Arhitekture mrežnih aplikacija

Internet protokoli nivoa aplikacije

- Mrežne aplikacije predstavljaju razlog postojanja računarskih mreža.
- Najpopularnije mrežne aplikacije
  - elektronska pošta;
  - Web;
  - trenutna razmjena poruka;
  - prijavljivanje na udaljene računare (na primjer Telnet);
  - P2P (peer-to-peer) razmjena datoteka;
  - transfer datoteka između dva naloga na dva računara (na primjer, FTP);
  - mrežne igrice;
  - protok uskladištenog video materijala u realnom vremenu;
  - Internet telefonija;
  - video konferencije u realnom vremenu.

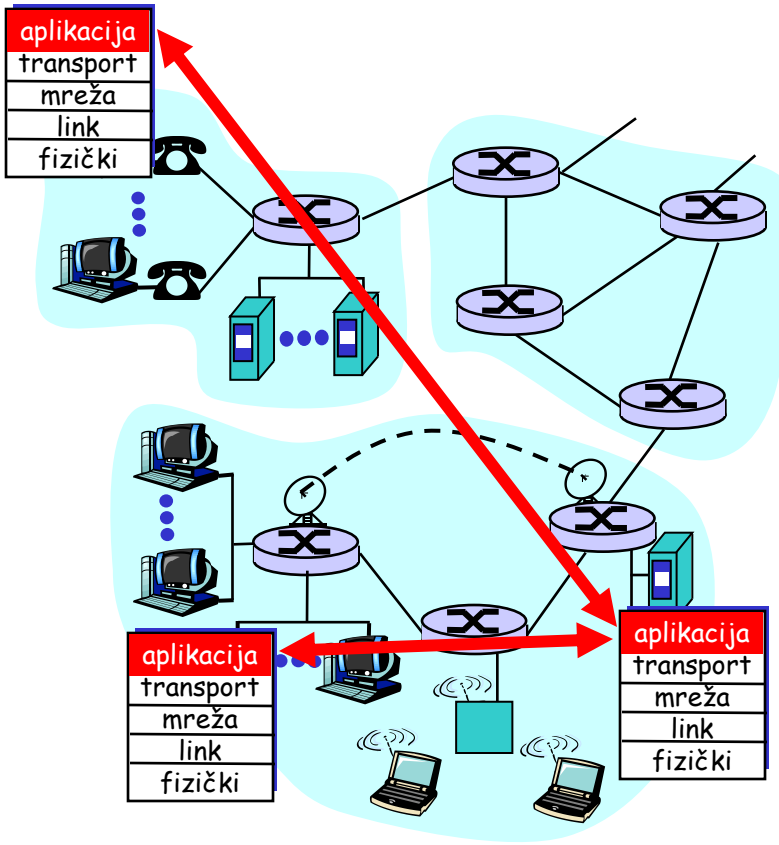
Kreiranje mrežne aplikacije

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

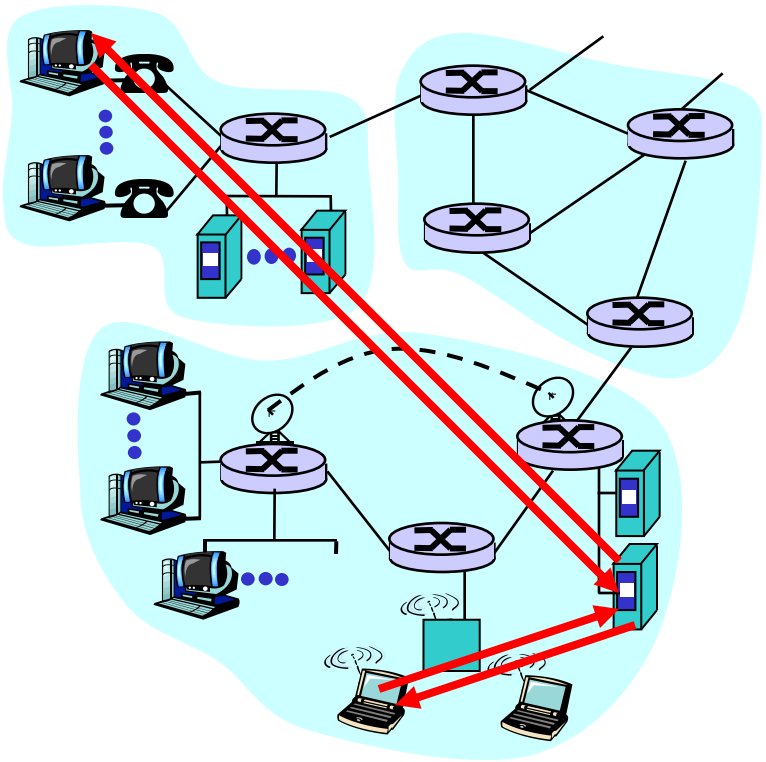
- mrežni uređaji na kičmi ne funkcionišu na nivou aplikacije
- ovakav dizajn dozvoljava brzi razvoj aplikacija



Arhitekture aplikacija

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

Klijent-server arhitektura



Server:

- Uvijek uključen
- Permanentna IP adresa
- Farme servera

Klijenti:

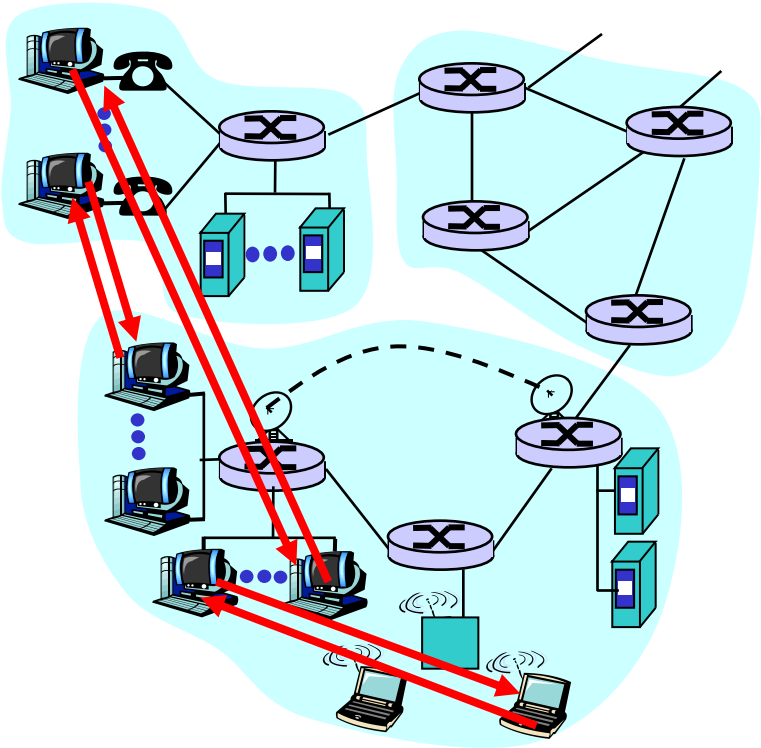
- Komuniciraju sa serverom
- Povezan povremeno
- Može imati dinamičku IP adresu
- Ne komuniciraju međusobno

P2P arhitektura

- Server nije uvijek uključen
- Proizvoljni krajnji sistemi mogu direktno komunicirati
- Peer-ovi su povremeno povezani i mogu mijenjati IP adrese
- Primjer: Gnutella

Vrlo skalabilni

Teški za upravljanje



Hibrid klijent-server i P2P

Napster

- P2P prenos fajlova
- Traženje fajlova centralizovano:
  - Peer-ovi registruju svoj sadržaj na centralnom serveru
  - Peer-ovi se raspituju kod istog centralnog servera za lociranje sadržaja

Instant messaging

- Čatovanje dva korisnika je P2P
- Detektovanje prisutnosti i lokacije je centralizovano:
  - Korisnik registruje svoju IP adresu na centralnom serveru 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

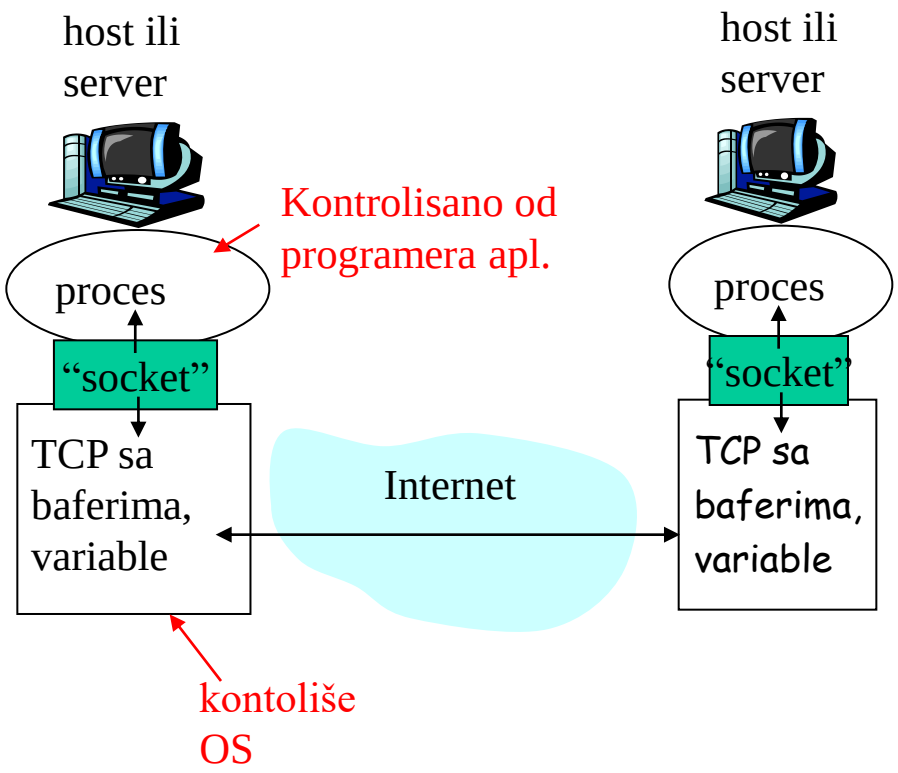
Klijent proces: proces koji inicijalizuje komunikaciju

Server proces: proces koji čeka da bude kontaktiran

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

Proces komunikacije mrežom

- Proces šalje/prima poruke preko svog "socket"-a
- "socket" je analogan vratima
  - proces koji 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)

Proces adresiranja:

- Za proces koji prima poruke, mora postojati identifikator
- Svaki host ima jedinstvenu 32-bitnu IP adresu
- P: Da li je IP adresa hosta na kojem se proces izvršava dovoljna za identifikaciju procesa?
- O:?

- 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 & poruke odgovora
- Tipove sintaksi poruka: koja su polja & kako su odvojena
- Semantika polja, npr., značenje informacija u poljima
- Pravila vezana kada i kako se šalju poruku i kako se odgovara na njih

Javni (*public*) protokoli:

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

Privatni (*proprietary*) protokoli:

- npr, KaZaA, Skype,…

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) zahtijeva malo kašnjenje

Brzina prenosa

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

Transportni servisni zahjevi zajednički za sve aplikacije

Aplikacija	Gubici	Brzina prenosa	Vrem. osjet.
file transfer	bez	elastičan	ne
e-mail	bez	elastičan	ne
Web dokumenti	bez	elastičan	ne
real-time audio/video	tolerantne	audio: 5kb/s-1Mb/s video:10kb/s-5Mb/s	da, 100's ms
stored audio/video	tolerantne	Isti kao gore	da, nekoliko s
Interaktivne igre	tolerantne	nekoliko kb/s i više	da, 100's ms
instant messaging	bez	elastičan	da i ne

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, pozdanost, kontrolu protoka, kontrolu zagušenju, 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 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	privatni (npr. RealNetworks)	TCP ili UDP
Internet telefonija	privatni (npr., Dialpad, Skype)	UDP (i TCP)

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 sadrži više referenci objekata
- Svaki objekat se adresira sa URL (Uniform Resource Locators)
- Primjer URL:

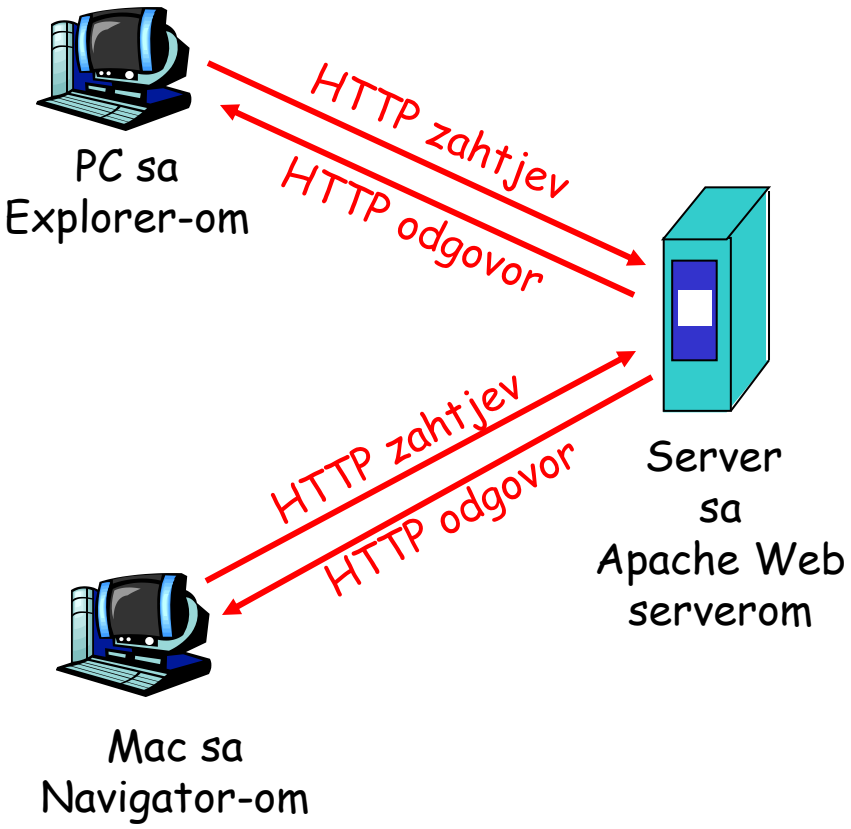
http://www.ethereal.com/distribution/win32

ime hosta | ime puta

HTTP pregled

HTTP: hypertext transfer protokol

- Web-ov protokol nivoa aplikacije
- klijent/server model
  - *klijent:* "browser" koji zahtijeva, prima, prikazuje Web objekte
  - *server:* Web server šalje objekte kao odgovor na zahtjeve
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2616 (1998)



HTTP pregled (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)

HTTP veze

Neperzistentni (neistrajan) HTTP

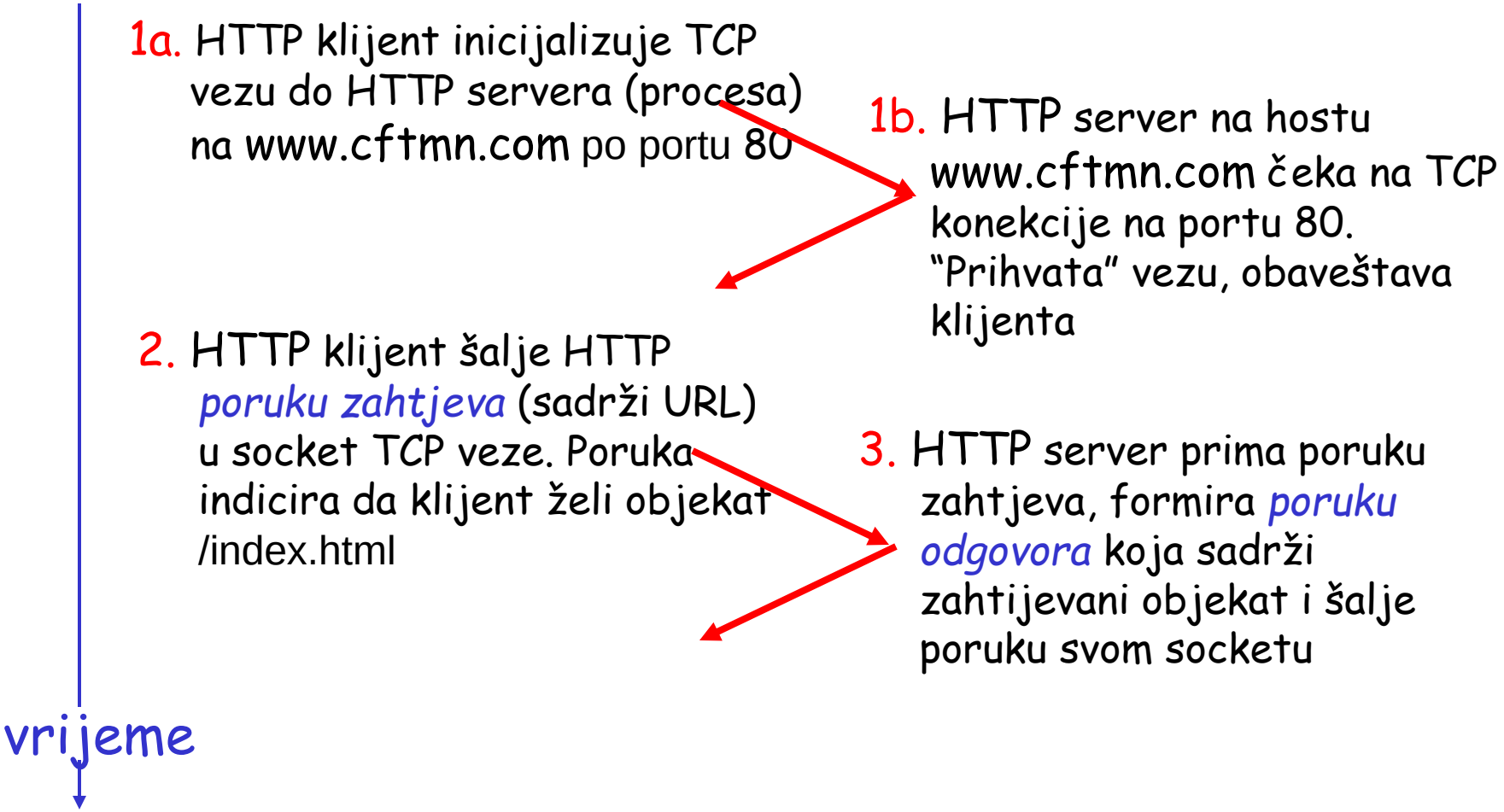
- Najviše jedan objekat je poslat preko TCP konekcije.
- HTTP/1.0 koristi neperzistentni HTTP

Perzistentni HTTP

- Više objekata može biti poslato preko jedne TCP veze između klijenta i servera.
- HTTP/1.1 koristi perzistentne veze u default modu

Neperzistentni HTTP

Pretpostavimo korisnik unese sledeći URL
http://www.cftmn.com/index.html



Neperzistentni HTTP (nastavak)

4. HTTP server zatvara TCP vezu.

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

6. Koraci 1-5 se ponavljaju za svaki objekat

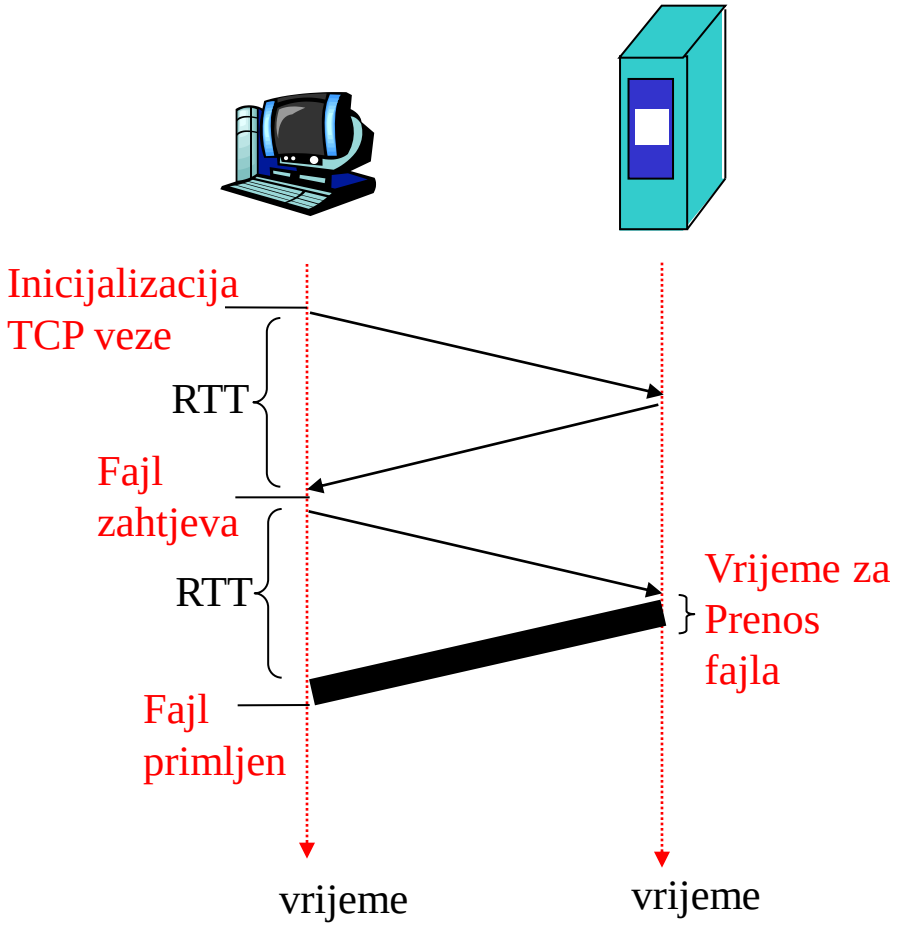
Modelovanje vremena odgovara

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:

- zahtjeva 2 RTT po objektu
- OS mora raditi i dodijeliti resurse hosta za svaku TCP vezu
- browser-i često otvaraju paralelne TCP veze za povlačenje upućenih objekata

Perzistentni HTTP

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

Perzistentni bez "pipelining":

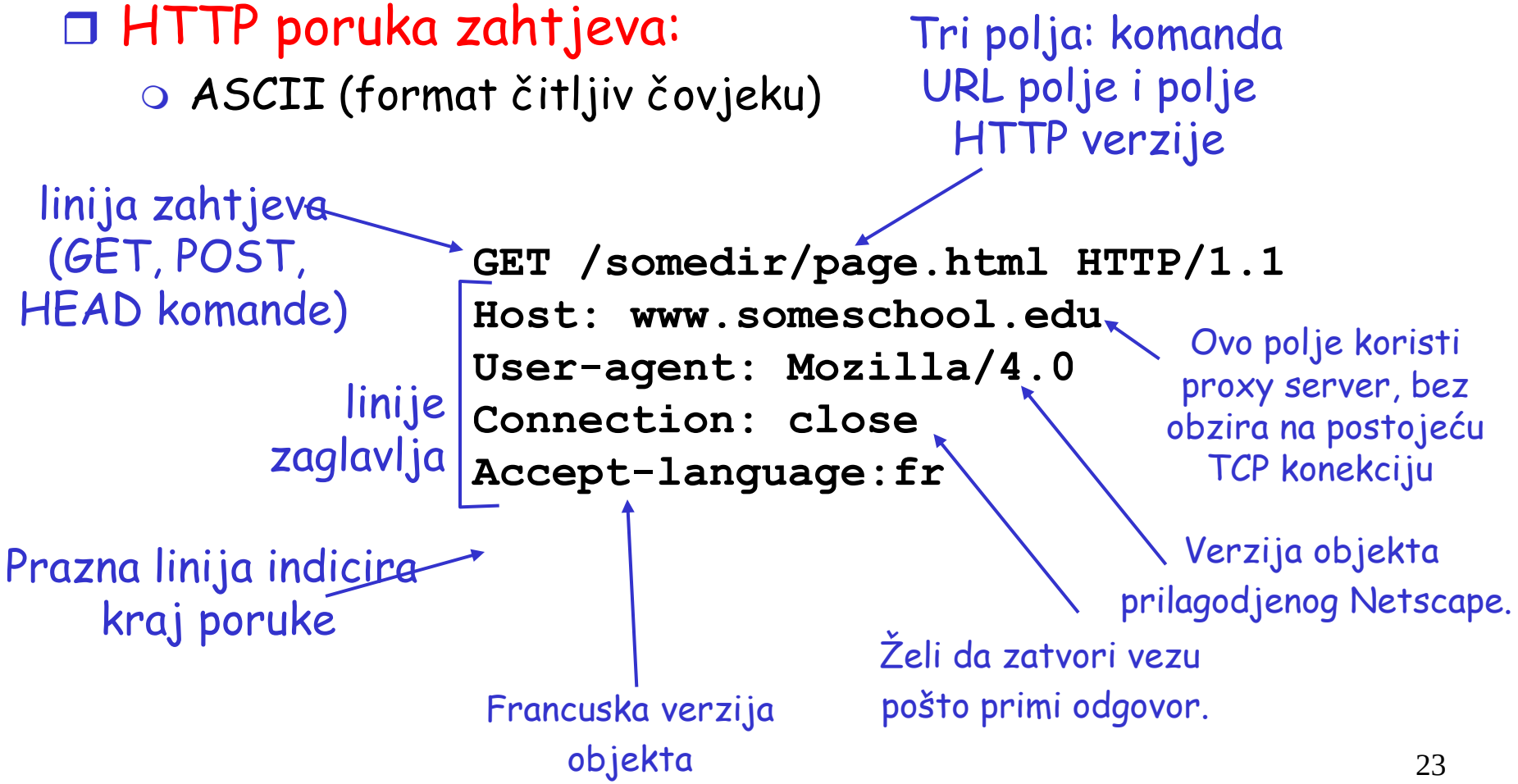
- Klijent šalje novi zahtjev samo kada je prethodni odgovor primljen
- jedan RTT za svaki upućeni objekat
- Kada nema zahtjeva TCP konekcija je slobodna

Perzistentni sa "pipelining":

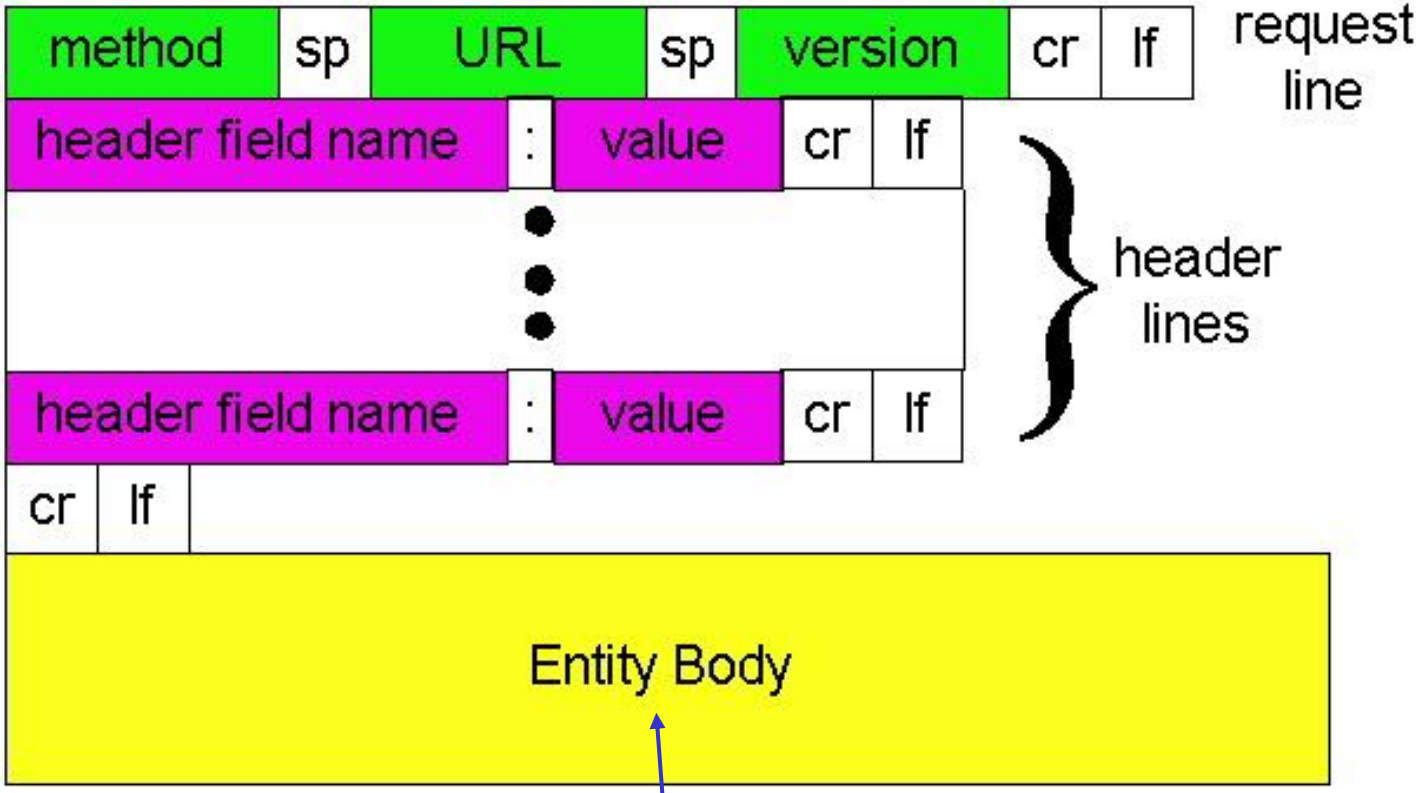
- default u HTTP/1.1
- klijent šalje zahtjeve odmah po dobijanju referenci objekata
- Veličine svega jedan RTT za sve reference objekata

HTTP poruka zahtjeva

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



HTTP poruka zahtjeva: opšti format



Prazan za GET, popunjen za POST sa informacijama koje se odnose na popunjvanjenekih formi (npr search). POST ima istu funkciju kao GET sa tom razlikom što korisnik traži objekat na bazi sadržaja koji je poslao. I GET se može koristiti u te svrhe preko URL polja.

Tipovi

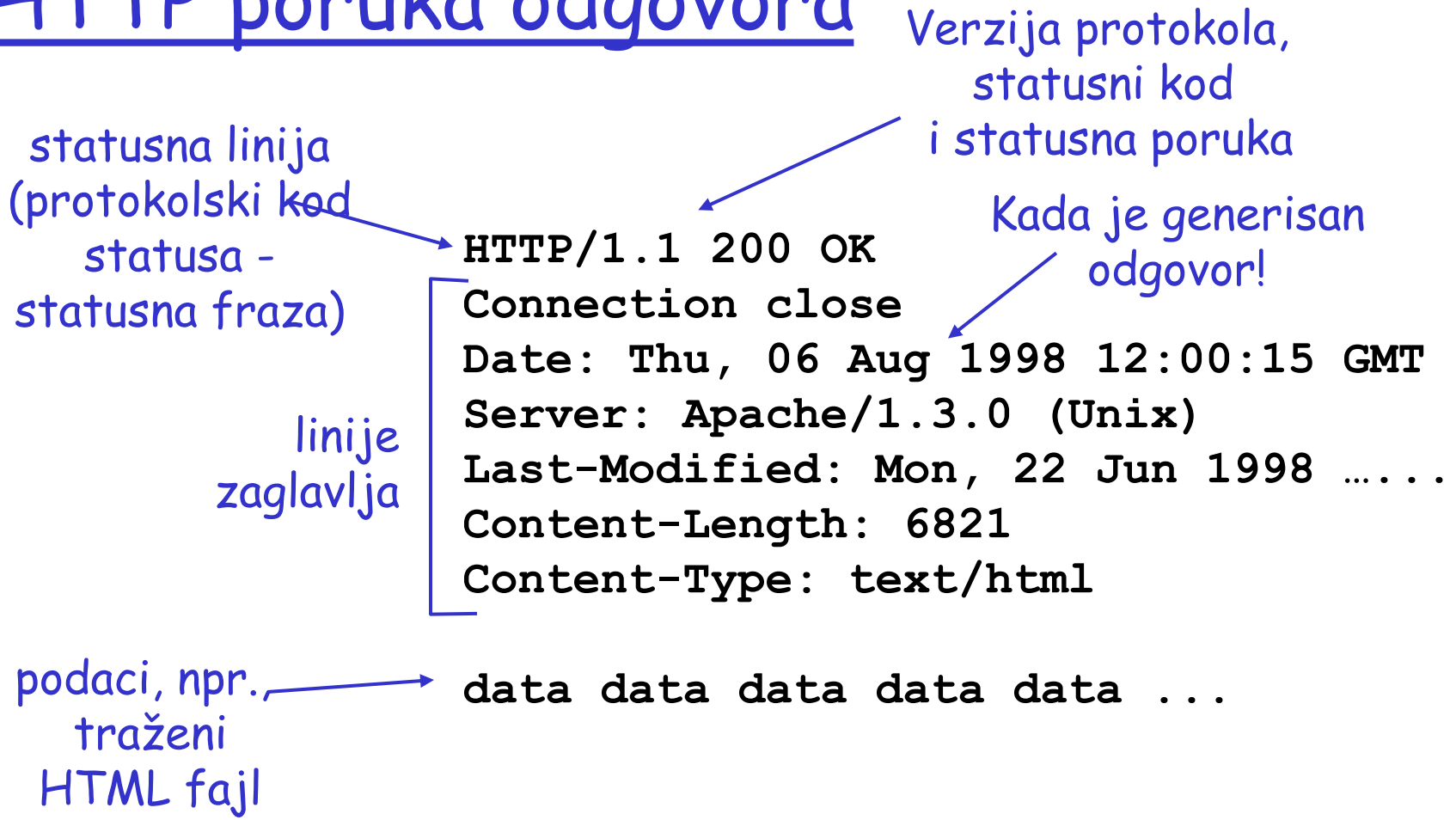
HTTP/1.0

- GET
- POST
- HEAD
  - Pita servera 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



HTTP kodovi statusa odgovora

U prvoj liniji u server->klijent poruci odgovora.

Nekoliko primjera kodova statusa i odgovarajućih poruka:

200 OK

- Zahtjev uspješan, zahtijevani objekat kasnije u ovoj poruci

301 Moved Permanently

- Zahtijevani objekat preseljen, nova lokacija specificirana kasnije u ovoj 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