

Lokálny algoritmus procesu

Päťica $(Z, I, \rightarrow^i, \rightarrow^s, \rightarrow^r)$, kde:

- Z - množina stavov
- $I \subseteq Z$ - množina počiatkových stavov
- $\rightarrow^i \subseteq Z \times Z$ - prechodová relácia pre internú udalosť
- $\rightarrow^r \subseteq Z \times M \times Z$ - prechodová relácia pre udalosť receive, M - množina možných správ
- $\rightarrow^s \subseteq Z \times M \times Z$ - prechodová relácia pre udalosť send

Distribovaný algoritmus

DA pre množinu procesov $\mathbb{P} = \{p_1, \dots, p_n\}$ je množina lokálnych algoritmov, jeden pre každý proces z \mathbb{P} .

Systém pod asynchrónnou komunikáciou

Systém indukovaný DA-om pre $\mathbb{P} = \{p_1, \dots, p_n\}$ pod asynchrónnou komunikáciou je $S = (C, \rightarrow, I)$, kde

- $C = \{(c_{p_1}, \dots, c_{p_n}, M) \mid (\forall p \subseteq \mathbb{P}. c_p \in Z_p), M \in \mathcal{M}(M)\}$, pozn. wtf znamená $M \in \mathcal{M}(M)$? $M \in \mathcal{P}(M)$?
- $\rightarrow = \bigcup_{p \in \mathbb{P}} \rightarrow_p$, kde \rightarrow_p je množina dvojíc $((c_{p_1}, \dots, c_{p_i}, \dots, c_{p_n}, M_1), (c_{p_1}, \dots, c'_{p_i}, \dots, c_{p_n}, M_2))$
 - $(c_{p_i}, c'_{p_i}) \in \rightarrow_{p_i}^i, M_1 = M_2$ (interná udalosť, neposielali sa správy)
 - $\exists m \in M (c_{p_i}, m, c'_{p_i}) \in \rightarrow_{p_i}^s, M_2 = M_1 \cup m$ (udalosť send, pribudla správa m)
 - $\exists m \in M (c_{p_i}, m, c'_{p_i}) \in \rightarrow_{p_i}^r, M_1 = M_2 \cup m$
- $I = \{(c_{p_1}, \dots, c_{p_n}, M) \mid (\forall p \subseteq \mathbb{P}. c_p \in I_p), M = \emptyset\}$

Systém pod synchronnou komunikáciou

Systém indukovaný DA-om pre $\mathbb{P} = \{p_1, \dots, p_n\}$ pod synchronnou komunikáciou je $S = (C, \rightarrow, I)$, kde

- $C = \{(c_{p_1}, \dots, c_{p_n}) \mid (\forall p \subseteq \mathbb{P}. c_p \in Z_p)\}$
- $\rightarrow = \bigcup_{p \in \mathbb{P}} \rightarrow_p \cup \bigcup_{p, q \in \mathbb{P}, p \neq q} \rightarrow_{pq}$, kde
 - \rightarrow_p je množina dvojíc $((c_{p_1}, \dots, c_{p_i}, \dots, c_{p_n}), (c_{p_1}, \dots, c'_{p_i}, \dots, c_{p_n})), (c_{p_i}, c'_{p_i}) \in \rightarrow_{p_i}^i$
 - \rightarrow_{pq} je množina dvojíc $((\dots, c_{p_i}, \dots, c_{p_j}, \dots), (\dots, c'_{p_i}, \dots, c'_{p_j}, \dots)), \exists m. (c_{p_i}, m, c'_{p_i}) \in \rightarrow_{p_i}^s \wedge (c_{p_j}, m, c'_{p_j}) \in \rightarrow_{p_j}^r$
- $I = \{(c_{p_1}, \dots, c_{p_n}) \mid (\forall p \subseteq \mathbb{P}. c_p \in I_p)\}$

Ak Q je invariant a P je Q -derivát, potom $Q \wedge P$ je invariant

Nech $S = (C, \rightarrow, I)$ a P, Q sú predikáty

$$\{P\} \rightarrow \{Q\} \Leftrightarrow (\forall \gamma \rightarrow \delta \in \rightarrow. P(\gamma) \Rightarrow Q(\delta))$$

P je invariantom systému S , ak $\forall \gamma \in I. P(\gamma) \wedge \{P\} \rightarrow \{P\}$.

P je Q -derivát, ak

1. $\forall \gamma \in I, Q(\gamma) \Rightarrow P(\gamma)$
2. $\{Q \wedge P\} \rightarrow \{Q \Rightarrow P\}$

Q je invariant, preto $\forall \gamma \in I.Q(\gamma)$ a keďže P je Q -derivát, $\forall \gamma \in I.Q(\gamma) \Rightarrow P(\gamma)$, teda $\{P(\gamma)\} \Rightarrow \{Q(\gamma) \wedge P(\gamma)\}$

Nech $\gamma \rightarrow \delta$, $Q(\gamma) \wedge P(\gamma)$. Keďže Q je invariant, platí $Q(\delta)$.

Keďže $\{Q \wedge P\} \rightarrow \{Q \Rightarrow P\}$, $Q(\delta) \Rightarrow P(\delta)$, teda $Q(\delta) \wedge P(\delta)$.

Predikát term, korektnosť, uviaznutie

Nech S je prechodový systém a P je predikát (cieľ).

term je predikát, ktorý je (ne)pravdivý v (ne)terminálnych konfiguráciách.

Systém S končí korektno, ak predikát ($term \Rightarrow P$) v S vždy platí.

Uviaznutie nastáva v terminálnej konfigurácii, kde sa cieľ P nedosiahol.

Norma

Nech S je prechodový systém a P je predikát. Funkcia f z C do dobre založenej množiny W sa nazýva norma (vzhľadom na P), ak $\forall \gamma \rightarrow \delta. (f(\gamma) > f(\delta) \vee P(\delta))$.

Lamportove hodiny

Nech $\Theta(a)$ je dĺžka najdlhšej postupnosti $(e_0, e_1, e_2, \dots, e_k)$ takej, že $e_0 \prec e_1 \prec e_2 \prec \dots \prec e_k = a$.

- Ak b je interná alebo send udalosť, a predchádzajúca udalosť toho istého procesora, potom $\Theta_L(b) = \Theta_L(a) + 1$.
- Ak r je receive, i je predchádzajúca udalosť toho istého procesora a s je odpovedajúca send udalosť, potom $\Theta_L(r) = \max \{\Theta_L(i), \Theta_L(s)\} + 1$.
- Ak a je prvá udalosť, $\Theta_L(a) = 0$.

Strom najkratších ciest

Pre každý vrchol v grafe existuje strom, kde pre každý vrchol je vzdialenosť od koreňa rovná minimálnej vzdialenosti oboch vrcholov v grafe.

Konštrukcia. Sink-tree. To sa dá vymyslieť „on-the-fly“.

Problémy Touegovho algoritmu

- kvôli voľbe pivota vyžadujeme znalosť všetkých vrcholov
- vyžadujeme informáciu $(d(u, w) + d(w, v) < d(u, v))$, ktorá nie je dostupná vo vrchole, ani v susedných vrchole.

Chandy-Misra:

$$d(u, v) = \begin{cases} 0, & u = v \\ \min_{w \in Neigh_u} \{\omega_{uv} + d(w, u)\}, & \text{inak} \end{cases}$$

Netchange

Algoritmus uchováva množinu susedov všetkých vrcholov, pri poruche/oprave kanála patrične upraví množiny a prepočíta podľa nich smerovacie tabuľky. V prípade, že sa vrcholu zmení najmenšia vzdialenosť do nejakého uzla, propaguje uzol túto informáciu jeho susedom (v procedúre *Recompute(v)*).

Premenné

- $Neigh_u$ - Aktuálna množina susedov vrchola u .
- $D_u[v]$ - odhad $d(u, v)$
- $Nb_u[v]$ - prvý vrchol na ceste s $D_u[v]$
- $ndis_u[w, v]$ - odhad $d(w, v)$

Správy

- $\langle mydist, v, d \rangle$ - vrchol oznamuje svoju vzdialenosť d od vrchola v .
Po prijatí spraví algoritmus odhad pre vrcholy idúce cez v a prepočíta tabuľky.
- $\langle fail, v \rangle$ - správa od w oznamuje poruchu kanála (v, w) .
 $Neigh_u \leftarrow Neigh_u - \{w\}$, prepočíta tabuľky pre vrcholy cez v .
- $\langle repair, v \rangle$ - správa od w oznamuje obnovenie kanála (v, w) .
 $Neigh_u \leftarrow Neigh_u \cup \{w\}$, prepočíta tabuľky pre vrcholy cez v a pošle ich vrcholu v .

KKM

...

GHS

...

Premenovanie

...

Acyklická orientácia, acyklické pokrytie

Acyklická orientácia grafu G je orientovaný acyklický graf, ktorý vznikne z grafu G zorientovaním hrán. Postupnosť G_1, \dots, G_B acyklických orientácií grafu G je acyklické pokrytie veľkosti B pre množinu ciest \mathcal{P} , ak $\forall p \in \mathcal{P}$, P možno zapísať ako zretazenie nanaajvýš B ciest P_1, \dots, P_k , $k \leq B$ takých, že P_i je cesta v G_i .

Ak existuje acyklické pokrytie veľkosti B , na prevenciu uviaznutia stačí vo vrchole nanaajvýš B bufferov. Pri generovaní packetu p vo vrchole u sa sa tento umiestni do bufferu $b_u[1]$. Pri forwardovaní packetu z vrchola u do vrchola w číslo bufferu ostáva, ak môžeme pokračovať v grafe G_i , inak preskočíme do grafu G_{i+1} .

Stabilizácia

...

Vzájomné vylúčenie (Dijkstra v kruhu)

...

Vzájomné vylúčenie Chandra-Toueg

...

Voľba šéfa v jednosmernom kruhu s $O(n \log n)$ správami

...

Singh

...

Dobrev

...

Rana

...

Detektory chýb

Detektor chýb je algoritmus, ktorý identifikuje procesy, ktoré prestali pracovať.

- T - čas
- $F(\tau)$ - množina procesov, ktoré v čase τ nepracujú.
- $Crash(F) = \bigcup_{\tau \in T} F(\tau)$ - množina chybných procesov
- $Corr(F) = \mathbb{P} - Crash(F)$ - množina korektných procesov.
- $H(p, \tau)$ - množina procesov, o ktorých procesor p predpokladá, že majú v čase τ fatálnu chybu.

Detektor je úplný vtedy, keď ak nejaký proces prestane pracovať, v konečnom čase ho každý procesor bude považovať za chybný. $\exists \tau \in T. \forall p \in Crash(F). \forall q \in Corr(F). \forall \tau' \in T. \tau' \geq \tau, p \in H(q, \tau')$

Detektor je (úplne) presný, ak podozrieva len o záj chybné procesy. $\forall \tau \in T. \forall p, q \in \mathbb{P}. p, q \notin F(\tau) \Rightarrow p \notin H(q, \tau)$

Konštrukcia úplne presného detektora

- každý proces pošle každých σ časových jednotiek správu *alive*
- existuje horný odhad μ na komunikačné zdržanie

Každý proces, od ktorého neprišla počas $\sigma + \mu$ časových jednotiek správa *alive*, prestal pracovať.

Vlnové algoritmy, traverzovanie

Vlnový algoritmus je DA, ktorý spĺňa

- terminácia - $\forall C. |C| < \infty$
- rozhodnutie - $\forall C. \exists e \in C$, e je (špeciálna interná) *decide* udalosť
- závislosť - v každom výpočte každej *decide* udalosti predchádza udalosť v každom procese

Traverzovací algoritmus je vlnový algoritmus, ktorý navyše spĺňa

- algoritmus má jediného iniciátora, ktorý jediný rozhoduje
- existuje úplné usporiadanie procesov

Traverzovací algoritmus je $f(x)$ -traverzovací, ak po vykonaní x krokov (posunov tokena), je navštívených $\max \{f(x), x\}$ vrcholov.

Algoritmus je f -traverzovací pre triedu sietí, ak

- je traverzovací pre túto triedu
- v každom výpočte je po $f(x)$ posunoch tokena navštívených aspoň $\min \{N, x + 1\}$ procesov.

Pre torus (kde vrcholy majú zmysel pre orientáciu) a hyperkocku existujú x -traverzovacie algoritmy.