ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

PDV 09 2018/2019

# Výpočet globálního stavu

Michal Jakob
michal.jakob@fel.cvut.cz
Centrum umělé inteligence, katedra počítačů, FEL ČVUT

OI

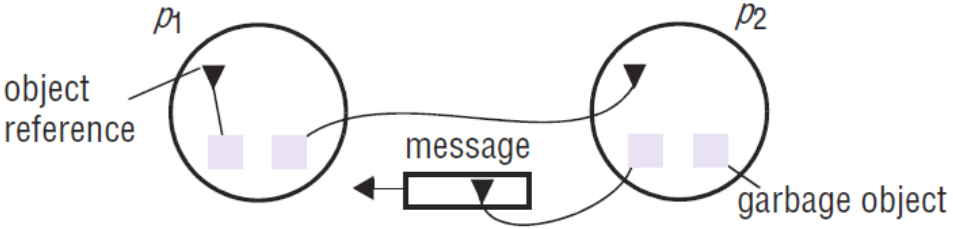
# Globální Stav

**Globální stav**: množina lokální **stavů procesů** v DS a **stavů** všech **komunikačních kanálů** v jednom okamžiku*.

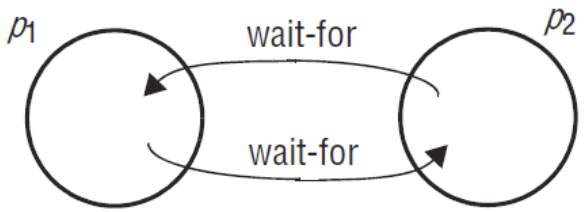
**Globální snapshot**: záznam globální stavu.

# Příklady

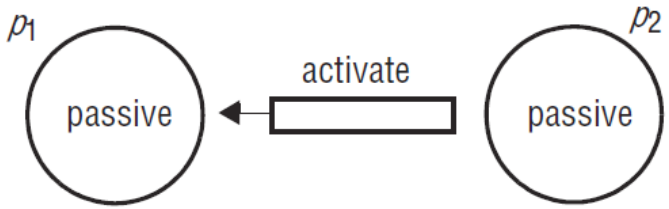
**Garbage collection**: nutnost identifikovat objekty, na které není *globálně* žádná reference.




**Detekce uváznutí (deadlock)**: nutnost identifikovat cykly *globálním* wait-for grafu.




**Detekce ukončení výpočtu**: nutnost zajistit, že *všechny* procesy jsou pasivní a v *žádném* kanálu přenosu není žádná potenciálně aktivační zpráva.




**Checkpointing** za účelem obnovení globálního stavu systému.

…

# Globální stav

Pokud bychom měli **globální hodiny**, tak zaznamenat globální stav jednoduché: všechny procesy by zaznamenaly stav v dohodnutý čas, tj. v jednom **fyzickém okamžiku**.

Jak zaznamenat globální stav **bez** globálních hodin?

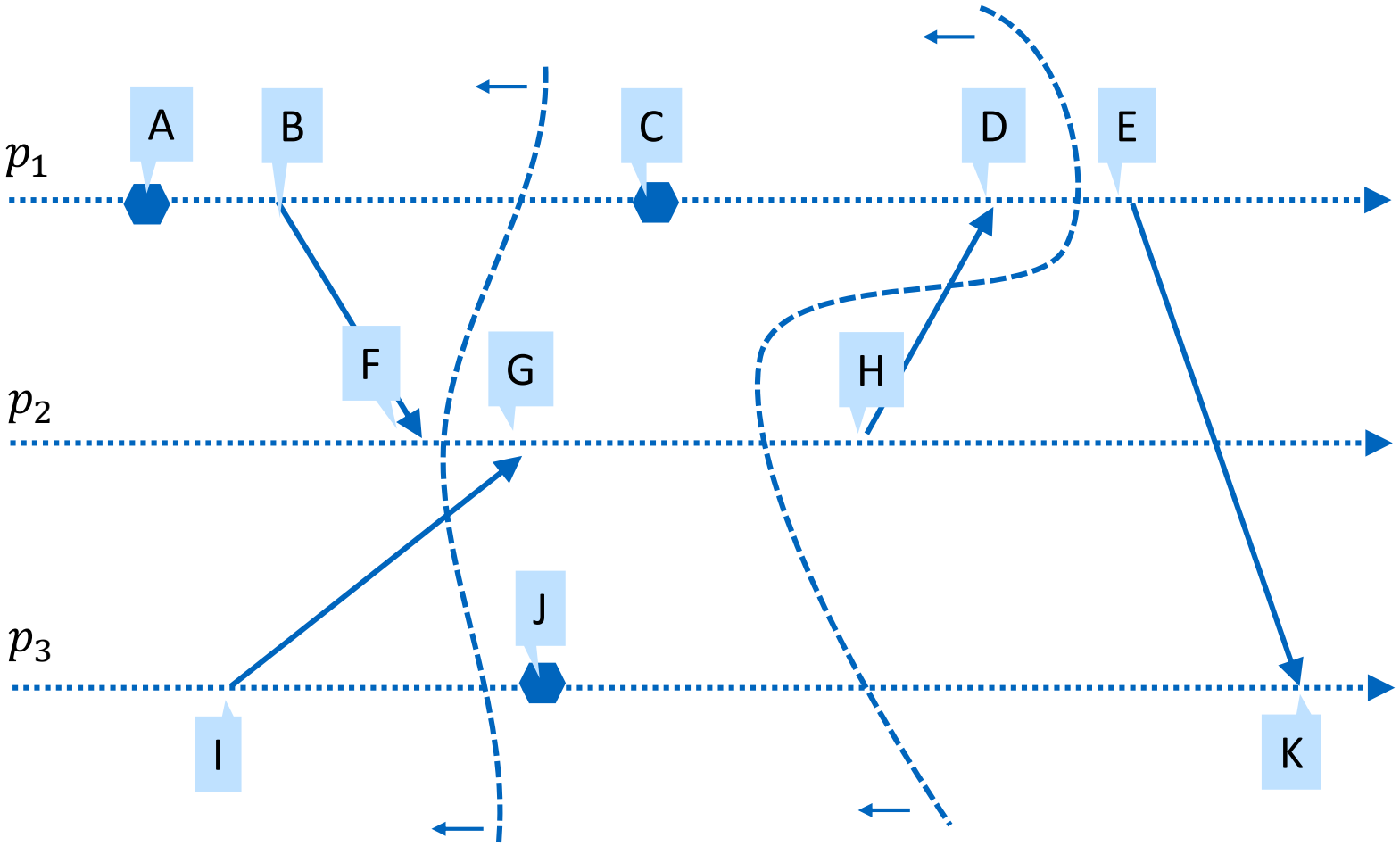
→ **Logický okamžik**

# Řez distribuovaného výpočtu

**Řez**: časová hranice v každém procesu a v každém komunikačním kanále.

- události, které nastanou před řezem, jsou **v řezu**
- události, které nastanou po něm, jsou **mimo řez**.

# Řez: Příklad



# Konzistentní řez

**Definice (optimistická)**

Řez $R$ je konzistentní pokud splňuje kauzalitu, tj. pokud pro každý pár událostí $e, f$ v systému platí:

$$f \in R \wedge e \rightarrow f \Rightarrow e \in R$$

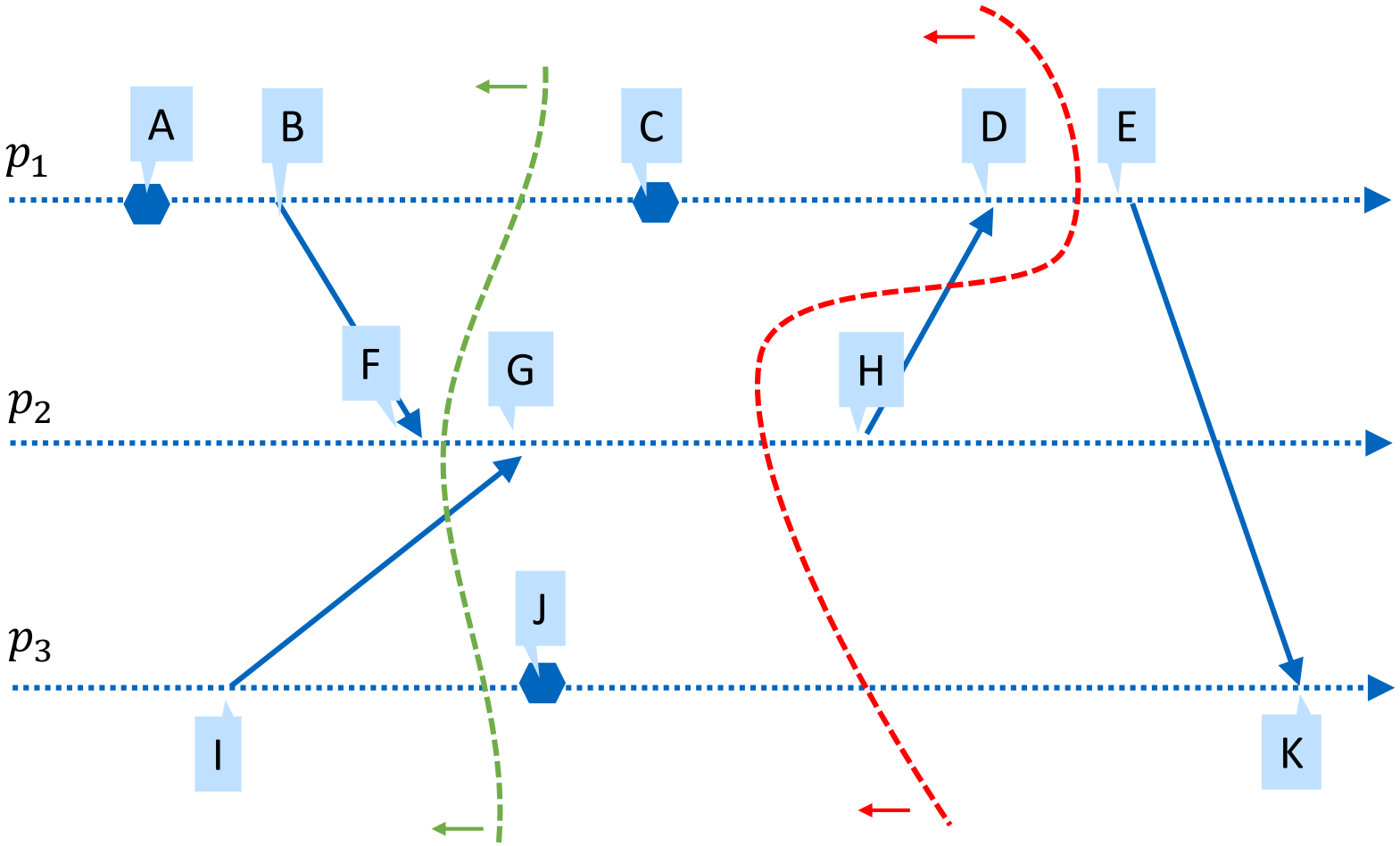
tj. pokud řez obsahuje nějakou událost, obsahuje i všechny, které ji předcházejí dle relace **stalo se před** (tj. nelze, aby v řezu byl „**důsledek**“ a nebyla tam „**příčina**“.)

Konzistentní řez = **logický okamžik**

**Konzistentní globální stav** odpovídá konzistentnímu řezu.

- Globální stav je konzistentní, pokud by mohl byt pozorován externím pozorovatelem.

# Řez: Příklad




**konzistentní** řez
(mohl, ale *nemusel* být pozorován)

**nekonzistentní** řez
(nikdy nemohl být pozorován)

# Výpočet konzistentního globálního stavu

# Problém výpočtu globálního stavu

**Cíl**: Zaznamenat **globální snapshot**, tj. stav pro každý proces a každý komunikační kanál.

**Požadavek:** Zaznamenání snapshotu by neměla **interferovat** s během distribuované aplikace a neměla by vyžadovat po aplikaci zastavení posílání aplikačních zpráv.

**Model**
(Existují i algoritmy se slabšími předpoklady)

- skupina $N$ procesů $p_1, \ldots, p_N$
- **FIFO perfektní komunikační kanál** mezi každým párem procesů, tj. zprávy se neduplikují, nevznikají, neztrácejí a jsou doručovány v pořadí odeslání (značení: $C_{i,j}$ kanál přenáší zprávy z procesu $p_i$ do procesu $p_j$)
- **asynchronní** systém: neznáma, ale **konečná latence**

**Předpoklad**: Každý proces je schopen zaznamenat svůj vlastní aplikační stav (případně low-level systémový stav).

# Chandy-Lamport algoritmus pro distribuovaný globální shapshot

(Vytváření snapshotu je distribuované.)

Speciální zpráva: **ZNAČKA**

Každý proces vykonává dvě pravidla:

- Každý proces může iniciovat vytvoření snapshotu
- pravidlo pro příjem ZNAČKY
- pravidlo pro odeslání ZNAČKY

# Chandy-Lamport algoritmus pro globální shapshot

**Zahájení tvorby snapshotu**

Iniciující **proces** $p_i$ odešle ZNAČKU všem ostatním procesům (i sobě)

**Pravidlo pro příjem ZNAČKY pro proces** $p_i$

Jakmile **proces** $p_i$ přijme ZNAČKU poslanou **kanálem** $C_{m,i}$

**if** ($p_i$ dosud nezaznamenal svůj stav) **then**

$p_i$ **zaznamená** svůj stav (a vykoná pravidlo pro odeslání ZNAČKY);

$p_i$ **zaznamená** stav kanálu $C_{m,i}$ jako prázdnou množinu;

$p_i$ **zapne zaznamenávání** zpráv doručených skrze všechny ostatní příchozí kanály $C_{j,i}$ kromě $C_{m,i}$

**else**

$p_i$ **zaznamená** stav kanálu $C_{m,i}$ jako množinu všech zpráv, které $p_i$ obdržel skrze $C_{m,i}$ od doby, kdy zahájil záznam $C_{m,i}$, a záznam **ukončí**

**end if**

**Pravidlo pro odeslání ZNAČKY pro proces** $p_i$

Poté, co **proces** $p_i$ **zaznamenal** svůj stav, tak pro každý odchozí **kanál** $C_{i,j}$**:**

$p_i$ **odešle** jednu ZNAČKU skrze $C_{i,j}$

(předtím než skrze $C_{i,j}$ pošle jakoukoliv jinou zprávu)

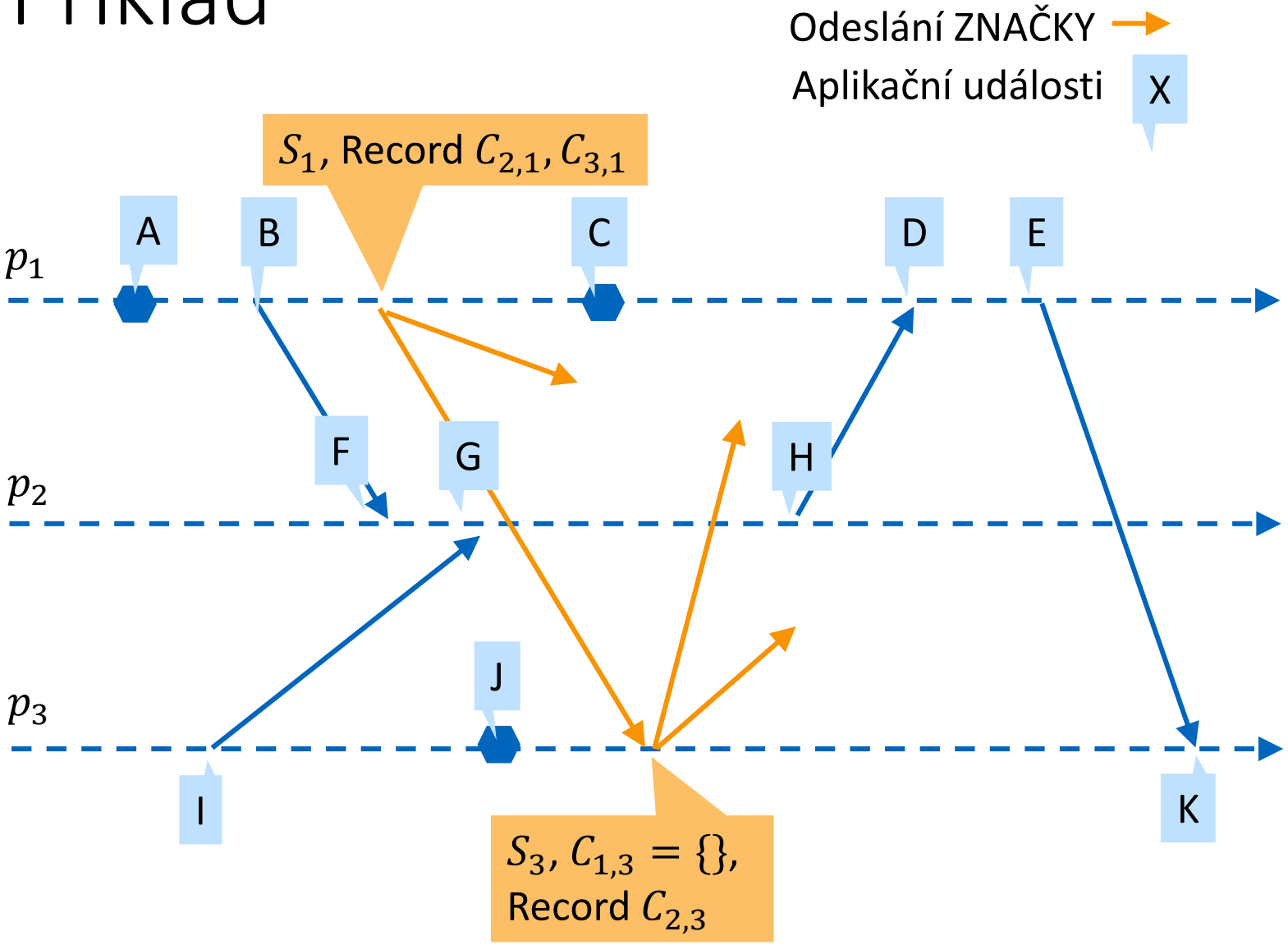
# Ukončení

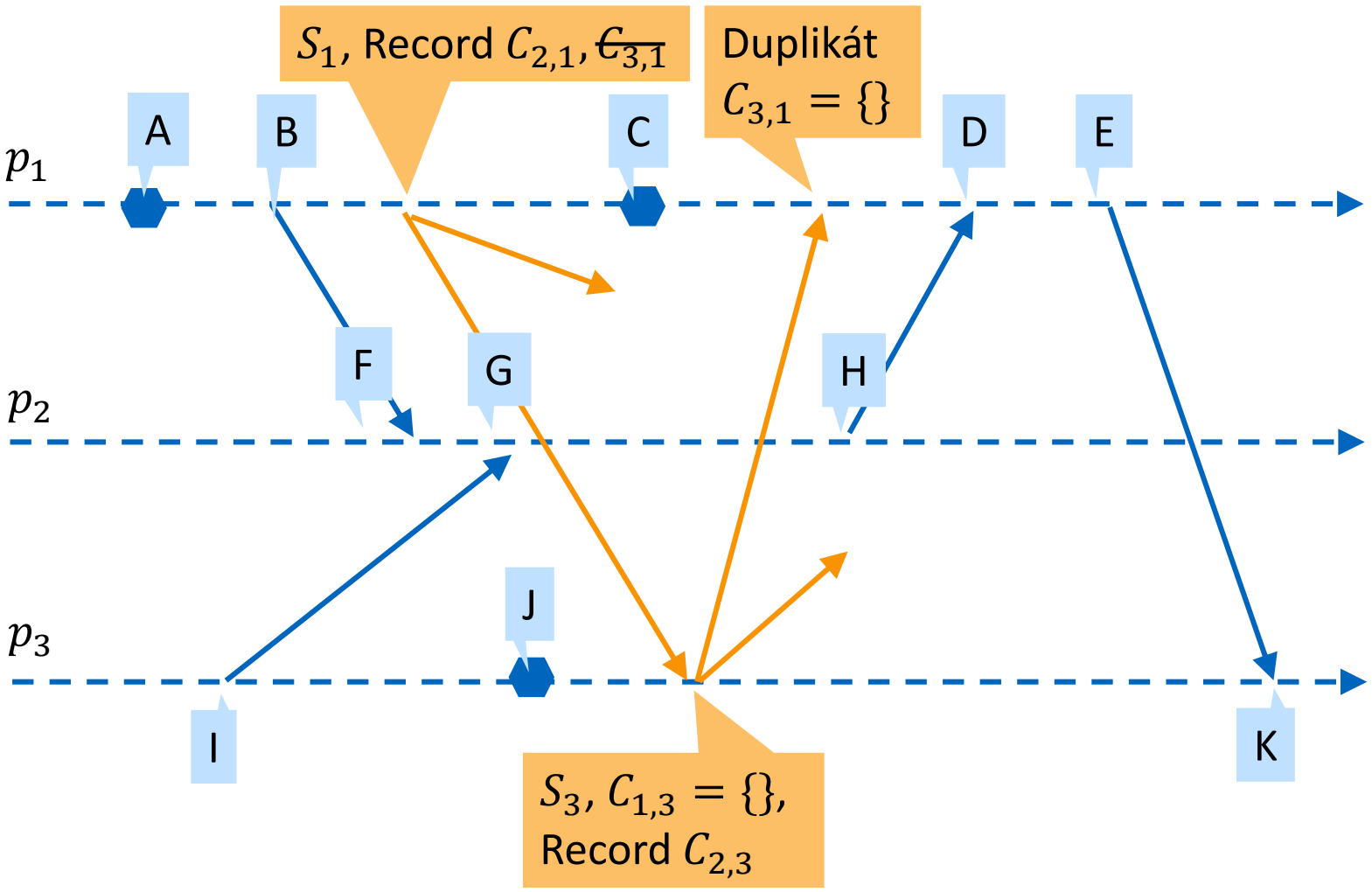
Algoritmus končí jakmile:

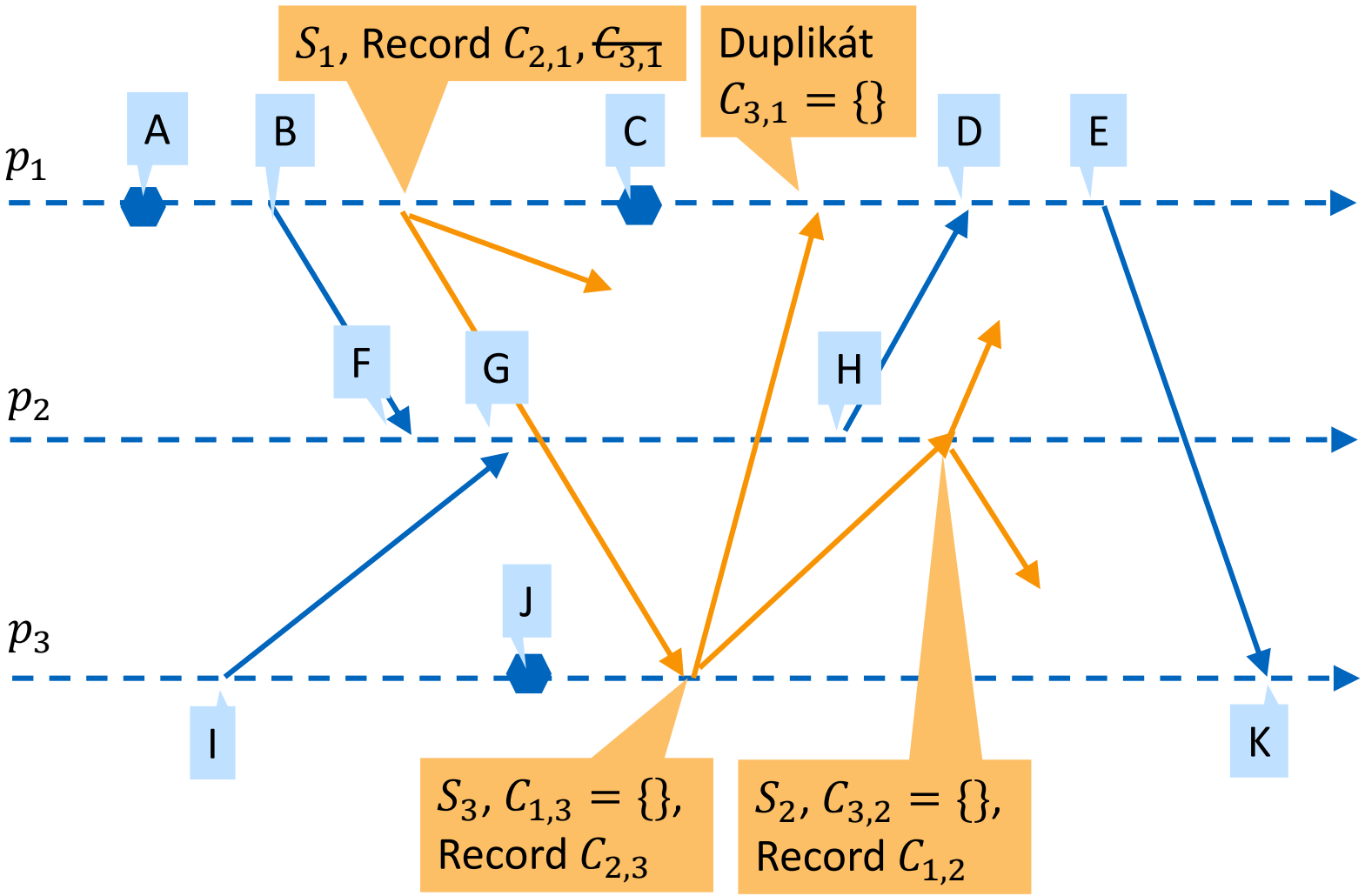
- všechny procesy přijaly Značku (aby zaznamenaly svůj stav) a
- všechny procesy přijaly Značku na všech $N - 1$ svých příchozích kanálech (aby zaznamenaly stav na všech kanálech)

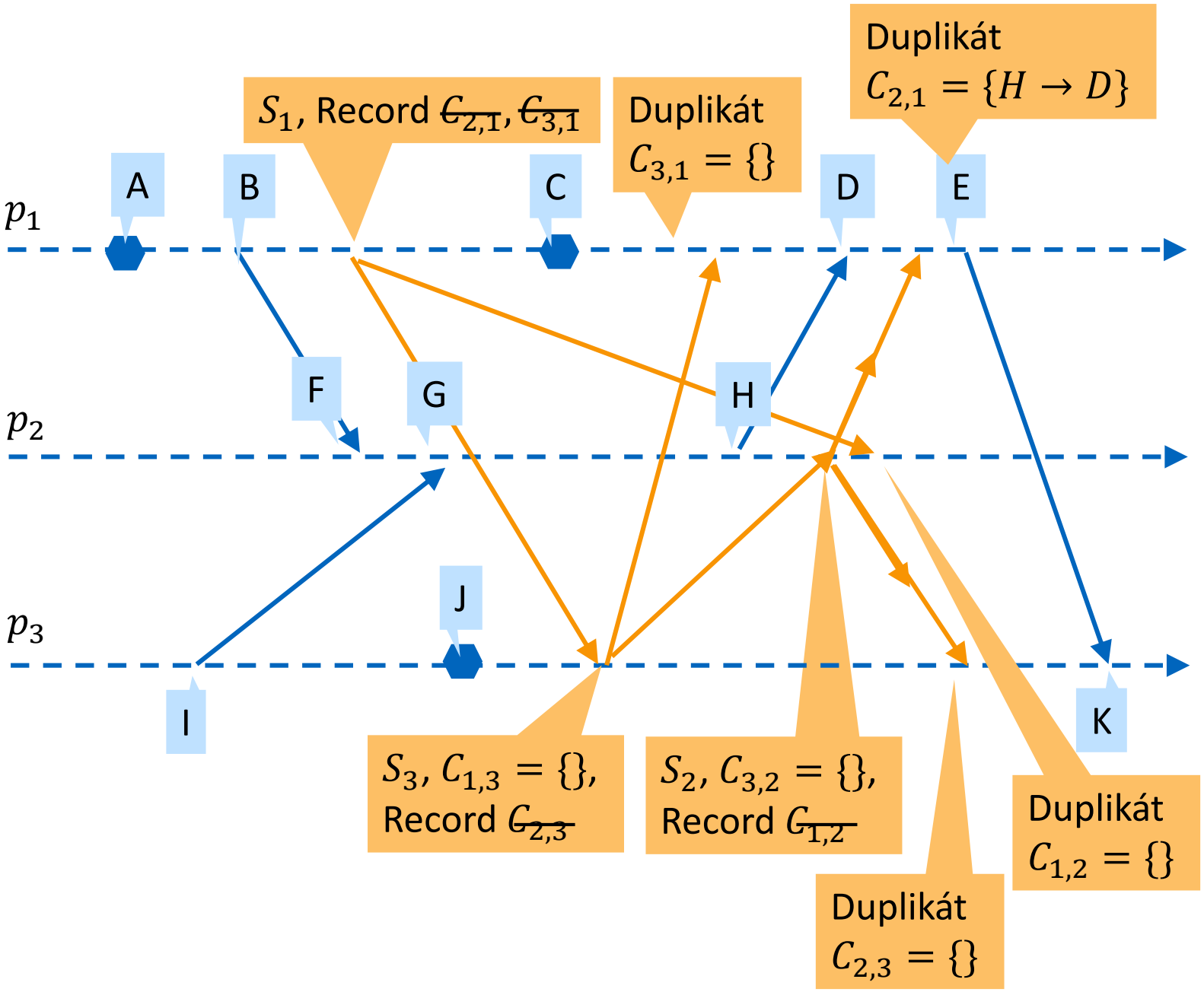
Následně (je-li potřeba) mohou být jednotlivé fragmenty globální stavu posbírány centrálním servery a poskládán **plný globální snaphot**.

# Příklad

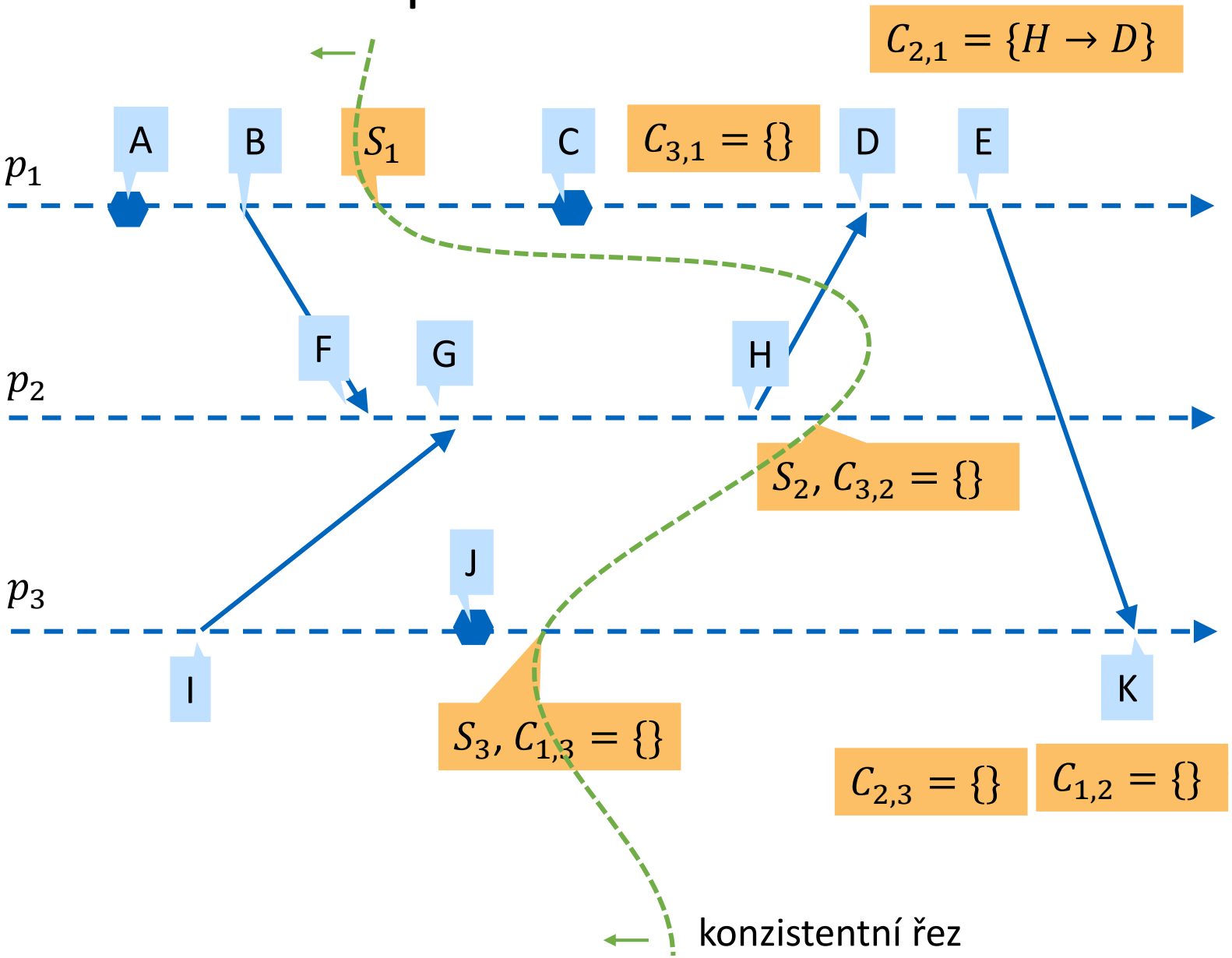

Odeslání ZNAČKY
Aplikační události
X
$S_1$, Record $C_{2,1}, C_{3,1}$
$p_1$
$p_2$
$p_3$
A
B
C
D
E
F
G
H
I
J
K
$S_3$, $C_{1,3} = \{\}$,
Record $C_{2,3}$


$S_1$, Record $C_{2,1}$, ~~$C_{3,1}$~~
Duplikát
$C_{3,1} = \{\}$
A
B
C
D
E
$p_1$
F
G
H
$p_2$
J
$p_3$
I
K
$S_3$, $C_{1,3} = \{\}$,
Record $C_{2,3}$


$S_1$, Record $C_{2,1}$, ~~$C_{3,1}$~~
Duplikát
$C_{3,1} = \{\}$
A
B
C
D
E
$p_1$
F
G
H
$p_2$
J
$p_3$
I
K
$S_3$, $C_{1,3} = \{\}$,
Record $C_{2,3}$
$S_2$, $C_{3,2} = \{\}$,
Record $C_{1,2}$


Duplikát
$C_{2,1} = \{H \rightarrow D\}$
$S_1$, Record ~~$C_{2,1}$~~, ~~$C_{3,1}$~~
Duplikát
$C_{3,1} = \{\}$
A
B
C
D
E
$p_1$
F
G
H
$p_2$
J
$p_3$
I
K
$S_3$, $C_{1,3} = \{\}$,
Record ~~$C_{2,3}$~~
$S_2$, $C_{3,2} = \{\}$,
Record ~~$C_{1,2}$~~
Duplikát
$C_{1,2} = \{\}$
Duplikát
$C_{2,3} = \{\}$

# Finální Snapshot



# Vlastnosti

# Vlastnosti Chandy-Lamport

Výsledkem Chandy-Lamport algoritmu pro výpočet globální snapshotu je **konzistentní řez.**

Důkaz:

- nechť $R$ je výsledný řez a $e_i$ a $e_j$ jsou události v procesech $P_i$ a $P_j$ tak, že $e_i \rightarrow e_j$
- pro konzistenci třeba dokázat implikaci: $e_j \in R \Rightarrow e_i \in R$
- tj. pokud $e_j \rightarrow < P_j$ zaznamená svůj stav $>$ pak $e_i \rightarrow < P_i$ zaznamená svůj stav $>$
- Přepokládejme, že tato implikace neplatí, tj. $e_j \rightarrow < P_j$ zaznamená svůj stav $>$ a $< P_i$ zaznamená svůj stav $> \rightarrow e_i$
- Uvažujme cestu aplikačních zpráv z $e_i$ do $e_j$ (přes další procesy)
- Vzhledem k FIFO kanálu musí ZNAČKY na všech spojích výše uvedené cesty předcházet aplikační zprávy
- Tedy, protože $< P_i$ zaznamená svůj stav $> \rightarrow e_i$, tak musí platit, že $P_j$ obdržel svoji značku před $e_j$
- A tedy $e_j \notin R$ → spor

# Korektnost v DS

## Živost (Liveness)

Garance, že v DS **časem** dojde k něčemu **dobrému** (bude dosažen žádoucí stav).

Příklady:

- Distribuovaný výpočet: výpočet skončí.
- Konsensus: všechny proces se shodnou na výstupní hodnotě.
- Úplnost při detekci selhání: každé selhání je časem detekováno.

## Bezpečnost (Safety)

Garance, že v DS **nikdy** nedojde k něčemu **špatnému** (nebude dosažen nežádoucí stav).

Příklady:

- Nedojde k uváznutí (deadlocku)
- Žádný objekt se nestane sirotkem
- Přesnost při detekci selhání
- Konsensus: Žádné dva procesy nevyprodukují různý výstup.

# V kontextu globálních stavů

Uvažujme vlastnost $\phi$ definovanou nad globálními stavy distribuovaného výpočtu.

**Živost vzhledem k $\phi$ v globálním stavu $S$**

$S$ je splněno v $\phi$ nebo existuje kauzální cesta mezi $S$ a **nějakým** stavem $S'$, ve kterém je $\phi$ splněno.

**Bezpečnost vzhledem k $\phi$ v globálním stavu $S$**

$\phi$ je splněno v $S$ a ve **všech** stavech $S'$ dosažitelných z $S$ je $\phi$ taky splněno.

# Vyhodnocení stabilních vlastností

**Stabilní vlastnost** je taková vlastnost, že jakmile je ve výpočtu **jednou** splněna, zůstává splněna **navždy**.

- příklad stabilní vlastnost živosti: výpočet skončil
- příklad stabilní vlastnosti porušující bezpečnost: nastalo uváznutí, objekt je sirotek (neukazuje na něj žádná reference)

Chandy-Lamport algoritmus lze použít pro detekci **stabilních** globálních vlastností.

(Lze ukázat, že pokud nějaká stabilní vlastnost splněna v globálním snapshotu zachyceným snapshot algoritmem, bude splněna i ve finálním stavu běhu snapshot algoritmu. Naopak, pokud v snapshotu nějaká stabilní vlastnost splněna není, nemohla být splněna ani ve stavu při zahájení algoritmu)

# Shrnutí

Schopnost **zachytit globální stav** distribuovaného výpočtu je důležitá.

Vytvoření snapshotů by nemělo nijak omezovat probíhající výpočet.

**Chandy-Lamportův algoritmus** vypočte globální snapshot.

Vypočtený globální snapshot odpovídá **konzistentnímu řezu**.

Globální snapshot může být využit k **detekci stabilních vlastností** výpočtu.