

PDV 09 2017/2018

Výpočet globálního stavu

Michal Jakob


michal.jakob@fel.cvut.cz

Centrum umělé inteligence, katedra počítačů, FEL ČVUT





Globální Stav

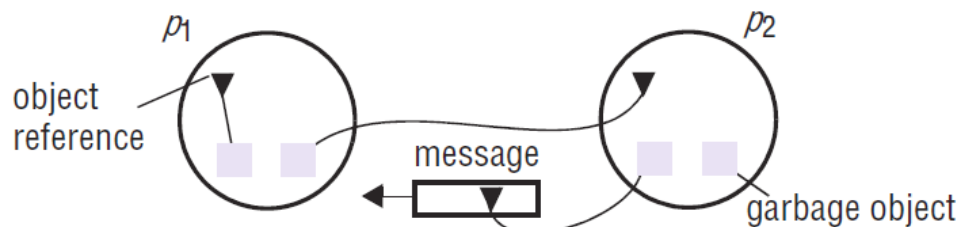


Globální stav: množina lokální **stavů procesů** v DS a **stavů** všech **komunikačních kanálů** v jednom okamžiku*.

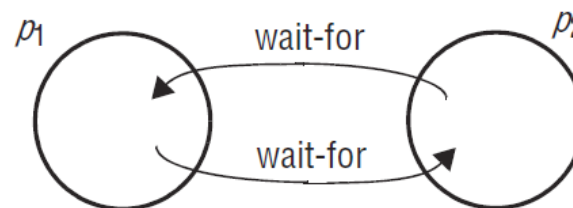
Globální snapshot: záznam globální stavu.

Příklady

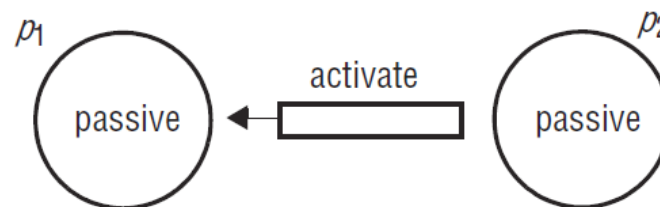
Garbage collection: nutnost identifikovat objekty, na které není *globálně* žádná reference.



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?

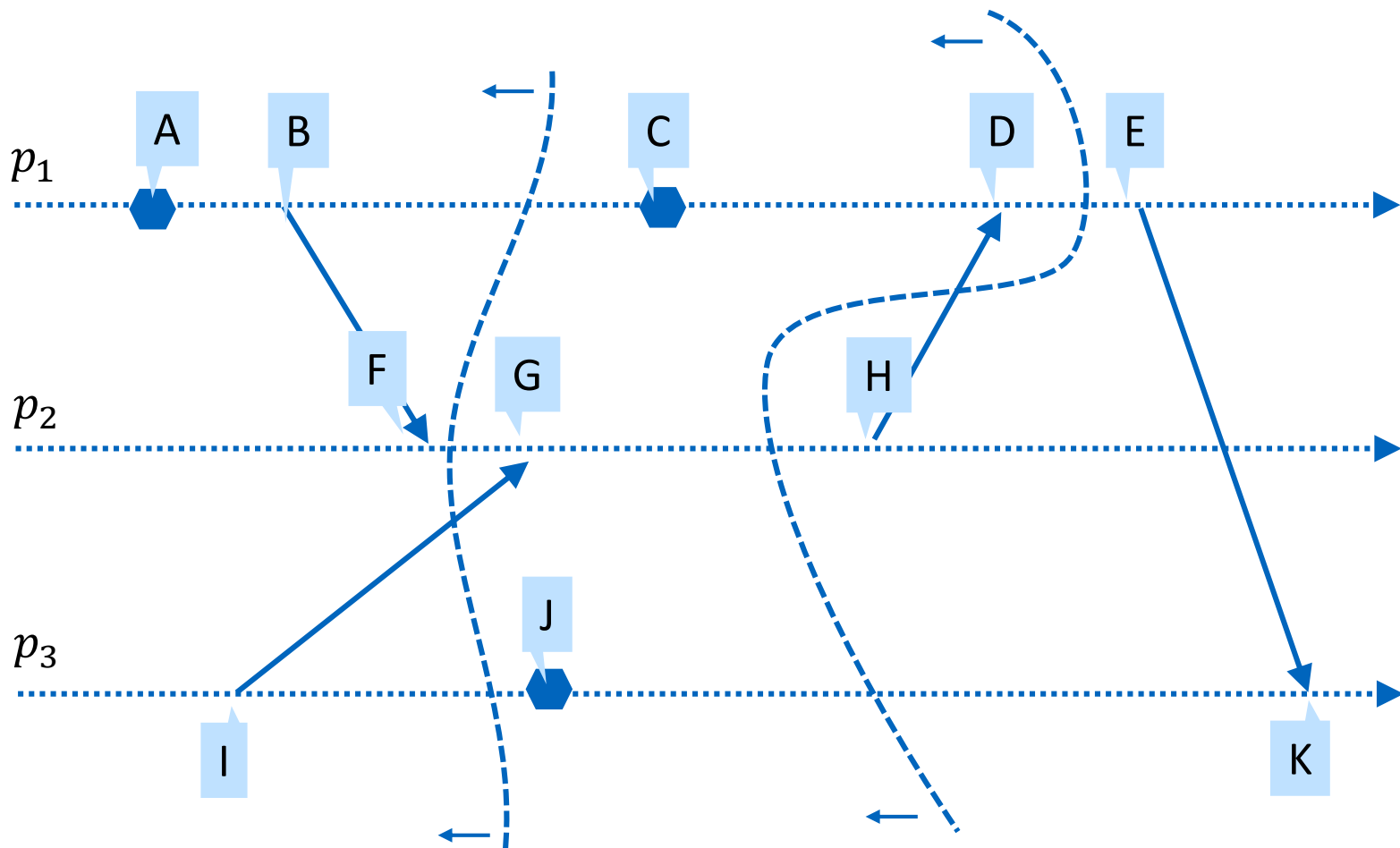
→ „logický okamžik“

Řez distribuovaného výpočtu

Řez: časová hranice v každém procesu a v každém kanálu.

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

Řez: Příklad



Konzistentní řez

Konzistentní řez R splňuje kauzalitu, tj. pro každý pár událostí e, f v systému platí:

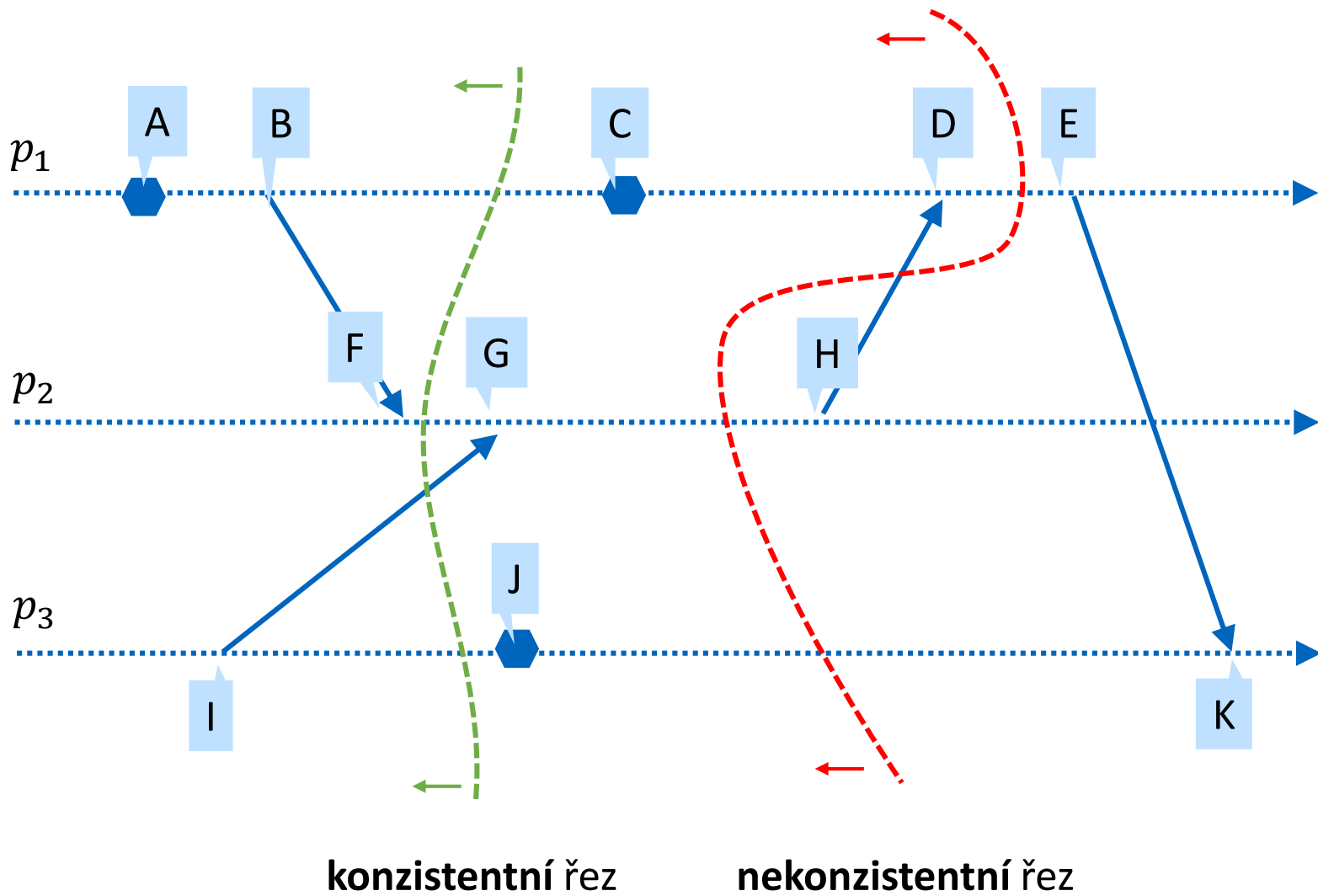
$$f \in R \wedge e \rightarrow f \Rightarrow f \in R$$

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 globální stav odpovídá konzistentní řezu.

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

Řez: Příklad





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
- FIFO perfektní komunikační kanál mezi každým párem procesů, tj. zprávy se **neduplikují, nevznikají, neztrácejí** a jsou doručovány **v pořadí** odeslání
- **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 snapshot

(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 snapshot

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 uložil svůj stav

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

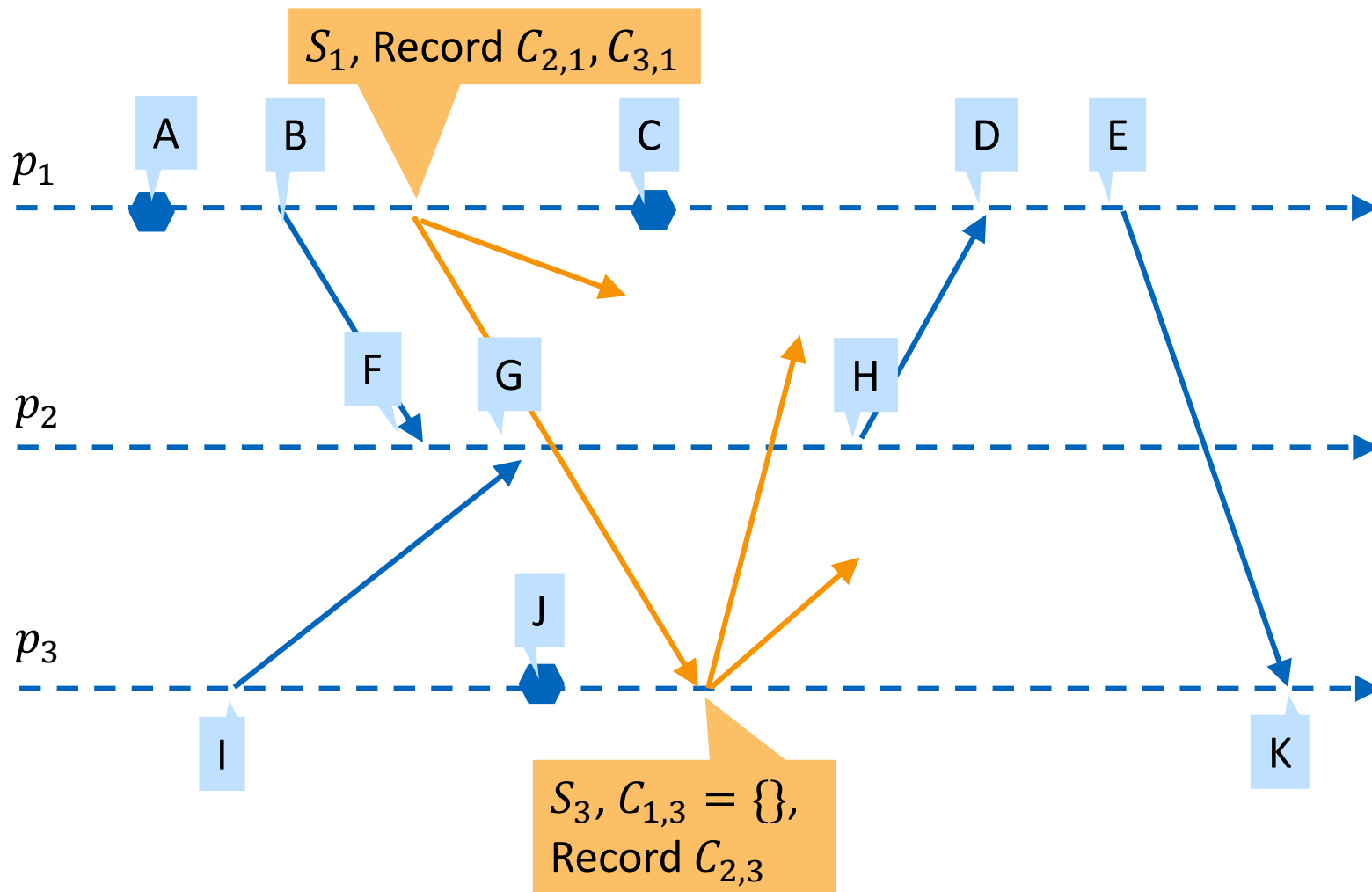
Algoritmus končí jmile:

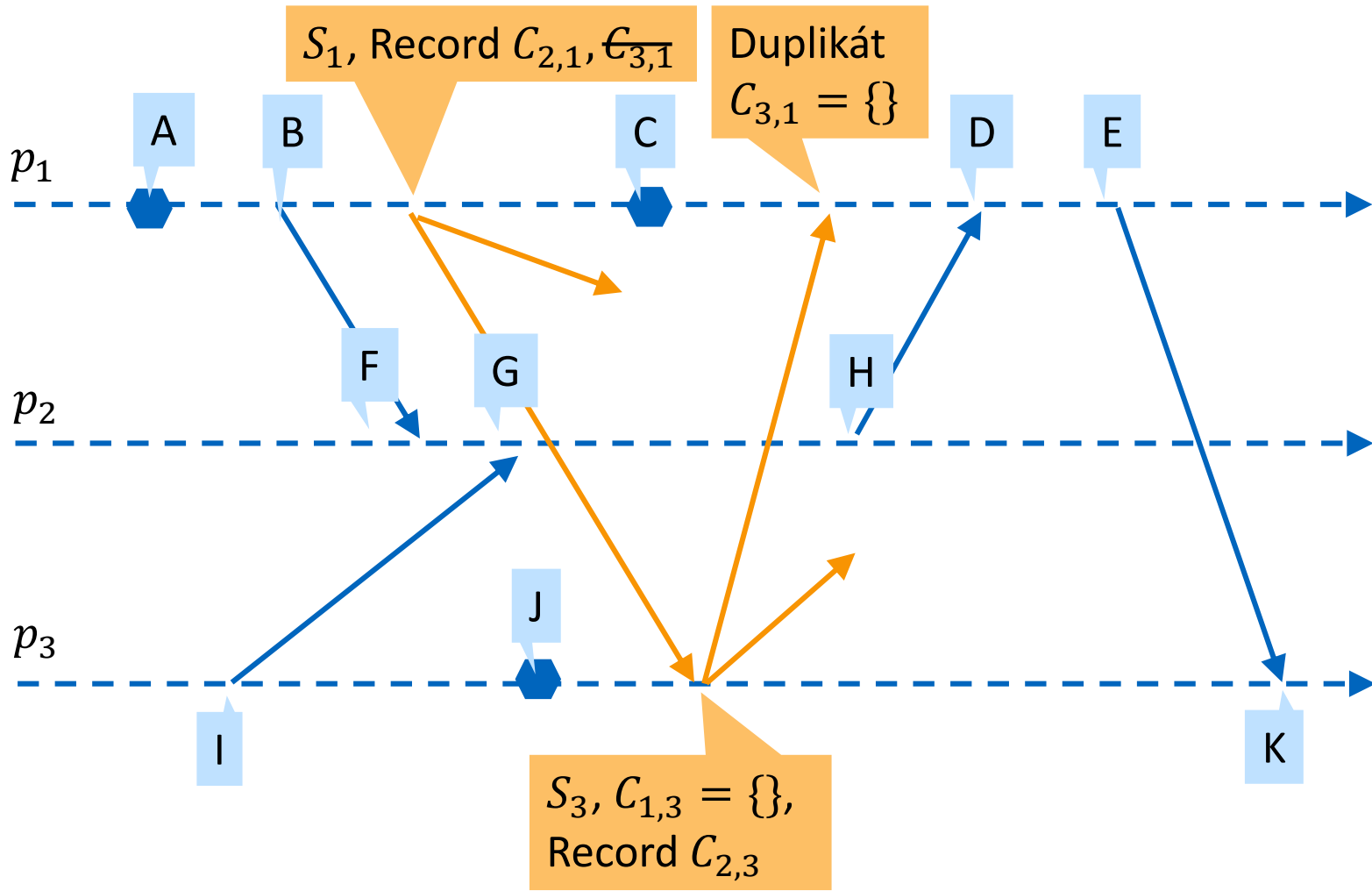
- všechny procesy přijaly Značku (aby zaznamenaly svůj stav) a
- všechny procesy přijaly Značku na všech $N - 1$ svých příchozích kanálech (aby zaznamenaly stav na všech kanálech)

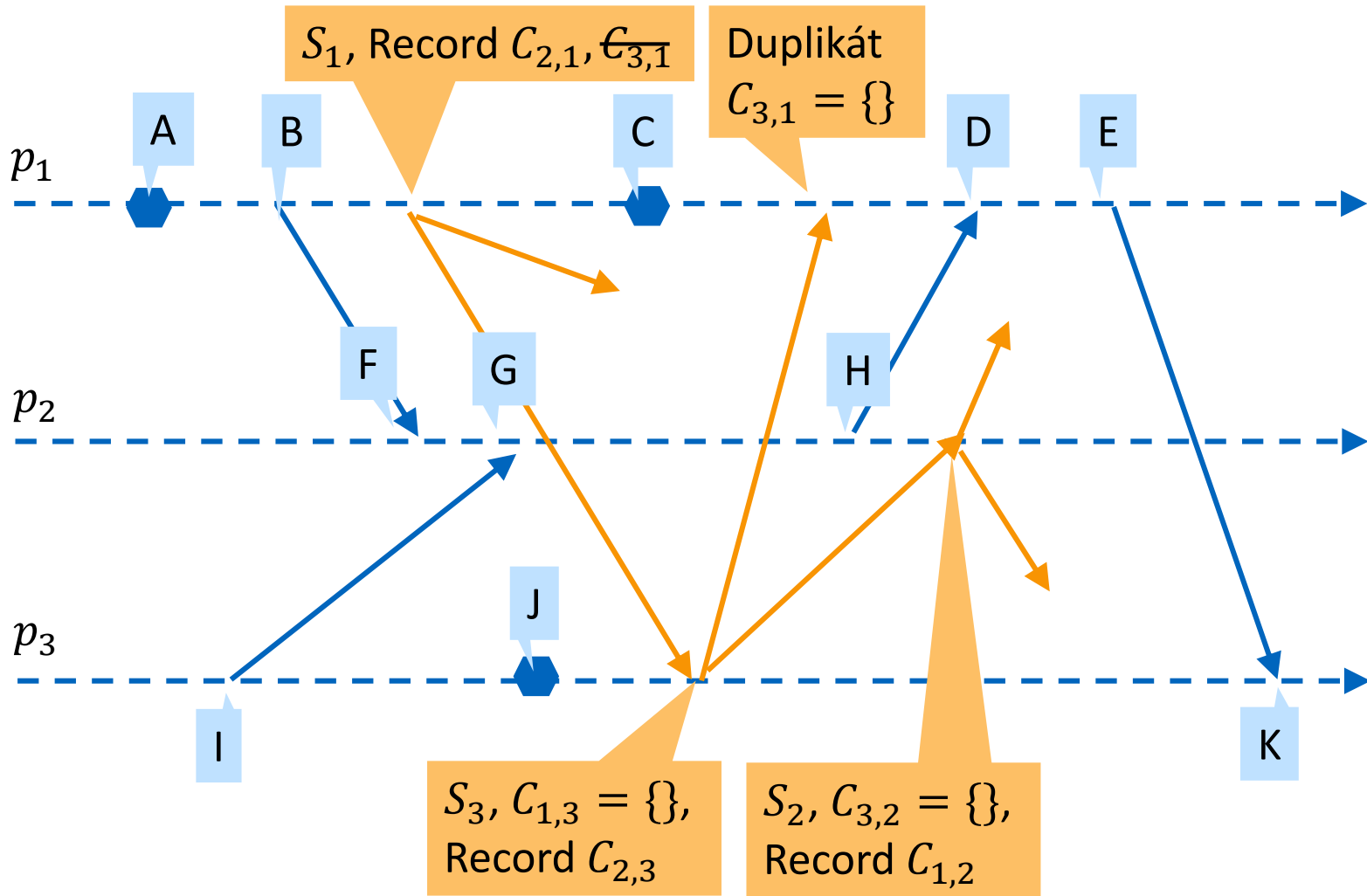
Následně (je-li potřeba) mohou být jednotlivé fragmenty globálního stavu posbírány centrálním serverem a poskládán **plný globální snapshot**.

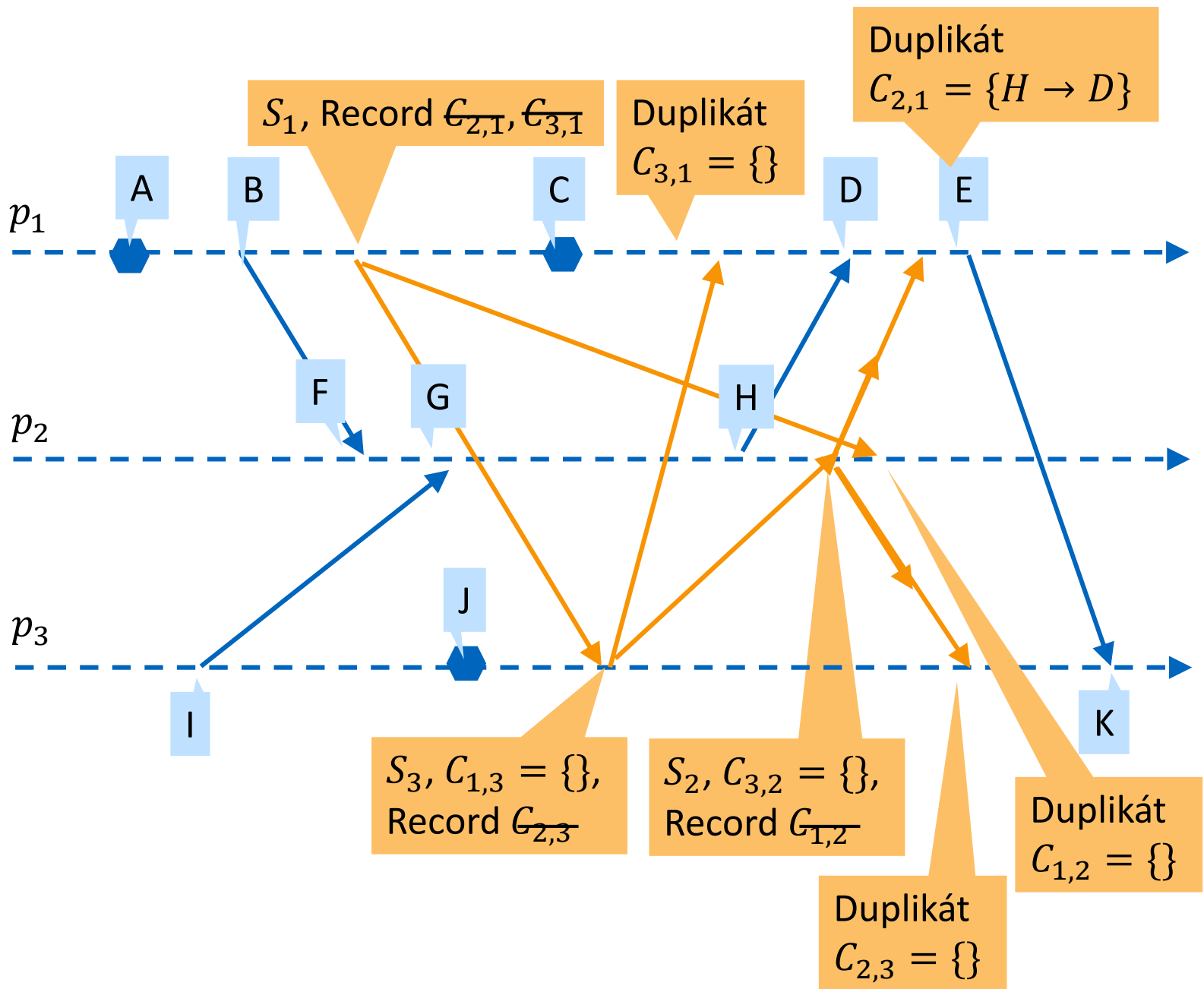
Příklad

Odeslání ZNAČKY →

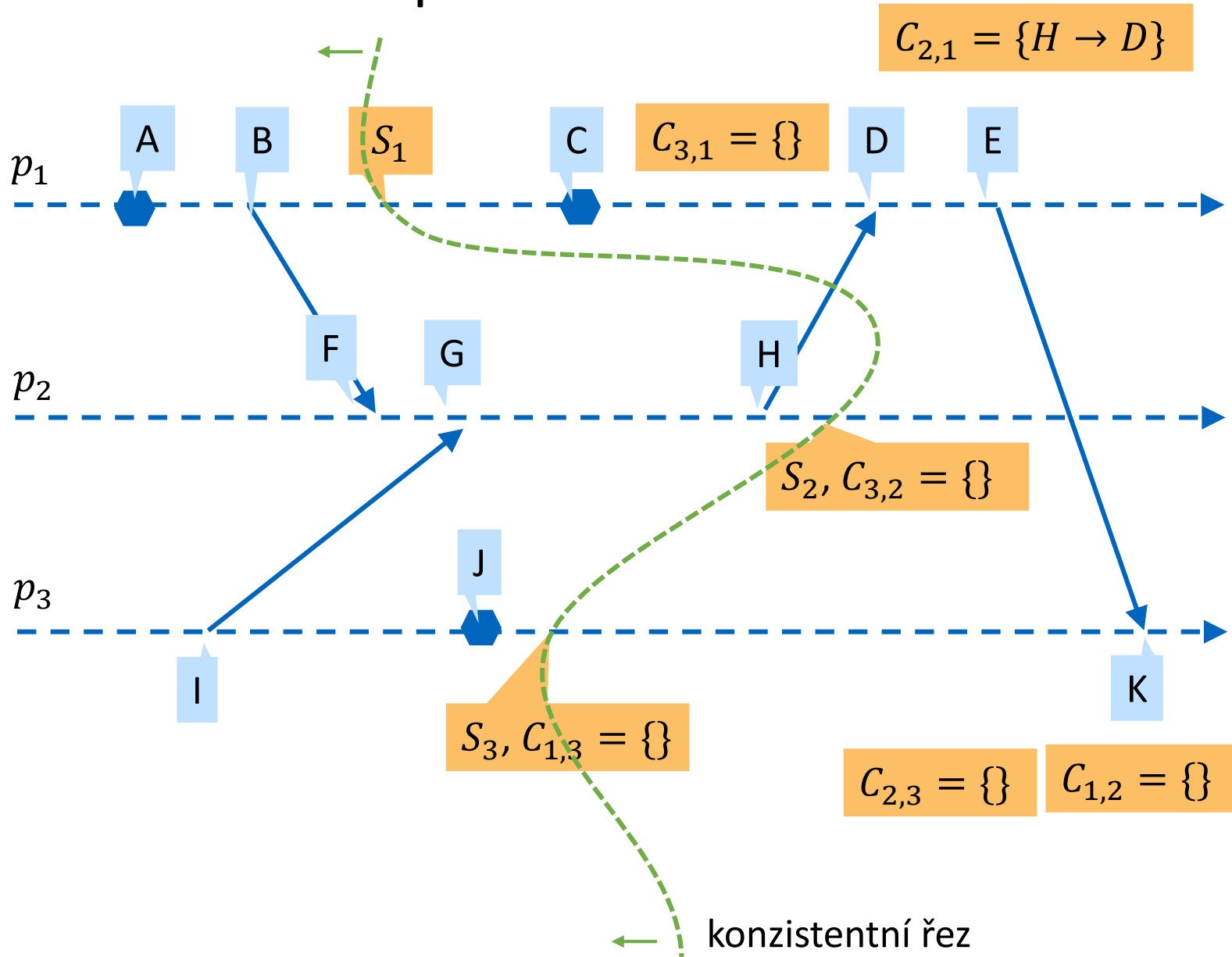








Finální Snapshot





Vlastnosti

Vlastnosti Chandy-Lamport

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

Důkaz:

- necht' 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 $e_j \rightarrow\langle P_j \text{ zaznamená svůj stav} \rangle$ pak $e_i \rightarrow\langle P_i \text{ zaznamená svůj stav} \rangle$
- Přepokládejme, že tato implikace neplatí, tj. $e_j \rightarrow\langle P_j \text{ zaznamená svůj stav} \rangle$ a $\langle P_i \text{ zaznamená svůj stav} \rangle \rightarrow e_i$
- 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 $\langle P_i \text{ zaznamená svůj stav} \rangle \rightarrow e_i$, tak musí platit, že P_j obdržel svoji značku před e_j
- A tedy $e_j \notin R \rightarrow$ 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 procesy se shodnou na výstupní hodnotě.
- Úplnost při detekci selhání: každé selhání je nakonec 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 ϕ definovanou nad globálními stavy distribuovaného výpočtu.

Živost vzhledem k ϕ v globálním stavu S

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

Bezpečnost vzhledem k ϕ v globálním stavu S

ϕ je splněno v S a ve **všech** stavech S' dosažitelných z S je ϕ taky splněno.

Vyhodnocení stabilních vlastností

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

- příklad stabilní vlastnosti 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 algoritmu. Naopak, pokud v snapshotu nějaká stabilní vlastnost splněna není, nemohla být splněna ani ve stavu při zahájení algoritmu)

Souhrn

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.