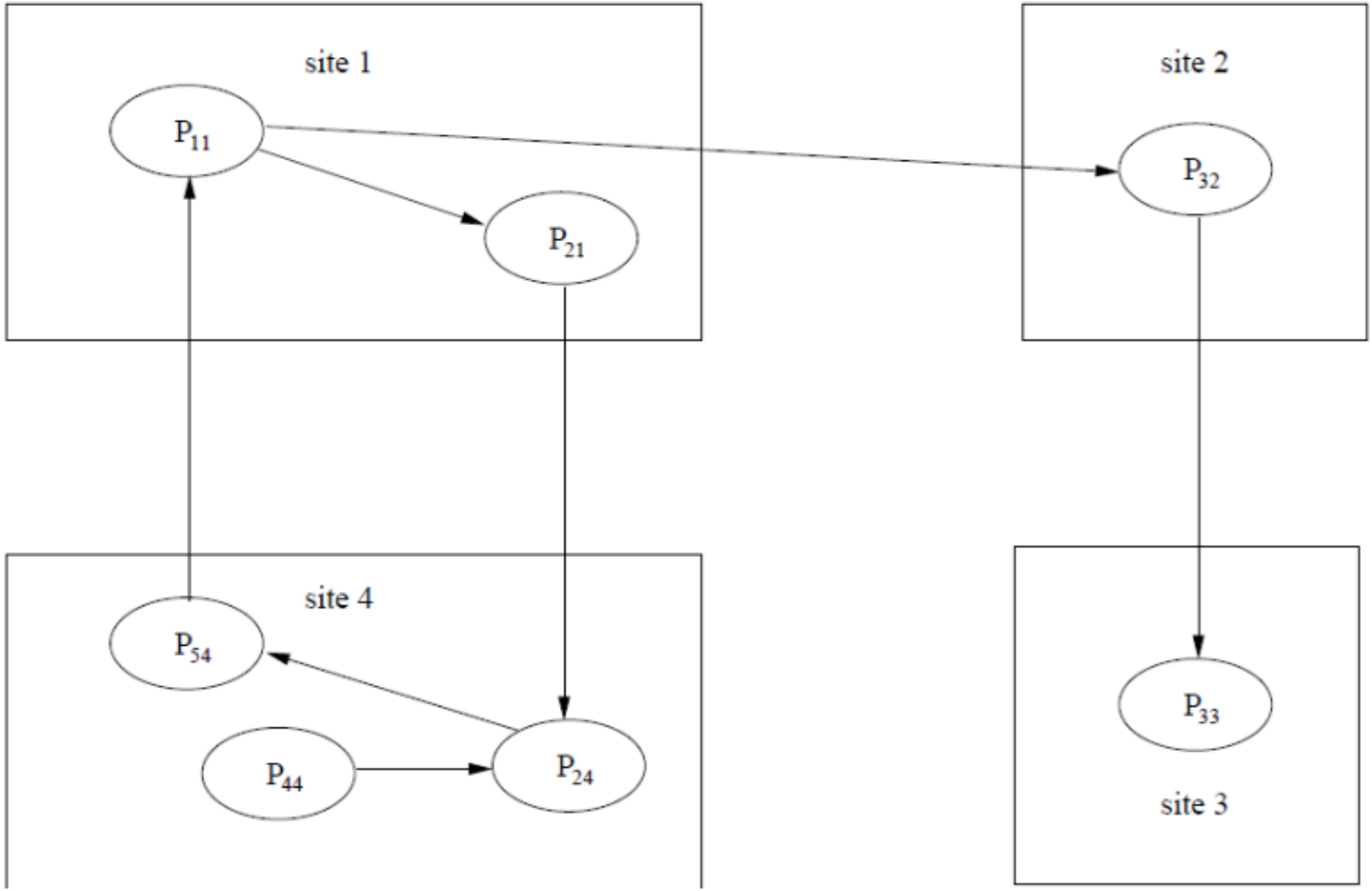
Detekce uváznutí v DS

PA 150 ◇ Principy operačních systémů

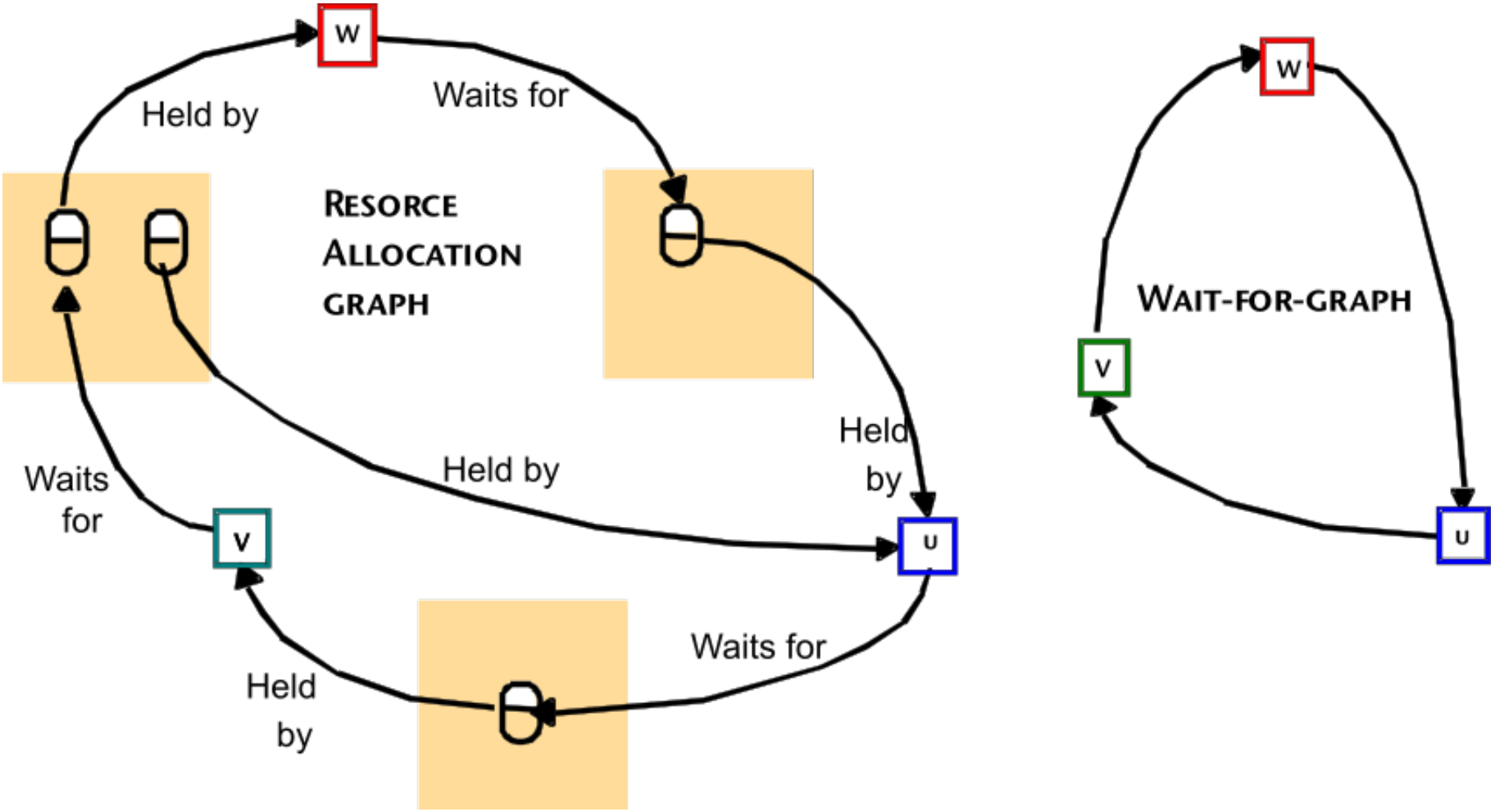
Jan Staudek
http://www.fi.muni.cz/usr/staudek/vyuka/

Verze : podzim 2020

Wait-for-Graph (mezi procesy běžícími v uzlech DS)



Příklad uváznutí v DS



Poznámky k výkladu uváznutí

- Procesy v DS mohou žádat o zdroje v libovolném pořadí
  - ✓ Pořadí není známé a priori
- Procesy mohou žádat o zdroje i když již mají přidělené jiné zdroje
- Předpoklady
  - ✓ zdroje jsou opakovaně přístupné
  - ✓ procesy požadují exkluzivní přístup ke zdroji
  - ✓ zdroje jsou v jediných exemplářích
- Stavy procesů
  - ✓ aktivní – má přidělené zdroje, které požadoval a je buď běžící nebo připravený k běhu
  - ✓ blokovaný – čeká na přidělení zdroje

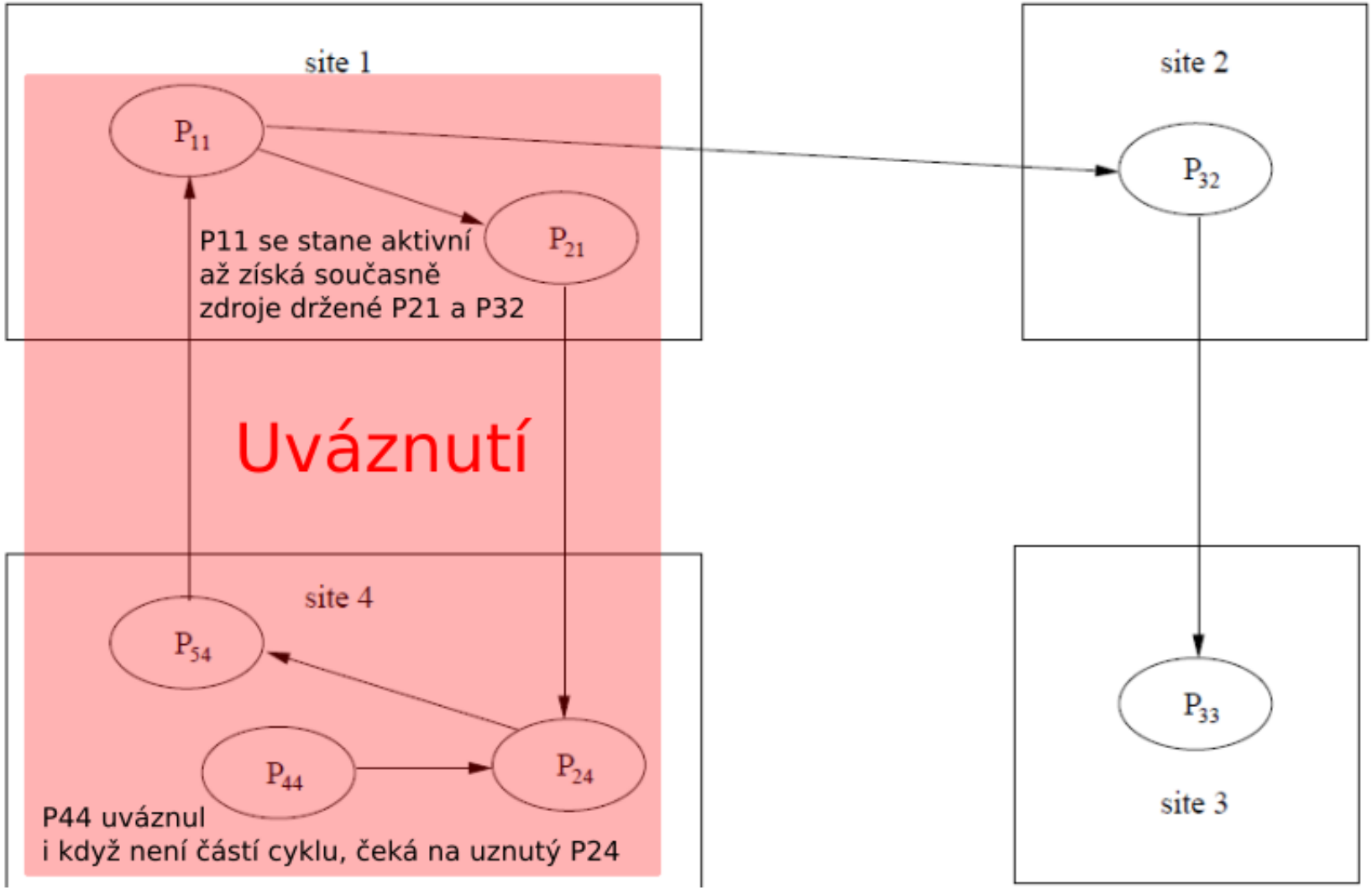
Principy řešení uváznutí

- Řazením použitelnosti zdrojů – prevence uváznutí
  - ✓ definuje se globální uspořádání všech systémových zdrojů
  - ✓ každý zdroj obdrží jedinečné pořadové číslo
  - ✓ proces smí požadovat zdroj s číslem i
    pouze když nevlastní zdroj s číslem větším než i
  - ✓ snadná implementovatelnost, možná neefektivita používání zdrojů
- Bankéřův algoritmus – obcházení uváznutí
  - ✓ jeden z procesů musí hrát roli bankéře – správce prostředků
  - ✓ v distribuovaném prostředí nesnadno implementovatelný, větší režie
- Detekce uváznutí
  - ✓ zkoumání stavu interakcí mezi procesy a zdroji,
    vyhledávání cyklického čekání
  - ✓ efektivní přístup ke zvládání uváznutí v distribuovaném prostředí

Modely uváznutí

- Jednoduchý model
  - ✓ Proces může požadovat přidělení pouze jednoho zdroje
  - ✓ Proces být pouze v jednom cyklu
  - ✓ Cyklus ve WFG je postačující podmínkou pro detekci uváznutí
- AND model
  - ✓ Proces může požadovat současné přidělení více zdrojů
  - ✓ Požadavek přidělení je splněný přidělením všech zdrojů současně
  - ✓ Z uzlu WFG může vystupovat více hran
  - ✓ Cyklus ve WFG je postačující podmínkou pro detekci uváznutí
  - ✓ AND model je obecnější model než jednoduchý model

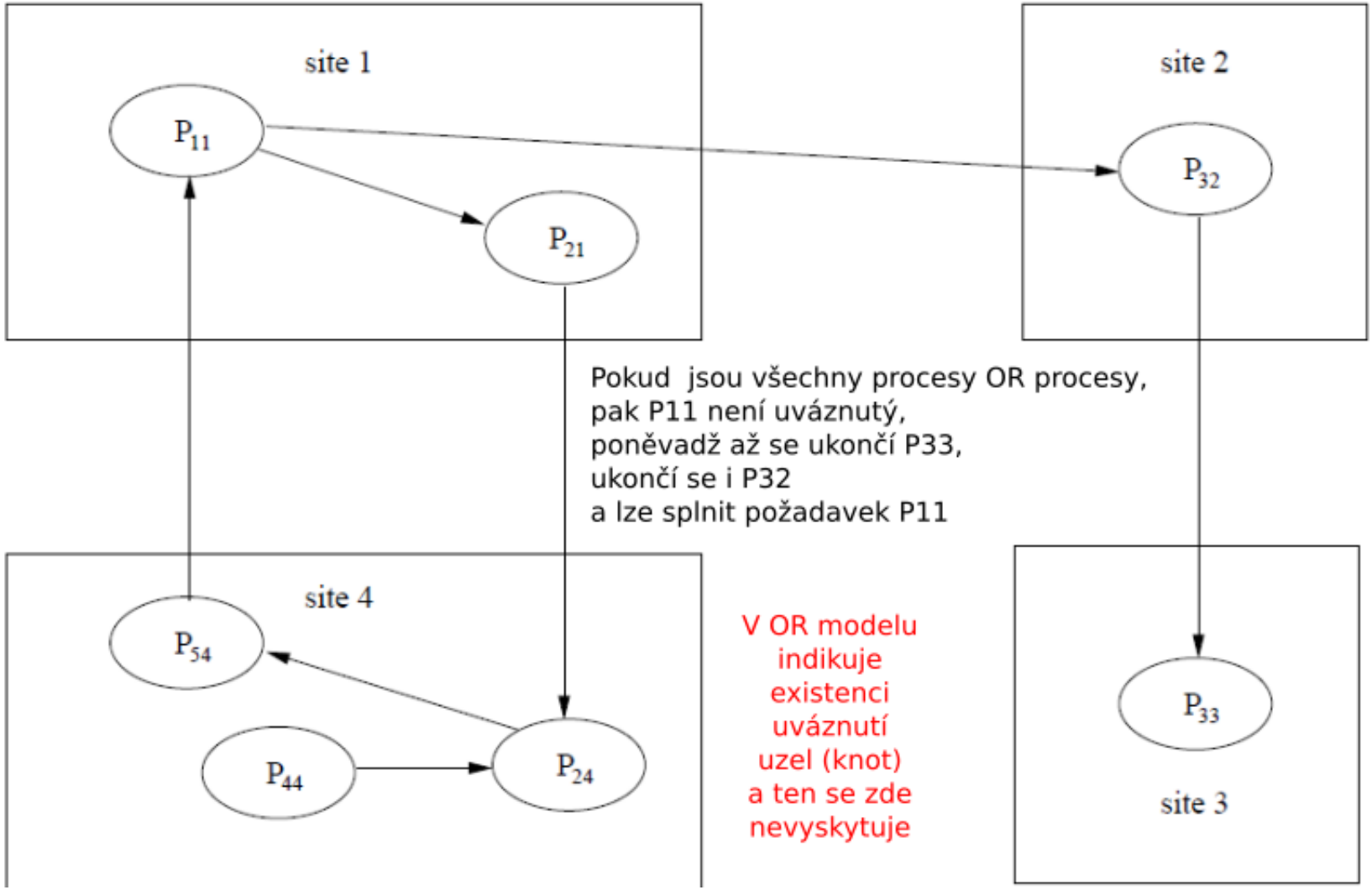
AND model


site 1
P_{11}
P_{21}
P11 se stane aktivní
až získá současně
zdroje držené P21 a P32
site 2
P_{32}
Uváznutí
site 4
P_{54}
P_{44}
P_{24}
P44 uváznul
i když není částí cyklu, čeká na uznutý P24
P_{33}
site 3

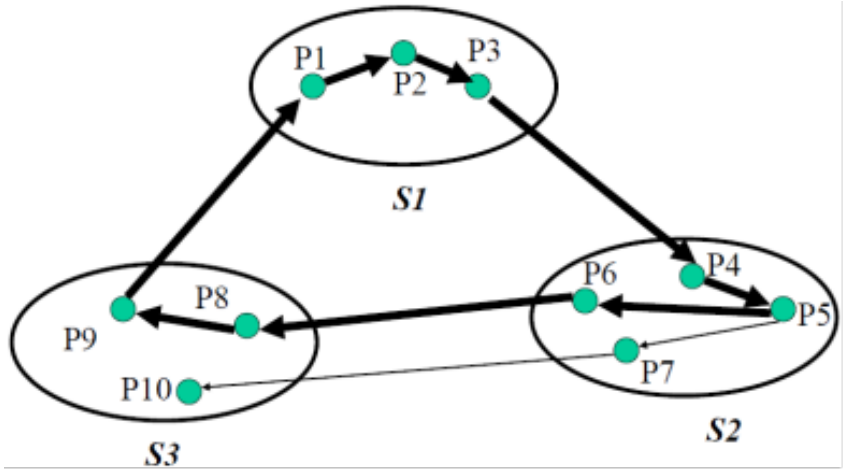
Modely uváznutí

- OR Model
  - ✓ Proces může najednou požadovat více zdrojů
  - ✓ Proces zůstává blokován dokud neobdrží alespoň jeden z požadovaných zdrojů
  - ✓ Cyklus ve WFG je nutnou podmínkou pro uváznutí
  - ✓ Uzel (knot) v grafu je postačující podmínkou pro detekci uváznutí
  - ✓ Knot (uzel): podmnožina orientovaného grafu taková, že počínajíc z libovolného uzlu podmnožiny nelze opustit knot po hranách grafu.

OR model

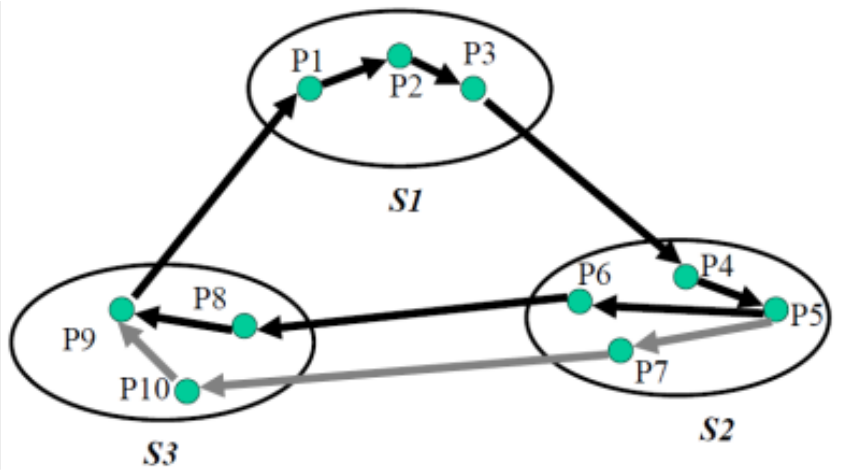

site 1
site 2
site 3
site 4
P_{11}
P_{21}
P_{32}
P_{54}
P_{44}
P_{24}
P_{33}
Pokud jsou všechny procesy OR procesy,
pak P11 není uváznutý,
poněvadž až se ukončí P33,
ukončí se i P32
a lze splnit požadavek P11
V OR modelu
indikuje
existenci
uváznutí
uzel (knot)
a ten se zde
nevyskytuje

Cyklus vs. uzel



Je zde cyklus nikoli uzel (knot)

Uváznutí v AND modelu, bez uváznutí v OR modelu



Jsou zde cykly i uzel (knot)

Uváznutí v AND mdelu i v OR modelu

Modely uváznutí

- AND–OR model
  - ✓ Kombinace požadavků na AND model a OR modelů
  - ✓ Např. *přiděl (y and (x or z)*
  - ✓ Existence uváznutí se testuje opakovaným testem uváznutí v OR modelu
- P-out-of-Q model, generalizovaný 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
  - ✓ Uzel (knot) v grafu je postačující podmínkou pro detekci uváznutí

Správa uváznutí pomocí detekce

- Spravá prevencí a obcházením je v DS neefektivní / nepraktické řešení
- Správa detekcí je vhodná pro DS
  - ✓ Uváznutí se musí
    a) odhalit (detekovat) a
    b) poté vyřešit
  - ✓ Pro odhalení uváznutí musíme umět
    a) udržovat WFG a
    b) hledat ve WFG cykly (uzly)
  - ✓ pro vyřešení se musí definovat aplikačně orientovaná politika

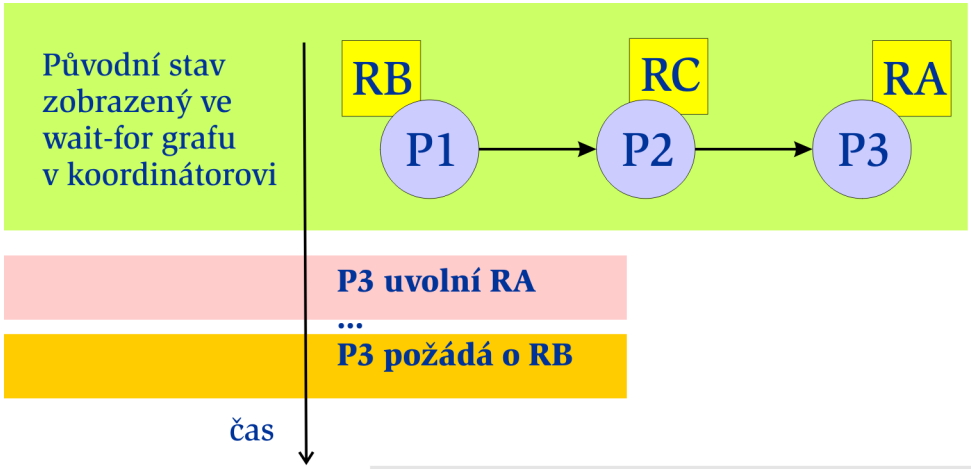
Detekce uváznutí, kritéria správnosti detekčního algoritmu

- Živost, *progress*, nezůstne žádné nedetekované uváznutí
  - ✓ všechna existující uváznutí musí algoritmus detekovat v konečném čase
  - ✓ jakmile se uváznutí vyskytne, spuštěný algoritmus nesmí čekat na žádnou další událost aby uváznutí detekoval
- Bezpečnost, *safety*, nedetekují se falešná uváznutí
  - ✓ algoritmus nesmí oznamovat neexistující, falešná uváznutí
  - ✓ tj. uváznutí detekované na základě konstrukce nekonzistentního WFG vytvořeného díky asynchronní komunikaci a neexist. společné paměti
- Řešení detekovaného uváznutí
  - ✓ jeden nebo více z uvázlých procesů se zruší (vrátí na počátek) a jejich zdroje se přidělí blokovaným procesů, které pak mohou dále běžet

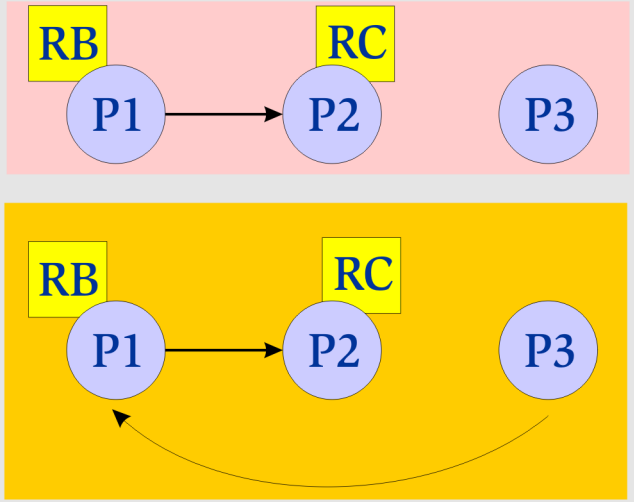
Metody detekce uváznutí

- Centralizované řízení
  - ✓ řídicí uzel v DS vytváří WFG a hledá v něm orientované cykly
  - ✓ WFG lze udržovat průběžně nebo budovat na žádost
  - ✓ řídicí uzel v DS řeší detekované uváznutí
  - ✓ negativa:
    možnost výpadku centra, zahlcení centra, detekce falešných uváznutí
- Distribuované řízení
  - ✓ WFG je rozprostřen po částech v různých uzlech DS
  - ✓ Kterýkoliv uzel DS může iniciovat proces detekce uváznutí
- Hierarchické řízení
  - ✓ uzly DS jsou uspořádány do hierarchie (strom)
  - ✓ uzly DS kontroluje cykly pouze u podřízených uzlů DS

Falešné cykly

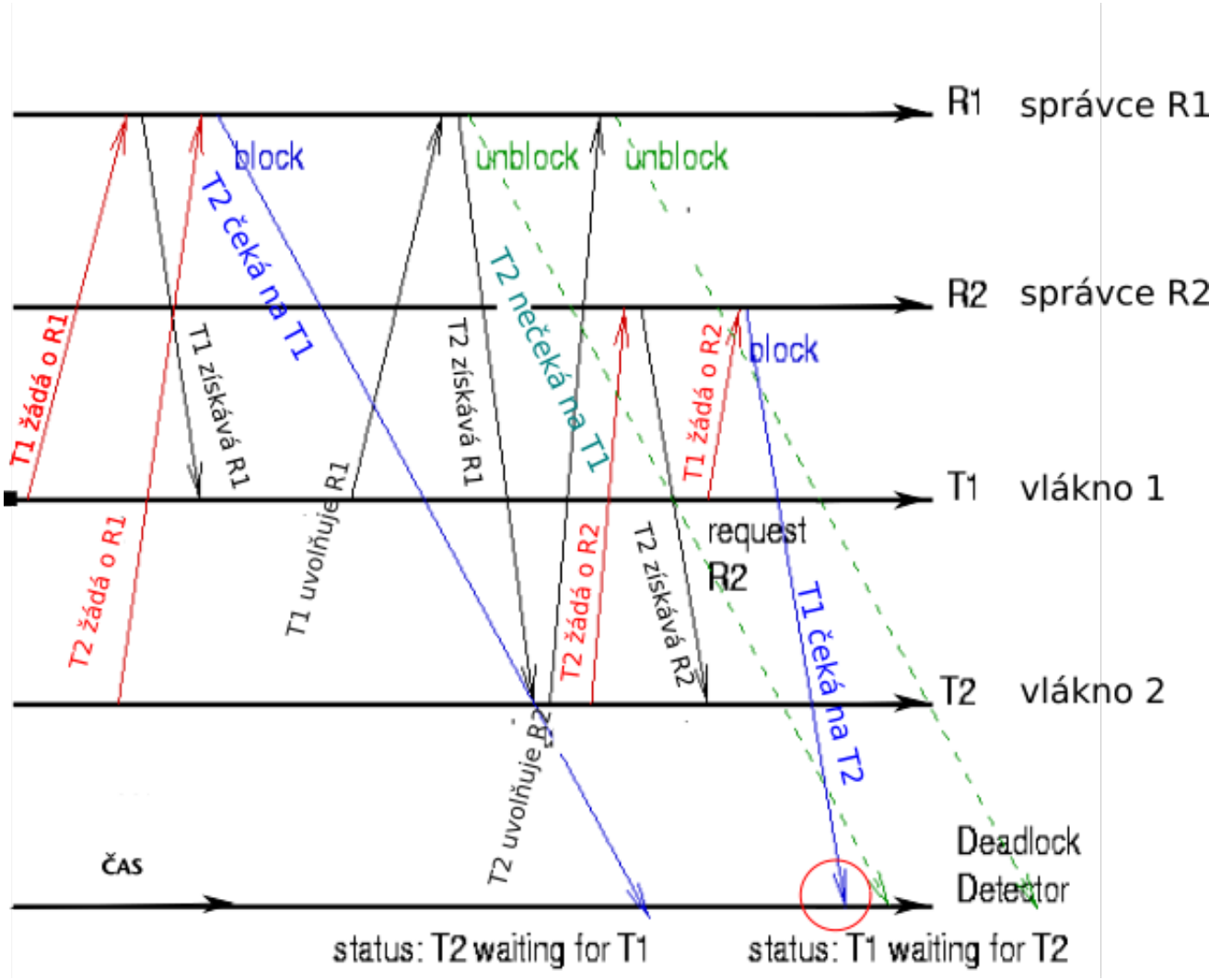


Stav zobrazený ve wait-for grafu v koordinátorovi když zpráva o přidělení dojde dříve než zpráva o uvolnění



Stav postupně zobrazovaný ve wait-for grafu v koordinátorovi když zpráva o uvolnění dojde dříve než zpráva o přidělení

Falešné cykly



Centralizovaný Ho-Ramamoorthy 2-fázový algoritmus

- pro AND i OR model
- každý uzel DS si udržuje stavovou tabulku o lokálních procesech
  (který proces na koho čeká, tj. de facto lokální WFG)
- řídicí uzel DS se periodicky dotazuje na obsah těchto tabulek ve všech uzlech
- řídicí uzel DS vytváří globální WFG, analyzuje jej, hledá v něm cykly a pokud je najde, pak hledá řešení uváznutí

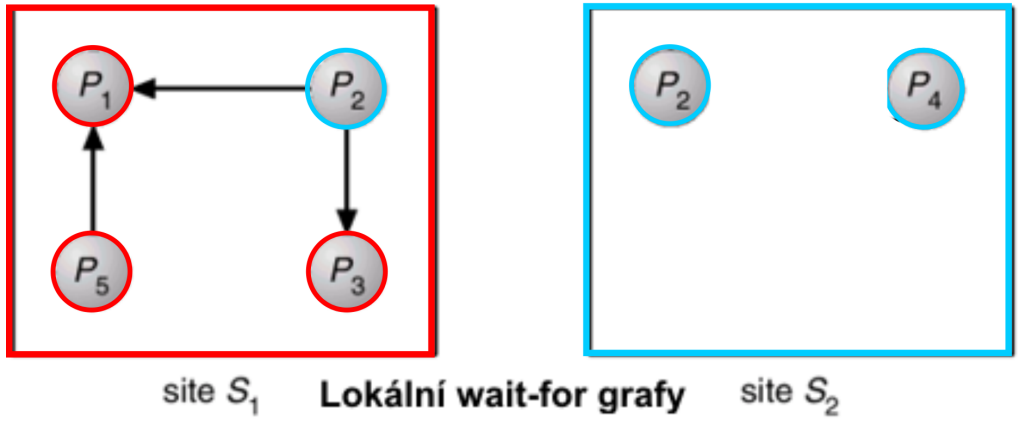
Centralizovaný Ho-Ramamoorthy 2-fázový algoritmus

- nutnou podmínkou pro detekci uváznutí je nalezení cyklu,
  pokud řídicí uzel DS najde cyklus v globálním WFG,
  požádá znovu o zaslání tabulek z participujících uzlů DS
- pokud je opět detekován cyklus,
  může se jednat o uváznutí
- může se ale jednat i o falešný cyklus,
  zdánlivé (*phantom*) uváznutí

Detekce uváznutí pomocí ,,wait-for"grafů, WFG

- Předpoklad a důsledek
  - ✓ každý alokovatelný zdroj ex. v jediném exempláři
  - ✓ cyklus ve ,,wait-for"grafu reprezentuje uváznutí
- Lokální ,,wait-for"graf, platný pro odpovídající uzel sítě
  - ✓ uzly v lok. WFG odpovídají jako lokálním tak i nelokálním procesům, pokud tyto procesy drží nebo požadují zdroje lokální v daném uzlu sítě
- Globální ,,wait-for"graf, platný pro celou síť
  - ✓ sjednocení lokálních lokálních ,,wait-for"grafů
- Cyklus v lokálním ,,wait-for"grafu $\Rightarrow$ existuje uváznutí
- Acykličnost lokálního ,,wait-for"grafu ještě neznamená neexistenci uváznutí
  - ✓ existenci uváznutí indikuje až globální ,,wait-for"graf

Detekce uváznutí pomocí ,,wait-for"grafů, WFG



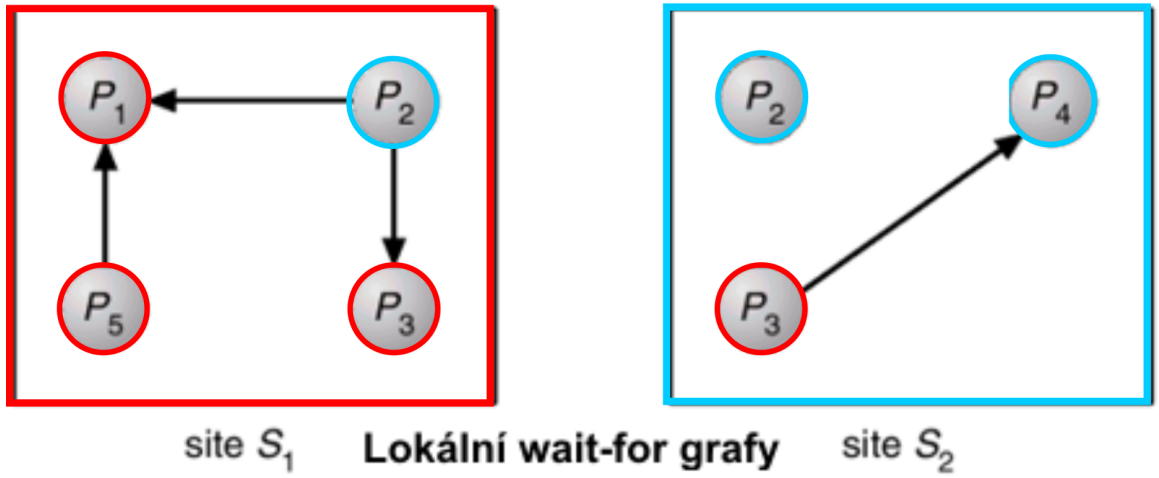
P1 běžící v S1 má přidělený zdroj v S1
P3 běžící v S1 má přidělený zdroj v S1
P5 běžící v S1 čeká na uvolnění zdroje v S1 drženého P1

P2 běžící v S2 má přidělený zdroj v S2 a
čeká na uvolnění zdrojů v S1 držených P1 a P3
P4 běžící v S2 má přidělený zdroj v S2

Lokální wait-for grafy neobsahují cyklus

Jestliže P_i běžící v S_2 žádá zdroj držený P_j běžícím v S_1, pošle P_i zprávu do S_1 a v lok. grafu v S_1 se zapíše hrana $P_i \rightarrow P_j$

Detekce uváznutí pomocí ,,wait-for"grafů, WFG



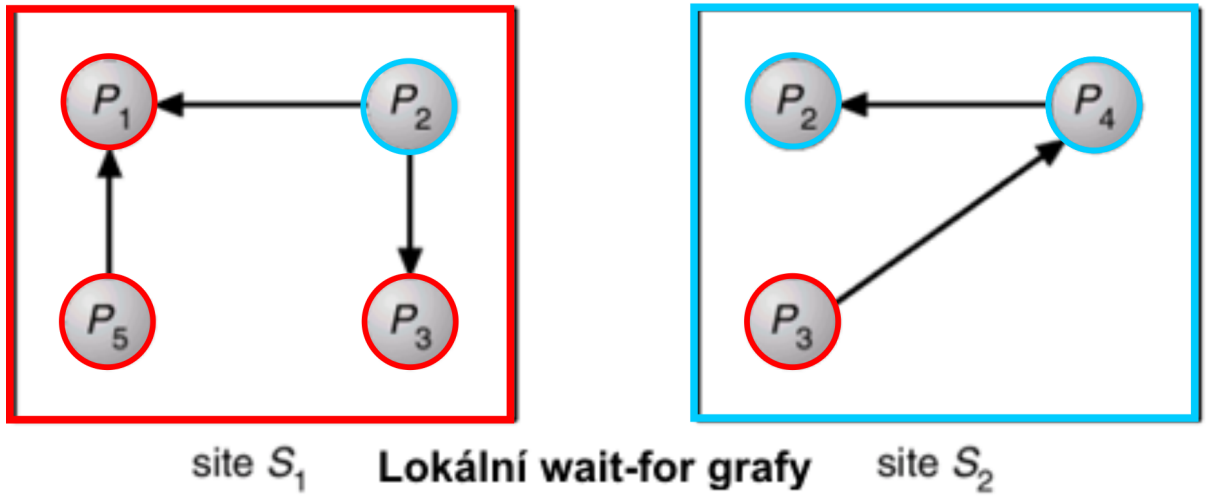
P1 běžící v S1 má přidělený zdroj v S1
P3 běžící v S1 má přidělený zdroj v S1 a
čeká na uvolnění zdroje v S2 drženého P4
P5 běžící v S1 čeká na uvolnění zdroje v S1 drženého P1

P2 běžící v S2 má přidělený zdroj v S2 a
čeká na uvolnění zdrojů v S1 držených P1 a P3
P4 běžící v S2 má přidělený zdroj v S2

Lokální wait-for grafy neobsahují cyklus

Jestliže P_i běžící v S_2 žádá zdroj držený P_j běžícím v S_1, pošle P_i zprávu do S_1 a v lok. grafu v S_1 se zapíše hrana $P_i \rightarrow P_j$

Detekce uváznutí pomocí ,,wait-for"grafů, WFG



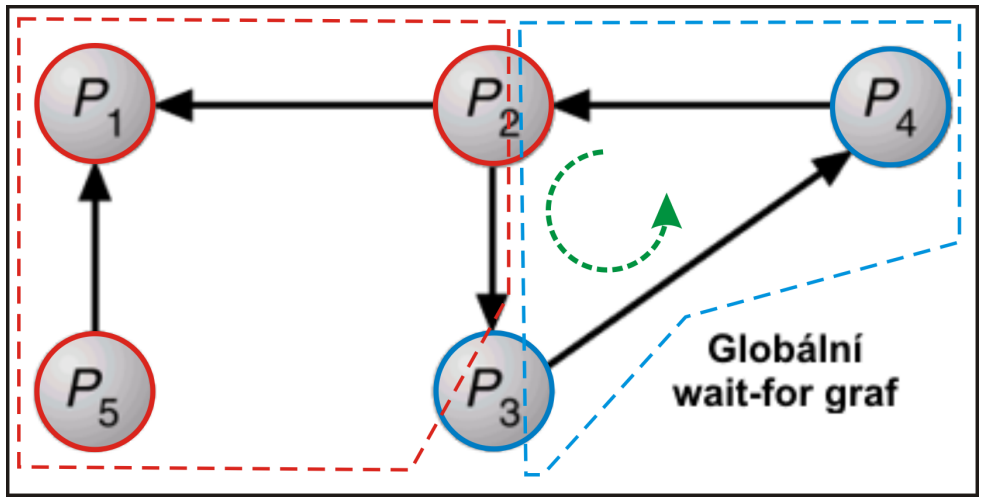
P1 běžící v S1 má přidělený zdroj v S1
P3 běžící v S1 má přidělený zdroj v S1 a
čeká na uvolnění zdroje v S2 drženého P4
P5 běžící v S1 čeká na uvolnění zdroje v S1 drženého P1
P2 běžící v S2 má přidělený zdroj v S2 a
čeká na uvolnění zdrojů v S1 držených P1 a P3
P4 běžící v S2 má přidělený zdroj v S2 a
čeká na uvolnění zdroje v S2 drženého P2

Lokální wait-for grafy neobsahují cyklus

Jestliže P_i běžící v S_2 žádá zdroj držený P_j běžícím v S_1, pošle P_i zprávu do S_1 a v lok. grafu v S_1 se zapíše hrana $P_i \rightarrow P_j$

Detekce uváznutí pomocí ,,wait-for"grafů, WFG

- Centrální uzel si vyžádá stav jednotlivých uzlů:



Globální wait-for graf obsahuje cyklus, systém je v uváznutí:
P_2 čeká na P_3,
P_3 čeká na P_4,
P_4 čeká na P_2

Distribuované algoritmy

- Každý uzel má stejné možnosti detekovat uváznutí
  - ✓ WFG je abstrakcí, kde každý uzel obsahuje svou část WFG
  - ✓ Obecně je detekce vyvolána stranou, kde nějaké vlákno čeká příliš dlouho ve frontě na zdroj
- 4 modely
  - ✓ Path-pushing: informace o cestě v grafu (vztah čekajícího procesu a přiděleného zdroje) je posílána z čekajícího uzlu do blokujícího uzlu (Obermarck), pro AND model
  - ✓ Edge-chasing: hranami WFG jsou posílány speciální zprávy (probe – sondy). Jestliže je sondovací zpráva přijata iniciátorem, je detekováno uváznutí (Chandy-Misra-Haas), pro AND model
  - ✓ Global state detection: získává se snímek (snapshot) o distribuovaném systému, konstruuje se a redukuje se WFG
  - ✓ Diffusion computation: WFG jsou posílány echo zprávy, obsahující dotaz na stav jednotlivých uzlů

Path-pushing, Obermarck

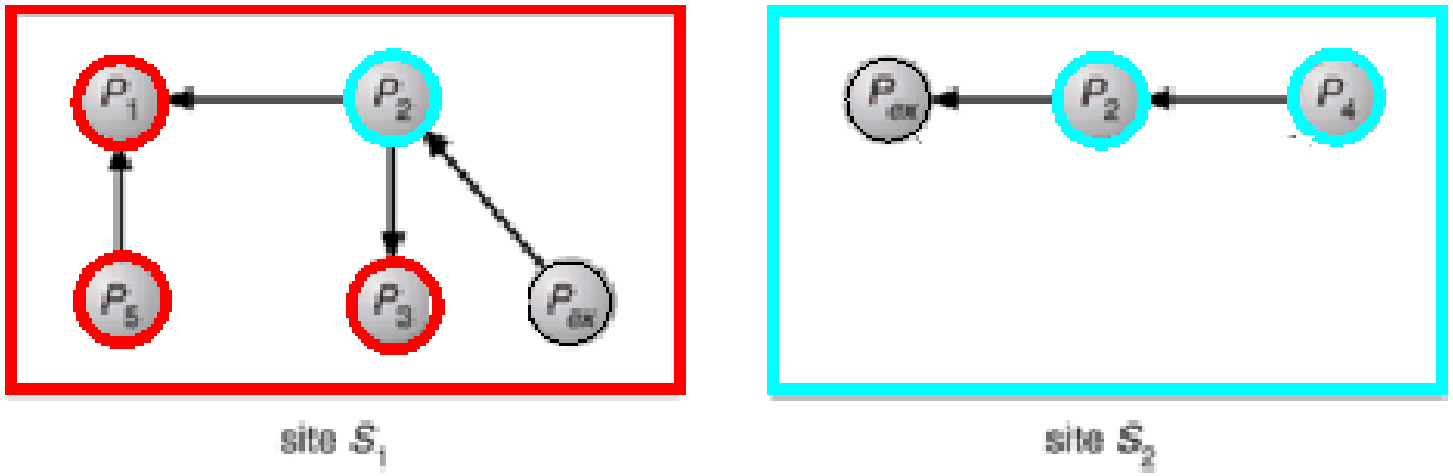
- Za detekci uváznutí sdílejí odpovědnost všechny uzly DS.
- V každém uzlu DS se konstruuje lokální wait-for graf, který reprezentuje část globálního wait-for grafu
- Do každého lokálního wait-for grafu se přidává dodatečný uzel P_{ex}
  - ✓ Hrana $P_i \rightarrow P_{ex}$ reprezentuje stav, ve kterém P_i čeká na zdroj držený procesem ve kterémkoliv jiném uzlu sítě
  - ✓ Hrana $P_{ex} \rightarrow P_i$ reprezentuje stav, ve kterém proces ve kterémkoliv jiném uzlu sítě čeká na zdroj držený procesem P_i v lokálním uzlu

Path-pushing, Obermarck

- Jestliže lokální wait-for grafu obsahuje cyklus,
  který neobsahuje uzel P_{ex},
  pak procesy v tomto uzlu jsou ve stavu uváznutí
- Cyklus obsahující uzel P_{ex} implikuje možnost uváznutí.

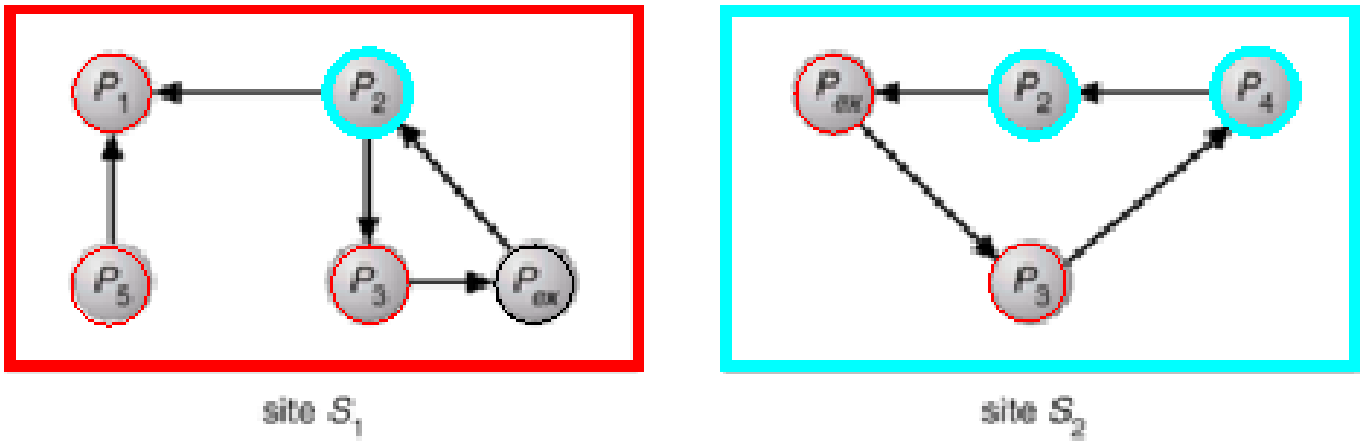
  Pro zjištění, zda procesy v DS uvázly či ne,
  se musí spustit distribuovaný algoritmus detekce uváznutí
- algoritmus může detekovat i falešná uváznutí protože snímek zjišťuje asynchronně

Path-pushing, Obermarck



P1 běžící v S1 má přidělený zdroj v S1
P3 běžící v S1 má přidělený zdroj v S1
P5 běžící v S1 čeká na uvolnění zdroje v S1 drženého P1
P2 běžící v S2 má přidělený zdroj v S2 a
čeká na uvolnění zdrojů v S1 držených P1 a P3
P4 běžící v S2 má přidělený zdroj v S2 a
čeká na uvolnění zdroje v S2 drženého P2

Path-pushing, Obermarck



P1 běžící v S1 má přidělený zdroj v S1

P3 běžící v S1 má přidělený zdroj v S1 a
čeká na uvolnění zdroje v S2 drženého P4

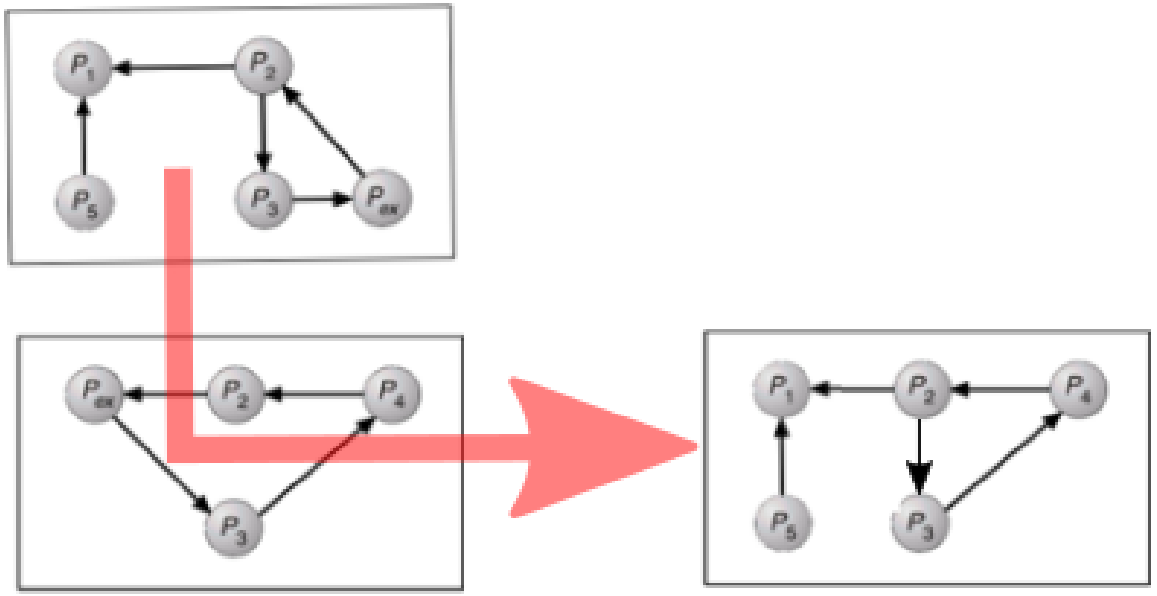
P5 běžící v S1 čeká na uvolnění zdroje v S1 drženého P1

P2 běžící v S2 má přidělený zdroj v S2 a
čeká na uvolnění zdrojů v S1 držených P1 a P3

P4 běžící v S2 má přidělený zdroj v S2 a
čeká na uvolnění zdroje v S2 drženého P2

Path-pushing, Obermarck

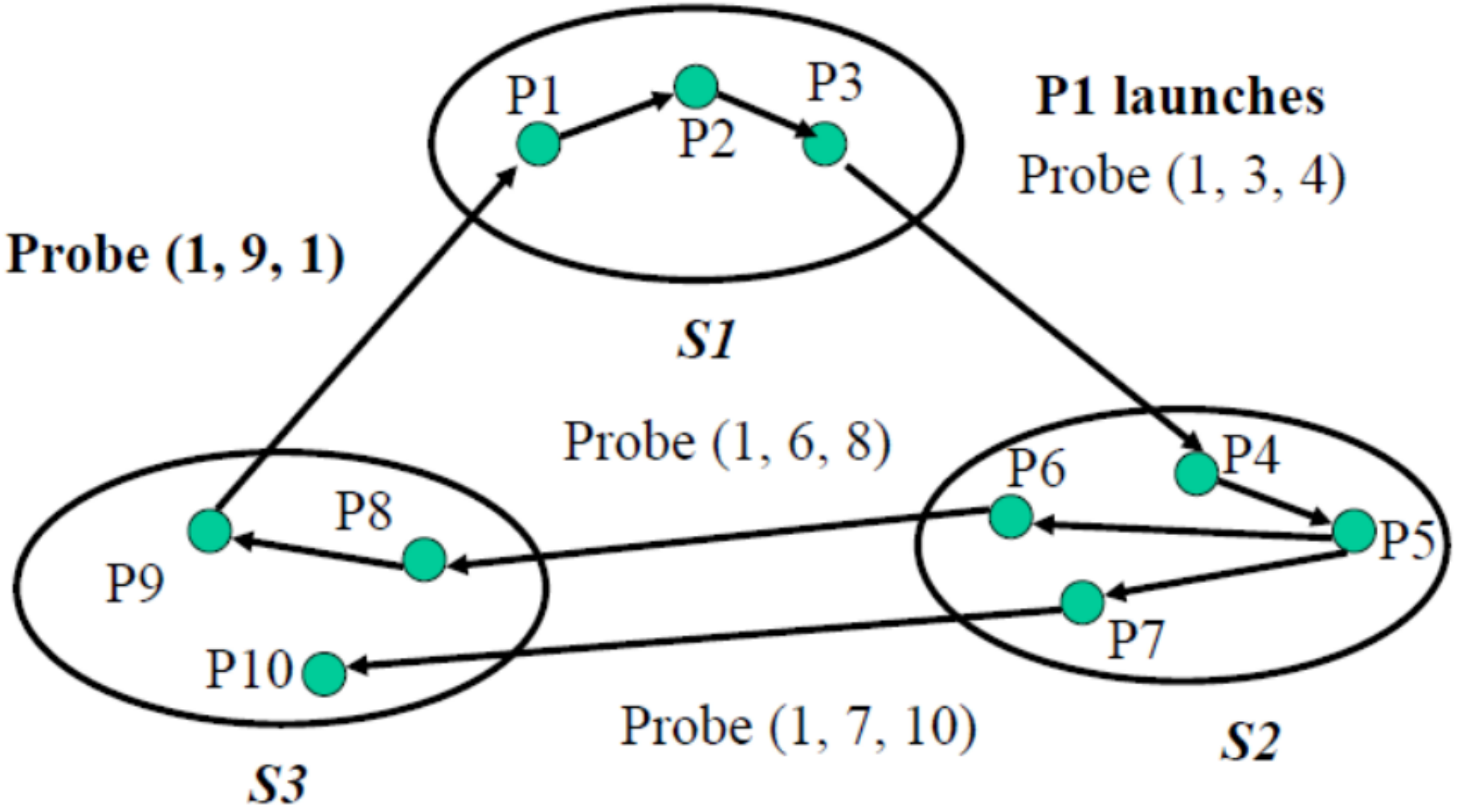
- Nechť s_1 odhalí cyklus $P_3 - P_{ex} - P_2 - P_3$
- Protože P_3 žádá zdroj z s_2, pošle s_1 do s_2 zprávu popisující cyklus v s_1
- s_2 koriguje svůj lokální graf a odhalí existenci uváznutí – cyklus, který neobsahuje P_{ex}:



Edge chasing, Chandy-Misra-Haas

- Blokovaný proces vysílá testovací zprávu (*probe*) procesu, který drží jím požadovaný zdroj
- Proces může čekat na více zdrojů najednou (AND model)
- Testovací zpráva obsahuje
  - ✓ ID blokovaného procesu
  - ✓ ID procesu vysílajícího testovací zprávu
  - ✓ ID cílového procesu testovací zprávy
- Když testovací zprávu přijme blokovaný proces, přepošle ji procesu(ům) držícím zdroj, který požaduje
  - ✓ ID blokovaného procesu se nemění, zbývající dva parametry se mění
- Jestliže blokovaný proces získá svoji testovací zprávu, detekovalo se uváznutí

Edge chasing, Chandy-Misra-Haas



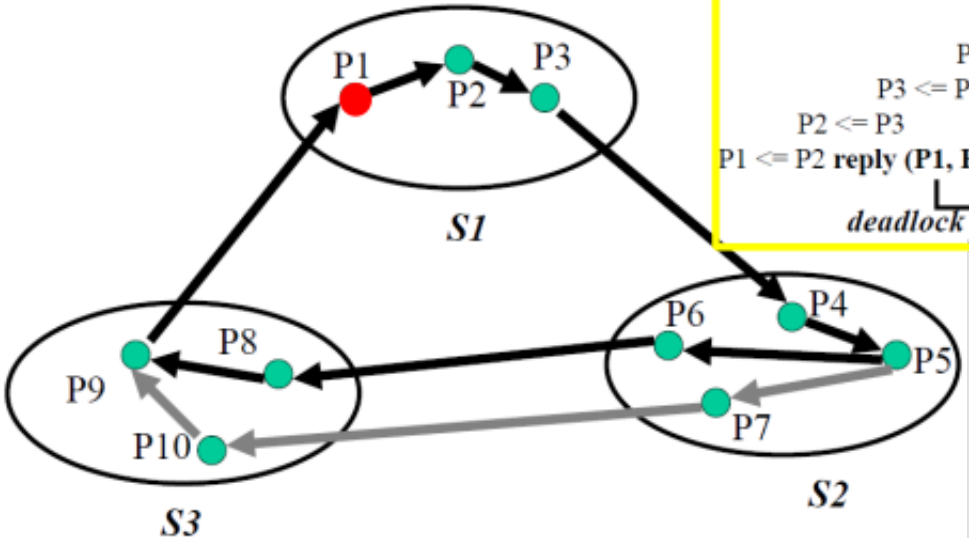
Difusní algoritmus, Chandy at al, OR model

- Výpočet detekce uváznutí difunduje skrz de facto systémový WFG, po hranách čekání mezi procesy
- Z uzlu vyhledávajícího detekci uváznutí jistého procesu se vysílají testovací zprávy (probes) a ty difundují hranami WFG do ostatních uzlů
- jestliže testovací zpráva dosáhne aktivní neblokovaný proces, je zahozena
- jestliže testovací zpráva dosáhne blokovaný proces, posílá se echo zpět iniciátorovi, každý uzel na cestě vždy jakmile dostane echo od všech procesů, kterým poslal testovací zprávu
- pokud všichni pošlou zpět echo iniciátorovi, nastalo uváznutí

Difusní algoritmus, Chandy at al

Testovací zpráva a odpověď obsahují:
ID iniciatora,
ID vysílajícího uzlu,
ID přijímajícího uzlu

P1 vyhledává uváznutí



P1 => P2 **message at P2 from P1** (P1, P1, P2)
P2 => P3 **message at P3 from P2** (P1, P2, P3)
P3 => P4 **message at P4 from P3** (P1, P3, P4)
P4 => P5 **ETC.**
P5 => P6
P5 => P7
P6 => P8
P7 => P10
P8 => P9 (P1, P8, P9), **now reply (P1, P9, P1)**
P10 => P9 (P1, P10, P9), **now reply (P1, P9, P1)**
P8 <= P9 **reply (P1, P9, P8)**
P10<= P9 **reply (P1, P9, P10)**
P6 <= P8 **reply (P1, P8, P6)**
P7 <= P10 **reply (P1, P10, P7)**
P5 <= P6 **ETC.**
P5 <= P7
P4 <= P5
P3 <= P4
P2 <= P3
P1 <= P2 **reply (P1, P2, P1)**

end condition

P5 cannot reply until both P6 and P7 replies arrive !

deadlock condition