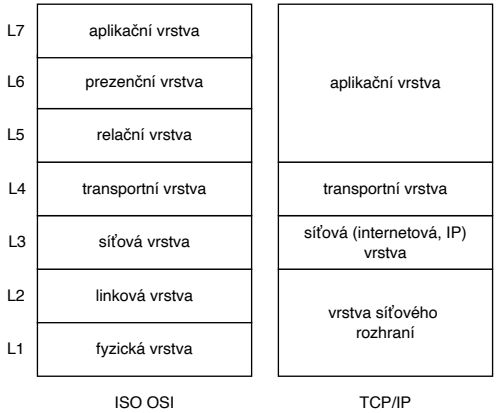
Počítačové sítě 1

aplikační vrstva

Martin Trnečka

Katedra informatiky
Univerzita Palackého v Olomouci

TCP/IP architektura



Aplikační vrstva

- tvořena aplikačními protokoly (popisují podobu dat)
  - obecně žádný společný prvek
- aplikační protokol = služba
- napojení na transportní vrstvu → socket (Socket API)
  - použití služeb nižších vrstev (běžné)
  - surová data (ruční vytvoření zprávy)
- bezpečnost → obecně nelze spoléhat na doplňkové zabezpečení na nižších vrstvách

Dynamická konfigurace hostitelského počítače

- Dynamic Host Configuration Protocol (DHCP)
  - IP adresa počítače
  - maska sítě
  - IP adresa výchozí brány
  - IP adresa jmenného (DNS) serveru
- klient-server služba
- klient port `udp/68`, server port `udp/67`

DHCP: Postup zajištění konfigurace

- klient odešle broadcast DHCP discover zprávu (zjišťuje zda existuje DHCP server)
  - klient nezná adresu odesílatele (svoji IP adresu) → `0.0.0.0`
  - klient nezná adresu příjemce (IP adresu serveru) → `255.255.255.255`
- server zachytí zprávu a pošle DHCP offer zprávu (nabídne konfiguraci)
  - server odpoví broadcast nebo unicast
  - nabídnutá IP je blokována
- pokud klient neobdrží DHCP offer (opakuje 4× s prodlevou 2 s, pak pauza 5 m)
- klient zašle DHCP request zprávu (vyžádá si nabídnutou IP)
- server zašle DHCP pack zprávu, klient po jejím obdržení získává IP adresu
- přidělená IP adresa má omezenou platnost (expirace)
  - při 50 % je opět poslána DHCP (request) zpráva požadující prodloužení expirace
  - obdržení DHCP pack → prodloužení expirace
  - server může prodloužení zamítnou (klient musí znovu zažádat o IP adresu)
- klient může vrátit přidělenou IP adresu (DHCP release zpráva)

DHCP: Prakticky

- je třeba určit rozsah použitelných IP adres
- doba expirace
- lze pevně určit vazbu mezi MAC a přidělenou IP (například pro síťovou tiskárnu)
- DHCP zprávy jsou standardně filtrovány na L3 zařízeních
- umístění serveru
  - DHCP server v lokální síti (obvykle domácí sítě, nepraktické u rozsáhlejších sítí)
  - DHCP server mimo lokální síť (broadcast neopustí lokální síť, nutný relay)
- v síti může být více DHCP $\rightarrow$ absence jednoho bodu selhání
- typicky různé rozsahy IP adres

DHCP: Bezpečnost

- UDP je nespolehlivý
  - DHCP má vlastní kontrolní součet
  - časovače a možnost opětovného poslání požadavku
- v síti může být více DHCP $\rightarrow$ bezpečnostní riziko
  - součástí konfigurace je i výchozí brána
  - útočník vytvoří nový DHCP server, který může předběhnout původní
  - skrze nový DHCP server může být nabídnuta kompromitovaná výchozí brána
  - blokování komunikace

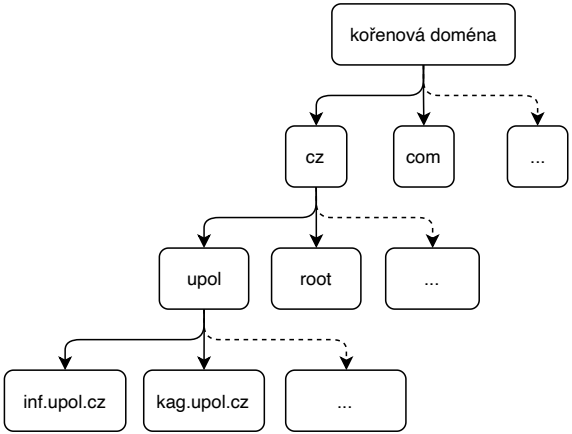
Domain Name System (DNS)

- IP adresy
  - kopírují fyzickou strukturu sítě → uživatelé vyžadují logickou organizaci
  - IP adresy jsou obtížně zapamatovatelné
- řešení: doménové jméno
- hierarchická organizace doménových jmen → stromová struktura
- decentralizace
- záznamy (vazba mezi doménovým jménem a IP adresou) uloženy na jmenných (DNS) serverech
- konkrétní příklad jmenné služby

Hierarchie doménových jmen

- doména, subdoména
- znak . oddělovač
- kořenová doména
  - nemá jméno (jméno je prázdný řetězec)
- top-level domény
  - generické – např. `com`, `net`, `info`, `edu`, ...
  - národní – např. `cz`, `sk`, `us`, ...
  - `.cz` vs `cz`
- doménové jméno
  - plně kvalifikované doménové jméno (FQDN), např. `phoenix.inf.upol.cz.` (tečka na konci)
  - částečně kvalifikované doménové jméno (PQDN), např. `phoenix` (resolver automaticky doplní suffix, včetně tečky)
- zóna = část záznamů spravovaných jmenným serverem

Hierarchie doménových jmen



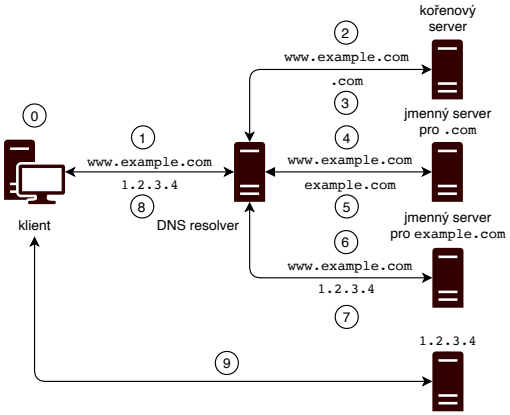
DNS servery

- kořenový server
  - reálně 13 serverů (jména a–m)
  - každý v mnoha kopiích, řešeno anycast
  - snížení celkové zátěže na DNS
  - neudržuje informace o záznamech v doménách
  - deleguje zodpovědnost na další servery (top-level domény)
- jmenný server
  - udržuje informace o záznamech (spravuje svoji zónu)
  - deleguje zodpovědnost na jmenné servery (subdomény)
  - má IP adresy kořenových serverů
  - primární server: vytváří, udržuje a aktualizuje soubor uchovávající zónu
  - sekundární server: pouze přebírá data (zone transfer) od primárního (případně dalších sekundárních)

Překlad doménového jména na IP adresu

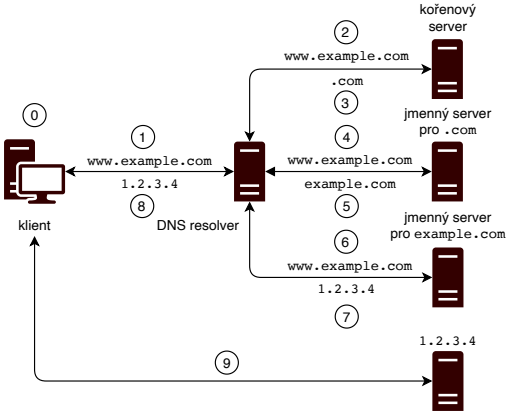
- klient-server služba
- protokol DNS
- port `udp/53` (pokud se odpověď vejde do 512 B) a `tcp/53`
- je třeba nalézt IP adresu k zadanému doménovému jménu
- řeší resolver
  - na klientovi část OS (služba poskytována aplikacím)
  - server (obvykle lokální jmenný server nebo veřejný resolver)
- dva typy dotazů
  - nerekurzivní – předání dotazu na jiný server, server vrací seznam dalších jmenných serverů
  - rekurzivní – klient požaduje a server vrací konečnou odpověď (server se stává řešitelem dotazu)

Příklad: Překlad doménového jména na IP adresu



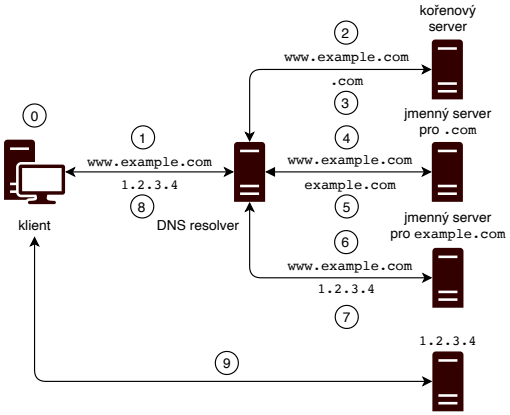
0. klient chce zjistit IP adresu ke jménu `www.example.com`, klient prohledá svoji cache zda nezná odpověď
1. klient pošle *rekurzivní dotaz* na DNS resolver (musí být nastaven), pokud DNS resolver zná odpověď (má ji v chache) pošle jí klientovi
2. pokud lokální jmenný server nezná odpověď pošle *nerekurzivní dotaz* na kořenový jmenný server

Příklad: Překlad doménového jména na IP adresu



3. kořenový jmenný server nezná odpověď, ale ví kdo je zodpovědný za doménu `com` v dotazu, pošle DNS resolveru jeho adresu
4. DNS resolver pošle *nerekurzivní dotaz* na jmenný server spravující doménu `.com`
5. jmenný server zodpovědný za doménu `.com` nezná odpověď, ale ví kdo je zodpovědný za doménu `example.com`, pošle DNS resolveru jeho adresu
6. DNS resolver pošle *nerekurzivní dotaz* na jmenný server spravující doménu `example.com`

Příklad: Překlad doménového jména na IP adresu



⑦ jmenný server spravující doménu `example.com` zná odpověď (`1.2.3.4`) a pošle ji DNS resolveru

⑧ DNS resolver předá `1.2.3.4` klientovi a uloží si údaje do cache

⑨ klient uloží `1.2.3.4` do cache a kontaktuje `1.2.3.4`

DNS: Poznámky

- vždy alespoň dva DNS servery (primární a sekundární)
  - všechny jsou vráceny v odpovědi (pořadí periodicky rotuje) → snížení zátěže
- typ odpovědi
  - autoritativní – poskytována primárním nebo sekundárním jmenným serverem
  - neautoritativní – odpověď vrácená z cache
- typy DNS serverů
  - primární, sekundární
  - cache
  - resolver (forwarder)

DNS: Reverzní překlad

- k IP adrese zjistit doménu
- typicky z důvodu bezpečnosti
- speciální top-level doména `arpa` se subdoménou `in-addr`
- postup
  - IP adresa se převrátí a přidá se `in-addr.arpa`
  - např. `158.184.80.13` → `13.80.194.158.in-addr.arpa`
  - provede se klasický překlad

DNS: Uložení záznamů

- záznamy uloženy v tzv. RR záznamech (větách)
- (vybrané) typy záznamů

typ	popis
`SOA`	informace o primárním serveru pro danou doménu (jméno, správce, TTL, ...)
`NS`	autoritativní jmenný server pro danou doménu
`A`	IPv4 adresa, uložení vazby doména IP adresa
`AAAA`	IPv6 adresa, uložení vazby doména IP adresa
`CNAME`	alias (alternativní jméno)
`PTR`	pro reverzní překlad
`MX`	poštovní servery

DNS prakticky

- software: Knot DNS vyvinut (cz.nic), Bind, ...
- chybná konfigurace → prodlevy při práce se sítí, nebo zdánlivá nefunkčnost
- nástroje
  - `nslookup`
  - `dig`
  - `dnswalk`
- veřejné DNS resolvery
  - `8.8.8.8`
  - `1.1.1.1`
  - `193.17.47.1` a `185.43.135.1` (pro ČR vhodnější)
- registrace domén

Ukázka interaktivní práce s `nslookup`

```
nslookup

> server
Default server: 158.194.80.254
Address: 158.194.80.254#53

> phoenix.inf.upol.cz
Name: phoenix.inf.upol.cz
Address: 158.194.80.13

> set type=ns
> inf.upol.cz
inf.upol.cz nameserver = ns1.inf.upol.cz.
inf.upol.cz nameserver = ns2.inf.upol.cz.

> set type=mx
> inf.upol.cz
inf.upol.cz mail exchanger = 12849 dx.spamfree.cz.
inf.upol.cz mail exchanger = 12801 ax.virusfree.cz.
inf.upol.cz mail exchanger = 12817 bx.virusfree.cz.
inf.upol.cz mail exchanger = 12833 cx.spamfree.cz.
```

Ukázka interaktivní práce s `nslookup`

```
> server 8.8.8.8
Default server: 8.8.8.8
Address: 8.8.8.8#53

> trnecka.inf.upol.cz
Server:   8.8.8.8
Address:  8.8.8.8#53

Non-authoritative answer:
trnecka.inf.upol.cz canonical name = phantom.inf.upol.cz.
Name: phantom.inf.upol.cz
Address: 158.194.80.10
Name: phantom.inf.upol.cz
Address: 158.194.92.12
Name: phantom.inf.upol.cz
Address: 158.194.80.9

> exit
```

Bezpečnost DNS

- doposud nebyl realizován úspěšný útok na (celosvětovou) DNS infrastrukturu
- zahlcení DNS serveru není jednoduché
  - cache, rotace v odpovědích, detekce automaticky generovaných dotazů
- hlavička DNS protokolu neobsahuje kontrolní součet → při použití UDP může znamenat bezpečnostní riziko
- odpovědi zneužívány pro DoS útoky (krátký dotaz, dlouhá odpověď)
- podvržení odpovědi → DNSSec
- DNS dotaz a odpověď jsou odposlechnutelné (nejsou šifrované)
  - např. v ČR využívané pro blokaci (lze triviálně obejít)
  - → DNS over TLS, DNS over HTTPS (velké rozdíly) případně VPN
  - stále problém důvěryhodnosti resolveru

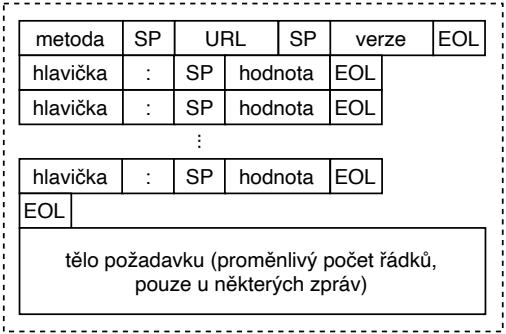
Secure shell (SSH)

- SSH-2 (SSH1-1 nebezpečné)
- port `tcp/22`
- čtyři komponenty
  - transportní protokol (SSH-TRANS) – vytváří bezpečné připojení nad TCP (šifrování a integrita dat)
  - autentizační protokol (SSH-AUTH) – autentizace klienta vůči serveru
  - spojovací protokol (SSH-CONN) – správa (multiplexing a demultiplexing) bezpečných spojení
  - aplikační protokol – data (využívají bezpečné kanály)
- SSH tunelování = použití bezpečného kanálu (např. telnet nebo SMTP skrze SSH), realizované přes forwardování portů (na klientovi a serveru)

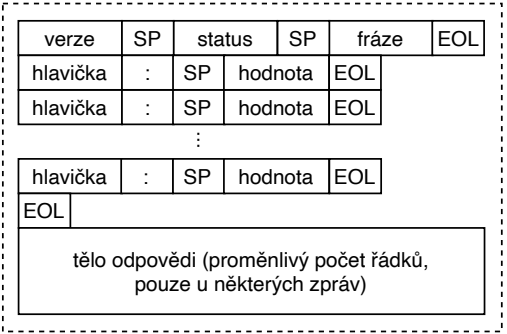
Hypertext Transfer Protocol (HTTP)

- přístup ke službě WWW
- HTTP/1, HTTP/1.1, HTTP/2, HTTP/3 (HTTP over QUIC)
- zaměříme se na HTTP/1.1
  - textová podoba (HTTP/2 je binární)
  - persistentní spojení (HTTP/1 je neperzistentní, vyžaduje TCP spojení pro každou zprávu)
  - absence šifrování → HTTPS (SSL/TLS)
- klient-server
- port `tcp`/80, port tcp/443 pro HTTPS
- bezstavový protokol (server neuchovává informace o klientovi)
  - cookies (uloženy na klientovi)
  - velikost a počet na doménu závisí na konkrétním klientovi

HTTP: Zprávy



request



response

- SP = mezera, EOL = nový řádek (CR + LF)

Protokol HTTP: Zjednodušené zprávy

- request

```
GET /pro-zajemce-o-studium HTTP/1.1
Host: www.inf.upol.cz
Connection: keep-alive
User-Agent: Mozilla/5.0
Accept-Language: cs
```

- response

```
HTTP/1.1 200 OK
Date: Mon, 29 Nov 2021 00:52:50 GMT
Server: Apache/2.4.10 (Debian)
Content-Type: text/html; charset=UTF-8

data
```

HTTP: Metody

metoda	popis
`GET`	požadavek na dokument (webovou stránku) na serveru
`HEAD`	pouze informace o dokumentu
`POST`	odeslání informace na server
`PUT`	odeslání dokumentu na server
`TRACE`	výpis příchozího požadavku
`CONNECT`	rezervováno
`DELETE`	odstranění dokumentu ze serveru
`PATCH`	změna dokumentu na serveru
`OPTIONS`	dostupné metody

HTTP: Hlavičky v request

metoda	popis
`user-agent`	identifikace klientského programu (prohlížeče)
`accept`	media
`accept-charset`	znaková sada
`accept-encoding`	kódování
`accept-language`	jazyk
`authorization`	oprávnění klienta
`host`	host a port klienta
`date`	aktuální datum
`upgrade`	preferovaný komunikační protokol
`cookie`	poslání cookie na server
`if-modifed-since`	podmíněný požadavek

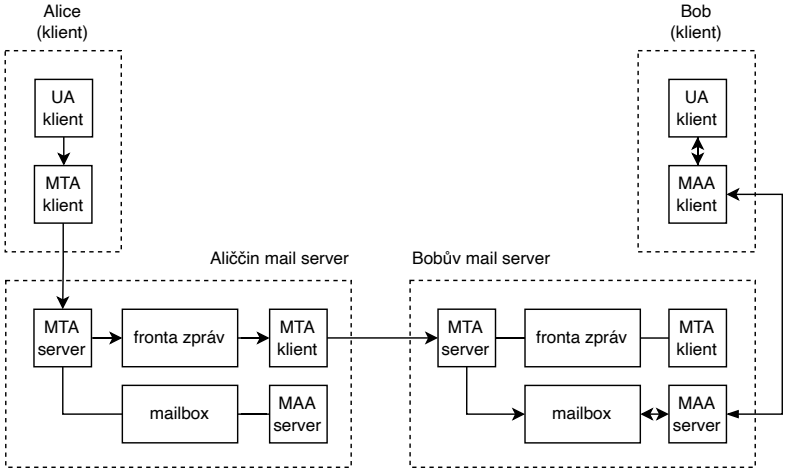
HTTP: Status v response

informativní		
kód	fráze	popis
100	continue	začátek požadavku přijat
101	switching	serveru vyhovuje změna protokolu
potvrzení		
kód	fráze	popis
200	OK	požadavek byl úspěšný
201	created	URL bylo vytvořeno
202	accepted	požadavek přijat, čeká na zpracování
204	no content	odpověď nemá tělo
přesměrování		
kód	fráze	popis
301	moved pernamently	dané URL bylo trvale změněno
302	moved temporarily	dané URL bylo dočasně změněno
303	not modified	dokument nebyl změněn
chyba klienta		
kód	fráze	popis
400	bad request	syntaktická chyba v požadavku
401	unauthorized	není dostatečné oprávnění
403	forbidden	přístup odepřen
404	not found	dokument nenalezen
405	method not allowed	daná metoda není podporována
406	not acceptable	požadovaný formát není akceptován
chyba serveru		
kód	fráze	popis
500	internal server error	chyba na straně serveru
501	not implemented	danou akci nelze provést
503	service unavailable	služba je dočasně nedostupná

HTTP: Hlavičky v response

metoda	popis
`date`	aktuální datum
`upgrade`	preferovaný komunikační protokol
`server`	informace o serveru
`set-cookie`	požadavek na uložení cookie
`content-encoding`	kódování
`content-language`	jazyk
`content-length`	délka dokumentu
`content-type`	typ medií
`location`	požadání klienta o poslaní požadavku na jinou URL
`accept-ranges`	server akceptuje daný rozsah bajtů
`last-mofified`	čas poslední změny

Elektronická pošta: Přehled

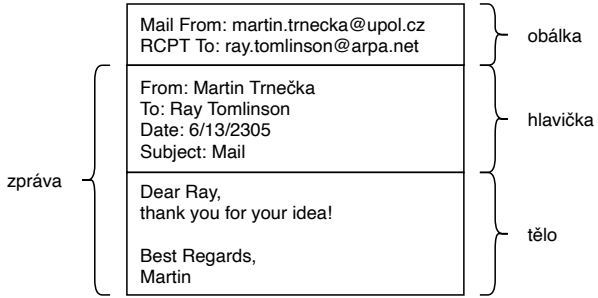


- UA = user agent, MTA = message transfer agent, MAA = message access agent
- web-based řešení využívá HTTPS pro spojení se serverem

Elektronická pošta

- UA = user agent
  - poštovní klient (aplikace)
- MTA = message transfer agent
  - klient-server
  - realizace odeslání zprávy
  - Extended Simple Mail Transfer Protocol (ESMTP)
- MAA = message access agent
  - klient-server
  - realizace stažení zprávy
  - Internet Mail Access Protocol 4 (IMAP4)

Elektronická pošta: Struktura zprávy



- lze přenášet pouze ASCII znaky
- Multipurpose Internet Mail Extension (MINE) – rozšíření hlavičky
  - neASCII znaky → zakódovány do ASCII
  - přílohy → zakódovány do ASCII → neefektivní

Elektronická pošta: Bezpečnost

- žádné zabezpečení na úrovni ESMTP, IMAP4
  - podvržení odesílatele
  - podvržení zprávy
- zabezpečení pomocí SSL/TLS na konci $\rightarrow$ „S“
- zabezpečení pomocí aplikačních protokolů
  - Pretty Good Privacy (PGP)
  - Secure MINE (SMINE)
- obálka obsahuje informace o zprávě a záznamy její cesty skrze síť

Decentralizace

- doposud vždy klient-server $\rightarrow$ centralizace
- decentralizace
  - např. Content Delivery Network (CDN), DNS $\rightarrow$ stále částečná centralizace
  - peer-to-peer (P2P) architektura

BitTorrent

- peer-to-peer protokol pro výměnu souborů
- klienti se podílejí na distribuci částečného souboru (*torrent*) výměnou *bloků* stejné velikosti (obvykle 256 kB)
- infrastrukturní server s *tracker* souborem
  - udržuje informace o klientech, kteří distribuují bloky daného torrent souboru
- výměna bloků je založena na obchodním algoritmu

BitTorrent: Zjednodušený postup sdílení

- Alice informuje tracker, že chce zahájit stahování torrent souboru, tracker ji eviduje a pošle ji podmnožinu již evidovaných klientů (mají bloky k výměně), Alice periodicky informuje tracker, že je aktivní
- Alice se pokusí navázat spojení s těmito klienty (klient, se kterým je navázáno spojení = soused, jejich počet se v průběhu mění (k trackeru se připojí se noví, kteří se stanou sousedy Alice, soused se odpojí, ...)
- Alice se pravidelně ptá sousedů na jejich seznam bloků, sousedé jí odpoví
- Alice si vybere (různé strategie, obvykle nejvzácnější), které bloky chce a pošle požadavek sousedům
- obchodní algoritmus (výměna požadovaných bloků): Alice udržuje 4 nejlepší (nejrychlejší) sousedy, kteří ji posílají bloky a posílá jim bloky na oplátku, přepočet cca každých 10 s, každých 30 s je náhodně zvolen další soused (Bob), kterému jsou poslány bloky, Alice se může stát nejlepší sousedkou Boba $\rightarrow$ Bob se může stát nejlepším sousedem Alice $\rightarrow$ princip „jak ty mě, tak já tobě“

Odbočka: CDN sítě

- distribuce dat v modelu klient-server
- centrální server + řady proxy serverů
- například: Google, Netflix, ...

Odbočka: Zastaralé protokoly

- telnet (port `tcp/23`)
- FTP (porty `tcp/20` a `tcp/21`)
- SMPT, POP3
- HTTP?
- zastaralé, ale jednoduché a elegantní
- při návrhu (programování) komunikace obvykle „znovuobjevování kola“

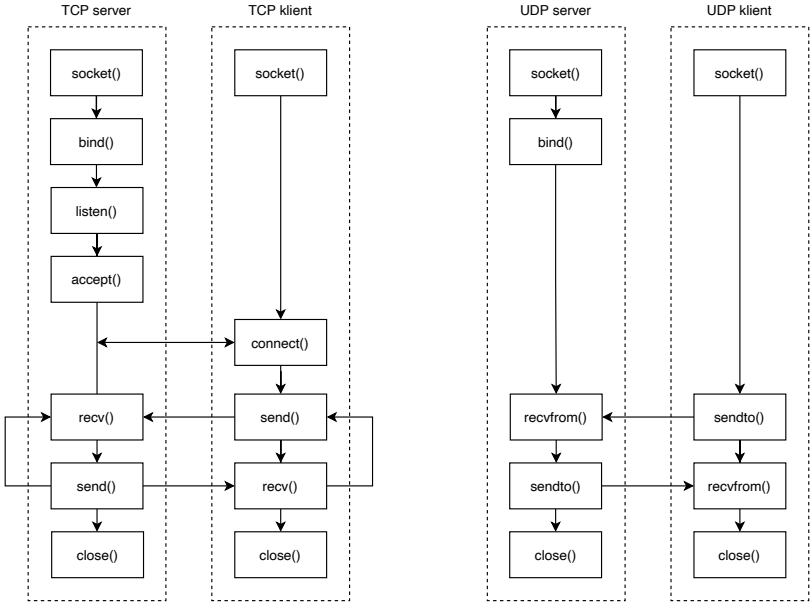
Další protokoly

- velké množství $\rightarrow$ ukázali jsme pouze zlomek
- správa sítě – Simple Network Management Protocol (SNMP)
- multimédia – Real-Time Transport Protocol (RTP), Real-Time Control Protocol (RTCP)
- synchronizace času – Network Time Protocol (NTP)
- soubory – Lightweight Directory Access Protocol (LDAP), Server Message Block (SMB)
- kvalita přenosu – Quality of Service (QoS)

Tvorba síťových aplikací

- napojení na síť pomocí socketu
- odesílání a příjem dat pomocí Socket API
  - ve většině jazyků (C/C++, Python, Java, C#, ...)
  - programátor obvykle potřebuje služby transportní vrstvy
  - lze obejít → konstrukce a odposlech (téměř) veškeré komunikace
- základní typy socketů
  - `SOCK_STREAM` (TCP)
  - `SOCK_DGRAM` (UDP)
  - `SOCK_SEQPACKET` (SCTP)
  - `SOCK_RAW`
- podpora různých typů adres, `AF_INET` = IPv4

Tvorba sťových aplikací: Socket API



Tvorba síťových aplikací (v jazyce Python)

- pouze ukázka
- `https://docs.python.org/3/library/socket.html`
- v příkladech klient-server budeme uvažovat následující kód

```
import socket as s

# nastavení
MSG_OK = b"ok"

# adresa a port serveru
server_address = "127.0.0.1"
server_port = 1227
```

- v ukázkách nejsou (pro jednoduchost) ošetřeny chybové stavy!
- absence konkurentnního zpracování požadavků → složitější, vyžaduje použití vláken

Tvorba síťových aplikací: UDP klient

```
# UDP klient
# otevření soketu, IPv4, UDP
client_socket = s.socket(s.AF_INET, s.SOCK_DGRAM)

# zpráva, je vyžadována sekvence bajtů
message = b"Hello, World!"

# poslání zprávy na server
client_socket.sendto(message, (server_address, server_port))

# čekání na přijetí zprávy od serveru, 2048 je velikost bufferu
message_from_server, server_address = client_socket.recvfrom(2048)
print(message_from_server)

# uzavření soketu
client_socket.close()
```

Tvorba síťových aplikací: UDP server

```
# UDP server
# otevření soketu, IPv4, UDP
server_socket = s.socket(s.AF_INET, s.SOCK_DGRAM)

# nastavení socketu
server_socket.bind((server_address, server_port))
print(f"The server is listening on port {server_port}.")

while 1:
    # čekání na příjem zprávy
    message, client_address = server_socket.recvfrom(2048)

    # poslání potvrzení o přijetí
    server_socket.sendto(MSG_OK, client_address)

    print(f"Message {message} received from {client_address}.")

server_socket.close()
```

Tvorba síťových aplikací: TCP klient

```
# TCP klient
# otevření soketu, IPv4, TCP
client_socket = s.socket(s.AF_INET, s.SOCK_STREAM)

# sekvence bajtů k přenosu
sequence = b"Hello, World!"

# navázání spojení se serverem
client_socket.connect((server_address, server_port))

# odeslání dat
client_socket.send(sequence)

# čekání na přijetí zprávy od serveru, 1024 je velikost bufferu
sequence_from_server = client_socket.recvfrom(1024)
print(sequence_from_server)

# uzavření soketu
client_socket.close()
```

Tvorba síťových aplikací: TCP server

```
# TCP server
# otevření soketu, IPv4, TCP
server_socket = s.socket(s.AF_INET, s.SOCK_STREAM)

# nastavení socketu
server_socket.bind((server_address, server_port))

# otevření portu
server_socket.listen(1)

while 1:
    # čekaní na spojení
    connection_socket, connection_address = server_socket.accept()

    # příjem dat, čeká se na zaplnění bufferu nebo konec spojení
    sentence = connection_socket.recv(1024)

    print(f"Sentence {sentence} received from {connection_address}.")

    # poslání sekvence klientovi
    connection_socket.send(MSG_OK)

    connection_socket.close()

server_socket.close()
```

Tvorba síťových aplikací: HTTP přes SSL/TLS

```
# SSL
import socket as s
import ssl

hostname = "www.inf.upol.cz"
port = 443

# vytvoření SSL/TLS vrstvy
context = ssl.create_default_context()

# vytvoření spojení
socket = s.create_connection((hostname, port))

# zabalení do SSL/TLS
sslSocket = context.wrap_socket(socket, server_hostname=hostname)

# HTTP požadavek, ruční konstrukce (existuje knihovna)
request = f"GET / HTTP/1.1\r\nHost: {hostname}:{port}\r\n\r\n"
sslSocket.send(request.encode())

# odpověď serveru
response = sslSocket.recv(4096)
print(response)
```

Tvorba síťových aplikací

- většina jazyků podporuje Socket API
  - C/C++ v UNIX a MacOS podpora přímo v C (BSD socket API), ve Windows Windows socket API (Winsock)
  - Java, součást `java.net.Socket`
  - C#, součást `System.Net.Sockets`
- přístup k nižším vrstvám TCP/IP
  - odlišné chování na různých architekturách
  - linkový rámec lze plnohodnotně manipulovat pouze na UNIXových systémech

Bezpečnost na aplikační vrstvě

- = bezpečnost jednotlivých služeb
  - chyby v knihovnách a software
  - zadní vrátka
  - prolomení šifrování (matematické, hrubá síla)
- autentizace
  - ukradení/odposlechnutí hesla
  - sociální inženýrství
- chránění vnitřní sítě
  - firewall
  - demilitarizovaná zóna (DMZ)
  - honeypots, traffic monitoring

Závěr: Poslední slide

- úvod do počítačových sítí (Internet)
  - principy fungování → zastarávají pomaleji
  - technologie → zastarávají rychleji
- úvod do (základní) bezpečnosti
- řada dalších zajímavých síťových záležitostí → KMI/POS2