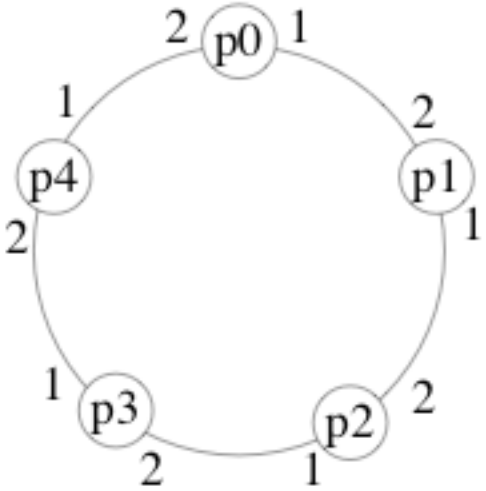
DISTRIBUIRANI ALGORITMI I SISTEMI

Iz kursa CSCE 668
Proleće 2014

Autor izvorne prezentacije:
Prof. Jennifer Welch

Prstenaste mreže

- U *orijentisanom* prstenu, procesori imaju isti pojam leve i desne strane



- Npr., ako se poruke uvek prosleđuju na kanal 1, one će kružiti po prstenu u smeru kazaljke na satu

Zašto proučavamo prstene?

- jednostavna polazna tačka, laka za analizu
- apstrakcija „token ring“ LAN mreže
- donje granice i nemogući rezultati za topologiju prstena su primenljivi na proizvoljne topologije

Definicija izbora lidera (LE = Leader Election)

- Svaki procesor ima stanja: **izabran** (pobedio) i **nije-izabran** (izgubio)
- Jednom kada uđe u stanje izabran, procesor zauvek ostaje u tom stanju (isto tako i za nije-izabran), tj. to je ireverzibilna odluka
- U svakom prihvatljivom izvršenju:
  - svaki procesor na kraju ulazi ili u stanje izabran ili nije-izabran
  - samo jedan procesor (**lider**) ulazi u stanje izabran

Upotrebe lidera

- Lider može koordinirati aktivnosti u sistemu:
  - određivanje razapinjućeg stabla koristeći lidera kao koren
  - rekonstrukcija izgubljenog tokena u „token-ring“ mreži
- Izučićemo izbor lidera u prstenima

Anonimni prsteni

- Kako modelirati slučaj kada procesori nemaju jedinstvene identifikatore?
- Prvi pokušaj: zahtevati da svi procesori imaju isti automat (state machine)
- Suptilna tačka: da li se alg. oslanja na poznavanje veličine prstena (broj procesora)?

Uniformni (Anonimni) algoritmi

- **Uniformni** algoritam ne koristi veličinu prstena (isti algoritam za sve veličine prstena)
  - Formalno, svi procesori za sve veličine prstena se modeliraju sa istim automatom
- **Neuniformni** algoritam koristi veličinu prstena (različit algoritam za svaku veličinu prstena)
  - Formalno, za svaku vrednost *n*, svi procesori u prstenu veličine *n* se modeliraju sa istim automatom A_n .
- Uočite da nema jedinstvenih identifikacija

Izbor lidera u anonimnim prstenima

- **Teorema:** Ne postoji algoritam za izbor lidera za anonimne prstene, čak i ako
  - algoritam poznaje veličinu prstena (neuniformno)
  - sinhroni model
- **Skica dokaza:**
  - Svaki procesor kreće iz istog stanja sa istim odlaznim por. (pošto je anoniman)
  - Svaki procesor prima iste por., radi iste izmene stanja, i šalje iste por. u rundi 1
  - Isto tako za runde 2, 3, …
  - Konačno, neki procesor bi trebao da uđe u stanje izabran. Ali onda bi svi ušli u to stanje.

Izbor lidera u anonimnim prstenima

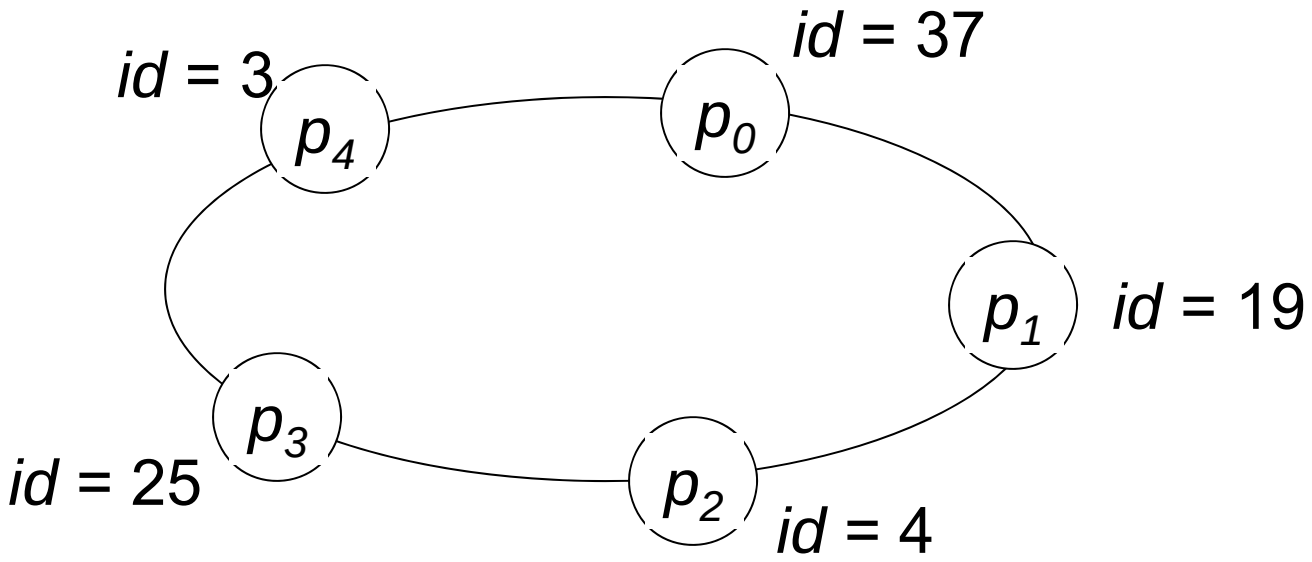
- Skica dokaza pokazue da ili je bezbednost (nikada neće biti izabrano više od 1 lidera) ili životnost (konačno će biti izabran bar 1 lider) narušena
- Pošto je teorema dokazana za neuniformne i sinhrone prstene, isti rezultat važi za slabije modle:
  - uniformni
  - asinhroni

Prsteni sa identifikatorima

- Pretpost. svaki procesor ima jedinstven id
- Ne treba brkati indekse i identifikacije:
  - **indeksi** idu 0 do *n - 1*; koriste se za analizu, oni nisu raspoloživi procesorima
  - **id-ovi** su proizvoljni nenegativni celi brojevi; oni su raspoloživi procesorima putem lokalne promenljive *id*

Specificiranje prstena

- Polazi se od najmanjeg id i daje se lista id-a po redosledu u smeru kazaljke na satu



- Primer: 3, 37, 19, 4, 25

Uniformni (neanonimni) algoritmi

- **Uniformni** algoritam: postoji jedan automat za svaki id, bez obzira na veličinu prstena
- **Neuniformni** algoritam: postoji jedan automat za svaki id i za svaku različitu veličinu prstena
- Ove definicije su prilagođene za problem izbora lidera u prstenu

Pregled LE alg. u prstenima sa identifikacijama

- Postoje algoritmi kada čvorovi imaju jedinstvene id
- Procenićemo ih prema njihovom (najgorem) *broju poruka*
- asinhroni prsten:
  - $\Theta(n \log n)$ poruka
- sinhroni prsten:
  - $\Theta(n)$ poruka pod određenim uslovima
  - inače $\Theta(n \log n)$ poruka
- Sve granice su asimptotski uske

Zagrevanje: Asinhroni LE algoritam sa $O(n^2)$ poruka

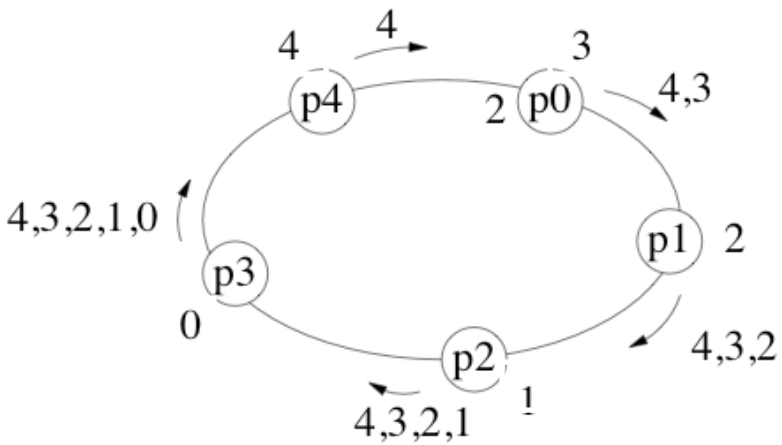
- pošalji vrednost svog *id* u levo
- kad se primi id *j* (sa desne strane):
  - if $j > id$ then
    - prosledi *j* u levo (ovaj procesor je izgubio)
  - if $j = id$ then
    - izaberi sebe (ovaj procesor je pobedio)
  - if $j < id$ then
    - ne radi ništa

Analiza $O(n^2)$ algoritma

- **Korektnost:** Izabira procesor sa najvećim id
  - por sa najvećim id prolazi kroz sve procesore
- **Vreme:** $O(n)$
- **Broj poruka:** Zavisi kako su id-ovi uređeni
  - najveći id putuje skroz oko prstena (n por.)
  - 2-gi najveći id putuje dok ne dođe do najvećeg
  - 3-ći najveći id putuje dok ne dođe do najvećeg ili drugog najvećeg
  - itd.

Analiza $O(n^2)$ algoritma

- Najgore uređenje id-a je u opadajućem redosledu:
  - 2-gi najveći izaziva $n - 1$ poruka
  - 3-ći najveći izaziva $n - 2$ poruka
  - itd.
- Ukupan br. poruka je $n + (n-1) + (n-2) + \ldots + 1 = \Theta(n^2)$



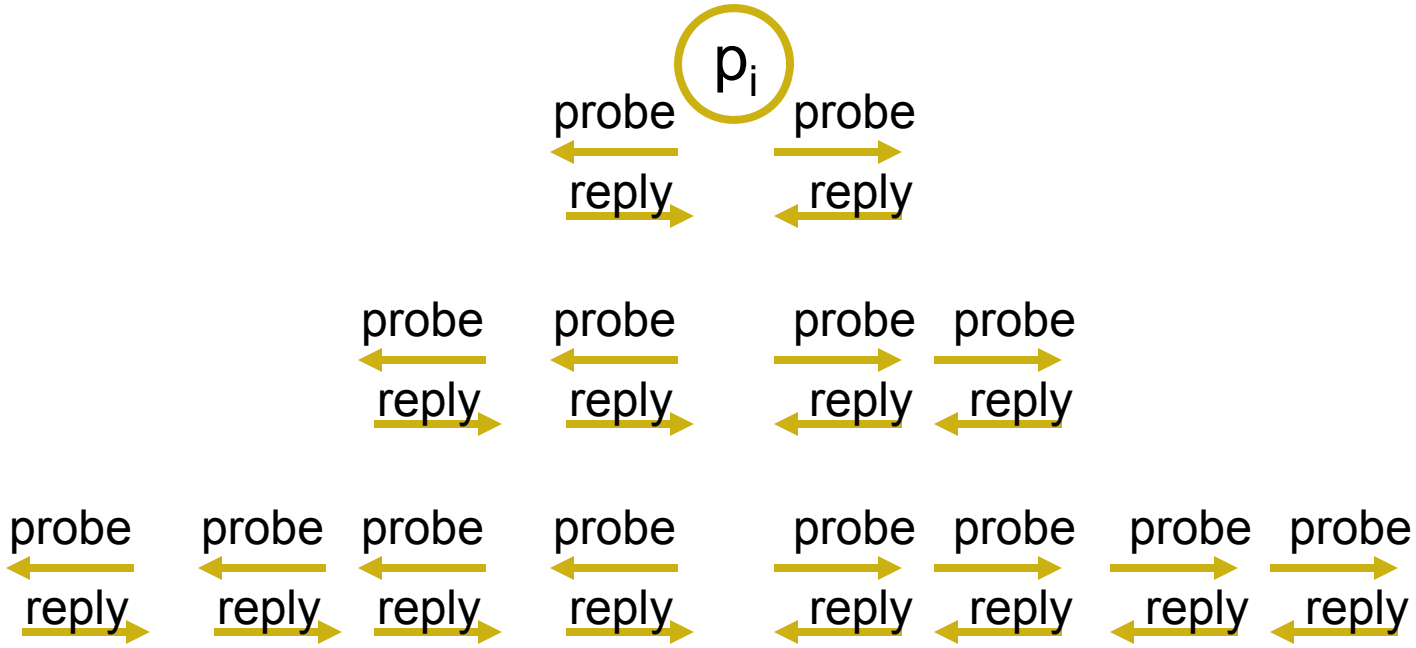
Da li je moguće koristiti manje poruka?

- $O(n^2)$ algoritam je jednostavan i radi i u sinhronom i u asinhronom modelu
- Ali da li možemo rešiti problem sa manje poruka?
- Ideja:
  - Pokušajmo da poruke sa manjim id prelaze kraća rastojanja u prstenu

O(*n* log *n*) algoritam

- Svaki proc. isprobava sukcesivno veće komšiluke u oba pravca (u levo i u desno)
  - veličina komšiluka se *uvostručava* u svakoj fazi
- Ako proba dosegne čvor sa većim id, isprobavanje se zaustavlja
- Ako proba dosegne kraj komšiluka, inicijatoru se šalje nazad odgovor
- Ako inicijator dobije nazad odgovore iz oba pravca, on prelazi na sledeću fazu
- Ako proc. primi probu sa svojim id, on izabira sebe

O(*n* log *n*) algoritam



Analiza O(*n* log *n*) algoritma

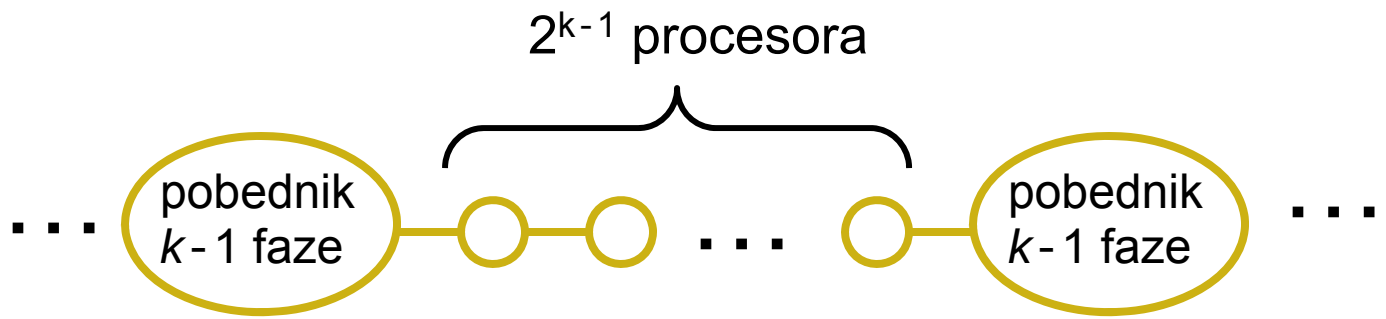
- **Korektnost:** Slično sa $O(n^2)$ algoritmom
- **Broj poruka:**
  - Svaka por. pripada određenoj fazi i inicira je određeni proc.
  - Distanca probe u fazi k je 2^k
  - Broj por. iniciran od strane proc. u fazi k je najviše $4*2^k$ (probe i odgovori u oba pravca)

Analiza O(*n* log *n*) algoritma

- Koliko procesora inicira probe u fazi k ?
- Za $k = 0$, svaki proc. inicira
- Za $k > 0$, svaki proc. koji je „pobednik" u fazi $k - 1$ inicira
  - „pobednik" znači da ima najveći id u svom 2^{k-1} komšiluku

Analiza O(*n* log *n*) algoritma

- Max broj pobednika u fazi *k* - *1* se dešava kada su oni pakovani najgušće što je moguće:



- ukupan broj pobednika u fazi *k* - *1* je najviše

$$n/(2^{k-1} + 1)$$

Analiza O(*n* log *n*) algoritma

- Koliko ima faza?
- U svakoj fazi broj pobednika (faze) se otprilike prepolovi
  - od $n/(2^{k-1} + 1)$ na $n/(2^k + 1)$
- Tako posle otprilike $\log_2 n$ faza, ostaje samo jedan pobednik
  - preciznije, max faza je $\lceil \log(n-1) \rceil + 1$

Analiza O(*n* log *n*) algoritma

- Ukupan broj poruka je zbir, preko svih faza, broja pobednika u toj fazi puta broj poruka nastalih od tog pobednika:

$$\leq 4n + n + \sum_{k=1}^{\lceil \log(n-1) \rceil + 1} 4 \cdot 2^k \cdot n/(2^{k-1}+1)$$

$$< 8n(\log n + 2) + 5n$$

$$= O(n \log n)$$

por. 0 faze

završne por.

por. za faze 1 do $\lceil \log(n-1) \rceil + 1$

Da li možemo bolje?

- $O(n \log n)$ algoritam je složeniji od $O(n^2)$ algoritma ali koristi manje poruka u najgorem slučaju
- Radi i u sinhronom i u asinhronom modelu
- Da li možemo još smanjiti broj poruka?
- Ne u asinhronom modelu…