SVEUČILIŠTE U ZAGREBU
Fakultet elektrotehnike i računarstva

Diplomski studij

Informacijska i komunikacijska tehnologija:

Telekomunikacije i informatika

Računarstvo:

Programsko inženjerstvo i informacijski sustavi

Računarska znanost

Raspodijeljeni sustavi

12. Sustavi s ravnopravnim sudionicima

Ak. god. 2020./2021.

Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava. *Tekst licence preuzet je s http://creativecommons.org/*

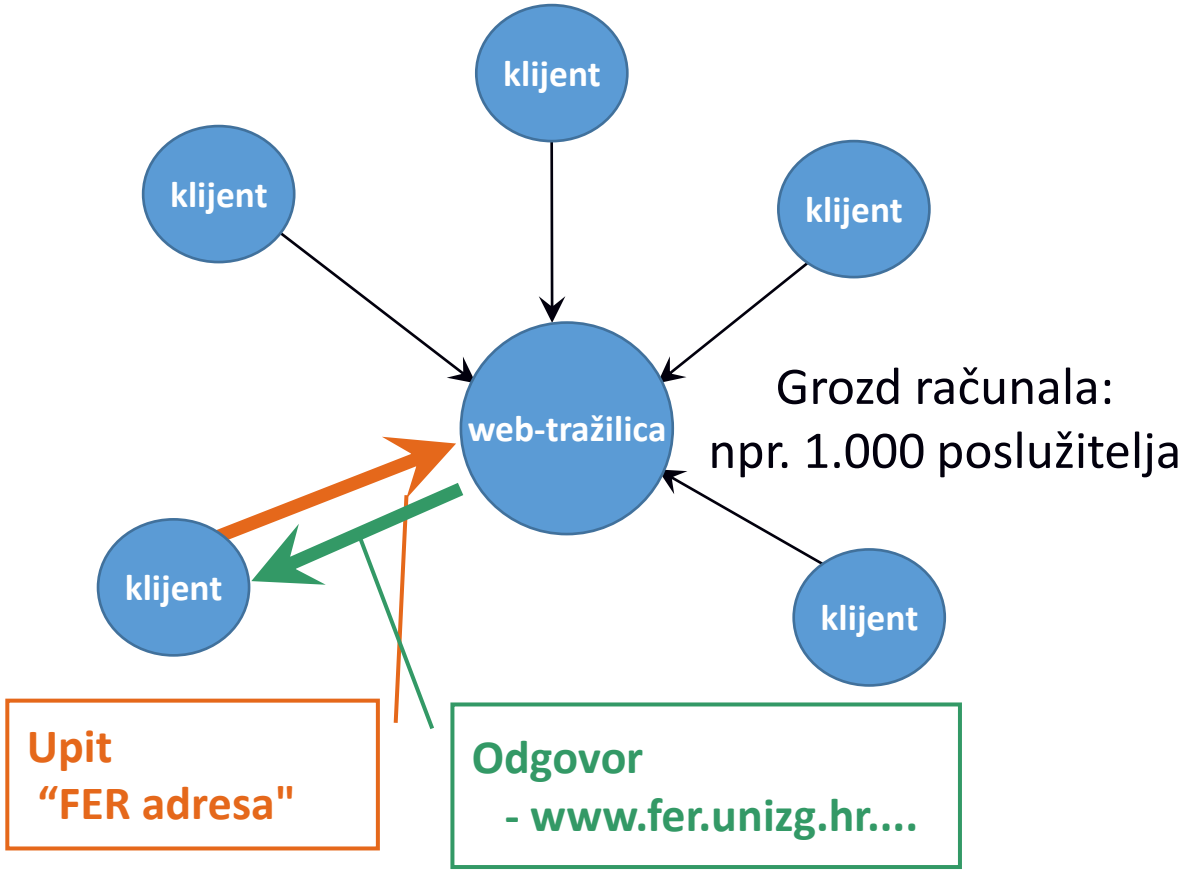
Sadržaj predavanja

- Centralizirani i decentralizirani raspodijeljeni sustavi
- Definicija sustava P2P
- Nestrukturirani sustavi P2P
- Strukturirani sustavi P2P
- Primjeri sustava P2P

Centralizirani raspodijeljeni sustavi (1)

Primjer: web-tražilica

- $45*10^9$ web stranica ≈ 225 TB tekstualnih dokumenata
- za održavanje indeksa veličine 50 TB treba oko 1.000 računala (ovisi o raspoloživoj radnoj memoriji računala)



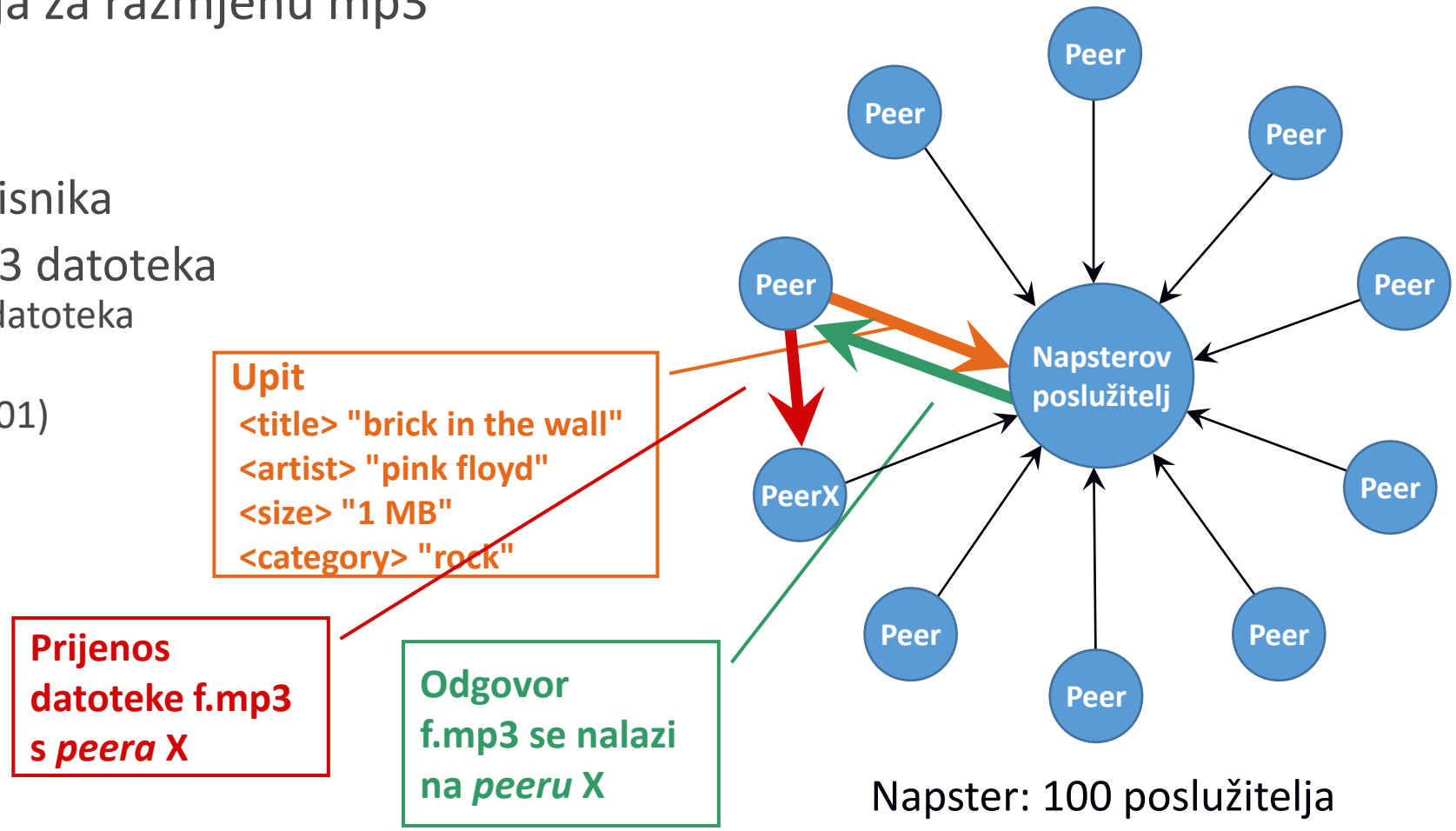
Centralizirani raspodijeljeni sustavi (2)

Model klijent-poslužitelj

- centralni koordinator koji prihvaća sve korisničke upite
- indeks dokumenata je raspodijeljen, pretraživanje je raspodijeljeno u grozdu računala (*cluster*), no organizacija pretraživanja je centralizirana
- prednosti
  - efikasnost, kratko vrijeme odgovora
  - relativno jednostavna organizacija indeksa, globalno rangiranje...
- nedostaci
  - cijena (infrastruktura, administracija...)

Decentralizirani raspodijeljeni sustavi (1)

- Primjer: aplikacija za razmjenu mp3 datoteka
  - npr. Napster
  - 1,570,000 korisnika
  - 2,000,000 mp3 datoteka
    (u prosjeku 220 datoteka po korisniku)
    (podaci za 02/2001)



Napster: 100 poslužitelja

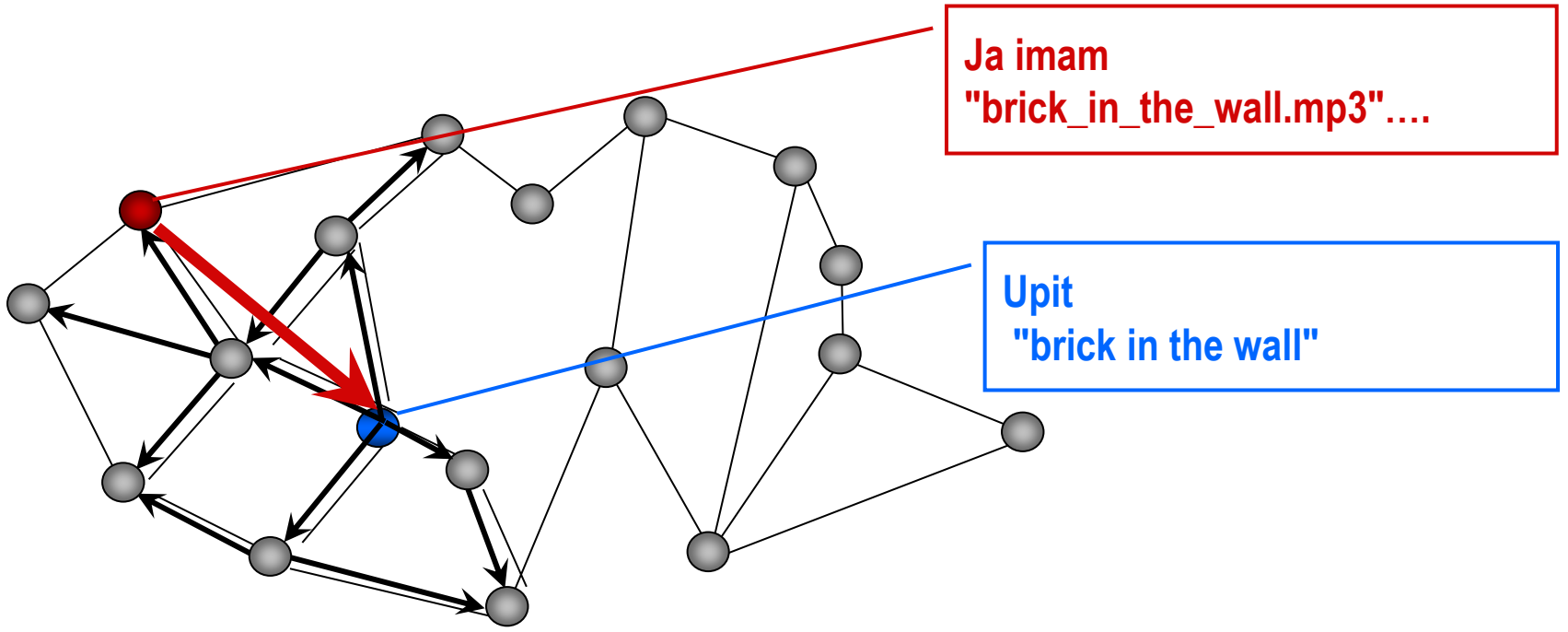
Decentralizirani raspodijeljeni sustavi (2)

- pretraživanje je i dalje centralizirano
  - postoji centralizirani indeks s podacima o lokaciji datoteka
- pohrana i *download* datoteke je decentraliziran
- broj potrebnih poslužitelja je znatno manji jer se resursno zahtjevne operacije izvode na decentralizirani način
- prednosti
  - dijeljenje resursa, svaki čvor (*peer*) "plaća" sudjelovanje u mreži vlastitim resursima (disk, mreža, datoteke)
  - znatno manja cijena infrastrukture i održavanja
- nedostaci
  - centralizirano pretraživanje i jedinstvena točka ispada

Decentralizirani raspodijeljeni sustavi (3)

Primjer: aplikacija za razmjenu datoteka

- npr. Gnutella
- 40.000 čvorova, $3 \cdot 10^6$ datoteka (podaci iz 08/2000)



Potpuno decentralizirani sustav

Decentralizirani raspodijeljeni sustavi (4)

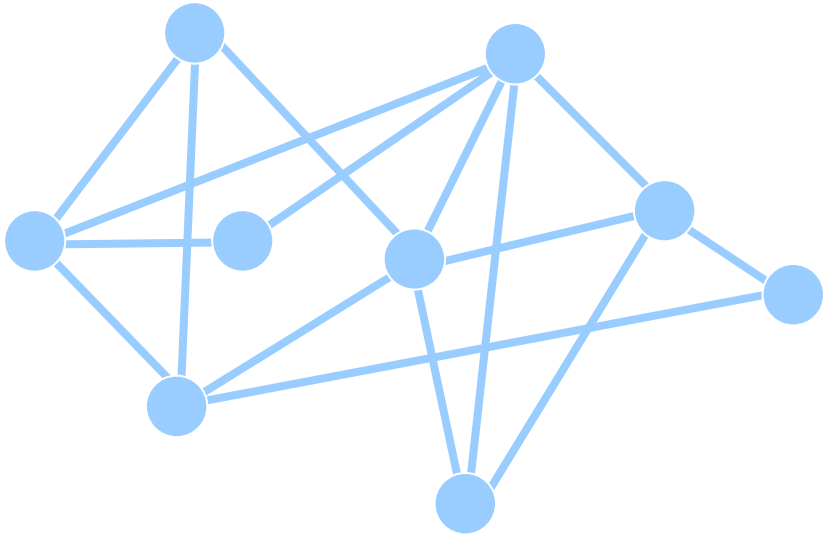
- Gnutella je primjer potpuno decentraliziranog sustava
  - svi čvorovi sudjeluju u procesu pretraživanja (ne postoji centralizirani indeks)
  - brzo pronalazi datoteke koje su replicirane na velikom broju čvorova
- prednosti
  - skalabilnost sustava
  - ne postoji posebna infrastruktura niti potreba za održavanjem sustava
  - ne postoji jedinstvena točka ispada
- nedostaci
  - velika količina generiranog mrežnog prometa
  - ne postoji garancija pronalaska tražene datoteke
  - problem: tzv. *free-riding*, postoje peerovi koji ne dijele datoteke

Sadržaj predavanja

- Centralizirani i decentralizirani raspodijeljeni sustavi
- Definicija sustava P2P
- Nestrukturirani sustavi P2P
- Strukturirani sustavi P2P
- Primjeri sustava P2P

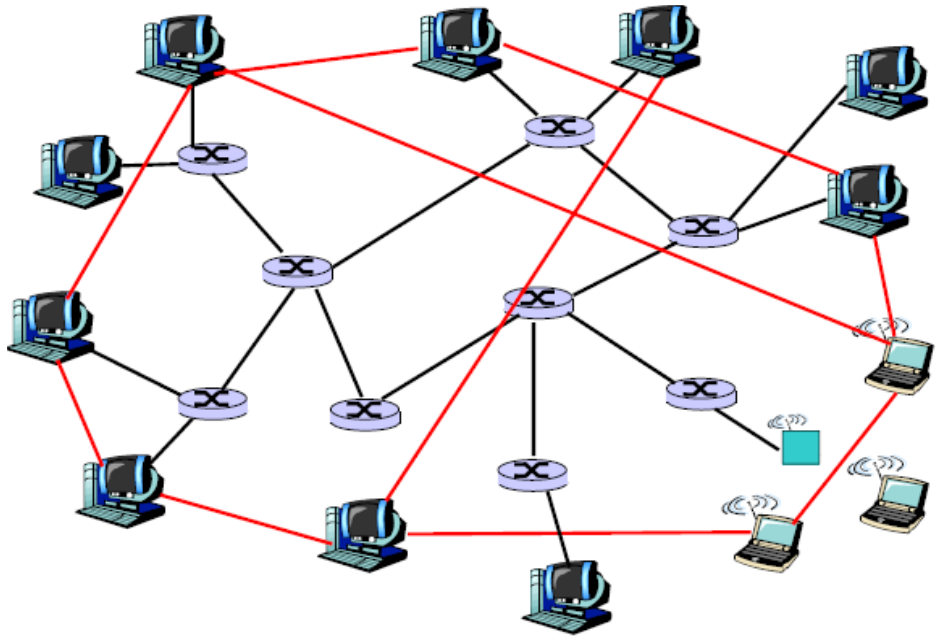
Definicija sustava peer-to-peer (P2P)

- mreža ravnopravnih sudionika tj. “čvorova” - *peerova*
- svaki *peer* istovremeno obavlja funkciju poslužitelja i klijenta
- svaki čvor “plaća” sudjelovanje u mreži nudeći dio vlastitih resursa (memorija, CPU, mreža) ostalim čvorovima
- *peerovi* ulaze i izlaze iz sustava po volji, dinamična i nestabilna topologija
- potencijalno sustav P2P nudi neograničene resurse (broj *peerova* nije ograničen)



Overlay network

- "prekrivajuća mreža" (*overlay network*) nad stvarnom mrežnom topologijom
- *peerovi* su programi koji se izvode na aplikacijskom sloju
- koristi resurse krajnjih računala koji čine posebnu mrežu na aplikacijskom sloju neovisnu o mrežnoj topologiji
- mreža *peerova* se konstantno mijenja



Mreža *peer*ova

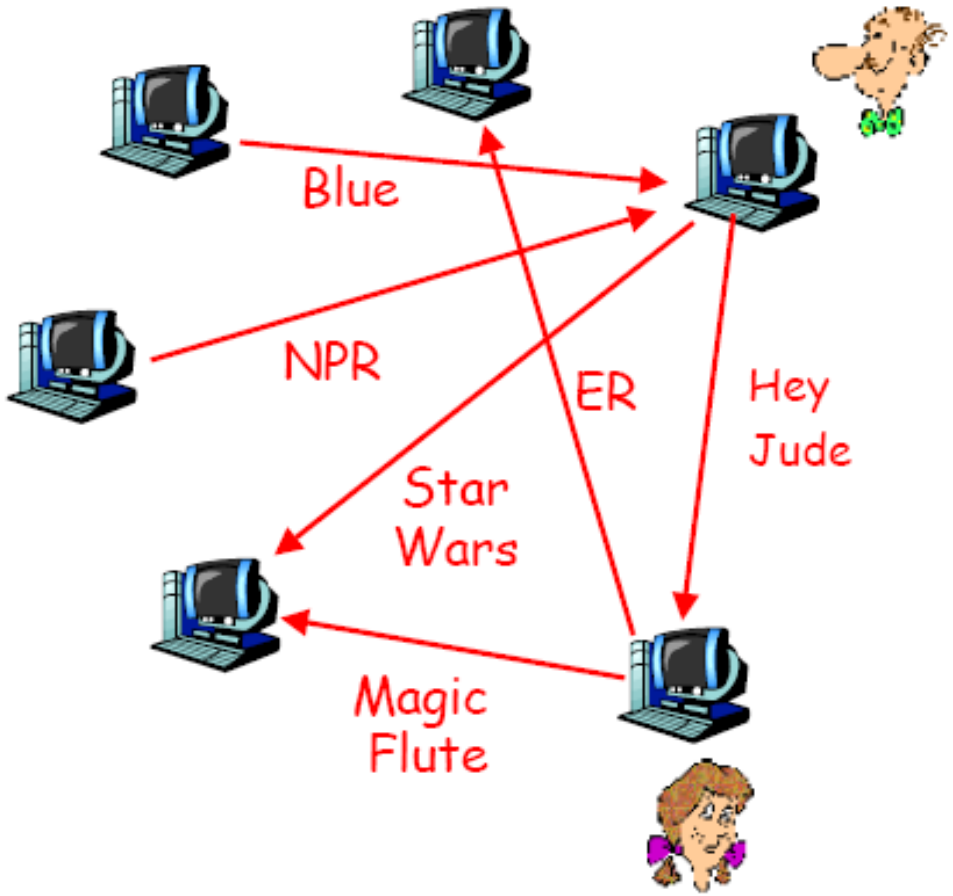
- Kada su 2 *peera* susjedi?
  - otvorena TCP konekcija ili
  - virtualne grane među *peerovima*, *peer* zna IP adresu drugog *peera*
- Kako se održava mreža *peerova*?
  - mreža je izrazito nestabilna
  - npr. *peer* periodički provjerava stanje susjeda (porukama *ping*)
  - ako je susjed nedostupan, briše se iz liste susjeda
  - potreban je poseban algoritam za otkrivanje novih susjeda
  - poseban algoritam za dodavanje novog *peer*a u postojeću mrežu (najčešće poznaje listu *peer*ova za inicijalni kontakt)

Obilježja sustava P2P

- decentralizirani raspodijeljeni sustav
  - nema centralizirane koordinacije među *peerovima*
  - ne postoji jedna točka ispada
- samoorganizirajuća mreža čvorova
  - *peerovi* su međusobno neovisni
  - dodavanje novih čvorova, izlazak čvora iz sustava te ispad čvorova je podržano organizacijom mreže P2P i definiranim protokolima
- globalni informacijski sustav bez velikih početnih ulaganja
  - raspodijeljena instalacija peerova i održavanje

Osnovna zadaća sustava P2P (1/2)

- Pronalaženje podataka, resursa, objekata (npr. datoteka) u sustavima P2P!



Osnovna zadaća sustava P2P (2/2)

Kako pronaći podatak ***d*** u mreži peerova?

- “naivno rješenje”: poslati upit svim peerovima u mreži
  - problemi: moram znati adrese svih peerova, što je s mrežnim prometom?
- “manje naivno rješenje”: poslati upit odabranim peerovima u mreži
  - problemi: kako odabrati peerove, hoću li sigurno pronaći podatak ***d***?
- “pametnije” rješenje
  - pohraniti podatak ***d*** na odabrani peer ***p*** (ili odabrane peerove): dovoljno je znati adresu peera ***p*** da mu možemo proslijediti upit
  - postoji algoritam koji povezuje peera ***p*** s podatkom ***d***, a svi peerovi u mreži znaju taj algoritam
  - isti algoritam se koristi pri pohranjivanju i traženju podatka

Vrste sustava P2P

- nestrukturirani sustavi
  - mrežna topologija nema definiranu strukturu (“manje naivno rješenje”)
  - mrežu *peerova* čini slučajan graf, npr. peer “poznaje” svoja četiri susjeda i preko njih pretražuje cijelu mrežu
  - primjeri: Freenet, Gnutella, KaZaA, BitTorrent

- strukturirani sustavi
  - mrežna topologija je definirana i ima posebnu strukturu (“pametnije” rješenje)
  - podatku ***d*** možemo pridijeliti ključ ***k*** (svaki peer može odrediti ***k*** za ***d***)
  - podatak ***d*** je pohranjen na *peeru* koji je “zadužen” za ključ ***k***, a ne na peeru koji ga kreira
  - primjeri: CAN, Chord, P-Grid, Pastry

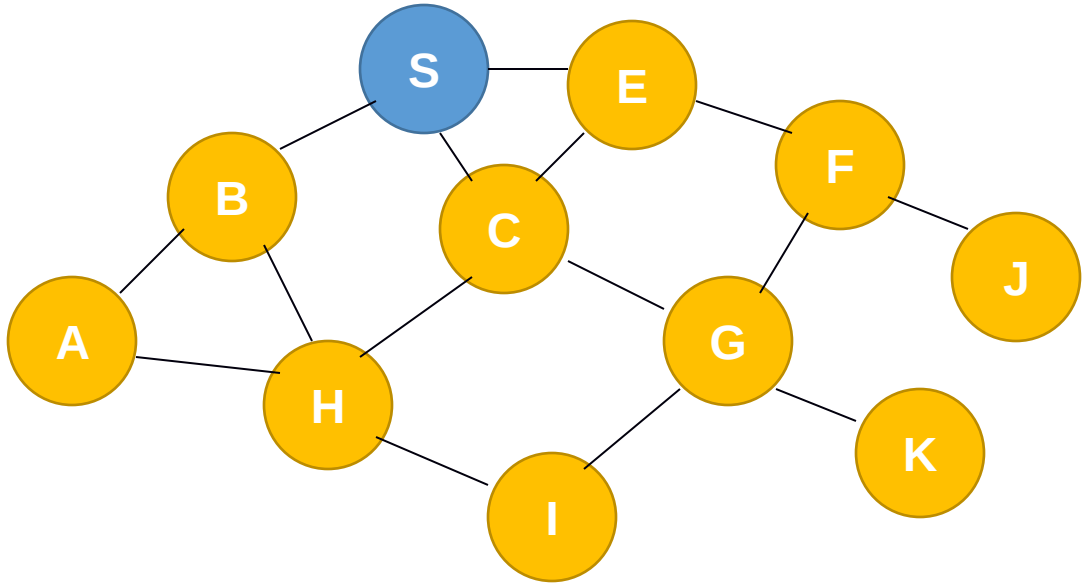
Sadržaj predavanja

- Centralizirani i decentralizirani raspodijeljeni sustavi
- Definicija sustava P2P
- Nestrukturirani sustavi P2P
- Strukturirani sustavi P2P
- Primjeri sustava P2P

Nestrukturirani sustavi P2P

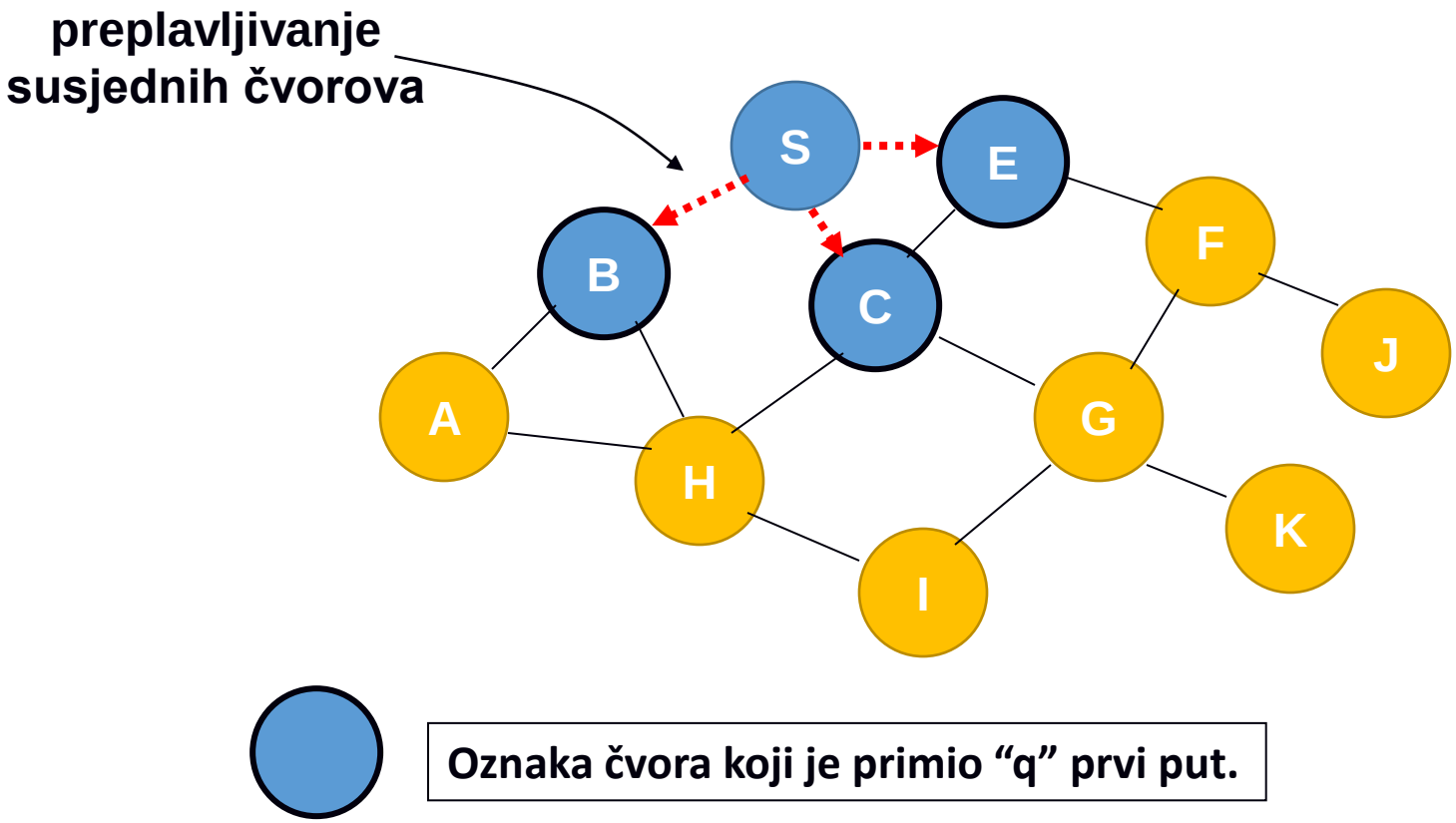
- podatak (npr. datoteka) je pohranjen na *peeru* koji ga kreira, ne postoji veza između podatka ***d*** i peera ***p***
- moguće je pohraniti kopiju podatka na peerovima koji ga kopiraju s originalnog peera
- pretraživanje se izvodi preplavljivanjem ili slučajnim izborom (*random walk*), itd.

Usmjeravanje upita: preplavljivanje

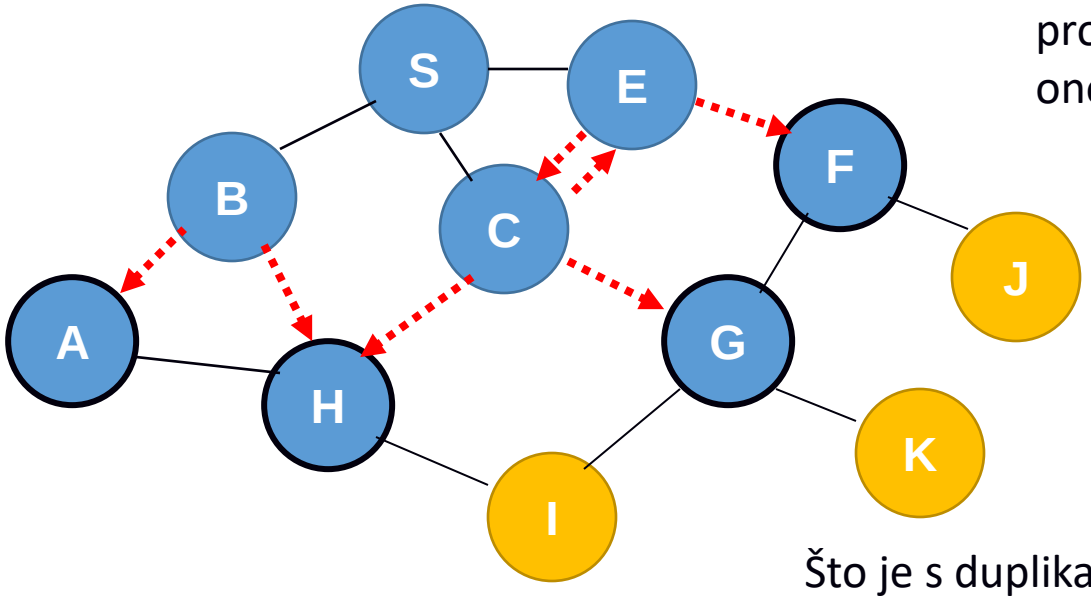


Oznaka čvora koji je izvor upita “q”.

Usmjeravanje upita: preplavljivanje



Usmjeravanje upita: preplavljivanje



Osnovno načelo:

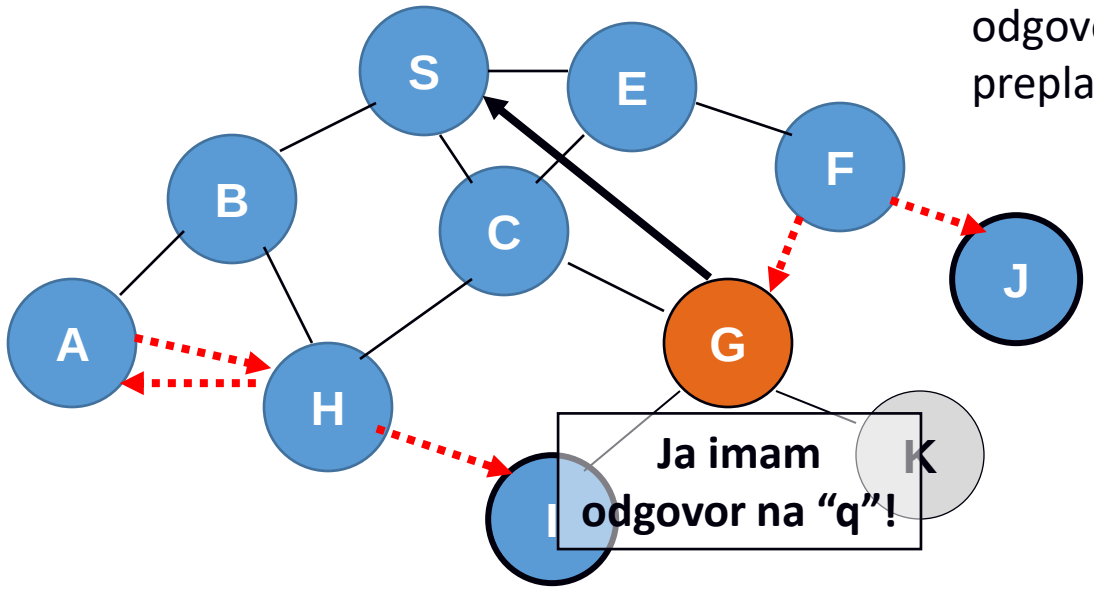
proslijedi upit svim susjedima osim onome od koga si upit primio

Što je s duplikatima?

Npr. H je primio upit od B i C. Ako upit ima jedinstveni identifikator, H uočava duplikat i ignorira ga.

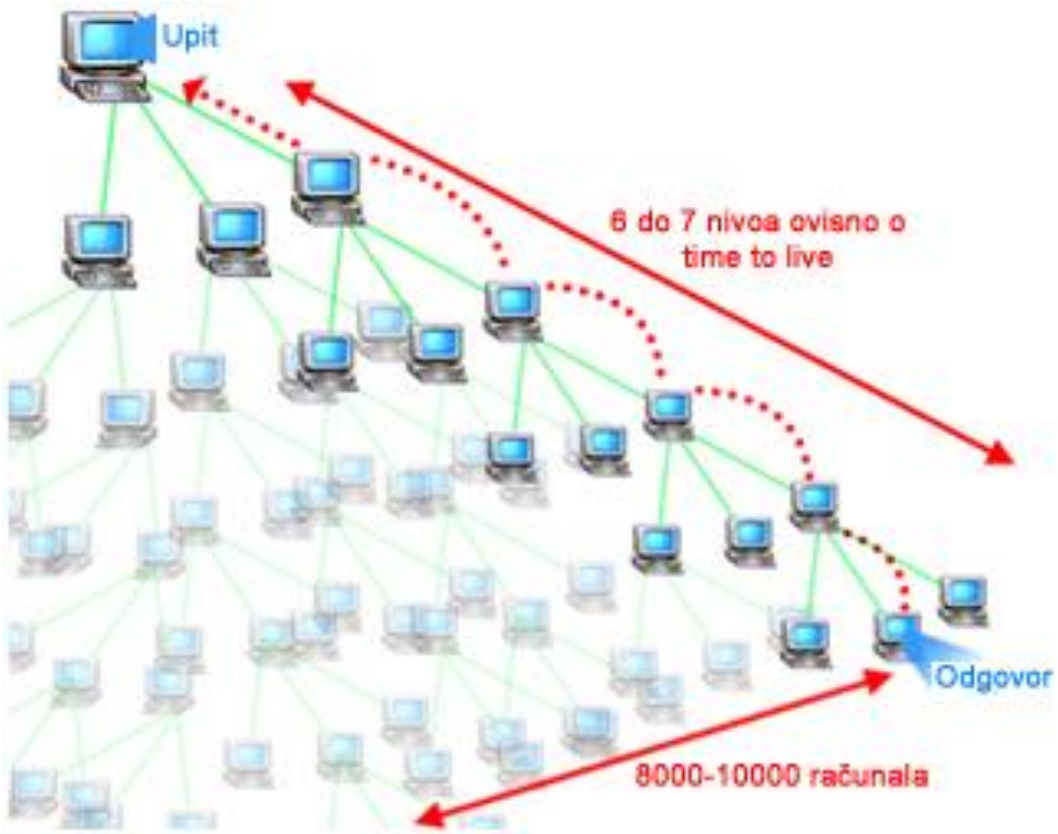
Usmjeravanje upita: preplavljivanje

G šalje odgovor direktno do S, no A, F i H ne znaju da je čvor G poslao odgovor na *q* i nastavljaju s preplavljivanjem svojih susjeda!



Usmjeravanje upita: slučajni izbor

- upit se prosljeđuje odabranom podskupu susjeda
- elementi podskupa odabiru se na slučajan način



Obilježja nestrukturiranih sustava P2P

- jednostavnost
  - jednostavan protokol za pronalaženje podataka
- robustnost
  - ne postoji jedna točka ispada
- niska cijena objavljivanja novog podatka
  - podatak ostaje pohranjen na peeru koji ga objavljuje
- velika cijena prilikom pretraživanja
  - generira se veliki mrežni promet
  - neskalabilno rješenje, komunikacijska složenost je $O(n^2)$, n je broj peerova
- dobro rješenje za pronalaženje podataka koji su replicirani na velikom broju *peerova*, ali ne za podatke pohranjene na malome broju *peerova*

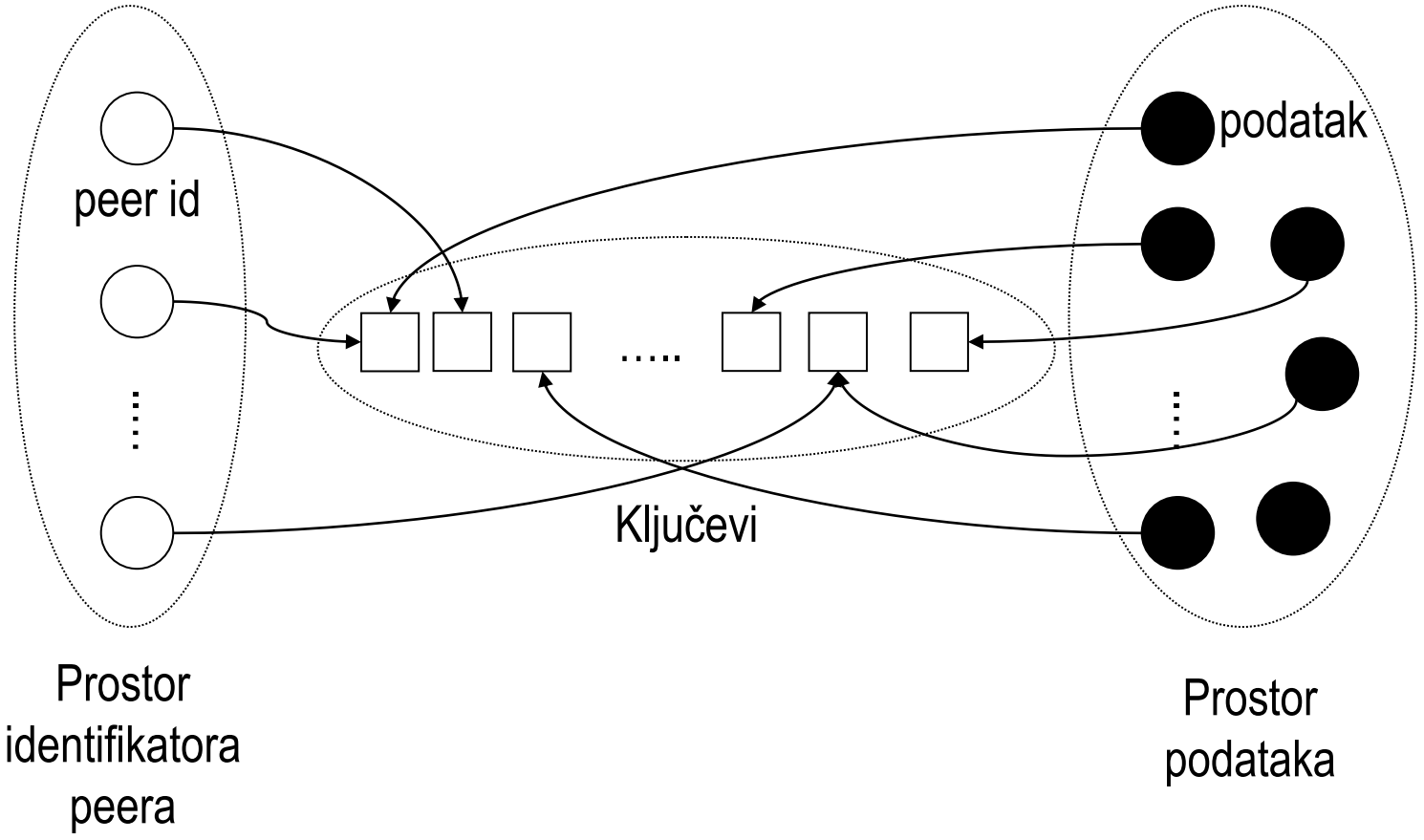
Sadržaj predavanja

- Centralizirani i decentralizirani raspodijeljeni sustavi
- Definicija sustava P2P
- Nestrukturirani sustavi P2P
- Strukturirani sustavi P2P
- Primjeri sustava P2P

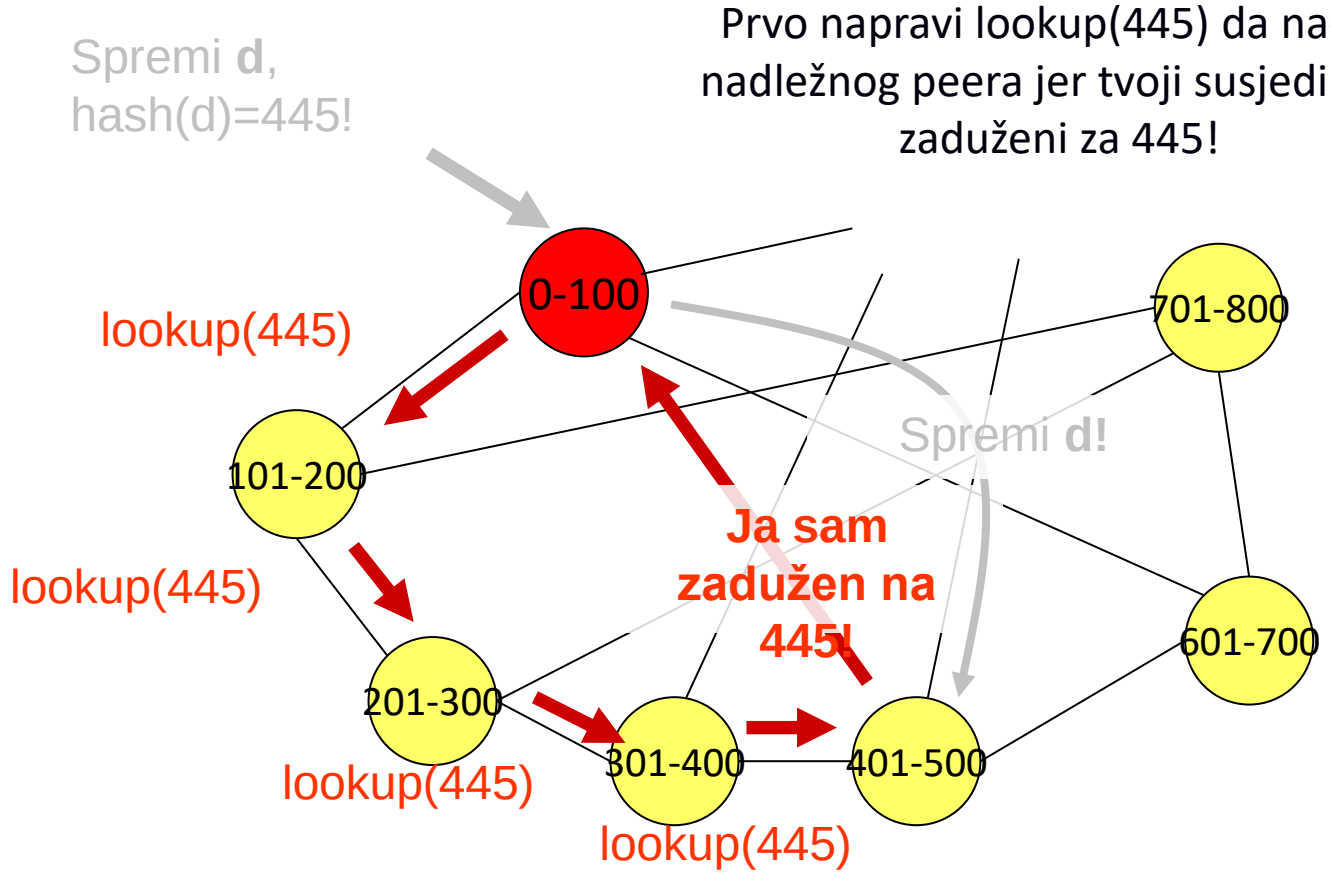
Strukturirani sustavi P2P

- za podatak ***d*** svaki peer može izračunati ključ ***k***
  - npr. ***k*** = hash(***d***), gdje je hash() hash funkcija
- za dani ključ ***k*** pronaći *peera* ***p*** koji je zadužen za prostor ključeva u koji spada ***k***
  - mreža peerova implementira metodu `lookup`(***k***) koja vraća identifikator *peera* za dani ključ ***k***
  - metoda `lookup`(***k***) je implementirana distribuirano, ako peer ne zna odgovor na upit, zna ga usmjeriti prema peeru s odgovorom
- za pohranjivanje podatka pronalazimo nadležnog peera i prosljeđujemo mu podatak
- prilikom pretraživanja pronalazimo nadležnog peera i prosljeđujemo mu upit koji opet sadrži podatak koji tražimo

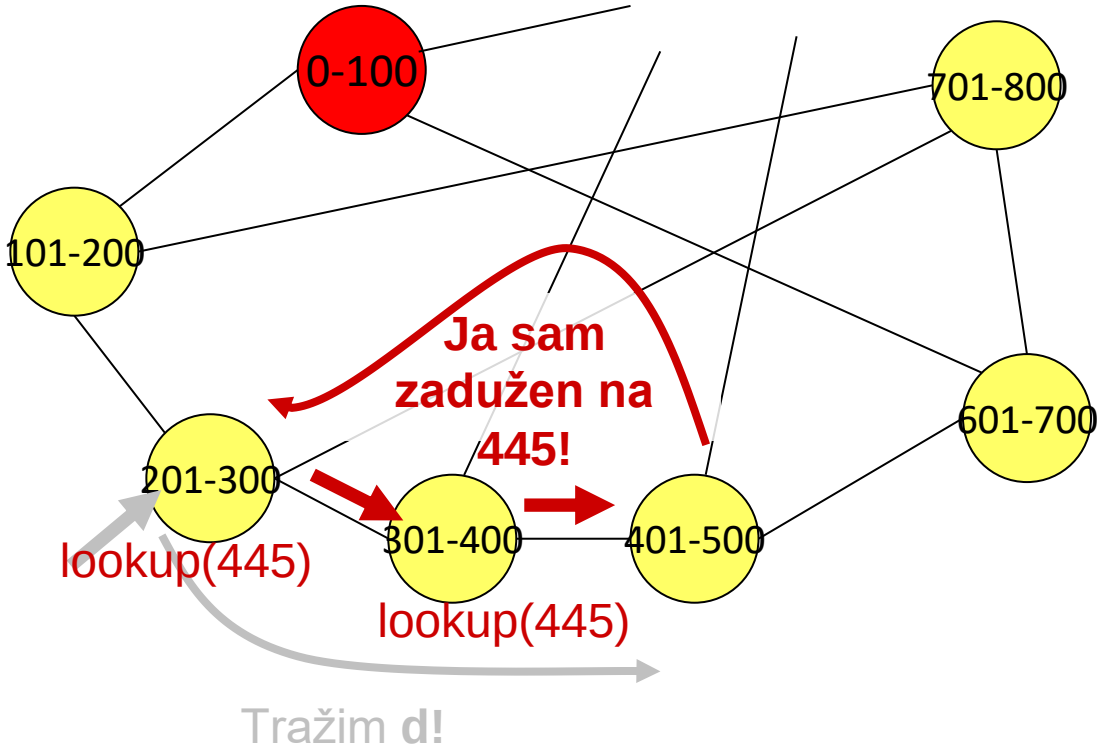
Odnos između peera, podatka i ključa



Ideja usmjeravanja: pohranjivanje podatka



Ideja usmjeravanja: upit



Osobine strukturiranih sustava P2P

- garantira pohranjivanje i pronalaženje podatka u O(log *n*) koraka (*n* je broj *peerova* u mreži)
  - skalabilno rješenje u smislu generiranog prometa u odnosu na nestrukturirane sustave
  - komunikacijska složenost **O**(*n**log *n*)
- povećana cijena objavljivanja novog podatka u odnosu na nestrukturirane sustave P2P
  - podatak se pohranjuje na *peer*u koji je za njega "zadužen"
- potrebno je održavati dodatne strukture podataka (tablice usmjeravanja) radi umjeravanje upita prema *peer*ovima koji pohranjuju tražene podatke

Sadržaj predavanja

- Centralizirani i decentralizirani raspodijeljeni sustavi
- Definicija sustava P2P
- Nestrukturirani sustavi P2P
- Strukturirani sustavi P2P
- Primjeri sustava P2P

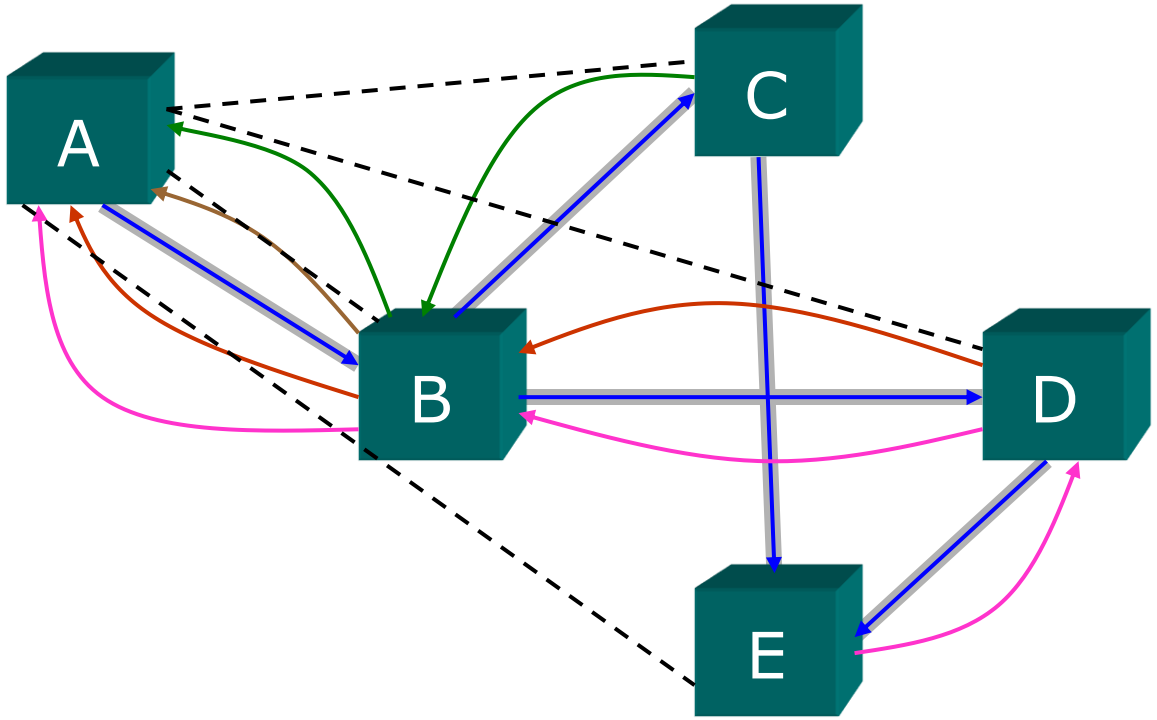
Gnutella

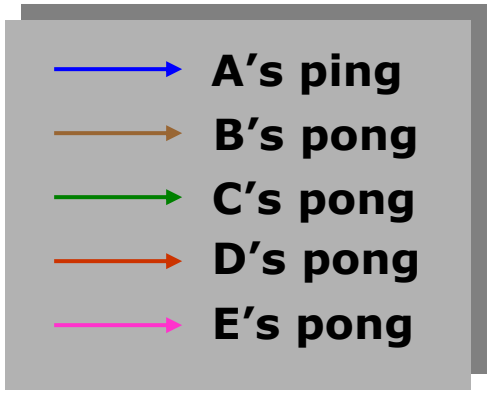
- primjer nestrukturiranog sustava P2P
- svaki peer u sustavu obavlja sljedeće funkcije:
  - pohranjuje odabrane dokumente
  - generira i usmjerava upite prema svojim susjedima
  - odgovara na upite ako pohranjuje dokument koji odgovara upitu
- koristi ograničeno preplavljivanje prilikom pretraživanja
  - svaki čvor šalje upit svim svojim susjedima
  - širenje upita je ograničeno parametrom *time-to-live* (TTL = 7)
  - svaki upit ima jedinstveni identifikator zbog petlji u mreži
- novi čvor se jednostavno povezuje u sustav tako da se spoji na barem jedan poznati Gnutella čvor

Gnutella: vrste poruka

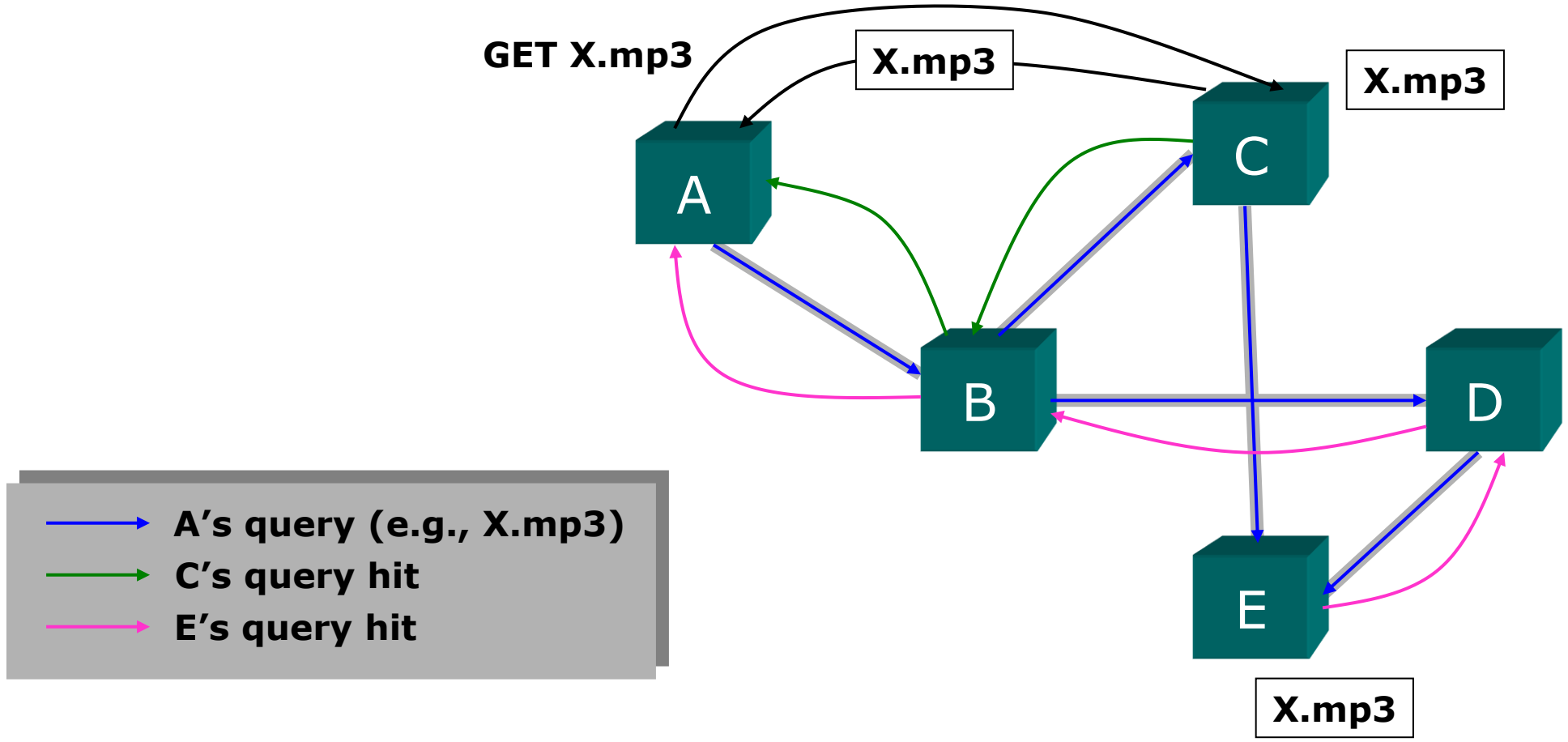
Type	Description	Contained Information
Ping	Announce availability and probe for other servents	None
Pong	Response to a ping	IP address and port# of responding servent; number and total kb of files shared
Query	Search request	Minimum network bandwidth of responding servent; search criteria
QueryHit	Returned by servents that have the requested file	IP address, port# and network bandwidth of responding servent; number of results and result set
Push	File download requests for servents behind a firewall	Servent identifier; index of requested file; IP address and port to send file to

Gnutella: održavanje mrežne topologije (Ping/Pong)

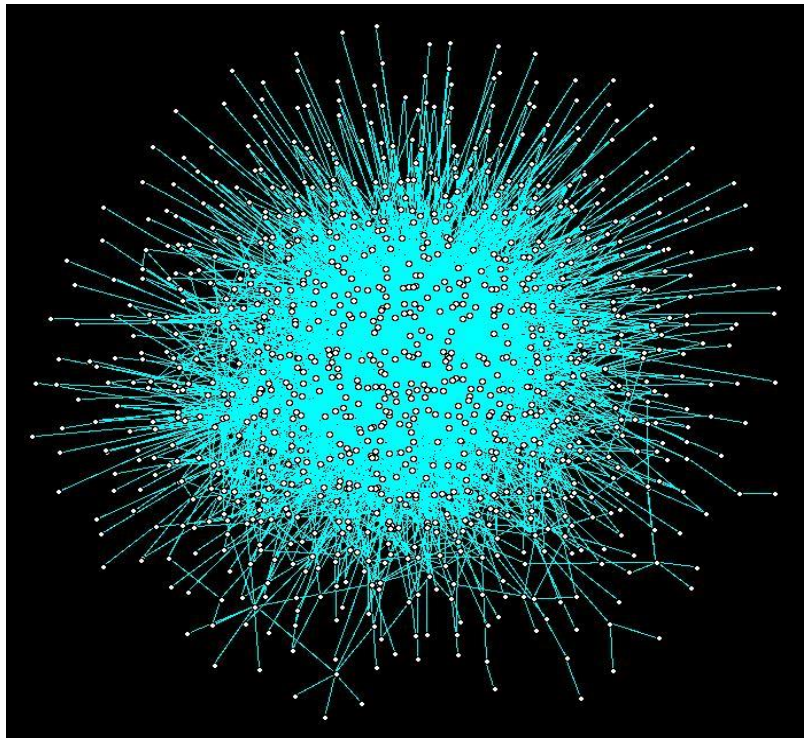


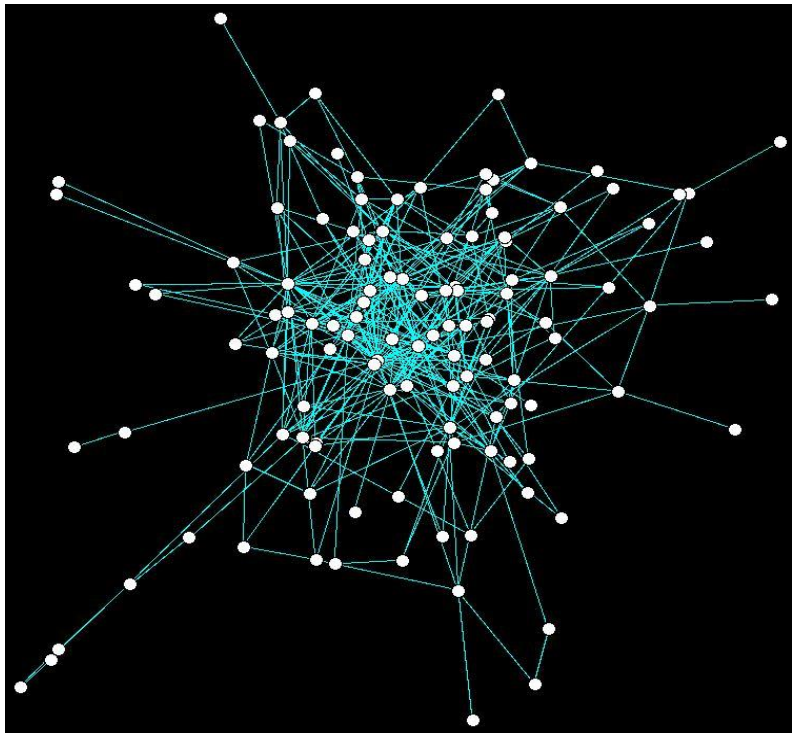


Gnutella: pretraživanje (Query/QueryHit/GET)



Topologija mreže Gnutella

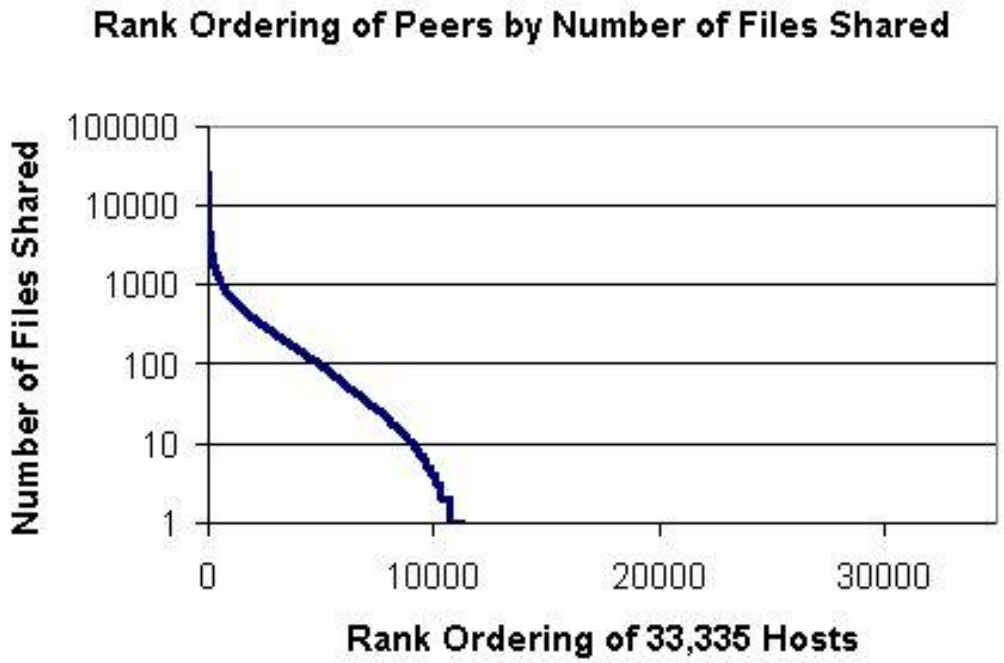




Jezgrena mreža

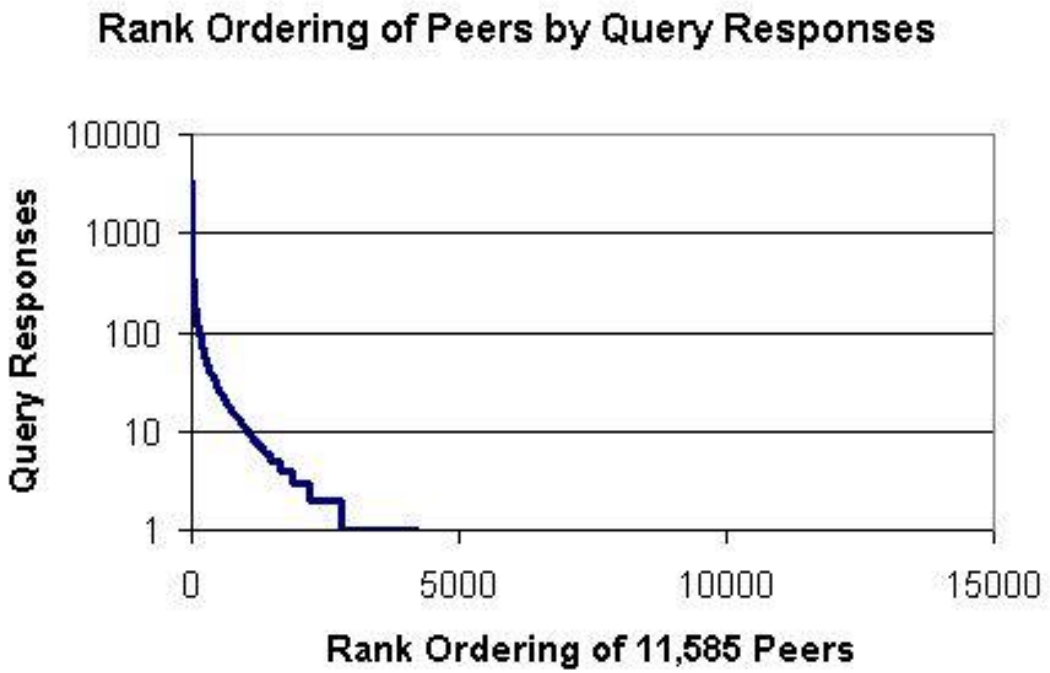
Gnutella: Free-riding (1/2)

- Veliki postotak korisnika su “*free riders*”
  - 66% peerova ne nudi vlastite datoteke
  - 73% peerova nude 10 ili manje datoteka
  - 1% peerova nude 37% svih datoteka
  - 10% peerova nude 87% svih datoteka



Gnutella: Free-riding (2/2)

- Veliki broj peerova nudi datoteke koje nikoga ne zanimaju
- od 11,585 peerova koji nude datoteke:
  - 1% peerova odgovara na 47% svih upita
  - 25% peerova odgovara na 98% svih upita
  - 63% peerova nikada ne odgovaraju na upite



Strategije pretraživanja

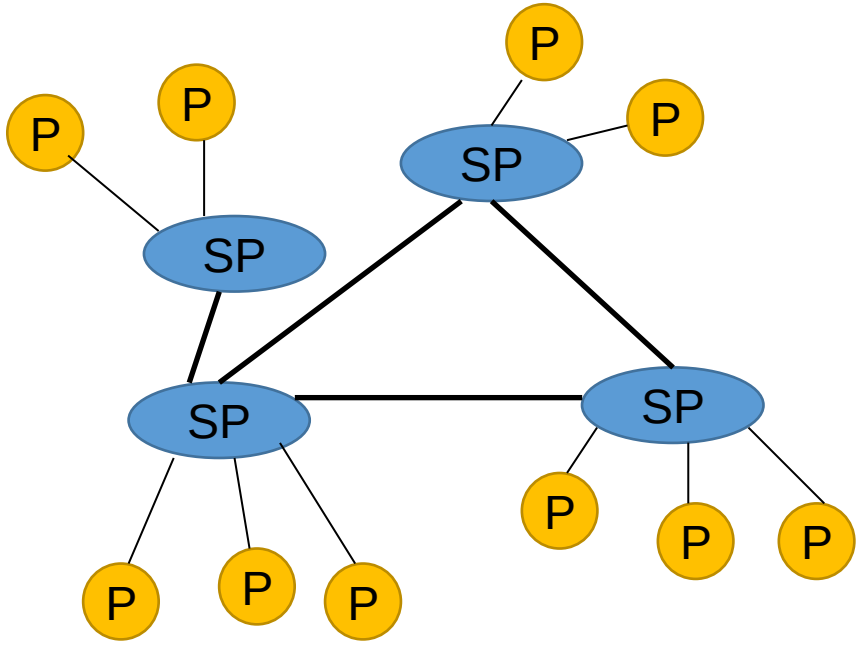
- Preplavljivanje
  - Čvor koji ne može odgovoriti na upit prvo provjerava kod izvorišnog čvora treba li proslijediti upit dalje, izvorišni čvor postaje preopterećen
  - Ograničenje udaljenosti od izvorišnog čvora (TTL)
  - Strategija širećeg prstena: TTL je inicijalno mali, a poveća se samo u slučaju da prethodni upit ne vrati rezultat (dobra strategija)
- Slučajna šetnja
  - upit se usmjerava jednom slučajno odabranom čvoru (značajno kašnjenje)
  - pokrene se *k* paralelnih šetnji (manje kašnjenje), no svaki “upit-šetač” treba periodički provjeravati je li upit zadovoljen (ako je šetnja se prekida)

Svojstva Gnutelle

- Robusna arhitektura
  - jednostavno održavanje mreže, jednostavno objavljivanje datoteka
  - otporna na značajne promjene mrežne topologije (*high churn*)
- Pogodno rješenje ako su datoteke često replicirane u mreži
- Neskalabilno rješenje u smislu generiranog prometa prilikom pretraživanja (strategija *k* paralelnih šetnji je bolja u smislu skalabilnosti od preplavljivanja)
- Ne garantira pronalaženje datoteke

Gnutella v6

- Posljednja verzija, uvodi posebne čvorove - *superpeer* (hijerahijska organizacija mreže)
  - *superpeer* je peer s dobrim resursima, na njega se spaja skup *peer*ova
  - *peer* na *superpeer*ovima sprema kopije vlastitih dokumenata te preko superpeerova vrši pretraživanje
  - upit se od *peer*a šalje *superpeer*u koji prvo provjerava svoje lokalne podatke pa ako nema odgovor, preplavljuje upitom svoje susjedne *superpeer*ove.

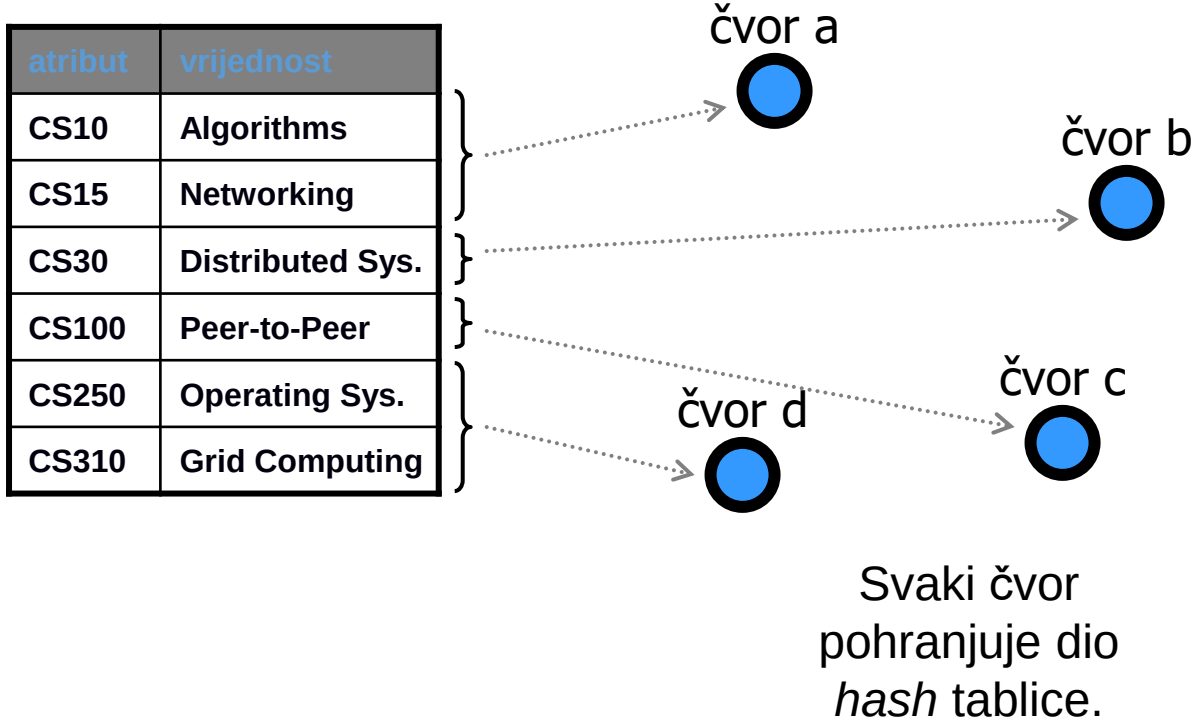


Chord

Izvor: Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. SIGCOMM Comput. Commun. Rev. 31, 4 (August 2001), 149–160. DOI: https://doi.org/10.1145/964723.383071

Primjer strukturiranog sustava P2P

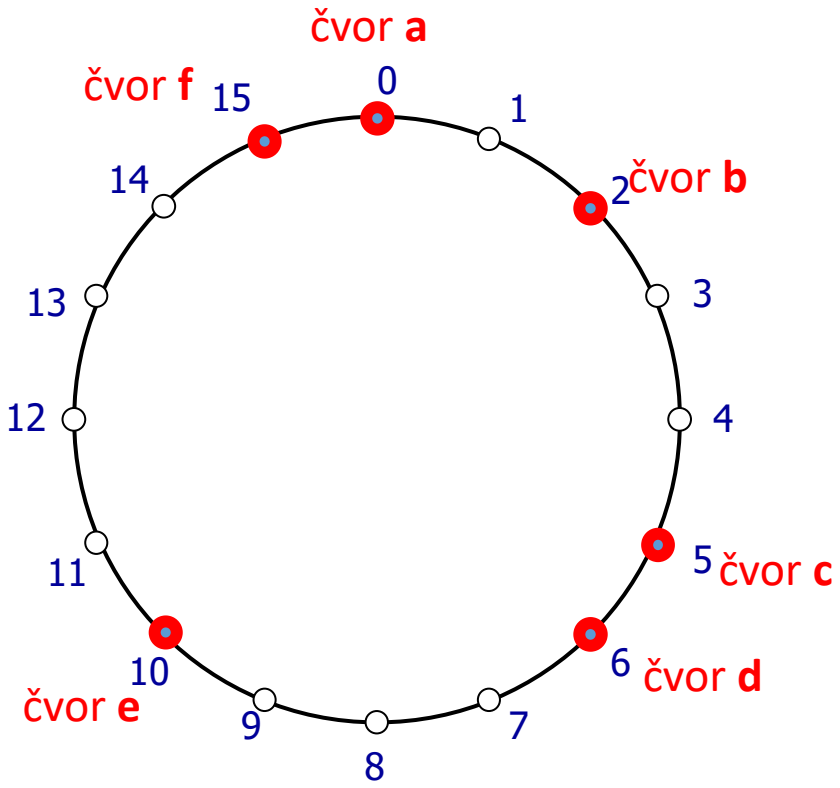
- koristi ideju raspodijeljene hash tablice - Distributed Hash Table (DHT)
- hash tablica je raspodijeljena na više čvorova.



Chord: organizacija prstena

- Koristi jednodimenzionalni prostor ključeva veličine 2^m
  - povezuje peerove i podatke iz hash tablice (atribut, vrijednost), vrijednost može biti bilo što, npr. podatak, dokument, objekt
  - prikazuje se kao prsten veličine $N = 2^m$
- Implementira samo jednu operaciju `lookup(`*k*`)` koja vraća identifikator peera nadležnog za *k*
- 2 hash funkcije: H_1(peer_ID) i $H_2(\text{atribut}_x)$, peer_ID je IP adresa peera
- Pretpostavka: *m* je dovoljno velik da postoji vrlo mala vjerojatnost kolizije za hash funkcije
  - Kolizija *hash* funkcije: proizvodi isti *hash* kod za različite parametre

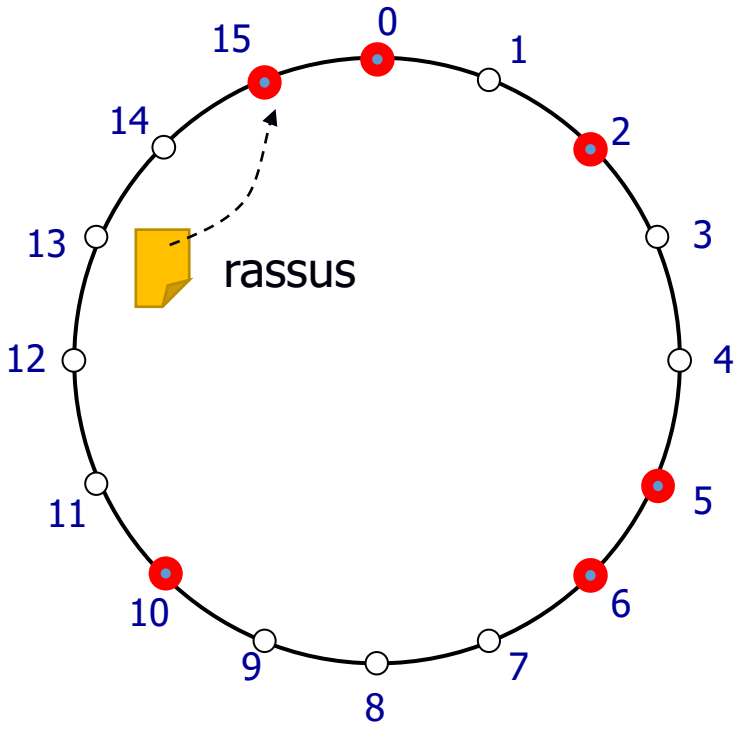
Chord: Kako složiti čvorove u prsten?



Mogući identifikatori čvorova:
{0,1, …, 15} , N = 16

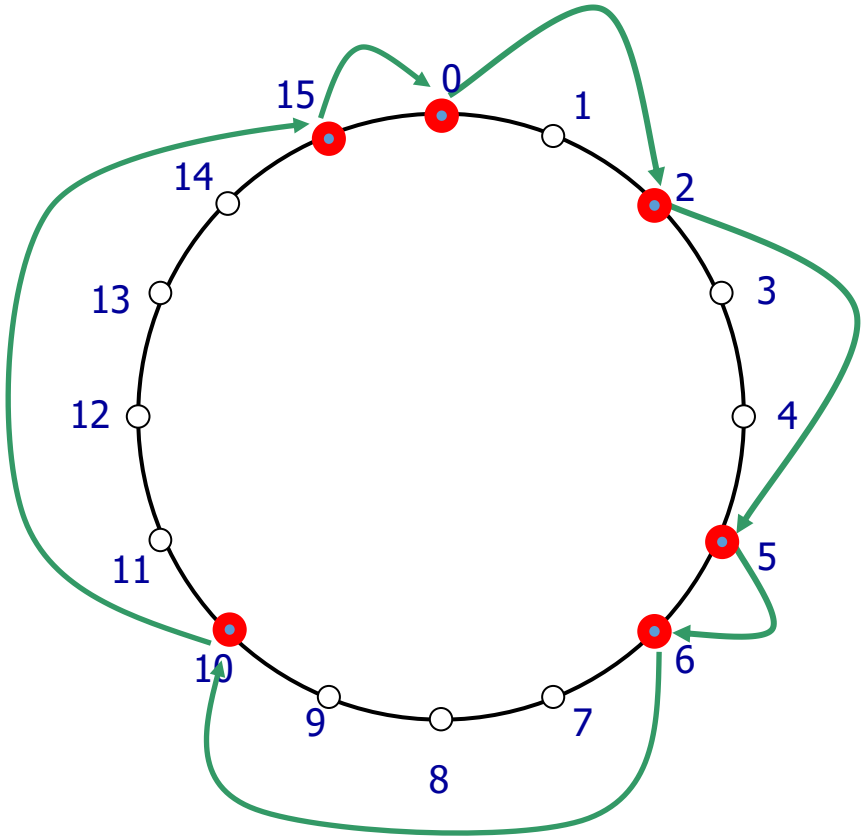
- Primjer mreže sa 6 čvorova {**a,b,c,d,e,f**}
- Prsten veličine $N = 16$ ($m = 4$), ovo je broj mogućih ključeva
- Čvorovima se jednoznačno pridjeljuju ključevi uz pomoć posebne funkcije H_1, npr. $H_1(a) = 0$.

Chord: Kako se podatak dodjeljuje čvoru?



- Podacima se pridjeljuju ključevi iz istog prostora {0,1, …, 15} koristeći funkciju H_2
- **(rassus, http://www.fer.hr/predmet/rassus)** dobiva ključ 13 jer vrijedi H_2(”**rassus**”) = 13
- Kako u mreži ne postoji čvor s ključem 13, podatak se pohranjuje na prvom sljedećem čvoru (to je u ovom slučaju čvor f kojemu je ključ = 15)

Chord: Kako ćemo povezati čvorove?



- Svaki čvor održava jedan pokazivač na sljedbenika, tj. na prvi sljedeći čvor na prstenu u smjeru kazaljke na satu

sljedbenik čvora 0 $\rightarrow$ čvor 2

sljedbenik čvora $2 \rightarrow$ čvor 5

sljedbenik čvora $5 \rightarrow$ čvor 6

...

Učinkovito pretraživanje?

Chord: Kako ubrzati pretraživanje?

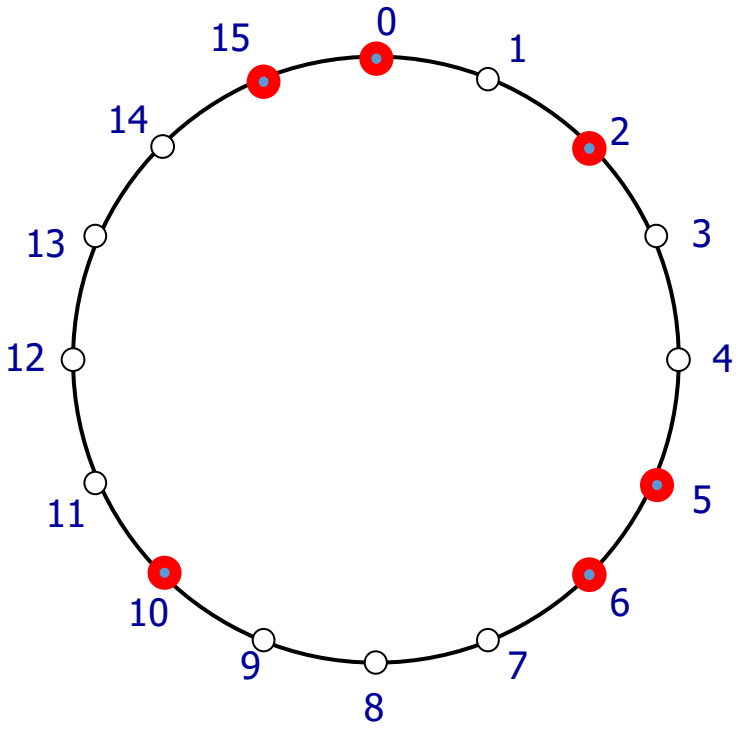
Primjer: tablica usmjeravanja za čvor s ključem 15

ključ + 2^i, $i = 0, \ldots, m\text{-}1$ ($N = 2^m$)

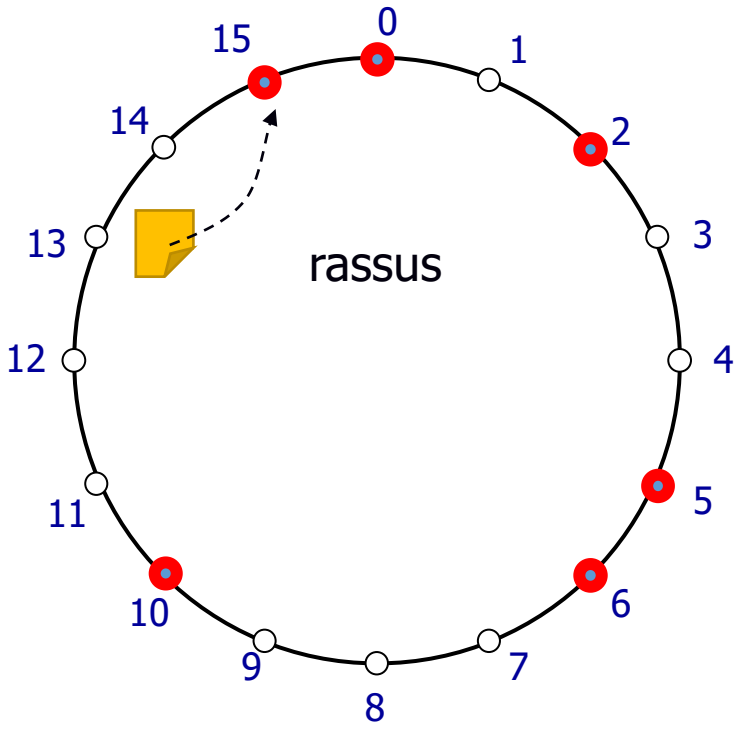
$15+2^0 =$	0	→ čvor 0
$15+2^1 =$	1	→ **čvor 2**
$15+2^2 =$	3	→ **čvor 5**
$15+2^3 =$	7	→ **čvor 10**

Za čvorove koji ne postoje, pokazivač se postavlja na prvog sljedbenika!

Koliko je zapisa u tablici usmjeravanja svakog čvora?



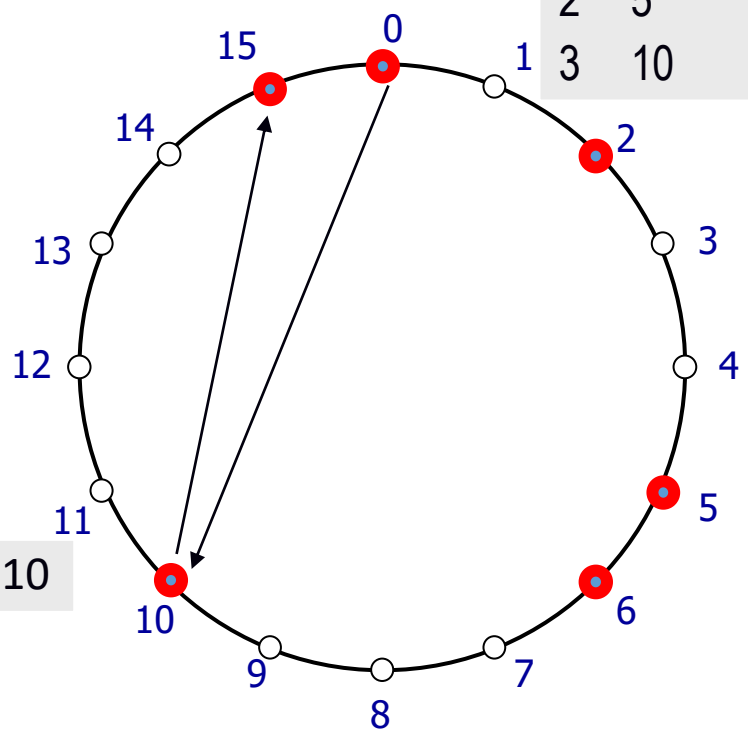
Chord: Kako pronaći podatak? (1)



- **lookup(13) ?**
- jer svaki peer može izračunati H_2(”**rassus**”) = 13
- vraća IP adresu peera zaduženog za ”**rassus**”, kontaktiramo peera direktno da dođemo do traženog URL-a

Chord: Kako pronaći podatak? (2)

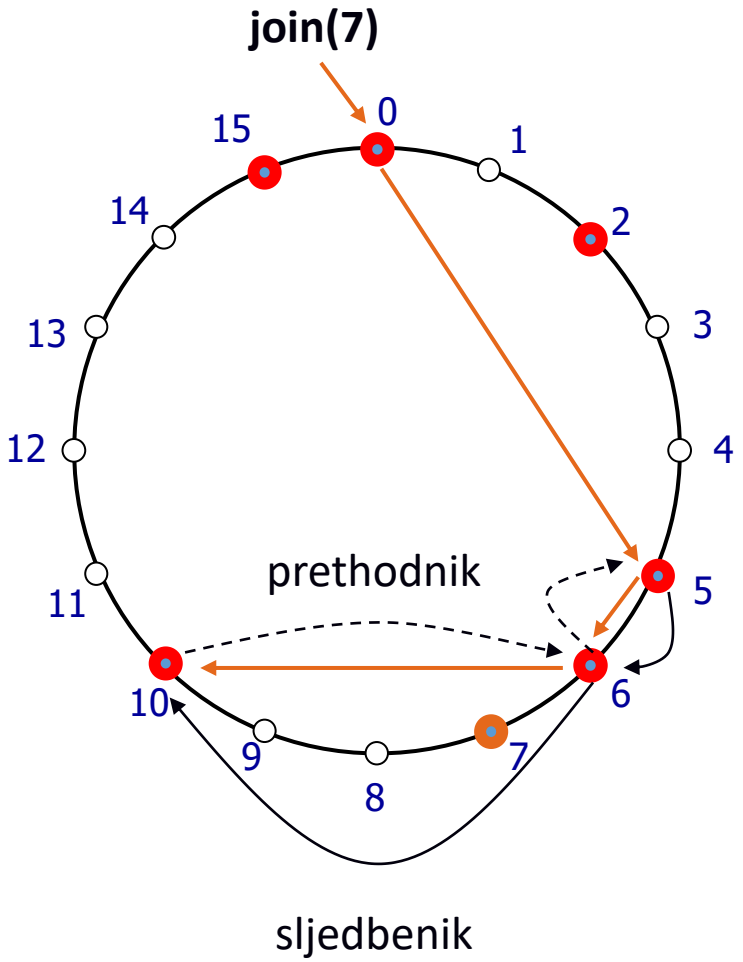
i	Tablica ***finger*** za p0
0	2
1	2
2	5
3	10



i	Tablica ***finger*** za p10
0	15
1	15
2	15
3	2

m zapisa u tablici *finger* (tablica usmjeravanja)

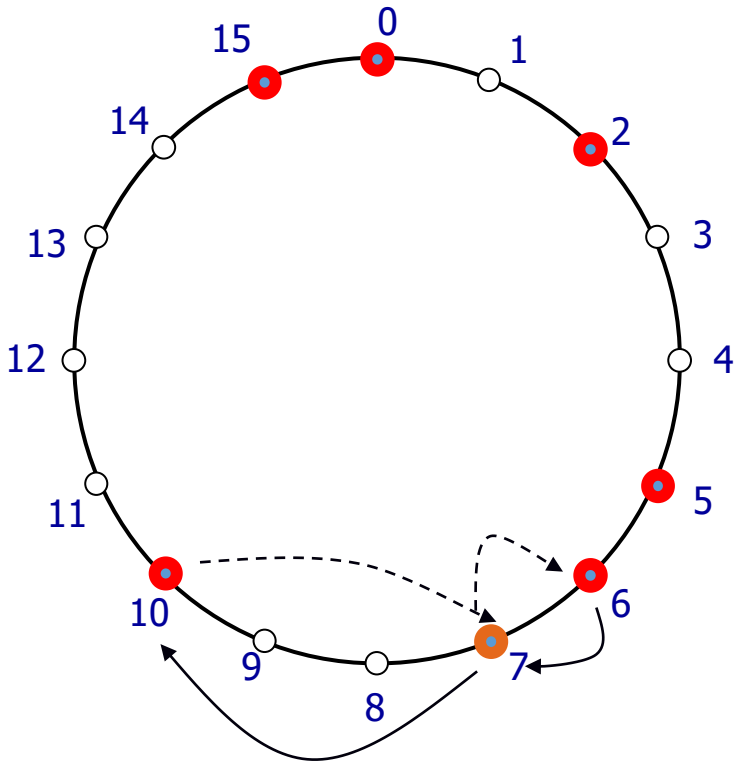
Chord: dodavanje peera (1)



- Dodatak u tablici usmjeravanja, svaki čvor ima pokazivač na prethodnika

1. korak: Čvor p7 pronalazi jednog peera iz mreže i koristeći lookup(7) pronalazi svog sljedbenika

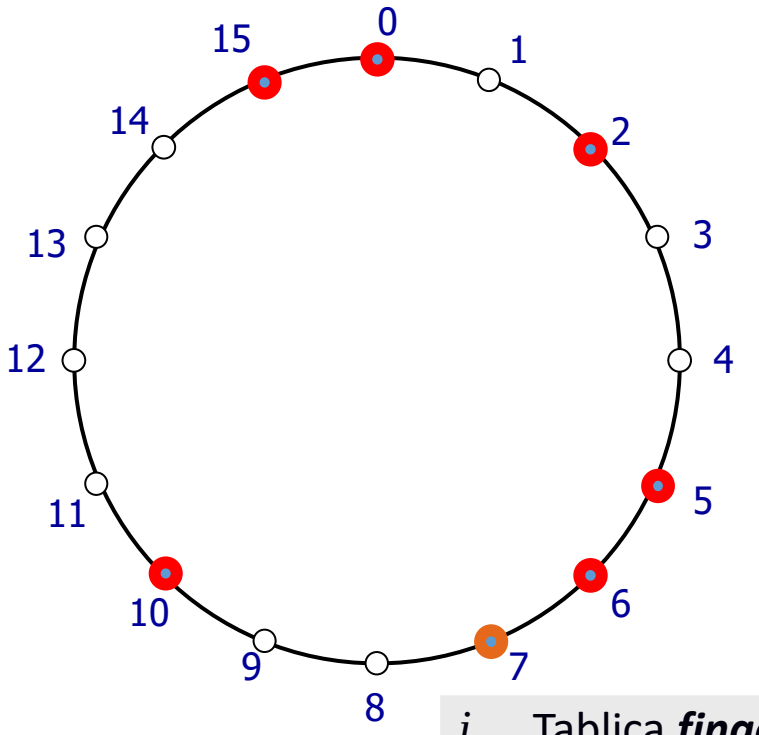
Chord: dodavanje peera (2)



2. korak: Čvor p7 podešava pokazivač na sljedbenika (p10) te od njega saznaje svog prethodnika (p6)

3. korak: Čvor p10 mijenja prethodnika (p7), a čvor p6 sljedbenika

Chord: dodavanje peera (3)

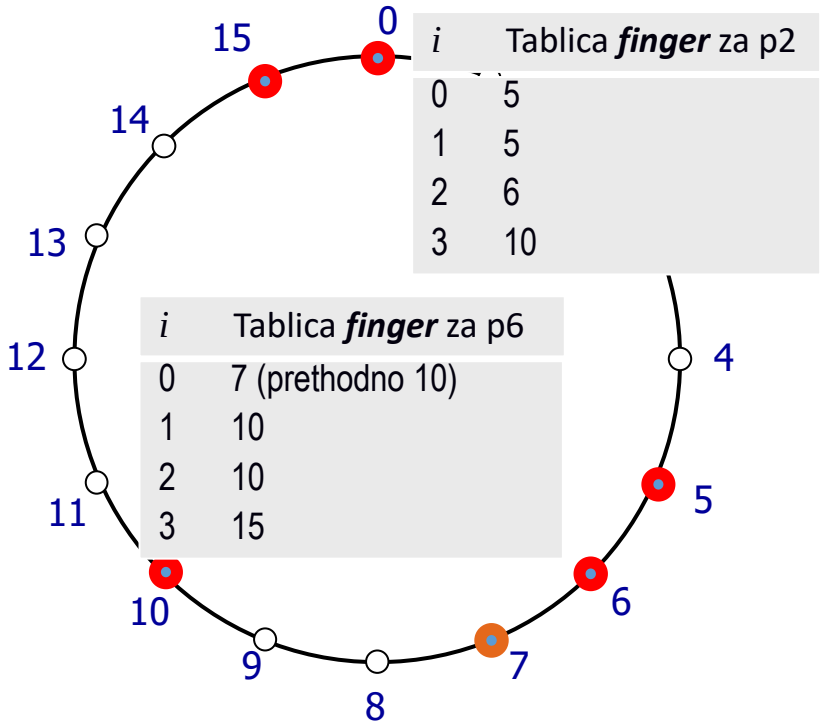


4. korak: Podesi tablicu *finger* na p7, za to p7 koristi svog prethodnika (p6) koji izvodi sljedeće operacije:

i	Tablica ***finger*** za p7
0	10
1	10
2	15
3	15

i	
0	lookup(7+1)
1	lookup(7+2)
2	lookup(7+4)
3	lookup(7+8)

Chord: dodavanje peera (4)



i	Tablica ***finger*** za p2
0	5
1	5
2	6
3	10

i	Tablica ***finger*** za p6
0	7 (prethodno 10)
1	10
2	10
3	15

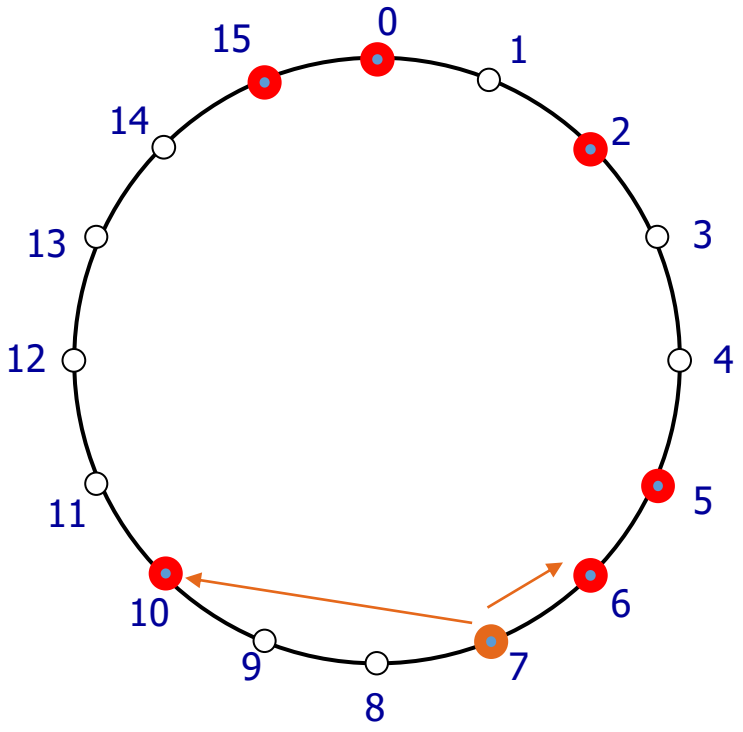
- Za i=0, lookup(7-1) vraća 6, u tablici na p6 mijenja se prvi redak u i=0, 7
- Za i=2, lookup(7-4) vraća 5 što je veće od 3 pa se vraćamo na prethodnika od p5, a to je p2; u tablici na p2 se treći redak (i=2) ne mijenja jer je trenutno 6 što je manje od 7

5. korak: Popravi tablice *finger* na sljedećim peerovima u mreži: $p7-2^i$ (ako postoje) ili na prethodniku tog čvora

Algoritam.

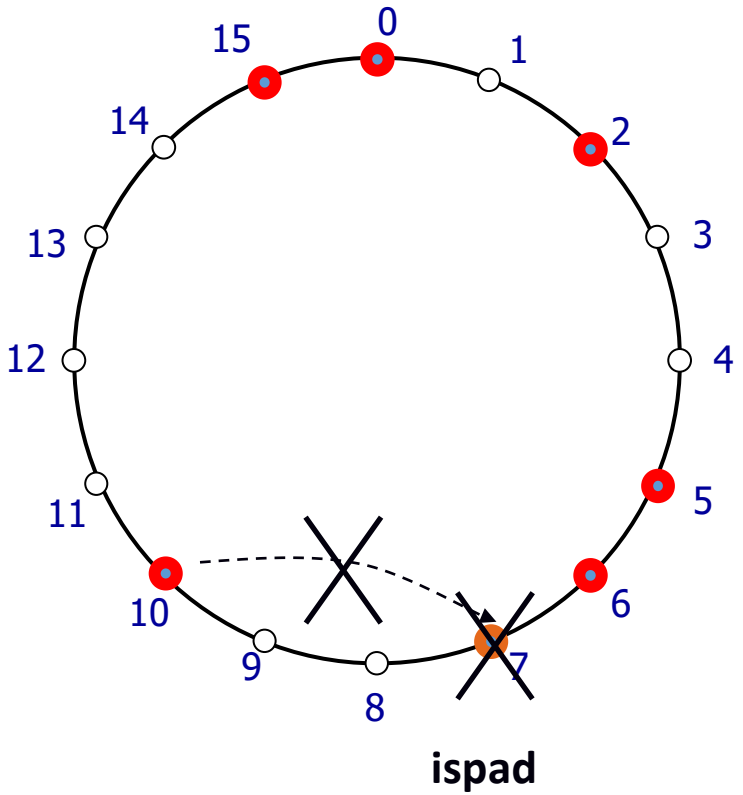
```
za i=0 do 3

 lookup (p7-2^i);

 ako si došao do čvora većeg od p7-2^i
vrati se do njegovog prethodnika (peer t)
inače peer t = p7-2^i;

 ako je element iz i-tog retka peera t
veći od p7, promijeni i-ti redak u tablici
usmjeravanja na 7
```

Chord: izlazak peera



1. korak: p7 obavještava svog sljedbenika i prethodnika o napuštanju mreže
2. korak: p7 šalje zahtjev ostalim peerovima u mreži da poprave tablice *finger* (slično postupku dodavanja čvora)

Chord: ispad peera



- Svaki čvor periodički kontaktira prethodnika i primjećuje njegov ispad
- Taj čvor može popraviti pokazivač na prethodnika samo ako ga drugi čvor dodaje kao svoga sljedbenik-a
- Svaki čvor održava listu sljedbenika, a ne samo jednog, ako prvi sljedbenik ne odgovara, čvor može kontaktirati sljedećeg na listi

Svojstva Chorda

- Ravnomjerno opterećenje čvorova
  - Za mrežu od *n* čvorova, svaki čvor je zadužen za najviše $(1+\varepsilon)N/n$ ključeva, gdje je *N* veličina adresnog prostora ključeva
- Skalabilnost algoritma *lookup*
  - kontaktira $O(\log_2 n)$ čvorova
- Skalabilna veličina tablice *finger* : *m*
- Nužno je određeno vrijeme stabilizacije za slučaj velikih promjena u mreži radi održavanja prstenaste strukture

Usporedba protokola Chord i Gnutella

Ulazni parametri simulacije

- Chord
  - veličina adresnog prostora = 2^{160}
  - broj sljedbenika *r* =10
  - za stabilizaciju prstena: 10s za ažuriranje tablice *finger*, 10s za stabilizaciju stanja prethodnika
- Gnutella
  - TTL za PING = 3
  - TTL za QUERY = 5

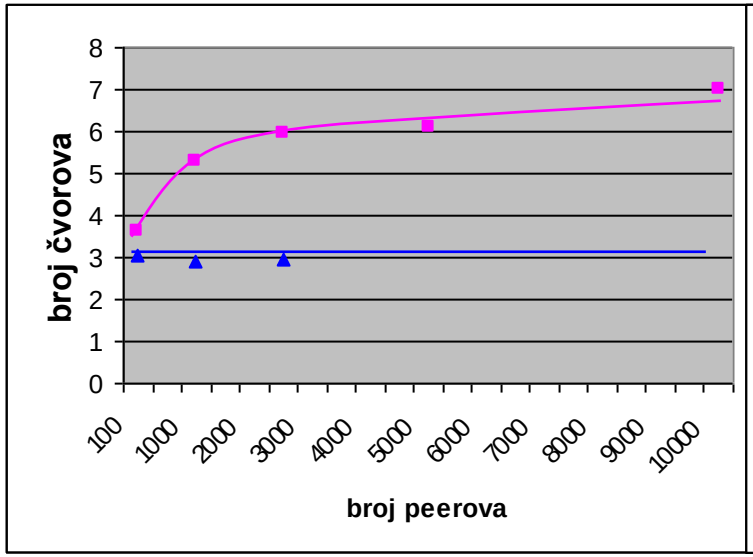
Izlazni parametri simulacije

- Prosječan broj posjećenih čvorova po upitu
- Prosječno vrijeme odziva
- Uspješnost (postotak riješenih upita)

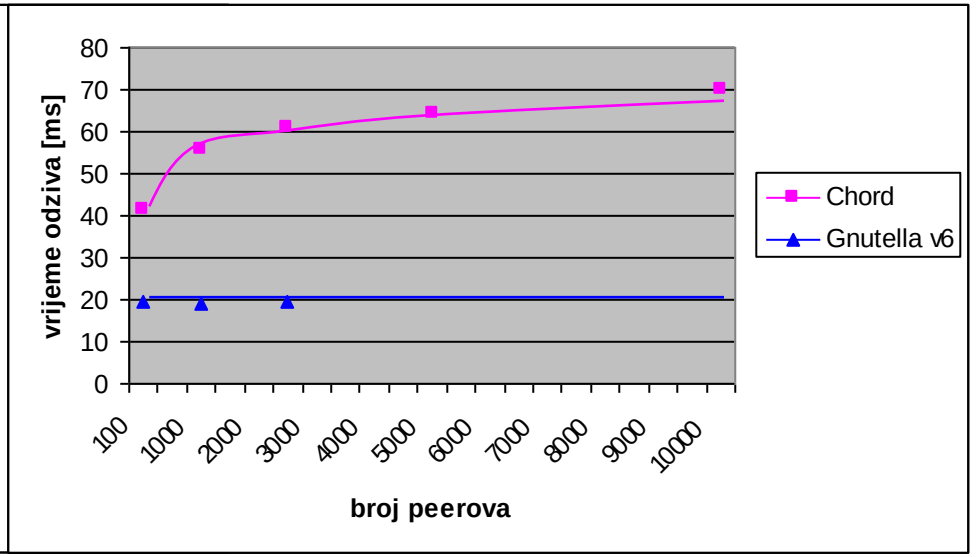
Simulator: PeerfactSim.KOM.
http://peerfact.kom.e-technik.tu-darmstadt.de/de/downloads/

Analiza skalabilnosti (1)

- Statični scenarij
  - inicijalno se formira mreža nakon koje slijedi proces stabilizacije (značajno za Chord) te potom počinje objavljivanje podataka i pretraživanje

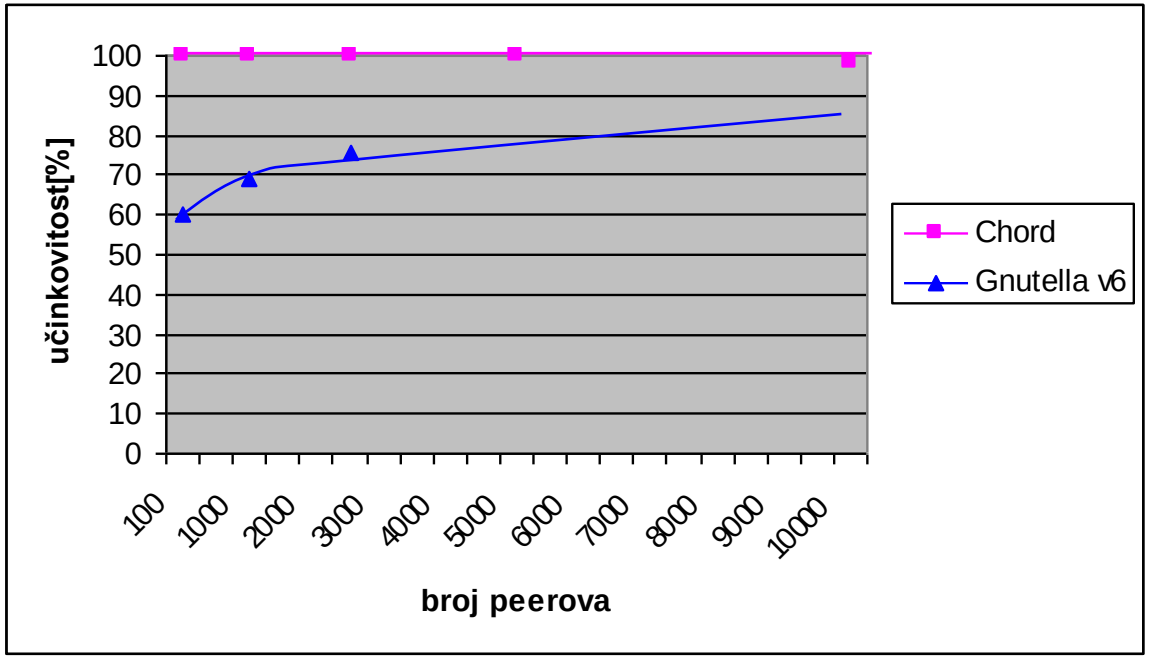


Prosječan broj čvorova po upitu



Prosječno vrijeme odziva

Analiza skalabilnosti (2)



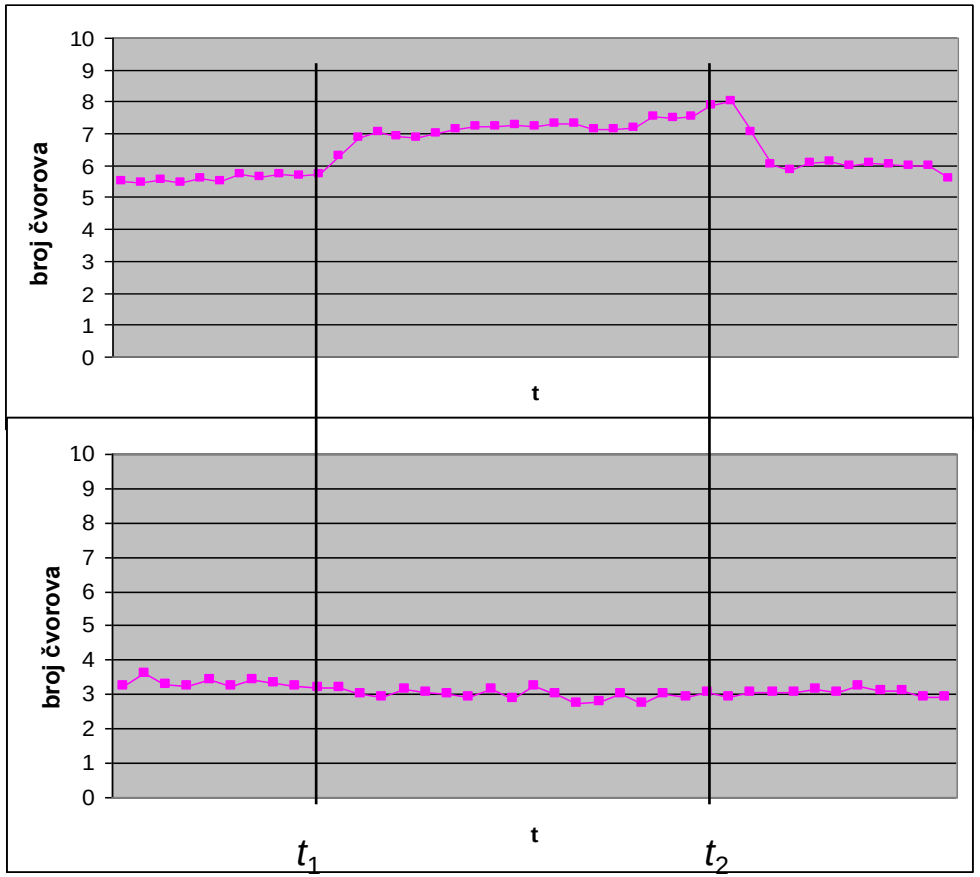
postotak riješenih upita

Analiza stabilnosti (1)

- Scenarij *churn*
  - u mreži je inicijalno 1000 čvorova, u t_1 se dodatno spaja 2000 čvorova, a u t_2 odspaja 2000 čvorova

Chord

Gnutella

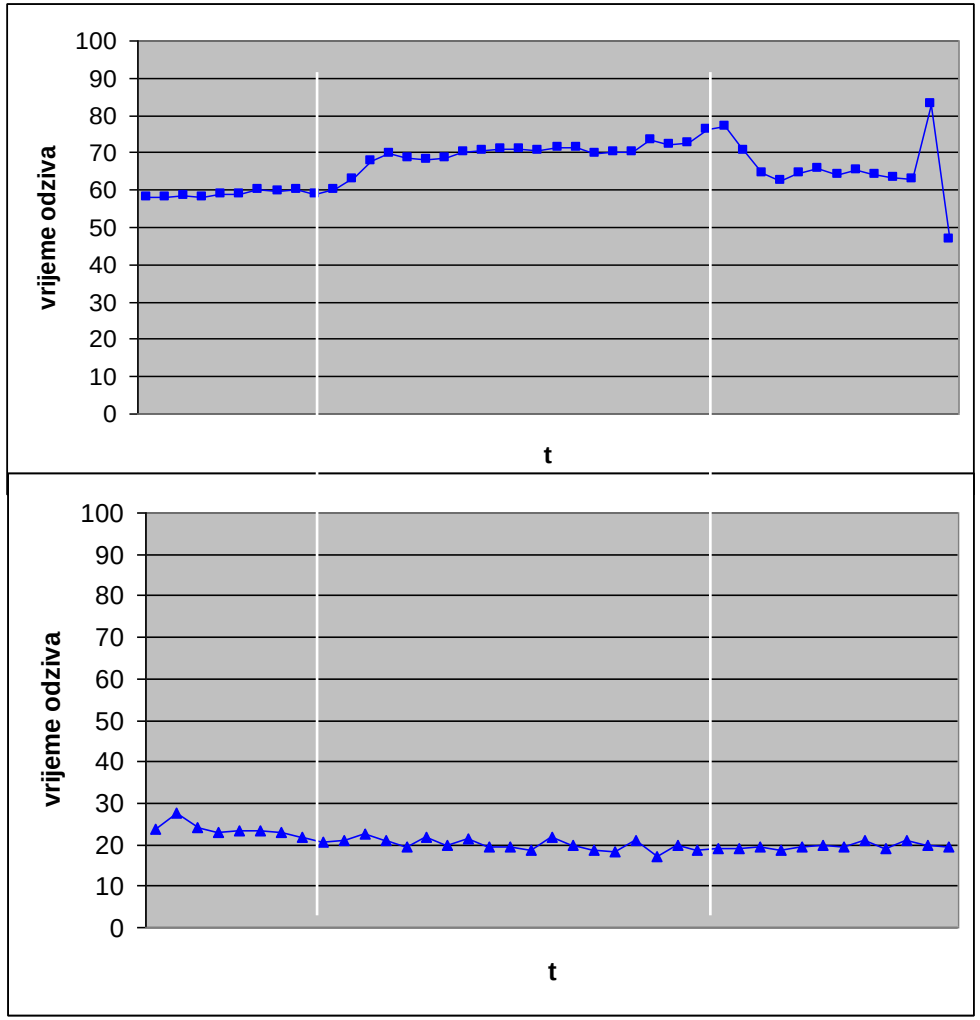


Promjena prosječnog broja čvorova po upitu u vremenu

Analiza stabilnosti (2)

Chord

Gnutella



Promjena prosječnog vremena odziva u vremenu

Zaključak

- Chord bilježi gotovo 100% uspješnost pri rješavanju upita
- Gnutella ima nižu učinkovitost koja raste za veće mreže zbog bolje povezanosti čvorova (povećana vjerojatnost pronalaska dokumenta)
- Prosječan broj čvorova po upitu te vrijeme odziva su konstantni za Gnutellu za povećani broj čvorova (TTL je konstantan), dok za Chord bilježimo logaritamski porast
- Chord je izrazito nestabilan prilikom scenarija churn zbog procesa stabilizacije koji nije završen (uspješnost je smanjena na 55%) dok je Gnutella neosjetljiva na churn (osim što je nešto smanjena uspješnost)

Literatura

- G. Coulouris, J. Dollimore, T. Kindberg, G. Blair: "Distributed Systems: Concepts and Design", poglavlje 10.1-10.4
- Eng Keong Lua Crowcroft, J. Pias, M. Sharma, R. Lim, S., A survey and comparison of peer-to-peer overlay network schemes, IEEE Communications Surveys & Tutorials, 7(2), Second Quarter 2005, pp. 72- 93. http://www.cl.cam.ac.uk/teaching/2005/AdvSysTop/survey.pdf
- Karl Aberer: Peer-to-Peer Data Management. Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2011

Pitanja za učenje i ponavljanje

- Za Chordov prsten sa slike navedite tablicu usmjeravanja za čvor 5.
- Navedite korake koji će pohraniti podatak *d* za koji vrijedi H(*d*)=13 ako podatak želi pohraniti čvor čiji je ključ jednak 2. Na kome čvoru će podatak biti pohranjen?
- Navedite korake za pronalaženje podatka *d* iz prethodnog pitanja ako upit za *d* dolazi s čvora s ključem 0. Koja je cijena u smislu komunikacije među peerovima?