nedelja informatike 3

Distribuirani sistemi

Nikola Jovanović

Matematička gimnazija
Nedelja informatike 3

16. decembar 2016.

Motivacija

- Sve jači a sve jeftiniji mikroprocesori
- Sve brže mreže
- $\Longrightarrow$ Jednostavno i isplativo povezivati računare u kompleksne sisteme

Dosadna definicija

- **Distribuirani sistem** je skup nezavisnih računara koji svojim korisnicima izgleda kao jedinstven koherentan sistem

Dosadna definicija

- **Distribuirani sistem** je skup **nezavisnih** računara koji svojim korisnicima izgleda kao **jedinstven** koherentan sistem

Primeri

- Internet
- Mail serveri
- Cloud computing
- Distribuirani fajl sistemi i skladišta podataka (Dropbox, GDrive)
- Telefonija, senzorske mreže, git...

Terminologija

- Proces
- Čvor
- Mašina
- Računar
- ...

Ciljevi

- Učiniti resurse dostupnim
- Sakriti distributivnost (transparencija)
  - Pristup
  - Lokacija
  - Migracija
  - Relokacija
  - Replikacija
  - Konkurentnost
  - Greške
- Napraviti otvorenu jasnu specifikaciju interakcija (razdvajanje polise od mehanizma)
- Omogućiti skalabilnost (decentralizovana rešenja)

Pogrešne pretpostavke

- Mreža je pouzdana
- Mreža je sigurna
- Mreža je homogena
- Topologija se ne menja
- Ne postoji kašnjenje
- Protok je beskonačan
- Cena transporta je nula
- Trajna memorija je trajna
- Postoji jedan administrator

Organizacija

- *Kako najbolje organizovati komponente jednog distribuiranog sistema?*
- Slojevi (podaci, procesiranje, UI)
- Centralizovane arhitekture
  - Klijent-server (e-mail)
- Decentralizovane arhitekture
  - Struktuirani *peer-to-peer* sistemi (DHT)
  - Nestruktuirani *peer-to-peer* sistemi (random graf, geometrijski modelirane mreže)

Organizacija

- *Kako najbolje organizovati komponente jednog distribuiranog sistema?*
- Slojevi (podaci, procesiranje, UI)
- Centralizovane arhitekture
  - Klijent-server (e-mail)
- Decentralizovane arhitekture
  - Struktuirani *peer-to-peer* sistemi (DHT)
  - Nestruktuirani *peer-to-peer* sistemi (random graf, geometrijski modelirane mreže)
  - Hibridni *peer-to-peer* sistemi (bitTorrent)

Komunikacija

- *Kako ostvariti pouzdanu komunikaciju između čvorova?*
- RPC (Remote Procedure Call)
  - *Client/server stub*
  - Maršalovanje, različiti tipovi podataka
  - Sinhrona/asinhrona komunikacija (*blocking calls*)
- MOM (Message Oriented Communication)
  - Kada druga strana nije uvek živa
  - *Sockets*
- *Streaming* (sinhronizacija?)
- *Broadcast* i *multicast*
- „Tračarenje” (*push vs pull*)

Identifikacija

- *Kako pronaći željenu mašinu/resurs?*
- Rutiranje zahteva
- Jedno ime - više adresa (replikacija i migracija)
- „Ravna” imena - *forwarding*, kućna lokacija, DHT (opet!)
- Struktuirana imena - *namespaces* (DNS, FS)
- Atributi - Youtube?

Replikacija

- *Kako pametno iskoristiti redundantnost?*
- $c \sim 300000 \frac{km}{s}$
- Pouzdanost i performanse
- Odabir lokacije za repliku
- Nalaženje najbliže replike

Otpornost na greške

- *Kako održati sistem u životu i kada se stvari ruše?*
- Arhitektura može dosta pomoći
- Oporavak od parcijalnog rušenja - *logs*
- Šta ako se neko ugasi usred transakcije?

Otpornost na greške

- *Kako održati sistem u životu i kada se stvari ruše?*
- Arhitektura može dosta pomoći
- Oporavak od parcijalnog rušenja - *logs*
- Šta ako se neko ugasi usred transakcije?
- Transakcije:
  - **A**tomicity - „Sve ili ništa”
  - **C**onsistency - Čuvanje invarijanti
  - **I**solation - Potpuna konkurentnost
  - **D**urability - Trajnost svake izvršene transakcije

Otpornost na greške

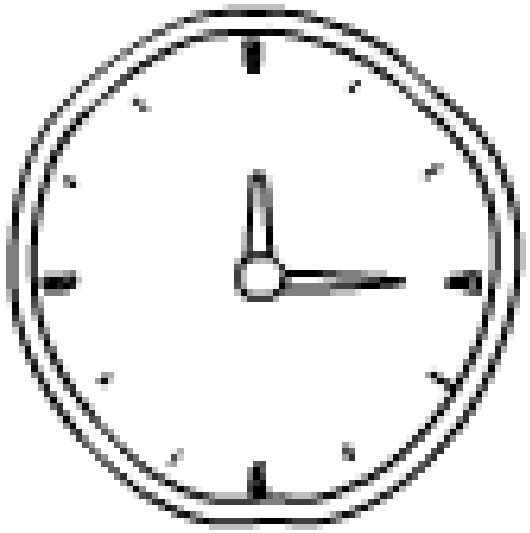
- *Kako održati sistem u životu i kada se stvari ruše?*
- Arhitektura može dosta pomoći
- Oporavak od parcijalnog rušenja - *logs*
- Šta ako se neko ugasi usred transakcije?
- Transakcije:
  - **A**tomicity - „Sve ili ništa”
  - **C**onsistency - Čuvanje invarijanti
  - **I**solation - Potpuna konkurentnost
  - **D**urability - Trajnost svake izvršene transakcije
- Još jedan akronim:
  - **C**onsistency - Svako čitanje daje najaktuelniji podatak
  - **A**vailability - Svaki zahtev dobija odgovor
  - **P**artition tolerance - Sistem radi i ako se mreža razdvoji
  - *Choose two*

Sigurnost

- *Kako se zaštititi od malicioznih napada?*
- Sigurni kanali komunikacije
- Autorizacija
- Ko će da se cima...

Sinhronizacija

- *Kako usaglasiti ceo sistem?*
- Koliko je zapravo sati?
- Šta se desilo pre?
- Smem li ja da diram ovo?
- Ko je tu glavni?



Sinhronizacija časovnika: Koliko je zapravo sati?

- Primer?

Sinhronizacija časovnika: Koliko je zapravo sati?

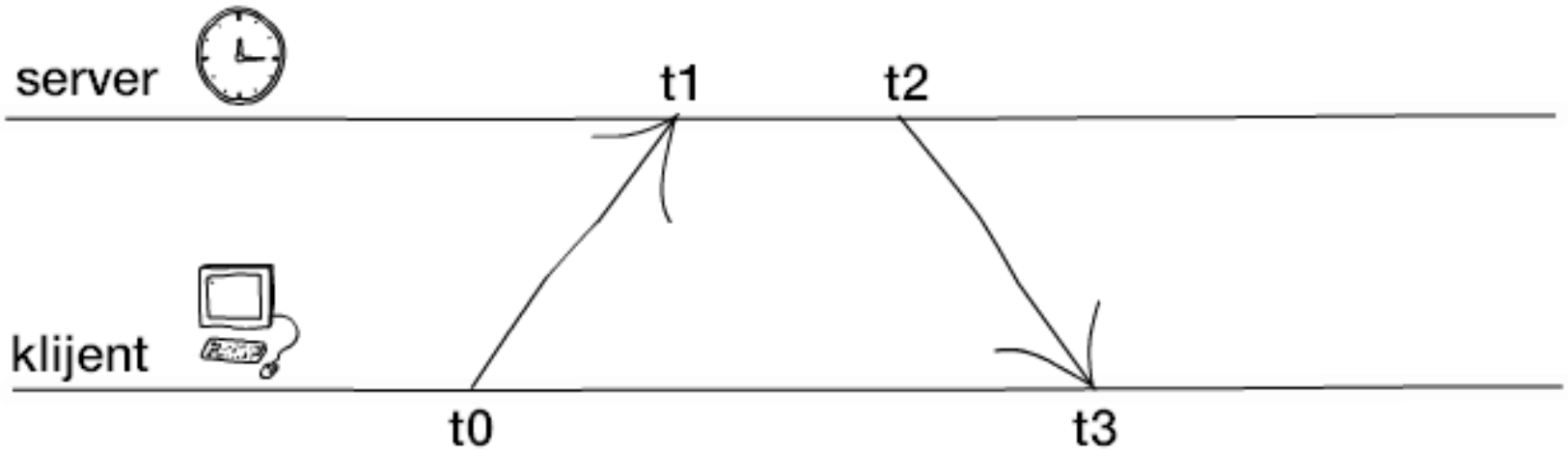
- Primer?
  - `make`
- U centralizovanom sistemu vreme je jednoznačno
- U distribuiranom sistemu svaka mašina ima svoj časovnik
- Nezanemarljivo trajanje komunikacije

O vremenu

- Kvarcni kristali - *drift*
- Astronomski časovnik (Sunce u najvišoj tački) - 17. vek
- Atomski časovnik (Cezijum 133, TAI) - 1948
  - Ali ovo sad nije u skladu sa astronomijom?
  - *Leap* sekunde (31.12.2016. 23:59:60) - UTC
  - *G*lobalni autoritet: radio stanice, sateliti (GPS - 4/29 satelita)

1.1 Network Time Protocol (NTP)

- Klijent bira skup pouzdanih time servera (Marzullo's algorithm) i pokušava da proceni svoj offset u odnosu na „tačno vreme"

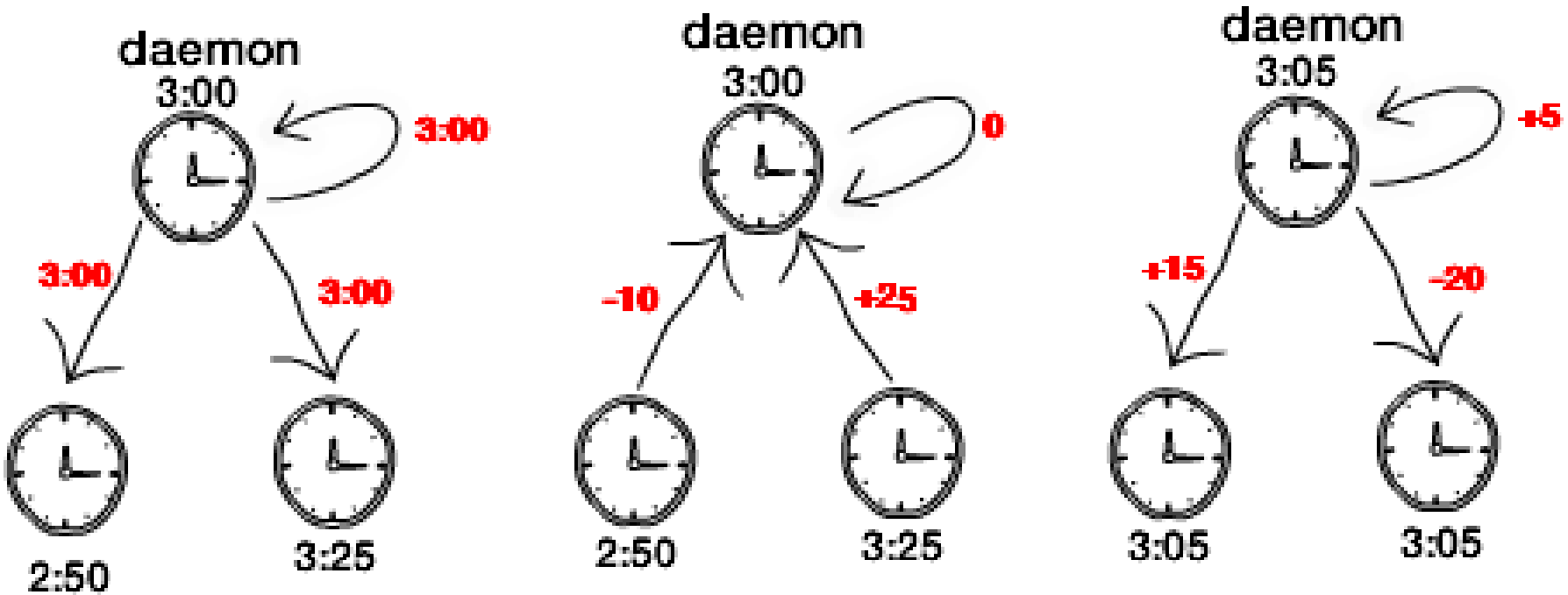


1.1 Network Time Protocol (NTP)

- $\theta = \frac{(t_1 - t_0) + (t_2 - t_3)}{2}$ - kašnjenje
- $\sigma = (t_3 - t_0) - (t_2 - t_1)$ - *round-trip delay*
- **Postepeno** nadoknađivanje kašnjenja
- Nivoi pouzdanosti - *strata*

1.2 Berkli algoritam

- Klijenti nemaju pristup time serverima niti komuniciraju sa eksternim svetom - bitno je samo da se slažu oko vremena
- *Daemon* periodično šalje upite i komande za sinhronizaciju



1.3 RBS algoritam

- Pretpostavka u senzornim mrežama: konstantno kašnjenje poruke
- Jedan čvor pošalje N poruka a ostali se međusobno sinhronizuju
- $Offset[p,q] = \frac{\sum_{k=1}^{N}(T_{p,k} - T_{q,k})}{N}$
- Drift pravi problem - linearna regresija:
  - $Offset[p,q](t) = \alpha t + \beta$

Logičke čuke: Šta se desilo pre?

- Pravo vreme nije neophodno za redosled događaja
- Praćenje sopstvenih i tuđih događaja (lokalni događaji i komunikacija), relativan poredak
- Ako dva procesa ne komuniciraju odnos njihovih časovnika nije bitan

2.1 Lamportovi logički časovnici

- Relacija „desilo-pre” („lanac”, $\rightarrow$):
  - 1. Ako su A i B događaji na istom procesu i A se desio pre B važi $A \rightarrow B$
  - 2. Ako je A događaj slanja poruke M a B događaj primanja poruke M važi $A \rightarrow B$
  - 3. Relacija $\rightarrow$ je tranzitivna
- **Konkurentni** događaji: ne važi ni $A \rightarrow B$ ni $B \rightarrow A$ - ne postoji kauzalni lanac

2.1 Lamportovi logički časovnici

- Algoritam - svaki proces ima svoj brojač i svaki događaj E ima svoj „timestamp” $T(E)$:
  1. Proces uvećava svoj brojač pre svakog lokalnog događaja
  2. Proces šalje vrednost svog brojača uz svaku poruku
  3. Proces koji primi poruku postavlja svoj brojač na brojač iz poruke ukoliko je veći; nakon toga uvećava brojač za 1 i smatra poruku primljenom
- Važi konzistentnost: $A \to B \implies T(A) < T(B)$
- Ne važi jaka konzistentnost: $T(A) < T(B) \not\Rightarrow A \to B$
  - Vektorski časovnici!

2.2 Vektorski časovnici

- Svaki proces i (od ukupno N) čuva vektor V_i dužine N:
  - $V_i[p]$ je poslednja informacija koju proces i ima o brojaču procesa p
- Algoritam:
  - 1. Proces i uvećava $V_i[i]$ pre svakog lokalnog događaja
  - 2. Proces i šalje vrednost svog vektora uz svaki poruku
  - 3. Proces i koji primi poruku uvećava $V_i[i]$ za 1 i popunjava svaki element u svom vektoru kao maksimum njegove trenutne vrednosti i vrednosti na toj poziciji u vektoru poruke
- Važi jaka konzistentnost: Ako je V_a u momentu kad se dogodio događaj A manji od V_b u momentu kad se dogodio događaj B $\implies A \to B$
- $V_a < V_b \iff (\forall i) V_a[i] \leq V_b[i] \wedge (\exists j) V_a[j] < V_b[j]$

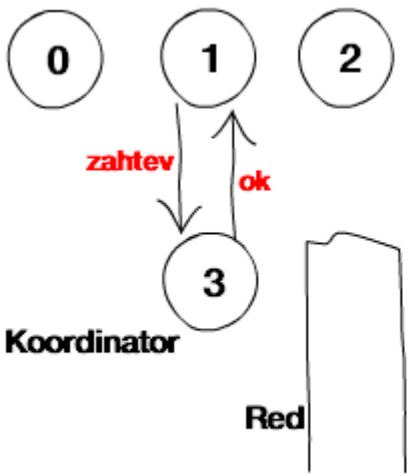
Kontrola pristupa: Smem li ja da diram ovo?

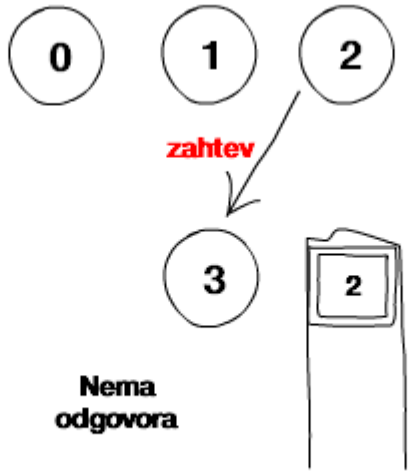
- Mutex - mehanizam koji omogućava ekskluzivni pristup resursima
- Poželjna svojstva algoritma:
  - Izbegavanje *deadlock*-a (svako čeka na nekog drugog)
  - Izbegavanje izgladnjivanja (neko dobije pristup više puta dok neko ne dobije nijednom)
  - Fer redosled (procesi dobijaju pristup onim redom kojim ga zahtevaju)
  - Otpornost na greške
- Rešenja zansovana na tokenima
- Rešenja zansovana na dozvolama

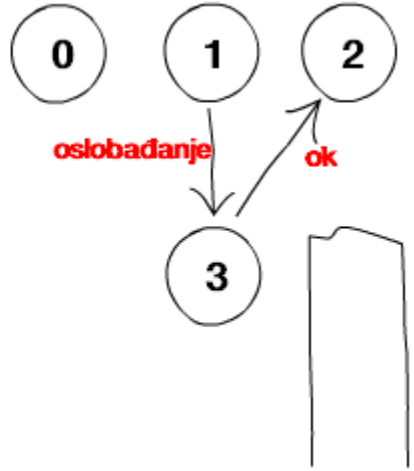
3.1 Centralizovani algoritam

- Bira se jedan koordinator (kako?)
- Procesi šalju zahteve za pristup resursu, koordinator odobrava (ako je resurs trenutno slobodan) ili stavlja u FIFO red (ako nije)
- Kada proces završi sa resursom obaveštava koordinatora

3.1 Centralizovani algoritam







3.1 Centralizovani algoritam

- Prednosti:
  - Fer redosled: FIFO red
  - Nema izgladnjivanja: Niko ne čeka zauvek
  - Jednostavno, samo 3 poruke (zahtev, ok, oslobađanje)
- Mane:
  - *Single point of failure*, ako se koordinator sruši sve pada u vodu; kako razlikovati mrtvog koordinatora od uobičajenog čekanja
  - Centralizovanost sistema može da utiče na performanse, sve ide kroz koordinatora

3.2 Decentralizovani algoritam

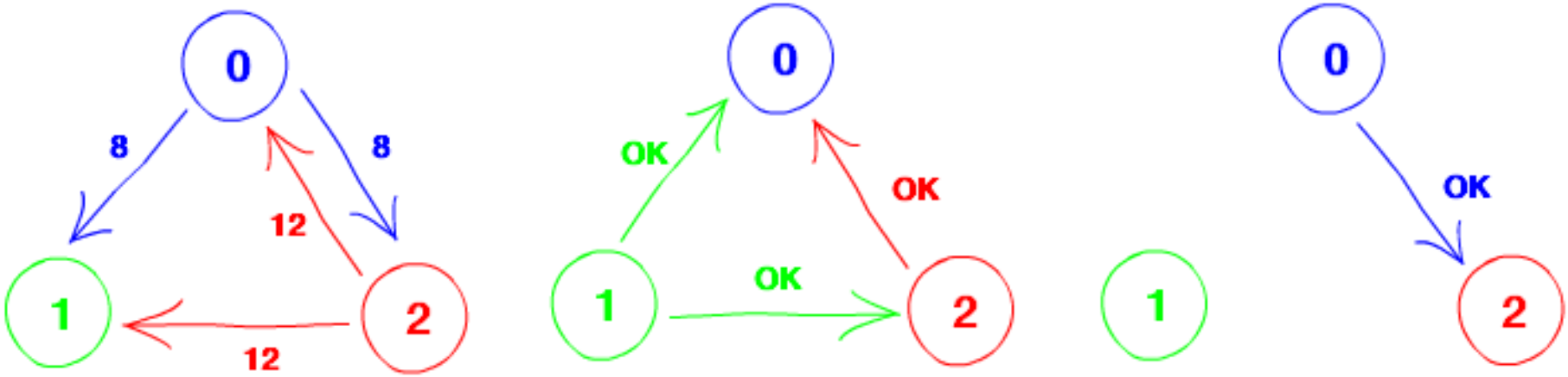
- Sistem glasanja baziran na DHT
- Svaki resurs je repliciran N puta, treba nam dozvola od bar $M > \frac{N}{2}$ procesa koji imaju taj resurs
- Mala šansa greške, čak i ako se individualni procesi sruše
- Mana: ako se mnogo procesa bori za isti resurs nastaje haos

3.3 Distribuirani algoritam

- Možemo bolje od probabilističke varijante
- Distribuirani algoritam zahteva totalni poredak svih događaja u sistemu (modifikacija Lamportovih logičkih časovnika)
- Svaki proces koji želi da pristupi resursu šalje svoj ID, ime resursa i svoje trenutno logičko vreme **svim ostalim procesima**:
  - 1. Proces koji primi zahtev, trenutno ne pristupa traženom resursu i nema nameru da mu pristupi odgovara OK
  - 2. Proces koji primi zahtev a trenutno pristupa traženom resursu ne odgovara ništa i stavlja zahtev u svoj red
  - 3. Proces koji primi zahtev a ima nameru da pristupi istom resursu poredi svoje logičko vreme sa vremenom zahteva; manje vreme pobeđuje (odgovara OK ili stavlja zahtev u red)

3.3 Distribuirani algoritam

- Svaki proces čeka odobrenje od svih drugih kako bi pristupio nekom resursu, a kada završi šalje OK svim procesima iz svog reda



3.3 Distribuirani algoritam

- Šta da je 2 poslao prvi?
- Nema deadlock-a, nema izgladnjivanja, nema *single point of failure*
- Sa druge strane imamo n *points of failure* - ako se iko sruši svi moraju da čekaju
- Zamenili smo loš algoritam n puta lošijim :)
- Primer kada distribuiranost smanjuje otpornost na greške (?!)

3.4 Token Ring algoritam

- Ustanovi se nasumična ciklična struktura, svaki proces zna sledećeg na krugu
- Proces 0 dobije jedinstveni token koji cirkuliše u krug
- Ko želi da pristupi resursu zadrži token dok ne završi (ne sme dva puta zaredom!)
- Problem?

3.4 Token Ring algoritam

- Ustanovi se nasumična ciklična struktura, svaki proces zna sledećeg na krugu
- Proces 0 dobije jedinstveni token koji cirkuliše u krug
- Ko želi da pristupi resursu zadrži token dok ne završi (ne sme dva puta zaredom!)
- Problem?
- Ako se proces koji ima token sruši to treba detektovati (teško?) i napraviti novi (jedinstven?) token
- Ako se proces koji nema token sruši prethodni proces šalje token „u prazno”
  - Fix: tražimo da proces potvrdi prijem tokena, ako ne potvrdi šaljemo sledećem na krugu (u ovom slučaju moramo da imamo celu konfiguraciju na svakoj mašini)

Algoritmi izbora: Ko je tu glavni?

- Algoritmi izbora koordinatora/vođe; svi procesi su jednaki ali neki moraju da budu jednakiji i preuzmu odgovornost
- Izazov: Složiti se oko vođe, obavestiti sve čvorove, prevazići činjenicu da su neki *down*
- Pretpostavimo da svaki proces ima dodeljen jedinstven, globalno poznat, prirodan broj - ID
- Algoritmi se uglavnom svode na nalaženje procesa sa najvećim ID

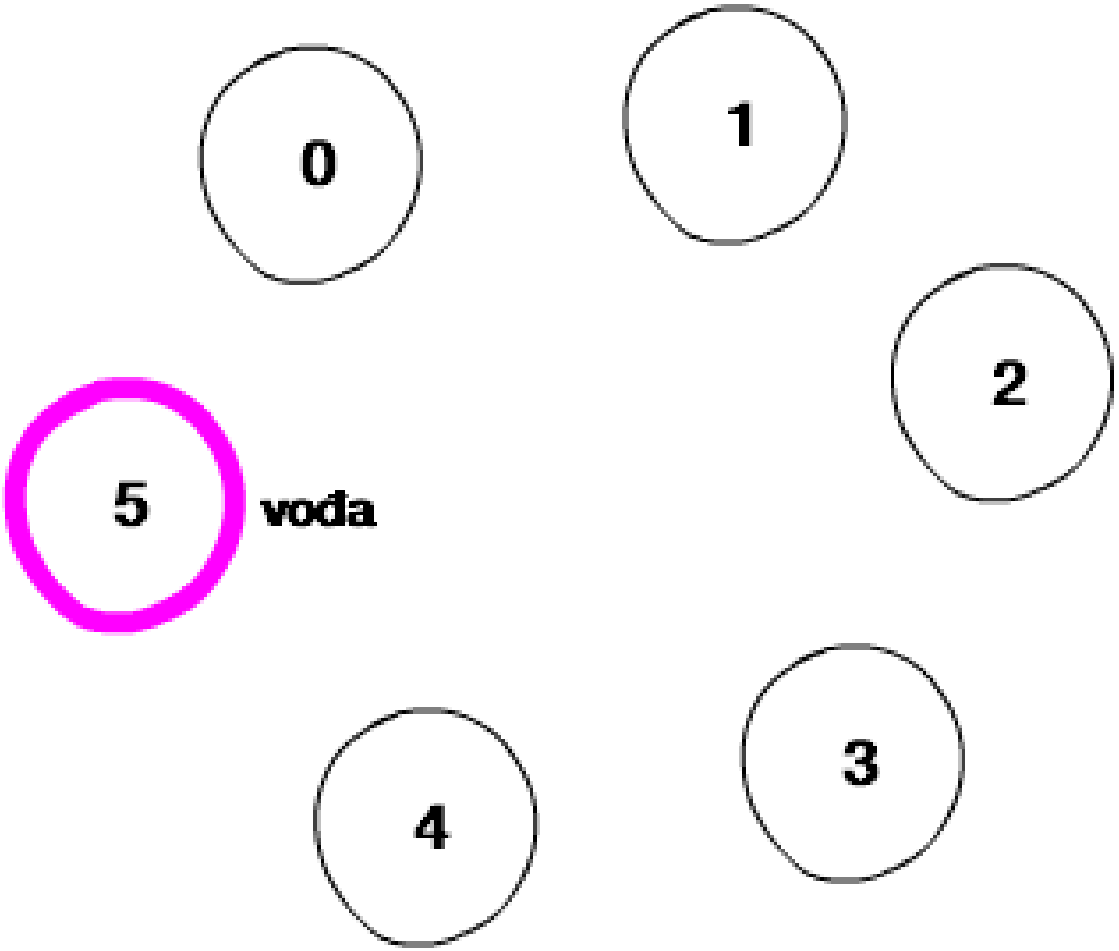
4.1 Bully algoritam

- Kada proces P primeti da je trenutni vođa *crash*-ovao šalje svim procesima koji imaju veći ID poruku $IZBORI$:
  - Ukoliko niko ne odgovori P je novi vođa i obaveštava ostale porukom $VODJA$
  - Ukoliko neko odgovori P odustaje i čeka da se novi vođa javi
- Kada proces Q primi poruku $IZBORI$ od procesa P:
  - Ukoliko važi $ID(P) > ID(Q)$ proces Q ne odgovara
  - U suprotnom proces Q šalje poruku OK i započinje nove izbore

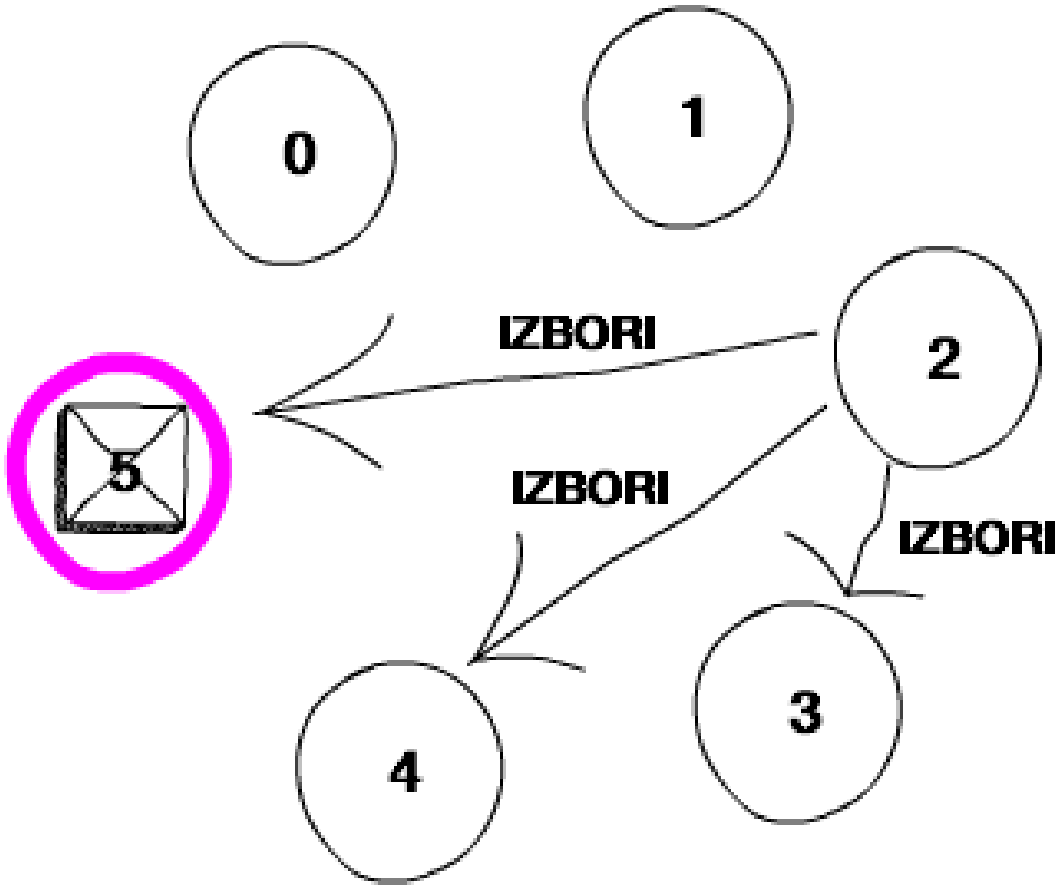
4.1 Bully algoritam

- Ako se stari koordinator ikad vrati ponovo će pokrenuti izbore i pobediti
- Bully (siledžija): najveći proces ućutka ostale
- Složenost: $\mathcal{O}(n^2)$
- Ako više procesa primeti odsustvo vođe u isto vreme to ne pravi problem

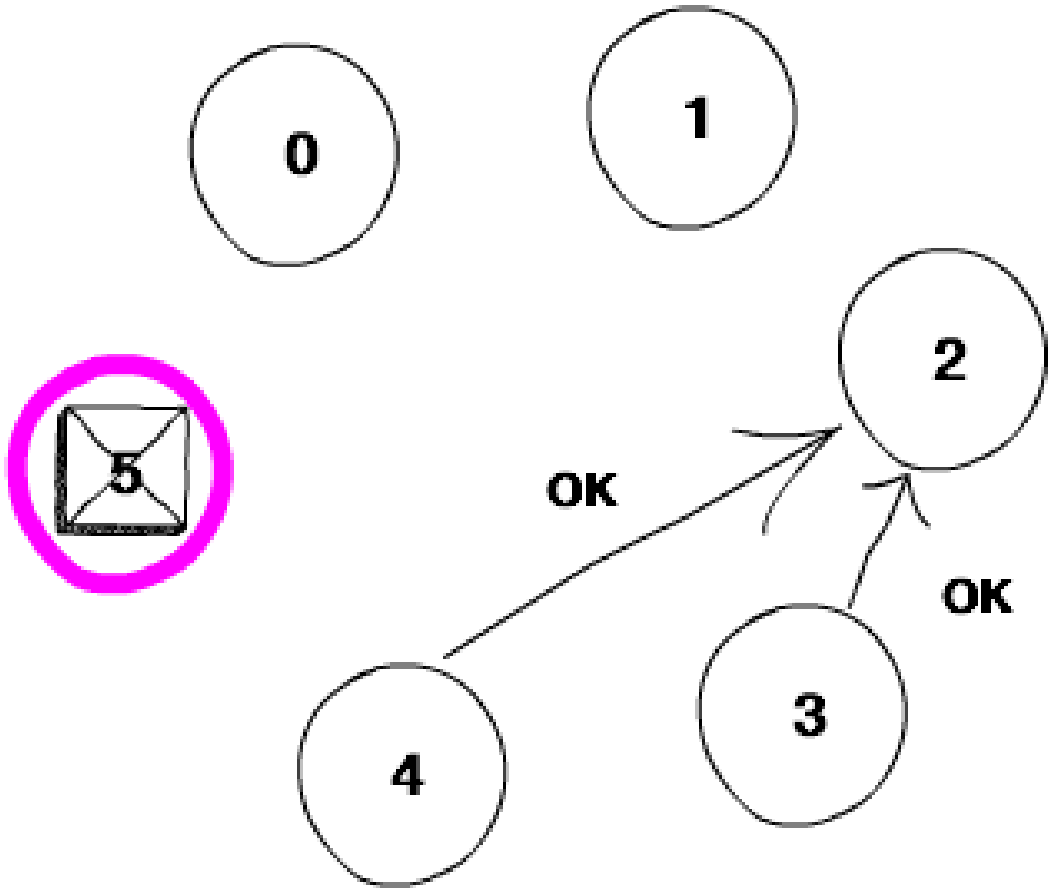
4.1 Bully algoritam



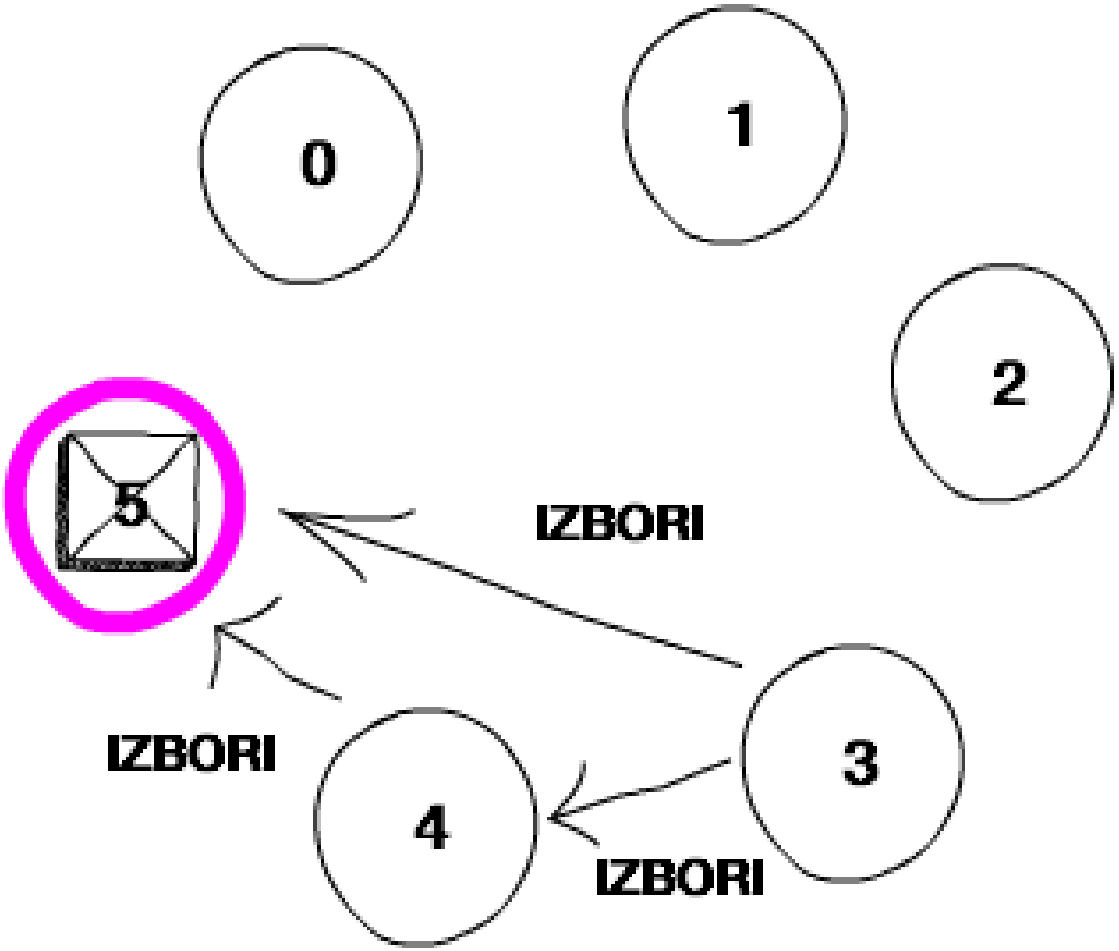
4.1 Bully algoritam



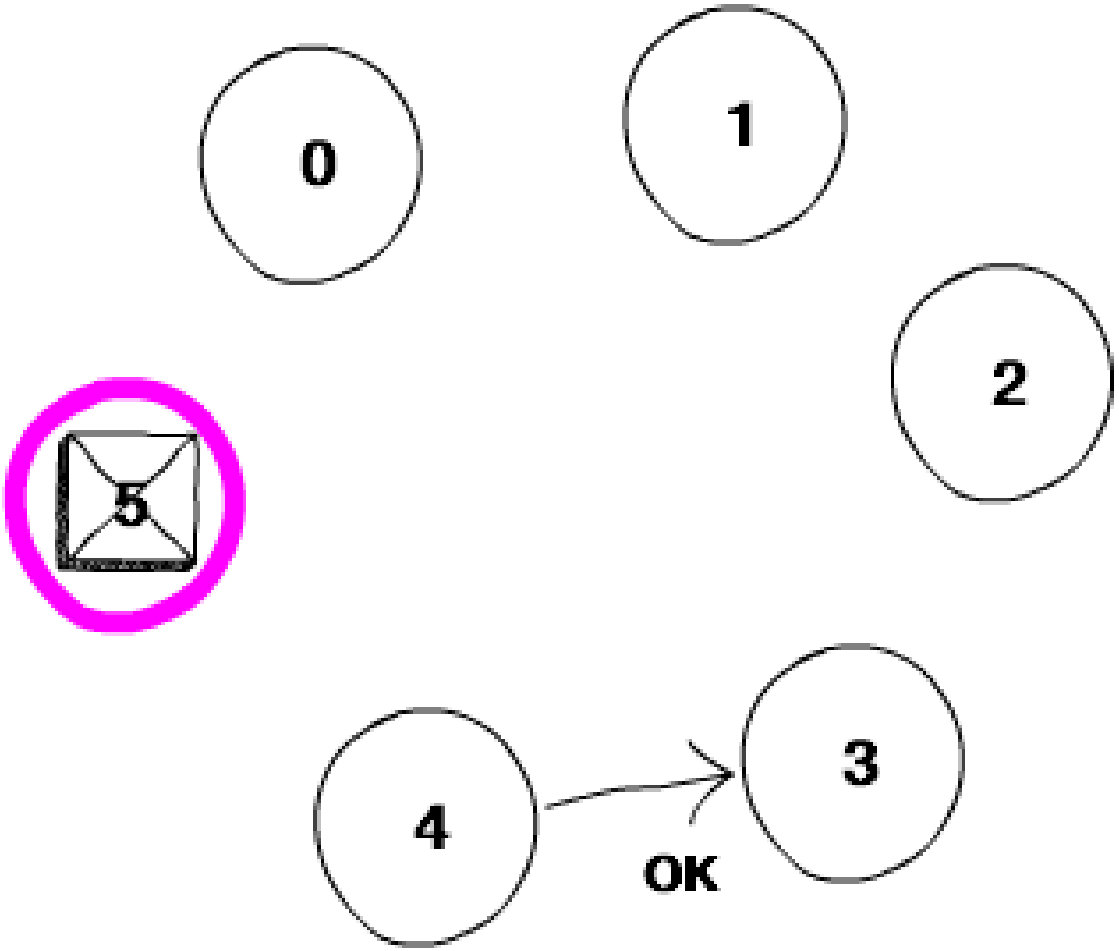
4.1 Bully algoritam



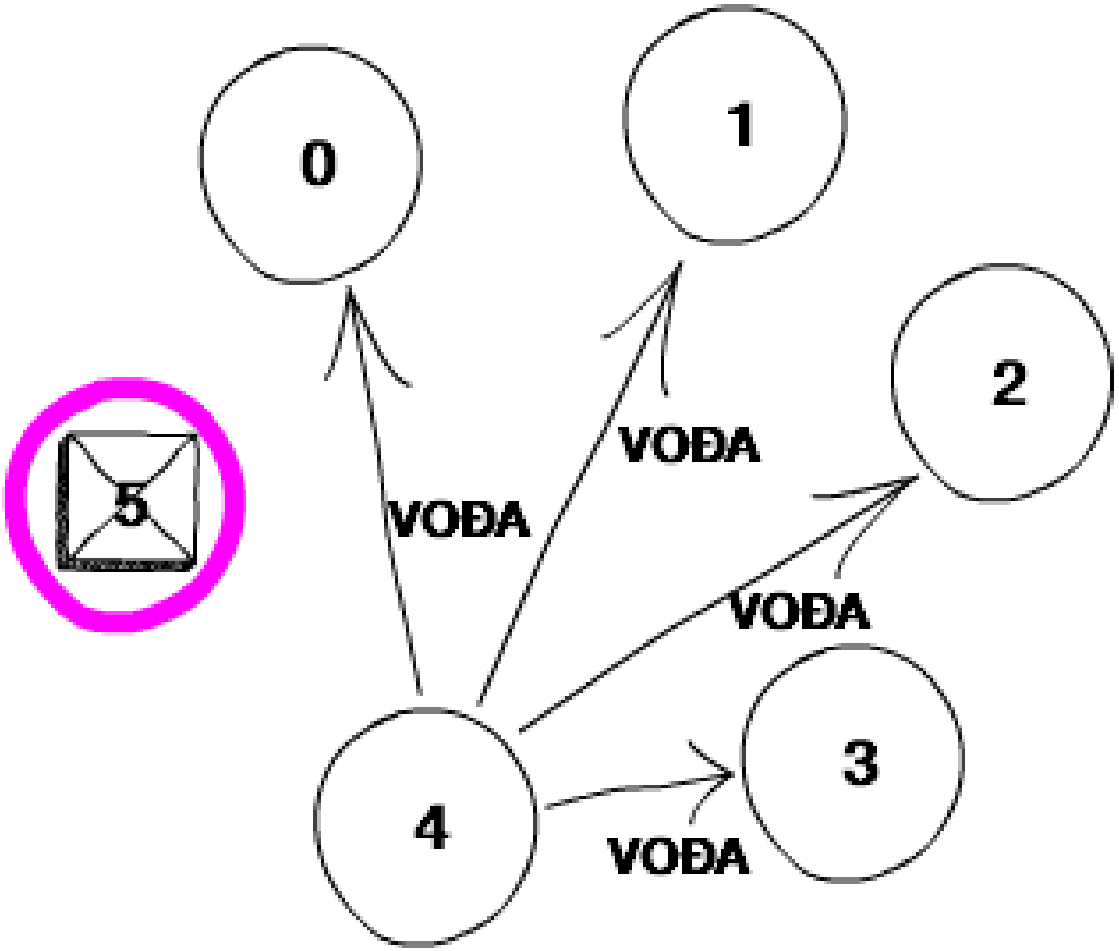
4.1 Bully algoritam



4.1 Bully algoritam



4.1 Bully algoritam



4.2 Ring algoritam

- Pravimo logički ciklus (ovaj put bez tokena), svako zna ID sledećeg procesa na krugu
- Kada proces P primeti da je trenutni vođa *crash*-ovao šalje poruku $IZBORI$ i svoj ID prvom sledećem procesu na krugu koji je živ
- Kada neki proces Q primi tuđu poruku $IZBORI$ dodaje u nju svoj ID

4.2 Ring algoritam

- Kada proces P primi sopstvenu poruku $IZBORI$ zna da su se svi kandidati upisali, izdvaja samo onog sa najvećim ID i prosleđuje poruku $VODJA$ dalje (kako bi informisao ostale da je odabran novi vođa)
- Složenost $\mathcal{O}(n)$
- Ako više procesa primeti odsustvo vođe u isto vreme to ne pravi problem

4.3 Izbori u wireless mrežama

- Do sad smo pravili dve pretpostavke (pouzdanost konekcije, statična topologija) koje nisu realistične u wireless mrežama
- Takođe, u wireless mrežama često imamo neki kriterijum po kom biramo vođu (npr. mobilni uređaj sa trenutno najviše baterije)

4.3 Izbori u wireless mrežama

- Predloženi algoritam - stablo pretrage:
  - Bilo koji čvor S započinje izbor tako što svim susedima šalje poruku $IZBORI$
  - Svaki čvor Q koji prvi put dobije poruku $IZBORI$ zabeleži pošiljaoca kao roditelja i širi poruku dalje
  - Svaki čvor Q koji ponovo dobije poruku $IZBORI$ šalje OK
  - Kada svi susedi P (osim roditelja) kažu OK, P vraća OK svom roditelju uz informaciju o najboljem kandidatu za kog zna (svaki čvor kombinuje saznanja svoje dece)
  - Na samom kraju S će znati koji čvor je najbolji kandidat za vođu (npr. ima najviše baterije) i objaviće tu informaciju svima

4.3 Izbori u wireless mrežama

- Složenost $\mathcal{O}(n)$
- Ako više procesa primeti odsustvo vođe u isto vreme to ne pravi problem, čvor učestvuje samo u izborima koje je pokrenuo čvor sa najvećim ID

4.4 Izbori u geometrijski modeliranim mrežama

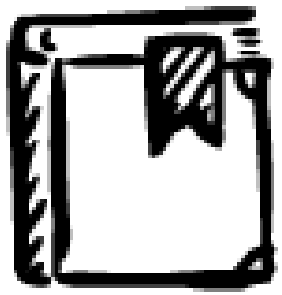
- Biramo N *superpeer*-ova
- Pustimo u promet N „odbojnih” tokena
- Algoritmima tračarenja širimo informaciju o silama
- Jaka sila može da „izbije” token iz nekog čvora
- Tokeni (intuitivno) završe raspoređeni po celom sistemu

Zaključci

- Postoje mnogi neočekivani izazovi
- Postoje mnoga neočekivana rešenja za neočekivane izazove
- Postoje mnogi neočekivani problemi sa neočekivanim rešenjima za neočekivane izazove

Šta dalje

- Tanenbaum i Van Steen, Distributed Systems: Principles and Paradigms
  - Pažljivo!
- code.google.com/codejam



Hvala na pažnji!

- Nemam XKCD za ovu priliku - gledajte u random sliku sa Google Images!