

PDV 13 2020/2021

Volba lídra (koordinátora)

Michal Jakob

michal.jakob@fel.cvut.cz

Centrum umělé inteligence, katedra počítačů, FEL ČVUT



Motivace

Design distribuovaných algoritmů pro řadu problémů distribuovaných výpočtů se **zjednoduší**, když jsou **asymetrické** a předpokládají, že jeden z procesů má roli **lídra** (a s ní spojenou logiku).

- konsensus, replikace, vyloučení procesů, ...

V situacích, kdy uvažujeme **selhání procesů**, musím být schopni lídra dynamicky nahradit.

Problém volby lídra

Ze skupiny procesů **vybrat lídra** (který bude řešit specifické úkoly) a **dát vědět všem procesům** ve skupině, kdo je lídrem.

Co se stane, když lídr selže?

- nějaký proces detekuje pomocí detektoru selhání a spustí nové volby

Algoritmus pro volbu lídra musí zajistit:

1. zvolí právě jednoho lídra z bezvadných procesů
2. všechny bezvadné procesy ve skupině se shodnou na tom, kdo je lídr

Systemový model

Skupina N procesů s unikátními identifikátory.

- známe všechny procesy, ale nevíme, které jsou aktivní (bezzvadné)

Procesy **mohou havarovat**.

FIFO perfektní komunikační kanál mezi každým párem procesů, tj. zprávy se neduplikují, nevznikají, neztrácejí a jsou doručovány v pořadí odeslání.

Asynchronní systém: neznáma, ale **konečná latence**.

Další požadavky

Každý proces může **vyvolat** volby.

Jeden proces může vyvolat v jeden okamžik pouze **jedny volby**.

Více procesů může vyvolat volby **současně** - pak požadujeme, aby se nakonec shodly na jednom lídrovi.

Výsledek volby lídra by **neměl záviset** na tom, který proces volby **vyvolal**.

Po skončení běhu algoritmu volby lídra má každý proces ve své proměnné *ELECTED* identifikátor lídra s nejvyšší hodnotou **volebního kritéria** (a nebo nikdo, tj. volby skončily neúspěšně).

Volební kritérium:

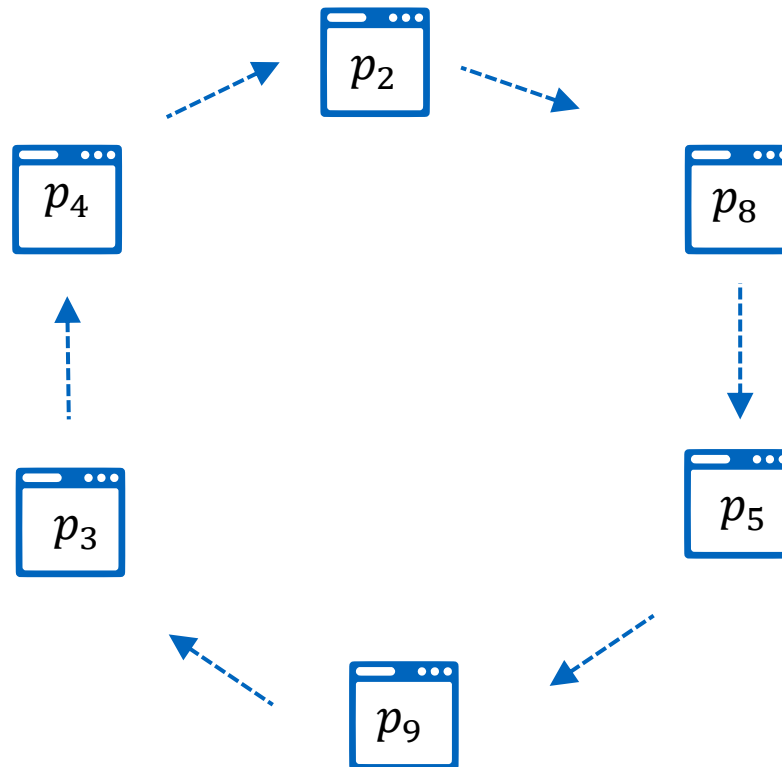
- typicky nejvyšší identifikátor, tj. IP adresa
- ale taky např: nejvíce RAM, diskového prostoru, nejvíce souborů v případě P2P sítí nebo nejúplnější log v případě RAFTu
- musí být fixní a známe všem procesům při zahájení volby

Kruhový algoritmus

Procesy jsou uspořádané do logického **kruhu**.

Zprávy jsou posílány dokola kruhu **v jednom směru**.

Procesy jsou schopné **detekovat selhání** ostatních procesů



Kruhový algoritmus

P_i : zahájení voleb (po detekci selhání)

P_i pošle po kruhu zprávu ELECTION(i) obsahující jeho identifikátor

P_i : zpracování zprávy ELECTION(j)

if $i < j$ **then** přepoše zprávu ELECTION(j)

if $i > j$ **then** nahradí zprávu za ELECTION(i) a pošle dál

if $i = j$ **then** P_i je nový koordinátor; odešle zprávu ELECTED(i)

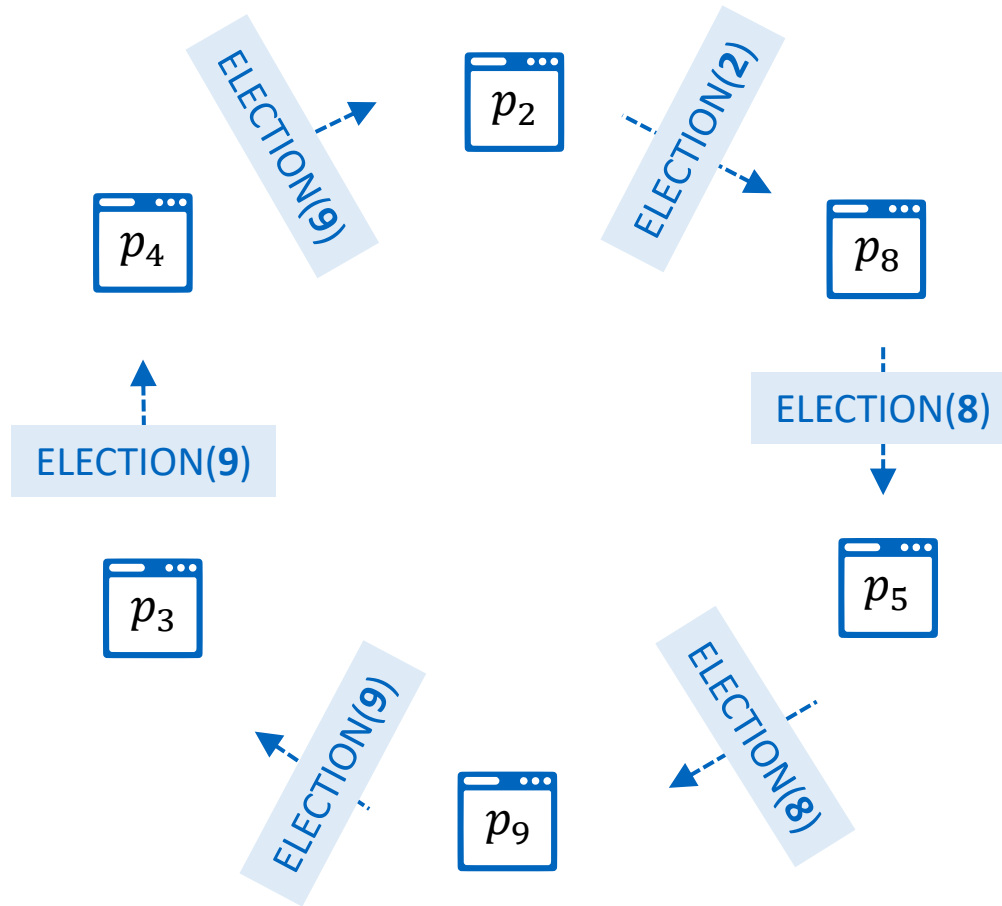
P_i : přijetí zprávy ELECTED(j)

Nastaví svou proměnou **elect** $_i := j$

if $i \neq j$ **then** přepoše zprávu ELECTED(j)

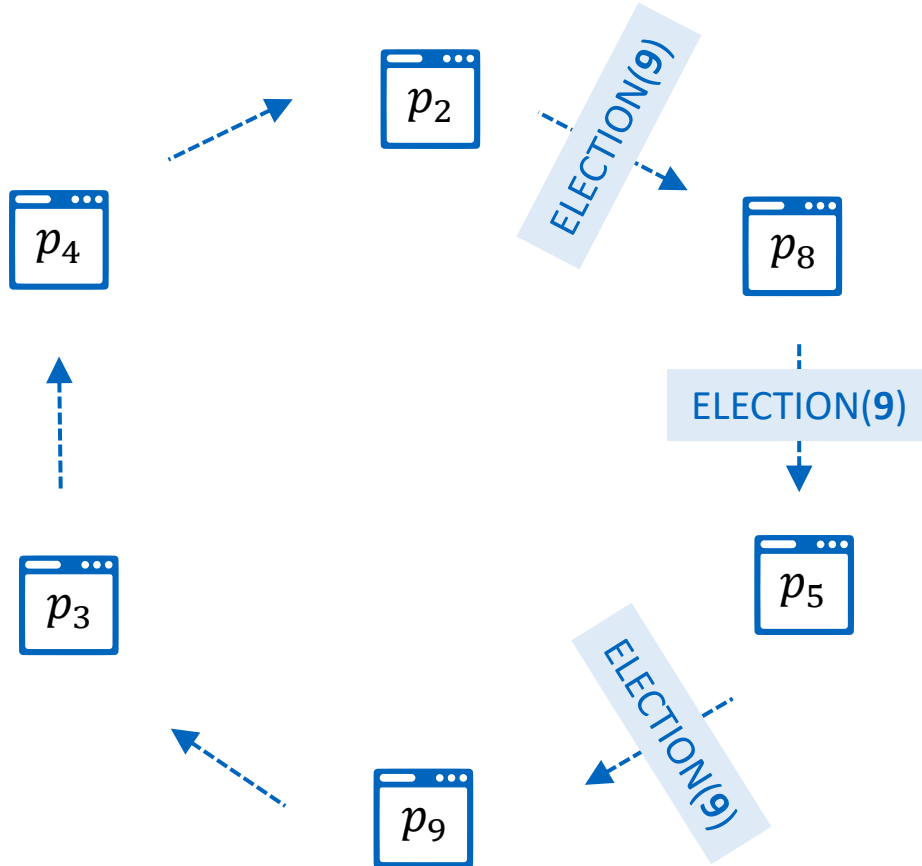
Příklad (volební kritérium = ID procesu)

Spustí volby

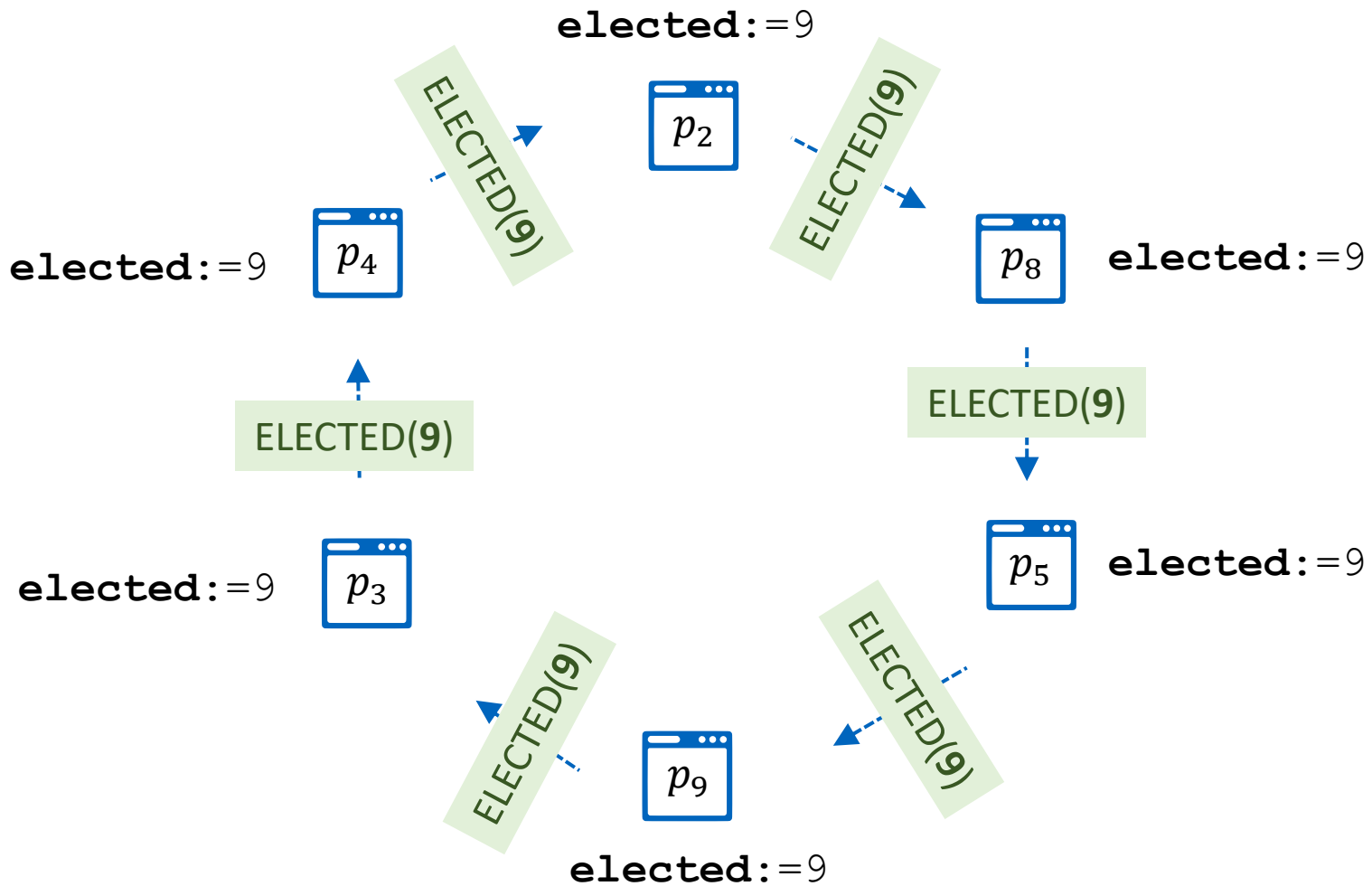


Příklad (volební kritérium = ID procesu)

Spustí volby



Příklad (volební kritérium = ID procesu)



Analýza

N procesů; předpokládáme, že v průběhu volby nedochází k selháním

Nejhorší případ

- iniciátor je následník budoucího lídra
- $N - 1$ zpráv pro zprávu ELECTION než se dostane od iniciátora P_i k budoucímu lídrovi P_j
- N přeposlání nezměněné zprávy ELECTION(j) okolo kruhu
- N přeposlání zprávy ELECTED(j) okolo kruhu
- Celkem $3N - 1$ zpráv
- Čas běhu: $3N - 1$ komunikačních latencí

Nejlepší případ

- iniciátor je budoucí lídr
- Celkem $2N$ zpráv
- Čas běhu: $2N$ komunikačních latencí

Pokud nejsou selhání, tak volba skončí **v konečném čase** (živost).
Pokud volby skončí úspěšně, tak všechny procesy znají **stejného lídra** (bezpečnost).

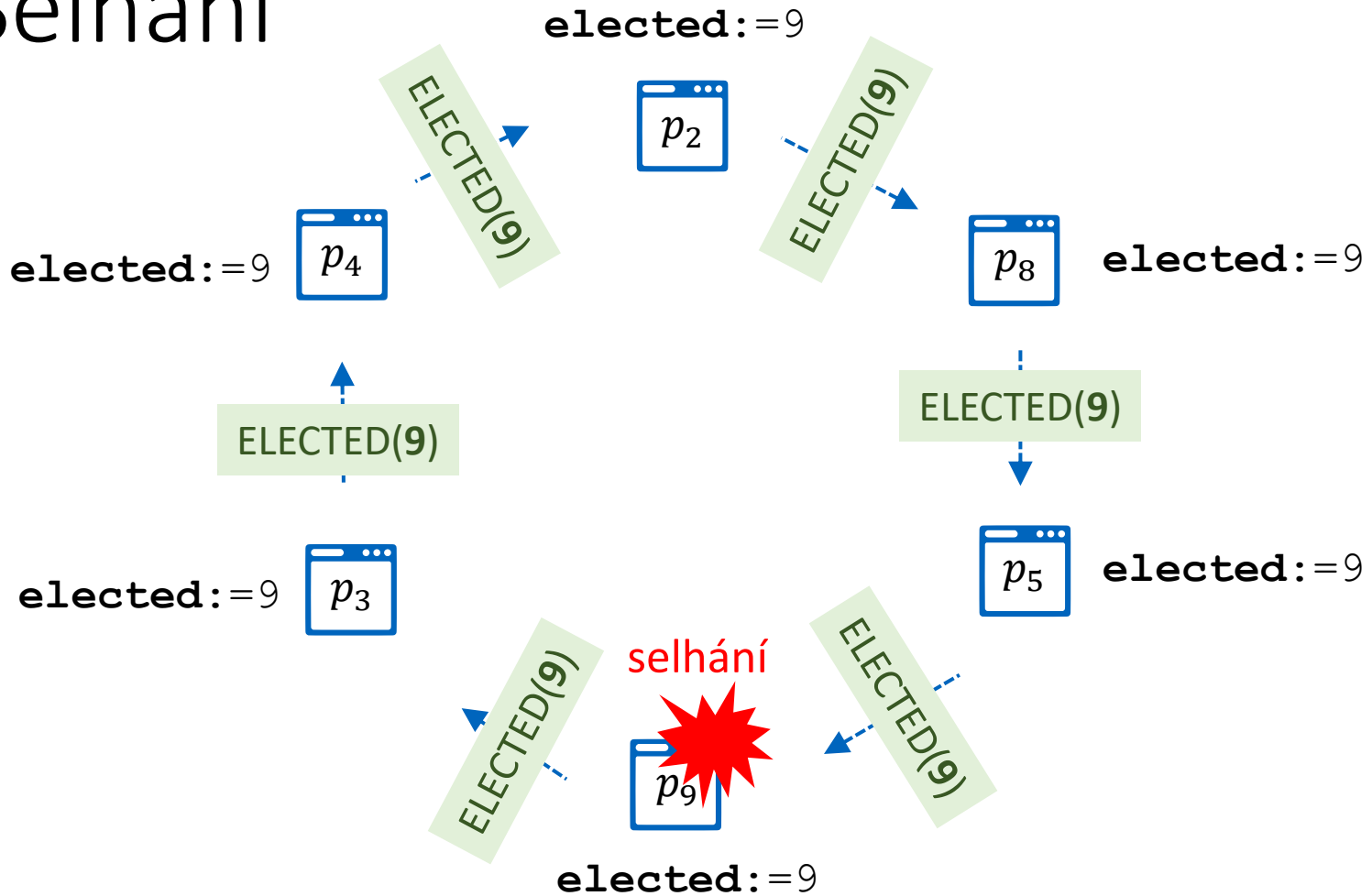
Více iniciátorů

Proces si pamatují nejvyšší maxID ze všech dosud obdržených ELECTION/ELECTED zpráv (od poslední úspěšné volby lídra).

Procesy ignorují ELECTION/ELECTED zpráv s nižším ID než maxID .

Dojde k rychlejší eliminaci zpráv s ELECTION zpráv s ID procesů, které nemohou být zvoleny.

Selhání



Selže-li P_9 po odeslání zprávy, tak `ELECTION(9)` bude cirkulovat do nekonečna i pokud dojde k obnovení topologie kruhu → **porušení živosti.**

Jak řešit selhání?

Možnost 1: Předchůdce nebo následovník rádoby lídra P_k detekuje selhání a spustí nové volby

- Pokud obdrží zprávu ELECTION(k), ale neobdrží odpovídající zprávu ELECTED(k)
- Pokud obdrží podruhé zprávu ELECTION(k) nebo ELECTED(k)

Ale co když selže předchůdce?
A předchůdce předchůdce? ...

Možnost 2: Využití detektoru selhání.

- Kterýkoliv proces po přijetí zprávy ELECTION(k) může detekovat selhání P_k pomocí svého lokálního detektoru selhání
- Jakmile detekuje selhání, iniciuje nové volby

Ale: Detektory selhání nemohou být zároveň úplné a přesné:

- neúplný detektor \Rightarrow selhání nemusí být detekované \Rightarrow systém zůstane bez lídra
- nepřesný detektor \Rightarrow lídr může být mylně detekován jako havarovaný \Rightarrow zbytečně opakované volby, potenciálně neomezený počet krát \Rightarrow narušení **živosti**



Algorithmus Bully

Algoritmus Bully („tyran“)

Klasický algoritmus pro volbu lídra.

Základní princip: proces, který **detekoval selhání** dosavadního lídra, **vyzve** procesy s **vyšším ID** ve volbách.

Bully algoritmus

P_i : zahájení voleb (po detekci selhání nebo jako reakce na volby)

```
if  $P_i$  má nejvyšší ID
    Pošli COORDINATOR zprávu
    všem procesům s nižšími
    identifikátory (volby končí).
else // zahájí volby
    Pošli ELECTION zprávu všem
    procesům s vyšším ID
    // následně čekání na odpověď
```

P_i : reakce na ELECTION

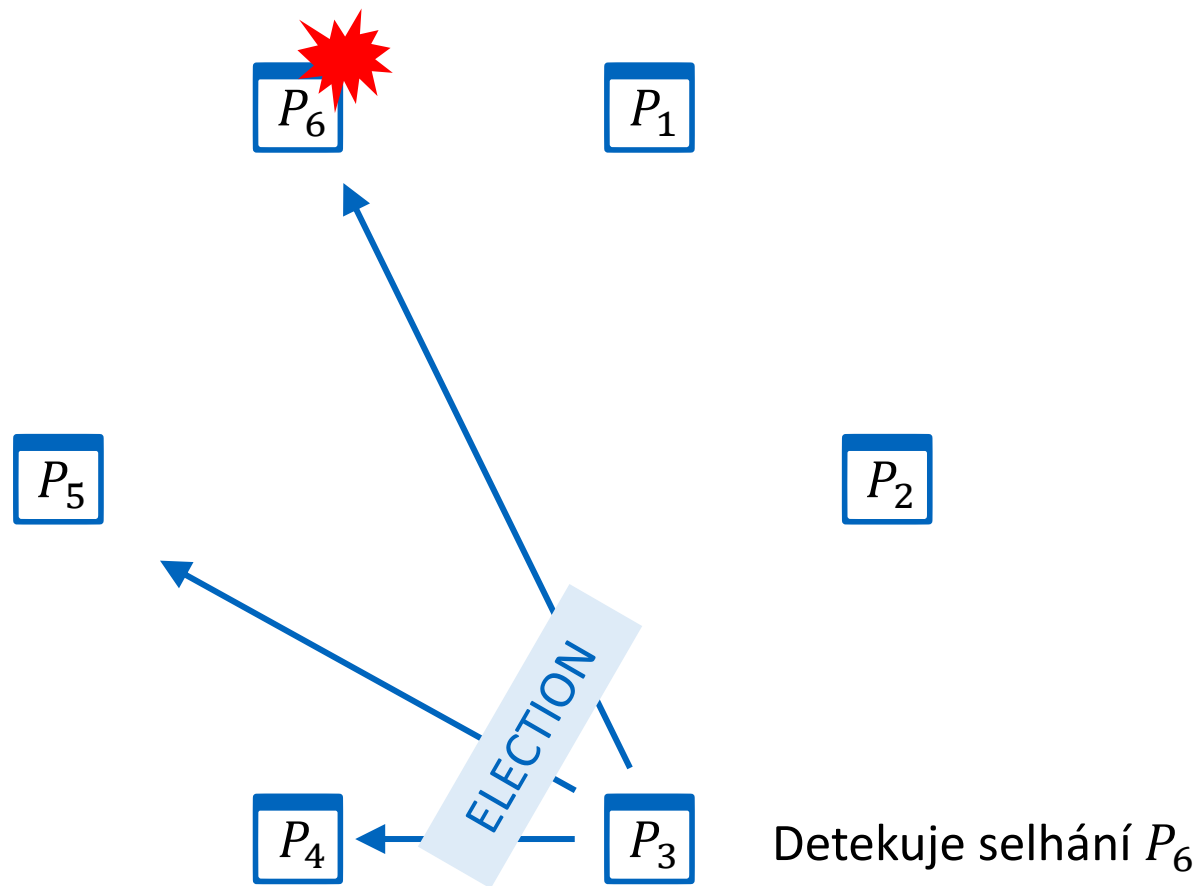
```
Odpověz OK
If pokud  $P_i$  dosud nezačal volby
    zahaj volby
```

P_i : čekání na odpovědi
(po vyvolání voleb)

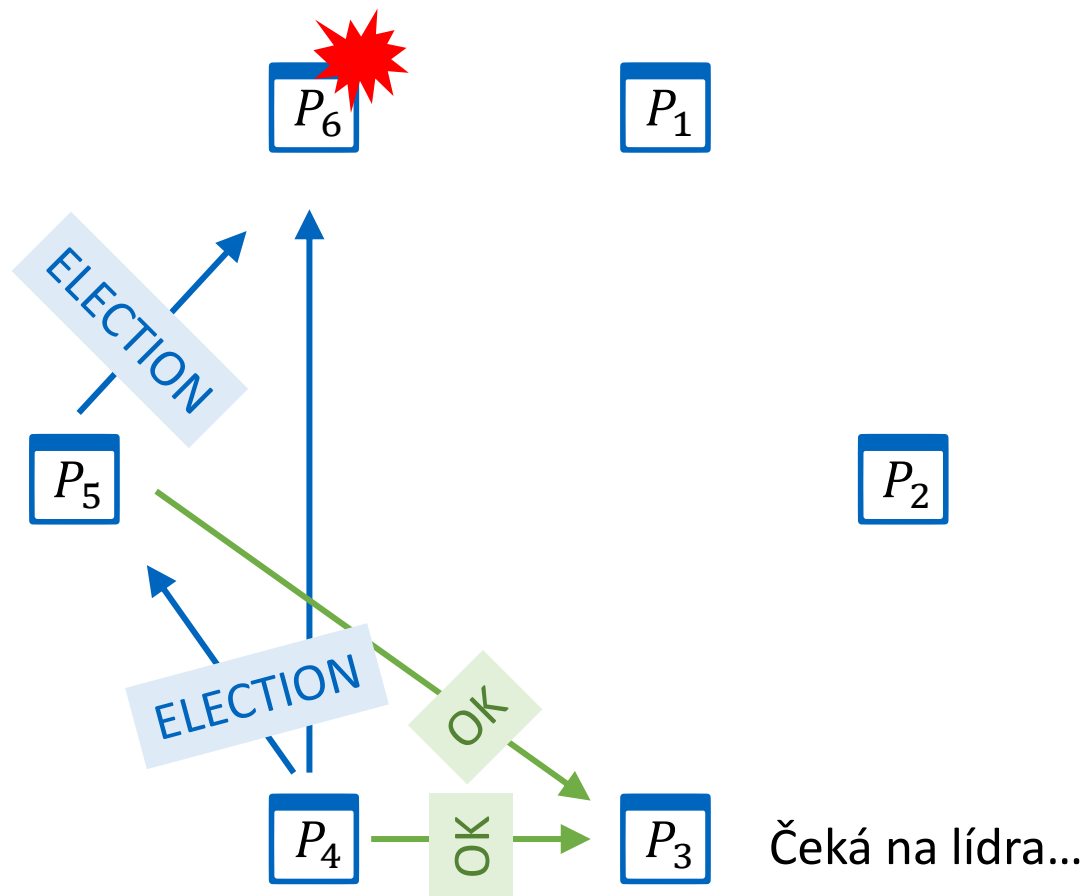
```
if nedorazí žádná odpověď v
časovém limitu  $T$ 
    prohleš se jako  $P_i$  za lídra;
    pošli COORDINATOR zprávu
    všem procesům s nižším ID;
    // volby skončily

else // existuje aktivní proces s
vyšším ID než  $P_i$ 
    čekej na zprávu
    COORDINATOR;
    pokud nedorazí v časovém
    intervalu  $T'$ , iniciuj nové volby
```

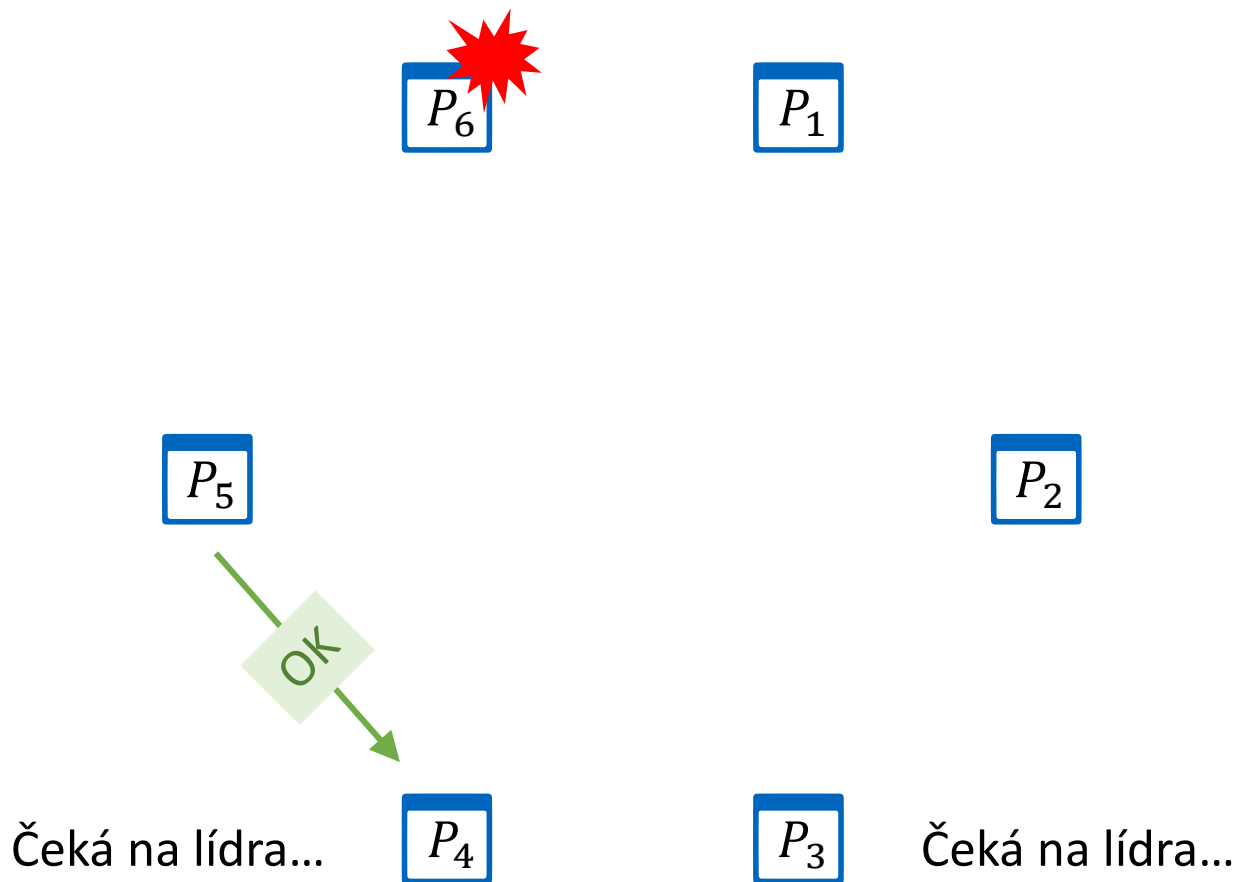
Bully algoritmus



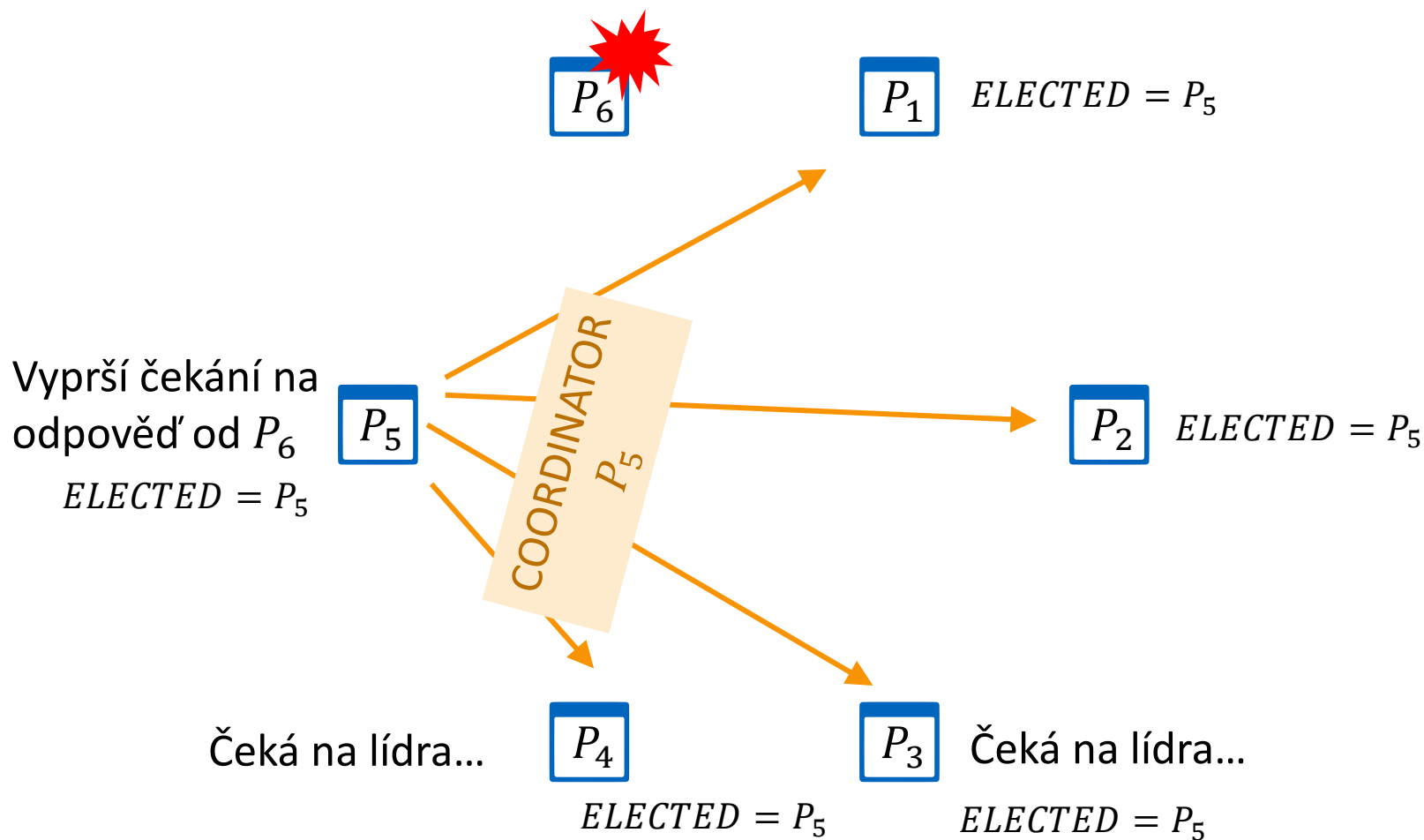
Bully algoritmus



Bully algoritmus

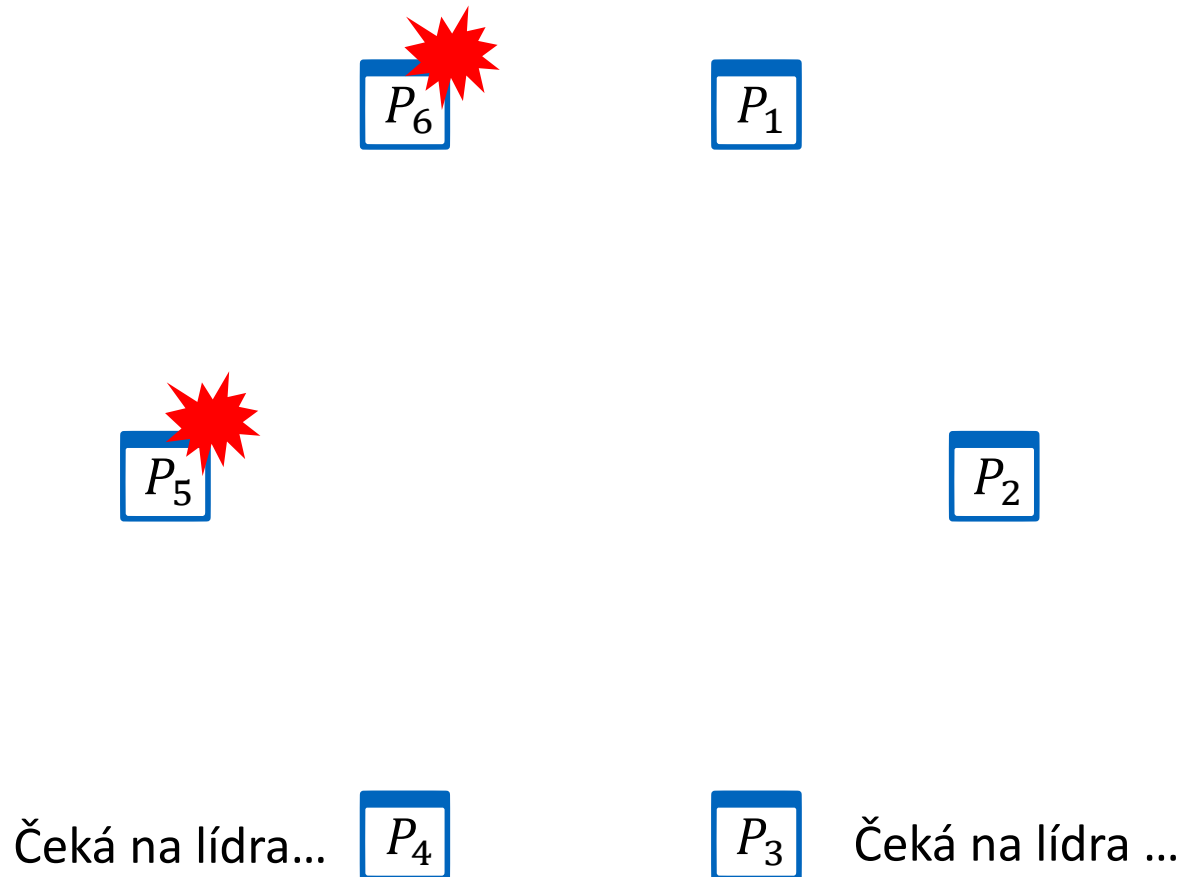


Bully algoritmus

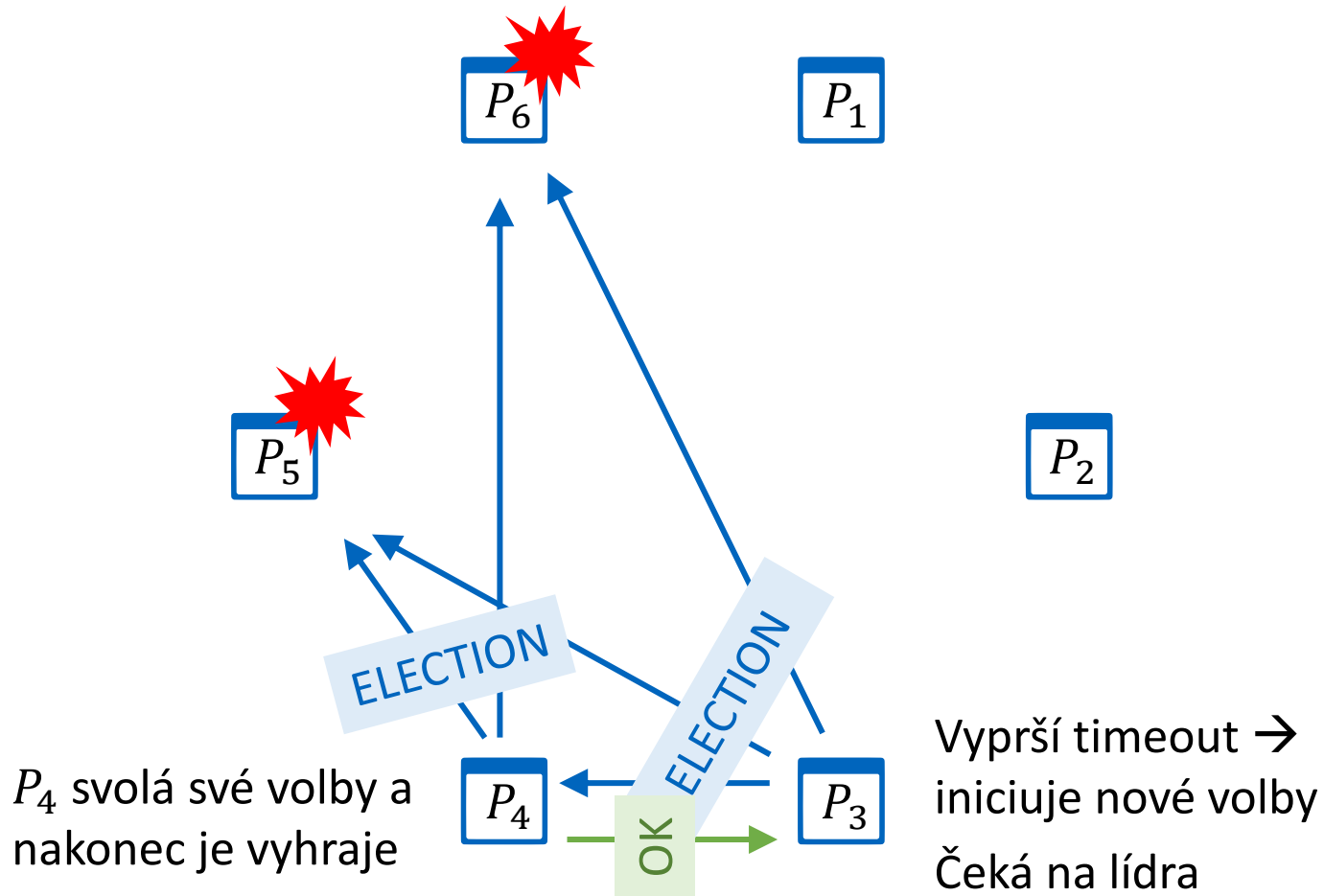


Volby jsou skončeny

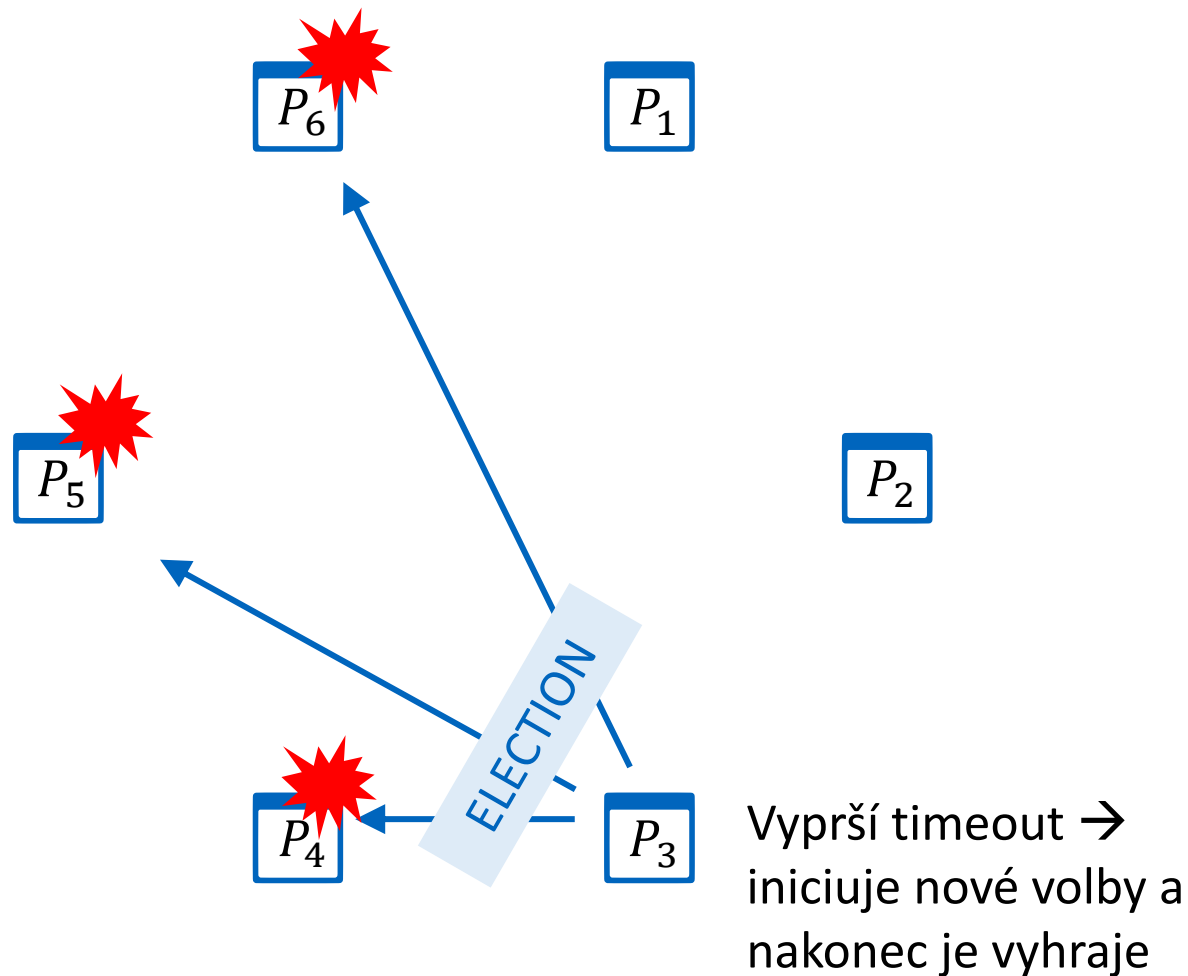
Selhání během volby



Selhání během volby



Selhání během volby



Analýza

Nejhorší případ – selhání lídra detekováno procesem s nejnižším ID.

Pro dokončení volby lídra je v nejhorším případě potřeba **čtyři komunikační latence**:

1. Proces s nejnižším ID pošle ELECTION zprávu ostatním procesům
2. Proces s druhým nejvyšším ID:
pošle OK procesům s nižšími ID
pošle ELECTION procesu s nejvyšším ID
3. Procesu s druhým nejvyšším ID vyprší timeout při čekání na odpověď procesu s nejvyšším ID
4. Proces s druhým nejvyšším ID rozešle zprávu COORDINATOR

Nejhorší případ – komunikační zátěž

- celkem $N - 1$ procesů pošle zprávu ELECTION procesům s vyšším ID
- tj. celkem: $N - 1 + N - 2 + \dots + 1 = \frac{N(N-1)}{2} = \mathbf{O(N^2)}$ zpráv

Analýza

Nejlepší případ: proces s druhým nejvyšším ID detekuje selhání lídra

Nejlepší případ - komunikační zátěž:

- $(N - 2)$ zpráv COORDINATOR
- čas do ukončení: **1 komunikační latence**

Bezpečnost

Bezpečnost volby lídra: Každý proces buď zvolí stejného lídra L nebo nezvolí žádného lídra.

Bully algoritmus bezpečnost **garantuje** pokud havarované procesy nejsou nahrazovány procesy se stejným ID (např. restartem).

Živost

Bully algoritmus pracuje s timeouts → **v asynchronním systému** jeho běh nemusí nikdy skončit → **živost není garantována.**

V synchronním systému:

- lze spočítat nejhorší jednosměrnou latence = doba přenosu zprávy + doba reakce na zprávu.
- pokud timeout T a T' nastavíme na násobek nejhorší jednosměrné latence, je **živost garantována**

V částečně synchronním systému lze garantovat **konečnou živost** (eventual liveness)

- pokud se latence dostane pod potřebnou mez na dostatečně dlouhou dobu, volby úspěšně skončí (pokud nebudou selhání)

Proč je volba lídra tak těžké?

Protože souvisí s problémem **konsensu!**

Pokud bychom byli schopni vyřešit volbu lídra, tak umíme řešit konsensus.

Stačí zvolit lídra a poslední bit jeho ID interpretovat jako výsledek konsensu.

Ale protože konsensus není řešitelný v asynchronních DS se selháními, není řešitelná ani volba lídra.

(řešitelnost = existence algoritmu garantujícího bezpečnost i živost současně)

Souhrn

Volba lídra důležitý problém v DS.

Bully algoritmus klasický algoritmus předpokládající selhání procesů, ale perfektní FIFO kanál.

Bully algoritmus garantuje **bezpečnost** (za předpokladu, že se havarované procesy nerestartují se stejným ID).

V asynchronním systému **negarantuje živost**; v synchronním nebo částečně synchronním systému lze (konečnou) živost (tzv. eventual liveness) dosáhnout vhodným **nastavením timeout.**