- Proces předání zprávy
  - poštou → list papíru - datová jednotka
    - → vložení do obálky - zapouzdření / encapsulation
    - → napsání adresy + nalepení známky - dodržování protokolu
    - → hození do schránky - odeslání
    - → pošta doručí zprávu - skrz uzly propojené různými médii
    - → obdrží ji příjemce - cílový uzel
    - → příjemce otevře obálku - decapsulation
    - → vytáhne zprávu - zpráva doručena
  - emailem → zabalování a zapouzdření podle správného protokolu ⇒ datová jednotka
    - → po místní síti doručíme zprávu síťové infrastruktuře
    - → ta ji nakonec doručí až na koncový uzel (počítač příjemce)
    - → příjemce ji rozbalí, dekóduje a zobrazí na displeji
- Odolnost
  - → americká armáda řešila problém výpadku telefonních sítí v případě útoku
  - → telekomunikační sítě - přepojování okruhů
    - → volání: síť najde posloupnost uzlů (okruh), kterým spojí obě strany
    - ⇒ rychlé, plynulé, ale při výpadu uzlu se spojení rozpadne
    - → chtěli síť odolnou vůči chybám
  - → přepojování paketů
    - → data rozdělíme na malé bloky - pakety - a každý paket si najde vlastní cestu k cílovému uzlu
    - → pokud je některý uzel napaden, tak si paket najde jinou cestu
    - ⇒ pomalejší, proměnlivá doba přenosu, spolehlivé
- Bezpečnost
  - → na počátku internetu se nikdo nebál softwarového útoku
    - ⇒ tehdejší protokoly nepovolovaly šifrování a důvěřovaly odesílateli i obsahu dat
  - → bálí se fyzického útoku na infrastrukturu - uzly a kabely - bombou

- bezpečnostní rizika
  - fyzické napadení infrastruktury
  - útok na data - neoprávněná manipulace
  - DoS (Denial of Service) - zahlcení zdroje dat - nemůže jet komunikovat s uživateli
  - DDoS (Distributed DoS) - útočník využije cizí servery, aby zvýšený provoz generovaly neviditelný místo něj

- Rozšiřitelnost - přidání počítače / sítě musí být snadné
  - LAN (Local Area Network)
    - Core vrstva - je připojená na infrastrukturu ISP (Internet Service Providera)
      - router = uzel propojující různé sítě
      - místní router je připojený k routeru ISP
      - několik hlavních switchů propojujících hlavní router se zbytkem LAN
    - Distribuční vrstva - vertikální
      - distribuuje konektivitu do všech částí budovy
    - Access vrstva - horizontální
      - umožňuje připojení koncovým zařízením
  - WAN (Wide Area Network) - opět rozdělení na 3 vrstvy
    - Tier 1 - klíčoví hráči internetu připojení přímo na páteř internetu
    - Tier 2 - společnosti (národní operátoři), jejímiž zákazníky jsou jiní ISP
    - Tier 3 - ISP, kteří připojují koncové zákazníky - společnosti, domácnosti s LAN

- Kvalita služeb
  - → přenosové parametry sítě
    - Latence - zpoždění = doba doručení
    - Jitter = rozptyl zpoždění - pravidelnost doručování
    - Ztrátovost - jak často dochází k tomu, že nějaký packet není doručen
    - Šířka pásma - rychlost - kolik dat lze zařídorat a přenášet = bandwith
  - → různé aplikace mají různé požadavky
    - multimediální aplikace - pravidelnost doručení
    - přenosy dat (email) - nízká ztrátovost dat

- Kvalita služeb
  - cíl – garance vymezeného toku pro konkrétní typ provozu
    – garance rychlejšího doručení prioritních zpráv
  - implementace
    – data obsahují klasifikaci QoS (Quality of Service)
    - strategie garance kvality
      – část kanálu vyhradíme jen pro prioritní zprávy
      → zaručená kvalita, plýtvání kapacitou
    - strategie best effort
      – u každého uzlu je prioritní fronta
      → efektivní využití média, není záruka kvality

- Vznik počítačových sítí

1) oddělené počítače – přenos dat na děrných štítcích
2) počítače a terminály – point to point komunikace
3) vznik LANek, které dohromady utvořily WAN
   → vznik klient-server aplikací, kde je část výkonu vykonává klient

- Základní dělení sítí
  - LAN – sdílení zdrojů na malé vzdálenosti – jednotné vlastnictví a řízení
  - WAN – globální sítě, vzdálený přístup, mnoho vlastníků – nepoužívá ji ten, kdo ji vybudoval
  → rozdíly se dnes stírají, ale sítě LAN bývají soukromé

- Veřejné a privátní sítě
  → když chceme propojit dvě LAN sítě přes veřejnou síť → problém s bezpečností
  ⇒ VPN (Virtual Private Network) – na hranici obou privátních sítí dáme zařízení, které enkóduje / dekóduje zprávy ⇒ vytvoří VPN tunel skrz veřejnou síť
  → pro počítače v obou částech LAN se celá síť tváří jako jedna LAN
  - případně může být jedna strana tunelu nahrazena SW na počítači

- RFC (Request For Comments)
  → standardizace internetu – veřejné
  → obsah dokumentů se nemění
  → ne všichni RFC vždy dodržují

- **Síťový model** – popisuje vrstvy, jejich strukturu a úkoly – OSI model
- **Síťová architektura** – model + konkrétní služby, technologie, protokoly,... TCP/IP
- **OSI (Open Systems Interconnection)**
  - model – vhodný pro dokumentaci a výuku
  - protokoly – nepraktické, budované shora

  1) fyzická – fyzický přenos bitů mezi uzly – hub, repeater
  2) linková – přenos datových rámců mezi sousedními uzly – switch
  3) síťová – směrování mezi sítěmi / mezi vzdálenými uzly – router
     ↳ přenos datových bloků s proměnlivou, ale omezenou délkou – paketů
  4) transportní – přenos dat neomezené délky mezi aplikacemi
     – segmentace příliš velkých bloků
  5) relační – řídí dialog mezi aplikacemi
  6) prezentační – datové konverze pro aplikace
  7) aplikační – komunikace mezi programy, interakce mezi uživatelem a aplikací

→ X.400 (Message Handeling System) = OSI pošta, komplikovaná, ale jednoznačná adresace
→ X.500 (Directory Access Protocol) – adresářové služby, telefonní seznam (adresace jako x400
→ následovníci:
- X.509 (Public Key Infrastructure) – správa veřejných klíčů
- LDAP (Lightweight DAP) – databáze informací o uživatelích a službách

• Rodina protokolů TCP/IP

- návrh odspodu, praktické → od jednoduchých ke složitým

← příklady

OSI	VRSTVA	PROTOKOLY		
7 6 5	aplikační	FTP HTTP SMTP	DNS SIP	NFS XDR RPC
4	transportní	TCP		UDP
3	síťová	IPv4 / IPv6		
2 1	síťové rozhraní	Ethernet, Wifi, ATM FDDI, SLIP, PPP, ...		

ICMP / ARP → stojí mimo hierarchii

← většinou je lepší definovat všechno v aplikační vrstvě, ale jsou výjimky → NFS + XDR + RPC

← Internet Protocol (vrstva 3)

← protokol podle média (vrstvy 2, 1)

• TCP (Transaction Control Protocol)

- pro spojované služby - telefonní spojení
- zaručeno spolehlivé / reliable doručení dat
  - → TCP data segmentuje na packety, potvrzuje úspěšné doručení, případně je posílá znovu
  - → implementace TCP je složitá, ale aplikace je jednodušší
  - → aplikace nemůže řídit komunikaci

• UDP (User Datagram Protocol)

- pro nespojované služby - pošta
- není zaručeno doručení ani pořadí paketů → unreliable = nespolehlivé
  - ↗ IP je také unreliable
- → kontrolu musí provádět aplikace → může řídit komunikaci
- → UDP je jednoduché, ale aplikace složitá

• Aplikační modely

• klient - server

- klient zná pevnou adresu serveru
- klient navazuje komunikaci, zadává požadavky
- server obvykle obsluhuje více klientů
- download (S → K), upload (K → S)
- např. DNS, WWW, SMTP

• peer-2-peer - P2P

- partneři neznají pevné adresy zdroje dat
- nejsou vyhraněné role → každý je zároveň klient i server
- BitTorrent

špatná pověst aplikací
→ někdo tam uploadne nelegální data
→ kdokoli si je může downloadovat
→ nelegální distribuce

• Adresování počítačů

• MAC adresa – HW, linková vrstva, ethernet: 8:0:20:AE:6:1F (6 B)
(Media Access Control)
– nerespektuje topologii sítě
– současné síťové karty mají MAC uloženou v paměti ⇒ lze sfalšovat

• IP adresa – SW, síťová vrstva, IPv4: 195.113.19.71 (4 B)
– přidělována počítači podle topologie sítě
– určuje jednoznačně síť a v jejím rámci počítač

• Doménová adresa – lidé, aplikační vrstva, www.mff.cuni.cz
– přidělení podle organizační struktury, hierarchie ←

• DNS (Domain Name System) IP ⇔ Domain

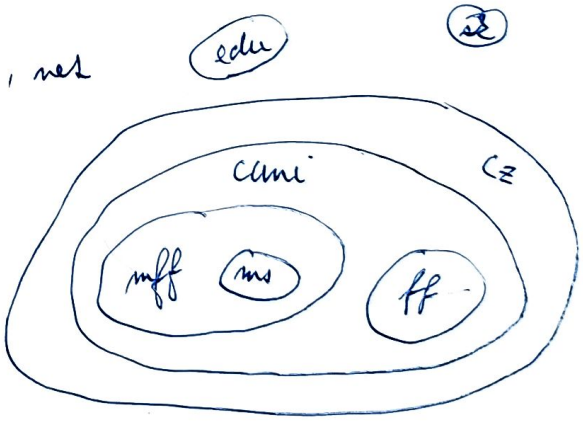
• ARP (Address Resolution Protocol) IP ⇔ MAC

• DNS

– hierarchická struktura zón, jež obsahují info o podřízených počítačích a zónách
– tyto informace jsou uložené v databázi nameserverů – poskytují klientům odpovědi
– každý počítač by měl mít přiřazené doménové jméno – ale může mít i více jmen

• správa domén (↳ podle domény, jejíž služby na něm běží)

• TLD (Top Level Domain) – spravuje ICANN – původně přísné podmínky
→ technické – arpa
→ rezortní pro USA – com, org, edu, mil, gov, net
→ ISO kódy zemí – cz, sk, ..., uk, eu



• SLD (Second Level Domain) – spravuje jí majitel
→ doména cz spravuje CZ.NIC
↳ cokoli pod cz je SLD
→ hierarchická správa domén

⇒ TLD spravuje centrální organizace a pak je ve správě celá hierarchie

- **IP adresy**
  - každý koncový uzel v síti TCP/IP musí mít IP adresu
  - IPv4 : 4 byty → 195.113.19.71
  - IPv6 : 16 bytů → 2001:718:1c03:a01::1 ← nejvíce nulových bloků nahrazujeme :: ← hex po 2B
  - → veřejné adresy (které mohou komunikovat se zbytkem světa) přidělují ISP
  - → soukromé adresy (které lze používat jen v rámci LAN) přidělují správce LAN
  - ↑ přiřazení adresních bloků síti – prefix
  - → přiřazení adresy počítači v síti
    - → statické – každý uzel má vždy stejnou adresu
    - → dynamické – adresa je přiřazena na vyžádání
    - → volné – k síti se může připojit kdokoli
    - → omezené – pro připojení je třeba se autentizovat

    } správa LAN

    - → platí i pro privátní adresy
    - ↳ výjimka: link-local adresa – počítač si určuje vlastní adresu v rámci segmentu sítě ve které je

- **Port** – 16 bit. číslo identifikující jeden konec spojení – aplikaci, která chce přijímat pakety
  - → destination-port musí klient znát → well-known services
  - → source-port přiděluje operační systém z neobsazených čísel portů
  - → spojení v TCP/IP: <zdrojová IP, zdrojový port, cílová IP, cílový port, TCP/UDP>
    - dva různé kanály stejné aplikace se musí lišit alespoň ve zdrojovém portu
    - stejná čísla portů lze použít pro 2 různé kom. kanály, pokud používají různé protokoly
  - → příklady well-known services
    - 21/TCP – FTP (File Transfer Protocol) – přenos souborů
    - 22/TCP – SSH (secure Shell) – vzdálené přihlášení a přenos souborů
    - 25/TCP – SMTP (Simple Mail Transfer Protocol) – přenos elektronické pošty
    - 53/* – DNS (Domain Name System) – překlady mezi doménami a IP
    - 80/TCP – HTTP (Hyper Text Transfer Protocol) – přenos webových stránek
      443/TCP – HTTPS – šifrované HTTP

- **Socket** – 1 konec komunikačního kanálu mezi klientem a serverem
  - adresa 1 konce kanálu ⇒ <IP adresa, port>

- **NAT (Network Address Translation)**
  - → princip, kdy lokální síť používá privátní adresy a ven se představuje nějakou veřejnou adresou = IP masquerading
  - → implementace i terminologie se v detailech liší
  - → princip: když chce klient z LAN něco poslat ven, tak router na perimetru LAN upraví obsah paketu tak, aby server odpověděl jemu
    - ⇒ router si uloží socket klienta – když přijde odpověď, tak mu ji přepošle

- **Adresování služeb**
  - → URI (Uniform Resource Identifier) – jednotný systém odkazů
    - historicky URL (umístění) a URN (název) dnes URI ~ URL

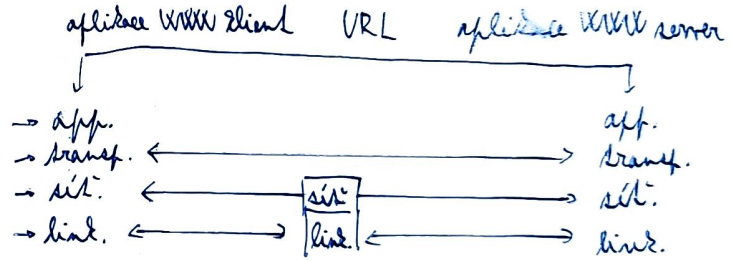
URI = schéma : [//] autorita [cesta] [?dotaz] [#fragment] [ ] = volitelné
(cesta – jako v file systému; fragment – identifikátor bodu na stránce (anchor))

autorita = [jméno [:heslo] @] adresa [:port]
(jméno, heslo – info o účastníkovi; adresa – doména; port – pokud server nepoužívá well-known port)

ftp://sunsite.mff.cuni.cz/Net/RFC

http://1.2.3.4:8080/q?ID=123#Local (doména, port)

mailto:ford@cuni.cz



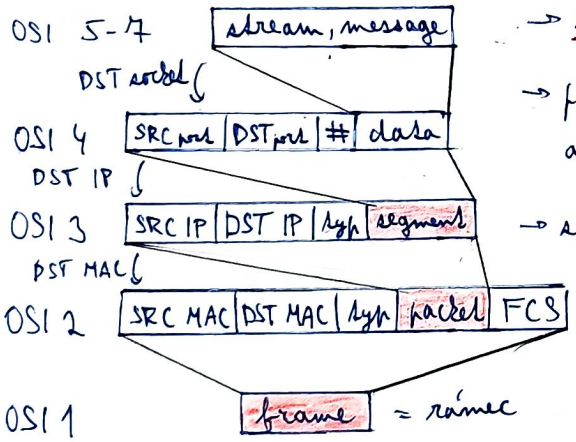
Klient ↔ switch ↔ router ↔ switch ↔ server

- **Datový tok v TCP/IP**
  - **aplikační vrstva** – klient adresuje server pomocí URL
    - → aplikační vrstva předává data spolu s cílovým socketem transportní vrstvě
  - **transportní vrstva** – identifikace obou konců kom. kanálu sockety
    - → předává data a cílovou adresu síťové vrstvě
  - **síťová vrstva** – identifikace IP adresami
    - → předává data a MAC adresu next-hop uzlu (cílový server – pokud je ve stejné síti; router – přes který se dostane do cílové sítě)
  - **linková vrstva** – identifikace MAC adresami
    - → fyzická vrstva doručí data na next-hop uzel, tak buď to je cílový server, nebo data předáme vyšším vrstvám. Nebo router → data dostane síťová vrstva → hop...

- **Multiplexing** – několik kom. kanálů v určité vrstvě používá stejný kanál v nižší vrstvě
- **Protokol** – obě strany musí dodržovat protokol.
  → na 1 vrstvě může probíhat komunikace ve více protokolech současně

- **Encapsulation**
  - mějme na vrstvě n protokol, který definuje **PDU** (Protocol Data Unit) této vrstvy.
  - SW n-té vrstvy přidá k PDU_n řídící informace a zavolá službu nižší vrstvy
  - tato funkce poté vrací n-té vrstvě info o selhání / úspěchu n-1 ní vrstvy
    ⇒ této výměně dat mezi vrstvami se narýná **interface** a jejímu formátu (PDU_n + příkazy) se říká **IDU** (Interface DU)
  - vrstva n-1 přirozeně řídící informace a sestaví PDU_{n-1} jako PDU_n a header s řídícími informacemi ← **zapouzdření**
  - header musí obsahovat identifikátor vrstvy n → příjemce provede decapsulation a demultiplexing a předá data vrstvě n.

- **Typy PDU v TCP/IP**



→ **zpráva** (nespojovaná app), **proud** (spojovaná app)

→ pokud aplikace využívá TCP, tak blok dat segmentuje a připraví header s # offsetem dat v rámci proudu

→ síťová vrstva k segmentu přidá header s číslem protokolu transportní vrstvy ⇒ **packet**

→ linková vrstva k packetu přidá header s číslem protokolu na síťové vrstvě a footer obsahující FCS (Frame Check Seq.), což je číslo vypočítané ze zbytku **framu / rámce**

→ v cílovém uzlu fyzická vrstva data dekóduje a předá linkové vrstvě
→ linková vrstva přepočítá FCS a zkontroluje, jestli sedí (pokud ne, obsah se změnil)
↳ zkontroluje cílovou MAC a podle čísla protokolu předá síťové vrstvě rozbalený packet
→ síťová vrstva zkontroluje IP a předá segment příslušnému SW transp. vrstvy
→ rozsah práce transportní vrstvy závisí na použitém protokolu
- UDP – zpráva je předána aplikaci
- TCP – segment je uložen a celý datový blok bude aplikaci předán až po přijetí všech segmentů

- **Bezpečnost**

- **Autentizace** = proces ověření totožnosti subjektu (uživatele) ← kdo jsem?
- **Autorizace** = proces, kterým autorita (server) přiřazuje identifikovanému subj. oprávnění

→ **lokálně lze autentikovat pomocí:**

- **znalostí** – heslo, PIN, ... ← co vím?
  - ⊕ snadná implementace, jednoduché sdílení
  - ⊖ důvěrný fakt může být vyzrazen
- **technických prostředků** – HW token, elektronický klíč ← co mám?
  - ⊕ je to bezpečnější
- **biometrie** – otisk prstu, sken sítnice
  - ⊕ nejbezpečnější

→ **vzdálená autentizace**

→ problém: kanál může být odposloucháván ⇒ vyzrazení hesla

⇒ **OTP (One Time Password)**

→ nebo pomocí kryptografie vytvořit bezpečný kanál

→ problém: většina protokolů nemá built-in ověření totožnosti

⇒ využívají framework **SASL** (Simple Authentication and Security Layer), který dokáže do většiny důležitých protokolů včlenit různé metody ověřování

→ pro každý protokol ∃ **profil**, který určuje, jak autentizaci v daném protokolu užívat

→ možnost využití autentizačního serveru, který používá speciální protokol na komunikaci s klientem a serverem

⇒ server = poskytovatel služby, a. server = poskytovatel identity

→ a. protokoly: LDAP, RADIUS, NTLM, Kerberos, SAML

- **OTP** = mechanismy umožňující nevyužívání plain-textové autentizaci uživatele

1) **historický** → vytištěný seznam jednorázových hesel
2) **challenge-response** → server pošle uživateli na klienta jedinečný kód a uživatel jej pomocí nějaké kalkulačky zkombinuje se svým heslem → odpověď
3) **HW tokeny** → uživatel dostane speciální autentizační zařízení, které je sesynchronizované se serverem a generují kódy pro identifikaci
   → platnost kódu je několik vteřin a 1 použití

- **Kryptografie**
  → velmi důležitá – pro šifrování a el. podpis se využívají 3 základní typy algoritmů

  - **symetrické šifrování**
    - historie: aditivní, transpoziční, substituční šifry, šifrovací mřížky, ...
    - dnes: metody založené na matematické teorii
    → pro šifrování i dešifrování se používá stejný klíč
    → příklady: DES, Blowfish, AES, RC4
    ⊕ rychlé, vhodné pro velká data
    ⊖ partneři si předem musí nějak bezpečně předat klíč

- **asymetrické šifrování**
  → pro šifrování a dešifrování se používá pár navzájem neodvoditelných klíčů
  → odesílatel veřejným klíčem zprávu zašifruje a příjemce ji tajným dešifruje
  → matematický základ – **jednocestné funkce**
    - násobení $x = a \cdot b$ ✗ rozklad na prvočinitele
    - diskrétní logaritmus $y = p^k \bmod q$ → $k = ?$

  ↳ lze šifrovat tajným a dešifrovat veřejným

  → příklady: RSA, DSA, ECDSA
  ⊕ veřejný klíč lze šířit, tajný uschovat ⇒ není problém sdíleného tajemství
  ⊖ pomalé, vhodné jen pro malá data
  → veřejný klíč je třeba pečlivě ověřovat !

- **hashovací funkce**
  → vytvoří pevný kód z daného textu
  → široké uplatnění → kontroly shod, hashovací tabulky → CRC, MD5
  → v kryptografii:
  → malá změna textu = velká změna hashe ← silně jednocestné hashování
  → jednocestnost ⇒ text je z hashe neodvoditelný
  → nalezení textu se stejným hashem je obtížné → SHA

- **Šifrování dat** – sym + asym
  - → text zašifrujeme symetrickou šifrou a její klíč zašifrujeme veřejným klíčem příjemce a toto celé mu odešleme
  - → příjemce svým tajným klíčem dešifruje klíč a pomocí něj i správu

- **Elektronický podpis** – asym + hash → je jedno jestli je nebo není šifrovaný
  - → odesílatel zvolí hashovací funkci, vezme text a vypočítá jeho hash
    - ↳ ten hash zašifruje svým soukromým klíčem
      - ⇒ text, zašifrovaný hash a hashovací fci pak odešle příjemci
  - → příjemce si sám spočítá hash přijatého textu a dešifruje přijatý hash veřejným klíčem odesílatele. Pokud oba hashe sedí, tak:
    1) nikdo nemanipuloval s textem ← vyšel by jiný hash
    2) správu skutečně odeslal někdo s přístupem k soukromému klíči odesílatele

    ✓ ne nutně odesílatel

- **Diffie-Hellmanův algoritmus** — např. symetrický klíč
  - → způsob výměny informací nezabezpečeným kanálem, aby oba získali sdílené tajemství
  - → používá diskrétní logaritmy
    1) A vygeneruje tajné číslo a a veřejná (prvo) čísla p, q
    2) A spočítá číslo $A = p^a \bmod q$ a pošle $p, q, A \to B$
    3) B zvolí tajné číslo b, spočte $B = p^b \bmod q$ a pošle $B \to A$
    4) A spočítá $s = B^a \bmod q$ a B také spočítá $s = A^b \bmod q$
  - → princip: $B^a = (p^b)^a = p^{ba} = p^{ab} = (p^a)^b = A^b$
    - bez znalosti a, b a při volbě velkých p, q je spočítání s neřešitelné → i při odchycení A, B
- **Autenticita veřejných klíčů**
  - → je třeba ověřit, že jmenovka = identifikační značka patří ke klíči
  - → autenticitu ověří třetí strana a připojí svůj podpis
    1) Web of trust – uživatelé potvrzují autenticitu klíčů dalších uživatelů
       - ↳ když to potvrdil někdo komu důvěřujeme → ✓
    2) Public Key Infrastructure – klíče podepisují speciální organizace
       - → Certification Authorities (CA) → podepíše klíč + značku a ten, kdo důvěřuje CA považuje za důvěryhodný i klíč

X.509

- **Certifikát**
  = klíč + identifikace vlastníka ← podepsané vydavatelem - např. CA
  → pokud důvěřujeme vydavateli, tak i klíči
  → **řetězec důvěry**: je třeba ověřovat CA → končíme se na certifikátu CA dokud nedospějeme k nějaké CA, které důvěřujeme

  → struktura certifikátu podle PKI
  - certifikát - verze certifikátu
    - sériové číslo
    - vydavatel
    - doba platnosti
    - vlastník veřejného klíče
    - info o klíči (algoritmus a klíč)
  - algoritmus pro elektronický podpis
  - elektronický podpis

  — uživatel stáhne certifikát a zkontroluje podpis pomocí veřejného klíče CA, která ho vydala

- **SSL & TLS**
  - SSL (Secure Socket Layer) se ve verzi 3.0 přejmenovala na TLS (Transport Layer Security) 1.0 → dnes se používají TLS 1.1 +
  - speciální mezivrstva mezi transportní a aplikační vrstvou umožňující autentifikaci a šifrování // HTTP + SSL
  - využívá to řada starších protokolů → HTTPS na portu 443
  - **princip**:
    1) klient pošle požadavek na SSL spojení + parametry
    2) server pošle odpověď + parametry + svůj certifikát
    3) klient ověří server a vygeneruje základ šifrovacího klíče → pošle ho serveru (zašifrovaný veřejným klíčem serveru)
    4) server rozšifruje základ klíče. Z toho základu vygenerují server i klient celý klíč.
    5) klient a server si navzájem potvrdí, že odteď bude jejich komunikace šifrovaná

- **Aplikační vrstva TCP/IP**
  - → spojuje funkce OSI 5, 6 a 7
  - → protokol na aplikační vrstvě definuje
    - průběh dialogu – kdo iniciuje spojení, ...
    - formát zpráv – textový / binární, struktura
    - sémantiku zpráv a informačních polí – která část zprávy znamená co
    - typy zpráv – jaké jsou požadavky a odpovědi na ně
    - interakci s transportní vrstvou – TCP / UDP – kdy, jak?

- **Domain Name System** – DNS (doménových / IP)
  - → klient-server aplikace pro překlad jmen na adresy a naopak
  - → binární protokol nad UDP i TCP, port 53
    - běžné dotazy (odpověď do 512 v non EDNS) se vyřizují v UDP
    - větší datové výměny probíhají v TCP
  - → klient se obrací na DNS servery, jejichž adresy má zadané ve své konfiguraci
    - → nakonec se dozví co potřebuje
    - → pokud odpověď neobsahuje potřebné info, měla by obsahovat odkazy na servery, kterých je třeba se ptát dál
  - → jednotkou dat je záznam (Resource Record – RR) např:

    `mff.cuni.cz 3600 IN (internet) A 195.113.19.48` TTL = doba platnosti v sekundách
    ↳ jméno záznamu ↳ TTL ↳ typ ↳ data
  - → každá zpráva obsahuje hlavičku a určitý počet záznamů
  - → typy DNS záznamů
    - **SOA** (Start Of Authority) – úvodní záznam → informace jako datum poslední změny, ...
    - **NS** – záznamy definující nameservery, které udržují databázi záznamů dané domény
    - **A** – IPv4 adresa pro dané jméno } převod jmen na adresy
    - **AAAA** – IPv6 adresa – " – }
    - **PTR** – reverzní záznam pro převod adres na jména

      IPv4: 1.2.3.4 → 4.3.2.1.in-addr.arpa

      IPv6: ::1 → 1.0.....0.0.ip6.arpa ← půlbyty odělené tečkami
    - **CNAME** – záznam pro tvorbu aliasů → alias ... kanonické (skutečné) jméno počítače
    - **MX** (Mail eXchanger) – říká, který server přijímá pro danou doménu (počítač) poštu

Typ	Jméno záznamu	Data
SOA	jméno domény	obecné informace o doméně
NS	jméno domény	jméno nameserveru domény
A	jméno počítače	IPv4 adresa počítače
AAAA	jméno počítače	IPv6 adresa počítače
PTR	reverzní jméno (např. pro IP adresu 1.2.3.4 je to `4.3.2.1.in-addr.arpa`, pro ::1 je to `1.0...0.ip6.arpa`)	doménové jméno počítače
CNAME	jméno aliasu	kanonické jméno počítače
MX	jméno domény/počítače	jméno poštovního serveru a jeho priorita

- **Aplikační vrstva TCP/IP**
  - → spojuje funkce OSI 5, 6 a 7
  - → protokol na aplikační vrstvě definuje
    - průběh dialogu – kdo iniciuje spojení, ...
    - formát zpráv – textový / binární, struktura
    - sémantiku zpráv a informačních polí – která část zprávy znamená co
    - typy zpráv – jaké jsou požadavky a odpovědi na ně
    - interakci s transportní vrstvou – TCP / UDP – kdy, jak?

- **Domain Name System – DNS** (doménových / IP)
  - → klient–server aplikace pro překlad jmen na adresy a naopak
  - → binární protokol nad UDP i TCP, port 53
    - běžné dotazy (odpověď do 512 v non EDNS) se vyřizují v UDP
    - větší datové výměny probíhají v TCP
  - → klient se obrací na DNS servery, jejichž adresy má zadané ve své konfiguraci
    - → nakonec se dozví co potřebuje
    - → pokud odpověď neobsahuje potřebné info, měla by obsahovat odkazy na servery, kterých je třeba se ptát dál
  - → jednotkou dat je záznam (Resource Record – RR) např:

    mff.cuni.cz 3600 IN (internet) A 195.113.19.48 TTL = doba platnosti v sekundách
    ↳ jméno záznamu ↳ TTL ↳ typ ↳ data
  - → každá zpráva obsahuje hlavičku a určitý počet záznamů
  - → typy DNS záznamů
    - SOA (Start Of Authority) – úvodní záznam → informace jako datum poslední změny, ...
    - NS – záznamy definující nameservery, které udržují databázi záznamů dané domény
    - A – IPv4 adresa pro dané jméno } převod jmen na adresy
    - AAAA – IPv6 adresa – " – } převod jmen na adresy
    - PTR – reverzní záznam pro převod adres na jména

      IPv4: 1.2.3.4 → 4.3.2.1.in-addr.arpa
      IPv6: ::1 → 1.0.... 0.0.ip6.arpa ← půlbyty oddělené tečkami
    - CNAME – záznam pro tvorbu aliasů → alias ... kanonické (skutečné) jméno počítače
    - MX (Mail eXchanger) – říká, který server přijímá pro danou doménu (počítač) poštu

- **Servery DNS**
  - **primární** (master) server – spravuje záznamy o doméně
  - **sekundární** – pravidelně stahují a zálohují aktuální obsah databáze z)
  - **caching-only** – pokud se k záznamům dostanou i nějaké další servery, tak si je jen dočasně uloží do cache – dokud je potřebují

→ každá doména (zóna) musí mít alespoň 1 autoritativní (prim./sek.) server

→ v SOA záznamu je uvedeno, jak často mají sekundární servery aktualizovat databázi → vyžádají si data od primárního

→ primární server může mimořádně říct sekundárům ať se aktualizují

→ pro výměnu dat se používá TCP

- **Vyřizování DNS dotazu**

→ uživatel zadá www.mff.cuni.cz

⇒ vygeneruje se dotaz pro nameserver v doméně, kde dotaz vznikl

→ bude **rekurzivní** = server přebírá odpovědnost za vyřízení dotazu

→ pokud tento nameserver nemá v cache info o hledané doméně → hledání

↳ neví nic o mff.cuni.cz, cuni.cz ani cz

⇒ obrátí se na **kořenový nameserver** – který ale neposkytuje rekurzivní odpověď

→ ve své databázi najde nejrelevantnější položku a tu pošle

↳ „pošli svůj dotaz nameserveru s adresou ...".

→ server si tuto info uloží do cache a pokračuje v dotazování k navrženému serveru

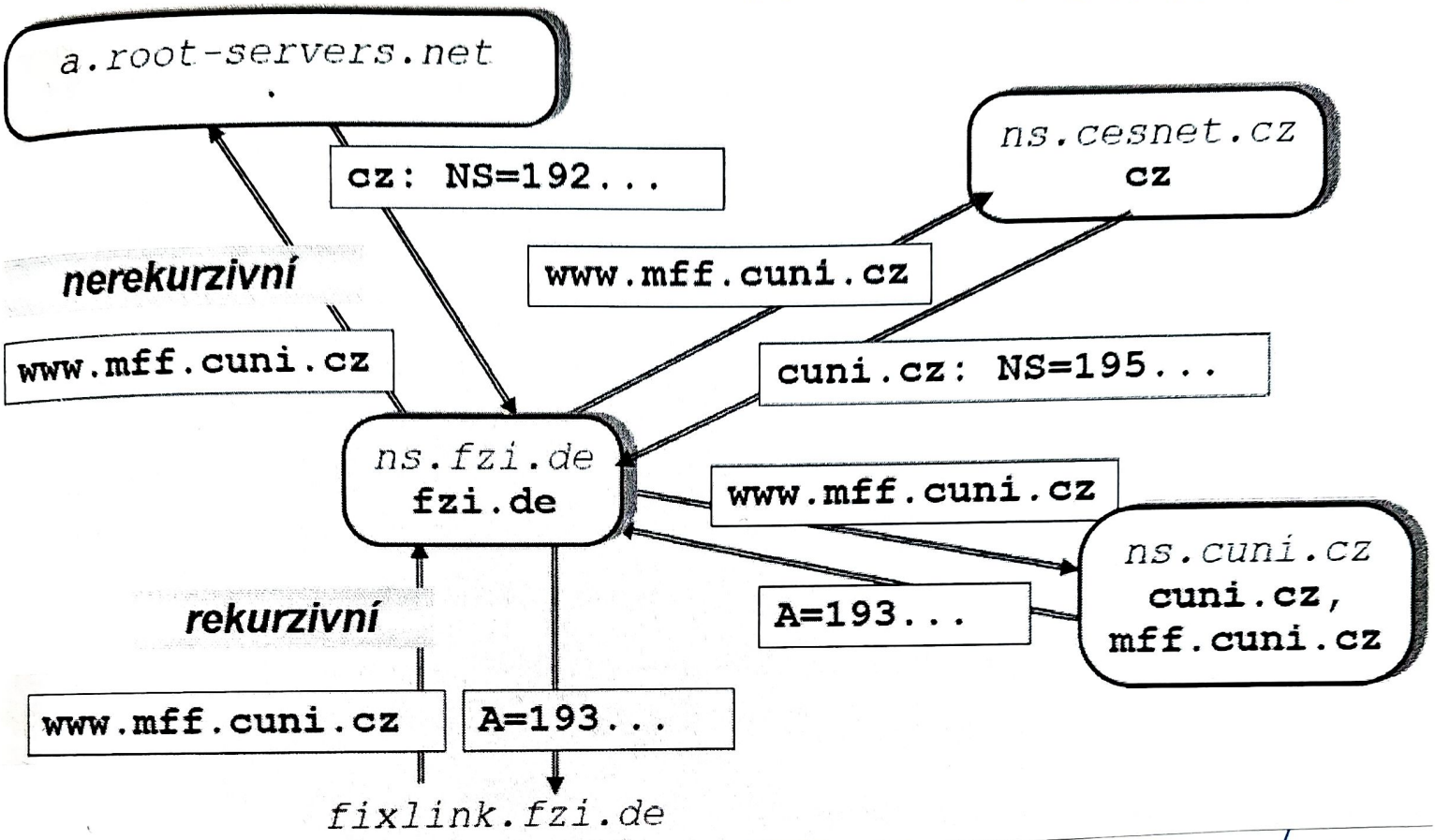
⇒ nakonec se dostane k nějakému autoritativnímu serveru, který zná odpověď

→ tuto konečnou odpověď pak přepošle klientovi

- **DNS dotaz a odpověď**

- **Dotaz**
  - ID – náhodné 2B číslo
  - FLAGS – příznaky
  - QUERY – 1 záznam bez dotazové části

- **Odpověď**
  - ID
  - FLAGS
  - QUERY
  - ANSWER – RR s odpověďmi
  - AUTHORITY – seznam nameserverů, které mohou dát autoritativní odpověď / info navíc
  - ADITIONAL – adresy nameserverů ↑
    ↳ adresy MX serverů z ANSWER, ...



klient posílá rádky s příkazy, server řádky s odpověďmi

↗ SMTP kódy

HTTP (404 - stránka nenalezena)

a, budou ještě další odpovědi

příkazy - např. heslo)

server ⇒ nepovedlo se)

- smysl příkaz opakovat)

- **Dotaz:**

```
ID:          n
FLAGS:       Recursion Desired
QUERY:       www.cuni.cz.   IN A
```

- **Odpověď:**

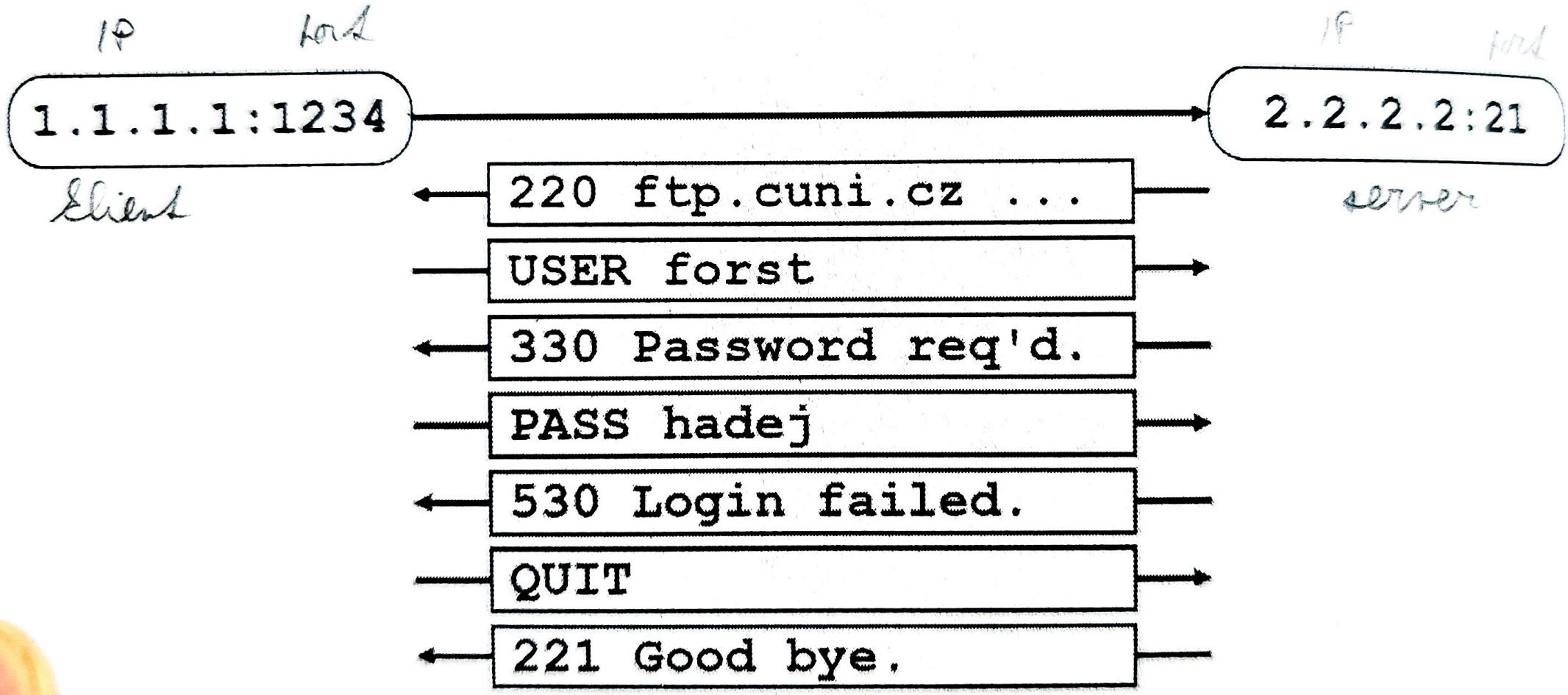
```
ID:          n
FLAGS:       Authoritative Answer
QUERY:       www.cuni.cz.   IN A
ANSWER:      www.cuni.cz.   IN CNAME tarantula
             tarantula      IN A     195.113.89.35
AUTHORITY:   cuni.cz.       IN NS    golias
ADDITIONAL:  golias         IN A     195.113.0.2
```

skutečné jméno www.cuni.cz

- **Bezpečnost DNS**
  - → problém útočníka: jak se dostat ke znění dotazu, abych mohl např. poslat falešnou odpověď?
    - → je těžké to odchytit
    - → šifrovat se to nevyplatí – náhodný zdrojový kód + ID
  - **cache-poisoning**
    - když klient pošle dotaz serveru útočníka, tak útočník může správně vyplnit sekci ANSWER, ale do AUTHORITY a ADITIONAL přidat falešné údaje o jiné doméně
      ⇒ získá kompletní kontrolu nad dotazy směřující do domény (třeba) cz
    - řešení: postupovat od root serverů a ptát se pouze autoritativních serverů
  - **DNSSEC** = DNS zabezpečené podpisy
    - je komplikované a rozšiřuje se pomalu

- **Diagnostika DNS**
  - → cmd: nslookup
  - → UNIX: dig

- **File Transfer Protocol – FTP**
  - nešifrovaný ↑
  - původně sloužil ke vzdálenému přístupu k vlastním datům pomocí otevřeného hesla !!!
  - ⇒ dnes hlavně anonymní přístup – uživatel anonymous/ftp, heslo je email
    - → uživatel tak získá přístup k volně dostupným datům
  - → je to textový protokol → klient naváže tzv. řídící spojení na server na portu 21
    - klient posílá řádky s příkazy, server řádky s odpověďmi
  - → kódy odpovědí (↗ SMTP taky)
    - každá odpověď začíná XXX kódem → převzalo to třeba HTTP (404 – stránka nenalezena)
      - 1XX = předběžná kladná odpověď (akce byla zahájena, budou ještě další odpovědi)
      - 2XX = definitivní kladná odpověď
      - 3XX = neúplná kladná odpověď (jsou nutné další příkazy – např. heslo)
      - 4XX = dočasná záporná odpověď (třeba je přetížený server ⇒ nepovedlo se)
      - 5XX = trvalá záporná odpověď (nepovedlo se a nemá smysl příkaz opakovat)

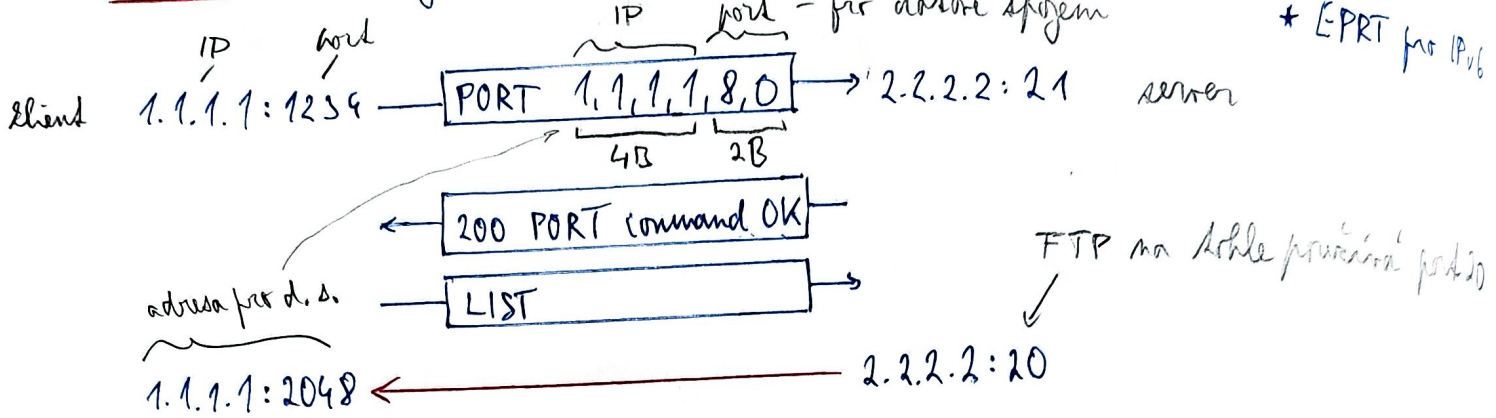

1.1.1.1:1234
2.2.2.2:21
client
server
220 ftp.cuni.cz ...
USER forst
330 Password req'd.
PASS hadej
530 Login failed.
QUIT
221 Good bye.

→ aktivní / pasivní datové spojení – FTP

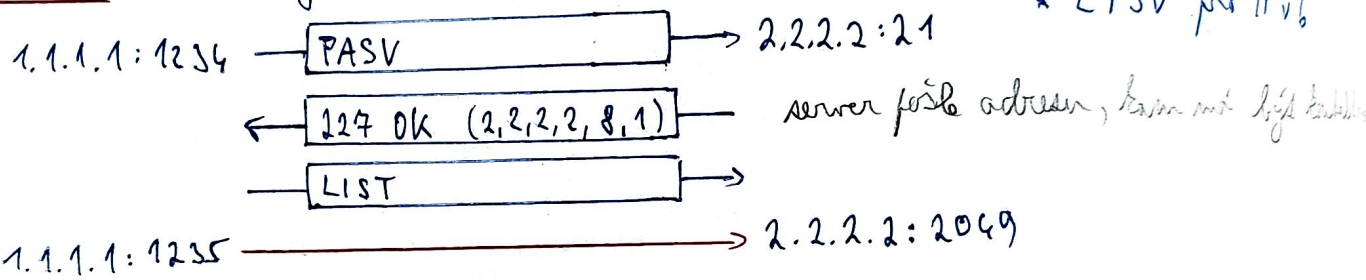
→ přenos dat probíhá po tzv. datovém spojení → přes TCP

→ musíme se dohodnout, kdo bude kanál otevírat a na jaký socket

• aktivní – navazuje server



• pasivní – navazuje klient



→ po skončení přenosu dat se datové spojení uzavře

→ aplikace pro FTP: WWW prohlížeč, správci souborů, cmd příkaz ftp

• Elektronická pošta

→ obecná služba existující i mimo internet

→ předávání zpráv (později i souborů) v lokální síti ⇒ offline

→ offline přístup k informačním službám – FTP archivy

→ na internetu používá SMTP na TCP portu 25 ← textový protokol

→ e-mailová adresa v internetu (typicky):

login@počítač

forst@ms.ms.mff.cuni.cz

↖ server, kde schránka leží

↗ název poštovní schránky

↳ bezpečnostní riziko

alias@doména

libor.forst@cuni.cz

častější

– původně krátké zprávy (64 kB), později posílání souborů

- **Příjem a odeslání pošty v SMTP**

→ uživatel vydá příkaz k odeslání dopisu

1) poštovní program zkontroluje adresu za @ a zjistí, který server bere poštu pro tu doménu

a) je možné, že mezi klientem a serverem nestojí žádné překážky ⇒ přímé doručení

2) **mail submission** - program předá dopis pomocí SMTP nějakému serveru, který má uložený v konfiguraci → nazývá se **mail-forwarder** ↑ MTA

3) uzly, které přijmají a doručují pošta nazýváme **Mail Transfer Agent (MTA)**

⇒ jednotlivé MTA si pomocí SMTP předávají dopis

⇒ server, který dopis přijme, si ho vždy dočasně uloží do fronty a pak pošle dál

4) pokud je cílový mailbot na serveru přístupném z internetu, tak se poslední MTA pokusí doručit dopis na tento server

→ správce toho serveru může v DNS nastavit nějaké **mail-exchangery**

⇒ potom to poslední MTA bude dopisy posílat na ně ↓ MX je obvykle přijmout dopis a doručit ho do síle

- **Přístup k poště z pohledu uživatele**

→ uživatel přistupuje k poštovnímu systému pomocí nějakého poštovního programu

⇒ **Mail User Agent (MUA)** - 2 možnosti připojení

- **přímé připojení** - uživatel se přijíjí na MTA, kde má svůj mailbot
  - aplikace má přístup k dopisům uživatele ⇒ příjem
  - zároveň —||— s službám MTA ⇒ odesílané zprávy řadí přímo do fronty MTA
- **připojení pomocí poštovního protokolu** - POP, IMAP
  - na MTA server se připojí klient toho protokolu a vodává uživatelovy poštovy ⇒ tyto poštovy ale slouží pouze ke čtení doručené pošty
  - tento program se současně připojí na nějaký další MTA server, kde pomocí SMTP dopisy odesílá

- **Ukázka SMTP protokolu**

→ MAIL FROM <ford@cuni.cz> ← nový dopis

← 250 ... obálka?

→ RCPT TO <medved@cuni.cz> ← příjemci jednotlivě → server je potvrzuje (250), může odmítnout

← 250 ... 450/550

→ DATA → začátek textu = to, co příjemce vidí

← 354 Enter mail, end with "." ← konec dopisu

→ From: <ford@cuni.cz>

→ To: <medved@cuni.cz>

→ ...

→ .

← 250...

→ QUIT

↳ pokud odpoví 250 → přebírá odpovědnost za doručení dopisu

→ pokud se to nepovede, tak píše Delivery Status Notification (DSN)

→ pozor na binární řád tvořící CR LF řádek!

⇒ odesílatel může do těla dopisu napsat jiného odesílatele než napsal do MAIL FROM!

→ když MTA odpověděl 250 na nějakého adresáta a nepovedlo se mu ho doručit, tak pošle DSN, kde nevyplňuje MAIL FROM ⇒ takové dopisy se často méně kontrolují ⇒ používají to spamovací enginy ⇒ některé správci prostě zakáží příjem dopisů tohoto formátu, což není dobře

- **Struktura dopisu**
  - → dopis se skládá ze záhlaví, které obsahuje hlavičky — jen ASCII 0–127
  - → pak je separátor (prázdná řádka) a následuje text dopisu – původně taky
    - ⇒ dnes rozšíření protokolu ESMTP, kde je ASCII 0–255
    - ⇒ je možné definovat strukturu a sémantiku těla dopisu pomocí MIME ⇒ soubory (pak přijd.)

- **Soubory a diakritika v poště**
  - → původně byly povoleny pouze ASCII 0–127 ⇒ kódování souborů / textů s non-ASCII znaky (binárních)
  - • UUENCODE
    - → vezmou se vždy 3B → rozdělí se na 4 × 6b. a tyto 6b. kódy se převedou na 4 tiskutelné znaky pomocí pevné tabulky
    - ⇒ 64 znaků: 26 velkých písmen, 10 číslic a 28 dalších znaků
    - → nafouknutí velikosti o 33% – je to šestný protokol Čau → R&%V
    - → zakódovaný soubor se vkládá mezi řádky begin, end
    - ⇒ problém: abychom soubor našli, tak musíme projet celý dopis
  - • Multipurpose Internet Mail Extention – MIME
    - – umožňuje strukturovat dokument (soubor) na hlavičku a tělo
      - ↳ hlavička mimo jiné obsahuje typ (text, image, audio) → důležitý je multipart
    - ⇒ multipart dokument obsahuje více MIME dokumentů
      - → pro každou část umožňuje popsat typ a formát (text/html), znakovou sadu a kódování...
      - → mail s přílohami se pošle jako multipart, kde první část je text a další části přílohy
    - → umožňuje používat diakritiku i v některých hlavičkách dopisu – např. předmět
    - – dnes používaný i mimo poštu
  - • Base64 – vychází z UUENCODE, jiná tabulka (a...z, A...Z, 0...9, +, /) a formát řádků ⇒ +33% (kódování)
  - • Quoted-Printable – asci znaky jsou ponechány beze změny ⇒ lepší čitelnost textových souborů
    - → non ASCII se nahrazují jako "=HH", kde HH je jejich hex hodnota ⇒ je třeba kódovat i =
    - ⇒ pro non ASCII máme +200% ⇒ vhodné pro kódování pouze ASCII souborů

nejdůležitější Hlavičky dopisu

`Date:`	datum pořízení dopisu
`From:`	autor (autoři) dopisu
`Sender:`	odesilatel dopisu
`Reply-To:`	adresa pro odpověď
`To:`	adresát(i) dopisu
`Cc:`	(carbon copy) adresát(i) kopie („na vědomí:“)
`Bcc:`	(blind cc) tajní adresáti kopie
`Message-ID:`	identifikace dopisu - slouží k vytváření vláken
`Subject:`	předmět dopisu
`Received:`	záznam o přenosu dopisu

• **Bezpečnost pošty** – uživatel

→ dopis se může dostat k bodvé lidem, kterým nebyl určen + SMTP není šifrované

řešení: šifrovat obsah dopisu → PGP (Pretty Good Privacy)

→ nikdy není jistý odesílatel → údaje v obálce a textu mohou být různé

částečné řešení: Sender Policy Framework

řešení: systém challenge-response, elektronický podpis

• **Bezpečnost pošty** – klient + server

– poštovní server, by měl posílat maily lokálních klientů / uživatelů komukoli a (↗ svých)

ostatní maily (přicházející zvenku) pouze lokálním uživatelům

⇒ ignorovat maily od cizích lidí pro cizí lidi + nedovolit cizincům posílat maily

– pokud server dovolí komukoliv aby se připojil a poslal mail kamkoli, tak je open relay

⇒ hrozí riziko zneužití pro rozesílání hromadných mailů

⇒ ∃ organizace, které vytváří seznam open-relay serverů – ignorace dopisů od nich

– když chce lokální uživatel poslat mail vzdáleně, tak to server bere jako cizího

⇒ ESMTP umožňuje autentikaci uživatele, což SMTP nemá built-in

↳ je to součást SASL profilu pro SMTP → příkaz AUTH

– klient může pomocí ESMTP příkazu STARTTLS požádat o zahájení SSL/TLS spojení

• **Ochrana proti spamu**

• **Grey-listing** – spam enginy obvykle neopakují pokus o doručení

⇒ poštovní server udržuje databázi tripletů <client (IP), sender (mail), recipient (mail)>

→ napoprvé odmítne poslat mail recipientovi odpovědí 450

→ klient se po cca 15 m. pokusí mail poslat znovu ⇒ teď už to přijme 250

(↙ když co to nezkouší dám na určitou dobu na blacklist)

• **Sender Polici Framework** – doména publikuje pomocí DNS jaké jsou její poštovní servery

⇒ pokud dopis poslal někdo z dané domény z jiného serveru ⇒ ignorujeme ho

→ problém: když má někdo nastavené forwardování pošty na jiné místo, tak to selže, protože odesílatel je stejný, ale najednou to posílá jiný stroj ⇒ dnes se nepoužívá

• **DomainKeys Identified Mail (DKIM)** – podobná myšlenka – seznam poštovních serverů

⇒ odesílací server dopis podepíše ⇒ forwardování funguje

• **Antispam algoritmy** – server na základě nastavené heuristiky odhaduje pst. že mail je spam → diskutabilní účinnost a riziko false positive

↗ textový protokol

- **Post Office Protocol** – POP
  - poštovní protokol pro přístup uživatele k mailboxu – další je IMAP
  - starý protokol → dnes podporován hlavně kvůli zpětné kompatibilitě ↗ dnes
  - hlavní nevýhody
    - → otevřené posílání hesla → ∃ rozšiřující příkaz pro šifrovanou autentikaci (↗ ne moc rozšířený ↓ spíš TLS)
    - → dopisy je nutno stahovat ze serveru celé
      - ↳ taky ∃ rozšíření pro stažení jen začátku – taky řídce implementované

nemá pořadí dopisy

- **Internet Message Access Protocol** – IMAP
  - modernější, ale složitější nástupce POP → má ho většina dnešních MUA
  - hlavní výhody
    - → server uchovává info o dopisech (stav)
    - → podpora více schránek (složek)
    - → protokol umožňuje vyžádat pouze část dopisu a vyhledávat v dopisech (↗ podle složitých podmínek)
    - → built-in možnost používat TLS
      - ⇒ naváže spojení na vyhrazený port – příkaz STARTTLS
    - → taky textový, ale uživatelsky přívětivější
    - → možnost zadat víc příkazů najednou

nemá pořadí dopisy

- **Princip distribuované databáze**
  - → databáze informací uložených na obrovském množství serverů (↗ aniž by o tom věděl)
  - → jsou provázané tak, že uživatel přechází pomocí odkazů ze serveru na server
  - ⇒ Gopher – 1. celosvětově rozšířená ↗
    - ↳ něco jako web dneska → při přihlášení se zobrazilo menu s odkazy, které vedly buď na další menu / text / formulář
    - → poskytoval jen textové informace → soubory si člověk musel stáhnout

- **Hypertext**
  - → text obsahující vazby umožňující pokračovat čtením podrobnější informace
  - → později myšlenka doplnění textu o obrázky, zvuk, ...
  - ⇒ realizace 1989 v CERNu → služba World Wide Web

• World Wide Web – WWW

→ distribuovaná hypertextová databáze

→ základní jednotkou informace je hypertextová **stránka** (dokument)
  ↳ server ji posílá na žádost klientům

→ dokumenty jsou psány v **HTML** – popisuje obsah i formu
  ↳ konkrétní zobrazení je v režii klienta resp. uživatele

→ dokumenty – **statické** – cesta v URL pak obvykle odpovídá skutečné cestě na disku serveru
  dynamické – generují se dynamicky podle požadavků klienta

→ přenos stránek zajišťuje **HTTP** – chybějící zabezpečení ⇒ TLS ⇒ HTTPS

• HTTP v. 1

→ textový protokol, převažuje verze 1.1, port 80

→ **klient naváže spojení na a posílá požadavek:**
  • úvodní řádka – metoda (GET), cesta, verze protokolu
  • hlavičky – Host = jméno serveru, na který se klient obrací
    – jazyky, kódování, stáří stránky
    – data pro autentikaci, ...
  • tělo – volitelné → např. když klient uploaduje na server dokument

→ **server mu odpoví**
  • úvodní řádka – verze protokolu, kód odpovědi (200), slovní popis (OK)
  • hlavičky – formální věci → čas poslední změny, ...
    – detaily protokolu – způsob přenosu, ...
    – vlastnosti posílaného dokumentu – např. jeho MIME hlavičky
  • tělo – požadovaný dokument / text chybové zprávy

→ **kódy odpovědí** – jako FTP
  → 1,2,3 významově stejné, 4xx = chyba na straně klienta
  5xx = chyba na straně serveru

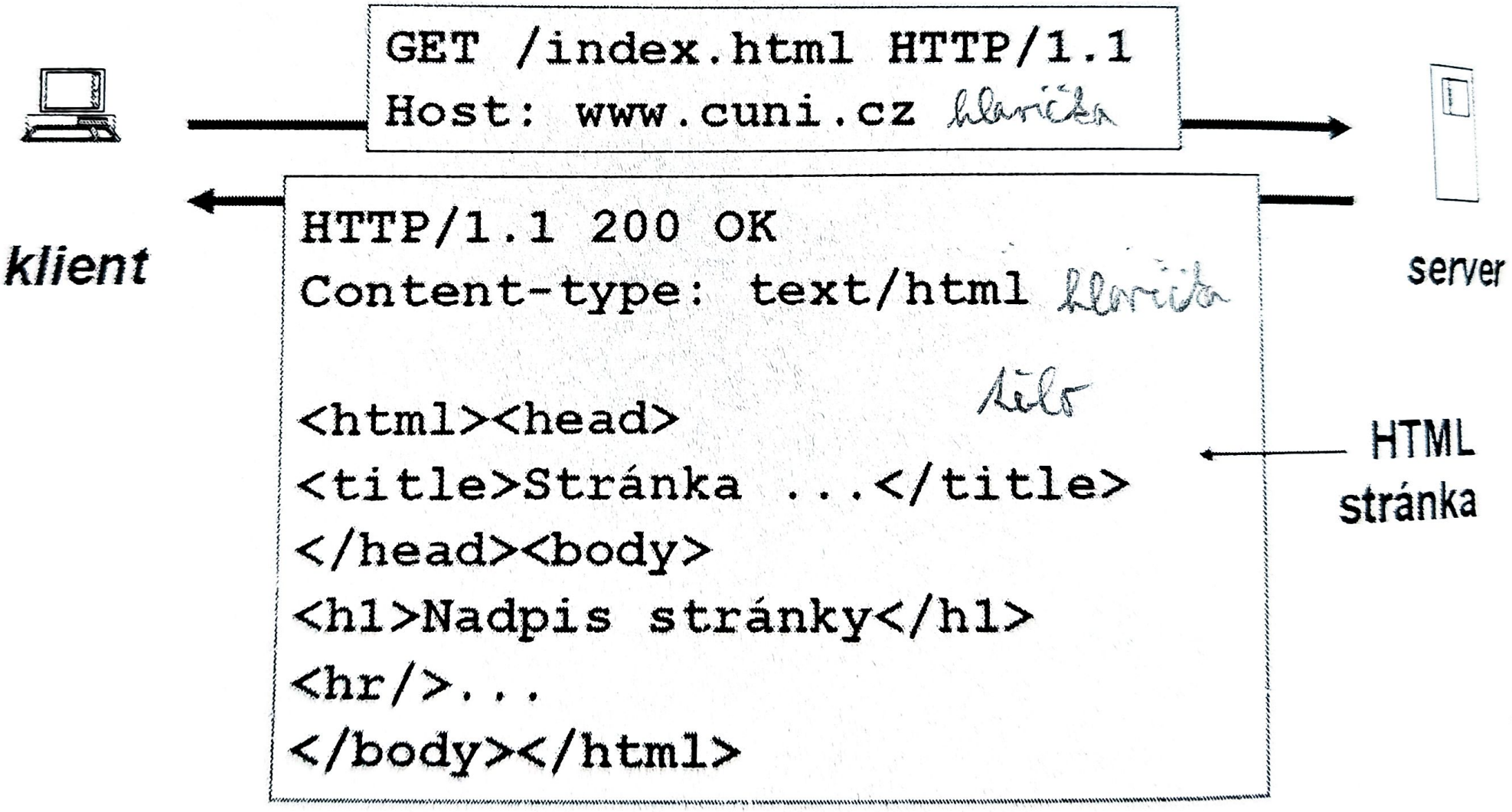
→ **metody HTTP**
  def: Metoda je **bezpečná** ≡ nemění obsah dokumentu – nikdy
  idempotentní ≡ opakované provolání má stejný efekt.

metoda	tělo požadavku	tělo odpovědi	bezpečná	idempotentní	
GET	—	dokument	✓	✓	← get doc.
HEAD	—	—	✓	✓	← get head
POST	parametry	dokument	×	×	← e.g. výsledky formuláře
PUT	dokument	—	×	✓	
DELETE	—	výsledek	×	✓	← úspěch/...
CONNECT	vytvoří ← tunel → skrzním je možné v HTTP realizovat jiné spojení v jiném protokolu				

→ bezpečnostní rizika
→ umožňuje obejít omezení dává bezpečnostní politikou sítě

Ukázka protokolu HTTP

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



- Vlastnosti HTTP v. 1
  - → požadavky jsou nezávislé ⇒ pokud se WWW stránka skládá z textu a 2 obrázků, tak to bude 3 nezávislé požadavky
  - → v 1.1: perzistentní spojení = po skončení požadavku nemusí klient zavírat TCP spojení
    ⇒ po 1 spojení může jít postupně více požadavků
    ↳ + klienti obvykle otevírají současně několik spojení
  - → celá komunikace je bezestavová ⇒ server neví, které požadavky patří k sobě, ...
    ⇒ pokud uživatel předá serveru nějaké info potřebné pro další práci (nastavení), tak by se server musel ptát u každého požadavku znovu
  - ⇒ cookies = data, které server vygeneruje na základě info od uživatele a pošle je klientovi pomocí hlaviček Set-Cookie
    - → prohlížeč si je uloží a při každém dalším požadavku je serveru pošle ve formě hlavičky Cookie ⇒ server zobrazuje správné nastavení, ...
    - → cookies nepředstavují přímé nebezpečí, ale server je může využít pro shromažďování info o uživateli + mohou být odcizeny - jsou uloženy v PC
- HTTP v. 2
  - binární protokol - lze na něj přejít v rámci HTTP/1 spojení
  - hlavní motivace → větší propustnost (rychlost)
  - metody - vlastní multiplexing → více streamů v rámci jednoho TCP spojení
    ↳ streamy se neblokují + se dají prioritizovat
    - server může poslat více dat, než klient požadoval, pokud soudí, že je bude potřebovat → pošle by obrázky na okraj rovnou (push)
    - kvůli autentikaci narůstá obsah hlaviček + často mají podobný obsah ⇒ lze je efektivně komprimovat
- HTML
  - 2014 verze 5
  - textový obsah stránky je doplněn značkami: strukturální → odstavec, sémantické → adresa, formátovací → tučně
  - je předchůdcem XML = Extensible ML

- **Telnet** – Telecommunication Network
  - velmi starý protokol
  - první řešení vzdáleného přihlášení na jiný stroj
    - → využívá emulaci terminálu (Network Virtual Terminal)
    - ⇒ protokol přenáší příkazy a reaka tak, že uživatel má podobné prostředí jako u reálného t.
    - → klient a server se musí domluvit kdo bude dělat práci
      - → když uživatel stiskne klávesu, někdo musí způsobit její zobrazení na obrazovce
        - ⇒ DO ECHO / DONT ECHO + WILL ECHO / WONT ECHO
      - → nevýhody polling : protokol neobsahuje příznak, zda jde o povel nebo odpověď
  - hlavní nevýhoda : otevřený přenos dat → později vyšlo rozšíření, ale až moc pozdě
  - dnes – občas se používá, když nehrozí odposlech hesel – segmenty LAN
    - hackování jiných protokolů, připojením na jejich server

- **Secure Shell** – SSH (Telnet, FTP)
  - bezpečná náhrada starých protokolů pro vzdálené přihlašování / přenos souborů
  - klient věřinje server + komunikace je šifrovaná
  - aktuální verze 2, port 22
  - SSH v2 navíc umožňuje: (současně NVT a přenášení souborů / terminál)
    - otevírat paralelně více zabezpečených kanálů
    - tunelovat zabezpečeným kanálem jiný provoz – možnost obcházení firewallu
    - zpřístupnit file systém tak, že se jeví jako lokální – SSHFS
  - Windows → klienti: putty, winscp
  - UNIX → příkazy: ssh, scp

- **Bezpečnost SSH** (potvrzuje uživatel)
  1, klient věřinje server – na základě kontroly veřejného klíče / certifikátu
  2, server věřinje uživateli – pomocí hesla / OTP nebo bez hesla ?
    - uživatel si může pro nějaké dvojice ⟨klient, server⟩ vygenerovat dvojici klíčů, přičemž ten veřejný uloží na server
    - ⇒ server pošle zašifrovanou výzvu a klient odpoví plaintextem
    - bezpečnostní riziko : když útočník získá přístup k účtu, tak se může přihlásit na všechny stroje používající stejnou dvojici klíčů nebo pokud je možné se ze stroje A přihlásit na B a reciprocně B → A pomocí stejné dvojice → ŠPATNĚ
  - ⇒ princip internetových červů – řešení: chránit tajný klíč heslem

- **Voice over IP – VoIP**
  - obecné označení technologií pro přenos hlasu pomocí TCP/IP
  - lze realizovat různými navzájem nekompatibilními způsoby
    - standard H.323
    - standard SIP
    - využití obecnějšího protokolu – SKYPE + HTTP
  - celá řada problémů
    - digitalizace hlasu, nalezení partnera, dohodnutí vlastností zařízení
    - propojení s běžnou telefonní sítí

- **H.323** (→ telekomunikační společnost.)
  - komplexní řešení multimediální komunikace od ITU (International Telegraph Union)
  - binární protokoly – sítění se každým bitem – založeno na ASN.1
  - celá řada dílčích protokolů – ne všechny jsou volně dostupné
    - H.225 / RAS (Reg. / Adm. / Status) pro vyhledávání partnera pomocí tzv. gatekeeper
    - Q.931 řeší navazování spojení
    - H.245 řeší řízení hovoru (dohodnutí používaných vlastnostech zařízení)
    - RTP kanály (Realtime Transfer Protocol) → přenos multimediálních dat – audio / video
    - RTCP (RTP Control Protocol) → řídí RTP kanály
  - dnes postupně nahrazováno SIP

- **Abstract Syntax Notation 1 – ASN.1**
  - metoda, jak definovat nějakou datovou strukturu / obsah jednotky dat protokolu pomocí formální definice → velmi užitečný nástroj
  - problém je implementace v H.323 – původně se vše zapisovalo jen bohužel bity (což je malo
    - → autoři zabudovali mechanismus umožňující budoucí rozšíření
      - ⇒ extrémně složitá implementace
      - ⇒ kupují se knihovny, které z textového zápisu v ASN.1 vygenerují kód, který realizuje zápis a čtení R.323
  - používá ho i X.509 – Public Key Infrastructure (PKI)

• Session Initiation Protocol – SIP

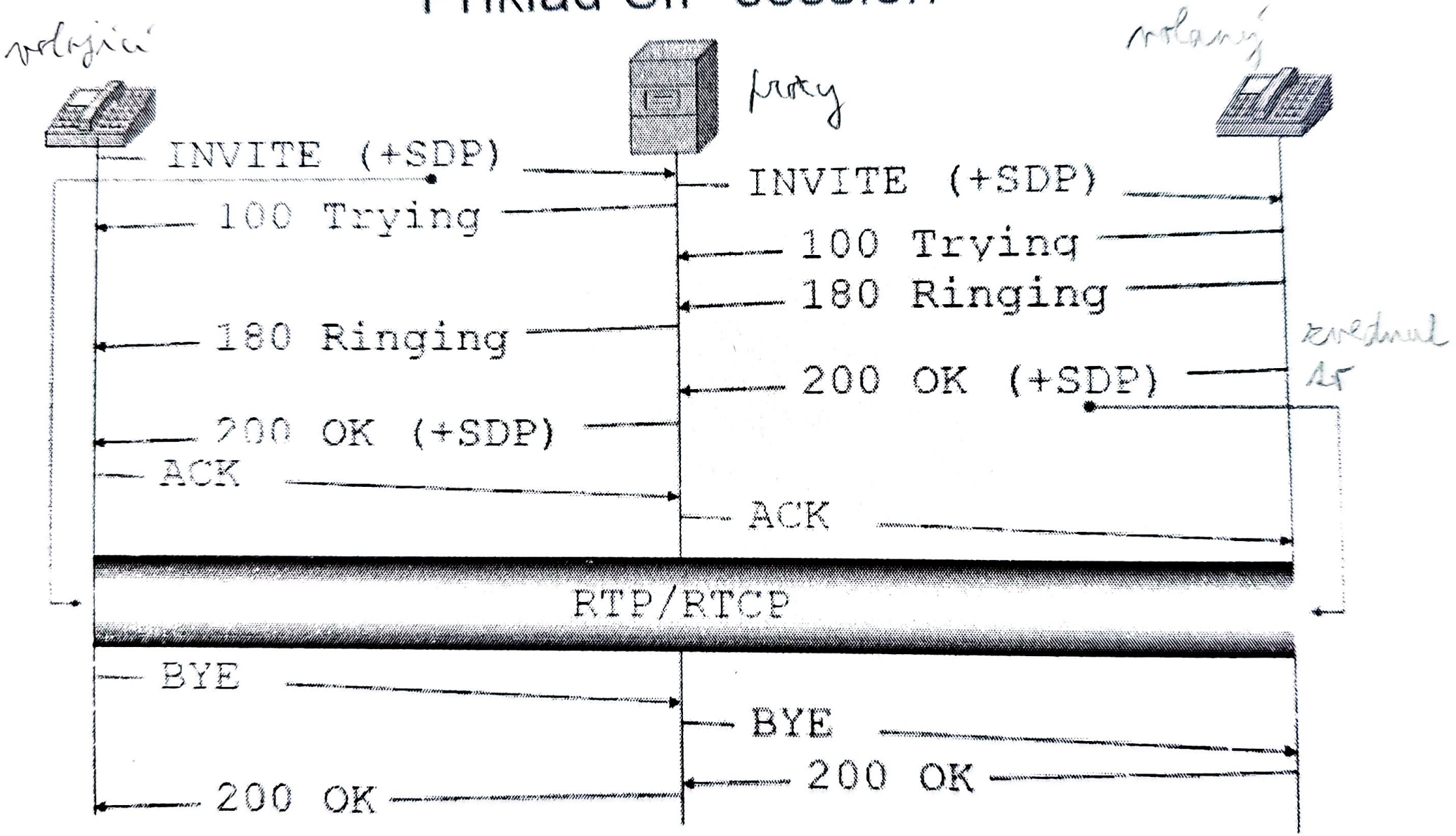
- náhrada složitého H.323 jednodušším protokolem   port TCP/UDP 5060
- architektura podobná HTTP, informace se přenášejí ve formě hlaviček
- neřeší vlastní přenos dat ⇒ využívá RTP + RCTP
→ řeší jen vyhledání partnera a navázání spojení
⇒ dohodu o parametrech datových kanálů řeší Session Description Protocol (SDP)
  ↳ textový protokol → řádky formátu keyword = value
  → přenášen pomocí SIP zpráv

- koncový uzel se může registrovat u registrátora ⇒ lze se propojit na telefonní síť
- používá proxy servery – usnadňují komunikaci přes hranice různých sítí
  ↳ jako u SMTP se během přenosu vkládají do zprávy hlavičky s cestou
    ⇒ protistrana může správně směrovat odpověď   ↳ Via, Record-Route

→ ukázka SIP hovoru

1, volající vyšle INVITE s volaným URL a nabídkou datových kanálů jako SDP zprávu
2, příkaz dorazí na nejbližší proxy ⇒ začne řešit nalezení cíle – podle své konfig. + URL
  ↳ nejdříve další proxy, …
  ⇒ poslední proxy upraví SDP – nějaký překlad adres ← to zaručení že bude posílat proxy a ten to bude přeposílat dál → něco jako NAT
3, proběhne nějaké vyzvánění – viz obrázek
4, když to volaný zvedne, tak pošle 200 OK + SDP zprávu s nabídkou datových kanálů
  ↳ proxy to zase přeloží
5, volající to potvrdí ACK
6, od tohoto okamžiku obě strany využívají datové kanály s RTP/RCTP

Příklad SIP session



- Sdílení systému souborů
  → vzdálené připojení cizího filesystému transparentně do lokálního (→ připojený disk se jeví jako součást místního systému adresářů)
  - Network File System - NFS
    - původně z firmy Sun Microsystems, dnes otevřený → RFC
    - primárně UDP, dnes i TCP — UNIX
    - připojený disk je identifikován jako server:cesta
    - autentikace pomocí protokolu Kerberos
    - má relační (RPC - Remote Procedure Call) a prezentační (XDR) vrstvy — Exchange Data Representation
  - Server Message Block - SMB
    - původně vyvinula IBM, potom Microsoft ⇒ není open
    - reverse engineering ⇒ ∃ open implementace Samba - umožňuje propojení Win a Unix
    - identifikace disku: UNC \\server\zdroj
    - autentikace pomocí obyčejného uživatelského jména a hesla - Win

- Network Time Protocol - NTP
  - synchronizace času mezi uzly sítě — UDP
    ⇒ stejné timestampy souborů + porovnávání času událostí na různých počítačích
  - klient kontaktuje NTP servery uvedené v konfiguraci → ty kontaktují NTP server od ISP
  - zdroje / NTP servery mají klasifikaci:
    - přesné zařízení = stratum 0 → atomové hodiny
    - server stratum N → řízený podle zdroje stratum N-1 (← prevence cyklů)
  - problém: zdroj pošle čas, ale není přesný kvůli latenci sítě
    ⇒ v odpovědi jsou timestampy určující interval, kde leží skutečný čas
    ⇒ když je zdrojů víc, tak se hledá průnik těch intervalů pomocí Marzullova alg.

- Bootstrap Protocol - BOOTP
  - starý protokol, sloužil k přiřazování IP adres bezdiskovým stanicím (→ pevné mapování IP ⟷ MAC; → nemohly mít permanentní úložiště)
  → stanice pošle všem uzlům v lokální síti svoji MAC
    ⇒ BOOTP server najde klienta v seznamu a pošle IP
    → pokud je odděluje router, tak musí umět tu žádost poslat dál tomu BOOTP serveru
      = BOOTP forwarding
  → postupně se ukázalo, že klientům by se hodily i další info:
    → adresy routerů, nameserverů, NTP serverů, mail forwarderů, ...
  ⇒ protokol se rozšiřoval až vznikl DHCP

• Dynamic Host Configuration Protocol – DHCP

- vznikl z BOOTP — IPv4: UDP porty 67 (server) 68 (klient)
- stejný formát zpráv ⇒ BOOTP klient může komunikovat s DHCP serverem
- kromě statické alokace adres i dynamická – MAC se mohou měnit + hodně klientů
- časově omezený pronájem adres – lease-time
- kooperace více serverů v 1 síti ⇒ mohou se lišit účelem / nabídkou adres
  ↳ klient si z jejich nabídek vybírá

- průběh DHCP

1, klient pošle broadcast DHCP DISCONER
   ⇒ jednotlivé DHCP servery v síti mu začnou posílat své nabídky DHCPOFFER
   → klient chvíli čeká a sbírá odpovědi ⇒ pak si z nich vybere tu nejlepší
2, klient pošle DHCP REQUEST s IP, kterou si vybral (↳ ideálně tu, co měl poslední; pak podle doby pronájmu)
   ↳ pořád broadcast, aby ostatní servery odblokovaly svoje nabídky pro něj
   ⇒ server to potvrdí DHCP ACK, že adresa je opravdu stále volná
3, od tohoto okamžiku začíná doba pronájmu
   → v polovině této doby klient pošle stejnou zprávu serveru DHCP REQUEST
   a, dostane odpověď ⇒ startuje mu nový interval doby pronájmu
   b, pokud odpověď nedostane, tak v 7/8 doby pronájmu pošle nový DHCP REQUEST, tentokrát broadcastem
   ⇒ pokud ani tentokrát adresu nedostane, tak po uplynutí doby pronájmu

• Prezentační vrstva – OSI 6

- představa o všeobecném modelu popisujícím kódování
  - datových typů, datových struktur, ...

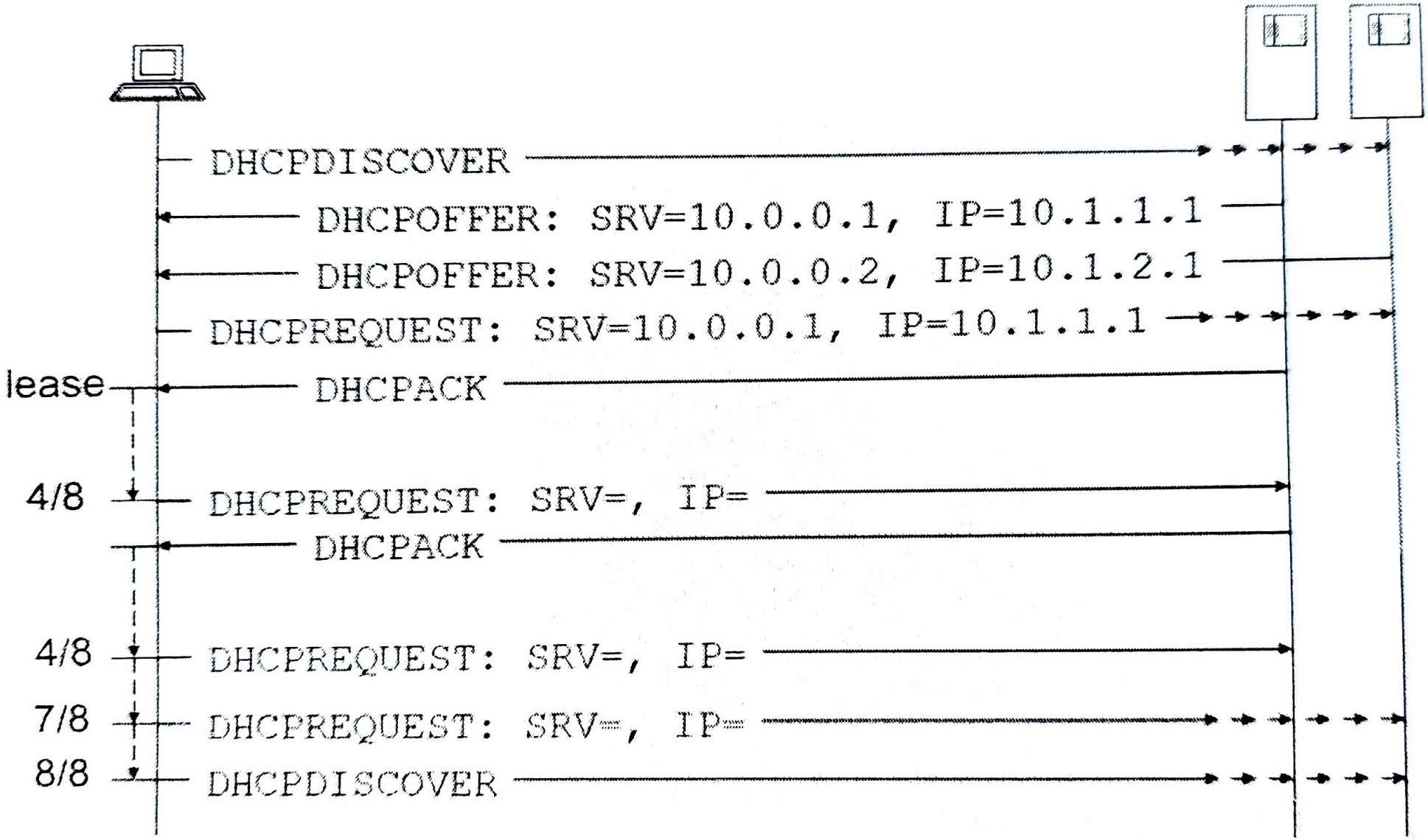
⇒ velmi složité – kdo a kdy dekóduje / kóduje

→ pokus o realizaci: ASN.1 – pokus dobrý, ale strašně složitá implementace

→ TCP/IP obecnou potřebu pohlcuje ⇒ konverzi provádí aplikace

→ praktické problémy:

- konce řádek: Win CR (0x0D) + LF (0x0A) vs. UNIX LF
- big endian vs little endian
  ⇒ TCP IP používá big endian ⇒ MSB se posílá jako první
  ⇒ ∃ knihovny pro konverzi


DHCPDISCOVER
DHCPOFFER: SRV=10.0.0.1, IP=10.1.1.1
DHCPOFFER: SRV=10.0.0.2, IP=10.1.2.1
DHCPREQUEST: SRV=10.0.0.1, IP=10.1.1.1
lease
DHCPACK
4/8
DHCPREQUEST: SRV=, IP=
DHCPACK
4/8
DHCPREQUEST: SRV=, IP=
7/8
DHCPREQUEST: SRV=, IP=
8/8
DHCPDISCOVER

- **Relační vrstva** – OSI 5
  - představa o obecném modelu dialogu
    - 1 dialog může obsahovat více spojení } obecně neproveditelné
    - pro 1 spojení může probíhat více dialogů }

⇒ TCP/IP začlenilo dialog do aplikačních protokolů

- SMTP – v rámci 1 spojení může být vyřízeno několik mailů (postupně)
- SIP – inicializuje dialog pomocí 1 řídicího spojení a 2 datovými kanály (audio + video)

- **Transportní vrstva** – OSI 4
  - rozpráší za end-to-end přenos dat mezi konverzními aplikacemi
  - zprostředkovává služby síťě aplikačním protokolům
  - umožňuje provozování více aplikací (klientů a serverů) na stejném uzlu ← porty
  - volitelně zabezpečuje spolehlivost přenosu dat } TCP
  - volitelně segmentuje data a opětovně je skládá } TCP
  - volitelně řídí tok dat – flow control ~ rychlost vysílání

⇒ multiplexing

- **TCP** – Transmission Control Protocol
  - pro spojované služby (telefonní hovor)
  - klient navazuje spojení → data tečou ve streamu
  - spojení řídí a zabezpečuje TCP
  - data ve spojení proudí oběma směry, protože protistrana potvrzuje doručení segmentů
  - méně pravidelné spolehlivé bezeztrátové spojení
- **UDP** – User Datagram Protocol
  - pro nespojované služby – zpráva
  - neexistuje spojení, data se posílají jako nezávislé zprávy
  - UDP je jednoduché, relaci řídí aplikace
  - pravidelný tok za cenu vyšší ztrátovosti
- SCPT, DCCP, MPTCP – další modifikace či kombinace

• Struktura UDP datagramu

→ v UDP hlavičce se přenáší pouze informace o multiplexingu - SRC a DST port a řídicí informace - délka a kontrolní součet

• Struktura TCP packetu

- aby TCP mohlo garantovat kompletnost přenosu, tak segmenty musí mít ID ~ offset → SEQ number (není to jako č. paketu +1 → zvětšuje se po bytech)
  ⇒ pro potvrzování packety: ACK number
- flags obsahuje příznaky
- Urgent pointer je pro out-of-band přenos
  ⇒ aplikace označí data jako urgentní příznakem URG
  ⇒ taková data se vloží do normální komunikace a jejich bázová adr. v rámci datového bloku (je v tom urgent pointru)

- TCP window
  - TCP posílá více bloků najednou → příklad 1 blok = 10B & okno = 40B
  ⇒ protistrana to potvrdí příznakem ACK a hodnotou ACK number nastavenou na offset konce dat, která byla doručena
    → může to poslat v rámci nějakého datového packetu - jinak by to bylo neefektivní
  ⇒ když dorazí ACK, tak vysílající posune okno
  ⇒ když se okno naplní, tak přeruší odesílání a čeká na ACK
    ⇒ když ACK nepřijde, tak znovu pošle první nepotvrzený blok dat

- navázání TCP spojení - three-way-handshake
  → sekvenční čísla ("offset") z bezpečnostních důvodů nezačíná od 0 ⇒ náhodné číslo (svoje SEQ num si vygeneruje vysílající)

A → | SYN | SEQ# C | ACK# 0 | → B

← | SYN, ACK | SEQ# D | ACK# C+1 |

| ACK | SEQ# C+1 | ACK# D+1 | → (jen hlavičky, Ø data)

⇒ offset u A se počítá od C
⇒ —||— B —||— D

- uzavření TCP spojení - jednostranné
  → A pošle FIN packet ⇒ říká, že už nebude posílat žádná data
  → než B pošle FIN, tak pořád může posílat data ⇒ A to bude potvrzovat ACK packety

A → | FIN | SEQ# X | ACK# Y | → B

← | ACK | SEQ# Y | ACK# X+1 |

→ protistrana hned / později provede totéž

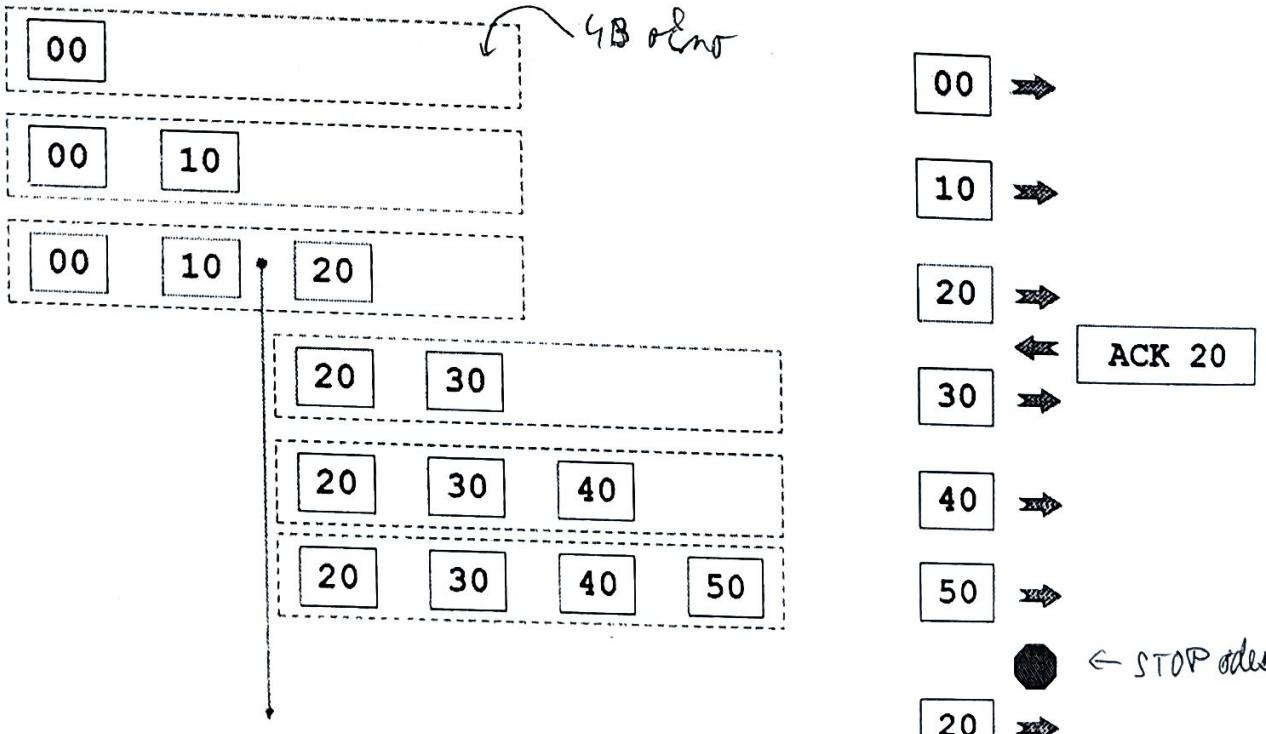
Source Port	Destination Port
Sequence Number	
Acknowledgement Number	
Data Offset / (rsvd) / Flags	Window
Checksum	Urgent Pointer
Options	
Data	

TCP

Source Port	Destination Port
Length	Checksum
Data	

UDP

e segmentuje data a opětovně je skládá



- TCP příznaky
  - SYN – packet slouží k synchronizaci čísel segmentů - inicializace SEQ num
  - ACK – packet potvrzuje doručení všech packetů až po ACK num (výjimka)
    ↳ packet může i nemusí obsahovat data
  - URG – packet obsahuje urgentní data, jejich hlavní adresa je v urgent pointru
  - FIN – odesílatel už nehodlá posílat žádná data
  - RST – reset – odesílatel odmítá příjmout spojení ⇒ označuje chybějící informaci
  - PSH – informuje příjemce, že obdržel kompletní blok a má ho předat aplikaci

- výpis existujících socketů
  cmd: **nestat -an** → vypíše seznam všech TCP i UDP serverů a otevřených TCP spojení
  Protokol, Local Adress, Foreign Address, State

- **Síťová vrstva** – OSI 3
  - hlavní funkce je přenos dat předaných transportní vrstvou od zdroje k cíli
  - využívá:
    - **adresace** – protokol síťové vrstvy definuje tvar a strukturu adres uzlů v síti
    - **zapouzdření** – řídící data potřebná pro přenos se přidají k PDU
    - **routing** – vyhledání nejvhodnější trasy k cíli přes mezilehlé sítě
    - **forwarding** – router, který není koncový předává packety o krok dál
    - **decapsulation** – vybalení dat a předání transportní vrstvě
  - příklady protokolů: IPv4, IPv6, IPX, AppleTalk

- **Internet Protocol** – IP
  - vlastnosti
    - **nespojovaná služba** – datagramy se doručují nezávisle
    - **best effort** – nespolehlivá, spolehlivost řeší vyšší vrstvy
    - **nezávislá na médiu** – vyšší vrstvy neřeší typ média
  - **adresy** – obsahují část s adresou sítě a část s adresou uzlu
    - IPv4 → 4B, IPv6 → 16B
  - **adresy přiděluje**
    - na vrcholu hierarchie stojí **IANA** (Internet Assigned Numbers Authority)
    - 5 světových regionů → regionální registrátoři → Evropa je pod RIPE NCC
    - dále ISP různých úrovní (privátní – správa, veřejné – ISP)
    - v LAN přiděluje IP adresy lokální správa sítě (ručně / automaticky – DHCP)

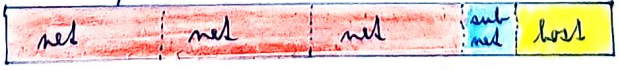
- **Struktura IPv4 datagramu**
  - délka hlavičky se udává v 32 bit slovech
  - **fragmentace** – když síťová vrstva dostane packet dlouhý tak, že při jeho zapouzdření by vznikl rámec delší než max. povolená délka pro danou linkovou vrstvu – **MTU (Maximum Transmission Unit)**

    ⇒ potom je třeba packet fragmentovat na více datagramů a posílat je postupně

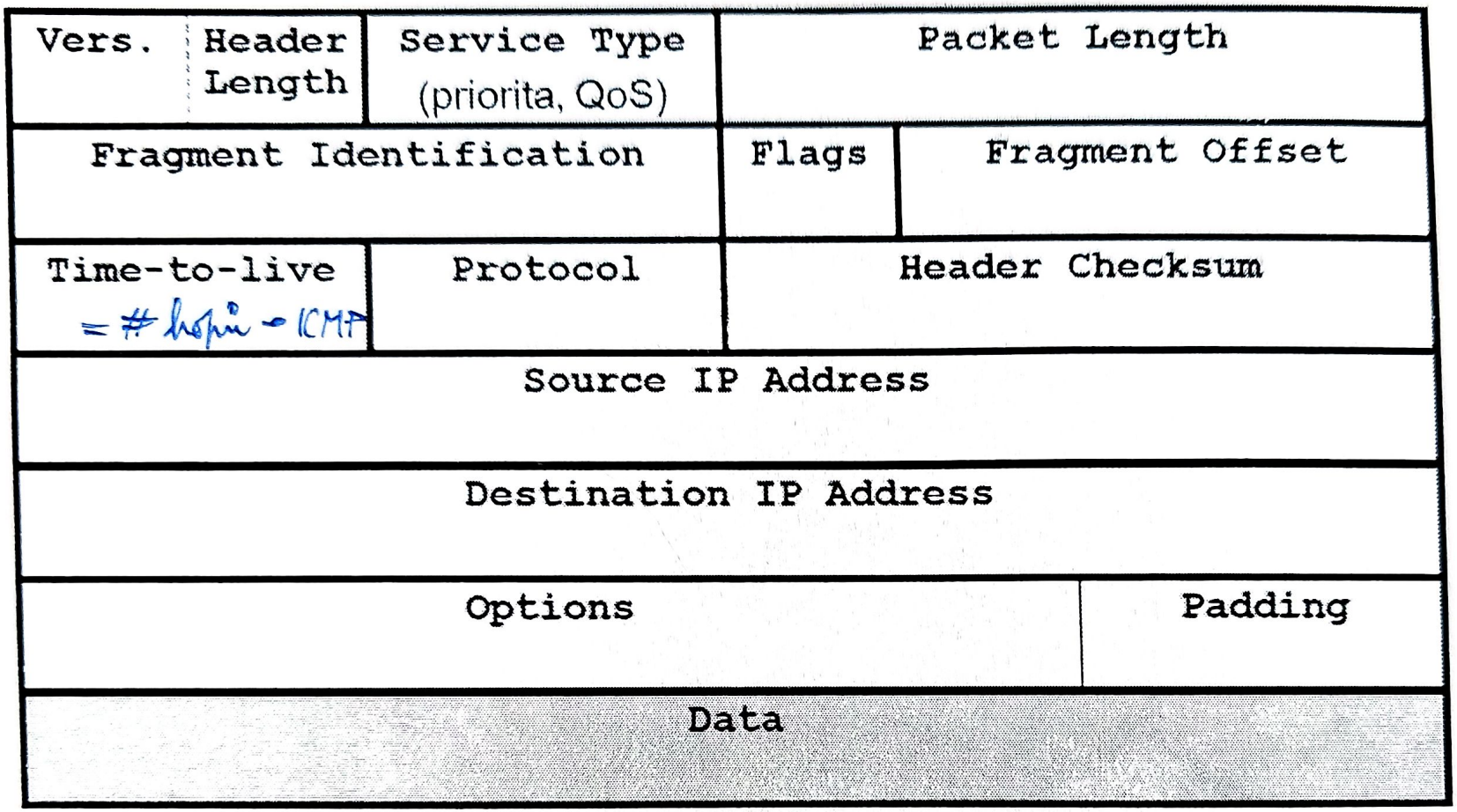
  → TCP se chce fragmentaci vyhnout ⇒ **Path MTU** – pakety se posílají s příznakem Do not fragment → odesílatel se dozví o problémech s velikostí MTU a může správně určit velikost segmentů
  - **TTL**, verze, číslo protokolu, kontrolní součet hlavičky, délka hlavičky
  - IP adresa odesílatele a příjemce

- **IPv4 adresy** (síť / počítač)
  - původně 1B, potom 3B, potom 1B+3B, nakonec třídy ABC → ABCD → ABCDE
  - **speciální adresy** – by design
    - **this host** – zdrojová adresa počítače s dosud nepřiřazenou adresou: 0.0.0.0/8
    - **loopback** – adresa "tohoto počítače" pro udělání smyčky: 127.0.0.1/8
    - **adresa sítě** – ⟨adresa sítě⟩⟨samé nuly⟩
    - **network broadcast** – ⟨adresa sítě⟩⟨samé jedničky⟩ ← všem v dané síti
    - **limited broadcast** – 255.255.255.255 ← všem v této síti
  - **speciální adresy** – by deffinition
    - **privátní adresy** – pro provoz v lokální síti, přiděluje správce, ven pomocí NAT
      - 1A: 10.0.0.0/8 16B: 172.16-31.0.0/16 256C: 192.168.*.0/24
    - **link-local adresy** – pouze pro spojení v rámci segmentu sítě, uzel si je volí sám
      - 169.254.1-254.0/16

- **Subnetting** = rozdělení sítě na podsítě rozšířením síťové části adresy

  C: 24/8 → 24/3/5: 

  → pomocí síťové masky: 1 ⟺ síť 0 ⟺ host 255.255.255.224 → adresa & maska = adresa sítě
  → nedoporučuje se používat subnet all-zeros a all-ones ⇒ 6 × 30 adres (70%)
  → často se ignorují třídy (A, B, C) ⇒ **classless mód**
  a uvádí se jen počet bitů adresy sítě ⇒ např. 193.84.57.1/27
  → pokud se v síti používají různé masky → **VLSM (Variable Length Subnet Mask)**
  → posun hranice sítě opačným směrem: **supernetting**



Třída	1.byte	2.byte	3.byte	4.byte	1. byte	Sítí	Adres
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	

třídy IPv4 adres

- Krize internetu
  - přeplňování routovacích tabulek
    ⇒ podstata problému: velký počet nesouvisle přidělených bloků rychle plní routovací tabulky
    ⇒ částečné řešení: realokace adres ⇒ CIDR (Classless InterDomain Routing) agregace
  - vyčerpávání adresového p.
    ⇒ kvůli hrubému členění sítí dochází k plýtvání
    ⇒ částečné řešení: přidělování bloků adres bez ohledu na třídy
    vracení nepoužívaných bloků
    privátní adresy + NAT ⇒ LAN s 1000 privátními + NAT = 1 veřejná
- IP verze 6
  - druhý vývoj, k IPv4 adoptována řada doplňujících nástrojů
  - přechod k IPv4 usnadňuje tunelování IPv4 a IPv6
  - konečná podoba adres: 128 b. (16 B) → 8·2 B
  - zápis: FEC0::1:800:5A12:3456/64
  - druhy adres:
    - unicastové - adresy 1 uzlu + zvláštní adresy:
      - Loopback (::1/128)
      - Link-Scope (FE80::/10) - dříve link-local
      - Unique-Local (FC00::/7) ~ privátní adresy v IPv4
    - multicastová - adresa skupiny uzlů
    - anycastová - de facto unicastová adresa přidělená více uzlům
      ⇒ více serverů po světě má stejnou adresu a my chceme ten nejbližší
      ⇒ routování na náš vzdálí doručení na ten nejbližší
    - chybějí broadcastové - používají se multicastové

• **Směrování / Routing**

- při směrování nějakého packetu chceme vždy najít **next-hop router**

⇒ **směrovací tabulka**

	destination (směr do internetu / síť)	mask (maska)	gateway (router)
- **default gateway**			
- další záznamy:	127.0.0.0	/8	127.0.0.1

→ next hop router vybereme jako ten nejspeciálnější záznam

→ použijeme masku ⇒ získáme adresu sítě kam vede ⇒ porovnáme s koncovou adresou

• **přímé záznamy** = záznamy popisující přímo připojené sítě / hosty

→ tyto záznamy vznikají v tabulce automaticky po konfiguraci síťového rozhraní

→ jako gateway je uvedena **naše vlastní adresa***, aby bylo jasné, že není třeba hledat next-hop router → tato adresa je v každé síti jiná

→ formální síťové rozhraní s loopback adresou - 1. záznam v příkladu

* adresa nějakého vlastního síťového rozhraní

• **nepřímé záznamy** = záznamy pro nepřímo propojené sítě/podsítě

→ gateway je tentokrát skutečná adresa next-hop routeru

→ podle vzniku dělíme záznamy na

• **implicitní** - vznikne automaticky nakonfigurováním síťového rozhraní

• **explicitní** - do tabulky se vadá příkazem - ručně / ho zavolá OS při startu PC

• **dynamické** - záznam se vytvoří v průběhu práce pomocí info od dalších uzlů v síti

→ směrovat by měla umět každá stanice (host?) v TCP/IP síti

→ maska určuje rozsah adresy sítě ~ uvažovanou část adresy destination

- **Směrovací algoritmus**

najdi v tabulce všechny vyhovující záznamy

↓

existuje —ne→ No route to host ⇒ packet se zahodí - nastane jedině když není nastavený default gateway

↓

zvol nejspeciálnější záznam (nejširší maska, nejdelší prefix)

↓

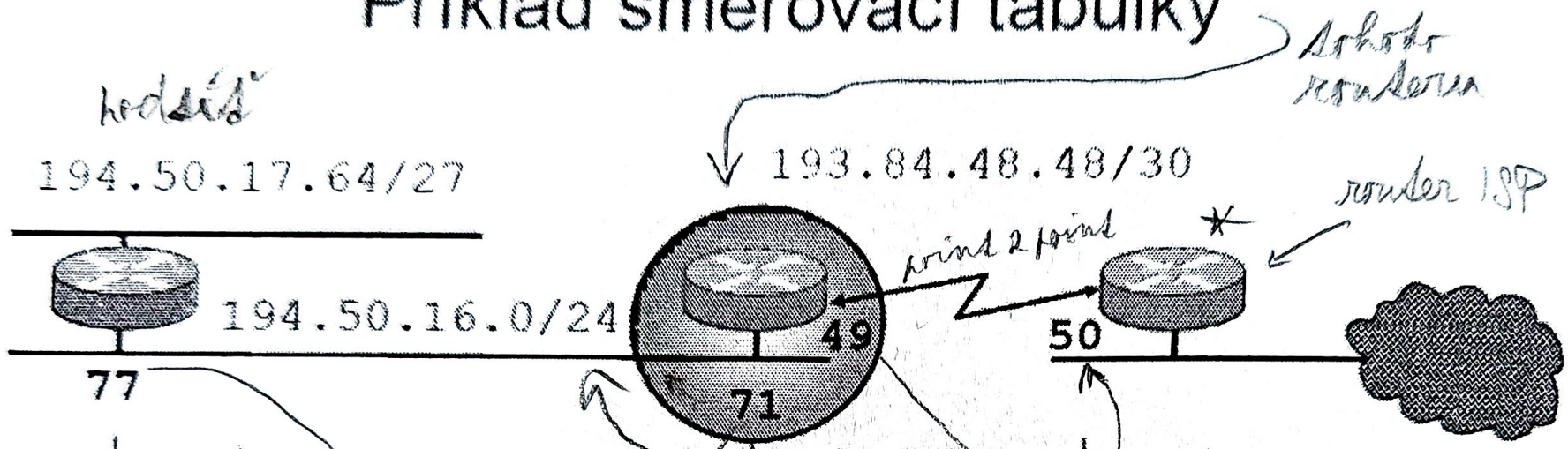
můj obraz —ano→ vrátit na vstup - jako by to právě přišlo ze sítě

↓

moje síť —ano→ pošli příjemci - přímý záznam

└—ne→ pošli next-hop routeru - nepřímý záznam

Příklad směrovací tabulky



Destination	Mask	Gateway	
127.0.0.0	255.0.0.0	127.0.0.1	direct, net
194.50.16.0	255.255.255.0	194.50.16.71	direct, net
193.84.48.50	255.255.255.255	193.84.48.49	direct, host
194.50.17.64	255.255.255.224	194.50.16.77	indirect, subnet
default	0.0.0.0 /0	193.84.48.50	default

- Konfigurace sítě
  - UNIX – IP adresa → ifconfig interface IP_adr [netmask maska]
    - defaultní router → route add default router
    - DHCP → dhclient interface
  - Windows – přes nastavení
    - v cmd je ifconfig, route

- Internet Control Message Protocol – ICMP
  - slouží pro posílání řídících informací pro IP např.:
    - Echo, Echo reply – testování dosažitelnosti počítače
    - Destination unreachable – nedostupný stroj / služba / síť, zakázaná fragmentace
      ↳ problém při routování / problém s UDP
    - Time exceeded – vypršel TTL ⇒ chyba v routování
    - Source quench – žádost o snížení rychlosti toku datagramů
    - Router solicitation, Router advertisement – vyhledávání routerů
    - Redirect – výzva ke změně záznamů v routovací tabulce
    - Parameter problem – chyba v záhlaví datagramu
  - používá IP datagramy, ale není to transportní protokol ⇒ ~OSI 3.5 ⇒ stojí mimo
  - ICMPv6 rozšířen → např. zprávy Neighbour Discovery Protocolu

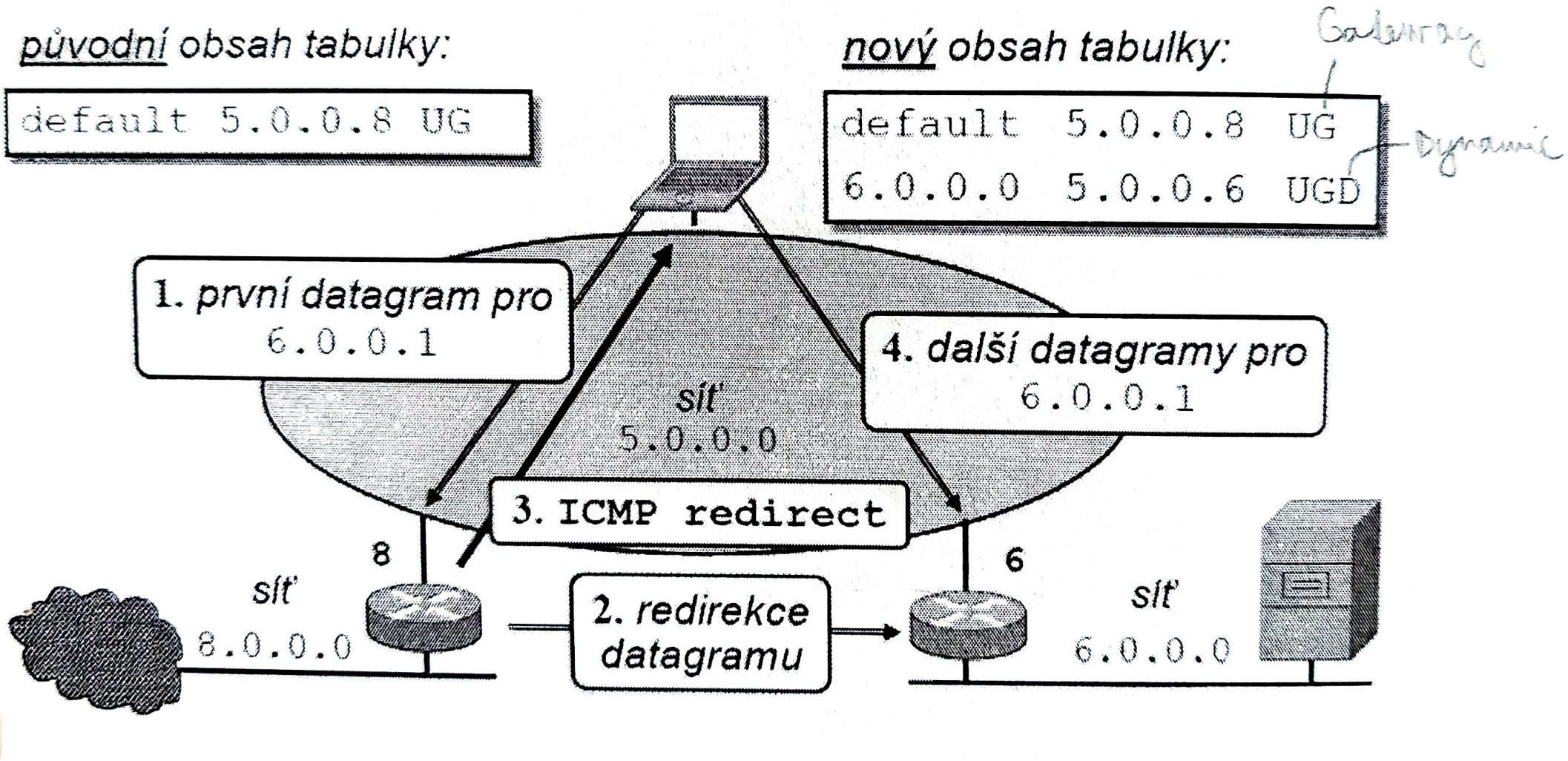
- Ping – základní program pro diagnostiku sítě
  → program s periodou 1s vysílá ICMP echo zprávy – dokud ho nepřerušíme!...
  → když zpráva dorazí na cílový stroj, tak odpoví ICMP echo reply
  ⇒ pokud dorazí zpět, tak ping vypíše řádek s časem cesty = round-trip time
  → nakonec vypíše round-trip min / avg / max / std-dev
  → na cílovém uzlu nemusí běžet žádný speciální program
  → nezaručuje dostupnost služeb – pouze síťové vrstvy
  → pokud odpověď nepřijde, tak nevyvrací dostupnost služeb – nějaký router může odstraňovat ping pakety

- **ICMP Time Exceeded**
  - používá pole TTL v IP záhlaví, aby nedocházelo k zacyklení paketů mezi routery
  - TTL udává # hopů, kterých se paket ještě může účastnit
  - ⇒ při každém hopu router sníží TTL ⇒ musí upravit header IP paketu
    if TTL == 0: zahodí paket a pošle odesílateli ICMP Time Exceeded zprávu
    else: pošle paket next-hop routeru
  - dnes je TTL defaultně 64

- **Diagnostika směrování** (nepřeložit adresy)
  - výpis routovací tabulky: `netstat -rn[n]` ≡ `route print`
  - ping - většinou nepomůže - moc různí
  - **traceroute** - 3x vyšle paket s TTL = 1 ⇒ zjistí 1. router
    3x vyšle paket s TTL = 2 ⇒ zjistí 2. router
    ⋮
    ⇒ zjistí kdy se přestane vrátit Time Exceeded ⇒ přibližně v jakého routeru je problém

- **Statické řízení routovacích tabulek**
  - počítač má někde uložené všechny potřebné záznamy a po bootu si je přidá do tabulky
  - např. když přes DHCP dostane adresu defaultního routeru pro danou síť
  - ⊖ nepružné při změnách sítě, problémy se subnettingem
  - ⊕ méně citlivé na problémy v síti, dostupné v libovolné síti
  - ⇒ vhodné pro jednodušší, stabilní sítě

- **ICMP Redirect**
  - umožňuje staticky řízeným routovacím tabulkám pokrýt složitější síť
  1) chceme poslat paket do sítě 6.0.0
     → default gateway 5.0.0.8
  2) paket dojde na router 5.0.0.8 a ten vidí, že ho má poslat routerem 5.0.0.6, neboť do stejné sítě, ze které přišel!
     ⇒ pošle mu ho, ale...
  3) pošle odesílateli (nám) ICMP redirect zprávu, abychom si do tabulky přidali nový záznam pro síť 6.0.0.0 ⇒ ten záznam bude s příznakem D

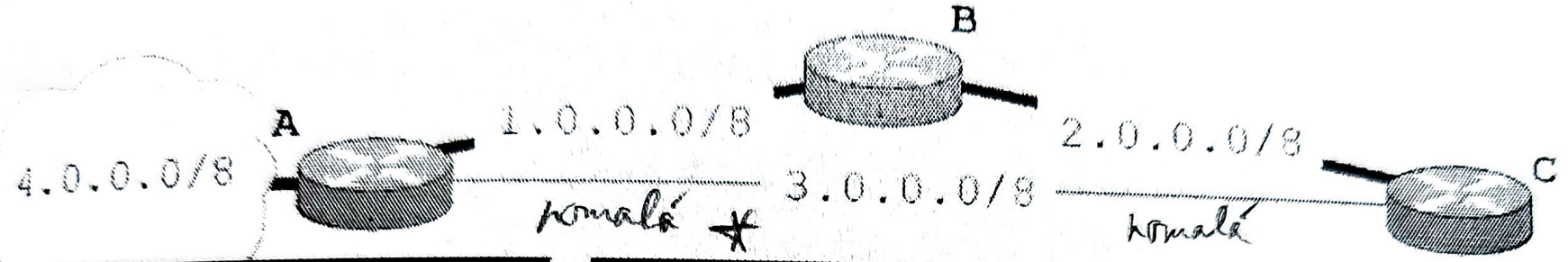

původní obsah tabulky:
default 5.0.0.8 UG
nový obsah tabulky:
default 5.0.0.8 UG
6.0.0.0 5.0.0.6 UGD
1. první datagram pro 6.0.0.1
síť 5.0.0.0
4. další datagramy pro 6.0.0.1
3. ICMP redirect
8
síť 8.0.0.0
2. redirekce datagramu
6
síť 6.0.0.0

- **Dynamické řízení směrovacích tabulek**
  - sousední routery si vyměňují info o síti pomocí routovacího protokolu
    → stanice se jím mohou řídit také, ale v režimu read-only
  - (+) jednoduché změny konfigurace, síť se sama opravuje, routovací tabulky se udržují automaticky
  - (−) citlivější na problémy, útoky
  - na počítači musí běžet program obsluhující protokol → BIRD – MFF
  - routovací protokoly pro lokální sítě se dělí na: ← interní routovací protokoly
    - Distance-vector protokoly – RIP
    - Link-state protokoly – OSPF

- **Distance vector protokoly**
  - router má u záznamů i „vzdálenost" ← a v dalším sloupci v tabulce [illegible]
  - svou tabulku periodicky posílá sousedům, ti si upraví svojí tabulku
  - (+) jednoduché, snadno implementovatelné
  - (−) pomalá reakce na chyby + chyba ve výpočtu 1 routeru ovlivní celou síť, ⇒ mohou vznikat smyčky
    omezený rozsah sítě, metrika nezohledňuje vlastnosti linek (rychlost, spolehlivost, ...)

  - **Routing Information Protocol – RIP**
    - nejstarší směrovací protokol
    - vlastnosti
      - metrikou je # routerů v cestě = hop count ↗ A—C přímo propojené routery mají metriku 1; A – D – C ⇒ cesta z A do C = 2
      - rozsah sítě je omezen na 15 hopů, 16 = inf
      - pro výpočet nejkratších cest používá Bellman-Fordův algoritmus
    - aktuálně verze 2 – UDP
      - umí subnetting včetně VLSM
      - obsahuje mechanismy na urychlení detekce chyb
    - starý ⇒ dostupný skoro všude
    - nepoužitelný pro velké, složité nebo dynamické sítě
    - metrika a kvalita linek
      → hop count nereflektuje vlastnosti linek ⇒ metriku některých pomalejších linek uměle zvětšíme
    → na začátku se všechny routery inicializují hodnotami pro přímo připojené sítě
    → když nějaký router rozesílá update a ostatní routery počítají metriky pro nové sítě, tak $v(A,C) = v(A,B) + v(B,C)$ ← přičtou k tomu vzdálenost sítě, ze které update přišel

* metrika linky mezi AC zvětšíme o 2 ⇒ r(A,C) = 3



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

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

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

r(A,C)

A rozesílá update:

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

r(A,B)

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

r(A,D)

B rozesílá update:

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

→ přes linku přišel C poslal update

přes jaký router

- counting to infinity - důvod, proč je nekonečno tak malé

→ Když v síti dojde k nějakému výpadku připojení do nějaké jiné sítě, tak na to sousední router A zareaguje nastavením metriky dané sítě na inf (16)

⇒ než mu ale vyprší perioda kdy bude moct rozeslat svoji tabulku, tak jiný router B rozešle update se starou vzdáleností ⇒ A si upraví tabulku ⇒ 16 se ztratí

⇒ po vypršení periody (30 s) A rozesílá update ⇒ B má ten záznam přes A, takže si ho zvětší ⇒ B rozesílá update ⇒ A += 1, ...

⇒ po nějaké době dojdeme k A = B = 16

⇒ kdyby inf byla větší, tak by tato nepřesnost tabulek byla větší

→ race condition = situace, kdy 2 nezávislé jevy mohou nastat v nevhodném pořadí

→ řešení v RIPv2

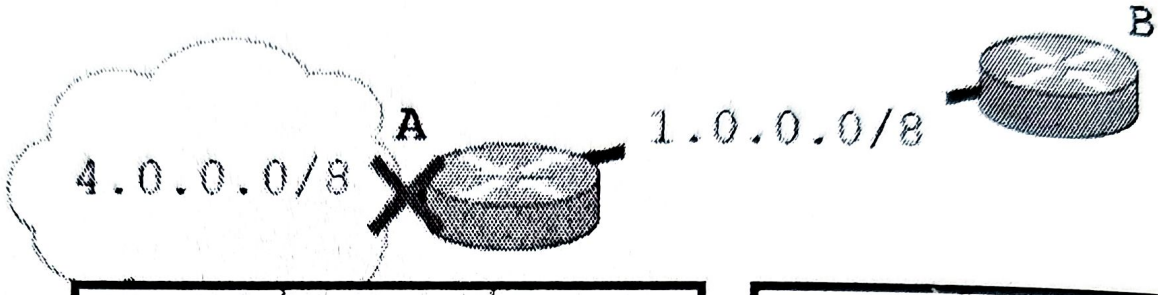
- Triggered updates - router při detekci problému rozešle update okamžitě
  ⇒ riziko race condition se sníží, ale neodstraní
- Horizon split - router neposílá sousedovi info o sítích, o kterých se dozvěděl od něj
  ⇒ problém to zabrání, ale pro každého souseda musí zvlášť připravit zvlášť
- Poison reverse - router sousedovi info o „jeho" sítích posílá, ale s metrikou 16

- Link state protokoly
  - každý router zná „mapu" celé sítě
  - routery si navzájem sdělují pouze stav svých linek ⇒ každý si sám modifikuje svoji mapu

⊖ výpočet mapy je náročnější na výkon CPU i paměť
při startu a na nestabilních sítích může výměna dat významně zatěžovat síť

⊕ pružná reakce na změny topologie, výměna dat pouze při změnách sítě
každý si mapu počítá sám, chyba nerušivě ostatní
síť je možné rozdělit na podsítě ⇒ ↑ rychlost výpočtu

- Open Shortest Path First - OSPF
  - k nalezení nejkratší cesty používá Dijkstrův algoritmus
  - používá hierarchický model sítě
    - oblast 0 = páteř sítě (backbone)
    - ostatní oblasti se připojují pouze na páteř ⇒ ∀ router zná mapu své oblasti a cestu k páteři
  - metriku je možné konfigurovat → implicitně to je path cost = Σ cen na cestě, cena = f(bandwidth)



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

Výpadek linky A/4:

4../8	-	16

perioda 30 s

B rozesílá update:

4../8	B	2+1

← přes B dám vědět krátkou cestu

A rozesílá update:

4../8	A	3+1

...

Stav po 7x30sec:

4../8	-	16

4../8	-	16

- **Autonomní Systémy**

Definice: blok sítí se společnou rozhraní politikou. ⇒ není zase tak těžké o přiřazení AS požádat

→ navržený 1982 pro snazší rozhraní na globální úrovni

→ identifikátor AS je jeho číslo, dřív 16 b, dnes 32 b.

→ v ČR: na počátku 2, dnes stovky

⇒ rozhraní mezi jednotlivými AS využívá Externí rozhraní protokoly - EGP - BGP

⇒ v AS Interní rozhraní protokoly - IGP - RIP, OSPF

→ dnes je nejrozšířenějším EGP Border Gateway Protocol - BGP

- **IP filtrování**

- router na perimetru má v konfiguraci uvedeno, jaký provoz je povolen a za jakých podmínek

- **přísná konfigurace: ven vybrané, dovnitř nic**

ven / dovnitř podle toho, kdo navazuje spojení

↳ podle toho kdo poslal SYN paket

  - OK pro protokoly s 1 datovým kanálem - HTTP, SMTP
  - problém u protokolů s více kanály - FTP, SIP

- **standardně: ven cokoli, dovnitř nic** / spojení navazuje server

  - naráží např u FTP s aktivním přenosem
  - nepoužitelné u protokolů s mnoha kanály - SIP

- řešení: SW na routeru musí částečně rozumět protokolu na aplikační vrstvě

- problém se službami uvnitř sítě, ke kterým by měli mít přístup všichni z internetu

  ↳ např www server, pošta

  → povolení výjimek je riskantní

  ⇒ lepší je zavést oddělený segment sítě, do které je přístup benevolentnější

  ⇒ DMZ = Demilitarizovaná zóna

- **Proxy servery** / pracují na hraničním routeru sítě

  - **Transparentní** - SW na routeru zachytí požadavek klienta, naváže svým jménem spojení na server a požadavek odešle

    - odpověď přijde zpět na router, ten ji uloží do cache a pošle původnímu žadateli (↗ pro další klienty)

  - **Netransparentní** - klienty je třeba nakonfigurovat, aby požadavky neposílali přímo, ale proxy serveru

    ⊕ proxy nemusí být nutně router ⊖ je nutná podpora v daném protokolu

- jsou významným bezpečnostním a výkonnostním prvkem sítě

  - umožňuje správě sítě kontrolovat činnost klientů, filtrovat škodlivý obsah zpráv
  - může věci z cache poslat dalším klientům

• Address Resolution Protocol – ARP

- umožňuje překládat mezi linkovými a síťovými adresami ethernet MAC <-> IP
- neznámé adresy se zjišťují broadcastem (šíří se pouze po linkovém segmentu) výzvou s cílovou MAC adresou ff:ff:ff:ff:ff:ff
- hledaný uzel (který se chová jako ARP server) odpoví unicastem ARP odpovědí s požadovanou MAC a přidá si info (IP a MAC) o žadateli do svojí ARP tabulky
- žadatel (ARP klient) si odpověď ukládá do ARP cache

! Nelze ověřit správnost odpovědi // ARP odpověď bez žádosti

! Gratuitous ARP – nevyžádané ARP – rychlejší změna dynamické sítě, riziko

- výpis ARP tabulky: arp -a

• Proxy ARP

A chce něco poslat B

1) host A posílá broadcastem ARP request s IP adresou B
2) router pozná, že dotaz k B nikdy nedoručí, takže sám pošle ARP reply se svojí vlastní MAC adresou
3) host A si k IP adrese B ve svojí ARP cache přiřadí MAC routeru
4) host A posílá data pro B s MAC adresou routeru

• Linková vrstva – OSI 2

- dělí se na 2 podvrstvy
  - Logical Link Control (LLC) – umožňuje různým protokolům síťové vrstvy přístup ke sdílenému médiu ⇒ multiplexing
  - Media Access Control (MAC) – řídí adresaci uzlů a přístup k médiu ⇒ kdo, kdy a jak může data odesílat a jak je přijímat

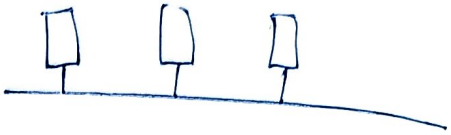
- TCP/IP ani se touto vrstvou nezabývá – součást síťového rozhraní

Definice: Síťový segment ≡ množina uzlů sdílejících sdílené médium.

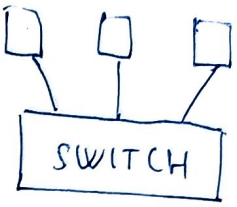
- PDU na linkové vrstvě = rámec / frame
  - liší se podle použitého média
  - obecně obsahuje: Synchronizační pole (~ start condition?),
    - hlavičku – MAC adresy, řídicí info LLC, datové pole
    - patičku – Frame Check Sequence (FCS) – detekce chyb

- **Typy síťových topologií**
  - topologie = uspořádání uzlů
    - fyzická – jak jsou propojeny? → kabelem
    - logická – jak spolu komunikují

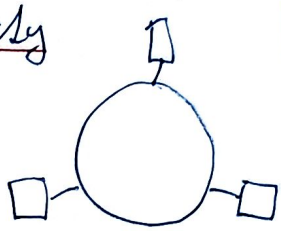
  - **Multipoint**
    - **Sběrnice** (Ethernet)
      - když se kabel přeruší, všechno se rozbije
      - kolize = když 2 počítače chtějí vysílat zároveň



    - **Hvězda** (ATM)
      - centrální prvek je dobře chráněný (konsynzum!)
      - je to switch - jednotlivé zásuvky = porty



    - **Kruh** (FDDI, Token-ring)
      - uzly jsou propojeny do kruhu



  - **Point-to-point**
    - **Přímé spojení kabelem** (RS-232)

    - **Propojení přes modemy**
      ↳ modem moduluje spojení aby šlo přenést po telefonní síti

    - **Bezdrátové propojení** - laser, radiové uzly
      ↳ WiFi má hvězdicovou topologii

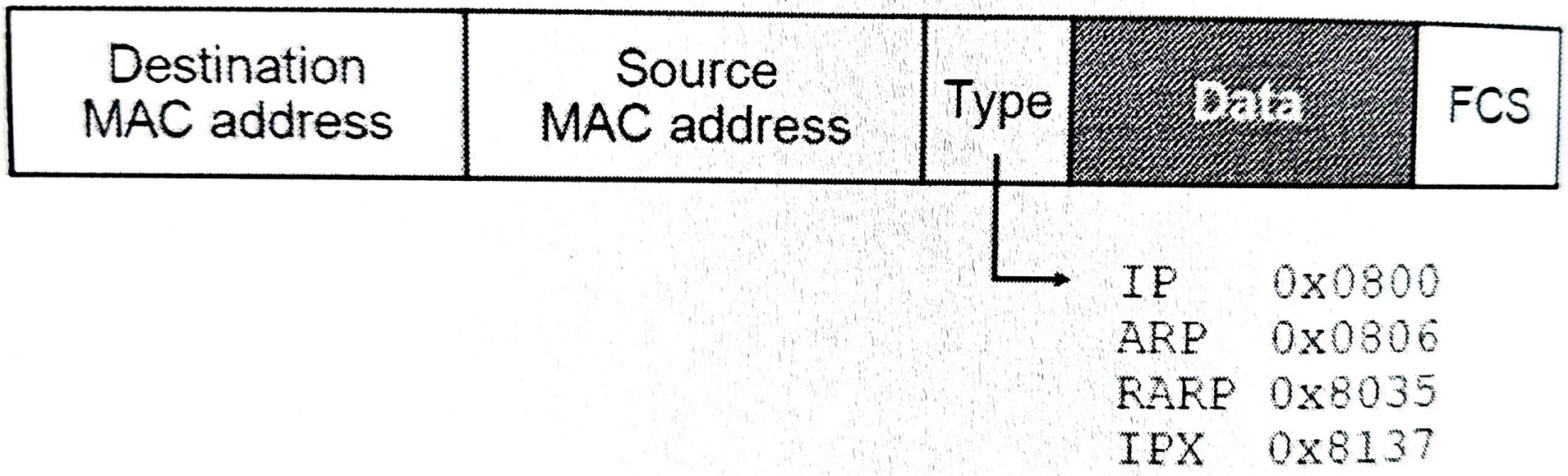
- **Způsob řízení přístupu k médiu**
  - **Multipoint**
    - **Deterministický** - vždy je určeno, kdo má vysílat ⇒ režie pomocí speciálního řídícího paketu
      - **Token-ring** - řídící prvek = speciální paket (token) - běhá po síti
        - vysílající odchytí token → přijímající ho rovněž vysílá
      - nebo je speciální řídící uzel, který ostatním uzlům posílá signály, kdyže mohou vysílat
    - **Nedeterministický** - Ethernet, musí se nějak řešit kolize
  - **Point-to-point** – halfduplex - riziko kolizí
    fullduplex - např. pro ethernet se zbavíme kolizí ⇒ ↑ propustnost

- Řešení kolizí
  - CSMA (Carrier Sense with Multiple Access)
    - uzel poslouchá „nosnou" a pokud není volno, čeká
  - CSMA/CD (Collision Detection) – Ethernet
    - během vysílání uzel taky poslouchá nosnou → detekuje případnou kolizi
    - při kolizi (obě) stanice ruší vysílání, upozorní ostatní, počkají určitou (náhodnou!) dobu a pokus opakuje
      → pokud opět dojde ke kolizi, tak se exponenciálně zvětšuje interval čekání
    - podmínka: doba vysílání rámce > doba šíření po segmentu (= kolizní okénko)
      ⇒ určuje max. délku segmentu sítě a min. velikost rámce
  - CSMA/CA (Collision Avoidance) – Wifi
    - Wifi používá hvězdicovou topologii – centrální prvek = Access point
      ⇒ každý přenos je vlastně point-to point
    - když je nosná volná, vysílá se celý rámec a čeká se na ACK
      ↳ pokud ACK nedorazí, nabájí se exponenciální čekání

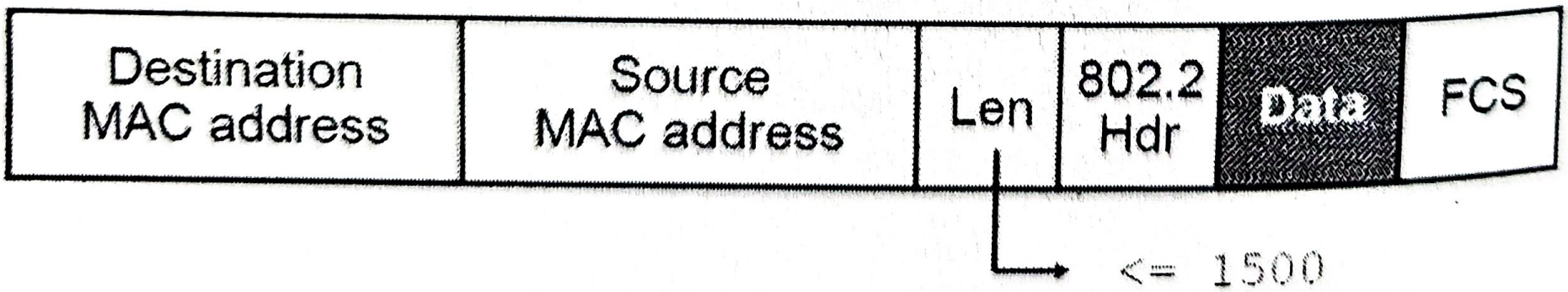
- Ethernet
  - historie – v Xerotu → standardizaci převzalo IEEE ⇒ 2 formáty: Ethernet II, IEEE 802.3
    ↳ nejsou veřejné! musí se koupit normy IEEE
  - vůdčí technologie pro lokální sítě
    - dokáží pružně reagovat na vývoj HW
  - řízení přístupu metodou CSMA/CD
    - při kolizi uzel vysílá „jam signál"
    - exponenciální čekání končí po 16 pokusech chybou
  - MAC adresy – 3 byty prefix výrobce, 3 byty adresa
    - dříve vypálená na kartě, dnes nastavitelná
    ↗ v 1 segmentu sítě nesmí být 2× stejná MAC
  - struktura ethernetového rámce
    - původní koncepce – 2B typ síťového protokolu
    - IEEE – délka + speciální LLC záhlaví

Struktura ethernetového rámce

Ethernet v2:



IEEE 802.3



- **Virtuální sítě** - VLAN

  - prostředek jak po 1 fyzické síti provozovat víc logických sítí
  - virtuální sítě jsou označeny 12 b. identifikátorem → VLANID
  - Ethernetový rámec se prodlouží o 32 b. dlouhý tag obsahující (protokol ID, QoS priorita a VLANID) = VLAN tag - vloží se před Type
  - virtuální sítě lze z pohledu koncových stanic provozovat transparentně
    ↳ switch ví, že na nějakém portu má VLAN s určitým VLANID
    → když přijde rámec s tím VLANID, tak tag odstraní a zbytek pošle do své VLAN
    → když přijde rámec z své VLAN, tak do něj přidá tag a pošle ho pryč
  - když nějaký uzel potřebuje mít přístup k rámcům ze všech sítí, tak se jeho port konfiguruje jako **trunk** a switch s rámci nic nedělá ⇒ obalování tagů nechá na uzlu
  - rámec se přidáním tagu prodlouží o 4B ⇒ musíme být schopni pracovat s rámci delšími než max. nebo nastavit max na max - 4B

- **Cyklický kontrolní součet** - CRC = Cyclic Redundancy Check

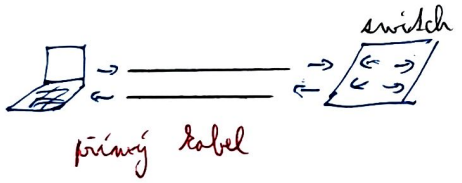
  - hashovací funkce používaná pro kontrolu konzistence dat - FCS, kontrolní součet IP hlavičky
  - → posloupnost bitů ⇒ polynom $10110 \sim x^4 + x^2 + x$
  - → vydělí se **charakteristickým polynomem** zadaného standardu, kolik b. má kontrolní pole
    ↳ pro CRC-16 to je $x^{16} + x^{15} + x^2 + 1$
  - → zbytek po dělení se převede zpět na bity a použije jako hash
  - jednoduchá HW implementace
  - velká síla - n-bit CRC detekuje na 100% chyby s lichým # bitů, chyby delší než n bitů

- **WiFi** = WLAN (Wireless LAN)

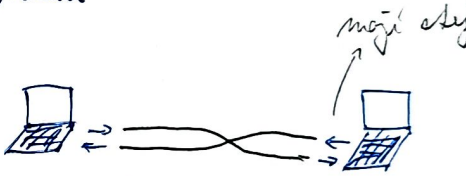
  - mnoho různých variant pod IEEE 802.11 < různá pásma 2.4 až 5 GHz / různé rychlosti 2 až 600 Mbps
  - **struktura sítě**
    - infrastrukturní - hvězdicových **Access points**
    - **ad-hoc** peer-to-peer sítě - komunikují spolu uzly z různých
  - SSID (Service Set ID) = řetězec pro rozlišení sítí
  - problém se zabezpečením !

- **Fyzická vrstva** – OSI 1
  - funkce: převod digitální info na analogovou a obráceně + přenos dat po konkrétním médiu
  - typy médií — metalické → el. pulzy
    optické → světelné pulzy
    bezdrátové → modulace vln
  - **baseband přenos** – přenáší přímo signál a kóduje ho
    ↳ Ethernet používá kód. Manchester 0:= falling edge, 1:= rising edge
  - **broadband přenos** – přenáší základní signál a moduluje ho – fázi, amplitudu, frekvenci

- **Nestíněná kroucená dvoulinka** – UTP = Unshielded Twisted Pair
  - dnes standardní kabel (metalický)
  - 4 páry Cu vodičů navzájem pravidelně zakroucené – snižuje to vyzařování i příjem elmg. záření
    ↳ 100 Mbps Ethernet používá jen 2 páry ⇒ je možné rozdělit
  - konektory: RJ 45
  - při připojení je třeba zohlednit povahu zařízení



přímý kabel



křížený kabel (crossover)

→ dnes většinou už není potřeba → používá se autodetekce MDI/MDIX

  - alternativa s dalším stíněním: STP

- **Optická vlákna** SiO_2
  - signál se šíří jako ~~viditelné~~ světlo křemíkovým vláknem
    → vysoké frekvence, velká bandwith (rychlost), malý útlum, žádné rušení
  - nevýhody: ↑ cena, náročnější manipulace (nelze ohýbat), nedotýkat do kabelu
  - druhy vláken
    - jednovidová (singlemode) – svítí se laserem ⇒ 1 paprsek, ↑ dosah + bandwith + cena
    - mnohovidová (multimode) – svítí se i LED

- **Segmentace sítě**
  - repeater – řeší větší dosah signálu
    – ve strukturované kabeláži hub
    } posílá dál všem
  - bridge – spojuje segmenty na linkové vrstvě
    – řeší větší propustnost (rozděluje kolizní doménu)
    ⇒ ↓ kolizí
    – ve strukturované kabeláži switch
    } pošle dál jen cílové MAC adrese

šíří se v ní limited broadcast

- full duplex ⇒ nejvyšší propustnost
- celá síť oddělená 1 routerem představuje 1 IP síť a také 1 broadcast doménu

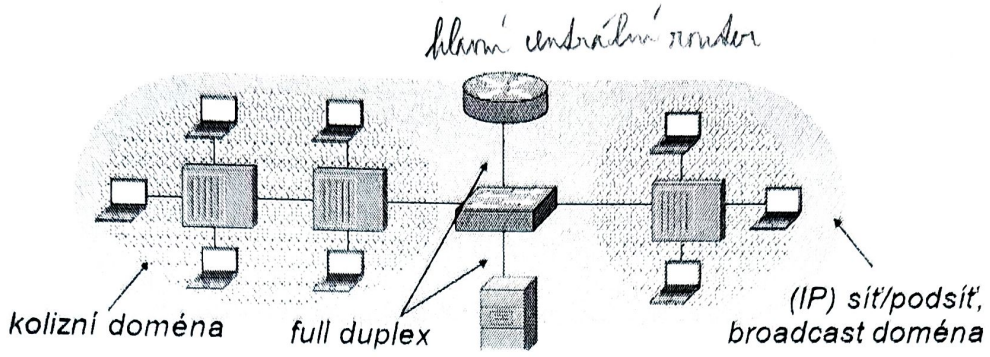
- **Learning bridge** – BUM (BUS)
  - režim práce switche, kdy si sám do své MAC tabulky přidává info o tom, za jakým portem jsou jaké adresy
  - když si zaplní celou tabulku, tak už bude všechny rámce posílat do správných portů
    ↳ s výjimkou broadcastů, neznámých unicastů a multicastů – ty bude posílat všem
    ⇒ BUS (Broadcast and Unknown Service) / BUM (Broadcast, Unknown and Multicast)

- **Spanning Tree Protocol** – STP
  - někdy chceme mít pro větší robustnost 2 segmenty propojené dvěma switchi
    ↳ kdyby ty switche fungovaly současně, tak learning bridge by selhal

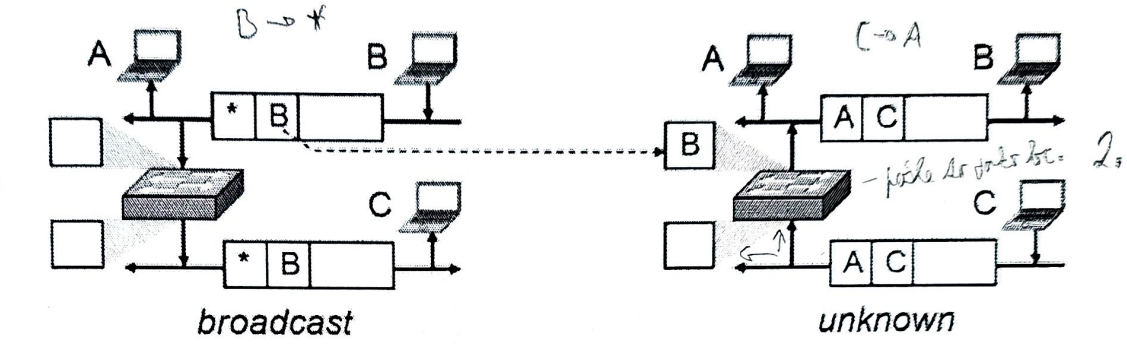
Důvod: graf je cyklický
Řešení: najít kostru = spanning tree

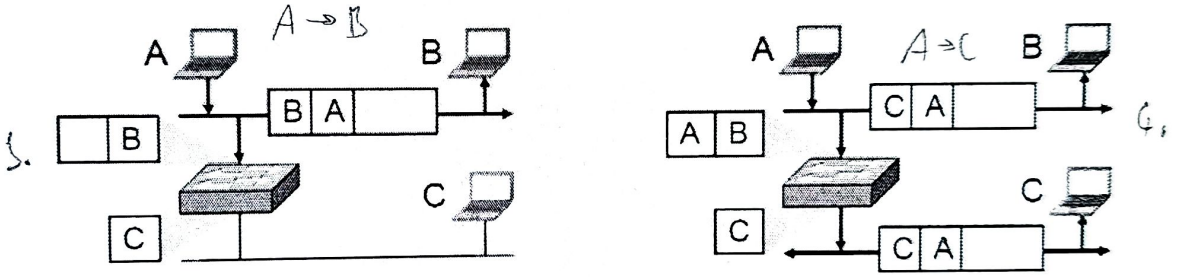
⇒ switche se musí dohodnout, který z nich bude v režimu forwarding a který v režimu blocking – monitorovat, jestli nedošlo k výpadku

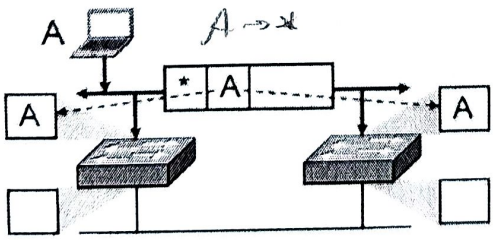
- STP má nezbytné timeouty, start může být pomalý
  ⇒ obvykle lze STA na portu potlačit (portfast) – je to na adminovi

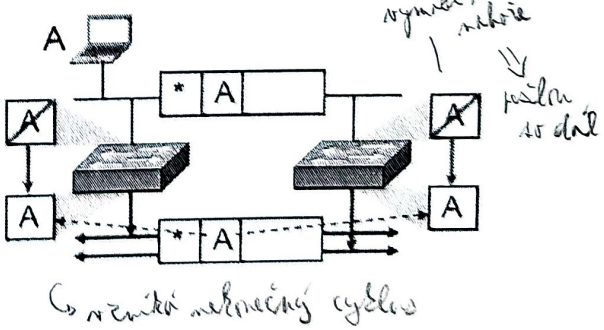


LEARNING BRIDGE









→ více bridgů