

Chang Roberts

const: ID : integer
 l_{in}, l_{out} : link
var: $leader$: integer

Init:

$leader := NULL$

Code:

send $\langle ID \rangle$
wait until $leader \langle \neq \rangle NULL$

On receipt $\langle i \rangle$:

if $i < ID$ **then send** $\langle i \rangle$
if $i = ID$ **then**
 $leader := ID$
 send $\langle leader, ID \rangle$

On receipt $\langle leader, x \rangle$:

$leader := x$
send $\langle leader, ID \rangle$

$O(n \log n)$ správ v priemernom prípade, $\Omega(n^2)$ v najhoršom

Hirschberg-Sinclair

- level l : dobýjať územie 2^l
- $\log n$ levelov
- $n/2^l$ vrcholov na leveli
- každý vrchol pošle 2^l správ

Franklin

- level l : poraziť susedov (na rovnakom leveli; "synchronizácia")
- $\log n$ levelov
- n správ na level

Dolev – simulácia na 1-smernom kruhu

- idea: presunúť identitu

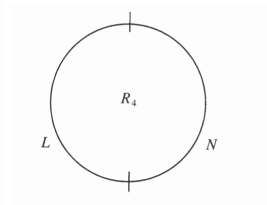
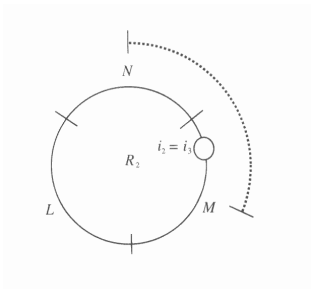
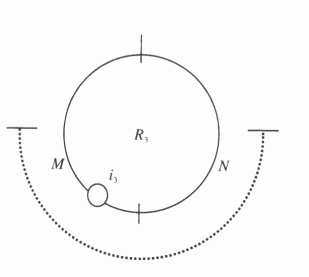
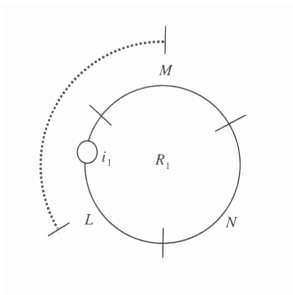
zapojiť do čiary L , vymení sa $C(L)$ správ

lema

Pre každé r existuje nekonečne veľa čiar dĺžky 2^r , kde $C(L) \geq r2^{r-2}$

- indukcia
- dve vrecia: vyberám trojice, chcem dve spojiť item pri spojení: dĺžka 2^{r+1} , počet správ $\geq r2^{r-1}$
- ešte treba 2^{r-1} správ; sporom

dolný odhad



- ľubovoľná topológia
- buduje sa kostra
- “segmenty”
- spájanie po najlacnejšej odchádzajúcej hrane:
 - A menší sa pripojí k väčšiemu
 - B rovnakí sa spoja
 - C väčší čaká
- veľkosť \Rightarrow level

```

var  $state_p$  :  $(sleep, find, found)$  ;
       $stach_p[q]$  :  $(basic, branch, reject)$  for each  $q \in Neigh_p$  ;
       $name_p, bestwt_p$  : real ;
       $level_p$  : integer ;
       $testch_p, bestch_p, father_p$  :  $Neigh_p$  ;
       $rec_p$  : integer ;

```

(1) As the first action of each process, the algorithm must be initialized:

```

begin let  $pq$  be the channel of  $p$  with smallest weight ;
       $stach_p[q] := branch$  ;  $level_p := 0$  ;
       $state_p := found$  ;  $rec_p := 0$  ;
      send  $\langle connect, 0 \rangle$  to  $q$ 
end

```

(2) Upon receipt of $\langle connect, L \rangle$ from q :

```

begin if  $L < level_p$  then (* Combine with Rule A *)
      begin  $stach_p[q] := branch$  ;
          send  $\langle initiate, level_p, name_p, state_p \rangle$  to  $q$ 
      end
      else if  $stach_p[q] = basic$ 
          then (* Rule C *) process the message later
          else (* Rule B *) send  $\langle initiate, level_p + 1, \omega(pq), find \rangle$  to  $q$ 
end

```

(3) Upon receipt of $\langle initiate, L, F, S \rangle$ from q :

```

begin  $level_p := L$  ;  $name_p := F$  ;  $state_p := S$  ;  $father_p := q$  ;
       $bestch_p := undef$  ;  $bestwt_p := \infty$  ;
      forall  $r \in Neigh_p$  :  $stach_p[r] = branch \wedge r \neq q$  do
          send  $\langle initiate, L, F, S \rangle$  to  $r$  ;
      if  $state_p = find$  then begin  $rec_p := 0$  ; test end
end

```

```

(4) procedure test:
  begin if  $\exists q \in \text{Neigh}_p : \text{stach}_p[q] = \text{basic}$  then
    begin testchp := q with  $\text{stach}_p[q] = \text{basic}$  and  $\omega(pq)$  minimal ;
      send  $\langle \text{test}, \text{level}_p, \text{name}_p \rangle$  to testchp
    end
  else begin testchp := undef ; report end
end

```

```

(5) Upon receipt of  $\langle \text{test}, L, F \rangle$  from q:
  begin if  $L > \text{level}_p$  then (* Answer must wait! *)
    process the message later
  else if  $F = \text{name}_p$  then (* internal edge *)
    begin if  $\text{stach}_p[q] = \text{basic}$  then  $\text{stach}_p[q] := \text{reject}$  ;
      if  $q \neq \text{testch}_p$ 
        then send  $\langle \text{reject} \rangle$  to q
        else test
      end
    end
  else send  $\langle \text{accept} \rangle$  to q
end

```

```

(6) Upon receipt of  $\langle \text{accept} \rangle$  from q:
  begin testchp := undef ;
    if  $\omega(pq) < \text{bestwt}_p$ 
      then begin  $\text{bestwt}_p := \omega(pq)$  ;  $\text{bestch}_p := q$  end ;
    report
  end

```

```

(7) Upon receipt of  $\langle \text{reject} \rangle$  from q:
  begin if  $\text{stach}_p[q] = \text{basic}$  then  $\text{stach}_p[q] := \text{reject}$  ;
    test
  end

```

-
- (8) **procedure** *report*:
- ```

begin if $rec_p = \#\{q : stach_p[q] = branch \wedge q \neq father_p\}$
 and $testch_p = undef$ then
 begin $state_p := found$; send $\langle report, bestwt_p \rangle$ to $father_p$ end
 end

```
- (9) Upon receipt of  $\langle report, \omega \rangle$  from  $q$ :
- ```

begin if  $q \neq father_p$ 
  then (* reply for initiate message *)
    begin if  $\omega < bestwt_p$  then
      begin  $bestwt_p := \omega$  ;  $bestch_p := q$  end ;
       $rec_p := rec_p + 1$  ; report
    end
  else (*  $pq$  is the core edge *)
    if  $state_p = find$ 
      then process this message later
    else if  $\omega > bestwt_p$ 
      then changeroot
    else if  $\omega = bestwt_p = \infty$  then stop
  end
end

```
- (10) **procedure** *changeroot*:
- ```

begin if $stach_p[bestch_p] = branch$
 then send $\langle changeroot \rangle$ to $bestch_p$
 else begin send $\langle connect, level_p \rangle$ to $bestch_p$;
 $stach_p[bestch_p] := branch$
 end
end

```
- (11) Upon receipt of  $\langle changeroot \rangle$ :
- ```

begin changeroot end

```


správnosť

ukázať, že sa zvolí práve jeden šéf: nenastane deadlock

počet správ

- testovacie správy: jeden test po každej hrane
- kostrové správy: fragment s n_i vrcholmi pri postupe o level $O(n_i)$ správ
- postupy na level l : dizjunktné vrcholy

- $f(x)$ -traverzovanie
- tokeny traverzujú/označujú územia
- levely: keď sa stretnú dva, vznikne nový
- naháňanie

```

var  $lev_p$       : integer    init -1 ;
     $cat_p, wait_p$  :  $\mathcal{P}$       init udef ;
     $last_p$       :  $Neigh_p$   init udef ;

begin if  $p$  is initiator then
  begin  $lev_p := lev_p + 1$  ;  $last_p := trav(p, lev_p)$  ;
       $cat_p := p$  ; send  $\langle annex, p, lev_p \rangle$  to  $last_p$ 
  end ;
  while ... (* Termination condition, see text *) do
    begin receive token  $(q, l)$  ;
      if token is annexing then  $t := A$  else  $t := C$  ;
      if  $l > lev_p$  then (* Case I *)
        begin  $lev_p := l$  ;  $cat_p := q$  ;
             $wait_p := udef$  ;  $last_p := trav(q, l)$  ;
            send  $\langle annex, q, l \rangle$  to  $last_p$ 
        end
      else if  $l = lev_p$  and  $wait_p \neq udef$  then (* Case II *)
        begin  $wait_p := udef$  ;  $lev_p := lev_p + 1$  ;
             $last_p := trav(p, lev_p)$  ;  $cat_p := p$  ;
            send  $\langle annex, p, lev_p \rangle$  to  $last_p$ 
        end
      else if  $l = lev_p$  and  $last_p = udef$  then (* Case III *)
         $wait_p := q$ 
      else if  $l = lev_p$  and  $t = A$  and  $q = cat_p$  then (* Case IV *)
        begin  $last_p := trav(q, l)$  ;
            if  $last_p = decide$ 
              then  $p$  announces itself leader
            else send  $\langle annex, q, l \rangle$  to  $last_p$ 
        end
      else if  $l = lev_p$  and ( $t = A$  and
         $q > cat_p$ ) or  $t = C$ ) then (* Case V *)
        begin send  $\langle chase, q, l \rangle$  to  $last_p$  ;  $last_p := udef$  end
      else if  $l = lev_p$  then (* Case VI *)
         $wait_p := q$ 
    end
  end

```

end

počet správ

- naháňacie: 1 na vrchol a level, spolu n za level
- objavovacie: $\sum_i f(n_i)$
- ak f je konvexná, t.j. $f(a) + f(b) \leq f(a + b)$, tak je $O(\log n(n + f(n)))$ správ

K_n : vplyv orientácie

zmysel pre orientáciu

Porty sú číslované podľa vzdialenosti vo fixnej Hamiltonovej kružnici.

- algoritmus: zajať fixný pattern $\{i[1..k], i[2k], i[3k], \dots, i[n-k]\}$
- prvá fáza $i[1..k]$
 - *level*, *ID*
 - *level* je počet zajatých
 - pripája sa celé územie (planarita!)
- druhá fáza $i[2k], i[3k], \dots, i[n-k]$
 - nastav *owner* vrcholom $i[1..k]$, *ack*
 - pošli *elect* do $i[2k], i[3k], \dots, i[n-k]$
 - *elect*: ak je v prvej fáze, alebo je slabší \Rightarrow accept

počet správ

- prvá fáza: $O(n)$: z planarity
- druhá fáza: max $O(n/k)$ kandidátov, každý pošle n/k správ $\Rightarrow O(n^2/k^2)$ správ

čas

- po zobudení (najsilnejšieho) $O(k)$, v najhoršom $O(n)$
- pridať wakeup fázu (poslať od $i[1], i[k]$) $\Rightarrow O(k + n/k)$
- $k := \sqrt{n}$

vylepšit' čas

- prvá fáaza: zajať $i[k], i[2k], \dots, i[n - k]$ sekvenčne
- vzniknú nezávislé "kruhy" R_0, \dots, R_{k-1} , $R_j = \{i[j + k], i[j + 2k], \dots, i[j + n - k]\}$
- druhá fáaza: zajať $i[1..k - 1]$ v $\log k$ krokoch
 - level, ID
 - v prvom kroku zajať $i[k/2]$
 - v l -tom kroku zajať $i[k/2^l], i[3k/2^l], \dots, i[(2^l - 1)k/2^l]$ ako v úplnom grafe (za zajatých bojuje šéf)
 - v l -tom kroku je $k/2^l$ kandidátov; je $\log n$ krokov $\Rightarrow O(k \log n)$ správ
- $k := n / \log n$

algoritmy založené na porovnaníach

- ekvivalentné okolia
- c -symetrický reťazec: pre každé $\sqrt{n} \leq l \leq n$ a pre každý segment S je $\lfloor \frac{cn}{l} \rfloor$ ekvivalentných
- bit-reversal je $1/2$ -symetrický
- c -symetrický reťazec, alg. nemôže skončiť po k kolách pre $\lfloor \frac{cn}{2k+1} \rfloor \geq 2$
- počet správ: $k = \lfloor \frac{cn-2}{4} \rfloor$, je aspoň $k + 1$ kôl
- aspoň $\lfloor \frac{cn}{2r-1} \rfloor$ aktívnych v r -tom kole pošle správu

algoritmy využívajúce integer ID

- rôzne rýchlosti
- v i -tom kroku test

V toruse rozmerov $n \times n$ (t.j. zacyklenej mriežke) sú na začiatku zobudené dva vrcholy. Napíšte algoritmus, pomocou ktorého sa každý z nich dozvie identifikátor druhého s použitím $O(n)$ správ.

Nájdite asymptoticky optimálny (čo do počtu správ) algoritmus na voľbu šéfa v úplnom bipartitnom grafe $K_{n,n}$. Dokážte jeho zložitosť a optimalitu.

★

broadcasting a voľba šéfa na (ne?)orientovanej hyperkočke s lineárnym počtom správ