DISTRIBUIRANI ALGORITMI I SISTEMI

Iz kursa CSCE 668
Proleće 2014

Autor originalne prezentacije:
Prof. Jennifer Welch

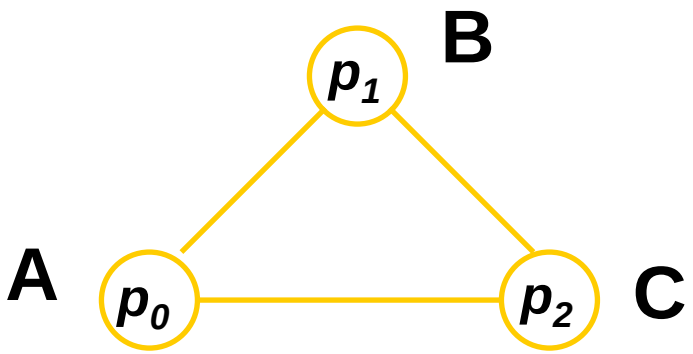
Konsenzus sa vizantijskim otkazima

- Koliko procesora je potrebno da se reši konsenzus kad je $f = 1$?
- Predpost. $n = 2$. Ako p_0 ima ulaz 0 a p_1 ima 1, jedan mora da promeni vred, ali ne oba. Šta ako je jedan proc u otkazu? Kako drugi to može znati?
- Predpost. $n = 3$. ako p_0 ima ulaz 0, p_1 ima ulaz 1, i p_2 je u otkazu, onda je potreban arbitar, ali p_2 može delovati zlonamerno

Donja granica za procesore za $f = 1$

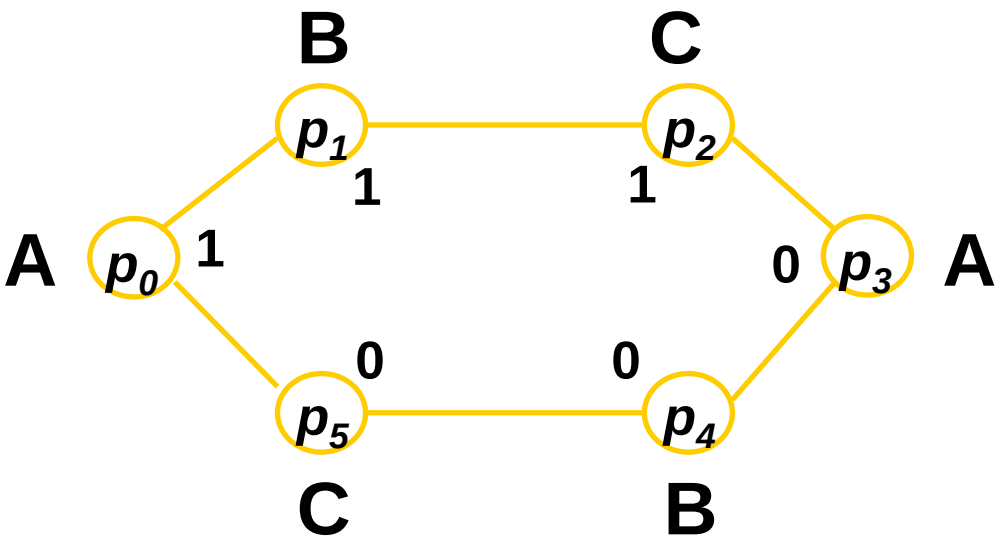
Teorema (10.7): Svaki algoritam konsenzusa za1 vizantijski otkaz mora imati bar 4 procesora

Dokaz: Predpost. radi kontradikcije da postoji algoritam konsenzusa $\mathcal{A}$ = (A,B,C) za 3 procesora i 1 vizantijski otkaz



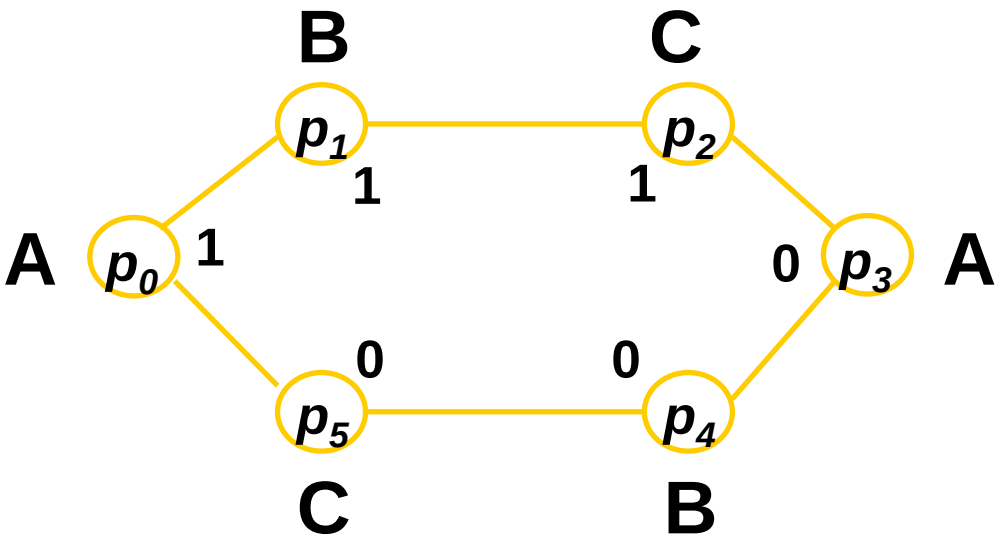
Specificiranje ponašanja otkaza

- Razmotrimo prsten sa 6 ispravnih procesora koji izvršavaju komponente od $\mathcal{A}$ na ovaj način:



Specificiranje ponašanja otkaza

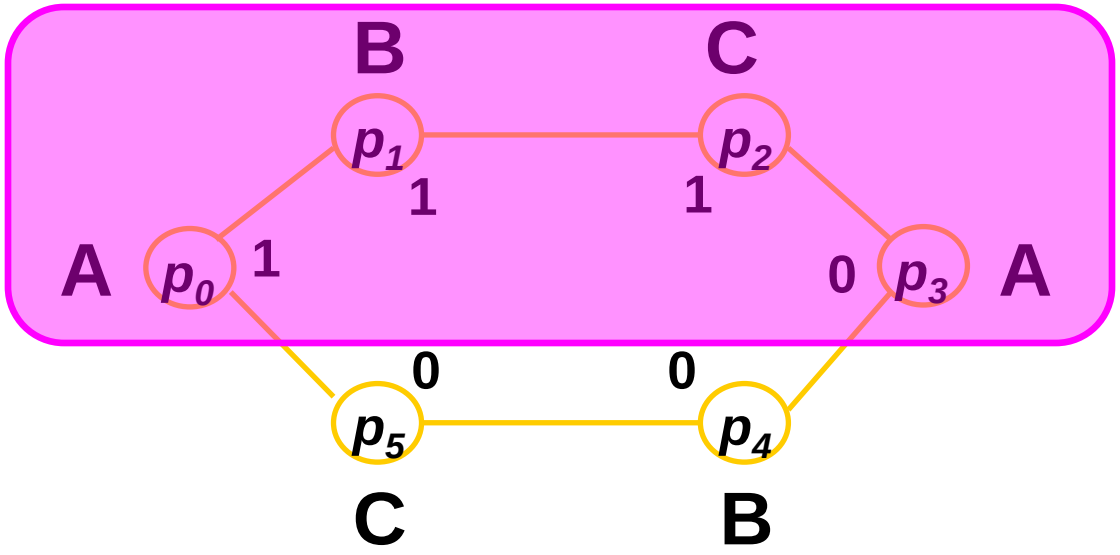
- Razmotrimo prsten sa 6 ispravnih procesora koji izvršavaju komponente od $\mathcal{A}$ na ovaj način:



- Ovo izvršenje β možda (ne) rešava konsenzus (ne mora da uspe)
- Ali procesori rade *nešto* – to ponašanje se koristi za specifikaciju ponašanja procesora u otkazu u izvršenju $\mathcal{A}$ u trouglu

Specificiranje ponašanja otkaza

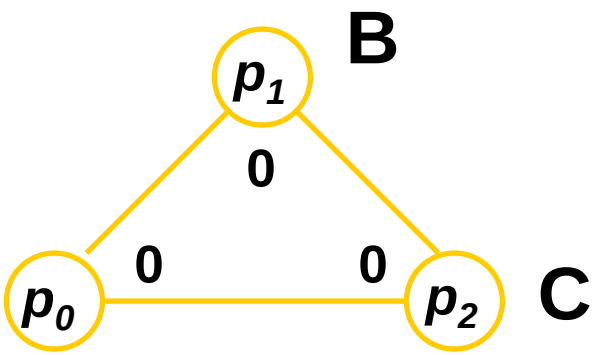
- Razmotrimo prsten sa 6 ispravnih procesora koji izvršavaju komponente od $\mathcal{A}$ na ovaj način:



- Ovo izvršenje β možda (ne) rešava konsenzus (ne mora da uspe)
- Ali procesori rade *nešto* – to ponašanje se koristi za specifikaciju ponašanja procesora u otkazu u izvršenju $\mathcal{A}$ u trouglu

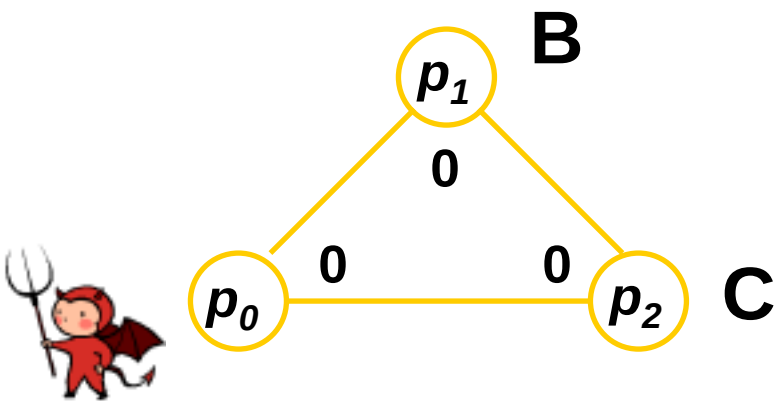
Priprema kontradikcije

- Neka je α_0 ovo izvršenje:



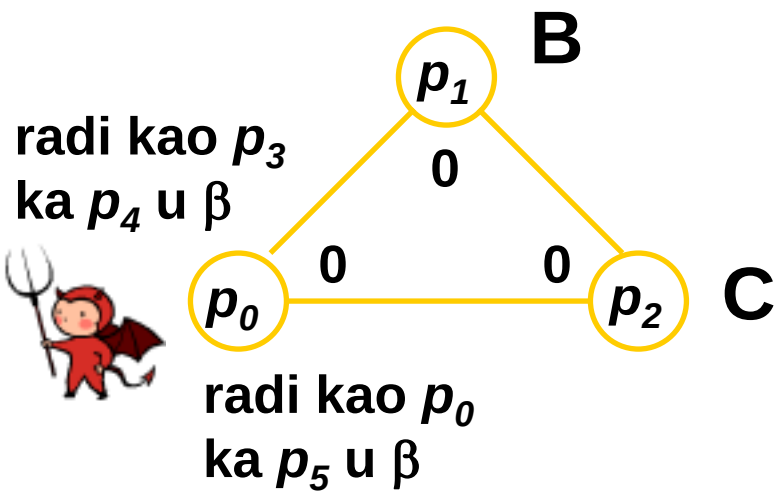
Priprema kontradikcije

- Neka je α_0 ovo izvršenje:



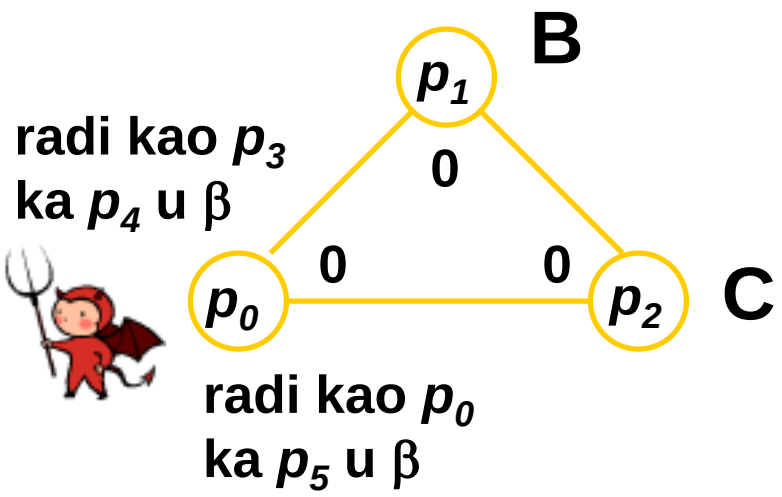
Priprema kontradikcije

- Neka je α_0 ovo izvršenje:



Priprema kontradikcije

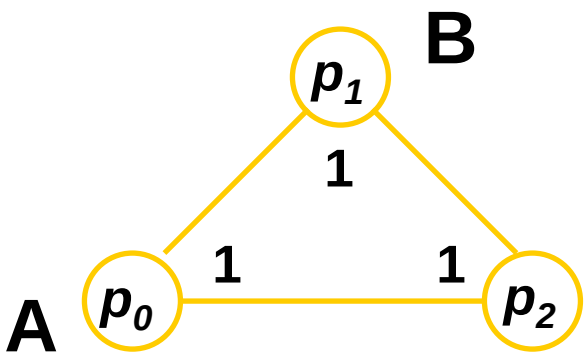
- Neka je α_0 ovo izvršenje:



p_1 i p_2 moraju da odluče 0

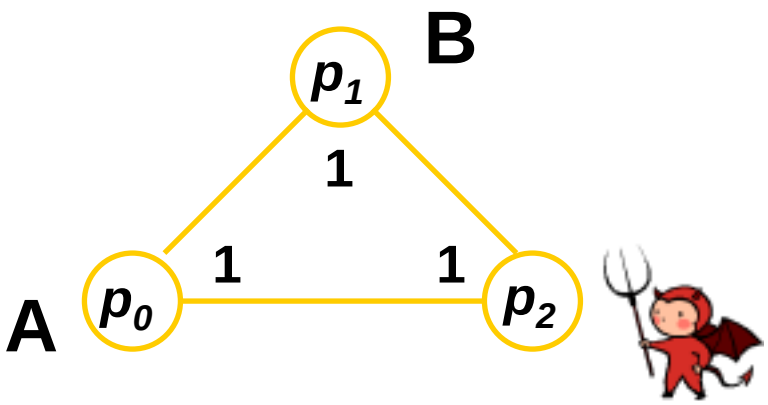
Priprema kontradikcije

- Neka je α_1 ovo izvršenje:



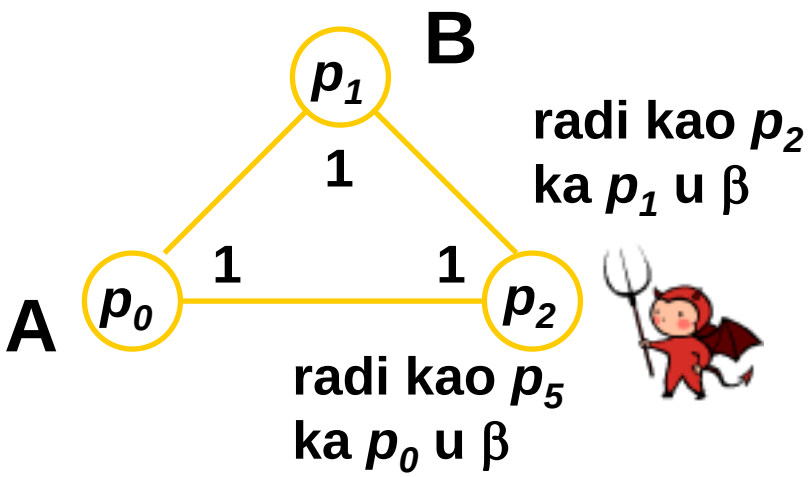
Priprema kontradikcije

- Neka je α_1 ovo izvršenje:



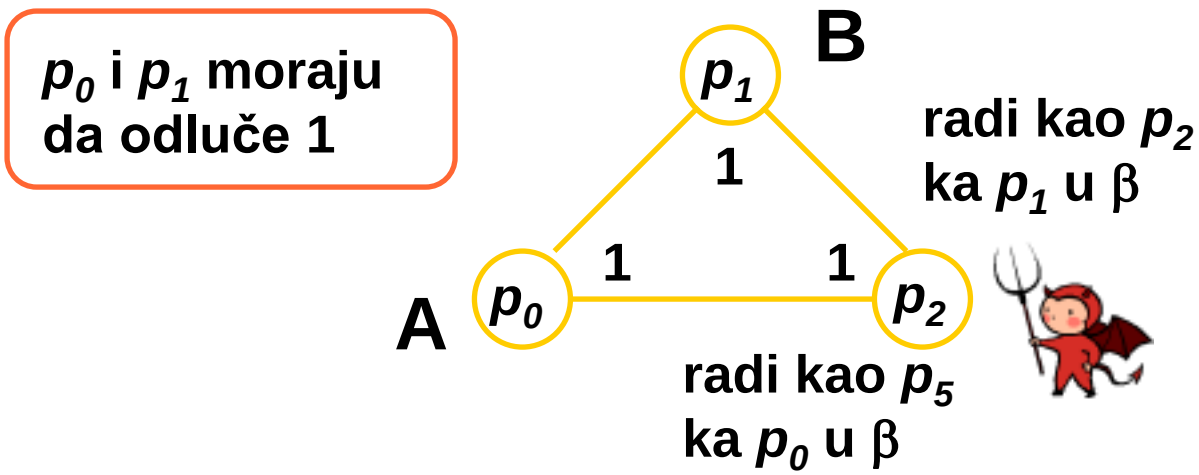
Priprema kontradikcije

- Neka je α_1 ovo izvršenje:



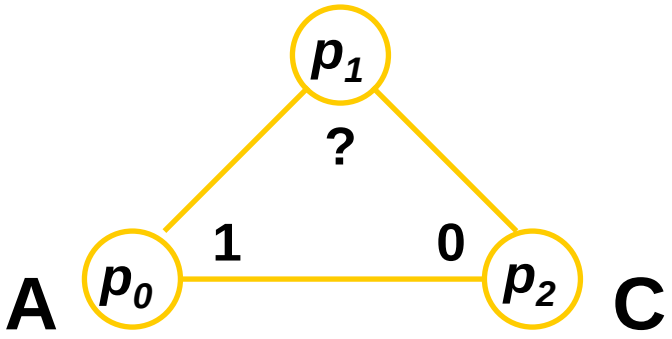
Priprema kontradikcije

- Neka je α_1 ovo izvršenje:



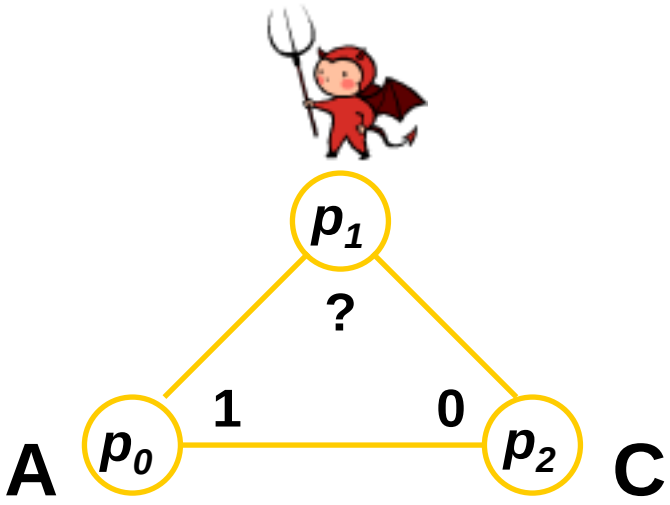
Kontradikcija

- Neka je γ ovo izvršenje:



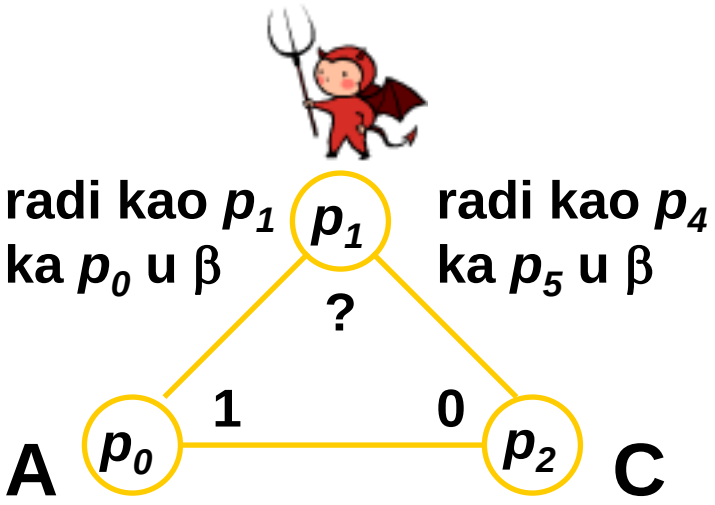
Kontradikcija

- Neka je γ ovo izvršenje:



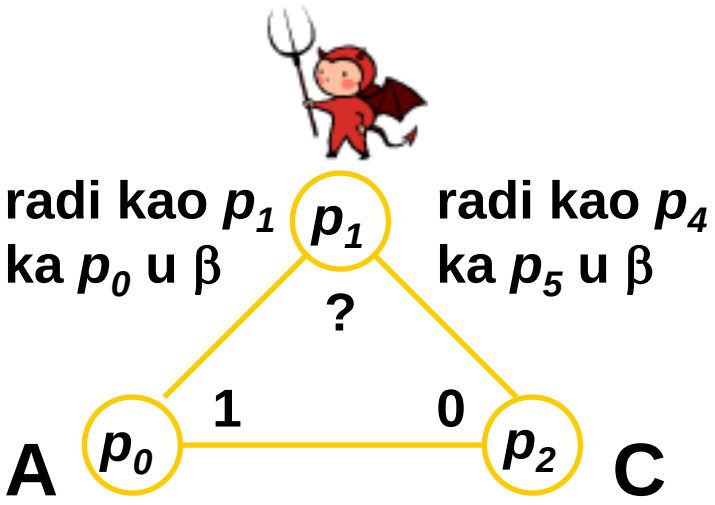
Kontradikcija

- Neka je γ ovo izvršenje:



Kontradikcija

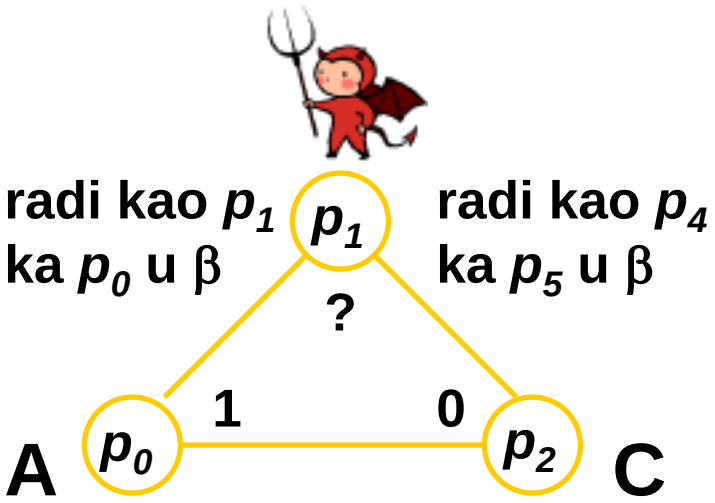
- Neka je γ ovo izvršenje:



Šta p_0 i p_2 odlučuju?

Kontradikcija

- Neka je γ ovo izvršenje:

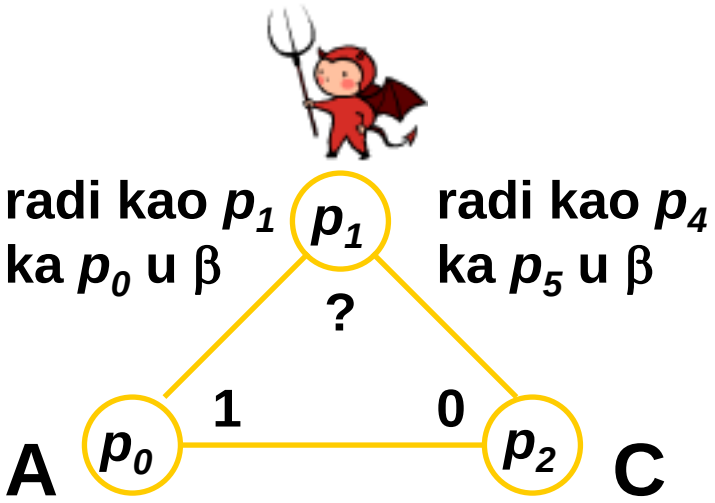


Šta p_0 i p_2 odlučuju?

pogled p_0 u γ = pogled p_0 u β = pogled p_0 u $\alpha_1 \Rightarrow$ p_0 odlučuje 1

Kontradikcija

- Neka je γ ovo izvršenje:



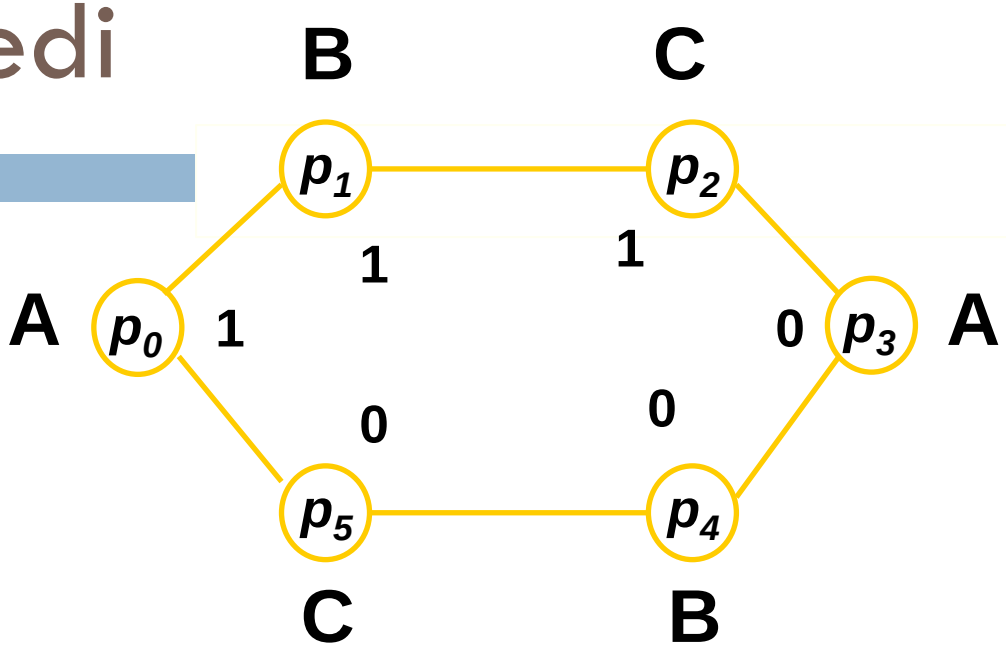
Šta p_0 i p_2 odlučuju?

pogled p_0 u γ = pogled p_0 u β = pogled p_0 u $\alpha_1 \Rightarrow$ p_0 odlučuje 1

pogled p_2 u γ = pogled p_5 u β = pogled p_2 u $\alpha_0 \Rightarrow$ p_2 odlučuje 0

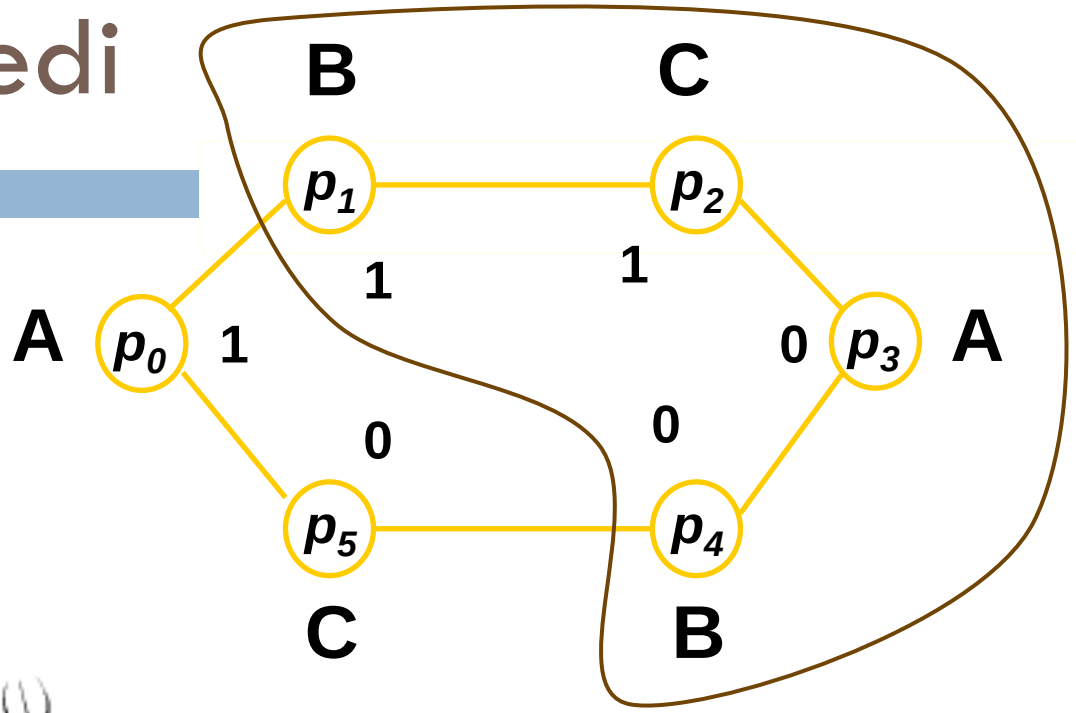
Pogledi

β:



Pogledi

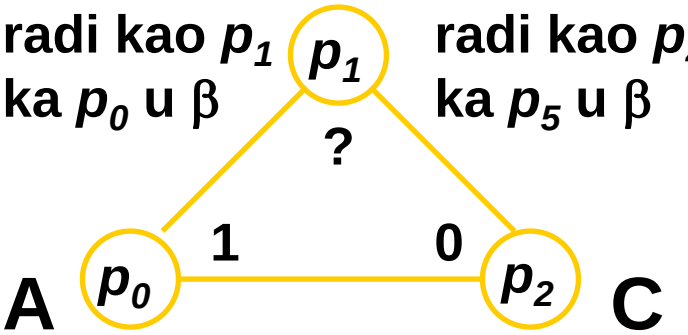
β:



γ:

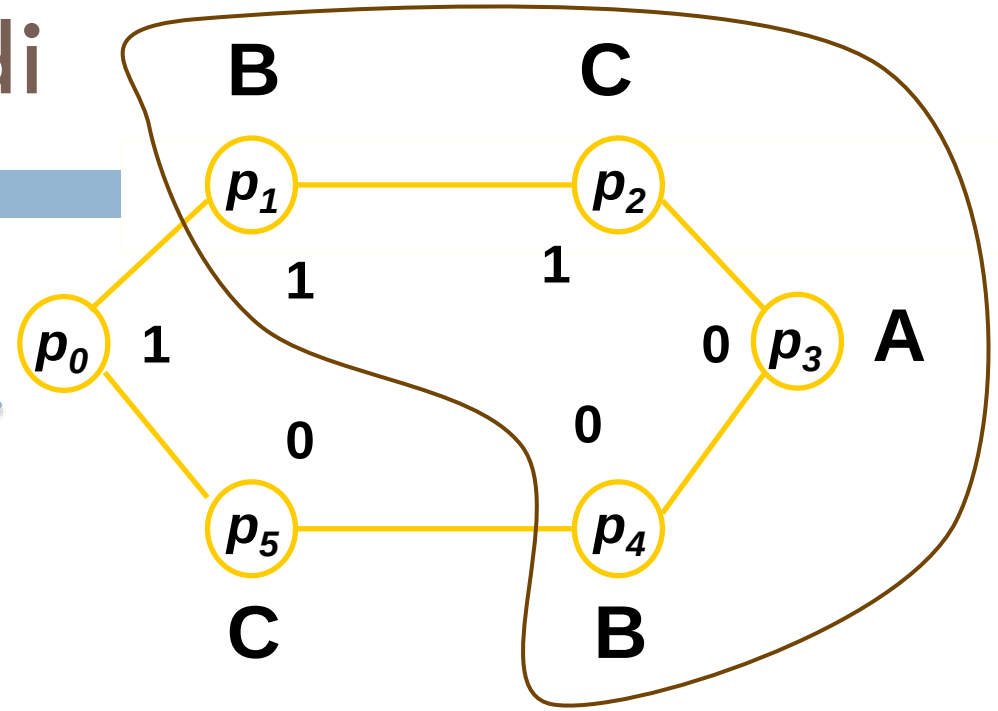
radi kao p_1 ka p_0 u β

radi kao p_4 ka p_5 u β



Pogledi

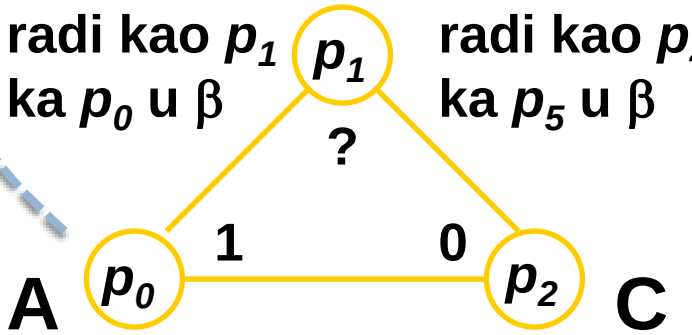
β:



γ:

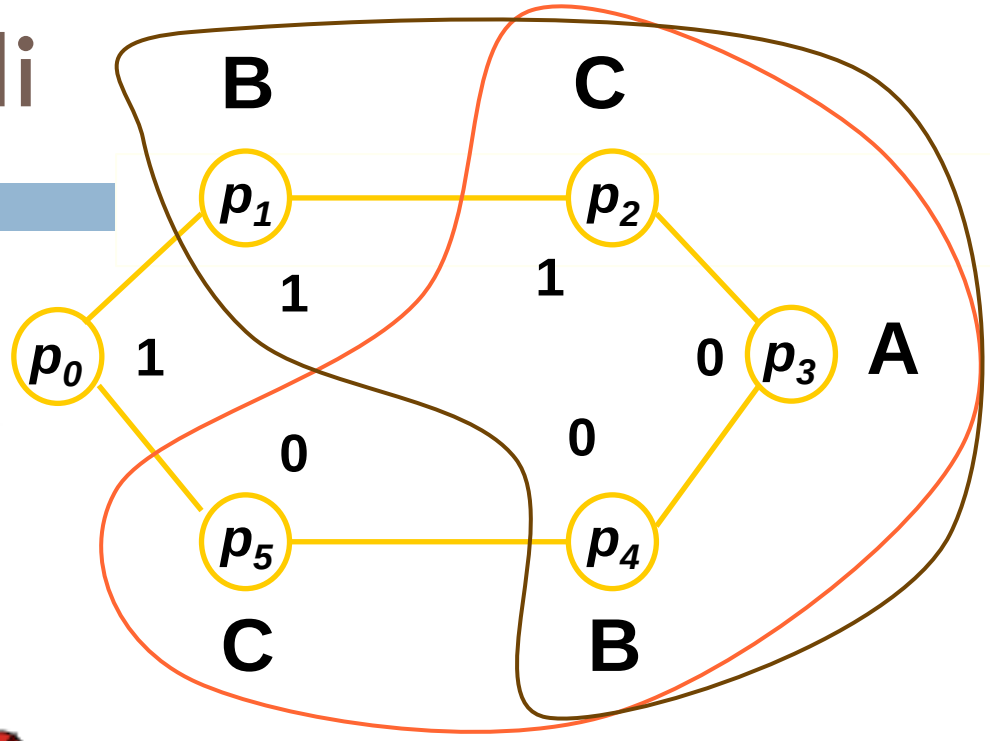
radi kao p_1 ka p_0 u β

radi kao p_4 ka p_5 u β



Pogledi

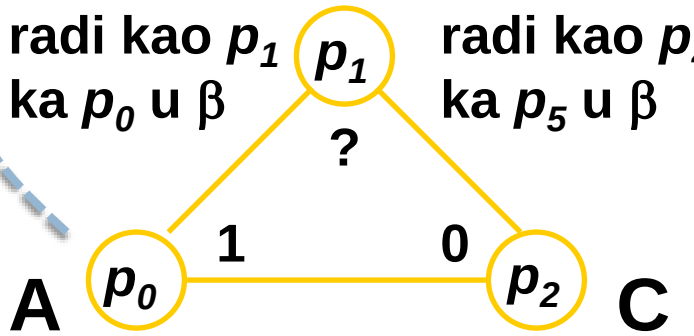
β:



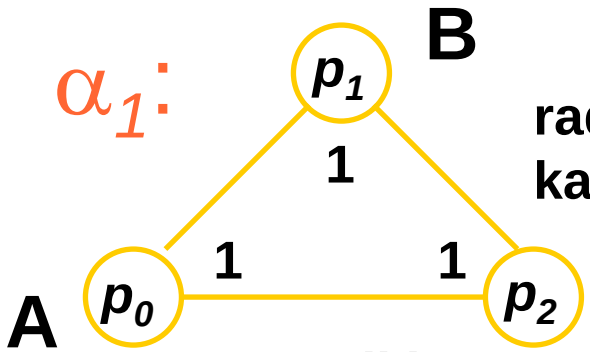
γ:

radi kao p_1
ka p_0 u β

radi kao p_4
ka p_5 u β



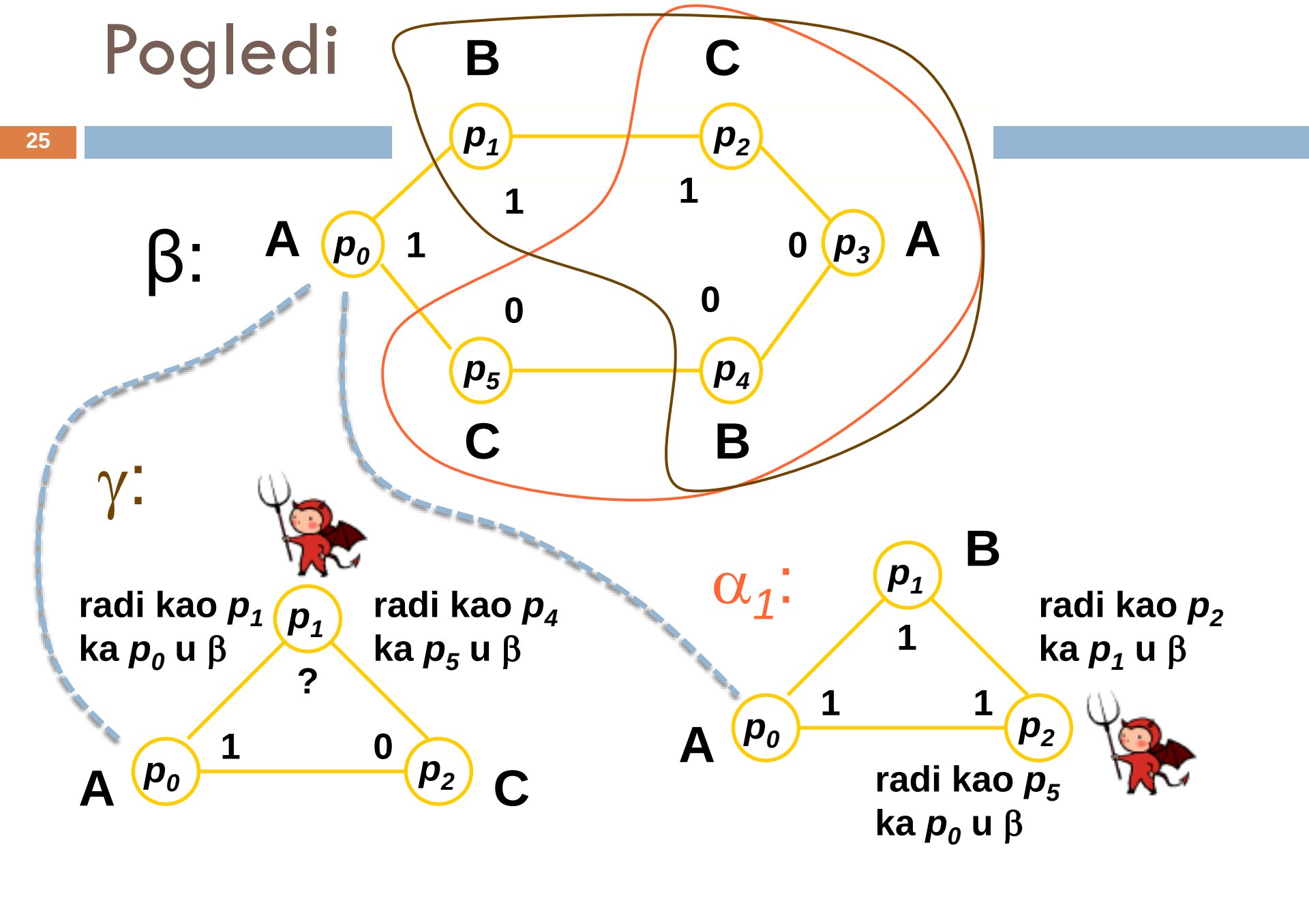
α_1:



radi kao p_2
ka p_1 u β

radi kao p_5
ka p_0 u β

Pogledi


β:
A
B
C
p_0
p_1
p_2
p_3
p_4
p_5
1
1
1
0
0
0
A
B
C
γ:
radi kao p_1
ka p_0 u β
radi kao p_4
ka p_5 u β
?
A
C
p_0
p_1
p_2
1
0
α_1:
B
A
p_0
p_1
p_2
1
1
1
radi kao p_2
ka p_1 u β
radi kao p_5
ka p_0 u β

Donja granica broja procesora za bilo koje f

Teorema 10.8: Bilo koji algoritam konsenzusa za f vizantijskih otkaza mora imati bar $3f+1$ procesora

Dokaz: Koristi redukciju na 3:1 slučaj

- Predpost. radi kontradickije da postoji algoritam $\mathcal{A}$ za $f > 1$ otkaza i $n = 3f$ procesora
- Koristimo $\mathcal{A}$ kao podprogram za konstruisanje algoritma za 1 otkaz i 3 procesora, što dovodi do kontradikcije

Redukcija

- Podelimo $n \leq 3f$ procesora u 3 skupa, P_0, P_1, i P_2, svaki veličine najviše f
- U slučaju $n = 3$, neka procesori:
  - p_0 simulira P_0
  - p_1 simulira P_1
  - p_2 simulira P_2
- Ako je 1 procesor u otkazu u sistemu sa $n = 3$, onda je najviše f procesora u otkazu u simuliranom sistemu
- Zato je simulirani sistem korektan
- Neka procesori u sistemu sa $n = 3$ odluče isto kao simulirani procesori, i njihove odluke će takođe biti korektne

Algoritam eksponencijalnog stabla

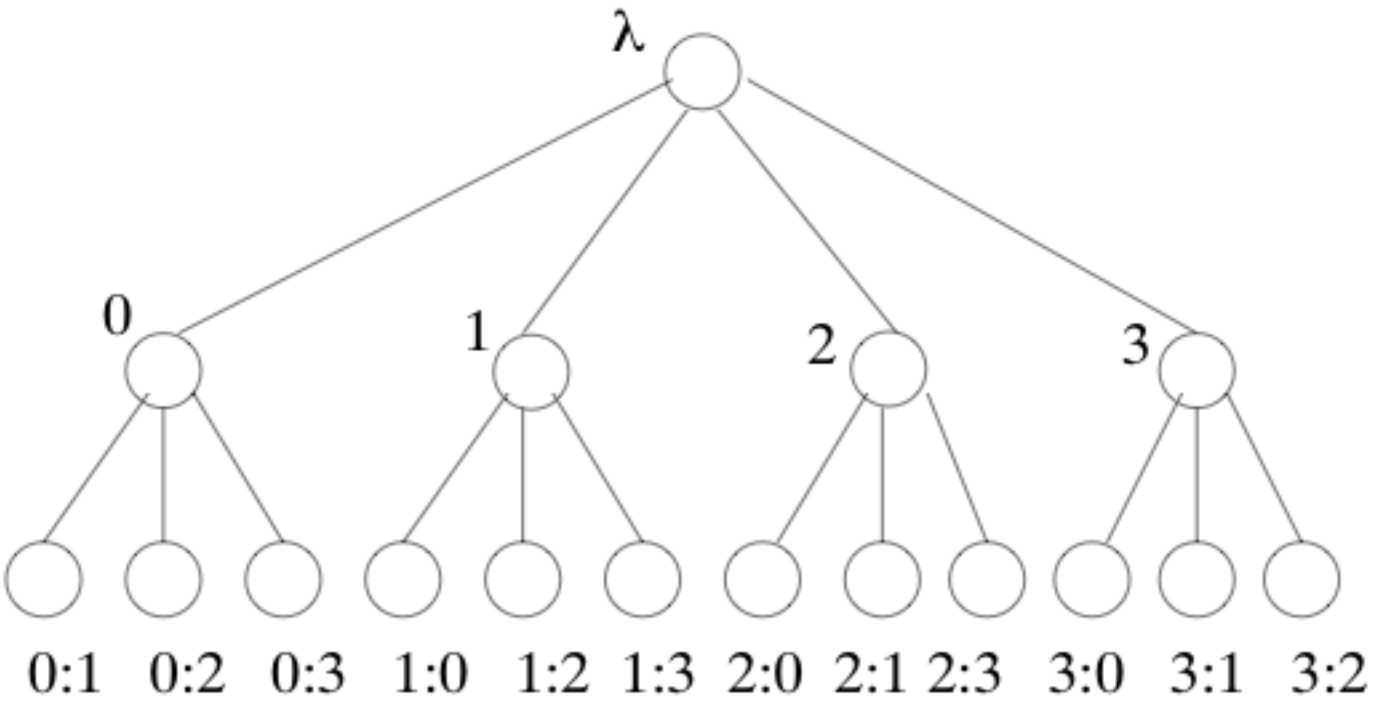
- Ovaj algoritam koristi:
  - $f + 1$ rudni (optimalan)
  - $n = 3f + 1$ procesora (optimalan)
  - eksponencijalnu veličinu poruka (suboptimalan)
- Svaki procesor održava stablo u svom lokalnom stanju
- Vrednosti u stablu se popunjavaju tokom $f + 1$ rundi
- Na kraju, vrednosti u stablu se koriste za odluku

Lokalno stablo

- Svaki čvor stabla je označen sa sekvencom jedinstvenih indeksa procesora
- Labela korena je prazna sekve. λ; koren ima nivo 0
- Koren ima n potomaka, označenih sa 0 do $n - 1$
- Čvor potomak označen sa i ima $n - 1$ potomaka, označenih sa $i : 0$ do $i : n-1$ (izostavlja se $i : i$)
- Čvor na nivou d označen sa v ima $n - d$ potomaka, označenih sa $v : 0$ do $v : n-1$ (izostavljajući bilo koji indeks koji se pojavljuje u v)
- Čvorovi na nivou $f + 1$ su listovi

Primer lokalnog stabla

Stablo za $n = 4$ i $f = 1$:



Popunjavanje čvorova stabla

- Inicijalno stavi svoj ulaz u koren (nivo 0)
- Runda 1:
  - šalji svima nivo 0 svog stabla
  - stavi vred *x* primljenu od svakog p_i u čvoru stabla označe. sa *i* (nivo 1); ako je potrebno, koristi podra. vred. (default)
  - "p_i mi je rekao da je ulaz od p_i bio *x*"
- Runda 2:
  - šalji svima nivo 1 svog stabla
  - stavi vred *x* primljenu od svakog p_i za svaki čvor stabla *k* u čvoru označe. sa *k* : *i* (nivo 2); ili podrazumevanu vred.
  - "p_i mi je rekao da je p_k rekao p_i da je ulaz od p_k bio *x*"
- Nastavi za $f + 1$ rundi

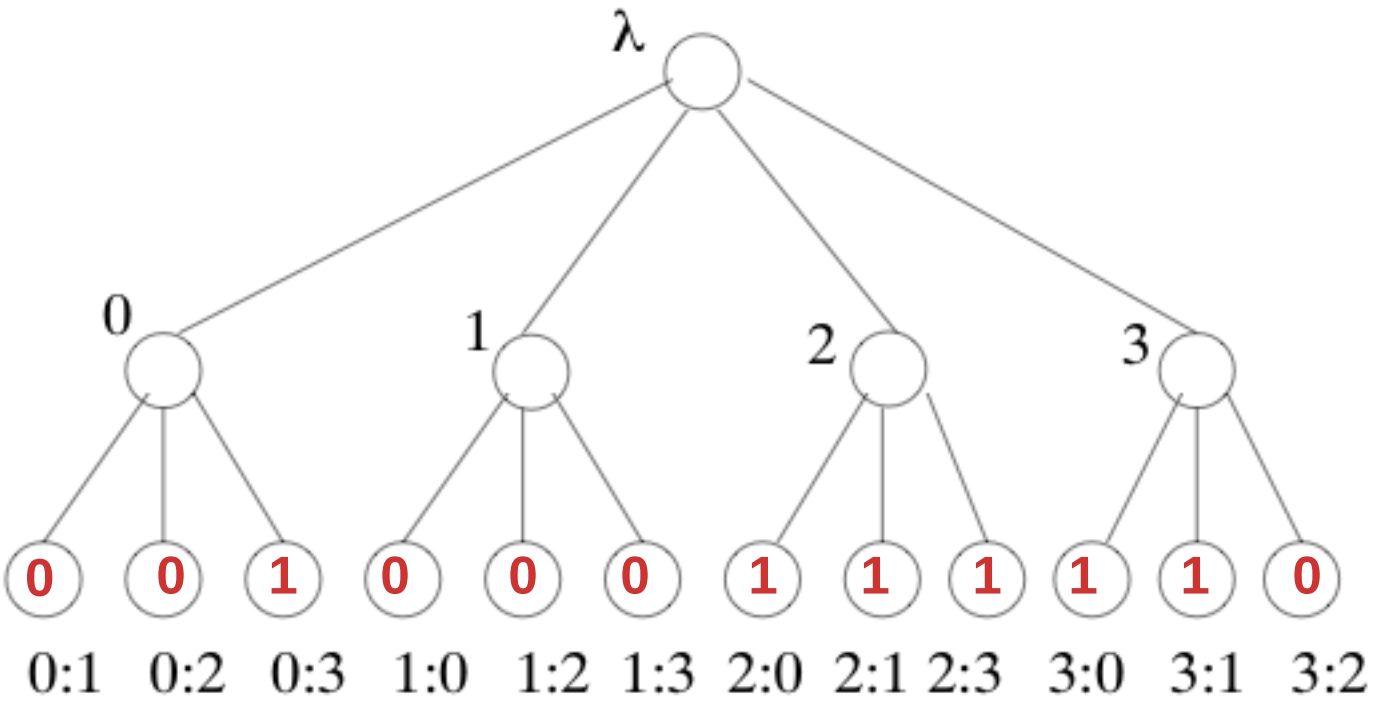
Odlučivanje

- U rundi $f + 1$, svaki procesor koristi vred u svom stablu da izračuna svoju odluku
- Rekurzivno računaj vred „rešenja“ za koren stabla, `resolve`(λ), na osnovu vred „rešenja“ za druge čvorove stabla:

$$\texttt{resolve}(\pi) = \begin{cases} \text{vred u čvoru označe. sa } \pi \text{ ako je to list} \\ \text{majority}\{\texttt{resolve}(\pi') : \pi' \text{ je potomak od } \pi\} \text{ inače (koristi podraz. vr. ako nema većine)} \end{cases}$$

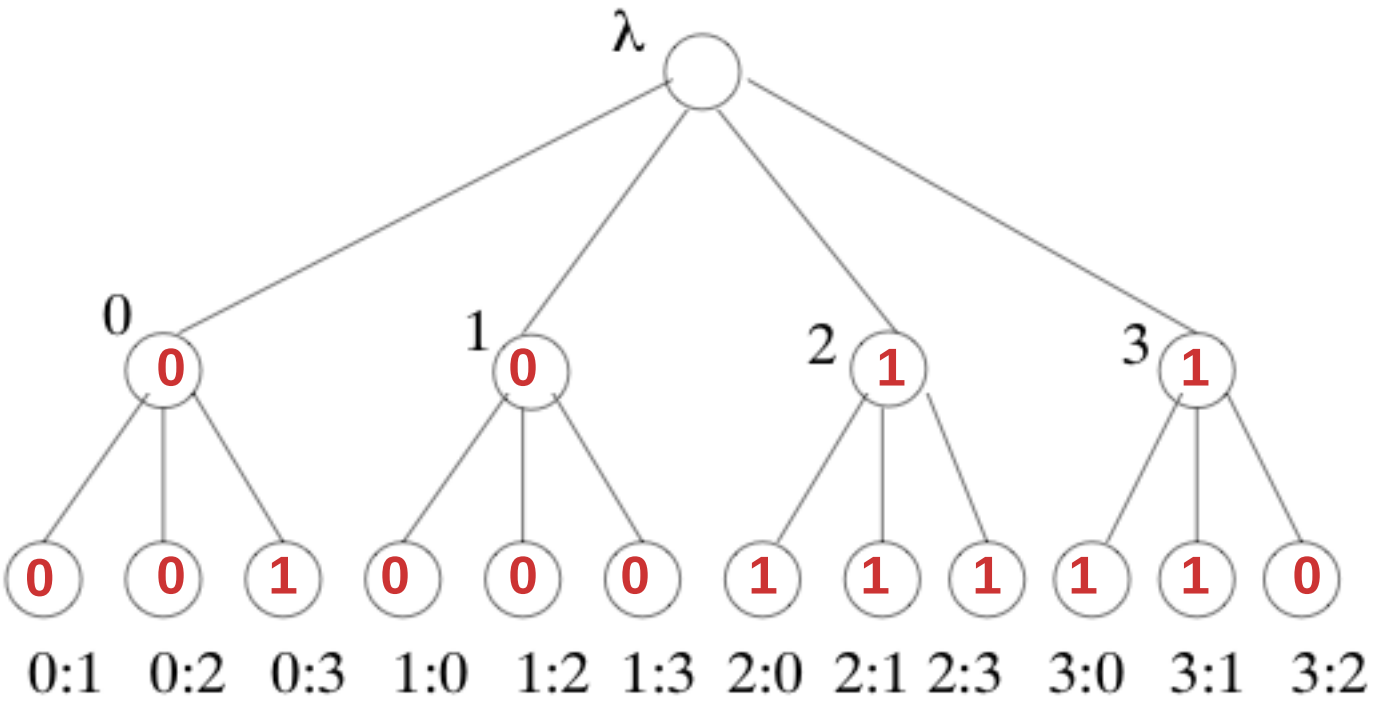
Primer vrednosti rešenja

Stablo za $n = 4$ i $f = 1$:



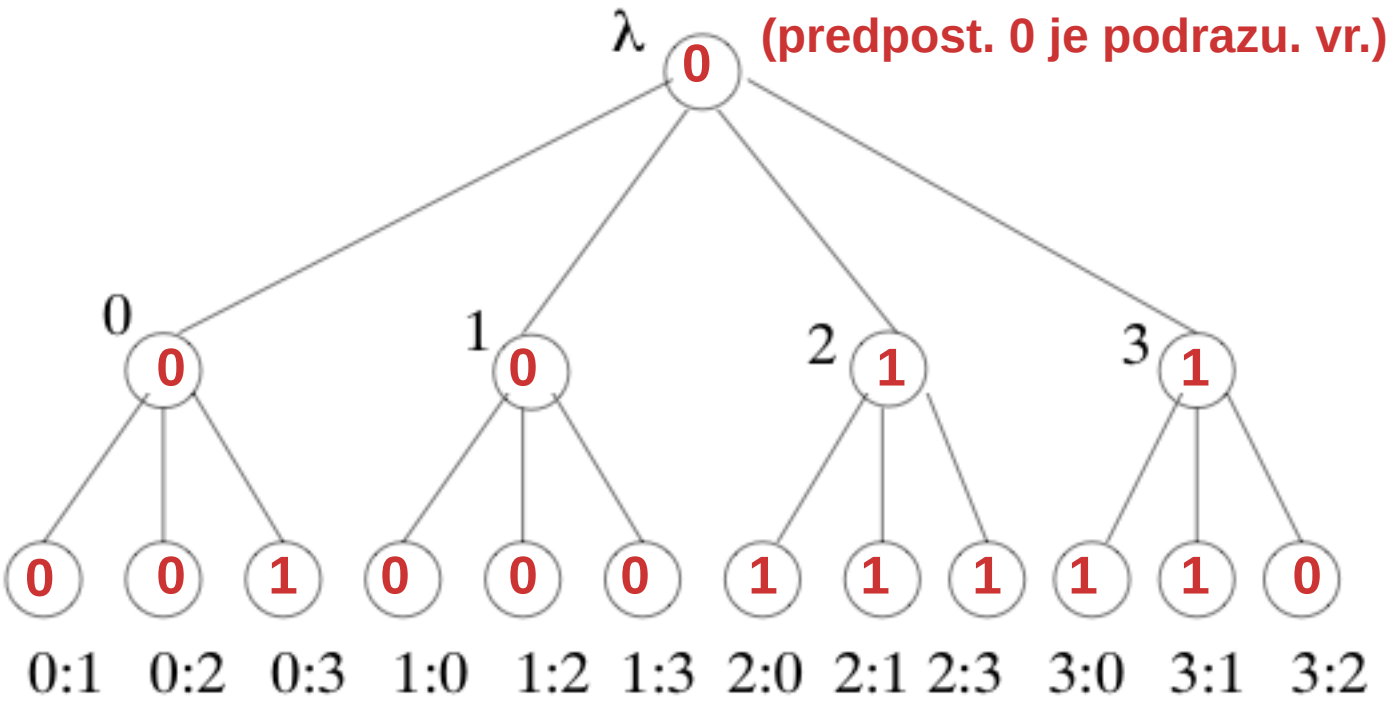
Primer vrednosti rešenja

Stablo za $n = 4$ i $f = 1$:



Primer vrednosti rešenja

Stablo za $n = 4$ i $f = 1$:



Složenost

Algoritam eksponencijalnog stabla koristi

- $n > 3f$ procesora
- $f + 1$ rundi
- poruke eksponencijalne veličine:
  - svaka poruka u rundi r sadrži:

    $n(n-1)(n-2)\ldots(n-(r-2))$ vrednosti
  - Kad je $r = f + 1$, ta veličina je eksponencijalna ako je f više od konstante u odnosu na n

Jedan polinomijalan algoritam

- Veličina por se može redukovati na polinomijalnu pomoću jednog jednostavnog algoritma
- Broj procesora se povećava na

$$n > 4f$$

- Broj rundi se povećava na

$$2(f + 1)$$

- Koristi se $f + 1$ faza, svaka uzima dve runde

Algoritam Kralj (Faze)

Kod za svaki procesor p_i:

pref := moj ulaz

prva runda faze k, $1 \leq k \leq f+1$:

šalji svima pref

primi pref vrednosti od drugih

neka je maj vred koja se pojavila $> n/2$ puta (0 ako takve nema)

neka je mult broj pojava od maj

druga runda faze k:

if $i = k$ **then** šalji svima maj // Ja sam kralj faze

primi tie-breaker (arbitražnu vred) od p_k (0 ako je nema)

if mult $> n/2 + f$

then pref := maj

else pref := tie-breaker

if $k = f + 1$ **then** odluči pref

Dogovor

Pomoću 2 leme dokazujemo dogovor:

- Pošto postoji $f + 1$ faza, bar jedna ima ispravnog kralja
- Lema ispravnog kralja implicira da na kraju te faze, svi ispravni procesori imaju istu preferencu
- Lema anonimne faze implicira da od te faze na dalje, svi ispravni procesori imaju istu preferencu
- Zati svi ispravni proc odlučuju isto

Mere složenosti

- broj procesora $n > 4f$
- $2(f + 1)$ rundi
- $O(n^2 f)$ poruka, svaka veličine $\log |V|$