

Aplikacijski sloj II: DNS, e-pošta, P2P i socketi

Predavanje 3 • kolegij Računalne mreže

doc. dr. sc. Sandi Baressi Šegota

Fakultet informatike Sveučilišta Jurja Dobrile u Puli

Sadržaj predavanja

- 1 DNS — imenički sustav
- 2 Elektronička pošta
- 3 P2P i distribucija sadržaja
- 4 Socket programiranje
- 5 Sažetak

DNS — imenički sustav

Ishodi učenja

Nakon ovog predavanja moći ćete:

- objasniti ulogu i hijerarhiju **DNS-a** te tijek razlučivanja imena;
- opisati rad **elektroničke pošte** (SMTP, IMAP/POP3, MIME);
- objasniti **P2P** distribuciju (BitTorrent) i njezinu skalabilnost;
- opisati **prilagodljivi streaming** i ulogu **CDN-a**;
- skicirati **socket** program (UDP i TCP).

Zašto DNS?

- ljudi pamte **imena** (`www.fipu.hr`), mreža koristi **IP adrese**
- **DNS** = distribuirana baza koja prevodi imena u adrese
- radi kao **aplikacijski protokol** na rubu, ali je temelj za sve ostale

Analogija

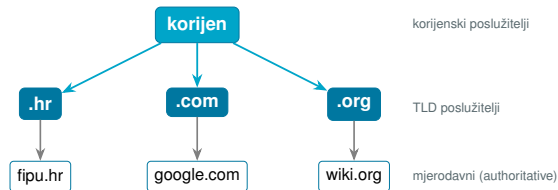
DNS je „telefonski imenik” Interneta: ime → broj (adresa).

Usluge DNS-a

- prijevod imena hosta u **IP adresu**
- **aliasi** (kanonsko ime i nadimci)
- usmjeravanje **e-pošte** (MX zapisi)
- **raspodjela opterećenja** (više adresa za isto ime)

Hijerarhija DNS-a

- **korijenski** poslužitelji
- **TLD** (.hr, .com, .org...)
- **mjerodavni** (authoritative) za domenu

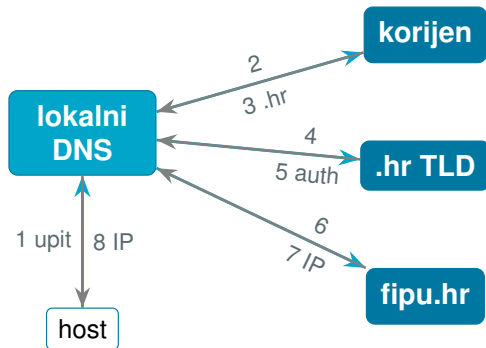


Vrste poslužitelja

- **Korijenski** — upućuju na TLD poslužitelje
- **TLD** — odgovorni za vršne domene (.hr, .com)
- **Mjerodavni** — čuvaju zapise konkretne organizacije
- **Lokalni (resolver)** — posrednik ISP-a; predmemorira odgovore

Razlučivanje imena

- **rekurzivni** upit: host → lokalni DNS
- **iterativni** upiti: lokalni DNS → korijen → TLD → auth
- odgovor se vraća i **predmemorira**



DNS predmemorija

- svaki poslužitelj **pamti** naučene odgovore (uz TTL)
- drastično smanjuje promet prema korijenu i TLD-u
- ubrzava razlučivanje za popularne domene

Kompromis

Dulji TTL = manje prometa, ali sporije širenje promjena adrese.

DNS zapisi

- **A** — ime → IPv4 adresa
- **AAAA** — ime → IPv6 adresa
- **NS** — mjerodavni poslužitelj za domenu
- **CNAME** — alias → kanonsko ime
- **MX** — poslužitelj e-pošte za domenu

DNS i transport

- koristi **UDP**, port 53 (brzo, mali upiti)
- pri velikim odgovorima ili zonskim prijenosima koristi **TCP**
- sigurnosna proširenja: **DNSSEC**, DNS-over-HTTPS/TLS

DNS u praksi: alati

- `nslookup www.fipu.hr` — brzo razlučivanje imena
- `dig +trace` — prikaz cijelog puta (korijen → TLD → auth)
- `host -t MX fipu.hr` — ispis MX zapisa

Savjet

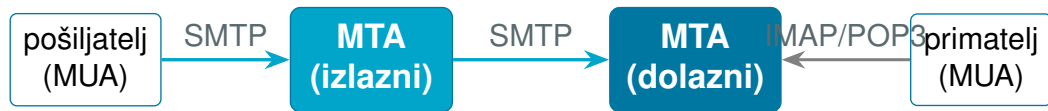
Ovi alati pokazuju i **TTL** i predmemorirane odgovore — korisno za dijagnostiku.

Sigurnost DNS-a

- **trovanje predmemorije** (cache poisoning) — ubacivanje lažnih zapisa
- **DNSSEC** — digitalni potpisi zapisa (integritet i autentičnost)
- **DoH/DoT** — DNS preko HTTPS-a/TLS-a (povjerljivost upita)

Elektronická pošta

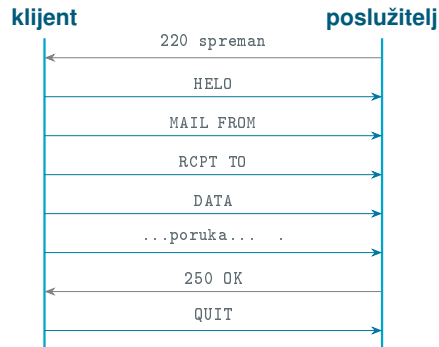
Komponente sustava e-pošte



- **MUA** (klijent), **MTA** (poslužitelji), **poštanski sandučić**
- slanje: **SMTP**; dohvat: **IMAP/POP3**

SMTP

- protokol za **slanje** pošte
- **push**: šalje prema poslužitelju
- TCP, port 25 (danas i 465/587)
- naredbe u ASCII-ju



Pristupni protokoli: IMAP vs POP3

IMAP	POP3
poruke ostaju na poslužitelju	preuzima i (obično) briše
mape, sinkronizacija	jednostavan
rad s više uređaja	jedan uređaj

Format poruke i MIME

- zaglavlja: From, To, Subject, Date
- tijelo odvojeno praznim retkom
- **MIME** omogućuje ne-ASCII sadržaj i privitke
- Content-Type, Content-Transfer-Encoding

Sigurnost e-pošte

- **SPF** — popis poslužitelja ovlaštenih slati za domenu
- **DKIM** — digitalni potpis poruke
- **DMARC** — politika i izvještavanje uz SPF/DKIM
- **TLS** — enkripcija prijenosa među poslužiteljima

Cilj

Smanjiti **lažiranje pošiljatelja** (spoofing) i **neželjenu poštu**.

Web-pošta i moderni klijenti

- pristup pošti kroz **preglednik** (HTTP/HTTPS) uz IMAP u pozadini
- sadržaj i privitci ostaju na poslužitelju, dostupni s više uređaja
- integracija s kalendarom, kontaktima, pretraživanjem

P2P i distribucija sadržaja

Skalabilnost: klijent–poslužitelj vs P2P

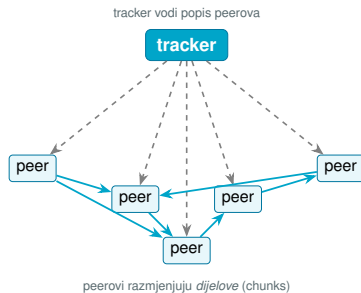
- kod **klijent–poslužitelj** kapacitet distribucije ograničava poslužitelj
- kod **P2P** svaki novi peer donosi i **vlastiti upload**
- vrijeme distribucije velikoj publici raste puno sporije

Poanta

P2P se dobro skalira jer resursi rastu s brojem sudionika.

BitTorrent

- datoteka podijeljena na **dijelove** (chunks)
- **tracker** vodi popis peerova
- peerovi razmjenjuju dijelove izravno



Mehanizmi BitTorrenta

- **rarest first** — prvo dohvati najrjeđe dijelove (ravnoteža)
- **tit-for-tat** — šalji onima koji šalju tebi (poticaj)
- **optimistic unchoke** — povremeno isprobaj nove peerove

Rezultat

Samoorganizirajuća, otporna i skalabilna distribucija bez središnjeg poslužitelja podataka.

Streaming: izazovi

- velika publika, **heterogena** propusnost i uređaji
- promjenjivi mrežni uvjeti tijekom reprodukcije
- cilj: **neprekidna** reprodukcija uz najbolju moguću kvalitetu

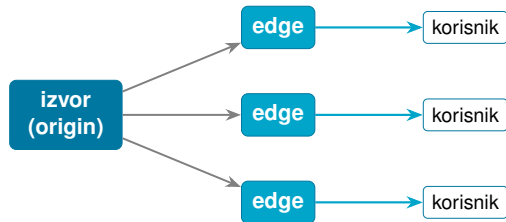
Prilagodljivi streaming (DASH)

- video u **segmentima** i više **kvaliteta**
- **manifest** (MPD) opisuje razine
- klijent bira kvalitetu prema **trenutnoj propusnosti**



Mreže za isporuku sadržaja (CDN)

- kopije sadržaja na **rubnim** poslužiteljima
- korisnik se usmjerava na **najbliži** edge
- manje kašnjenje, rasterećen izvor



korisnik se preusmjerava na *najbliži* edge

Kako CDN bira poslužitelj?

- najčešće preko **DNS-a**: ime sadržaja razlučuje se u adresu bliskog edge-a
- uzima u obzir **geografiju**, opterećenje i mrežnu blizinu
- primjeri: veliki streaming i web servisi

Vrijeme distribucije: intuicija

- **klijent–poslužitelj**: vrijeme raste **linearno** s brojem korisnika (izvor je usko grlo)
- **P2P**: nakon početnog širenja, peerovi pomažu jedni drugima
- za veliku publiku P2P dramatično skraćuje ukupno vrijeme

Zašto CDN? Koristi

- **manje kašnjenje** — sadržaj je blizu korisnika
- **rasterećenje izvora** — edge poslužuje većinu zahtjeva
- **otpornost** — raspodijeljena infrastruktura podnosi vršna opterećenja
- **ušteda** u prometu prema izvoru

Socket programiranje

Socket API

- **socket** = sučelje između aplikacije i transporta
- dva tipa: **UDP** (bez veze) i **TCP** (s vezom)
- poslužitelj „sluša”, klijent „se spaja” ili „šalje”

UDP socketi

- nema uspostave veze
- svaka poruka nosi **odredišnu adresu**
- jednostavno, ali bez jamstva dostave



UDP: skica poslužitelja (Python)

server_udp.py

```
s = socket(AF_INET, SOCK_DGRAM)
s.bind(("", 12000))
while True:
    data, addr = s.recvfrom(2048)
    s.sendto(data.upper(), addr)
```

TCP socketi

- **uspostava veze** prije prijenosa
- poslužitelj: **welcome** socket + **connection** socket po klijentu
- pouzdan tok bajtova

klijent
socket()
connect()
send()/recv()
close()

TCP veza

poslužitelj
socket()
bind()
listen()
accept() # nova veza
send()/recv()
close()

TCP: skica poslužitelja (Python)

server_tcp.py

```
s = socket(AF_INET, SOCK_STREAM)
s.bind(("", 12000)); s.listen(1)
while True:
    conn, addr = s.accept()
    msg = conn.recv(1024)
    conn.send(msg.upper()); conn.close()
```

UDP vs TCP socketi

UDP socket	TCP socket
bez veze	uspostava veze
<code>sendto/recvfrom</code>	<code>connect/accept</code> , <code>send/recv</code>
adresa uz svaku poruku	adresa jednom, pri spajanju
brz, bez jamstva	pouzdan tok bajtova

UDP: skica klijenta (Python)

client_udp.py

```
s = socket(AF_INET, SOCK_DGRAM)
s.sendto(poruka, ("host", 12000))
odg, addr = s.recvfrom(2048)
print(odg); s.close()
```

TCP: skica klijenta (Python)

client_tcp.py

```
s = socket(AF_INET, SOCK_STREAM)
s.connect(("host", 12000))
s.send(poruka)
odg = s.recv(1024)
print(odg); s.close()
```

Istovremeni klijenti (konkurentnost)

- poslužitelj mora posluživati **više klijenata odjednom**
- pristupi: **dretve/procesi** po vezi, ili **asinkroni** I/O (event loop)
- **welcome** socket prima nove veze; svaka dobiva svoj **connection** socket

Sažetak

Ključne poruke

- **DNS** je distribuirana, hijerarhijska baza imena; oslanja se na predmemoriju
- **e-pošta**: SMTP za slanje, IMAP/POP3 za dohvat, MIME za sadržaj
- **P2P** (BitTorrent) skalira distribuciju; resursi rastu s brojem peerova
- **DASH** i **CDN** omogućuju kvalitetan streaming velikoj publici
- **socket API** temelj je mrežnog programiranja (UDP i TCP)

Sljedeće predavanje

Transportni sloj I: UDP i pouzdani prijenos podataka.

Pitanja za provjeru

- Koja je razlika između rekurzivnog i iterativnog DNS upita?
- Zašto DNS predmemorija toliko pomaže?
- Koje protokole koristi e-pošta za slanje, a koje za dohvat?
- Zašto se P2P bolje skalira od klijent–poslužitelja?
- U čemu se razlikuju UDP i TCP socketi?

Hvala na pažnji!

Pitanja i rasprava

doc. dr. sc. Sandi Baressi Šegota • FIPU