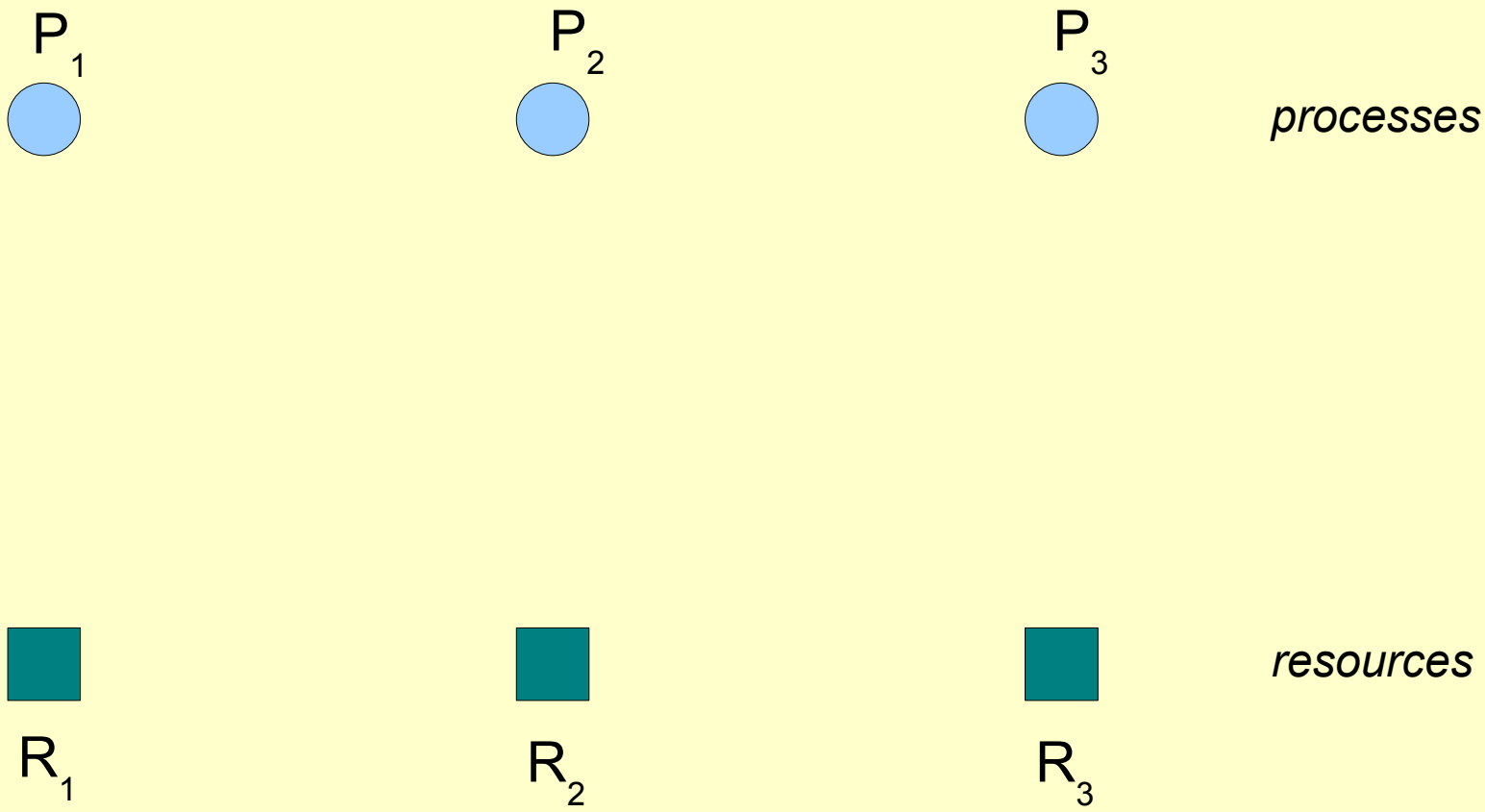
Distribuované systémy a výpočty

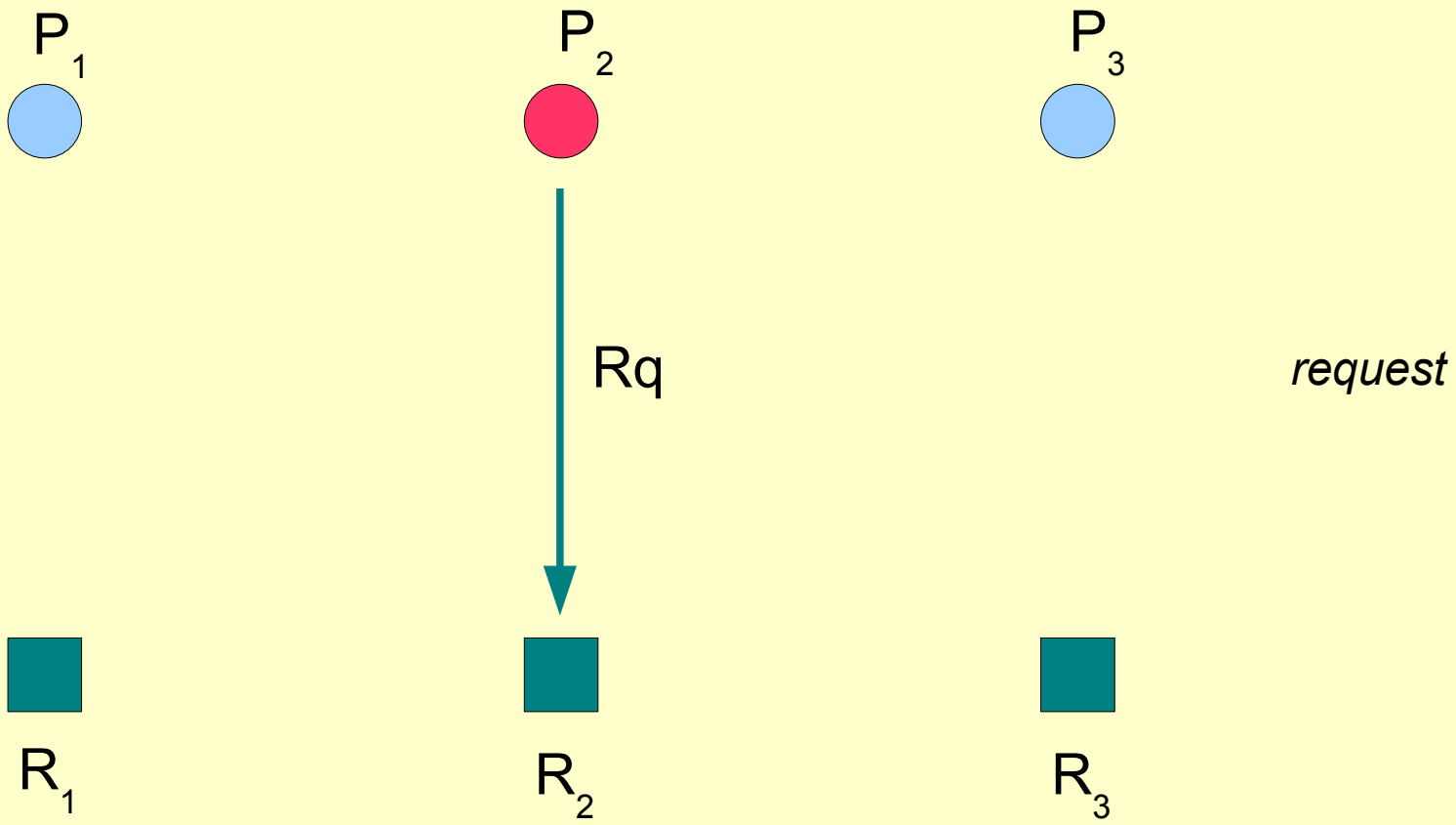
X36DSV

Jan Janeček
(dnes Peter Macejko)

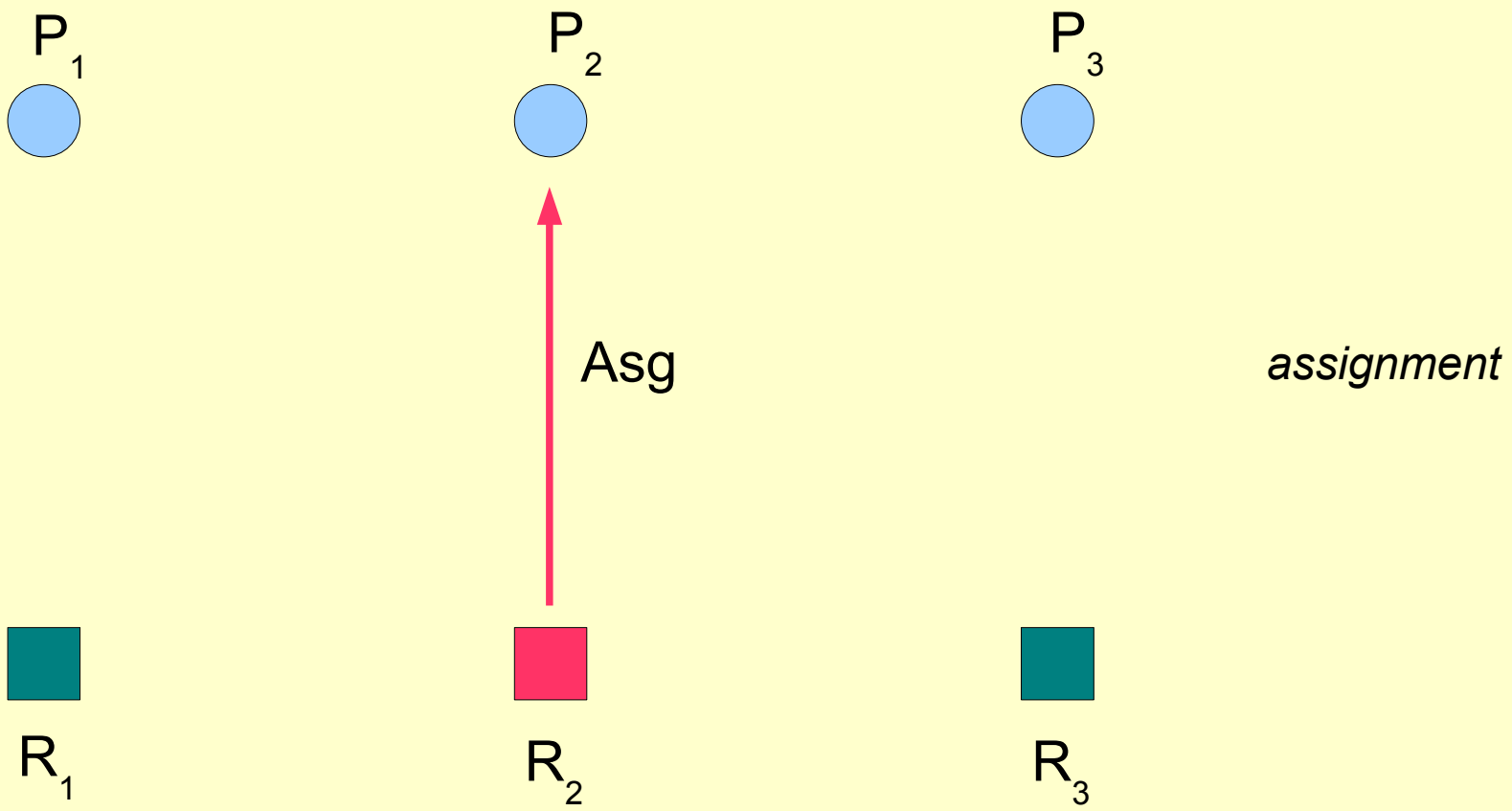
Zablokování na prostředcích



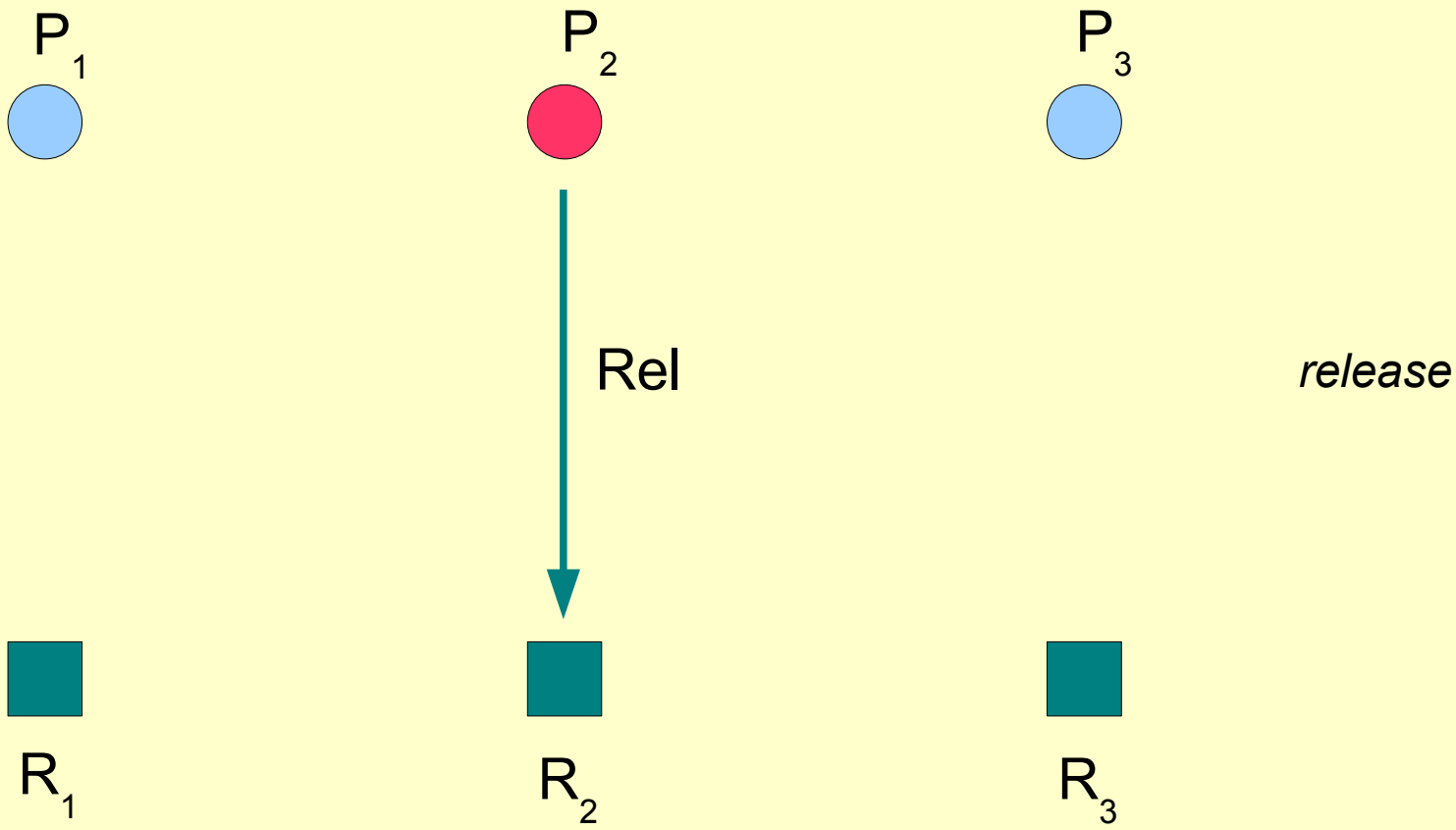
Zablokování na prostředcích



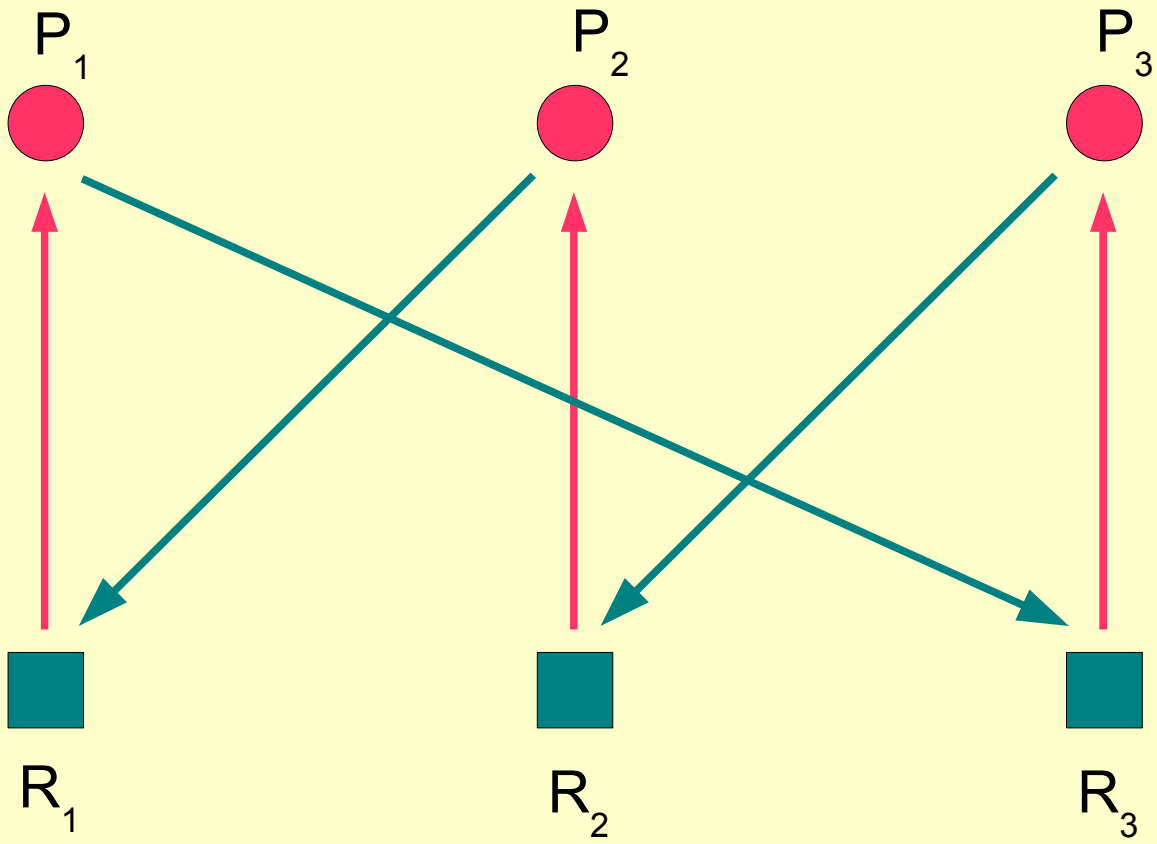
Zablokování na prostředcích



Zablokování na prostředcích

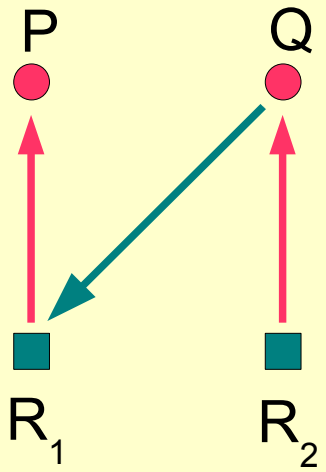


Zablokování na prostředcích

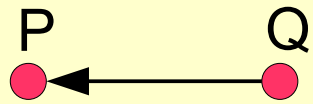


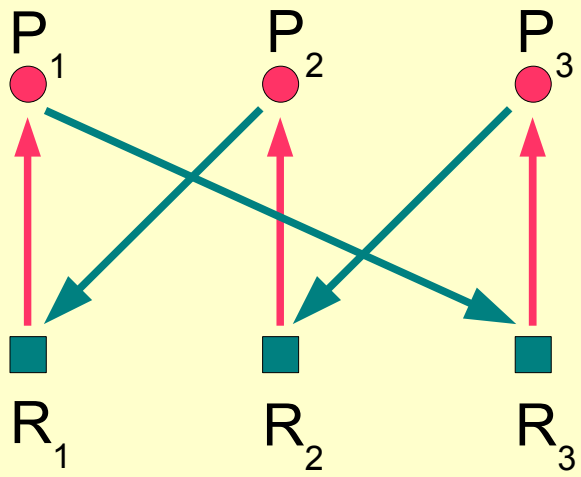
deadlock !

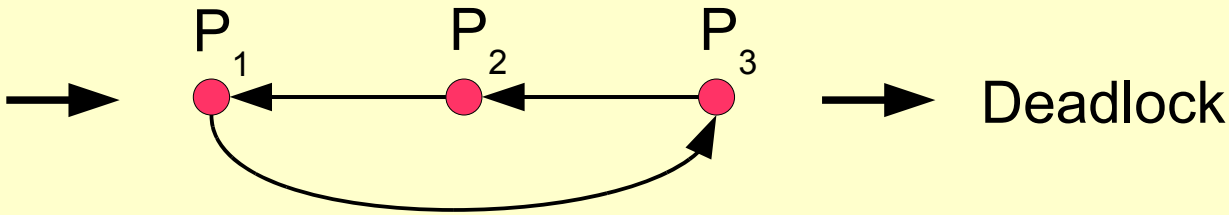
Zablokování na prostředcích



Relace *závislosti*
Q závisí na P







Zablokování na prostředcích

Deadlock – Coffmanovy podmínky

1) Přidělení prostředku je výlučné v čase.
2) Při čekání na prostředek může mít už jiný přidělený.
3) Uvolnění prostředku jen z vlastní iniciativy.
4) V grafu závislostí může vzniknout cyklus.

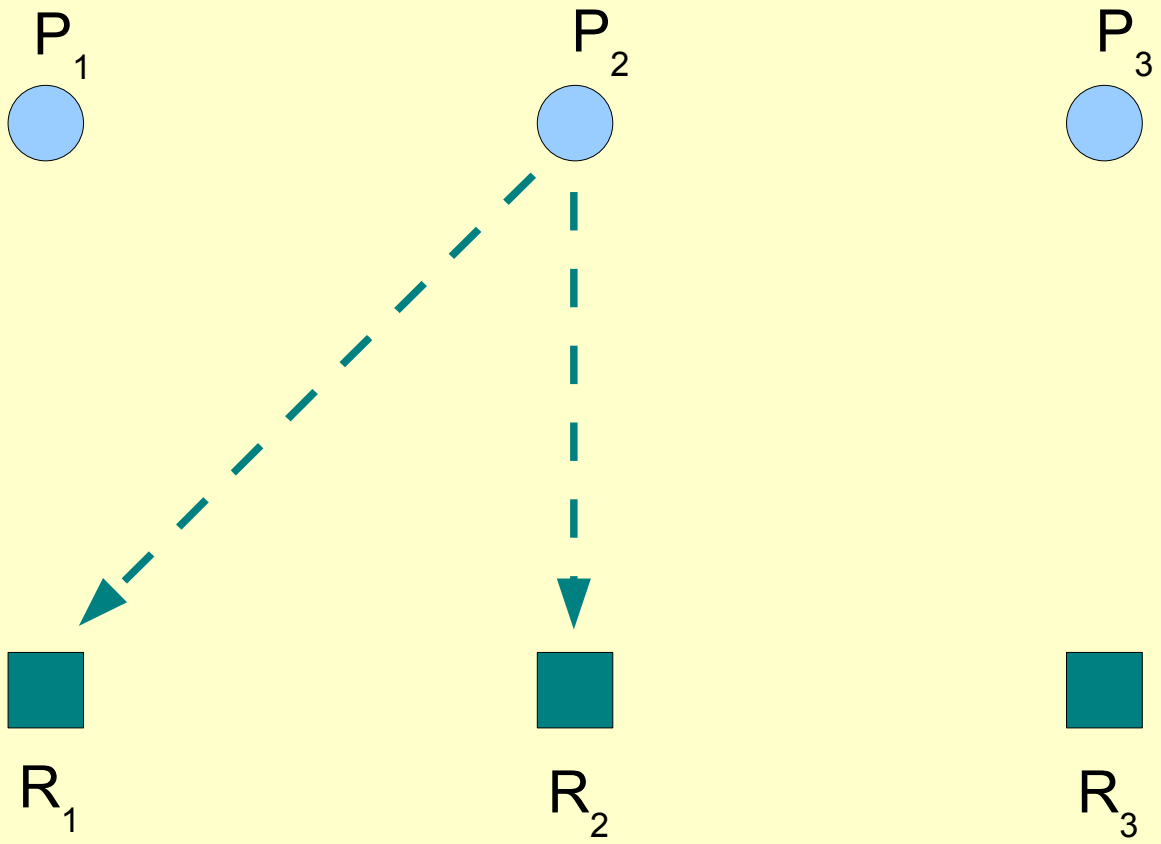
Řešení

a) preventivní – pesimistické – **apriorní**
b) dodatečné – optimistické – **aposteriorní**

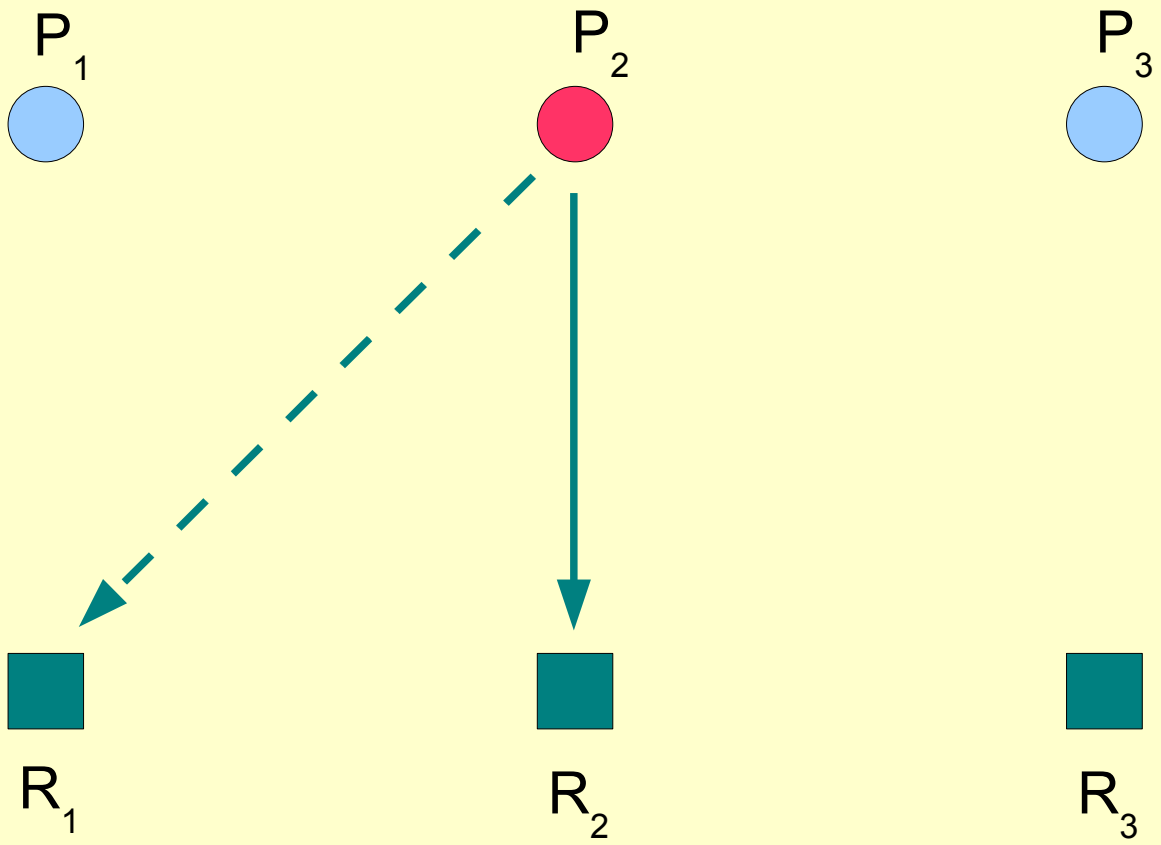
Prevence – Lomet

Lomet
- plná upořadatelnost požadavků
- předběžné žádosti – modifikovaný graf závislostí
- globální x lokální informace
- užití v DB serverech (globální)

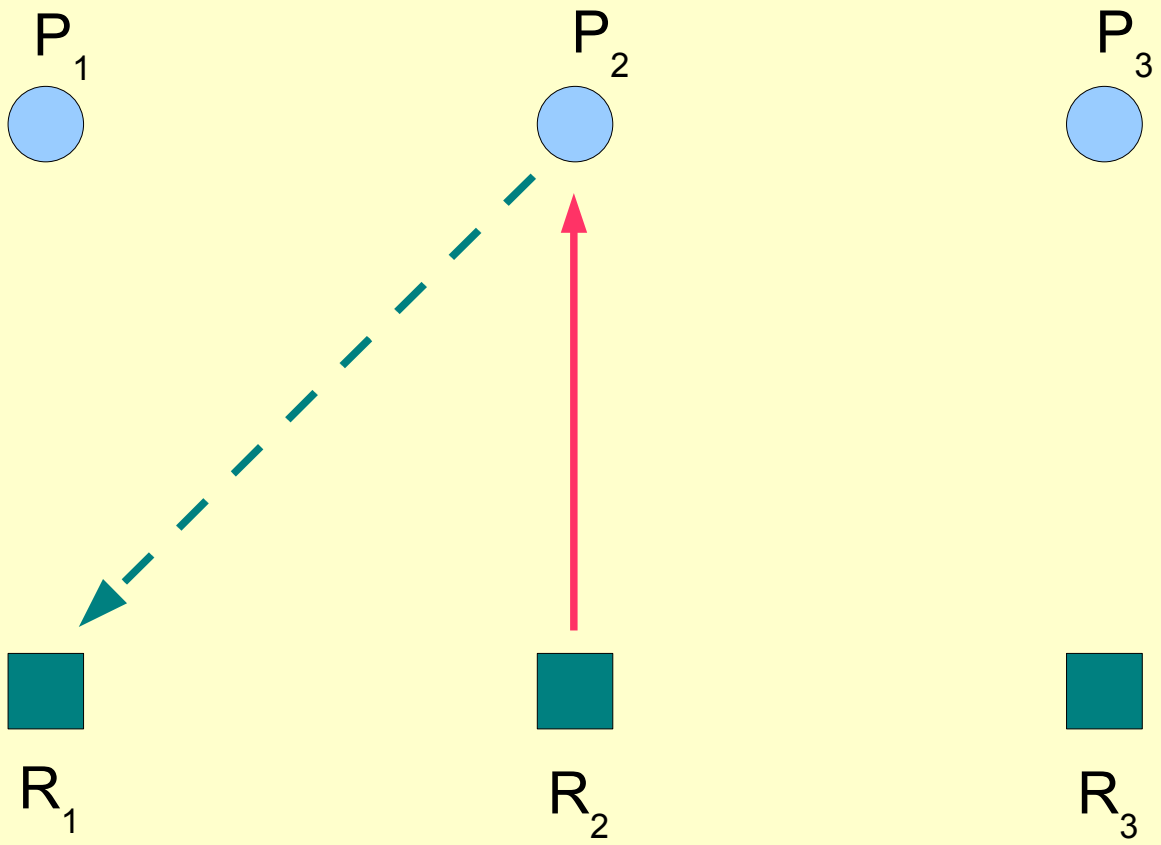
Prevence - Lomet



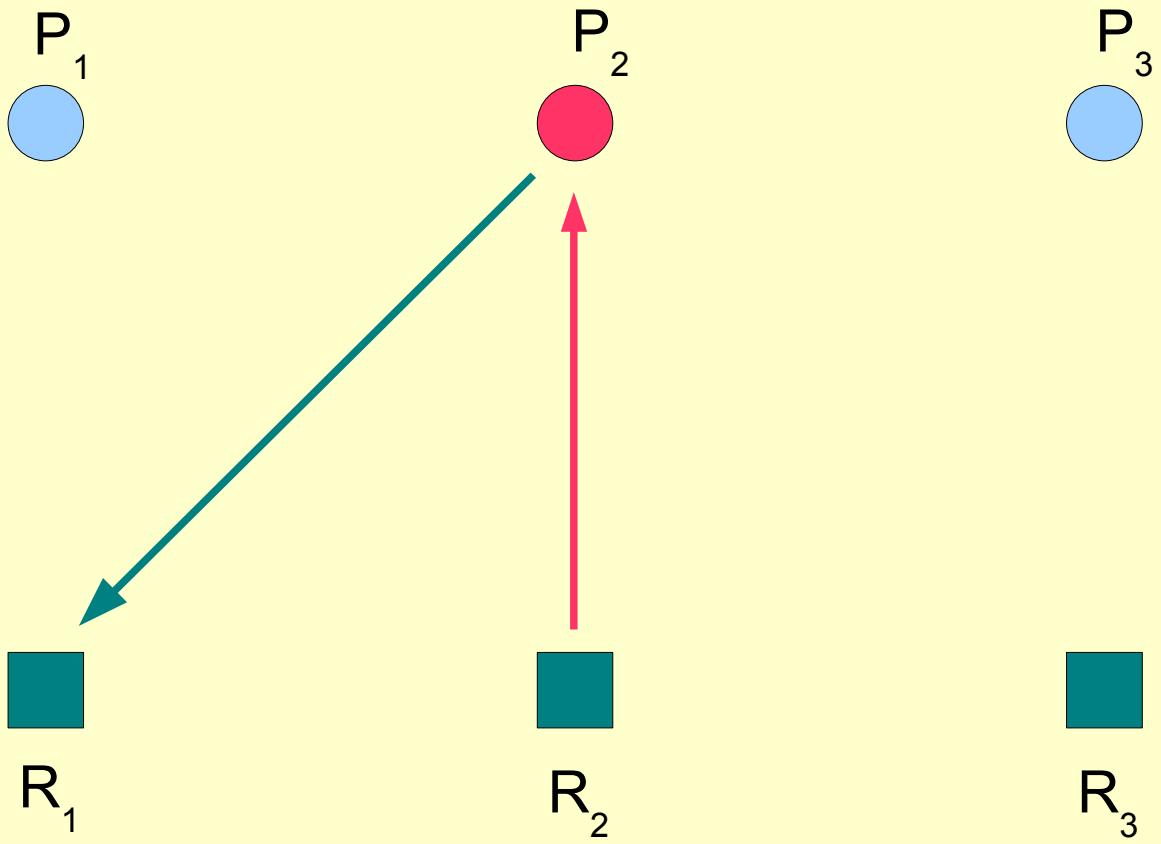
Prevence – Lomet



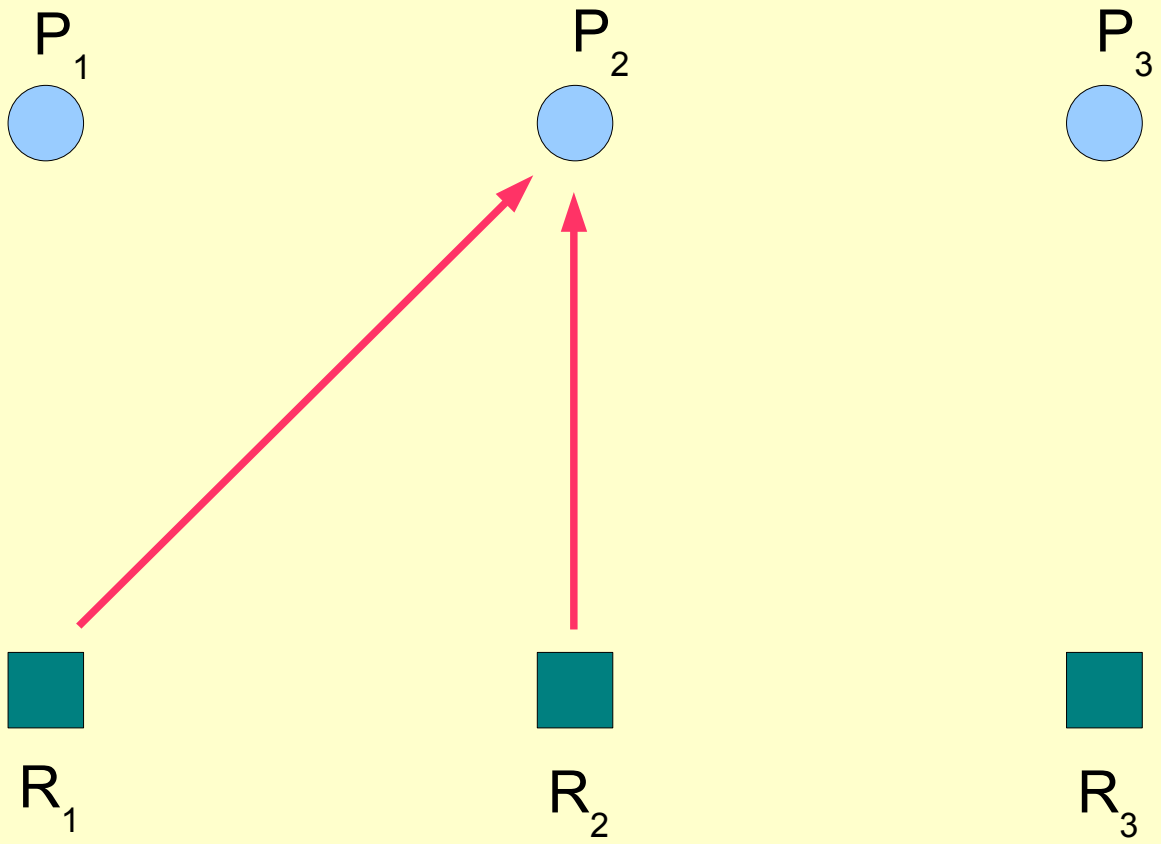
Prevence – Lomet



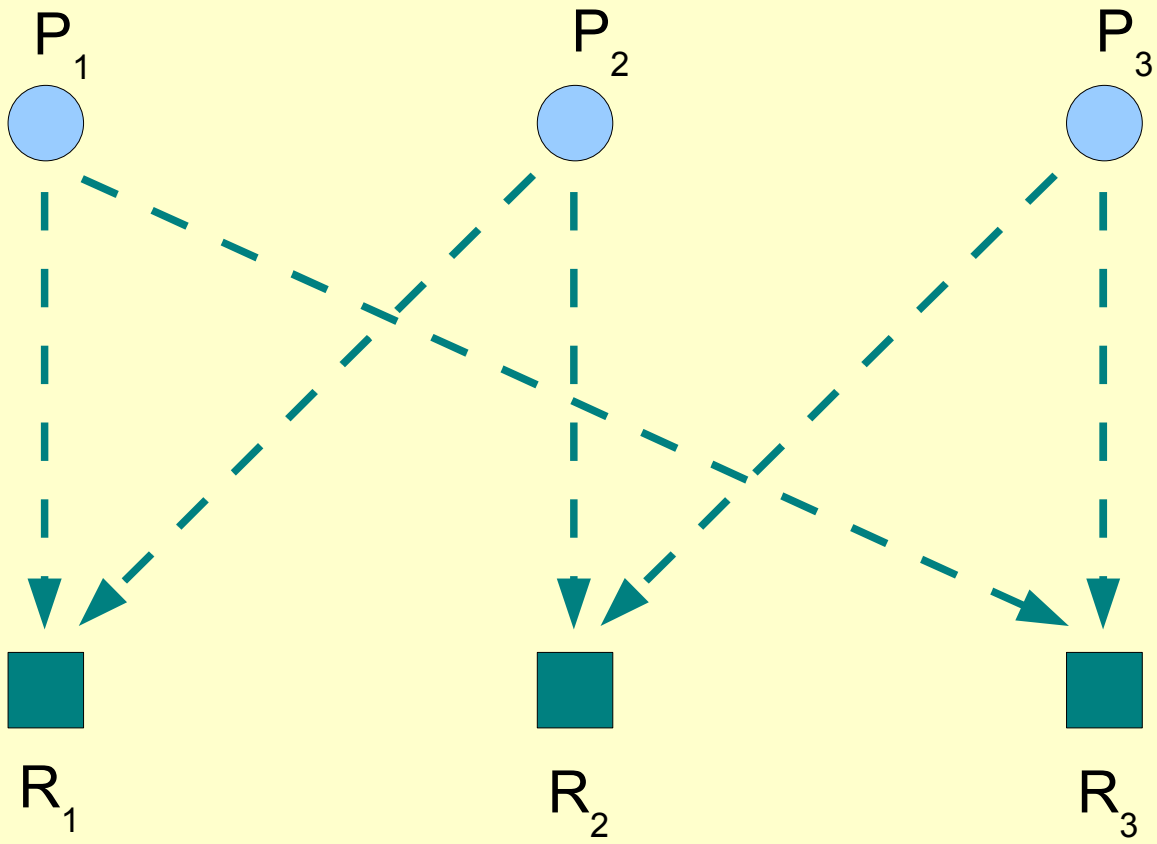
Prevence – Lomet



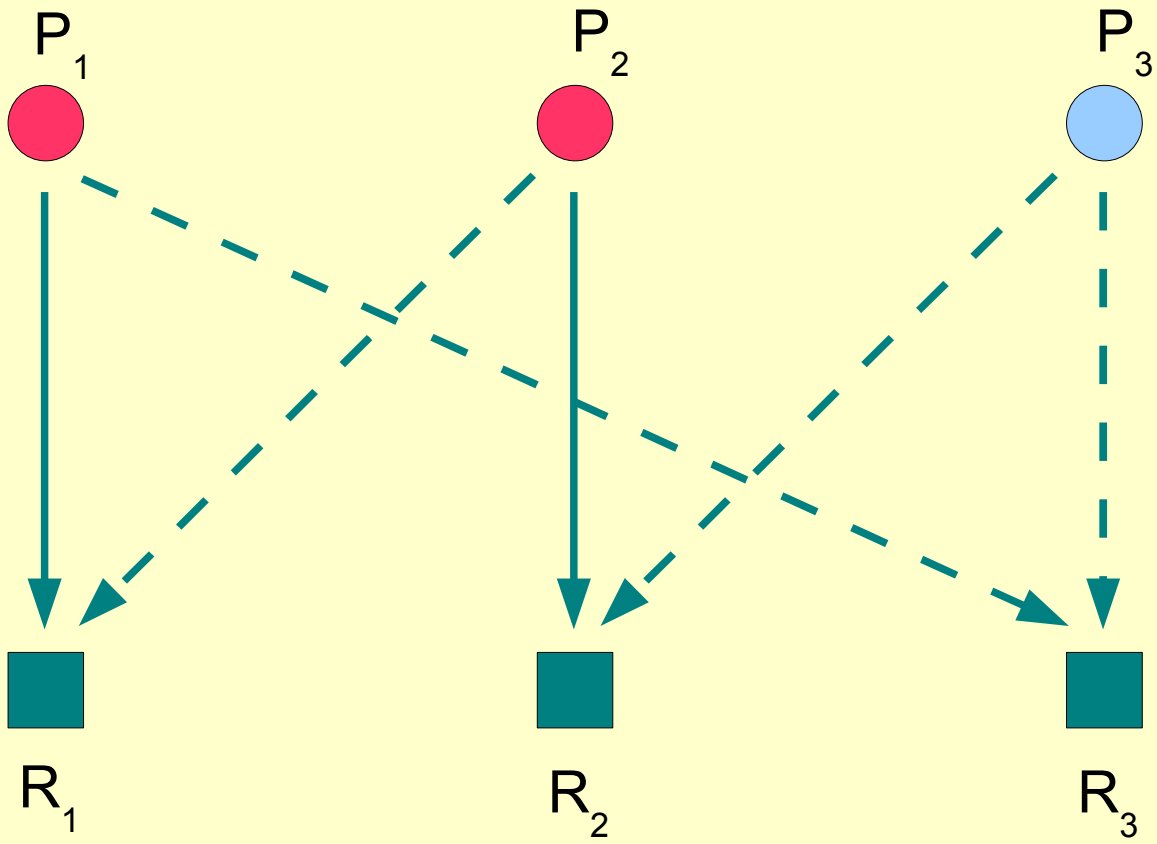
Prevence – Lomet



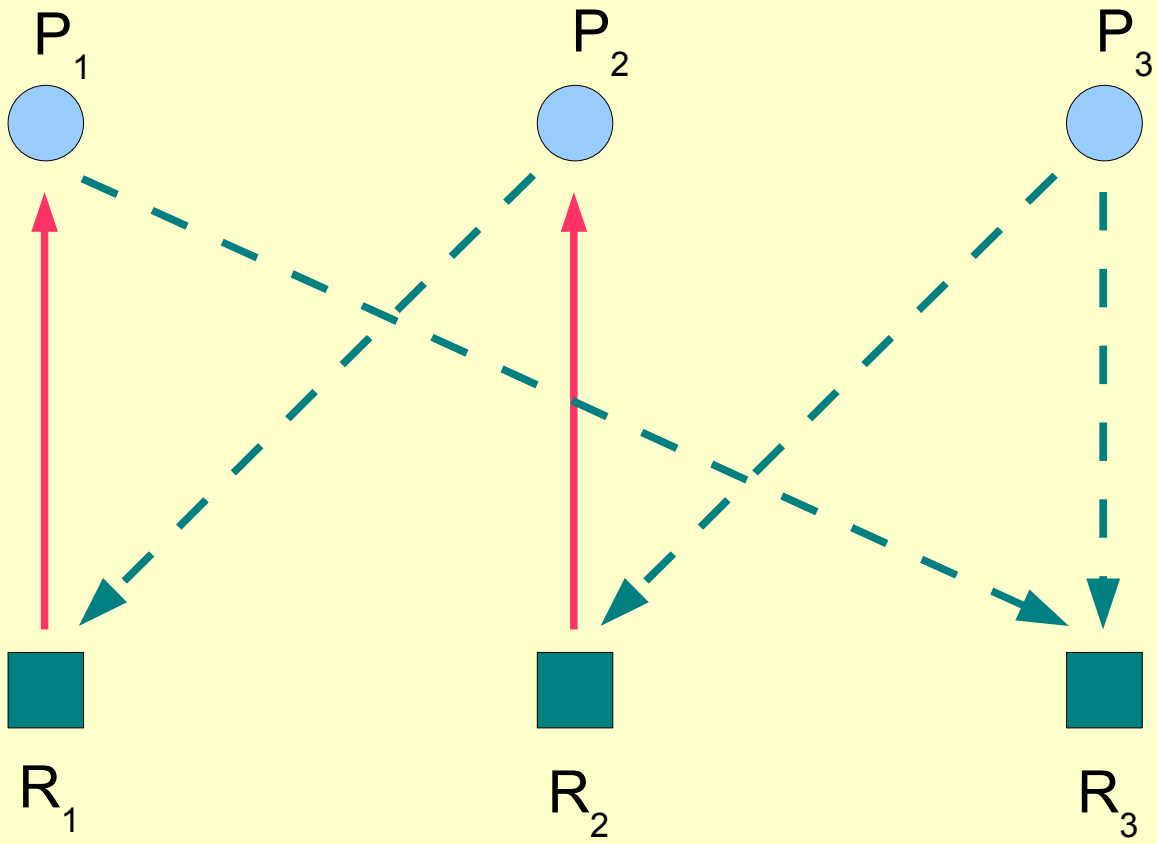
Prevence – Lomet



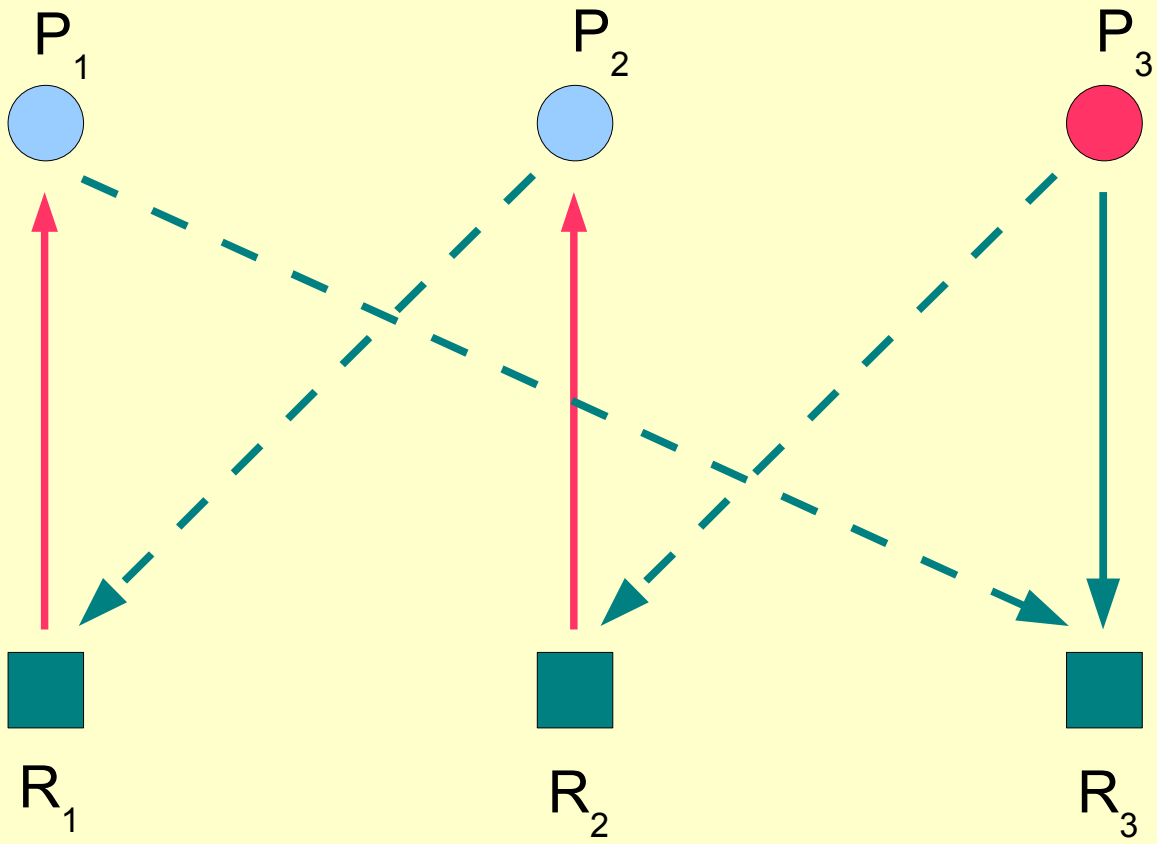
Prevence – Lomet



Prevence – Lomet

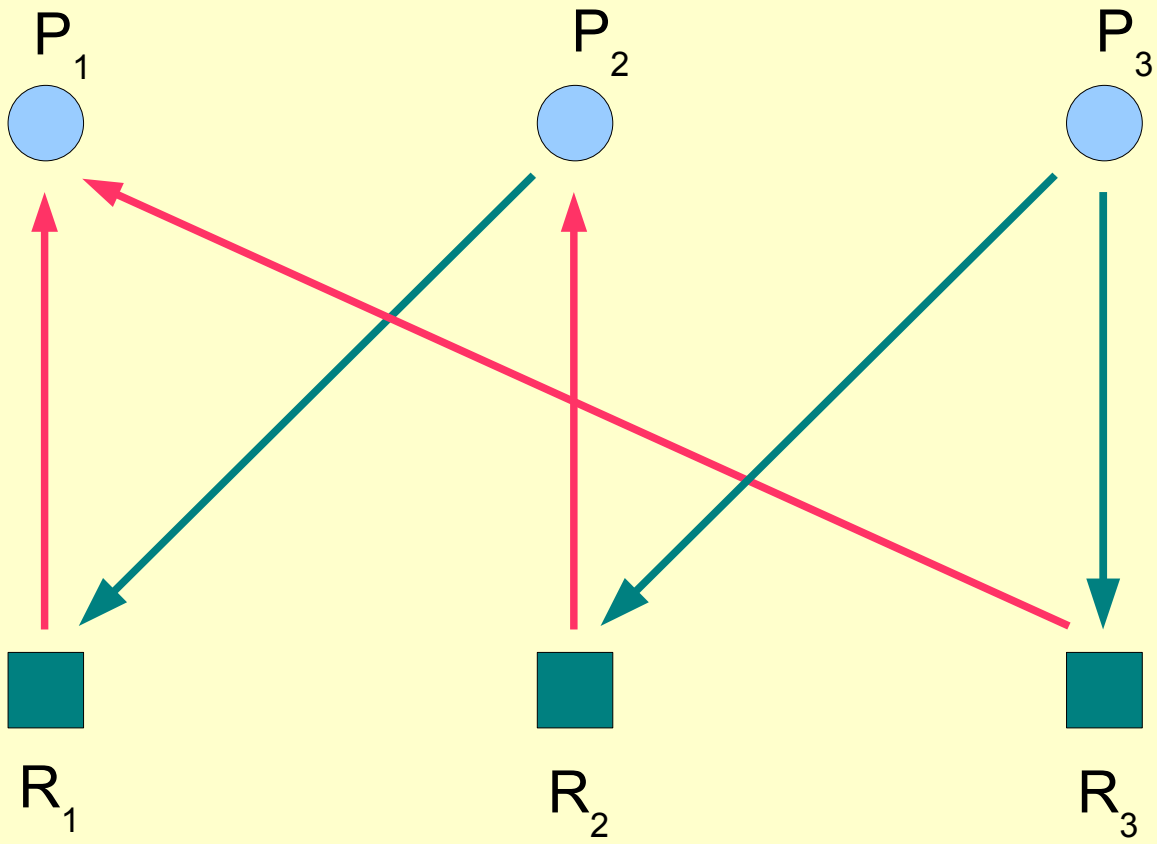


Prevence – Lomet



a tady musím počkat ...

Prevence – Lomet



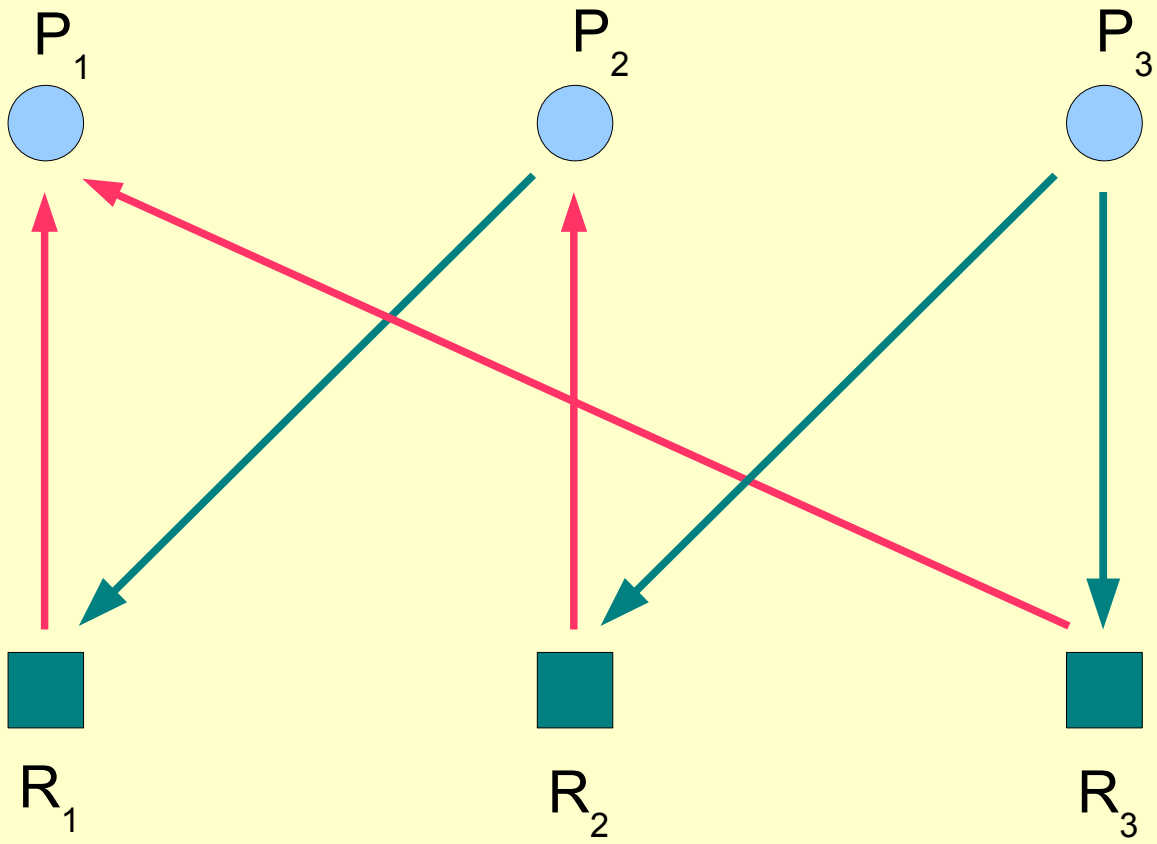
ale jak to zařídit ?

Prevence – Lomet

Lomet – globální informace
- globální graf závislostí
- při možnosti vzniku cyklu nepřidělím
- distribuovaná verze pomocí výlučného přístupu

Lomet – lokální informace
- graf závislostí pouze pro jeden zdroj
- plná uspořadatelnost požadavků (pomocí VP)
- silnější podmínka (horší efektivita)

Prevence – Lomet (lokální)



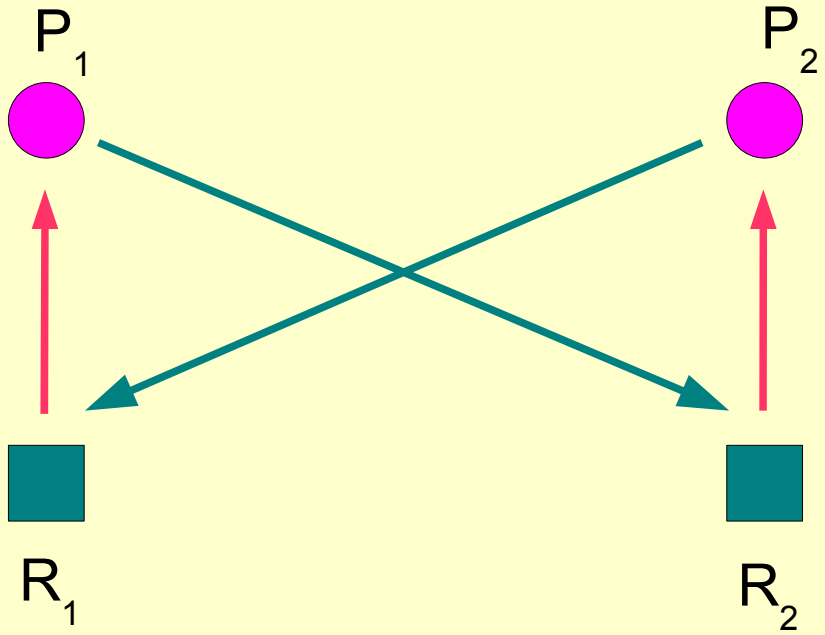
ale jak to zařídit ?

… časové známky, ...?

Detekce

aposteriori

optimistická strategie

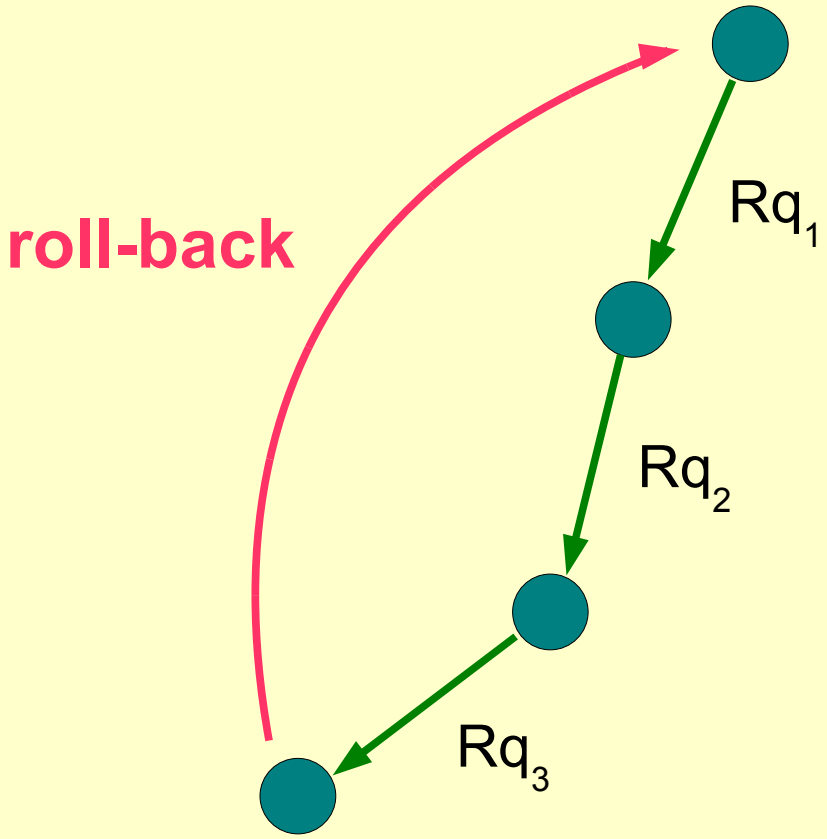


při uváznutí

- výběr procesu (oběti)
- uvolnění prostředků
- vrácení výpočtu procesu

Detekce

transakce



Detekce

Systém:

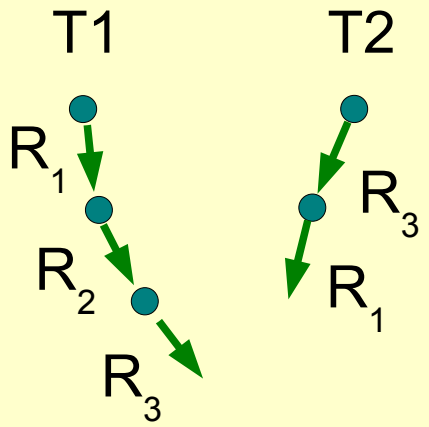
transakce T1 a T2
sdílený prostředek R1 – přidělen T1
časové známky začátku transakcí e(T1) a e(T2)
přijde žádost od T2
T_{old} = T1, T_{new} = T2

Řešení 1

if e(T_{new}) < e(T_{old}) **then** halt T_{new}
else kill T_{new}

Řešení 2

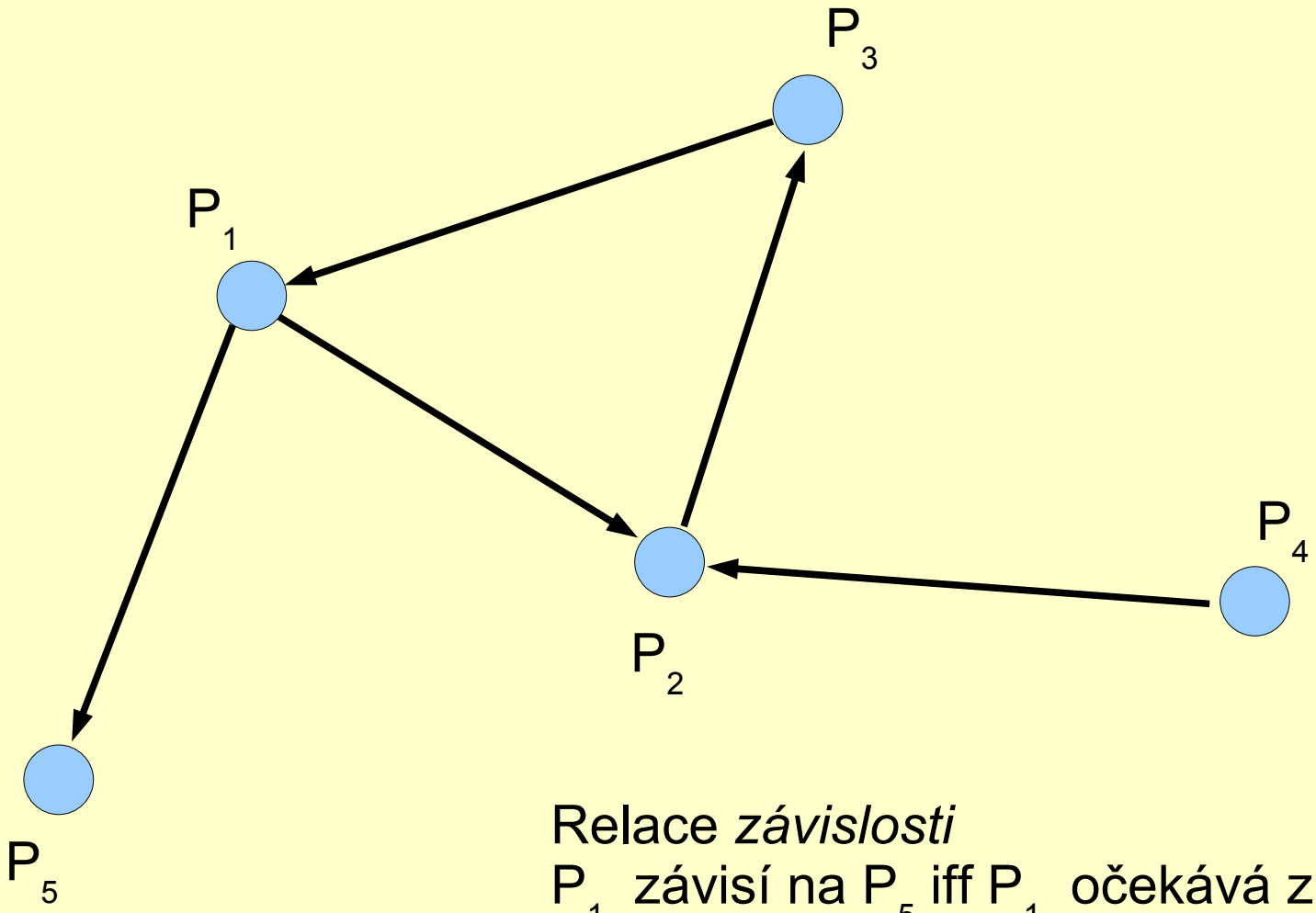
if e(T_{new}) < e(T_{old}) **then** kill T_{old}
else halt T_{new}



!!! opakování transakce vždy s původní časovou známkou !!!

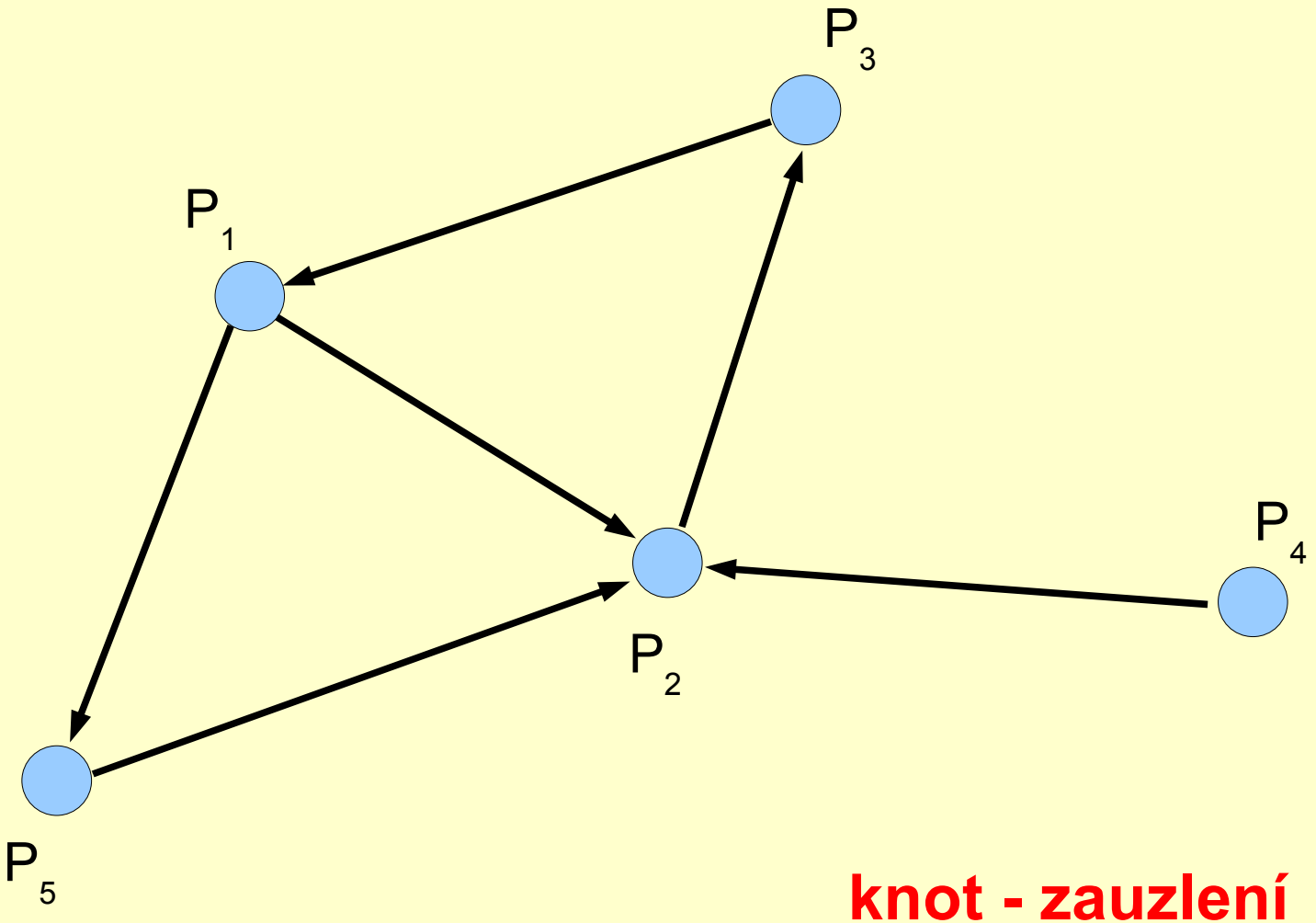
Silnější, ekvivalentní nebo slabší podmínka deadlocku?

Zablokování při komunikaci



Relace *závislosti*
P_1 závisí na P_5 iff P_1 očekává zprávu odP_5

Zablokování při komunikaci



knot - zauzlení

Zablokování při komunikaci

Proces P
- aktivní x pasivní
- množina závislostí P – dependency set of P – DS(P)

Zablokování na množině S
- 1) každý P z S je pasivní
- 2) pro každý P z S platí, že DS(P) je podmnožinou S
- knot = zauzlení – podgraf grafu závislostí odpovídající dané S

Zdánlivé zablokování
- problém se zjišťováním globálního stavu

Zablokování při komunikaci

Chandy – Misra - Hass

Proměnné – legenda

Last[i] - poslední známé číslo testu od procesu i
Wait[i] - informace o aktivitě procesu i
Parent[i] - informace o rodiči procesu v testu od i
Number[i] - počet nezodpovězených dotazů v testu od i

Komunikační primitiva – legenda

QUESTION(k,m,j,i) a ANSWER(k,m,j,i) kde
k – id zahajovacího procesu, m – pořadové číslo testu,
j – odesílatel žádosti, i – příjemce žádosti

inicializace

```
begin
  for i:=1 to n do
    begin
     Last[i] := 0; Wait[i] := F
    end
end
```

Zablokování při komunikaci

Chandy – Misra - Hass

```
when decision START DETECTION and State=PASSIVE do
  begin
    Last[i] := Last[i]+1;
    Wait[i] := T;
    for j in DSet do
      send QUESTION(i,Last[i],i,j) to j
    Number[i] := card(DSet)
  end

when received ANY OTHER MESSAGE do                    zpráva aplikace
  begin
    State := ACTIVE;
    for i:=1 to N do
      Wait[i] := F
  end
```

Zablokování při komunikaci

Chandy – Misra - Hass

příjem dotazu

```
when received QUESTION(k,m,j,i) and State=PASSIVE do
  if m>Last[k] then
    begin
      Last[k] := m;
      Parent[k] := j;
      Wait[k] := T;
      for r in DSet do
        send QUESTION(k,m,i,r) to r;
      Number[k] := card(DSet)
    end
  else
  if Wait[k] and m=Last[k] then
    send ANSWER(k,m,i,j) to j
```

Zablokování při komunikaci

Chandy – Misra - Hass

```
when received ANSWER(k,m,r,i) and State=PASSIVE do
  if m=Last[k] and Wait[k] then
    begin
      Number[k] := Number[k]-1;
      if Number[k]=0 then
        if k=i then
            { Pi je zablokován }
        else
        send ANSWER(k,m,i,Parent[k]) to Parent[i]
        Wait[k] := F
    end
```