Dana Pardubská

Katedra informatiky, FMFI UK, Bratislava

+421 2 602 95 158

pardubska@fmph.uniba.sk
http://www.dcs.fmph.uniba.sk/~pardubska/brno

Čo je distribuovaný systém?

Prepojenie sústavy **autonómnych** počítačov, procesov, procesorov

Autonómny- vlastné riadenie
Prepojenie- komunikácia/ výmena informácie

Prečo distribuované systémy?

pôvodne veľa procesov rozdistribuovaných vo veľkej geograficej ploche

neskôr lokálne siete, multiprocesorové systémy so spoločne zdieľanou pamäťou

- Výmena informácií (WAN)
- Zdieľanie zdrojov
- Zvýšenie spoľahlivosti replikáciou
- Zvýšenie výkonu využitím paralelizmu
- Zjednodušenie návrhu využitím špecializácie

rozdiely DA

spôsob medziprocesorovej komunikácie

shared memory / point-to-point posielanie správ / broadcasting / remote procedure calls

časovanie

— **synchrónne systémy** (spoločný čas)

— **asynchrónne systémy** (rôzne rýchlosti, rôzne poradie krokov)

— **čiastočne synchronizované** (čiastočné info o časovaní udalostí)

model chýb

— spoľahlivý vs. nespoľahlivý hardware

— tolerancia nejakého množstva chybného správania

chyby procesora	stop / crash / byzantínske
chyby kanálov	strata / duplikácia / modifikácia správ

DA - sústreďujeme sa na algoritmy s vysokým **stupňom neurčitosti** a veľkou **nezávislosťou** jednotlivých **akcií**

- neznámy počet procesorov
- neznáma topológia
- nezávislé vstupy na rôznych miestach
- viacero programov (rôzny čas začiatku, rôzne rýchlosti, ..)
- nedeterminizmus procesorov
- neistý čas doručenia správ
- neznáme poradie doručovania správ
- chyby procesorov a komunikácie/liniek

Distribuované vs. Centralizované algoritmy

znalosť globálneho stavu

pojem globálneho času

nedeterminizmus/determinizmus

Kvôli dôkazom korektnosti/ zložitosti / neriešiteľnosti používame **model**

Príklad - spoľahlivá výmena informácií cez nespoľahlivé médium

Procesy a,b; procedúry kontroly siete NCP -A,B; správa m

Naviazanie a ukončenie komunikácie

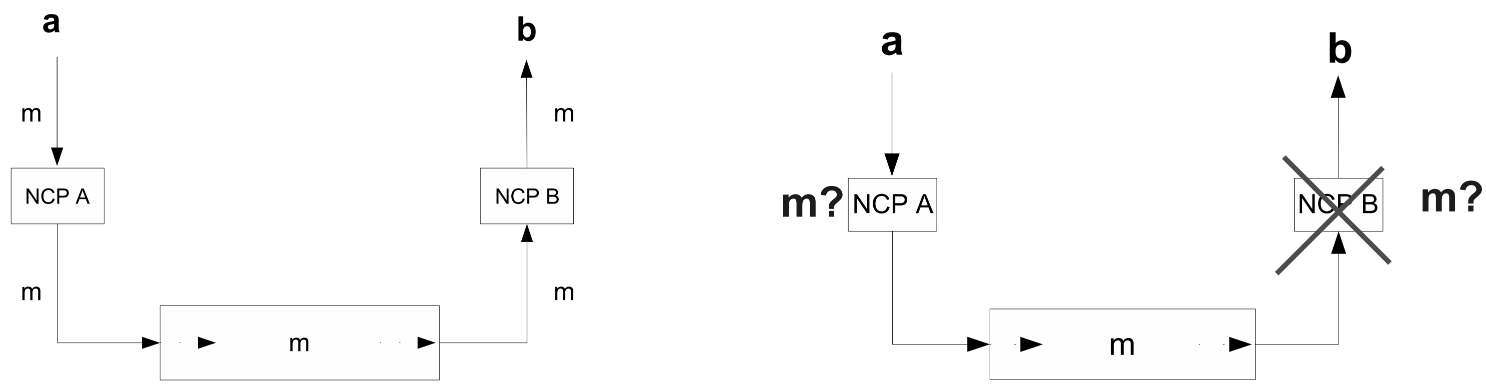
Informačná jednotka sa môže stratiť alebo duplikovať, NCP môže zlyhať (reštart v uzavretom stave)

Spoľahlivá výmena nie je dosiahnuteľná

→ Inicializácia komunikácie procesom a

→ NCP A a NCP B začnú komunikáciu, počas ktorej B doručí m do b

→ NCP B sa zrúti a je reštartovaný v uzavretom stave



V tejto situácii ani A ani B nevedia, či m bolo doručené

Konverzácia jednou správou- $<\mathbf{data}, \mathbf{m}>$

1. NCP A **send** $<\mathbf{data}, \mathbf{m}>$, notify, close
2. NCP B receive <data, m>, deliver m, close

- strata, keď sa správa nedoručí $\Rightarrow$ zavádza sa **potvrdenie príjmu** - nedochádza k duplikácii

Konverzácia dvomi správami- $<\mathbf{data}, \mathbf{m}>$, - $<\mathbf{ack}>$

1. NCP A **send** $<\mathbf{data}, \mathbf{m}>$
2. NCP B receive <data, m>, deliver m, **send** $<\mathbf{ack}>$, close
3. NCP A receive <ack>, notify, close

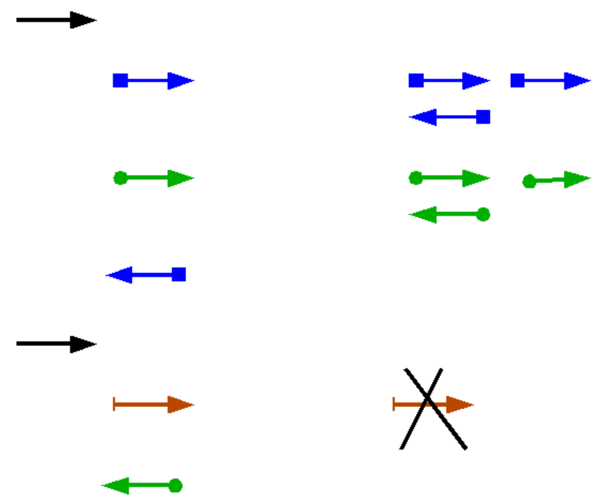
zavádza sa **vypršanie času a opakované poslanie správy**

1. NCP A **send** $<\mathbf{data}, \mathbf{m}>$
2. NCP B receive <data, m>, deliver m, **send** $<\mathbf{ack}>$, close
3. DN <ack> sa stratí
4. NCP A timeout, **send** $<\mathbf{data}, \mathbf{m}>$
5. NCP B receive <data, m> deliver m, **send** $<\mathbf{ack}>$, close
6. NCP A receive <ack>, notify, close

opätovné poslanie správy $\rightarrow$ možnosť duplikácie

Pri zavedení potvrdenia sa môžu strážené správy stratiť

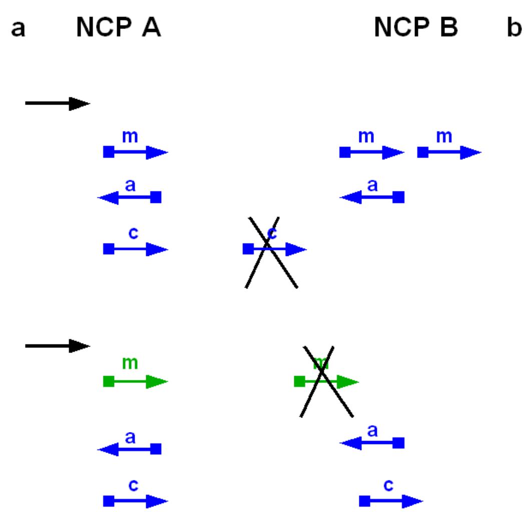
1. NCP A **send < data, m1 >**
2. NCP B receive <data, m1> deliver m1,
   send < ack >, close
3. NCP A timeout, **send < data, m1 >**
4. NCP B receive <data, m1> deliver m1,
   send < ack >, close
5. NCP A receive <ack>, notify, close
6. NCP A **send < data, m2 >**
7. DN <data, m2> lost
8. NCP A receive <ack>, notify, close



Dvojnásobné potvrdenie m1 spôsobí, že sa nedozvieme o strate m2

Konverzácia tromi správami- $< \mathbf{data}, \mathbf{m} >$, $< \mathbf{ack} >$, $< \mathbf{close} >$

1. NCP A **send** $< \mathbf{data}, \mathbf{m} >$
2. NCP B receive <data, m>, deliver m, **send** $< \mathbf{ack} >$
3. NCP A receive <ack>, notify, **send** $< \mathbf{close} >$, close
4. NCP B receive <close>, close



1. NCP A **send** $< \mathbf{data}, \mathbf{m1} >$
2. NCP B receive <data, m1>, deliver m1, **send** $< \mathbf{ack} >$
3. NCP A receive <ack>, notify, **send** $< \mathbf{close} >$, close
4. DN <close> sa stratí
5. NCP A **send** $< \mathbf{data}, \mathbf{m2} >$
6. DN <data, m2> sa stratí
7. NCP B retransmit <ack> (krok 2)
8. NCP A receive <ack>, notify, **send** $< \mathbf{close} >$, close
9. NCP B receive <close>, close

strata <ack> nevedie k duplikovaniu; strata <close> môže spôsobiť re-send <ack>

Konverzácia tromi číslovanými správami- $<\mathbf{data},\mathbf{m},\mathbf{x}>$, $<\mathbf{ack},\mathbf{x},\mathbf{y}>$, $<\mathbf{close},\mathbf{x},\mathbf{y}>$

1. NCP A **send** $<\mathbf{data},\mathbf{m},\mathbf{x}>$
2. NCP B receive <data, m,x>, deliver m, **send** $<\mathbf{ack},\mathbf{x},\mathbf{y}>$
3. NCP A receive <ack,x,y>, notify, **send** $<\mathbf{close},\mathbf{x},\mathbf{y}>$, close
4. NCP B receive <close,x,y>, close

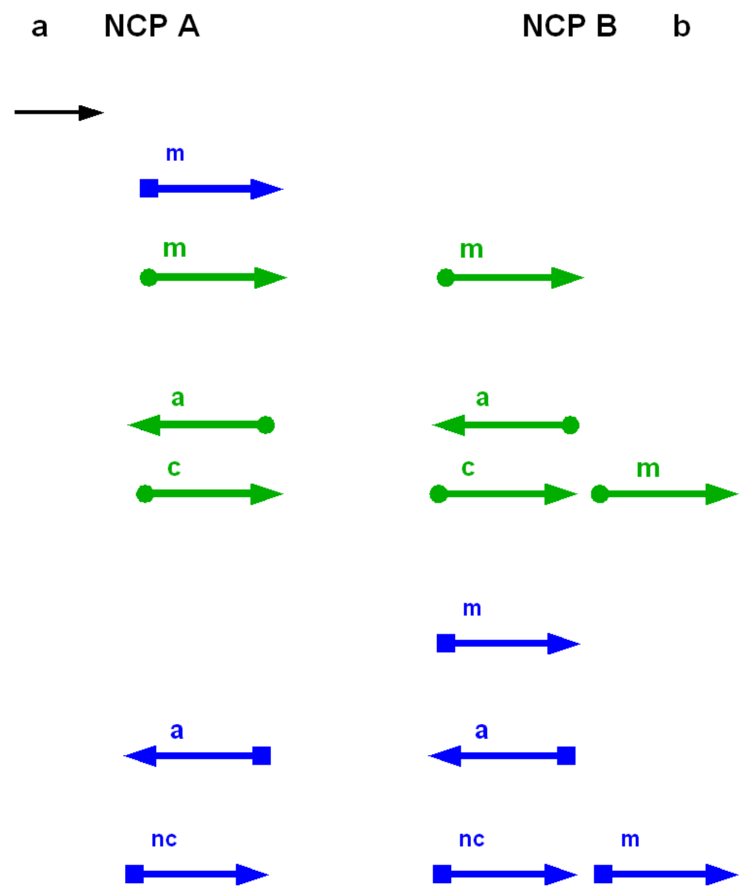
Overíme platnosť <data, m,x>

1. NCP A **send** $<\mathbf{data},\mathbf{m},\mathbf{x}>$
2. NCP B receive <data, m,x> **send** $<\mathbf{ack},\mathbf{x},\mathbf{y}>$
3. NCP A receive <ack,x,y>, notify, **send** $<\mathbf{close},\mathbf{x},\mathbf{y}>$, close
4. DN <close,x,y> sa stratí
5. NCP B timeout, retransmit <ack,x,y>
6. NCP A receive <ack,x,y>, reply <nocon,x,y>
7. NCP B receive <nocon,x,y>, deliver m, close

B musí doručiť dáta aj keď A nepotvrdí spojenie s x,y - môže nastať duplikácia

Môže nastať **duplikácia**

1. NCP A **send < data, m, x >**
2. NCP A timeout,
   retransmit <data, m,x>
3. NCP B receive <data, m,x>(step 2)
   send < ack, x, y1 >
4. NCP A receive <ack,x,y1>, notify,
   send < close, x, y1 >, close
5. NCP B receive <close,x,y1>,
   deliver m, close
6. NCP B receive <data, m,x>(step 1),
   send < ack, x, y2 >
7. NCP A receive <ack,x,y2>,
   reply < nocon, x, y2 >
8. NCP B receive <nocon,x,y2>,
   deliver m, close



Konverzácia štyrmi správami

$\mathbf{send} < \mathbf{open}, \mathbf{x}, \mathbf{y} >, \mathbf{send} < \mathbf{data}, \mathbf{x}, \mathbf{y} >, \mathbf{send} < \mathbf{agree}, \mathbf{x}, \mathbf{y} >, \mathbf{send} < \mathbf{ack}, \mathbf{x}, \mathbf{y}$

Vzájomná dohoda na id. číslach konverzácie pred doručením dát

1. NCP A $\mathbf{send} < \mathbf{data}, \mathbf{m}, \mathbf{x} >$
2. NCP B receive <data, m,x>, $\mathbf{send} < \mathbf{open}, \mathbf{x}, \mathbf{y} >$
3. NCP A receive <open,x,y>, $\mathbf{send} < \mathbf{agree}, \mathbf{x}, \mathbf{y} >$
4. NCP B receive <agree,x,y>, deliver m, $\mathbf{send} < \mathbf{ack}, \mathbf{x}, \mathbf{y} >$, close
5. NCP A receive <ack, x,y>, notify, close

Stále dochádza k duplikácii, keď NCP padne. Dá sa to zmodifikovať tak, aby A zazna-menalo a ukončilo po prijatí <noncon,x,y>; tým zabránime duplicite, ale môžu nastat straty

. . . □

Model distribuovaných výpočtov

Distribuovaný systém

sieť – G=(V, E), |V|=n procesorov, |E|=m kanálov

Procesor

lokálna pamäť

komunikácia so susedmi výmenou správ

správanie popísané stavmi, riadené správami

Formálny popis

Burns(1980), Lynch, Fischer(1981), Angluin(1980)

Predpoklady modelu distribuovaných výpočtov

⇒ **Asynchrónnosť**

procesory

komunikácia

poradie správ

⇒ **Nie je centrálne riadený**

⇒ Procesory majú jednoznačné **identifikačné čísla**

(anonymné siete – bez IČ)

⇒ Komunikácia procesorov **výmenou správ**

⇒ **Čas** lokálneho výpočtu zanedbateľný vzhľadom k času potrebnému na prenos

⇒ Znalosť o **topológii**

striktne lokálna (zoznam susedov)

lokálna + štruktúra siete (kruh, strom, ...)

⇒ **Plne distribuovaný systém** - každý má rovnaký algoritmus

Prechodový systém $S = (C, \longrightarrow, I)$, kde

$\mathbf{C}$ je množina konfigurácií

$\longrightarrow$ je prechodová relácia, $\gamma \;\longrightarrow\; \delta$

$\mathbf{I}, \mathbf{I} \;\subseteq\; \mathbf{C}$, je množina počiatočných konfigurácií

Vykonanie (execution) S je maximálna postupnosť $E = (\gamma_0, \gamma_1, \gamma_2, ...)$, kde $\gamma_0 \in I, \forall i\; \gamma_i \longrightarrow \gamma_{i+1}$

Terminálna konfigurácia γ- neexistuje $\delta :\; \gamma \longrightarrow \delta$

δ je **dosiahnuteľná z** γ ($\gamma \rightsquigarrow \delta$; $\gamma \longrightarrow^* \delta$)

$$\exists \gamma = \gamma_0, \gamma_1, \gamma_2, ...\gamma_k = \delta,\; \gamma_i \longrightarrow \gamma_{i+1}\; \forall 0 \;\leq\; i < k$$

δ je **dosiahnuteľná**, ak $\gamma \rightsquigarrow \delta$ pre $\gamma \in I$

- DS= množina procesov + komunikačný podsystém
- Prechodový systém-proces; konfigurácia-**stav**; prechod-**udalosť**
- Udalosť – interná, send, receive

Lokálny algoritmus procesu

$(Z, I, \mapsto^i, \mapsto^s, \mapsto^r)$, kde

Z-množina stavov

I-počiatočné stavy ($I \subseteq Z$)

$\mapsto^i \subseteq Z \times Z$ **interná udalosť**

$\mapsto^s \subseteq Z \times M \times Z$ **send**; M je množina možných správ

$\mapsto^r \subseteq Z \times M \times Z$ **receive**

$$c \mapsto d \Leftrightarrow (c, d) \in \mapsto^i \text{ alebo } \exists m \in M((c, m, d) \in \mapsto^s \cup \mapsto^r)$$

Distribuovaný algoritmus pre množinu procesov $\mathbb{P} = \{ p_1, \ldots, p_N\}$ je množina lokál-nych algoritmov, jeden pre každý procesor z $\mathbb{P}$

Prechodový systém indukovaný distribuovaným algoritmom pre $\mathbb{P} = \{ p_1, \ldots p_N\}$ poc
asynchrónnou komunikáciou je $\mathbf{S} = (\mathbf{C}, \rightarrow, \mathbf{I})$, kde

1. $\mathbf{C} = \{(c_{p1}, \ldots, c_{pN}, M) : (\forall p \in \mathbb{P} : c_p \in Z_p), M \in \mathcal{M}(M)\}$

2. $\rightarrow = (\cup_{p\in\mathbb{P}} \rightarrow_p)$,kde $\rightarrow_p$ je množina dvojíc

   (2a) $(c_{pi}, c'_{pi}) \in \mapsto^i_{pi},\ M1 = M2$

   (2b) $\exists m \in M(c_{pi}, m, c'_{pi}) \in \mapsto^s_{pi},\ M2 = M1 \cup m$

   (2c) $\exists m \in M(c_{pi}, m, c'_{pi}) \in \mapsto^r_{pi},\ M1 = M2 \cup m$

3. $\mathbf{I} = \{(c_{p1}, \ldots, c_{pN}, M) : (\forall p \in \mathbb{P} c_p \in I_p) \wedge M = \emptyset\}$

Prechodový systém indukovaný distribuovaným algoritmom pre $\mathbb{P} = \{ p_1, \ldots p_N\}$ poc
synchrónnou komunikáciou je $\mathbf{S} = (\mathbf{C}, \rightarrow, \mathbf{I})$, kde

1. $\mathbf{C} = \{(c_{p1}, \ldots , c_{pN}) : (\forall p \in \mathbb{P} : c_p \in Z_p)\}$

2. $\rightarrow \; = \; (\bigcup_{p\in\mathbb{P}} \rightarrow_p) \; \cup \; (\bigcup_{p,q\in\mathbb{P}:p\neq q} \rightarrow_{pq})$, kde

   $\rightarrow_p$ je množina dvojíc

   $$(c_{p1}, \ldots , c_{pi}, \ldots , c_{pN}), \; (c_{p1}, \ldots , c'_{pi}, \ldots , c_{pN}) : \; (c_{pi}, c'_{pi}) \in\mapsto^i_{pi}$$

   $\rightarrow_{pq}$ je množina dvojíc

   $$(\ldots , c_{pi}, \ldots , c_{pj}, \ldots), \; (\ldots , c'_{pi}, \ldots , c'_{pj}, \ldots) :$$
   $$\exists m \in M (c_{pi}, m, c'_{pi}) \in\mapsto^s_{pi} \& (c_{pj}, m, c'_{pj}) \in\mapsto^r_{pj}$$

3. $\mathbf{I} = \{(c_{p1}, \ldots , c_{pN}) : (\forall p \in \mathbb{P} c_p \in I_p)\}$

Overovanie vlastností prechodových systémov

Bezpečnosť a životnosť sú predikátmi na množine konfigurácií

Požiadavka bezpečnosti/fairness (tvrdenie P je pravdivé v **každej** konfigurácii každé-ho vykonania algoritmu)

Požiadavka životnosti/liveness (tvrdenie P je pravdivé v **nejakej** konfigurácii každého vykonania algoritmu)

Invarianty

$S = (C, \rightarrow, I)$, P, Q sú predikáty. Píšeme

$\{\mathbf{P}\} \rightarrow \{\mathbf{Q}\} \Leftrightarrow (\forall\, \gamma \rightarrow \delta$ ak $P(\gamma)$ tak $Q(\delta))$

Definícia 1 *Tvrdenie P je* ***invariantom*** *systému S, ak*

- $\forall\, \gamma \in I,\ P(\gamma)$
- $\{P\} \rightarrow \{P\}$

Fakt 1 *Ak P je invariantom S, tak P platí v každej konfigurácii každého vykonania S*

Fakt 2 *Nech Q je invariant S, $Q \Rightarrow P(\forall\gamma \in C)$. Potom P platí v každej konfigurácii každého vykonania*

Definícia 2 *Nech S je prechodový systém, P,Q tvrdenia. Hovoríme, že P je **Q-derivát**, ak*

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

Fakt 3 *Ak Q je invariant a P je Q-derivát, tak $Q \ \wedge \ P$ je invariant.*

Q je invariant, $Q(\gamma)$ platí $\forall \gamma \in I$ & (def) $\forall \gamma \in I,\ Q(\gamma) \ \Rightarrow\ P(\gamma)$

$$\Downarrow$$

$$\{P(\gamma)\},\ \text{ preto } \{Q(\gamma) \wedge P\gamma\}$$

Nech $\gamma \ \rightarrow \ \delta,\ Q(\gamma) \wedge P(\gamma)$; Q je invariant (teda $Q(\delta)$)
Keďže $\{Q \wedge P\} \ \rightarrow \ \{Q \Rightarrow P\}$, tak $Q(\delta) \Rightarrow P(\delta)$

Teda $Q(\delta) \wedge P(\delta)$

Životnosť - tvrdenie je pravdivé v **nejakej** konfigurácii každého vykonania

Nech S je prechodový systém, P (cieľ) predikát.
term-predikát, ktorý je (ne)pravdivý v (ne)terminálnych konfiguráciách

Definícia 3 *Systém S **končí korektne** ak predikát* $(term \Rightarrow P)$ *v S vždy platí.*
Uviaznutie *- terminálna konfigurácia, cieľ P sa nedosiahol*

Dobre založená množina - množina s usporiadaním, pre kt. neexistuje nekonečná klesajúca postupnosť

Čiastočné usporiadanie $(W, <)$ je dobre založené, ak neexistuje nekonečná klesajúca postupnosť $w1 > w2 > \ldots$

Nech S je prechodový systém, P tvrdenie. Funkcia f z C do dobre založenej W sa nazýva **norma** (vzhľadom na P) ak $\forall\, \gamma \rightarrow \delta :\ (\ \mathbf{f}(\gamma) >\ \mathbf{f}(\delta)\ \text{ alebo } \mathbf{P}(\delta))$

Fakt 4 *Nech $S = (C, \rightarrow, I)$, P (cieľ) predikát. Ak S skončí korektne a norma f existuje tak P platí v niektorej konfigurácii každého vykonania.*

Dôkaz: Nech $E = (\gamma_0, \gamma_1, \gamma_2, ...)$ je vykonanie.

1. E je **konečná**, $E = (\gamma_0, \gamma_1, \gamma_2, ...\gamma_k)$, potom $P(\gamma_k)$ platí
2. E je **nekonečná**.

   Uvažujme najdlhší prefix $E' = (\gamma_0, \gamma_1, \gamma_2, ...)$ taký, že P neplatí.

   Nech $F = (f(\gamma_0), f(\gamma_1), f(\gamma_2), ...)$.

   f je norma $\Rightarrow$ $f(\gamma_0) > f(\gamma_1) > f(\gamma_2) > ....$

   Je to klesajúca postupnosť, preto je konečná;

   $F = (f(\gamma_0), f(\gamma_1), f(\gamma_2), ..., f(\gamma_k))$ a $P(\gamma_{k+1})$

Poradie(causality order)

Nech $E = (\gamma_0, \gamma_1, \gamma_2, ...)$ je vykonanie. Relácia $\prec$, ktorej hovoríme **poradie udalostí** v E, je najmenšia relácia, ktorá spĺňa

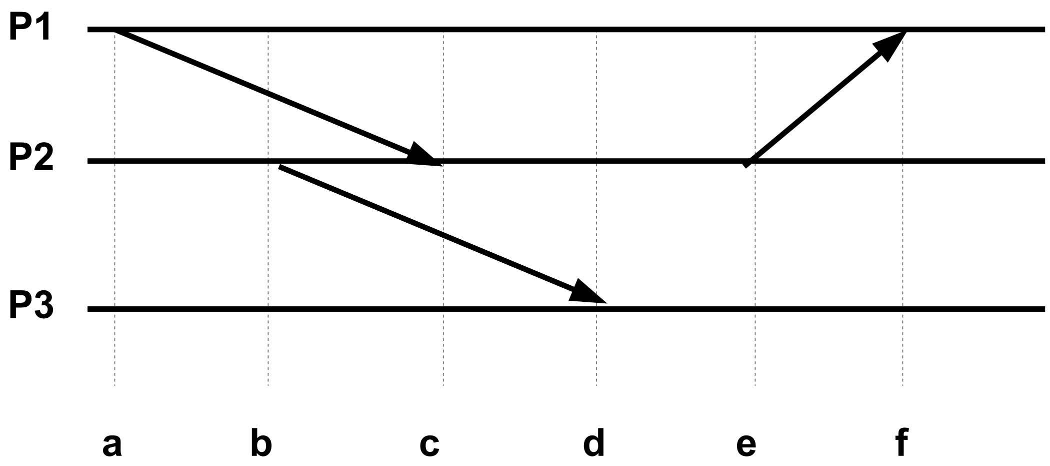
- Ak e, f sú rôzne udalosti toho istého procesora, e sa vykonalo pred f, tak $e \prec f$
- Ak s je send, r odpovedajúca receive udalosť, tak $s \prec r$
- $\prec$ je tranzitívna

Čiastočné usporiadanie $\mathbf{a} \le \mathbf{b}$: $a \prec b \vee a = b$

Súbežne bežiace procesy $(\mathbf{a}||\mathbf{b})$ ani $a \le b$, ani $b \le a$

Time-space diagram

Ku každému vykonaniu $E = (\gamma_0, \gamma_1, \gamma_2, ...)$ existuje odpovedajúca postupnosť udalost
$F = (e_0, e_1, e_2, ...)$ takých, že e_i spôsobilo prechod $\gamma_i \rightarrow \gamma_{i+1}$



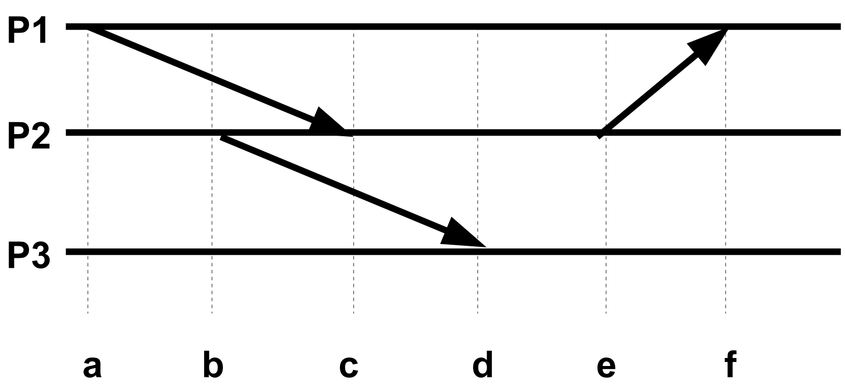
$$F = (a, b, c, d, e, f)$$

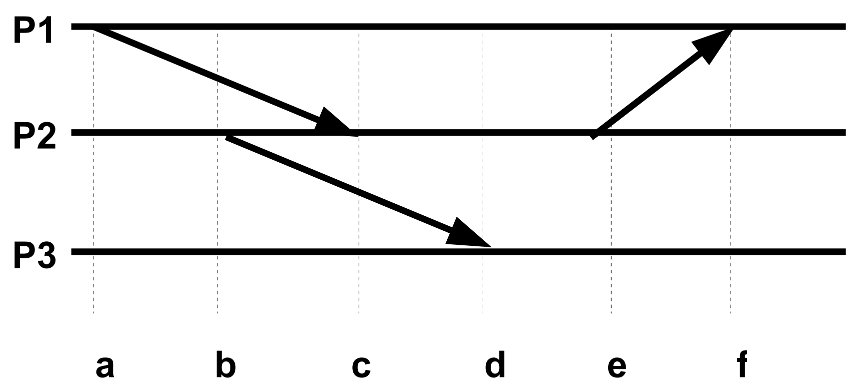
Závislé a nezávislé udalosti

Tvrdenie 1 *Nech γ je konfigurácia, e_p a e_q sú udalosti rôznych procesov p,q, aplikovateľne v γ. Potom e_p je aplikovateľné v $e_q(\gamma)$, e_q je aplikovateľné v $e_p(\gamma)$ a $e_p(e_q(\gamma)) = e_q(e_p(\gamma)$*

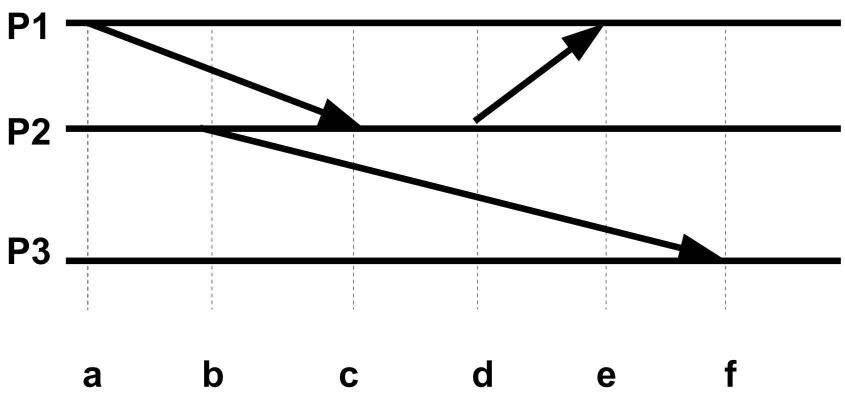
- Správy sú jednoznačne spojené s konkrétnym procesorom
- e_p, e_q nemôžu byť odpovedajúce send/receive udalosti

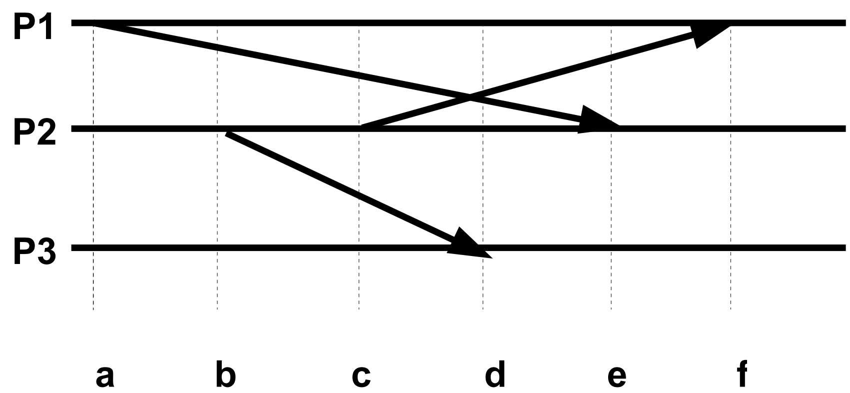
Rôzne vykonania





$$a<c\,,\,b<d\,,\,e<f\,,\,c<e$$





Ekvivalencia vykonaní

$E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ je vykonanie,

$\mathcal{E} = (e_0, e_1, e_2, \ldots)$ odpovedajúca postupnosť udalostí,

$\mathcal{F} = (f_0, f_1, f_2, \ldots)$ konzistentná permutácia $\mathbf{f_i \leq f_j \Rightarrow i \leq j}$

F vykonanie s postupnosťou udalostí $\mathcal{F}$.

Tvrdenie 2 *$\mathcal{F}$ definuje jednoznačné vykonanie F, ktoré začína v počiatočnej konfiguráci E, má rovnaký počet udalostí ako E, posledná konfigurácia F je poslednou konfiguráciou E.*

Ekvivalentné vykonania definujú triedu ekvivalencie $\approx$

Výpočet distribuovaného algoritmu je trieda ekvivalencie vykonaní toho algoritmu

Logické hodiny Môžeme definovať hodiny, ktoré budú vyjadrovať poradie/causality

Nech Θ je funkcia z množiny udalostí do **usporiadanej** množiny. Θ sú **logické hodiny**, ak **a $\prec$ b implikuje $\boldsymbol{\Theta}$(a) $<$ $\boldsymbol{\Theta}$(b)**

- **Poradie**- ak je vykonanie definované postupnosťou udalostí $\mathcal{E} = (e_0, e_1, e_2, ...)$, definujeme $\Theta(e_i) = i$
- **Real-time hodiny** – procesor má hardwareové hodiny (vlastne to kazí definíciu DS)
- **Vektorové hodiny** – $a \prec b \Leftrightarrow \Theta(a) < \Theta(b)$ (súbežne bežiace procesy majú neporovnateľné hodnoty)

  $\Theta_v(a) = (a_1, a_2, ..., a_n)$, kde

  a_i je maximálne číslo udalosti e v procesore p_i, pre ktorú $e \leq a$

Lamportove hodiny

$\Theta(a)$ definujeme ako dĺžku najdlhšej postupnosti $(e_0, e_1, e_2, ..., e_k)$ takej, že
$e_0 \prec e_1 \prec e_2 \prec ... \prec e_k = a$

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

Realizácia

Θ_p číslo poslednej udalosti v procesore p

Θ_m číslo udalosti-posielania správy m

Topológia siete

Sieť = množina procesorov+komunikačný subsystém (graf)

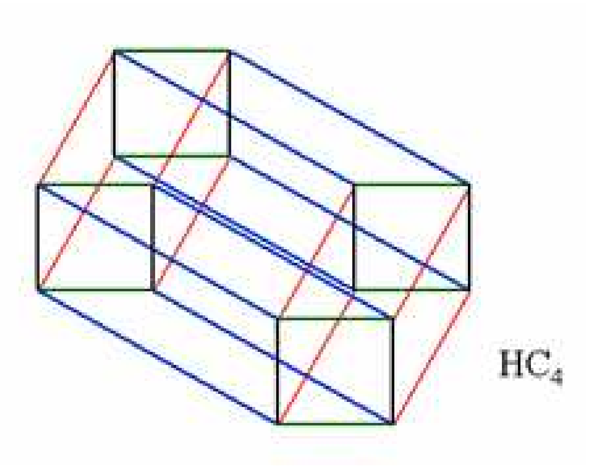
- kruh (tokren ring),
- strom (rýchle výpočty; kostra - prenositeľné)
- hviezda (master-slave; zraniteľnosť centra)
- úplný graf
- hyperkocka

$$HC_n = (V, E), N = 2^n, V = \{0,1\}^n$$

$$((a_0, a_1, ..., a_n), (b_0, b_1, ..., b_n)) \in V$$

$$\Updownarrow$$

$$\exists! i \; a_i \neq b_i$$



HC_4

Vlastnosti kanálov

- **spoľahlivosť** (správy sa nestrácajú/neduplikujú/nevznikajú-predpokladáme)
- **FIFO** (zachováva poradie, v akom doň vstupujú-nepredpokladáme)
- **kapacita** (def. predpokladá neobmedzenú kapacitu)

Vedomosti procesu/procesora

Topologická informácia

- Čísla procesorov, polomer, topológia
- Zmysel pre orientáciu(konzistentné označenie hrán so smermi)

Identita procesu (pomenovaná sieť, anonymná sieť)

Susedné identity (nepredpokladáme; priame vs. nepriame adresovanie)

Zložitosť DA

Počet správ (total)

Bitová(väčšinou správy dĺžky logn)

Časová
idealizované meranie času

- Čas spracovania je nulový
- Prenosový čas najviac jednotka času

Časová zložitosť je čas spotrebovaný vzhľadom na tieto predpoklady

Priestorová(vzhľ. na jeden proces)

Predpoklady

Ak nepovieme inak

- Sieť je silne súvislá
- Komunikácia je asynchrónna
- Kanály sú FIFO
- Kanály nestrácajú správy
- Správy sú doručené v konečnom čase

Routing – smerovanie packetov do cieľa

routovacia tabuľka

$RT_u(v) = w \Leftrightarrow$ packet smerujúci do vrchola v je z vrchola u posunutý do w

routovanie

1. výpočet routovacích tabuliek
2. samotné posúvania packetov podľa nich

Kritériá dobrého routovania

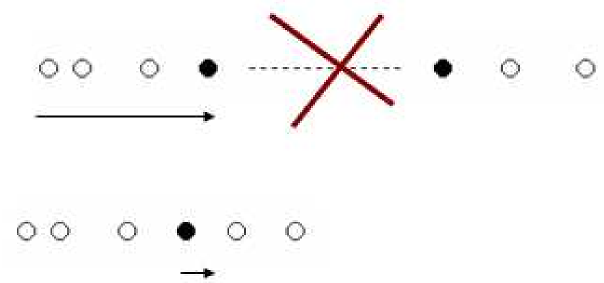
- **Korektnosť** (každý packet je v konečnom čase doručený)
- **Zložitosť** počítania tabuliek (čas, pamäť, správy)
- **Efektívnosť**(používanie optimálnych ciest)
- **Robustnosť** (prepočítavanie tabuliek pri zmene v topológii)
- **Adaptabilita** tabuliek, ktorá eliminuje/znižuje preťaženosť kanálov.
- Predpokladáme, že vrcholy sú **obsluhované rovnomerne**

Smerovanie packetov podľa cieľa

optimány algoritmus existuje, ak

- cena posielania je nezávislá od momentálneho používania
- cena zreťazenia ciest je súčet ciest jednotlivých ciest
- graf neobsahuje cyklus zápornej dĺžky

Fakt 5 *Ak existuje cesta z u do v, tak existuje jednoduchá optimálna cesta.*



sink tree

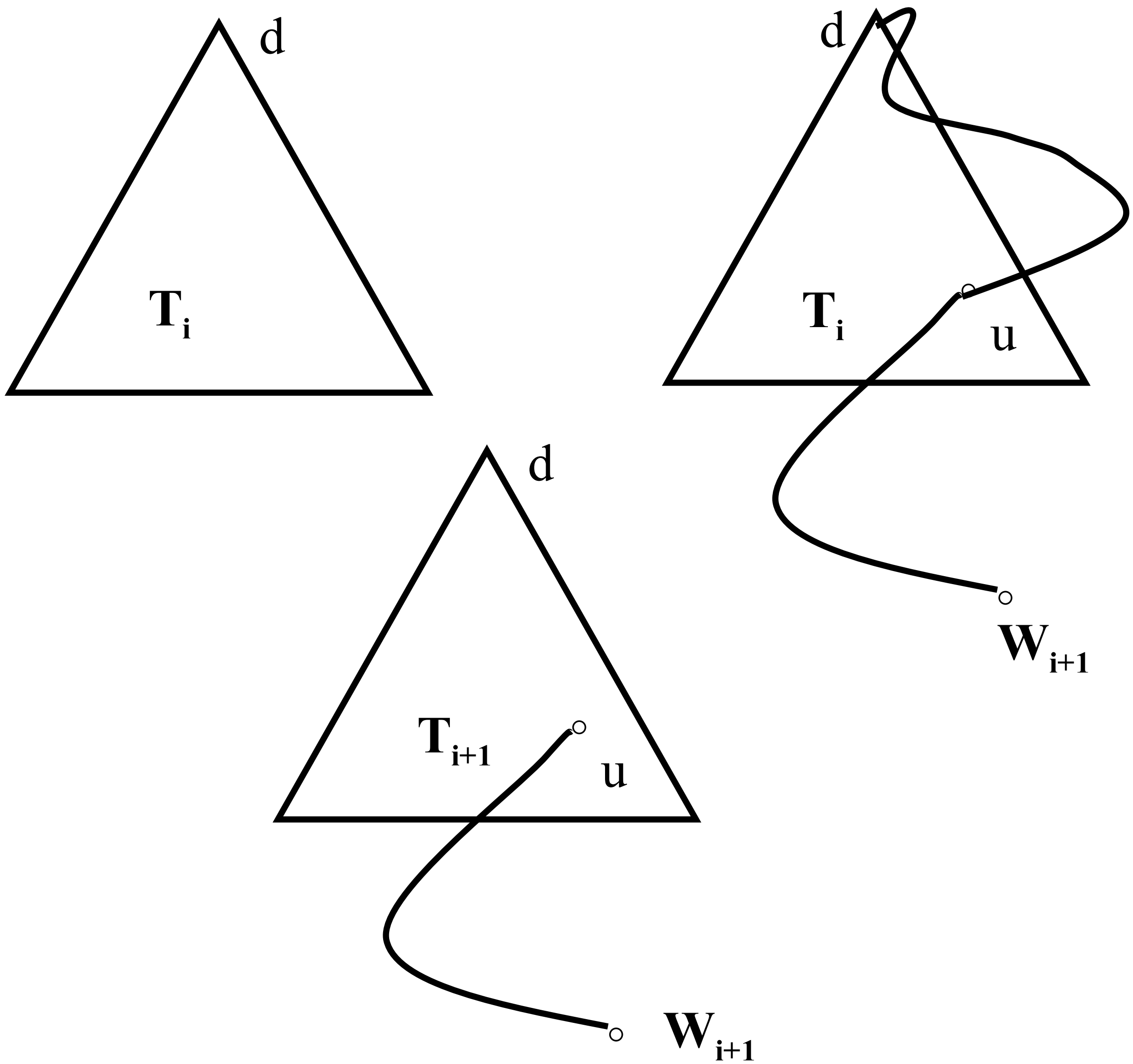
Fakt 6 *Nech $G = (V, E)$ je súvislý graf. Pre každý vrchol/cieľ $d \in V$ existuje strom $T_d = (V, E_d)$, $E_d \subseteq E$ tak, že pre každý vrchol $v \in V$ je $v - d$ cesta v strome T_d najkratšou $v - d$ cestou v grafe G.*

Konštrukcia – iteratívne $T_0,\ T_1, \ldots, T_m$

- $T_0 = (\{d\}, \emptyset)$
- $T_i = (V_i, E_i)$, kde T_i je podgraf grafu G, T_i je podgraf grafu T_{i+1}
- $\forall w \in V_i$ je $w - d$ cesta v T_i optimálnou $w - d$ cestou v G.

  Nech $w_{i+1} \notin V_i$ a nech $\mathbf{w_{i+1} - d}$ je optimálna cesta v G. Nech u je prvý vrchol na tejto ceste, ktorý *patrí* do V_i.

  T_{i+1} vznikne z T_i **pripojením** cesty $\mathbf{w_{i+1} - u}$ k vrchulu $\mathbf{u}$.
- proces skončí vytvorením stromu T_m so všetkými vrcholmi. Vtedy $T_d := T_m$


d
$\mathbf{T_i}$
d
$\mathbf{T_i}$
u
$\mathbf{W_{i+1}}$
d
$\mathbf{T_{i+1}}$
u
$\mathbf{W_{i+1}}$

Destination based routing

- Vypočítaj sink-tree T_d $\forall d \in V$
- $T_u[d]$ je otec vrcholu u v sink-tree s koreňom d

Algoritmus(packet prijatý/vygenerovaný vrcholom u smeruje do d)
if $d = u$ **then** *deliver* lokálne
else send packet do $T_u[d]$

všetky najkratšie cesty

$G = (V, E)$ je ohodnotený orientovaný graf bez záporných cyklov; váhy $\omega(\mathbf{u}, \mathbf{v})$ môžu byť aj záporné

Vychádzame zo **sekvenčného Floyd-Warshall alg.**

$\mathbf{d^S(u, v)}$ dĺžka najkratšej $u - v$ cesty s *vnútornými* vrcholmi z množiny S

inicializácia:

$d^S(u, u) \leftarrow \emptyset$

$d^\emptyset(u, v) \leftarrow \omega(u, v)$ pre $(u, v) \in E, u \neq v$

$d^\emptyset(u, v) \leftarrow \infty$ pre $(u, v) \notin E, u \neq v$

iterácia:

$d^{S \cup \{w\}}(u, v) = min\{d^S(u, v), d^S(u, w) + d^S(w, v)\}$

Floyd-Warshall

$S \leftarrow \emptyset$
for all $u, v \in V$ **do**
  if u=v **then** $D_U[v] \leftarrow \emptyset$
  else if $(u, v) \in E$ **then** $D_u[v] \leftarrow \omega(u, v)$
  else $D_u[v] \leftarrow \infty$
                  ▷ inicializáci

while $S \neq V$ **do**
  pick $w \in V \setminus S$;   ▷ w-pivo

  for all $u, v \in V$ **do**
    $D_u[v] \leftarrow \min\{D_u[v], D_u[w] + D_w[v]\}$ ▷ nie je kratšie ísť cez w ?

  $S \leftarrow S \cup \{w\}$ ▷ zväčšenie množiny S

sekvenčná zložitosť $O(n^2) + n \cdot O(n^2) = \mathbf{O(n^3)}$

Paralelná verzia - **TOUEG**ov algoritmus

Predpoklady

– cykly sú **pozitívne**

– na začiatku výpočtu všetky **vrcholy poznajú** V

– vrcholy **poznajú** svojich susedov (**Neigh**$_{\mathbf{u}}$)

– výber pivota **podľa dopredu zvolenej** stratégie

S$_{\mathbf{u}}$ množina (vnútorných) vrcholov
D$_{\mathbf{u}}$ pole váh/vzdialeností
Nb$_{\mathbf{u}}$ pole vrcholov(susedov na najkratších cestách)

Jednoduchý Toueg - pre vrchol u;

$S_u := \emptyset$;
for all $v \in V$ **do**
    inicializácia;

while $S \neq V$ **do**
    voľba pivota w;
    distribúcia $D_w[v]$ *po sieti*;
    for all $v \in V$ **do**

        aktualizácia $D_u[v], Nb_u[v]$;

    $S_u := S_u \cup \{w\}$

algoritmus jednoduchýTOUEG

inicializácia

$S_u \leftarrow \emptyset$;
for all $v \in V$ **do**
  if $u = v$ **then**
    $D_u[v] \leftarrow 0$;
    $Nb_u[v] \leftarrow \perp$
  else if $(u, v) \in E$ **then**
    $D_u[v] \leftarrow \omega(u, v)$;
    $Nb_u[v] \leftarrow v$
  else
    $D_u[v] \leftarrow \infty$;
    $Nb_u[v] \leftarrow \perp$

iterácia

while $S \neq V$ **do**
  pick $\mathbf{w} \in \mathbf{V} \setminus \mathbf{S}$

  if $u = w$ **then** **broadcast** $\mathbf{D_w}$
  else receive D_w

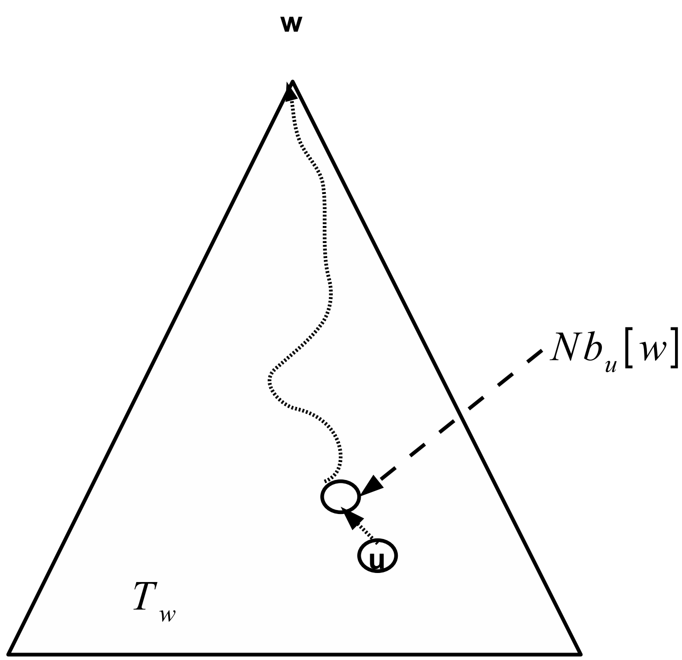
  for $v \in V$ **do**
    if $D_u[w] + D_w[v] < D_u[v]$ **then**
      $D_u[v] \leftarrow D_u[w] + D_w[v]$
      $Nb_u[v] \leftarrow Nb_u[w]$
  $S_u \leftarrow S_u \cup \{w\}$

Fakt 7 *Algoritmus* jednoduchýToueg *skončí po N opakovaniach hlavného cyklu. Po jeho skončení je v $D_u[v]$ dĺžka najkratšej u-v cesty, $Nb_u[v]$ obsahuje identifikáciu prvého vrcholu na nej (ak cesta existuje) alebo má nedefinovanú hodnotu.*

Vylepšenie:

- ak $D_u[w] = \infty$, u nemení svoje tabuľky vo fáze s pivotom w
  $\Rightarrow$ broadcast $D_w[v]$ iba po strome T_w

- ak $D_u[w] < \infty$, u pozná otca, ale nie synov v T_w, preto
  v oznámi svojmu susedovi u, či je jeho synom v T_w alebo nie



u oznámi $Nb_u[w]$, že je jeho syn vo fáze s pivotom w

Inicializácia D_u, Nb_u

Začni fázu (pre pivota w)

for $x \in Neigh_u$ **do** ▷ Vytvor v T_w spojenie otec-syn

  if $Nb_u[w] = x$ **then** send $\langle ys, w\rangle$ do x

  else send $\langle nys, w\rangle$ to x

$numRec_u \leftarrow 0$ ▷ u musí prijať $|Neigh_u|$ správ

while $numRec_u < |Neigh_u|$ **do**

  receive $< ys, w >$ alebo $< nys, w >$

  $numRec_u \leftarrow numRec_u + 1$

if $D_u[w] < \infty$ **then** ▷ Zúčastni sa fázy (s pivotom w)

  if $u \neq w$ **then** receive $< dtab, w, d >$ od $Nb_u[w]$

  for $x \in Neigh_u$ **do**

    if $< ys, w >$ bol prijatý od x **then**

      send $< dtab, w, d >$ do x ; ▷ lokálny update

  $S_u \leftarrow S_u \cup \{w\}$

$\langle ys, w\rangle \backslash \langle nys, w\rangle$ – **som** \ **nie som** tvoj syn vo fáze s pivotom w

$\langle dtab, w, D\rangle$– broadcast riadka D_w vo fáze s pivotom w

Veta 1 *Algoritmus vylepšený Toueg skončí. Po jeho skončení je v* $D_u[v]$ *dĺžka najkratše*
$u - v$ *cesty,* $Nb_u[v]$ *obsahuje identifikáciu prvého vrchola na nej (ak cesta existuje) alebo*
má nedefinovanú hodnotu.

Nech **W** *je počet bitov postačujúci na kódovanie mien aj váh. Potom počas vykonania*
algoritmu sa prekomunikuje

- $\mathbf{O(N)}$ *správ - kanál*
- $\mathbf{O(N|E|)}$ *správ - celkovo*
- $\mathbf{O(N^2W)}$ *bitov – kanál*
- $\mathbf{O(N^3W)}$ *bitov – celkovo* $N^2.NW + 2N|E| = O(N^3W)$

Pamäť potrebná v jednotlivých procesoroch je $\mathbf{O(NW)}$ *bitov.*

Od sekvenčných algoritmov k distribuovaným

- premenné sekvenčného algoritmu sa distribuujú medzi vrcholy;výpočty sú lokálne
- ak treba vzdialenú premennú, vyžaduje sa komunikácia
- znižovanie komunikácie využitím vlastností sekvenčného algoritmu

Ešte stále nám ostali **zlé vlastnosti Touegovho** algoritmu

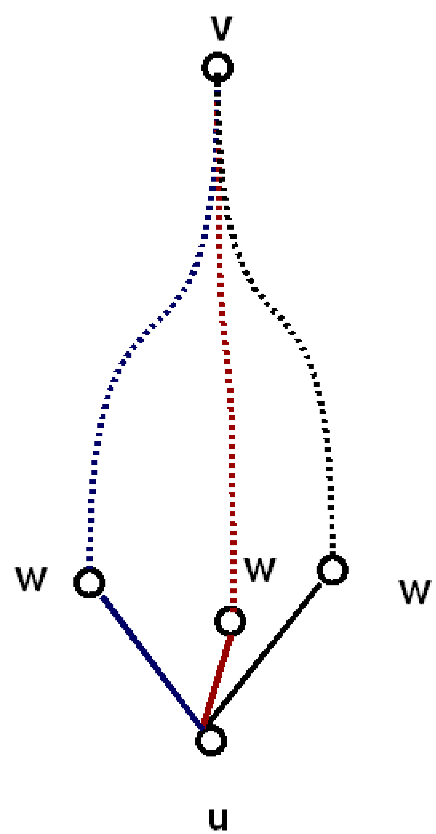
- kvôli pivotovi vyžadujeme znalosť všetkých vrcholov
- vyžadujeme informáciu, ktorá nie je dostupná ani vo vrchole ani v jeho susedoch ?$d(u,w) + d(w,v) < d(u,v)$?

$$\Downarrow$$

Chandy-Misra

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

- Min hop
- Lokálna komunikácia
- Výpočty nezávislé
- Vzhľadom na lokálnosť jednoduchšia adaptabilita



sekvenčný Dijkstra– z každého do **s**

inicializácia:
$T \leftarrow \{s\}; S := V \setminus s; d_s \leftarrow 0; Nb_s \leftarrow s$

$d_u \leftarrow \omega(u,s); Nb_u \leftarrow s$ pre $(u,s) \in E$
$d_u \leftarrow \infty; Nb_u \leftarrow \perp$ pre $(u,s) \notin E$

výpočet
while $S \neq \emptyset$ **do**
  nech v je taký, že $d_v = min\{d_w, w \in S\}$;
  $T \leftarrow T \cup \{v\}; S \leftarrow S - \{v\}$;
  for $w \in S$ **do**
    if $d_w < \omega(w,v) + d_v$ **then**
      $d_w \leftarrow \omega(w,v) + d_v$
      $Nb_w \leftarrow v$

budujeme vlastne sink-tree s koreňom s:

- vo vrchole u bude informácia d_u a Nb_u
- S, T sú v grafe uložené implicitne; vrchol, ktorý "dostáva trvalú značkuöznámi všetkým ostatným, aká je jeho vzdialenosť ku koreňu
- každé vylepšenie oznamuje vrchol všetkým ostatným

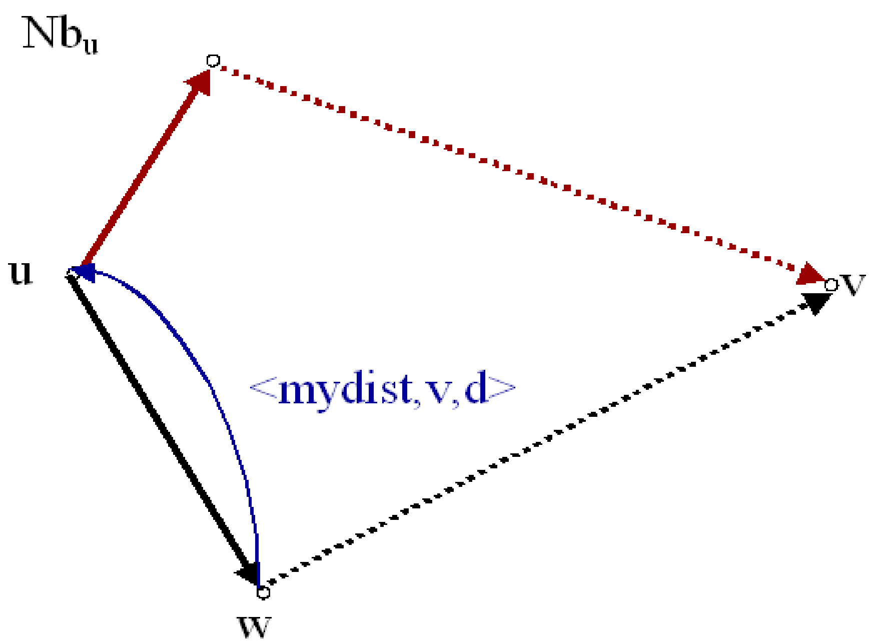
Inicializácia - $D_u[w] := \infty; Nb_u := \perp$

Výpočet pre vrchol v (koreň v kostry sa "zobudil")
$D_v[v] := 0;$
$\forall w \in Neigh_v$ send $< mydist, v, 0 >$ do w

while nie je koniec **do**
  for $w \in V$ **do**
    prijmi správu;
    spracuj prijatú správu

Spracovanie $< \mathbf{mydist}, \mathbf{v}, \mathbf{d} >$

receive $< mydist, v, d >$ od w;
if $d + \omega(uv) < D_u[v]$ **then begin**
$D_u[v] := d + \omega(uv); Nb_u[v] := w;$
forall $x \in Neigh_u$ **do** send $< mydist, v, D_u[v] >$ do x
end;

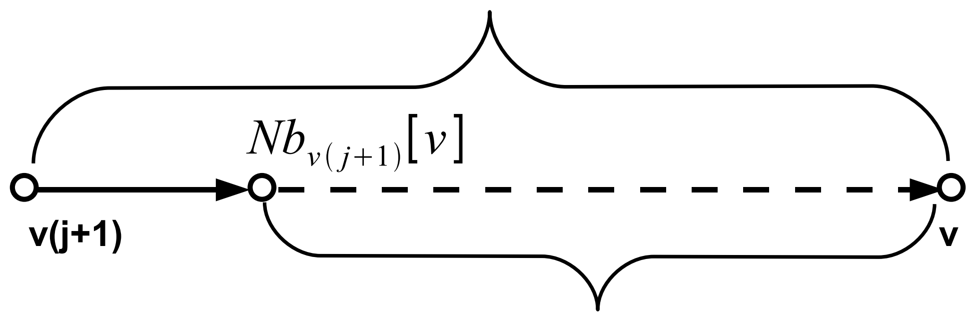


Fakt 8 *V každej realizácii algoritmu Chandy-Misra(CHM) sa dosiahne konfigurácia, v ktorej* $\forall u \in V\, D_u[v] \leq d(u, v)$.

Kvôli <u>dôkazu</u> tohto tvrdenia si očíslujeme vrcholy v T_v tak, že otec má menšie číslo ako syn. Potom tvrdenie dokážeme matematickou indukciou.

IP $\forall j \leq N-1$ dosiahneme konfiguráciu, v ktorej

$$D_{v(j+1)}[v] = \omega(v(j+1), vi) + d(vi, v) = d(v(j+1), v)$$



$$Nb_{v(j+1)}[v] = vi, 1 \leq j \wedge d_{vi}[v] = d(vi, v)$$

zložitosť algoritmu Chandy-Misra

Každý vrchol posiela správy svojim susedom vždy, keď sa dozvie o vylepšení. Keďže každý vrchol si môže vylepšiť info max $\mathbf{N-1}$ krát, dostávame pre **počet správ** vymenených pri výpočte **najkratších ciest do v**

$$O(N\Sigma_{u\in V}|Neig_u|) = \mathbf{O(N|E|)}$$

Ak počítame najkratšie cesty ku každému vrcholu (teda N krát opakujeme vysvetlený postup), dostávame

- $O(N^2|E|)$ správ/ $O(N^2|E|W)$ bitov pre výpočet **všetkých najkratších ciest**
- $O(N^2)$ **správ** / $O(N^2W)$ **bitov na kanál**

Netchange - najkratšie cesty v prítomnosti chýb kanálov

predpoklady

- Vrcholy poznajú veľkosť siete
- Kanály sú FIFO
- Vrcholy sa dozvedia o poruche/oprave susedných kanálov
- Cena cesty je počet kanálov na nej

Od algoritmu vyžadujeme

1. Ak po konečnom počte zmien ostane topológia siete nezmenená, algoritmus v konečnom čase skončí.
2. V okamihu, keď algoritmus skončí, pre vrchol u platí:
   $Nb_u[v]$=local, ak $u = v$
   $Nb_u[v] = w$, kde w je prvý vrchol na najkratšej $u - v$ ceste
   $Nb_u[v] = \perp$ ak $u - v$ cesta neexistuje

premenné

Neigh$_{\mathbf{u}}$- aktuálna množina susedov vrchola u

D$_{\mathbf{u}}$[v] – odhad $d(u, v)$

Nb$_{\mathbf{u}}$[v] – prvý vrchol na ceste s $D_u[v]$

ndis$_{\mathbf{u}}$[w, v] – odhad $d(w, v)$

Inicializácia v u:

for all $w \in Neigh_u, v \in V$ **do** $ndis_u[w, v] := N$

for all $v \in V$ **do** $D_u[v] \leftarrow N; Nb_u[v] \leftarrow \perp$

$D_u[u] \leftarrow 0; Nb_u[u] \leftarrow$local;

for all $w \in Neigh_u$ **do** send $< mydist, u, 0 >$ do w;

$\langle \mathbf{mydist}, \mathbf{v}, \mathbf{d} \rangle$ – vrchol, od ktorého prišla, oznamuje svoju vzdialenosť d od vrchola v
$\langle \mathbf{fail}, \mathbf{v} \rangle$ – správa prijatá kanálom od vrchola w oznamuje poruchu kanálu (v, w)
$\langle \mathbf{repair}, \mathbf{v} \rangle$ – kanál (v, w) bol obnovený

$\langle \mathbf{repair}, \mathbf{v} \rangle$

receive $\langle repair, v \rangle$
$Neigh_u \leftarrow Neigh_u \cup \{w\}$
for all $v \in V$ **do**
  $ndist_u[w, v] \leftarrow N$
  send $\langle mydist, v, D_u[v] \rangle$ do w

$\langle \mathbf{fail}, \mathbf{v} \rangle$

receive $\langle fail, w \rangle$
$Neigh_u \leftarrow Neigh_u - \{w\}]$
for all $v \in V$ **do**
  Recompute(v)

$\langle \mathbf{mydist}, \mathbf{v}, \mathbf{d} \rangle$

receive $\langle mydist, v, d \rangle$
$ndist_u[w, v] \leftarrow d$
Recompute(v)

Recompute(v)

if $v = u$ **then**
$D_u[v] \leftarrow 0$
$Nb_u[v] \leftarrow local$

else
$D \leftarrow 1 + \min\{ndist_u[w, v] \mid w \in Neigh_u\}$
$\triangleright$ nech w_D je ten vrchol, pre ktorý sa dosahuje minimum
if $D < N$ **then**
$D_u[v] \leftarrow D$
$Nb_u[v] \leftarrow w_D$
else
$D_u[v] \leftarrow N$
$Nb_u[v] \leftarrow \perp$

if $D_u[v]$ sa zmenilo **then**
for all $x \in Neigh_u$ **do** send $\langle mydist, D\rangle$ do x

Netchange – korektnosť

hovoríme, že $\mathbf{up(u, w)}$ platí $\Leftrightarrow$ obojsmerný kanál (u, w) pracuje korektne

$\mathbf{Q_{uw}}, \mathbf{Q_{wu}}$ – rady, ktoré modelujú kanál (u, w)

Korektnosť sa ukáže pomocou invariantov.

$\mathbf{P(u, v, w)} \equiv$

1. $up(u, w) \Leftrightarrow w \in Neigh_u$
2. $up(u, w) \wedge Q_{uw}$ obsahuje $< mydist, v, d >$ potom v poslednej $d = D_w[v]$
3. $up(u, w) \wedge Q_{uw}$ neobsahuje $< mydist, v, d >$ potom $ndist_u[w, v] = D_w[v]$

$\mathbf{L(u, v)} \equiv$

4. $u = v \Rightarrow (D_u[v] = 0 \wedge Nb_u[v] = local)$
5. $(u \neq v \wedge \exists w \in Neigh_u \ ndist_u[w, v] < N - 1) \Rightarrow$
   $(D_u[v] = 1 + min\{ndist_u[w, v]\} = 1 + ndist_u[Nb_u[v], v])$
6. $(u \neq v \wedge \forall w \in Neigh_u \ ndist_u[w, v] \geq N - 1) \Rightarrow (D_u[v] = N \wedge [Nb_u[v] = \perp)$

Netchange – stabilizácia

predikát $\quad\quad \textbf{stable} \equiv \forall u, w (up(u,w) \Rightarrow Q_{uw}$ neobsahuje $< mydist, v, d >)$

Hovoríme, že **konfigurácia γ je stabilná**, ak $stable(\gamma)$.

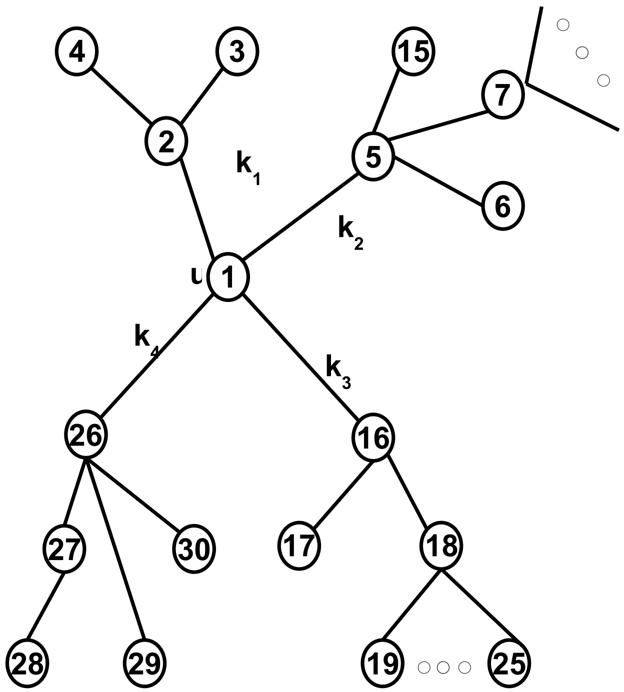
Nech $\quad\quad \mathbf{t_i}(\gamma) = \sharp < mydist, .., i > \; v \; \gamma + \sharp(u,v)$, pre ktoré v γ $D_u[v] =$

norma $f(\gamma) = (t_0(\gamma), t_1(\gamma), \ldots, t_N(\gamma))$

Fakt 9 *Spracovanie $< mydist, v, i >$ znižuje hodnotu f.*

Fakt 10 *Ak po konečnom počte topologických zmien ostane topológia siete nemenná, algoritmus dosiahne stabilnú konfiguráciu.*

Routing s kompaktnými tabuľkami



- také adresovanie, ktoré minimalizuje veľkosť routovacej tabuľky
- Tree-labeling -kanálom routujeme interval adries
- topológia je strom

vo vrchole 1

adresa	2	3	...	20	...	30
kanál	k_1	k_1		k_3		k_4

alebo kompaktnejšie

kanál	k_1	k_2	k_3	k_4
adresy	2,4 (2...4)	5, 15 (5...15)	16,25 (16...25)	26,30(26...30)

Tree labeling — Santoro/Khatib

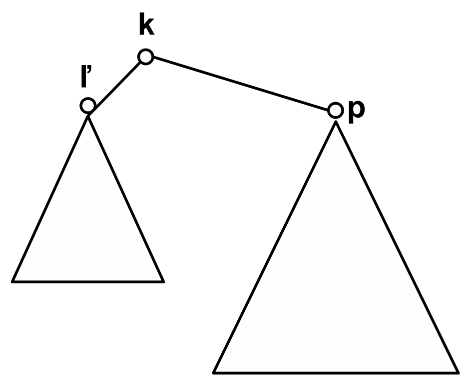
vrcholy – $0, 1, \ldots N-1$

$$\textbf{cyklický interval } [a,b) = \begin{cases} \{a, a+1, \ldots, b-1\} & \text{ak a < b;} \\ \\ \{a, a+1, \ldots, N, 0, 1, \ldots, b-1\} & \text{ak b < a.} \end{cases}$$

Ak $[a_1, b_1), \ldots, [a_k, b_k)$ sú cyklické intervaly na odchádzajúcich kanáloch vrchola u, tak
na routovaciu tabuľku v u stačí pamäť $\mathbf{k} \cdot \log \mathbf{N}$

Fakt 11 *Vrcholy stromu T možno očíslovať tak, že množina adries routovaných cez jeden výstupný kanál tvorí cyklický interval.*

napr.preorder – koreň, ľavý, pravý



Tree labeling – Správa (m, d) pre d (prijatá/generovaná) v u

if $d = u$ **then** spracuj lokálne
else
  nech w je také, že $d \in [a_w, b_w)$
  send (m, d) po kanáli v

– v stromoch je optimálne
– v súvislých grafoch použijeme kostru.
G je graf, T jeho kostra, $d_G(u, v), d_T(u, v$
cena najkratšej cesty medzi u, v v grafe G
resp. kostre T.

Fakt 12 *: Neexistuje ohraničenie pomeru* $\frac{d_T(u,v)}{d_G(u,v)}$. *A to ani v prípade min-hop.*

kruh

Majme súvislý grafu so symetrickým ohodnotením hrán ($d(u, v) = d(v, u)$). Označme
D_G polomer grafu G. Potom

Fakt 13 *Existuje taká kostra* T *grafu* G*, že* $\forall u, v \in V \ d_T(u, v) \leq 2 \cdot D_G$.

Nech T je optimálny sink-tree pre vrchol w z centra. O vzdialenosti $d_T(u, v)$ platí

$$\begin{aligned} d_T(u, v) &\leq d_T(u, w) + d_T(w, v) \\ &= d_G(u, w) + d_G(w, v) \leq 2 \cdot D_G \end{aligned}$$

Nevýhody routovania po strome

– nepoužívame hrany mimo stromu (mrhanie prostriedkami)
– používame len stromové hrany (zahltenie)
– výpadok jednej hrany spôsobí kolabs systému

ILS(interval labeling scheme) je značkovanie vrcholov a hrán siete značkami zo Z_N tak že

- vrcholy majú rôzne značky
- v uzle majú všetky kanály rôzne značky

Hovoríme, že ILS je platná, ak routovaním podľa nej každá správa v konečnom čase dosiahne svoj cieľ.

Veta 2 *Ku každému súvislému grafu existuje platná ILS.*

$\mathbf{l_w}$ značka vrchola w
$\alpha_{\mathbf{uv}}$ značka hrany (u, v)
$\mathbf{k_w} = l_w + |T(w)|$ $T(w)$ označuje podstrom s koreňom w

- vrcholy značkujeme preorderom(koreň w, podstrom $l_w, l_{w+1}, \ldots, l_{w+|T(w)|-1}$)
- značka hrana (u, w)

  (u, w) je frond (nestromová hrana), tak $\alpha_{uw} = l_w$

  w je T-syn vrchola u, tak $\alpha_{uw} = l_w$

  w je T-otec vrchola u, $k_u \neq N$ alebo u nemá frond do koreňa, tak $\alpha_{uw} = k_u$

  w je T-otec vrchola u, $k_u = N$ a u má frond do koreňa, tak $\alpha_{uw} = l_w$.

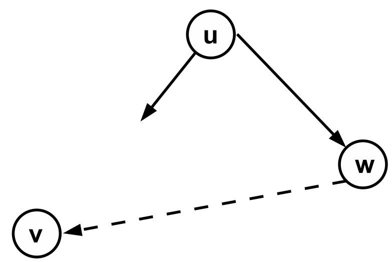
Nech **lca(u,v)** je najbližší spoločný predchodca vrcholov u, v. Potom správe, ktorá sa nachádza vo vrchole u a smeruje do vrchola v, priradíme hodnotu

$$f_v(u) = (-lca(u,v), l_u)$$

vrchol u posúva správu pre vrchol v do w

Lema 1 *V situácii, keď podľa DFS ILS posúva vrchol u správu pre vrchol v do vrchola w platí*

$$\begin{aligned} l_u > l_v &\Rightarrow l_w < l_u \\ l_u < l_v &\Rightarrow l_w \le l_v \\ l_u < l_v &\Rightarrow f_v(w) < f_v(u) \end{aligned}$$



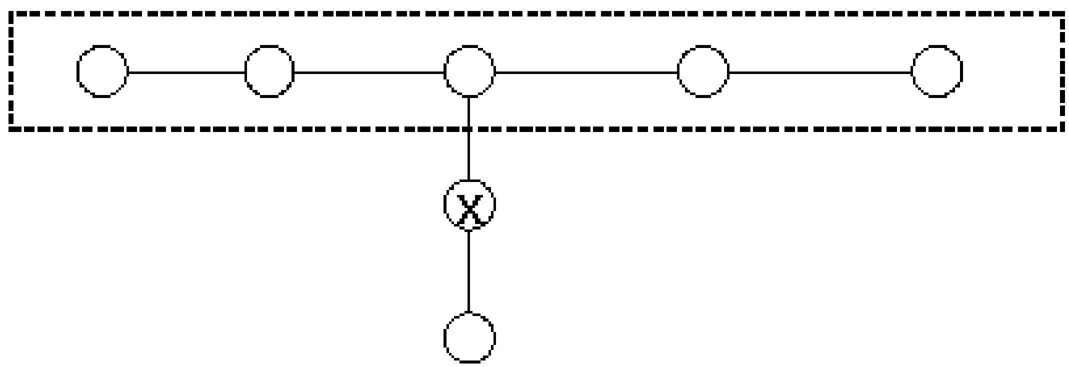
Fakt 14 *Existuje sieť G taká, že pre každú platnú ILS existujú v G dva vrcholy u, v také, že na doručenie packetu z u do v treba aspoň $\frac{3}{2}D_G$ hopov.*

Fakt 15 *Existuje min-hop ILS pre kruh.*

Fakt 16 *Existuje min-hop ILS pre $n \times n$ mriežku.*

lineárne ILS schémy – intervaly priradené jednotlivým kanálom sú lineárne

Fakt 17 *Existuje sieť, pre ktorú neexistuje platná lineárna ILS*



Fakt 18 *Existuje min-hop ILS pre hyperkocku*

PLS(prefix labeling scheme) nad abecedou Σ je značkovanie vrcholov a hrán siete reťaz-cami nad Σ^* tak, že

- vrcholy majú rôzne značky
- v uzle majú všetky kanály rôzne značky

PLS je *platná*, ak routovaním podľa nej každá správa v konečnom čase dosiahne svoj cieľ.

packet pre d **(prijatý/generovaný) v** u

if $d = l_u$ **then** doruč lokálne
else
nech α_i je najdlhšie číslo kanála také, že $\alpha_i < d$;
send packet pre d po kanáli α_i

Veta 3 *Ku každému súvislému grafu G existuje platná PLS.*

Konštrukcia tree–PLS Nech T je kostra grafu G.

- koreň je označený ϵ
- ak w je syn u tak $l_w = l_u.a_i$
  ak $u_1, \ldots, u_k$ sú rôzni synovia vrchola u, tak $a_1 \neq \ldots \neq a_k$ a $l_{u_i} = l_u.a_i$
- ak uw je frond, tak $\alpha_{uw} = l_w$
- ak w je syn u, tak $\alpha_{uw} = l_w$
- ak w je otec u, tak $\alpha_{uw} = \epsilon$ okrem prípadu, keď u má frond do koreňa; vtedy $\alpha_{uw} = l_u$

Fakt 19 *Pre popísanú tree–PLS platí*

1. *ak $u \neq v$ tak v u existuje kanál označený prefixom l_v*
2. *v situácii, keď u posiela správu do v cez w*
   (a) *ak $u \in T[v]$, tak w je predchodca u, resp. v je predchodca w*
   (b) *ak u je predchodca v tak w je predchodca v bližšie k v (ako u)*
   (c) *ak $u \notin T[v]$, tak w je predchodca v alebo $d_T(w, v) < d_T(u, v)$*

Dôsledok 1 *Pre každý graf G s priemerom D_G existuje PLS, ktorá každý packet doruč*
$\leq 2D_G$ hopov.