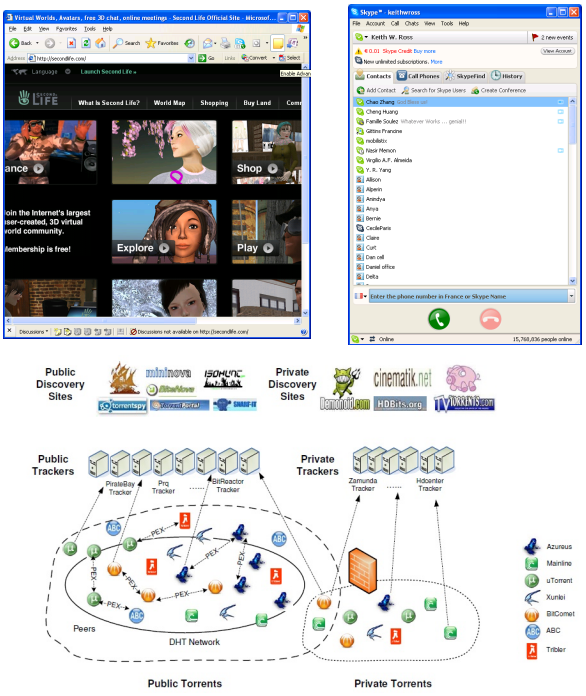
©material copyright 1996-2020
J.F.Kurose and K.W. Ross, All Rights Reserved



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
- TicToc
- ...

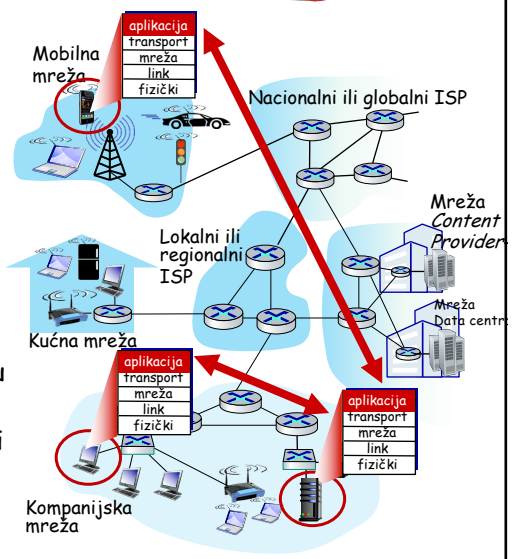
Kreiranje Internet aplikacije

Napisati programe koji

- se izvršavaju na različitim krajnjim sistemima I komuniciraju preko mreže.
- Softver Web servera komunicira preko mreže sa softverom browser-a

Ne piše se softver za mrežne uređaje

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

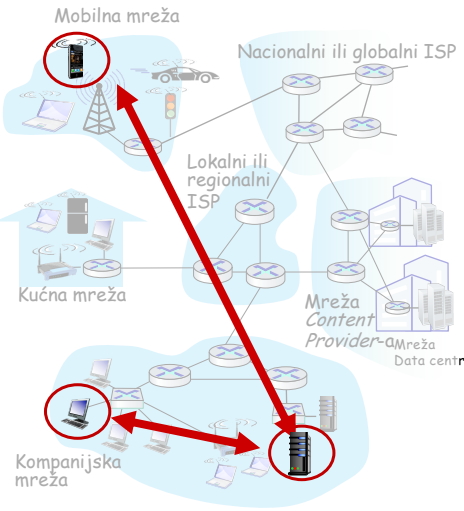


Arhitekture Internet aplikacija

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

5

Klijent-server arhitektura



Server:
- Uvijek aktivan
- Po pravilu permanentna IP adresa
- Standalone, Farma servera, Data centri

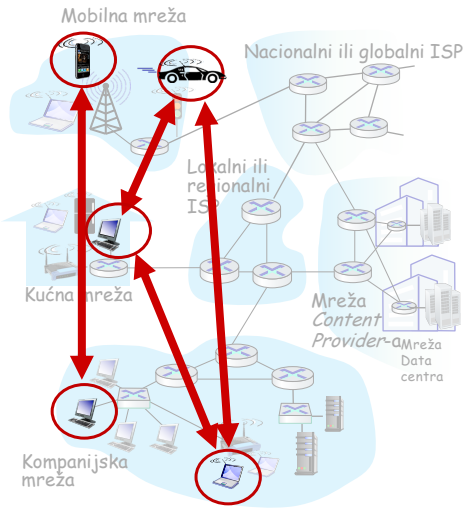
Klijenti:
- Komuniciraju sa serverom
- Mogu biti povremeno povezani
- Mogu imati dinamičku IP adresu
- Ne komuniciraju međusobno

Primjeri:
- Web, e-mail, file transfer,...

6

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



7

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 radi pronalaženja IP adrese korisnika sa kojima želi da čatuje

8

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

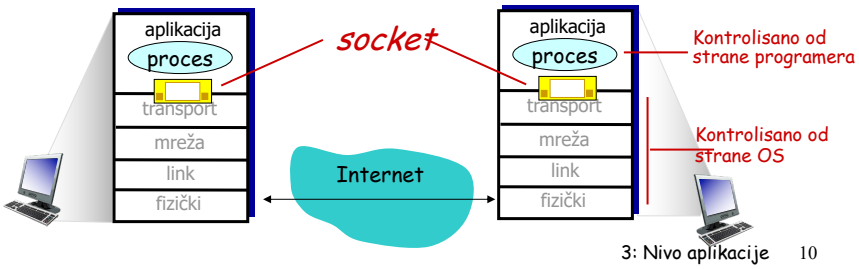
Klijent proces: proces koji inicijalizuje komunikaciju

Server proces: proces koji čeka da bude kontaktiran

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

Socket-i

- Proces šalje/prima poruke preko svog *socket*-a
- *Socket* je sličan 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* prijemne strane
  - Po dva su socket-a angažovana (jedan na izvorištu drugi na odredištu)



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?

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

- Identifikator uključuje i IP adresu i broj porta vezan za proces na hostu.
- Primjer brojeva porta:
  - HTTP server: 80
  - Mail server: 25
- VIŠE KASNIJE

Protokol nivoa aplikacije definiše

- Tipove poruka koje se razmjenjuju, npr., zahtjevi i poruke odgovora
- Sintaksu poruka tj. koja su polja i kako su odvojena
- Semantika polja, odnosno 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, Zoom,...

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

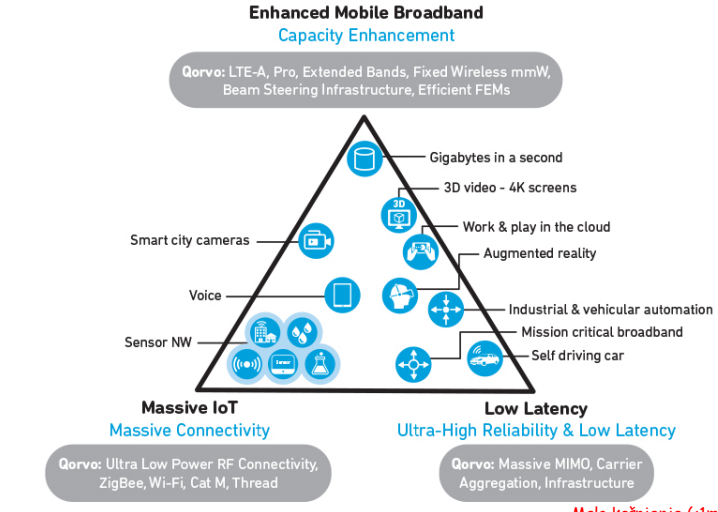
Zaštita
- Enkripcija, integritet podataka, …

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-5Mb/s	da, 10' ms
Stored audio/video	tolerantne	Isti kao gore	da, nekoliko s
Interaktivne igre	tolerantne	nekoliko kb/s i više	da, 10' ms
Instant messaging	bez	elastičan	da i ne

"5G trougao"

Velika brzina prenosa
(>100Mb/s, u piku više od 10Gb/s)



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 obezjeđuje:* tajming, garantovanje minimalnog opsega, sigurnost

UDP servisi:

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

P: Zašto oba? Zašto UDP?

Internet aplikacije: aplikacija, transportni protokoli

Aplikacija	Protokoli nivoa aplikacije	Transportni protokol
e-mail	SMTP [RFC 5321]	TCP
Interaktivne igre	WOW, FPS (privatni)	UDP ili TCP
Web	HTTP1.1 [RFC 7320]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP [RFC 7320], DASH	TCP
Internet telefonija	SIP [RFC 3261], RTP [RFC 3550], ili privatni (Skype,…)	TCP ili UDP

Zaštita i TCP

TCP i UDP *socket*-i

- Nema enkripcije
- Tekstualne poruke se prenose preko Interneta bez zaštite

TLS (Transport Layer Security)

- Omogućava privatnost komunikacije enkripcijom TCP konekcije i UDP komunikacije
- Integritet podataka
- Autorizacija od kraja do kraja
- Protokol nivoa aplikacije
- Aplikacije koriste TLS biblioteke, koje "komuniciraju"sa TCP
- Tekstualne poruke se šalju enkriptovane preko Interneta
- HTTPS=HTTP+TLS

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:

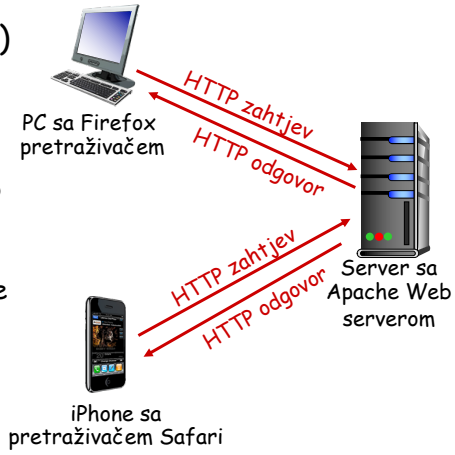
http://www.ucg.ac.me/index.html

protokol 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 konekcija se zatvara poslije završetka razmjene HTTP poruka

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

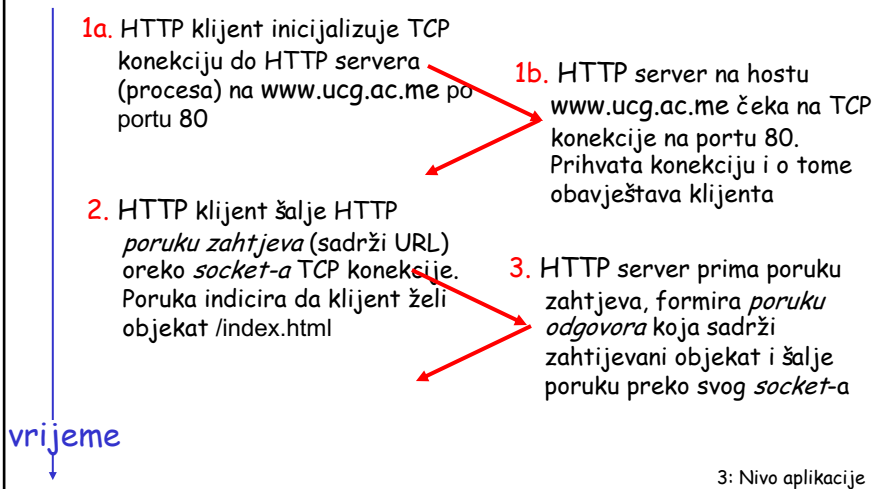
- Najviše jedan objekat se šalje preko TCP konekcije.
- Povlačenje više objekata podrazumijeva otvaranje više konekcija

Perzistentni HTTP

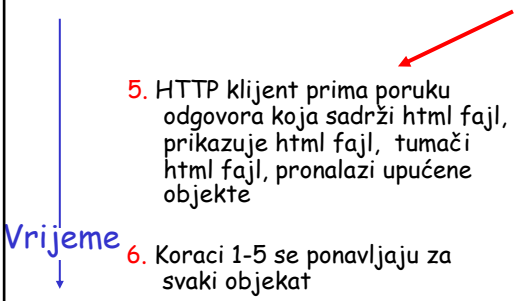
- Više objekata može biti poslato preko jedne TCP konekcije između klijenta i servera.

Neperzistentni HTTP

Neka korisnik unese sledeći URL http://www.ucg.ac.me



Neperzistentni HTTP(nastavak)



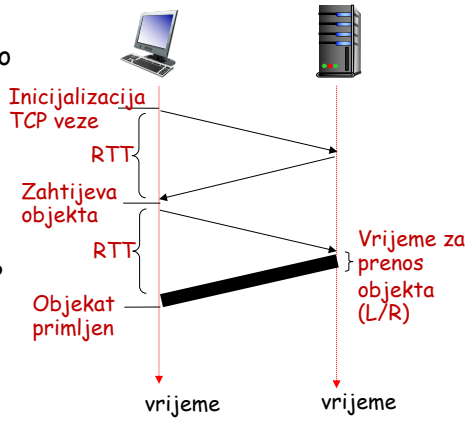
Neperzistentni HTTP: vrijeme odgovora

RTT (Round Trip Time):

- vrijeme prenosa paketa zanemarljive veličine 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 objekta
  - 2RTT+L/R
- Ako su index.html fajl i N-1 referenciranih objekata jednaki vrijeme njihovog prenosa je
  - N*(2RTT+L/R)



Perzistentni HTTP

Problemi neperzistentnog HTTP-a:

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

Perzistentni HTTP

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

Perzistentni bez "pipelining":

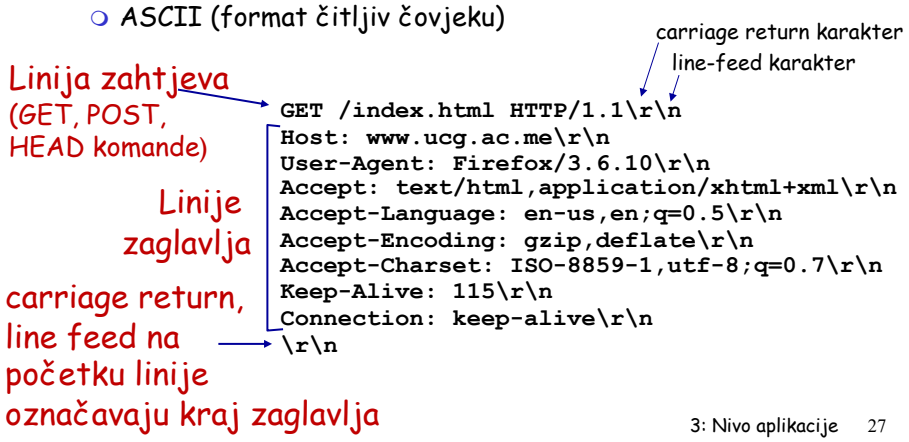
- klijent šalje novi zahtjev samo kada je prethodni odgovor primljen
- jedan RTT za svaki preneseni objekat
- kada nema zahtjeva TCP konekcija je slobodna
- ako su index.html fajl i N-1 referenciranih objekata jednaki vrijeme njihovog prenosa je
  - (N+1)RTT+N*L/R

Perzistentni sa "pipelining":

- klijent šalje zahtjeve odmah po dobijanju referenci objekata
- jedan RTT za sve referencirane objekte
- ako su index.html fajl i N-1 referenciranih objekata jednaki vrijeme njihovog prenosa je
  - 3RTT+N*L/R

HTTP poruka zahtjeva

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



Tipovi

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

HEAD:

- Zahtijeva samo zaglavlje koje se šalje ako je specificirani URL zahtijevan GET porukom

GET:

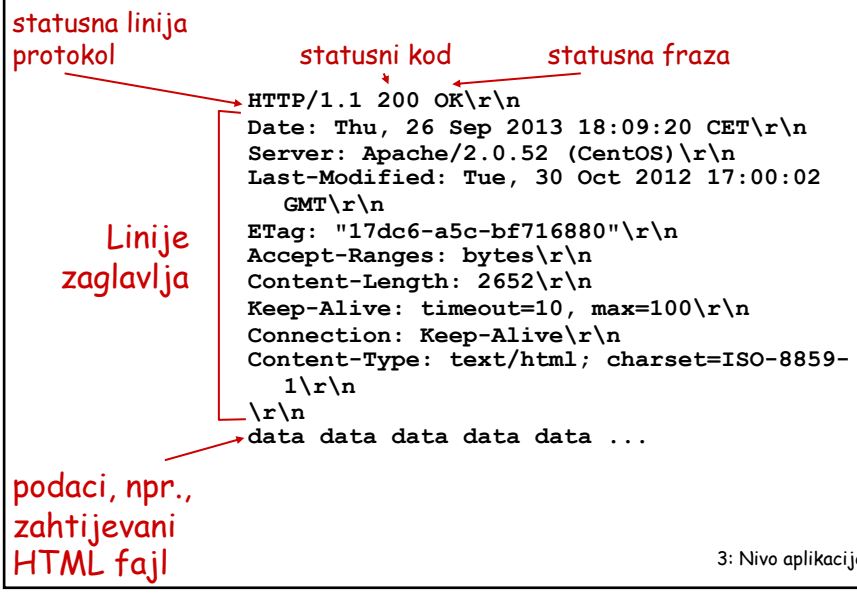
- Služi za povlačenje objekata sa servera
- Koristi se i za slanje korisnikovih podataka u sklopu URL polja HTTP GET poruke zahtjeva (poslije simbola '?'):

PUT:

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

```
www.google.com/animalsearch?monkeys&banana
```

HTTP poruka odgovora



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:

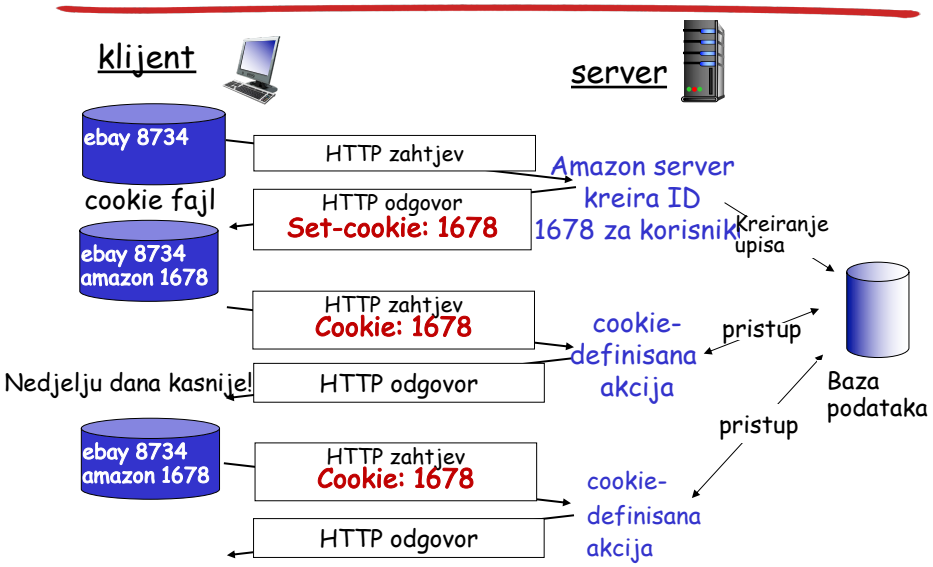
1. Linija zaglavlja Set-cookie u HTTP poruci odgovora
2. Linija zaglavlja Cookie u HTTP poruci zahtjeva
3. Cookie fajl se čuva na korisnikovom hostu i održava se od strane korisnikovog browser-a
4. Baza podataka na Web sajtu

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
- praćenje stanja korisnikove sesije (Web e-mail)

Pored toga

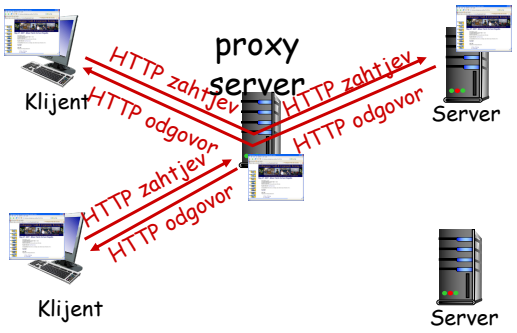
Cookies i privatnost:

- 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: ispunjavanje zahtjeva klijenta 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

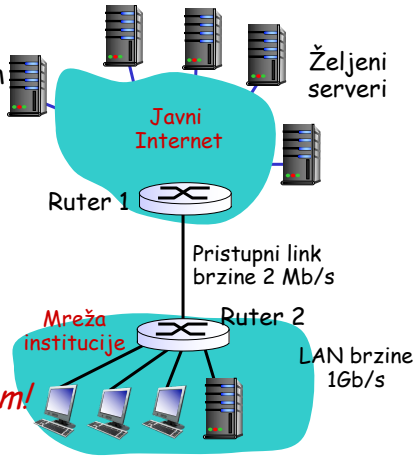
Primjer:

Pretpostavke:

- Srednja veličina objekta: 100000 b
- Srednji broj zahtjeva prema željenim serverima: 19 zahtjeva/s
- RTT između Rutera 1 i željenog servera: 2s
- Brzina na pristupnom linku: 2Mb/s

Posledice:

- Srednja brzina dolaznog saobraćaja: 100000 b*19 zahtj./s= 1,9 Mb/s
- Iskorišenje LAN mreže: (1,9Mb/s)/(1Gb/s)=0,0019=0,19%
- Iskorišenje pristupnog linka (1,9Mb/s)/(2Mb/s)=0,95= 95%
- Ukupno kašnjenje= RTT na Internetu + kašnjenje na pristupnom linku + LAN kašnjenje = 2s + minuti + ms = minuti



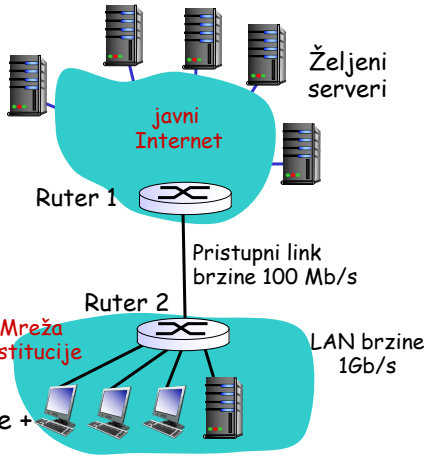
Primjer: brži pristupni link

Pretpostavke:

- Srednja veličina objekta: 100000b
- Srednji broj zahtjeva: 19 zahtjeva/s
- RTT od Rutera 1 do željenog servera:
- Brzina prenosa pristupnog linka: 100Mb/s

Posledice:

- Srednja brzina dolaznog saobraćaja: 100000 b*19 zahtj./s= 1,9 Mb/s
- Iskorišenje LAN mreže: (1,9Mb/s)/(1Gb/s)=0,0019=0,19%
- Iskorišenje pristupnog linka (1,9Mb/s)/(100Mb/s)=0,019= 1,9%
- Ukupno kašnjenje= Internet kašnjenje + kašnjenje na pristupnom linku+ LAN kašnjenje = 2s + ms + ms = 2s



Troškovi: povećanje brzine pristupa je skupo!

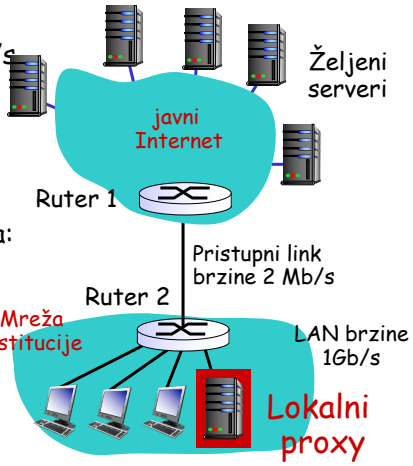
Primjer: Lokalni proxy

Pretpostavke:

- Srednja veličina objekta: 100000b
- Srednja brzina zahtjeva: 19 zahtj./s
- RTT od Rutera 1 do željenog servera: 2s
- Brzina pristupa: 2Mb/s

Posledice:

- Srednja brzina dolaznog saobraćaja: 100000 b*19 zahtj./s= 1,9 Mb/s
- Iskorišenje LAN mreže: (1,9Mb/s)/(1Gb/s)=0,0019=0,19%
- Iskorišćenje pristupnog linka= ?
- Ukupno kašnjenje= ?



Kako izračunati iskorišćenje pristupnog linka i kašnjenje?

Troškovi: proxy nije skup!

Primjer: Lokalni proxy

Izračunavanje iskorišćenja i kašnjenja:

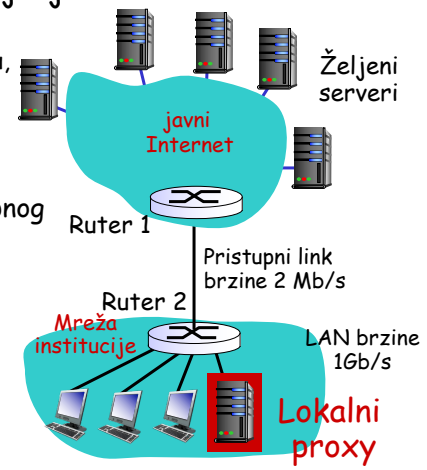
- Neka je vjerovatnoća pogađanja 0,4
  - 40% zahtjeva posluži na proxy serveru,
  - 60% zahtjeva na željenom serveru

Iskorišćenje pristupnog linka:

- 60% zahtjeva koristi pristupni link
- Srednja brzina prenosa preko pristupnog linka: 0,6*1,9 Mb/s = 1,14Mb/s
- Iskorišćenje pristupnog linka: 1,14/2 = 0,57

Ukupno kašnjenje

- 0,6 * (kašnjenje od željenih servera) +0,4 * (kašnjenje do proxy servera) = = 0,6*2s + 0,4* 1ms = ~ 1,2s
- Manje kašnjenje nego za pristupni link brzine 100Mb/s



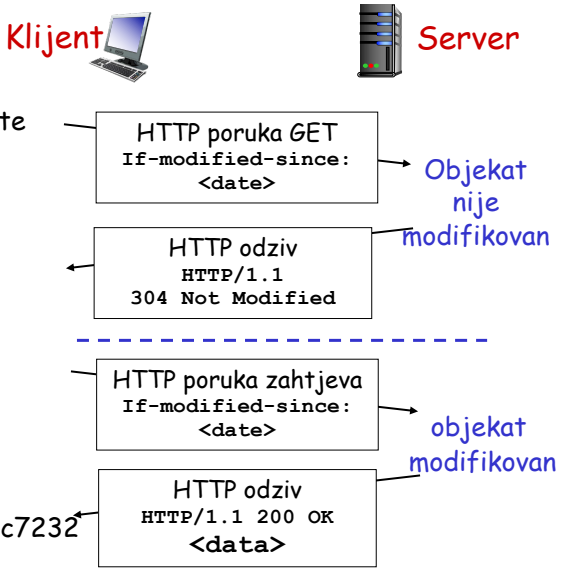
Conditional GET

- Cilj: ne slati objekat ako cache klijenta 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

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 *Head-Of-Line* (HOL) blokiranje
- Oporavak od gubitaka (retransmisija izgubljeni TCP segmenta) usporava prenos objekata

Cilj: dodatno smanjiti kašnjenje u slučaju stranice sa više referenciranih objekata

HTTP/2

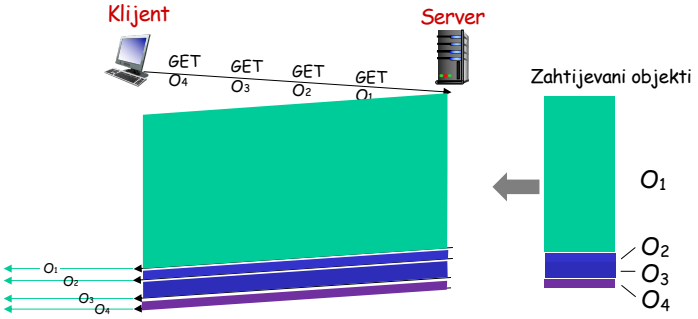
HTTP/2:

- [RFC 7540, 2015]
- povećava fleksibilnost servera u slanju objekata klijentu
- Potekao iz SPDY protokola (Google)
- Poruke zahtjeva, statusni kodovi, većina polja zaglavlja su isovjetna kao kod HTTP 1.1
- Redosled slanja zahtijevanih objekata je baziran na prioritetima koje je definisao klijent (ne mora biti FCFS)
- Omogućava slanje nezahtijevanog referenciranog objekta klijentu
- Dijeli objekte na frejmove, raspoređuje frejmove radi smanjenja HOL blokiranja

HTTP/2: ublažavanje HOL blokiranja

HTTP 1.1:

klijent zahtijeva veliki objekat (na primjer video fajl) i 3 manja objekta

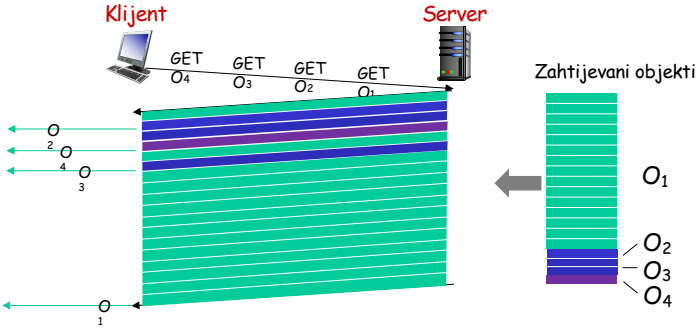


Slanje objekata po redosledu zahtjeva: O_2, O_3, O_4 čekaju iza O_1

HTTP/2: ublažavanje HOL blokiranja

HTTP/2:

objekti se dijele ne frejmove, "izmiješano" slanje frejmova



O_2, O_3, O_4 se brzo prenose, malo kašnjenje se dodaje O_1

HOL blokiranje još vijek postoji jer se sve obavlja po jednoj TCP konekciji na kojoj se garantuje pouzdani tok bajta!

HTTP/3

HTTP/2 funkcioniše preko jedne TCP konekcije:

- Oporavak od gubitka paketa korišćenjem TCP pouzdanog prenosa i dalje usporava prenos objekata jer se obavlja na nivou bajta za sve HTTP-ove tokove jedne TCP konekciju
- Kao i kod HTTP 1.1, *browser*-i otvaraju više paralelnih HTTP tokova radi smanjenja usporavanja i povećanja propusnosti
- Nema zaštite preko Vanilla TCP konekcije pa se za zaštitu koristi TLS

HTTP/3:

- Više pipeline prenosa preko UDP protokola korišćenjem QUIC protokola
- QUIC (Google uveo za Chrome, IETF za ostale browsere)
- QUIC protokol garantuje pouzdani prenos preko UDP prokola uz mnogo manje kašnjenje od TCP protokola!
- QUIC nije implementiran u OS.
- QUIC je podržan od značajnog broja Web klijenata i Web servera
- Ima zaštitu na nivou aplikacije a ne na nivou bajta kao kod TLS-a

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