

PDV 09 2021/2022

# Čas a kauzalita v DS

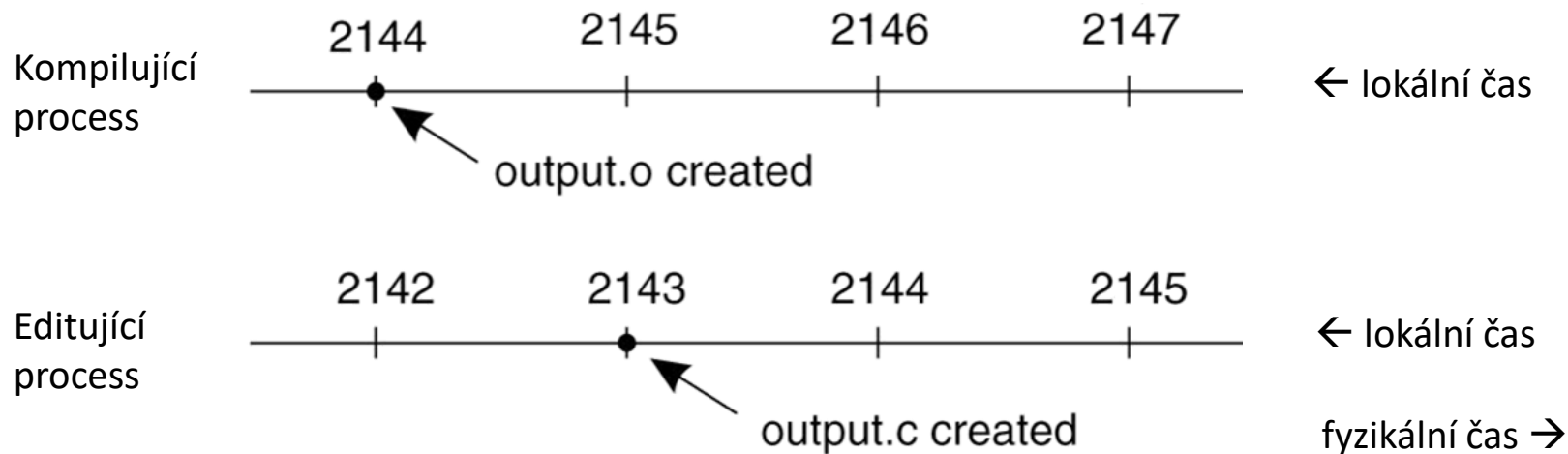
Michal Jakob

[michal.jakob@fel.cvut.cz](mailto:michal.jakob@fel.cvut.cz)

Centrum umělé inteligence, katedra počítačů, FEL ČVUT



# Distribuovaná kompilace



$2143 < 2144 \rightarrow$  kompilace nebude spuštěna

**Důsledek nesynchronizovaného času – chybná verze binárního souboru**



# Synchronizace fyzických hodin



# Mimoběžnost vs. Drift

## Mimoběžnost hodin (clock skew)

Rozdíl v **času** hodin dvou procesů.

(jako vzdálenost dvou jedoucích automobilů)

Mají-li dvoje hodiny nenulovou mimoběžnost, jsou **nesynchronizované**.

## Drift hodin (clock drift)

Rozdíl v **rychlosti** (frekvenci) hodin dvou procesů.

(jako rozdíl v rychlosti jedoucích automobilů)

Mají-li dvoje hodiny **nenulový drift**, tak se jejich mimoběžnost v čase bude (nakonec) **zvyšovat**

- jsou-li popředu pomalejší hodiny, tak nejdříve dojde k vyrovnání času



# Synchronizace

Uvažujeme skupinu procesů

## Externí synchronizace

- Čas  $C_i$  hodin každého procesu  $p_i$  je udržován v rozmezí  $\delta$  od času  $S$  externích **referenčních hodin**, tj. v každém okamžiku  $|C_i - S| \leq \delta$
- Externí hodiny mohou být napojeny na UTC nebo na atomové hodiny
- Algoritmy: např. Cristianův, NTP

Externí synchronizace s rozmezím  $\delta$  **implikuje** interní synchronizaci s rozmezím  $2 * \delta$ .

## Interní synchronizace

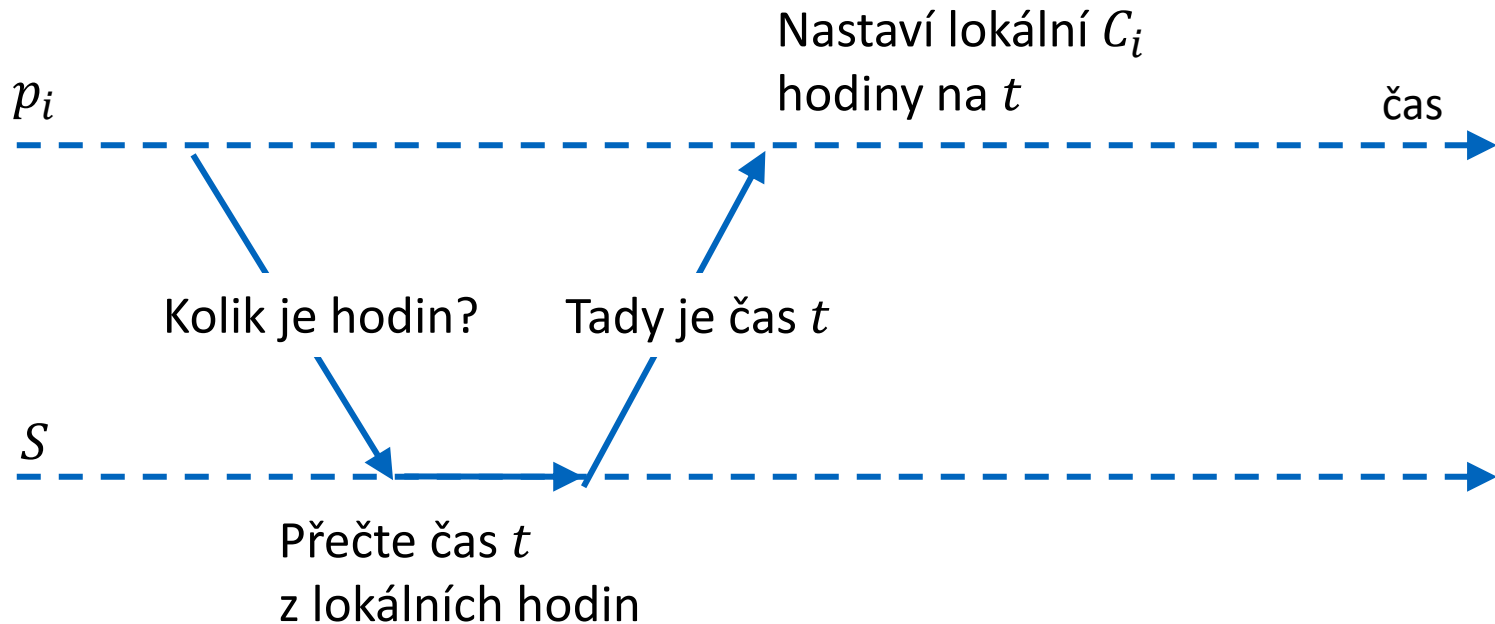
- Každý pár procesů  $(p_i, p_j)$  má hodnoty času svých hodin v rozmezí  $\delta$ , v každém okamžiku tj.  $|C_i - C_j| \leq \delta$
- Algoritmy: např. Berkeley

Interní synchronizace **neimplikuje** externí synchronizaci.



# Synchronizace fyzických hodin

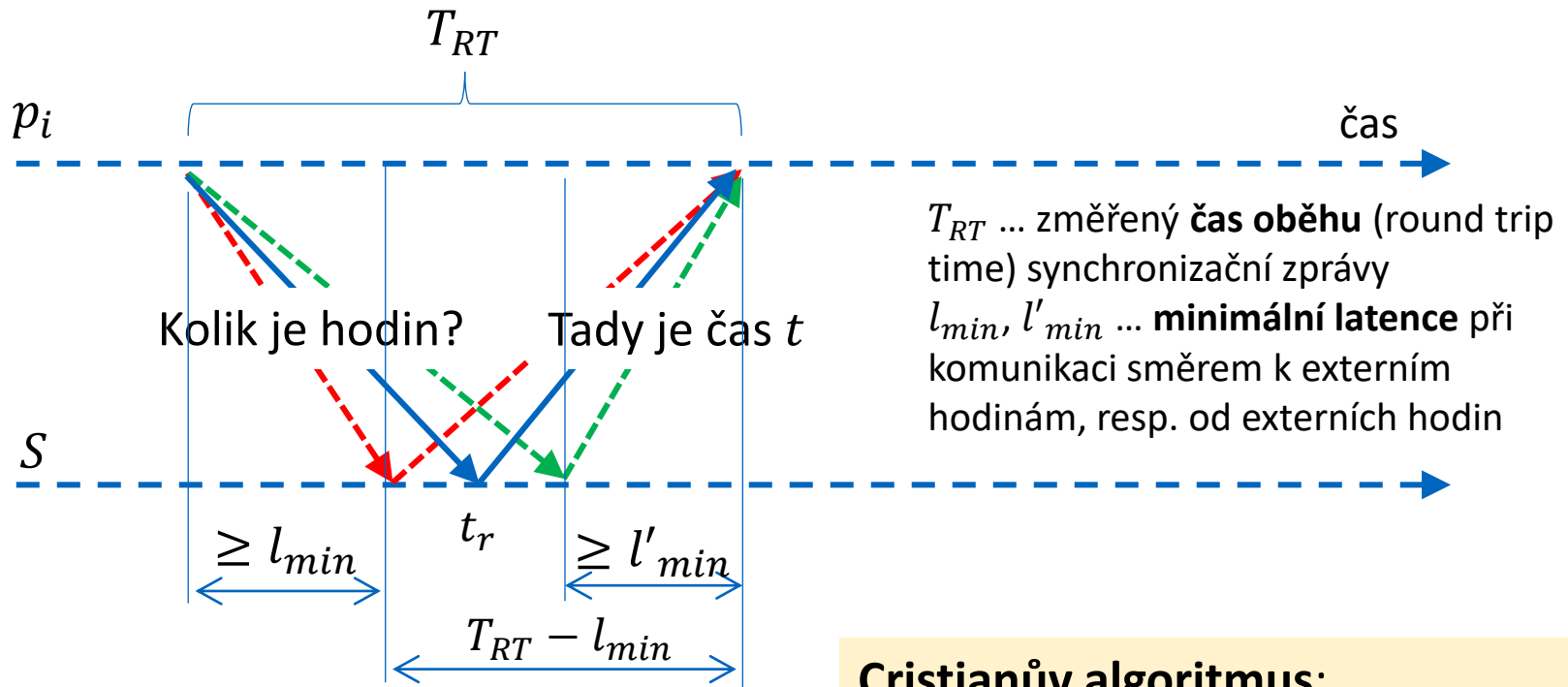
**Externí synchronizace:** všechny procesy  $p_i$  se synchronizují s externím časovým serverem  $S$ .



- ❓ Co je špatně?
- V okamžiku, kdy  $p_i$  obdrží odpověď, se už **čas posunul**.
  - Míra nepřesnosti  $C_i$  závisí na komunikační **latenci**.
  - V asynchronním systému je latence **konečná**, ale **neomezená (a neznámá!)**  $\Rightarrow$  neomezená je i **nepřesnost** hodin.



# Cristianův Algoritmus



Skutečný čas v okamžiku, kdy  $p_i$  přijme odpověď je  $[t_r + l'_{min}, t_r + T_{RT} - l_{min}]$



$t_r$  ... skutečný čas, kdy  $S$  odpověděl na dotaz

**Cristianův algoritmus:**

$$C_i := t + \frac{T_{RT} - l_{min} + l'_{min}}{2}$$

Chyba je **omezena**, tj. maximálně  $(T_{RT} - l_{min} - l'_{min})/2$

(Je-li chyba měření příliš vysoká, je možné poslat více zpráv a výsledek průměrovat.)



# Pozor

Lokální čas je možné posunout libovolně **dopředu**...

...ale **nikoliv dozadu**!

- posun dozadu by mohl narušit lokální uspořádání události v procesu

Je možné zvýšit nebo snížit **rychlost** hodin





# Jak často synchronizovat?

**Maximální rychlost driftu** (maximum drift rate  $MDR$ ) hodin

**Absolutní  $MDR$**  je definováno relativně vůči UTC (universal coordinated time). UTC je **správný (přesný)** čas v každém okamžiku.

Vzájemná maximální rychlost driftu mezi dvěma procesy se stejnou absolutní  $MDR$  je  $2 * MDR$ .

Je-li maximální **tolerovaná mimoběžnost** mezi jakýkoliv párem hodin je  $M$ , pak je třeba hodiny synchronizovat aspoň každých  $M / (2 * MDR)$  časových jednotek.



# Network Time Protocol (NTP)

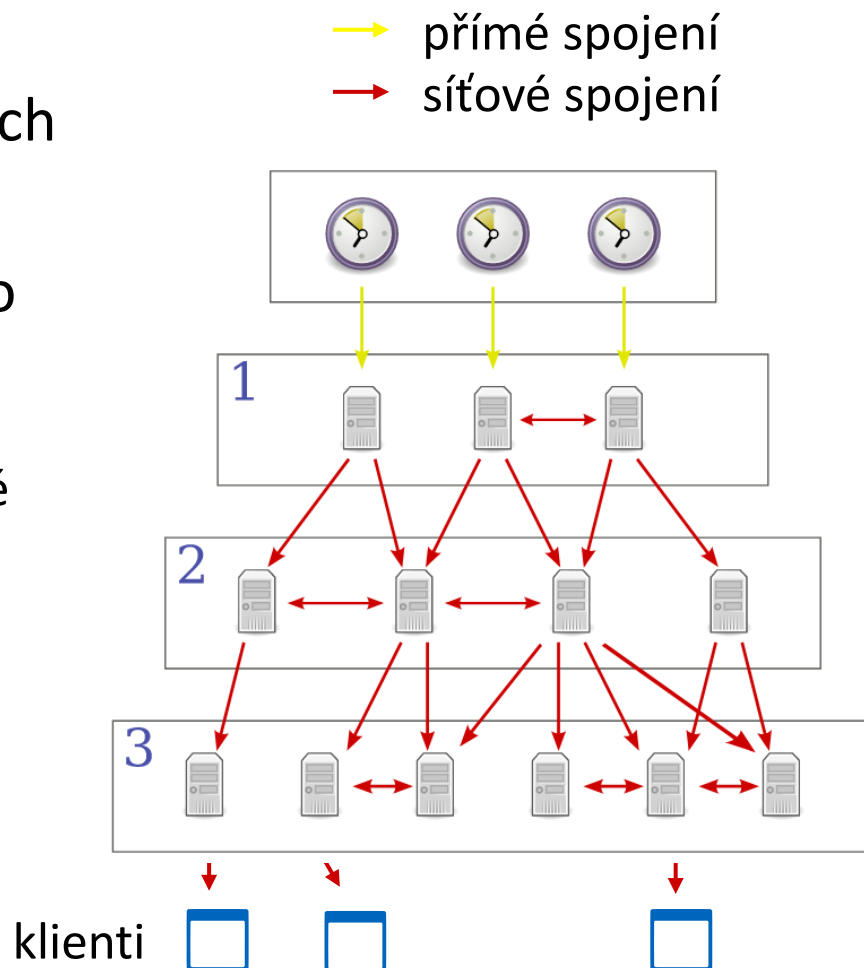
Využíván od roku 1985 pro synchronizaci času v počítačových sítích s proměnlivou latencí.

NTP servery jsou uspořádány do **stromu**

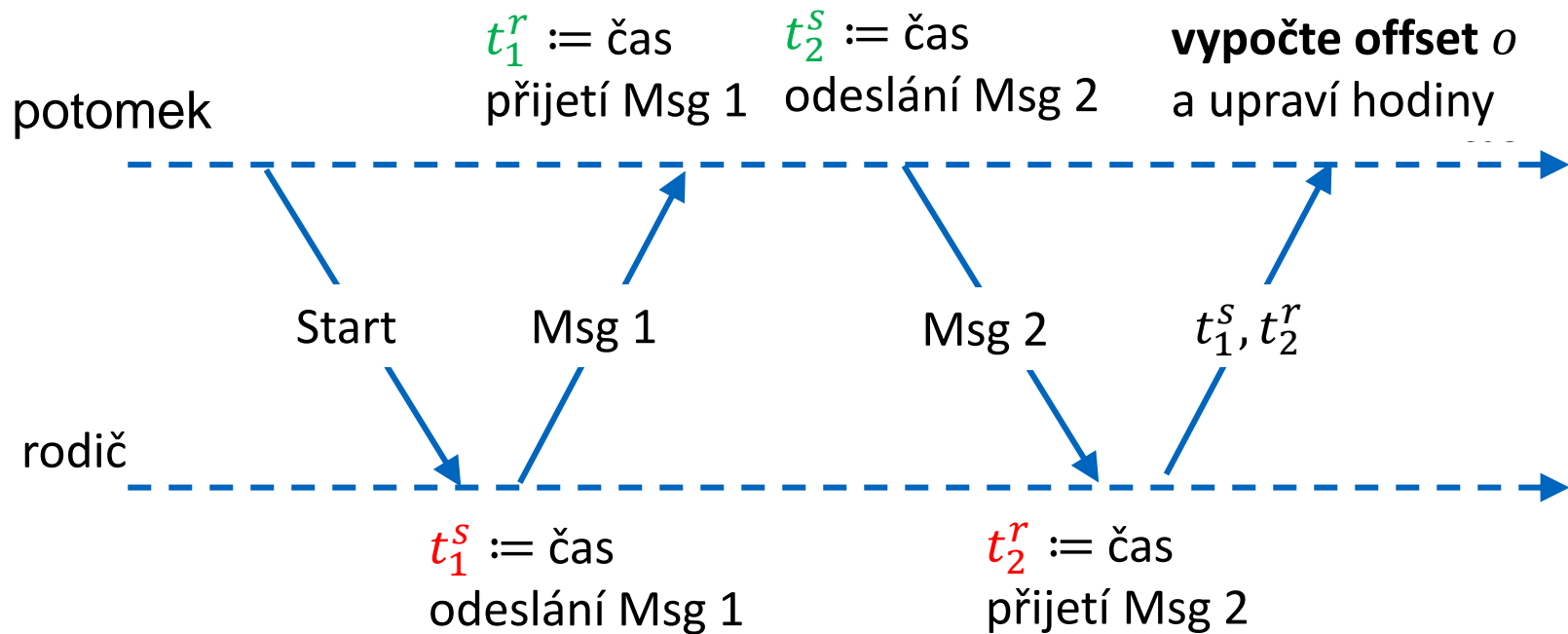
- uzly synchronizují se svými rodiči a někdy i dalšími servery na stejné úrovni
- klienti tvoří listy stromu

Dosažitelná přesnost:

- internet: **desítky ms**
- LAN: **1 ms**



# Základ protokolu



Potomek spočítá **offset** mezi svým časem a časem rodiče:

$$o = \frac{(t_1^r - t_2^r + t_2^s - t_1^s)}{2}$$



# Odvození

$$\text{Proč } o = \frac{(t_1^r - t_2^r + t_2^s - t_1^s)}{2}?$$

Předpokládejme skutečný offset je  $o_{real}$ , tj. čas potomka je popředu o  $o_{real}$  a čas rodiče je pozadu o  $o_{real}$ .

Předpokládejme, že latence pro zprávu 1 je  $l_1$  a pro zprávu 2 je  $l_2$ .

- hodnoty  $l_1$  a  $l_2$  **nejsou známé**

Pak platí

- $t_1^r = t_1^s + l_1 + o_{real}$
- $t_2^r = t_2^s + l_2 - o_{real}$

$$o_{real} = \frac{(t_1^r - t_2^r + t_2^s - t_1^s)}{2} + \frac{l_2 - l_1}{2}$$

$$o_{real} = o + \frac{(l_2 - l_1)}{2}$$

$l_1$  a  $l_2$  jsou  
nezáporné

$$|o - o_{real}| \leq \frac{l_2 + l_1}{2}$$

tj. chyba je omezená **časem oběhu zpráv**



# Nicméně

Stále **nenulová** chyba synchronizace.


Dokud bude **latence nenulová a neznámá**, chyby se nezbavíme!

Pomocí fyzických hodin lze uspořádat jen události v „**pomalých**“ distribuovaných výpočtech

- tj. když intervaly mezi kroky výpočtu trvají výrazně déle než je mimoběžnost hodin
- pro standardně rychlé výpočty bychom potřebovali synchronizaci hodin řádově v nanosekundách

V praxi jsou ale výpočty řádově rychlejší → lze problém uspořádání události vyřešit **bez potřeby synchronizovat fyzické hodiny?**





# Logické hodiny



# Uspořádání událostí v DS

Synchronizace fyzických hodin je jeden z přístupů k uspořádání událostí v DS. Vzhledem k omezené přesnosti synchronizace lze ale použít jen uspořádání událostí, mezi kterými uplyne **dostatečné množství času**.

Alternativní přístup: Co kdybychom místo absolutního/fyzického času přiřazovali událostem **logické časové značky**?

- Pokud by přiřazení logických značek respektovalo **kauzální vztah** mezi událostmi, tak by fungovalo.

**Kauzální vztah** mezi událostmi: první událost může ovlivnit druhou.



# Model

Uvažujeme **asynchronní DS** sestávající z **procesů**.

Každý proces má **stav** (hodnoty **proměnných**).

Každý proces vykonává **akce**, aby změnil svůj stav. Akce může být **instrukce** nebo **poslání zprávy (send, receive)**.

**Událost** je výskyt akce. Poslání zprávy generuje dvě události: **odeslání** a **přijetí**.

Každý proces má **lokální hodiny**.

Události v rámci procesu mohou mít přiřazeny **časové značky (timestamps)**, a mohou tak být linerárně **seřazeny**.

- Ale my potřebujeme řadit globálně v celém DS.



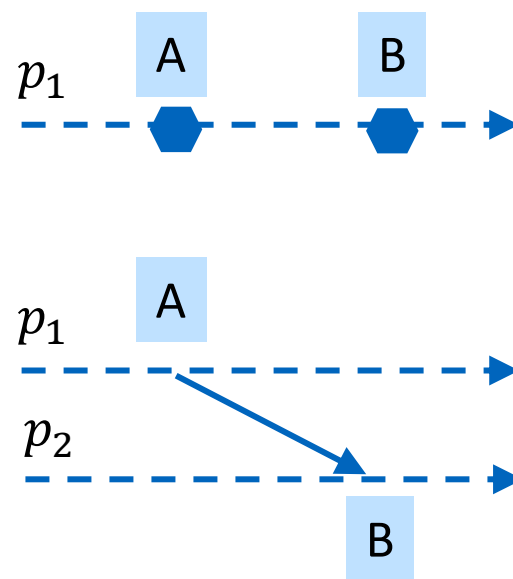


# Relace „stalo se před“

Potenciální kauzalitu zachycujeme pomocí relace „stalo se před“

## Definice (Relace $\rightarrow$ *stalo se před*)

- Jsou-li  $A$  a  $B$  události ve stejném procesu  $p$  a pokud  $A$  předchází  $B$ , pak  $A \rightarrow B$ .
- Je-li  $A$  odeslání zprávy a  $B$  je přijetí této zprávy, pak  $A \rightarrow B$ .
- Pokud  $A \rightarrow B$  a  $B \rightarrow C$ , pak  $A \rightarrow C$



# Kauzální závislost/nezávisost

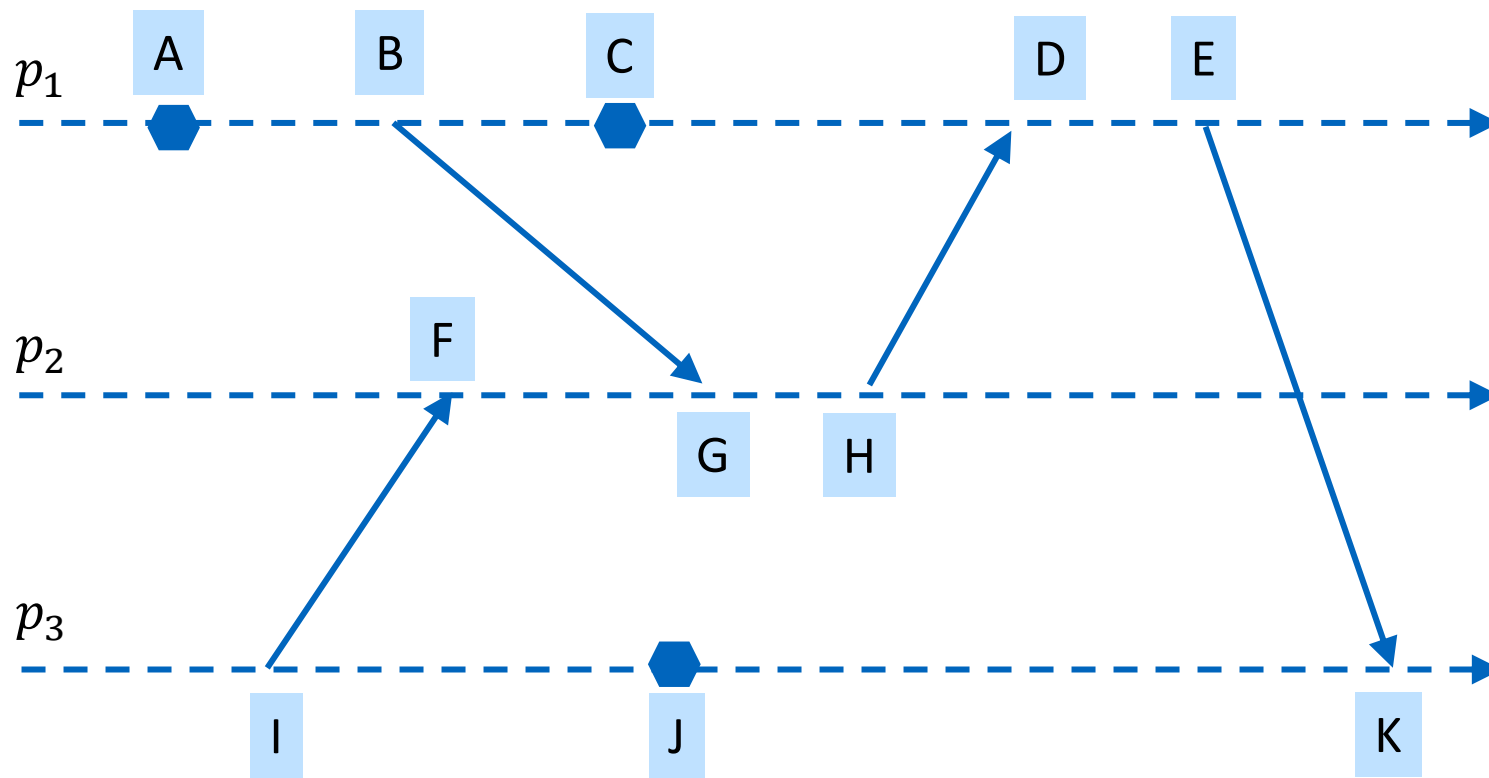
Relace *stalo se před* zavádí **částečné uspořádání událostí** → potenciální **kauzální závislost**

$e_1 \rightarrow e_2$ : potenciálně **kauzálně závislé** události  
(*může* být kauzální vztah, tj.  $e_1$  *mohla* ovlivnit  $e_2$  – ale *nemusela*)

$e_1 \parallel e_2$ : **současné** události  
(kauzální vztah *určitě* není, tj.  $e_1$  nemohla ovlivnit  $e_2$  a  $e_2$  nemohla ovlivnit  $e_1$ )



# Příklad: Stalo se před →



A → B	I → F	C    G
B → G	F → K	A    J
A → G	A → K	C    H
	C → K	

⬡ instrukce  
→ zpráva

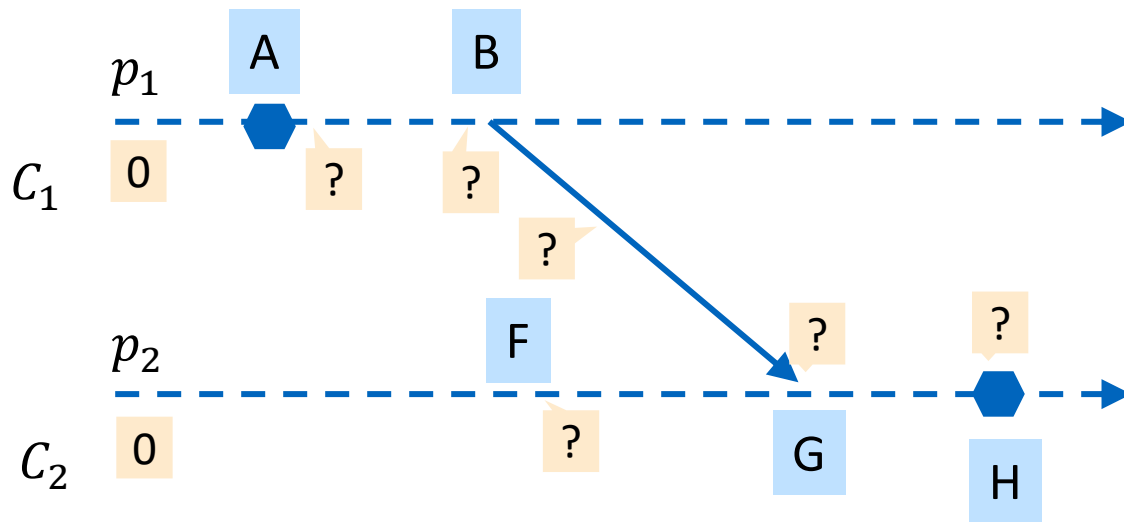
# Lamportovy logické hodiny

Jak přiřadit události  $e$  časovou značku  $C(e)$  tak, aby **respektovaly kauzalitu**, tj. jestliže  $e_1 \rightarrow e_2$ , pak  $C(e_1) < C(e_2)$ ?

→ **Lamportovy logické hodiny**: Každý proces má své logické hodiny, které se **synchronizují podle přijímaní zpráv**.

- navržené Leslie Lamportem v 70. letech
- používané prakticky ve všech distribuovaných systémech (a všech cloudových platformách)

# Příklad: Lamportovy hodiny



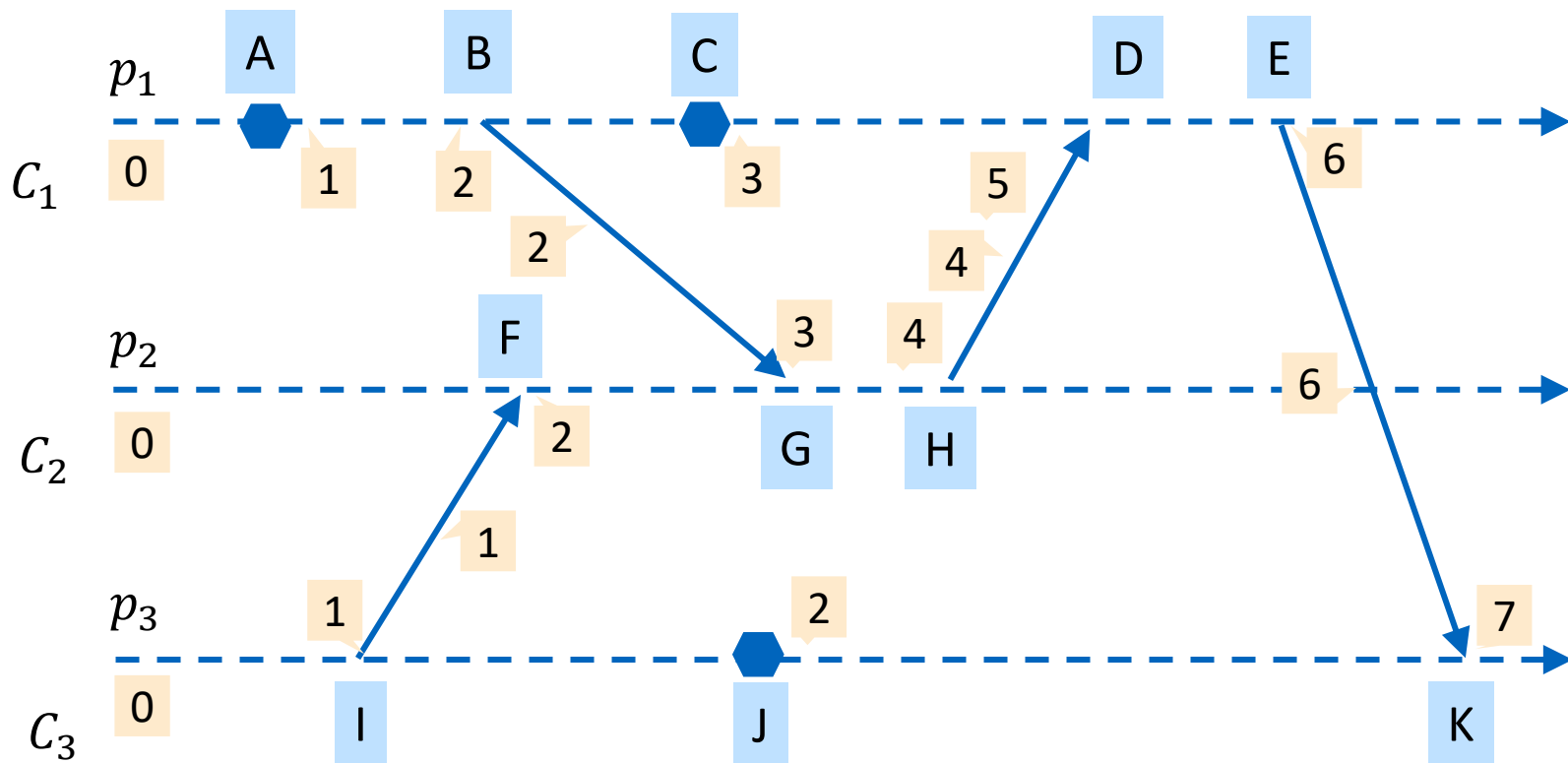
# Synchronizace logických hodin

## Synchronizace logických hodin

Každý proces  $p_i$  si udržuje lokální logické hodiny  $C_i$  a nastavuje

1. Po každé události, která se odehraje v  $p_i$ , se  $C_i$  **inkrementuje** o 1.
2. Každé zprávě  $m$  **odeslané** procesem  $p_i$  je přiřazena časová značka  $ts(m) = C_i$ .
3. Kdykoliv proces  $p_j$  přijme zprávu  $m$ , tak
  - I. upraví své lokální hodiny  $C_j$  na  $\max\{C_j, ts(m)\}$ ; a poté
  - II. provede krok 1 předtím, než předá  $m$  aplikaci ( $\Leftarrow$  přijetí zprávy je událost)

# Příklad: Lamportovy hodiny



A → B	1 < 2	G → H	3 < 4
B → G	2 < 3	F → K	2 < 7
A → G	1 < 3	H → K	4 < 7
		C → K	3 < 7

C    G?	3 = 3
A    J?	1 < 2
C    H?	3 < 4

Lamportovy hodiny  
neimplikují kauzalitu!

# Logické hodiny a kauzalita

Pár **současných** událostí **nemá kauzální cestu** od jedné události ke druhé (ani jedním směrem).

Lamportovy časové značky pro současné události mohou, ale **nemusí** mít stejnou hodnotu!

- **Platí:** jestliže  $e_1 \rightarrow e_2$ , pak  $C(e_1) < C(e_2)$
- **Neplatí:** jestliže  $C(e_1) < C(e_2)$ , pak  $e_1 \rightarrow e_2$

Jak zajistit, že z **uspořádání** časových značek jednoznačně poznáme potenciální **kauzalitu**?





# Vektorové hodiny

# Vektorové hodiny

Předpokládejme skupinu procesů  $\{p_1, \dots, p_N\}$

Každý proces si udržuje **vektor** celočíselných hodin  $V_i[1 \dots N]$

- $j$ -tý element vektorových hodin procesu  $V_i[j]$  je znalost procesu  $i$  o událostech v procesu  $j$

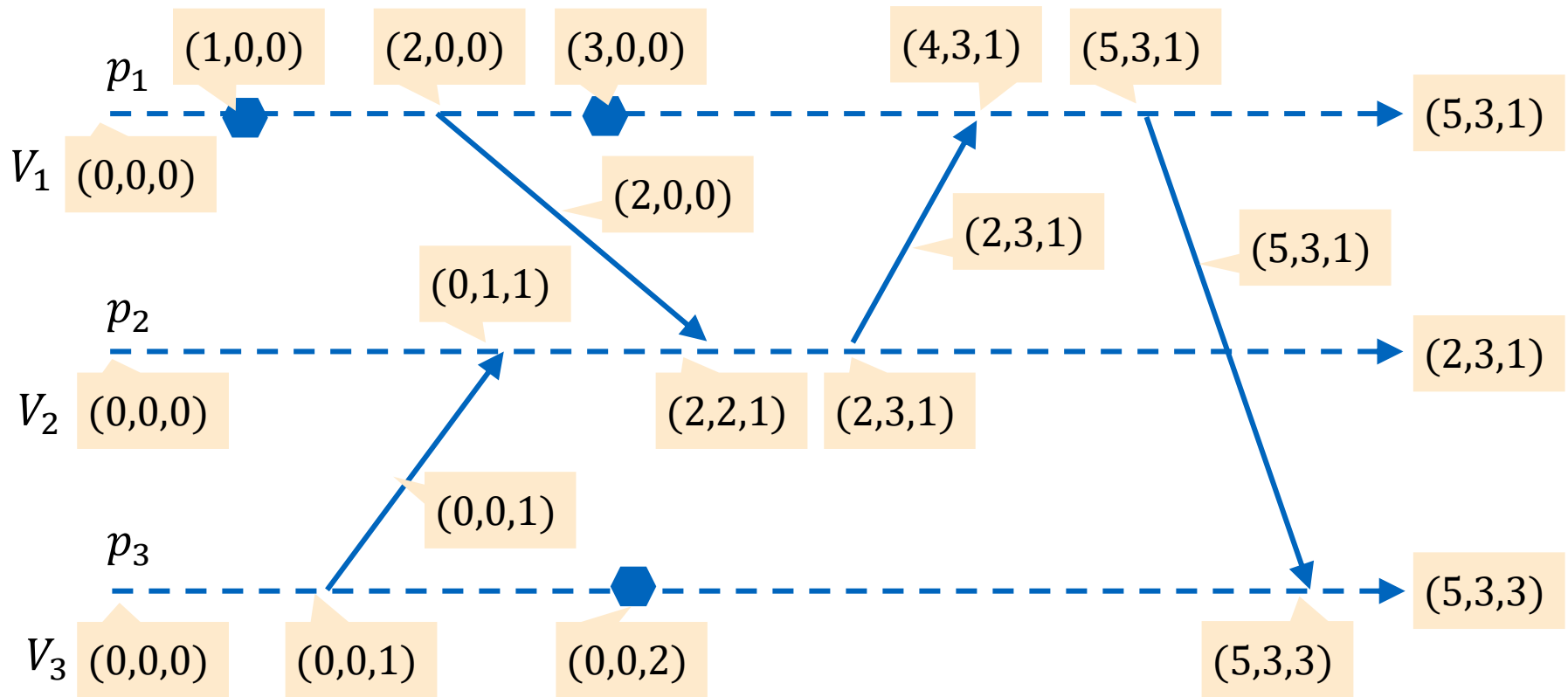
# Synchronizace vektorových hodin

## Synchronizace vektorových hodin

Každý proces  $p_i$  si udržuje **lokální vektorové hodiny**  $V_i$  a nastavuje

1. Před **provedením akce** v procesu  $p_i$  se  $V_i$  **inkrementuje** o 1, tj.  $V_i[i] := V_i[i] + 1$
2. **Pošle-li** proces  $p_i$  zprávu  $m$  procesu  $p_j$ , **nastaví vektorovou časovou značku**  $ts(m)$  zprávy  $m$  na  $V_i$  (poté, co provedl krok 1)
3. Proces  $p_j$  po **přijetí** zprávy  $m$ 
  - nastaví své hodiny  $V_j[k] := \max(V_j[k], ts(m)[k])$  pro všechna  $k = 1, \dots, N$  (tzv. **sloučení**)
  - poté **inkrementuje**  $V_j[j]$  a předá zprávu  $m$  aplikaci.

# Vektorové hodiny: příklad



# Vektorové hodiny a kauzalita

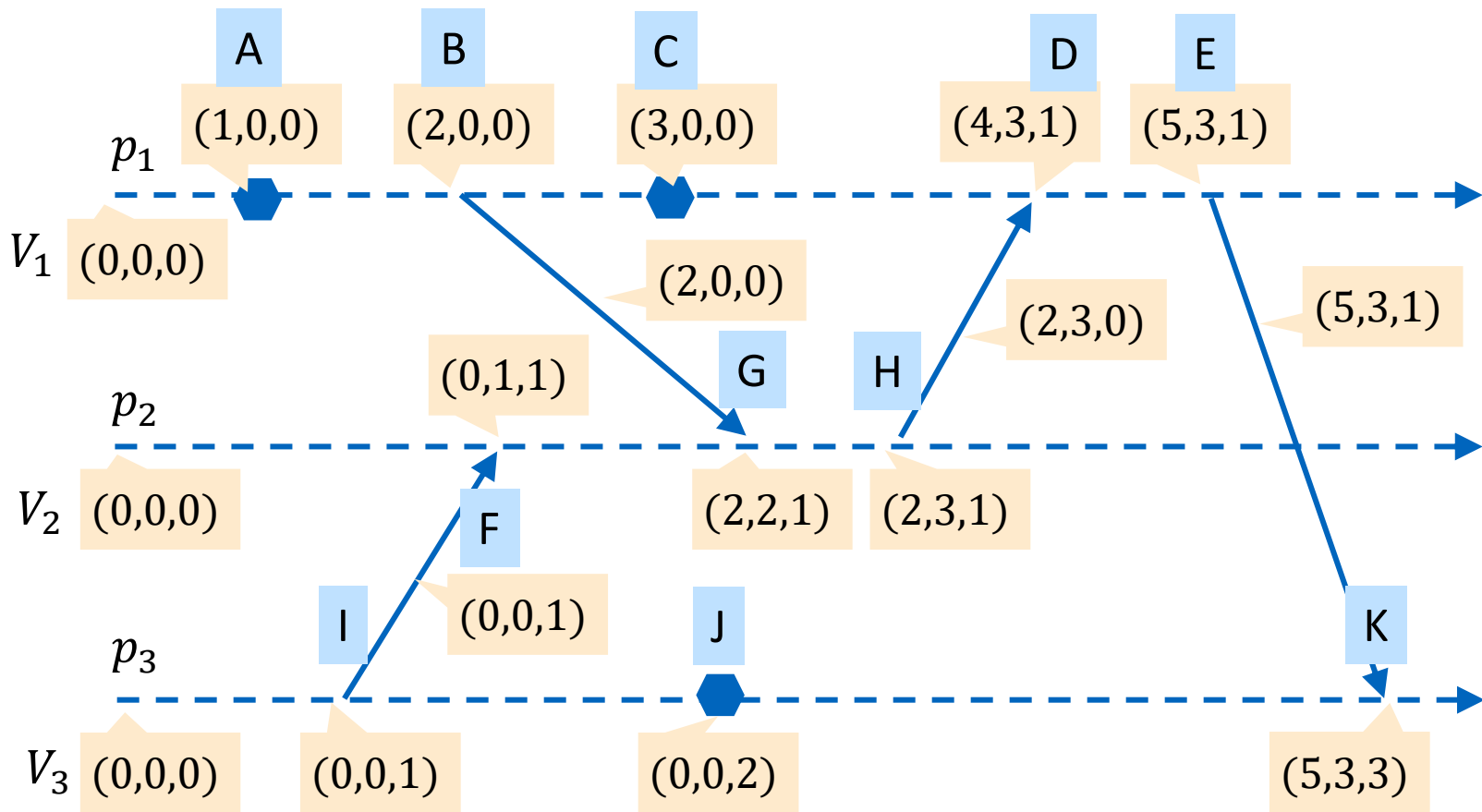
## Upořádání vektorových časových značek:

- $V_1 = V_2$  iff  $V_1[i] = V_2[i]$  pro všechna  $i = 1, \dots, N$
- $V_1 < V_2$  iff  $V_1[i] \leq V_2[i]$  pro všechna  $i = 1, \dots, N$  a existuje  $j$ , že  $V_1[j] < V_2[j]$
- Pokud:  $\neg(V_1 < V_2) \wedge \neg(V_2 < V_1)$ , pak píšeme  $V_1 \parallel V_2$

Pro vektorové hodiny platí:

$e_1 \rightarrow e_2$  iff pro časové události platí:  $V_{e_1} < V_{e_2}$

# Vektorové hodiny: kauzalita



$A \rightarrow B$   $(1,0,0) < (2,0,0)$

$B \rightarrow G$   $(2,0,0) < (2,2,1)$

$A \rightarrow G$   $(1,0,0) < (2,2,1)$

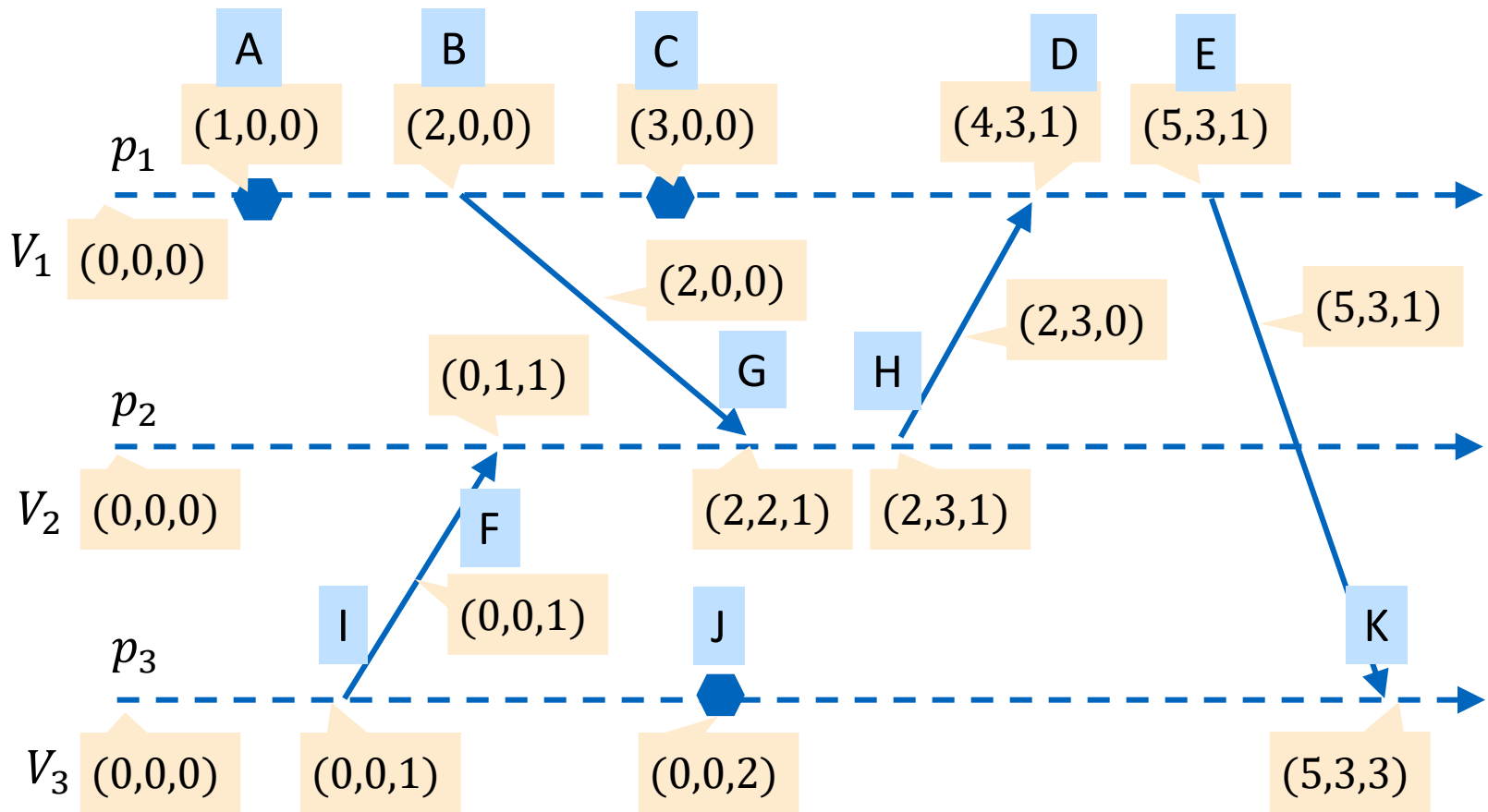
$I \rightarrow H$   $(0,0,1) < (2,3,1)$

$F \rightarrow K$   $(0,1,1) < (5,3,3)$

$I \rightarrow K$   $(0,0,1) < (5,3,3)$

$C \rightarrow K$   $(3,0,0) < (5,3,3)$

# Souběžné události



$C \parallel G$  (3,0,0)  $\parallel$  (2,2,1)  
 $A \parallel J$  (1,0,0)  $\parallel$  (0,0,2)  
 $C \parallel H$  (3,0,0)  $\parallel$  (2,3,1)

vektorové hodiny rozliší  
souběžné události

# Logické hodiny: Souhrn

## Lamportovy (skalární) logické hodiny

- respektují **kauzalitu**
- ale nerozliší současné události

## Vektorové hodiny

- implikují kauzalitu
- potřebují více místa, ale rozliší **současné události**

Lze dokázat, že pro zachycení kauzality ve skupině  $N$  procesů je potřeba **vektorové hodiny délky  $N$**  pro každý proces

- Existují algoritmy, které redukují množství dat potřebných pro udržbu vektorových hodin (např. Raynal a Singhal, 1996)



# Čas v DS: Shrnutí

Přesně **synchronizované globální** hodiny v DS (s nenulovou latencí přenosu zpráv) **neexistují**.

Lze synchronizovat s určitou přesností:

- Cristianův algoritmus
- NTP
- Berkely algoritmus

... ale chyba je **nenulová** a je funkci **doby oběhu zprávy** (RTT)

Nutností synchronizovat hodiny se můžeme vyhnout využitím **logických hodin**.