

Sieťový model

- nezávislé zariadenia (procesy, procesory, uzly)
- posielanie správ
- lokálny pohľad (číslovanie portov)
- asynchrónne, spoľahlivé správy
- topológia siete
- wakeup/terminácia

zložitosť: v závislosti od počtu procesorov

- počet správ / bitov
- čas (beh normovaný na max. dĺžku 1)

horný odhad pre problém

Existuje algoritmus, ktorý **pre všetky** topológie, vstupy, časovania,
... pracuje správne a vymení najviac $f(n)$ správ

dolný odhad pre problém

Pre každý algoritmus, **existuje kombinácia** topológie, vstupu, časovania,
... že buď nepracuje správne alebo vymení aspoň $f(n)$ správ

alternating bit

- A opakovane posielala správu so "sequence number" 0
- B začne posielat' ACK0
- A prejde na sequence number 1

problém: latencia

sliding window

- A posielala naraz niekoľko správ (frames)
- keď dostane ACK, posunie "okno"

dvojsmerné vyvážené posúvanie okna

- konštanta l – veľkosť okna
- i -ty frame sa môže poslať, keď sa doručili $0..i - l$
- zároveň je to acknowledgement

dvojsmerné vyvážené posúvanie okna

- a – prvý (odoslaný) frame, na ktorý neprišiel ACK
- s – prvý neprijatý frame
- posiela sa z intervalu $a \leq i < s + l$
- pri prijatí i : $a := \max(a, i - l + 1)$

fairnes

- na začiatku je jeden iniciátor
- treba informovať všetkých

shout-and-echo

- iniciátor pošle **shout**
- ak príde **shout** do nového vrchola
 - označí hranu
 - pošle po neoznačených **shout**
 - čaká na všetky **echo**
 - pošle **echo**
- ak príde **shout** do navštíveného – okamžite **echo**

zložitosť

- $4m$ správ
- $2diam(G)$ čas

traverzovanie

- na začiatku je jeden iniciátor
- treba informovať všetkých
- v každom okamihu je len jeden *token*

$f(x)$ -traverzovanie

Na objavenie x vrcholov treba $\max\{f(x), n\}$ správ.

priamočiaro

```
var  $used_p[q]$  : boolean    init false for each  $q \in Neigh_p$  ;  
                                (* Indicates whether  $p$  has already sent to  $q$  *)  
     $father_p$  : process    init undef ;
```

For the initiator only, execute once:

```
begin  $father_p := p$  ; choose  $q \in Neigh_p$  ;  
         $used_p[q] := true$  ; send  $\langle tok \rangle$  to  $q$   
end
```

For each process, upon receipt of $\langle tok \rangle$ from q_0 :

```
begin if  $father_p = undef$  then  $father_p := q_0$  ;  
        if  $\forall q \in Neigh_p : used_p[q]$   
            then decide  
        else if  $\exists q \in Neigh_p : (q \neq father_p \wedge \neg used_p[q])$   
            then begin if  $father_p \neq q_0 \wedge \neg used_p[q_0]$   
                then  $q := q_0$   
                else choose  $q \in Neigh_p \setminus \{father_p\}$   
                    with  $\neg used_p[q]$  ;  
                 $used_p[q] := true$  ; send  $\langle tok \rangle$  to  $q$   
            end  
        else begin  $used_p[father_p] := true$  ;  
            send  $\langle tok \rangle$  to  $father_p$   
        end  
end
```

- zložitosť $2m$ (čas aj správy)
- ako ušetriť čas?

Awerbuch: $4m$ správ, $4n - 2$ čas

```
var  $used_p[q]$  : boolean    init false for each  $q \in Neigh_p$  ;  
                                   (* Indicates whether  $p$  has already sent to  $q$  *)  
 $father_p$  : process    init  $undef$  ;
```

For the initiator only, execute once:

```
begin  $father_p := p$  ; choose  $q \in Neigh_p$  ;  
    forall  $r \in Neigh_p$  do send  $\langle vis \rangle$  to  $r$  ;  
    forall  $r \in Neigh_p$  do receive  $\langle ack \rangle$  from  $r$  ;  
     $used_p[q] := true$  ; send  $\langle tok \rangle$  to  $q$   
end
```

For each process, upon receipt of $\langle tok \rangle$ from q_0 :

```
begin if  $father_p = undef$  then  
    begin  $father_p := q_0$  ;  
        forall  $r \in Neigh_p \setminus \{father_p\}$  do send  $\langle vis \rangle$  to  $r$  ;  
        forall  $r \in Neigh_p \setminus \{father_p\}$  do receive  $\langle ack \rangle$  from  $r$   
    end ;  
    if  $p$  is the initiator and  $\forall q \in Neigh_p : used_p[q]$   
    then  $decide$   
    else if  $\exists q \in Neigh_p : (q \neq father_p \wedge \neg used_p[q])$   
    then begin if  $father_p \neq q_0 \wedge \neg used_p[q_0]$   
        then  $q := q_0$   
        else choose  $q \in Neigh_p \setminus \{father_p\}$   
            with  $\neg used_p[q]$  ;  
         $used_p[q] := true$  ; send  $\langle tok \rangle$  to  $q$   
    end  
    else begin  $used_p[father_p] := true$  ;  
        send  $\langle tok \rangle$  to  $father_p$   
    end  
end
```

For each process, upon receipt of $\langle vis \rangle$ from q_0 :

```
begin  $used_p[q_0] := true$  ; send  $\langle ack \rangle$  to  $q_0$  end
```

Cheung'83: kubická komunikácia, lineárny čas

- každá správa obsahuje počítadlo hopov
- na začiatku iniciátor: všetkým susedom pošle 1
- každý vrchol p : lokálnu vzdialenosť $dist_p := \infty$
- on recv i : $dist_p := \min\{i, dist_p\}$, ak sa zmenila, pošli všetkým

Cheung,Zhu'87: kvadratická komunikácia aj čas

- buduje kostru po vrstvách

kombinácia?

naivne

```

const:   N      : integer
           ID     : integer
           Neigh : [1...N-1] link
var:    leader: boolean
           count : integer
           i      : integer

```

Init:

```

count := 0
leader := false

```

Code:

```

for i = 1 to N - 1 do
    send  $\langle \text{elect}, ID \rangle$  to Neigh[i]
while count < N - 1 wait
for i = 1 to N - 1 do
    send  $\langle \text{leader}, ID \rangle$  to Neigh[i]
leader := true

```

On receipt $\langle \text{elect}, id_i \rangle$ from *Neigh*[*i*]:

if $id_i > ID$ **send** $\langle \text{accept} \rangle$ **to** *Neigh*[*i*]

On receipt $\langle \text{accept} \rangle$ from *Neigh*[*i*]:

count + +

On receipt $\langle \text{leader}, id_i \rangle$ from *Neigh*[*i*]:

Skonči algoritmus

voľba šéfa: úplné grafy II

const: N : integer
 ID : integer
 $Neigh$: $[1 \dots N-1]$ link

var: $leader$: boolean
 $state$: {active, captured, killed}
 $level$: integer
 $parent$: link
 msg : {victory, defeat}
 i : integer

Init:

$state := active$
 $level := 0$
 $leader := false$

Code:

```
for  $i = 1$  to  $N - 1$  do
  send  $\langle capture, [level, ID] \rangle$  to  $Neigh[i]$ 
  receive  $\langle accept \rangle$  from  $Neigh[i]$ 
   $level ++$ 
 $leader := true$ 
for  $i = 1$  to  $N - 1$  do
  send  $\langle leader, ID \rangle$  to  $Neigh[i]$ 
```

Dead:

loop forever

On receipt $\langle capture, [level_i, id_i] \rangle$ from $Neigh[i]$:

```
if  $state \in \{active, killed\}$  and  $[level_i, id_i] > [level, ID]$ 
   $state := captured$ 
   $parent := Neigh[i]$ 
  send  $\langle accept \rangle$  to  $parent$ 
  goto Dead
```

```
else if  $state = captured$ 
  send  $\langle help, [level_i, id_i] \rangle$  to  $parent$ 
  receive  $msg$  from  $parent$ 
  if  $msg = defeat$ 
    send  $\langle accept \rangle$  to  $Neigh[i]$ 
     $parent := Neigh[i]$ 
```

On receipt $\langle help, [level_i, id_i] \rangle$ from $Neigh[i]$:

```
if  $[level_i, id_i] < [level, ID]$ 
  send  $\langle victory \rangle$  to  $Neigh[i]$ 
else
  send  $\langle defeat \rangle$  to  $Neigh[i]$ 
  if  $state = active$ 
     $state := killed$ 
  goto Dead
```

On receipt $\langle leader, id_i \rangle$ from $Neigh[i]$:

Skonči algoritmus

Lema 1

V ľubovoľnom výpočte existuje pre každý level $l = 0, \dots, N - 1$ aspoň jeden proces, ktorý bol počas výpočtu na leveli l .

Lema 2

Nech v je aktívny proces (*state = active*) s levelom l . Potom existuje l zajatých procesov ktoré patria v (t.j. ich premenná *parent* ukazuje na v).

⇒ práve jeden proces je šéf

Lemma 3

ľubovoľnom výpočte je najviac $N/(l + 1)$ procesov, ktoré niekedy dosiahli level l .

⇒ maximálne $\sum_{l=1}^{N-1} \frac{N}{l+1} = N(\mathbf{H}_N - 1) \approx N \log N$ postupov o level (=správ)

Treba $\Omega(n \log n)$ správ

- “globálny” algoritmus
- graf indukovaný novými správami
- udržujem výpočet:
 - jeden súvislý komponent, ostatné vrcholy izolované
 - $e(k)$ – koľko viem vynútiť na komponente k
 - $e(2k + 1) = 2e(k) + k + 1$

Chang Roberts

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

Init:

$leader := NULL$

Code:

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

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 $\langle leader, ID \rangle$

Koľkokrát sa pohla správa i ?

$$P(i, k) = \frac{\binom{n-i}{k-1} \cdot (i-1)}{\binom{n-1}{k-1} \cdot (n-k)}$$

$$E[X_i] = \sum_{k=1}^{n-i+1} k \cdot P(i, k)$$

$$E[X] = n + \sum_{i=2}^n E[X_i] = n + \sum_{i=2}^n \sum_{k=1}^{n-i+1} k \cdot P(i, k)$$

lema

$$\sum_{j=k}^{n-1} \frac{j!}{(j-k)!} = k! \binom{n}{k+1}$$

dôkaz

$$\sum_{j=k}^{n-1} \frac{j!}{(j-k)!} = \sum_{j=k}^{n-1} \frac{k!j!}{k!(j-k)!} = k! \sum_{j=k}^{n-1} \binom{j}{k} = k! \binom{n}{k+1}$$

$$\begin{aligned}
n + \sum_{i=2}^n \sum_{k=1}^{n-i+1} k \cdot \frac{\binom{n-i}{k-1} \cdot (j-1)}{\binom{n-1}{k-1} \cdot (n-k)} &= n + \sum_{j=1}^{n-1} \sum_{k=1}^j k \cdot \frac{\binom{j-1}{k-1} \cdot (n-j)}{\binom{n-1}{k-1} \cdot (n-k)} = \\
&= n + \sum_{j=1}^{n-1} \sum_{k=1}^j \frac{k(j-1)!(n-j)(k-1)!(n-k)!}{(k-1)!(j-k)!(n-1)!(n-k)} = \\
&= n + \sum_{k=1}^{n-1} \left[\frac{k(n-k-1)!}{(n-1)!} \sum_{j=k}^{n-1} \frac{(j-1)!(n-j)}{(j-k)!} \right] = \\
&= n + \sum_{k=1}^{n-1} \left[\frac{k(n-k-1)!}{(n-1)!} \left(\sum_{j=k}^{n-1} \frac{n(j-1)!}{(j-k)!} - \sum_{j=k}^{n-1} \frac{j!}{(j-k)!} \right) \right] = \\
&\sum_{j=k}^{n-1} \frac{(j-1)!}{(j-k)!} = (k-1)! \binom{n-1}{k} \text{ a } \sum_{j=k}^{n-1} \frac{j!}{(j-k)!} = k! \binom{n}{k+1} \\
&= n + \sum_{k=1}^{n-1} \frac{k(n-k-1)!}{(n-1)!} \left[n \cdot (k-1)! \binom{n-1}{k} - k! \binom{n}{k+1} \right] = \\
&= n + \sum_{k=1}^{n-1} \frac{n}{k+1}
\end{aligned}$$