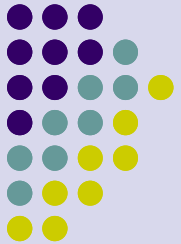
Distribuované algoritmy - přehled



Přednášky z Distribuovaných systémů

Ing. Jiří Ledvina, CSc.

Distribuované vzájemné vyloučení

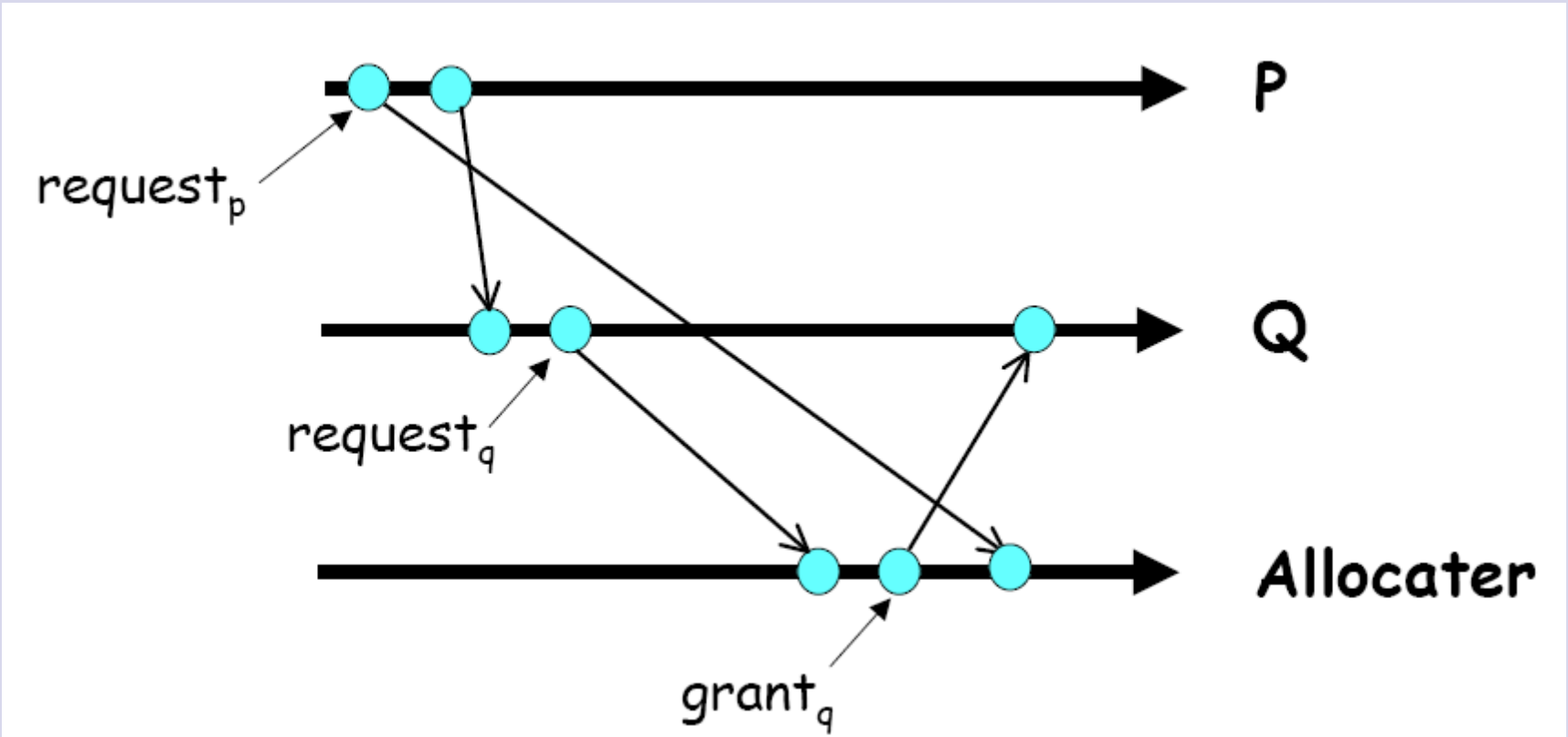
- Základní rozdělení
  - Centralizované metody (sequencer)
  - Decentralizované metody
    - Založené na soupeření
    - Založené na předávání pověření

Distribuované vzájemné vyloučení

- Existuje pevný počet procesů, které sdílí jeden zdroj.
- Zdroj může používat v jednu chvíli pouze jeden proces.
- Podmínky
  - Proces, kterému bylo povoleno užívat zdroj jej musí uvolnit dříve, než jej začne používat jiný proces.
  - Požadavky procesů musí být zpracovány v pořadí, ve kterém vznikly.
  - Jestliže platí, že zdroj je procesy v konečném čase uvolněn, pak musí být také každý požadavek v konečném čase zpracován.

Distribuované vzájemné vyloučení

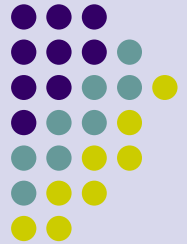
- Centralizované řešení



Distribuované vzájemné vyloučení

- Decentralizované metody založené na soupeření
  - Lamportův algoritmus – libovolná topologie, časové značky (3 fáze)
  - Ricart a Agrawala – libovolná topologie, časové značky (2 fáze)
  - Maekawa – vytváření hlasovacího quora
- Decentralizované metody založené na předávání pověření
  - Suzuki a Kasami – broadcast
  - LeLann – logický kruh
  - Reymond – stromová struktura, rozšíření pro sdílení K identických zdrojů

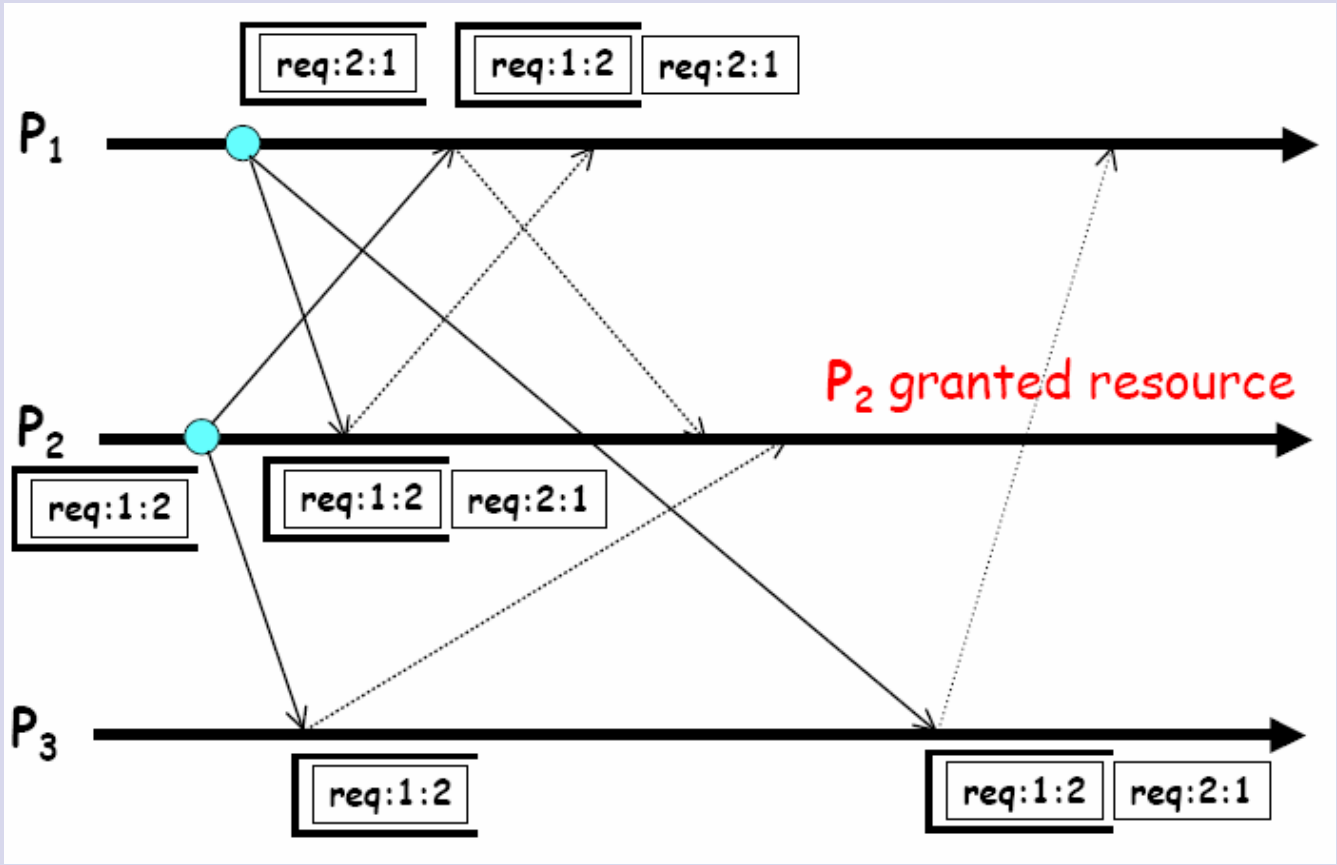
Lamportův algoritmus



- Decentralizovaný algoritmus vzájemného vyloučení
- Předpokládá obousměrné FIFO kanály mezi procesy
- Každý proces udržuje vlastní frontu požadavků
- Používá časových značek k uspořádání požadavků
- Vyžaduje 3(n-1) zpráv
- Probíhá ve třech fázích
  - Požadavek (req)
  - Potvrzení (ack)
  - Uvolnění (rel)

Lamportův algoritmus

Příklad (req: časová značka :proces)

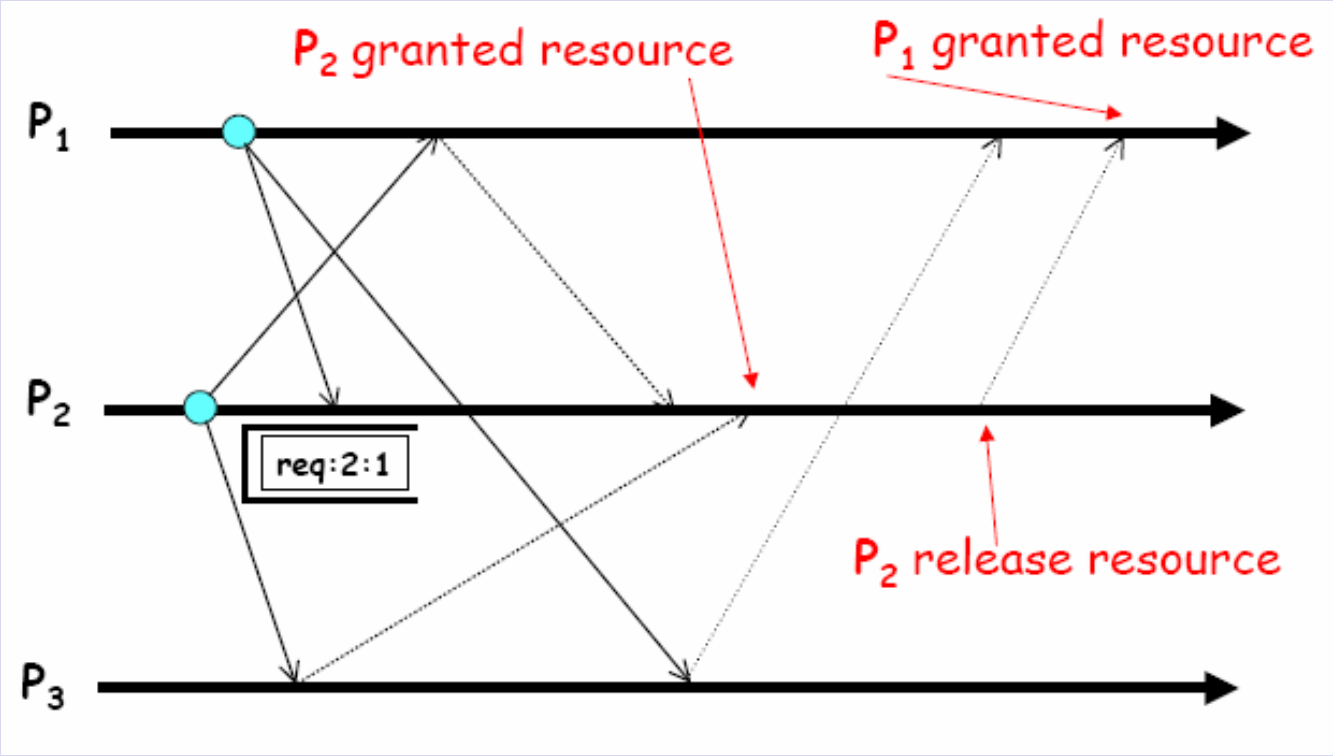


Lamportův algoritmus

- Algoritmus
  - Proces P požaduje zdroj posláním požadavku do všech procesů zasláním zprávy **req**
  - Při příjmu požadavku je požadavek zařazen do fronty požadavků uspořádané dle časových značek, poslání zprávy **ack**
  - Ukončení zpracování požadavku – odstranění požadavku z fronty a poslání zprávy **rel** ostatním procesům
  - Přijetí zprávy **rel** od procesu P – odstranění požadavku procesu P z fronty
  - Povolení zpracování požadavku – požadavek je na prvním místě ve frontě a je potvrzen ostatními procesy

Algoritmus vzájemného vyloučení Ricard - Agrawala

- Optimalizace Lamportova algoritmu snížením počtu zpráv na 2(n-1)



Algoritmus vzájemného vyloučení Ricard - Agrawala

- Algoritmus
  - Proces P požaduje zdroj posláním požadavku do všech procesů zasláním zprávy **req**
  - Při příjmu požadavku je požadavek potvrzen zprávou **ack** pouze tehdy, jestliže přijímající proces zdroj neužívá nebo o něj právě nežádá. Jinak je požadavek zařazen do fronty.
  - Ukončení zpracování požadavku – poslání zprávy **ack** procesům, kterým bylo poslání předtím odepřeno (a odstranění požadavků z fronty)
  - Povolení zpracování požadavku – požadavek je potvrzen ostatními procesy

Algoritmus Maekawa

- Vzájemné vyloučení zajištěno vytvořením hlasovacího quora
- Vyžaduje souhlas $(2\sqrt{N})-1$ procesů

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

Algoritmy založené na předávání pověření

- Suzuki a Kasami – broadcast
- LeLann – logický kruh
- Raymond – stromová struktura, rozšíření pro sdílení K identických zdrojů

Algoritmus vzájemného vyloučení Suzuki-Kasami

- Metoda využívá distribuci pověření mezi jednotlivými procesy
- Každý uzel udržuje celočíselné pole největších sekvenčních čísel obdržených od jednotlivých procesů
- Proces posílá ostatním pověření, které obsahuje
  - frontu požadavků a
  - vektor LN posledních požadavků jednotlivých procesů

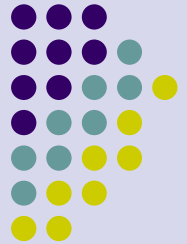
Algoritmus vzájemného vyloučení Suzuki-Kasami

- Pokud chce proces vstoupit do kritické sekce a nemá pověření
  - Zvýší vlastní TS ve vektoru TS a pošle ostatním požadavek (i, TS_i)
- Pokud proces přijme požadavek od i-tého procesu
  - Nastaví položku vektoru TS[i] na maximum
  - Jestliže má pověření a TS[i]=LN[i]+1, pošle pověření do P_i

Algoritmus vzájemného vyloučení Suzuki-Kasami

- Pokud P_i opouští kritickou sekci
  - Nastaví LN[i] v pověření na vlastní vektor časových značek
  - Frontu v pověření nastaví tak, že do ní zahrne všechny procesy, které v ní nejsou a mají časovou značku požadavku o 1 vyšší než je časová značka v pověření
  - Pověření se pošle prvnímu ve frontě (ten se z fronty odstraní)
  - Fronta je posílána v pověření

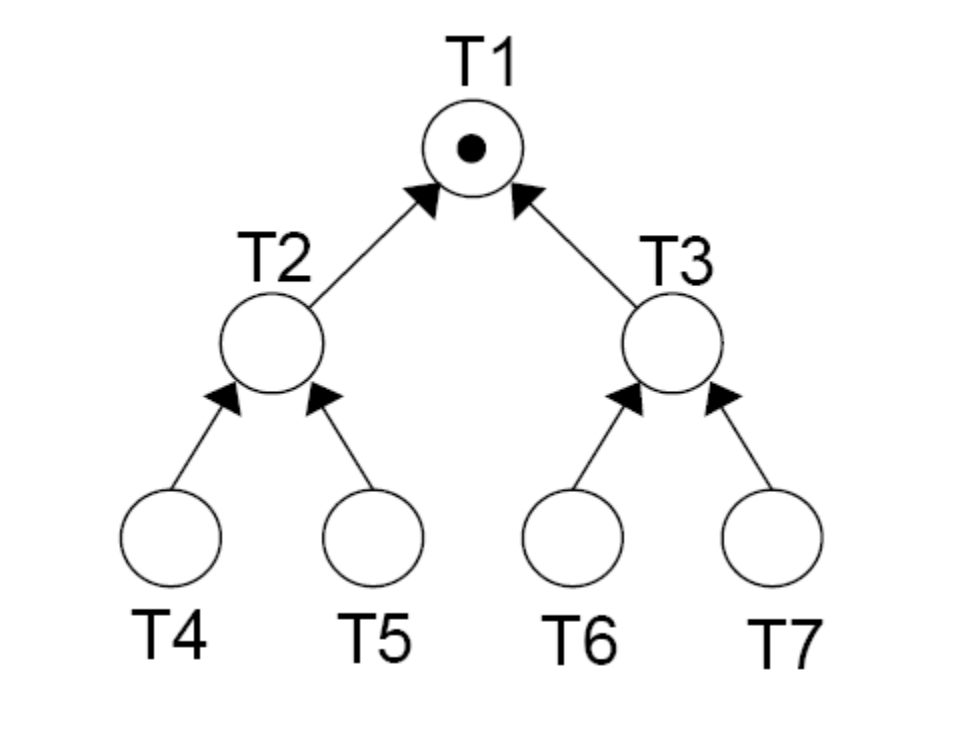
LeLann algoritmus



- Procesy jsou uspořádány do logického kruhu
- Na začátku má pověření proces 0
  - Pověření je posíláno v kruhu (přenos zpráv dvoubodovými spoji)
  - Po obrdžení pověření může proces vstoupit do kritické sekce
  - Po opuštění CS předává pověření dál
- Problém s detekcí ztráty pověření
- Jednoduché rozšíření kruhu

Raymodův algoritmus

- Procesy jsou uspořádány do stromu



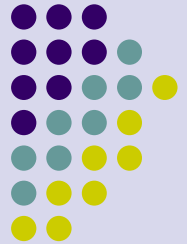
Raymodův algoritmus

- Pracuje na bázi předávání pověření
- Proces udržuje frontu požadavků na pověření od procesů nižších úrovní
- Požaduje-li vstup do CS a fronta je prázdná, posílá požadavek nadřazenému a řadí se do fronty
- Požaduje-li podřízený proces vstup do CS, předá mu pověření nebo jej zařadí do fronty
- Obdrží-li pověření, předá ho prvnímu ve frontě

Algoritmy výběru 1 z N

- úplný graf – Bully algoritmus
- kruhová topologie
  - logický kruh – předávání dle seznamu, rekonstrukce kruhu
  - fyzický kruh – předávání dle sousedství uzlů
- stromová topologie – logický strom

Algoritmy shody



- Shoda na hodnotě
- Interaktivní konzistentnost – shoda na vektoru hodnot
- Problém Byzantinských generálů

Algoritmy detekce ukončení

- asynchronní systémy
- synchronní systémy
  - Dijkstra-Scholten – difuzní výpočty
    - jeden iniciátor
    - uspořádání do stromu, iniciátor je kořen
    - detekce ukončení podle počtu ukončených podřízených
  - Shavit-Francez
    - více iniciátorů
    - všechny procesy se účastní vlny
    - proces který není iniciátorem pokračuje ve vlně
    - pokud strom kolabuje, iniciátory pokračují ve vlně