Skupinové zasílání zpráv, Multicasting

PA 150 ◇ Principy operačních systémů

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

Verze : podzim 2020

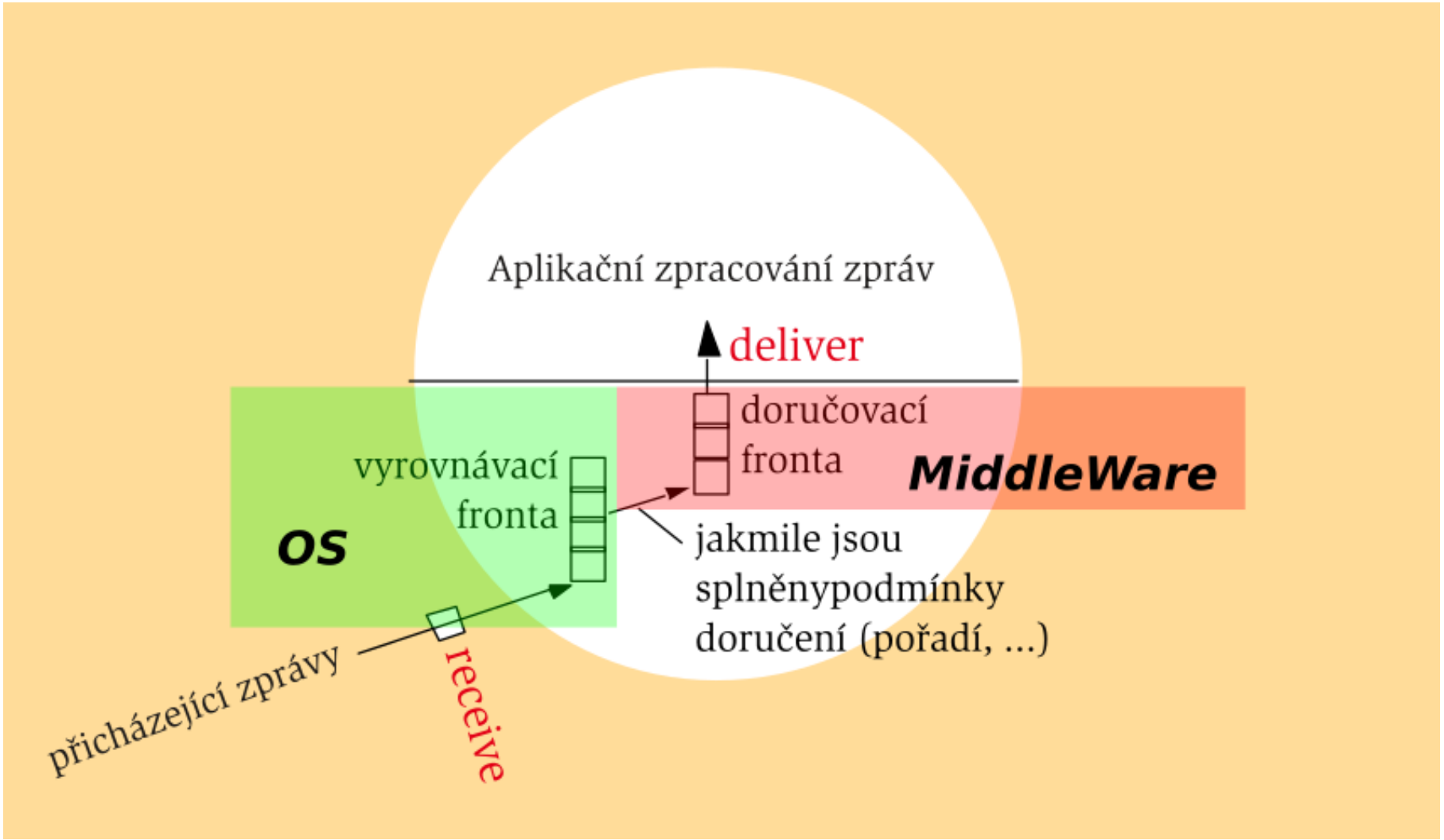
Model skupinové komunikace (Multicast Communication)

- Řešení problému sdělení zprávy všem procesům náležejících do skupiny procesů, složení skupiny je známé
- Procesy mezi sebou komunikují spolehlivými 1:1 kanály
- Procesy mohou krachovat (krach = výpadek = *fail-stop*, proces buď běží správně nebo je nečinný)
- Vysílající proces vysílá zprávu m skupině procesů g jedinou operací, *multicast*(g, m)
  - ✓ ta je implementovaná v middleware násobnými operacemi na úrovni služeb OS, *send*(p, m), kde p jsou id procesů ze skupiny g, $p \in g$
- Zpráva m je unikátně identifikovatelná, nese sebou jedinečný id odesílatele a jednoznačný id skupiny cílových procesů, g=*group*(m)

Problém skupinové komunikace (Multicast Communication)

- Uzel procesu v adresované skupině zprávu přijme
  (operací *receive*) na úrovni služeb OS,
  middleware ji doručí (operací *deliver*) aplikačnímu procesu
- Procesu ve skupině g přijatou zprávu m doručí operace
  deliver(m), komplementární operace k operaci *multicast*,
  s vlastnostmi:
  - ✓ Zpráva m musí být doručena všem procesům aktivním ve skupině g
  - ✓ Pokud zprávu m získá jeden ze skupiny procesů g,
    musí ji získat všechny procesy aktivní ve skupině g

Problém skupinové komunikace (Multicast Communication)



Bázové řešení skupinové komunikace

- *B-multicast*(g, m)

  Basic-multicast – se (na úrovni middleware) implementuje násobným provedením spolehlivých operací *send*(p, m) z úrovně OS ke všem procesům p ze skupiny g

  ✓ Provedení spolehlivých operací *send*(p, m) lze zparalelnit pomocí vláken odpovídajících ve vysílajícím procesu procesům ve skupině g

  ✓ Pak ale potvrzení přijmutí zprávy mají tendenci se kumulovat, vysílač zprávy je nemusí stačit zpracovávat, opakuje proto vysílání, zpráv, což způsobuje příchod dalších potvrzení – *ack-implosion*

- *B-delivered(m)*, komplementární operace k *B-multicast*, zprávu m pro proces p přijímá v uzlu procesu p OS operací *receive (m)* a middleware ji doručuje procesu p

Bázové řešení skupinové komunikace

- Bázové řešení skupinové komunikace nezaručuje splnění požadavku doručení zprávy všem procesům ve skupině
  - ✓ pokud vysílající uzel vypadne během vysílání operacemi *send*(*p*, *m*) implementujícími *B-multicast*(*g*, *m*), některé procesy zprávu dostanou, jiné ne a procesy ve skupině *g* nemají šanci tuto skutečnost (kdo zprávu dostal a kdo ne) zjistit

- Spolehlivé řešení skupinové komunikace požaduje:

  zpráva je doručena všem procesům ve skupině,
  pokud ji získá jeden ze skupiny procesů,
  musí ji získat všechny procesy ve skupině

Spolehlivé (*Reliable*) řešení skupinové komunikace

- Požadované vlastnosti operací *R-multicast*(g, m) a *R-deliver*(m)
  - ✓ integrita – spolehlivá 1:1 komunikace procesů

    korektně se chovajícímu procesu se doručuje zpráva m nejvýše jednou
    (zprávy lze jednoznačně identifikovat:
    např. id zdroje + pořadí zprávy ve zdroji)
  - ✓ validita –

    jestliže korektní proces skupinově rozešle zprávu m,
    zpráva m nakonec bude doručena
  - ✓ dosažení shody –

    jestliže je zpráva m doručena jednomu korektnímu procesu skupiny,
    je doručena všem korektním procesům ve skupině

    společně s validitou se jedná o zárukou živosti algoritmu –
    pokud *send* odvozený z *multicast* vyšle alespoň jednou zprávu
    k procesu skupiny, nesmí výpadek originálního vysílače ohrozit
    doručení zprávy ke všem procesům

Spolehlivé (*Reliable*) řešení skupinové komunikace

- Spolehlivost se dosahuje nadstavbou bázového řešení

On initialization
Received := {}; // *stavová proměnná v každém procesu*

For process p to R-multicast message m to group g
B-multicast(*g*, *m*); // $p \in g$ is included as a destination

On B-deliver(*m*) *at process q with g = group*(*m*)
if ($m \notin$ *Received*)
then
Received := *Received* $\cup$ {*m*};
if ($q \neq p$) *then B-multicast*(*g*, *m*); *end if*
R-deliver m;
end if

Spolehlivé (*Reliable*) řešení skupinové komunikace

- Je zajištěna validita a shoda
  zprávu by některý z procesů ve skupině neobdržel pouze kdyby žádný proces skupiny neudělal *B-multicast*, tj. kdyby žádný nefungoval
- integritu podporuje podpůrný komunikační systém řešící *B-multicast*
- Spolehlivé rozesílání je zaručeno i v asynchronním systému, o čase se nic nepředpokládá
- Každá zpráva je každému procesu skupiny g zaslána $|g|$-krát, :-((

Skupinová komunikace se zachováním pořadí zpráv

- Jestliže je důležité aby se doručování zpráv se odehrávalo v definovaném pořadí, existují 3 typy řazení multicastu:
  - ✓ FIFO řazení –
    provede-li korektní proces *multicast(g, a)* před *multicast(g, b)*,
    pak každý korektní proces v g, který získává zprávu b,
    získá zprávu a před získáním zprávy b
  - ✓ kauzální řazení –
    jestliže *multicast(a)* předchází *multicast(b)* v g z hlediska zasílání zpráv mezi procesy ve skupině g,
    pak každý korektní proces v g, který získává zprávu b,
    získá zprávu a před získáním zprávy b
  - ✓ totální řazení –
    jestliže některý korektní proces v g
    získává zprávu a před získáním zprávy b,
    pak každý korektní proces v g, který získává zprávu b,
    získá zprávu a před získáním zprávy b

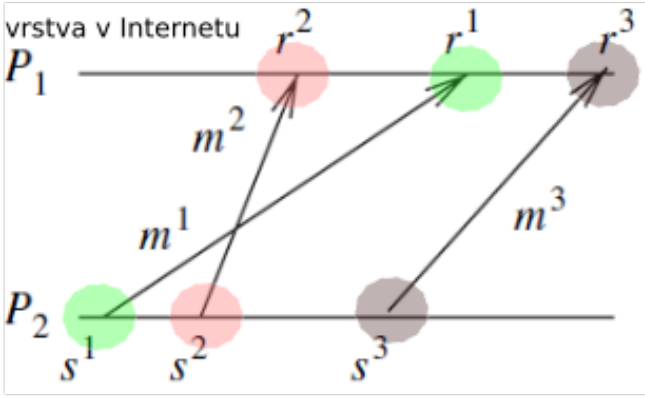
Skupinová komunikace se zachováním pořadí zpráv

- Kauzální řazení přirozeně implikuje FIFO řazení
- Kauzální řazení a FIFO řazení jsou částečná uspořádání
- Definice řazení nepředpokládá ani neimplikuje spolehlivost doručení
  - ✓ Např. při totálním řazení jestliže je korektnímu procesu p doručena zpráva a po ní zpráva b, procesu q může být doručena a bez doručení b nebo jiných zpráv řazených za a

Paradigmata doručování zpráv v asychr. DS

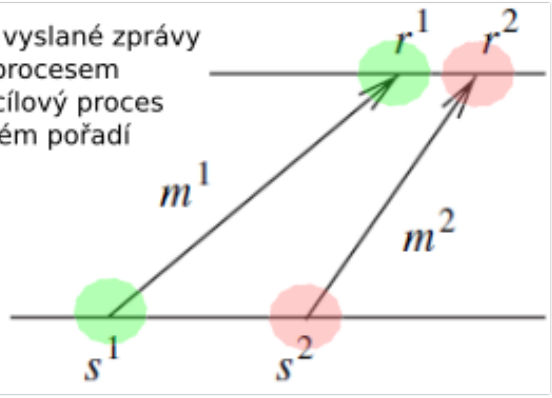
non-FIFO komunikace

např. síťová vrstva v Internetu



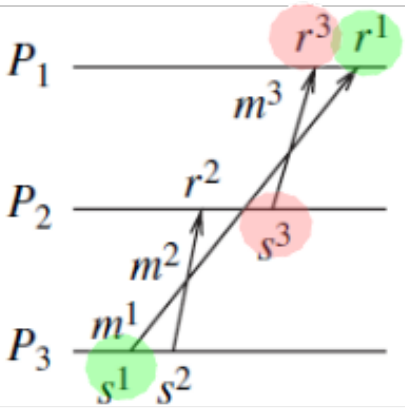
FIFO komunikace

Po sobě vyslané zprávy jedním procesem přijímá cílový proces ve stejném pořadí



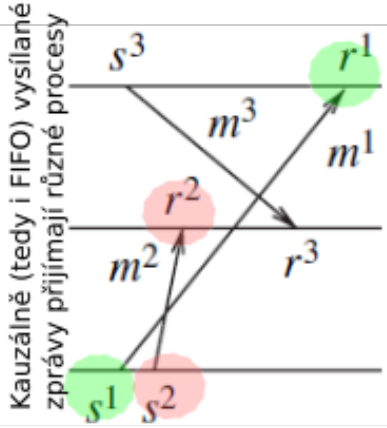
Komunikace nesplňující kauzalitu:

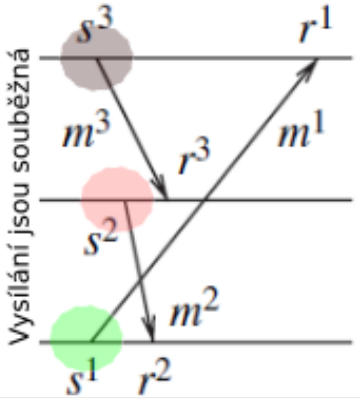
s1 → s3, r3 → r1

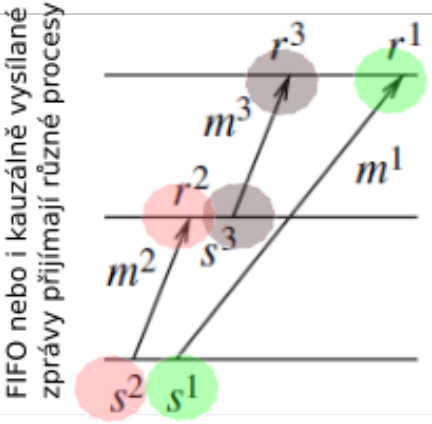


Komunikace splňující kauzalitu

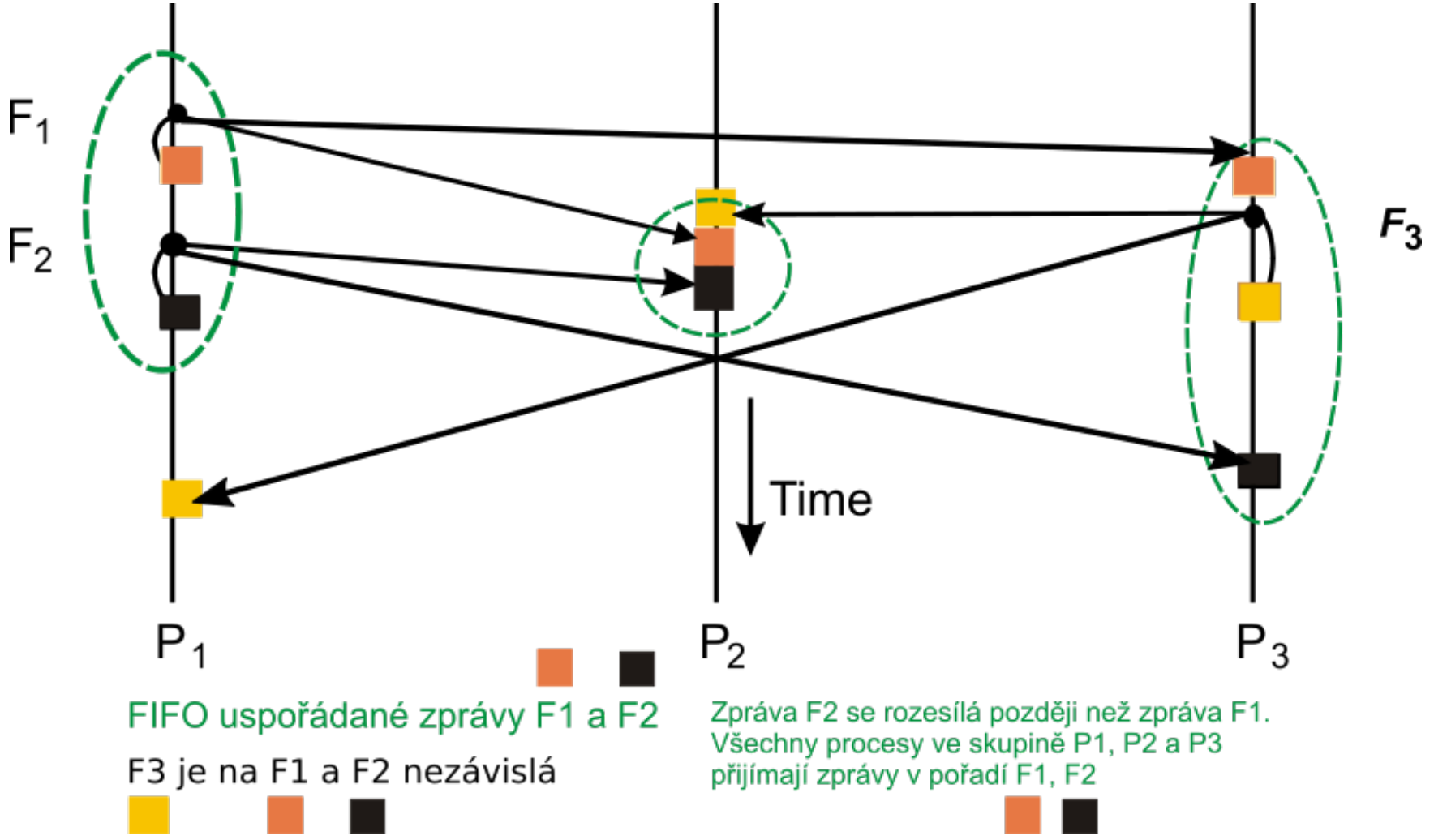
Pokud po sobě vyslané zprávy přijme jeden cílový proces, pak tyto zprávy přijímá ve stejném pořadí







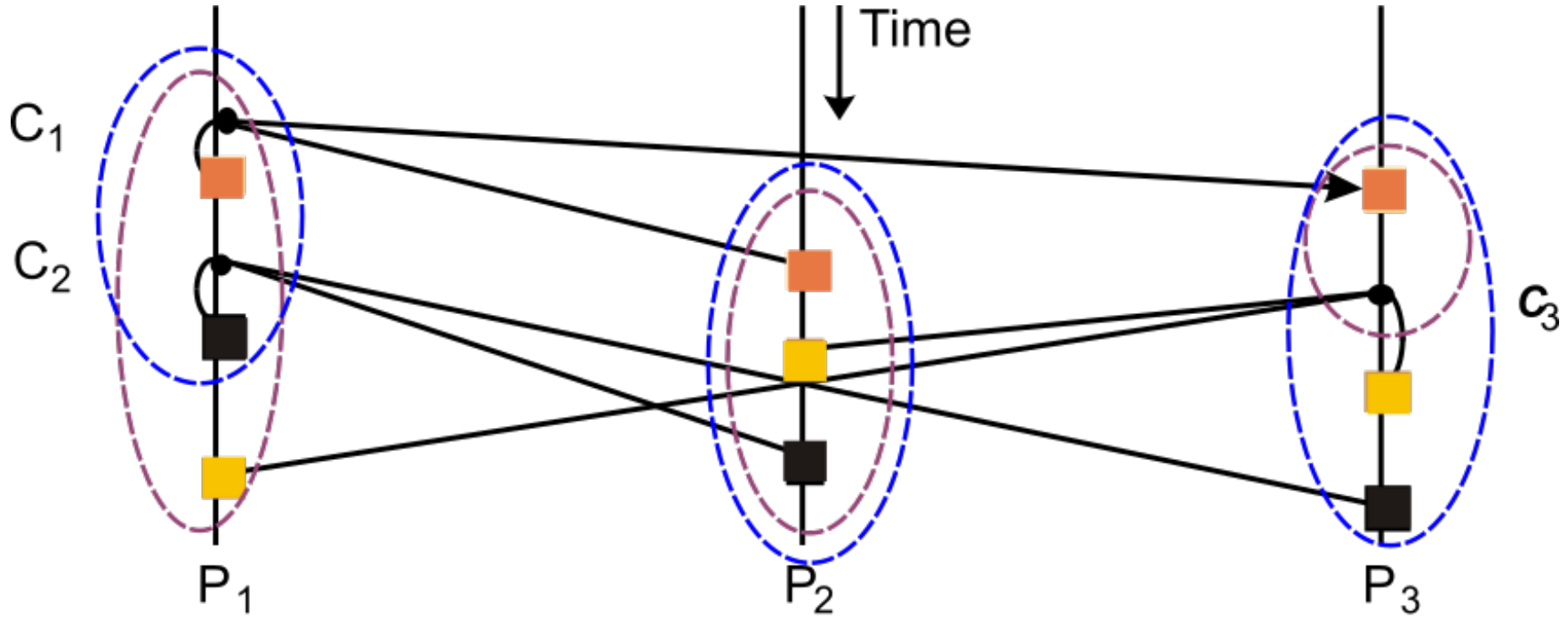
Skupinová komunikace se zachováním FIFO pořadí zpráv



Implementace FIFO řazení

- Pro každý proces p ze skupiny g middleware (MW) udržuje proměnné
  - ✓ S_g^p – čítač zpráv vyslaných procesem p do skupiny g
  - ✓ R_g^q – pro každý proces q ze skupiny g číslo poslední zprávy ze zpráv zaslaných do g procesem q a doručených procesu p
- Při vysílání operací *multicast* přidává MW ke zprávě $S = S_g^p$ a poté S_g^p inkrementuje o 1, zprávy z p pořadově čísluje
- Při přijetí zprávy od q pro p MW kontroluje R_g^q a S ze zprávy
  - ✓ jestliže $S = R_g^q + 1$, MW zprávu doručí p a nastaví $R_g^q = S$
  - ✓ jestliže $S > R_g^q + 1$, MW zprávu zapamatuje ve vyrovnávací frontě a doručí ji procesu p až bude platit $R_g^q + 1 = S$
- Jestliže se použije spolehlivé rozesílání, *R-multicast*, získáme spolehlivé FIFO rozesílání

Skupinová komunikace se zachováním kauzálního pořadí

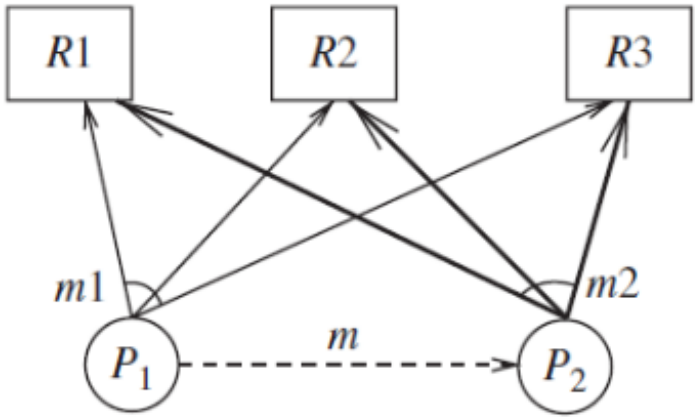


Kauzálně uspořádané zprávy C1 a C3 — mulitcast C3 následuje po multicast C1

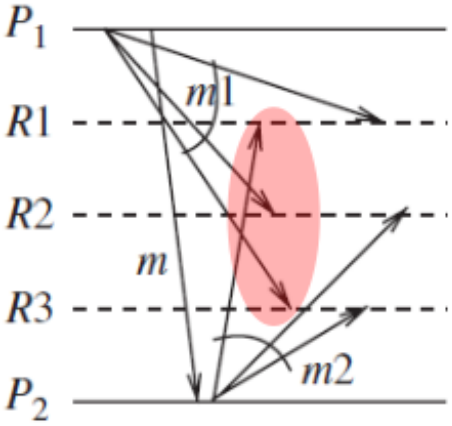
Kauzálně uspořádané zprávy C1 a C2 — mulitcast C2 následuje po multicast C1
(kauzální řazení implikuje i FIFO řazení)

Mezi zprávami C3 a C2 kauzální vztah neexistuje

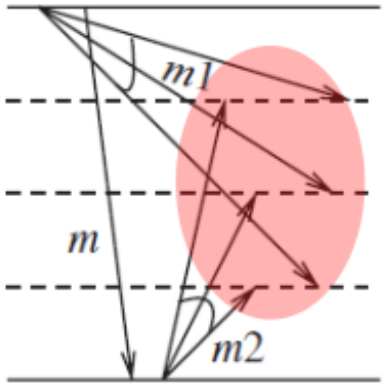
Skupinová komunikace se zachováním kauzálního pořadí



(a)



(b)



(c)

P1 a P2 jsou procesy běžící ve dvou různých uzlech DS a korigují 3 replikace jednoho zdroje ve 3 různých uzlech DS

Korekce prováděné P2 musí proběhnout po korekcích prováděných P1, protože vysílání P2 kauzálně následuje po vysílání P1 díky vyslání zprávy *m* z P1 do P2 po vyslání zpráv m1

Porušení kauzality

Implementace kauzálního řazení

- Bázová idea
  - ✓ Respektování relace *stalo-se-před* na bázi logického času hnaného rozesíláním a doručováním (*multicast*) zpráv
  - ✓ Místo prostého časového razítka TS,
    ve zprávách použijeme vektorové časové razítko V
  - ✓ Vektorové časové razítko V má tolik prvků kolik je procesů ve skupině,
    iniciálně jsou všechny prvky V nulové
  - ✓ Každý proces p_i udržuje své vlastní vektorové časové razítko V_i
  - ✓ Když proces rozesílá zprávu procesům ve skupině,
    nejprve inkrementuje svůj čas ve svém V a poté zprávu s V rozešle
  - ✓ Když se přijme zpráva m pro proces q od procesu p, může mu být doručena až když se mu doručí zprávy, které ji kauzálně předcházejí

    tj. až se q doručí všechny předchozí zprávy vyslané procesem p a
    až se q doručí všechny zprávy, které byly doručeny procesu p do doby,
    kdy p rozeslal m

Implementace kauzálního řazení vektorovými čas. razítky

Algorithm for group member p_i $(i = 1, 2\ldots, N)$

On initialization

$V_i^g[j] := 0$ $(j = 1, 2\ldots, N)$;

To CO-multicast message m to group g

$V_i^g[i] := V_i^g[i] + 1$;

B-multicast$(g, <V_i^g, m>)$;

On B-deliver$(<V_j^g, m>)$ *from* p_j, *with* $g = group(m)$

place $<V_j^g, m>$ in hold-back queue;

wait until $V_j^g[j] = V_i^g[j] + 1$ and $V_j^g[k] \leq V_i^g[k]$ $(k \neq j)$;

CO-deliver m; // after removing it from the hold-back queue

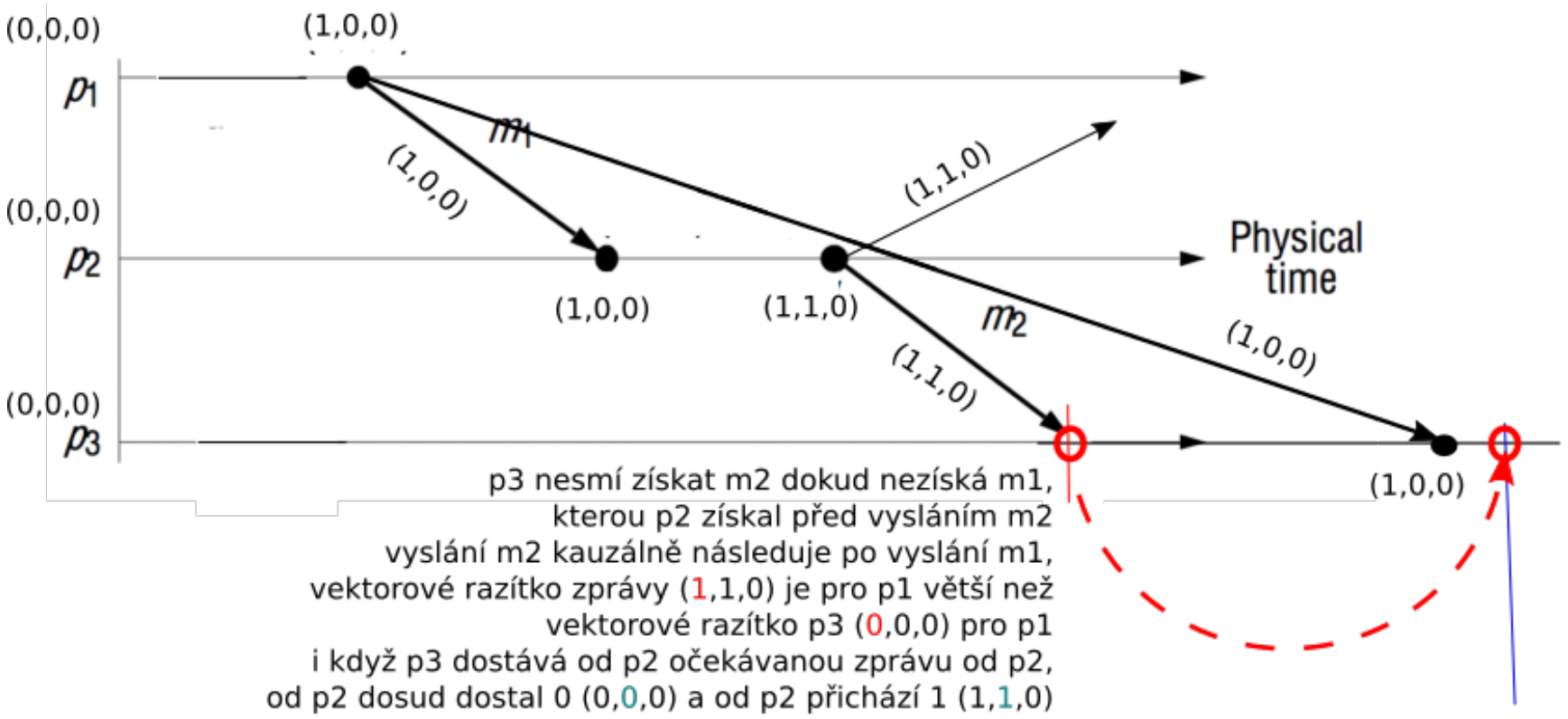
$V_i^g[j] := V_i^g[j] + 1$;

p_j získal všechny zprávy, které získal p_j do doby rozeslání této zprávy

p_i získal od p_j všechny předcházející zprávy vyslané p_j

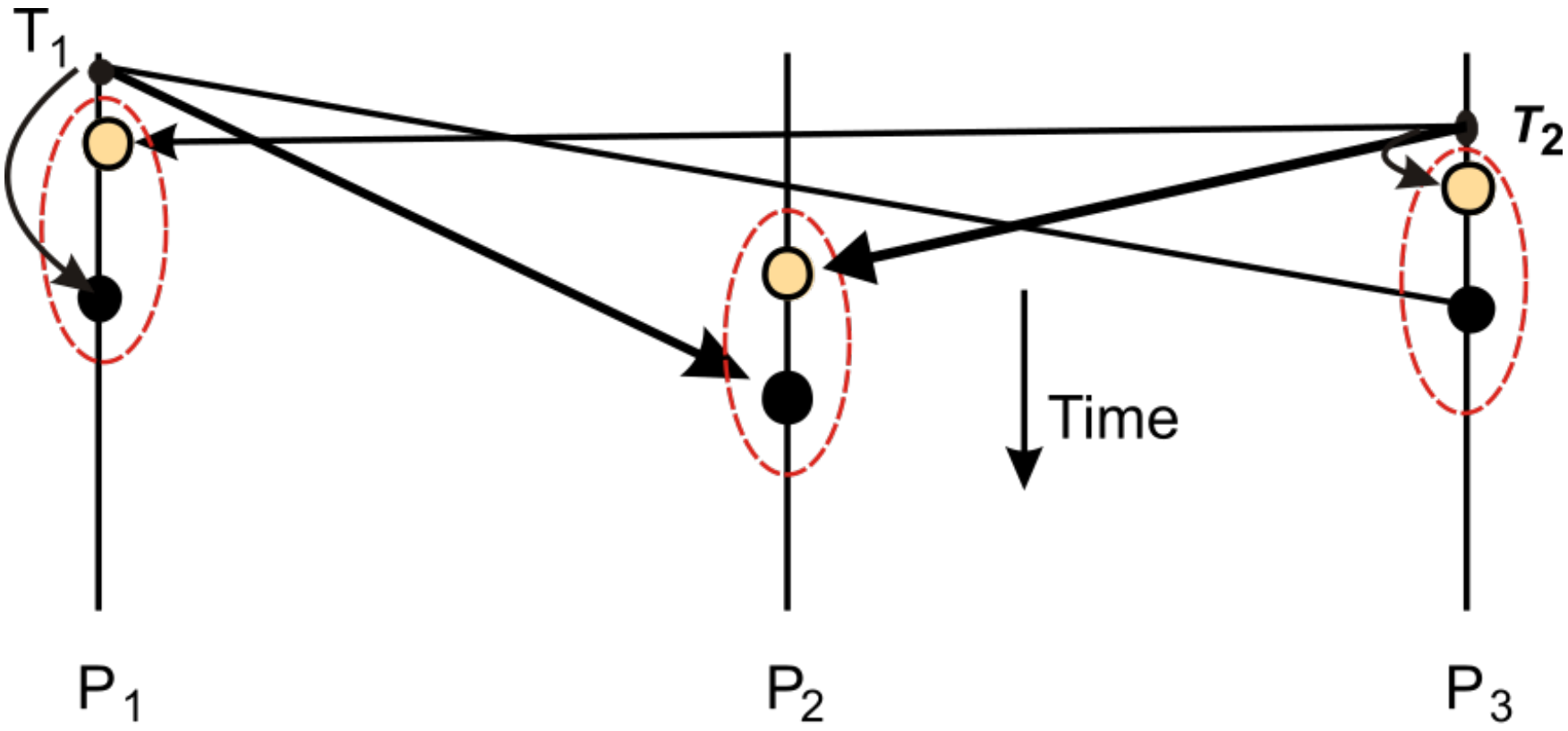
- ✓ Jestliže se použije spolehlivé rozesílání, *R-multicast*, získáme spolehlivé kauzální rozesílání

Kauzálního řazení vektorovými čas. razítky



p3 už smí získat m2 protože už získal m1,
kterou p2 získal před vysláním m2
vyslání m2 kauzálně následuje po vyslání m1
vektorové razítko zprávy (1,0,0) není pro p1 větší než
vektorové razítko p3 (1,0,0) pro p1

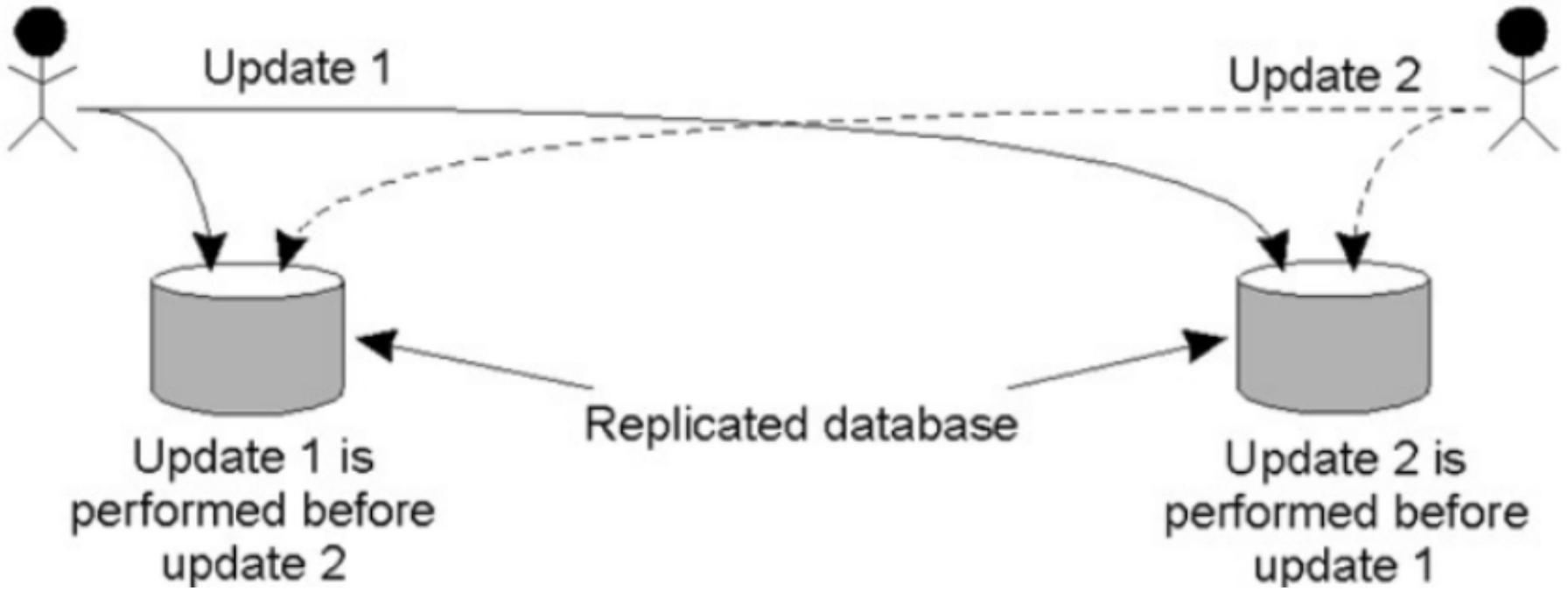
Skupinová komunikace se zachováním totálního pořadí



Totálně řazené zprávy T1 a T2

Poněvadž P1 získal zprávu T2 před zprávou T1,
získává zprávu T2 před zprávou T1
každý korektní proces ve skupině P1, P2, P3

Ilustrace dopadu nedodržení totálního pořadí



- Obě replikace databáze se ocitnou v nekonzistentním stavu

Implementace totálního řazení,

- Analogické řešení jako FIFO řazení, čítač rozesílaných zpráv
  ale musí být ve skupině procesů jedinečný
- centralizované řešení
  - ✓ DS podporuje pouze FIFO kanály
  - ✓ Zprávy rozesílá všem procesům ve skupině
    jeden centrální proces CP
  - ✓ Proces P rozesílající zprávu M pošle tuto zprávu CP
  - ✓ CP rozešle všem procesům ve skupině zprávu $\{P, M\}$

Implementace totálního řazení, *sequencerem*

- Řešení totálního řazení pomocí *sequenceru*
  - ✓ Čítač rozesílaných zpráv udržuje jeden proces ve skupině - *sequencer*
  - ✓ Každý proces, který chce rozesílat zprávu ji s jedinečným indexem rozešle procesům ve skupině a *sequenceru*
  - ✓ jedinečný index – např. id procesu + pořadové číslo zprávy v procesu
  - ✓ přijatou zprávu middleware umístí do vyrovnávací fronty
  - ✓ *sequencer* určí jemu doručené zprávě její pořadové číslo a toto číslo s indexem zprávy rozešle procesům ve skupině
  - ✓ Middleware pro každý proces ve skupině doručuje přijaté zprávy z vyrovnávací fronty v pořadí pořadových čísel
  - ✓ *Sequencer* z hlediska komunikační zátěže a spolehlivosti je úzkým profilem řešení

Implementace totálního řazení, *sequencerem*

1. Algorithm for group member p

On initialization: $r_g := 0$;

To TO-multicast message m to group g
    B-multicast($g \cup \{sequencer(g)\}$, <*m*, *i*>);

On B-deliver(<*m*, *i*>) *with* $g = group(m)$
    Place <*m*, *i*> in hold-back queue;

On B-deliver(m_{order} = <"order", *i*, *S*>) *with* $g = group(m_{order})$
    wait until <*m*, *i*> in hold-back queue and $S = r_g$;
    TO-deliver m; // (after deleting it from the hold-back queue)
    $r_g = S + 1$;

2. Algorithm for sequencer of g

On initialization: $s_g := 0$;

On B-deliver(<*m*, *i*>) *with* $g = group(m)$
    B-multicast(*g*, <"order", *i*, s_g>);
    $s_g := s_g + 1$;

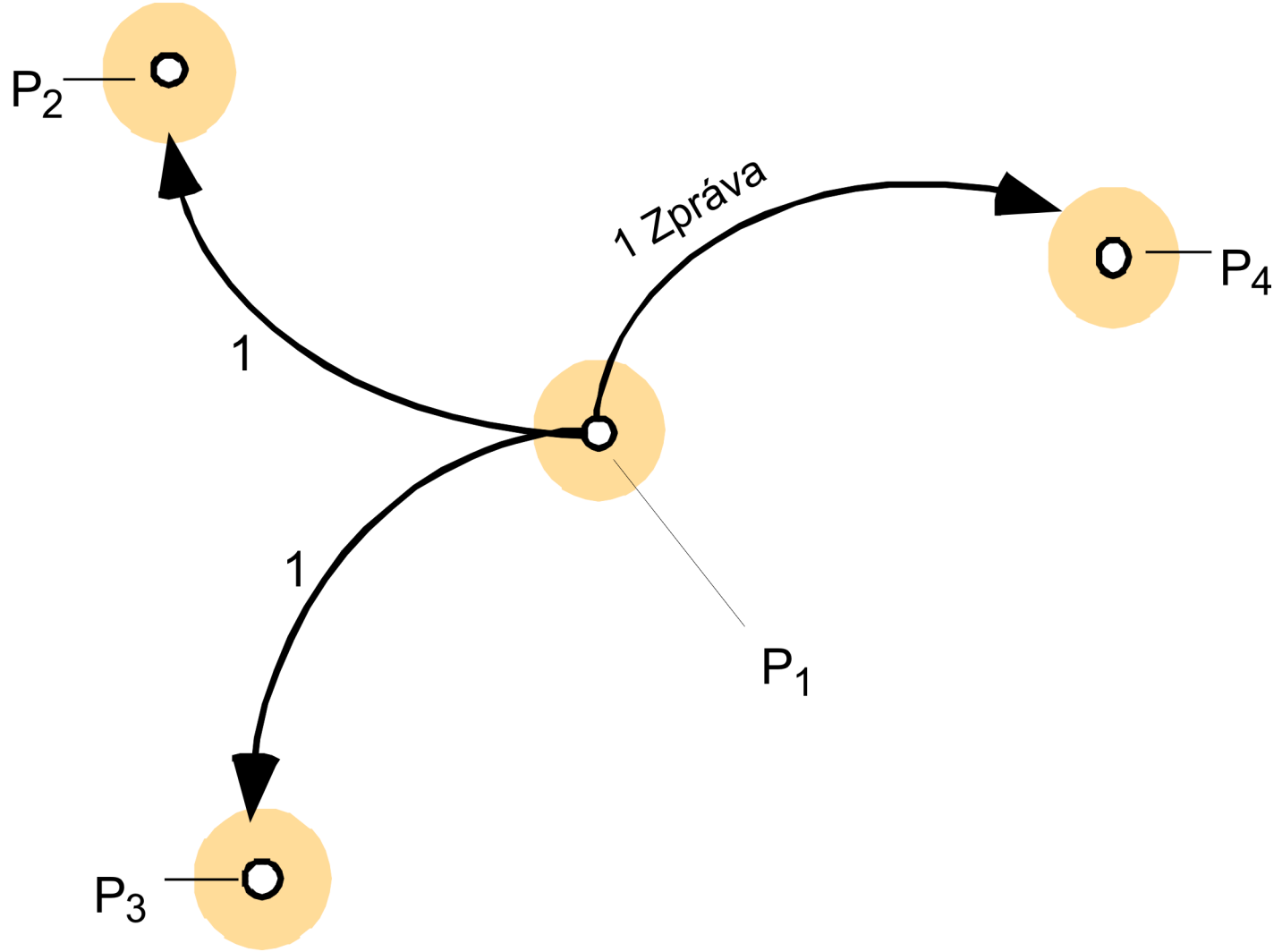
Implementace totálního řazení, *dohodou*

- Třífázový distribuovaný algoritmus
- Princip algoritmu:
  - ✓ Každý proces q si pamatuje v proměnné A_g^q nejvyšší jemu známé dohodnuté pořadové číslo zprávy některým procesem rozeslané do g
  - ✓ každý proces q vytváří a udržuje v proměnné P_g^q jím navrhované pořadové číslo zprávy
  - ✓ proces p rozešle základním rozesláním (B-*multicast*) všem procesům ve skupině jedinečně indexovanou (i) rozesílanou zprávu $< m, i >$, procesy si ji zapamatují na úrovni *receive*

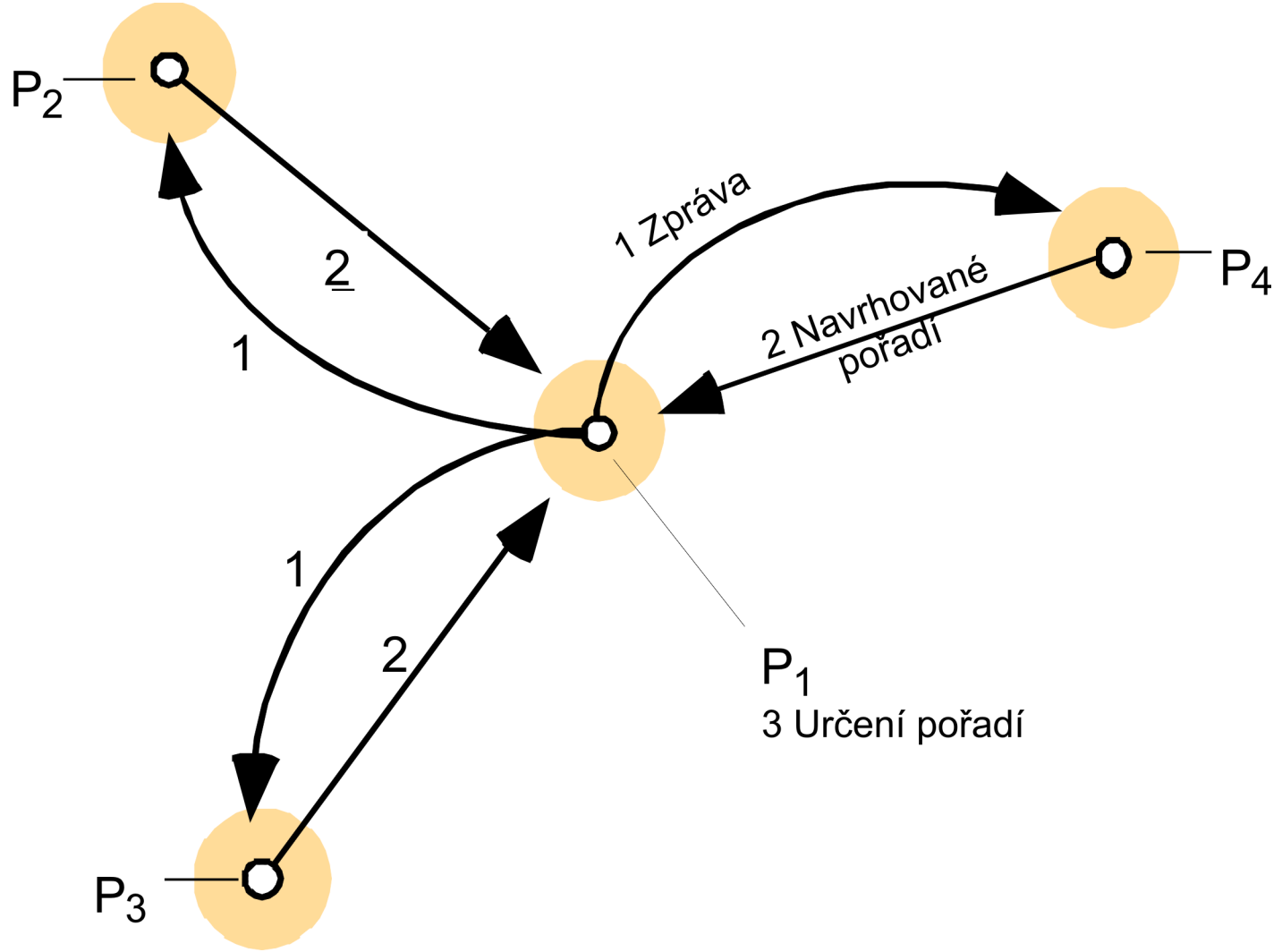
Implementace totálního řazení, *dohodou*

- ✓ proces q, který přijal zprávu $< m, i >$ vrátí procesu p
  jím navrhované pořadové číslo zprávy $< m, i >$
  jako hodnotu $P_g^q = max(A_g^q, P_g^q) + 1$
- ✓ proces p vybere největší navrhované číslo zprávy $< m, i >$,
  označme je a, a
  toto základním rozesíláním rozešle všem procesům skupiny jako
  dohodnuté pořadové číslo zprávy $< m, i >$
- ✓ každý proces q ve skupině si nastaví $A_g^q = max(A_g^q, a)$
- ✓ zpráva $< m, i >$ je pak doručena každému procesu ve skupině
  v takto dohodnutém pořadí

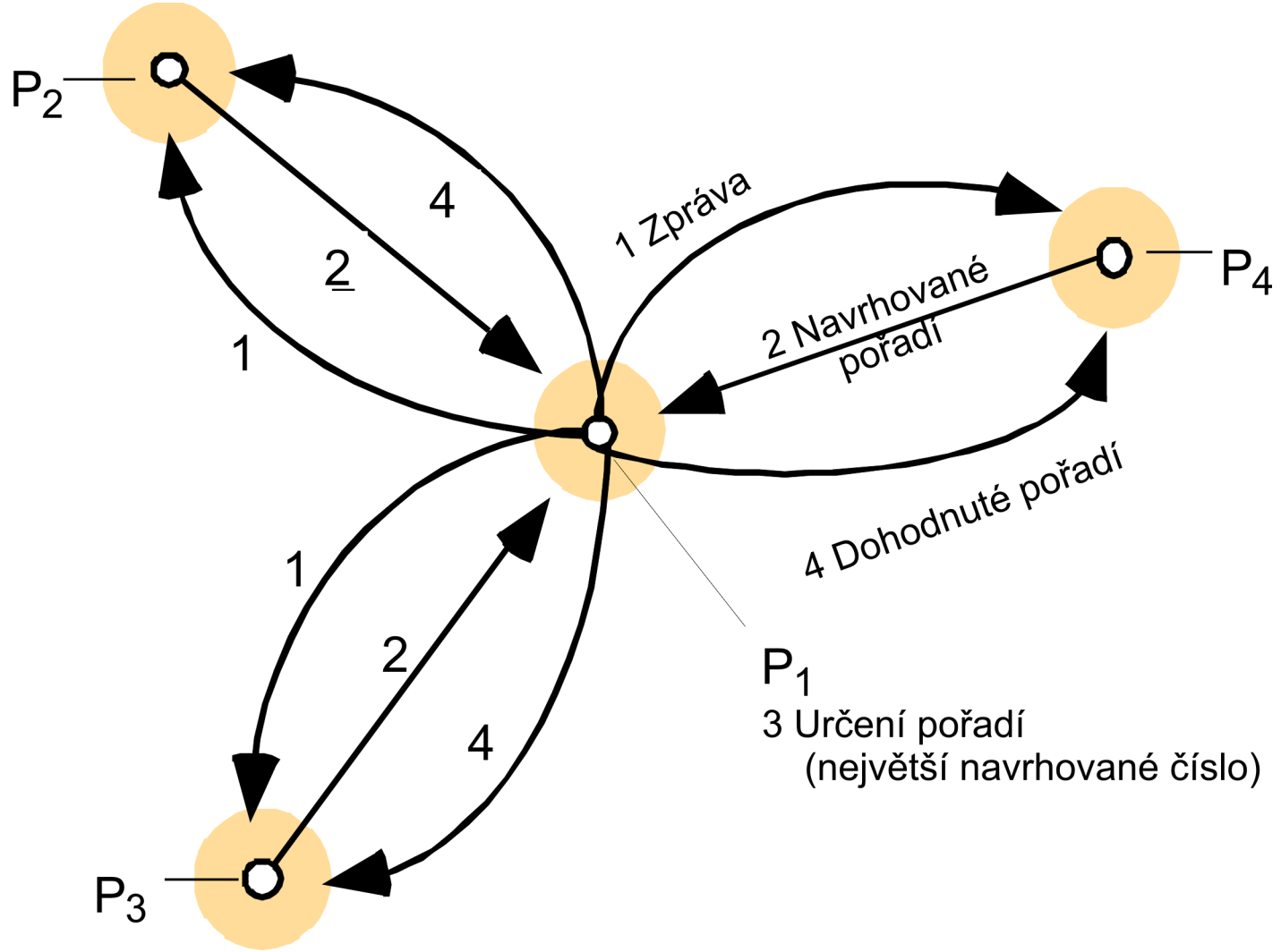
Implementace totálního řazení, *dohodou*



Implementace totálního řazení, *dohodou*



Implementace totálního řazení, *dohodou*



Implementace totálního řazení, *dohodou*