Volba vůdce, Leader Election

PA 150 ◇ Principy operačních systémů

Jan Staudek
http://www.fi.muni.cz/usr/staudek/vyuka/

Verze : podzim 2020

Volební problém – Kdy a proč se volí vedoucí uzel ?

- Existuje potřeba spustit nový centrální proces např.
  - ✓ koordinátora vzájemného vyloučení či transakcí, resp.
  - ✓ nový centrální časový server, resp.
  - ✓ ztratil se token a jediný uzel musí vygenerovat nový token, resp.
  - ✓ hledá se kostra grafu, ve které má být vůdce kořen (stromu)
- Programování distribuovaných aplikací je typicky jednodušší v prostředí typu *master*/*slave*, (tj. 1/n procesů), např.
  - rozesílání zpráv ke všem uzlům po kostře grafu organizované z kořene
  - koordinace distribuovaných transakcí
- Roli vedoucího objektu vyžaduje řada správ výjimek, např.
  - odstranění uváznutí, generování tokenu (peška)
- Dynamické určování vedoucího uzlu má řadu předností, např.
  - nevadí výpadek centrálního uzlu – koordinátora
  - množina činných procesů, ze kterých se volí, nemusí být statická

Volební problém

- Definice problému
  - ✓ každý výpočet podle volebního algoritmu ve skupině procesů v DS musí v konečném čase končit konfigurací, ve které je jeden jediný uzel v roli vůdce (*leader*, *master*, ...)

- Každý proces ve skupině procesů se době své existence nachází v jednom ze tří stavů:
  nerozhodnutý, ve skupině procesů běží volba
  vůdce, volba ve skupině procesů skončila
  není vůdce, volba ve skupině procesů skončila

Volební problém

- Skupina procesů v DS je skupinou anonymních procesů
  pokud procesy ve skupině nemají jedinečné id
- Skupina procesů v DS je skupinou neanonymních procesů
  pokud procesy ve skupině mají jedinečné id

Volební problém

- Algoritmus volby je
  - uniformní pokud procesy neznají počet procesů ve skupině procesů
  - neuniformní pokud procesy znají počet procesů ve skupině procesů
- Uniformní algoritmus
  - ✓ stejný algoritmus pro jakýkoliv rozměr skupiny
  - ✓ každý proces v každém rozměru skupiny je modelovaný stejným stavovým strojem
- Neuniformní algoritmus
  - ✓ algoritmy pro různé rozměry skupiny se liší
  - ✓ pro každé n každý proces ve skupině rozměru n je modelovaný stejným stavovým strojem A_n

Volební problém

- Řešení volby vůdce ve skupině anonymních procesů neexistuje
  - ✓ nelze splnit podmínku bezpečnosti
  - ✓ v každém kroku volby všichni dostávají shodné zprávy, každý se může prohlásit za vůdce
- Specifikace algoritmu pro skupinu neanonymních procesů
  - ✓ procesy mají jedinečný id z totálně uspořádané množiny
  - ✓ všechny uzly řeší stejný lokální algoritmus
  - ✓ algoritmus volby vůdce je decentralizovaný algoritmus, iniciátorem může být kterýkoliv proces DS, algoritmus může iniciovat více iniciátorů souběžně
  - ✓ algoritmus v každém provedení dosáhne terminální konfiguraci, ve které je ve skupině procesů DS jediný proces (uzel, komponenta) ve stavu vůdce

Volba, volební běh, účastník volby

- Mějme skupinu neanonymních procesů, ve které je jeden proces vůdce, ostatní procesy nejsou vůdci
- Roli vůdce přebral proces, který byl vítěz provedení volby
- Proces, který spouští jeden konkrétní běh volebního algoritmu, vyvolává volební běh, je *iniciátorem*
- Každý proces může najednou vyvolat pouze jeden volební běh
  $\Rightarrow$ v prostředí N procesů se může současně odehrávat v rámci jedné volby až N volebních běhů
- Každý proces má právo i povinnost volit
- Volba končí ukončením všech volebních běhů aktivovaných v aktivované volbě

Volba, volební běh, účastník volby

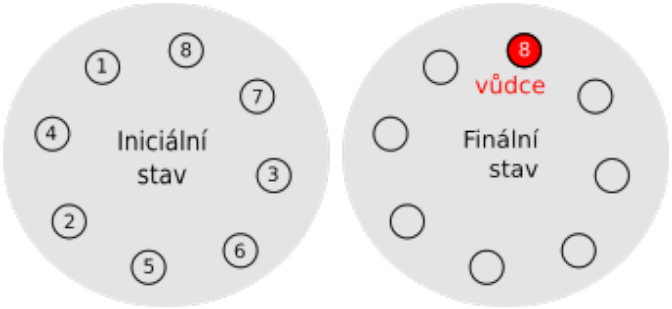
- Každý proces může být v době řešení algoritmu
  - ✓ účastník volby (*participant*),
    pak participuje alespoň v jednom volebním běhu,
    do ukončení volby je ve stavu nerozhodnutý
  - ✓ ,,neúčastník'' volby (*non-participant*),
    pak neparticipuje v žádném volebním běhu
    jeho stav není definovaný
- Po ukončení volby se každý uzel nachází v jednom ze 2 stavů
  - ✓ vůdce, vedoucí uzel, vítěz volby (*master, elected, winner*) nebo
  - ✓ není vůdce, je podřízený uzel (*not-elected, slave, loser*)
- roli vůdce (*master*) získá po ukončení volebního běhu jediný uzel

Volební algoritmus, vlastnosti

- výběr zvoleného procesu (master) musí být jednoznačný, i když se realizuje více volebních běhů souběžně
  - ✓ Bez omezení obecnosti lze zavést podmínku vítězné volby – volí se proces s nejvyšší prioritou, s největším id, ...
  - ✓ priorita může být i strukturovaná –

    např. chceme-li zvolit pro roli master proces běžící na procesoru s nejmenší průměrnou výpočetní zátěží, jako id procesoru zavedeme hodnotu $\{1/load, i\}$, kde

    $load>0$ je % doby procesoru věnované aplikačním procesům a i je jedinečný index procesoru pro odlišení procesorů se stejnou zátěží



Volební algoritmus, vlastnosti

- Každý proces P_i si udržuje proměnnou $elected_i$
  - ✓ iniciálně obsahuje prázdnou hodnotu ($\perp$)
  - ✓ po ukončení volebního algoritmu obsahuje id zvoleného procesu
- Všechny procesy mají definovanou prioritu
  - ✓ procesy znají škálu hodnot priorit
  - ✓ každý proces zná svoji prioritu
- Ilustrace akcentů spuštění volebního běhu
  - ✓ proces P_i pošle koordinátorovi zprávu a nedostane do doby T odpověď (např. nedostane časové razítko od časového serveru)
  - ✓ proces P_i nedostane do doby T token, ...

Volební algoritmus, podmínky řešení

- bezpečnostní podmínka řešení volebního algoritmu
  - ✓ rozhodnutí uzlu být vedoucím (zvoleným) procesem je zvoleným procesem nezměnitelné
  - ✓ vedoucím procesem je nejvýše jeden proces, je jím běžící proces s nejvyšší prioritou, P_{max} (s největším id, ...)
  - ✓ každý účastník volby má $elected_i = \perp$ nebo $elected_i = id\ s\ P_{max}$
- podmínka živosti řešení volebního algoritmu
  - ✓ na volbě participují všechny uzly a nakonec všechny uzly nastaví $elected_i \neq \perp$
  - ✓ jeden z uzlů se nakonec stane vedoucí uzel
- dokud se proces nestane účastníkem volby, může mít $elected_i$ nastaveno na id procesu zvoleného v předešlé volbě

Studované volební algoritmy

- Ring algorithm – volba směrovaná po komunikačním kruhu (LeLann, 1977)
- Ring algorithm – volba směrovaná po komunikačním kruhu (Chang, Roberts, 1979)
- Ring algorithm – volba směrovaná po komunikačním kruhu (Hirschberg-Sinclair, 1980)
- Bully algorithm – volba v obecné topologii (Garcia-Molina, 1982)

Volební algoritmus, Ring algorithm – LeLann, 1977

- Iniciátor posílá svůj ID po kruhu, spouští volební běh
- Kruh je jednosměrný, každý proces má jednoho souseda, komunikace je ve FIFO režimu
- Pokud volební běh dosáhne uzel dosud neparticipující na volbě, spouští volební běh tohoto uzlu
- Z přicházejících zpráv si každý uzel postupně zapamatovává ID ostatních procesů a tyto zprávy přeposílá dál po kruhu.
- Jakmile získá zprávu se svým ID, zná ID všech procesů na kruhu a ze seznamu ID zjišťuje kdo bude vůdce (nejmenší/největší ID, . . .)

Volební algoritmus, Ring algorithm – LeLann, 1977

- Každý proces si v průběhu algoritmu volby vypracovává seznam aktivních procesů s jejich prioritami
- Po ukončení volby každý proces v seznamu rozpozná šéfa
- Nechť proces P_i detekuje výpadek šéfa, pak jeho další kroky inicializují volební běh:
  - ✓ P_i si vytvoří nový, počátečně prázdný, seznam aktivních procesů,
  - ✓ P_i posílá svému (pravému) sousedovi volící zprávu elect(i) a do svého seznamu aktivních procesů zapisuje i

. . .

Volební algoritmus, Ring algorithm – LeLann, 1977

- Nechť proces P_i dostane zprávu elect(j) od levého souseda P_j, pak reaguje jedním z následující postupů:

  ✓ Zpráva elect(j) je první ze zpráv elect, které P_i dostal:

  Proces P_i se stává účastníkem volby a

  i) vytvoří si nový seznamu aktivních procesů, počátečně prázdný, a zapíše do něj i a j

  ii) a pošle zprávy elect(j) a elect(i) svému pravému sousedovi

  ✓ Zpráva elect(j) není první ze zpráv elect, a přitom platí $i \neq j$:

  Proces P_i je již účastníkem volby a

  i) do svého seznamu aktivních procesů přidá j a

  ii) přepošle elect(j) svému pravému sousedovi

Volební algoritmus, Ring algorithm – LeLann, 1977

- ✓ Zpráva elect(j) není první ze zpráv elect,
  a přitom platí $i = j$:

  proces P_i drží seznam aktivních procesů, který obsahuje priority všech procesů v systému a P_i si může zjistit kdo bude v dalším období koordinátor a žádnou zprávu dál neposílá

  Nastaví si svoji proměnnou $elected_i = id$ na P_{max}

Volební algoritmus, Ring algorithm – LeLann, 1977

- Algoritmus má kvadratickou komunikační složitost, každý proces z N procesů způsobí generování N zpráv
- Algoritmus je uniformní,decentralizovaný, asynchronní
- Obnovený proces po výpadku nemusí spouštět volební běh, může id koordinátora zjistit např. dotazem poslaným po kruhu

- Následující algoritmus, *Chang, Roberts*, vylepšuje LeLanův algoritmus tím, že odstraňuje z kruhu volební běhy procesů o kterých je jasné, že nebudou zvoleni

Volební algoritmus, Ring algorithm – volba po kruhu

- Chang, Roberts, 1979
- N procesů, každý proces P_i má definovaný komunikační kanál ke svému následníkovi $P_{(i+1) mod\ N}$
- každý proces si uchovává id *master* v $elected_i$
  - ✓ $elected_i$ obsahuje id procesu s maximální prioritou nebo $\perp$
- každý proces je ve stavu participant (účastník) nebo non-participant volby
- iniciálně není žádný proces účastníkem volby
- proces P_i zahajující svůj volební běh se označí za účastníka volby a pošle volící zprávu $election(P_i)$ svému (levému) sousedovi

Volební algoritmus, Ring algorithm – Chang, Roberts

- Uzel P_j po získání zprávy $election(P_i)$ od svého pravého souseda reaguje následovně
  - ✓ $P_i > P_j$:

    P_j přepošle $election(P_i)$ svému levému sousedovi, běží volební běh P_i
  - ✓ $P_i < P_j$ a P_j dosud není účastník volby:

    P_j pošle $election(P_j)$ svému levému sousedovi, startuje svůj volební běh, označí se za účastníka volby
  - ✓ $P_i < P_j$ a P_j je účastník volby:

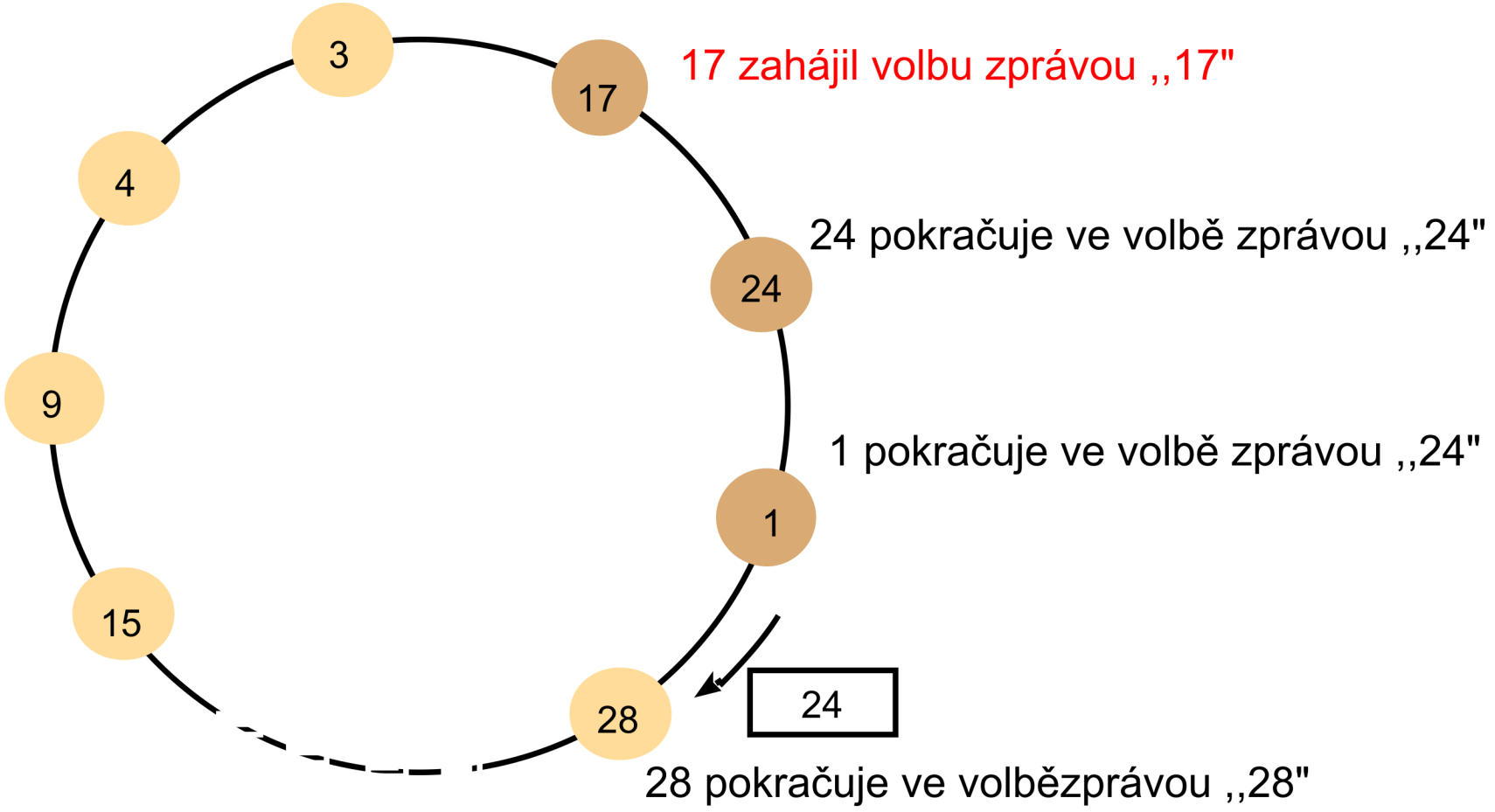
    P_j přijatou zprávu nepřeposílá, jeho volební běh již běží volební běh P_i ruší
  - ✓ $P_i = P_j$:

    P_j získal volící zprávu, kterou on vyslal, přepne se do stavu vedoucí uzel a označí se za neúčastníka volby a pošle po kruhu ukončovací zprávu $elected(P_i)$ oznamující identitu vedoucího uzlu

Volební algoritmus, Ring algorithm – Chang, Roberts

- Uzel P_i po získání zprávy $elected(P_j)$ od svého pravého souseda reaguje následovně
  - ✓ označí se za ne-účastníka volby
  - ✓ poznačí si identifikaci nového vedoucího uzlu
  - ✓ a pokud není novým koordinátorem, zprávu $elected(P_j)$ přepošle svému levému sousedovi
- sledování stavů účastník / ne-účastník co nejdříve likviduje zbytečné násobné volící běhy, vždy dříve než se oznámí výsledek vítězné volby
- výpadek uzlu znamená výpadek celé úlohy, v praxi by se musela vyvolávat rekonstrukce kruhu
- Komunikační kanály musí dodržovat FIFO pořadí přenosu zpráv

Volební algoritmus, Ring algorithm – Chang, Roberts



pokud v kruhu procesů není proces s vyšší identifikací, 28 získá po čase zprávu ,,28" a pošle po kruhu zprávu, že je zvolený 28

Volební algoritmus, Ring algorithm – Chang, Roberts

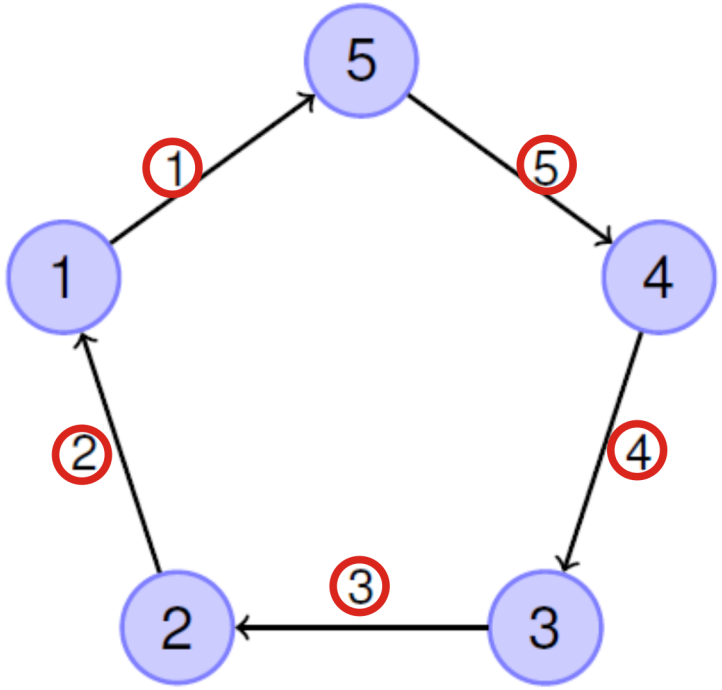
- správnost algoritmu, splnění podmínky bezpečnosti
  - ✓ pouze uzel P_j s maximální hodnotou P_j získá od svého pravého souseda ukončovací zprávu a vstoupí do stavu vedoucí uzel
  - ✓ uzly $P_i \neq P_j$, kde P_j je maximální hodnota identifikace, nikdy nevstoupí do stavu vedoucí uzel
- Jestliže volbu spustí jediný proces, pak v nejhorším případě (proces s nejvyšší id je jeho pravý soused), algoritmus generuje $3N - 1$ zpráv
  - ✓ běh volebního algoritmu se spouští dvakrát, druhý se ukončí volbou
  - ✓ druhý běh se spouští po předání prvních $N - 1$ zpráv $\boldsymbol{election}$
  - ✓ druhý běh způsobí předání dalších N zpráv $\boldsymbol{election}$
  - ✓ zvolený uzel způsobí vyslání N zpráv $\boldsymbol{elected}$

Volební algoritmus, Ring algorithm – Chang, Roberts

- Komunikační složitost algoritmu spuštěného souběžně všemi potenciálními iniciátory je $O(N^2)$, časová složitost je $O(N)$
- Algoritmus je asynchronní a uniformní a decentralizovaný

Volební algoritmus, Ring algorithm – Chang, Roberts

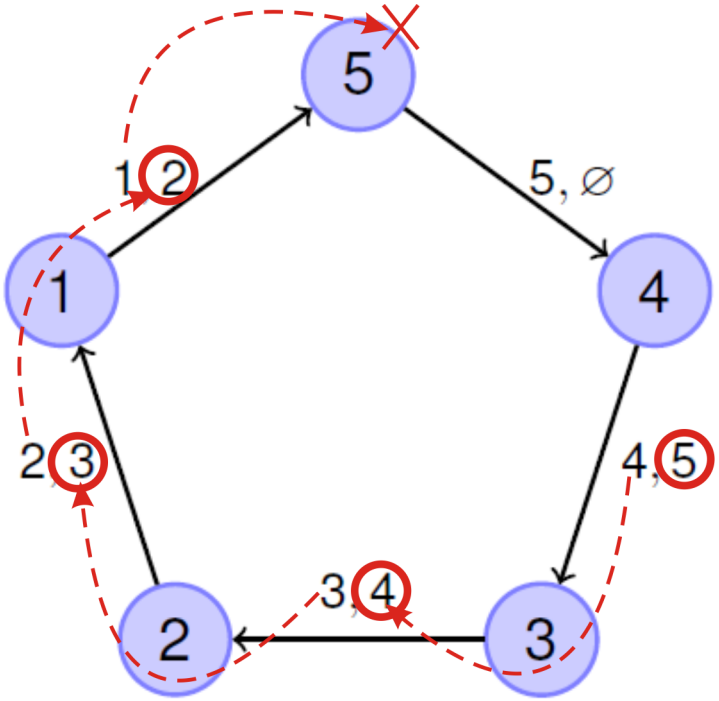
Assume all p_is are initiators.



Messages transmitted:

⟨1⟩	1 times
⟨2⟩	1 times
⟨3⟩	1 times
⟨4⟩	1 times
⟨5⟩	1 times
total	5 times

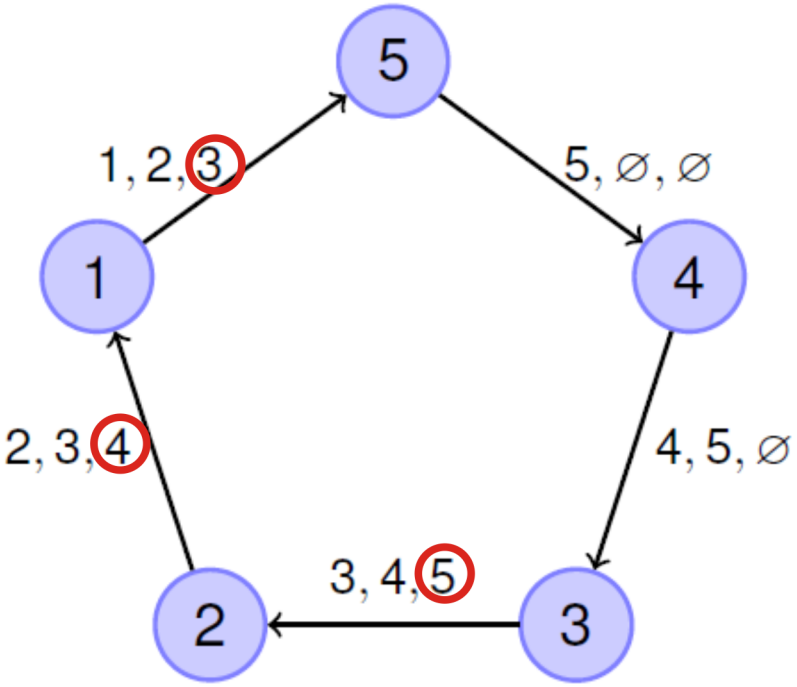
Volební algoritmus, Ring algorithm – Chang, Roberts



Messages transmitted:

⟨1⟩	1 times
⟨2⟩	2 times
⟨3⟩	2 times
⟨4⟩	2 times
⟨5⟩	2 times
total	9 times

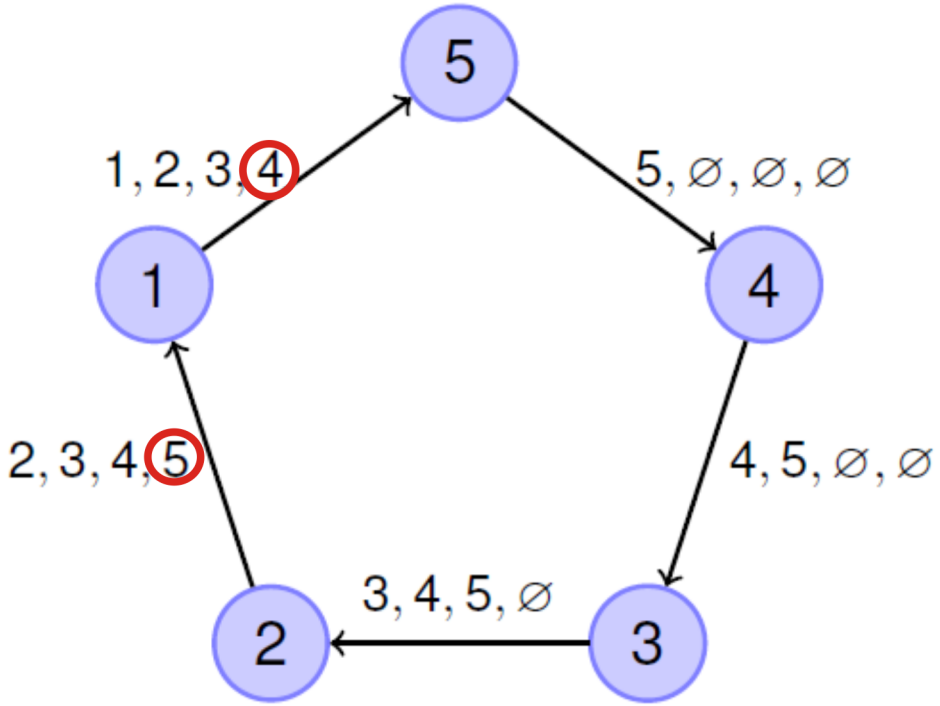
Volební algoritmus, Ring algorithm – Chang, Roberts



Messages transmitted:

⟨1⟩	1 times
⟨2⟩	2 times
⟨3⟩	3 times
⟨4⟩	3 times
⟨5⟩	3 times
total	12 times

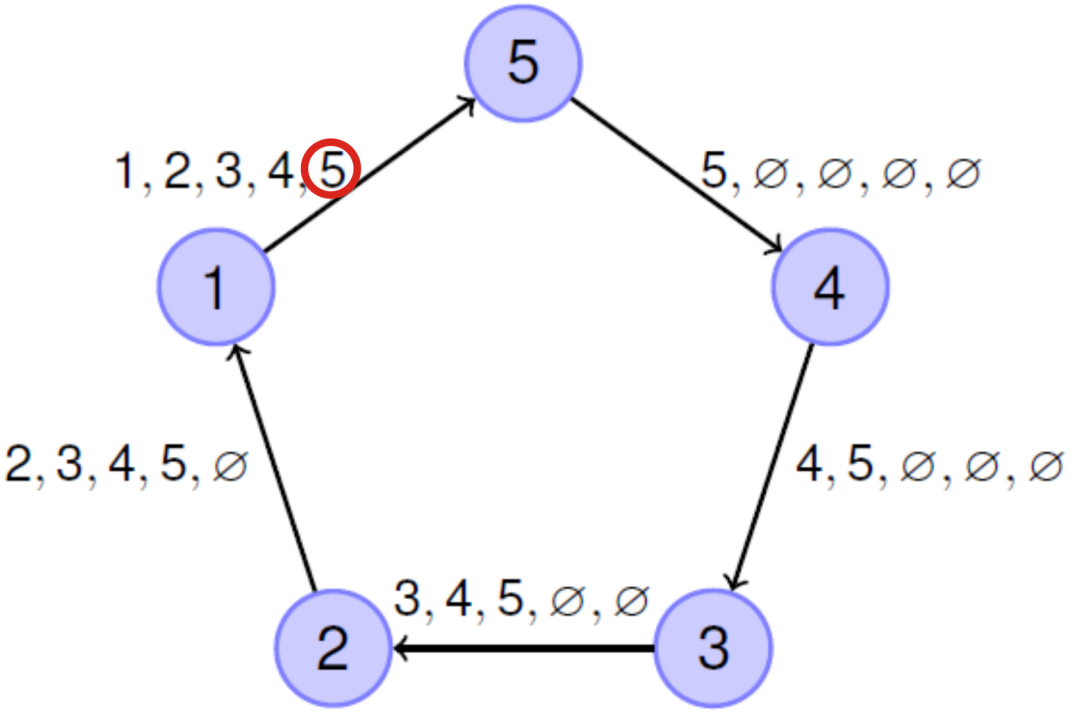
Volební algoritmus, Ring algorithm – Chang, Roberts



Messages transmitted:

⟨1⟩	1 times
⟨2⟩	2 times
⟨3⟩	3 times
⟨4⟩	4 times
⟨5⟩	4 times
total	14 times

Volební algoritmus, Ring algorithm – Chang, Roberts



Messages transmitted:

⟨1⟩	1 times
⟨2⟩	2 times
⟨3⟩	3 times
⟨4⟩	4 times
⟨5⟩	5 times
total	15 times

- ✓ plus 5 zpráv oznamujících číslo zvoleného uzlu, 5, takže 20 zpráv
- ✓ $(n + (n-1) + \ldots + 1) + n = n(n+1)/2 + n$, složitost $O(n^2)$
- ✓ $5(5+1)/2 + 5 = 30/2 + 5 = 20$

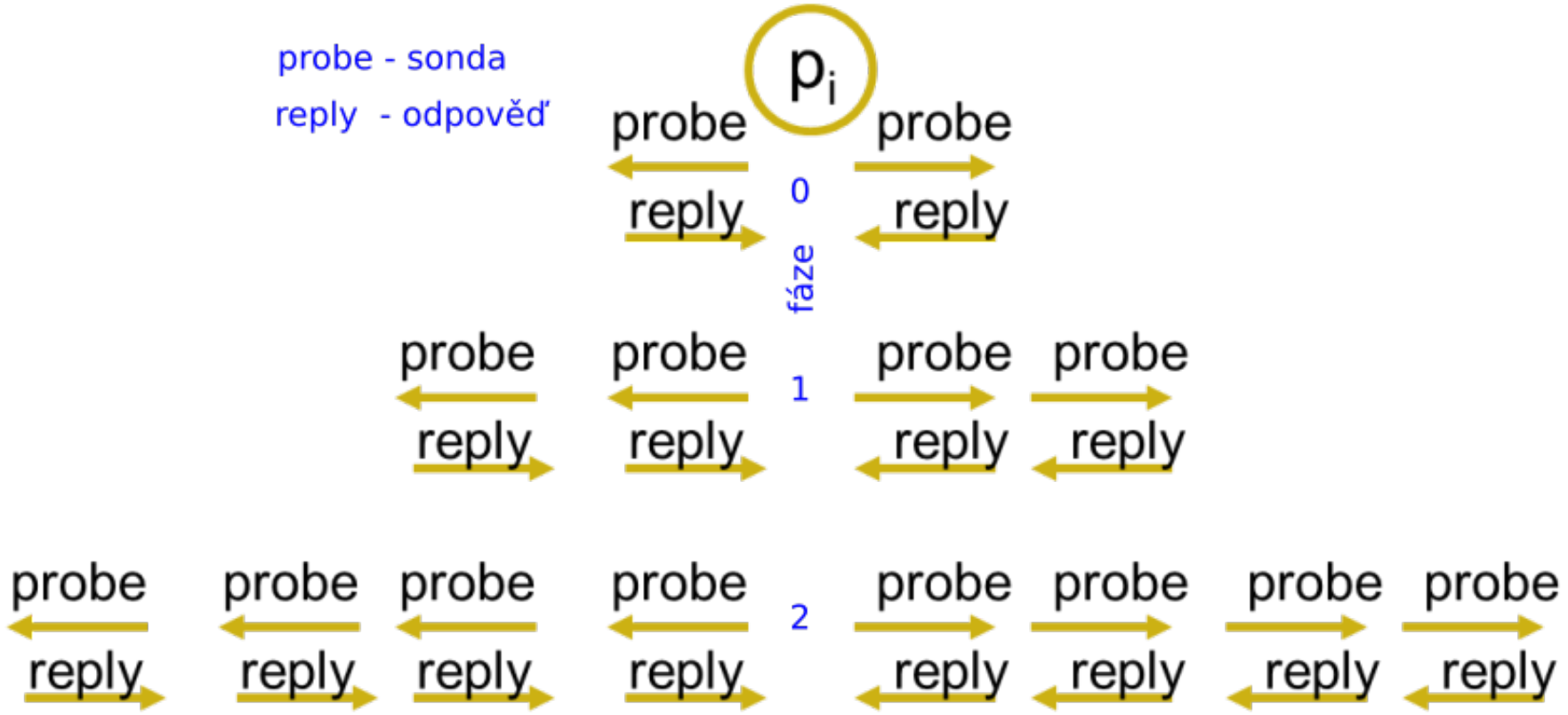
Volba vůdce, Ring algorithm, snížení komunikační složitosti

- Lze snížit komunikační složitost pod $O(n^2)$?
- Zkusme šířit zprávy s menším ID na menší vzdálenost po kruhu
  - ✓ Ukážeme, že lze dosáhnout komunikační složitost $O(n.\log\ n)$ v nejhorším případě uspořádání uzlů na kruhu
  - ✓ Za cenu použití komplikovanějšího algoritmu

Volba vůdce, Ring algorithm, snížení komunikační složitosti

- Každý proces bude postupně zkoumat vzdálenější a vzdálenější sousedy v obou směrech na kruhu
- V každé fázi zkoumání zdvojnásobí vzdálenost zkoumání
- Jestliže jeho sonda dosáhne proces s větším ID, zkoumání se zastaví
- Jakmile sonda dosáhne cíleného souseda, vrací se odpověď iniciátorovi
- Jestliže iniciátor dostane odpověď z obou směrů, přejde do další fáze
- Jakmile proces získá sondu se svým ID, volí se za vůdce

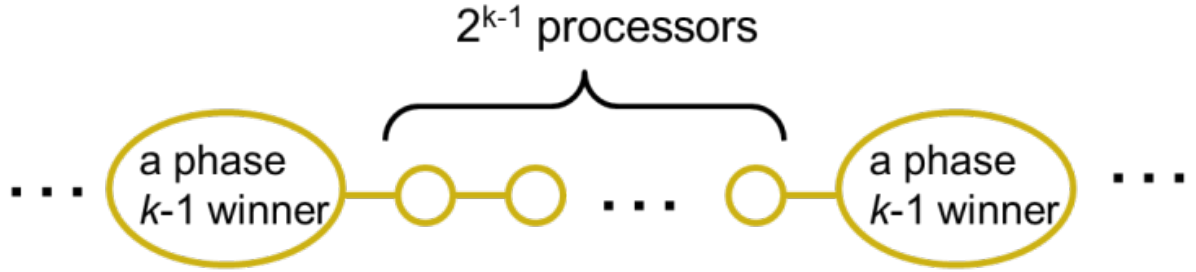
Volba vůdce, Ring algorithm, snížení komunikační složitosti



- ✓ každá zpráva náleží konkrétní fázi a má svého iniciátora
- ✓ Délka průzkumu ve fázi k je 2^k
- ✓ v každé fázi se přenese až 4×2^k zpráv

Volba vůdce, Ring algorithm, snížení komunikační složitosti

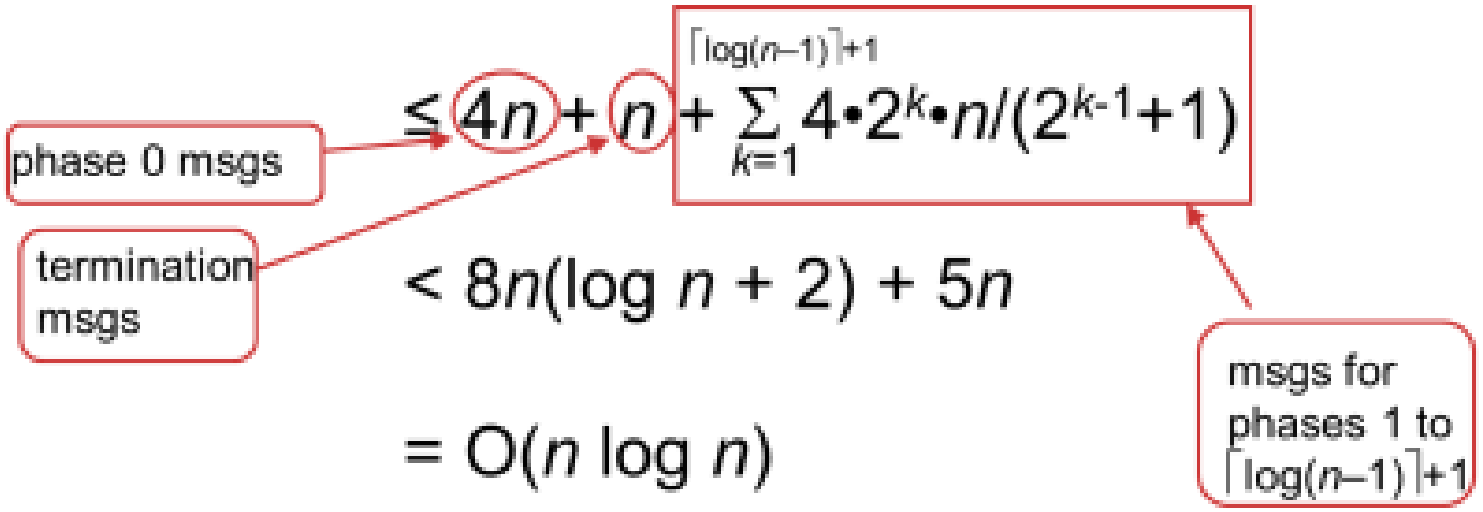
- Ve fázi $k = 0$ může zahájit zkoumání každý proces
- Ve fázi $k > 0$ může zahájit zkoumání každý vítěz fáze fáze $k - 1$
  - ✓ vítěz má největší ID při zkoumání do vzdálenosti 2^{k-1}
- Maximální počet vítězů fáze $k - 1$ nastane pokud jsou naskládaní tak hustě, jak je to možné



  - ✓ ve fázi $k - 1$ bude nejvýše $n/(2^{k-1} + 1)$ vítězů
  - ✓ v každé fázi se možný počet vítězů snižuje cca na polovinu od $n/(2^{k-1} + 1)$ do $n/(2^k + 1)$

Volba vůdce, Ring algorithm, snížení komunikační složitosti

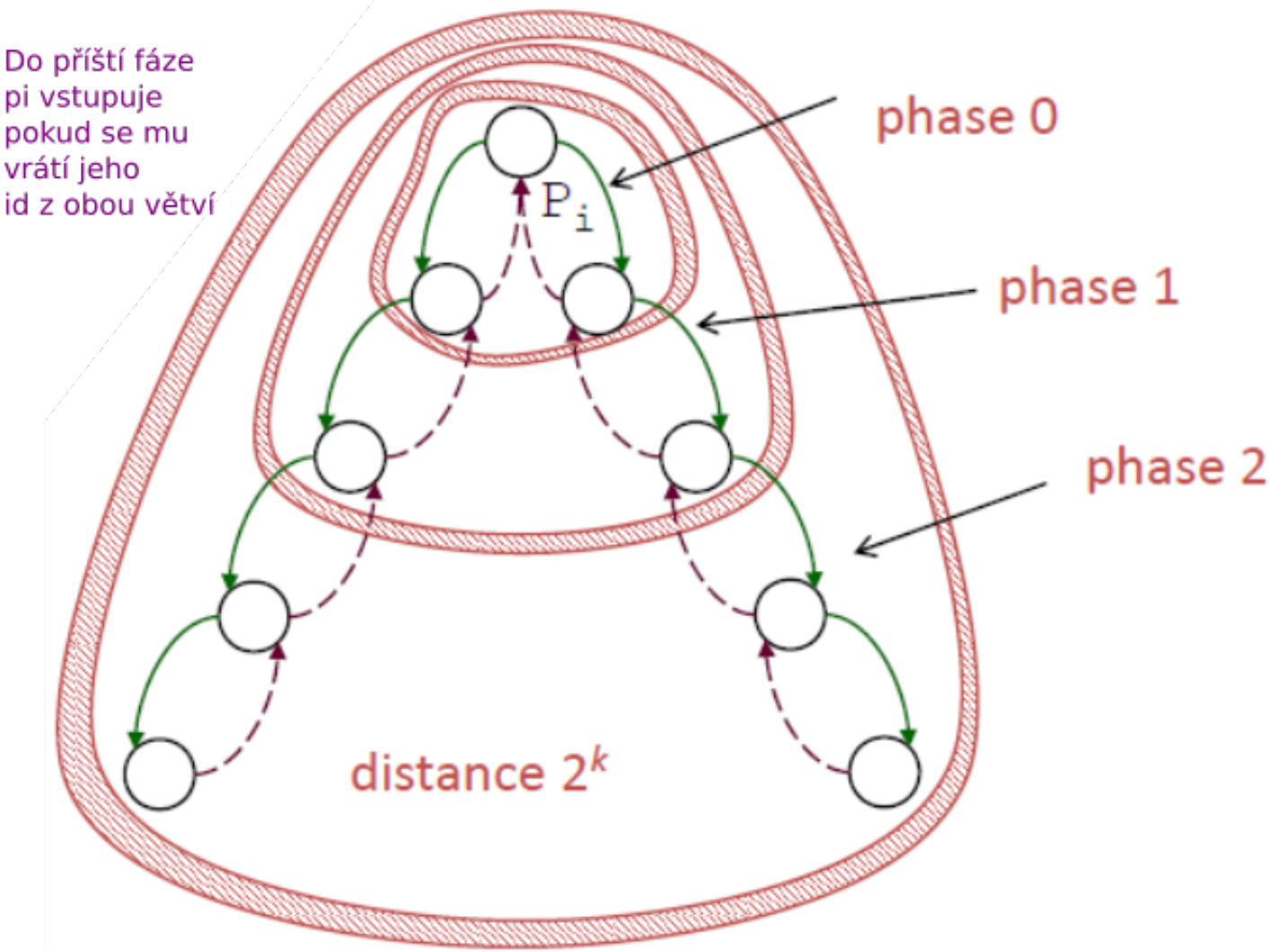
- Celkový počet zpráv je sumou přes všechny fáze
  - počtů vítězů v těchto fázích krát
  - počty zpráv zahajovaných těmito vítězi



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

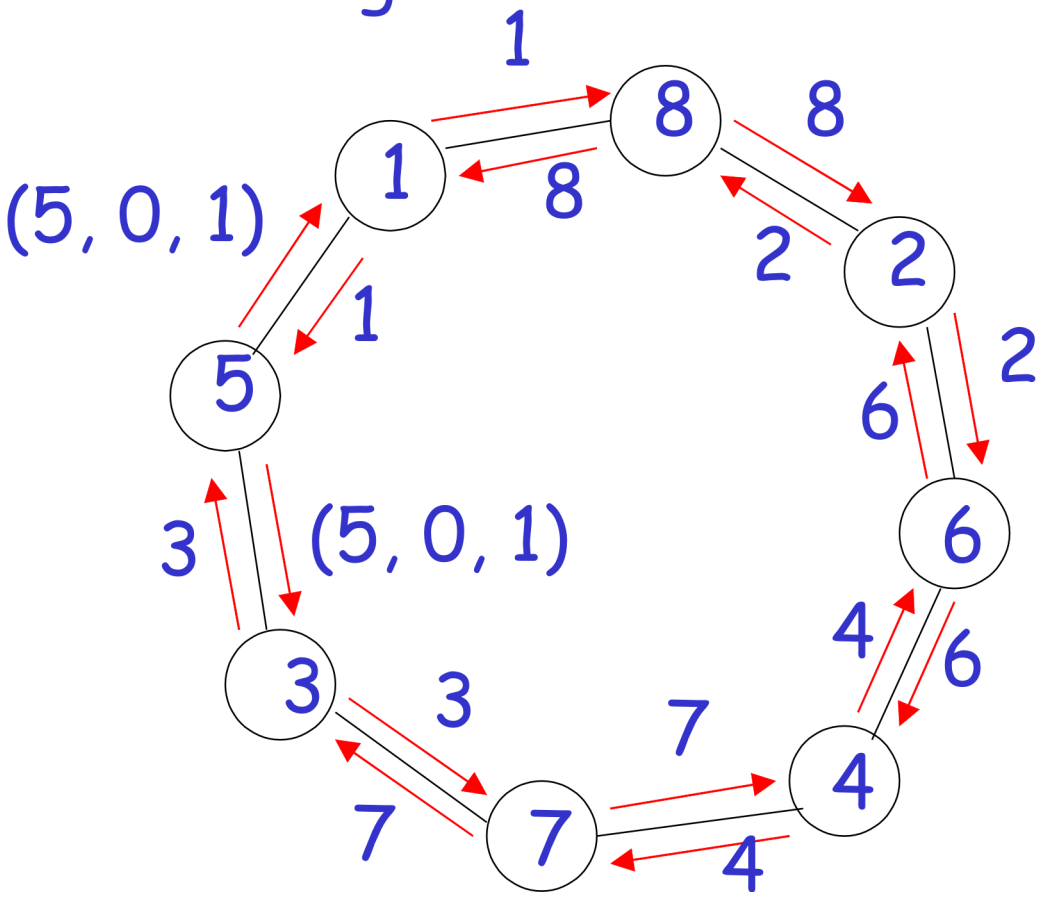
- r. 1980, topologie – obousměrný (neorientovaný) kruh
- Iniciátor v jednotlivých fázích $r = 0, 1, 2, \ldots$ posílá svoji identitu oběma směry do vzdálenosti 2^r vlnou nesoucí průzkumnou zprávu (jsem p_i, mám nejvyšší prioritu ?)
  - ✓ očekává její pozitivní potvrzení po odrazu v posledním z vlnou oslovených uzlů
- Identity cestující po kruhu jsou likvidované stejně jako v algoritmu Chang-Roberts
- Proces p_i je vítěz ve fázi $r = 0, 1, \ldots$ pokud má nejvyšší id ze všech procesů, které jsou od něho vzdálené až 2^r skoků
- Do fáze $r + 1$ vstupují pouze vítězové fáze r
- Ostatní procesy pouze přenášejí zprávy mezi sousedy

Volební algoritmus, Ring algorithm – Hirschberg-Sinclair



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

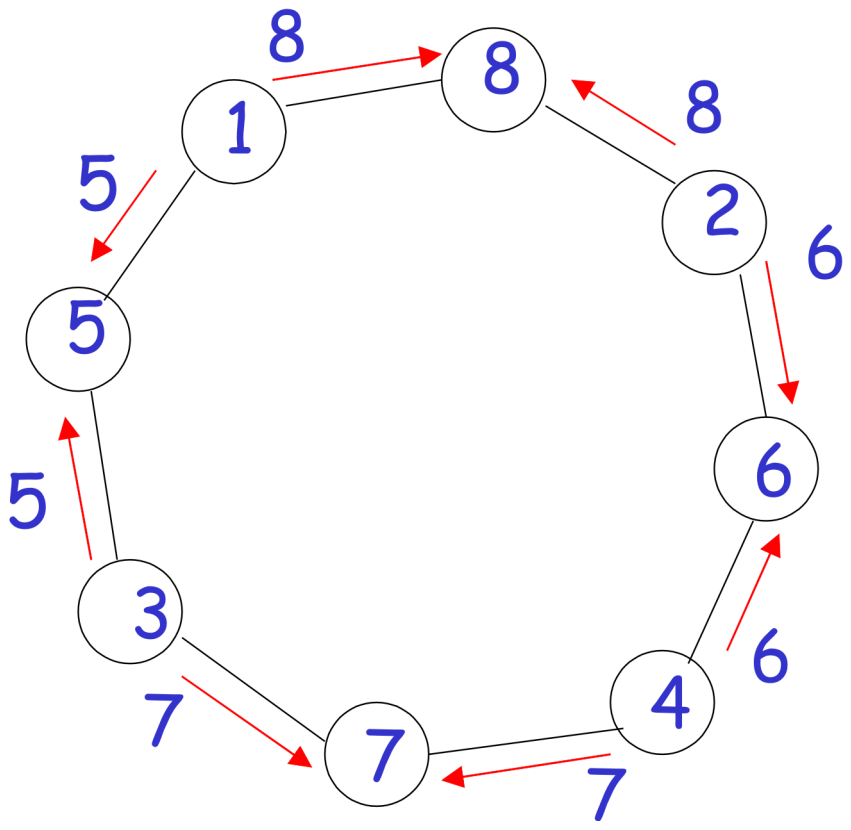
Phase 0: send(id, current phase, step counter) to 1-neighborhood



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

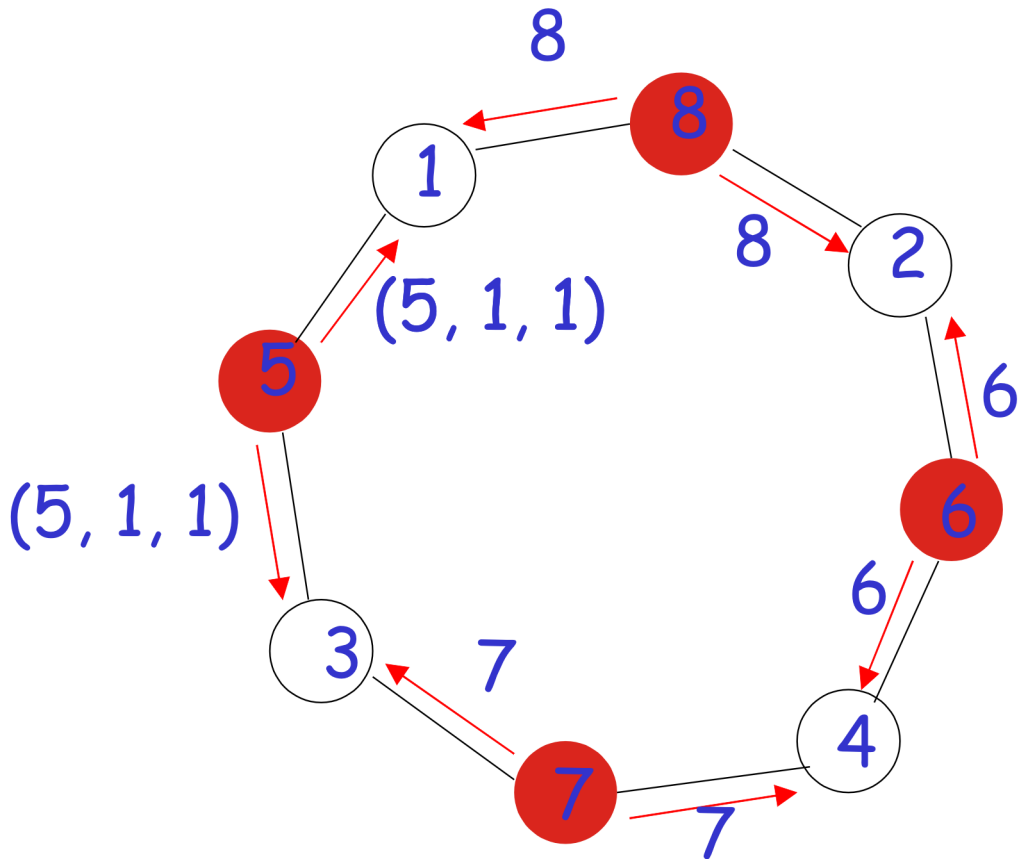
If: received **id** > current id

Then: send a reply(**id**)



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

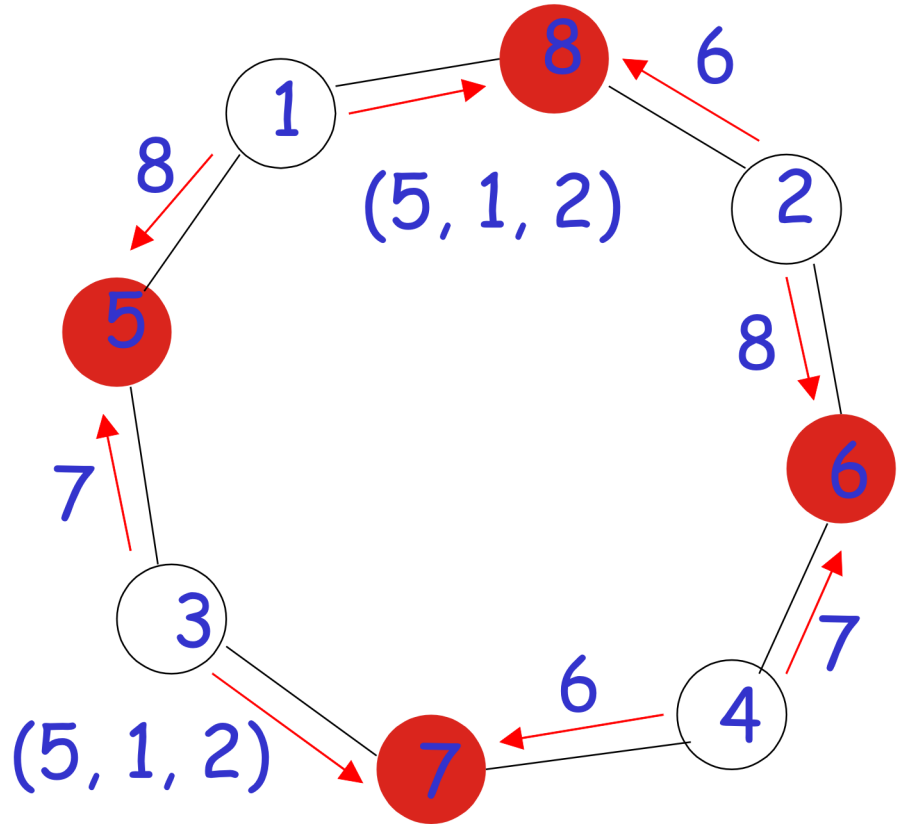
Phase 1: send(id,1,1) to left and right adjacent in the 2-neighborhood



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

If: received **id** > current id

Then: forward(**id**,1,2)

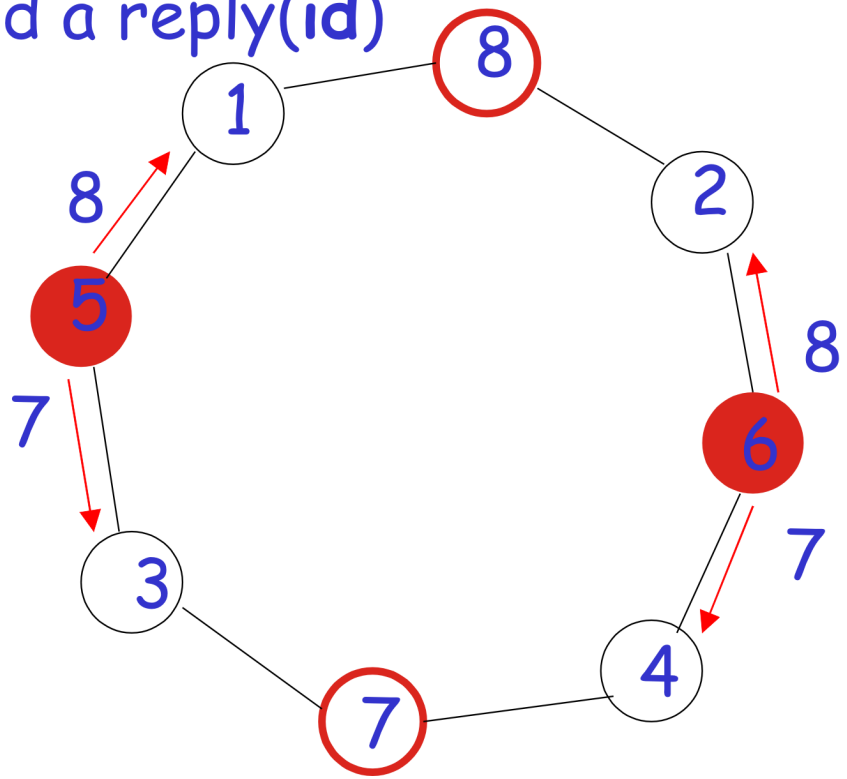


Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

At second step: since step counter=2, I'm on the boundary of the 2-neighborood

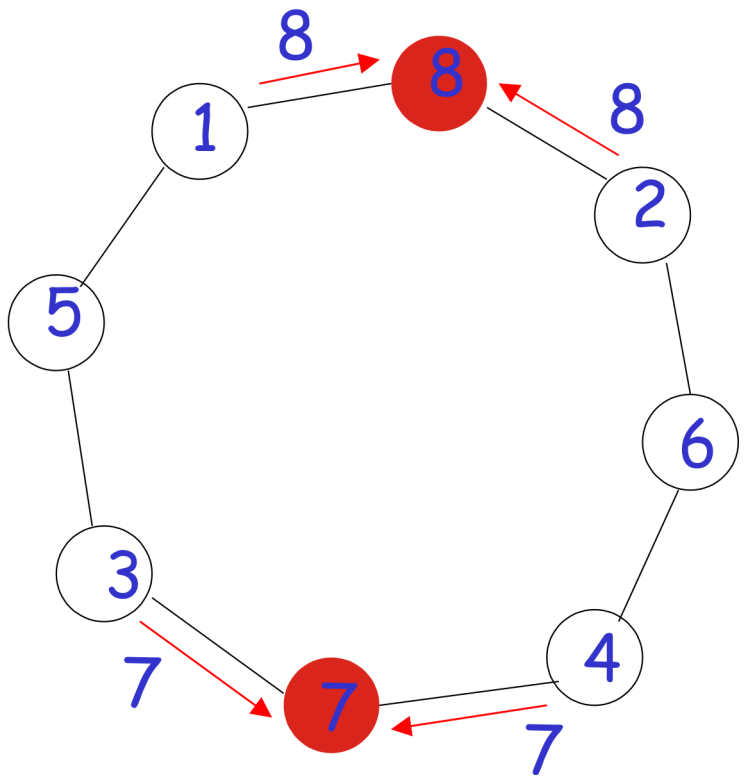
If: received **id** > current id

Then: send a reply(**id**)



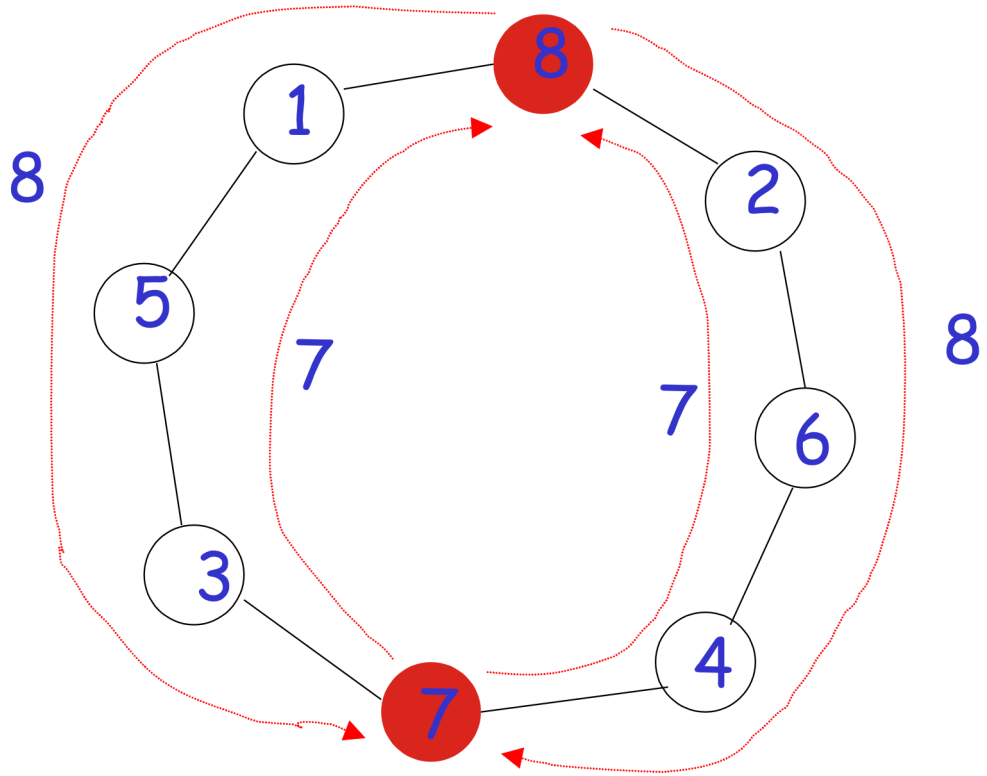
Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

If: a node receives a reply with another id
Then: forward it
If: a node receives both replies
Then: it becomes a temporal leader



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

Phase 2: send id to 2^2-neighborhood

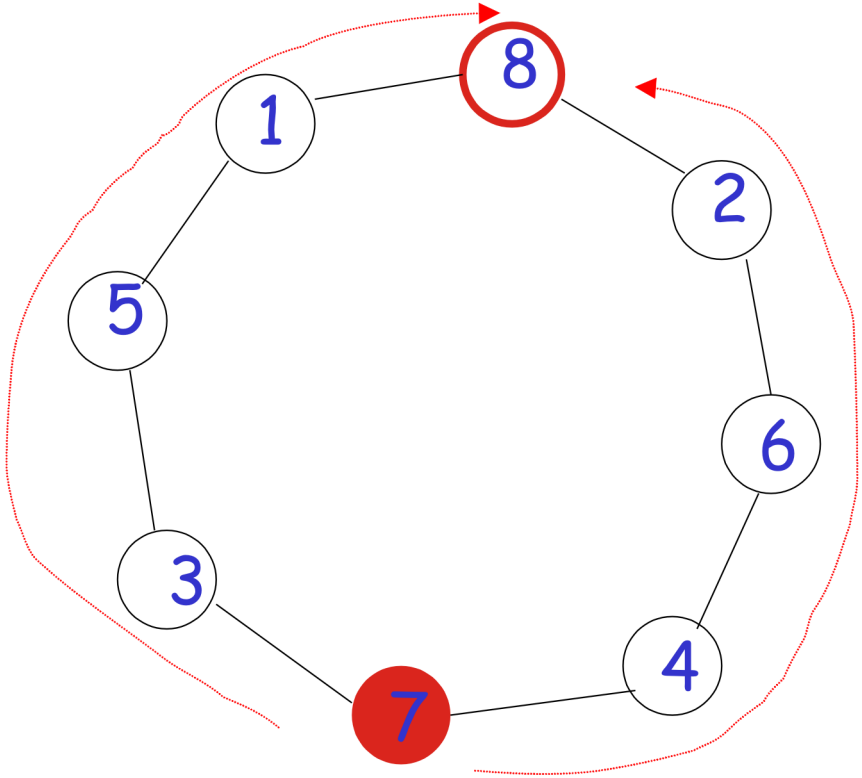


Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

At the 2^2 step:

If: received **id** > current id

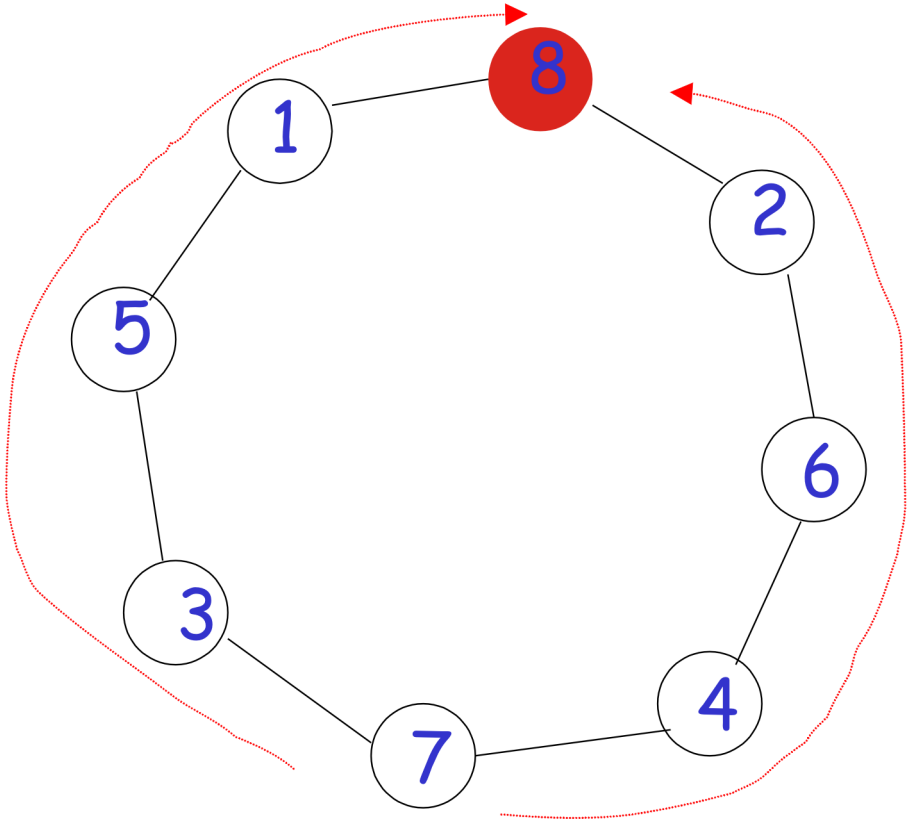
Then: send a reply(**id**)



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

If: a node receives both replies

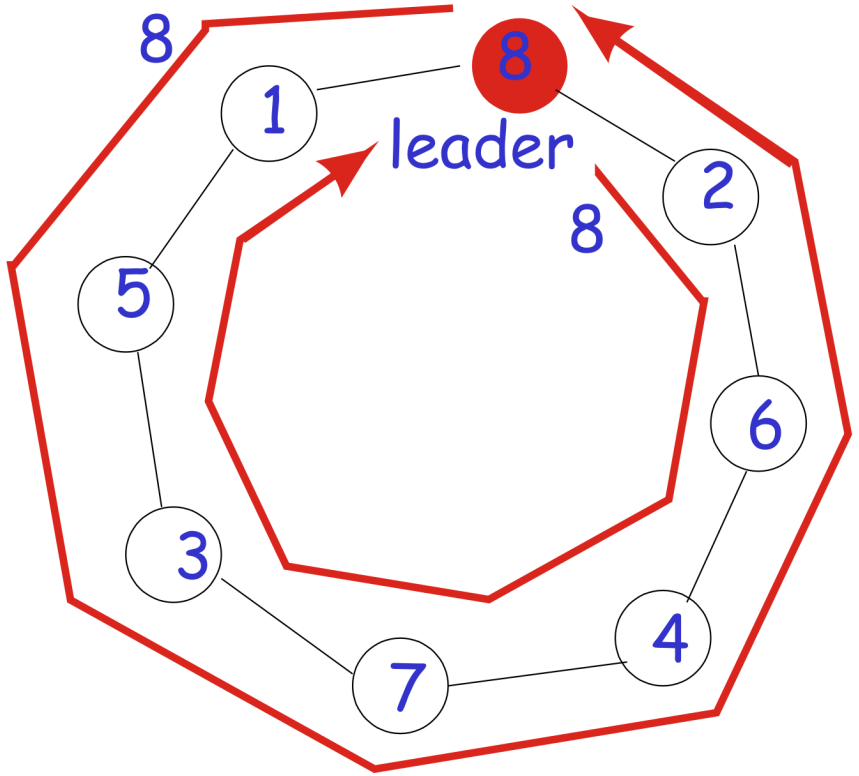
Then: it becomes the temporal leader



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

Phase 3: send id to 8-neighborhood

Þ The node with id 8 will receive its own probe message, and then becomes leader!



Volební algoritmus, Ring algorithm – Hirschberg-Sinclair

- n uzlů, $O(log\ n)$ fází
- Délka zkoumání ve fázi k je 2^k
- Počet zpráv souvisejících s jedním dočasně vedoucím procesem ve fázi k je nejvýše 4×2^k
  - ✓ ve fázích $0, 1, \ldots, i, \ldots, log\ n$ to jsou počty zpráv $4, 8, \ldots, 2^{i+1}, \ldots 2^{log\ n+1}$ a počty dočasných vítězů $n, n/2, \ldots, n/2^i, \ldots, n/2^{log\ n-1}$
  - ✓ zatímco ve fázi 0 každý z n iniciátorů generuje vysílání 4 zpráv ve fázi 1 každý z $n/2$ iniciátorů generuje po 8 zprávách atd
- Složitost z hlediska počtu zpráv je tudíž $O(n.log\ n)$

Volební algoritmus, Bully algorithm

- Bully
  - tyran, osoba, která obvykle otravuje a zastrašuje menší nebo slabší lidi
  - proces s nejvyšší prioritou vynucující si vůdcovskou roli
- Použitelný v případech, ve kterých
  - proces může poslat zprávu každému z ostatních procesů v definované množině procesů
  - se mohou vyskytovat výpadky uzlů (procesů)
  - komunikační systém je spolehlivý, zprávy se neztrácí
- Výpadek uzlu se pozná uplynutím časového intervalu (timeout)

Volební algoritmus, Bully algorithm

- Když se proces rozhodne volit, zvolí sebe a
  zašle o tom zprávu všem procesům s vyšší prioritou,
  spouští svůj volební běh
- Když proces přijme zprávu o volebním běhu jiného procesu,
  má určitě vyšší prioritu,vrátí mu zpět zamítavou odpověď
  a sám spouští svůj volební běh – vyšle žádosti všem
  prioritnějším procesům
- Když procesu dostane zamítavou odpověď, existuje proces
  s vyšší prioritou a proces se svůj volební běh končí
- Když proces nedostane zamítavou odpověď, volbu vyhrál,
  je novým *master* procesem a pošle o tom zprávu všem
  ostatním procesům

Volební algoritmus, Bully algorithm

- Proces P_i, který rozpozná výpadek šéfa – tyrana,
  se pokouší prohlásit za nového šéfa sebe
  - ✓ spuštěním volebního běhu zjišťuje,
    zda on má ze všech právě činných procesů nejvyšší prioritu,
    (musí tudíž všechny kooperující procesy znát)
  - ✓ nemá-li nejvyšší prioritu, dá spuštěním volebního běhu podnět
    k volební aktivitě právě činným procesům s vyšší prioritou
- Pokud obnoví svůj běh vypadlý proces,
  spouští okamžitě svůj volební běh
  - ✓ Pokud není aktivní žádný proces s vyšší prioritou než je jeho priorita,
    přebere tak roli šéfa a všichni se o tom dozví
  - ✓ nebo se dozví, kdo je šéfem

Volební algoritmus, Bully algorithm

Algoritmus

- P_i zjistil nečinnost dosavadního šéfa
  - ✓ P_i pošle volící zprávu – $election(P_i)$

    všem procesům s vyšším prioritním číslem než má on, kterou spouští nový volební běh

    P_i se pokouší prohlásit sebe za šéfa
  - ✓ P_i čeká dobu T na obdržení odpovědí s odmítnutím této volby od procesů s vyšší prioritou, $reject$
  - ✓ Pokud iniciátor volby P_i nedostane do doby T žádné odmítnutí volby
    - není aktivní žádný proces s vyšší prioritou než je priorita P_i a
    - P_i se zprávou $coordinator(P_i)$,

      zaslanou všem procesům, prohlašuje za šéfa

. . .

Volební algoritmus, Bully algorithm

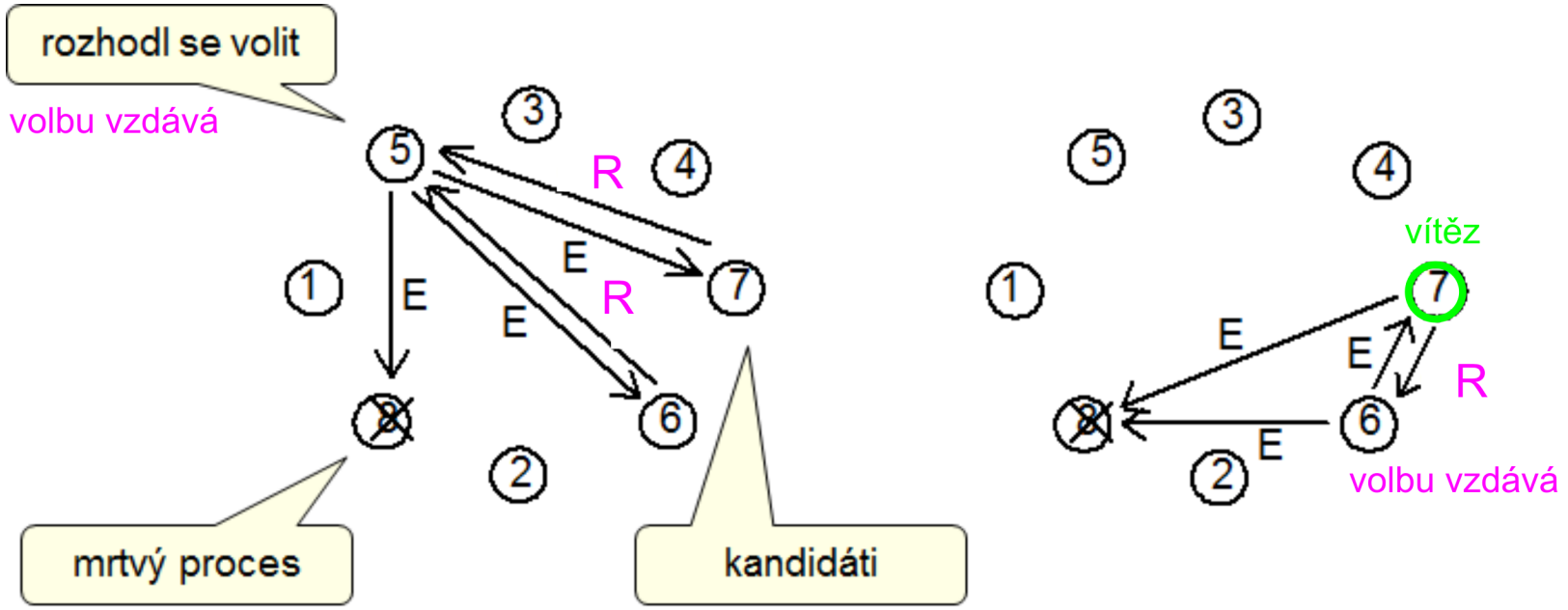
- ✓ Pokud iniciátor volby P_i dostane do doby T odmítnutí volby, $reject$
  - je aktivní proces s vyšší prioritou než i, např. j a
  - P_i čeká dobu T_1 na obdržení zprávy od P_j,
    že pro další období bude roli šéfa vykonávat P_j, $coordinator(P_j)$
- ✓ Pokud iniciátor P_i nedostane do doby T_1 zprávu od P_j,
  že P_j přebírá roli šéfa,
  - P_i startuje svůj volební běh znovu (P_j mezitím vypadl)

Volební algoritmus, Bully algorithm

- Pokud P_i není šéf, může kdykoliv dostat od jiného procesu P_j jednu ze dvou zpráv $coordinator(P_j)$ nebo $election(P_j)$
  - ✓ $coordinator(P_j)$ a $j > i$, pak platí, že
    - P_j je nový koordinátor, P_i si to zapamatuje v proměnné $elected$
  - ✓ $coordinator(P_j)$ a $j < i$, pak platí, že
    - P_i byl v době volby vypadlý a už obnovil svoji činnost, ale ještě nespustil volební běh a proto spouští volební běh
  - ✓ $election(P_j)$ a $j < i$, pak
    P_j startoval volební běh neprávem, P_i pošle P_j odpověď $reject$ a zahajuje algoritmus své vlastní volby, pokud tak dosud neučinil
  - ✓ $election(P_j)$ a $j > i$, pak
    čeká na zprávu $coordinator(P_j)$
    a pokud ji do timeoutu nezíská, spouští volební běh

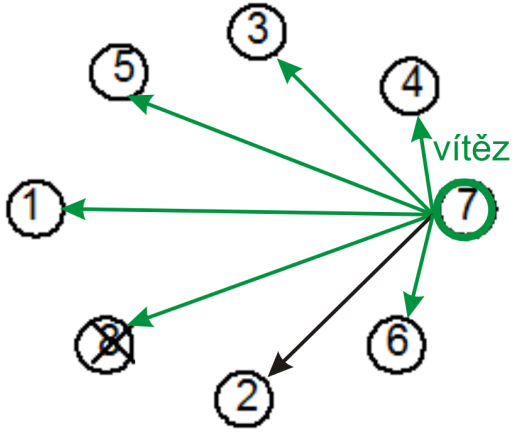
Volební algoritmus, Bully algorithm

- Složitost Bully algoritmu
  - ✓ $O(N^2)$ zpráv v nejhorším případě
  - ✓ tj. když volbu zahájí proces s nejnižší identifikací, pak postupně zahajuje volbu $N - 1$ procesů rozesíláním až $N - 1$ zpráv



Volební algoritmus, Bully algorithm

- Složitost Bully algoritmu
  - ✓ $O(N^2)$ zpráv v nejhorším případě
  - ✓ tj. když volbu zahájí proces s nejnižší identifikací, pak postupně zahajuje volbu $N - 1$ procesů rozesíláním až $N - 1$ zpráv



Volební algoritmus, Bully algorithm

1. 4 aktivní procesy, $P_{1,\ldots,4}$, P_4 je koordinátor
2. P_1 a P_4 vypadnou. P_2 nedostane do doby T odpověď od P_4
   a pošle proto P_3 svou volící zprávu
3. P_3 odpoví P_2 odmítnutím a zašle svou volící zprávu P_4
4. P_2 dostane odmítnutí své volby od P_3 a po dobu T_1 čeká
   na potvrzení, že P_3 je koordinátor
5. P_4 mlčí, je stále vypadlý, do doby T neodmítne P_3 a
   P_3 se prohlásí za koordinátora a všem to sdělí
6. Později, když se P_1 obnoví, zašle svou volící zprávu P_2, P_3 a P_4
7. P_2 a P_3 odpoví odmítnutím P_1 a spustí svoje volby a P_3 se opět
   stejným postupem zvolí za koordinátora
8. Až se obnoví P_4 sdělí to P_1, P_2 a P_3, volbu nezahajuje,
   nezná proces vyšší prority