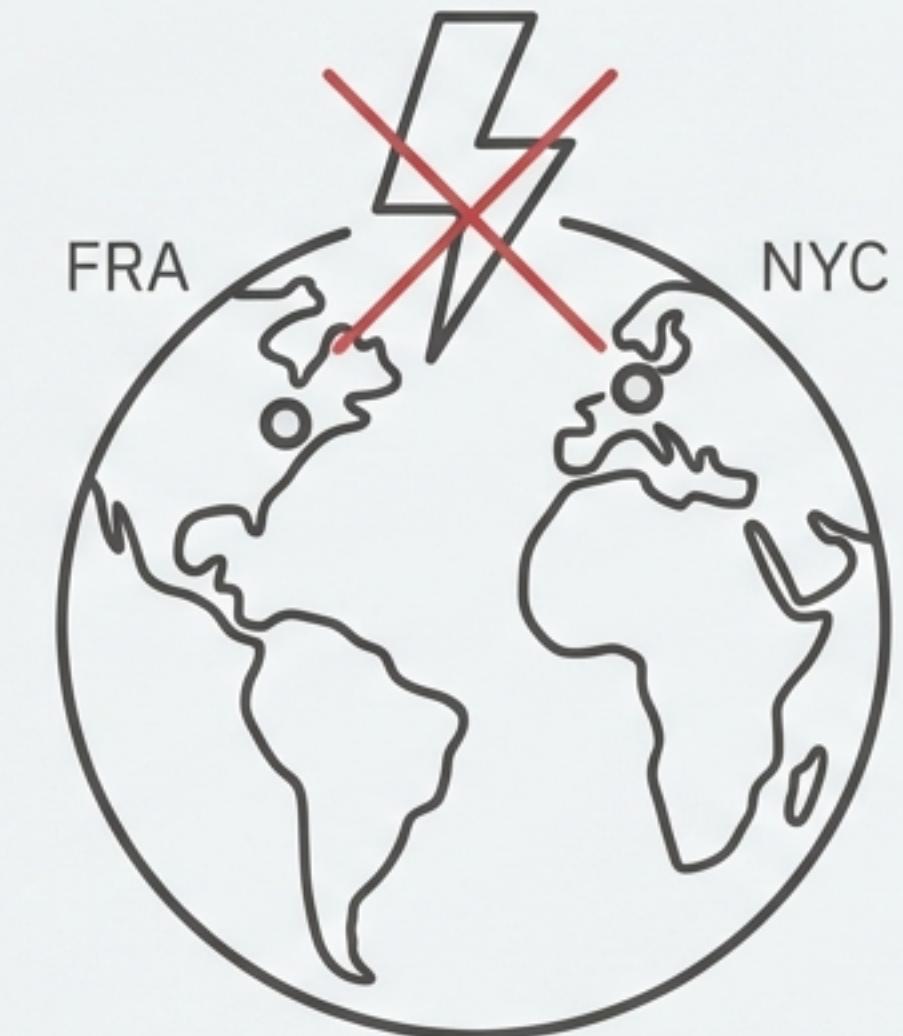


# Problem: Nie istnieje uniwersalne „teraz”

- Scenariusz rezerwacji lotu: Dwie osoby – jedna we Frankfurcie, druga w Nowym Jorku – klikają „rezerwuj” w tej samej chwili. Kto był pierwszy?
- W systemach rozproszonych, składających się z odseparowanych przestrzeni procesów wymieniających komunikaty, tradycyjne pojęcie czasu zawodzi.
- Brak jednego, uniwersalnego zegara dla globalnej sieci komputerów.
- Opóźnienia w transmisji wiadomości są niepomijalne, co sprawia, że pojęcie „jednoczesności” traci sens.
- Fundamentalne wyzwanie: Jak ustalić kolejność zdarzeń między rozproszonymi węzłami?



# Przełom Lamporta: Przyczynowość zamiast czasu zegarowego

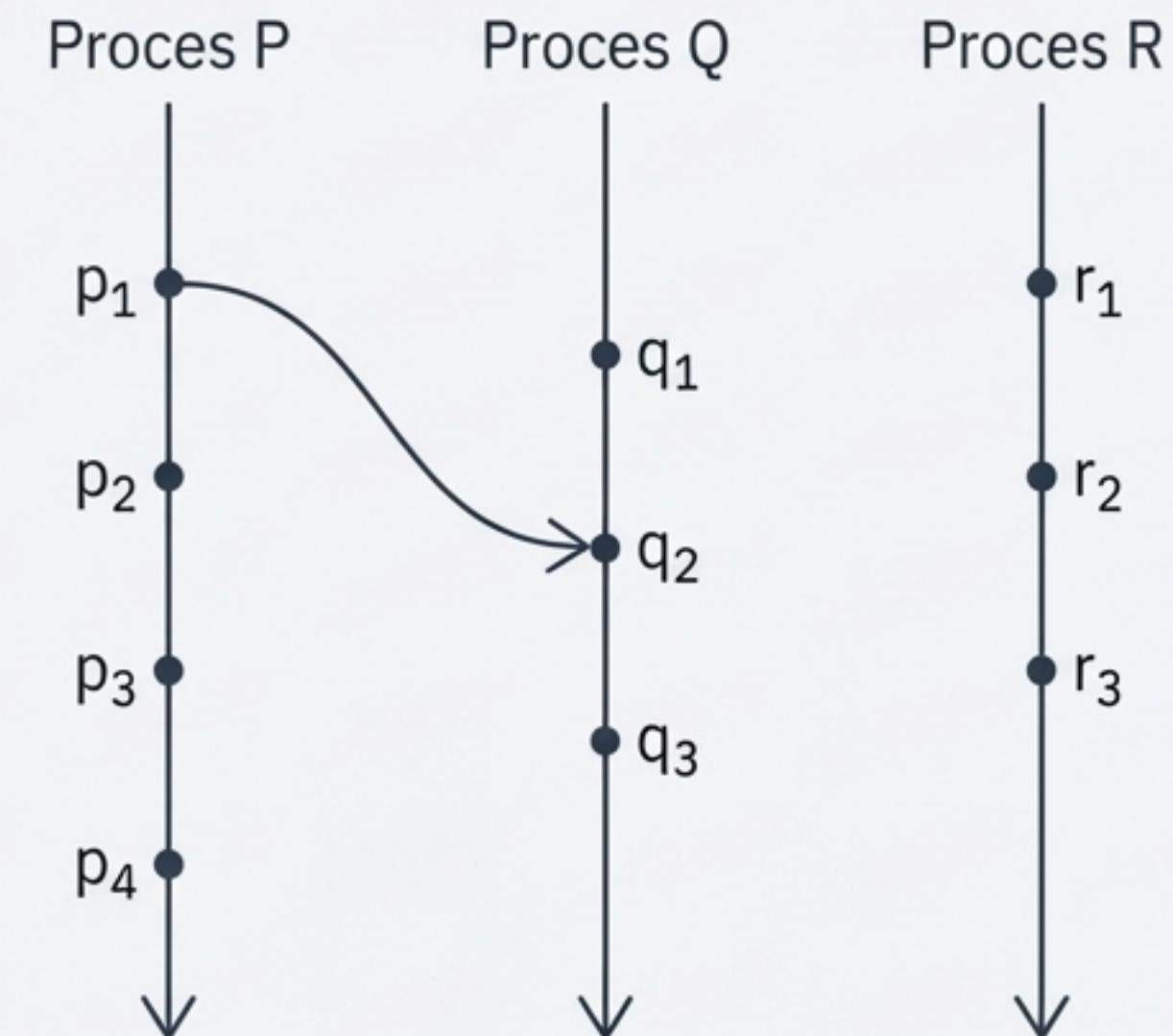
**Kluczowa idea:** Zamiast pytać „o której godzinie?”, pytajmy „co mogło być przyczyną czego?”.

Relacja „zdarzyło się przed” (oznaczana jako  $\rightarrow$ ) definiuje **porządek częściowy (partial ordering)** zdarzeń.

Zgodnie z definicją Lamporta, jest to najmniejsza relacja spełniająca trzy warunki:

- 1. Wewnątrz procesu:** Jeśli zdarzenia  $a$  i  $b$  zachodzą w tym samym procesie i  $a$  występuje przed  $b$ , to  $a \rightarrow b$ .
- 2. Między procesami:** Jeśli  $a$  jest wysłaniem wiadomości przez jeden proces, a  $b$  jest jej odebraniem przez inny, to  $a \rightarrow b$ .
- 3. Przechodniość:** Jeśli  $a \rightarrow b$  oraz  $b \rightarrow c$ , to  $a \rightarrow c$ .

Porządkować można tylko zdarzenia połączone związkiem przyczynowo-skutkowym.



# Zdarzenia współbieżne: Gdy nie ma związku przyczynowego

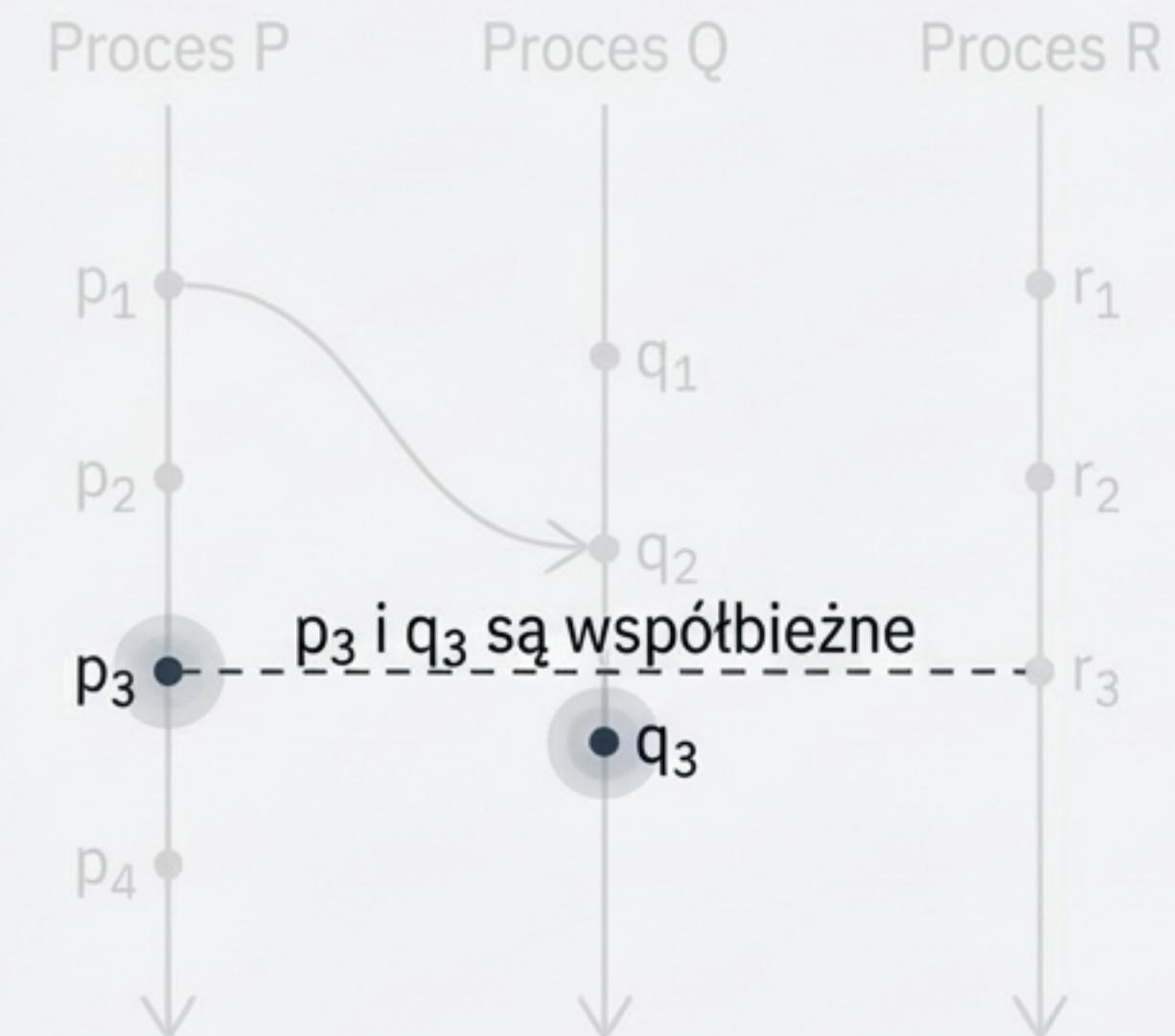
Dwa różne zdarzenia  $a$  i  $b$  są współbieżne (concurrent), jeśli ani  $a \rightarrow b$ , ani  $b \rightarrow a$ .

Oznacza to, że nie mogły na siebie przyczynowo wpływać.

System nie jest w stanie stwierdzić, które z nich wydarzyło się pierwsze w sensie fizycznym.

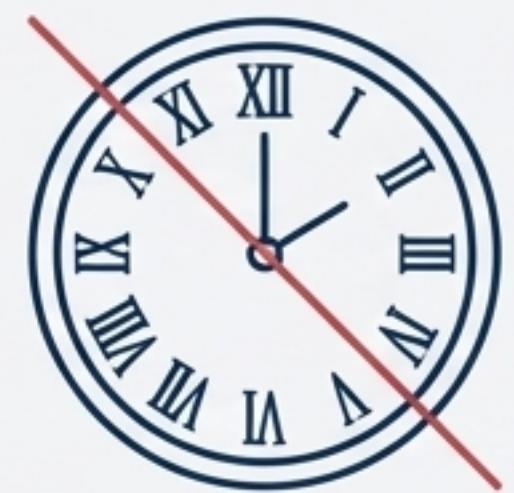
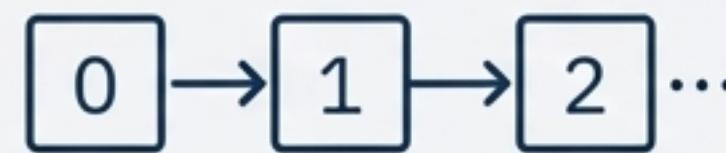
Przykład: Rezerwacje z Frankfurtu i Nowego Jorku są zdarzeniami współbieżnymi, jeśli nie wymieniły między sobą żadnych informacji.

System musi być zaprojektowany tak, by jawnie radzić sobie z tą niejednoznacznością.



# Zegary logiczne: Implementacja porządku przyczynowego

- Jak zaimplementować relację  $\rightarrow$ ? Każdy proces  $P_i$  utrzymuje własny zegar logiczny  $C_i$  – prosty, rosnący licznik.
- Cel: przypisywanie numerów zdarzeniom w celu odzwierciedlenia przyczynowości, a nie mierzenia sekund.
- **Warunek Zegara (Clock Condition):** Dla dowolnych zdarzeń  $a, b$ : jeśli  $a \rightarrow b$ , to musi zachodzić  $C(a) < C(b)$ .
- Wartość zegara odzwierciedla historię przyczynową, a nie czas fizyczny. Zdarzenia współbieżne mogą mieć dowolną relację czasową.



# Algorytm: Dwie proste zasady aktualizacji zegara

Aby zagwarantować spełnienie Warunku Zegara, procesy muszą przestrzegać dwóch reguł implementacyjnych (IR1 i IR2):

**Zasada 1 (Zdarzenia lokalne):** Każdy proces  $P_i$  inkrementuje swój zegar  $C_i$  między dwoma kolejnymi zdarzeniami w tym procesie.

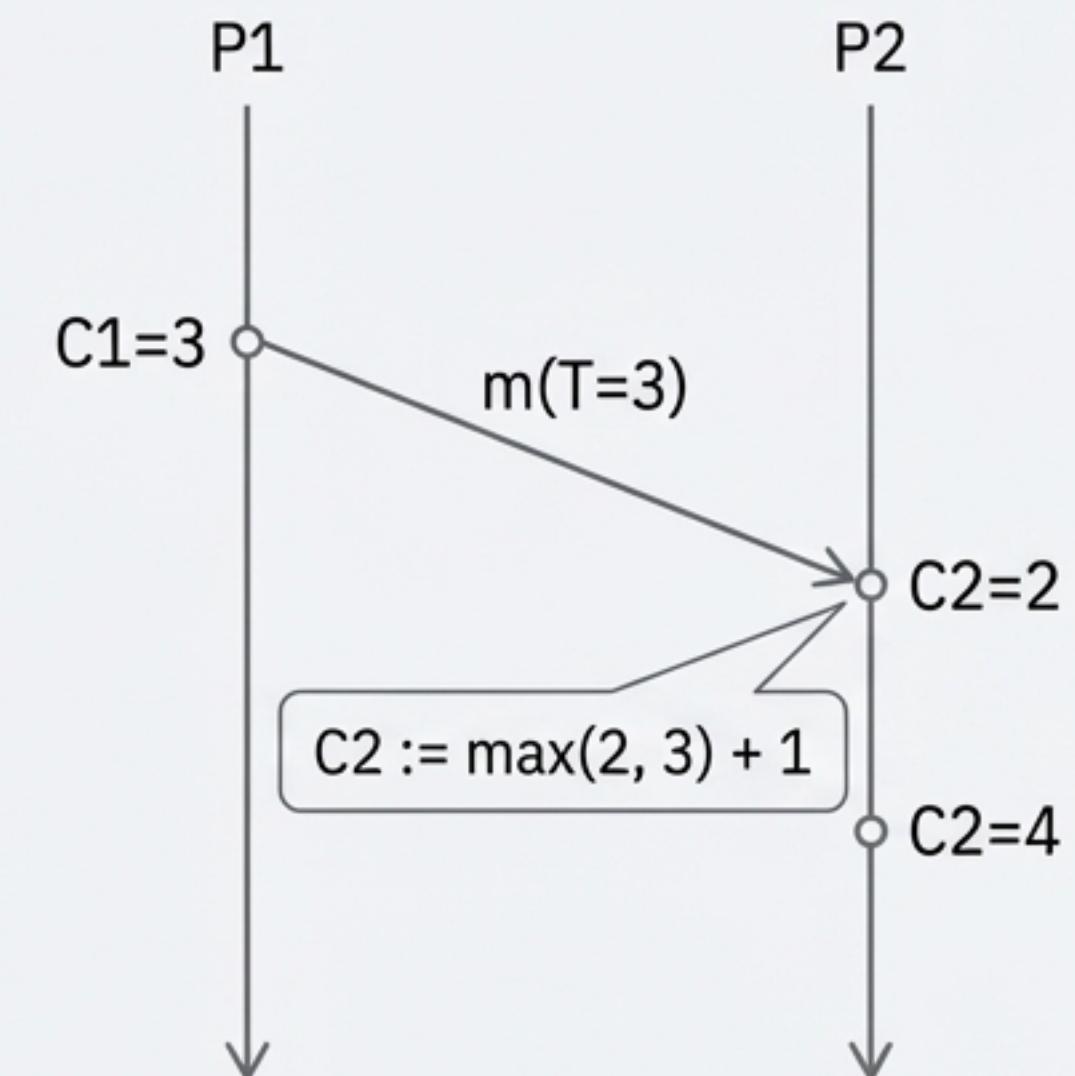
$$C := C + 1$$

**Zasada 2 (Komunikacja):**

(a) Gdy proces  $P_i$  wysyła wiadomość  $m$ , dołącza do niej jej sygnaturę czasową  $T_m = C_i$ .

(b) Gdy proces  $P_j$  odbiera wiadomość  $m$  z sygnaturą  $T_m$ , aktualizuje swój zegar  $C_j$  tak, by był większy od jego obecnej wartości i większy od  $T_m$ . Najczęściej:  $C_j := \max(C_j, T_m) + 1$ .

Wiadomości przenoszą „wiedzę” o historii przyczynowej nadawcy.



# Od porządku częściowego do całkowitego: Rozstrzyganie remisów

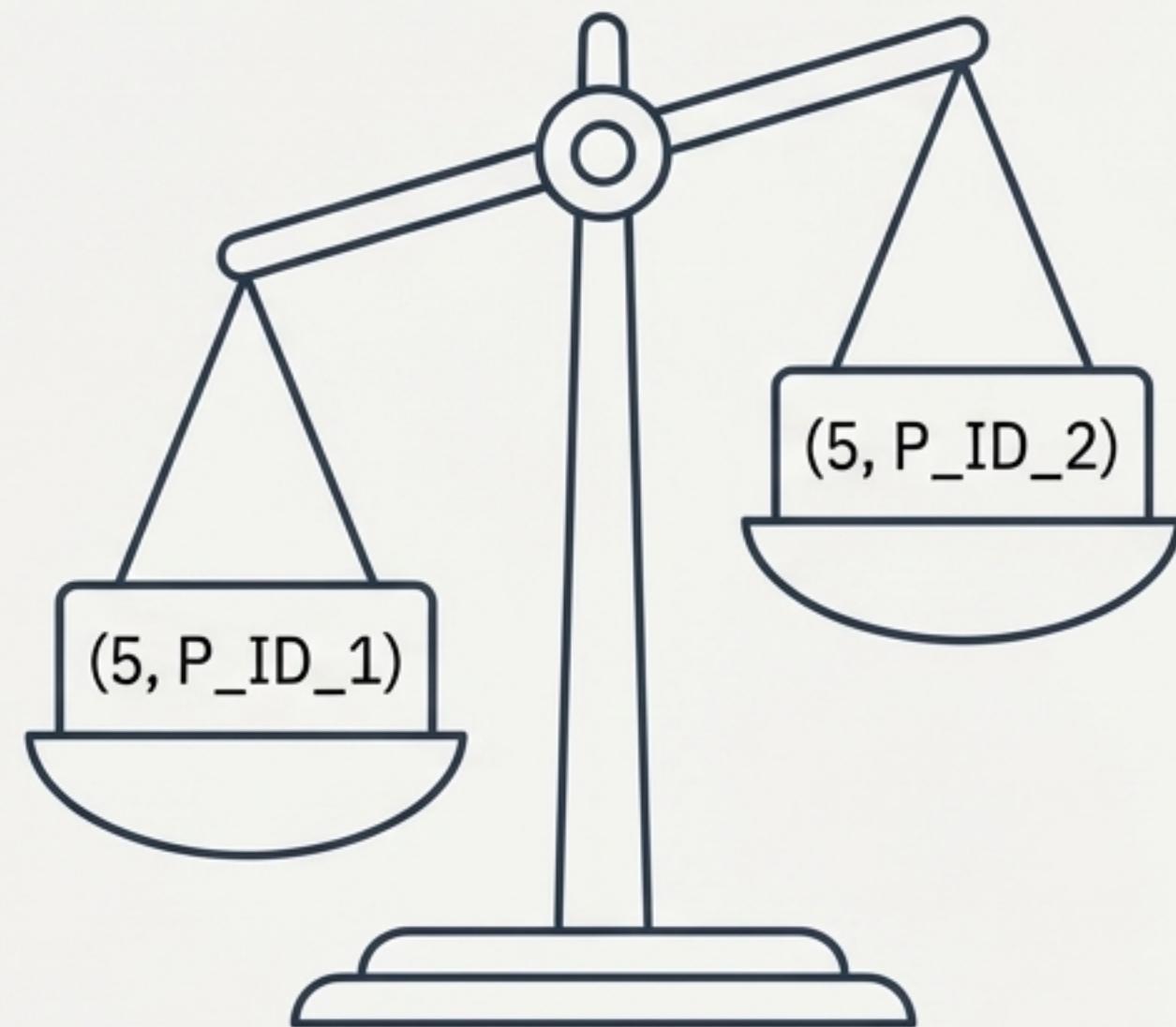
Sam porządek częściowy nie wystarcza do podejmowania jednoznacznych decyzji. Co jeśli dwa żądania dostępu do zasobu są współbieżne?

Potrzebujemy jednego, spójnego uszeregowania **wszystkich** zdarzeń w systemie – **porządku całkowitego** (*total ordering*), oznaczanego  $\Rightarrow$ .

**Rozwiązanie:** Rozszerzamy porządek częściowy. Do rozstrzygania remisów (...) używamy dowolnego, arbitralnego porządku na procesach (np. po ich unikalnych ID).

**Zasada:** Zdarzenie  $a$  w procesie  $P_i$  dzieje się przed ( $\Rightarrow$ ) zdarzeniem  $b$  w procesie  $P_j$ , jeśli:

1.  $C_i(a) < C_j(b)$ , LUB
2.  $C_i(a) = C_j(b)$  oraz  $P_i < P_j$  (reguła łamania remisu).



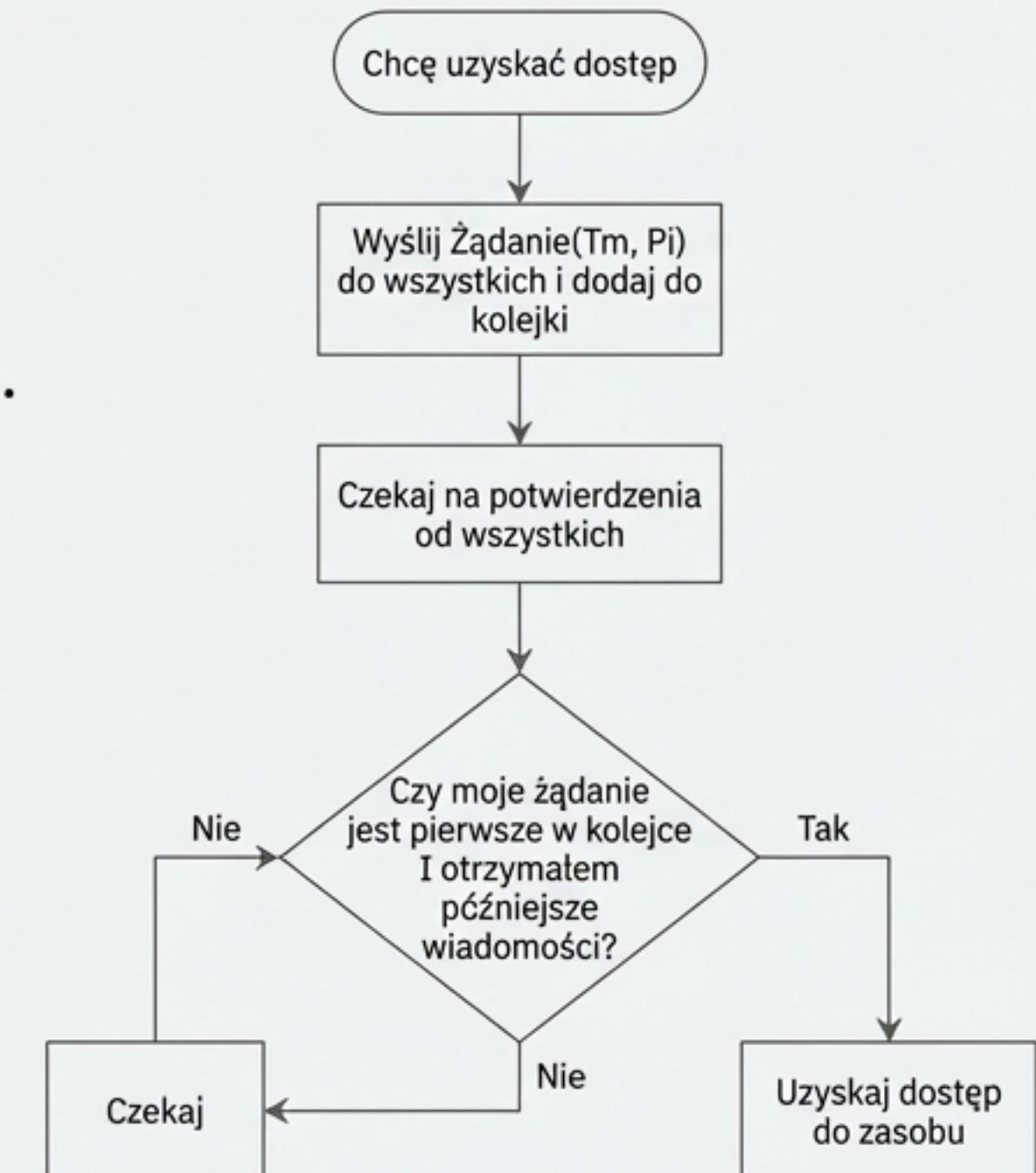
Wygrywa (jeśli  $ID_1 < ID_2$ )

# Zastosowanie: Zdecentralizowane wzajemne wykluczanie

Porządek całkowity  $\Rightarrow$  pozwala stworzyć w pełni zdecentralizowany algorytm dostępu do współdzielonego zasobu.

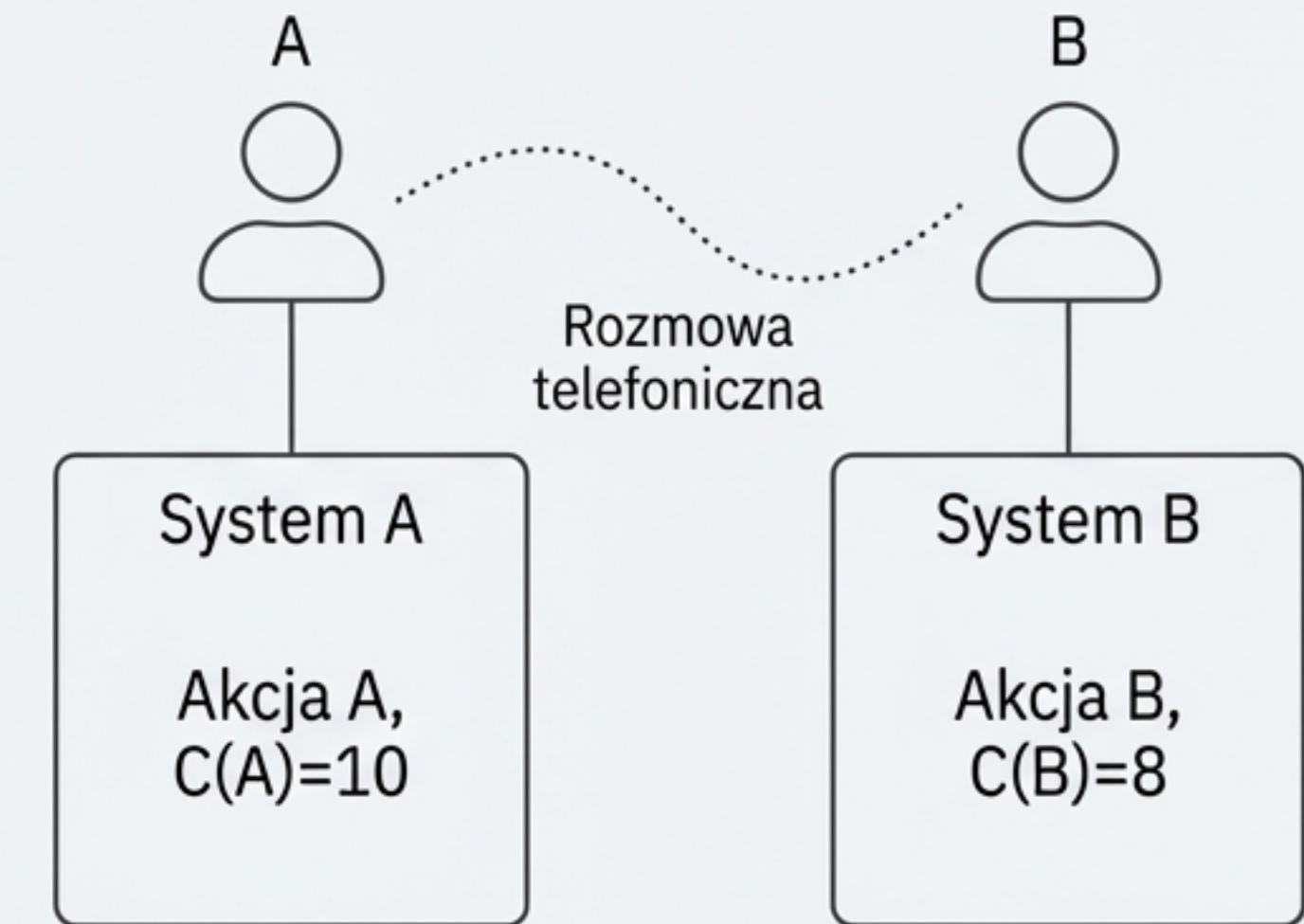
## Kroki algorytmu (dla procesu Pi):

1. Aby zażądać zasobu, wyślij wiadomość Żądanie( $T_m, P_i$ ) do wszystkich procesów i umieść ją w swojej lokalnej kolejce żądań.  
 $T_m$  to aktualny czas zegara  $C_i$ .
2. Po otrzymaniu żądania od  $P_j$ , umieść je w swojej kolejce i odeślij potwierdzenie.
3. Aby zwolnić zasób, usuń swoje żądanie z kolejki i wyślij wiadomość Zwolnienie do wszystkich.
4. Przyznaj sobie dostęp do zasobu, gdy:
  - (i) Twój żądanie jest pierwsze w Twojej kolejce (zgodnie z  $\Rightarrow$ ).
  - (ii) Otrzymałeś wiadomości od wszystkich innych procesów z sygnaturą czasową późniejszą niż Twój żądanie.



# Problem anomalii: Gdy logika kłoci się z rzeczywistością

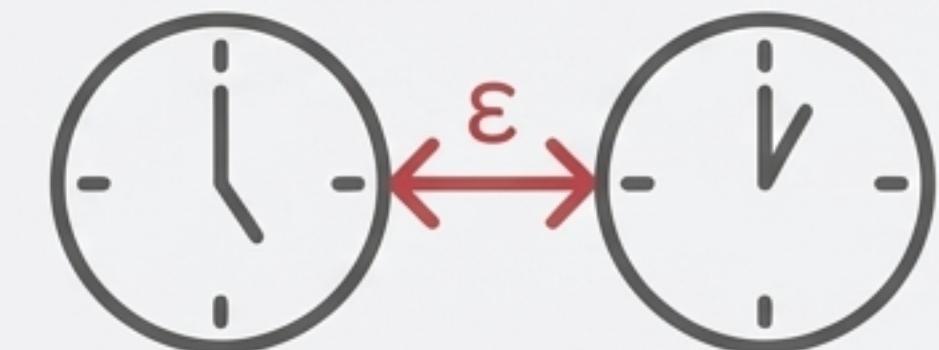
- Systemy logiczne mogą prowadzić do „anomalnego zachowania”.
- Przykład Lamporta: Osoba A wykonuje akcję A na komputerze A. Następnie dzwoni do osoby B, która wykonuje akcję B na komputerze B.
- Jest całkowicie możliwe, że akcja B otrzyma niższy znacznik czasowy i w porządku całkowitym ⇒ zostanie uszeregowana *przed* akcją A.
- Dzieje się tak, ponieważ system nie ma wiedzy o przyczynowości zachodzącej na zewnątrz (rozmowa telefoniczna).
- Narusza to ludzką percepcję kolejności zdarzeń. Wymaga to wprowadzenia zegarów fizycznych.



Porządek w systemie: B ⇒ A.  
Porządek w świecie rzeczywistym: A → B.

# Rozwiązanie: Synchronizacja z zegarami fizycznymi

- Aby uniknąć anomalii, system zegarów musi spełniać **Silny Warunek Zegara (Strong Clock Condition)**, który uwzględnia również zdarzenia zewnętrzne.
- Osiąga się to przez synchronizację zegarów z czasem fizycznym, przy zachowaniu dwóch warunków:
  - **PC1 (Ograniczony dryf):** Szybkość pracy wszystkich zegarów musi być zbliżona do rzeczywistej.  
 $|dC_i(t)/dt - 1| < \kappa$  dla małej stałej  $\kappa$ .
  - **PC2 (Ograniczone odchylenie):** Maksymalna różnica między wskazaniami dowolnych dwóch zegarów jest ograniczona.  
 $|C_i(t) - C_j(t)| < \varepsilon$  dla małej stałej  $\varepsilon$ .
- Lamport przedstawia algorytm, który pozwala utrzymać te warunki poprzez okresową synchronizację.



# Wpływ i dziedzictwo: Fundamenty współczesnych systemów

- Praca „Time, Clocks, and the Ordering of Events in a Distributed System” (1978) stała się kamieniem węgielnym informatyki.
- Zdefiniowała fundamentalne pojęcia i problemy w systemach rozproszonych.
- Jej zasady napędzają dziś:
  - Rozproszone bazy danych (np. Google Spanner, Amazon DynamoDB)
  - Systemy chmurowe i wielkoskalowe przetwarzanie danych
  - Technologię blockchain i mechanizmy konsensusu w kryptowalutach
- Algorytmy konsensusu, takie jak Paxos (również autorstwa Lamporta) i Raft, są zbudowane na tych ideach.
- Koncepcja została rozwinięta w postaci Zegarów Wektorowych, które precyzyjniej śledzą przyczynowość.



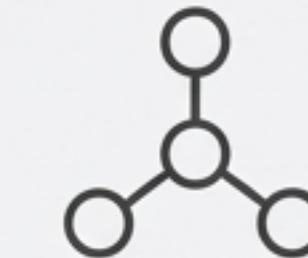
Bazy danych



Systemy chmurowe



Blockchain



Konsensus

# Pytanie do Ciebie

Lamport jasno rozdzielił zdarzenia wewnętrz systemu od tych na zewnątrz (jak rozmowa telefoniczna). Dziś ta granica coraz bardziej się zaciera. Nasze systemy nieustannie wchodzą w interakcje z ludźmi, miliardami urządzeń IoT i setkami innych systemów.

Gdzie w takim świecie kończy się „system”? Jak idee Lamporta muszą ewoluować, gdy zewnętrzna i wewnętrzna przyczynowość splatają się w sposób, którego nikt w 1978 roku nie mógł sobie wyobrazić?

