PDV 10 2022/2023

Globální stav a jeho snapshot

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.

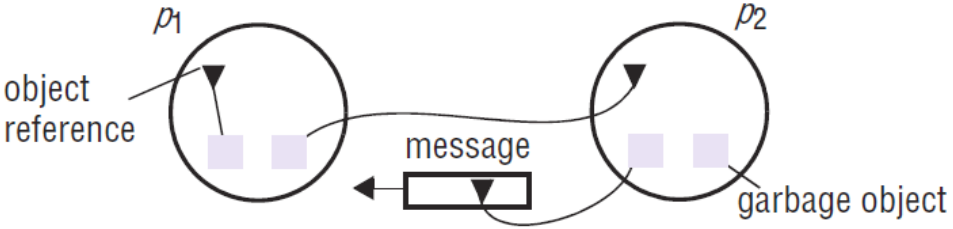
Globální stav se mění, kdykoliv dojde v systému **k akci** (události)

- vykonání instrukce v procesu
- poslání nebo přijetí zprávy

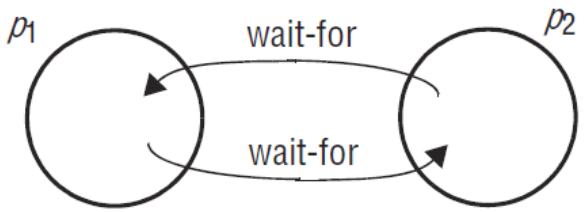
Globální snapshot: záznam globální stavu.

Příklady

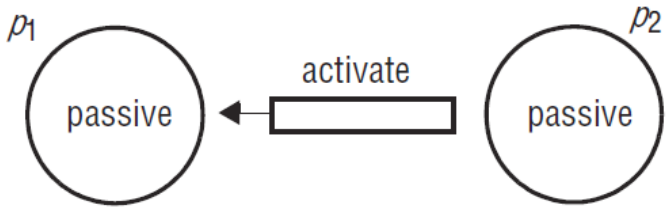
Garbage collection: nutnost identifikovat objekty, na které není *globálně* žádná reference.



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 snapshot

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

Fyzický okamžik by garantoval **konzistenci** zaznamenaného globálního stavu.

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

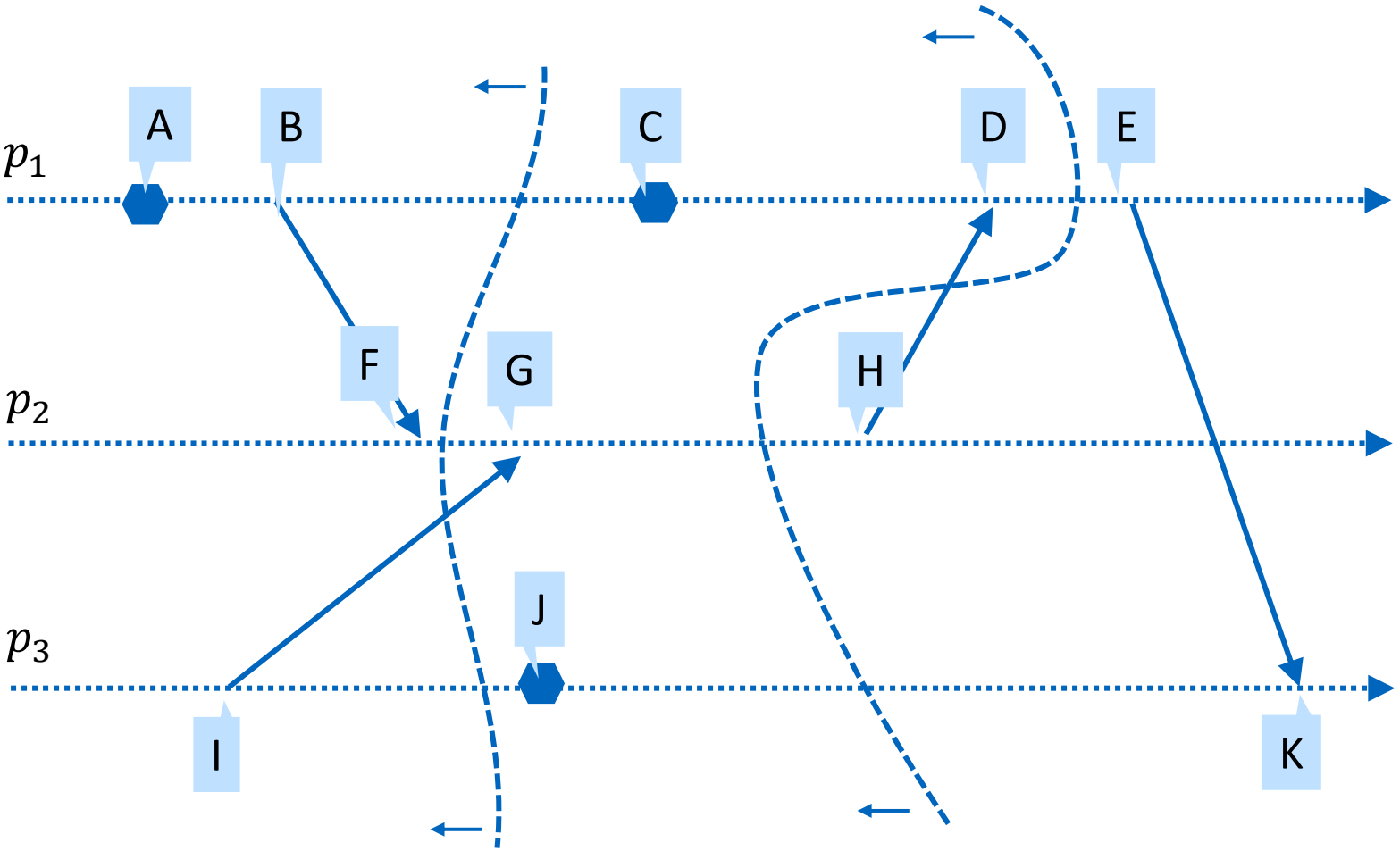
→ **Logický okamžik**

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

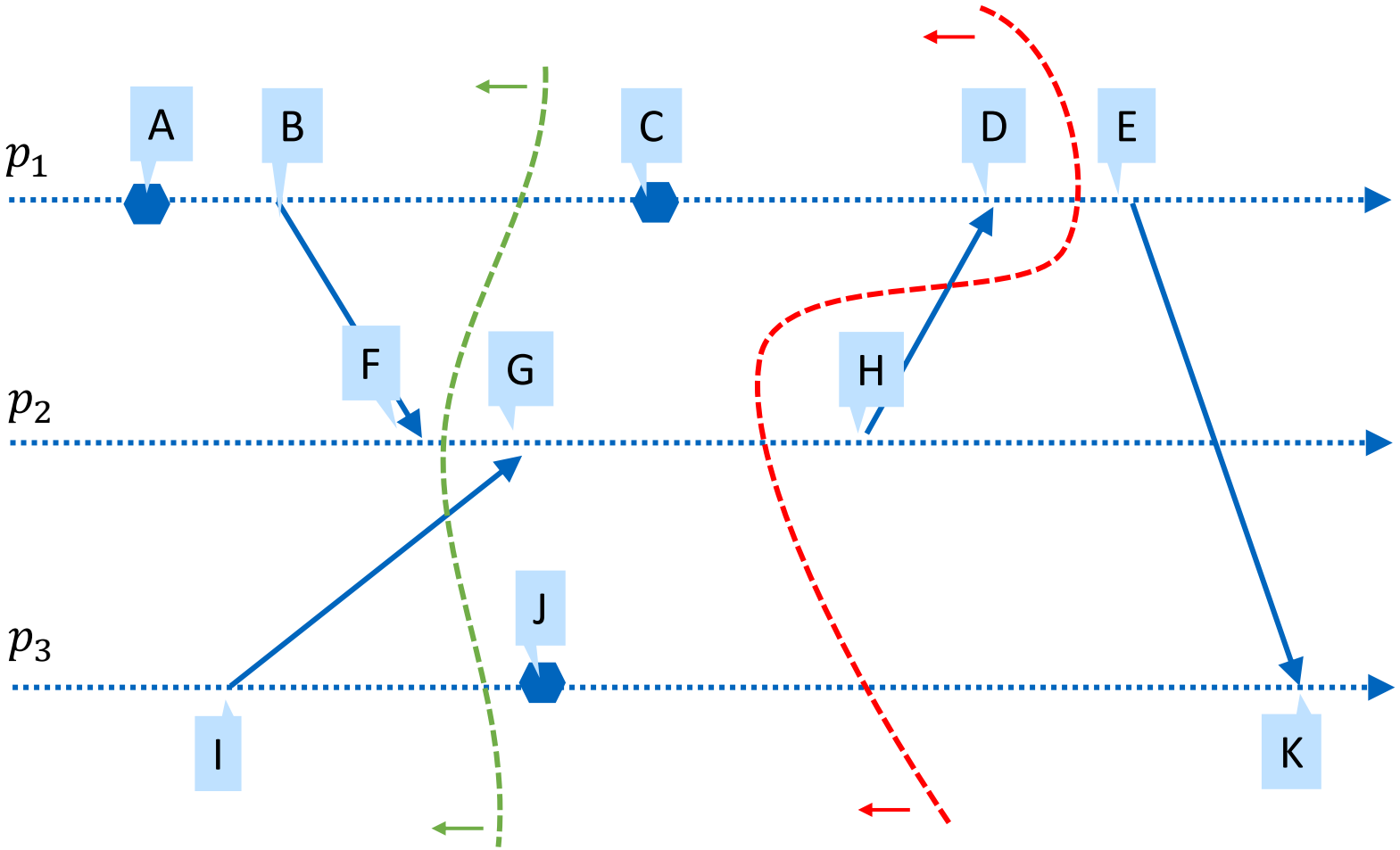
Řez R je konzistentní pokud splňuje kauzalitu, tj. pokud pro každý pár událostí e, f v systému platí:

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

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**

Řez: Příklad



konzistentní řez
(mohl být zpozorován)

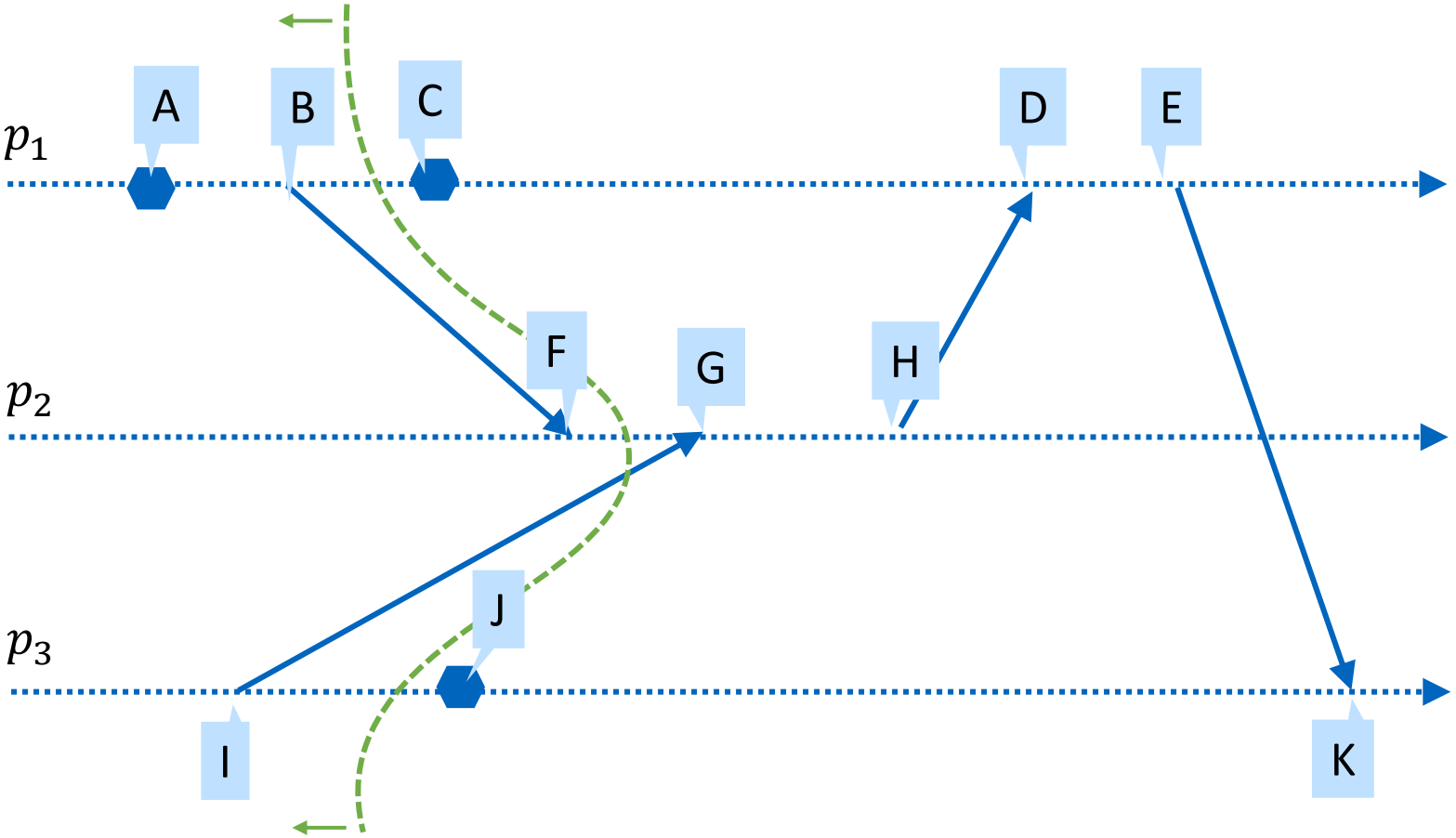
nekonzistentní řez
(*nikdy* nemohl být zpozorován)

Pozorovatelnost řezu

Konzistentní globální snapshot odpovídá konzistentnímu řezu.

- Globální snapshot je konzistentní, pokud by **mohl** (ale nikoliv musel) být zpozorován externím pozorovatelem daného distribuovaného výpočtu.
- Nekonzistentní řez by nemohl **nikdy** být zpozorován externím pozorovatelem daného distribuovaného výpočtu.

Řez: Příklad



tentýž **konzistentní** řez

(ale nemohl být v reálu zpozorován –
F nastalo ve fyzickém čase až po C a J)

Výpočet konzistentního globálního snapshotu

Problém výpočtu globálního snapshotu

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 $p_1, \dots, p_N$
- **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 ▪**

Jeden (libovolný) z procesů iniciuje vytvoření snapshotů.

Procesy reagují na příjem zprávy **ZNAČKA ▪ .**

Chandy-Lamport algoritmus pro globální shapshot

Zahájení tvorby snapshotu

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

Příjem ZNAČKY ▪ procesem p_i kanálem $C_{m,i}$

if (p_i dosud nezaznamenal svůj stav) **then**
- p_i **zaznamená** svůj stav;
- p_i **zaznamená** stav kanálu $C_{m,i}$ jako prázdnou množinu;
- p_i *každým* odchozím **kanálem $C_{i,j}$ odešle** jednu ZNAČKU ▪ (předtím než skrze $C_{i,j}$ pošle jakoukoliv jinou zprávu);
- 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 $C_{m,i}$;
- p_i **ukončí** záznam kanálu $C_{m,i}$;

end if

Ukončení

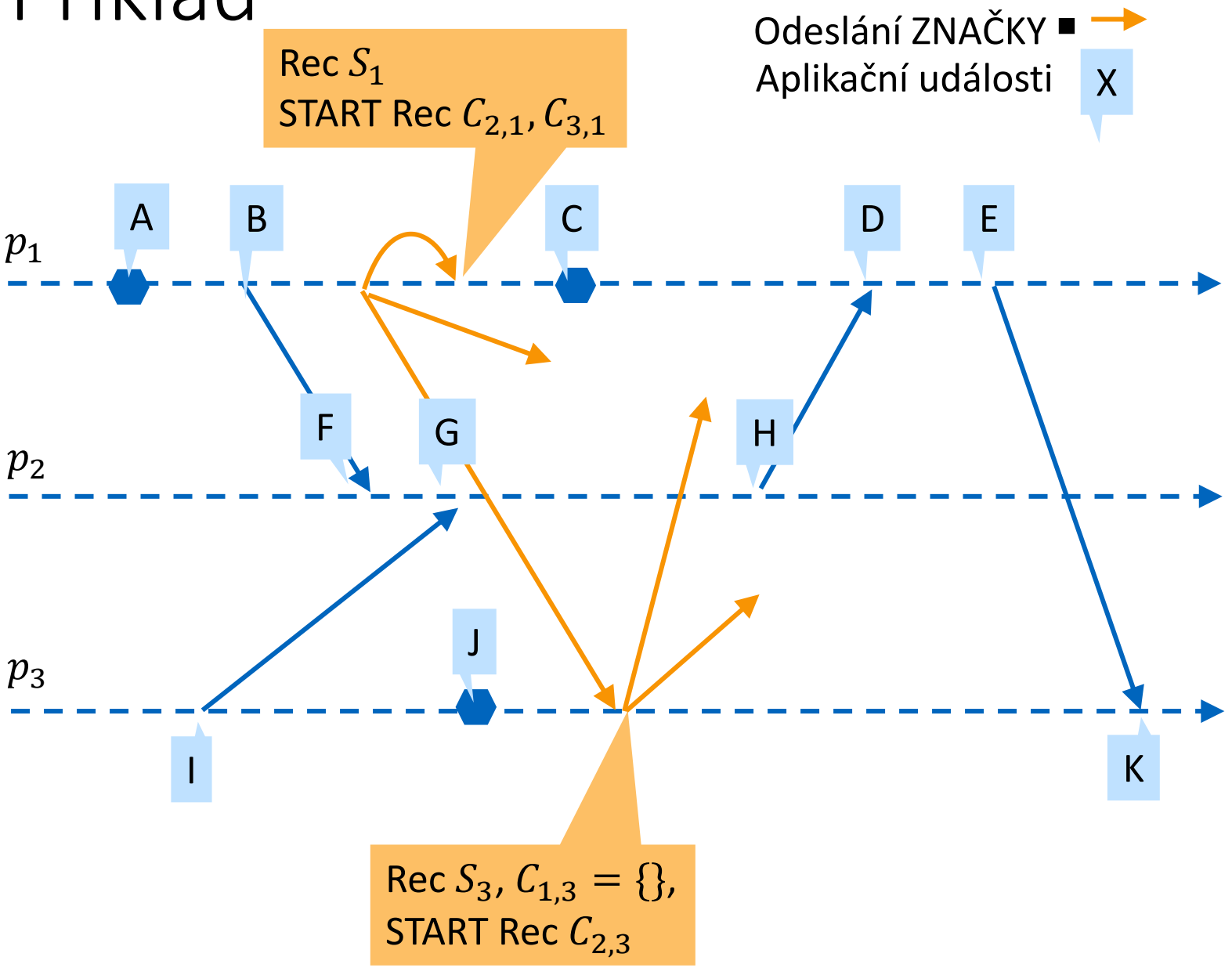
Ukončení algoritmus

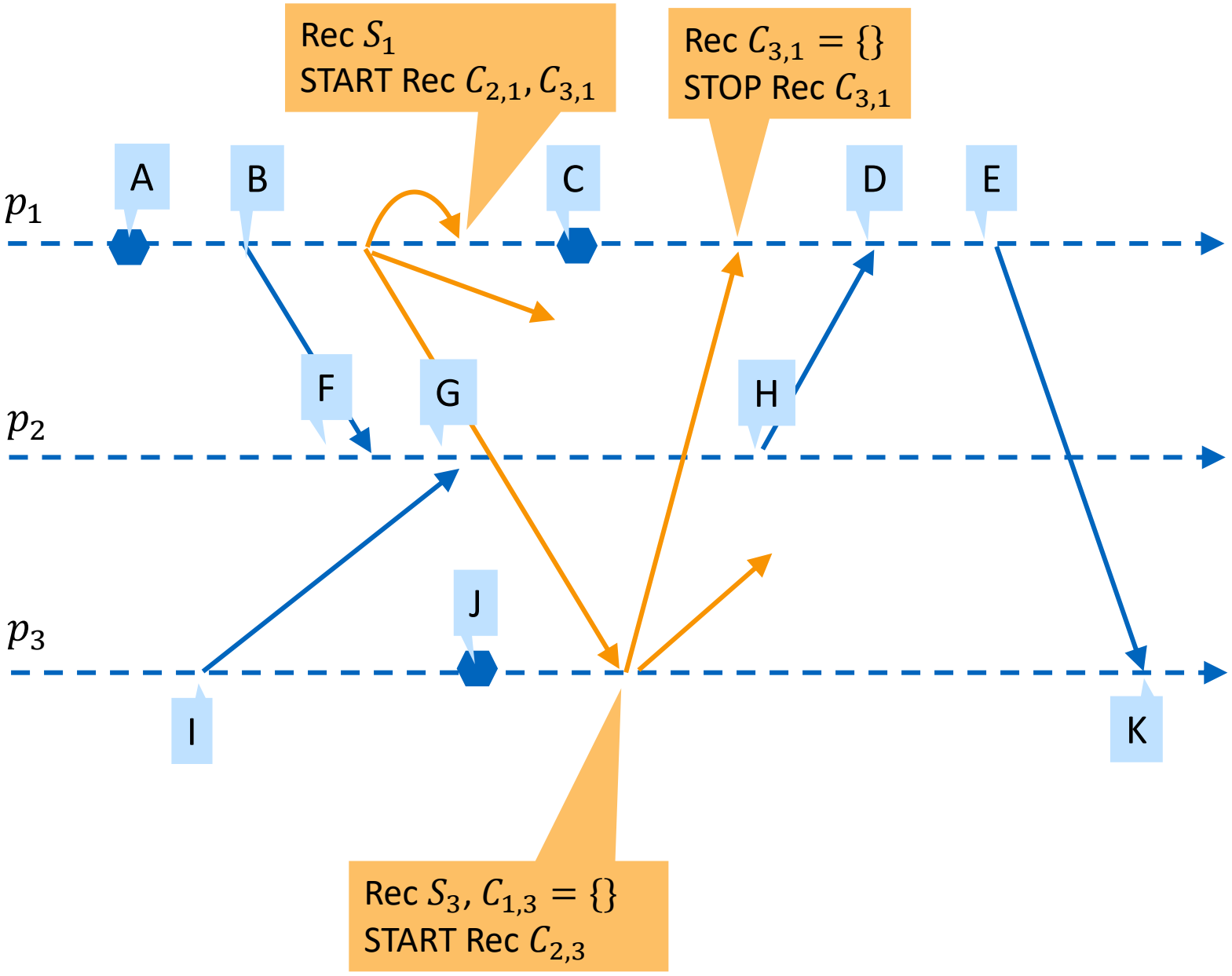
Algoritmus je ukončen jakmile:

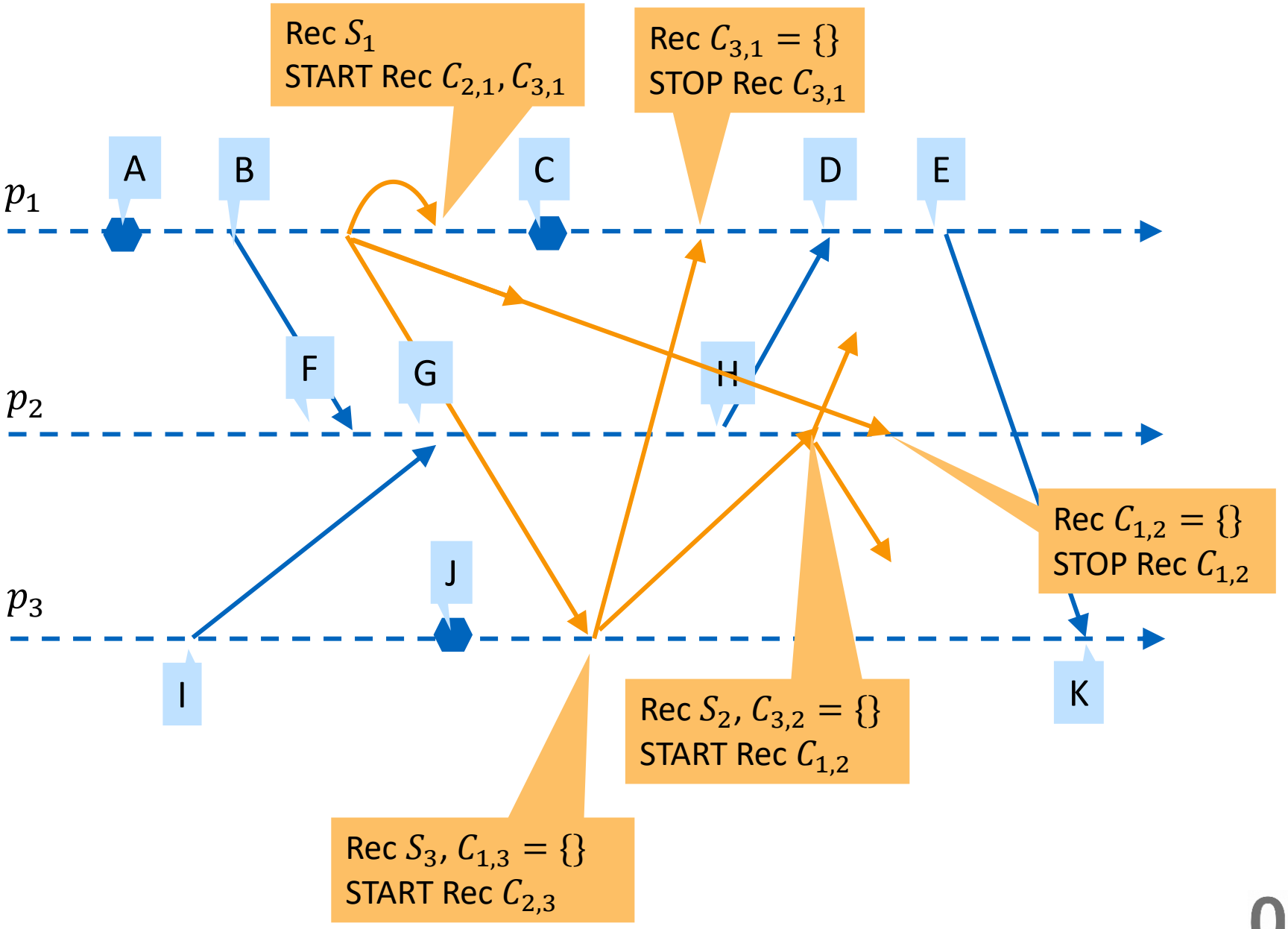
1. všechny procesy zaznamenaly svůj stav
   a
2. všechny procesy přijaly ZNAČKU ▪ na všech $N - 1$ svých příchozích kanálech (aby zaznamenaly stav na všech kanálech)

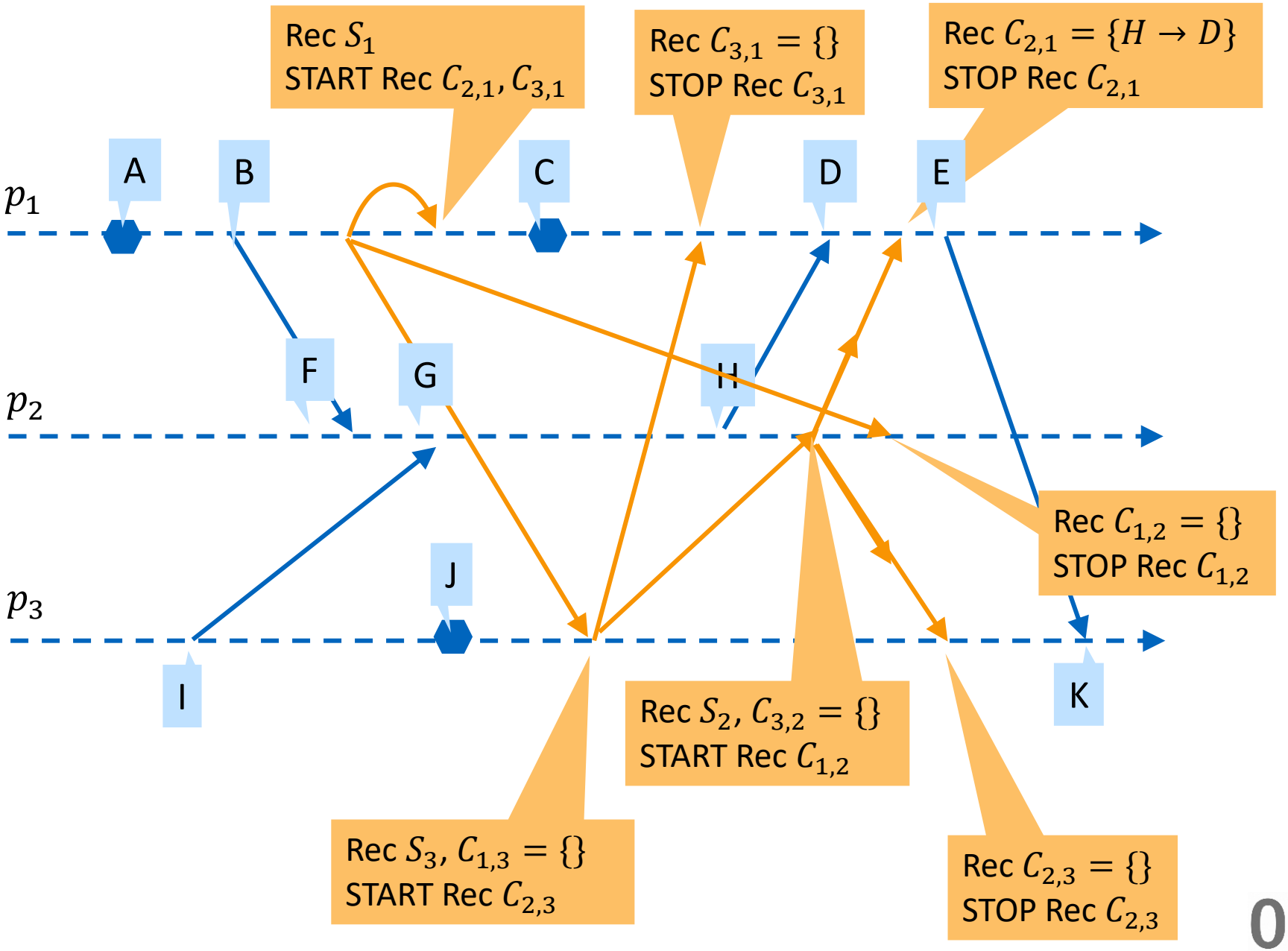
Následně (je-li potřeba) mohou být jednotlivé fragmenty globální stavu posbírány centrálním servery a poskládán **plný globální snaphot**.

Příklad

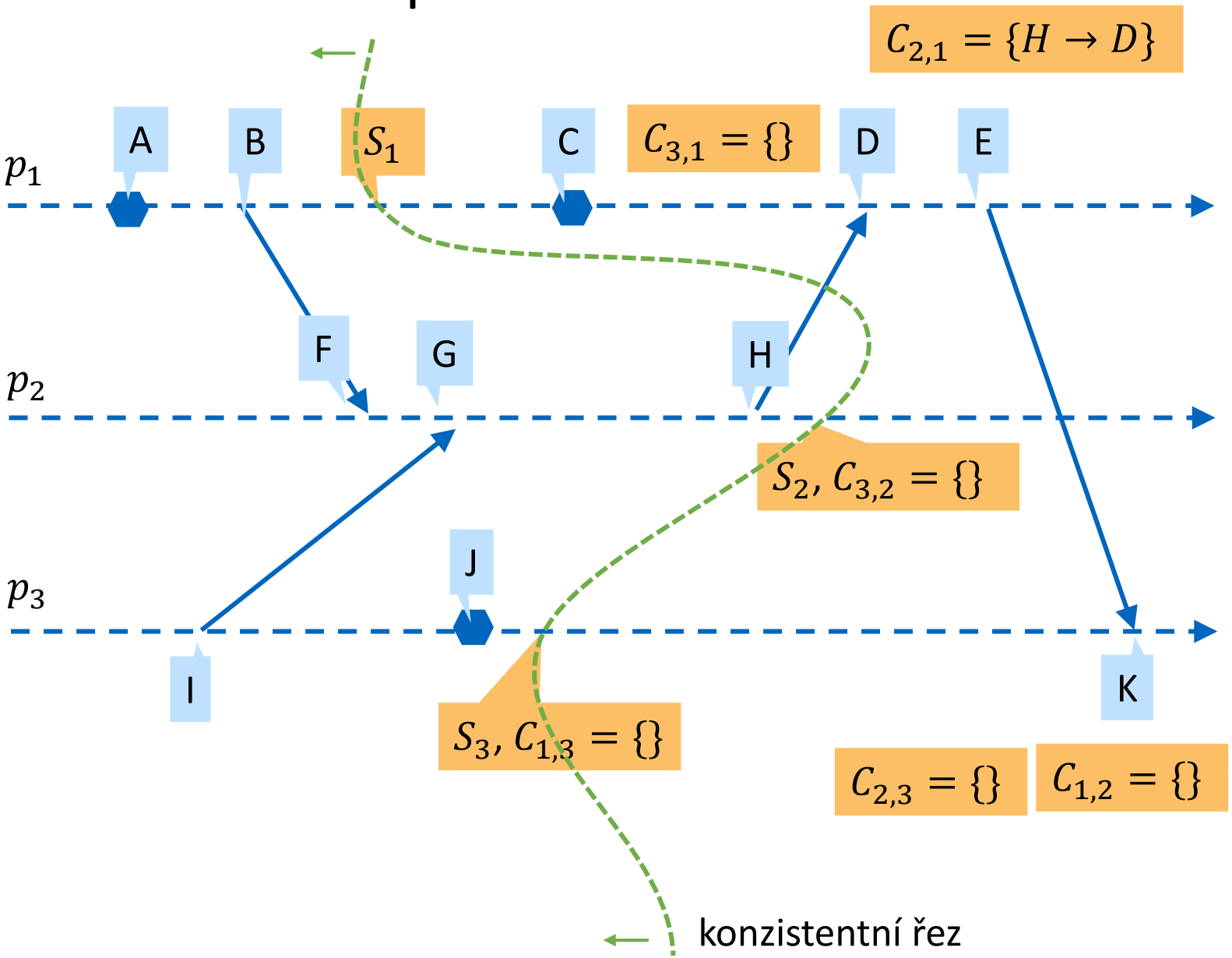

Odeslání ZNAČKY
Aplikační události
X
Rec S_1
START Rec $C_{2,1}, C_{3,1}$
p_1
p_2
p_3
A
B
C
D
E
F
G
H
I
J
K
Rec S_3, $C_{1,3} = \{\}$,
START Rec $C_{2,3}$


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


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


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

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 (a tedy i zaznamenal svůj stav) před e_j
  (tj. nemůže nastat: "P_i zaznamená svůj stav" $\rightarrow e_i \rightarrow e_j \rightarrow$ "P_j zaznamená svůj stav")
- A tedy $e_j \notin R$ → spor

Korektnost distribuovaného výpočtu

Živost (Liveness)

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

Příklady:

- Distribuovaný výpočet: výpočet skončí.
- Úplnost při detekci selhání: každé selhání je časem detekováno.
- Globální snapshot: algoritmus doběhne.

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
- Globální snapshot: Snapshot je konzistentním řezem.

Vyhodnocení stabilních vlastností

Uvažujme **vlastnost distribuovaného výpočtu** jako logickou formuli definovanou nad globálním stavem distribuovaného výpočtu.

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)

Příklad *nestabilní* vlastnosti: ve výpočtu není žádný havarovaný proces, v komunikačních kanálech je právě přenášeno N zpráv, na vstup do kritické sekce čekají právě dva procesy …

Vyhodnocení stabilních vlastností

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

Je daná **stabilní** vlastnost *V* **splněna** v globálním snapshotu zachyceným snapshot algoritmem?

ANO

Vlastnost *V* **bude** ve výpočtu splněna i ve *fyzickém* okamžiku **doběhnutí** snapshot algoritmu.

NE

Vlastnost *V* **nemohla být** ve výpočtu splněna ani ve *fyzickém* okamžiku **zahájení** snapshot algoritmu.

fyzický okamžik zahájení algoritmu < logický okamžik zachyceného snapshotu < fyzický okamžik ukončení algoritmu

Další algoritmy pro globální snapshot

Nigamanth a A.G. Sivilotti: Proces zaznamená svůj snapshot, až pokud obdrží zprávu od procesu, od kterého dříve obdrželo ZNAČKU → méně zpráv v běhu.

Spezialetti-Kearns: Umí se vypořádat se více souběžnými iniciátory, snapshoty neduplikuje, ale částečné snapshots spojí dohromady → rychlejší průběh.

Venkatesan inkrementální snapshot: vhodný pro opakované snapshotování, pamatuji si poslední snapshot a zaznamenává pouze inkrement → nižší počet zpráv.

Dále existují algoritmy pro případy, kdy kanály nejsou FIFO.

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.

Pravidla samostatné práce

1.Kód či text (dále jen kód) každé domácí úlohy musí být vypracován samostatně. Je běžné v programátorské praxi, že autor pro větší efektivitu své práce cizí kód přejímá a/nebo přizpůsobuje. Nicméně, pro studijní účely je takový postup nežádoucí[1)]. Případné použití cizího kódu je obvykle explicitně v zadání úlohy povoleno.

2.Výše uvedené se týká i nežádoucího použití AI nástrojů umožňujících automatizovanou tvorbu kódů jako jsou ChatGPT, GitHub Copilot, Google Bard, apod.

…

1.V případě prvního odevzdaného plagiátu musí autor vypracovat úlohu znovu a samostatně. Řešení musí být funkční. I přesto bude řešení hodnoceno nulovým bodovým (procentuálním) ziskem. Bodové (procentuální) podmínky zápočtu jakož i termín odevzdání řešení se přitom nemění. U jména studenta se pak objeví žlutá karta.

2.V případě opakovaného[2)] odevzdání plagiátu neudělí cvičící jeho autorovi zápočet a přednášející bez odkladu klasifikuje autora stupněm F - nedostatečně z celého předmětu. U jména studenta se pak objeví červená karta.

…

1.Posluchač, který se svým hodnocením nebo klasifikací nesouhlasí, se může obrátit na vedoucího katedry, která předmět zajišťuje, a dále postupovat podle Studijního a zkušebního řádu ČVUT, zejména podle článku 9, odstavce 3. nebo článku 10, odstavce 9.

2.V případě prohřešku držitele červené karty následuje oznámení Disciplinární komisi FEL se žádostí o vyloučení ze studia.

Příští přednáška (pravděpodobně)
v **KN:E-301**