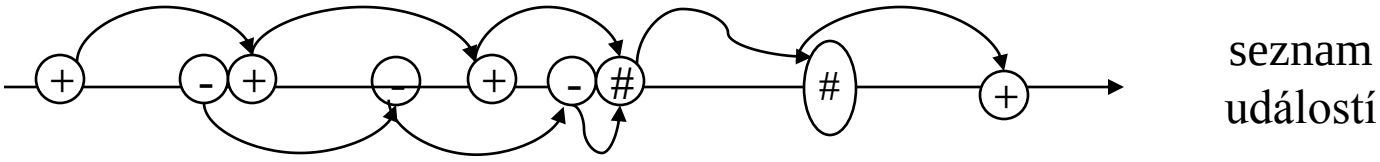
Paralelní simulace

Kvaziparalelní prostředí - charakteristika:

- přísně sériový výpočet řízený seznamem událostí (SQS)
- splynutí fází (událostí) všech procesů do jediné sekvenční posloupnosti

poznámka: ne vždy jsou události v SQS vázány vztahem příčina - následek

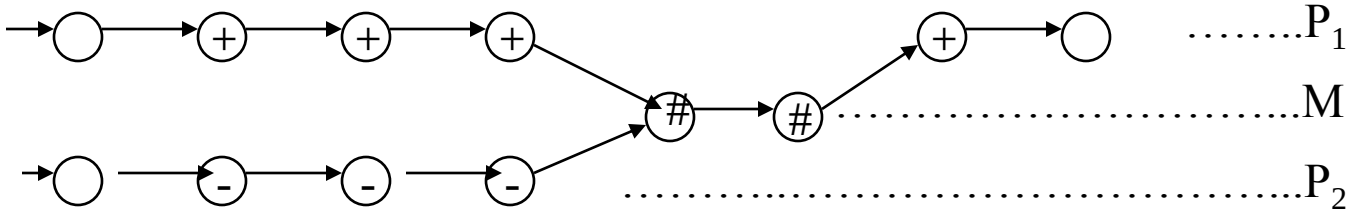


příklad: uvažujme 3 samostatné procesy:

P_1……průchod polotovarů typu 1 výrobní halou

P_2…….průchod polotovarů typu 2 výrobní halou

M…….proces sdílené obsluhy (montáže) obou typů



Synchronizační strategie

Poznámky:

- vlastní paralelizace problému - vyplývá z definice paralelně probíhajících procesů na simulovaném systému ,
- mapování procesů na procesory - závisí na vzájemném počtu procesů a procesorů: při velkém počtu dynamicky vznikajících procesů nutno předpokládat kvaziparalelní sdílení jednoho procesoru více procesy (dále pro jednoduchost předpokládejme, že každý proces je prováděn na jiném procesoru),
- nutná synchronizace všech zůčastněných procesorů ,
- výsledný efekt paralelně prováděného programu musí být stejný jako efekt jeho provedení v kvaziparalelním prostředí; toho lze dosáhnout pokud výsledný efekt každého procesu respektuje příčinné závislosti

Výchozí předpoklady pro budování paralelního prostředí :

- paralelní výpočet řízený více seznamy událostí,
- jednotlivé části (procesory) navzájem komunikují pomocí zpráv,

Synchronizační strategie paralelní simulace

Dvě základní skupiny algoritmů synchronizačních strategií:

- konservativní synchronizační algoritmy

  (metoda nulových zpráv (Chandy - Misra - Bryant), metoda uváznutí a zotavení procesů, synchronní metoda)
  - přísně respektují příčinné závislosti (local causality constraint),
  - jejich algoritmy jsou založeny na rozlišení tzv. bezpečných událostí (neexistuje možnost pozdějšího výskytu dalších událost s menší hodnotou časové známky.

- optimistické synchronizační algoritmy

  (metody Time Warp)
  - nerespektují pravidla příčinných závislostí, ale detekují jejich porušení,
  - pomocí zpětných běhů (tj. návratů k menším hodnotám modelového času) anulují efekty všech předčasně provedených událostí, pak pokračují novým dopředným během.

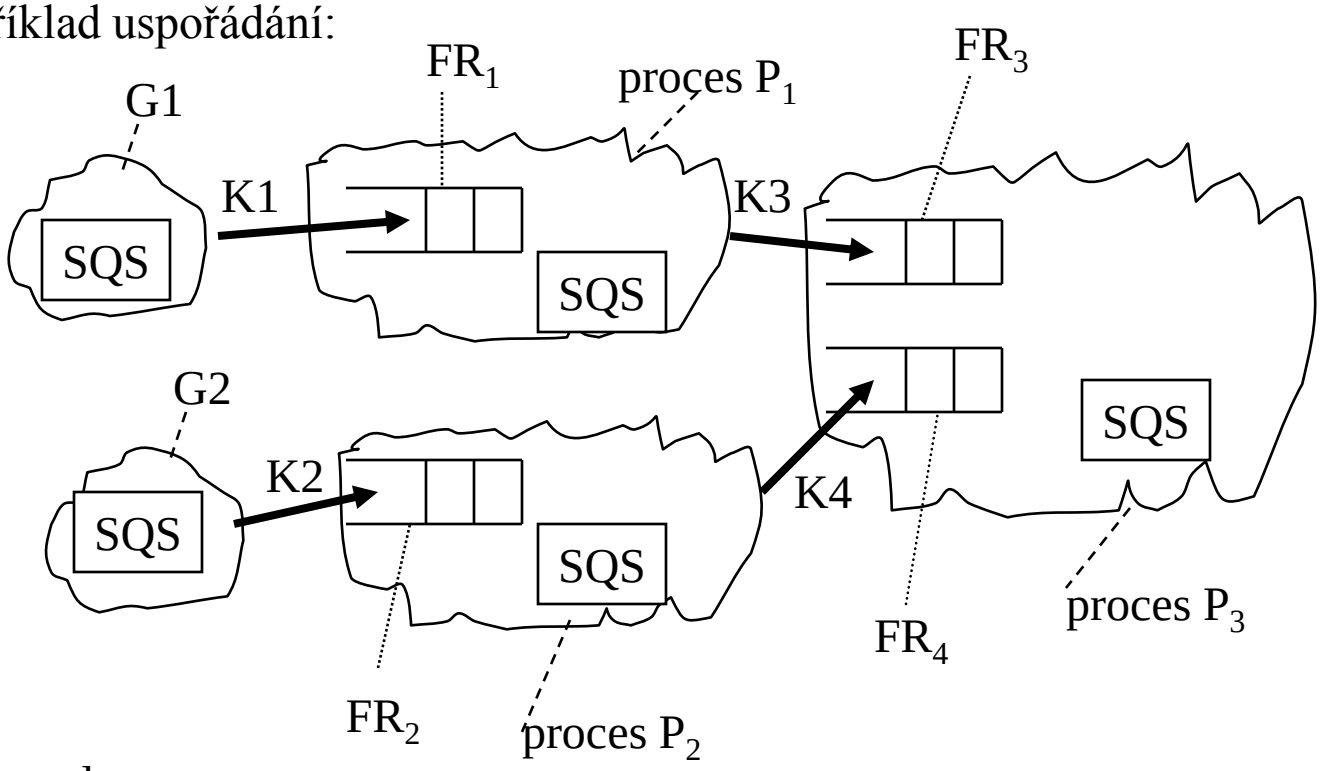
Konservativní algoritmy

předpoklad:

- mezi procesy existují statické komunikační kanály,
- posloupnost zpráv přenášených na každém spoji tvoří z hlediska časových známek neklesající posloupnost,
- mezi-procesorové komunikační služby jsou spolehlivé a zachovávají pořadí přenášených zpráv..

Synchronizace ve struktuře bez zpětných vazeb

příklad uspořádání:



legenda:

G1,G2…….procesy - generátory požadavků,

K1,..,K4 ….komunikační kanály plnící vstupní fronty zpráv FR_1,..,FR_4,

SQS…….lokální seznamy událostí (pro každý procesor).

Konservativní algoritmy

- posloupnost událostí z předchozího příkladu:

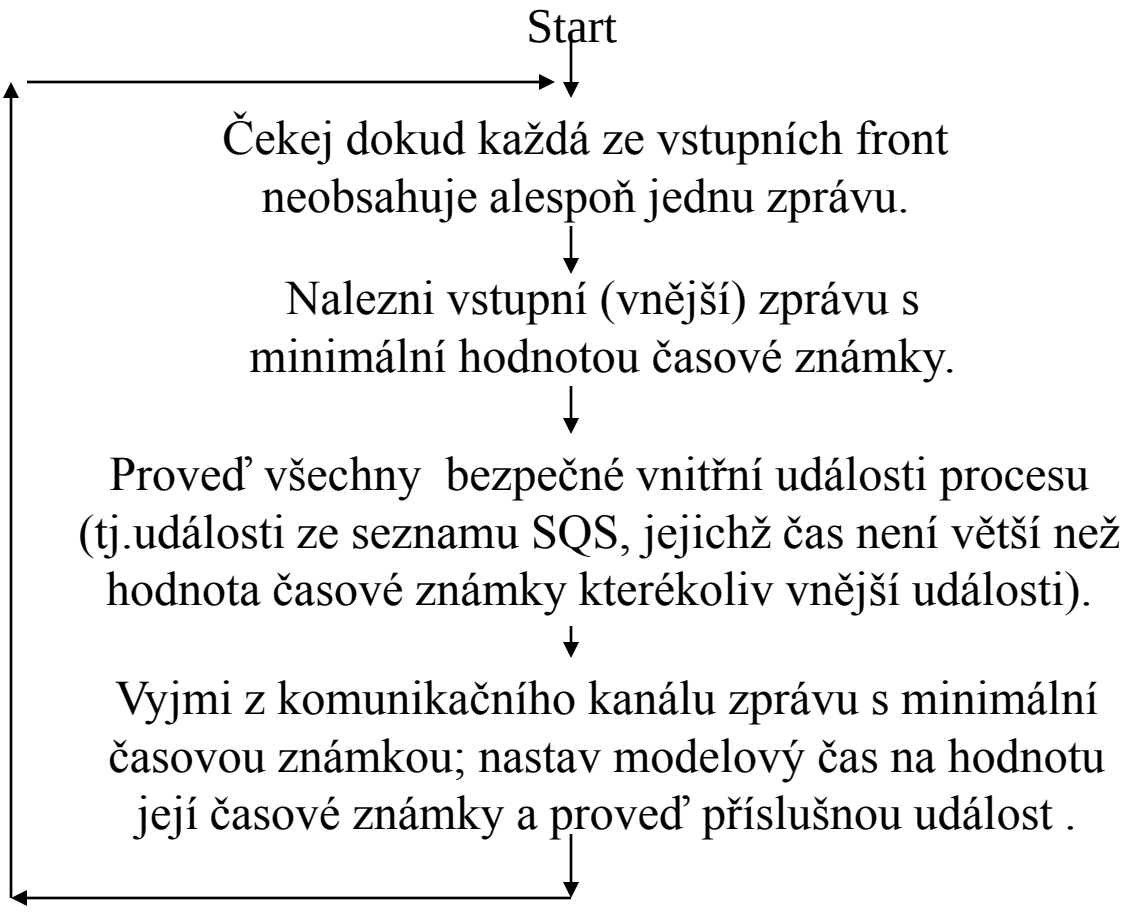
procesy G1, G2: nejsou závislé na žádném jiném procesu => jsou omezeny v podstatě pouze velikostí front FR_1 a FR_2.

procesy P_i (i = 1,2) : bezpečné události procesu P_i leží v intervalu:

< okamžitá hodnota modelového času procesu P_i, hodnota časové známky poslední události ve frontě FR_i >

proces P_3: interval bezpečných událostí: <okamžitá hodnota modelového času procesu P_3, min (hodnota poslední časové známky v FR_3, hodnota poslední časové známky ve frontě FR_4) >

Algoritmus 1: řízení událostí v procesoru:



Konservativní algoritmy

Struktury se zpětnou vazbou: možnost uváznutí procesů:

- vzájemnou komunikaci procesů reprezentujme orientovaným grafem (uzly grafu representují procesy, hrany representují komunikační kanály),
- jestliže se v orientovaném grafu vyskytuje cyklus a jestliže v každém uzlu výše zmíněného cyklu existuje alespoň jedna prázdná fronta => při použití algoritmu 1 dojde k uváznutí (každý proces cyklu čeká na nějakou zprávu)

Opatření zabraňující zablokování:

- Každý proces generuje výstupní zprávy určené jiným procesům pokud možno předvídavě (příslušné časové známky jsou větší než okamžitá hodnota modelového času vysílacího procesu).
- Pokud nelze určit časovou známku příští výstupní zprávy procesu P_i , pak proces P_i predikuje hodnotu nejmenšího časového intervalu $Tmin_i$, během něhož nebude požadovat na jiném procesoru provedení nějaké jím specifikované události (tj. příští zpráva vyslaná tímto procesem nemůže mít menší hodnotu časové známky než je $time_i + Tmin_i$). Tato predikční schopnost procesu vyplývá z jeho sémantiky a hodnota Tmin se označuje jako předstih (lookahead) daného procesu; touto svou vlastností umožňuje proces P_i ostatním procesům (které přijímají jeho zprávy) pokračovat v provádění jiných událostí až do času $time_i + Tmin_i$).

Metoda nulových zpráv

Opatření zabraňující zablokování (pokračování):

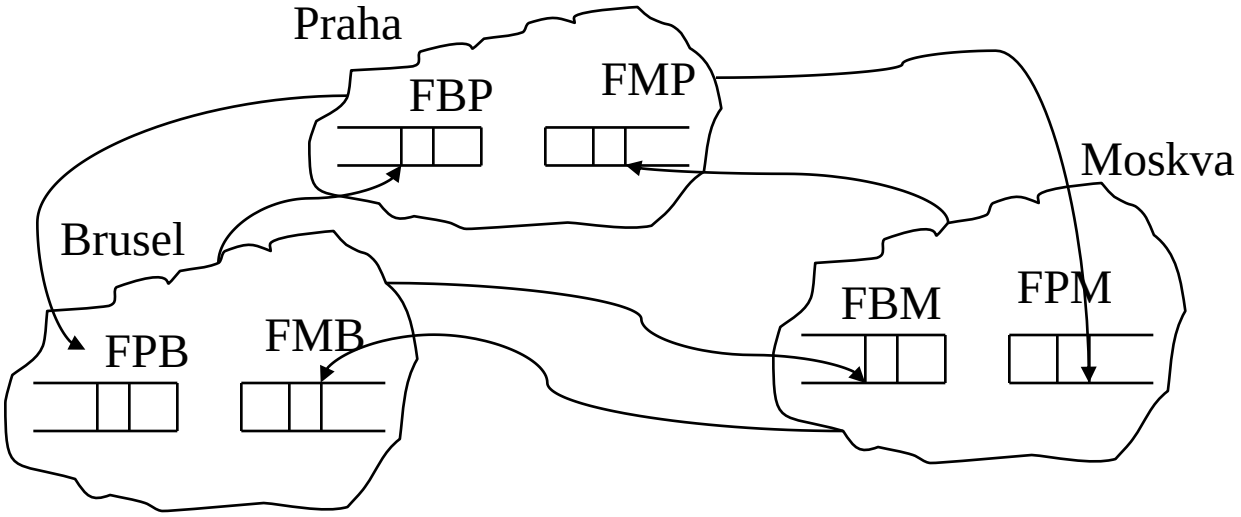
- Pro šíření hodnot předstihů mezi procesory se používá tzv. mechanismus nulových zpráv (null messages). Nulové zprávy nerepresentují žádnou skutečnou událost; definují však horní mez modelového času do níž může pokročit simulace procesů na daném procesoru aniž by riskovala přijetí dalších zpráv a v důsledku toho narušení neklesající posloupnosti přidružených časů. Možnosti odesílání nulových zpráv:
  - do všech výstupních směrů na začátku simulace a po každém přijetí jiné nulové zprávy nebo normální zprávy, která nemá za následek výstup nějaké jiné normální zprávy (velký počet zpráv),
  - blokovanému procesoru na jeho žádost (delší časové zdržení).
- Hodnota předstihu procesu se může měnit v průběhu simulace a má vliv na efektivitu simulačního algoritmu; volba této hodnoty se provádí dle odhadu fyzikálních vlastností simulovaného objektu v daném procesoru (zpoždění logického členu, doba obsluhy v SHO, atd.
- Mechanismus nulových zpráv selhává v případě výskytu cyklu ve kterém všechny objekty vykazují nulovou hodnotu předstihu.

Demonstrace použití nulových zpráv

Příklad: letecká doprava mezi městy Praha, Brusel, Moskva

předpoklad:

- existují lety v obou směrech mezi každými dvěma letišti,
- simulační program sestavíme pomocí tří paralelně prováděných procesů (každý pro popis událostí na letištích v uvedených městech)



- okamžitá situace:

Praha: FMP: 7.15, 10.00 (plánované přílety);

FBP: 0; poslední přílet :5.00 => modelový čas = 5.00

Moskva: FPM: 0, poslední přílet : 6.00 => modelový čas = 6.00

FBM: 8.30,10.00 ;

Brusel: FMB: 0, poslední přílet :5.30 => modelový čas = 5.30

FPB: 6.00, 7.00, 11.00,

Demonstrace použití nulových zpráv

Pokračování příkladu: předstih = minimální doba přeletu mezi kterýmikoliv městy: 1.30 hod.

čas nulových zpráv z Prahy: 5.00 + 1.30 = 6.30

čas nulových zpráv z Bruselu: 5.30 + 1.30 = 7.00

čas nulových zpráv z Moskvy: 6.00 + 1.30 = 7.30

Situace po přijetí nulových zpráv:

Praha: pokračování simulace na intervalu 5.00 až 7.00

Moskva: pokračování simulace na intervalu 6.00 až 6.30

Brusel: pokračování simulace na intervalu 5.30 až 6.30

Poznámky:

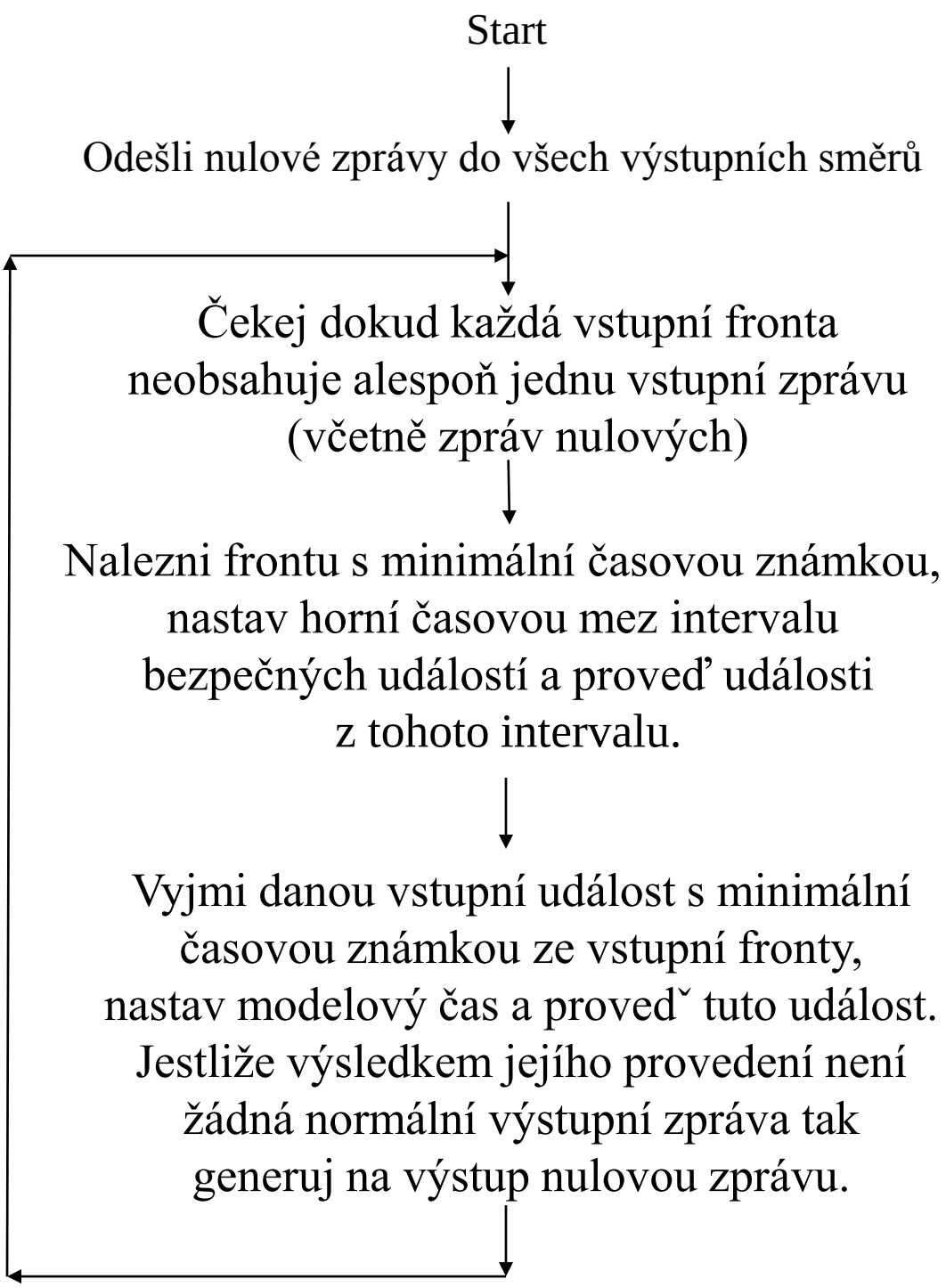
a) v uvedených „bezpečných“ intervalech (vymezených nulovými zprávami) existují nějaké naplánované vnitřní události (např. odlety letadel) => jejich provedení => případné naplňování front

b) neexistují vnitřní události: =

- Brusel přijme vstupní zprávu pro čas 6.00 (přílet) a odešle nulové zprávy s časem 7.30,
- Praha přijme vnější zprávu v 7.15 a odešle nulové zprávy s časem 8.45,
- Brusel přijme vstupní zprávu pro čas 7.00 (přílet) a odešle nulové zprávy s časem 8.30,
- Moskva přijme vnější zprávu v 8.30 a odešle nulové zprávy s časem 10.00,
- ……..

Null message algorithm

Algoritmus 2: (Chandy -Misra- Bryant):



Metoda uváznutí a zotavení procesů

- jde o další z konservativních strategií, která je založena na původním algoritmu 1, ale bez použití nulových zpráv

Princip: v průběhu simulace dochází ke střídání dvou fází:

- provádění simulace a s průběžnou detekcí uváznutí simulačních procesorů (shromažďováním vhodných informací v průběhu simulace),
- zotavení z uváznutí (pomocí řídícího procesu - efekt: uvolnění nejpomalejšího procesoru).

Zotavení procesů z uváznutí:

- řídící proces vysílá všem simulačním procesorům žádost o vrácení hodnoty času první z dosud neprovedených událostí,
- simulační procesor vrací vyžádané časové hodnoty řídícímu procesu,
- řídící proces provede výpočet minimální hodnoty; tím určí simulační procesor s minimální okamžitou hodnotou modelového času a zajistí jeho uvolnění,
- uvolnění nejpomalejšího simulačního procesoru umožní jeho další postup v čase což následně způsobí uvolnění jiných simulačních procesorů.

Metoda uváznutí a zotavení procesů

Simulace s průběžnou detekcí uváznutí: metoda je založena na dynamickém vytváření a udržování grafu reprezentujícího nezablokované simulační procesory:

- uzly grafu reprezentují zotavené procesory,
- hrany vycházející z určitého uzlu reprezentují zprávy, které byly z odpovídajícího procesoru odeslány s cílem zotavit nějaký jiný procesor,

Princip:

- každý nově zotavený procesor se začlení do stromu zotavených procesorů tím způsobem, že vytvoří list k procesoru od něhož obdržel zotavovací zprávu,
- simulační procesor po zotavení provádí vnitřní a vstupní události a generuje výstupní zprávy ostatním procesům (tím zotavuje tyto zablokované procesory); takto simulační procesor pokračuje až do vyčerpání některé své vstupní fronty,
- zablokovaný procesor (tj. procesor s některou prázdnou vstupní frontou), který je listem (nemá ve stromě nezablokované následovníky), je ze stromu vyjmut (tj. je zablokovaná celá větev),
- v průběhu simulace narůstá strom zotavených procesorů (vlivem odesílání zpráv a zotavováním procesorů) nebo degeneruje (vlivem zablokování procesorů),
- pokud řídící proces je listem (tj. bez nezablokovaných následovníků) je zablokován celý simulační výpočet .

Synchronní metoda

Princip: v průběhu simulace jsou postupně vymezovány tzv. bezpečné zóny; jde o časové úseky ve kterých jednotlivé procesy nemohou být vzájemně ovlivňovány.

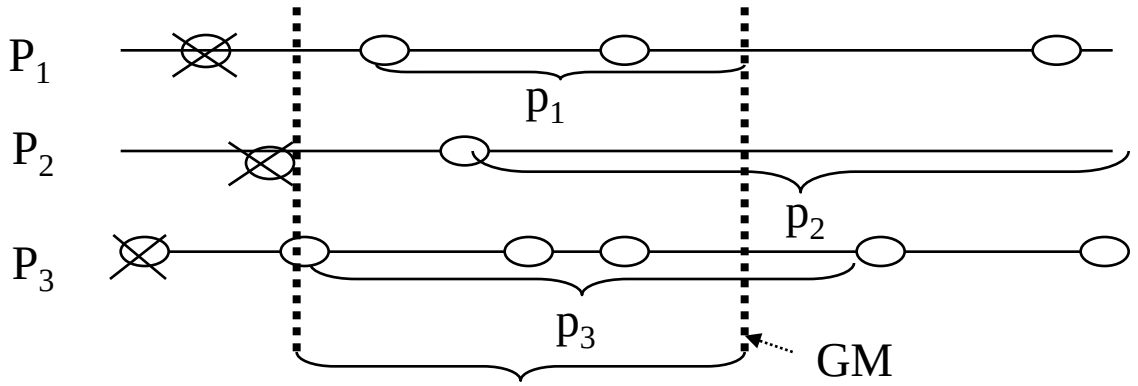
$t\_next\_ev_i$……čas nejbližší dosud neprovedené události procesu P_i,

p_i……...hodnota predikce procesu P_i (lokahead),

$t\_next\_ev_i + p_i$….lokální minimum (procesu P_i); nejmenší možná hodnota časové známky příští výstupní zprávy vyslané procesem P_i,

$GM = \min (t\_next\_ev_0 + p_0, \ldots\ldots\ldots\ldots, t\_next\_ev_n + p_n)$

……globální minimum: nejmenší hodnota ze všech lokálních minim



globální bezpečná zóna

……*záznam dosud neprovedené události*

……*bývalá pozice záznamu poslední provedené události*

synchronizace procesů v jednotlivých krocích - pomocí centrální či distribuované bariéry bariéry

Synchronní metoda

Implementace centrální bariéry :

- řídící proces dostává od jednotlivých procesorů zprávy informující o dosažení bariery a obsahující hodnoty lokálních minim modelového času,
- po obdržení všech zpráv řídící proces vypočte novou hodnotu globálního minima a vyšle všem procesorům zotavovací zprávu obsahující největší časovou hodnotu bezpečné zóny.

Implementace distribuované bariéry:

- není třeba centrální řídící proces, jeho roli hraje jeden z procesů,
- všechny procesory jsou organizovány ve formě vyváženého (např. binárního) stromu,
- listy stromu odesílají svému otci zprávy o dosažení bariéry spolu s hodnotou jejich lokálního minima,
- každý vnitřní uzel dostává zprávy od všech svých „dětí“ o dosažení bariéry spolu s hodnotou lokálního minima „příslušných větví“,
- pokud vnitřní uzel dosáhl bariéry a obdržel zprávy od všech svých dětí tak vypočte dílčí hodnotu minima (jde o minimum vypočtené z lokálních minim dětí včetně svého lokálního minima) a odesílá zprávu svému otci,
- všechny procesy dosáhly bariéry pokud kořen stromu dosáhl bariéry a zároveň obdržel zprávy od všech svých dětí,
- kořen stromu vypočte hodnotu globálního minima (horní mez společné bezpečné zóny) a tuto rozešle spolu s uvolňovací zprávou svým dětem; tyto předávají tuto zprávu dále svým dětem atd.

Modifikace synchronní metody

- řídící proces stanoví pro jednotlivé procesy individuální zóny,
- přesnější určení bezpečnostních zón = efektivnější využití paralelismu.
- příklad: letecká doprava Praha, Brusel, Tokio
  - okamžité časy - Praha:12.00, Brusel:12.15, Tokio:13.00,
  - nejkratší doba přeletu: 1.30 hod. =>

    globální minimum = 12.00 + 1.30 =13.30 hod.,
  - pro Tokio zbytečně velké omezení.

- individuální zóny se stanoví na základě „**vzdáleností mezi procesy“**

  p_{ij}……..hodnota predikce procesu P_i vzhledem k procesu P_j,

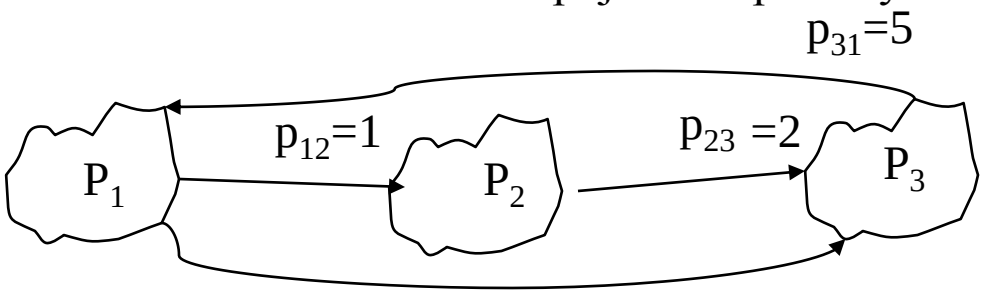
  $dc_{mn}(i)$…..délka i-té cesty mezi P_m a P_n (součet predikcí na i - té cestě z P_m do P_n),

  d_{mn}……...vzdálenost mezi procesy m, n (min ($dc_{mn}(i)$) přes všechny cesty z P_m do P_n),

- zjišťování vzdáleností d (rozšíření původní funkce řídícího procesu):
  - řídící proces vyšle žádost všem ostatním procesům o zaslání hodnot jejich predikcí a komunikačních spojů;
  - po obdržení odpovědí udržuje matici vzdáleností mezi jednotlivými procesy (nutno provádět dynamicky při změně hodnot predikcí)

Modifikace synchronní metody

- příklad: možné komunikační spoje mezi procesy



- $p_{13}=6$
  - matice vzdáleností pro spojení z předešlého příkladu:

$$\begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{ccc} P_1 & P_2 & P_3 \end{array} \\ \begin{pmatrix} \min(p_{13}, p_{12}+p_{23}) + p_{31}, & p_{12}, & \min(p_{13}, p_{12}+p_{23}) \\ p_{23} + p_{31}, & p_{23}+p_{31}+p_{12}, & p_{23} \\ p_{31}, & p_{31}+p_{12}, & p_{31}+\min(p_{13}, p_{12}+p_{23}) \end{pmatrix} \end{array}$$

$$= \begin{pmatrix} 8 & 1 & 3 \\ 7 & 8 & 2 \\ 5 & 6 & 8 \end{pmatrix}$$

$time_i$…….čas nejbližší dosud neprovedené události v procesu P_i

IMP_j…….individuální minimum (dolní mez časové známky pro zprávy přijaté procesem $P_j = \min(time_j + d_{ij})$)

Příklad: možná ovlivňování procesu P_3:

nechť t_1, resp. t_2 je hodnota první dosud neprovedené události v procesu P_1, resp. P_2.

- vliv P_1 na P_3: nejdříve v čase $t_1 + d_{13} = t_1 + 3$
- vliv P_2 na P_3: nejdříve v čase $t_2 + d_{23} = t_2 + p_{23} = t_2 + 2$

Modifikace synchronní metody

- příklad: použití předešlé matice
  - nechť $time_1 = 71$, $time_2 = 72$, $time_3 = 73$ ….nejmenší časové hodnoty dosud neprovedených událostí jednotlivých procesů),
  - lokální minima (nejmenší hodnoty časových známek výstupních zpráv):
    - $LMP_1 = 71 + 1 = 72$
    - $LMP_2 = 72 + 2 = 74$
    - $LMP_3 = 73 + 5 = 78$
  - globální minimum (globální bezpečná zóna): 72
    (viz standardní synchronní metoda),
  - individuální minima (bezpečné zóny) jednotlivých procesů:
    $IMP_1 = \min (time_1 + 8, time_2 + 7, time_3 + 5) = 78,$
    $IMP_2 = \min (time_1 + 1, time_2 + 8, time_3 + 6) = 72,$
    $IMP_3 = \min (time_1 + 3, time_2 + 2, time_3 + 8) = 74,$

- poznámky:
  - další modifikace: zúžení individuálních bezpečných zón na velikost okna T_w (reprezentující „rozumnou vzdálenost“) pro zanedbání vlivu „vzdálených procesů“ => redukce režijní doby potřebné pro meziprocesorovou komunikaci,
  - $IMP_i = \min (time_j + d_{ji})$ pro všechna j pro která $d_{ji} <= T_w$

Konservativní metody - dodatek

Další potíže při paralelní simulaci:

- změny mezi-procesorových komunikačních vazeb v průběhu simulace
  - na začátku simulace vytvořit všechny možné procesy a možné vazby mezi nimi = malé efektivita dusící paralelismus,
  - dynamické vytváření procesů a komunikačních kanálů v průběhu simulace.

- reprodukovatelnost paralelního výpočtu
  - předpoklad: reprodukovatelnost výpočtu dílčích událostí
  - závisí na pořadí vyhodnocení „současných událostí“ (událostí se stejnou hodnotou časové známky) - toto pořadí musí být zachováno při každém opakovaném paralelním výpočtu
  - opatření: rozšíření časové známky o nižší skryté řády (sloužící k rozlišení současných událostí) a o automatické přiřazení hodnoty těmto řádům

Konservativní metody - dodatek

Vnitřní struktura časové známky:

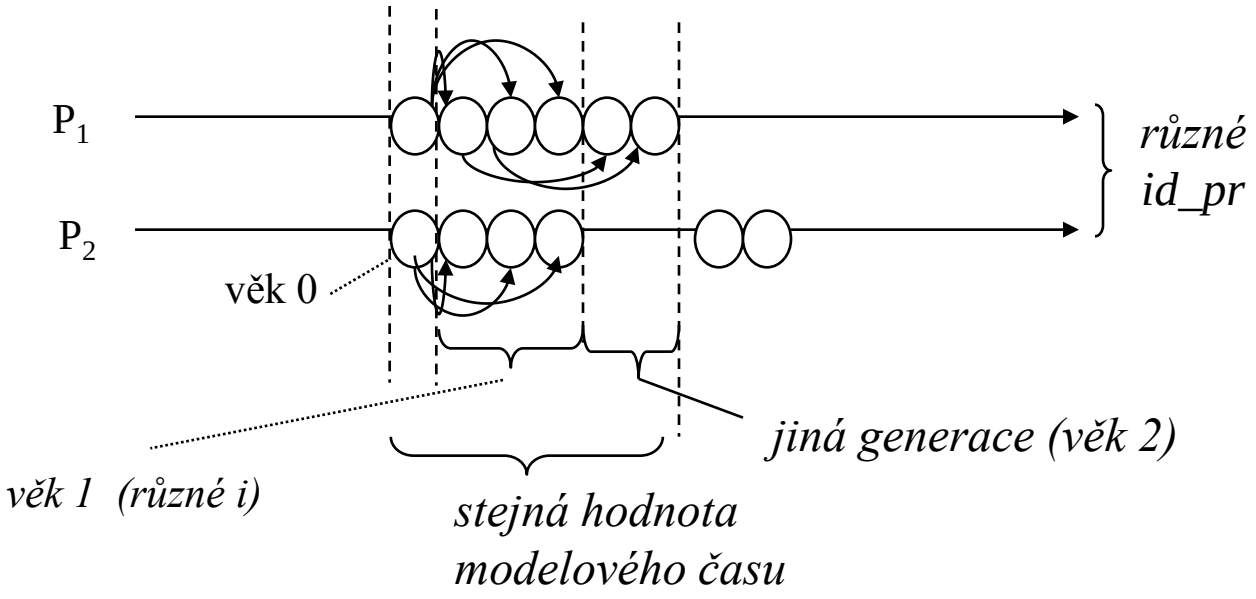
hodnota modelového času	věk	id_pr	i

věk…….začíná od nuly a vytváří posloupnost generací událost (v případě generování událostí se stejnou časovou známkou

id_pr : i…..rozliší příslušníky jedné generace se stejným věkem

id_pr……..identifikace procesu

i……pořadové číslo události generované procesem id_pr



Optimistické synchronizační přístupy

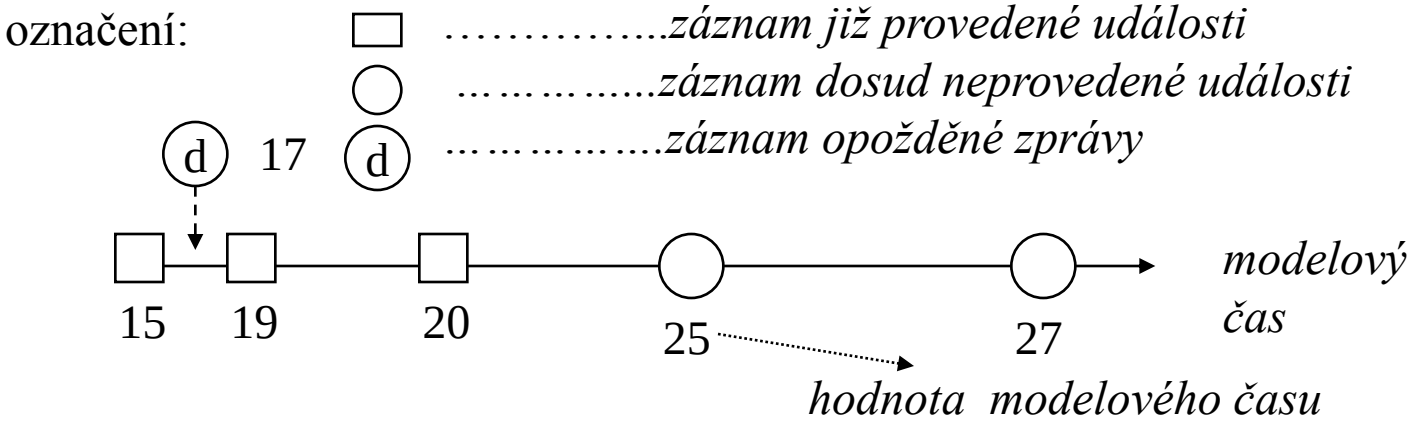
- dovolují porušit závislosti typu příčina - následek„
- ve srovnání s konservativními přístupy jde o robustnější algoritmy,
- chybí prvotní synchronizace, procesy nerozlišují bezpečné či jiné události; jsou vybavené mechanismy pro eliminaci vlivu chybně provedených událostí,
- 1. algoritmus tohoto typu: Time warp („deformace času ve smyslu návratu do historie“),
- přístupy optimistické povahy používané v jiných oblastech:
  - proudové zpracování -pipelining (optimistický výběr instrukcí),
  - přístupová metoda CSMA/CD (optimistický přístup ke sběrnici - kolise),
  - distribuované databáze (optimistické zahájení transakcí - detekce konfliktu).

Předpoklady optimistických metod :

- procesy komunikují prostřednictvím předávání zpráv po spolehlivých spojích,
- komunikační kanály nemusí zachovávat pořadí zpráv,
- jednotlivé procesy nemusí vysílat zprávy uspořádané dle hodnot časových známek,
- synchronizační strategie nejsou závislé na stavu vstupních komunikačních kanálů mezi procesory,
- neplatí žádné omezení o cyklech procesů s nulovou hodnotou předstihu.

Základní charakteristika metody Time warp

- sekvenční i paralelní konservativní přístupy: likvidují po provedení událostí příslušné záznamy,
- metody Time warp: musí zachovat do určité „hloubky“ i historii (kvůli zpětným běhům),
- důvod návratu procesu v čase: obdržení zprávy (vnější události) s menší hodnotou časové známky než je okamžitá hodnota modelového času daného procesu.



Důsledky přijetí opožděné zprávy s časovou známkou 17:

- inicializace„zpětného“ běhu:
  - pro obnovení stavu procesu, který existoval po provedení události v čase 15 (eliminuje vlivy již provedených událostí v časech 19, 20 na lokální proměnné daného procesu)
  - pro eliminaci vlivu zpráv již odeslaných jiným procesům v průběhu provádění událostí v časech 19, 20 (rozesláním eliminujících zpráv – anti-messages)
- inicializace opravného „dopředného“ běhu
  - pro případné nové provedení událostí v časech 17, 19 a 20

Základní charakteristika metody Time warp

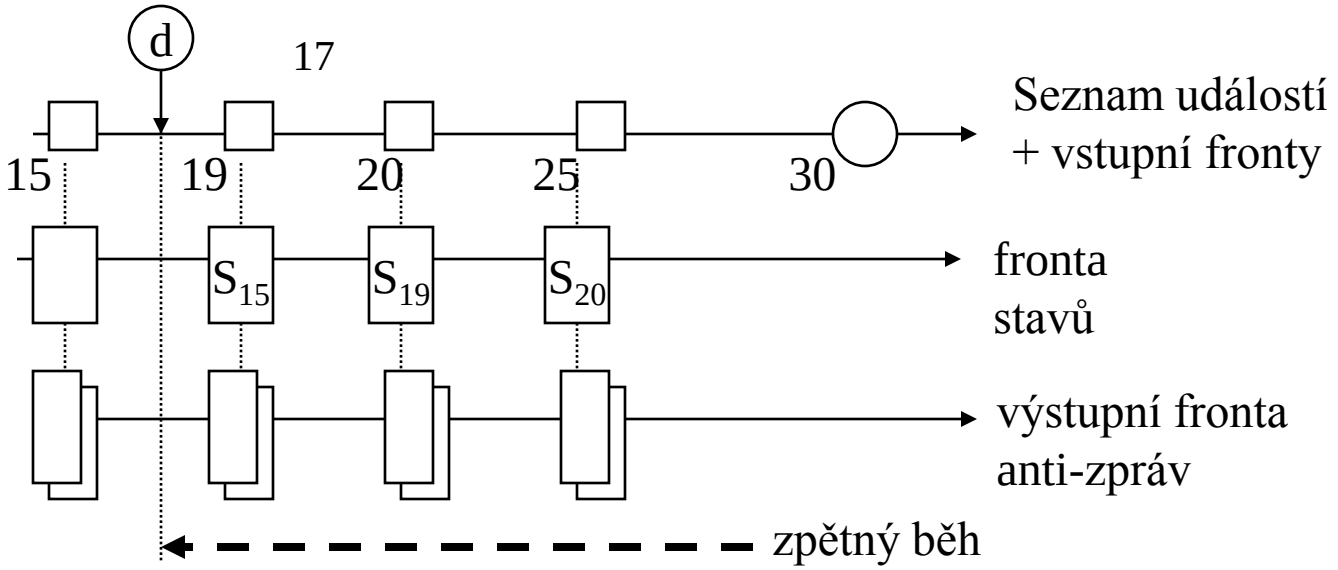
Základní lokální funkce procesů:

a) obnovení původních hodnot stavových proměnných procesu (rolling back state variables)

- před provedením každé události se vytvoří záznam s kopiemi všech stavových proměnných (copy state saving) nebo se vytvoří záznam s adresami modifikovaných stavových proměnných (incremental state saving) a tento se spolu s příslušnou hodnotou modelového času uloží do fronty stavů
- zmíněná fronta stavů umožní návrat procesu do jeho historie

b) eliminace vlivu odeslaných zpráv

- kromě každé odesílané výstupní zprávy proces automaticky vytváří druhou identickou zprávu s nastaveným příznakem „anti“; v okamžiku odeslání původní zprávy je příslušná anti-zpráva vložena spolu s její časovou známkou do fronty anti-zpráv,
- zmíněné anti-zprávy jsou odesílány procesem při zpětném běhu za účelem eliminace jim přidružených původních zpráv.



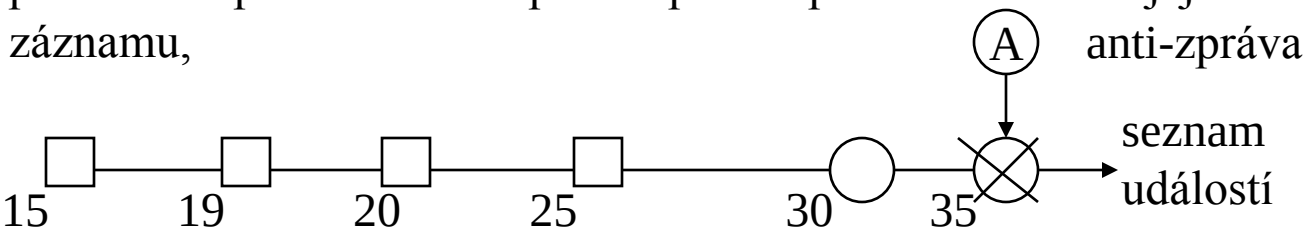
S_{15}…….stav procesu po ukončení zpětného běhu

Metoda Time warp - zpracování antizpráv

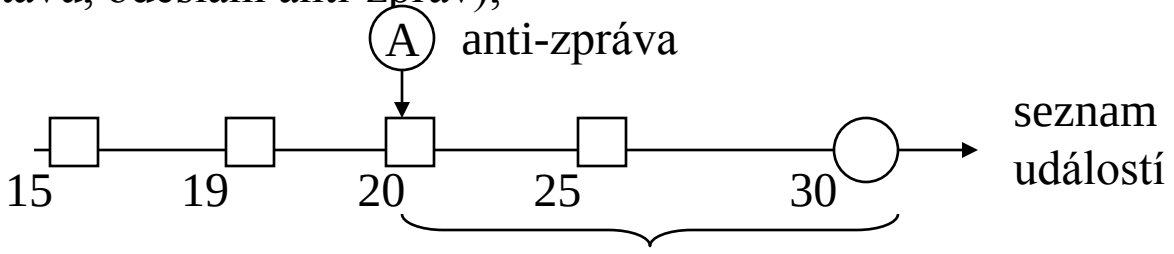
Vliv anti-zpráv:

rozlišíme 3 možné reakce procesu na přijetí anti-zprávy (dle stupně zpracování původní sdružené zprávy v přijímacím procesu):

a) událost odvozená od sdružené vstupní zprávy nebyla dosud provedena: příslušná anti-zpráva způsobí pouze odstranění jejího záznamu,



b) událost odvozená od sdružené zprávy byla již provedena: dojde ke zpětnému běhu z důvodu eliminace vlivu této události (nastavení stavu, odeslání anti-zpráv),



c) původní sdružená zpráva nebyla (z důvodu nějakého zpoždění) procesem dosud přijata: anti-zpráva se zařadí do seznamu událostí a po přijetí původní zprávy bude pouze odstraněna (bez ohledu na to zda okamžitá hodnota modelového času procesu překročila její časovou známku či ne).

Metoda Time warp - eliminace výstupních zpráv

Agresivní rušení:

- zpětný běh ruší záznamy provedených událostí v předešlém dopředném běhu (vyjma událostí naplánovaných v menším čase než je dolní mez zpětného běhu a událostí ve vstupních frontách) a všechny zprávy, které byly odeslány jiným procesům,
- následující dopředný běh generuje znovu všechny výstupní zprávy.

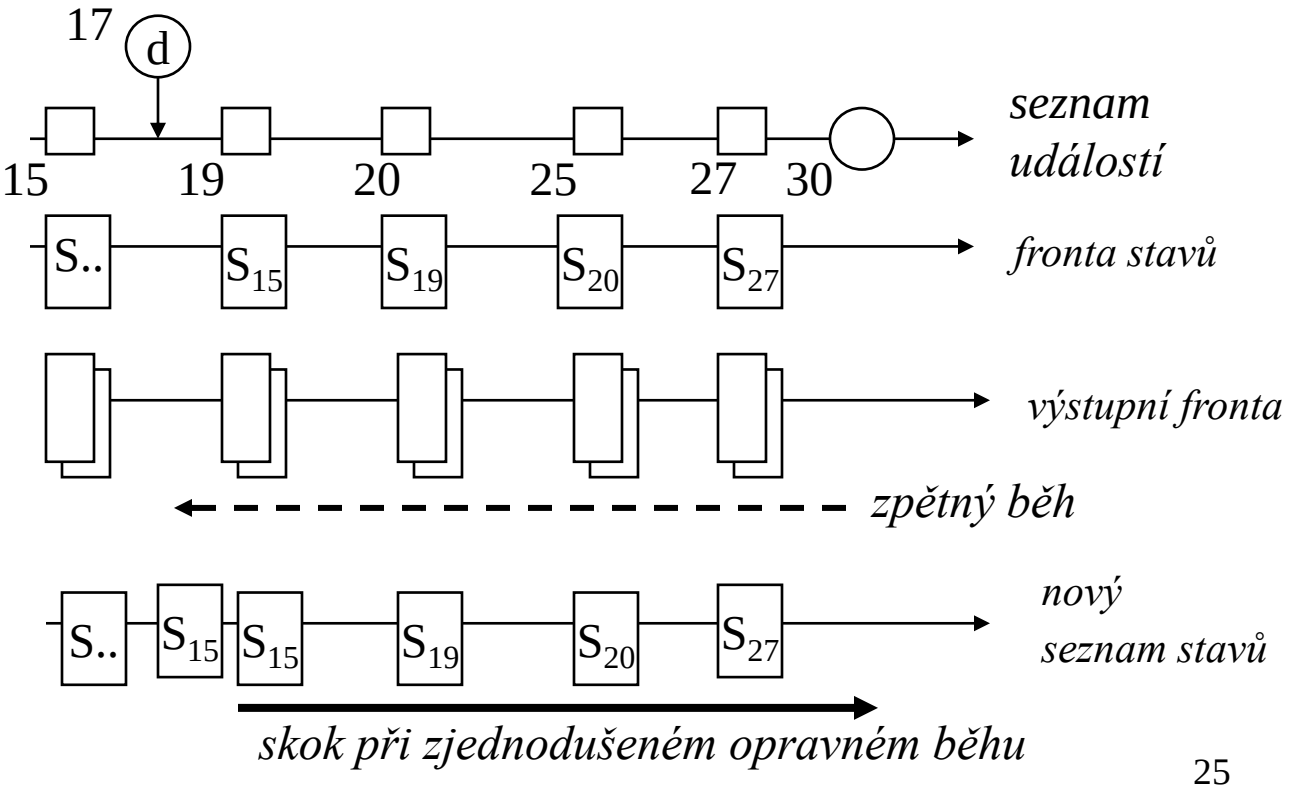
Opožděné, líné (lazy cancelation) rušení - tento přístup vychází z předpokladu, že většina zpráv odeslaných v předchozím dopředném běhu zůstane v platnosti a jejich rušení ve zpětném běhu je zbytečné:

- všechny anti-zprávy v průběhu zpětného běhu zůstávají zachovány,
- v průběhu následujícího opravného dopředného běhu se anti-zprávy odesílají pouze v případě, že by nedošlo k opakovanému vyslání původních sdružených zpráv; v opačném případě se anti-zprávy zachovávají pro případ dalších zpětných běhů :
  - porovnávají se všechny nově generované výstupní zprávy s existujícími anti-zprávami,
  - v případě existence sdružené anti-zprávy je právě generovaná výstupní zpráva zrušena,
  - v případě neexistence sdružené anti-zprávy je generovaní zprávy důsledek opravného běhu; nově generovaná výstupní zpráva je odeslána a nová anti-zpráva je zařazena do seznamu anti-zpráv.
- výhody: odpadá násilné rušení událostí, které budou pravděpodobně obnoveny
- nevýhody: opožděné vyslání anti-zpráv umožňuje větší rozšíření chyb,

Metoda Time warp: optimalizace opravných běhů

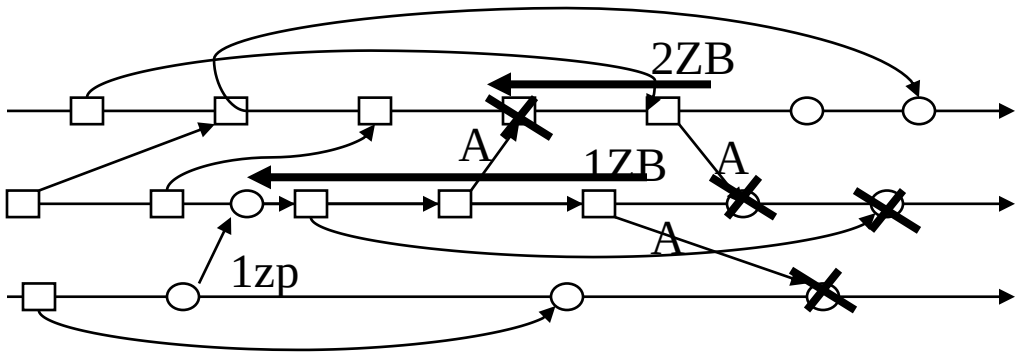
jde o snahu redukovat opakování zbytečných opravných běhů:

- příjem opožděné zprávy způsobí vložení nové události do seznamu událostí a zpětný běh procesu,
- při zpětném běhu procesu se nevysílají anti-zprávy (viz opožděné rušení) ani neruší záznamy stavů procesu v seznamu stavů,
- po provedení nově vložené zpožděné události lze zahájit a nebo někdy i vynechat opravný dopředný běh:
  - po provedení události specifikované zpožděnou zprávou se porovná okamžitý stav procesu se záznamem původního stavu před provedením této události v předešlém dopředném běhu,
  - jsou-li oba stavy stejné a nedošlo-li k jiným změnám v seznamu událostí na „intervalu zpětného běhu“ pak zpožděná událost nezmění efekt opravného běhu vůči původnímu běhu a provedení všech původních událostí tohoto „intervalu zpětného běhu“ lze přeskočit.



Metoda Time warp:charakteristika zpětných běhů

- prvotní zpětný běh nějakého procesu je způsoben přijetím externí zprávy od jiného „pomalejšího“ procesu a je omezen zdola hodnotou časové známky té opožděné zprávy, která daný zpětný běh přivodila,
- hodnota časové známky zmíněné opožděné zprávy nemůže být menší než okamžitá hodnota modelového času „pomalejšího“ procesu, který tuto zprávu vygeneroval (důsledek nezáporné hodnoty předstihu),
- následkem rozesílání anti-zpráv v průběhu prvotního zpětného běhu mohou nastat odvozené zpětné běhy i jiných procesů a jejich dolní meze jsou omezeny časovými známkami zmíněných anti-zpráv; návrat těchto odvozených zpětných běhů směrem do historie nemůže být „hlubší“ než návrat prvotního zpětného běhu,
- návrat zmíněného prvotního zpětného běhu nemůže být proveden do menší hodnoty modelového času než je okamžitá hodnota modelového času onoho „pomalého“ procesu v okamžiku vygenerování zmíněné externí zprávy, která byla podnětem prvotního i všech odvozených zpětných běhů,
- u nejpomalejšího ze všech zúčastněných procesů nemůže dojít k prvotnímu zpětnému běhu.

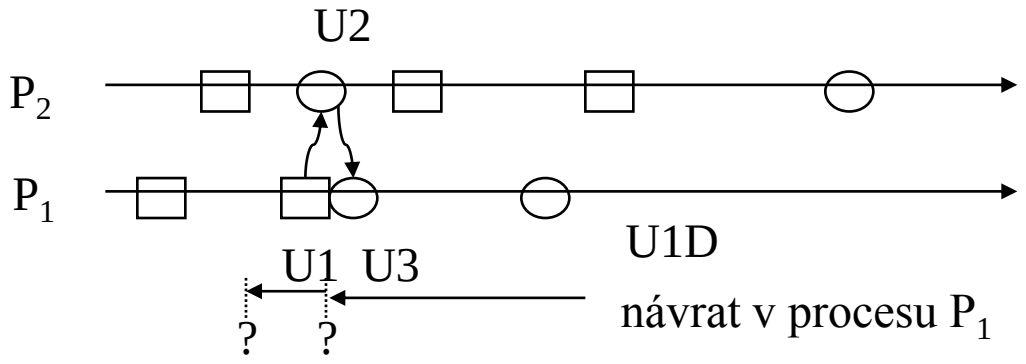


1zp……..prvotní zpožděná zpráva,
1ZB, resp. 2ZB…prvotní, resp. druhotný zpětný běh

Metoda Time warp: případné zacyklení

Důvod zacyklení: následek obousměrné komunikaci procesů a s nulovou hodnotou predikce:

situace:



- předpokládejme, že při provedení události U1 v procesu P_1 dojde k vyslání zprávy U2 s nulovou hodnotou předstihu do procesu P_2 ; proces P_1 nechť přistoupí k provádění další události U1D,
- přijetí zpožděné zprávy U2 v procesu P_2 způsobí zpětný běh a předpokládejme, že při zpracování zprávy U2 proces P_2 generuje pro proces P_1 zprávu U3 také s nulovou hodnotou předstihu,
- přijetí opožděné zprávy U3 v procesu P_1 způsobí také jeho zpětný běh; pokud by se v průběhu tohoto běhu eliminovala i událost U1, dojde při následném opravném běhu k novému provedení této události a ke vzniku cyklů: U1-U2-U3-U1-U2-U3…...….

Možná řešení umožňující zabránit vzniku cyklů:

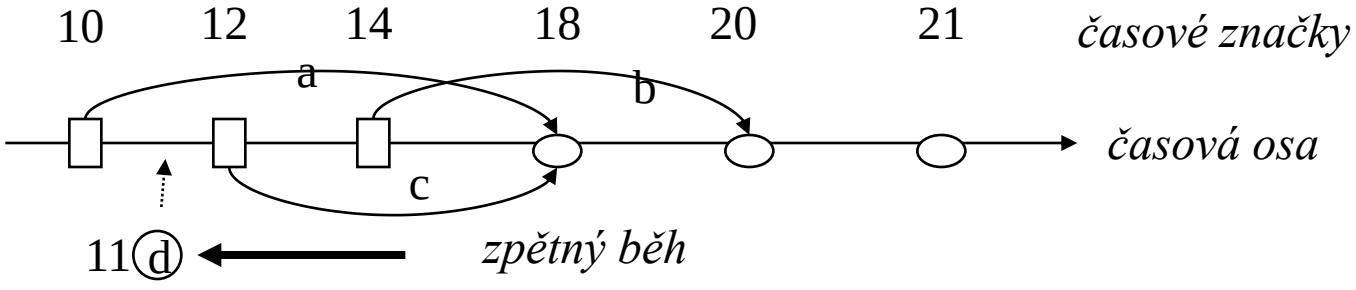
1) zpětný běh nezahrnuje eliminaci vlivu všech těch provedených událostí, které byly v minulosti provedeny ve stejném čase jako je hodnota časové známky opožděné zprávy, která zpětný běh vynutila
2) eliminace v průběhu zpětného běhu nepostihne pouze ty události na nichž zpožděná zpráva U3 závisí (tuto závislost je možné zjistit pomocí rozšířené časové známky o položku věk (věk = 0, 1, 2, 3,….)

Metoda Time warp: rušení budoucích událostí

strategie vyplývá z existence zpětných běhů:

- složitější situace než rušení (**canceling**) při sekvenčním výpočtu nebo v případě konservativních metod
- má-li nastat zrušení nějaké budoucí události při kterémkoliv dopředném běhu, pak se záznam dané události ze seznamu událostí neodstraní, ale provede se pouze nastavení příznaku jejího zrušení; operace se nazývá **retraction** a umožňuje pozdější eliminaci tohoto „zrušení“ při případném zpětném běhu

ukázka situace:



- a,b….operace plánování budoucích událostí při dopředném běhu
- c……operace retraction - zrušení již naplánované události při dopředném běhu; zde nejde o absolutní výmaz, ale o nastavení příznaku, který by způsobil (pokud zůstane zachován) ignorování této zprávy,
- příchod opožděné zprávy s hodnotou časové značky 11 způsobí návrat do stavu procesu v čase 10 => při agresivním rušení v průběhu zpětného běhu bude událost v čase 20 zrušena, u události 18 bude pouze eliminiván příznak zrušení, který byl nastaven operací c.

Metoda Time warp: globální funkce systému

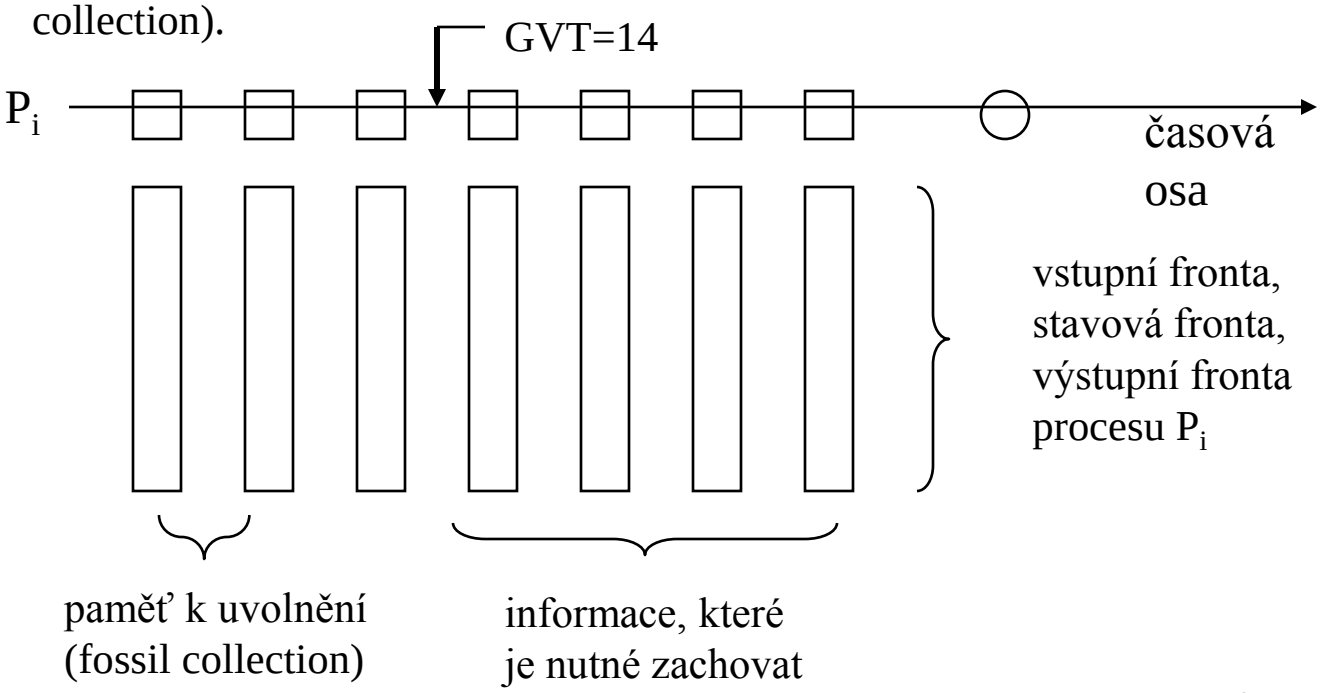
samotné procesy nejsou při svých dopředných bězích nijak omezovány ;
výjimky tvoří: nucené návraty vlivem opožděných zpráv, vyčerpání přidělené paměti, provedení nevratných I / O operací =>

nezbytné funkce globální povahy:

- uvolňování již nepotřebné paměti (SQS + všechny fronty)
- provádění nevratných událostí (I/O operace)
- výpočet globálního virtuálního času GVT (global virtual time)

charakteristika GVT:

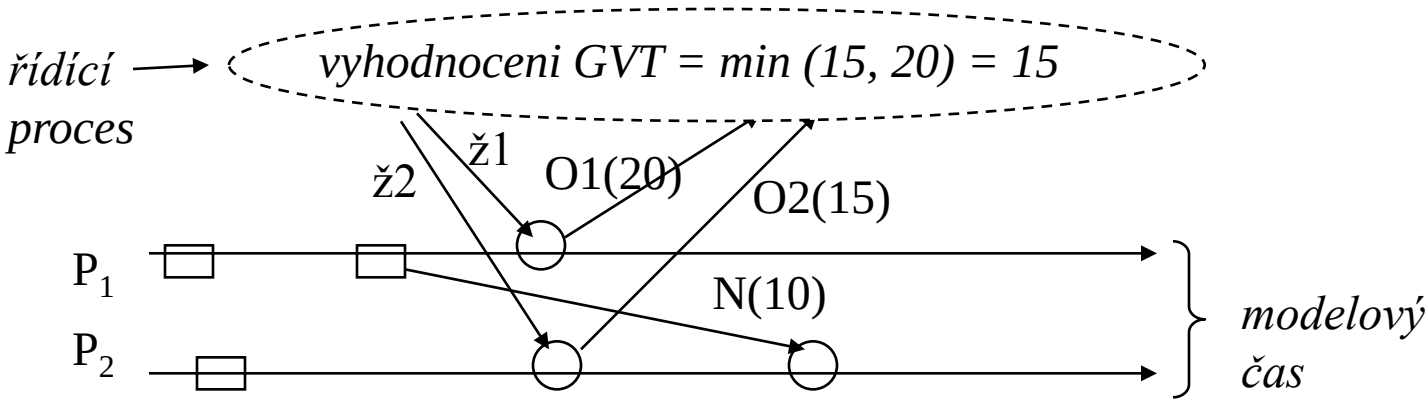
- GVT : hodnota minimální časové známky mezi všemi dosud neprovedenými událostmi nebo částečně provedenými událostmi či pozitivními zprávami či anti-zprávami,
- GVT representuje možnou dolní mezní hodnotu modelového času pro příští možný návrat každého procesu,
- GVT umožňuje uvolňování „historických“ úseků paměti ze vstupních , stavových a výstupních front, které nebudou dále zapotřebí (fossil collection).



Metoda Time warp: určení globálního času

- použití řídícího procesu: z filosofie metody Time warp vyplývají následující problémy:

1) Vliv dosud nedoručených zpráv mezi procesy



ž1, ž2…..žádosti o minimální hodnoty modelových časů

O1(15), O2(20)….odpovědi procesů s příslušnými hodnotami modelových časů

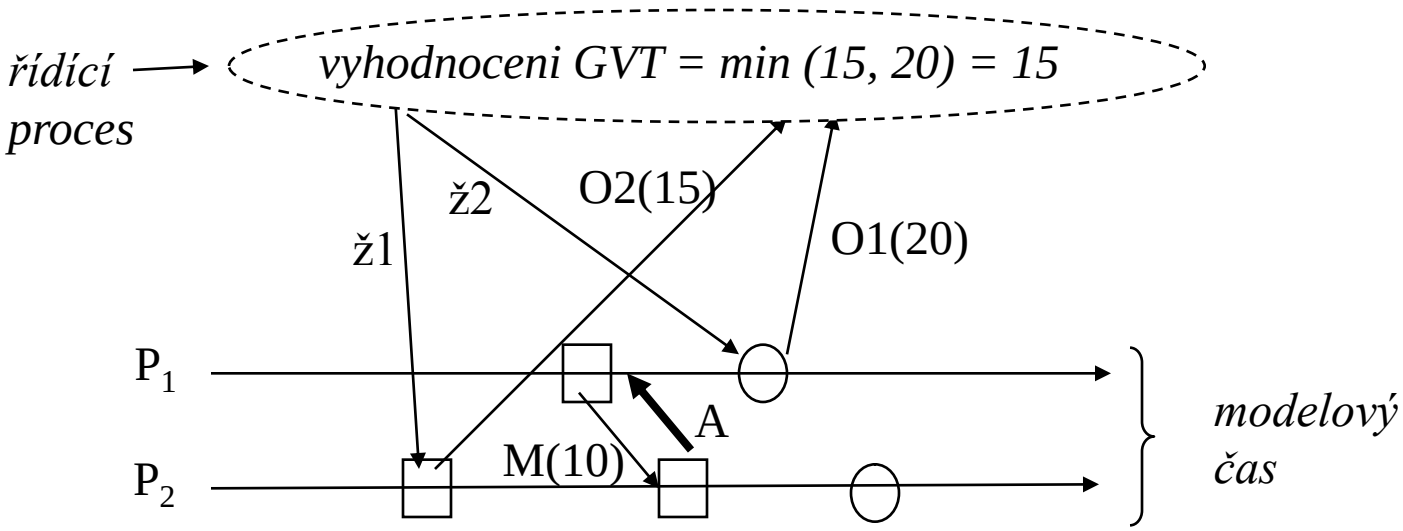
N(10)….dosud nedoručená zpráva s časovou známkou 10, která vyvolá zpětný běh procesu P_2 a znehodnotí výslednou minimální hodnotu 15

Eliminace vlivu nedoručených zpráv:

- zavedení potvrzování zpráv mezi komunikujícími procesy
- vysílací proces zahrne do výpočtu hodnoty svého lokálního minima i časové známky všech jím vyslaných, ale dosud nepotvrzených zpráv (v našem případě proces P_1 správně nahlásí hodnotu lokálního minima 10)

Metoda Time warp: určení globálního času

2) Vliv nesynchronizovaného hlášení jednotlivých procesů způsobeného možným zpožděním žádostí řídícího procesu:



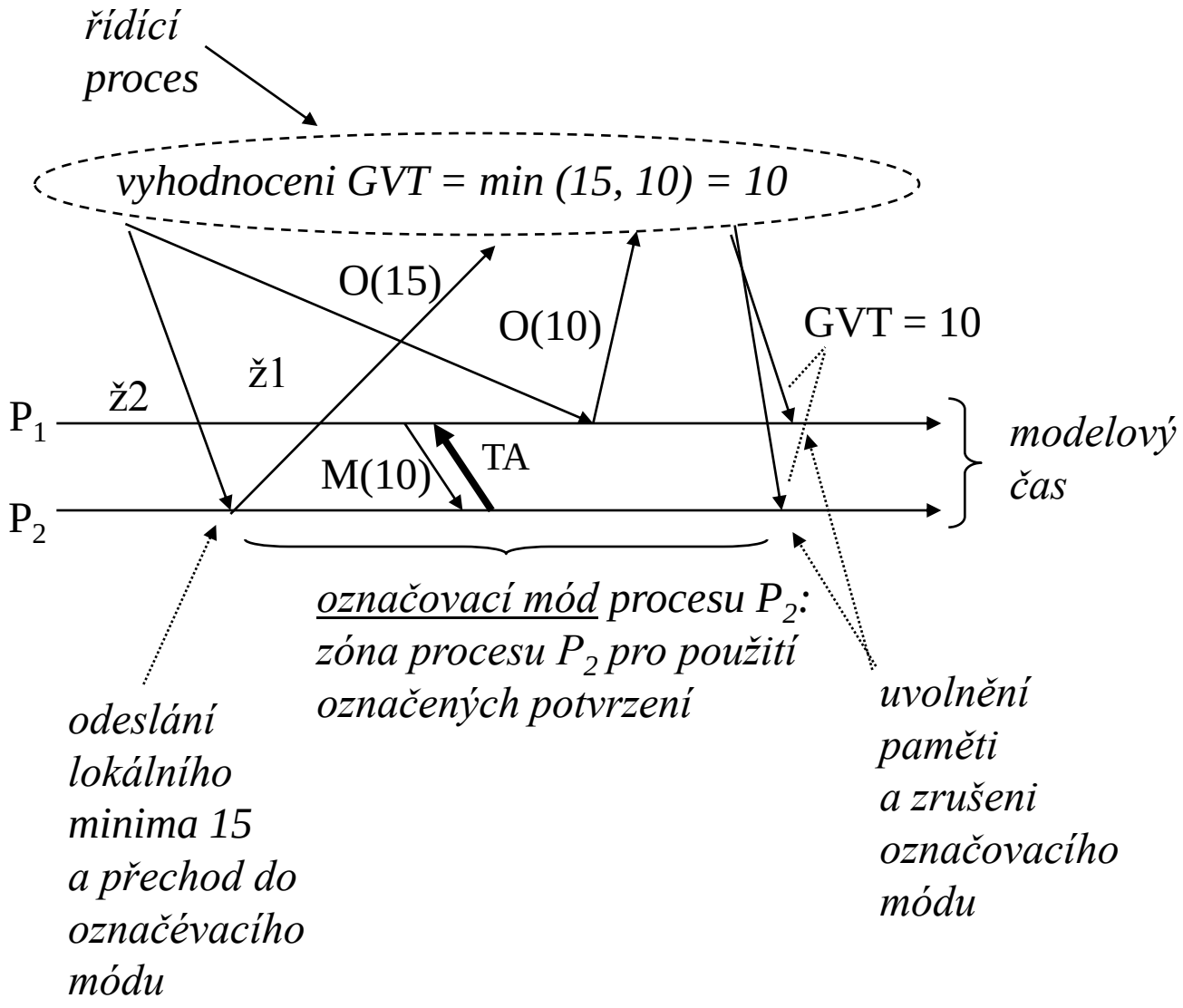
A…….potvrzení zprávy M (10) => P_1 je zbaven zodpovědnosti uvažovat hodnotu časové známky 10

M(10)….zpráva s časovou známkou 10 a odeslaná procesem P_1 způsobila zpětný běh procesu P_2, ale neovlivnila hodnotu 15 jeho lokálního minima a jako potvrzená nebyla ani uvažována procesem P_1 při hlášení jeho lokálního minima 20.

Metoda Time warp: určení globálního času

Eliminace vlivu nesynchronizovaného hlášení:

- použití tzv. označených potvrzení (nezbavuje zodpovědnosti)
- označená potvrzení přinutí přijímací proces zahrnout časové známky jim příslušných zpráv do výpočtu lokálních minim



Metoda Time warp: Samadiho algoritmus

1) Řídící proces odešle všem procesům žádosti o výpočet lokálních minim.
2) Po obdržení žádosti o lokální minimum každý proces vypočte minimální hodnotu z časových známek dosud neprovedených událostí, odeslaných ale dosud nepotvrzených zpráv a antizpráv a časových známek všech označených potvrzení obdržených od přijetí poslední hodnoty GVT. Takto vypočtenou hodnotu proces odešle centrálnímu procesu a přejde do „označovacího módu“.
3) Je-li proces v „označovacím módu“, pak na všechny přijaté zprávy a antizprávy odpovídá pomocí označených potvrzení.
4) Po přijetí lokálních minim od všech procesů vypočte centrální řídící proces hodnotu globálního virtuálního času GVT a tuto odešle všem procesům.
5) Po přijetí GVT každý proces opustí „označovací mód“.

Poznámky: Time Warp algoritmus nijak neomezuje paralelně běžící procesy; důsledky:

- vyčerpání dostupné paměti (především u „rychlých procesů“),
- zvětšování dostupné paměti podporuje odvahu optimistických procesů.
- existují snahy o efektivnější využívání paměti: celkové omezení paměťových nároků na hodnotu k * Mem_{min} (Mem_{min}…velikost potřebné paměti v případě seriového výpočtu).