Algoritmy v prítomnosti chýb

robustnosť - korektne pracujúce procesy po chybe iných procesov pracujú korektne
stabilizácia - hoci chyba v inom procese môže spôsobiť dočasne nekorektný výpočet korektného procesu, systém sa v konečnom čase z chyby zotaví

Predpokladáme, že **kanály sú spoľahlivé, chybné môžu byť procesory**

Modelovanie chýb

1. ***od začiatku mŕtve*** *procesory*
2. ***fatálne*** chyby -po nejakom počte krokov procesor prestane pracovať
3. ***byzantské*** správanie - procesor nepracuje korektne

Hierarchia modelov chýb $\mathbf{1} \subseteq \mathbf{2} \subseteq \mathbf{3}$

Zmiešané chyby a chyby časovania $\mathbf{2t} < \mathbf{n}$, $\mathbf{3t} < \mathbf{n}$, $\mathbf{2b} + t < n$

Úlohy o dohode - každý korektný procesor nastaví dohodnutú výstupnú hodnotu

terminovanie všetky korektné procesy niekedy rozhodnú

konzistentnosť vzťah medzi rozhodnutiami jednotlivých procesov

1. **o zhode/dohode** - spoľahlivé procesy sa dohodnú na bitovej zhode y; $x_i = y_i, y_i = y$ pre tie i, kde p_i je spoľahlivý procesor
2. **o interaktívnej konzistencii** - každý spoľahlivý procesor vypočíta rovnaký vektor hodnôt $y = (y_1, \ldots, y_n)$
3. **o generáloch** - na začiatku má jeden procesor P inicializovanú hodnotu x. Spoľahlivé procesory skončia s rovnakou hodnotou y. Ak je P spoľahlivý, $y = x$

netrivialita výstup závisí od vstupu

Praktické problémy

1. distribuované databázy - potvrdenie/zamietnutie zmeny
2. distribuovaný výpočet -dohoda o vstupoch (pri replikácii)
3. výber
4. približná dohoda (výstup je z $\{1, \ldots, k\}$; môžu sa líšiť o 1)

Robustnosť – asynchrónne výpočty

Fischer, Lynch,Paterson ***deterministický*** algoritmus zhody ***nie je odolný*** voči fatálnym chybám(ani jedinej)

Moran, Wolfstahl rozšírené na výber – *pravdepodobnostné* (alebo synchronizované) algoritmy môžu byť odolné voči fatálnym chybám

Bracha, Toueg pravdepodobnostná zhoda odolná voči $\mathbf{t} < \mathbf{N/2}$ fatálnym alebo $\mathbf{t} < \mathbf{N/3}$ byzantským chybám

Robustnosť– synchronizácia *deterministický* algoritmus odolný voči $\mathbf{t} < \mathbf{N/3}$ byzantským, $\mathbf{t} < \mathbf{N}$ fatálnym chybám

vplyv autentifikácie

Robustnosť vs.Stabilizácia

S prevencia pred dočasne nekorektným správaním

R prevencia pred trvalou chybou obmedzeného počtu procesorov

G,P existuje spôsob, ako z robustného spraviť robustný a stabilizujúci

A,H neexistuje deterministické robustné a stabilizujúce riešenie pre výber a veľkosti kruhu postupnosť $\sigma = (e_1, \ldots, e_k)$ je **aplikovateľná** na konfiguráciu γ, ak

e_1 je aplikovateľná na γ, e_2 je aplikovateľná na $e_1(\gamma)$,...

$\gamma \hookrightarrow^{\sigma} \delta$ $\quad$ $\sigma(\gamma) = \delta$ $\quad$ $S \subseteq P$ a σ obsahuje **iba** udalosti z S, tak $\gamma \hookrightarrow_{\mathbf{S}} \delta$

Fakt 1 *Ak sú σ_1, σ_2 aplikovateľné na γ a* neexistuje *proces, ktorý sa zúčastňuje aj σ_1 aj σ_2, tak σ_2 je aplikovateľná na $\sigma_1(\gamma)$, σ_1 je aplikovateľná na $\sigma_2(\gamma)$ a $\sigma_2(\sigma_1(\gamma)) = \sigma_1(\sigma_2(\gamma))$*

t-f-korektný výpočet – aspoň $N - t$ procesov realizuje nekonečný výpočet a každá správa poslaná korektnému procesu je prijatá

odolnosť t- maximálny prípustný počet chybných procesov

t-f-robustná zhoda:

terminovanie v každom t-f-korektnom výpočte všetky korektné procesy rozhodnú

zhoda ak v dosiahnuteľnej konfigurácii $y_p \neq b \neq y_q$, p, q sú korektné procesy, potom $y_p = y_q$

netrivialita pre $v = 0, 1$ existuje dosiahnuteľná konfigurácia $\gamma(v)$, taká že pre nejaký korektný proces $p(v)$, $y_{p(v)} = v$

konfigurácia:

v-rozhoduje	ak $y_p = v$
rozhoduje	ak 0-rozhoduje, alebo 1-rozhoduje
je ***v-valentná***	ak všetky z nej dosiahnuteľné rozhodujúce konfigurácie v-rozhodujú
bivalentná	dosiahnuteľná je aj 0-rozhodujúca aj 1-rozhodujúca
univalentná	alebo 0-valentná alebo 1-valentná

Fakt 2 *Ak γ je dosiahnuteľná konfigurácia t-f-robustného algoritmu, S podmnožina aspoň $N - t$ procesov, tak existuje z nej dosiahnuteľná rozhodujúca konfigurácia δ, $\gamma \hookrightarrow_S \delta$.*

fork/vetvenie – taká konfigurácia γ $t - f$-robustného algoritmu, že existuje množina procesov T, $|T| \leq t$ a konfigurácie γ_0, γ_1 také, že

$$\gamma \hookrightarrow_T \gamma_0, \gamma \hookrightarrow_T \gamma_1 \text{ a } \gamma_v \text{ je } v\text{-valentná}$$

Lema 1 *Vo výpočte korektného algoritmu neexistuje dosiahnuteľné vetvenie.*

Lema 2 *Nech A je* 1-f-robustný *algoritmus zhody. Potom existuje počiatočná konfigurácia, ktorá je bivalentná.*

Označme ***krok*** s prijatie a spracovanie správy, resp. spontánny krok.

Lema 3 *Nech γ je dosiahnuteľná bivalentná, s aplikovateľný pre proces p v γ. Potom existuje taká postupnosť udalostí σ, že s je aplikovateľný v $\sigma(\gamma)$ a $s(\sigma(\gamma))$ je bivalentná.*

Veta 1 ***Neexistuje*** *asynchrónny deterministický* 1-f-robustný *algoritmus zhody.*

Ak áno, vychádzajúc z bivalentnej počiatočnej konfigurácie γ_0 by sme MI vedeli skonštruovať nekonečne dlhý výpočet, ktorý nerozhoduje. (γ_i potom $\exists s_i$, že aplikovaním s_i dostaneme bivalentnú γ_{i+1})

$\square$

zoslabenie

▷ modelu chyby: od začiatku mŕtve procesy – deterministický výber

▷ koordinácie – premenovanie je riešiteľné

▷ determinizmu – randomizované riešenie aj pre byzantské chyby

▷ terminovania – stačí terminovanie len korektných – riešenie aj pre byzantské chyby

▷ synchronizácia

Od začiatku mŕtve procesy- zhoda na podmnožine korektných procesov(Fischer, Lynch, Paterson) (($t < N/2$) ⇒ dohoda, výber)

orientovaný graf procesov, ktoré nie sú mŕtve -tzv. knot (silne súvislý bez odchádzajúcich hrán)

premenné: $Succ_p, Alive_p, Rcvd_p$ – množiny procesorov, iniciované $\emptyset$

begin

shout $\langle name, p \rangle$; ▷ $\forall q \in P$ send $\langle name, p \rangle$ do q

while $\sharp Succ_p < L$ **do**

receive $\langle name, q \rangle$

$Succ_p \leftarrow Succ_p \cup \{q\}$

end while

shout $\langle pre, p, Succ_p \rangle$;

$Alive_p \leftarrow Succ_p$;

while $Alive_p \not\subseteq Rcvd_p$ **do**

receive $\langle pre, q, Succ \rangle$;

$Alive_p \leftarrow Alive_p \cup Succ \cup \{q\}$;

$Rcvd_p \leftarrow Rcvd_p \cup \{q\}$

end while

vypočítaj knot

end

Zhoda a výber

knot K $\Rightarrow$ leader má max. identitu
$\Rightarrow$ broadcast v K

korektný proces sa stane leadrom $\Rightarrow$ dohoda (broadcast leadrovho vstupu)

Distribuovaná úloha - $T : X^N \rightarrow \mathcal{P}(D^N)$

Algoritmus je **t-f-robustný pre T** ak:

terminovanie v každom $t - f-$korektnom výpočte každý korektný proces terminuje

konzistentnosť ak sú všetky procesy korektné, vektor rozhodnutia $\overrightarrow{d} \in T(\overrightarrow{x})$

zhoda - $D_{\text{zhoda}} = \{(0, \ldots, 0), (1, \ldots, 1)\}$

výber - $D_{\text{výber}} = \{(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)\}$

približná zhoda - $D_\epsilon = \{(d_1, \ldots, d_N) : max(d_i) - min(d_j) \leq \epsilon\}$

premenovanie - $D_{\text{premenovanie}} = \{(d_1, \ldots, d_N), i \neq j \Rightarrow d_i \neq d_j\}$

Deterministické premenovanie - v prípade $t < N/2$

Veta 2 *Pre $t \geq N/2$ t-f-korektný algoritmus na premenovanie **neexistuje**.*

Algoritmus ABND– $t < N/2$

- udržiava info o tých, ktorých videl
- pri každej zmene množiny mien ju zakričí ostatným $\Rightarrow$ usporiadanie na podmnožinách
- počíta, koľkokrát dostal info o aktuálnej množine; nech pre $V = V_p$ je to $N - t$, potom $V = V_p$ je *stabilná* množina
- menom je (s, r), kde $s = |V_p|, r$ poradie x_p vo V_p

Lema 4 *Stabilné množiny sú úplne usporiadané.*

Lema 5 *V každom t-f-korektnom výpočte dosiahne každý proces stabilnú množinu.*

Veta 3 *Algoritmus ABND rieši problém premenovania v prítomnosti $t < N/2$ fatálnych chýb, pričom mená prideľuje z množiny veľkosti $K = (N - t/2)(t + 1)$.*

ABND—Deterministické premenovanie v prítomnosti fatálnych chýb

premenné:

V_p množina identifikátorov;
c_p celé číslo

$V_p \leftarrow \{x_p\}$; $c_p \leftarrow 0$; shout $\langle set, V_p \rangle$;
while true **do**
  receive $\langle set, V \rangle$;
  if $V = V_p$ **then**
    $c_p \leftarrow c_p + 1$;
    if $c_p = N - t \wedge y_p = b$ **then** ▷ V_p je stabilná
      $y_p \leftarrow (\sharp V_p, rank(V_p, x_p))$
    end if
  else if $V \subseteq V_p$ **then** ignoruj
  else
    if $V_p \subset V$ **then** $c_p \leftarrow 1$
    else $c_p \leftarrow 0$;
    end if
    $V_p \leftarrow V_p \cup V$; shout $\langle set, V_p \rangle$
  end if
end while

Rozhodovací graf úlohy T

$\mathbf{G_T} = (\mathbf{V}, \mathbf{E})$, kde

$V = D_T$

$E = \{(\overrightarrow{d_1}, \overrightarrow{d_2}) : \overrightarrow{d_1}, \overrightarrow{d_2}$ sa líšia práve v jednej zložke $\}$

Netrivialita - pre každý vektor $\overrightarrow{d}$ existuje taká počiatočná konfigurácia, že sa procesy dohodnú na $\overrightarrow{d}$

Podľa toho, aký je G_T je úloha súvislá alebo nesúvislá.

Veta 4 ***Neexistuje*** *netriviálny 1-f-robustný rozhodovací algoritmus pre nesúvislú distribuovanú úlohu.*

Podmienka netriviality je dosť silná.

Dohoda v prítomnosti t fatálnych chýb - negatívny výsledok

Konvergencia: $\lim_{k\to\infty}$Pr[korektný proces po k krokoch nerozhodol]$= 0$

Nech $\mathbf{R(q, p, k)}$ je udalosť, keď v kole k proces p dostane od q jeho správu k-teho kola medzi prvými $N - t$ správami kola k.

Korektné časovanie

1. $\exists \epsilon > 0\ \forall p, q, k :\ \ Pr[R(p, q, k)] \geq \epsilon$
2. $\forall k$ a rôzne procesy p, q, r sú udalosti $R(q, p, k)$ a $R(q, r, k)$ nezávislé

Veta 5 ***Neexistuje*** *t-f-robustný protokol na výpočet dohody pre $t \geq N/2$.*

Ak by taký protokol P existoval, potom by platilo:

1. P má bivalentnú počiatočnú konfiguráciu
   Rozdeľme procesy do dvoch množín S, T veľkosti $\lfloor N/2 \rfloor, \lceil N/2 \rceil$
2. Ak γ je dosiahnuteľná konfigurácia, tak γ je alebo $S - 0-$valentná a $T - 0-$valentná, alebo $S - 1-$valentná a $T - 1-$valentná
3. P nemá dosiahnuteľnú bivalentnú konfiguráciu $\square$

Bracha/Toueg - Pravdepodobnostný algoritmus zhody v prítomnosti t fatálnych chýb

Nech $t < N/2$.

Inicializácia Každý korektný proces si náhodne zvolí hodnotu 1 alebo 0.

k**-te kolo**: Nech p je **korektný** proces

- p pošle $(k, val_p, weight_p)$ každému procesu (vrátane seba samého)
- p počká, kým dostane $N - t$ správ (k, b, w)
  Ak $w > N/2$, $val_p \leftarrow b$ (b je jednoznačné)
  Inak $val_p \leftarrow 0$ (ak väčšina volila 0), resp. $val_p \leftarrow 1$ (ak väčšina volila 1)
  Váha hodnoty val_p je počet správ, ktoré volili val_p.
- p ignoruje správu (l, b, w) ak $l < k$ a uloží si správu (l, b, w) ak $l > k$
- Ak $w > N/2$ pre viac ako t správ (l, b, w), rozhodne p hodnotou b.

Keď p rozhodne b, rozšíri správu $(k + 1, b, N - t)$ a $(k + 2, b, N - t)$ a terminuje.

while $y_p = b$ **do**
  $witness_p[0], witness_p[1], msgs_p[0], msgs_p[1] \leftarrow 0, 0, 0, 0$
  shout$\langle vote, round_p, value_p, weight_p\rangle$
  while $msgs[0] + msgs[1] < N - t$ **do**
    receive $\langle vote, r, v, w\rangle$
    if $r > round_p$ **then** send $\langle vote, r, v, w\rangle$ do p ▷ nasledujúce kolo
    else if $r = round_p$ **then**
      $msgs_p[v] \leftarrow msgs_p[v] + 1$
      if $w > N/2$ **then** $witness_p[v] \leftarrow witness_p[v] + 1$
      end if
    else ignoruj ▷ $r < round$
    end if
  end while
  if $witness_p[0] > 0$ **then** $value_p \leftarrow 0$
  else if $witness_p[1] > 0$ **then** $value_p \leftarrow 1$
  else if $msgs_p[0] > msgs_p[1]$ **then** $value_p \leftarrow 0$
  else $value + p \leftarrow 1$
  end if
  $weight_p \leftarrow msgs_p[value_p]$
  if $witness_p[value_p] > 1$ **then** $y_p \leftarrow value_p$
  end if
  $round_p \leftarrow round_p + 1$
end while
shout $\langle vote, round_p, value_p, N - t\rangle$ ▷ pomôž ostatným
shout $\langle vote, round_p + 1, value_p, N - t\rangle$

Proces potvrdzuje hodnotu, ak váha jeho voľby je $N - t$

Lema 6 *V žiadnom kole neexistujú dva procesy, ktoré potvrdzujú rôzne hodnoty.*

Lema 7 *Ak proces rozhodne, tak všetky korektné procesy rozhodnú rovnako v priebehu dvoch nasledujúcich kôl.*

Lema 8 $\lim_{k\to\infty}$ *Pr[v kole* $l \leq k$ *k rozhodnutiu nedošlo]=0.*

Lema 9 *Ak všetky procesy začnú so vstupom v, tak všetky procesy rozhodnú v v kole 2.*

Veta 6 *Pre* $t < N/2$ *je algoritmus* BT *pravdepodobnostný t-f-robustný algoritmus dohody.*

Algoritmus dohody v prítomnosti byzantských chýb

$\hookrightarrow$- odvodenie postupnosťou *korektných* krokov

t-b-robustný algoritmus - odolný voči t byzantským chybám

Veta 7 *Pre $t \geq N/3$* ***neexistuje*** *t-b-robustný algoritmus pre problém dohody*

- netrivialita $\Rightarrow$ existencia bivalentnej počiatočnej konfigurácie
  S, T sú také, že $|S| \geq N - t, |T| \geq N - t, |S \cap T| \leq t$
- dosiahnuteľná konfigurácia γ je alebo $S - 1-$valentná a $T - 1-$valentná
  alebo $S - 0-$valentná a $T - 0-$valentná
- neexistuje dosiahnuteľná bivalentná konfigurácia

Bracha/Toueg - pravdepodobnostná dohoda v prítomnosti t byzantských chýb

– Korektné procesy šíria svoju voľbu a čakajú na $N - t$ prichádzajúcich správ

– Korektný proces rozhodne b, keď dostane viac ako $\mathbf{\frac{N-t}{2} + t = \frac{N+t}{2} < N - t}$ b-hlasov

Komplikácia: byzantský proces môže rôznym procesom posielať rôzne voľby

Riešenie: Overovanie prichádzajúcich správ echo mechanizmom - voľba sa akceptuje, ak ju potvrdí aspoň $\frac{N-t}{2} + t = \frac{N+t}{2}$ echo správ

Bracha/Toueg - pravdepodobnostná dohoda v prítomnosti t byzantských chýb

Inicializácia Každý korektný proces si náhodne zvolí hodnotu 1 alebo 0.

***k*-te kolo**: Nech p je **korektný** procesor

- p pošle (in, k, val_p) každému procesu (vrátane seba samého)
- ak p dostane $(in, l, b), l \leq k$ od q, rozpošle (ec, q, l, b)
- p počíta prichádzajúce (ec, q, k, b) správy. Keď dostane viac ako $\frac{N+t}{2}$ takých správ, p akceptuje b-voľbu procesu q
- p sleduje, aby nedostal rovnakým kanálom násobné správy $(in, q, l, -), (ec, q, l, -)$-prichádzali by od byzantského procesu. Prvú z nich berie vážne.
- p ignoruje $(ec, q, l, b), l < k$ a uchováva $(in, q, l, b), (ec, q, l, b), l > k$
- kolo je ukončené, keď p akceptuje $N - t$ hlasov. Podľa väčšiny nastaví val_p
- Ak je viac ako $\frac{N+t}{2}$ akceptovaných hlasov b, p hlasuje za b

```
while true do
    for all v, q do msgs_p[v] ← 0; echos_p[q, v] ← 0
    end for
    shout ⟨vote, in, p, value_p, round_p⟩
    while msgs_p[0] + msgs_p[1] < N − t do
        receive ⟨vote, t, r, v, rn⟩ od q
        if ⟨vote, t, r, ∗, rn⟩ od q už prišlo then ignoruj        ▷ q opakuje, musí byť byzantinsky
        else if t = in a q ≠ r then ignoruj                        ▷ q klame, musí byť byzantinsky
        else if rn > round_p then send ⟨vote, t, r, v, rn⟩ do p
        else                                                       ▷ t ∈ {in, ec}
            if t = in then shout ⟨vote, ec, r, v, rn⟩
            end if
            if t = ec then
                if rn = round_p then
                    echos_p[r, v] ← echos_p[r, v] + 1;
                    if echos_p[r, v] = ⌊(N + t)/2⌋ + 1 then msgs_p[v] ← msgs_p[v] + 1
                    end if
                else ignoruj
                end if
            end if
        end if
    end while
    if msgs_p[0] > msgs_p[1] then value_p ← 0
    else value_p ← 1
    end if
    if msgs_p[value_p] > (N + t)/2 then y_p ← value_p
    end ifround_p ← round_p + 1
end while
```

korektnosť

- Ak korektný proces p v kole k akceptuje voľbu v korektného procesu r, tak korektný proces r v k-tom kole hlasoval za v.
- Ak korektné procesy p, q akceptujú v kole k voľbu procesu r, akceptujú rovnakú voľbu.
- Ak všetky korektné procesy začnú kolo k, tak akceptujú dostatočne veľa hlasov k tomu, aby kolo úspešne ukončili.
- Ak korektný proces rozhodne v kole k pre v, tak všetky korektné procesy volia v v kole k a každom ďalšom.
- $\lim_{k\to\infty}$ Pr[korektný proces p nerozhodol pred kolom k]=0.
- Ak všetky korektné procesy začnú so vstupom v, v konečnom čase rozhodnú pre v.

Veta 8 *Algoritmus BT II je pravdepodobnostným t-b-robustným algoritmom protokolom dohody pre* $t < N/3$.

Asynchrónny broadcast v prítomnosti t byzantských chýb

Ak je generál korektný, všetci dôstojníci nasledujú jeho rozkaz.

Veta 9 ***Neexistuje*** *1-b-robustný algoritmus asynchrónneho broadcastu, ktorý spĺňa konvergenciu, zhodu, závislosť a to ani vtedy, ak konvergenciu vyžadujeme iba v prípade, keď generál poslal aspoň jednu správu.*

Definícia 1 *t-b-robustný broadcastovací algoritmus je algoritmus s nasledujúcimi vlastnosťami*
Slabé terminovanie - *korektné procesy alebo všetky rozhodnú alebo nerozhodne žiaden. Ak je generál korektný, rozhodnú všetky korektné procesy.*

Zhoda- *ak korektné procesy rozhodnú, rozhodnú rovnako*

Závislosť- *Ak je generál korektný, všetky procesy rozhodnú jeho vstup.*

Asynchrónny broadcast

```
var  msgs_p[(in, ec, re), 0..1]   :integer   init:0

if general then shout ⟨vote, in, x_p⟩
end if
while y_p = b do
    receive ⟨vote, t, v⟩ od q
    if ⟨vote, t, v⟩ od q už prišlo then ignoruj                          ▷ q opakuje
    else if t = in a q ≠ g then ignoruj                 ▷ q simuluje g, je byzantinsky
    else msgs_p[t, v] ← msgs_p[t, v] + 1
        if t = in then
            if msgs_p[in, v]=1 then shout ⟨vote, ec, v⟩
            end if
        end if
        if t = ec then
            if msgs_p[ec, v] = ⌈(N + t)/2⌉ + 1 then shout ⟨vote, re, v⟩
            end if
        end if
        if t = re then
            if msgs_p[re, v] = (t + 1) then shout ⟨vote, re, v⟩
            end if
        end if
        if msgs_p[re, v] = 2t + 1 then y_p ← v
        end if
    end if
end while
```

korektnosť

- Neexistujú korektné procesy, ktoré by poslali ready správy s rôznymi hodnotami.
- Ak korektný proces rozhodne, všetky korektné procesy rozhodnú rovnakú hodnotu.
- Ak je generál korektný, všetky korektné procesy rozhodnú pre jeho vstup.

Veta 10 *Pre $t < N/3$ je algoritmus BT III asynchrónny t-b-robustný broadcastovací algoritmus.*

Chyby v synchronizovaných systémoch

synchronizovaný systém:

- práca v kolách
- v i-tom kole
  - pošle správy
  - príjme správy z tohto kola
  - zmení stav na základe stavu a prijatých správ
- idealizovaný prípad - garantujeme, že sa správa príjme v tom istom kole
- realistickejšie- ABDN (siete s ohraničeným zdržaním)

Pre synchronizované systémy fatálne chyby nie sú problém - neposlanie informácie spôsobí použitie preddefinovanej hodnoty $\Rightarrow$ byzantské procesy sú "účinné" len ak posielajú nekorektné hodnoty

Broadcast v synchronizovaných DS

- terminovanie, zhoda, závislosť
- **simultánnosť**- všetci rozhodnú v rovnakom takte

Veta 11 ***Neexistuje*** *t-b-robustný broadcastovací protokol* ***pre*** $\mathbf{t \geq N/3}$.

Existuje DETERMINISTICKÝ t-b-robustný broadcastovací protokol pre $t < N/3$

$\Rightarrow$ rekurzívny $broadcast(N, t)$, kde N je počet vrcholov, t počet byzantských chýb.

Broadcast(N,0)

posielanie: generál: send $\langle val, x_g \rangle$
dôstojníci: neposielajú

príjem správy kola 1:

rozhodnutie: generál: x_g
dôstojníci: ak prišla správa od generála, tak rozhodne x_g, inak $undef$

Idea **Broadcast(N,t)**

$t > 0$ - po prijatí správy od "generála" v rámci $broadcast(N, t)$ v ďalšom kole spúšťa $broadcast(N, t-1)$

je prechod $t \rightarrow t-1$ **korektný?**

Algoritmus Broadcast(N,t), $t > 0$

kolo 1

generál: send$\langle val, x_g \rangle$ všetkým; dôstojníci: neposielajú

príjem správy kola 1:

dôstojník p:

- ak prišla správa $\langle val, x \rangle$ od generála v kole 1, tak $x_p := x_g$, inak $x_p := undef$;
- oznámi x_p ostatným dôstojníkom ako keby bol generál v $broadcast_p(N-1, t-1)$ v nasledujúcom kole

kolo t+1

príjem správy kola $t+1$;

generál – rozhodne x_g

dôstojník p:

$broadcast_q(N-1, t-1)$ rozhodol v každom dôstojníkovi q

Nech $W_p[q]$ = rozhodnutie v $broadcast_q(N-1, t-1)$

$y_p \leftarrow major\{W_p\}$

Analýza algoritmu Broadcast(N,t)

terminácia Ak začne $broadcast(N, t)$ v kole 1, každý proces rozhodne v kole $t + 1$

závislosť Ak je generál korektný, f chybných procesov, $N > 2f + t$, tak všetky korektné procesy rozhodnú vstup generála.

zhoda Všetky korektné procesy rozhodnú rovnakú hodnotu.

Veta 12 *Pre $t < N/3$ je algoritmus broadcast(N,t) t-b-robustný broadcastovací protokol.*

- exponenciálna zložitosť správ

Polynomiálny t-b-broadcast (Dolev, Fischer, Fowler)

Na začiatok **predpokladáme** $N = 3t + 1$

L=t+1 - aspoň jeden je korektný **H=2t+1** - aspoň L ich je korektných

správy - $\langle bm, v\rangle$, kde v je alebo hodnota 1 alebo meno procesora
algoritmus je nesymetrický — 0, ak nie je dostatok dôkazov, že generál zakričal 1

pamäť - $R_p[q, v]$ je T práve vtedy keď p dostalo v od procesu q monotónne

aktivity:

podpora p podporuje q v i-tom kole, ak má dosť dôvodov si myslieť, že q poslalo 1

$$\begin{aligned} DS_P &= \{q : R_p[q, 1]\} \\ IS_p &= \{q : \#\{r : R_p[r, q]\} \geq L\} \\ S_p &= DS_p \cup IS_p \end{aligned}$$

potvrdenie ak má dosť správ o q $C_p = \{q : \#\{r : R_p[r, q]\} \geq H\}$

inicializácia ak má dosť dôvodov pre rozhodnutie 1

1. je generál
2. v kole 1 správa $\langle bm, 1\rangle$ od generála
3. v poslednom kole potvrdil dostatočne veľa dôstojníkov

if ini_p **then** shout $\langle bm, 1\rangle$
end if
for all $q \in S_p$ **do** shout $\langle bm, q\rangle$
end for

prijatie všetkých správ kola i;

if $i = 1$ a $R_p[r, 1]$ **then** $ini_p \leftarrow true$
end if
if $\#C_p^L \geq Th(i)$ **then** $ini_p \leftarrow true$ $\triangleright\ Th(i) = L + max(0, \lfloor i/2 \rfloor - 1)$
end if
if $i = 2t + 3$ **then if**$\#C_p \geq H$ **then** $y_p \leftarrow 1$
else $y_p \leftarrow 0$
end if

O algoritme platí:

- splŕňa terminovanie, simultánnosť, závislosť
- Ak L procesov iniciuje na konci i-teho kola, $i < 2t$, potom všetky korektné procesy rozhodnú 1 //generál klame
- Ak aspoň L procesov iniciuje a i je minimálne číslo kola, na konci ktorého aspoň L korektných procesov iniciuje, tak $i < 2t$. //generál klame

Veta 13 *Algoritmus DFF je deterministický t-b-robustný broadcastovací protokol.*

Autentifikácia — zvyšuje t na $N-1$

$S_p(M)$ - elektronický podpis procesu p pre správu M

1. ak je p korektný, len p môže *efektívne* vypočítať $S_p(M)$ - podpis správy M
2. každý proces môže rozhodnúť, či $S_p(M) = S$

správa M podpísaná procesom p - $\langle M \rangle : p$

Algoritmus Lamport, Shostak, Pease [LSP]

kolo 1 generál zakričí podpísanú správu $\langle val, x_g \rangle : g$
kolo 2,..,t+1 pridanie podpisu pri preposielaní správy (tým, ktorý v tom zozname nie sú);
správa $\langle val, x_g \rangle : g : p_2 : \ldots : p_i$ je platná pre (prijímajúci) proces p, ak

1. všetky procesy sú korektné
2. všetky podpisy sú od rôznych procesov
3. p sa v zozname podpisov nevyskytuje

proces si udržiava množinu W_p hodnôt v platných správach
kolo t+1 proces rozhoduje podľa W_p
ak $W_p = \{v\}$, rozhodne v, inak 0

Veta 14 *Pre $t < N$ je algoritmus LSP korektný $t-b-$robustný broadcastovací protokol, ktorý používa $t+1$ kôl.*

Algoritmus Dolev, Strong — všetky W_p jednoprvkové alebo sú všetky viacprvkové
Zakomponovanie do LSP - preposlanie iba dvoch rôznych akceptovaných správ

Veta 15 *Algoritmus DS je $t-b-$robustný broadcastovací protokol, ktorý používa $t+1$ kôl a posiela $2N^2$ správ.*

Implementácie digitálnych podpisov — verejný, osobný kľúč

ElGamal diskrétny algoritmus: $g^0 = 1, g^1, g^2, \ldots, g^{P-2}$- $P-1$ rôznych prvkov

vie sa P, g $\Rightarrow$ <u>súkromný kľúč</u> -d <u>verejný kľúč</u> - $e = g^d$

<u>platný podpis</u> pre správu M - dvojica (r, s), pre kt. platí $g^M = e^r r^s$

Ak poznáme d: a nesúdeliteľné s $P-1$;
$r = g^a (mod\ P)$; $s = (M - dr)a^{-1}$ (mod (P-1))
$e^r \cdot r^s = e^r (g^a)^{(M-dr)a^{-1}} = g^{dr} g^{M-dr} = g^M$

RivestShamirAdleman verejný kľúč - veľké čísko n také, že p pozná jeho faktorizáciu
a exponent e taký, že $de = 1$
podpis procesu p pre správu M je $M^e \mod n$
súkromný kľúč - d; $(M^d)^e = M$

...

Synchronizácia hodín

ABD (asynchronous bounded delay) - máme lokálne hodiny a horný odhad na doručenie správy

$\mathbf{C_p(t) = T}$ - hodiny procesu p majú v reálnom čase t hodnotu T
$\mathbf{c_p}$ označuje C_p^{-1}; $t = c_p(T) = C_p^{-1}(T)$
sklz je ρ-ohraničený, ak pre t_1, t_2 také, že neexistuje hodnota C medzi t_1, t_2 platí

$$(t_2 - t_1)(1+\rho)^{-1} \leq C(t_2) - C(t_1) \leq (t_2 - t_1)(1+\rho)$$

Hodiny sú δ-synchronizované v t, ak $|C_p(t) - C_q(t)| \leq \delta$. Chceme **globálnu synchronizáciu**.

Zdržanie správy je medzi $\sigma_{min}, \sigma_{max}$: $\sigma_{min} \leq \tau - \sigma \leq \sigma_{max}$

Veta 16 *Existuje deterministický protokol synchronizácie hodín C_p, C_s s presnosťou $\frac{1}{2}(\delta_{max} - \delta_{min})$, ktorý posiela 1 správu. Neexistuje deterministický algoritmus, ktorý dosahuje väčšiu presnosť.*

S pošle $\langle time, C_S \rangle$

p príjme $\langle time, T \rangle$ a nastaví si čas na $T + \frac{1}{2}(\delta_{max} - \delta_{min})$

t-b-distribuovaná synchronizácia hodín

Procesy sa nepresne dohodnú na *priemernom* čase. Predpokladajú sa ideálne hodiny (bez sklzu)

1. **fáza** p si vypýta info správou $\langle ask \rangle$ a počká $2\delta_{max}$ na odpoveď

2. **fáza** p prefiltruje prijaté správy (korektné hodnoty sa líšia nanajvýš o δ)
   A_p je množina tých časov, ktoré to spĺňajú

výstup priemer prefiltrovaných hodnôt; zamietnuté si vygenerujú hodnotu, kt. prežije
$intvl(A) = [min(A), max(A)]; width(A) = max(A) - min(A);$
$estimator(A) \in intvl(A)$, resp. $min(A), max(A), (max(A) + min(A))/2, \ldots$

var: $x_p, y_p, esti_p$:real
V_p, A_p :multimnožina reálnych čísel

$V_p \leftarrow \emptyset$
for all $q \in \mathbb{P}$ **do** send $\langle ask \rangle$ do q
end for
čakaj $2\delta_{max}$ $\triangleright$ spracovanie $\langle ask \rangle$ a $\langle val, x \rangle$
while $\sharp V_p < N$ **do** insert(V_p, ∞)
end while
$A_p \leftarrow \{x \in V_p : \ \sharp\{y \in V_p : \ |y - x| \leq \delta\} \geq N - t\}$
$esti_p \leftarrow estimator(A_p)$
while $\sharp A_p < N$ **do** insert$(A_p, esti_p)$
end while
$y_p \leftarrow \left(\sum A_p\right)/N$

po prijatí $\langle ask \rangle$ od q – send $\langle val, x_p \rangle$

po prijatí $\langle val, x \rangle$ od q
if ešte taká správa od q neprišla **then** insert(V_p, x)
end if

Presnosť tohto algoritmu je $2t\delta/N$

Synchronizácia hodín-nutná adaptácia

- nie je známe zdržanie správ - pripočítame $(\delta_{max} + \delta_{min})/2$
- čas v priebehu výpočtu postupuje - $C_p = C_p + \Delta_p$

var: $C_p, \Delta_p, esti_p$:real
D_p, A_p :multimnožina reálnych čísel

$D_p \leftarrow \emptyset$
for all $q \in \mathbb{P}$ **do** send $\langle ask \rangle$ do q
end for
čakaj $2\delta_{max}$ ▷ spracovanie $\langle ask \rangle$ a $\langle val, x \rangle$
while $\sharp D_p < N$ **do** insert(D_p, ∞)
end while
$A_p \leftarrow \{x \in D_p : \ \sharp\{y \in D_p : \ |y - x| \leq \delta + (\delta_{max} - \delta_{min})\} \geq N - t\}$
$esti_p \leftarrow estimator(A_p)$
while $\sharp A_p < N$ **do** insert$(A_p, esti_p)$
end while
$\Delta_p \leftarrow \left(\sum A_p\right)/N; \quad C_p \leftarrow C_p + \Delta_p$

po prijatí $\langle ask \rangle$ od q – send $\langle val, C_p \rangle$

po prijatí $\langle val, C \rangle$ od q – ak prvá taká, insert $(D_p, (C + \frac{1}{2}(\delta_{max} + \delta_{min})) - C_p)$

Presnosť tohto algoritmu je $(\delta_{max} - \delta_{min}) + \frac{2t}{N}[\delta + (\delta_{max} - \delta_{min})]$