10. Election algoritmi

Proces odabira voditelja u raspodijeljenom sustavu se odnosi na odabir jednog procesa kojemu se pridjeljuje uloga organizatora za obavljanje zadtka raspodijeljenog na više ćvorova. Prije nego zadatak počme, čvorovi ne znaju koji od njih će biti organizator ili nisu u mogućnosti komunicirati s trenutnim organizatorom. Izvršenjem election algoritma mora biti odabran jedan organizator te ostali čvorovi moraju biti upoznati s odabirom.

10.1 Odabir voditelja

U algoritmu izbora (engl. *Election algorithm*), procesi među sobom odabiru jednog koji će dijelovati kao voditelj. Ovaj proces najšesće djeluje kao organizator izvršavanja raspodijeljenog zadatka, a može poslućiti i kao korijen najmanjeg stabla koje spaja sve čvorove te pokretač nekog centraliziranog algoritma. Kako bi se voditelj odabrao, čvorovi moraju biti u mogućnosti međusobno komunicirati.

Algoritam odabira voditelja može inicirati jedan ili više procesa, ali zahtjevamo da svaki proces ima isti lokalni algoritam. Da bi algoritam funkcionirao svaki čvor mora znati je li algoritam gotov te je li on sam odabrani voditelj.

Svaki algoritam za odabir voditelja se mora izvršiti u konačnom broju koraka te svaki proces mora biti upoznat s time kada je algoritam gotov te koji je proces odabran za voditelja, od čega mora biti točno jedan.

Ovisno o tome kakvom strukturom su povezani čvorovi, imamo različite algoritme za odabir voditelja:

- prestenasta struktura - LeLann-Chang-Roberts (LCR) algoritam, Franklinov algoritam, Hirschberg i Sinclair (HS) algoritam
- mesh
- hiperkocka
- algoritam za odabir voditelja u stablu - *tree election algorithm*

Ovdje ćemo se fokusirati na odabir voditelja u među čvorovima spojenim u prsten.

10.2 Algoritmi prstenaste strukture

Kako bi ovi algoritmi funkcionirali svaki proces mora imati jedinstven ID iz potpuno uređenog skupa. U svakom od algoritama koje ćemo navesti veličina prstena N nije poznata čvorovima te se za voditelja odabire čvor s najvećim ID-om. Svi ostali čvorovi moraju znati koji čvor je odabran za voditelja.

Inicijalno, procesi koji iniciraju algoritam su kandidati za voditelja - može biti jedan čvor, nekoliko ili svi čvorovi. Na početku ovih algoritama za sve inicijatore kažemo da su aktivni, a neinicijatore da su pasivni čvorovi. Pasivni čvorovi nisu u utrci da postanu voditelji.

10.2.1 LeLann-Chang-Roberts (LCR) algoritam

Kod ovog algoritma pretpostavljamo usmjereni prsteni - poruke će ići u jednome smjeru. Algoritam funkcionira na sljedeći način:

- Jedan ili više čvorova preuzima inicijativu te inicira algoritam na način da čvor pi pošalje *election* poruku sa svojim ID čvoru P_{i+1}.
- U slučaju da p_i primi poruku koja sadrži ID koji je veći od njegovog, prosljedi poruku s ID-om kojeg je primio.
- U slučaju da p_i primi poruku koja sadrži ID koji je manji od njegovog, a ujedno nije aktivan, proces postaje aktivan i pošalje *election* poruku sa svojim ID-om.
- U slučaju da p_i primi poruku koja sadrži ID koji je manji od njegovog, ali je p_i već aktivan, onda proces odbacuje poruku.
- U slučaju da p_i porukom primi svoj ID, onda se počinje ponašati kao voditelj.

Proces koji je postao voditelj se počme ponašati na sljedeći način:

- Sebe označi kao pasivni čvor te pošalje *elected* poruku sa svojim ID-om svom susjedu.
- Proces koji primi *elected* poruku sebe označi kao pasivnog te prosljedi istu poruku dalje svome susjedu.
- Kada voditelj primi *elected* poruku, dakle poruka je napravila puni krug, odbaci poruku i time je algoritam gotov.

Pseudokod algoritma je dan ispod (10.1).

Primjer 10.1: LCR ukratko

```
boolean participant = false;
int leader_id = null;

Za pokrenuti odabir:
        send(ELECTION<my_id>);
        participant := true;

Po primitku poruke ELECTION<j>:
        if(j > my_id) then send(ELECTION<j>);
        if(j = my_id) then send(LEADER<my_id>);
        if((my_id > j) ∧ (¬participant) then
                send(ELECTION<my_id>);
        participant := true;

Po primitku poruke poruke LEADER<j>:
        leader_id = j;
        if(my_id ≠ j ) then send(LEADER<j>);
```

10.2.2 Hirschberg i Sinclair (HS) algoritam

Kod ovog algoritma odabir voditelja se vrši po k-susjedstvima nekog čvora p_i. k-susjedstvo čvora p_i je skup čvorova koji su najviše k udaljeni od čvora p_i - k lijevih i k desnih susjeda.

Algoritam djeluje u asinkronim fazama: čvor p_i nastoji postati voditelj u fazi k među svojih 2^k susjeda; samo ako je čvor p_j pobijednik u danoj fazi, tj. ima najveći ID u svom 2^k susjedstvu, može nastaviti u fazu $k+1$. Stoga se veličina susjedstva podupla u svakoj fazi. Sve manje čvorova napreduje u sljedeću fazu sve dok samo jedan pobjednik ne ostane u cijelom prstenu.

Inicijalno svi čvorovi p_i iniciraju kandidaturu (faza 0), npr. nakon što dobiju zahtjev za odabiranje voditelja.

ELECTION poruka koju kandidati šalju se sastoji od tri polja:

- ID kandidata.
- Trenutnog broja faze k.
- Brojač d, koji je inicijalno na početku svake faze 0, ali se inkrementira za 1 svaki put kada se poruka prosljedi sljedećem čvoru 2^k susjedstva.

Algoritam funkcionira na sljedeći način:

- Ako čvor primi poruku ELECTION<i, k, d> gdje je $d = 2^k$ onda je to posljednji čvor 2^k susjedstva čvora p_i s ID-om $ID = i$.
- Ako čvor p_i koji prima ELECTION poruku ima veći ID, onda proguta poruku, a u suprotnom inkrementira d za 1.
- Ako poruka stigne do čvora p_j udaljenog za 2^k od čvora p_i, onda p_j pošalje natrag poruku REPLY koja se prosljeđuje dok ne stigne do čvora p_i.
- Ako čvor p_i primi REPLY poruke iz oba smjera, onda je pobjednik svog 2^k susjedstva.
- Čvor p_i koji primi ELECTION poruku sa svojim ID-om, onda postaje voditelj prstena.
- Odabrani čvor se predstavi svim ostalim čvorovima kao voditelj.

Primjer 10.2: HS algoritam ukratko

```

Da bi se inicirao postupak odabira (faza 0):
```

```
3           send(ELECTION<my_id, 0, 0>) to left and right;
4
5   Po primitku poruke ELECTION<j, k, d> from left (right):
6           if((j > my_id) ∧ (d < 2^k)) then
7                   send(ELECTION<j, k, d + 1>) to right (left);
8           if((j > my_id) ∧ (d = 2^k)) then
9                   send(REPLY<j, k>) to left (right);
10          if(my_id = j) then announce itself as leader;
11
12  Po primitku poruke REPLY<j, k> from left (right):
13          if(my_id ≠ j) then
14                  send(REPLY<j, k>) to right (left);
15          else
16                  if(already received REPLY<j, k>)
17                          send(ELECTION<j, k + 1, 1>) to left
                               (right);
```

10.3 Zadatci

1. Proučite kako funkcionira bully algoritam za odabir voditelja te ga implementirajte koristeći Akka.NET.

10.4 Izvori

1. http://www.cs.rug.nl/~eirini/DS_slides/leader_election.pdf, 24.05.2016. u 21:45.
2. https://en.wikipedia.org/wiki/Distributed_minimum_spanning_tree, 24.05.2016. u 21:45.
3. https://en.wikipedia.org/wiki/Leader_election, 24.05.2016. u 21:45.
4. http://www.cs.utexas.edu/~lorenzo/corsi/cs380d/past/01F/notes/leader2.ps, 24.05.2016. u 21:45.
5. https://en.wikipedia.org/wiki/Chang_and_Roberts_algorithm, 24.05.2016. u 21:45.
6. *Distributed Algorithms: An intuitive approach*, Wan Fokkink