16. Internetové protokoly

Obsah

- Klient, server
- DNS (Domain Name System)
- IPv4 a IPv6, adresa
- URI/URL
- Paket, stream, datagram
- TCP/UDP port
- HTTP protokol
- Přenos souborů
- RFC (Request for Comments)

Klient, server

Klient-server je základní architektura komunikace na internetu, kde role jsou jasně rozdělené.

Server

- **Server** poskytuje služby nebo prostředky, které klienti využívají
- Pasivně čeká na požadavky od klientů (naslouchá na určité IP adrese a portu)
- Je schopen obsluhovat více klientů současně
- Typicky běží nepřetržitě
- Má obvykle statickou IP adresu

Klient

- **Klient** je aplikace nebo zařízení, které iniciuje komunikaci
- Aktivně zasílá požadavky na server
- Zpracovává odpovědi od serveru
- Může se připojit k různým serverům
- Typicky má dynamickou IP adresu

Průběh komunikace

1. Server naslouchá na konkrétní IP adrese a portu
2. Klient iniciuje spojení na tuto adresu a port
3. Server přijme spojení
4. Klient odešle požadavek
5. Server zpracuje požadavek a odešle odpověď
6. Klient zpracuje odpověď
7. Spojení může být ukončeno nebo udržováno pro další požadavky

DNS (Domain Name System)

Domain Name System (DNS) je hierarchický decentralizovaný systém pro překlad doménových jmen na IP adresy a naopak.

Princip funkce DNS

- Lidé si lépe pamatují doménová jména (např. `www.example.com`) než IP adresy (např. `93.184.216.34`)
- DNS překládá doménová jména na odpovídající IP adresy
- Usnadňuje změnu IP adresy serveru bez nutnosti informovat uživatele

Hierarchická struktura DNS

- **Kořenové DNS servery** (Root DNS servers)
- **Servery domén nejvyšší úrovně** (TLD servers) - `.com`, `.org`, `.net`, `.cz` atd.
- **Autoritativní DNS servery** - spravují záznamy pro konkrétní domény
- **Lokální DNS servery** - poskytovatelé internetových služeb, firemní sítě

Proces vyhledávání DNS

1. Klient zadá doménové jméno (např. `www.example.com`)
2. Požadavek je odeslán na lokální DNS server
3. Pokud lokální server nemá odpověď v cache, ptá se kořenového DNS serveru
4. Kořenový server odkáže na TLD server pro `.com`
5. TLD server odkáže na autoritativní server pro `example.com`
6. Autoritativní server vrátí IP adresu pro `www.example.com`
7. Lokální server si uloží odpověď do cache a předá ji klientovi

Reverzní DNS (Reverse DNS)

- Opačný proces - převod IP adresy na doménové jméno
- Používá speciální doménu `in-addr.arpa` pro IPv4 a `ip6.arpa` pro IPv6
- IP adresa se zapisuje v obráceném pořadí (např. pro IP `192.168.1.1` je dotaz na `1.1.168.192.in-addr.arpa`)

Typy DNS záznamů

- **A** (Address) - mapování doménového jména na IPv4 adresu
- **AAAA** - mapování doménového jména na IPv6 adresu
- **CNAME** (Canonical Name) - alias pro jiné doménové jméno
- **MX** (Mail Exchange) - určuje e-mailové servery pro doménu
- **NS** (Name Server) - určuje autoritativní DNS servery pro doménu
- **PTR** (Pointer) - používá se pro reverzní DNS
- **SOA** (Start of Authority) - obsahuje administrativní informace o zóně
- **TXT** - textový záznam, může obsahovat libovolné informace

IPv4 a IPv6, adresa

IPv4

- 32-bitová adresa (4 byty)
- Zapisuje se jako čtyři dekadická čísla oddělená tečkami (např. `192.168.1.1`)
- Každé číslo má rozsah 0-255 (osmi bitové číslo)
- Celkový počet možných adres: 2^32 = přibližně 4,3 miliardy
- Vzhledem k růstu internetu již nedostatek IPv4 adres
- Některé rozsahy jsou rezervovány pro speciální účely:
  - Privátní sítě: `10.0.0.0/8`, `172.16.0.0/12`, `192.168.0.0/16`
  - Localhost: `127.0.0.0/8` (typicky `127.0.0.1`)

IPv6

- 128-bitová adresa (16 bytů)
- Zapisuje se jako osm skupin čtyř hexadecimálních číslic oddělených dvojtečkami (např. `2001:0db8:85a3:0000:0000:8a2e:0370:7334`)
- Zjednodušený zápis:
  - Vynechání počátečních nul v každé skupině (`2001:db8:85a3:0:0:8a2e:370:7334`)
  - Nahrazení jedné nejdelší skupiny nul `::` (`2001:db8:85a3::8a2e:370:7334`)
- Celkový počet možných adres: 2^128 = přibližně 3,4 × 10^38
- Prakticky nevyčerpatelný adresní prostor
- Nemá kontrolní součet hlavičky (na rozdíl od IPv4), což zrychluje směrování

Rozdíly mezi IPv4 a IPv6

- IPv6 má mnohem větší adresní prostor
- IPv6 má zjednodušenou hlavičku (efektivnější směrování)
- Vestavěná podpora pro zabezpečení (IPsec)
- Lepší podpora pro mobilitu
- Automatická konfigurace (nevyžaduje DHCP)
- Efektivnější směrování díky hierarchické struktuře adres

URI/URL

URI (Uniform Resource Identifier)

- **URI** je obecný identifikátor zdroje
- Nadmnožina zahrnující URL a URN
- Jednoznačně identifikuje zdroj
- Může, ale nemusí poskytovat informace o přístupu ke zdroji

URL (Uniform Resource Locator)

- **URL** je speciální typ URI, který kromě identifikace zdroje poskytuje i informace o jeho umístění a způsobu přístupu k němu
- Obsahuje protokol, doménové jméno nebo IP adresu, port (volitelně) a cestu k zdroji

Struktura URL

`protokol://uživatel:heslo@doména:port/cesta?parametry#fragment`

- **Protokol** - způsob přístupu ke zdroji (http, https, ftp, mailto, file, ...)
- **Uživatel a heslo** - volitelná autentikace (zřídka používaná z bezpečnostních důvodů)
- **Doména** - doménové jméno nebo IP adresa serveru
- **Port** - volitelný, specifikuje port na serveru (výchozí porty: HTTP - 80, HTTPS - 443)
- **Cesta** - umístění zdroje na serveru
- **Parametry** - volitelné dodatečné informace pro server (formát klíč=hodnota, oddělené &)
- **Fragment** - volitelný identifikátor konkrétní části zdroje (např. pozice na webové stránce)

Příklady URL

- `https://www.example.com` - základní URL s protokolem a doménou
- `https://www.example.com:8443` - URL s nestandardním portem
- `https://www.example.com/path/to/resource.html` - URL s cestou k souboru
- `https://www.example.com/search?q=query&lang=cs` - URL s parametry
- `https://www.example.com/page.html#section2` - URL s fragmentem

Paket, stream, datagram

Paket

- **Paket** je základní jednotka dat přenášená v počítačových sítích
- Obsahuje hlavičku (metadata) a data (payload)
- Hlavička obsahuje informace potřebné pro doručení (zdrojovou a cílovou adresu, kontrolní součty, ...)
- Má omezenou velikost (typicky maximálně 1500 bytů v Ethernetu - MTU)
- Umožňuje efektivní routování a paralelní přenos dat v síti

Stream

- **Stream** je souvislý tok dat
- Data jsou přenášena jako sekvence paketů
- Na rozdíl od jednotlivých paketů představuje souvislý proud dat

- Typicky používán v TCP, kde se o správné řazení paketů stará protokol
- Příjemce vidí data jako nepřerušovaný proud, i když fyzicky jsou přenášena po částech

Datagram

- **Datagram** je samostatná jednotka dat, která nese všechny informace potřebné pro doručení
- Podobný paketu, ale používá se především v kontextu UDP
- Na rozdíl od TCP streamů, datagramy jsou nezávislé a mohou být doručeny v libovolném pořadí
- Může být ztracen, duplikován nebo doručen mimo pořadí bez automatické opravy

Srovnání

Vlastnost	Paket	Stream	Datagram
Velikost	Omezená (MTU)	Neomezená	Omezená
Souvislost	Samostatný	Sekvence paketů	Samostatný
Pořadí	Nezaručeno	Zaručeno (TCP)	Nezaručeno
Doručení	Nezaručeno	Zaručeno (TCP)	Nezaručeno
Protokol	IP	TCP	UDP

TCP/UDP port

Port

- **Port** je 16-bitové číslo (0-65535), které identifikuje konkrétní aplikaci nebo službu na zařízení
- Umožňuje současnou komunikaci více aplikací přes jednu IP adresu
- Kombinace IP adresy a portu jednoznačně identifikuje konkrétní službu na konkrétním zařízení

Kategorie portů

- **Dobře známé porty** (0-1023)
  - Standardizované porty pro běžné služby
  - Vyžadují privilegovaný přístup pro naslouchání
  - Příklady: HTTP (80), HTTPS (443), FTP (21), SSH (22), DNS (53), SMTP (25)
- **Registrované porty** (1024-49151)
  - Pro méně běžné aplikace a služby
  - Mohou být zaregistrovány u IANA
- **Dynamické/privátní porty** (49152-65535)
  - Pro dočasné použití (např. klientské porty)
  - Nejsou registrovány u IANA

TCP (Transmission Control Protocol)

- Spojově orientovaný protokol (navazuje spojení)
- **Spolehlivý** - zajišťuje doručení všech paketů
- **Dodržuje pořadí** - pakety jsou doručeny ve správném pořadí
- **Kontroluje chyby** - detekuje a opravuje chyby v přenosu
- **Řídí tok dat** - přizpůsobuje rychlost přenosu pro předcházení zahlcení
- Vhodný pro aplikace vyžadující spolehlivost (webové stránky, e-mail, přenos souborů)
- Vyšší režie než UDP

Funkce TCP: 1. **Navázání spojení** (three-way handshake: SYN, SYN-ACK, ACK) 2. **Přenos dat** s potvrzením přijetí 3. **Opětovné odeslání** ztracených nebo poškozených paketů 4. **Řízení toku** dat pro optimální využití sítě 5. **Řízení zahlcení** sítě 6. **Ukončení spojení** (four-way handshake: FIN, ACK, FIN, ACK)

UDP (User Datagram Protocol)

- Bezstavový protokol (nenavazuje spojení)
- **Nespolehlivý** - nezaručuje doručení datagramů
- **Nedbá na pořadí** - datagramy mohou být doručeny v jiném pořadí
- **Nekontroluje chyby** (kromě základního kontrolního součtu)
- **Neřídí tok dat** - odesílá datagramy bez ohledu na stav sítě
- Nízká režie a latence
- Vhodný pro aplikace, kde je důležitější rychlost než spolehlivost (online hry, streamované média, VoIP)

Funkce UDP: 1. Jednoduchý přenos datagramů bez navazování spojení 2. Minimální režie (malá hlavička) 3. Rychlý přenos bez čekání na potvrzení

Srovnání TCP a UDP

Vlastnost	TCP	UDP
Spolehlivost	Vysoká	Nízká
Pořadí paketů	Zachováno	Nezaručeno
Kontrola chyb	Ano	Minimální
Rychlost	Pomalejší	Rychlejší
Režie	Vyšší	Nižší
Spojení	Stavové (s navázáním)	Bezstavové

HTTP protokol

HTTP (Hypertext Transfer Protocol) je aplikační protokol pro distribuované, spolupracující, hypermediální informační systémy. Je základem datové komunikace pro World Wide Web.

Vlastnosti HTTP

- **Bezstavový** - server si neuchovává informace o předchozích požadavcích klienta
- Model **požadavek-odpověď** (request-response)
- Textový protokol - požadavky a odpovědi jsou v čitelném formátu
- Běží typicky na portu 80 (HTTP) nebo 443 (HTTPS)

HTTP požadavek (request)

HTTP požadavek se skládá z: 1. **Metoda** (GET, POST, PUT, DELETE, ...) 2. **URL** cesta k požadovanému zdroji 3. **Verze protokolu** (HTTP/1.0, HTTP/1.1, HTTP/2, HTTP/3) 4. **Hlavičky** (headers) - metadata o požadavku 5. **Tělo** (body) - volitelná data (např. při POST požadavku)

Příklad HTTP požadavku:

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html,application/xhtml+xml
```

HTTP odpověď (response)

HTTP odpověď se skládá z: 1. **Verze protokolu** 2. **Stavový kód** (200 OK, 404 Not Found, 500 Internal Server Error, ...) 3. **Stavová zpráva** 4. **Hlavičky** - metadata o odpovědi 5. **Tělo** - požadovaná data (HTML stránka, obrázek, JSON, ...)

Příklad HTTP odpovědi:

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2022 22:38:34 GMT
```

```
Content-Type: text/html; charset=UTF-8
Content-Length: 138

<!DOCTYPE html>
<html>
<head>
    <title>Example Page</title>
</head>
<body>
    <h1>Hello, World!</h1>
</body>
</html>
```

HTTP metody

- **GET** - získání zdroje
- **POST** - odeslání dat ke zpracování
- **PUT** - nahrazení cílového zdroje zaslanými daty
- **DELETE** - odstranění zdroje
- **HEAD** - podobné jako GET, ale vrací pouze hlavičky
- **OPTIONS** - zjištění podporovaných metod pro daný zdroj
- **PATCH** - částečná úprava zdroje

Stavové kódy HTTP

- **1xx** (Informační) - požadavek přijat, pokračuje se ve zpracování
- **2xx** (Úspěch) - požadavek byl úspěšně přijat a zpracován
  - 200 OK - standardní odpověď pro úspěšný požadavek
  - 201 Created - požadavek byl úspěšný a byl vytvořen nový zdroj
- **3xx** (Přesměrování) - je třeba provést další akci pro dokončení požadavku
  - 301 Moved Permanently - zdroj byl trvale přesunut
  - 302 Found - zdroj byl dočasně přesunut
- **4xx** (Chyba klienta) - chyba na straně klienta
  - 400 Bad Request - server nemůže zpracovat požadavek kvůli chybě klienta
  - 401 Unauthorized - vyžadována autentizace
  - 403 Forbidden - server rozumí požadavku, ale odmítá ho provést
  - 404 Not Found - požadovaný zdroj nebyl nalezen
- **5xx** (Chyba serveru) - server nemohl splnit platný požadavek
  - 500 Internal Server Error - obecná chyba serveru
  - 503 Service Unavailable - server momentálně nemůže zpracovat požadavek

HTTPS

HTTPS (HTTP Secure) je zabezpečená verze HTTP s použitím šifrování pomocí TLS (Transport Layer Security, dříve SSL).

- Šifruje celou komunikaci mezi klientem a serverem
- Chrání před odposloucháváním a man-in-the-middle útoky
- Ověřuje identitu serveru pomocí certifikátů
- Běží na portu 443
- Je standardem pro bezpečné webové stránky

Vývoj HTTP

- **HTTP/0.9** (1991) - Jednoduchý, pouze GET metoda
- **HTTP/1.0** (1996) - Přidány hlavičky, metody a stavové kódy
- **HTTP/1.1** (1997) - Persistent connections, chunked transfer, virtual hosting
- **HTTP/2** (2015) - Binární, multiplexování, komprese hlaviček, server push
- **HTTP/3** (2022) - Založený na QUIC místo TCP, lepší podpora mobility

Přenos souborů

Pro přenos souborů po internetu existuje několik specializovaných protokolů, každý s vlastními výhodami a nevýhodami.

FTP (File Transfer Protocol)

- Standardní protokol pro přenos souborů
- Používá **oddělené kanály** pro řízení (port 21) a data (port 20 nebo dyn. port)
- Podporuje autentizaci uživatele
- Umožňuje procházení adresářové struktury
- Podporuje obousměrný přenos souborů
- **Nevýhody**: data jsou přenášena nešifrovaně, problémy s firewally

SFTP (SSH File Transfer Protocol)

- Protokol pro bezpečný přenos souborů přes SSH
- Šifruje veškerá data včetně autentizace
- Používá port 22 (stejný jako SSH)
- Nabízí plnou funkčnost pro práci se soubory (přenos, mazání, přejmenování, ...)
- Běžně používaný v Unixových systémech a pro správu serverů

SCP (Secure Copy Protocol)

- Jednoduchý protokol pro bezpečný přenos souborů založený na SSH
- Méně funkcí než SFTP, zaměřený pouze na kopírování souborů
- Používá port 22
- Jednoduchá syntaxe příkazů, vhodný pro skripty

FTPS (FTP Secure)

- FTP s dodatečnou vrstvou zabezpečení (SSL/TLS)
- Zachovává všechny funkce FTP, ale přidává šifrování
- Existují dva módy:
  - Explicitní (port 21, klient musí požádat o zabezpečení)
  - Implicitní (port 990, spojení je vždy zabezpečené)

HTTP/HTTPS

- Běžně používaný pro stahování souborů z webových serverů
- Jednoduchá implementace
- Podporuje resumování přenosů (s hlavičkou Range)
- Výhoda: téměř univerzální dostupnost, průchod firewally

WebDAV (Web Distributed Authoring and Versioning)

- Rozšíření HTTP protokolu pro správu souborů na webovém serveru
- Umožňuje vytváření, mazání a přesouvání dokumentů
- Podporuje uzamykání souborů pro současnou editaci
- Často používaný pro webové disky (cloud storage)

BitTorrent

- Decentralizovaný protokol pro peer-to-peer sdílení souborů
- Vhodný pro distribuci velkých souborů mnoha uživatelům
- Rozloží zátěž mezi všechny účastníky sítě
- Soubor je rozdělen na části, které lze stahovat od různých peerů současně

rsync

- Efektivní protokol pro synchronizaci souborů a adresářů
- Přenáší pouze rozdíly mezi soubory, šetří šířku pásma
- Může pracovat přes SSH pro zabezpečený přenos
- Široce používaný pro zálohování a synchronizaci serverů

RFC (Request for Comments)

RFC (Request for Comments) jsou formální dokumenty, které popisují specifikace, protokoly, postupy a koncepty související s internetem a počítačovými sítěmi.

Charakteristika RFC

- Publikovány organizací **IETF** (Internet Engineering Task Force)
- Slouží jako standard pro internetové protokoly a technologie
- Mají rostoucí číselnou posloupnost (RFC 1, RFC 2, ...)
- Jsou textové dokumenty, historicky s pevnou šířkou 72 znaků na řádek
- Jakmile je RFC publikováno, jeho text se nemění (je možné ho pouze nahradit novým RFC)

Proces tvorby RFC

1. **Internet-Draft** (I-D) - počáteční návrh, platný max. 6 měsíců
2. **Revize a diskuze** v rámci IETF
3. **Schválení** příslušnou pracovní skupinou
4. **Přidělení čísla RFC** a publikace

Stavy RFC

- **Standards Track** - určen k standardizaci
  - **Proposed Standard** - počáteční fáze
  - **Draft Standard** - stabilní a implementovaný (zrušeno v roce 2011)
  - **Internet Standard** - plně přijatý standard
- **Informational** - poskytuje informace, ale není standardem
- **Experimental** - experimentální protokol nebo funkce
- **Best Current Practice** (BCP) - osvědčené postupy
- **Historic** - zastaralý nebo nahrazený novým RFC

Významné RFC

- **RFC 791** - Internet Protocol (IP)
- **RFC 793** - Transmission Control Protocol (TCP)
- **RFC 768** - User Datagram Protocol (UDP)
- **RFC 2616** - Hypertext Transfer Protocol (HTTP/1.1)
- **RFC 7540** - Hypertext Transfer Protocol Version 2 (HTTP/2)
- **RFC 1035** - Domain Names - Implementation and Specification
- **RFC 959** - File Transfer Protocol (FTP)
- **RFC 5321** - Simple Mail Transfer Protocol (SMTP)
- **RFC 2818** - HTTP Over TLS (HTTPS)
- **RFC 2460** - Internet Protocol, Version 6 (IPv6)

Význam RFC

- Zajišťují interoperabilitu internetových technologií
- Poskytují vývojářům a implementátorům jasné specifikace
- Dokumentují historii vývoje internetu
- Usnadňují tvorbu nových protokolů a rozšíření
- Slouží jako referenční dokumentace

Jak číst RFC

- **Abstrakty** shrnují obsah a cíl RFC
- **Status** dokumentu určuje jeho postavení
- **Citace** odkazují na související RFC
- **Klíčová slova** (MUST, SHOULD, MAY, OPTIONAL) mají specifický význam definovaný v RFC 2119

Shrnutí

- **Klient-server** architektura tvoří základ internetové komunikace, kde server poskytuje služby a klient je využívá.
- **DNS** překládá doménová jména na IP adresy, což usnadňuje uživatelům navigaci na internetu.
- **IPv4** (32-bit) a **IPv6** (128-bit) jsou protokoly pro adresování zařízení v síti, přičemž IPv6 řeší problém vyčerpání IPv4 adres.
- **URI/URL** poskytují standardizovaný způsob identifikace a lokalizace zdrojů na internetu.
- **Pakety**, **streamy** a **datagramy** jsou různé způsoby organizace dat při přenosu po síti.
- **TCP** zajišťuje spolehlivý přenos dat se zachováním pořadí, zatímco **UDP** poskytuje rychlejší, ale méně spolehlivý přenos.
- **HTTP** je základní protokol pro web založený na modelu požadavek-odpověď.
- Pro **přenos souborů** existuje několik specializovaných protokolů jako FTP, SFTP, WebDAV a další.
- **RFC** jsou standardizační dokumenty, které definují internetové protokoly a zajišťují jejich kompatibilitu.