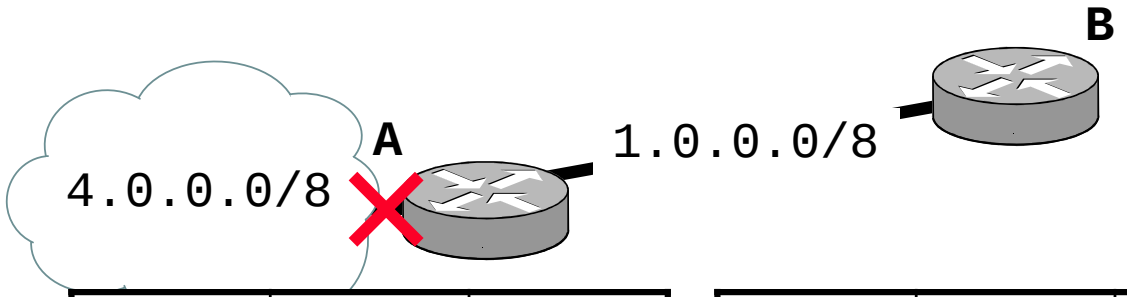
| | | |
|---|---|---|
| 1../8 | - | 1 |
| 2../8 | - | 1 |
| 3../8 | **A** | **3+1** |
| 4../8 | **A** | **1+1** |

| | | |
|---|---|---|
| 1../8 | **A** | **1+3** |
| 2../8 | - | 3 |
| 3../8 | - | 1 |
| 4../8 | **A** | **1+3** |

B sends update:

| | | |
|---|---|---|
| 1../8 | **B** | **1+1** |
| 2../8 | - | 3 |
| 3../8 | - | 1 |
| 4../8 | **B** | **2+1** |

# Counting to infinity




| | | |
|---|---|---|
| 1../8 | - | 1 |
| 2../8 | B | 2 |
| 3../8 | - | 3 |
| 4../8 | - | 1 |

| | | |
|---|---|---|
| 1../8 | - | 1 |
| 2../8 | - | 1 |
| 3../8 | A | 4 |
| 4../8 | A | 2 |

Line A/4 failure:

| | | |
|---|---|---|
| 4../8 | **-** | **16** |

B sends update:

| | | |
|---|---|---|
| 4../8 | **B** | **2+1** |

A sends update:

| | | |
|---|---|---|
| 4../8 | **A** | **3+1** |

...

After 14x30 sec:

| | | |
|---|---|---|
| 4../8 | **-** | **16** |

| | | |
|---|---|---|
| 4../8 | **-** | **16** |

# Link state protocols

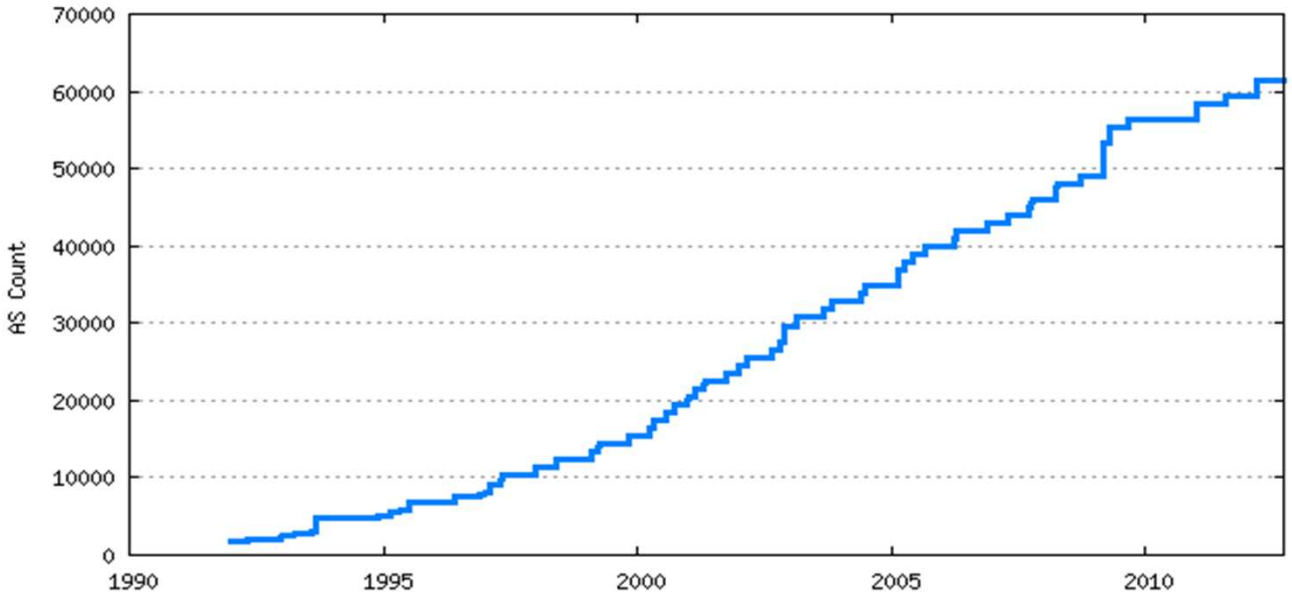
- Basic idea:
  - every router knows the entire network „map“
  - routers send neighbors state of all their links, every router uses this data for rebuilding the network map
- Disadvantages:
  - building map is CPU and memory consuming
  - during the start and in too unstable networks, the data exchanges can bring heavy network load
- Advantages:
  - flexible reaction to network topology changes
  - every router builds own map, error does not affect others
  - network can be divided to smaller areas (build speed!)
  - data exchange only in case of a failure

# Open Shortest Path First

- The most widespread link-state internal routing protocol
- Properties:
  - uses Dijkstra algorithm for the shortest path searching
  - uses a hierarchical network model:
    - area 0 is the backbone
    - other areas are connected only to the backbone
    - each router knows own area map and the path to the backbone
  - configurable metrics, by default it is *path cost*, the sum of „prices“ along the path (the price depends on the bandwidth)
- Uses own transport layer protocol (number 89) and multicast addresses 224.0.0.5 and 224.0.0.6
- Current version 2 for IPv4 (RFC 2328) and IPv6 revision marked as version 3 (RFC 5340)

# Autonomous systems

- Definition: network block with common routing policy
- Established in 1982: easier routing at the global level, using of *External Routing Protocols* (EGP)
- The most frequent EGP: Border Gateway Protocol (BGP)
- AS Identification: 16bit number, nowadays moving to 32bit
- In the Czech republic: starts with 2, hundreds today

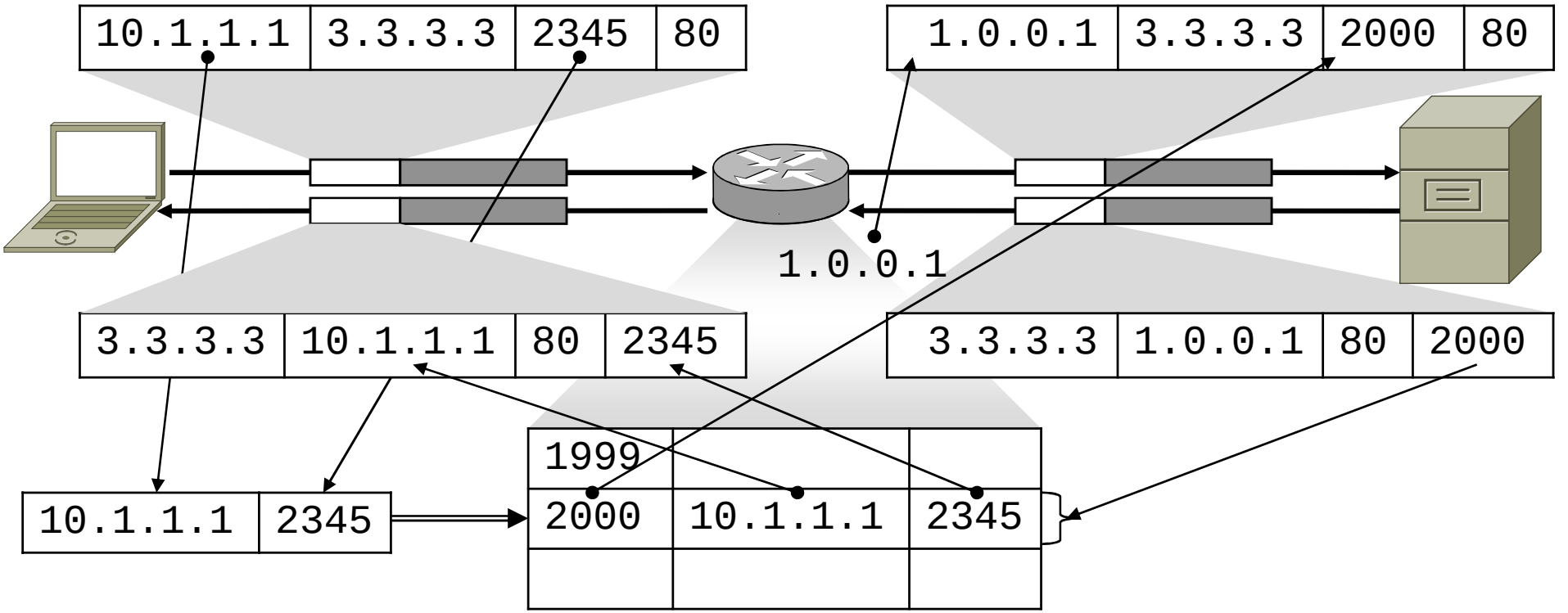


# IP filtering

- Router on the network perimeter can be configured which traffic is allowed to ingress and egress
- Strict configuration: selected out, nothing in
  - good for single-channel protocols (HTTP, SMTP)
  - poor for multiple-channel ones (FTP, SIP)
- Usual configuration: all out, nothing in
  - conflicts with e.g. FTP with active data transfer
  - unusable for protocols with many of channels (SIP)
- Better solution: configuring a router, which partially understands the application layer
- Problem with services „inside“ (e.g. www server, e-mail)
  - allowing exceptions is risky
  - separated network segment, demilitarized zone (DMZ)

# Network address translation (NAT)

- General principle: a local network with *private* addresses using for outside traffic *public* ones (or other private ones)
- Other term: *IP masquerading*
- Implementation and terminology can slightly differ in details



# Proxy server

- Transparent model:
  - SW on a **router** captures a connection, stores the request, makes its own connection to the target and sends the request.
  - The response is delivered to the router, stored (for further clients) and forwarded to the original requestor.
  - No configuration changes on client needed.
- Nontransparent model:
  - Clients need to be **configured**, to send requests to the local proxy instead of the target server (can be done automatically).
  - Proxy server need not to run on the router.
  - Only for protocols having the support for proxy usage.
- Important security and performance issue:
  - network administrator can effectively control user activities
  - outside traffic can be reasonably reduced

# Data link layer (OSI 2)

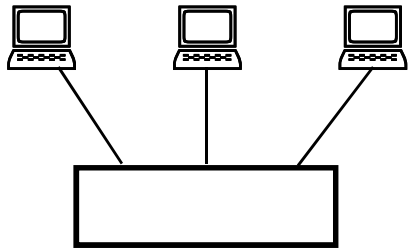
- Separated into two sublayers:
  - Logical Link Control (LLC) multiplexes various network layer protocols approaching a media
  - Media Access Control (MAC) controls addressing and media access: who, when and how may send and receive data
- TCP/IP does not deal with this layer („network interface“)
- Network segment (physical network):
  - set of nodes sharing the same media
- Data link layer PDU: frame
  - format depends on the media used
  - in general: synchronization field, header (addresses, type, control data), data field and trailer (error detection)

# Physical topology

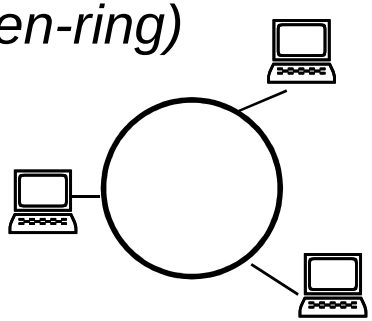
## Multipoint

*Bus (e.g. Ethernet)*

*Star (e.g. ATM)*

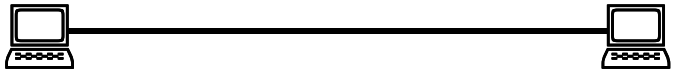


*Ring (e.g. FDDI, Token-ring)*

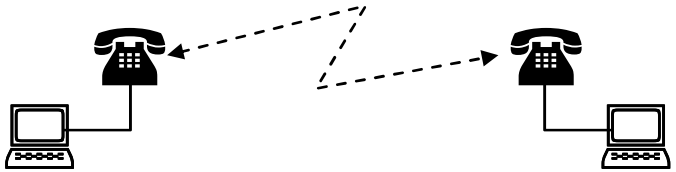


## Point-to-point

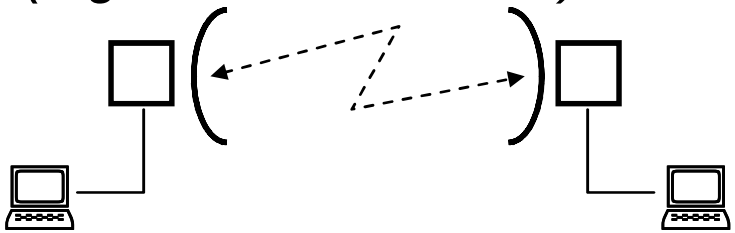
*Direct cabeling (e.g. serial, coaxial, UTP, fibre-optical)*



*Connection using modems*



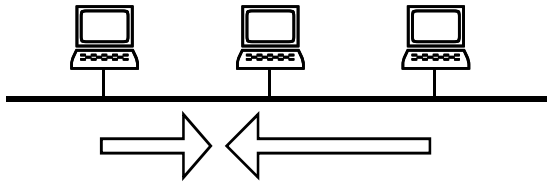
*Wireless (e.g. laser, microwave)*



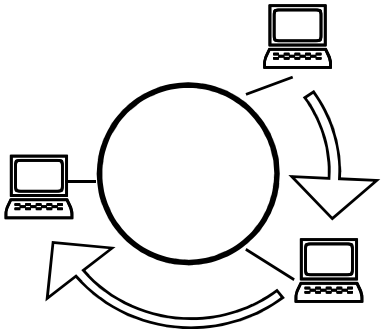
# Media access control types

## Multipoint

*Nondeterministic - collisions*

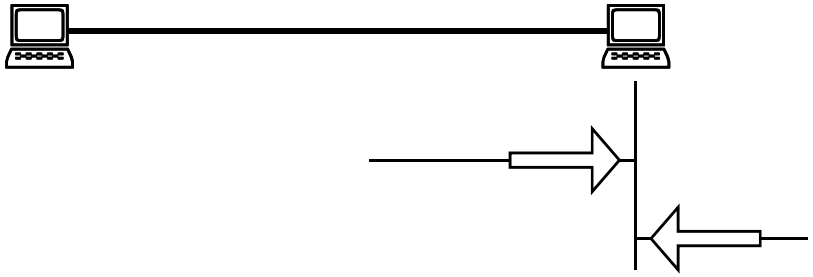


*Deterministic - overhead*

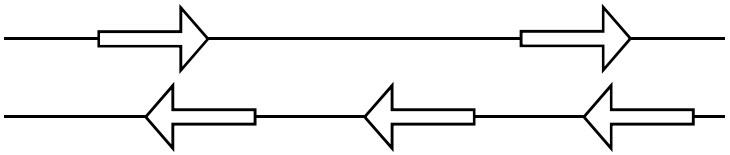


## Point-to-point

*Half duplex*




*Full duplex*



# Collision solution

- CSMA (Carrier Sense with Multiple Access)
  - a node listens carrier traffic, if not idle, waits
- CSMA/CD (Collision Detection), e.g. Ethernet
  - during the transmission checks a collision occurrence
  - if it occurs, the node stops the transmission, alerts other nodes, waits some (random) time and repeats the attempt, usually the period is increasing (*exponential waiting*)
  - frame transmission time < time to traverse the whole segment (*collision window*); segment length and min. frame size limited
- CSMA/CA (Collision Avoidance), e.g. WiFi
  - when the carrier is idle, the node sends the entire frame and waits for acknowledgement (ACK)
  - if the carrier is not idle, or the ACK does not come, the *exponential waiting* is started

# Ethernet

- History:
  - the first LAN attempts in Xerox company
  - standardization taken over by IEEE (Feb 1980 → IEEE 802)
  - two most common formats Ethernet II, IEEE 802.3
- Currently the top technology for LANs
  - can flexibly react to progressive HW evolution
  - can adapt to a wide range of transmission media
- Media access on multipoint links controlled by CSMA/CD
  - a „jam signal“ is sent by sender when a collision is detected
  - exponential waiting terminates after 16 attempts by an error
- Addresses:
  - 3 bytes prefix (producer, multicast...), 3 bytes node number
  - formerly „burned-in“ in NIC, nowadays programmable

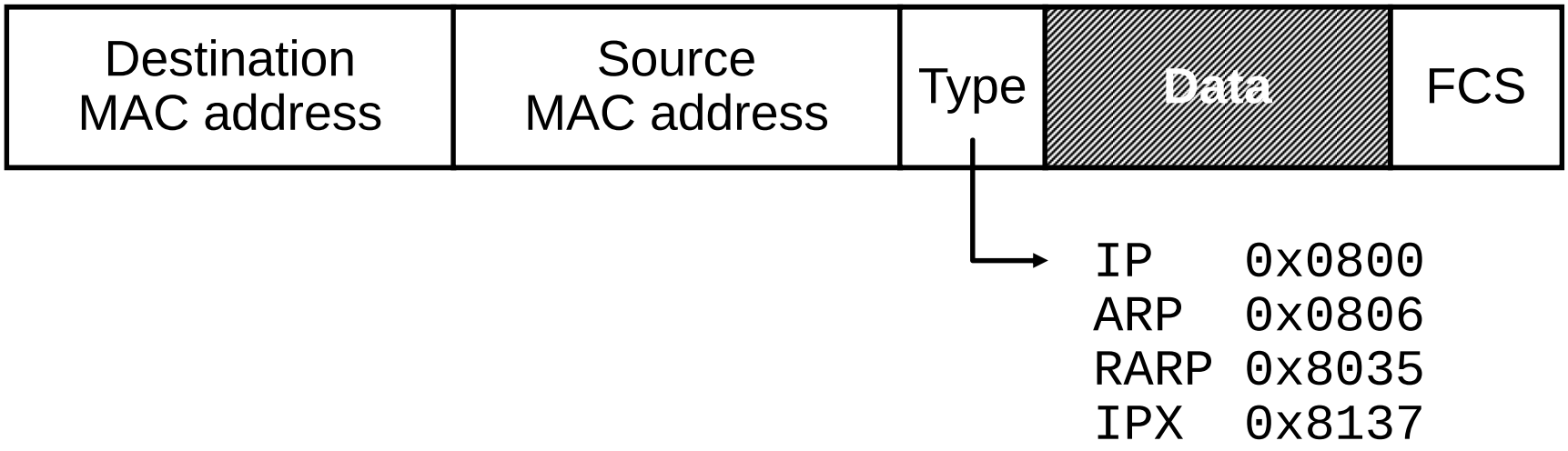
# IEEE 802.3 standards

| Standard | Year | Identification | Bandwidth | Media |
|---|---|---|---|---|
| 802.3 | 1983 | 10BASE5 | 10 Mbit/s | thick coaxial cable |
| 802.3a | 1985 | 10BASE2 | 10 Mbit/s | thin coaxial cable |
| 802.3i | 1990 | 10BASE-T | 10 Mbit/s | twisted pair (UTP) |
| 802.3j | 1993 | 10BASE-F | 10 Mbit/s | fiber optic |
| 802.3u | 1995 | 100BASE-TX,FX | 100 Mbit/s | UTP or fiber optic |
| 802.3z | 1998 | 1000BASE-X | 1 Gbit/s | fiber optic |
| 802.3ab | 1999 | 1000BASE-T | 1 Gbit/s | UTP |
| 802.3ae | 2003 | 10GBASE-SR,... | 10 Gbit/s | fiber optic |
| 802.3an | 2006 | 10GBASE-T | 10 Gbit/s | fiber optic |

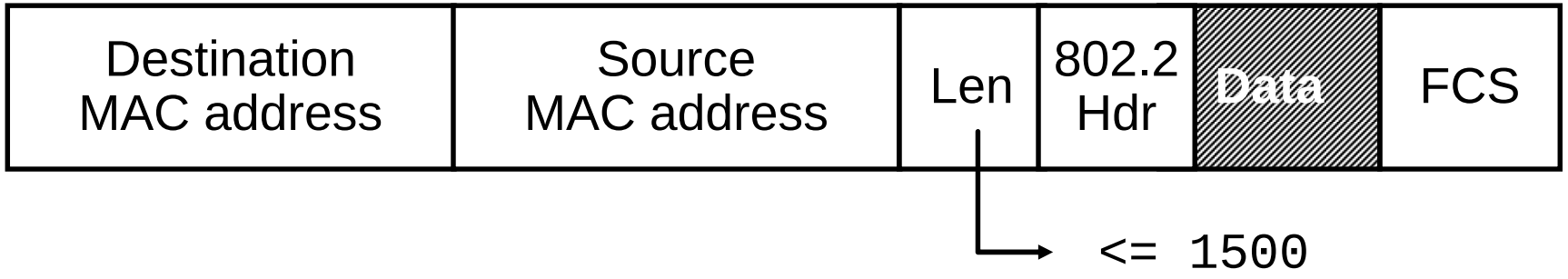
Unlike RFC, IEEE standards are licensed.

# Ethernet frame format

**Ethernet v2:**

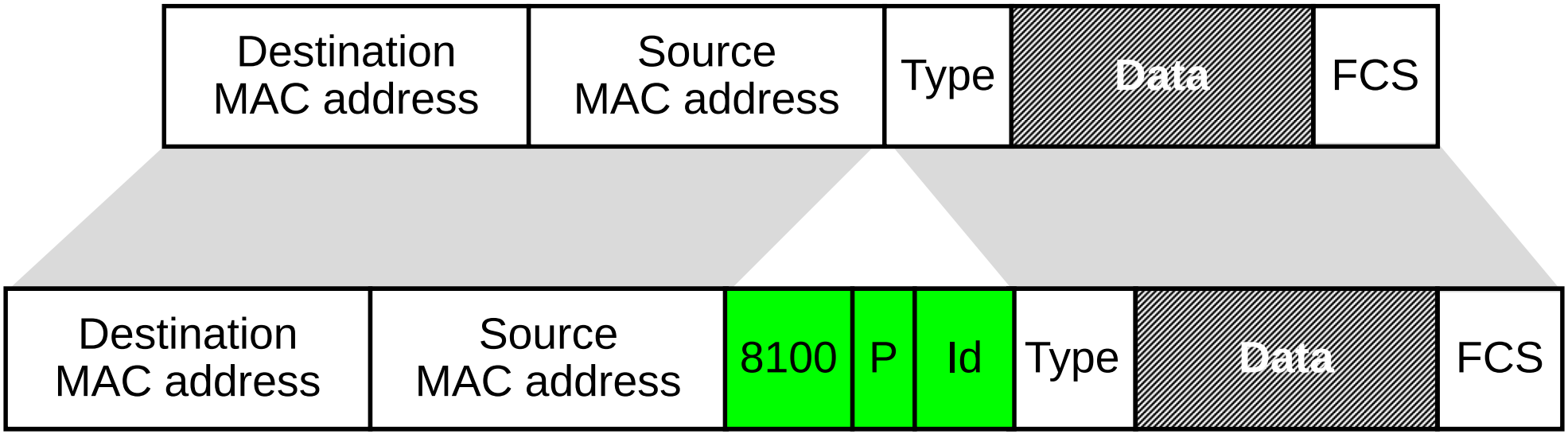



**IEEE 802.3**



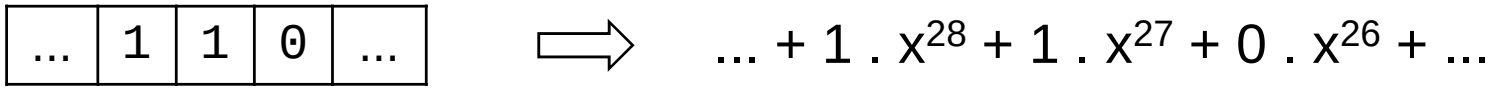
# Virtual networks (VLAN)

- VLANs allow to operate more independent local networks on a single physical infrastructure
- Networks marked by a 12bit identification (VLANID)
- Ethernet frame must be enlarged by a 32bit tag (tag protocol identifier 0x8100, QoS priority and VLANID)
- Tagging can be done by a switch, transparently for end nodes



# Cyclic Redundancy Check

- Hash function used for data integrity checks on many levels, e.g. the Frame Check Sequence (FCS) in the Ethernet
- Bit sequence is considered as a binary polynomial coefficients




- The polynomial is divided by so called *characteristic polynomial* (e.g. for CRC-16 it is $x^{16} + x^{15} + x^2 + 1$)
- The remainder is converted back to bits and used as a hash
- Simple implementation (also pure HW solutions)
- Big strength, *n*-bit CRC can detect:
  - 100% of errors with odd number of bits, or shorter than *n* bits
  - longer errors with high probability, too

# Wi-Fi

- Wireless network, another name: WLAN (wireless LAN)
- Many various models commonly called IEEE 802.11 (802.11a, b, g, n, y,...):
  - various frequencies (2,4 to 5 GHz)
  - various speeds (2 to 600 Mbps)
- WiFi devices embedded to almost all communication tools
- Network structure:
  - ad-hoc peer-to-peer network
  - infrastructure of access points (AP)
- SSID (Service Set ID): string (up to 32 characters) for network distinguishing
- Problem: security!

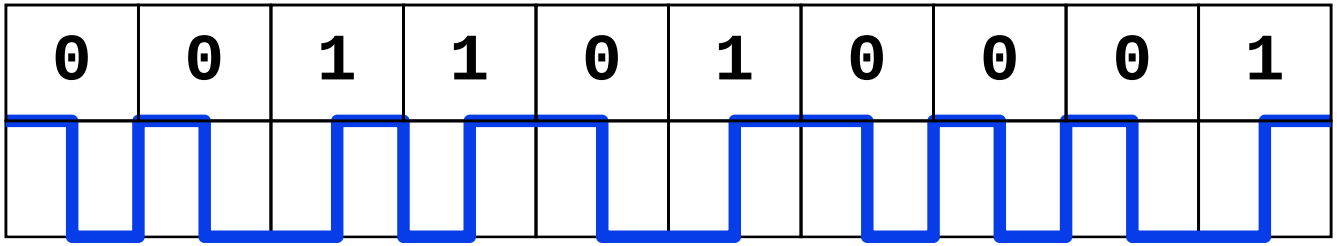
# Physical layer (OSI 1)

- Layer function:
  - data transfer over physical media
  - conversion from digital data to analogue signal and v.v.

- Various media types
  - metallic: electric pulses
  - optical: light pulses
  - wireless: wave modulation

# Data transfer modes

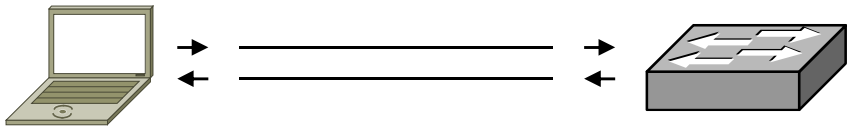
- Analogue vs. digital
  - in fact, everything is analogue (e.g. electric current)
  - digital: thresholds exist that decide whether a signal level belongs to proper interval (less impact of noise)
  - convertors: A→D (and back) *codec* (coder/decoder) D→A *modem* (modulator/demodulator),
- Baseband vs. broadband
  - baseband carries directly encoded signal itself, the Ethernet uses so called Manchester:



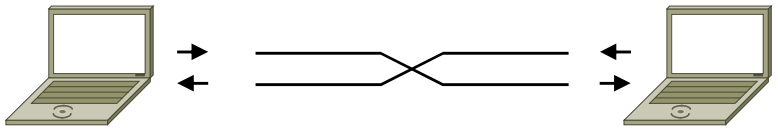

  - broadband carries the basic signal and modulates it (phase, amplitude, frequence)

# Unshielded twisted pair (UTP)

- The most common structured cabling media nowadays
- 4 pairs of copper conductors twisted around each other
  - twisting lowers both emission and reception of electromagnetic radiation (lower interference)
- 100Mb Ethernet uses only 2 pairs (cable can be “divided”)
- Connectors: RJ 45
- Cabling must reflect the nature of devices
  - nowadays usually the MDI/MDIX autodetection available



straight cable

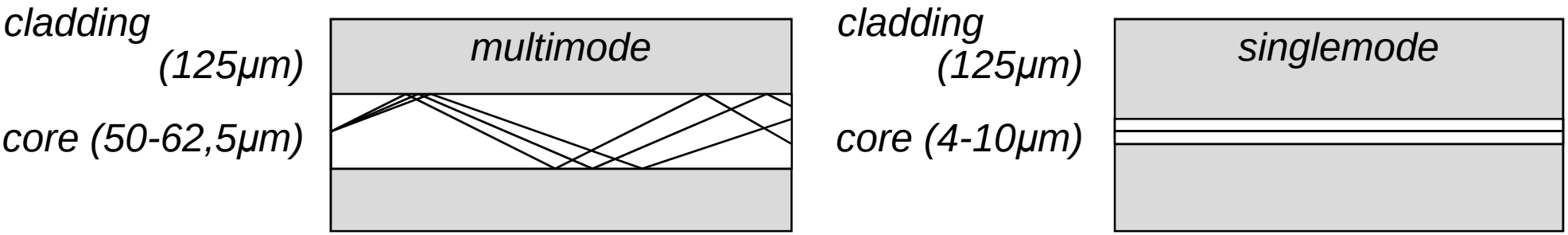


crossover

- Option: cable with metallic shielding (STP)

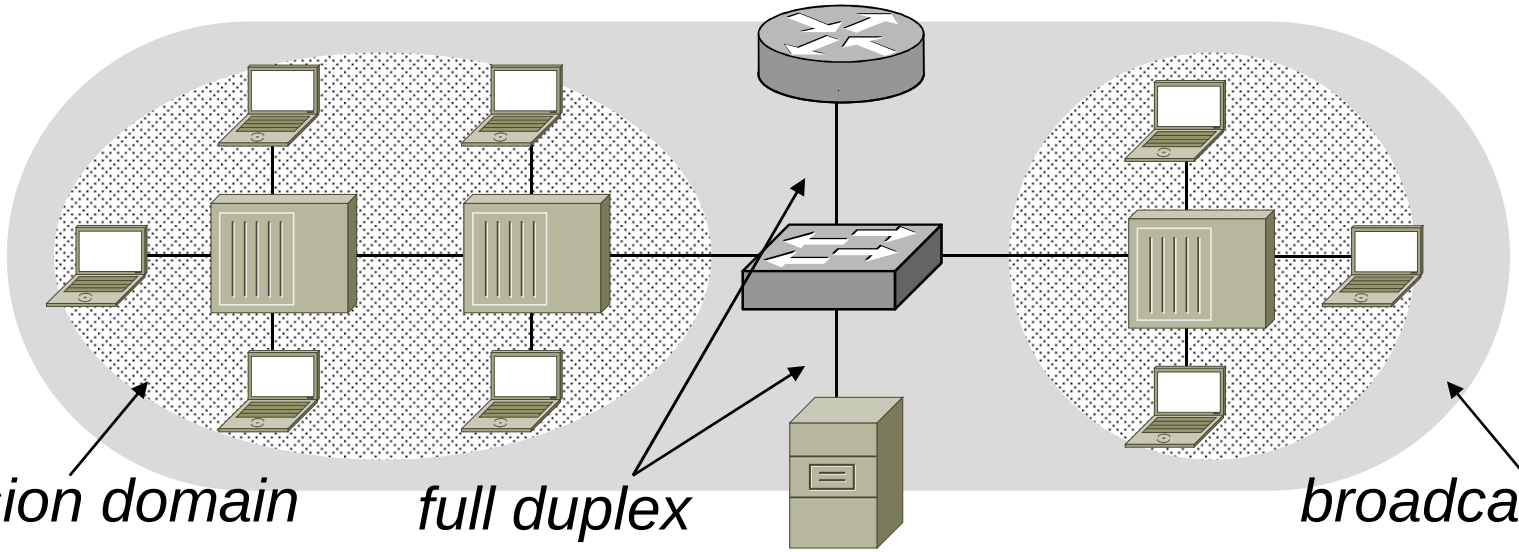
# Optical fiber

- Signal is carried as visible light through a fiber of $SiO_2$
  - high frequency, large bandwidth
  - low attenuation, no interference
- Disadvantages:
  - higher price, demanding installation, don't look into cable
- Fiber types:
  - singlemode fiber - light source: laser =>
    higher radius, bandwidth („speed", not speed), price
  - multimode fiber - light source: LED

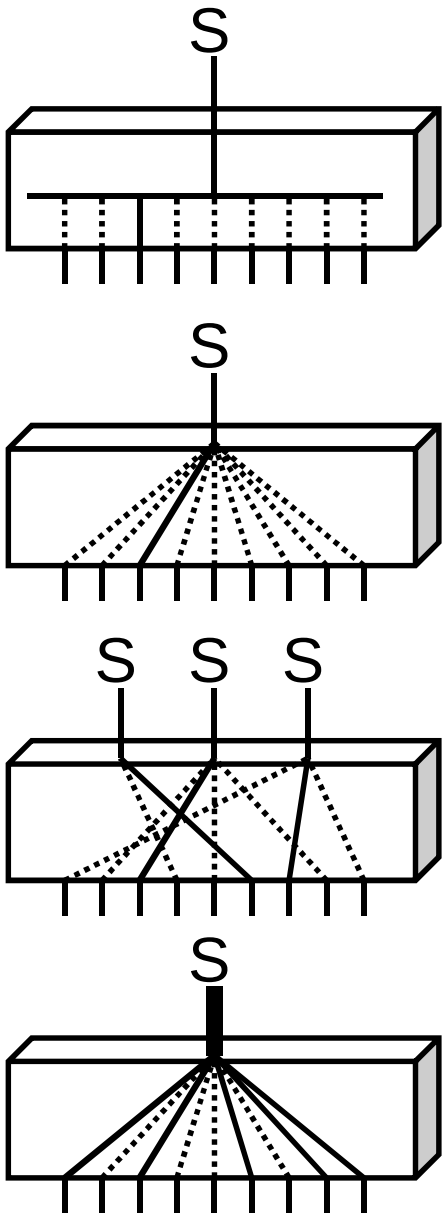


# Network devices (repeater, bridge)

- Repeaters connect segments on the physical layer
  - solves: larger radius (eliminates cable attenuation)
  - does not solve: throughput (collision probability increases)
  - terminology of structured cabling: *hub*
- Bridges connect segments on the data link layer
  - solves: larger throughput (by splitting the collision domain)
  - terminology of structured cabling: *switch*

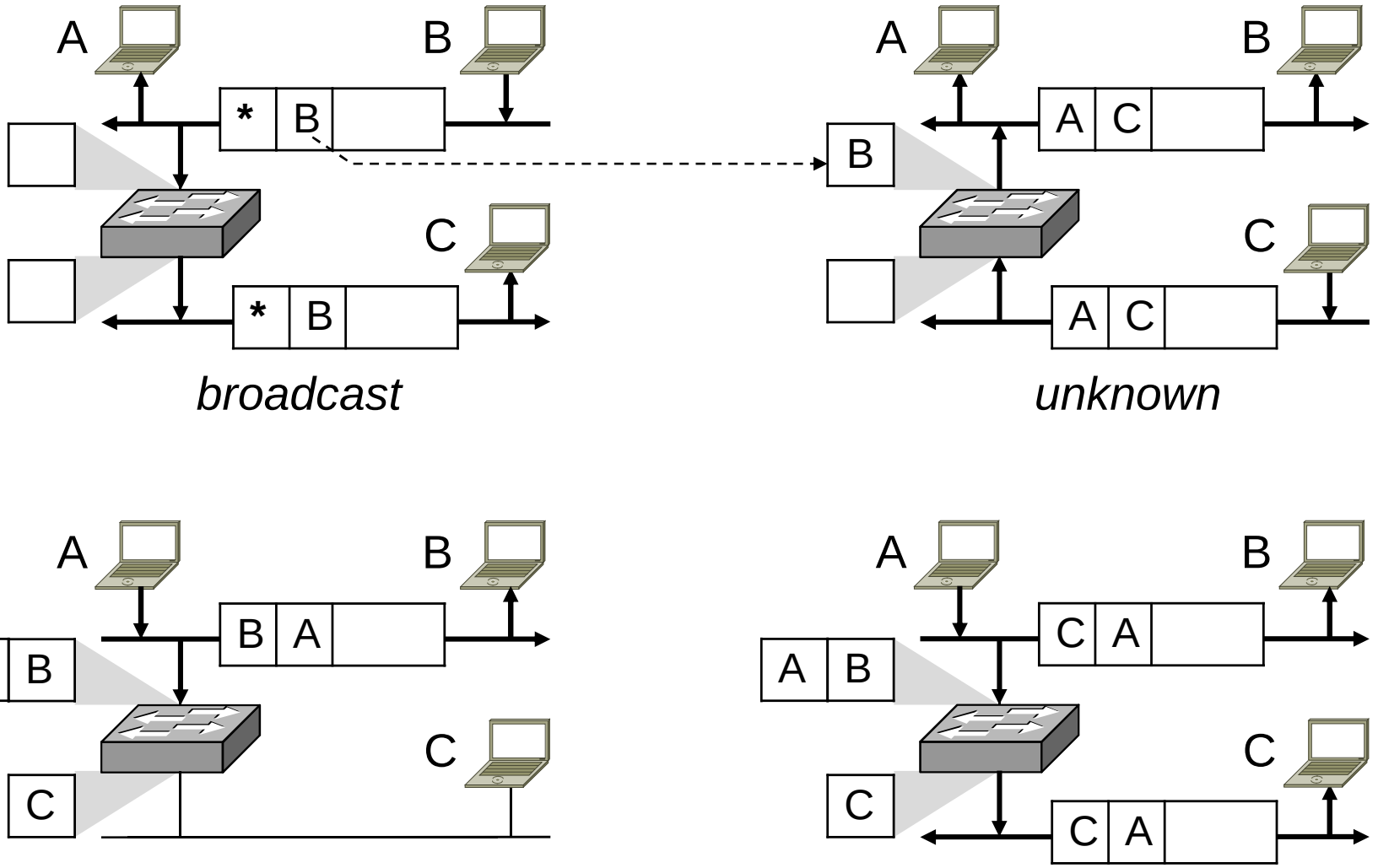


# Hub vs. switch comparison



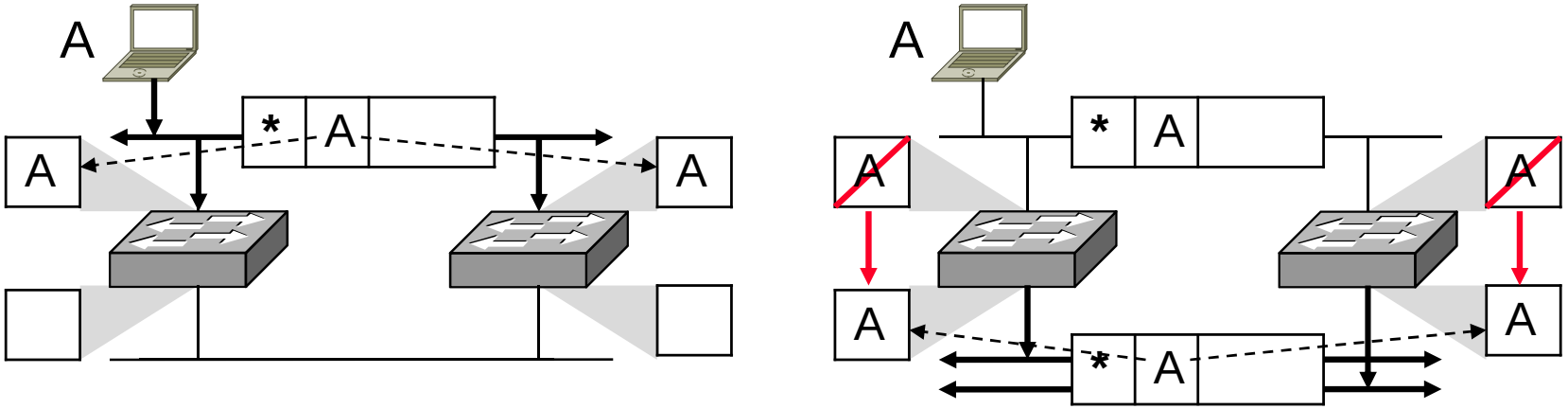

- HUB
  Σ 10 Mbit/s
- Switch
  Σ 10 Mbit/s
- Switch, more servers
  Σ > 10 Mbit/s
- Switch with uplink
  Σ up to 100 Mbit/s

# Learning bridge



# Spanning Tree Algorithm

- Motivation: if the network contains a backup switch, learning does not work and the network is flooded by forwarded frames




- Reason: the graph contains a loop
- Solution: to find an acyclic subset, spanning tree
- Switches must agree, which acts as backup one (forwarding no data, only monitoring status)
- STA consumes some time, switch ports start is slow
  - usually, the STA can be suspended („faststart“), use carefully

# The End