Anonymné siete

Predpoklady:

- procesy nemajú identitu
- pracujú podľa rovnakého algoritmu
- neexistuje leader

Veta 1 *Pri asynchrónnom posielaní správ v anonymnej sieti* **neexistuje** *terminujúci algoritmus na voľbu leadra.*

DÔKAZ: Uvažujme jednosmerný cyklus veľkosti N. V symetrickej konfigurácii sú všetky procesy v rovnakom stave a všetky kanály obsahujú rovnaké správy.

- počiatočná konfigurácia je symetrická
- Ak je γ_0 symetrická a $\gamma_0 \rightarrow \gamma_1$, potom $\gamma_1 \rightarrow \gamma_2 \rightarrow \ldots \gamma_N$, kde γ_N je symetrická

Máme teda nekonečnú realizáciu

Východisko - **pravdepodobnostné algoritmy**

Pravdepodobnostné algoritmy

proces má dva lokálne algoritmy $\vdash^0, \vdash^1$; $\rho_p : N \to \{0,1\}; \rho : P \to \{\rho_p\}_{p \in P}$
v ρ-výpočte ide k-ty krok procesu p podľa $\vdash^{\rho_p(k)}$.

ρ-terminujúci, ak sa pri ρ-výpočte dosiahne terminálna konfigurácia

terminujúci, ak je ρ-terminujúci pre každé ρ

ρ-čiastočne korektný, ak je čiastočne korektný v $\forall$ ρ-výpočte

ρ-korektný, ak je ρ-terminujúci a ρ-čiastočne korektný

čiastočne korektný, ak je je ρ-čiastočne korektný pre každé ρ

korektný, ak je ρ-korektný pre každé ρ

terminujúci s pravdepodobnosťou p, ak je pravdepodobnosť toho, že je ρ-terminujúci, aspoň p

čiastočne korektný s pravdepodobnosťou p, ak je pravdepodobnosť toho, že je ρ-čiastočne korektný $\geq p$

korektný s pravdepodobnosťou p, ak je pravdepodobnosť toho, že je korektný, aspoň p

Klasifikácia pravdepodobnostných algoritmov

Sherwood - korektné pravdepodobnostné algoritmy.

Fakt 1 *Ak existuje korektný pravdepodobnostný algoritmus vzhľadom na výstupnú podmienku ψ, tak existuje korektný deterministický algoritmus vzhľadom na ψ.*

Monte Carlo, ak

- terminuje
- pravdepodobnosť toho, že sú všetky terminálne konfigurácie korektné, je nenulová

Las Vegas, ak

- pravdepodobnosť toho, že terminuje, je nenulová
- všetky terminálne konfigurácie sú korektné

$\longrightarrow$ Znalosť vs. *absencia znalosti* veľkosti siete

$\longrightarrow$ Termináciu správami/*procesmi*

$\longrightarrow$ *determinizmus* vs. pravdepodobnostné algoritmy

$\longrightarrow$ *Las Vegas* vs. Monte Carlo

centralizované výpočty

— veľkosť siete vypočítame echo-algoritmom (pri budovaní kostry procesy evidujú aj počty prvkov v podstrome. Iniciátor končí výpočet so znalosťou veľkosti siete.)

— iniciátor zabezpečí priradenie mien procesom (každému synovi pošle interval identifikačných čísel pre jeho podstrom.)

Fakt 2 *Existuje deterministický centralizovaný algoritmus na výpočet veľkosti siete, ktorého počet správ je* $2|E|$ *a čas* $O(N)$.
Existuje deterministický centralizovaný algoritmus na pomenovanie procesov pôvodne anonymnej siete, ktorého počet správ je $2|E| + (N-1)$ *a čas* $O(N)$

synchrónne vs. asynchrónne posielanie správ

Zatiaľ čo v prípade asynchrónneho posielania správ nedokážeme v prípade anonymného úplného grafu zvoliť leadra, synchrónne posielanie správ umožňuje symetriu narušiť.

Fakt 3 *Existuje deterministický procesmi terminujúci algoritmus výberu v anonymnej sieti dvoch procesov so synchrónnou komunikáciou.*

Proces môže byť v troch stavoch- *sleep, leader, lost*; počiatočný stav je *sleep*.
$send \Rightarrow leader \vee receive \Rightarrow lost$ ◇

Deterministické algoritmy - Počítanie na kruhu so znalosťou veľkosti

p_i	proces
$x_i \in X$	vstup procesu p_i na začiatku výpočtu
SX	množina všetkých postupností nad X
f je cyklická na SX	ak $f(x) = f(y) \forall y$, y je cyklickým posunom x; napr. $\sum, \vee, \wedge, \ldots$
$result_p$	predpokladáme, že každý proces má premennú $result_p$. Algoritmus počíta funkciu f, ak $result_p = f(x_0, \ldots, x_{N-1})$

Veta 2 *Pri znalosti veľkosti kruhu sa každá cyklická funkcia* ***dá*** *vypočítať deterministickým procesom terminujúcim algoritmom, ktorý pošle* $N(N-1)$ *správ.*

Veta 3 *Každý deterministický procesmi končiaci algoritmus na výpočet* $\wedge, \vee, \sum$ ***vyžaduje*** *(v najhoršom prípade)poslanie aspoň* $N(N-1)$ *správ.*

Počítanie na kruhu bez znalosti jeho veľkosti

Veta 4 ***Neexistuje*** *deterministický procesom terminujúci algoritmus na výpočet nekonštantnej funkcie, ak je veľkosť kruhu neznáma.*

$a = f(x) \neq f(y) = b$.
C_x – v procese p vypočíta $result_p = f(x)$; K - max dĺžka stopy správy
C_y – v procese q vypočíta $result_q = f(y)$.

Uvažujme väčší kruh, ktorý obsahuje úsek S dĺžky $K + 1$ taký, že

- posledný proces v S má rovnaký vstup ako p v C_x; nech je to proces p'
- i-ty predchodca p' má rovnaký vstup ako i-ty predchodca p v C_x

Algoritmus A bude **v segmente** S počítať ako vo výpočte $C_x \Rightarrow result_{p'} = a$

Uvažujme taký kruh, v ktorom bude okrem segmentu S aj **segment** R pre simuláciu výpočtu C_y. V tomto segmente skončí procesor (napr.) q' s výsledkom $result_{q'} = b$.

□

Veta 5 *Funkcie $\vee, \wedge$ vieme vypočítať deterministickým správami terminujúcim algoritmom na anonymnom kruhu bez znalosti veľkosti, pričom počet poslaných správ je nanajvýš N.*

Itai-Rodeh -pravdepodobnostná voľba so znalosťou veľkosti kruhu

Anonymný jednosmerný kruh veľkosti N. Základ — algoritmus Chang-Roberts: každý proces poslal svoj token a iba token najmenšieho z nich sa vrátil svojmu majiteľovi.

identifikačné čísla Najprv sa pokúsime narušiť symetriu tým, že si každý proces náhodne vygeneruje svoje identifikačné číslo z množiny $\{0, \ldots, N-1\}$.

počítanie krokov Kedže sa môže stať, že si niekoľko procesov vygeneruje rovnaké identifikačné číslo, budeme počítať kroky. Token je jednoznačne u svojho odosielateľa, ak spravil N krokov.

rovnaké identifikácie Dodáme tokenu boolovskú premennú un, ktorá je na začiatku nastavená na $true$. Ak počas prechodu kruhom prechádza cez proces s rovnakým identifikačným číslom, prestaví si $un := false$.

prechod do nového kola Ak sa vráti token s hodnotou $un = false$, proces vie, že mal minimálnu hodnotu, ale nie je jediný. Preto spustí dalšie kolo-opäť si vygeneruje novú identitu. V tokene sa nesie info aj o úrovni $\langle \mathbf{tok}, \mathbf{level}, \mathbf{id}, \mathbf{hops}, \mathbf{un} \rangle$

Veta 6 *Algoritmus Itai-Rodeh je čiastočne korektným algoritmom výberu na anonymnom kruhu známej veľkosti, ktorý terminuje s pravdepodobnosťou 1.*

Mierne vylepšená verzia Itai-Rodeh — keď token prechádza procesom s rovnakou identifikáciou, začína nová úroveň.

Kanály sú **FIFO** (pôvodne nebolo treba); procesy sú aktívne alebo pasívne.

Aktívny proces p pracuje v každom kole nasledovne

- p si vygeneruje náhodnú identifikáciu id_p a pošle $(id_p, 1)$
- ak p príjme (u, h), kde $id_p > u$, p sa stane pasívnym a iba posunie $(u, h + 1)$
- ak p príjme (u, h), kde $id_p < u$, p ignoruje token
- ak p príjme (id_p, h), kde $h < N$, začne nové kolo volieb
- ak p príjme (id_p, N), stane sa leadrom

Itai-Rodeh ukázali, že **priemerný počet správ** je **O(NlogN)**

– v priemere $e\frac{N}{N-1}$ kôl volieb

– očakávaný počet správ v jednom kole $O(NlogN)$

Algoritmus výberu v ľubovoľnej sieti známej veľkosti

Vychádzame z ECHO algoritmu so zánikom, náhodne generujeme identity. Na začiatku sú aktívne iba identifikátori - náhodne si vygenerujú identitu a spustia vlastnú vlnu.

Nech p vlny v je zasiahnutý vlnou w:

- ak $w < v$, p prejde do vlny w
- ak $v < w$, p pokračuje v svojej vlne v
- ak $v = w$, prichádzajúcu správu spracujeme echo algoritmom

Ak proces p *rozhodne*, vypočíta veľkost skonštruovaného stromu. Ak strom pokrýva celú sieť, proces sa stáva leadrom. Ak nie, spúšťa ďalšie kolo volieb.

Ak kanály nie sú FIFO, ďalšie kolo volieb má vlnu vyššej úrovne.

Výpočet veľkosti siete

Veta 7 *Neexistuje procesom terminujúci algoritmus na výpočet veľkosti anonymného jednosmerného kruhu, ktorý je korektný s pravdepodobnosťou $r > 0$.*

A je pravdepodobnostný algoritmus s ρ-výpočtom C, ktorý terminuje s výsledkom N

L je najväčší počet krokov, ktorý nejaký proces spravil vo výpočte C

K je maximálna dĺžka stopy správy, ktorá prechádzala cez p

Väčší kruh má segment dĺžky $K+1$, p_0 je posledný proces, p_i jeho i-ty predchodca. Nech ρ':

- prvých L bitov ρ'_{p_0} je rovnakých ako v ρ_{p_0}
- prvých L bitov ρ'_{p_i} je rovnakých ako v ρ_{p_i}, $i \leq K$

Pravdepodobnosť, že **ρ' bude mať také vlastnosti**, je $2^{-(K+1)L}$ $(*)$

Nech $r > 0$ je daná pravdepodobnosť. Nech R je také, že $(1 - 2^{-(K+1)L})^R < r$.

Potom kruh veľkosti $R(K+1)$ má R rôznych segmentov dĺžky $K+1$. Pre každý z nich je pravdepodobnosť toho, že na ňom odpovie nesprávne, $2^{-(K+1)L}$.

Preto je pravdepodobnosť toho, že **žiaden proces neskončí s nesprávnou odpoveďou** $(1 - 2^{-(K+1)L})^R < r$.

Veta 8 ***Neexistuje správami terminujúci*** *algoritmus na výpočet veľkosti anonymného kruhu, ktorý je korektný s pravdepodobnosťou 1.*

DÔKAZ: Uvažujme kruh $C_N = p_0, \ldots, p_{N-1}$ a pravdepodobnostný algoritmus A s ρ-výpočtom, v ktorom proces p skončí s odpoveďou $result_p = N$. Nech L je maximálny počet udalostí realizovaných nejakým procesom v ρ-výpočte.

Opät skonštruujeme väčší kruh, ktorý bude nesprávne odpovedať. Nech tento nový kruhu je veľkosti $2N$; $C_{2N} = q_0, \ldots, q_{2N-1}$. Nech ρ' je taká, že prvých L bitov ρ'_{q_i} sa rovná prvým L bitom ρ_{q_i}, čo sa rovná prvým L bitom $\rho'_{q_{i+N}}, i < N$. Potom vo výpočte na väčšom kruhu máme 2 procesy, ktoré nesprávne odpovedia $result = N$.

Aká je pravdepodobnosť, že ρ' má príslušné vlastnosti? Je to 2^{-2NL}. Preto algoritmus A nevypočíta správny výsledok s pravdepodobnosťou väčšou ako $1 - 2^{-2NL}$

Itai-Rodeh: Monte Carlo výpočet veľkosti kruhu

Udržiavanie odhadu $\mathbf{est_p}$ veľkosti kruhu v každom procese p, pričom v každom okamihu $est_p \leq N$.

Náhodne vygenerujeme mená, pričom používame mená z množiny $\{1, \ldots, R\}$. Tu R je konštanta, ktorá určuje pravdepodobnosť korektnosti.

Posielaný token - $\langle \mathbf{est_p}, \mathbf{id_p}, \mathbf{s} \rangle$; s je počet krokov, ktoré token spravil.

Inicializácia - $est_p = 2, id_p := \mathbf{random(R)}$; send $\langle est_p, id_p, 1 \rangle$

Po prijatí $\langle est, id, s \rangle$

- ak $est < est_p$, p ignoruje správu
- Nech $est > est_p$ (ak ešte token nespravil potrebný počet krokov, preberieme odhad a pošleme ho ďalej. Ak potrebný počet krokov spravil, je kruh evidentne väčší ako ten väčší odhad)
  - ak $s < est$ tak (send $\langle est, id, s+1 \rangle$, $est_p \leftarrow est$; $id_p := random(R)$; send $\langle est_p, id_p, 1 \rangle$) (1)
  - ak $s = est$ tak (nastav $est_p \leftarrow est + 1$; $id_p := random(R)$; send $\langle est_p, id_p, 1 \rangle$) (2)
- Nech $est = est_p$ (treba zvážit aj možnosť, že to je "môj"token, ktorý sa vrátil naspäť)
  - ak $s < est$, p pošle $\langle est, id, s+1 \rangle$
  - ak $s = est$ a $id \neq id_p$ tak ($est_p \leftarrow est + 1$; $id_p := random(R)$; send $\langle est_p, id_p, 1 \rangle$) (3)
  - ak $s = est$ a $id = id_p$ tak p ostane pasívny

$est_p \leq N$

- Nech $est > est_p$
  - ak $s < est$ tak (send $\langle est, id, s+1 \rangle$ a nastav $est_p = est$) (1)
  - ak $s = est$ tak nastav $est_p = est + 1$ (2)
- Nech $est = est_p$ (treba zvážiť aj možnost, že to "môj"token, ktorý sa vrátil naspäť)
  - ak $s < est$, p pošle $\langle est, id, s+1 \rangle$
  - ak $s = est$ a $id \neq id_p$ tak $est_p := est + 1$ (3)
  - ak $s = est$ a $id = id_p$ tak p ostane pasívny

(1.) Ked zväčšujem $est_p := est$, je kruh určite aspoň veľkosti est

(2,3) Token, ktorý spravil est krokov prišiel do procesu, z ktorého nevyšiel. Preto je veľkosť kruhu aspoň $est + 1$

Počet správ je $O(N^3)$

Každý proces vygeneruje nanajvýš $N-1$ rôznych tokenov, token s odhadom est sa dostane do vzdialenosti est.

V terminálnej konfigurácii $\forall p, q\ est_p = est_q$

Kedže pri zvyšovaní est_p posielame token susedovi, platí $est_p \leq est_{Next_p}$. V terminálnej konfigurácii preto platia rovnosti.

Pravdepodobnosť nesprávnej odpovede

K nesprávnej odpovedi dôjde len vtedy, ak sa všetky procesy p stali pasívnymi skôr, ako bolo treba. Čiže v kruhu existoval vo vzdialenosti est proces q, ktorého $id_q = id_p \wedge est_q = est_p = est$.

Nech $\mathbf{f = gcd(N, est)}$ ak f delí $(j - i) \Longrightarrow id_{p_i} = id_{p_j}$

Rozdelenie procesov do f skupín veľkosti N/f; $\forall$ procesy v jednej skupine majú rovnaké meno.

Pravdepodobnosť toho, že **všetky procesy v skupine si zvolili rovnaké meno** je $\mathbf{(1/R)^{\frac{N}{f}-1}}$

Pravdepodobnosť toho, že sa to stalo **vo všetkých skupinách** je

$$(1/R)^{f(\frac{N}{f}-1)} = (1/R)^{N-f}$$

$est \in \{2, \ldots, N-1\}$, $f \leq N/2$, preto $\sum_{est \in \{2,\ldots,N-1\}} (1/R)^{(N-f)} \leq \mathbf{(N-2)/(R^{N/2})}$

Veta 9 *Algoritmus Itai-Rodeh(Monte Carlo) terminuje správami, pričom sa pošle nanajvýš $O(N^3)$ správ. V okamihu terminovania*

$$\forall p, q \; est_p = est_q \; a \; (est_p = N \; s \; pravdepodobnosťou \; aspoň \; 1 - (N-2)(1/R)^{N/2})$$

Synchronizácia

ÁNO globálny čas (idealizovane diskrétny čas)
NIE synchronizovaná komunikácia(nevyžaduje celkovú synchronizáciu)
M -množina správ, **PM** - multimnožiny správ

V synchronizovanom výpočte spracovávame info na všetkých hranách naraz

Definícia 1 *Synchronizovaný proces je štvorica* $p = (Z, I, MG, \vdash)$, *kde*

1. Z *je množina stavov*
2. I *je podmnožina počiatočných stavov*
3. $MG : Z \to PM$ *je generátor správ*
4. $\vdash$ *je relácia na* $Z \times PM \times Z$; $(c, M) \vdash d$ *namiesto* $(c, M, d) \in \vdash$

Definícia 2 *Synchronizovaný systém (synchronizovaných) procesov* $P = \{p_1, \ldots, p_N\}$, *kde* p_i *je proces* $(Z_i, I_i, MG_i, \vdash_i)$, *je prechodový systém* $S = (C, \to, I)$, *kde*

1. $C = \{(c_1, \ldots, c_N) \forall i, c_i \in Z_i\}$
2. $\to$: $(c_1, \ldots, c_N) \to (d_1, \ldots, d_N)$ *ak* $\forall p_i \in P; (c_i, \{m \in (\cup_{p_j \in P} MG_j(c_j)) :$ *miesto doručenie pre* m *je* $p_i\}) \vdash_i d_i$
3. $I = \{(c_1, \ldots, c_N) : \forall i, c_i \in I\}$

Zefektívnenie synchronizáciou

kódovanie časom rozdiel v posielaní správ možno považovať za celočíselnú hodnotu

implicitné správy ak neposielam, posielam nejakú implicitnú hodnotu

selekcia časom čas správneho výpočtu "diskriminuje"nekorektné

Namiesto synchronizácie - **Asynchrónne siete s ohraničeným zdržaním** (**ABD**)

Definícia 3 *ABD sieť je distribuovaný systém*

1. *spracovanie udalosti v procese spotrebuje nulový čas*
2. *proces má fyzikálne hodiny -* $CLOCK_p^{(t)}$ *je hodnota hodín procesu* p *v čase* t*. Ak proces medzi* t_1 *a* t_2 *neprestavuje hodnotu svojich hodín, tak* $CLOCK_p^{(t_1)}$-$CLOCK_p^{(t_2)} = t_2 - t_1$
3. *fyzikálny čas medzi poslaním s prijatím správy je ohraničený hodnotou* μ*: ak správa poslaná v čase* σ *bola prijatá v čase* τ*, tak* $\sigma < \tau < \sigma + \mu$

ABD synchronizátor -zosynchronizujú sa hodiny s presnosťou σ a krok sa vykonáva po $\sigma + \mu$ jednotkách hodín

Dve fázy ABD synchronizátora - synchronizácia hodín a simulácia

init:
if not $started_p$ **then**
  $CLOCK_p \leftarrow 0$
  for all $q \in Neigh_p$ **do** *send* (start) do q

I_p ▷ po prijatí (start) v p
receive (start); init

$P_p^{(i)}$: ▷ štart i-teho takt
if $CLOCK_p = 2i\mu$ **then**
  if $i = 1$ **then** $state_p \leftarrow$počiatočný stav
  else $state_p \leftarrow d_p$ pre $(state_p, M) \vdash d_p$
    $pulse_p \leftarrow i$
    $M' = MG(state_p)$; pošli všetky správy z M'

M_p ▷ prijatie základnej i-správy
receive and store message

Fakt 4 *V každom čase t po inicializovaní procedúry init susedmi p, q platí:*

$$|CLOCK_p^t - CLOCK_q^t| < \mu$$

Nech $state_p^{(i)}$ označuje stav procesu p po $i+1$ zmenách stavu-keď začína $(i+1)$-ý takt
$\gamma_i = (state_{p1}^{(i)}, \ldots, state_{pN}^{(i)})$

Veta 10 *$C = (\gamma_0, \gamma_1, \ldots)$ je výpočet synchronizovaného algoritmu*

Zložitosť synchronizácie - meriame časom a počtom správ

- inicializácia - $2|E|$ správ, čas $O(D)$
- simulácia - žiadne dodatočné správy, jeden takt 2μ jednotiek času

Výber v synchronizovaných sieťach - pri známej veľkosti siete

–D-horný odhad na priemer (N-1, ak N poznáme)

–p rozpošle správu v čase t_0 - všetci ju majú v $t_0 + (D-1)$, stojí to $2|E|$ správ

ak proces nedostane rozposielanú správu v takte $i + (D-1)$ alebo skôr, nik nerozposlal správu v takte i alebo skôr

proces p počká na správu, alebo kým je čas pD a vtedy spustí rozposielanie (najmenší vyhráva)

počet správ $O(|E|)$, čas $p_0 D + (D-1)$

rozšírenie - kostra, leader sa stane koreňom

Výber v synchronizovaných sieťach - bez znalosti veľkosti siete

Vychádzame z traverzovania so zánikom, využijeme selekciu časom

Iniciátor p voľby šéfa spustí traverzovanie, svoj token si označí; token číslo p prijatý v takte i sa posúva v takte $i + 2^p$.

Token

- zanikne, ak prechádza cez proces, ktorý videl token s menším číslom
- úspešne skončí traverzovanie v p a stane sa leadrom

počet správ- $\mathbf{O(W)}$, kde W je počet správ traverzovacieho algoritmu

počet správ $\leq W + \Sigma_{p \in S - \{p_0\}} \frac{2^{p_0} W}{2^p}$
$< W + \sum_{i>0} \frac{W}{2^i} = 2W$

Synchronizácia začiatku- najprv ich zobudíme (čas D, počet správ $2|E|$)

Algoritmy synchronizácie

jednoduchý: v každom takte posiela proces správu susedom

α-synchronizátor: Po všetkých hranách proces posiela informáciu $\langle$**iamsafe**$\rangle$. Keď všetci susedia potvrdia, proces pokračuje ďalej.

– netreba inicializačnú fázu

β-synchronizátor: Analogicky, ale komunikácia prebieha len po kostrových hranách.

– Koreň posiela správu $\langle$**pulse**$\rangle$ dolu stromom, aby oznámil, že môže nastať ďalší krok. Ostatné vrcholy odpovedajú $\langle$**ts**$\rangle$.

– vyžaduje sa inicializačná fáza na vybudovanie kostry

γ-synchronizátor: V inicializačnej fáze sú vrcholy rozdelené do skupín. V rámci skupiny algoritmus pracuje ako β-synchronizátor, medzi skupinami pracuje ako α-synchronizátor.

dôraz na

$\Rightarrow$ rozdelenie na triedy

$\Rightarrow$ určenie leadra v rámci triedy

$\Rightarrow$ určenie jednej hrany na komunikáciu medzi triedami

Centralizovaný asynchrónny BFS - synchronizácia zakomponovaná do alg.

Na začiatkuje iniciátor na úrovni 0, ostatní ∞. Po kole $f \geq 0$ každý vrchol vo vzdialenosti f vie, ktoré vrcholy sú vo vzdialenosti $f-1$.

V kole $f+1$

- $(forward, f)$ putuje po strome *dole*, od iniciátora k vrcholom vo vzd. f
- vrchol vo vzd. f posiela $(explore, f+1)$ všetkým susedom, kt. nie sú vo vzd. $f-1$
- vrchol na úrovni f ignoruje $(explore, f+1)$
- ak vrchol na úrovni ∞ dostane $(explore, f+1)$, nastaví úroveň na $f+1$. Otcom sa stane ten, od koho prišla správa. Naspäť pošle $(reverse, T)$
- ak vrchol vo vzd.$f+1$ dostane $(explore, f+1)$ vie, že správa prišla od vrchola na úrovni f. Pošle späť $(reverse, F)$
- neiniciátor na úrovni f (resp. $< f$) čaká, kým nie sú zodpovedané všetky správy $(explore, f+1), (forward, f)$. Potom pošle $(reverse, b)$ otcovi, pričom $b = T \Leftrightarrow$ do svojho podstromu pridal nový vrchol
- iniciátor čaká na zodpovedanie všetkých $(forward, f), (explore, 1)$správ. Ak boli pridané nejaké vrcholy, pokračuje kolom $f+2$, inak terminuje

Počet správ $\mathbf{O(N^2)}$

- maximálne N kôl
- v každom kole ide po stromovej hrane jedna forward alebo explore, jedna reverse správa
- nestromové hrany (celkovo) pošlú jednu explore a odpovedajú reverse alebo explore

synchronizácia dominuje nad "výpočtom"

Čas - $\mathbf{O(N^2)}$

kolo č. f sa ukončí v $2f$ časových jednotkách

Centralizovaný asynchrónny BFS-Frederickson-(l) úrovní v jednom kole

Na začiatkuje iniciátor na úrovni 0, ostatní ∞

Po kole k každý vrchol vo vzdialenosti $f \leq kl$ patrí do úrovne f a vie, ktoré vrcholy sú v úrovni $f-1$.

V kole $k+1$

- dole stromom ide $(forward, kl)$ od iniciátora k vrcholom na úrovni kl
- vrchol úrovne kl pošle $(explore, kl+1, l)$ susedom, ktorí nie sú v úrovni $kl-1$
- Ak vrchol u dostane $(explore, f, m)$, $f < level_u$, nastaví si $level_u := f$, otcom sa stane ten, od koho prišla správa, u prípadne pošle $(reverse, F)$ svojmu pôvodnému otcovi
  Ak $m > 1, u$ pošle $(explore, f+1, m-1)$ susedom, o ktorých nevie, že sú na úrovni $\leq f+1$
  Ak $m = 1$ (teda $f = (k+1)l$), u pošle späť $(reverse, T)$
- Ak vrchol u dostane $(explore, f, m)$, pričom $f \geq level_u$, pošle naspäť $(reverse, F)$
- Vrchol na úrovni $kl \leq f < (k+1)l$ (resp. $0 < f < kl$), počká na potvrdenie všetkých $(explore, f+1)$ (resp.$(forward, kl)$ správ správou $(reverse, b)$. Potom pošle $(reverse, b)$ otcovi, kde $b = T \Leftrightarrow$ pribudli nové vrcholy
- Iniciátor počká na potvrdenie všetkých $(forward, kl)$ (a $(explore, 1)$)správ. Pokračuje kolom $k+2$, ak v kole $k+1$ pribudli vrcholy, inak terminuje.

Počet správ - $O(\frac{N^2}{l} + l|E|)$

- N/l kôl
- po stromovej hrane ide max. N/l forward, N/l odpovedí
- po každej hrane ide max $2l$ explore správ a $2l$ odpovedajúcich reverse správ

Čas - $O(\frac{N^2}{l})$

- Úrovne $kl + 1$ až $(k + 1)l$ v čase $O(N)$

Ak $l = \frac{N}{\sqrt{|E|}}$, tak čas aj správy sú $O(N\sqrt{|E|})$