

# Distribučovaný deadlock



Distribučované systémy

Lekce 6

Ing. Jiří Ledvina, CSc.

## Předpoklady

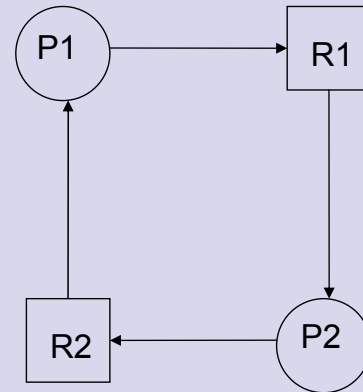


- Systém má pouze znovupoužitelné zdroje
- Existuje pouze výlučný přístup ke zdrojům
- Každý zdroj existuje pouze v jedné kopii
- Proces je ve stavu běžící nebo blokován
- Běžící – proces má všechny zdroje
- Blokováný – čeká na jeden nebo více zdrojů



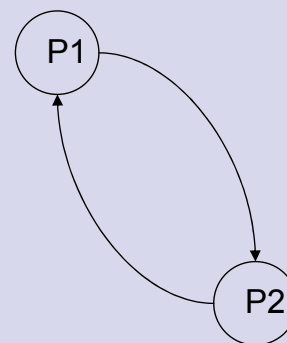
## Grafické znázornění - RAG

- Modelem je graf
- Uzly grafu jsou procesy
- Hrany grafu jsou nevyřízené požadavky procesů nebo přiřazené zdroje
- Graf je proto označován RAG (Resource Allocation Graph)
- $P1 \rightarrow R1$  znamená, že P1 čeká na R1
- Nebo  $R2 \rightarrow P1$  znamená, že R2 je přidělen P1
- $P2 \rightarrow R2 \rightarrow P1 \rightarrow R1 \rightarrow P2$
- Deadlock je orientovaný cyklus v tomto grafu



## Grafické znázornění - WFG

- Modelem může být i WFG (Wait-For-Graph)
- Uzly grafu jsou procesy
- Hrany grafu jsou vazby mezi procesy
- Vyjadřují vzájemné čekání
- $P1 \rightarrow P2$  znamená, že P1 čeká na P2
- $P1 \rightarrow P2 \rightarrow P1$
- Deadlock je orientovaný cyklus v tomto grafu





# Modely deadlocku

- Jednoduchý model
  - Uzel může být pouze v jednom cyklu
  - Přidělení jednoho zdroje
  - Cyklus WFG je postačující podmínkou pro detekci deadlocku
- AND model
  - Tentýž jako předchozí případ
  - proces může současně požadovat více zdrojů
  - zůstává blokován dokud neobdrží všechny požadované zdroje
  - Cyklus WFG je postačující podmínkou pro detekci deadlocku
- OR model
  - proces může najednou požadovat více zdrojů
  - zůstává blokován dokud neobdrží alespoň jeden z požadovaných zdrojů
  - Nalezení uzlu v grafu (knot) je postačující podmínkou pro detekci deadlocku



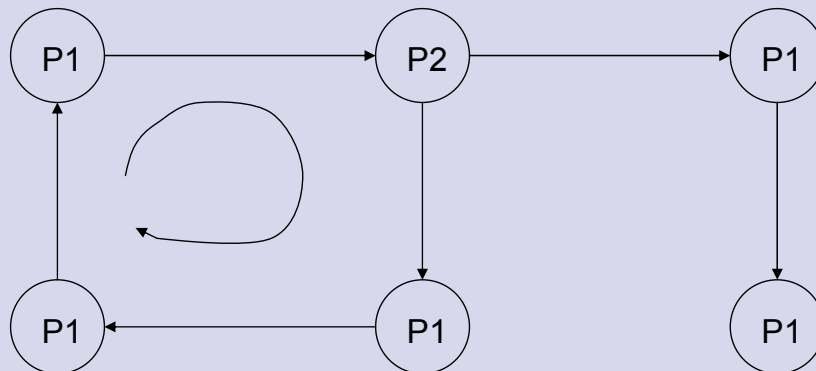
# Modely deadlocku

- AND – OR model
  - Kombinace požadavků na AND model a OR model
- P-out-of-Q model
  - Získání P z Q zdrojů
  - Např. přístup k replikám (stačí přístup k P replikám z Q replik)
  - Speciální případy
    - P = 1 ... OR model
    - P = Q ... AND model
  - Nalezení uzlu v grafu (knot) je postačující podmínkou pro detekci deadlocku



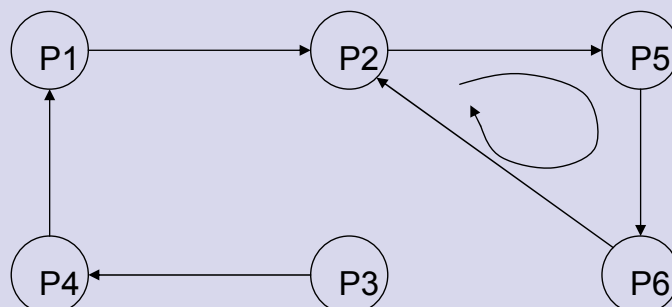
## Příklad AND modelu

- Přítomnost cyklu



## OR model

- Přítomnost uzlu
- Knot (uzel): podmnožina orientovaného grafu taková, že počínajíc z libovolného uzlu podmnožiny je nemožné opustit knot po hranách grafu.



# Strategie zpracování deadlocku



- Akceptování deadlocku
  - ignorování problému a víra, že deadlock nenastane (UNIX)
- Prevence deadlocku – obtížné
  - zabránění vzniku podmínek vedoucích k deadlocku
  - vždy drahé
- Předcházení deadlocku – před alokací zdroje kontrola možného deadlocku
  - obtížné protože bych potřeboval mít informaci o globálním stavu každého uzlu který ovládá zdroje
  - cílem je udržení bezpečných stavů
- Detekce a odstranění deadlocku

# Strategie zpracování deadlocku prevence



- Prevence může být dosaženo zrušením jedné ze čtyř následujících podmínek
  - 1. výlučný přístup ke zdrojům
    - většina prostředků vyžaduje výlučný přístup
    - jedná se např. o paměť, periferie
    - lze obejít virtualizací (virtuální tiskárna, virtuální paměť)
  - 2. postupné přidělování
    - zdroje jsou přidělovány postupně, ne najednou
    - lze řešit ohodnocením zdrojů a zachováním pořadí při jejich přidělování i uvolňování

# Strategie zpracování deadlocku prevence



- Prevence může být dosaženo zrušením jedné ze čtyř následujících podmínek
  - 3. nepreemptivní plánování
    - prostředek může uvolnit pouze proces, který jej vlastní
    - řešení opět ve virtualizaci
  - 4. neomezené čekání
    - předchozí podmínky vedou k neomezenému čekání na přidělení zdrojů
    - odstranění některé z předchozích podmínek vede k prevenci deadlocku

# Prevence deadlocku uspořádání podle časových značek



- Uspořádání požadavků – prevence proti vzniku cyklů
  - Požadavky
    - Globální čas (Lamportův logický čas)
    - Atomické transakce
  - Transakcím jsou přiřazeny časové značky
    - Schéma wait-die (zemřít)
      - Starší čeká na uvolnění zdroje, který vlastní mladší
      - Mladší končí (roll-back) pokud žádá o zdroj, který vlastní starší
    - Schéma wound-wait (postřelit)
      - Starší požaduje zdroj držený mladším, mladší končí (roll-back)
      - Mladší požaduje zdroj držený starším - čeká

# Strategie zpracování deadlocku - předcházení deadlocku



- **Zavádíme jisté a nejisté stavy**
  - ze stavu jistý nemůžeme dosáhnout deadlocku vézt přímo přidělením jednoho zdroje
  - předpokládejme tři procesy a 10 stejných zdrojů
  - A žádá o jeden zdroj, může být přidělen?
  - B žádá o jeden zdroj, může být přidělen?

Thread	Current	Max	Balance
A	2	4	2
B	3	6	3
C	3	8	5

# Strategie zpracování deadlocku detekce a odstranění



- **Detekce deadlocku hledání cyklů**
  - Vytváří se a následně redukuje **Resource Allocation Graph**
  - RAG je bipartitní graf s dvojími uzly
    - Procesy (vlákna) - kroužky
    - Zdroje – obdelníčky
    - Orientované hrany
  - WFG (Wait-for-Graph) – redukovaný RAG
    - Procesy (vlákna)
    - Orientované hrany
  - Obnova po deadlocku představuje ukončení běhu některého z procesů (vláken) a je založena na ad-hoc technikách

# Strategie zpracování deadlocku detekce a odstranění



- Algoritmy detekce deadlocku musí zajistit
  - Nesmí nedetekovat žádný deadlock
  - Musí detekovat všechny deadlocky v konečném čase
- Nesmí detekovat falešné deadlocky (phantom, false)
  - jsou způsobeny dobou zpoždění (latencí) sítě
  - představují principiální problém při budování korektního distribuovaného detekčního algoritmu

## Falešný deadlock



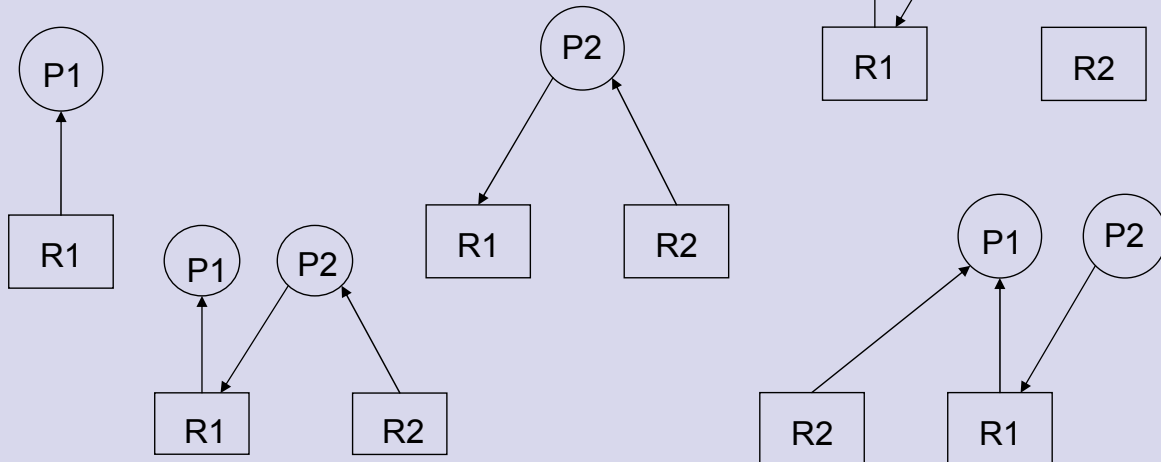
- Phantom (false) deadlock
- Předpokládejme, že procesy manipulují se zdroji pomocí
  - Request – žádost o zdroj
  - Acquire – získat, přidělit zdroj
  - Release – uvolnění zdroje
- Zprávy nemusí přicházet ve správném pořadí
- Příklad:
  - Zpráva A: Release ( P2, R2 )
  - Zpráva B: Request ( P1, R2 )





# Falešný deadlock

- Správné pořadí doručení
  - Zpráva A: Release ( P2, R2 )
  - Zpráva B: Request ( P1, R2 )



27.10.2008

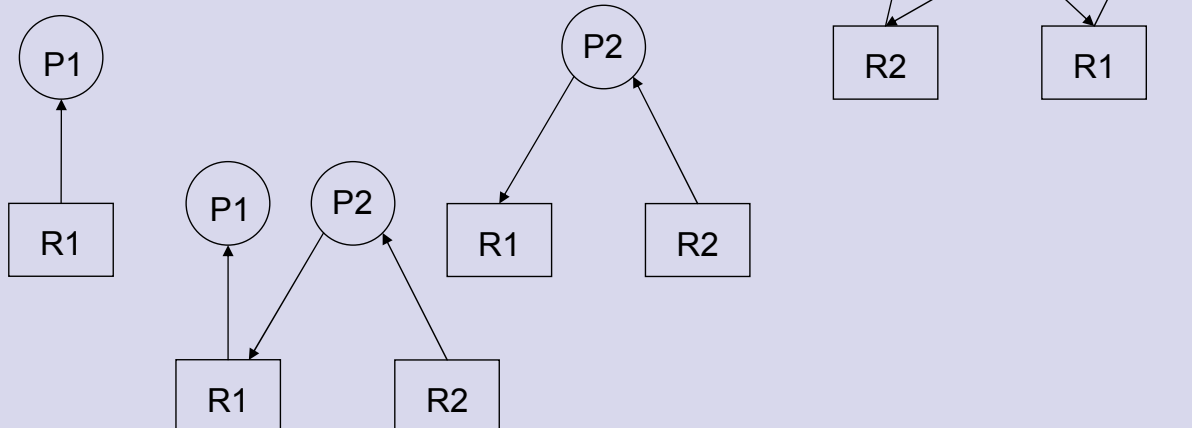
Distribuívaný deadlock

17



# Falešný deadlock

- Chybné pořadí doručení
  - Zpráva B: Request ( P1, R2 )
  - Zpráva A: Release ( P2, R2 )



27.10.2008

Distribuívaný deadlock

18



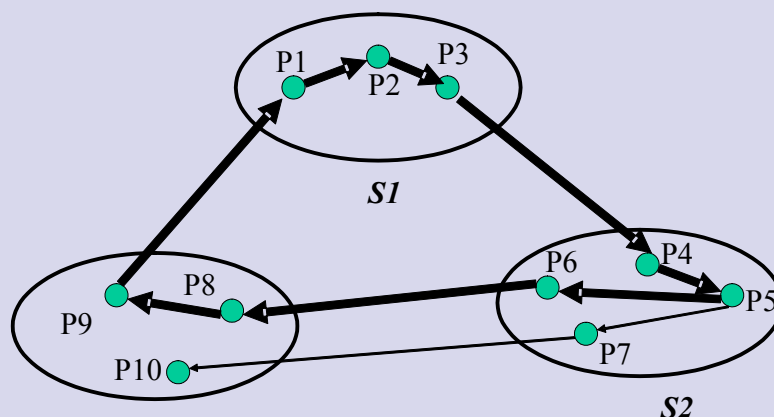
## Detekce deadlocku, cyklus a uzel

- AND modelu požadavků požaduje aby všechny právě požadované zdroje byly přiděleny jako podmínka pro odblokování výpočtu
  - v tomto modelu je detekce cyklu postačující podmínkou pro detekci deadlocku
- OR model požadavků dovoluje vytvářet při výpočtu požadavky na více různých zdrojů a odblokování výpočtu je možné pokud je alespoň jeden uspokojen
  - v tomto modelu je cyklus podmínkou nutnou
  - knot je podmínkou postačující

## Příklad



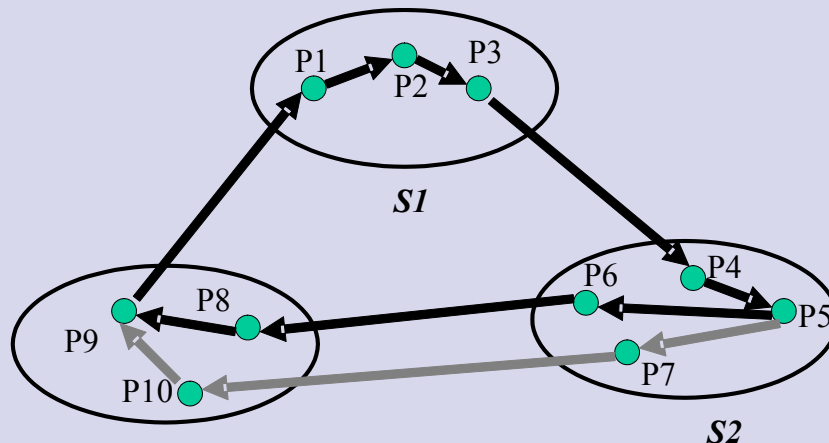
- v AND modelu je to deadlock
- v OR modelu není, protože chybí uzel





## Příklad

- v AND modelu je to deadlock
- v OR modelu také



## Možnosti řešení deadlocku v distribuovaných systémech



- Centralizované řízení
  - řídicí uzel vytváří WFG a kontroluje orientované cykly
  - WFG může být udržován průběžně nebo budován na přání při požadavku FWG jednotlivými stranami
- Distribuované řízení
  - WFG je rozprostřen na různých uzlech. Kterýkoliv uzel může iniciovat proces detekce deadlocku
- Hierarchické řízení
  - uzly jsou uspořádány do hierarchie
  - uzel kontroluje cykly pouze u podřízených



# Centralizované algoritmy

- Centralizované algoritmy
- Jednoduchá koncepce
  - každý uzel posílá zprávu do hlavního detekčního uzlu
  - hlavní detekční uzel vytváří a analyzuje WFG
  - pokud je deadlock detekován, hlavní detekční uzel řídí odstraňování deadlocku
- Závažné problémy
  - výskyt chyby
  - zahlcení sítě
  - detekce falešného deadlocku



# Centralizované algoritmy

- Ho-Ramamoorthy 2-fázový algoritmus (AND i OR)
  - každý uzel si udržuje stavovou tabulku o přidělených (uzamčených) zdrojích a čekajících procesech
  - řídicí uzel se periodicky dotazuje na obsah těchto tabulek ve všech uzlech. Tabulky jednoho uzlu stahuje najednou (atomická operace)
  - řídicí uzel vytváří WFG, analyzuje jej, hledá smyčky a pokud je najde, pak hledá řešení
  - nutnou podmínkou pro detekci deadlocku je nalezení smyčky, pokud najde smyčku, požádá znovu o zaslání tabulek z ostatních uzlů
  - pokud je opět detekován cyklus, může se jednat o deadlock
  - může se jednat i o falešný (false) nebo zdánlivý (phantom) deadlock



# Centralizované algoritmy

- Ho-Ramamoorthy 1-fázový algoritmus
  - každý uzel udržuje dvě tabulky
    - stavovou tabulku zdrojů
    - stavovou tabulku procesů
  - tabulka zdrojů
    - transakce mohou být uzamčeny nebo čekají na zdroje
  - tabulka procesů
    - zdroje uzamčené transakcemi nebo čekající v transakcích
  - řídicí uzel periodicky soustřeďuje tyto tabulky ze všech uzlů
  - z transakcí vytváří WFG společný pro obě tabulky
  - nalezení cyklu znamená deadlock



# Distribované algoritmy

- Každý uzel má stejné možnosti detekovat deadlock
  - WFG se stává abstrakcí, kde každý uzel obsahuje svou část WFG
  - v zásadě detekce je vyvolána stranou, kde nějaké vlákno čeká příliš dlouho ve frontě na zdroj
- Existují 4 modely které mohou být použity k realizaci algoritmu pro detekci deadlocku
  - **Path-pushing**: informace o cestě v grafu (vztah čekajícího procesu a přiděleného zdroje) je posílána z čekajícího uzlu do blokujícího uzlu (Obermarck)
  - **Edge-chasing**: hranami WFG jsou posílány speciální zprávy (probe – sondy). Jestliže je sondovací zpráva přijata iniciátorem, je detekován deadlock (Chandy-Misra-Haas).
  - **Global state detection**: získává snímek (snapshot) o distribuovaném systému, konstruuje a redukuje WFG (zametá hrany)
  - **Diffusion computation**: WFG jsou posílány echo zprávy, obsahující dotaz na stav jednotlivých uzlů

# Path-pushing (propagace nalezených cest)



- Obermarckův algoritmus (AND model)
  - založen na databázovém modelu, který používá transakční zpracování
  - strana, která detekuje ve svém částečném WFG cyklus, přenesou objevenou cestu členům úplně uspořádané transakce.
  - Transakce s nejvyšší prioritou detekuje deadlock (Ex => T1 => T2 => Ex”)
  - algoritmus může detekovat i falešné deadlocky protože snímek zjišťuje asynchronně

# Edge-chasing algorithm (odstranění hran)



- Chandy-Misra-Haas algoritmus (AND model)
  - proces  $P_i$  detekuje deadlock posláním zprávy (probe)  $P(i, j, k)$ . Tato zpráva je posílána procesem  $P_j$  procesu  $P_k$ .
  - Tato zpráva putuje po hranách grafu mezi stranami tak dlouho, pokud se nezruší, nebo dokud nedoputuje do  $P_i$ . V tom případě implikuje detekci deadlocku.
  - Tedy:
    - $P_j$  závisí na  $P_k$ , jestliže existuje posloupnost  $P_j, P_{i_1}, \dots, P_{i_m}, P_k$
    - $P_j$  je lokálně závislé na  $P_k$ , jestliže předchází podmínka a  $P_j, P_k$  jsou na téže straně.
    - Každý proces si udržuje vektor závislostí:  $D_i(j)$  je true  $P_i$  ví, že  $P_j$  je na něm závislé. (inicializováno na false pro  $i$  a  $j$ ).

# Edge-chasing algorithm

## Algoritmus Chandy-Misra-Hass



- Odeslání požadavku (testovací zprávy)
  - jestliže  $P_i$  je lokálně závislý na sobě, pak nastal deadlock.
  - Jestliže pro všechna  $P_j$  a  $P_k$  taková že
    - $P_i$  je lokálně závislé na  $P_j$
    - $P_j$  čeká na  $P_k$  a
    - $P_j$  a  $P_k$  jsou na různých uzlech, pak pošli  $P(i,j,k)$  do uzlu s  $P_k$ .

# Edge-chasing algorithm

## Algoritmus Chandy-Misra-Hass



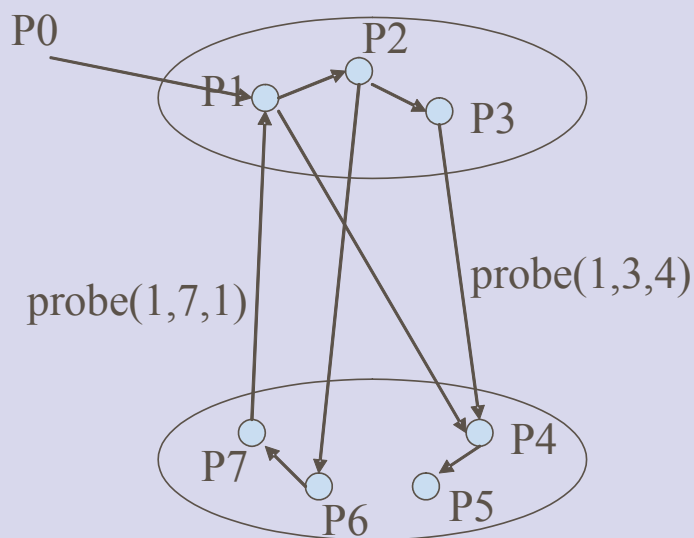
- Příjem požadavku (testovací zprávy)
  - jestliže je
    - $P_k$  blokováno a
    - $D_k(i)$  je false a
    - $P_k$  neodpověděl na všechny požadavky  $P_j$ ,
  - pak
    - $D_k(i) := \text{true}$ ;
    - jestliže  $(k == i)$  pak  $P_i$  je uvíznuté
    - jinak pro všechna  $P_m$  a  $P_n$  taková že
      - $P_k$  je lokálně závislé na  $P_m$ , a
      - $P_m$  čeká na  $P_n$ , a
      - $P_m$  a  $P_n$  jsou na různých uzlech, pošli  $P(i,m,n)$  do uzlu s  $P_n$

# Edge-chasing algorithm

## Algoritmus Chandy-Misra-Hass



- Výkonnost
  - pro detekci deadlocku, který zahrnuje  $m$  procesů nad  $n$  stranami (uzly) je třeba  $m(n-1)/2$  zpráv  $P$
  - délka zprávy je 3 slova (id)
  - zpoždění při detekci deadlocku je  $O(n)$ .



# Edge-chasing algorithm

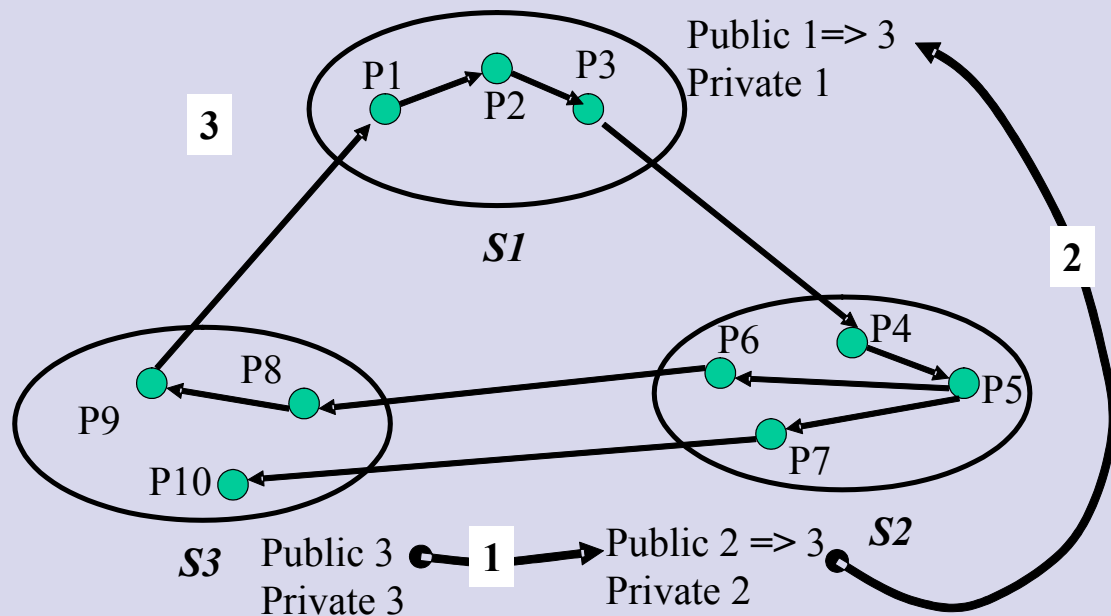
## Mitchell-Meritt Algorithm (pro AND model)



- Propaguje zprávy v opačném směru
- Používá značení veřejné/privátní zprávy
- Zprávy mohou v kterémkoliv uzlu svá změnit návěští at each site
- Pokud se objeví v uzlu zpráva se souhlasným veřejným návěštím, je detekován deadlock (pouze v uzlu s největším veřejným návěštím v cyklu)



## Edge-chasing algorithm Mitchell-Meritt Algorithm (pro AND model)



27.10.2008

Distribovaný deadlock

33

## Algoritmy založené na difuzním zpracování



- Výpočet detekce deadlocku difunduje skrz systémový WFG
  - zprávy jsou posílány z výpočetního procesu uzlu a difundují hranami WFG do ostatních uzlů
  - jestliže dotaz dosáhne aktivní neblokovaný výpočet, je zahozena
  - jestliže dotaz dosáhne blokový výpočet, pošle se echo zpět iniciátorovi
  - pokud všichni pošlou zpět echo, nastal deadlock

27.10.2008

Distribovaný deadlock

34

# Difuzní zpracování Chandy-Misra-Hass



- Čekající výpočet na uzlu  $x$  posílá periodicky všem výpočtům, které také čekají (závislá množina), zprávu s označením ID iniciátoru a ID cíle
- Každý z těchto výpočtů se opakovaně dotazuje závislé podmnožiny členů (pouze pokud jsou také sami blokovány) označujíc každý dotaz ID iniciátoru, vlastním ID a a novým cílovým ID na který čeká.
- Výpočet neposílá echo dokud nedostane odpovědi od všech členů závislé podmnožiny, kterým poslal výzvu (požadavek)
- Jestliže dostane odpověď ode všech, pošle předchůdci odpověď s ID iniciátoru, vlastním ID a ID nejvzdálenějšího závislého uzlu.
- Jestliže iniciátor obdrží odpovědi na echo ode všech členů závislé podmnožiny, rozpozná deadlock pokud ID iniciátoru a ID nejvzdálenějšího uzlu jsou stejná.

# Difuzní zpracování Chandy-Misra-Hass



- Inicializace blokováním procesem  $P_i$ :
  - pošle  $query(i,i,j)$  všem procesům  $P_j$  v závislé množině  $DS_i$  z  $P_i$ ;
  - $num(i) := |DS_i|$ ;  $wait(i) := true$ ;
- Blokováný proces  $P_k$  přijme  $query(i,j,k)$ :
  - jestliže jde o prvotní dotaz od  $P_k$ 
    - posílá  $query(i,k,m)$  všem  $P_m$  v  $DS_k$ ;
    - $num_k(i) := |DS_k|$ ;  $wait_k(i) := true$ ;
  - jinak jestliže  $wait_k(i) == true$  pak pošli  $reply(i,k,j)$  do  $P_j$ .
- Proces  $P_k$  přijme  $reply(i,j,k)$ 
  - jestliže  $wait_k(i)$  pak
    - $num_k(i) := num_k(i) - 1$ ;
  - jestliže  $num_k(i) = 0$  pak
    - jestliže  $i = k$  pak nastal deadlock
  - jinak pošli  $reply(i, k, m)$  do  $P_m$ , který poslal prvotní dotaz.



## Detekce globálního stavu

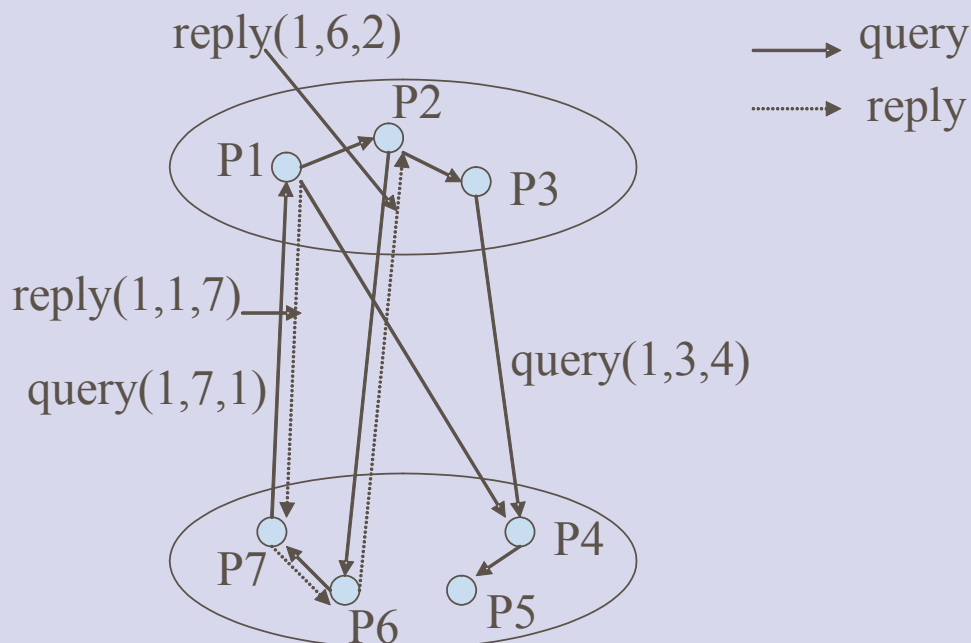
- Založena na dvou skutečnostech z distribuovaných systémů
- Konzistentní snímek můžeme dostat bez umrtvení probíhajícího výpočtu
- Konzistentní snímek nemusí reprezentovat stav systému v libovolném čase, ale jestliže zastavíme systém před inicializací shromáždění snímku, bude stav zahrnut do snímku.



## Detekce globálního stavu Kshemkalyani-Singhal algorithm

- Iniciátor výpočtu snímku systému vyšle FLOOD zprávu všemi odcházejícími hranami
- Výpočty přijímající zprávu FLOOD buď vrátí zprávu ECHO (pokud na ní není závislá nebo propaguje zprávu FLOOD svým závislým
- ECHO zprávy jsou analogické přenosu požadavků přes hrany v resource allocation graph (RAG)
  - Když je ECHO objeven jako odpověď na FLOOD oblast ve WFG kde je iniciátor se začne redukovat
  - Jestliže závislí nevrátí ECHO jako ukončení, takové uzly reprezentují část kde je deadlock
  - Ukončení je dosaženo sečtením vah ECHO a SHORT zpráv (vracející počáteční FLOOD váhy)

# Detekce globálního stavu Kshemkalyani-Singhal algorithm



27.10.2008

Distribovaný deadlock

39

## Hierarchická detekce deadlocku



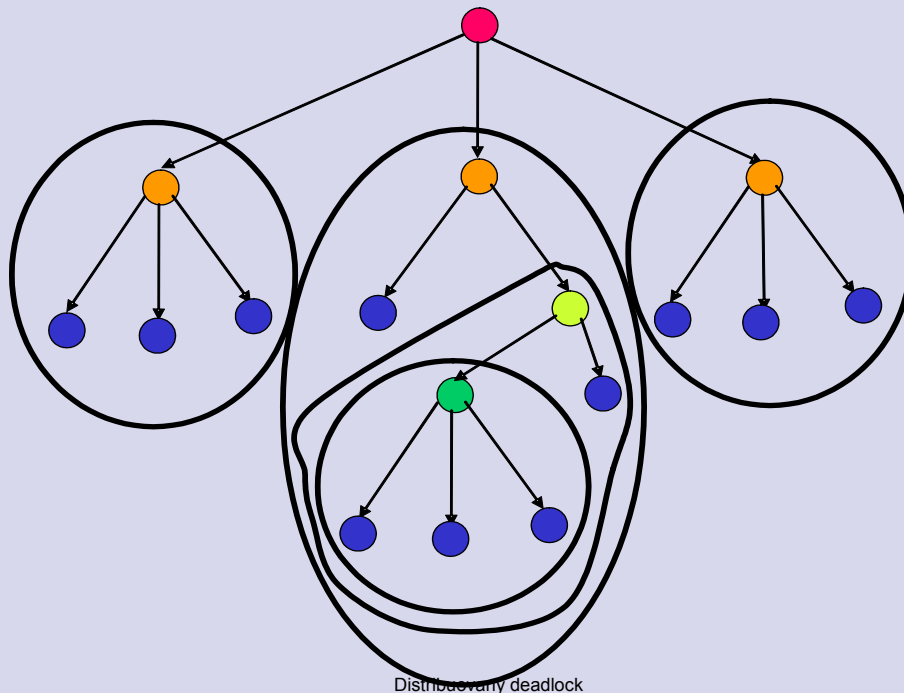
- Tyto algoritmy reprezentují kompromis mezi zcela centralizovaným řešením a decentralizovaným řešením
- Řídicí uzel je informován z podřízených uzlů periodicky (centralizovaný algoritmus), řídicí uzly jsou uspořádány do hierarchie
- Nadřazený řídicí uzel (kořen stromu), vnitřní uzly slouží pro řízení větví.

27.10.2008

Distribovaný deadlock

40

# Hierarchická detekce deadlocku



27.10.2008

Distribuovaný deadlock

41

# Hierarchická detekce deadlocku Menasce-Muntz Algorithm



- Listové kontrolery přidělují zdroje
- Hranové kontrolery jsou schopné k hledání deadlocku ve zdrojích rozprostřených v podstromu
- Může být ovládáno zahlcení sítě
- Chyby uzlů jsou méně kritické než u centralizovaného řešení
- Detekce může být prováděna několika cestami
  - Kontinuální informování o alokacích
  - Periodické informování o alokacích

27.10.2008

Distribuovaný deadlock

42

# Hierarchická detekce deadlocku Menasce-Muntz Algorithm



- Používá pouze 2 úrovně
  - hlavní řídicí uzel
  - řídicí uzly clusteru
- Řídicí uzly clusteru jsou schopné detekovat deadlock svých členů a oznamují závislosti mimo cluster hlavnímu řídicímu uzlu.
- Používá jednofázový Ho-Ramamoorthy algoritmus pro centralizovanou detekci
- Hlavní řídicí uzel je schopen detekovat deadlock uvnitř clusteru
- Přiřazování uzlů do clusteru je dynamické

# Hierarchická detekce deadlocku Menasce-Muntz Algorithm

