

3: Nivo aplikacije

Nivo aplikacije

- Principi protokola nivoa aplikacije
- Web i HTTP

Igor Radusinović
Slavica Tomović
ETF, UCG

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*
- *Blockchain*
- ...

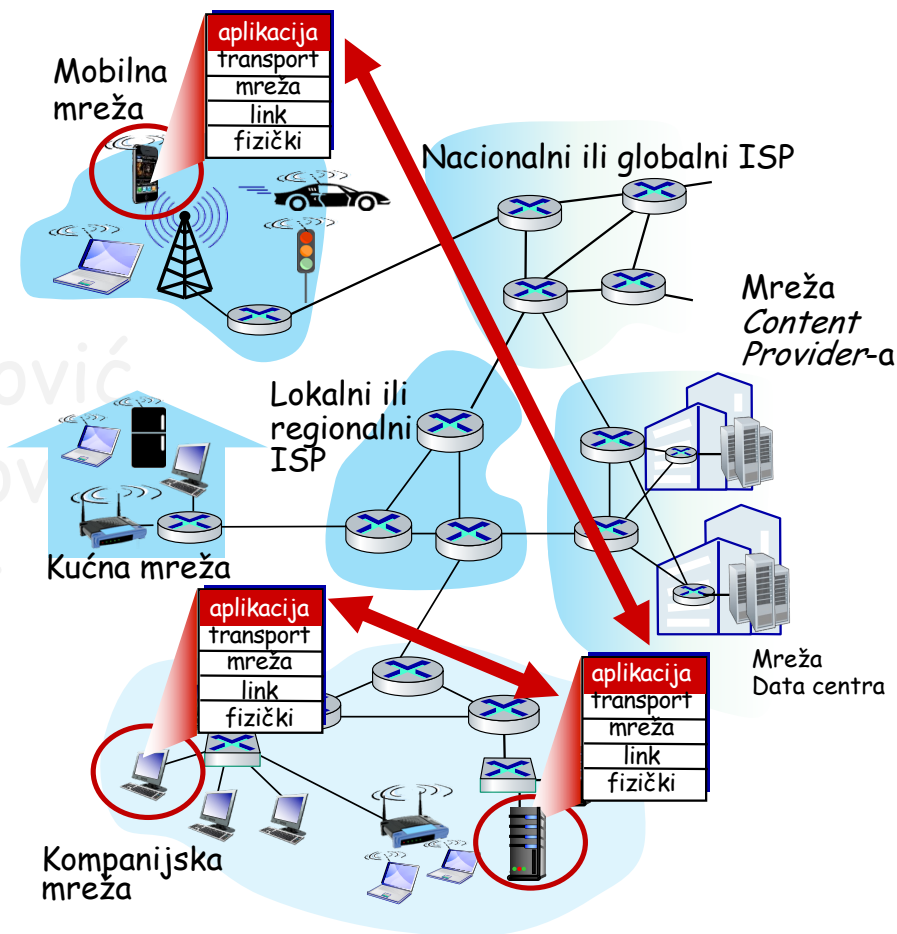
Kreiranje mrežne aplikacije

Pisanje programa koji

- se izvršavaju na različitim krajnjim sistemima i
- komuniciraju preko mreže.
- npr. web server softver komunicira 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

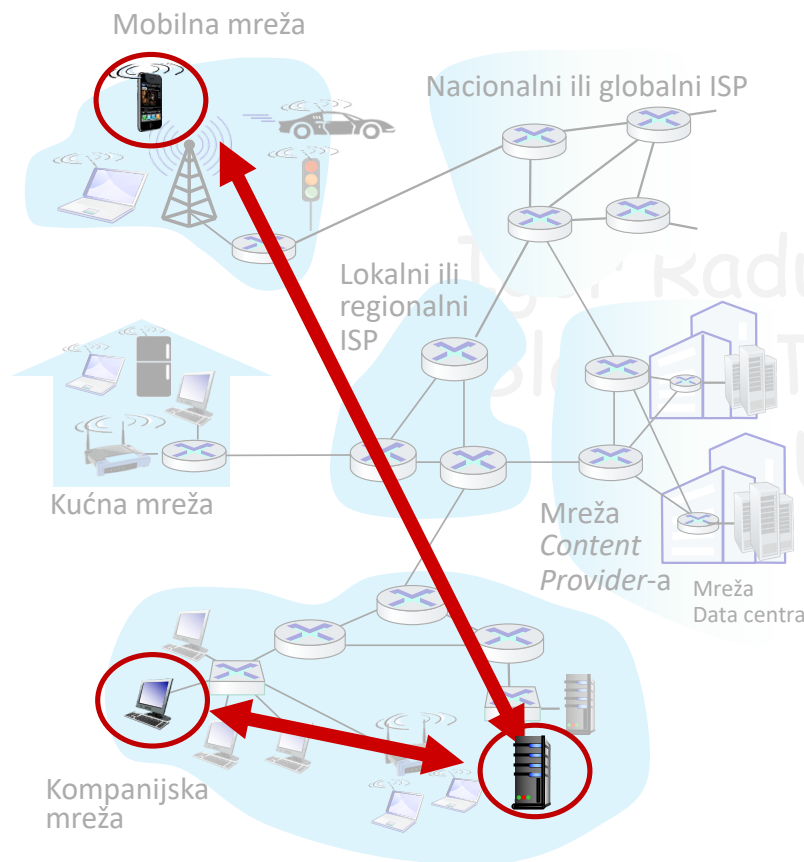


Arhitekture Internet aplikacija

- Klijent-server
- Peer-to-peer (P2P)
- Hibrid klijent-server i P2P
- ...

Igor Radusinović
Slavica Tomović
ETF, UCG

Klijent-server arhitektura



Server:

- Uvijek aktivan
- Po pravilu permanentna IP adresa
- Data centri

Klijenti:

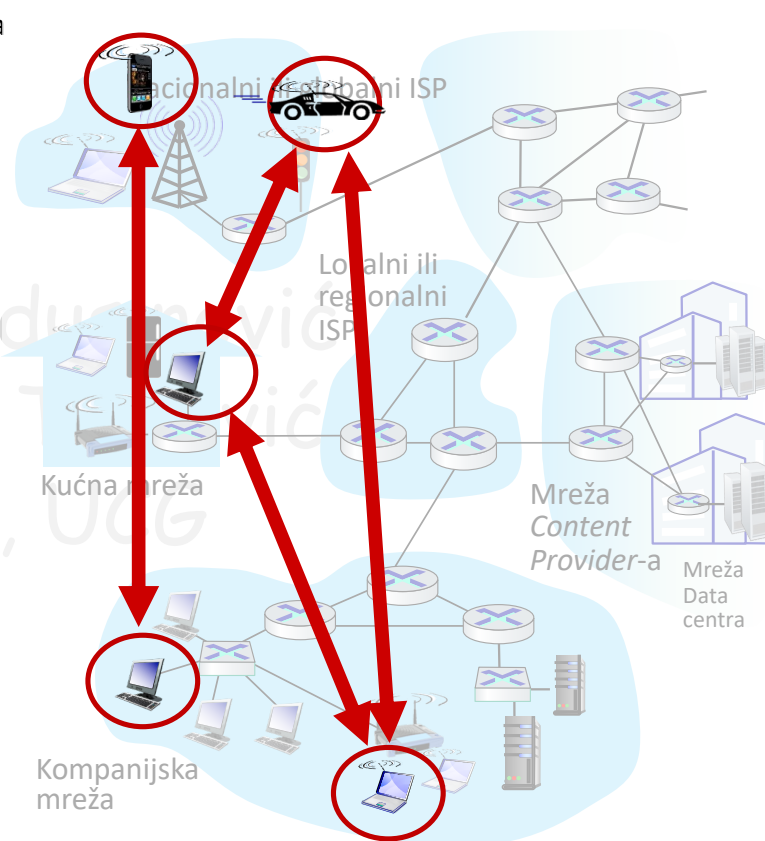
- Komuniciraju sa serverom
- Mogu biti povremeno povezani
- Mogu imati dinamičku IP adresu
- Ne komuniciraju međusobno

Primjeri:

- Web, e-mail, file transfer

P2P arhitektura

- Proizvoljni krajnji sistemi mogu direktno komunicirati bez učešća servera
- Peer zahtijeva servis od drugog peer-a, nudeći servis drugim peer-ovima
 - *skalabilnost- novi peer-ovi donose nove kapacitete, ali i nove zahtjeve*
- Peer-ovi se povremeno povezuju i mogu da mijenjaju IP adrese
 - Složeno upravljanje
- Primjer: P2P file sharing



Hibrid Klijent-server i P2P arhitektura

Skype/Viber

- voice-over-IP P2P aplikacije
- centralizovani server: pronalaženje strane sa kojom se želi komunicirati
- klijent-klijent komunikacija je direktna bez posredovanja servera

Instant messaging

- Čatovanje dva korisnika je P2P
- Detektovanje prisutnosti i lokacije je centralizovano:
 - Korisnik registruje svoju IP adresu na centralni server kada hoće da čatuje
 - Korisnik kontaktira centralni server da pronade IP adrese korisnika sa kojima želi da čatuje

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 razmjenom **poruka**

Klijent proces: proces koji inicijalizuje komunikaciju

Server proces: proces koji čeka da bude kontaktiran

- Napomena: aplikacije sa P2P arhitekturom imaju i klijent i server procese

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

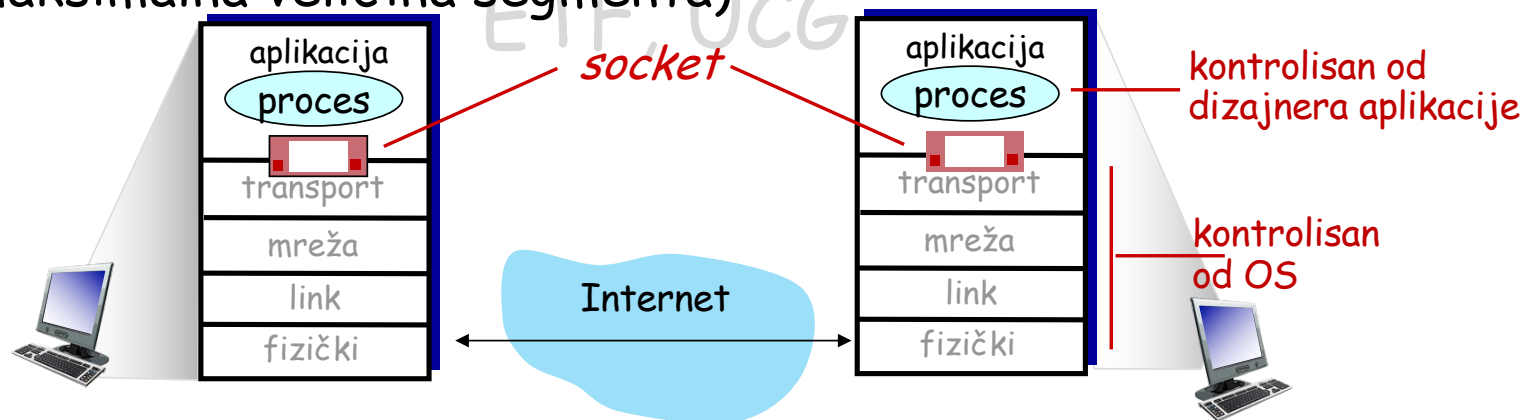
Klijentski proces: proces koji inicijalizuje komunikaciju

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
- ❑ Tipove sintaksi poruka: koja su polja & kako su odvojena
- ❑ Semantiku polja, 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, ...

Koji transportni servisi su potrebni aplikacijama?

Gubici 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

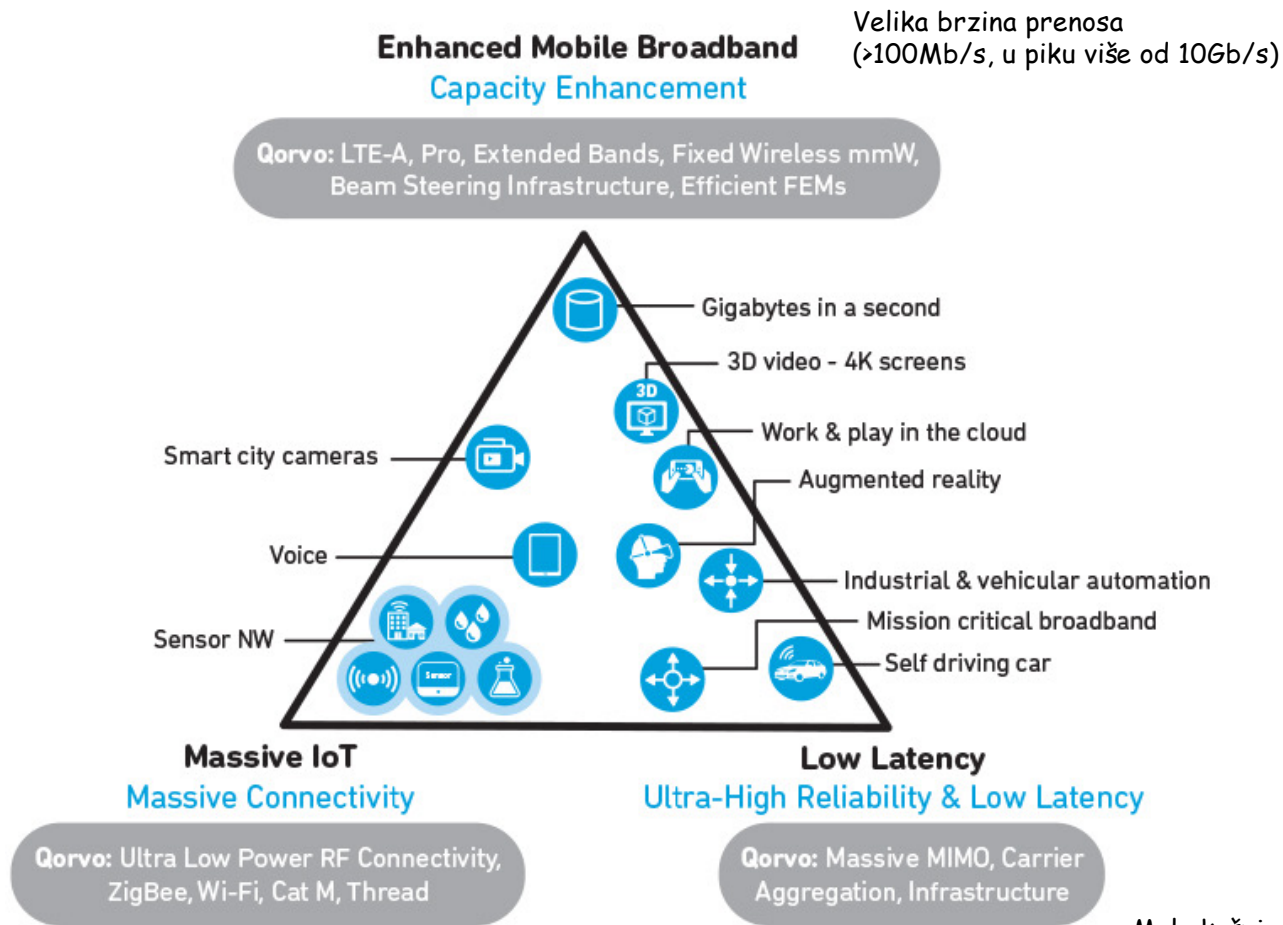
Zaštita

- ❑ 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"



10 do 100 puta više uređaja nego što podržavaju 4G i današnji WiFi

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

Malo kašnjenje (<1ms)
Mala vjerovatnoća gubitka (reda 10⁻⁹)

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 minimalnog opsega, zaštitu

UDP servisi:

- ❑ Nepouzdana prenos podataka između procesa slanja i prijema
- ❑ Ne obezbjeđuje: uspostavljanje veze, pouzdanost, 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
Streaming multimedija	HTTP (e.g., YouTube), DASH	TCP
Internet telefonija	SIP, RTP, proprietary (e.g., Skype)	TCP ili UDP
Interaktivne igre	WOW, FPS	UDP ili TCP

Zaštita i TCP

TCP & UDP

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

Transport Layer Security (TLS)

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

TLS je 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

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

Perzistentni HTTP

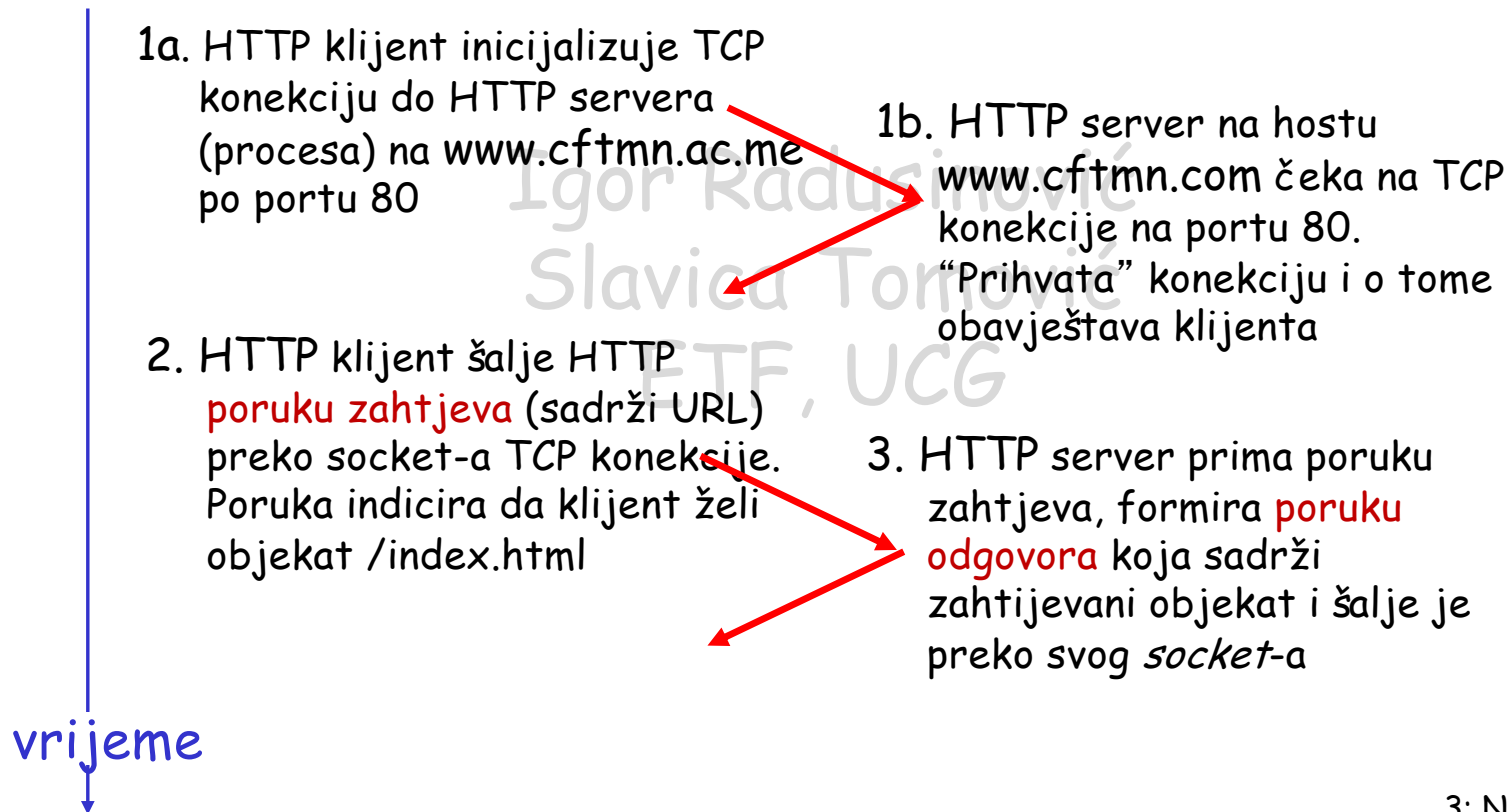
- Više objekata može biti poslato preko jedne TCP konekcije između klijenta i servera.

Igor Radusinović
Slavica Tomović
ETF, UCG

Neperzistentni HTTP

Neka korisnik unese sledeći URL

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



Neperzistentni HTTP(nastavak)

- Vrijeme ↓
4. HTTP server zatvara TCP konekciju.
 5. HTTP klijent prima poruku odgovora koja sadrži html fajl, tumači html fajl, prikazuje html fajl i pronalazi upućene objekte
 6. Koraci 1-5 se ponavljaju za svaki objekat
-

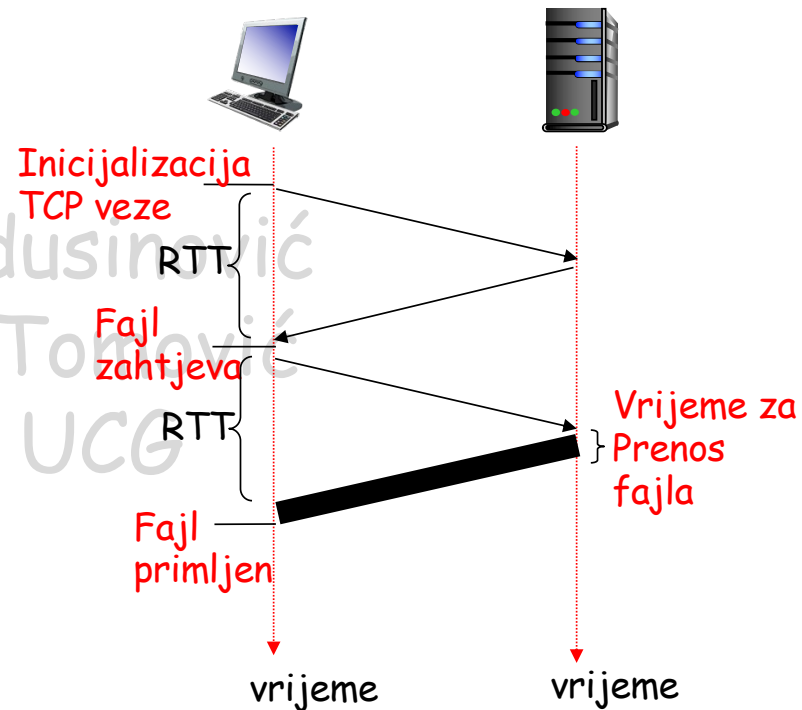
Modelovanje vremena odgovora

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 klijent/servera se šalju istom vezom
- ❑ Klijent šalje poruke zahtjeva odmah po dobijanju referenci objekata
- ❑ Potrebno 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

Igor Radusinović

```
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 u 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:

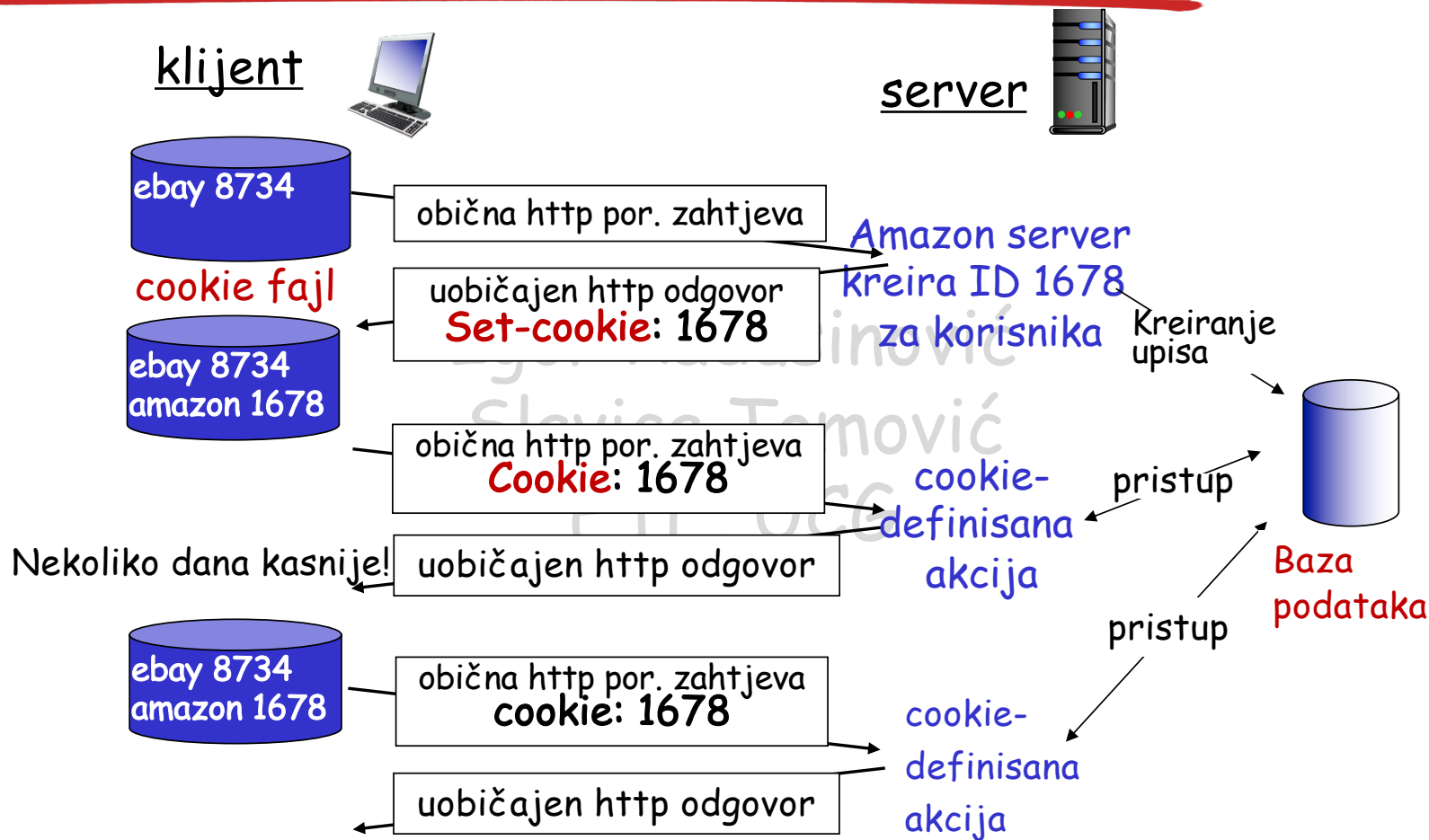
- 1) *Linija zaglavlja Set-cookie* u HTTP poruci odgovora
- 2) *Linija zaglavlja Cookie* u HTTP poruci zahtjeva
- 3) *Cookie fajl* se čuva na korisnikovom hostu i održava se od strane korisnikovog browser-a
- 4) Baza *podataka* na Web sajtu

Primjer:

- Neko 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 ID i kreira odgovarajuću informaciju u bazi podataka za ID

<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)

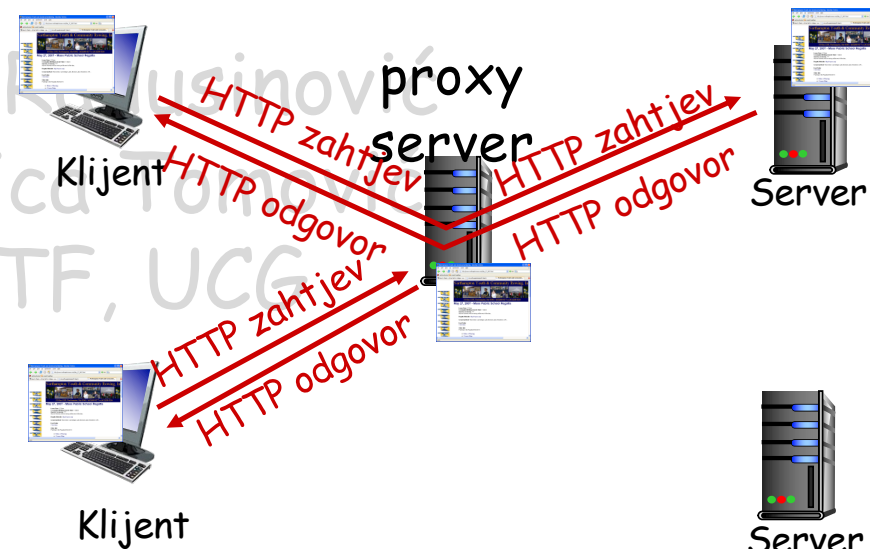
Cookies i privatnost: Pored toga

- 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

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 i kao klijent i kao server
- Tipično proxy instalira ISP (univerzitet, kompanija, rezidencijalni ISP)

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

Igor Radušinović
Slavica Tomović
ETF, UCG

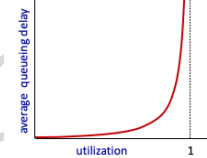
Primjer:

Pretpostavke:

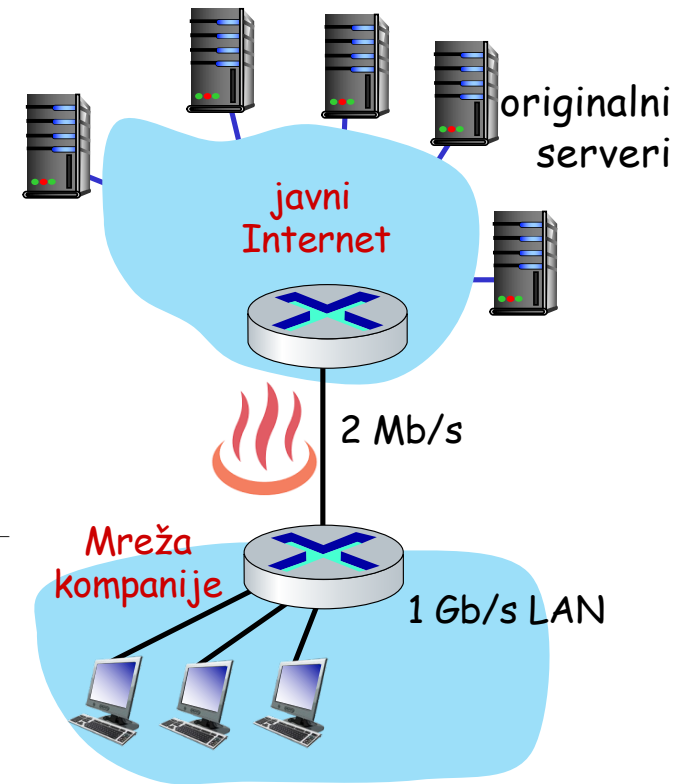
- Srednja veličina objekta: 100000 bita
- Srednji broj zahtjeva prema željenim serverima: 19 zahtjeva/s
- Srednja brzina : 1.9Mb/s
- RTT od rutera institucije do željenog servera: 2s
- Brzina na pristupnom linku: 2Mb/s

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!



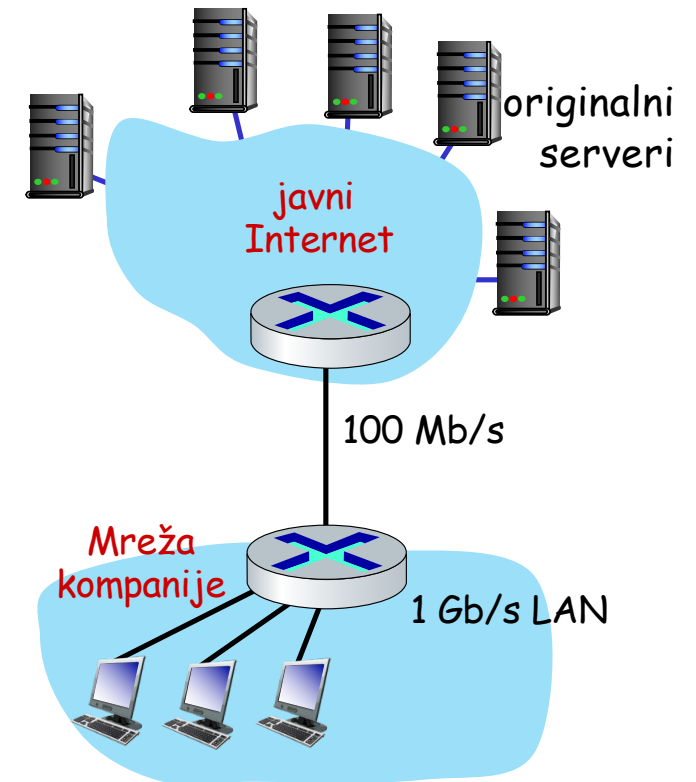
Primjer: brži pristupni link

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:

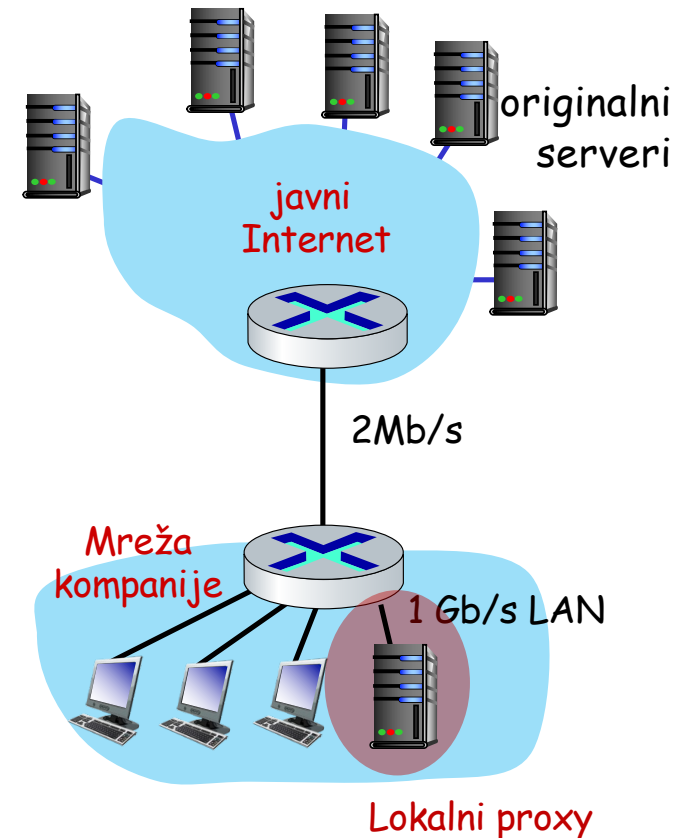
- Srednja veličina objekta: 100000 bita
- Srednja brzina zahtjeva: 19 zahtjeva/s
- Srednja brzina: 1.9Mb/s
- RTT od rutera institucije do željenog servera: 2s
- Brzina pristupa: 2Mb/s

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

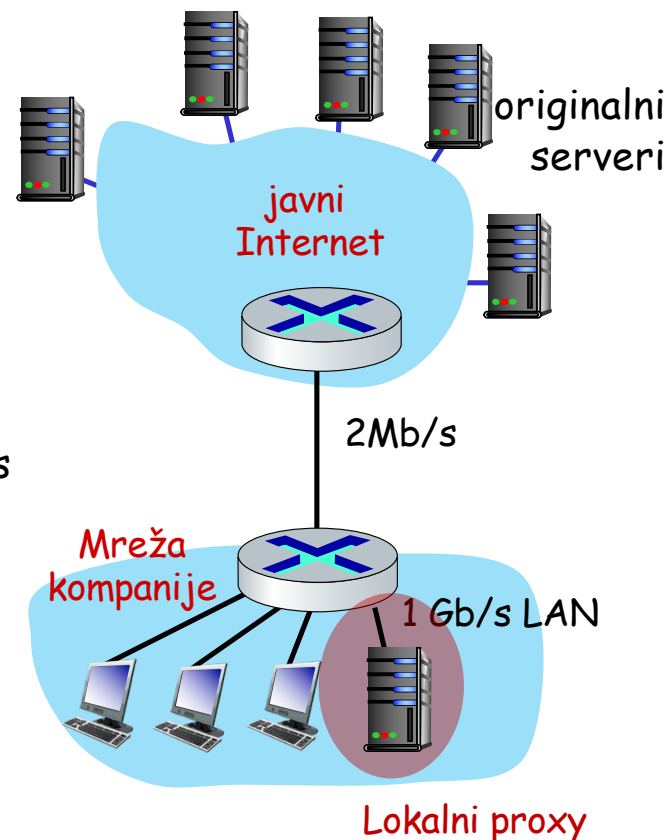
Iskorišćenje pristupnog linka:

- 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

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

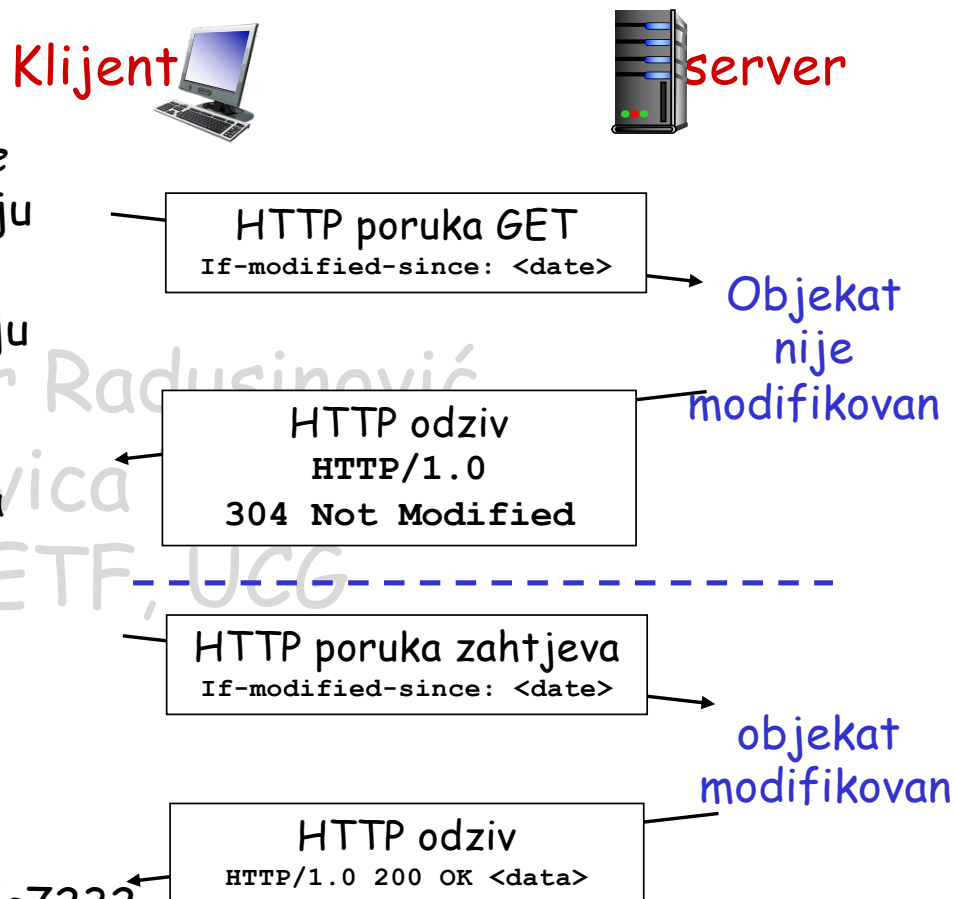
Manje nego pristupni link od 100Mb/s



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 više objektnih 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

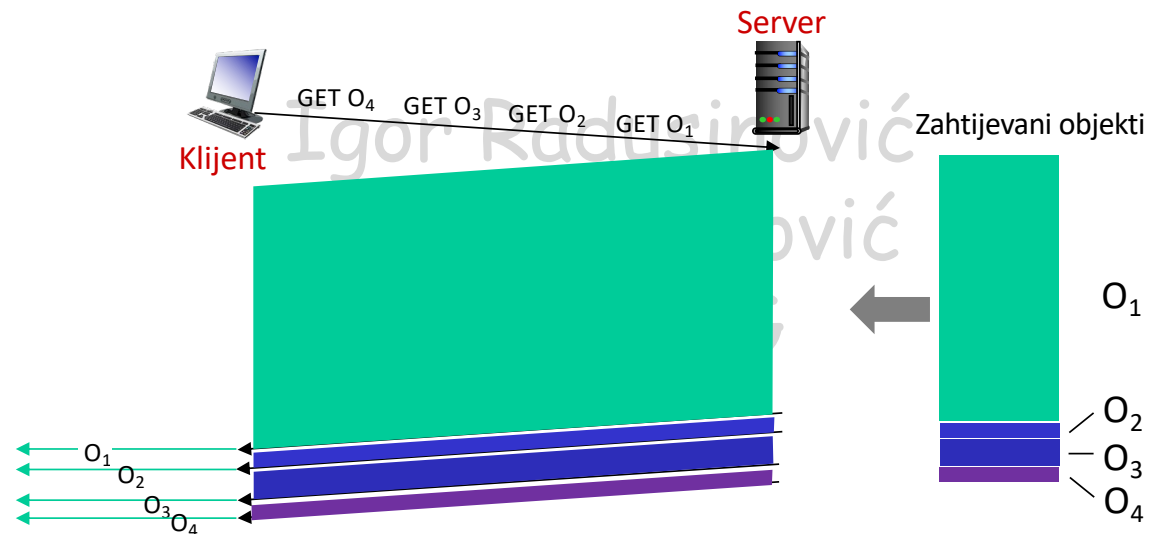
Cilj: smanjiti kašnjenje u slučaju više objektnih zahtjeva

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

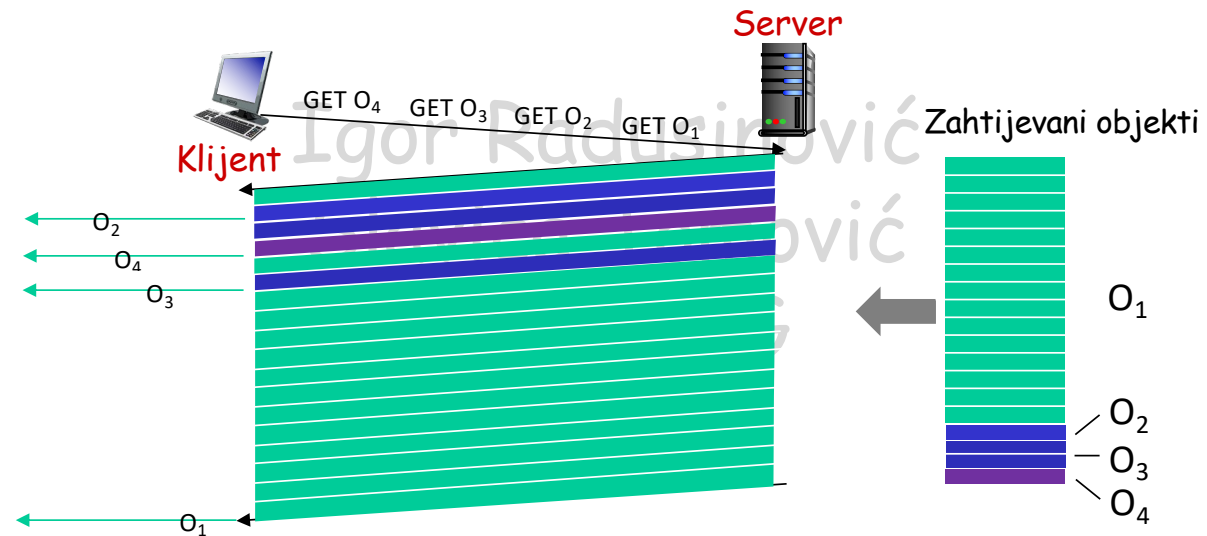
HTTP 1.1: klijent zahtjeva veliki objekat (na primjer video fajl) i 3 manja objekta



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:

- ❑ Više pipeline prenosa preko UDP protokola (QUIC)
- ❑ Ima zaštitu ugrađenu u QUIC protokolu

Nivo aplikacije

Ispitna pitanja

1. Arhitekture aplikacija
2. Navedi i objasni transportne servise koje zahtijevaju mrežne aplikacije
3. HTTP
4. Neperzistentni i perzistentni HTTP
5. HTTP poruke zahtjeva
6. HTTP poruke odgovora
7. Cookies
8. Web proxy servis
9. Objasniti razlike između različitih verzija HTTP protokola