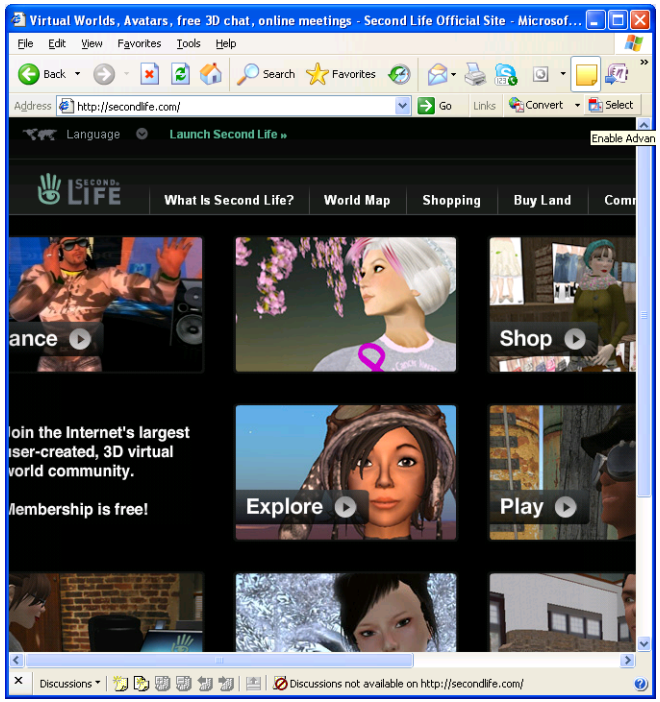
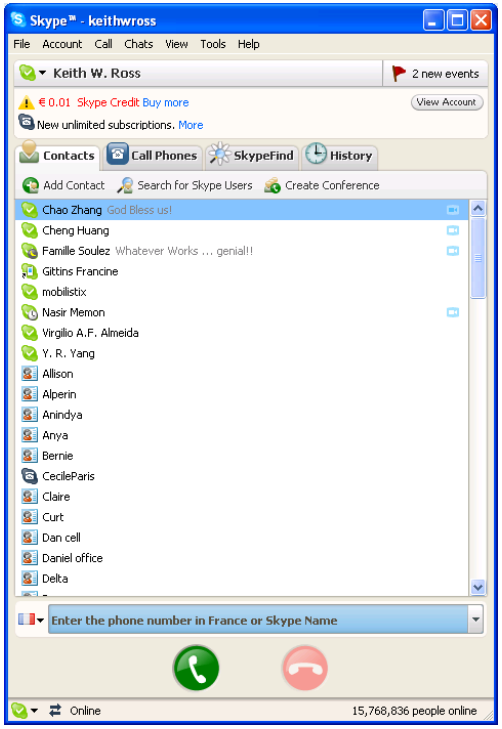
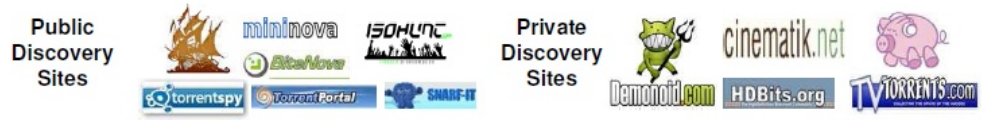
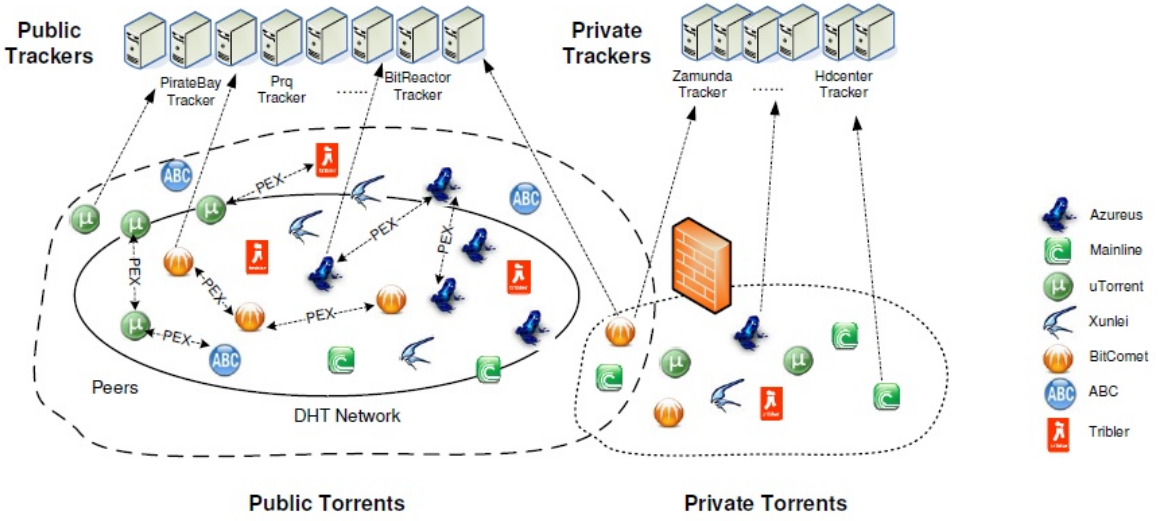
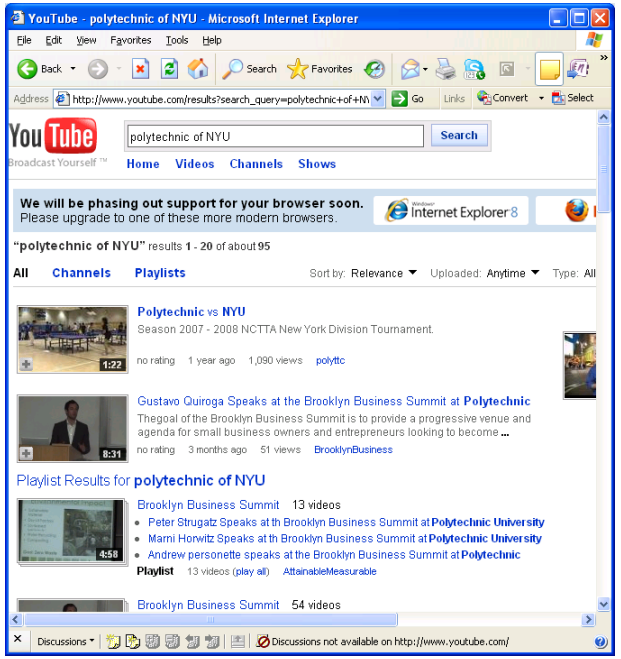










Nivo aplikacije

- Principi protokola nivoa aplikacije
- Web
  - HTTP

Primjeri Internet aplikacija

- E-mail
- Web
- "Instant messaging"
- "Remote login"
- "P2P file sharing"
- "Multi-user" mrežne igre
- "Streaming stored" video klipovi
- Internet telefon
- "Real-time" video konferencija
- "Grid computing"
- Društvene mreže
- ...

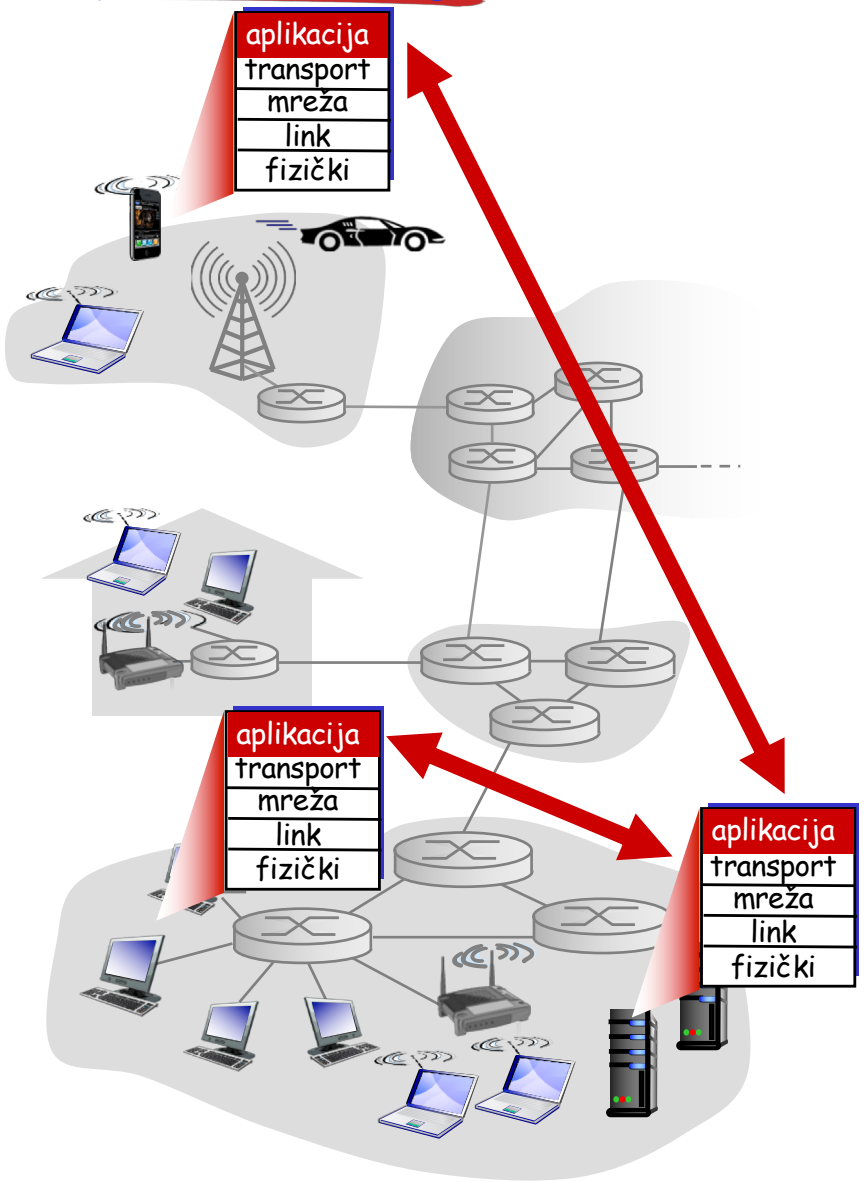
Kreiranje Internet aplikacije

Napisati programe koji

- se izvršavaju na različitim krajnjim sistemima i
- komuniciraju preko mreže.
- npr., Web: Web server software komunicira preko browser software

Ne piše se softver za uređaje na kičmi mreže

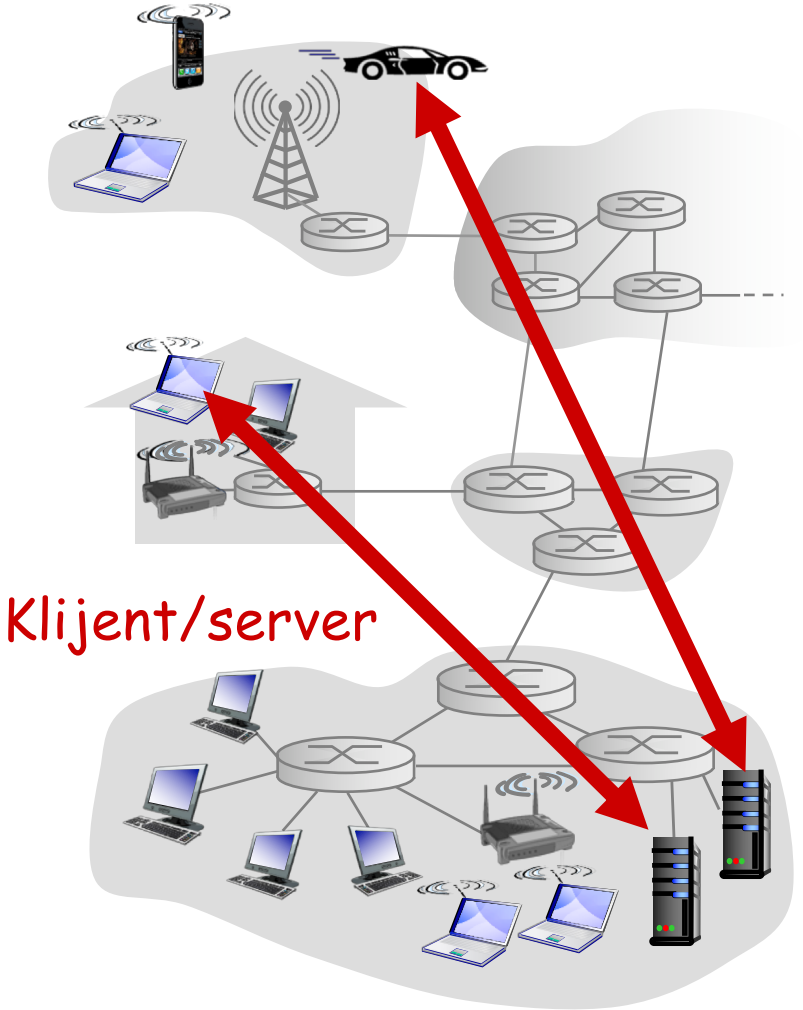
- mrežni uređaji na kičmi ne funkcionišu na nivou aplikacije
- ovakav dizajn dozvoljava brzi razvoj aplikacija



Arhitekture Internet aplikacija

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

Klijent-server arhitektura



server:

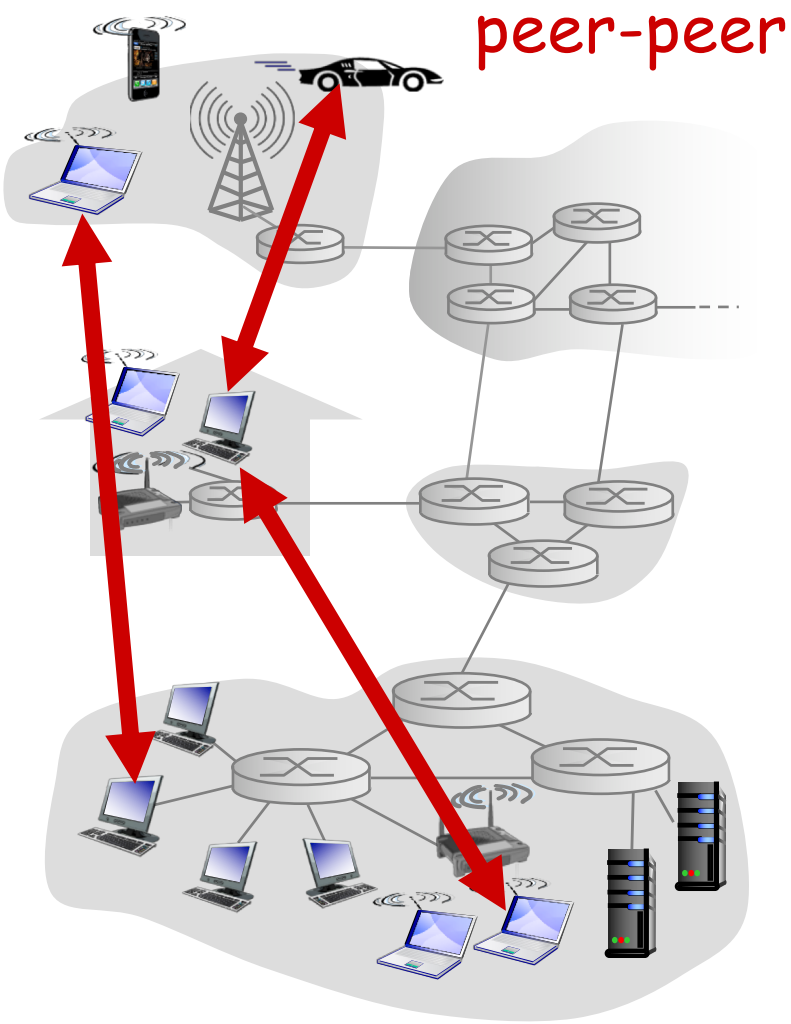
- Uvijeka aktivan
- Po pravilu permanentna IP adresa
- Data centri

klijenti:

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

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



Hibrid Klijent-server i P2P arhitektura

Skype

- voice-over-IP P2P aplikacija
- centralizovani server: pronalaženje adrese udaljene strane
- klijent-klijent konekcija 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 pronađe 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

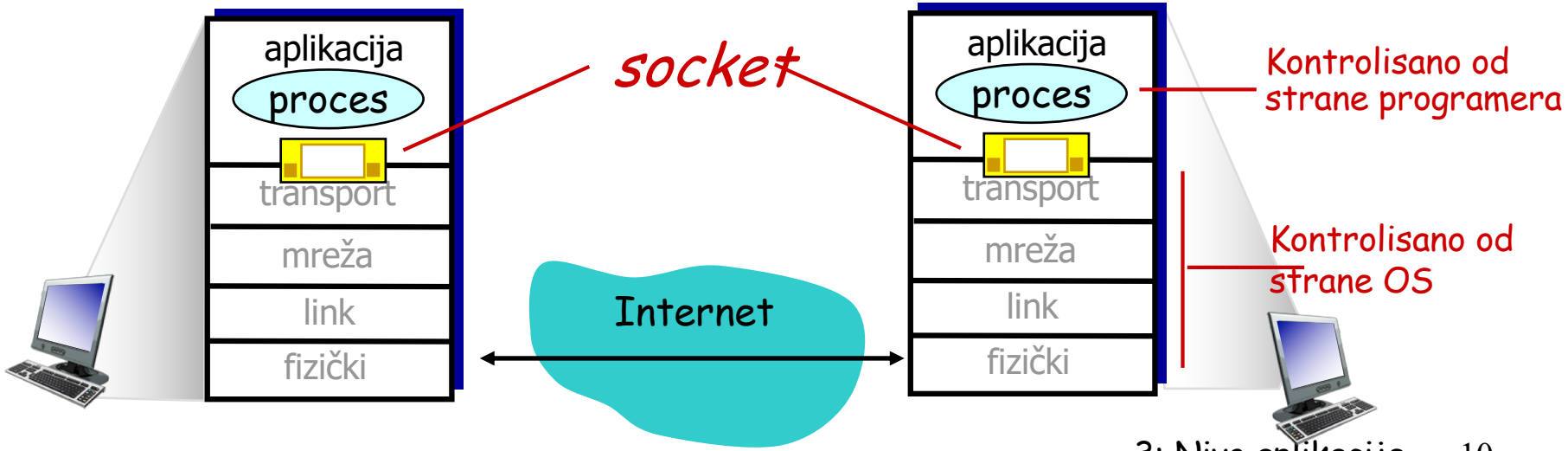
Klijent proces: proces koji inicijalizuje komunikaciju

Server proces: proces koji čeka da bude kontaktiran

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

Soketi

- Proces šalje/prima poruke preko svog "socket"-a
- "socket" je analogan vratima
  - Proces šalje poruke preko socketa
  - proces koji šalje se oslanja na transportnu infrastrukturu na drugoj stani vrata koja prenosi poruku do "socket" 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
- Prisjetiti se komande 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

O: 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
- Semantika polja, npr., značenje informacija u poljima
- Pravila vezana kada i kako se šalje poruka i kako se odgovara na njih

Javni (public) protokoli:

- Definisani u RFC-ovima
- Dozvoljavaju interoperatibilnost
- npr, HTTP, SMTP

Privatni (proprietary) protokoli:

- npr, Skype, Viber,...

Koji transportni servisi su potrebni aplikacijama?

Gubici podataka

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

Vrijeme

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

Brzina prenosa

- Neke aplikacije (npr., multimedija) zahtijevaju preciziranje minimalne dostupne brzine prenosa
- Druge aplikacije ("elastične aplikacije") koriste onoliko opsega koliko mogu dobiti

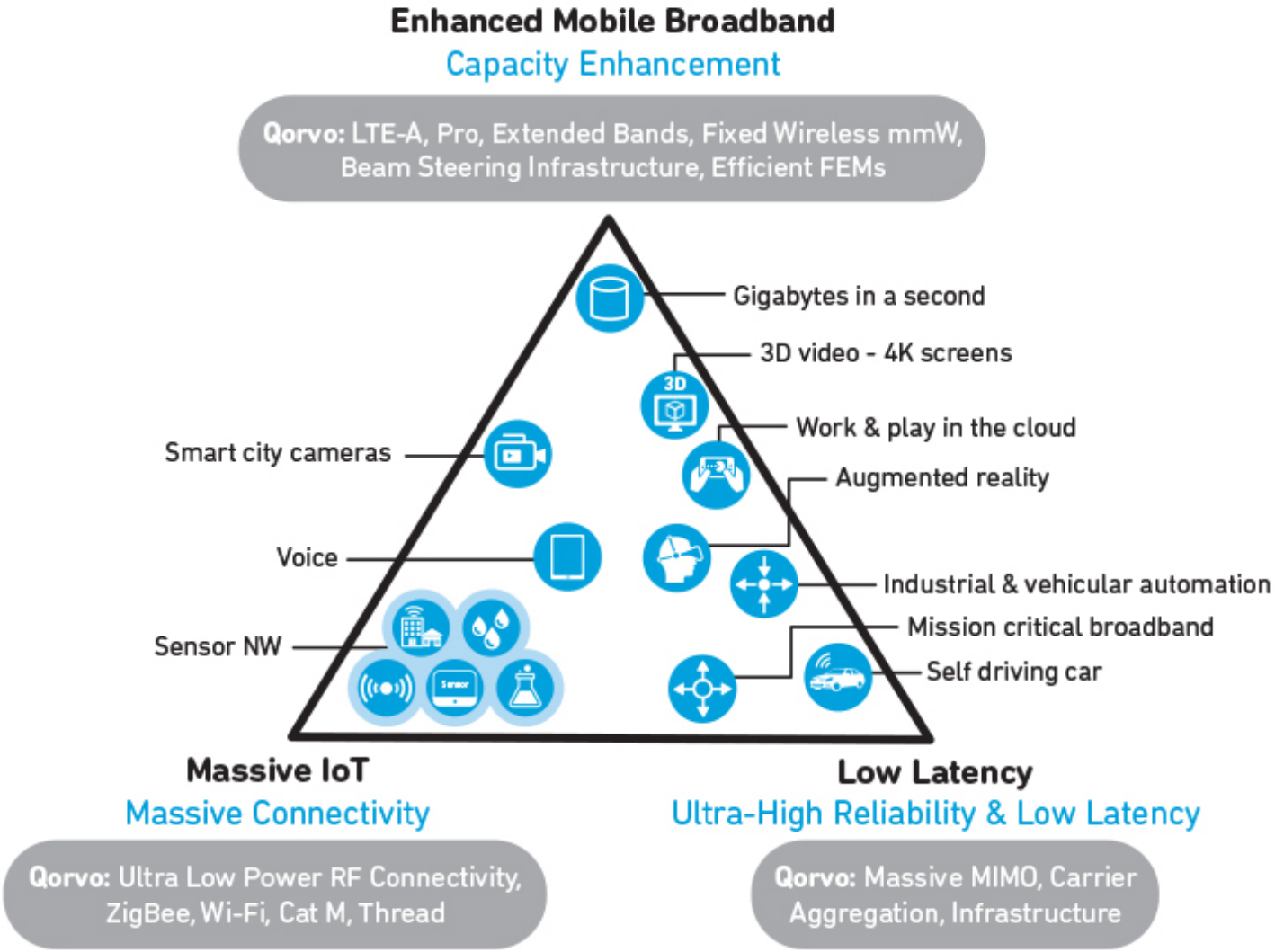
Zaštita

- Enkripcija, integritet podataka, ...

Transportni servisni zahjevi zajednički za sve aplikacije

Aplikacija	Gubici	Brzina prenosa	Vrem. osjet.
File transfer	bez	elastičan	ne
E-mail	bez	elastičan	ne
Web	bez	elastičan	ne
Real-time audio/video	tolerantne	audio: 5kb/s-1Mb/s video:10kb/s-	da, 100' ms
Stored audio/video	tolerantne	5Mb/s	da, nekoliko s
Interaktivne igre	tolerantne	Isti kao gore	da, 100' ms
Instant messaging	bez	nekoliko kb/s i više 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 obezjeđuje:* tajming, garantovanje minimalnog opsega

UDP servisi:

- nepouzdani prenos podataka između procesa slanja i prijema
- ne obezbjeđuje: uspostavljanje veze, pouzdanost, kontrolu protoka, kontrolu zagušenja, tajming, ili garantovani opseg

P: Zašto oba? Zašto UDP?

Internet aplikacije: aplikacija, transportni protokoli

Aplikacija	Protokoli nivoa aplikacije	Transportni protokol
e-mail	SMTP [RFC 2821]	TCP
udaljeni terminal	Telnet [RFC 854]	TCP
Web	HTTP [RFC 7230]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	Privatni ili javni	TCP ili UDP ili RTP
Internet telefonija	Privatni (npr., Dialpad, Skype)	UDP (i TCP)

Zaštita i TCP

TCP & UDP

- Nema kriptovanja
- Tekstualne lozinke se prenose preko Interneta

SSL (Secure Sockets Layer)

- Omogućava enkripciju TCP konekcije
- Integritet podataka
- Autorizacija od kraja do kraja

SSL je na nivou aplikacije

- Aplikacije koriste SSL biblioteke, koje "komuniciraju" sa TCP

SSL socket API

- Tekstualna lozinka se šalje kriptovana preko Interneta

Web i HTTP

Termini

- Web stranica se sastoji od objekata
- Objekat može biti HTML fajl, JPEG slika, Java "applet", audio fajl,...
- Web stranica se sastoji od osnovnog HTML-fajla koji može sadržati reference više objekata
- Svaki objekat se adresira sa URL (Uniform Resource Locators)
- Primjer URL:

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

ime hosta ime puta

Pregled HTTP-a

HTTP: HyperText Transfer Protocol

- Web-ov protokol nivoa aplikacije
- klijent/server model
  - *klijent:* Web browser šalje zahtjeve, prima i prikazuje Web objekte
  - *server:* Web server šalje objekte kao odgovor na zahtjeve

Pregled HTTP-a (nastavak)

Koristi TCP:

- klijent inicijalizuje TCP vezu (kreira socket) prema serveru, port 80
- server prihvata TCP vezu od klijenta
- HTTP poruke zahtjeva i odgovora (poruke protokola nivoa aplikacije) se razmjenjuju između "browser"-a (HTTP klijent) i Web servera (HTTP server)
- TCP veza se zatvara

HTTP je "stateless"

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

Pored toga

Protokoli koji nadziru "stanje" su kompleksni!

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

HTTP konekcije

Neperzistentni (neistrajan) HTTP

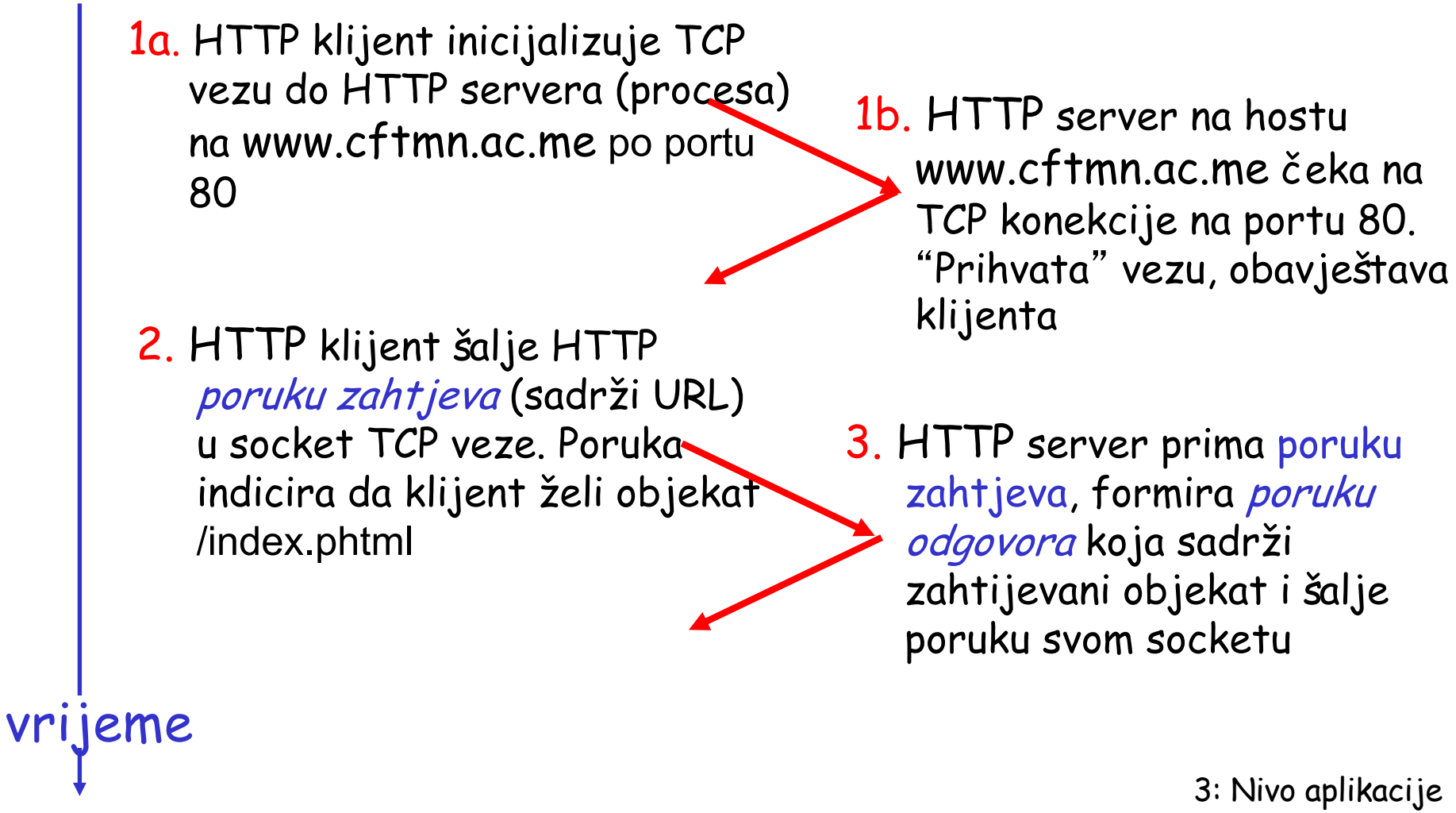
- Najviše jedan objekat se šalje preko TCP konekcije.
- Povlačenje više objekata podrazumijeva otvaranje više konekcija

Perzistentni HTTP

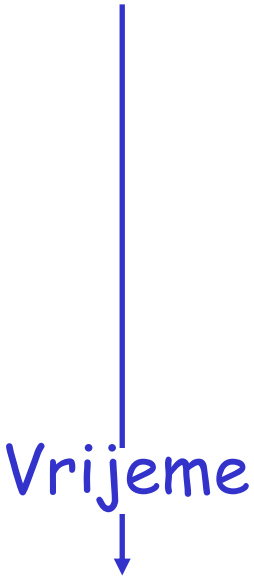
- Više objekata može biti poslato preko jedne TCP konekcije između klijenta i servera.

Neperzistentni HTTP

Pretpostavimo da korisnik unese sledeći URL
http://www.cftmn.ac.me/index.html



Neperzistentni HTTP(nastavak)



4. HTTP server zatvara TCP vezu.

5. HTTP klijent prima poruku odgovora koja sadrži html fajl, prikazuje html fajl, tumači html fajl, pronalazi upućene objekte

6. Koraci 1-5 se ponavljaju za svaki objekat

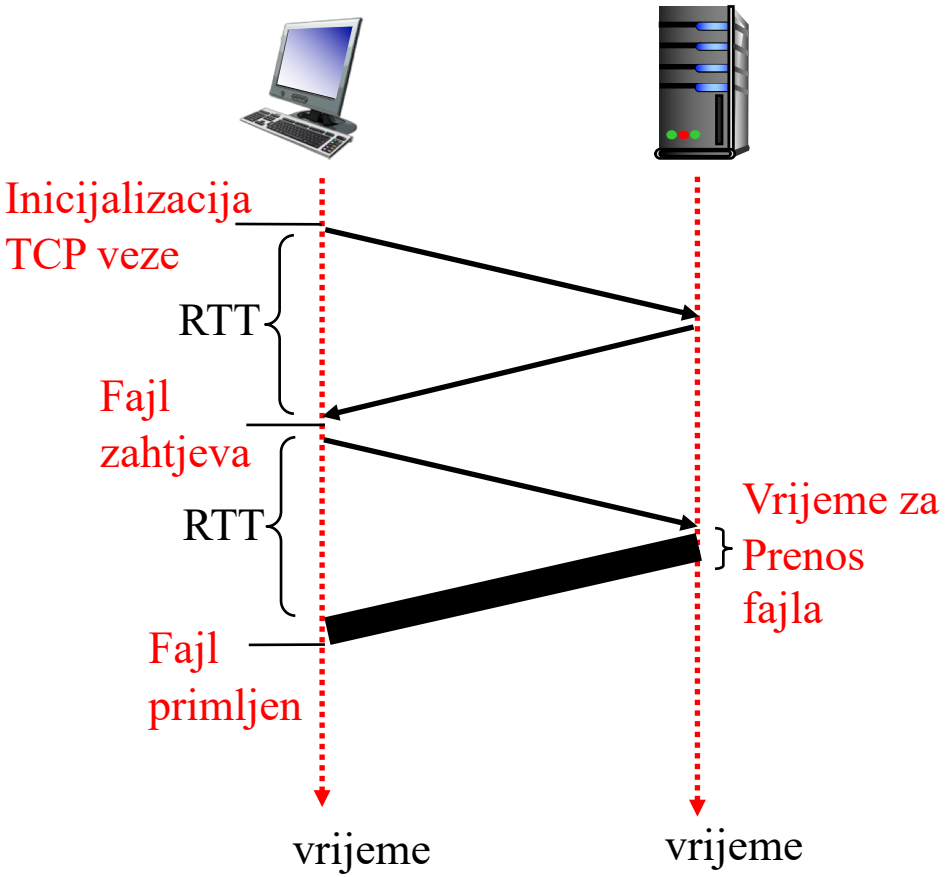
Modelovanje vremena odgovora

Definicija RTT (Round Trip Time): vrijeme prenosa malog paketa od klijenta do servera i nazad.

Vrijeme odgovora:

- jedan RTT za inicijalizaciju TCP veze
- jedan RTT za HTTP zahtjev i vraćanje prvih nekoliko bajtova HTTP odgovora
- Vrijeme prenosa fajla

ukupno = 2RTT+vrijeme prenosa fajla



Perzistentni HTTP

Problemi neperzistentnog HTTP-a:

- zahtijeva 2 RTT po objektu
- OS mora raditi i dodijeliti resurse hosta za svaku TCP vezu
- Problem je što browser-i često otvaraju paralelne TCP veze za povlačenje zahtijevanih objekata

Perzistentni HTTP

- server zadržava vezu otvorenu poslije slanja odgovora
- sekvencijalne HTTP poruke između istog klijent/servera se šalju istom vezom
- Zatvara konekciju poslije određenog vremena neaktivnosti

Perzistentni bez "pipelining":

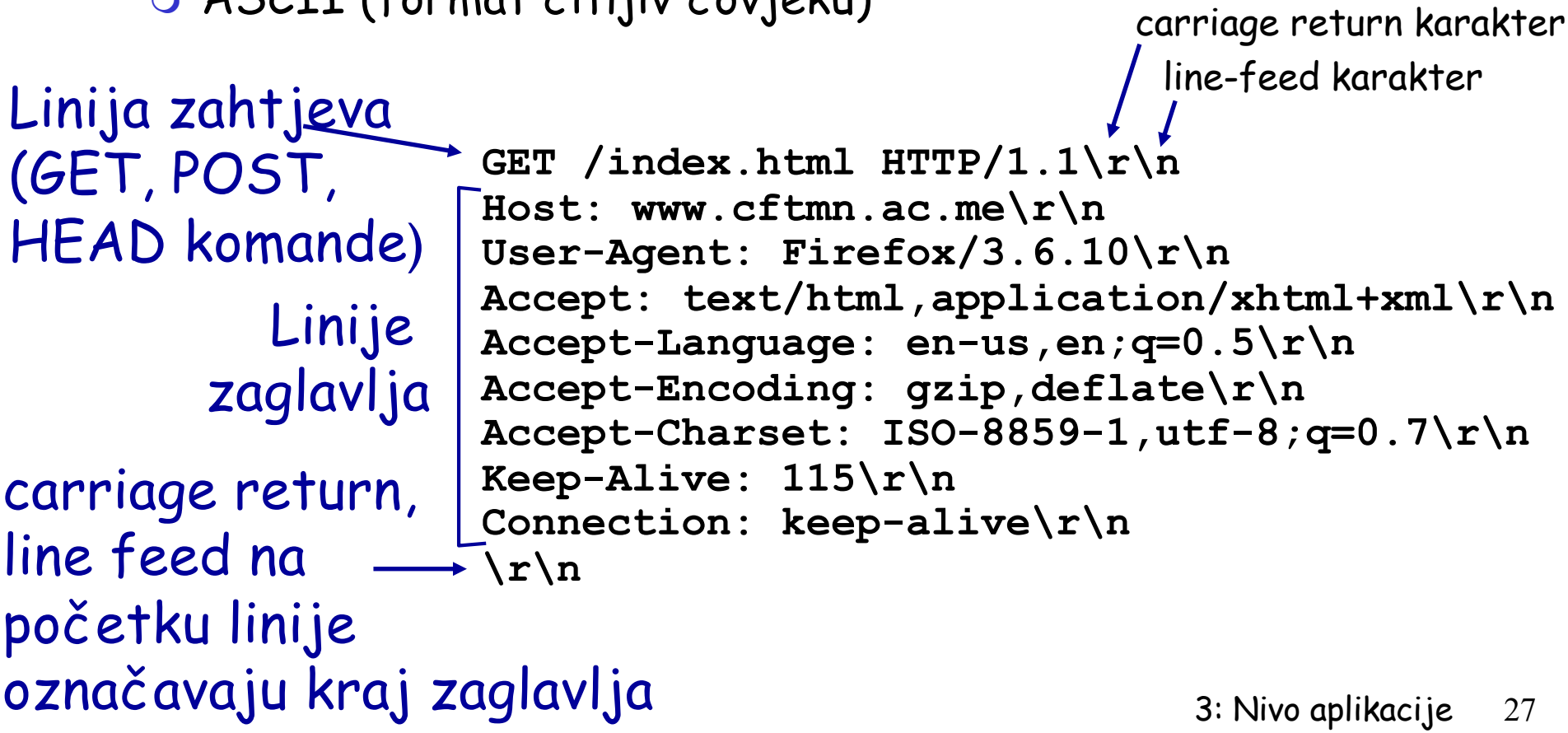
- Klijent šalje novi zahtjev samo kada je prethodni odgovor primljen
- jedan RTT za svaki upućeni objekat
- Kada nema zahtjeva TCP konekcija je slobodna

Perzistentni sa "pipelining":

- klijent šalje zahtjeve odmah po dobijanju referenci objekata
- Veličine svega po jedan RTT za svaki referencirani objekat

HTTP poruka zahtjeva

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



Tipovi

HTTP/1.0

- GET
- POST
- HEAD
  - Pita server da pusti traženi sadržaj (otklanjanje grešaka)

HTTP/1.1

- GET, POST, HEAD
- PUT
  - Uploaduje fajl na mjesto u Web serveru definisano u URL polju
- DELETE
  - Briše fajl definisan u URL polju

HTTP poruka odgovora

statusna linija
(protokol statusni kod statusna fraza)

Linije zaglavlja

```
HTTP/1.1 200 OK\r\n
Date: Thu, 26 Sep 2013 18:09:20 CET\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2012 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
   1\r\n
\r\n
data data data data data ...
```

podaci, npr., zahtijevani HTML fajl

HTTP kodovi statusnog odgovora

U prvoj liniji u server->klijent poruci 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

- Verzija HTTP protokola nije podržana

Cookies: vode računa o "stanju"(RFC 6265)

Mnogi Web sajtovi koriste cookies

Četiri komponente:

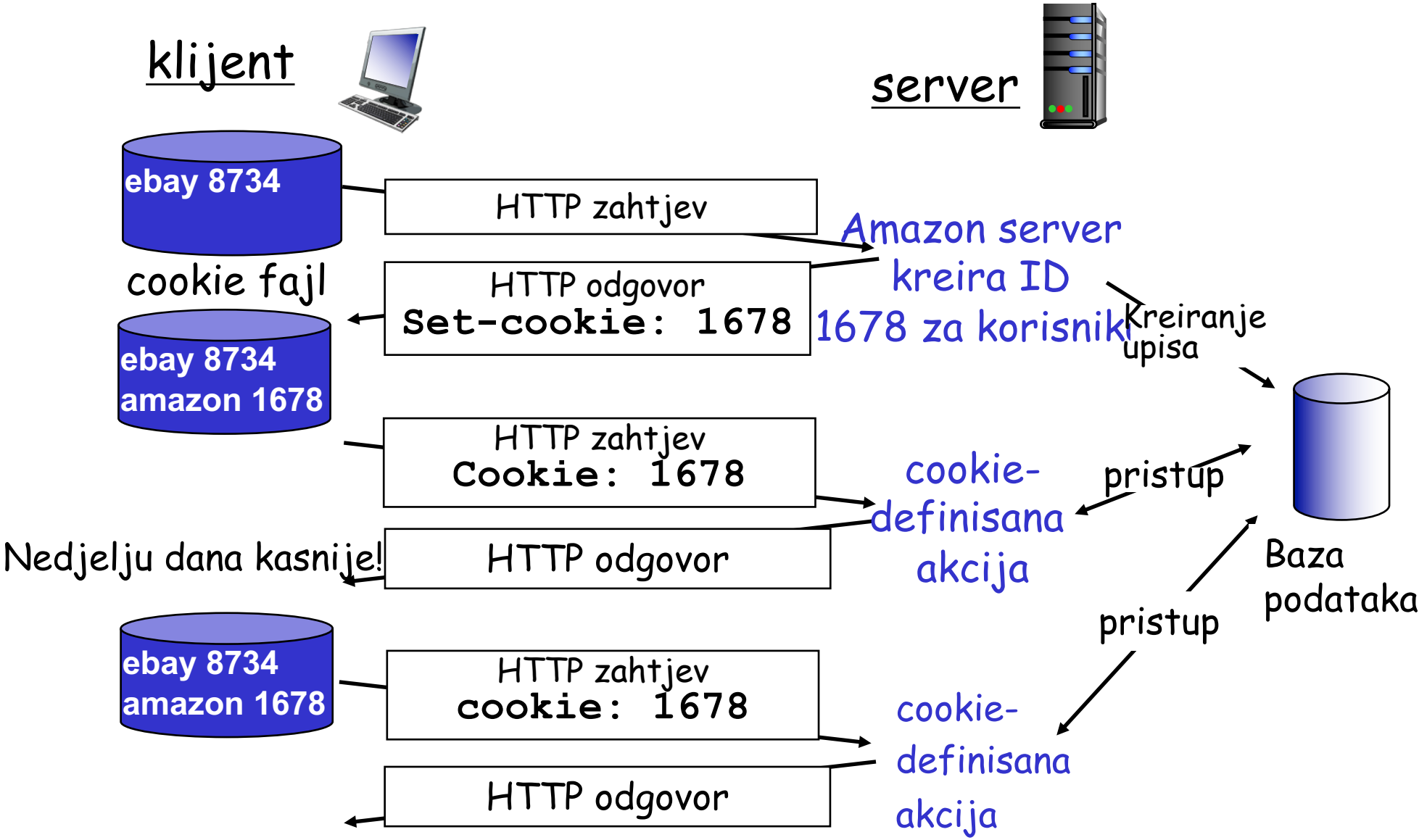
1) Linija zaglavlja Set-cookie u HTTP poruci odgovora
2) Linija zaglavlja Cookie u HTTP poruci zahtjeva
3) Cookie fajl se čuva na korisnikovom hostu i održava se od strane korisnikovog browser-a
4) Baza podataka na Web sajtu

Primjer:

- Neko pristupa Internetu uvijek preko istog hosta
- 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)

Pored toga

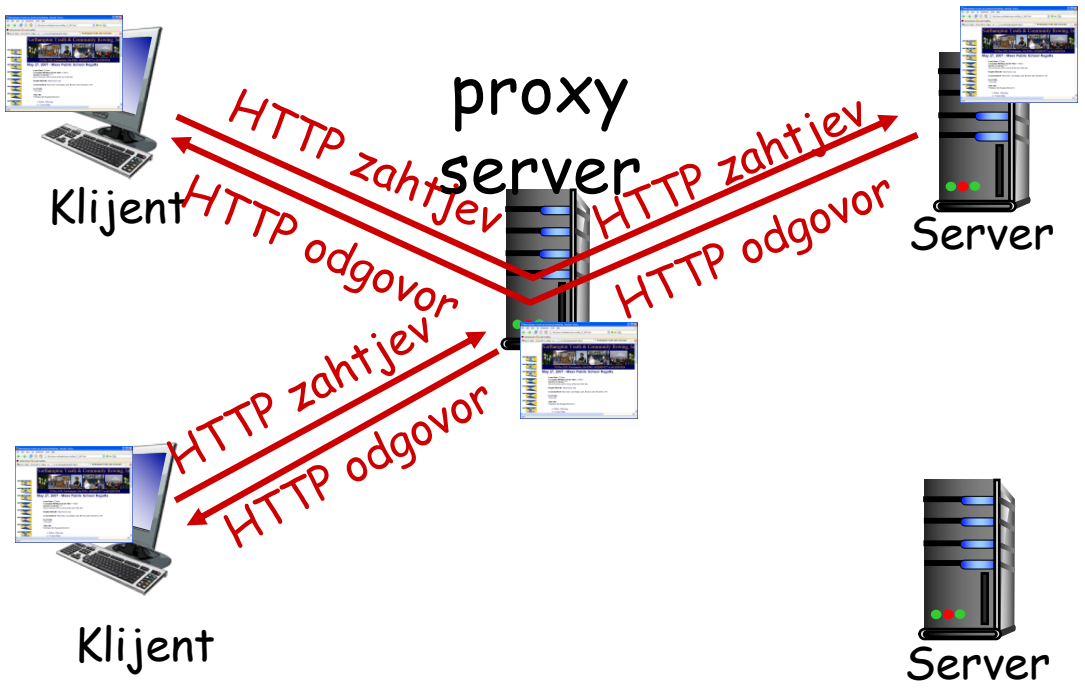
Cookies i privatnost:

- Cookies dozvoljavaju sajtu da dosta nauči o korisniku
- Mogu se prikupiti imena i kontakt podaci
- Pretraživači koriste cookies da nauče više o korisnicima
- Kompanije dobijaju dodatne informacije preko weba

Web "caches" (proxy server)

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
  - ili proxy zahtijeva objekat od željenog servera, tada vraća objekat 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

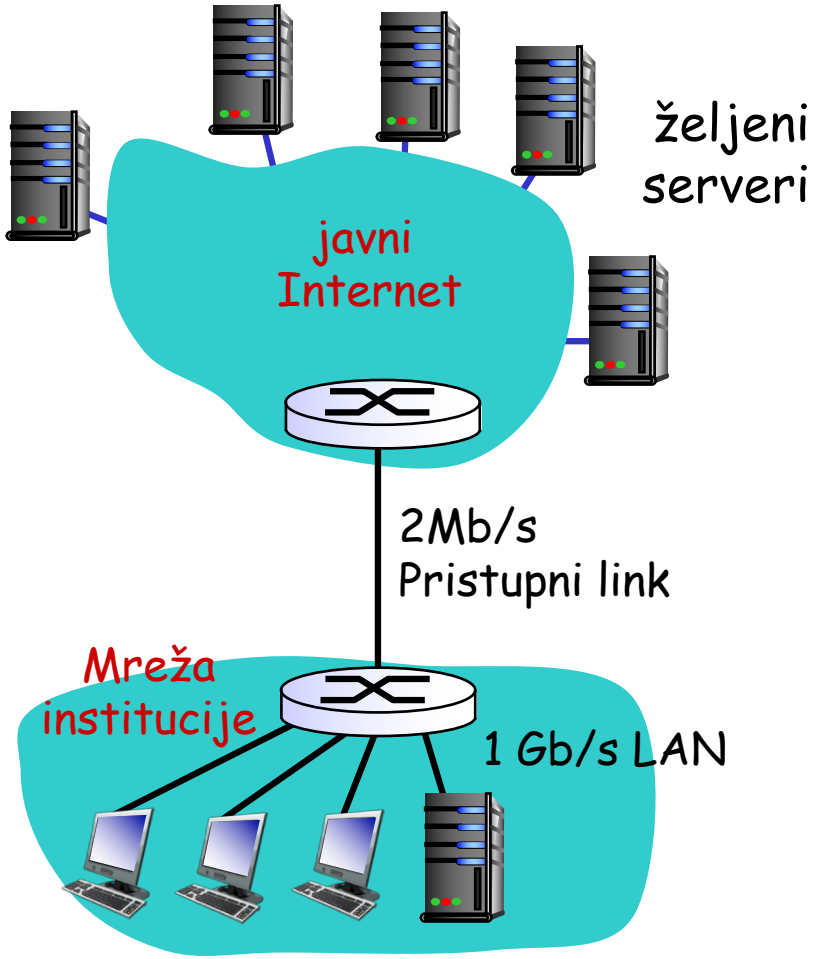
Primjer:

Pretpostavke:

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

Posledice:

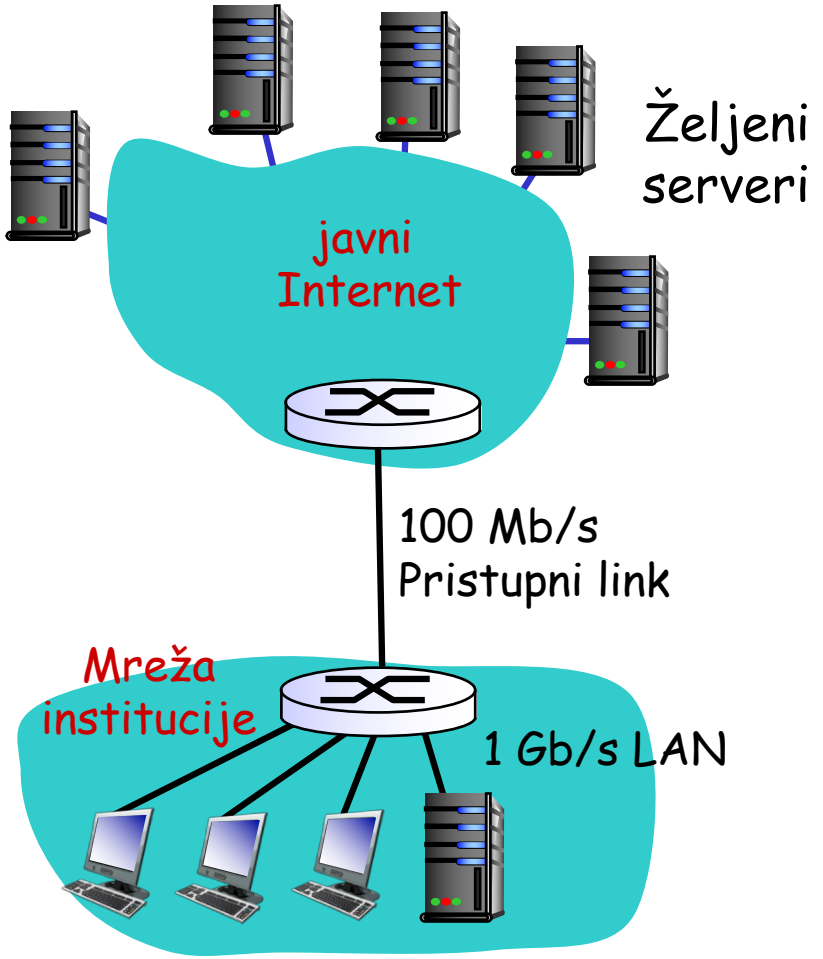
- Iskorišenje LAN-a: 0.19% *problem!*
- Iskorišenje pristupnog linka = 95%
- Ukupno kašnjenje= kašnjenje na Internetu+ kašnjenje u pristupu+ LAN kašnjenje
  = 2s + minuti + ms



Primjer: brži pristupni link

pretpostavke:

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



posledice:

- Iskorišenje 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:

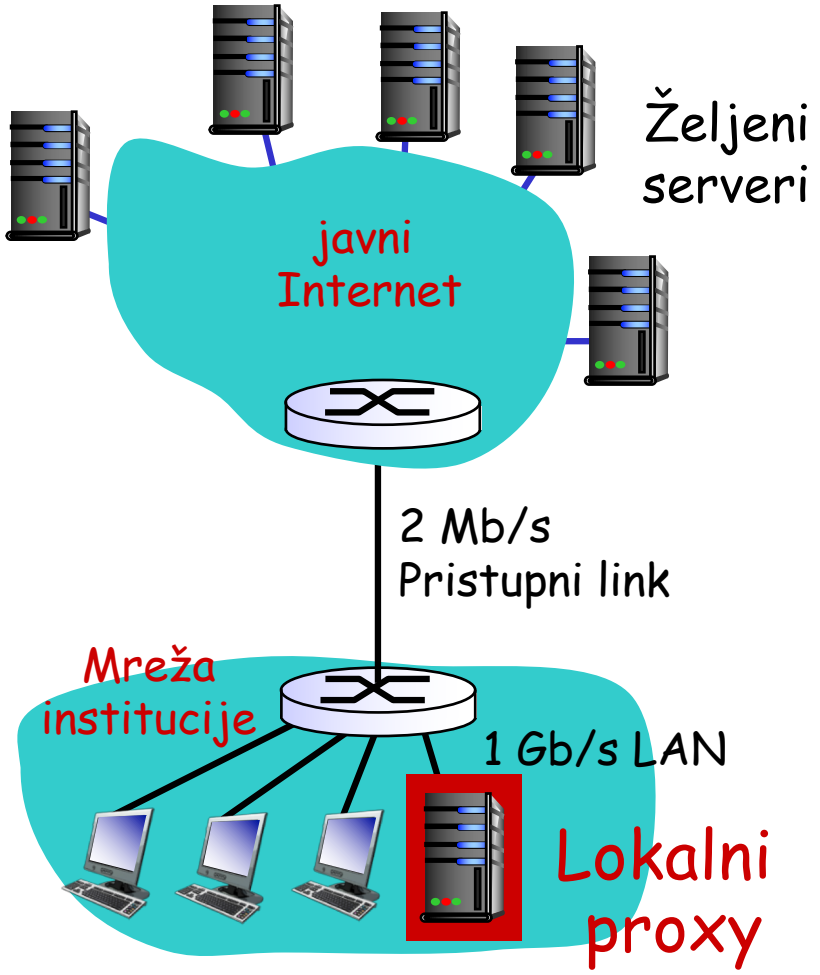
- Srednja veličina objekta: 100000b
- Srednja brzina zahtjeva: 19zahtjeva/s
- Srednja brzina: 1.9Mb/s
- RTT od rutera institucije do željenog servera: 2s
- Brzina pristupa: 2Mb/s

posledice:

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

Kako izračunati iskorišćenje i kašnjenje?

Troškovi: proxy nije skup!



Primjer: Lokalni proxy

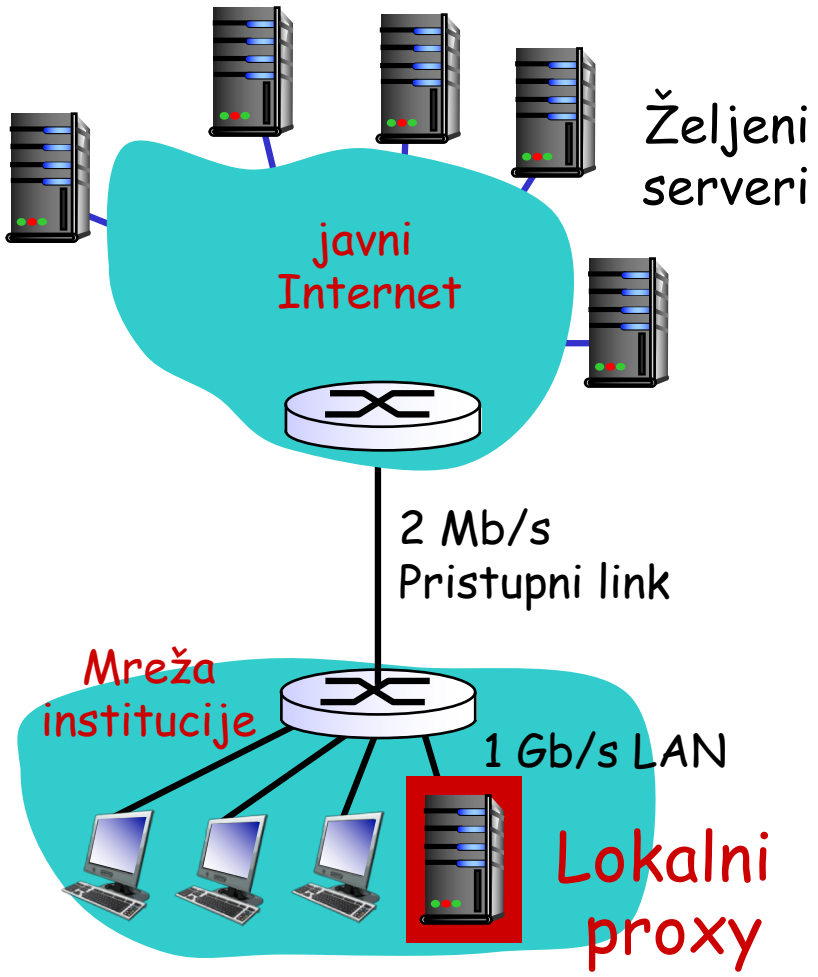
Izračunavanje iskorišćenja i kašnjenja:

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

Iskorišćenje pristupnog linka:

- 60% zahtjeva koristi pristupni link

- Brzina prenosa preko pristupnog linka= 0.6*1.9 Mb/s = 1.14Mb/s
  - iskorišćenje = 1.14/2 = .57
    - Ukupno kašnjenje
      - = 0.6 * (kašnjenje od željenih servera) +0.4 * (kašnjenje do proxy servera)
      - = 0.6 (2.0) + 0.4 (~ms)
      - = ~ 1.2s
      - Manje nego 100Mb/s pristupni link



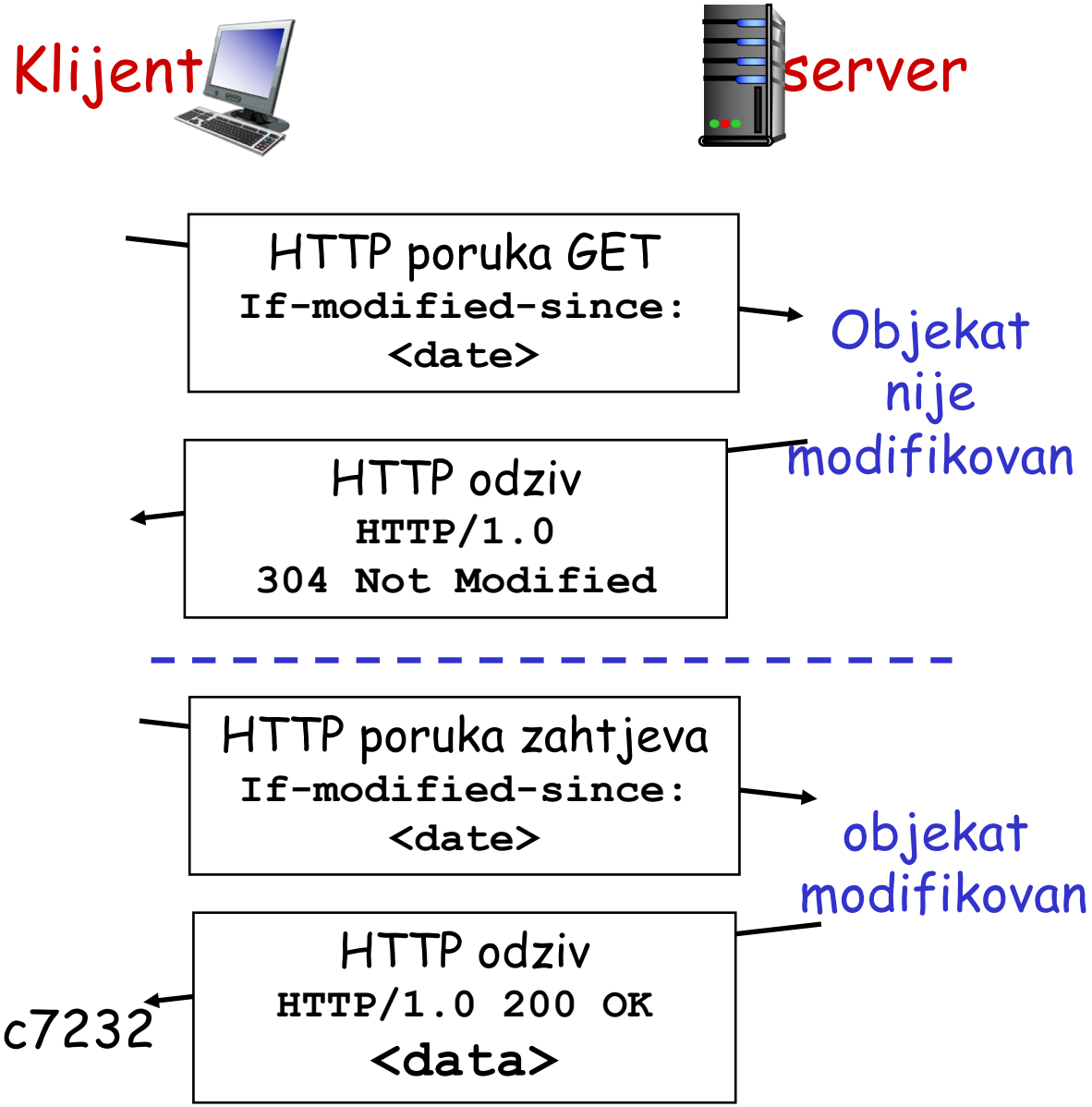
Conditional GET

- Cilj: ne slati objekat ako cache ima up-to-date sačuvanu verziju
- cache: 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



Fog computing

Fog Computing: Helping the Internet of Things Realize Its Potential
Issue No. 08 - Aug. (2016 vol. 49), ISSN: 0018-9162 pp: 112-116
DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/MC.2016.245
Amir Vahid Dastjerdi , University of Melbourne
Rajkumar Buyya , University of Melbourne