

2: Nivo aplikacije

Nivo aplikacije

- ❑ Principi protokola nivoa aplikacije
- ❑ Web i HTTP
- ❑ FTP
- ❑ Elektronska pošta na Internetu (SMTP, POP3, IMAP)
- ❑ DNS
- ❑ P2P
- ❑ Video *streaming* i CDN

Primjeri mrežnih aplikacija

- E-mail
- Web
- *Instant messaging*
- *Remote login*
- *P2P file sharing*
- *Multi-user mrežne igre*
- *Streaming stored video klipovi (Netflix, Hulu, YouTube,...)*
- Internet telefon
- *Real-time video konferencija*
- *Grid computing*
- Društvene mreže
- *Cloud computing*
- *Fog computing*

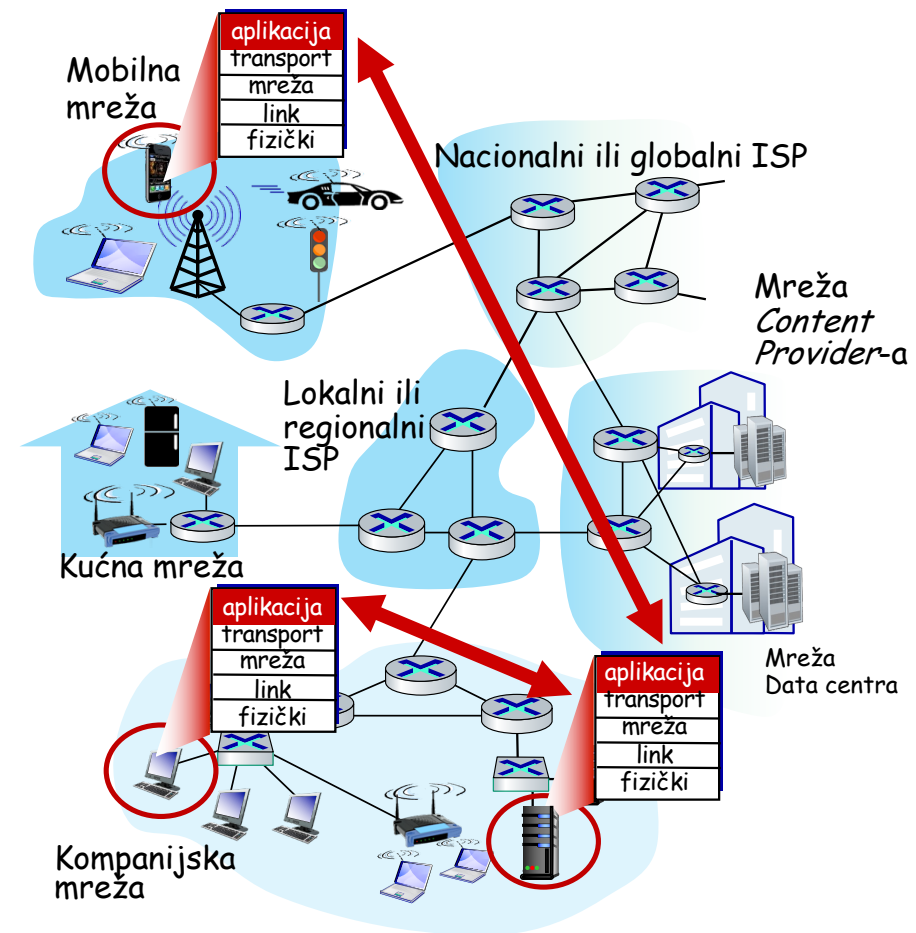
Kreiranje mrežne aplikacije

Pisanje programa koji

- se izvršavaju na različitim krajnjim sistemima i
- komuniciraju preko mreže.
- primjer je komunikacija web server softvera sa web browser softverom

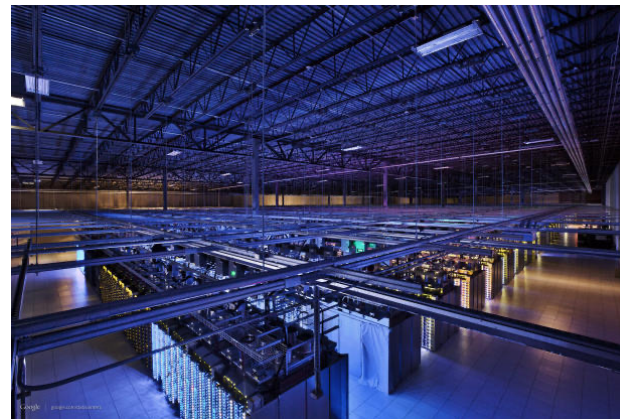
Softver se ne piše za uređaje na kičmi mreže

- mrežni uređaji na kičmi uglavnom ne funkcionišu na nivou aplikacije
- ovakav dizajn dozvoljava brzi razvoj aplikacija



Google Data Centri

- ❑ Procijenjena cijena jednog data centra: stotine miliona \$
- ❑ Google je svake godine potroši nekoliko milijardi \$ u razvoj data centara
- ❑ Svaki data centar troši stotine MW električne energije



Google Data Centri

USA

Berkeley County, South Carolina
Council Bluffs, Iowa
The Dalles, Oregon
Douglas County, Georgia
Henderson, Nevada
Jackson County, Alabama
Lenoir, North Carolina
Loudoun County, Virginia
Mayes County, Oklahoma
Midlothian, Texas
Montgomery County, Tennessee
New Albany, Ohio
Papillion, Nebraska
Storey County, Nevada

South America

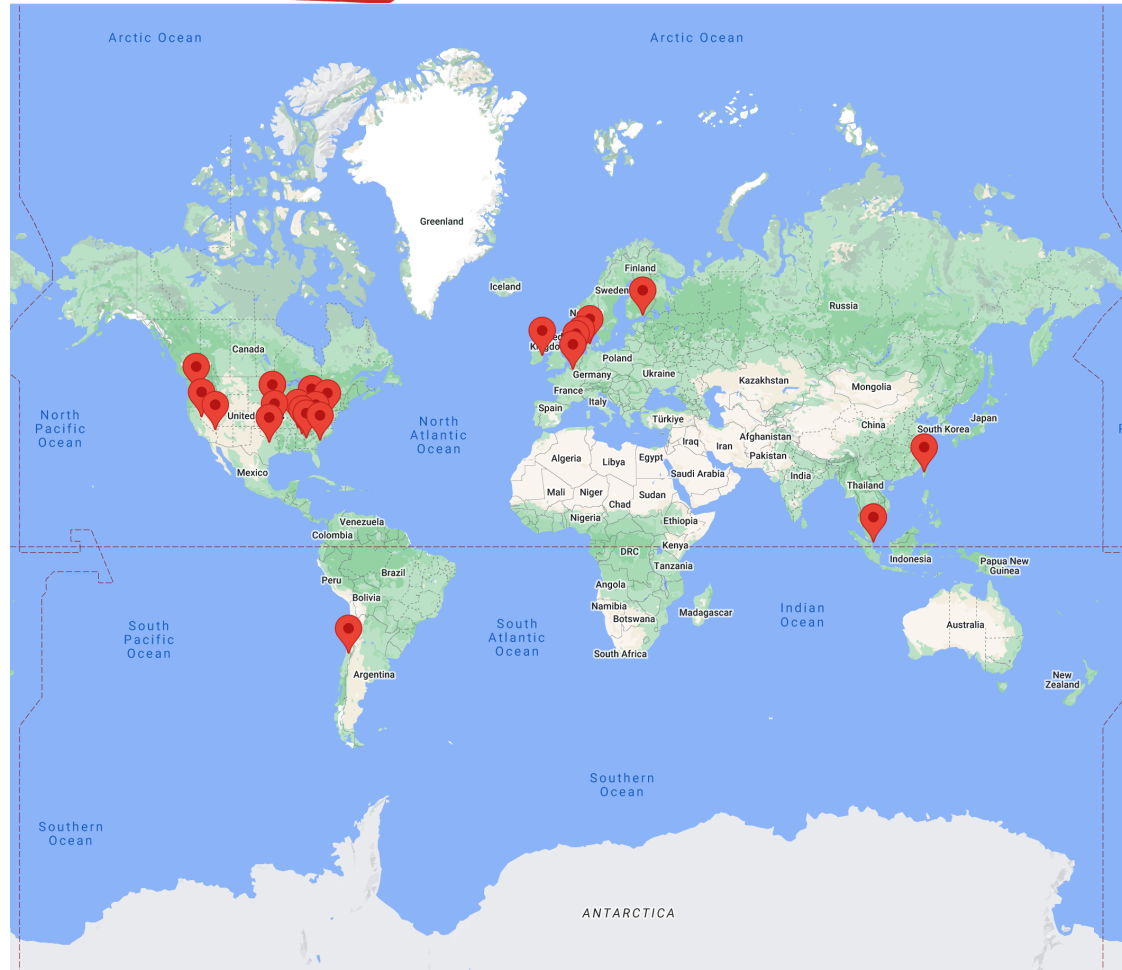
Quilicura, Chile

Europe

Dublin, Ireland
Eemshaven, Netherlands
Fredericia, Denmark
Hamina, Finland
Middenmeer, Netherlands
St. Ghislain, Belgium

Asia

Changhua County, Taiwan
Singapore



<http://www.google.com/about/datacenters/inside/locations/index.html>

Komuniciranje procesa

Proces: program koji se izvršava na hostu.

- ❑ U samom hostu, dva procesa komuniciraju na bazi inter procesne komunikacije (definisane u OS).
- ❑ Procesi na različitim hostovima komuniciraju razmjenu poruka

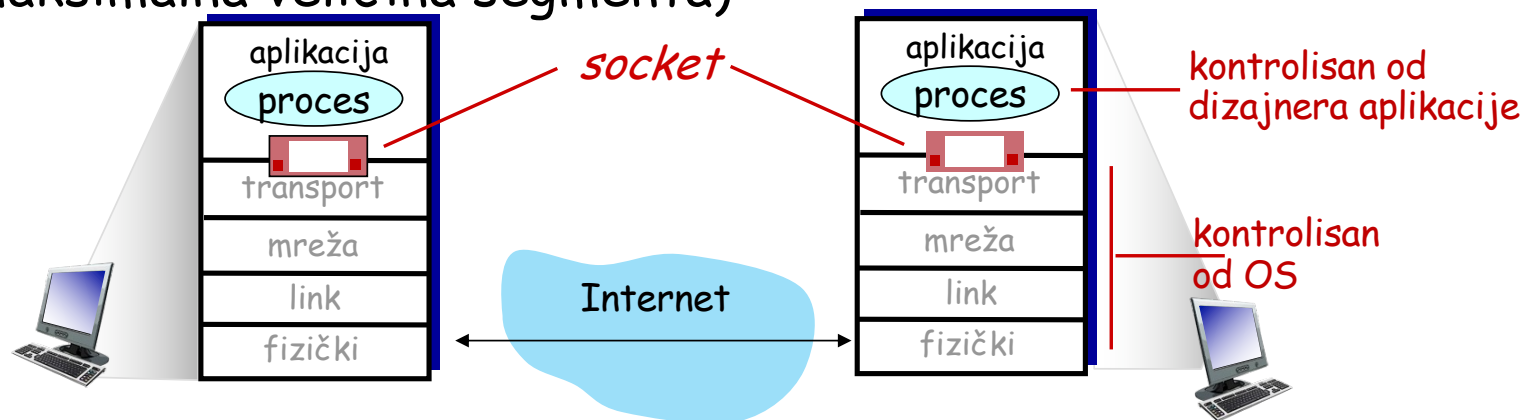
Klijentski proces: proces koji inicijalizuje komunikaciju

Serverski proces: proces koji čeka da bude kontaktiran

Aplikacije sa P2P arhitekturom imaju i klijent i server procese

Soketi

- ❑ Proces šalje/prima poruke preko svog *socket*-a
- ❑ *Socket* je analogan vratima
 - Proces šalje poruke preko *socket*-a
 - proces koji šalje se oslanja na transportnu infrastrukturu na drugoj stani vrata koja prenosi poruku do *socket*-a prijemnog procesa
- ❑ API: (1) izbor transportnog protokola; (2) mogućnost specificiranja nekoliko parametara (maksimalna veličina bafera i maksimalna veličina segmenta)



Adresiranje

- ❑ Za proces koji prima poruke, mora postojati identifikator
- ❑ Svaki host ima jedinstvenu 32 bitnu IP adresu
- ❑ Komanda ipconfig...
- ❑ P: Da li je IP adresa hosta na kojem se proces izvršava dovoljna za identifikaciju procesa?
- ❑ Identifikator uključuje i IP adresu i broj porta vezan za proces na hostu.
- ❑ Primjer brojeva porta:
 - HTTP server: 80
 - Mail server: 25
- ❑ VIŠE KASNIJE

○: Ne, mnogi procesi se mogu izvršavati na istom hostu

Protokol nivoa aplikacije definiše

- ❑ Tipove poruka koje se razmjenjuju, npr., zahtjevi & poruke odgovora
- ❑ Sintaksu poruka: koja su polja & kako su raspoređena
- ❑ Semantiku poruka: odnosno značenje informacija u poljima
- ❑ Pravila vezana kada i kako se šalju poruke i na koji način se odgovara na njih

Javni (*public*) protokoli:

- ❑ Definisani u RFC-ovima
- ❑ Dozvoljavaju interoperabilnost
- ❑ HTTP, SMTP, FTP,...

Privatni (*proprietary*) protokoli:

- ❑ Skype, Viber, Zoom,...

Koji transportni servisi su potrebni aplikacijama?

Integritet podataka

- ❑ Neke aplikacije (audio) mogu tolerisati određeni nivo gubitaka
- ❑ Druge aplikacije (file transfer, telnet) zahtijevaju 100% pouzdani transfer podataka

Vrijeme

- ❑ Neke aplikacije (Internet telefonija, interaktivne igre) zahtijevaju malo kašnjenje

Brzina prenosa

- ❑ Neke aplikacije (multimedija) zahtijevaju preciziranje minimalne dostupne brzine prenosa
- ❑ Druge aplikacije (“elastične aplikacije”) koriste onoliko opsega koliko mogu dobiti

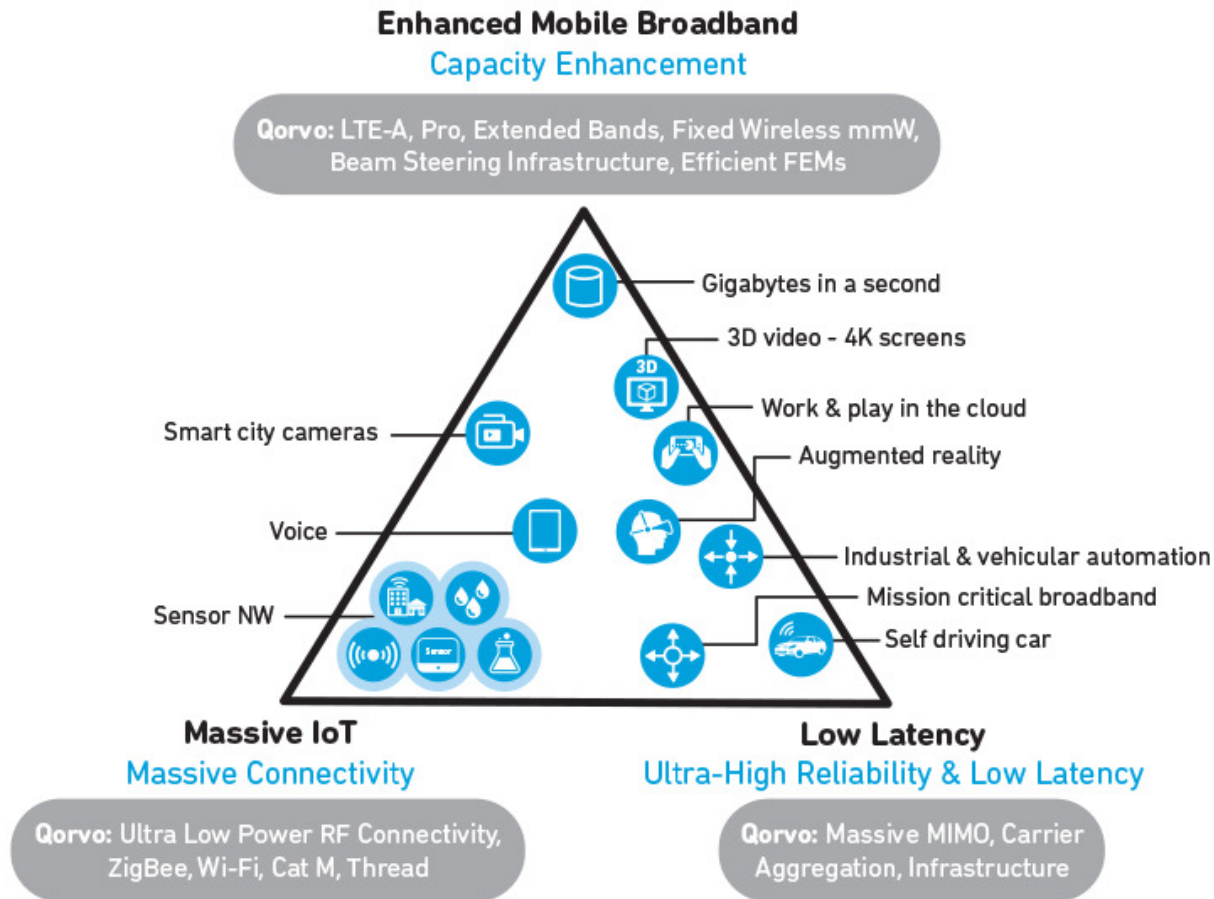
Zaštita

- ❑ Enkripcija, integritet podataka, ...

Transportni servisni zahjevi zajednički za sve aplikacije

Aplikacija	Gubici	Brzina prenosa	Vrem. osjet.
<i>File transfer</i>	bez	elastičan	ne
E-mail	bez	elastičan	ne
Web dokumenti	bez	elastičan	ne
<i>Real-time</i> audio/video	tolerantne	audio: 5kb/s-1Mb/s video:10kb/s-5Mb/s	da, 10-ak ms
<i>Streaming</i> audio/video	tolerantne	isti kao <i>real-time</i>	da, nekoliko s
Interaktivne igre	tolerantne	nekoliko kb/s i više	da, 10-ak ms
<i>Text messaging</i>	bez	elastičan	da i ne

"5G trougao"



(Source: Qorvo, Inc., from ITU-R IMT 2020 requirements)

Servisi transportnih protokola Interneta

TCP servisi:

- ❑ konektivnost: uspostavljanje komunikacije se zahtijeva između klijentskih i serverskih procesa
- ❑ pouzdani transport između procesa slanja i prijema
- ❑ kontrola protoka: pošiljalac ne smije da "zaguši" prijemnik
- ❑ kontrola zagušenja: usporava pošiljaoca kada je mreža zagušena
- ❑ Ne obezbjeđuje: tajming, garantovanje minimalne brzine prenosa, zaštitu

UDP servisi:

- ❑ Nepouzdana prenos podataka između procesa slanja i prijema
- ❑ Ne obezbjeđuje: uspostavljanje veze, pozdanost, kontrolu protoka, kontrolu zagušenju, tajming, garantovani opseg, zaštitu

Zašto oba? Zašto UDP?

Internet aplikacije: aplikacija, transportni protokoli

Aplikacija	Protokol nivoa aplikacije	Transportni protokol
e-mail	SMTP [RFC 2821]	TCP
Udaljeni terminal	Telnet [RFC 854]	TCP
Web	HTTP1.1 [RFC 7320]	TCP
File transfer	FTP [RFC 959]	TCP
<i>Streaming</i> audio/video	HTTP (RFC 7230), DASH	TCP
Internet telefonija	SIP, RTP, privatni	TCP ili UDP
Interaktivne igre	WOW, FPS	UDP ili TCP

Zaštita i TCP

TCP & UDP

- ❑ Nemaju enkripciju
- ❑ Tekstualne poruke se prenose bez sigurnosne zaštite preko Interneta

Transport Layer Security (TLS)

- ❑ Omogućava enkripciju TCP konekcije
- ❑ Čuva integritet podataka
- ❑ Obezbjeđuje autorizacija od kraja do kraja

TLS je implementiran na nivou aplikacije

- ❑ Aplikacije koriste TLS biblioteke
- ❑ Tekst koji se šalje preko *socket-a* na Internet je enkriptovan

Web i HTTP

Termini

- Web stranica se sastoji od objekata, koji se mogu nalaziti na različitim serverima
- Objekat može biti HTML fajl, JPEG slika, Java "applet", audio fajl,...
- Web stranica se sastoji od osnovnog HTML-fajla koji sadrži više referenci objekata adresiranih pomoću URL (*Uniform Resource Locators*)

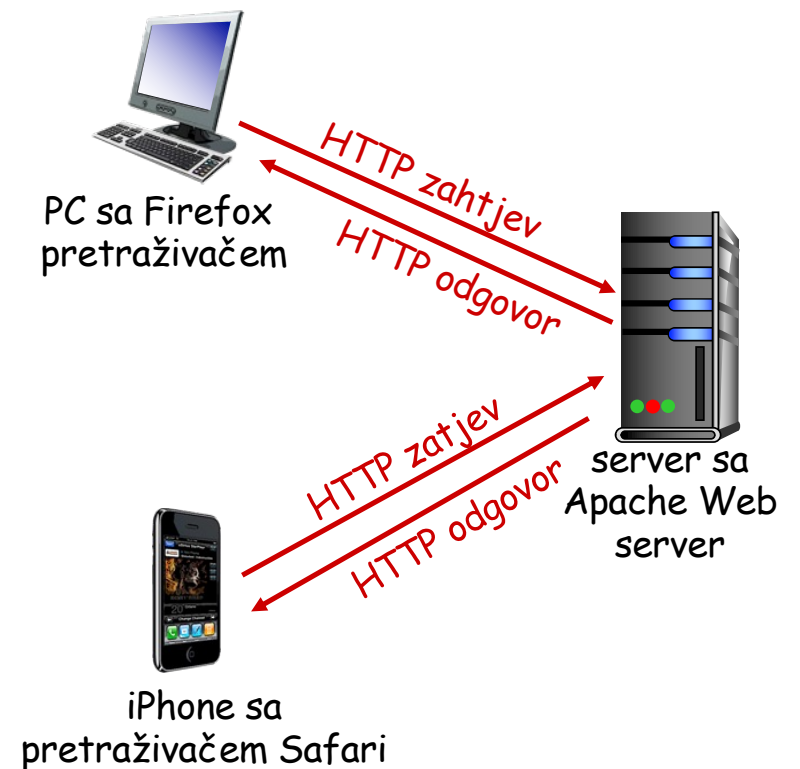
http://www.cftmn.ac.me/index.html

ime hosta ime puta

Pregled HTTP-a

HTTP (*HyperText Transfer Protocol*)

- Protokol nivoa aplikacije Web aplikacije
- klijent/server model
 - klijent: *Web browser* koji zahtijeva, prima, prikazuje Web objekte
 - server: Web server šalje objekte kao odgovor na zahtjeve *Web browser-a*



Pregled HTTP-a (nastavak)

Koristi TCP:

- ❑ klijent inicijalizuje TCP konekciju (kreira *socket*) prema serveru na portu 80
- ❑ server prihvata TCP konekciju sa klijentom
- ❑ HTTP poruke zahtjeva i poruke odgovora (poruke protokola nivoa aplikacije) se razmjenjuju između *browser-a* (HTTP klijent) i Web servera (HTTP server)
- ❑ TCP konekcija se zatvara

HTTP je *stateless*

- ❑ server ne čuva informacije o prethodnim korisnikovim zahtjevima (ne raspoznaje korisnike)

Pored toga

Protokoli koji nadziru stanje su kompleksni!

- ❑ Ranije stanje mora biti nadzirano
- ❑ ako server/klijent "padne", njihovi uvidi u "stanje" mogu biti inkonzistentni, moraju biti ponovo razmotreni

HTTP konekcije

Neperzistentni (neistrajni) HTTP

- ❑ Otvara se TCP konekcija
- ❑ Najviše jedan objekat se šalje preko TCP konekcije.
- ❑ Zatvara se TCP konekcija
- ❑ Povlačenje više objekata podrazumijeva otvaranje više konekcija

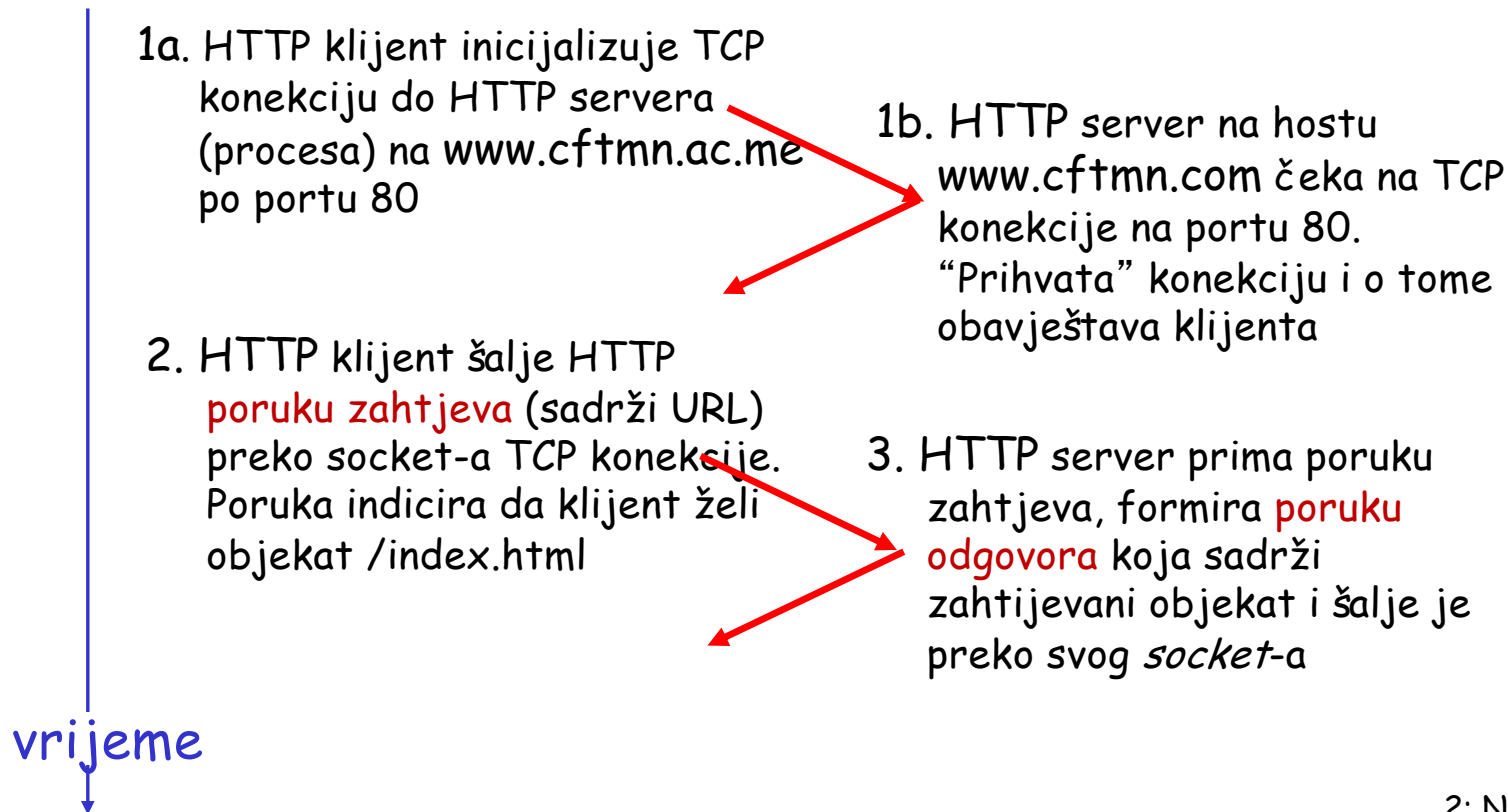
Perzistentni HTTP

- ❑ Otvara se TCP konekcija
- ❑ Više objekata može biti poslato preko jedne TCP konekcije između klijenta i servera.
- ❑ Zatvara se TCP konekcija

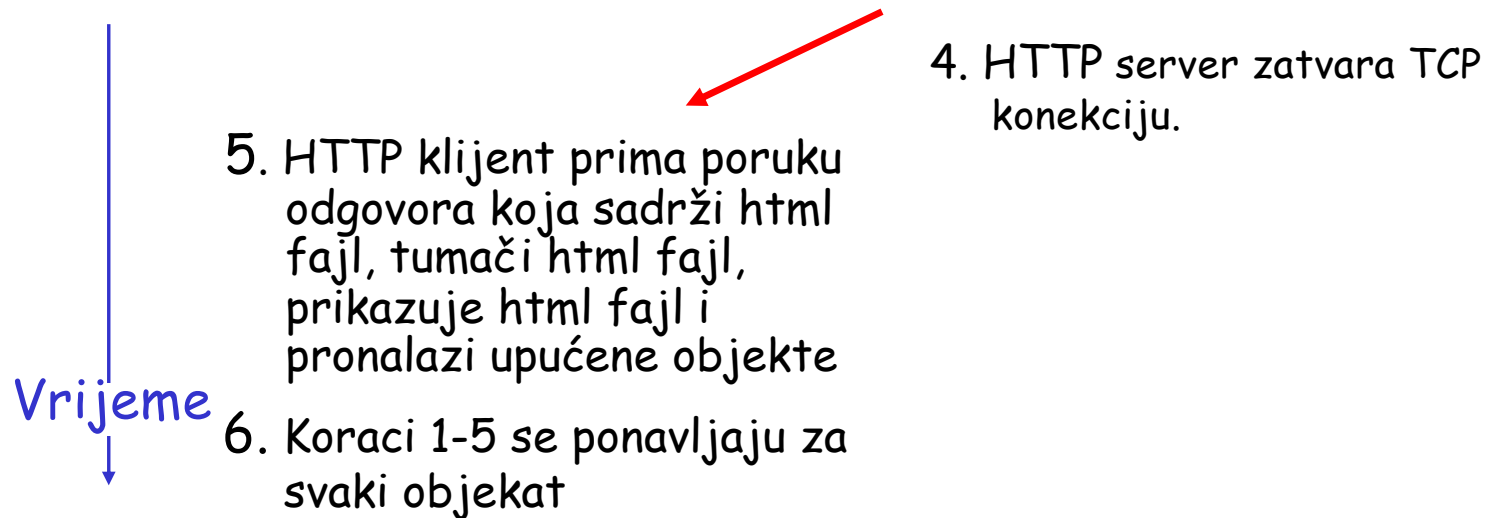
Neperzistentni HTTP

Neka korisnik unese sledeći URL

`http://www.cftmn.ac.me/index.html`



Neperzistentni HTTP(nastavak)



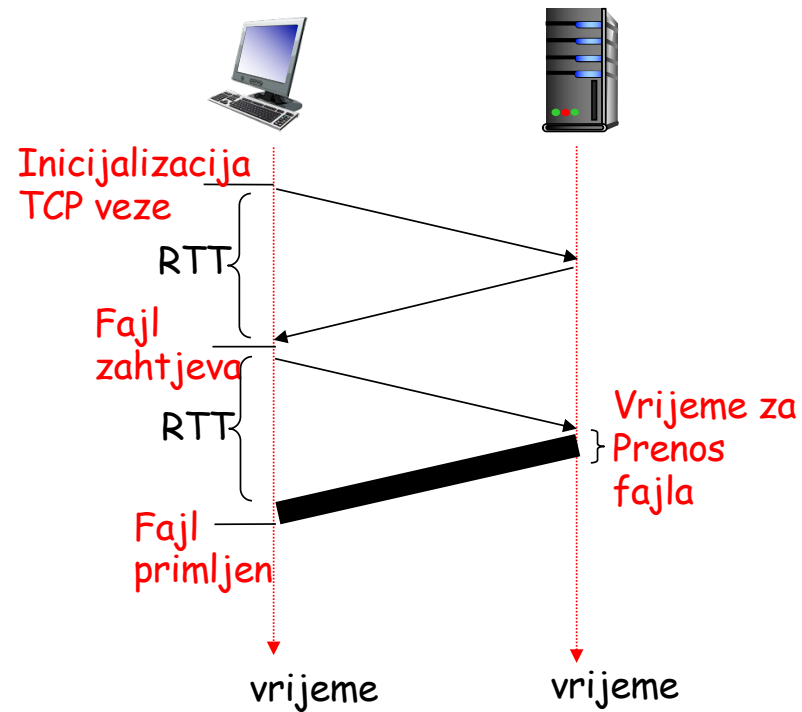
Modelovanje vremena odgovora

Definicija RTT (*Round Trip Time*):
vrijeme prenosa malog paketa od izvora do
destinacije i nazad.

Vrijeme odgovora:

- RTT za inicijalizaciju TCP veze
- RTT za slanje HTTP poruke zahtjeva i
prijem prvih nekoliko bajtova HTTP poruke
odgovora
- Vrijeme prenosa fajla

$2RTT + \text{vrijeme prenosa fajla}$



Perzistentni HTTP (HTTP1.1)

Problemi neperzistentnog HTTP:

- ❑ Zahtijeva 2 RTT po objektu
- ❑ Operativni sistem mora pratiti i dodijeliti resurse hosta za svaku TCP vezu
- ❑ Problem je što *browser*-i često otvaraju paralelne TCP veze za povlačenje zahtijevanih objekata

Perzistentni HTTP 1.1

- ❑ Server zadržava vezu otvorenu nakon slanja poruke odgovora
- ❑ Sekvencijalne HTTP poruke između istog para klijent/server se šalju istom vezom
- ❑ Klijent šalje poruke zahtjeva odmah po dobijanju referenci objekata
- ❑ Potreban je po jedan RTT za svaki referencirani objekat
- ❑ Zatvara konekciju poslije određenog vremena neaktivnosti

HTTP poruka zahtjeva

- Dva tipa HTTP poruka: *zahtjev, odgovor*
- HTTP poruka zahtjeva:
 - ASCII (format čitljiv čovjeku)

Linija zahtjeva
(GET, POST,
HEAD komande)

Linije
zaglavlja

carriage return,
line feed na
početku linije
označavaju kraj zaglavlja

```
GET /index.html HTTP/1.1\r\n
Host: www.cftmn.ac.me\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10.15; rv:80.0) Gecko/20100101 Firefox/80.0 \r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Connection: keep-alive\r\n
\r\n
```

carriage return karakter
line-feed karakter

Tipovi zahtjeva

POST:

- ❑ Web stranice često sadrže forme za unos podataka
- ❑ Podaci koje je unio korisnik se šalju od klijenta server u tijelu HTTP POST poruke zahtjeva

GET:

- ❑ Služi za povlačenje objekata sa servera
- ❑ Koristi se i za slanje korisnikovih podataka u sklopu URL polja HTTP GET poruke zahtjeva (poslije simbola '?'):

HEAD:

- ❑ Zahtijeva samo zaglavlje koja se šalju ako je specificirani URL zahtijevan GET porukom

PUT:

- ❑ *Upload*-uje novi fajl (objekat) na server
- ❑ U potpunosti mijenja fajl koji postoji na specificiranom URL-u sa sadržajem u tijelu HTTP PUT poruke zahtjeva

www.google.com/animalsearch?monkeys&banana

HTTP poruka odgovora

Statusna linija (protokol,
statusni kod, statusna fraza)

Linije
zaglavlja

```
HTTP/1.1 200 OK\r\n
Date: Tue, 08 Sep 2020 00:53:20 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.9
      mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 01 Mar 2016 18:57:50 GMT
ETag: "a5b-52d015789ee9e"
Accept-Ranges: bytes
Content-Length: 2651
Content-Type: text/html; charset=UTF-8
\r\n
data data data data data ...
```

podaci, npr.,
zahtijevani
HTML fajl

HTTP kodovi statusnog odgovora

Nalaze se u statusnoj liniji serverove poruke odgovora.
Nekoliko primjera kodova statusa i odgovarajućih poruka:

200 OK

- Zahtjev uspješan, zahtijevani objekat se nalazi u poruci

301 Moved Permanently

- Zahtijevani objekat preseljen, nova lokacija specificirana u poruci (Lokacija:)

400 Bad Request

- Server ne razumije poruku zahtijeva

404 Not Found

- Zahtijevani dokument nije pronađen na ovom serveru

505 HTTP Version Not Supported

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

Cookies: vode računa o “stanju” (RFC 6265)

Mnogi Web sajtovi koriste *cookies*

Četiri komponente:

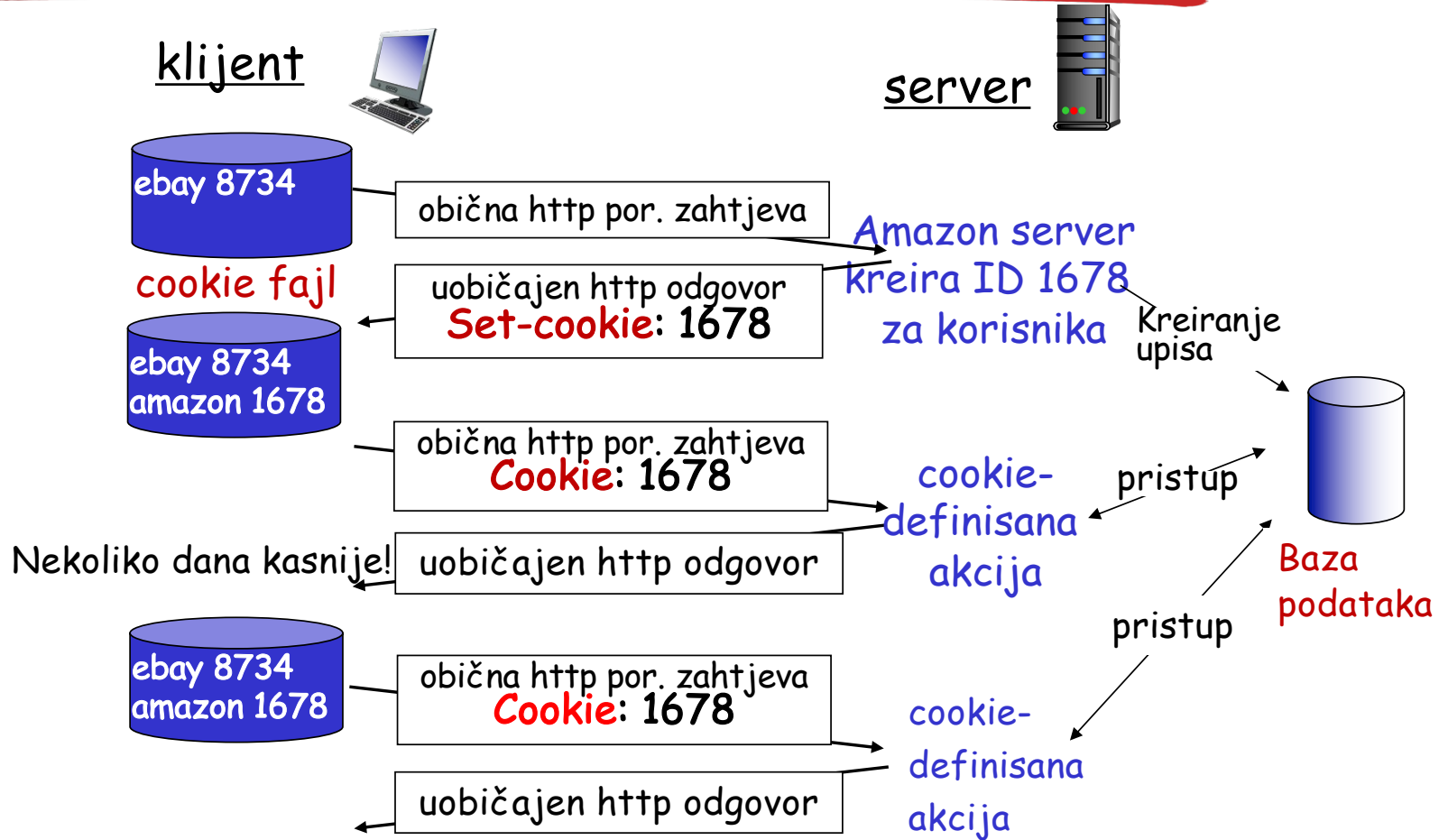
- 1) *Linija zaglavlja Set-cookie* u HTTP poruci odgovora
- 2) *Linija zaglavlja Cookie* u HTTP poruci zahtjeva
- 3) *Cookie fajl* se čuva na korisnikovom hostu i održava se od strane korisnikovog browser-a
- 4) Baza *podataka* na Web sajtu

Primjer:

- Korisnik pristupa Internetu uvijek preko istog PC-a
- Posjećuje specifične *e-commerce* sajtove po prvi put
- Kada inicijalni HTTP zahtjevi dođu na sajt, sajt kreira jedinstveni cookie ID i kreira odgovarajuću informaciju u bazi podataka za cookie ID
- Sledeći HTTP zahtjevi nose isti cookie ID tako da web sajt može pratiti aktivnost korisnika

<https://tools.ietf.org/html/rfc6265>

Cookies: vode računa o "stanju" (nastavak)



Cookies: vode računa o “stanju” (nastavak)

Šta *cookies* donose?

- ❑ autorizaciju
- ❑ *shopping cards*
- ❑ preporuke
- ❑ stanje korisnikove sesije (Web e-mail)

Cookies i privatnost: Pored toga

- ❑ *Cookies* dozvoljavaju vlasniku sajtu da dosta nauči o korisniku
- ❑ Korisnici mogu dostaviti imena i kontakt podatke
- ❑ Web pretraživači koriste *cookies* da nauče više o korisnicima
- ❑ Kompanije dobijaju dodatne informacije preko weba

GDPR (EU General Data Protection Regulation) i cookies

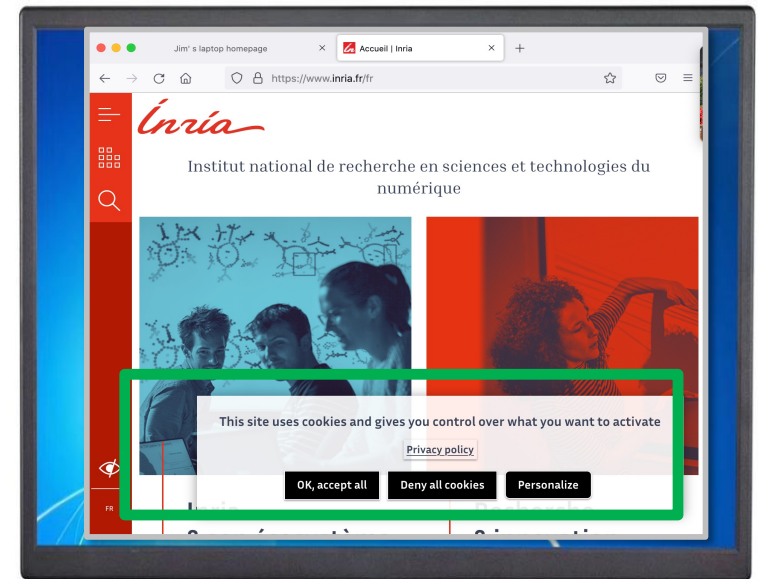
"Natural persons may be associated with online identifiers [...] such as internet protocol addresses, cookie identifiers or other identifiers [...].

This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them."

GDPR, recital 30 (May 2018)



Kada cookie može identifikovati osobu, onda se može smatrati ličnim podatkom koji je podložan GDPR regulaciji ličnih podataka

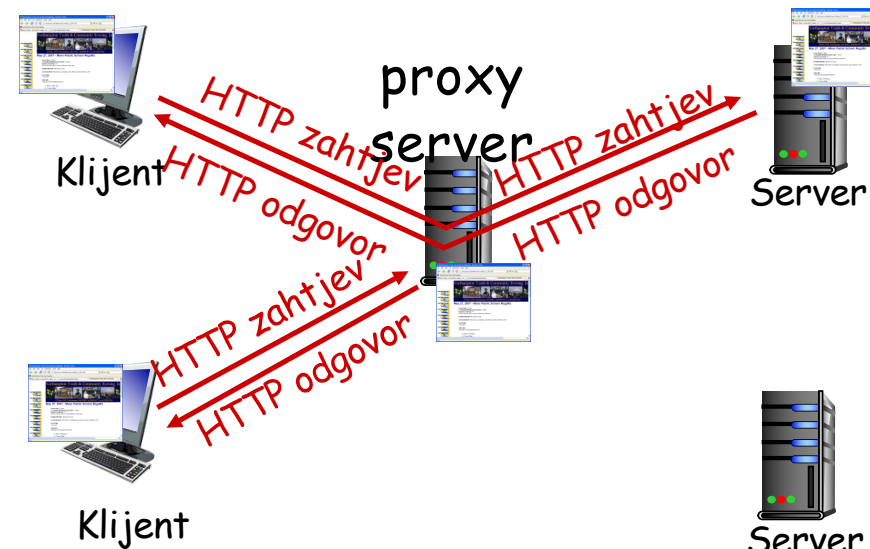


Korisnik ima eksplicitnu kontrolu da li je ili nije dozvoljen cookie

Web caches (proxy serveri)

Cilj: zadovoljenje klijentovog zahtjeva bez uključivanja originalnog servera

- Korisnik *setuje browser*: Web pristup preko *proxy servera*
- *browser šalje sve HTTP zahtjeve proxy serveru*
 - objekat u *proxy-u*: *proxy šalje objekat browser-u*
 - Ako objekat nije na *proxy-u*, *proxy* zahtijeva objekat od željenog servera i nakon dobijanja ga prosleđuje klijentu



<https://tools.ietf.org/html/rfc7234>

Više o *proxy* serveru

- *Proxy* server radi kao
 - Klijent originalnog servera
 - Server za originalnog klijenta

- server u zaglavlju odgovora obavještava o tome da li je dozvoljeno čuvanje objekta

```
Cache-Control: max-age=<seconds>
```

```
Cache-Control: no-cache
```

Zašto *proxy* server?

- Smanjuje vrijeme odziva na zahtjev.
- Smanjuje saobraćaj na linku institucije prema Internetu.
- Internet sa *proxy* serverom omogućava “slabim” provajderima sadržaja efikasniju predaju sadržaja

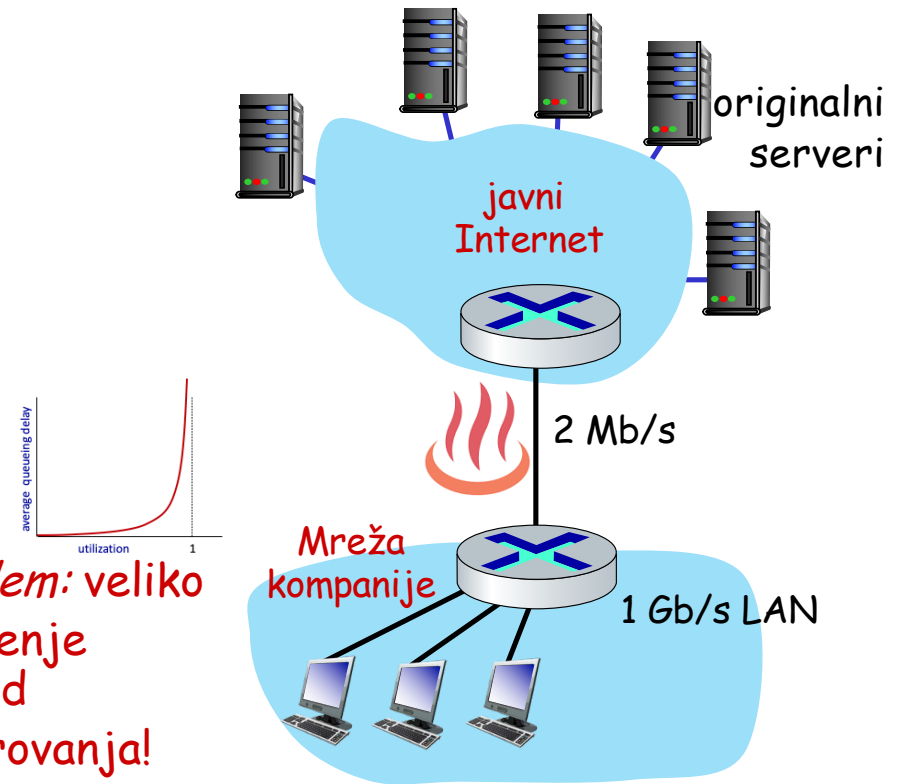
Primjer:

Pretpostavke:

- Srednja veličina objekta: 100000 bita
- Srednji broj zahtjeva prema željenim serverima: 19 zahtjeva/s
- Srednja brzina : 1.9Mb/s
- RTT od rutera institucije do željenog servera: 2s
- Brzina na pristupnom linku: 2Mb/s

Posledice:

- Iskorišćenje LAN-a: 0.19%
- Iskorišćenje pristupnog linka = 95%
- Ukupno kašnjenje = kašnjenje na Internetu + kašnjenje u pristupu + LAN kašnjenje
= 2s + minuti + ms



problem: veliko kašnjenje uslijed baferovanja!

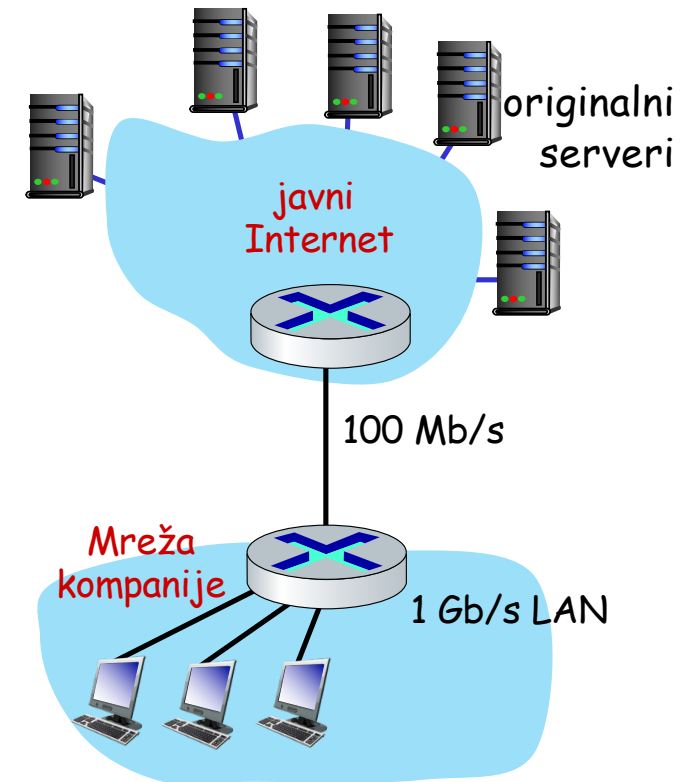
Primjer: brži pristupni link

Pretpostavke:

- Srednja veličina objekta: 100000 bita
- Srednji broj zahtjeva: 19 zahtjeva/s
- Srednja brzina: 1.9Mb/s
- RTT od rutera institucije do željenog servera: 2s
- Brzina pristupnog linka: 100Mb/s

Posledice:

- Iskorištenje LAN-a: 0.19%
 - Iskorišćenje linka = 1.9%
 - Ukupno kašnjenje = Internet kašnjenje + pristupno kašnjenje + LAN kašnjenje
- = 2s + ms + ms



Troškovi: povećanje brzine pristupa je skupo!

Primjer: Lokalni proxy

Pretpostavke:

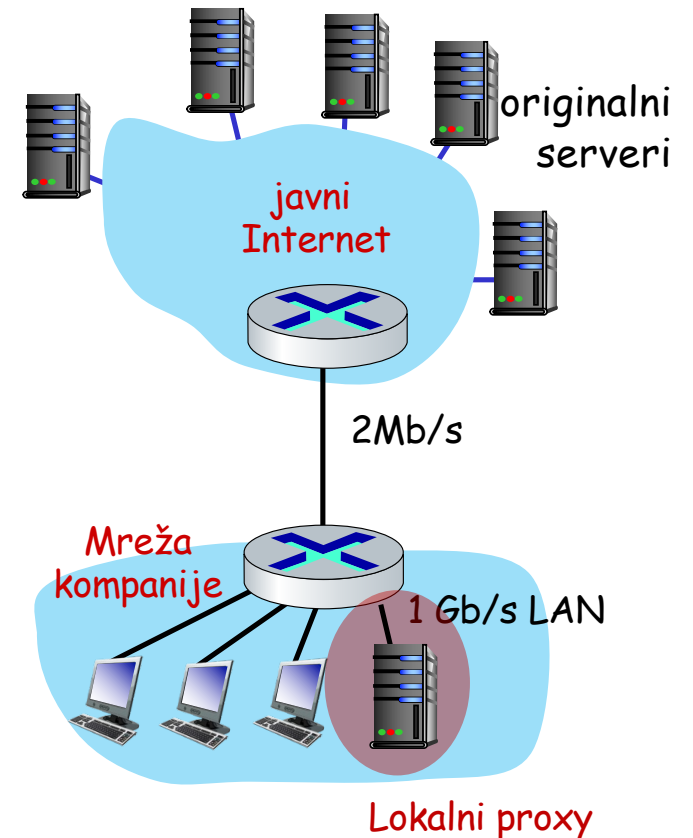
- Srednja veličina objekta: 100000 bita
- Srednja brzina zahtjeva: 19 zahtjeva/s
- Srednja brzina: 1.9Mb/s
- RTT od rutera institucije do željenog servera: 2s
- Brzina pristupa: 2Mb/s

Posledice:

- Iskorišćenje LAN-a: 0.19%
- Iskorišćenje pristupnog linka= ?
- Ukupno kašnjenje= ?

Kako izračunati iskorišćenje i kašnjenje?

Troškovi: proxy nije skup!



Primjer: Lokalni proxy

Izračunavanje iskorišćenja i kašnjenja:

- Neka je vjerovatnoća pogađanja 0.4
 - 40% zahtjeva se posluži na proxy serveru, 60% zahtjeva na željenom Internet serveru

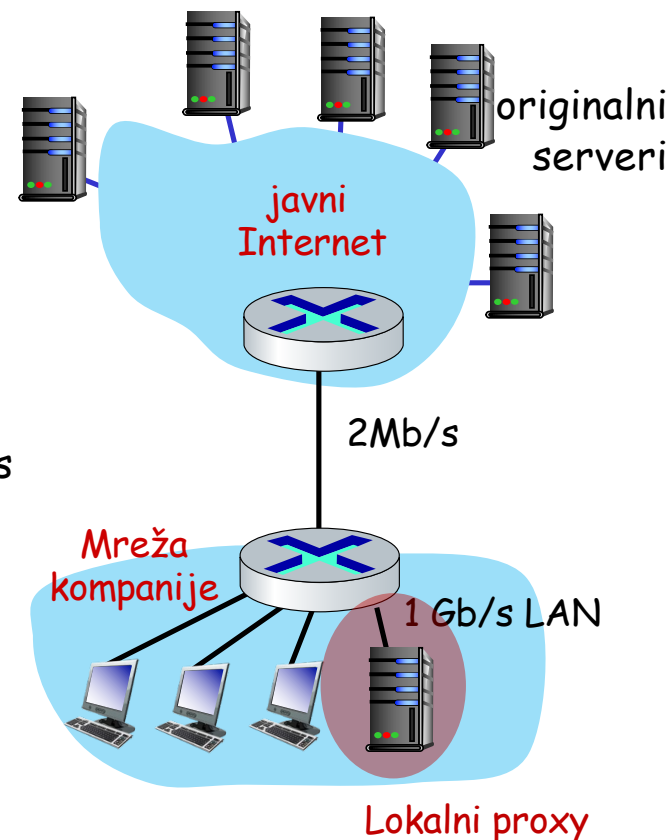
Iskorišćenje pristupnog linka:

- 60% zahtjeva koristi pristupni link
- Brzina prenosa preko pristupnog linka = $0.6 * 1.9 \text{ Mb/s} = 1.14 \text{ Mb/s}$
- iskorišćenje = $1.14 / 2 = .57$

Ukupno kašnjenje

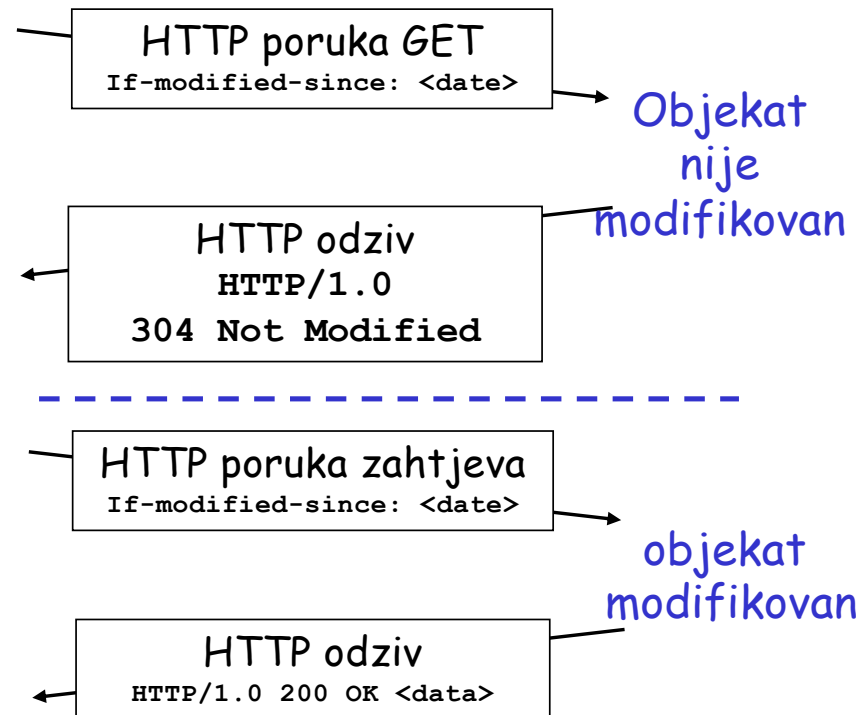
$$\begin{aligned} &= 0.6 * (\text{kašnjenje od željenih servera}) + 0.4 * \\ &(\text{kašnjenje do proxy servera}) \\ &= 0.6 (2.0) + 0.4 (\sim \text{ms}) \\ &= \sim 1.2 \text{ s} \end{aligned}$$

Manje nego pristupni link od 100Mb/s



Browser caching: Conditional GET

- **Cilj:** ne slati objekat ako *cache* ima *up-to-date* sačuvanu verziju
- **klijent:** specificira datum čuvanja kopije u HTTP zaglavlju
`If-modified-since: <date>`
- **server:** odgovor ne sadrži objekat ako je sačuvana kopija up-to-date:
`HTTP/1.0 304 Not Modified`



<https://tools.ietf.org/html/rfc7232>

HTTP/2

Cilj: smanjiti kašnjenje u slučaju višeobjektnih HTTP zahtjeva

HTTP 1.1: uvodi više, pipeline GET poruka preko jedne TCP konekcije

- ❑ server odgovara *redosledno* (FCFS: *first-come-first-served*) na GET zahtjeve
- ❑ zbog FCFS može se desiti da mali objekat mora da čeka prenos iza velikog objekta, odnosno da doživi HOL (*head-of-line*) blokiranje
- ❑ Oporavak od gubitaka (retransmisija izgubljenog TCP segmenta) usporava prenos objekata

HTTP/2

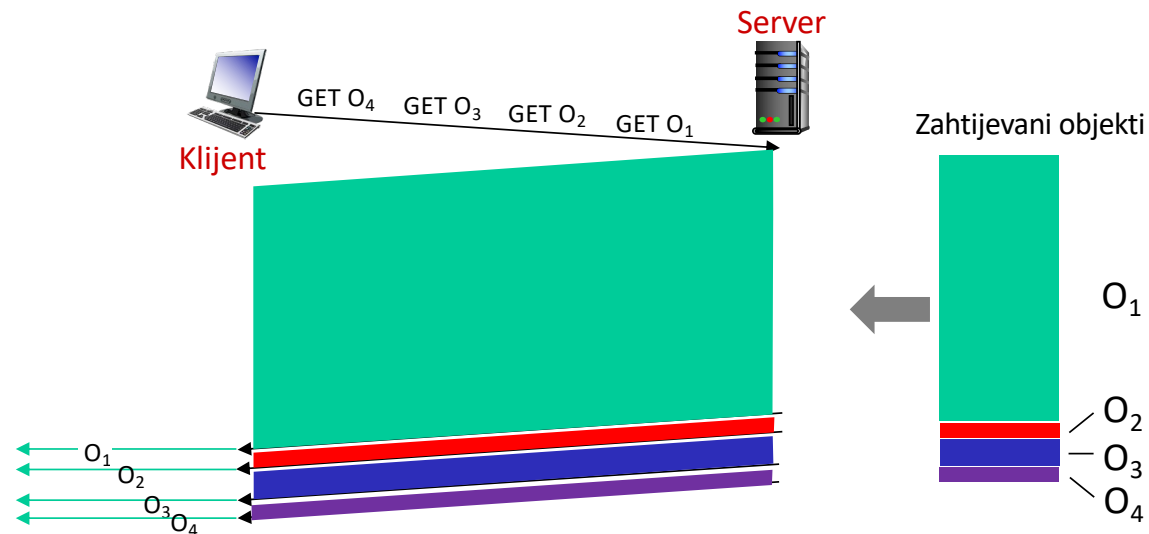
Cilj: smanjiti kašnjenje u slučaju višeobjektnih HTTP zahtjeva

HTTP/2: [RFC 7540, 2015] povećava fleksibilnost servera prilikom slanja objekata klijentu:

- ❑ Poruke zahtjeva, statusni kodovi, većina polja zaglavlja su istovjetna kao kod HTTP 1.1
- ❑ Redosled slanja zahtijevanih objekata je baziran na prioritetima koje je definisao klijent (ne mora biti FCFS)
- ❑ Omogućava *slanje* nezahitijevanih objekata klijentu
- ❑ Dijeli objekte na frejmove, raspoređuje frejmove radi smanjenja HOL blokiranja

HTTP/2: ublažavanje HOL blokiranja

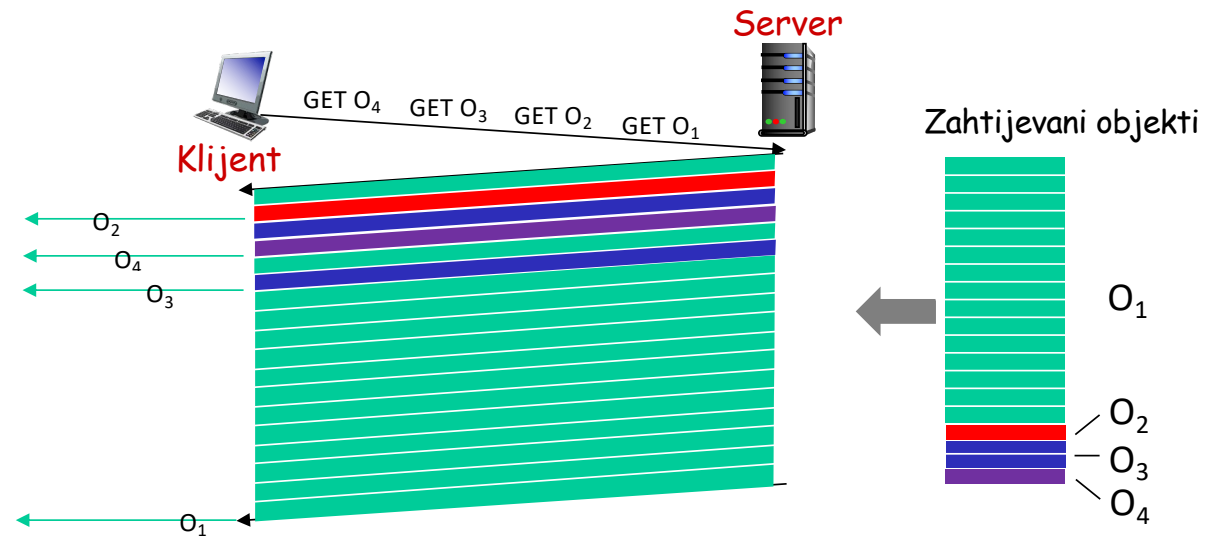
HTTP 1.1: klijent zahtjeva veliki objekat (na primjer video fajl) i 3 manja objekta



Slanje objekata po redosledu zahtjeva: O₂, O₃, O₄ čekaju iza O₁

HTTP/2: ublažavanje HOL blokiranja

HTTP/2: objekti se dijele ne frejmove, "izmiješano" slanje frejmova



O₂, O₃, O₄ se brzo prenose, malo kašnjenje se dodaje O₁

HTTP/3

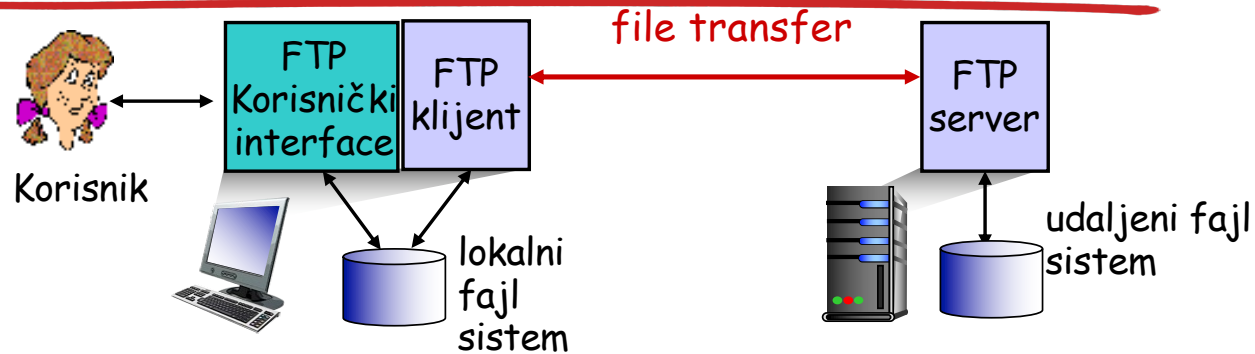
HTTP/2 funkcioniše preko jedne TCP konekcije:

- ❑ Oporavak od gubitka paketa i dalje usporava prenos objekata
 - ❑ Kao i kod HTTP 1.1, *browser*-i otvaraju više paralelnih TCP konekcija radi smanjenja usporavanja i povećanja propusnosti
- ❑ Nema zaštite korišćenjem obične TCP konekcije

HTTP/3:

- ❑ Zaštita
- ❑ Detekcija greške po objektu
- ❑ Kontrolu protoka preko UDP protokola (QUIC)

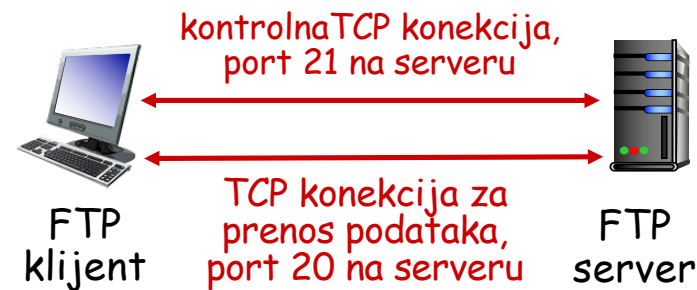
FTP (*File Transfer Protocol*)



- ❑ transfer fajla od/do udaljenog hosta
- ❑ klijent/server model
 - klijent: strana koja inicijalizuje prenos (ili od/do udaljenog hosta)
 - server: udaljeni host
- ❑ ftp: RFC 959
- ❑ ftp server: port 21

FTP: kontrolna veza i veze za prenos podataka

- FTP klijent kontaktira FTP server na port 21, definišući TCP kao transportni protokol
- Klijent dobija zahtjev od servera za autorizacijom preko **kontrolne TCP konekcije**
- Klijent pregleda udaljene direktorijume slanjem komandi preko kontrolne TCP konekcije
- Kada server primi komandu za prenos fajla, server otvara **TCP konekciju za prenos podataka** do klijenta (port 20)
- Poslije slanja jednog fajla server zatvara TCP konekciju za prenos podataka.



- Server otvara drugu TCP vezu podataka za prenos drugog fajla.
- Kontrola veze: *out of band*
- FTP server nadzire stanje konekcije: trenutni direktorijum, ranija identifikacija (*statefull*)

FTP komande i odgovori

Komande:

- ❑ Komande se šalju kao 7 bitni ASCII tekst preko kontrolne TCP konekcije, identično kao HTTP.
- ❑ `USER ime`
- ❑ `PASS lozinka`
- ❑ `LIST` vraća spisak fajlova u direktorijumu
- ❑ `RETR imefajla` povlači fajl
- ❑ `STOR imefajla` smješta fajl na udaljeni host

Odgovori

- ❑ Odgovori sadrže statusne kodove i fraze (kao u HTTP)
- ❑ `331 Username OK, password required`
- ❑ `125 data connection already open; transfer starting`
- ❑ `425 Can't open data connection`
- ❑ `452 Error writing file`

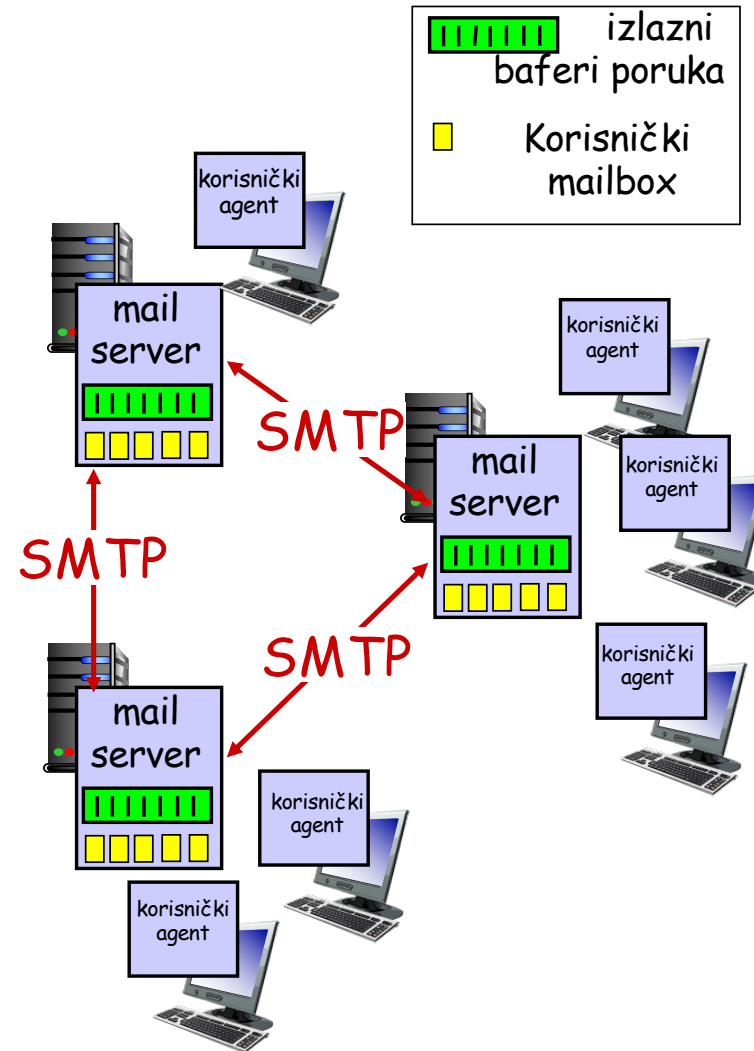
Elektronska Pošta

Tri glavne komponente:

- ❑ korisnički agenti
- ❑ mail serveri
- ❑ SMTP (*Simple Mail Transfer Protocol*)

Korisnički Agent

- ❑ mail reader
- ❑ sastavljanje, editovanje i čitanje mail poruka
- ❑ Outlook, iPhone mail client, ...
- ❑ odlazne, dolazne poruke se čuvaju na hostu



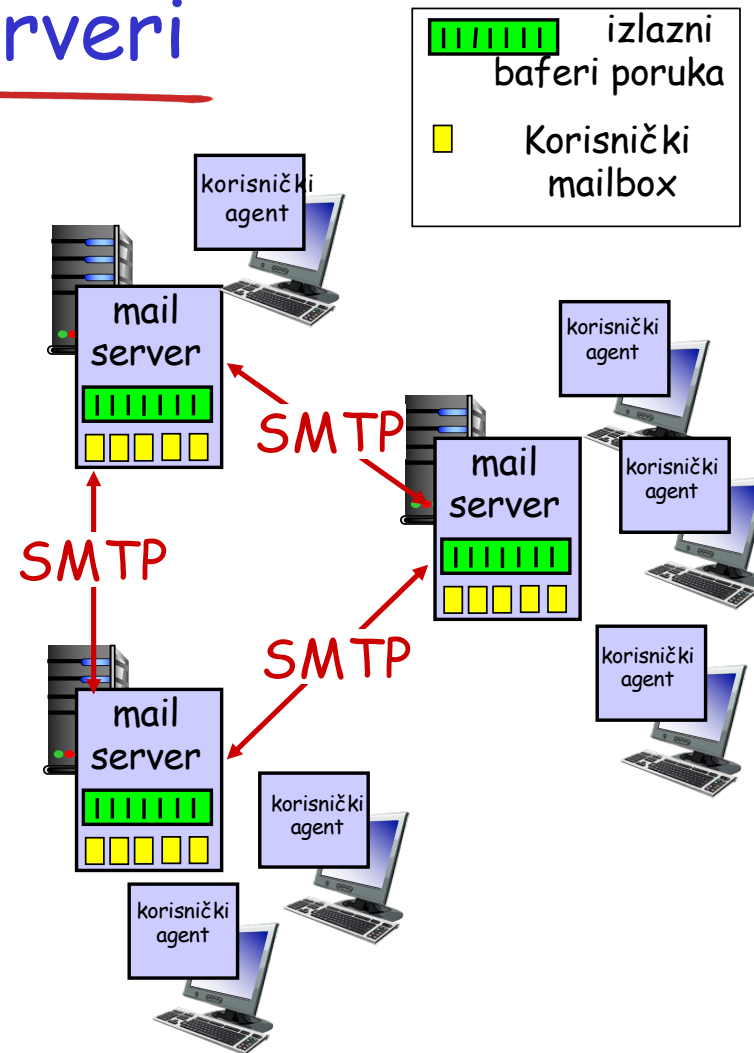
Elektronska Pošta: mail serveri

Mail Serveri

- Mailbox sadrži dolazne poruke korisnika
- Red čekanja odlaznih poruka koje trebaju da se pošalju

SMTP protokol između mail servera za slanje email poruka

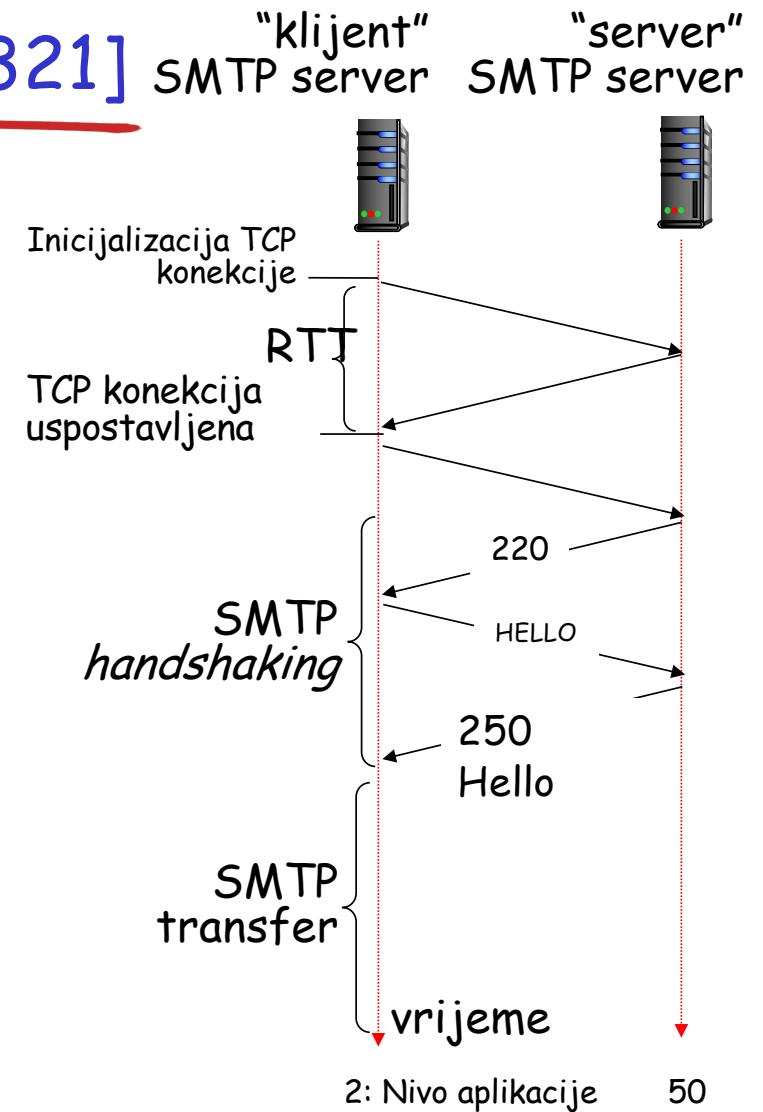
- klijent: slanje mail serveru
- “server”: prijem sa mail servera



Elektronska Pošta: SMTP [RFC 5321]

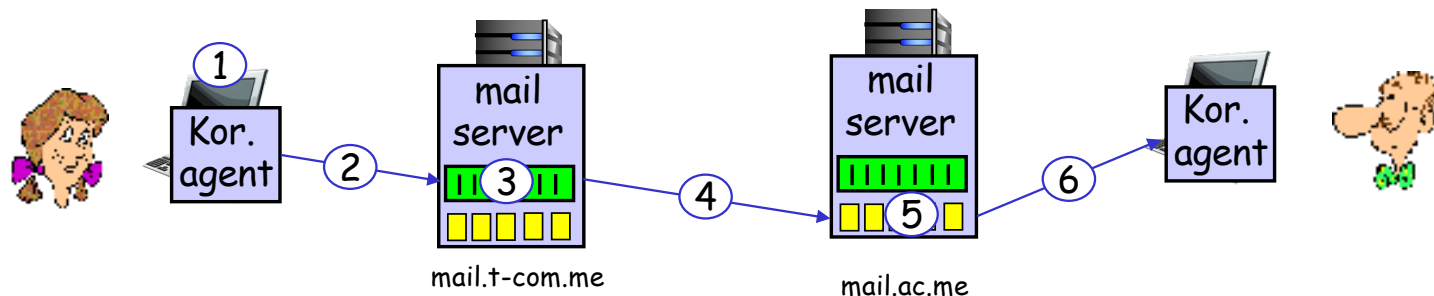
- koristi TCP za pouzdani transfer email poruke od klijenta do servera po portu 25
- direktan transfer: od servera pošiljaoca do servera primaoca
- Tri faze transfera
 - *SMTP handshaking* (upoznavanje)
 - SMTP prenos poruke
 - SMTP zatvaranje
- komanda/odgovor interakcije
 - komande: ASCII tekst
 - odgovor: status kod ili fraza
- Poruke moraju biti u potpunosti 7-bitne ASCII.

Zašto? Zašto je ovo danas problematično? Što mislite kako je kod HTTP-a?



Scenario slanja poruke

- 1) Korisnik A koristi korisnički agent da sastavi poruku i adresira je na korisnikb@ac.me
- 2) Korisnički agent korisnika A šalje poruku njenom mail serveru; poruka se smješta u red čekanja
- 3) Klijentska strana SMTP otvara TCP konekciju sa mail serverom korisnika B
- 4) SMTP klijent šalje poruku korisnika A preko TCP konekcije
- 5) Mail server korisnika B prima poruku i SMTP-ov serverski dio smješta poruku u mailbox Korisnika B
- 6) Korisnik B aktivira svoj korisnički agent da pročita poruku



Primjer SMTP interakcije

```
S: 220 mail.ac.me
C: HELO mail.t-com.me
S: 250 Hello mail.t-com.me, pleased to meet you
C: MAIL FROM: <korisnika@mail.t-com.com>
S: 250 korisnika@mail.t-com.me... Sender ok
C: RCPT TO: <korisnikb@ac.me>
S: 250 korisnikb@ac.me ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C:   How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 ac.me closing connection
```

SMTP: zapažanja

- ❑ SMTP koristi perzistentne TCP konekcije
- ❑ SMTP zahtijeva poruke koje moraju biti sedmobitnom ASCII formatu
- ❑ SMTP server koristi `CRLF.CRLF` da odredi kraj poruke

Poređenje sa HTTP:

- ❑ HTTP: "pull"
- ❑ SMTP: "push"
- ❑ Oba imaju ASCII komande/odgovore, kodove statusa, ali se razlikuju po tijelima poruke.
- ❑ HTTP: svaki objekat se smješta u sopstvenoj poruci odgovora
- ❑ SMTP: više objekata se šalje u višedjelnoj (*multipart*) poruci

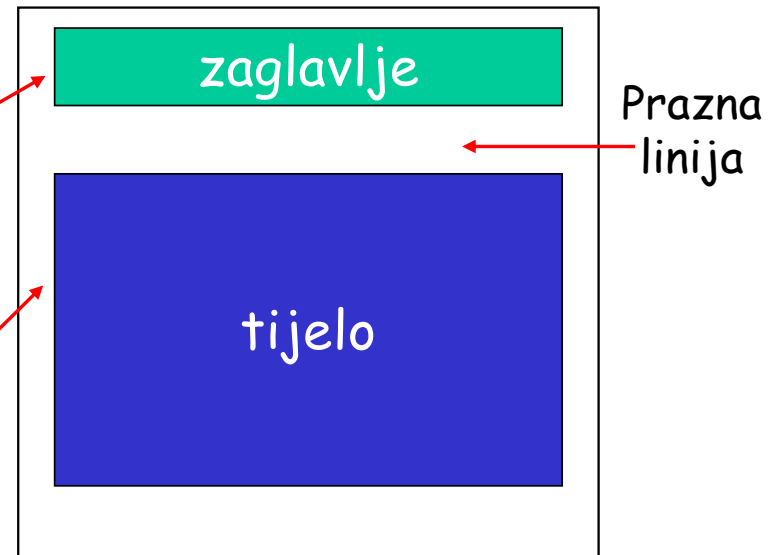
Format mail poruke (RFC 5322)

SMTP: protokol za razmjenu email poruka
RFC 5322: **standard za format tekstualnih poruka**

- Zaglavlja linija, npr.,
 - To:
 - From:
 - Subject:

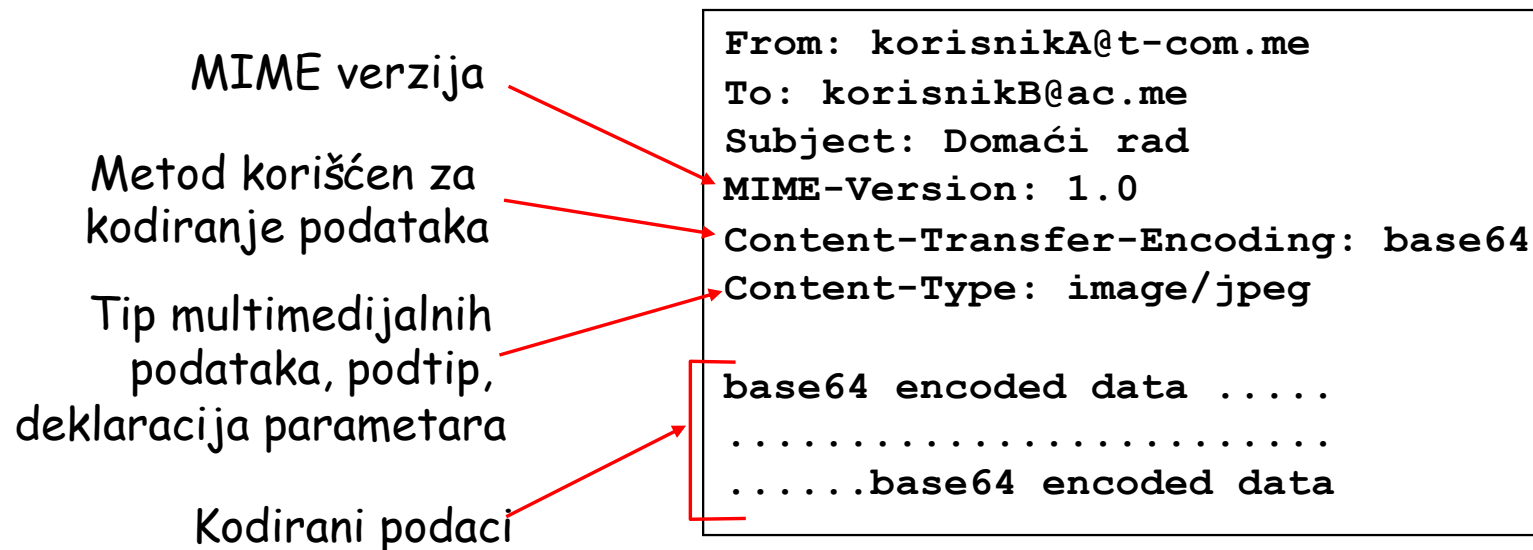
Različito od SMTP komandi SMTP MAIL FROM, RCPT TO:!

- tijelo
 - poruka, samo ASCII karakteri



Format poruke: multimedija

- MIME: multimedia mail extension, RFC 2045, 2046, 2047, 2048, 2049
- Dodatne linije u zaglavlju poruke deklariraju tip MIME sadržaja



MIME tipovi (Tip sadržaja: tip/podtip; parametri)

Tekst

- Primjeri podtipova: `plain`, `html`

Slike

- Primjeri podtipova : `jpeg`, `gif`

Audio

- Primjeri podtipova : `basic` (8-bit mu-law kodiranje), `32kadpcm` (32 kb/s kodiranje)

Video

- Primjeri podtipova : `mpeg`, `quicktime`

Aplikacije

- Drugi podaci koji moraju biti obrađeni odgovarajućim programom prije “gledanja”
- Primjeri podtipova : `mword`, `octet-stream`

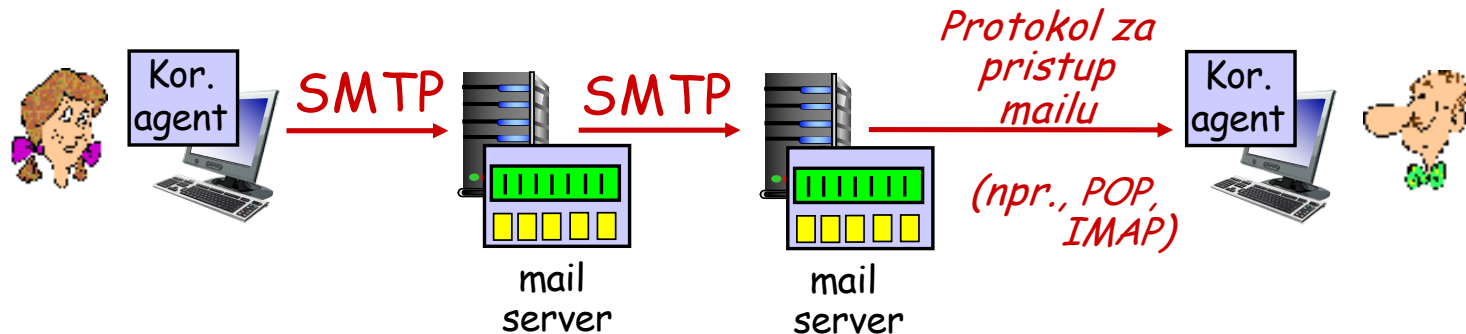
Višedjelni tip

```
From: korisnikA@t-com.me
To: korisnikB@ac.me
Subject: Picture
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=StartOfNextPart
```

```
--StartOfNextPart
Dear Bob, Please find a picture of me.
--StartOfNextPart
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.....base64 encoded data
--StartOfNextPart
Do you want the recipe?
```



Protokoli Mail pristupa



- ❑ SMTP: predaja/smještanje e-mail poruka na serveru primaoca
- ❑ **Protokol za pristup mailu: povlačenja e-mail sa servera**
 - POP: Post Office Protocol [RFC 1939]
 - autorizacija (agent <-->server) i povlačenje sadržaja
 - Port 110
 - IMAP: Internet Mail Access Protocol [RFC 3501]
 - Više funkcija (složeniji od POP protokola)
 - Port 143
 - Manipulacija sačuvanim porukama na serveru
 - HTTP: gmail, Hotmail, Yahoo! Mail, nude web interfejs za SMTP slanje poruka i IMAP/POP prijem poruka.

DNS (Domain Name System)

Ljudi imaju mnogo dokumenata za identifikaciju:

- Ime/prezime, JMBG, broj pasoša,...

Internet hostovi, ruteri:

- IP adresa (32 bita) - koristi se za adresiranje datagrama
- "ime" servera, npr., mail.ac.me - koriste ga ljudi

P: Kako mapirati IP adrese i imena?

Domain Name System:

- **Distribuirana baza podataka** implementirana kao hijerarhija velikog broja **servera imena**
- Protokol nivoa aplikacije
- Hostovi, ruteri, serveri imena komuniciraju radi utvrđivanje imena (adresa/ime translacija)
 - napomena: ključna Internet funkcija, implementirana kao protokol nivoa aplikacije
 - Kompleksnost na "ivici" mreže
- Port 53,
- UDP,
- RFC 1034 i 1035.

DNS

Zašto ne centralizovani DNS?

- ❑ Jedna tačka otkaza
- ❑ Obim saobraćaja
- ❑ Centralizovana baza podataka
- ❑ Nadzor

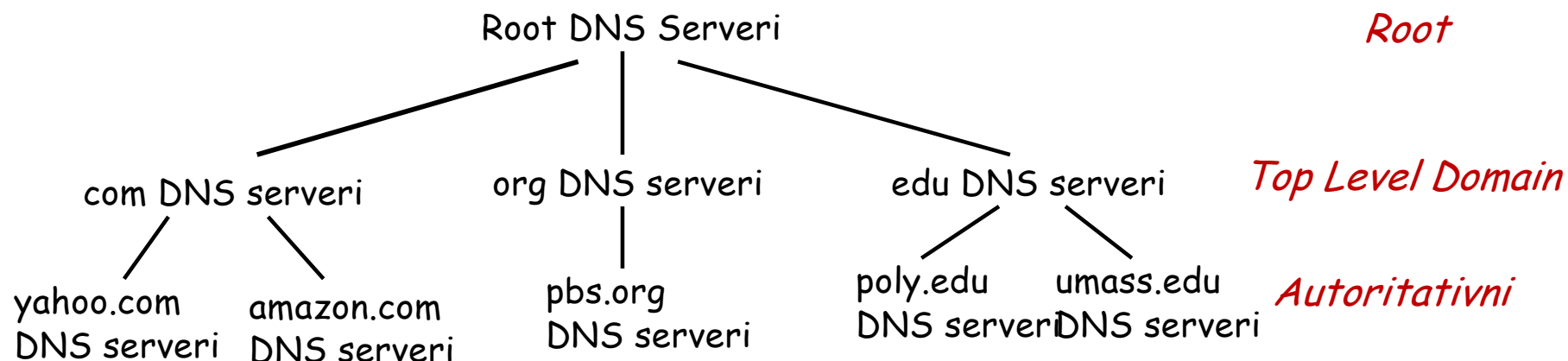
Ne odgovara!

- ❑ Comcast DNS serveri: 600 milijardi DNS upita/dnevno
- ❑ Akamai DNS serveri: 2.2 biliona DNS upita/dnevno

DNS servisi

- ❑ Translacija imena hosta u IP adresu
- ❑ Host *aliasing*
 - Kanonska (www.yahoo.akadns.com) i alias imena (www.yahoo.com)
- ❑ Mail server *aliasing*: "mail.ac.me" u "ac.me"
- ❑ Distribucija opterećenja
 - Replikacija Web servera: setovanje IP adresa za jedno kanoničko ime

Distribuirana i hijerarhijska baza podataka



Klijent želi IP adresu za "www.amazon.com":

- ❑ Klijent pita root server da mu pronade com DNS server
- ❑ Klijent pita jedan od com DNS servera da pronade amazon.com DNS server
- ❑ Klijent pita amazon.com DNS server da mu pošalje IP adresu www.amazon.com

DNS: "Root" serveri imena

- Kontaktiraju ih lokalni serveri imena kada ne mogu da pronadu ime
- root server imena:
 - kontaktira autoritativni server imena ako mapiranje nije poznato
 - dobija mapiranje
 - vraća mapiranje lokalnom serveru imena



postoji 13 svetskih "root" servera imena pri čemu je svaki repliciran mnogo puta (preko 200 servera u US)
www.root-servers.org

TLD i Autorizacioni serveri

- Top-level domain (TLD) server su odgovorni za com, org, net, edu, etc, i sve *top-level* domene zemalja uk, fr, ca, jp, me...
 - Network Solutions: nadzire servere za .com, .net TLD
 - Educause: .edu TLD
- Autoritativni DNS serveri: DNS serveri organizacije obezbjeđuju mapiranja imena hostova u IP adrese za servere organizacije (npr., Web i mail).
 - Može biti nadziran od strane organizacije ili servis provajdera

Lokalni DNS

- ❑ Striktno ne pripada hijerarhiji
- ❑ Svaki ISP (rezidencijalni ISP, kompanijski, univerzitet) ima jedan.
 - Još se zove “default DNS”
- ❑ Kada host napravi DNS upit, upit se šalje na njegov lokalni DNS server
 - Ponaša se kao DNS proxy, prosleđuje upite u hijerarhiju.
- ❑ Svaki ISP ima lokalni DNS name server
- ❑ Kako pronaći sopstveni?
 - ❑ MacOS: % scutil --dns
 - ❑ Windows: >ipconfig /all

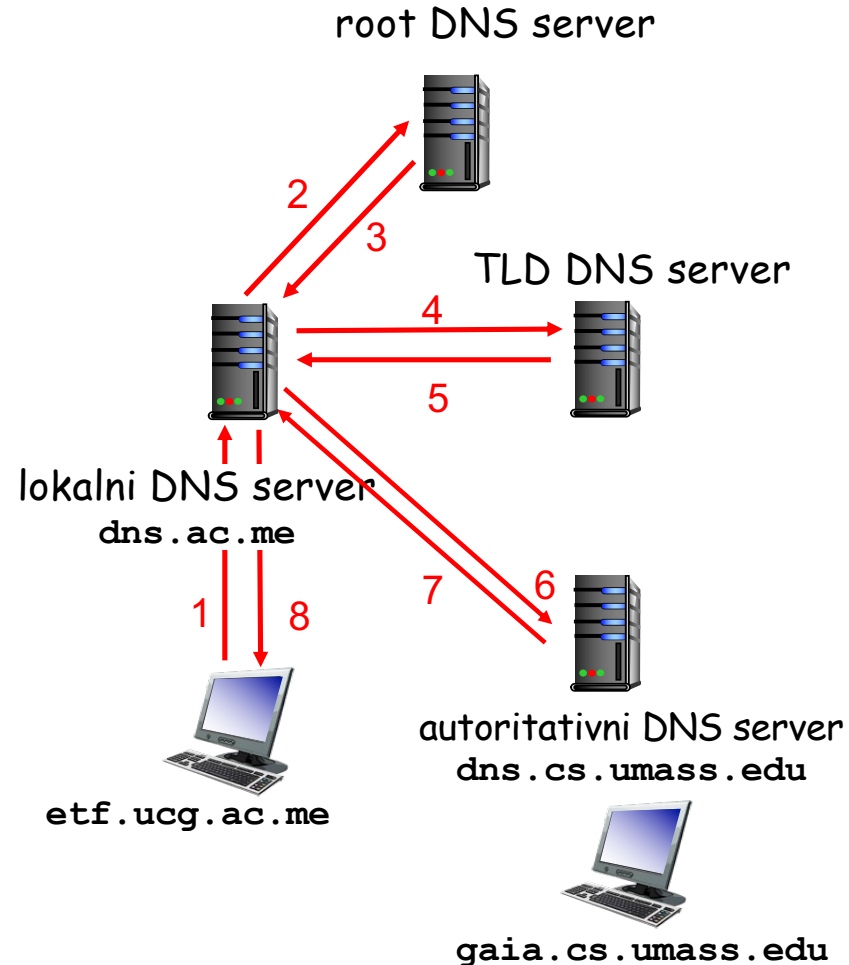
Primjer 1

- Host `cis.ac.me` želi IP adresu za `gaia.cs.umass.edu`

Iterativni upit:

Kontaktirani server odgovara sa imenom servera kojeg treba kontaktirati

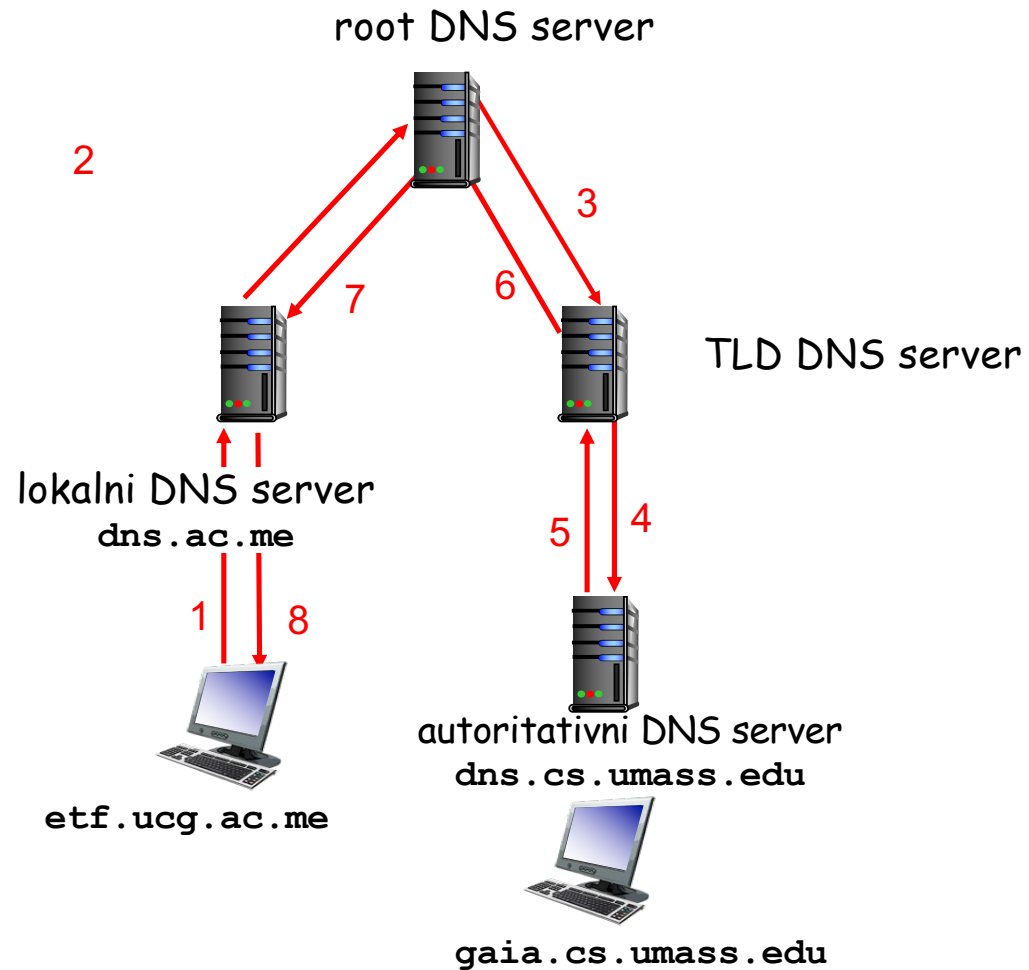
“Neznam ovo ime, ali pitaj ovaj server”



Primjer 2

Rekurzivni upit:

- Stavlja problem utvrđivanja imena na kontaktirani DNS
- *Veliko opterećenje?*



DNS: "caching" i "updating"

- Kada server imena definiše mapiranje on ga čuva:
 - Pri čemu se sačuvani podaci posle izvjesnog timeout perioda brišu
 - TLD serveri su tipično sačuvani u lokalnim DNS-ovima
 - Na taj način se root name serveri rijetko posjećuju
- *update/notify* mehanizmi su definisani od IETF za slučaje kada keširani podaci nisu ažurni
 - RFC 2136

DNS zapisi

DNS:

distribuirana baza podataka koja sadrži zapise resursa (resource records (RR))

RR format: (name, value, type, ttl)

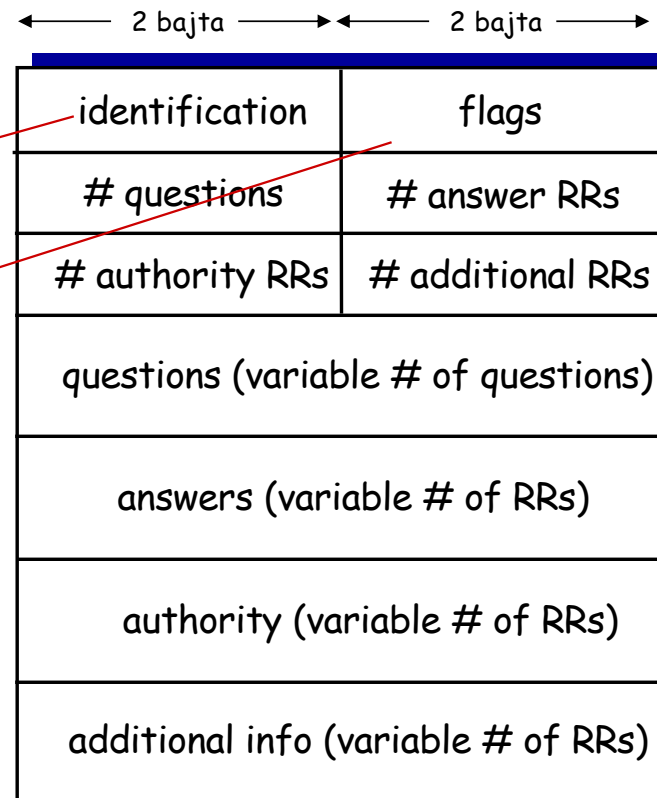
- Type=A
 - name je ime hosta
 - value je IP adresa
- Type=NS
 - name je domen
 - value je IP adresa autoritativnog name servera za ovaj domen
- Type=CNAME
 - name je alias ime nekog “kanoničkog” (stvarnog) imena
 - value je kanoničko ime
 - `www.ibm.com` je u stvari `e2874.x.akamaiedge.net`
- Type=MX
 - value je kanonično ime mail servera čiji je name alias ime.

DNS protokol, poruke

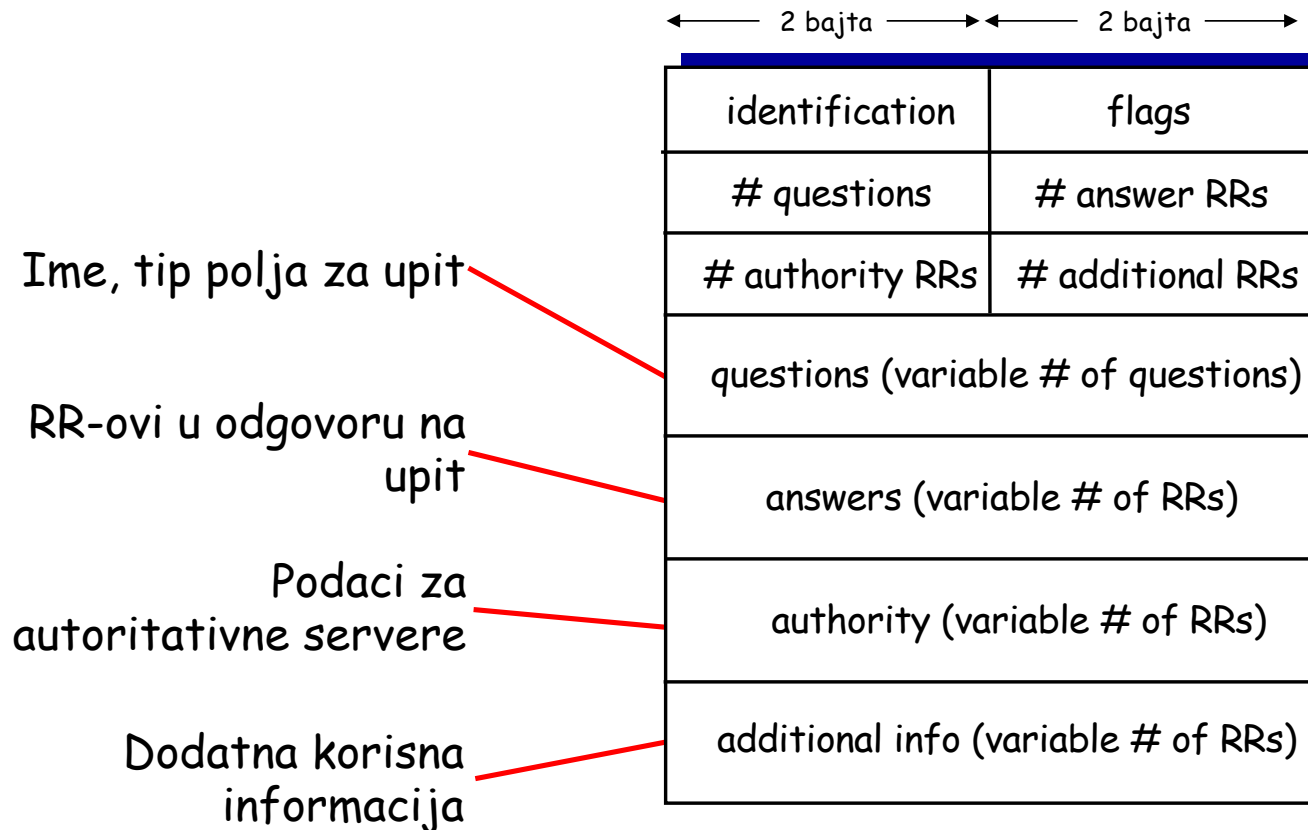
Upiti i odgovori, imaju isti format

Zaglavlje poruke

- ❑ identifikacija: 16 bitni broj za upit, odgovor na upit koristi isti broj
- ❑ oznake:
 - Upit ili odgovor
 - Poželjne rekurzije
 - Dostupne rekurzije
 - Odgovor (broj pojavljivanja tipova)



DNS protokol, poruke



Ubacivanje zapisa u DNS

- ❑ Primjer: osnovan je novi start up “Network Utopia”
- ❑ Registracija imena networkutopia.com u registar (Network Solutions)
 - Potrebno je dostaviti registru imena i IP adrese autoritativnog name server (primarnog i sekundarnog)
 - Registar ubacuje dva RR u sve .com TLD servere:

(networkutopia.com, dns1.networkutopia.com, NS)

(dns1.networkutopia.com, 212.212.212.1, A)

- ❑ Postavlja u autoritativni server Type NS zapis za www.networkutopia.com i Type A zapis za dns1.networkutopia.com
- ❑ Kreira autoritativni lokalni server sa IP adresom 212.212.212.1
 - type A zapis za www.networkutopia.com
 - type MX zapis za networkutopia.com
- ❑ Kako se može saznati IP adresa nekog Web sajta?
- ❑ Kako poslati DNS poruku upita direktno DNS serveru?
 - **nslookup** komanda sa MSDOS comand prompta
 - Pomoću odgovarajućih sajtova

Napadi na DNS

DDoS napadi

- ❑ Bombardovanje root servera prekomjernim saobraćajem
 - Neuspješan do sada
 - Filtriranje saobraćaja
 - Lokalni DNS serveri keširaju IP adrese TLD servera, što obezbjeđuje zaobilazanje root servera
- ❑ Bombardovanje TLD servera
 - Mnogo opasnije!

Indirektni napadi

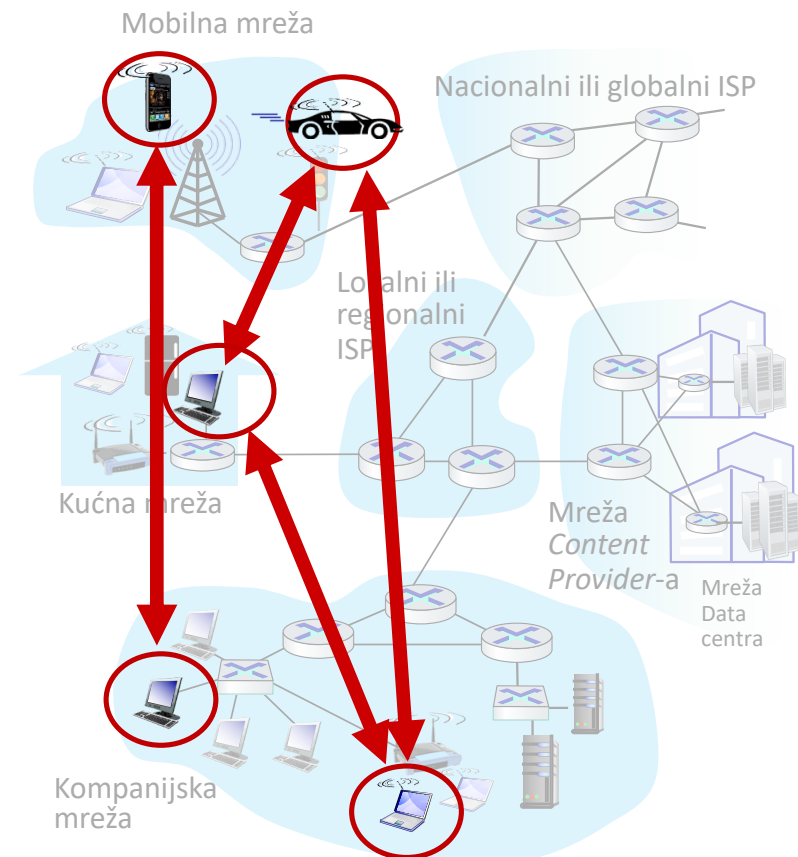
- ❑ Man-in-middle
 - Presrijetanje upita
- ❑ DNS *poisoning*
 - Slanje pogrešnih odgovora povezanih sa DNS serverom, koji se keširaju

Korišćenje DNS za DDoS

- ❑ Slanje upita sa ukradenih IP adresa: cilj je IP
- ❑ Zahtijeva potvrdu

P2P (*Peer-to-peer*) arhitektura

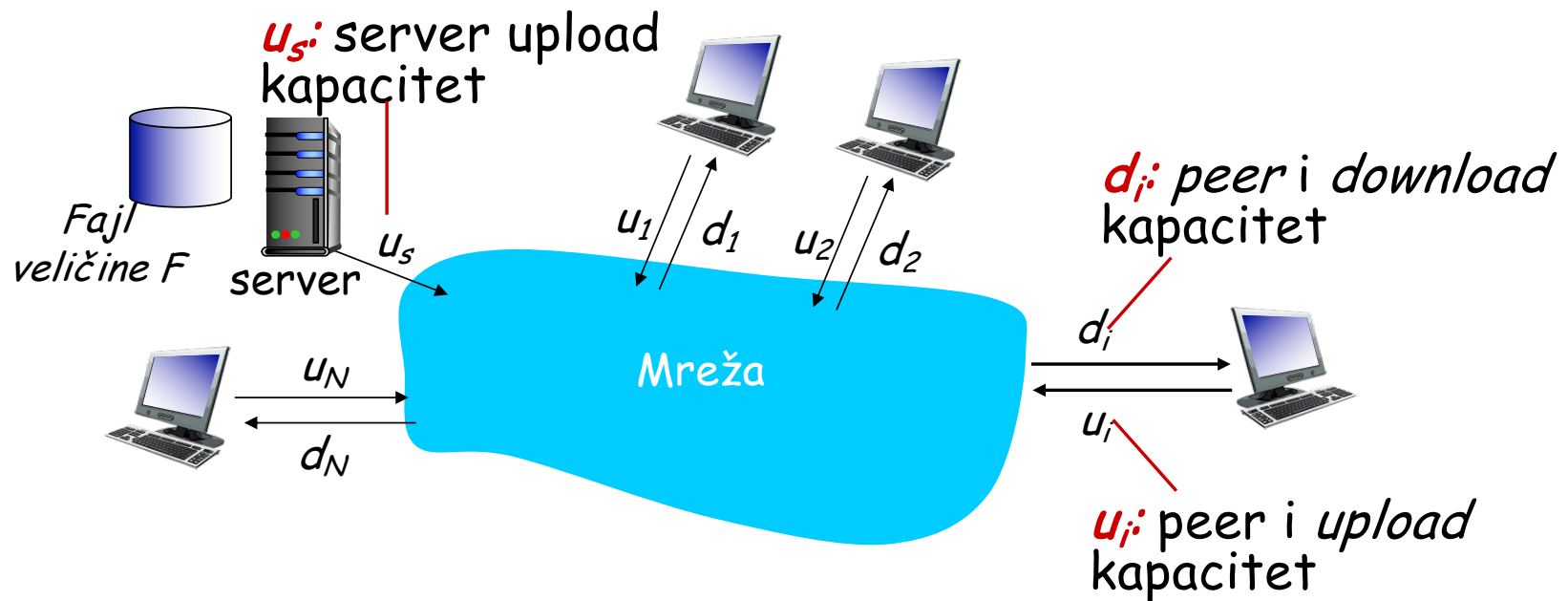
- ❑ Server nije uvijek aktivan
- ❑ Krajnji sistemi direktno komuniciraju
- ❑ *Peer*-ovi zahtijevaju servis od drugih *peer*-ova, nudeći za uzvrat servis ostalim *peer*-ovima
 - ❑ Samoskalabilnost jer novi *peer*-ovi donose nove servisne kapacitete ali i zahtjeve
- ❑ *peer* se povremeno povezuje i mijenja IP adresu
 - ❑ Kompleksno upravljanje
- ❑ P2P *file sharing* (BitTorrent), *streaming* (KanKan), VoIP (Skype)



Distribucija fajla: Klijent-server ili P2P

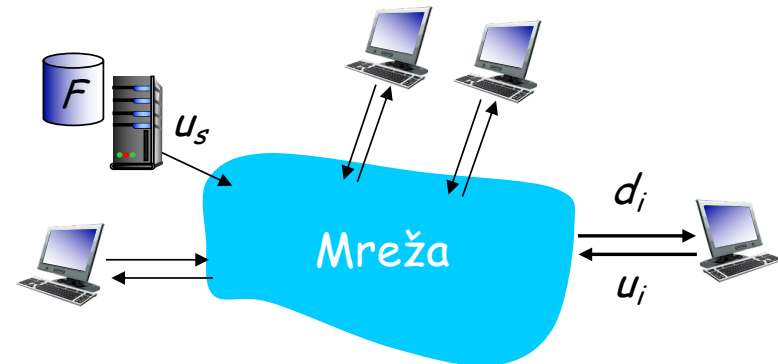
Za koje vrijeme se file veličine F može distribuirati od servera do N peer-ova?

- upload/download kapaciteti *peer*-ova su limitirani



Vrijeme distribucije fajla: klijent-server

- ❑ Server mora sukcesivno poslati N kopija fajla:
 - ❑ Vrijeme za slanje jedne kopije: F/u_s
 - ❑ Vrijeme za slanje N kopija: NF/u_s
- ❑ Svaki klijent mora primiti kopiju fajla
 - ❑ d_{min} = min *download* brzina klijenta
 - ❑ maksimalno vrijeme klijentovog povlačenja fajla je: F/d_{min}

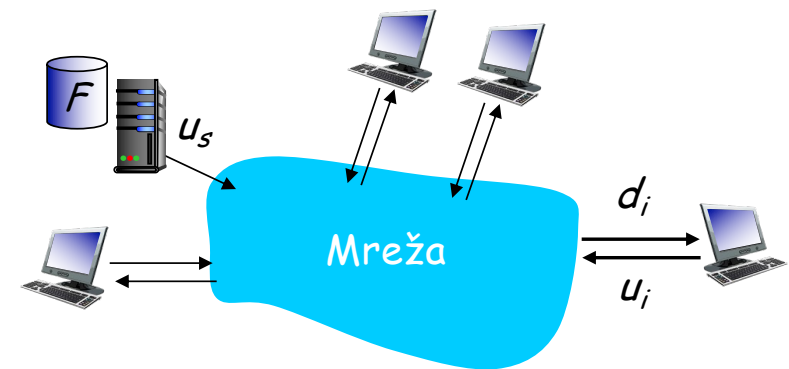


$$D_{c-s} \geq \max\{NF/u_s, F/d_{min}\}$$

Linearno raste sa N

Vrijeme distribucije fajla: P2P

- ❑ Server mora poslati makar jednu kopiju fajla:
 - ❑ Vrijeme za slanje fajla: F/u_s
- ❑ Svaki klijent mora povući kopiju fajla
 - ❑ maksimalno vrijeme klijentovog povlačenja fajla: F/d_{\min}
- ❑ Svi klijenti moraju povući NF bita
 - ❑ Maksimalna brzina slanja koja ograničava brzinu povlačenja je $u_s + \sum u_i$

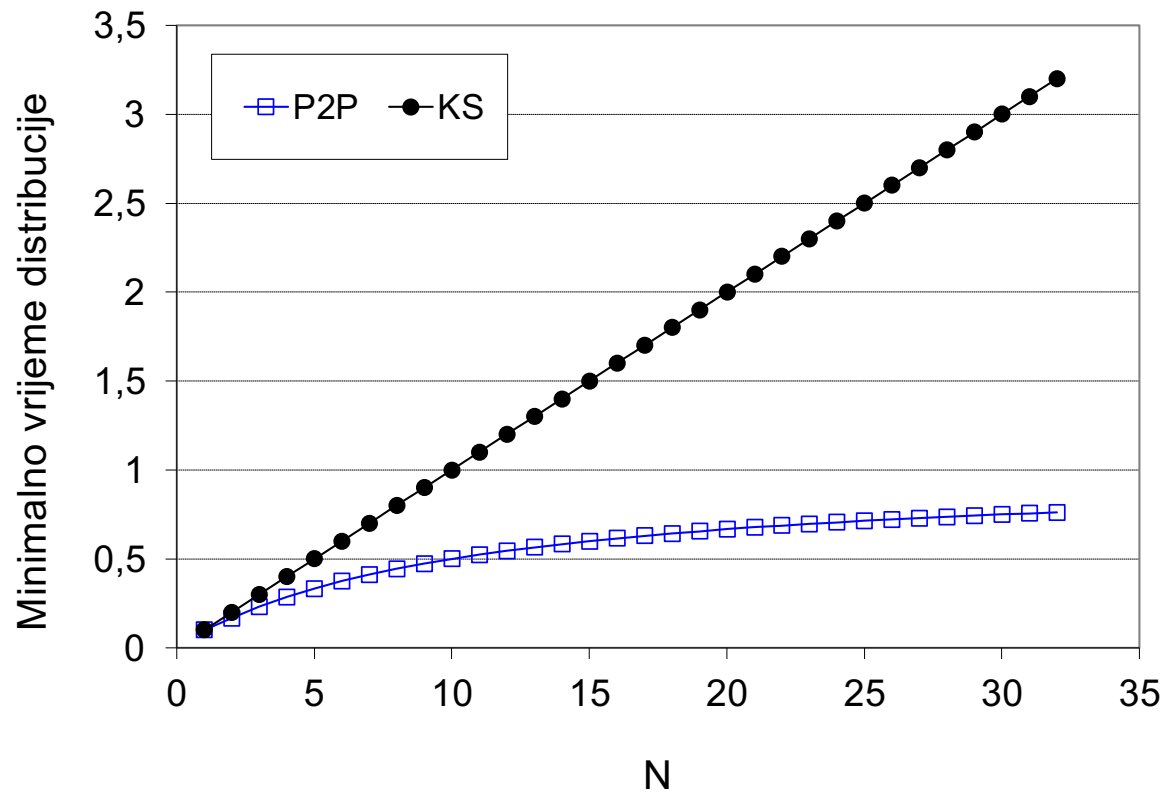


$$D_{P2P} \geq \max\{F/u_s, F/d_{\min}, NF/(u_s + \sum u_i)\}$$

Raste linearno sa N ali svaki peer dodaje svoj kapacitet

Jedno poređenje KS i P2P

Klijentova brzina uploada je u , $F/u = 1h$, $u_s = 10u$, $d_{min} \geq u_s$

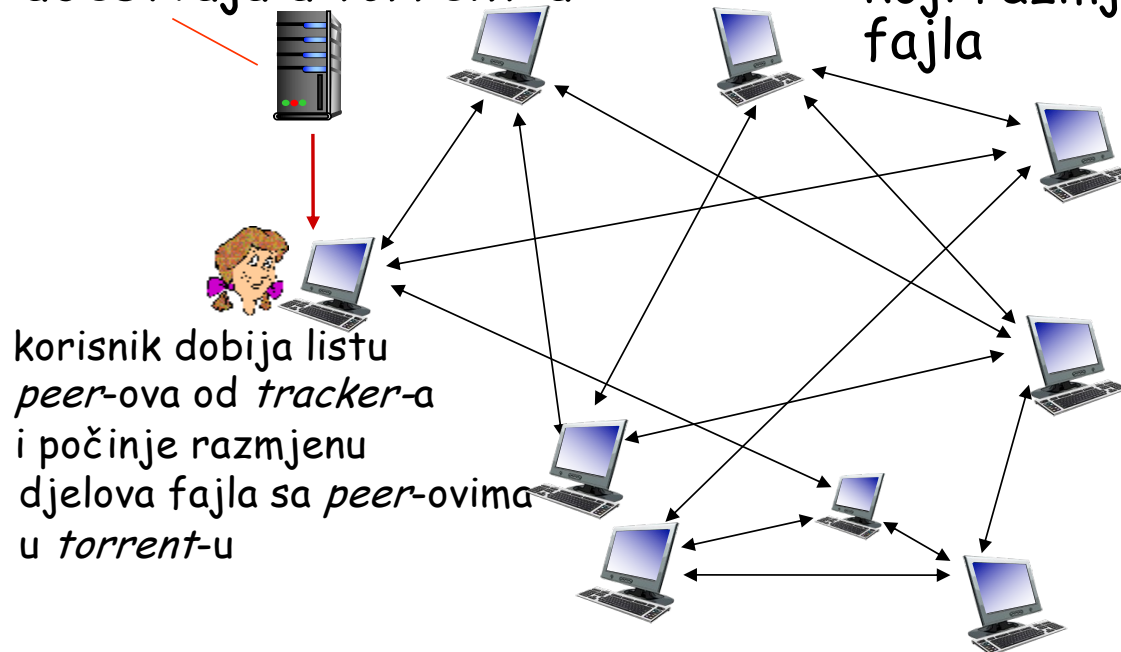


P2P distribucija fajla: BitTorrent

- ❑ Fajl se dijeli na djelove (*chunks*) veličine 256kB, 512kB ili 1MB
- ❑ *Peer*-ovi u *torrent*-u šalju/primaju djelove fajla

Tracker: prati *peer*-ove koji učestvuju u *torrent*-u

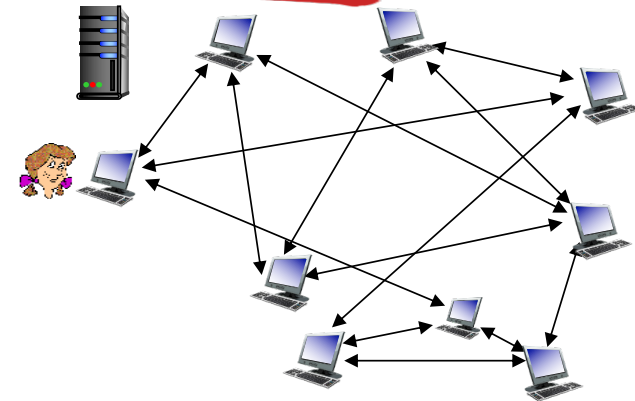
Torrent: grupa *peer*-ova koji razmjenjuju djelove fajla



korisnik dobija listu *peer*-ova od *tracker*-a i počinje razmjenu djelova fajla sa *peer*-ovima u *torrent*-u

P2P distribucija fajla: BitTorrent

- ❑ *peer* se pridružuje *torrent-u*:
 - ❑ Nema djelova fajla, ali će ih vremenom prikupiti od ostalih *peer-ova*
 - ❑ Registruje se kod *tracker-a* kako bi dobio listu *peer-ova* i povezuje se na podskup *peer-ova* (“susjedi”)
- ❑ dok povlači, *peer* šalje djelove fajla drugim *peer-ovima*
- ❑ *peer* može promijeniti *peer-ove* sa kojim razmjenjuje djelove
- ❑ *peer-ovi* mogu dolaziti i odlaziti
- ❑ Jednom kada *peer* povuče kompletan fajl, može napustiti ili ostati u *torrent-u*



BitTorrent: zahtijevanje i slanje djelova fajla

Zahtijevanje djelova:

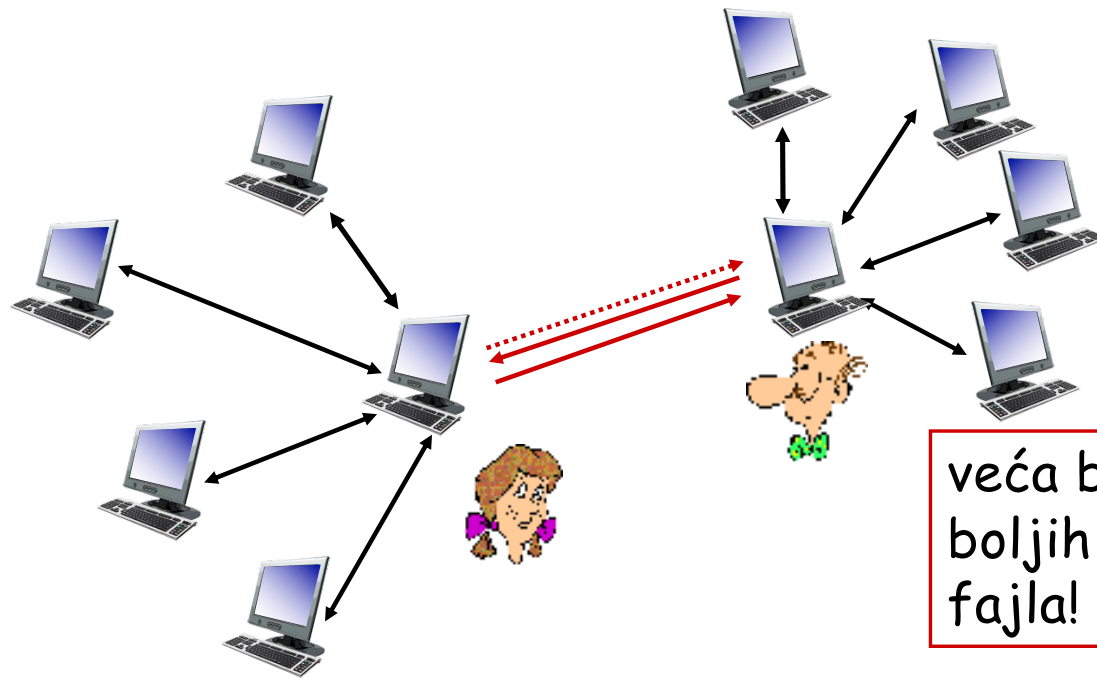
- ❑ U bilo kojem trenutku, različiti *peer*-ovi mogu imati različite podskupove djelova fajla
- ❑ Periodično *peer* pita svaki drugi *peer* za listu djelova koje posjeduje
- ❑ *Peer* zahtijeva nedostajuće djelove tražeći prvo one koji su najrjeđi

Slanje djelova:

- ❑ *Peer* šalje djelove fajla četvorici *peer*-ova koji mu dostavljaju djelove fajla najvećom brzinom
 - ❑ Drugi *peer*-ovi ne dobijaju djelove
 - ❑ Svakih 10s se određuje top 4
- ❑ svakih 30s: slučajno bira drugi *peer* i počinje da mu šalje djelove fajla
 - ❑ Pokušava da aktivira novog *peer*-a
 - ❑ Novi *peer* može ući u njegov top 4

BitTorrent: slanje dijelova fajla

- (1) *Peer* pokušava da aktivira drugog *peera*
- (2) *Peer* postaje jedan od Top 4 drugog *peera* tako da drugi *peer* uzvraća
- (3) Drugi *peer* postaje prvom jedan od Top 4



veća brzina slanja: pronalaženje
boljih partnera i brže dobijanje
fajla!

Video Streaming i CDN

- ❑ Prenos *streaming* video saobraćaja je glavni potrošač Internet kapaciteta danas
 - ❑ Netflix, YouTube, Amazon Prime: 80% rezidencijalnog ISP saobraćaja (2020)
- ❑ Kako doseći milijardu korisnika?

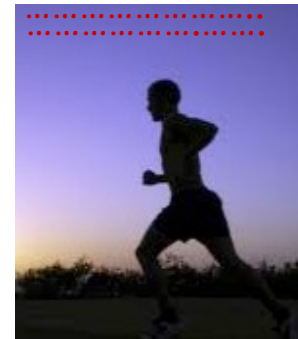


- ❑ Kako prevazići heterogenost korisnika (žični, mobilni, visoke ili niske brzine prenosa)?
- ❑ Moguće rješenje je distribuirana infrastruktura nivoa aplikacije

Multimedija: video

- ❑ Video je sekvenca slika koje se prikazuju konstantnom brzinom (npr 24 slike/s)
- ❑ Digitalna slika je matrica piksela koji se predstavljaju bitima
- ❑ Kodiranje je korišćenje redundanse u i između slika radi smanjenja broja bita koji se koriste za kodiranje slike
 - ❑ prostorno (u slici)
 - ❑ vremensko (od slike do slike)

Primjer prostornog kodiranja:
umjesto slanja više vrijednosti iste boje šalju se dvije vrijednosti : vrijednost koja odgovara boji i broj ponavljanja



frejm i

Primjer vremenskog kodiranja: umjesto slanja kompletnog frejma šalje se samo razlika između frejmova

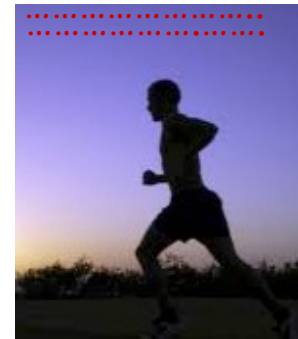


frejm $i+1$

Multimedia: video

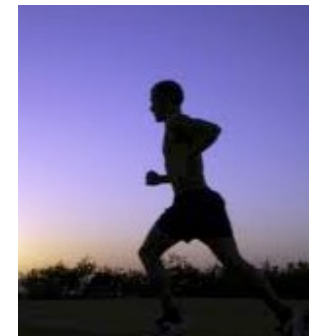
- ❑ CBR: (*constant bit rate*): video kodiran fiksnom brzinom
- ❑ VBR: (*variable bit rate*): video kodiran brzinom koja se mijenja u funkciji promjena prostornog i vremenskog kodiranja
- ❑ Primjeri:
 - ❑ MPEG 1 (CD-ROM) 1.5 Mb/s
 - ❑ MPEG2 (DVD) 3-6 Mb/s
 - ❑ MPEG4 (često korišćen na Internetu, 64 kb/s - 12 Mb/s)

Primjer prostornog kodiranja:
umjesto slanja više vrijednosti iste boje šalju se dvije vrijednosti : vrijednost koja odgovara boji i broj ponavljanja



frejm i

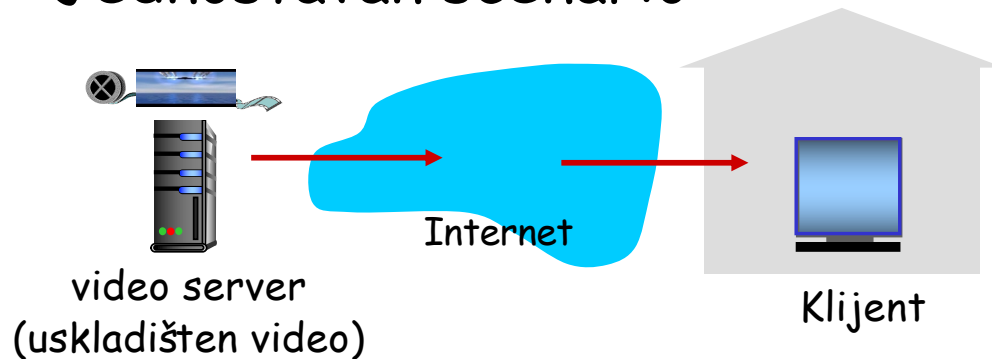
Primjer vremenskog kodiranja: umjesto slanja kompletnog frejma šalje se samo razlika između frejmova



frejm $i+1$

Streaming video

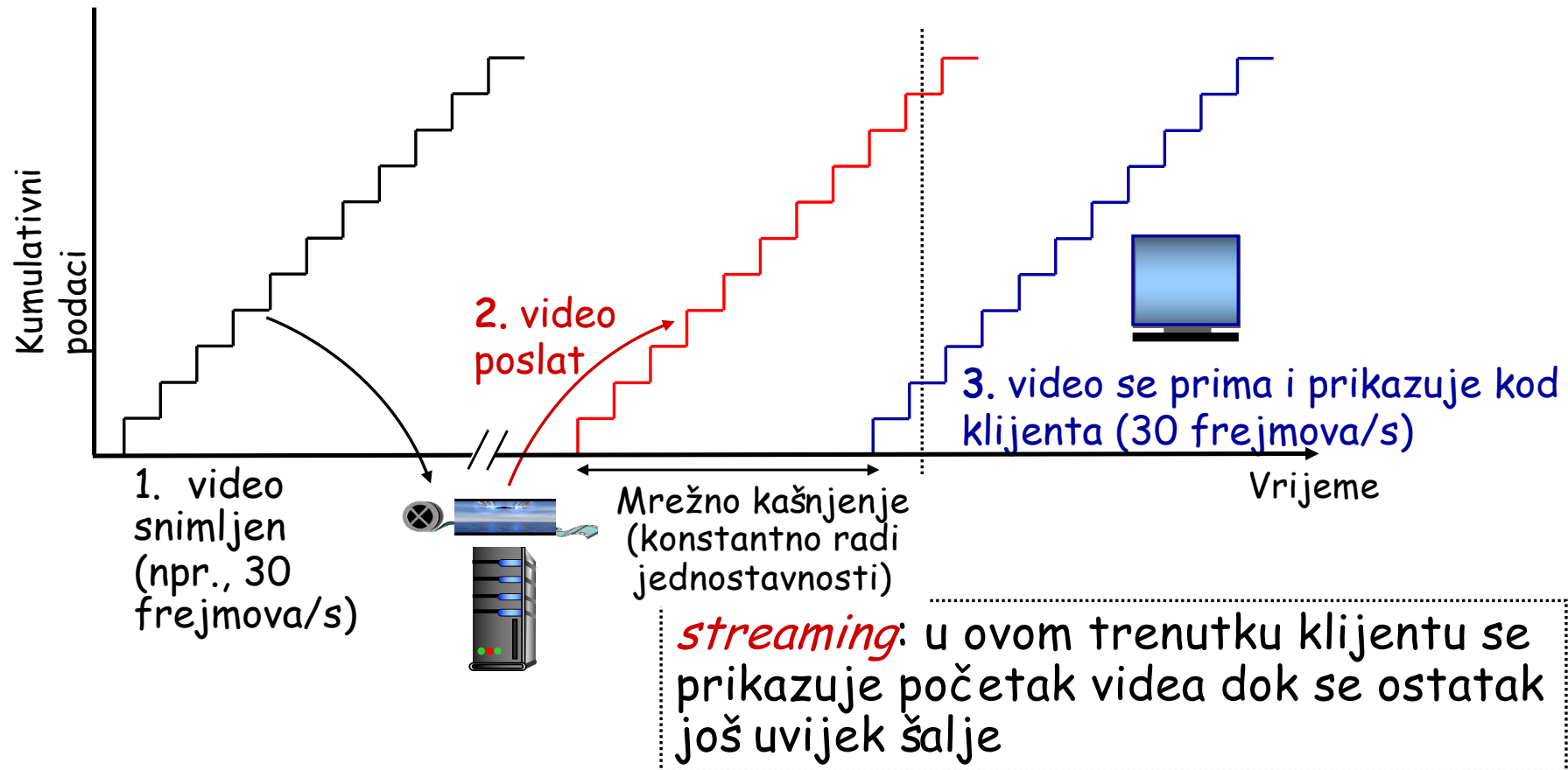
Jednostavan scenario:



Ključni izazovi:

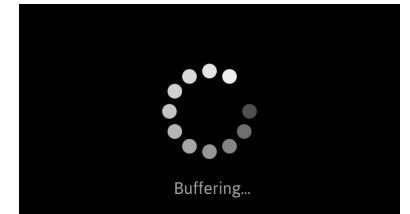
- ❑ Kapacitet mreže između servera i klijenta varira u vremenu saglasno varijaciji zagušenja u rezidencijalnoj mreži, pristupnoj mreži, jezgru mreže i video serveru
- ❑ Gubitak paketa i kašnjenje uslijed zagušenja će usporavati video i obarati kvalitet

Streaming video

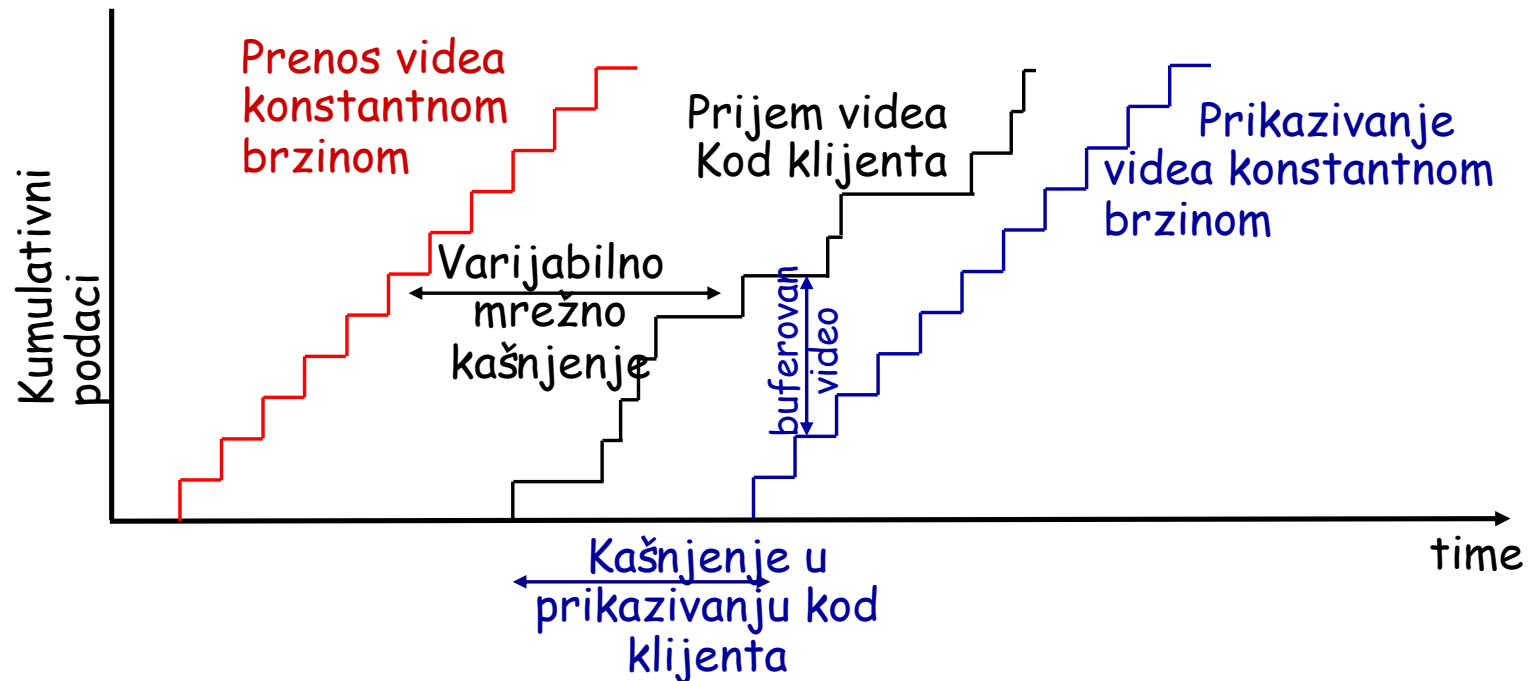


Streaming video: izazovi

- ❑ Kontinualno prikazivanje je izazov jer se mrežno kašnjenje mijenja tako da je na klijentskoj strani potreban bafer
- ❑ Drugi izazovi:
 - ❑ Klijentove interakcije (početak, pauza, naprijed, nazad,...)
 - ❑ Video paketi mogu biti izgubljeni što izaziva retransmisije



Streaming video: baferovanje



- baferovanje kod klijenta i kašnjenje prikazivanja kompenzuju mrežno kašnjenje i jitter

Streaming multimedia: DASH

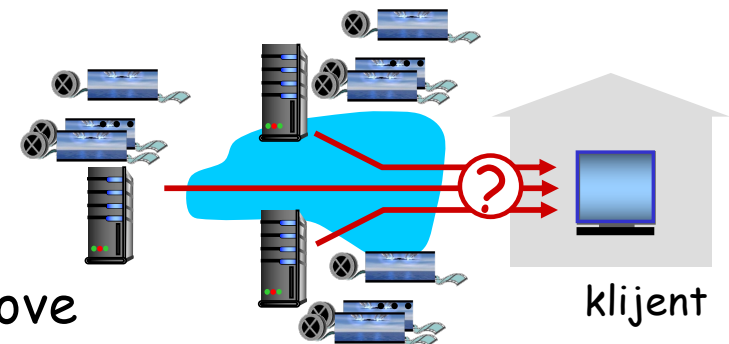
*Dynamic, Adaptive
Streaming over
HTTP*

server:

- dijeli video fajl na djelove
- svaki dio se kodira različitom brzinom
- različite brzine kodiranja su smještene u različite fajlove
- fajlovi se repliciraju u različite CDN čvorove
- manifest fajl**: obezbjeđuje URLove za različite djelove videa

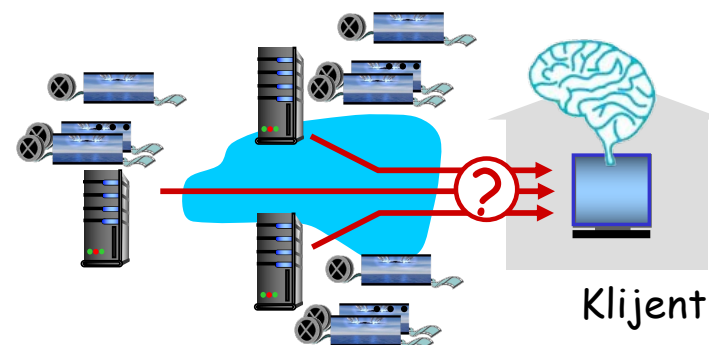
klijent:

- periodično estimira propusnost mreže između servera i klijenta
- konsultuje **manifest** zahtijevajući u jednom trenutku jedan dio video fajla
 - bira maksimalnu brzinu kodiranja koju podržava data propusnost mreže
 - može birati različite brzine kodiranja u različitim trenucima zavisno od propusnosti mreže, kao i različite servere



Streaming multimedia: DASH

- klijent određuje
 - kada se zahtijeva dio fajla tako da se ne isprazni ili ne prepuni bafer
 - zahtijevanu brzinu kodiranja (veći kvalitet znači veću potrebnu propusnost)
 - traženi dio fajla odnosno bira server sa većom propusnošću ili server do kojeg je manje kašnjenje



Streaming video = kodiranje + DASH + baferovanje prikazivanja

Content distribution networks (CDNs)

Kako slati video sadržaj izabran između mnoštva videa stotinama ili hiljadama simultanih korisnika?

Opcija 1: jedan, veliki mega server

- Jedna tačka otkaza
- Tačka zagušenje mreže
- Dugačak i vjerovatno zagušen put do udaljenih korisnika

Rješenje nije skalabilno!

Content distribution networks (CDNs)

Kako slati video sadržaj izabran između mnoštva videa stotinama ili hiljadama simultanih korisnika?

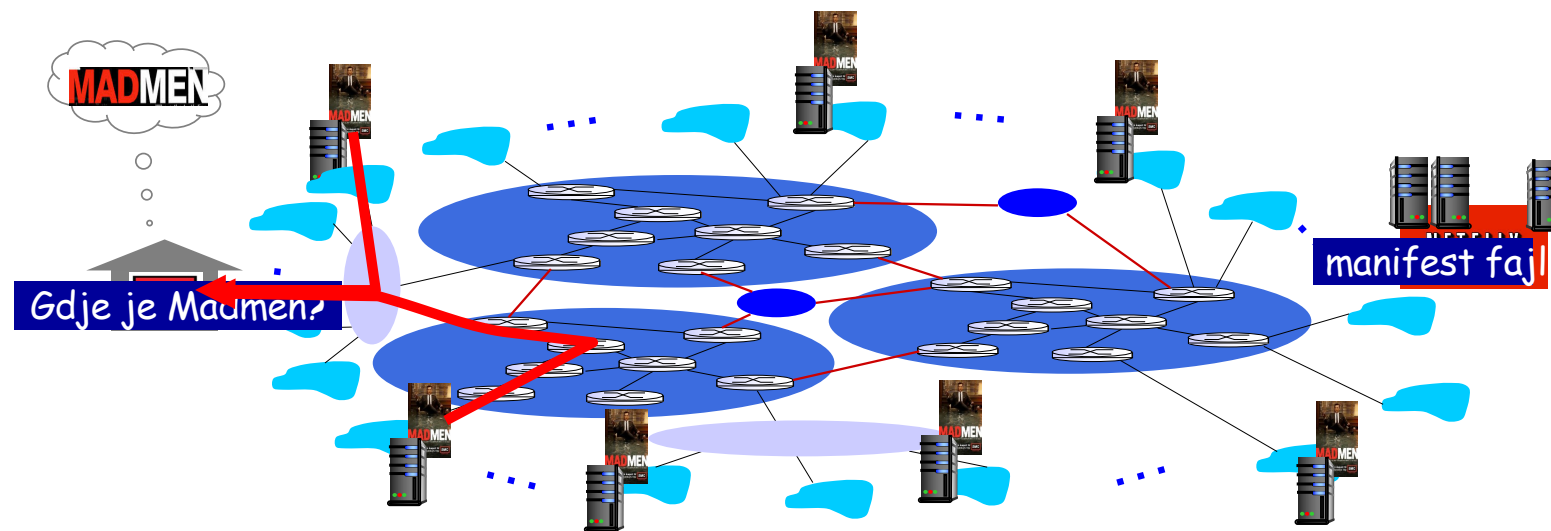
Opcija 2: smještati i slati kopije videa sa više geografski distribuiranih sajtova

- ❑ *enter deep*: smjestiti CDN servere duboko u mnogo pristupnih mreža
 - ❑ Blizu korisnicima
 - ❑ Akamai: 240,000 servera u radu u više od 120 zemalja (2015)
- ❑ *bring home*: manji broj (desetak) velikih klastera u POP-vima blizu pristupnih mreža
 - ❑ Kao kod Limelight



Content distribution networks (CDNs)

- smještaju kopije sadržaja na CDN čvorove
- Korisnik traži sadržaj, a od operatora dobija manifest
 - Koristeći manifest klijent dobija sadržaj najvećom mogućom brzinom
 - Može birati različite brzine ili kopije ako je mreža zagušena



Content distribution networks (CDNs)



OTT izazovi: Kako se "izboriti" sa Internetom?

- Koji sadržaj smjestiti u kojem CDN čvorištu?
- Sa kojeg CDN čvorišta povlačiti sadržaj? Kojom brzinom?

Pristup CDN sadržaju: Netflix

