Overview

- Po co analizować algorytmy rozproszone?
- Gdzie są używane?
- Zapis formalny algorytmów rozproszonych
- Podstawowe abstrakcje: procesy, kanały, warstwy
- Specyfikacja: własności bezpieczeństwo i żywotność
- Założenia nt. środowiska
- Założenia nt. czasu
- Opis kroków algorytmu rozproszonego

```
upon event <Event1, attr1, attr2, ... > do
// Something
trigger <Event2, attr1, attr2, ... >
```

Własności Safety i Liveness

Two fundamental classes of properties

- Safety – „something bad never happens”
- Liveness – „something good should eventually happen”

Theorem by Alpern and Schneider (Inf. Proc. Letters, 1985):

"Any specification can be expressed as the conjunction of a safety property and a liveness property."

Safety i Liveness: przykład

Problem 1: uzyskanie z egzaminu z AR oceny pozytywnej

- Safety: in case you take the exam, do not fail it
- Liveness: you eventually have to take the exam

Problem 2: posortowanie rosnąco ciągu liczb

- Safety (partial corectness): if the algorithm terminates, then the output sequence is the sorted input sequence
- Liveness (termination): eventually the algorithm will terminate, e.g., respond with some sequence

Safety vs. Liveness

- Zasada składniowa: eventually – liveness
- Zasada intuicyjna: jeśli system nie robiąc nic, spełnia ten warunek, prawdopodobnie jest to warunek bezpieczeństwa (safety)
- Formalna wskazówka:

  Safety
  - warunek naruszony w skończonym czasie
  - warunek spełniony w nieskończonym czasie

  Liveness
  - warunek spełniony w skończonym czasie (eventually)
  - warunek naruszony w nieskończonym czasie (kiedy algorytm się nie skończy)

Safety vs. Liveness

Problem skoordynowanego ataku:

- Generał A wybiera czas ataku t i wysyła posłańca m1 z tą informacją do generała B, co oznaczamy m1(t)
- Generał B czeka na m1, po otrzymaniu m1(t) wysyła posłańca m2 do generała A, czeka do chwili t i atakuje
- Generał A czeka na m2, następnie po otrzymaniu m2 czeka do chwili t i atakuje.

Które z poniższych własności są spełnione przez rozważany protokół. Czy są to własności spełniające warunek safety czy liveness?

Safety vs. Liveness

1. Jeśli jeden z generałów atakuje o czasie t, to drugi także atakuje w tym samym czasie

FAŁSZ: Jeśli m2 będzie zagubione to B zaatakuje a A nie

Safety: „one could instruct an observer to wait until one general attacks and at that point in time check whether the other general also attacks.If not, then there is no chance to satisfy the property again. This is characteristic of a safety property.

2. Jeśli jeden z generałów atakuje o czasie t, to drugi także atakuje w tym samym czasie

PRAWDA: Jeśli generał A atakuje o czasie t to dostał m2, czyli B dostał m1;

Safety

Safety vs. Liveness

3. Jeśli generał B nigdy nie atakuje, to generał A nigdy nie atakuje.

PRAWDA: Jeśli generał B nigdy nie atakuje to nie dostał m1, czyli nie odesłał m2, więc generał A nigdy nie atakuje;

Nieokreślone:

- system nic nie robiący gwarantuje własność – sugeruje to safety;
- „ any finite behavior can be “made good” again” – co sugeruje liveness; if general B attacks, then the property is only violated as long as general A never attacks. So if A eventually attacks, then the property is satisfied. This indicates a liveness property.

Safety vs. Liveness

4. Jeśli generał A nigdy nie atakuje, to generał B nigdy nie atakuje.

FAŁSZ: m2 jest zagubione, więc B atakuje, podczas gdy A nigdy nie atakuje;

nieokreślone

5. Nie zdarzy się sytuacja, w której generał B atakuje a generał A nie atakuje

FAŁSZ: B może atakować, a A nie;

nieokreślone

Safety vs. Liveness

6. Ostatecznie generał B atakuje

FAŁSZ: gdy m1 jest zagubione, to B nie zaatakuje;

liveness - there is no finite point in time where this property can be violated

7. Jeśli posłańcy m1 i m2 nie zostaną przechwyceni, to ostatecznie obaj generałowie zaatakują

PRAWDA;

liveness - does not state that both generals must attack at the same time. Any finite behavior can be made good again: If some general attacks then the other general should also attack. If no general has attacked yet, this can be made good again by letting both generals attack.

Safety vs. Liveness

8. Jeśli posłańcy m1 i m2 nie zostaną przechwyceni, to ostatecznie obaj generałowie zaatakują w czasie t

PRAWDA / FAŁSZ – w zależności od momentu czasu t

safety: wait for the time when one general attacks and then check if the other general attacks at the same instant. If not, then you have a finite point in time where the property is violated.

Detektory błędów

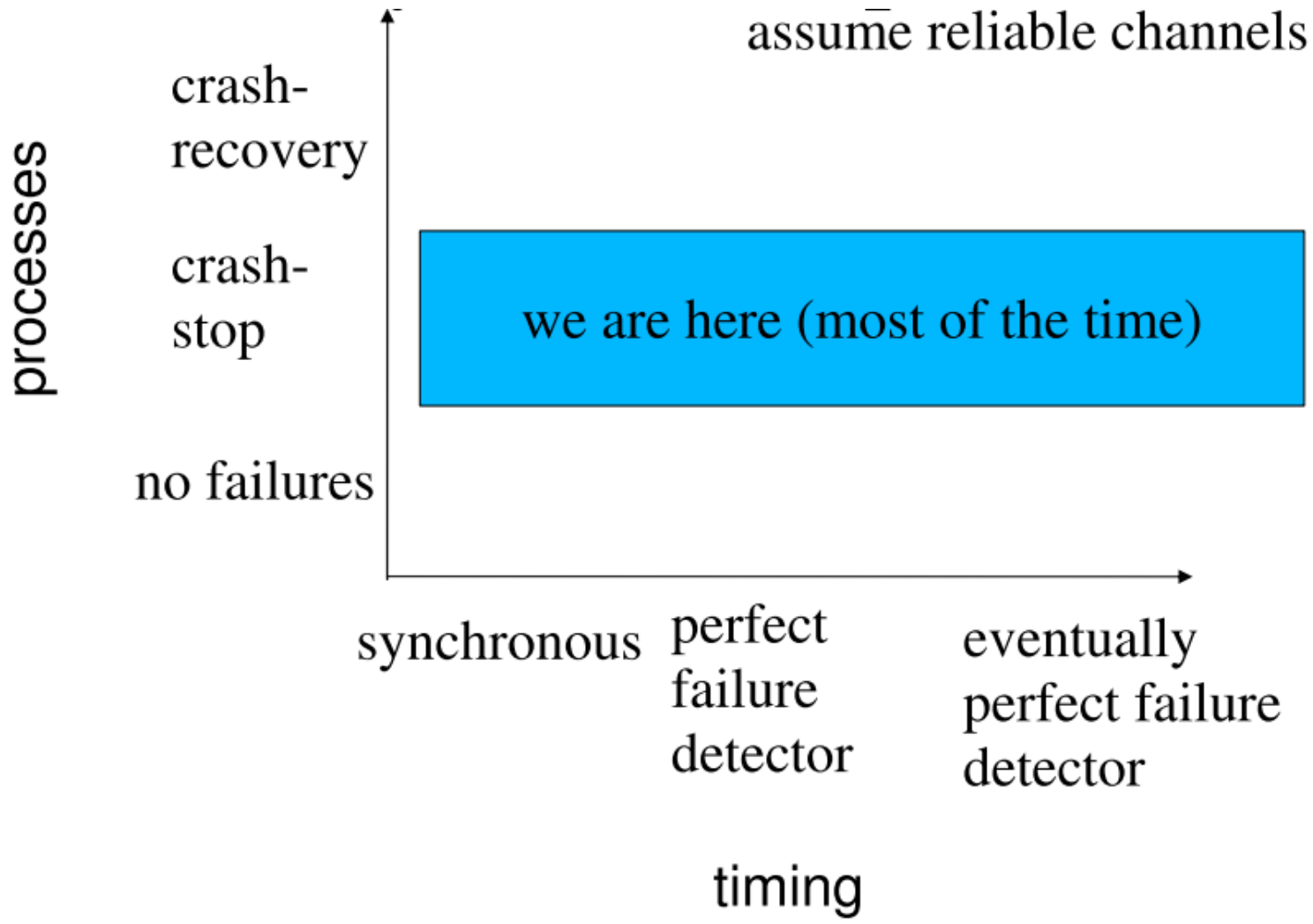
Perfect Failure Detector:

- PFD1: eventually every process that crashes is permanently detected by every correct process (strong completeness).
- PFD2: no process is detected by any process before it crashes (strong accuracy).

Eventually Perfect Failure Detector:

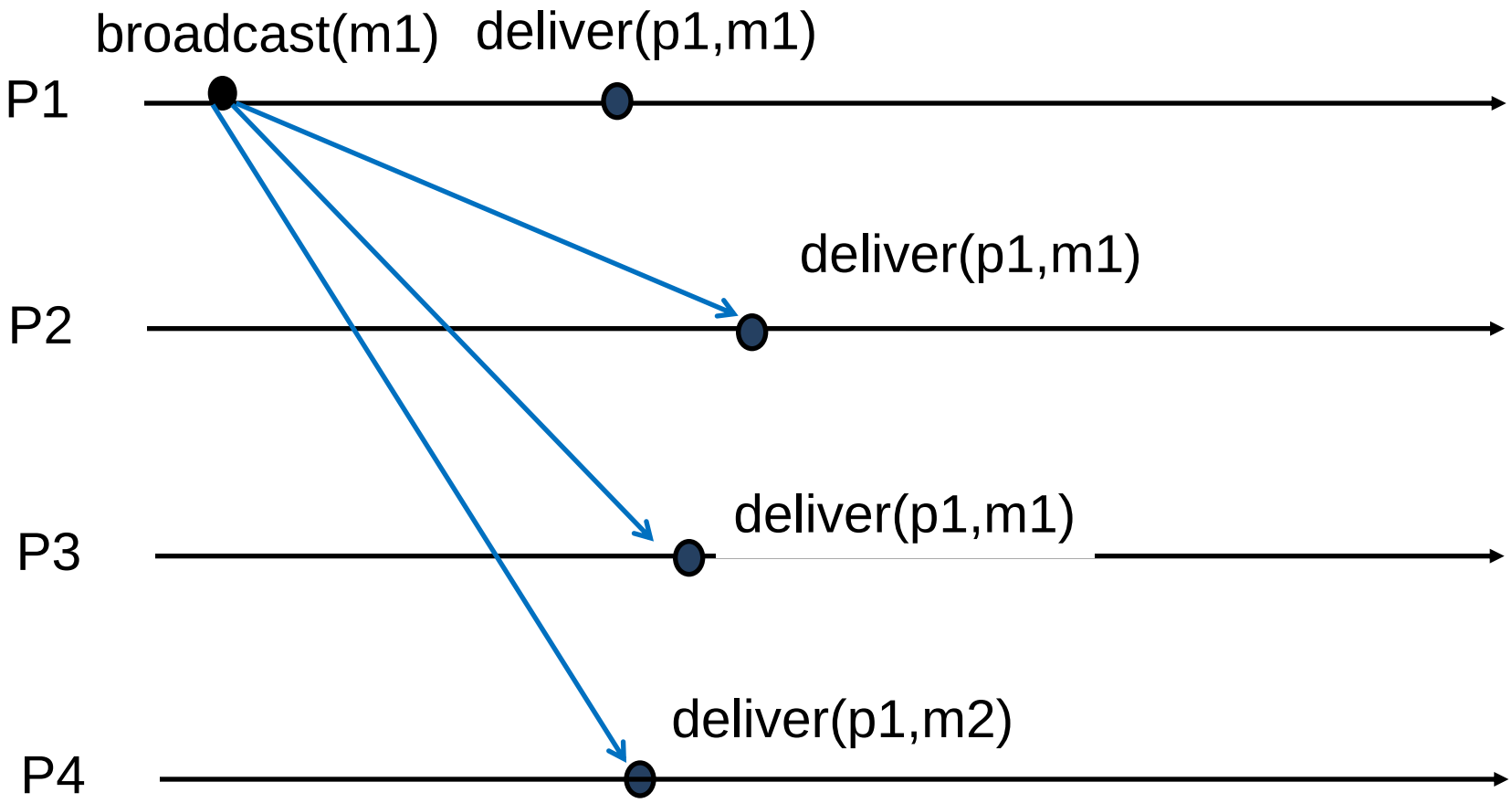
- PFD1
- Eventually, no correct process is ever suspected

Założenia

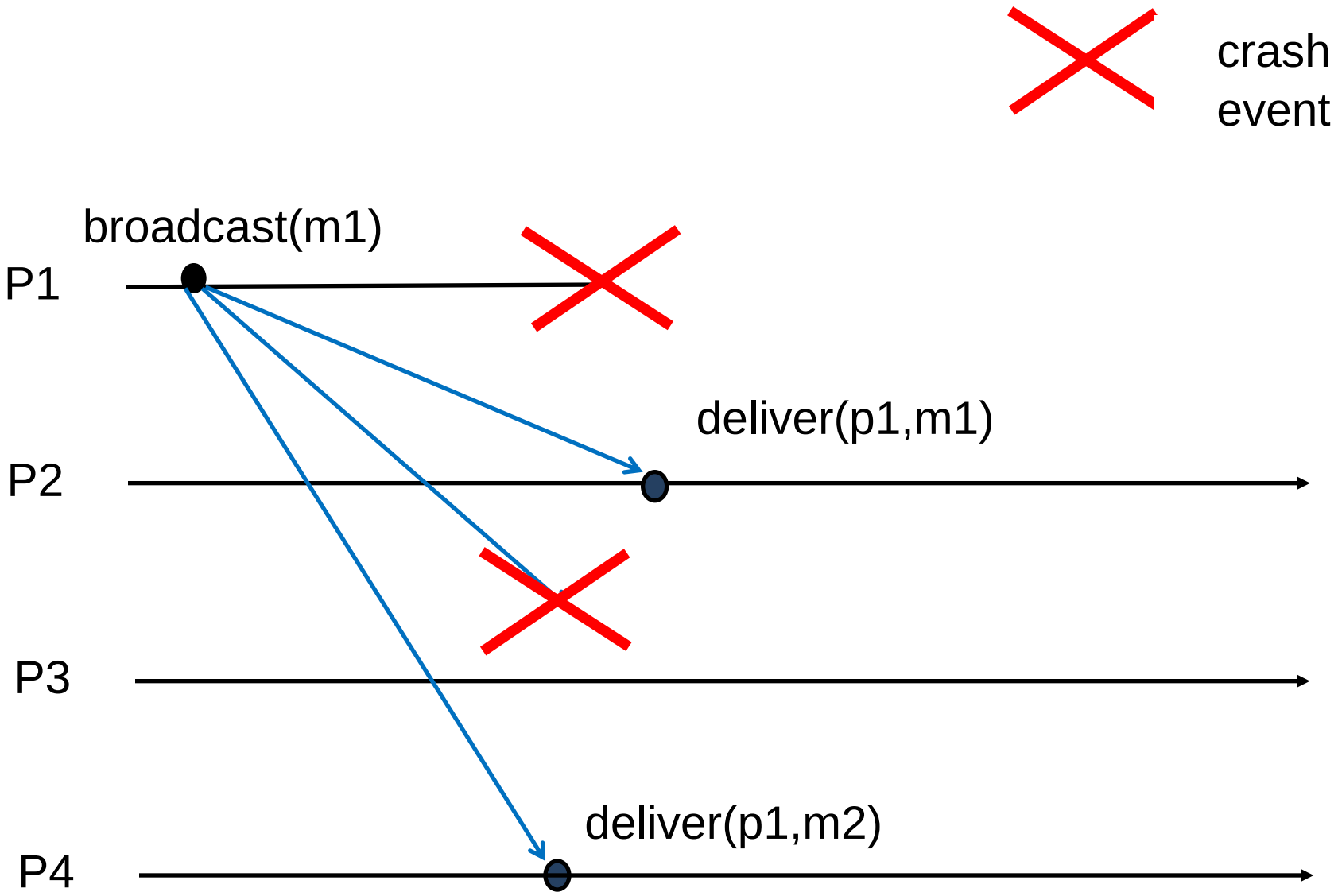

assume reliable channels
processes
crash-recovery
crash-stop
no failures
we are here (most of the time)
synchronous
perfect failure detector
eventually perfect failure detector
timing

Usługa rozgłaszania

Wysłanie wiadomości do grupy procesów



Unreliable broadcast



Reliable Broadcast - abstrakcje

- Best-effort broadcast – gwarantuje niezawodność tylko gdy nadawca jest poprawny
- Reliable broadcast – gwarantuje niezawodność niezależnie od poprawności nadawcy
- Uniform reliable broadcast – rozważa także zachowanie niepoprawnych węzłów
- FIFO reliable broadcast – RB z porządkiem FIFO
- Causal reliable broadcast – RB z porządkiem causal
- Probabilistic reliable broadcast – gwarantuje niezawodność z wysokim prawdopodobieństwem; skalowalny
- Total order (atomic) reliable broadcast - RB + całkowity porządek dostarczania wiadomości

Specyfikacje – BEB, RB, URB

Best-effort Broadcast (BEB)

- Ważność (Jeżeli procesy Pi i Pj są poprawne, to każda wiadomość m rozgłaszana przez Pi jest ostatecznie dostarczona do Pj)
- Brak powielenia (m jest dostarczona co najwyżej raz)
- Brak samogeneracji (m nie jest dostarczona, jeśli nie była
- wysłana)

Którą z własności spełnia ważność? Liveness

Specyfikacje – BEB, RB, URB

Reliable Broadcast (RB)

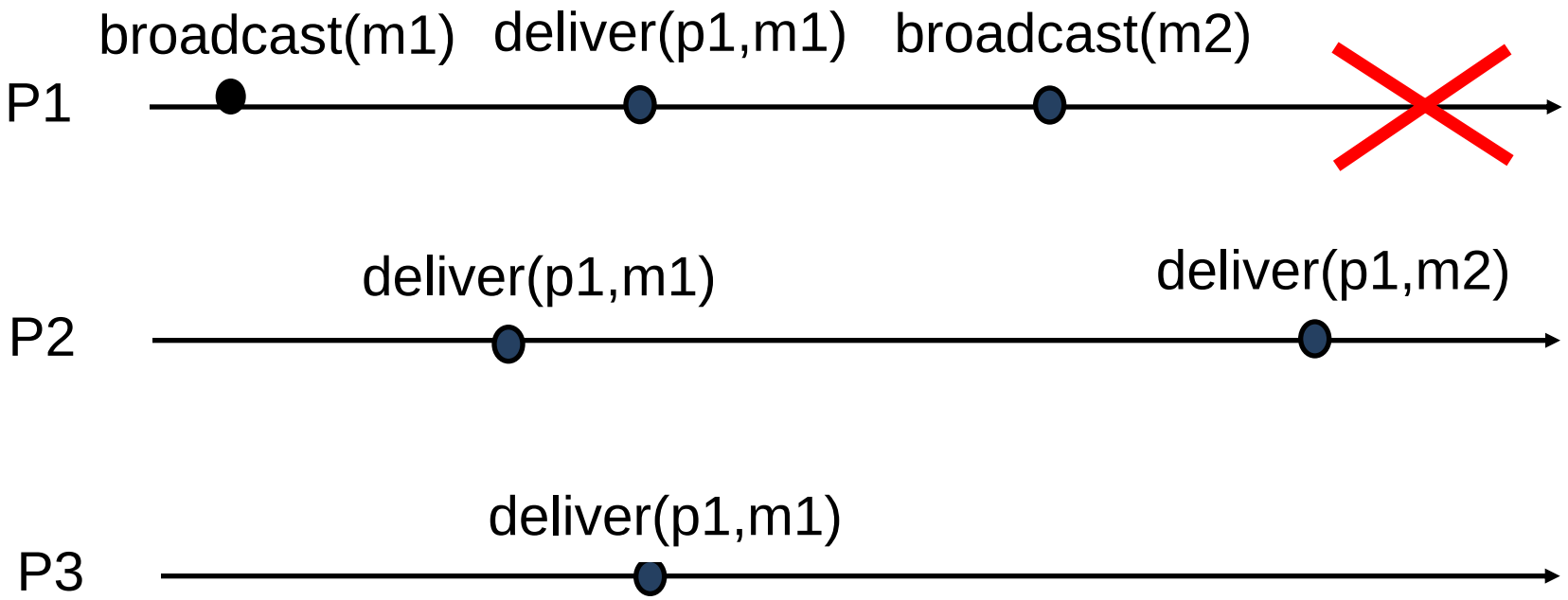
- Własności BEB
- Zgodność (jeśli wiadomość m została odebrana przez pewien poprawny proces, to ostatecznie wszystkie poprawne procesy odbiorą m)

Uniform Reliable Broadcast (URB)

- Własności BEB
- Jednolita zgodność (jeśli wiadomość m została odebrana przez pewien proces (poprawny lub nie), to ostatecznie wszystkie poprawne procesy odbiorą m)

Zadanie 1

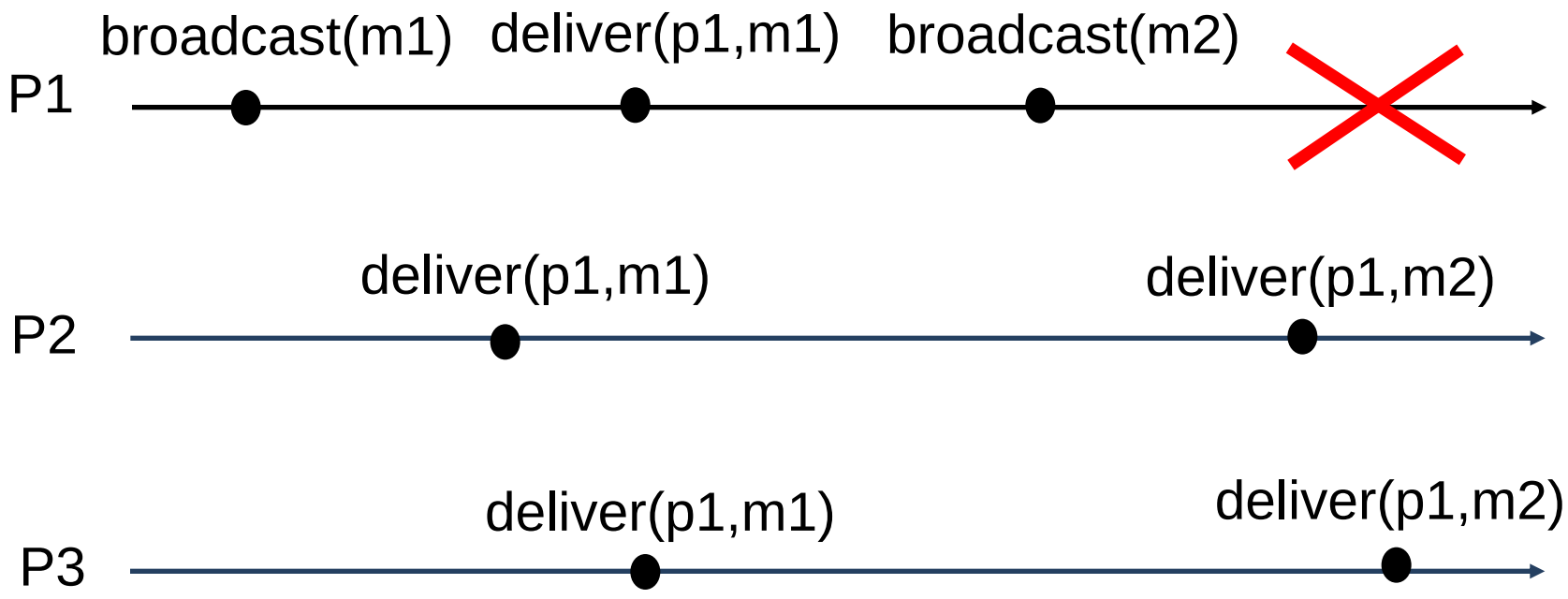
Jaki rodzaj rozgłaszania przedstawiony jest na rys?



P1 i P3 nigdy nie odbiorą m2 - BEB

Zadanie 2

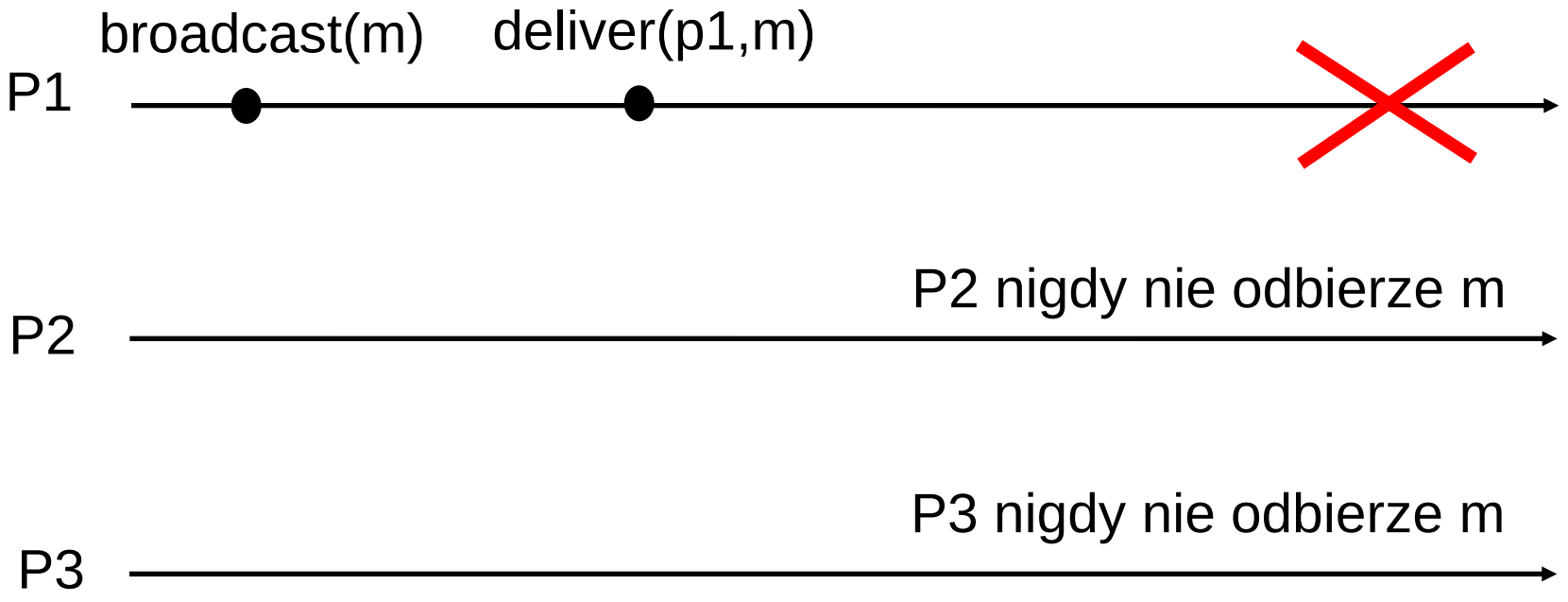
Jaki rodzaj rozgłaszania przedstawiony jest na rys?



Wszystkie poprawne procesy odbiorą m1 i m2 – BEB i RB i URB

Zadanie 3

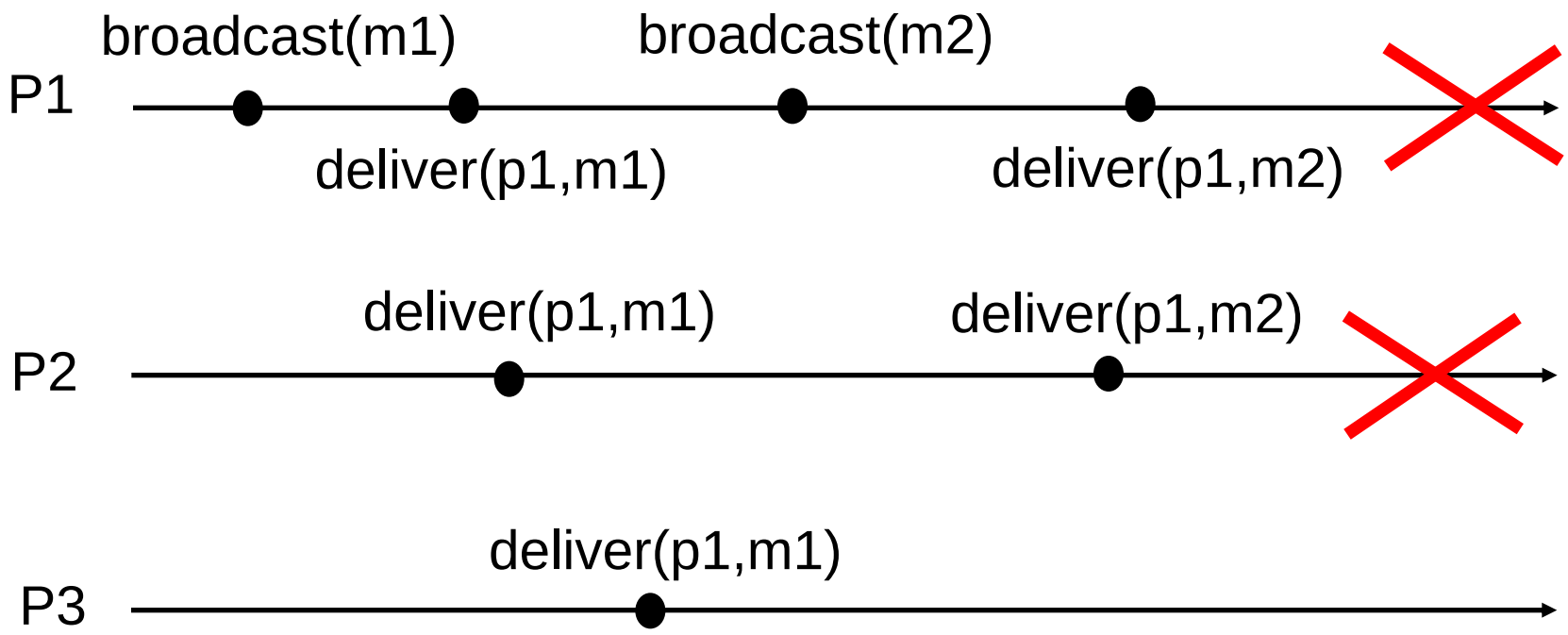
Jaki rodzaj rozgłaszania przedstawiony jest na rys?



Żaden poprawny proces nie odbierze m – BEB i RB

Zadanie 4

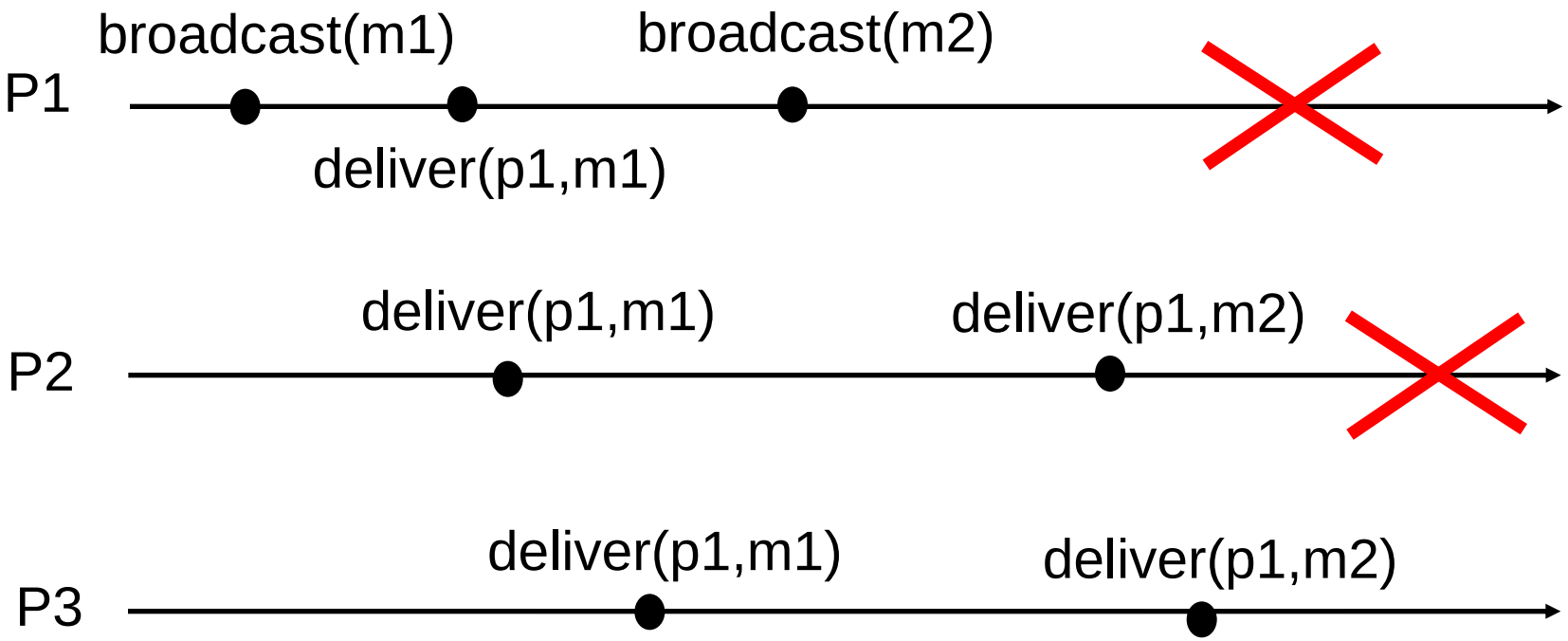
Jaki rodzaj rozgłaszania przedstawiony jest na rys?



Żaden poprawny proces nie odbierze m2, ale wszystkie niepoprawne odbiorą m2 (we don't care) – BEB i RB

Zadanie 5

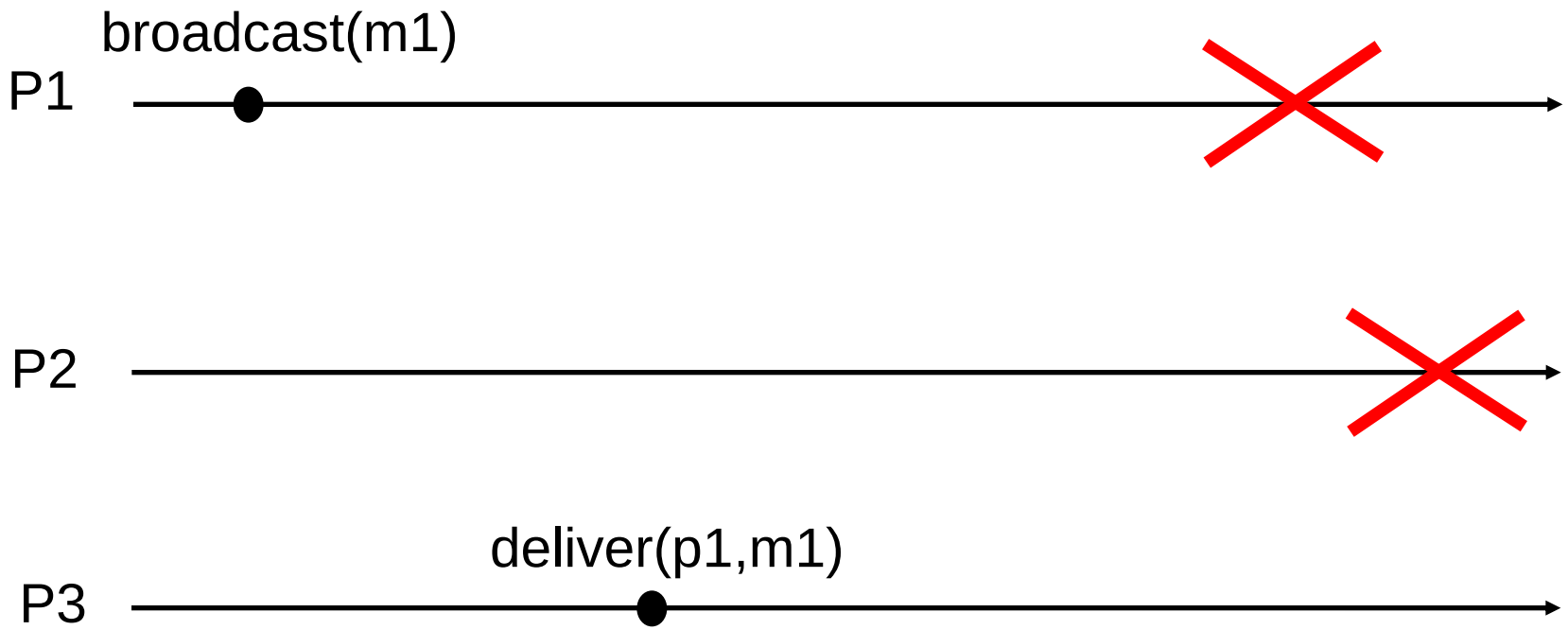
Jaki rodzaj rozgłaszania przedstawiony jest na rys?



BEB, RB i URB

Zadanie 6

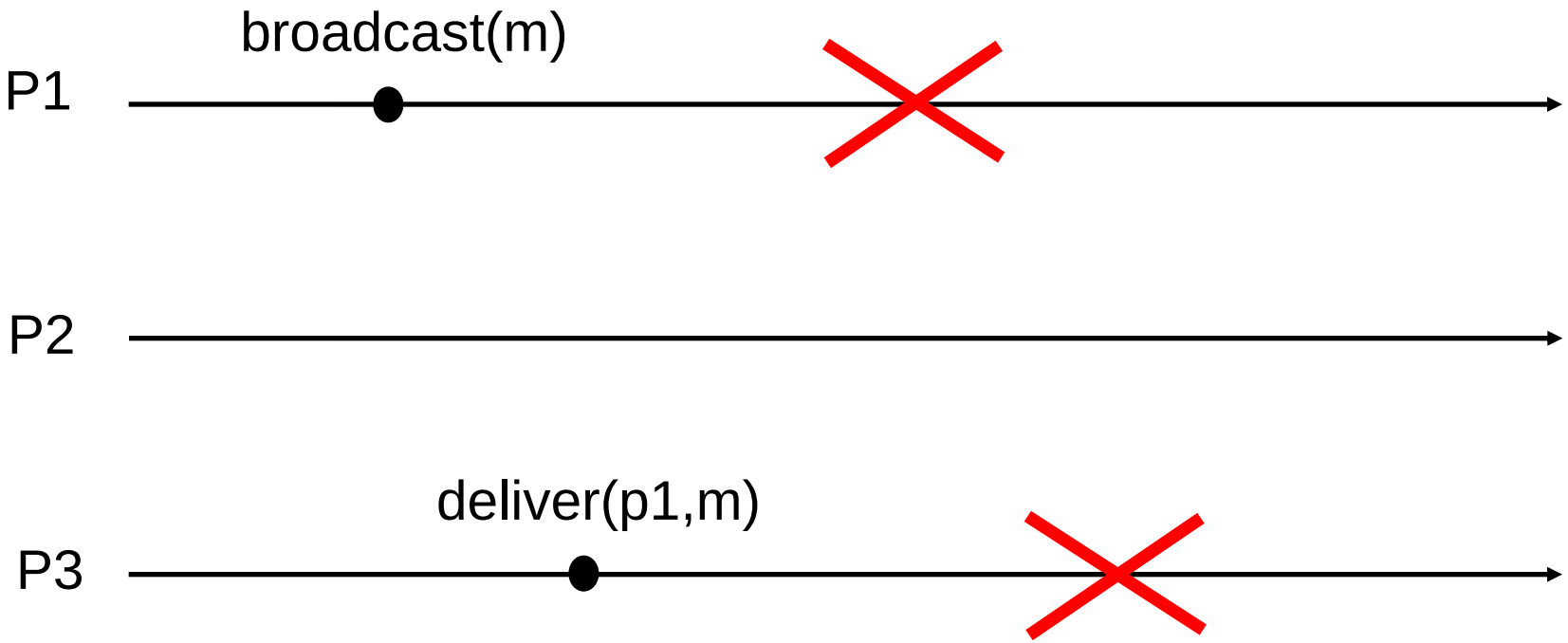
Czy URB jest spełniony?



Tak

Zadanie 7

Podaj przykład przetwarzania, które spełnia własności RB, ale nie spełnia własności URB



Jest spełniona własność zgodności, ale nie jednolitej zgodności

Zależności pomiędzy BEB, RB, URB

B = { BEB, RB, URB}

For any x,y in B: x => y iff

1) any execution of x satisfies the properties of y
2) There exists an execution of y which does not satisfy the properties of x

RB => BEB

1) any execution of RB is also execution of BEB
2) There exists an execution of BEB which is not RB

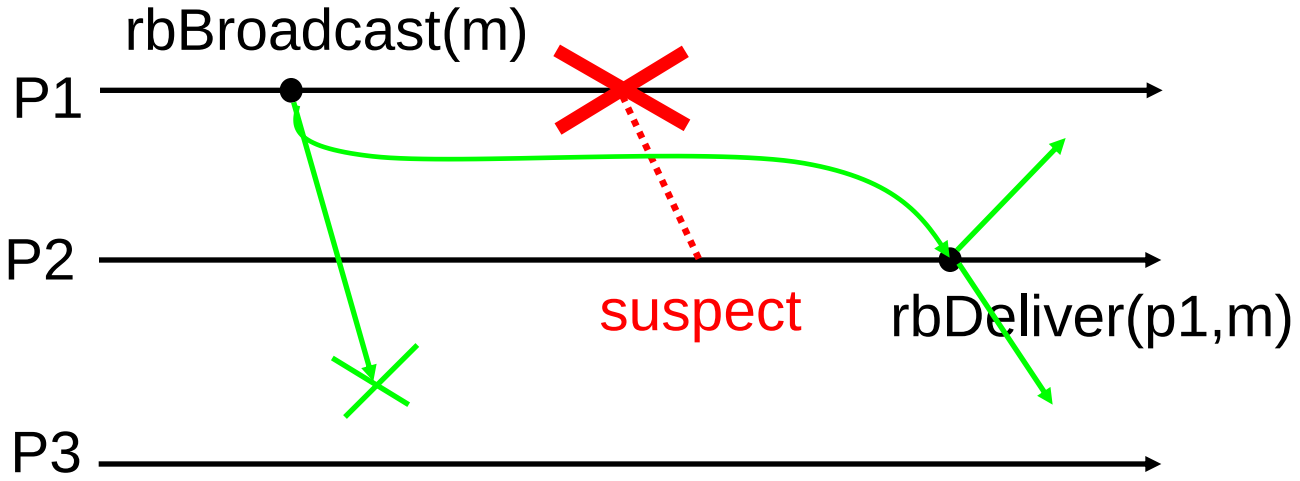
URB => RB

1) any execution of URB satisfies RB
2) There exists an execution of RB which is not URB

Zadanie 8

Algorytm 1 – Reliable Broadcast.

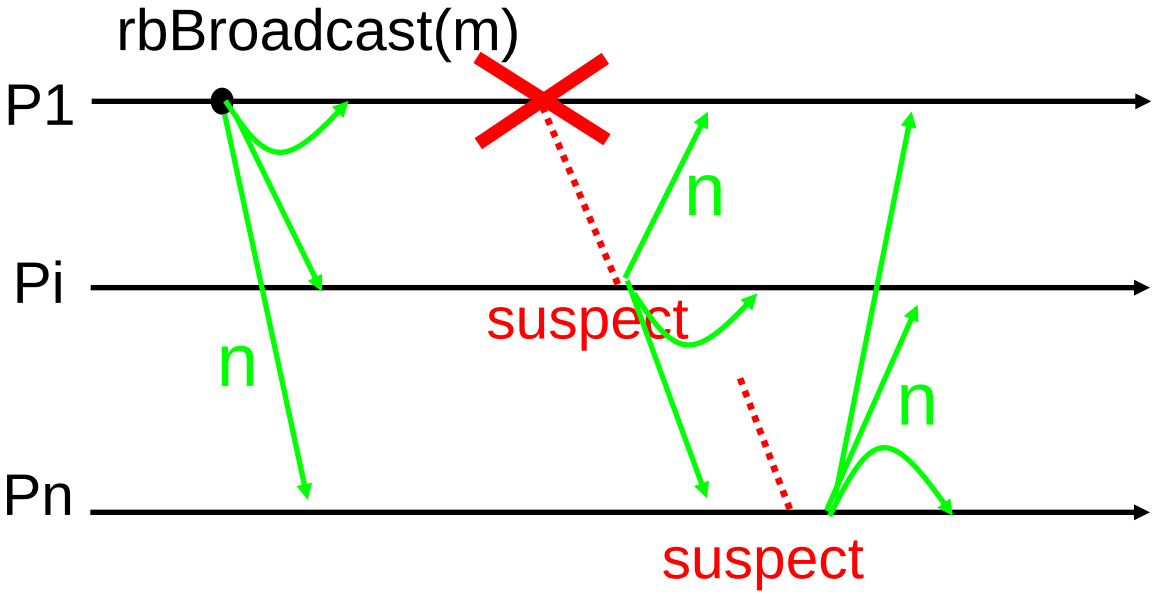
Czy można dokonać optymalizacji tego algorytmu poprzez ponowne rozgłoszenie wiadomości tylko w przypadku wykrycia procesu niedziałającego (pomijając każdorazowe rozgłaszanie wiadomości po jej dostarczeniu)?



„Premature optimization is the source of evil” (Knuth)

Zadanie 9

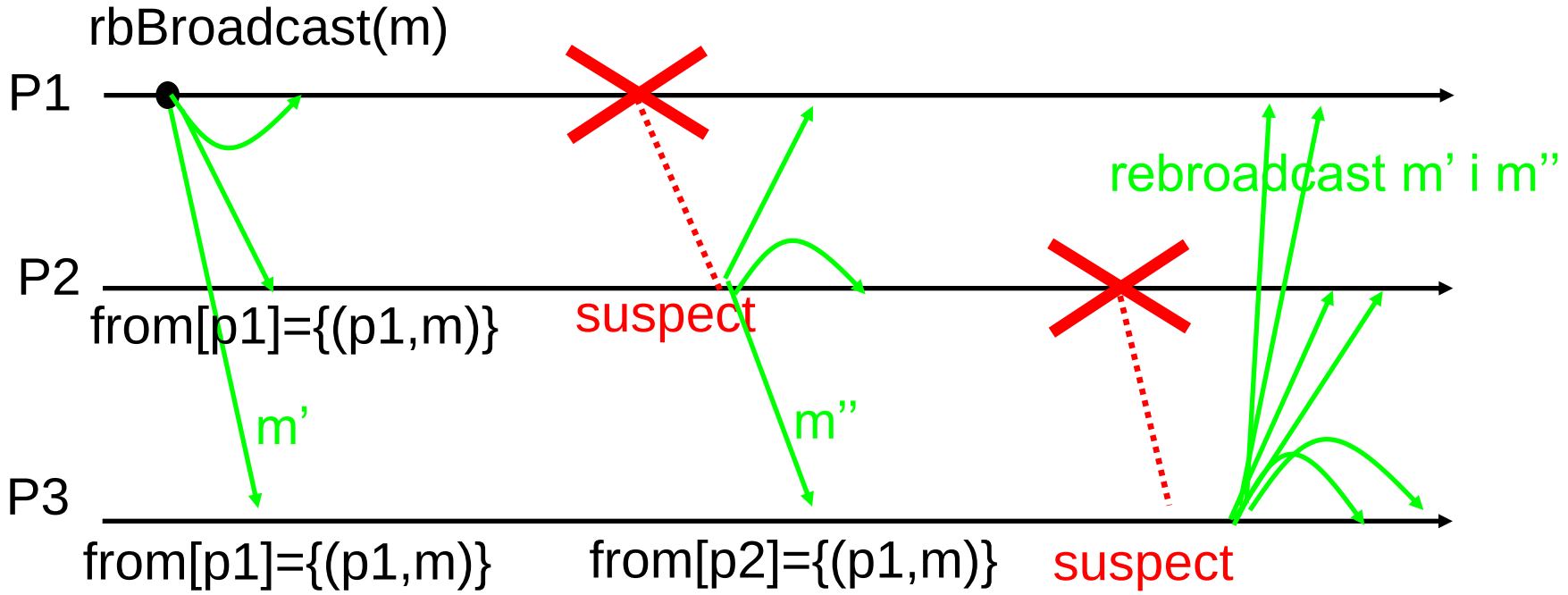
- **Zakładamy, że mamy n procesów (wywołanie BEB powoduje wysłanie n wiadomości). Podaj przykład przetwarzania, w którym po pojedynczym wywołaniu RB wysłana została maksymalna możliwa liczba wiadomości.**
- **Jaka jest pesymistyczna złożoność komunikacyjna RB?**



Każdy proces przesyła n*n =n^2 wiadomości

Zadanie 10

Dlaczego pesymistyczna złożoność komunikacyjna to nie n^3 ?

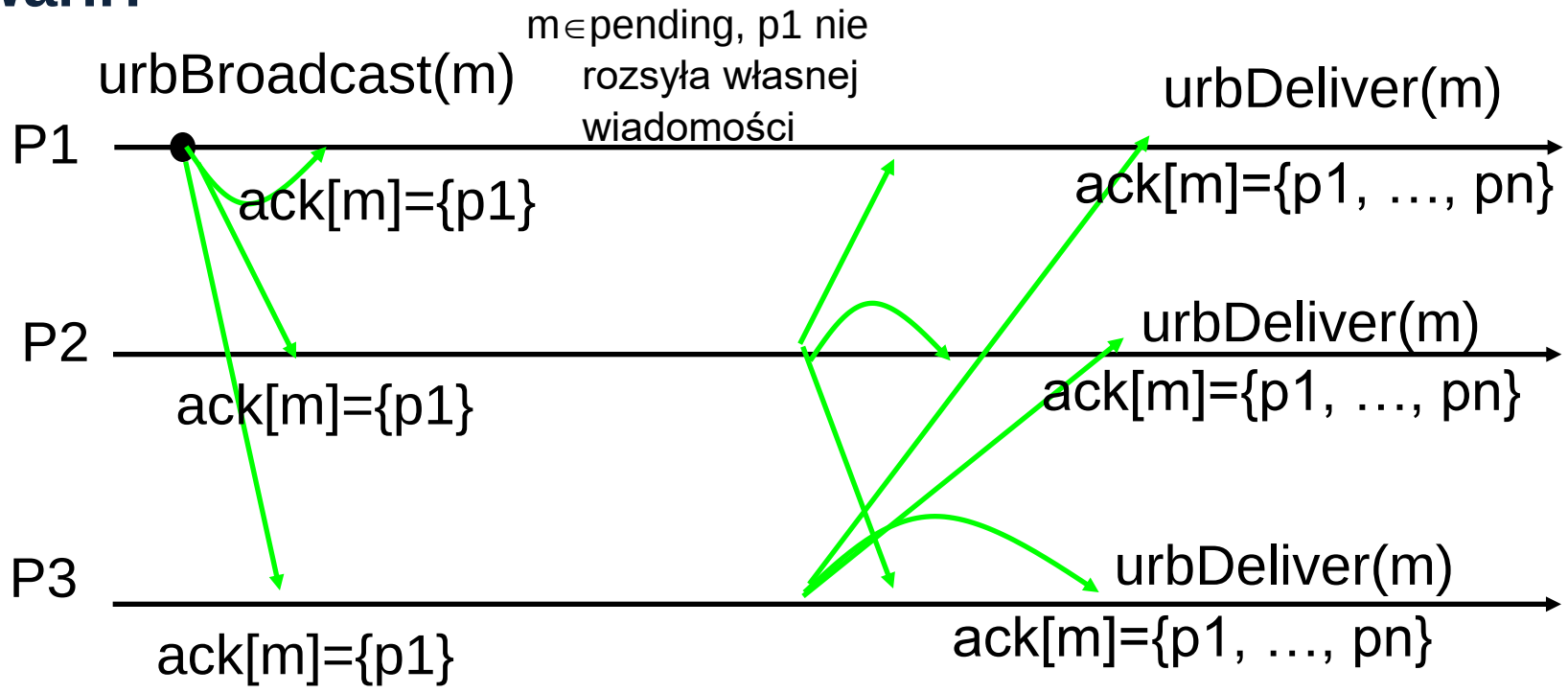


Zbiór delivered zapobiega dostarczeniu wiadomości do procesu po raz drugi. Jeśli p2 przestaje działać i p3 podejrzewa upadek, wtedy from[p2] w p3 nie zawiera żadnej wiadomości

Zadanie 11

Algorytm 2 – Uniform Reliable Broadcast

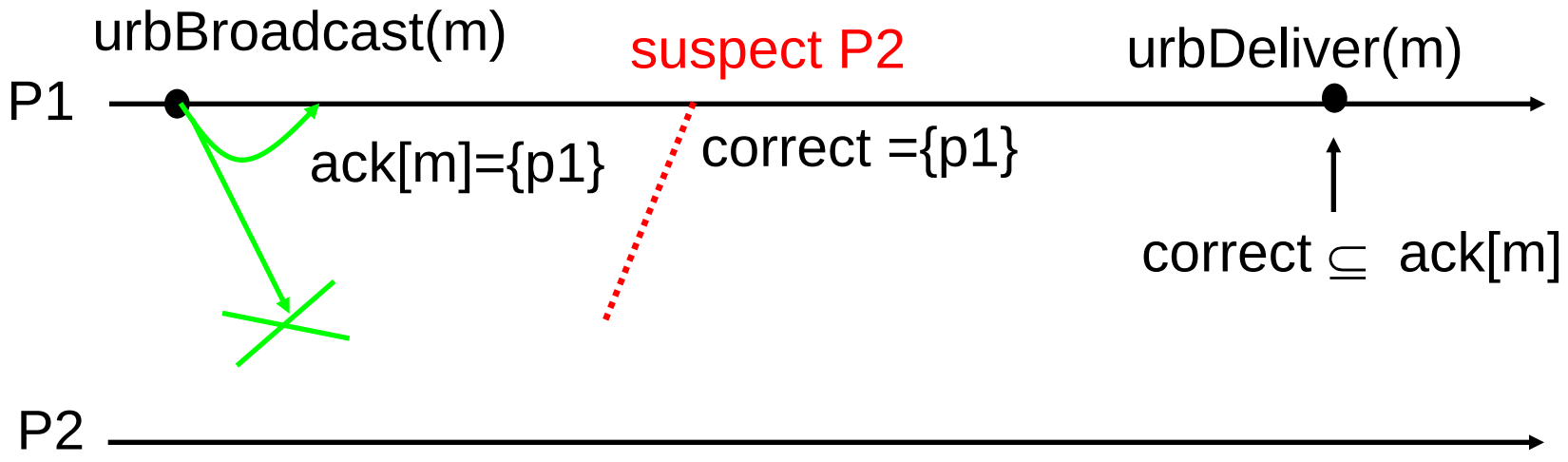
Jaka jest złożoność komunikacyjna dla przypadku optymistycznego, w którym żaden proces nie uległ awarii?



Pierwotny urbBroadcast – n wiadomości, koszt rozgłaszania $(n - 1)n$ wiadomości. Złożoność n^2

Zadanie 12

Czy algorytm URB będzie poprawny, jeśli użyjemy eventual perfect failure detector zamiast perfect failure detector?



Odp: Nie, patrz kontrprzykład

Zadanie 13

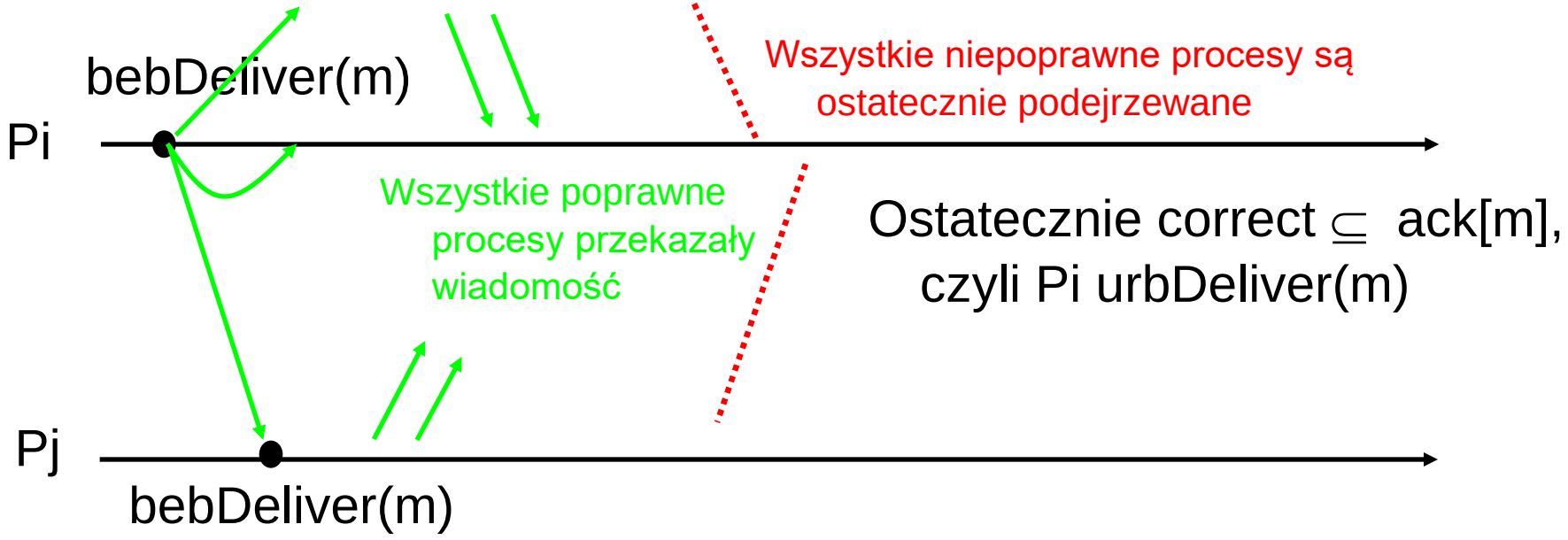
Udowodnij poprawność RB

- RB1, RB2, RB3 – wynikają z własności niezawodnych łączy
- RB4 (zgodność):
  - Załóżmy, że istnieje poprawny proces Pi, który odebrał wiadomość m (rbDeliver), rozgłoszoną przez pewien proces Pk (rbBroadcast)
  - Jeśli Pk jest poprawny, to z BEB1 wszystkie poprawne procesy odbiorą m (wywołując najpierw bebDeliver, a potem rbDeliver)
  - Jeśli Pk nie działa prawidłowo, to z własności kompletności detektora błędów, Pi odkryje upadek i rozgłosi m (bebBroadcast) do wszystkich procesów
  - Z zał., że Pi jest poprawny i z BEB1, wszystkie poprawne procesy bebDeliver i rbDeliber m

Zadanie 13

Udowodnij poprawność URB

- Jednolita zgodność: *dlaczego jeżeli istnieje proces, który odebrał wiadomość m, to wszystkie poprawne procesy także ostatecznie odbiorą m?*
- Uproszczenie: Jeśli poprawny proces Pi bebDeliver wiadomość m, to Pi ostatecznie urbDeliver m



Zadanie13

- Lemat:

  Jeśli poprawny proces Pi bebDeliver wiadomość m, to Pi ostatecznie urbDeliver m

- Dowód:
  - Dowolny poprawny proces który bebDeliver m, bebBroadcast m (linia 15)
  - Z własności ważności BEB1, ostatecznie Pi bebDeliver m od każdego poprawnego procesu
  - Z własności kompletności detektora błędów wynika, że każdy niepoprawny proces jest ostatecznie wykluczony ze zbioru *correct*
  - Ostatecznie warunek correct $\subseteq$ ack[m] będzie prawdziwy
  - Stąd Pi urbDeliver m

Zadanie13

- **URB1- Ważność:**
  - Jeżeli poprawny proces Pi urbBroadcast wiadomość m, to Pi ostatecznie bebBroadcast m
  - Z własności BEB1 – każdy poprawny proces Pj ostatecznie bebDeliver m.
  - Z Lematu – Pj ostatecznie urbDeliver m.
- **URB2, URB3 –** wynikają z BEB2 i BEB3
- **URB4 – jednolita zgodność:**
  - Załóżmy, że istnieje proces Pi, który urbDeliver wiadomość m
  - Z algorytmu i własności kompletności i dokładności detektora błędów, każdy poprawny proces bebDeliver wiadomość m, lub to dopiero zrobi.
  - Z lematu, każdy poprawny proces urbDeliver m

Causal Order

m1 $\rightarrow$ m2, gdy:

a) obie wiadomości zostały rozgłoszone przez ten sam proces, m1 przed m2

b) m1 została odebrana przez pewien proces Pi, a m2 została rozgłoszona przez Pi po odebraniu m1

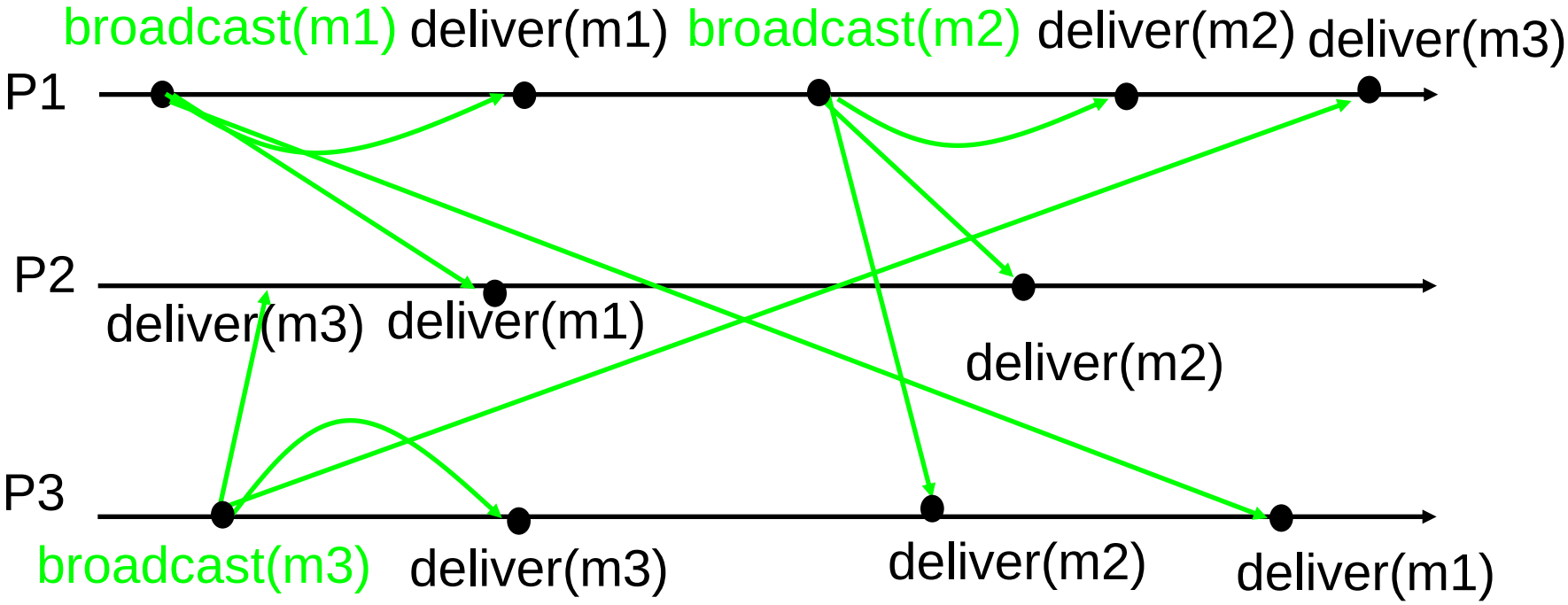
c) Istnieje wiadomość m3 taka, że dla m1 i m3, lub m3 i m2 zachodzi a) lub b)

Specyfikacje – FIFO, CO

- **Reliable FIFO Broadcast (RFB)**
  - RB1, RB2, RB3, RB4
  - FIFO Order: Jeżeli proces Pi rozgłosił wiadomość m1 przed m2, to każdy proces Pj nie odbierze m2 jeśli nie odebrał wcześniej m1
- **Reliable Casual Broadcast (RCB)**
  - RB1, RB2, RB3, RB4
  - Casual order: Proces Pi nie odbierze wiadomości m2, dopóki nie odebrał wszystkich wiadomości m1, takich, że m1 $\rightarrow$ m2

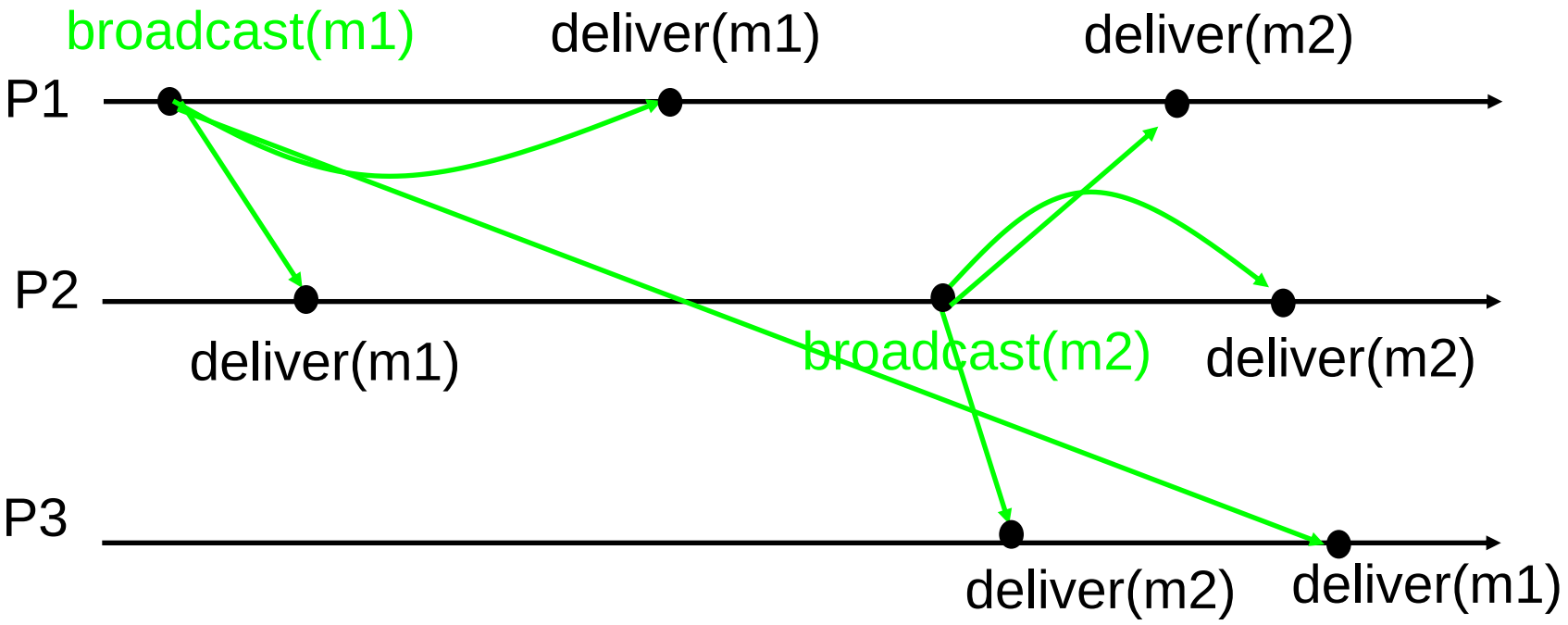
Zadanie 14

Zaproponuj przetwarzanie spełniające RB, ale nie spełniające FIFO



Zadanie 15

Podaj przykład przetwarzania, które spełnia warunki RFB i nie spełnia warunków RCB

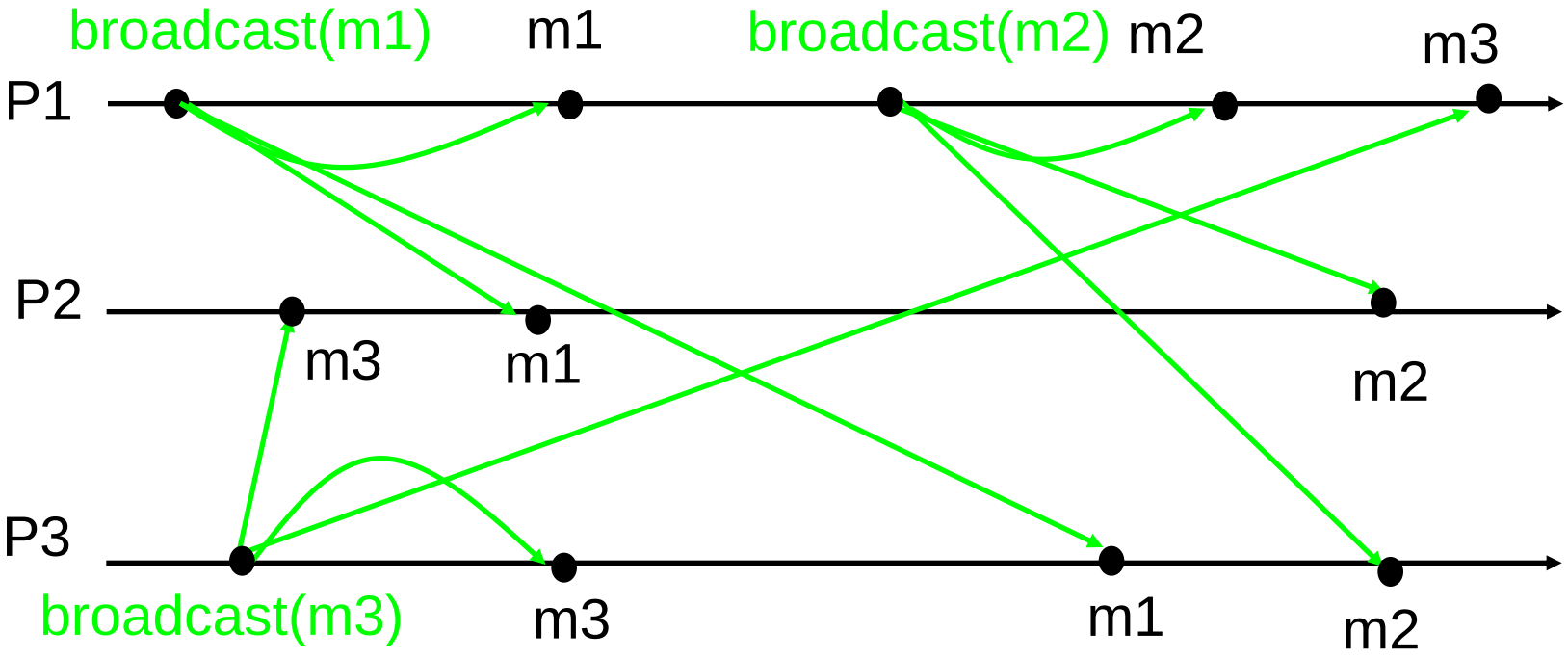


$m1 \rightarrow m2$

m2 odebrane przed m1

Zadanie 16

Zaproponuj przetwarzanie spełniające RB, RFB i RCB



Zadanie 17

Zaproponuj implementację RFB, do implementacji wykorzystaj RB

```
upon event <init> do
  buffer := { } // rbDelivered but not rfbDelivered
  next [ 1..n ] := (1, ... , 1) // next msg to rfbDeliver
  seq := 1 // local sequence number

upon event <rfbBroadcast, m> do
  trigger <rbBroadcast, self, seq, m>
  seq := seq + 1
```

Zadanie 17

Zaproponuj implementację RFB, do implementacji wykorzystaj RB

```
upon event <rbDeliver, src, sn, m> do
  buffer := buffer ∪ (src, sn, m)
  while (∃(src', sn', m') ∈ buffer :
          src' = src and sn' = next [src] ) do
           trigger <rfbDeliver, src, m'>
           next [src] := next [src] + 1
           buffer := buffer \ (src', sn', m')
   end // while
```

Zadanie 18

Udowodnij poprawność RFB.

Szkic rozwiązania:

- **RB1, RB2, RB3, RB4**
  - wynikają bezpośrednio z wykorzystywanego protokołu RB
- **FIFO order :**
  - Zakładamy, że m jest otrzymane przed m’
  - Pokaż, że istnieje proces Pi, który rozgłosił m przed m’
  - Podstawowy pomysł: elementy tablicy next[…] tylko wzrastają

Zadanie 19

Algorytm 3 - xyzBroadcast wykorzystujący mechanizm FIFOBroadcst.

A. **Cz ten algorytm spełnia własności RB?**

B. **Czy spełnia porządek FIFO**

C. **Czy spełniony jest porządek CO?**

Zadanie 19 – rozwiązanie pkt. A

- Algorytm wykorzystuje RFB jako mechanizm przesyłania danych
- Każda wiadomość wysłana za pomocą `xyzBroadcast` (l.4) jest także wysłana za pomocą `rfbBroadcast` (l.5)
- Każda wiadomość odebrana za pomocą `rfbDeliver` (l. 7) jest też `xyzDelivered` (l.10)
- Zatem własności ważności i zgodności RB są spełnione.
- Własność braku powielenia wynika z zastosowania zmiennej `delivered`.
- Własność braku samogeneracji wynika z faktu, że żadna nowa wiadomość nie jest wprowadzona w algorytmie

Zadanie 19 – rozwiązanie pkt. B

- Wiadomości są przekazywane za pomocą RFB
- Dlatego nie biorąc pod uwagę zmiennej `rcntDlvrs` wiadomości są dostarczane w porządku FIFO
- Wykorzystanie `rcntDlvrs` nie narusza FIFO:
  - Załóżmy, że wiadomości m1 i m2 są rozgłaszane przez ten sam proces
  - Obie wiadomości wysyłane są indywidualnie poprzez podwójne wywołanie `rfbBroadcast`.
  - Z własności RFB (porządek FIFO), wiadomość m1 będzie dostarczona za pomocą `rfbDeliver` przed m2.
  - To oznacza, że m1 będzie także dostarczona za pomocą xyzDeliver przed m2, co pokazuje zachowanie porządku FIFO dla xyzBroadcast.

Zadanie 19 – rozwiązanie pkt. C

- Ideą algorytmu jest przesyłanie danych partiami.
- Wiadomości wysyłane przez dany proces są otrzymywane w kolejności FIFO.
- Rozważmy wiadomości m1 i m2 otrzymane z zachowaniem FIFO. Wszystkie wiadomości odebrane pomiędzy odbiorem m1 i m2 są rozsyłane z wiadomością m2 (l. 5 – zmienna `rcntDlvrs` zawiera zbiór wiadomości odebranych od ostatniego rozgłaszania – l.12) – są to wiadomości od których przyczynowo zależy m2.
- Wszystkie wiadomości od których przyczynowo zależy m1 były rozesłane z wiadomością m1 i nie trzeba ich ponownie rozsyłać.
- Stąd algorytm zapewnia przyczynowe uporządkowanie wiadomości

Zadanie 20

Algorytm 4 – Reliable Causal Order Broadcast (RCB)

Zaproponuj rozszerzenie algorytmu o garbage collection.

- By zredukować rozmiar przesyłanych wiadomości, można usunąć wiadomość m ze zbioru past, jeśli m została dostarczona i wszystkie poprawne procesy potwierdziły dostarczenie m
- Potrzebne będzie wprowadzenie potwierdzenia przez wysłanie broadcast [ACK, m] do wszystkich procesów, po odebraniu m przez rbDelivered,
- Potrzebny będzie perfect failure detector

Zadanie 20 - rozwiązanie

- **upon event** < Init > **do**
  - delivered := past := { }
  - correct := { p1, ..., pn };
  - **forall** m: ack[m] := { }
- **upon event** < crash, pi > **do**
  - correct := correct \ { pi }
- **upon** <for some m in delivered:
  self not in ack[m]> **do**
  - ack[m] := ack[m] U { self };
  - **trigger** < rbBroadcast, [ACK,m]>;
- **upon event** <rbDeliver, pi, [ACK, m]> **do**
  - ack[m] := ack[m] U { pi };
  - **if** correct $\subseteq$ ack[m] **do**
    - past := past \ { [sm, m] };

Total Order Broadcast (TO)

- RB1, RB2, RB3, RB4
- Globalne uporządkowanie wiadomości (total order):

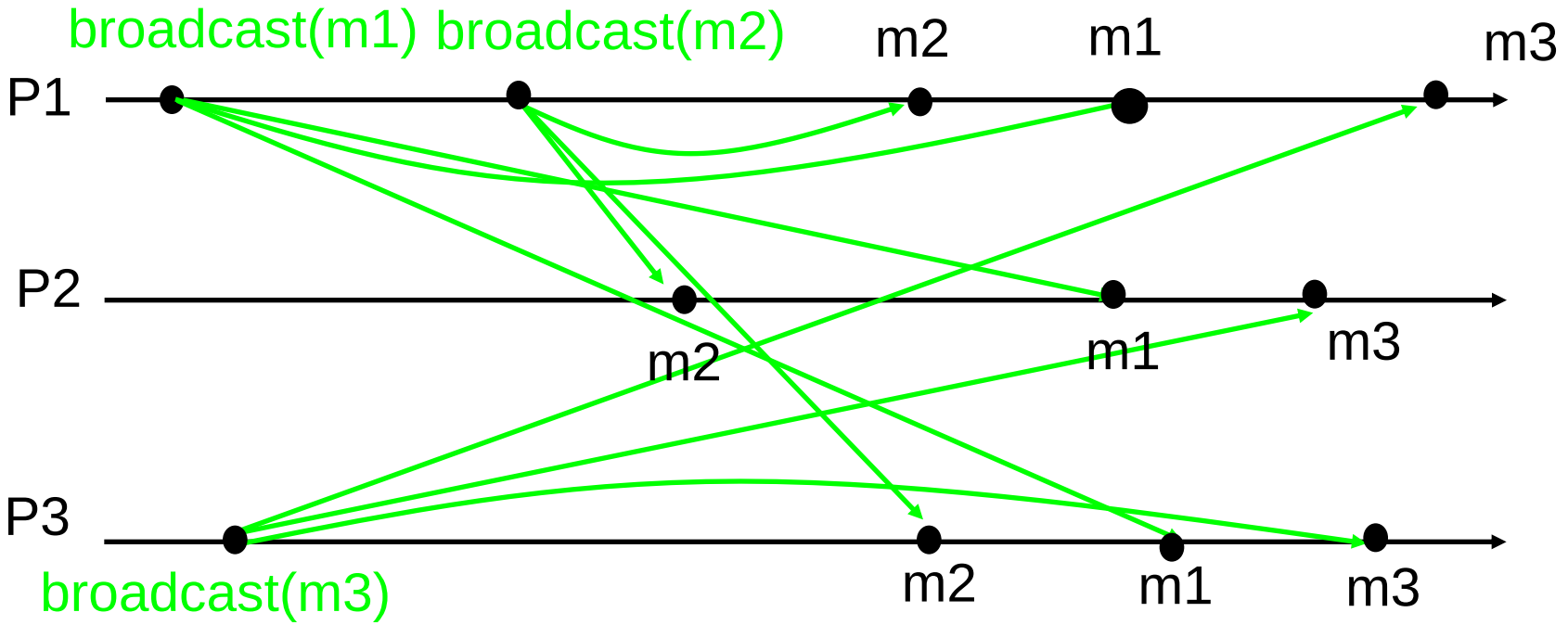
  Wszystkie procesy odbierają wiadomości w tym samym porządku, tj. jeśli Pi i Pj są poprawnymi procesami, które odbierają wiadomość m i jeśli Pi odbiera m' przed m, to także Pj odbiera m' przed m.

- Jednolite globalne uporządkowanie wiadomości (uniform total order)

  jeśli Pi i Pj odbierają wiadomość m i jeśli Pi odbiera m' przed m, to także Pj odbiera m' przed m.

Zadanie 21

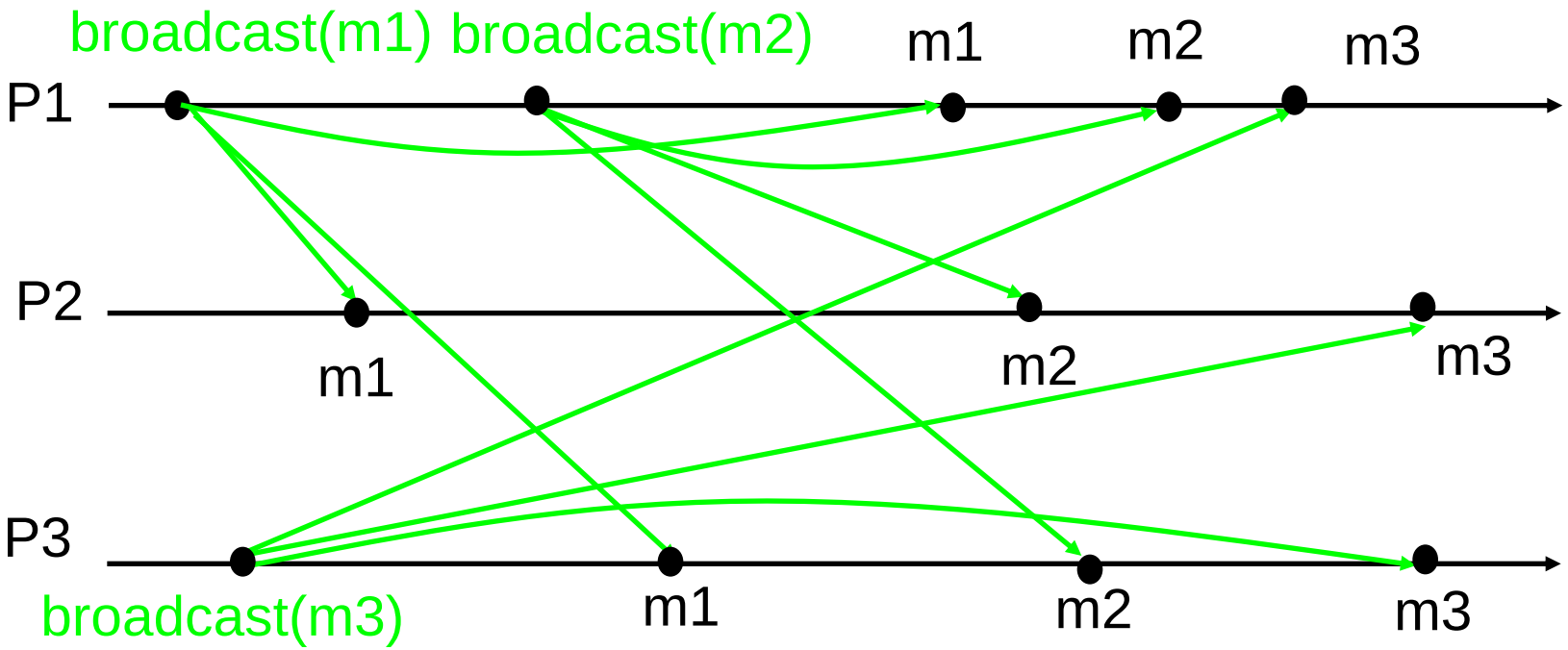
Czy na poniższym rys. jest zapewniony TO?



TO – tak ; FIFO – nie

Zadanie 21

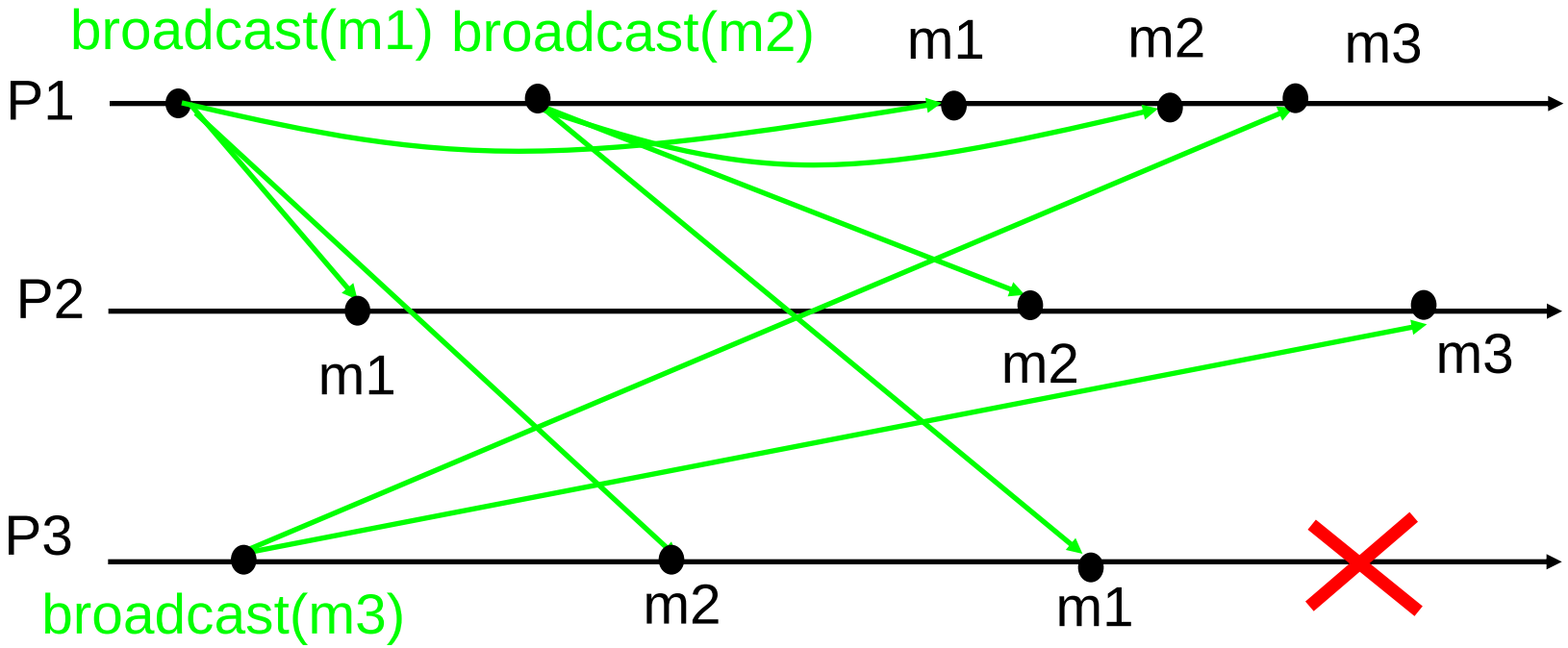
Czy na poniższym rys. jest zapewniony TO?



TO – tak; FIFO – tak; casual – tak

Zadanie 21

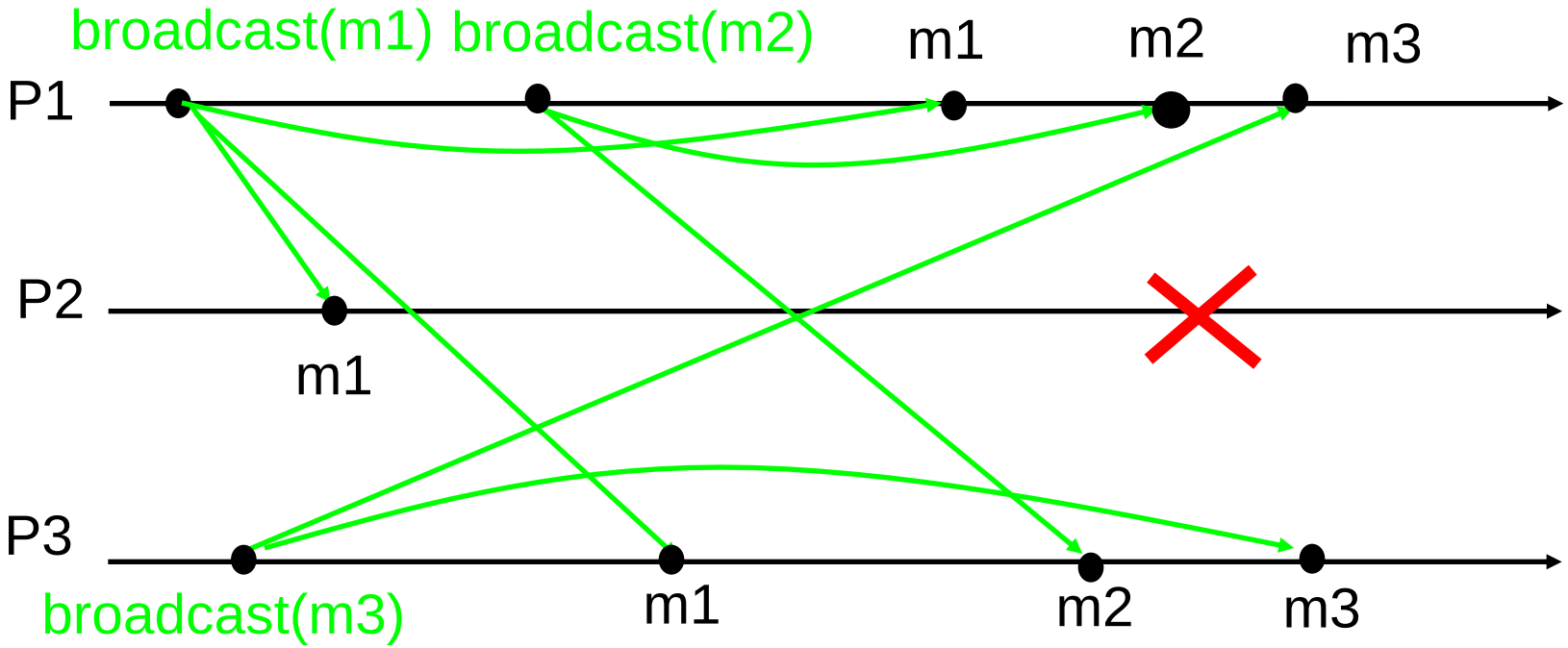
Czy na poniższym rys. jest zapewniony uniform TO?



Uniform TO – nie; URB – tak; uniform FIFO – nie

Zadanie 22

Czy na poniższym rys. jest zapewniony uniform TO?



Uniform TO – tak; casual – tak

Zadanie 23

Udowodnij, że wykonanie TO, które zapewnia porządek FIFO, zapewnia także uporządkowanie przyczynowe

- Założenia:
  1. Dostarczane wiadomości są globalnie uporządkowane (TO)
  2. Wiadomości są dostarczane w porządku FIFO
  3. Istnieją wiadomości m1 i m2 dla któych: m1 $\rightarrow$ m2 (m1 casually precedes m2)
- Teza:

  Dla rozważanego protokołu - dla każdego poprawnego procesu Q dostarczenie wiadomości m1 i m2 spełnia CO

Zadanie 23 – dowód

A. Istnieje proces P, dla którego :

   1. Dostarczenie wiadomości m1 do P odbywa się przed rozgłoszeniem wiadomości m2 przez ten proces, lub
   2. Rozgłoszenie wiadomości m1 odbywa się przed rozgłoszeniem wiadomości m2 przez ten proces

   Dowód: wynika to z założenia 3.

B. Załóżmy, że spełniony jest warunek A1

   Teza: ∀ poprawnego procesu Q: dostarczenie wiadomości m1 odbywa się przed dostarczeniem wiadomości m2.

Dowód: Jeżeli w P dostarczenie m1 odbywa się przed rozgłoszeniem m2, to dla P odebranie m1 także odbywa się przed odebraniem m2.

Z własności TO – ta zależność zachodzi dla wszystkich procesów.

Zadanie 23 – dowód

C. Zał: Spełniony jest warunek A2

Teza: ∀ poprawnego procesu Q: dostarczenie wiadomości m1 odbywa się przed dostarczeniem wiadomości m2.

Dowód: Jeżeli w P rozgłoszenie m1 odbywa się przed rozgłoszeniem m2, to z własności FIFO: dla wszystkich procesów dostarczenie m1 musi odbywać się przed dostarczeniem m2

B i C kończą dowód.

Zadanie 24

Czy można skonstruować BEB spełniający własność przyczynowego uporządkowania wiadomości, nie będący jednocześnie RCB (tj. dla którego nie jest zachowana własność zgodności RB)?

Odp: NIE

Dowód niewprost:

1. Załóżmy, że istnieje algorytm rozgłaszania, zapewniający przyczynowe uporządkowanie wiadomości i nie będący RB, ale BEB
2. Niech abstrakcje implementujące ten algorytm nazwane będą coBroadcast i coDeliver.

Zadanie 24 – dowód cd.

3. Jedynym przypadkiem spełniającym własności BEB i nie zapewniającym niezawodności jest naruszenie własności zgodności:
    a) Istnieje wykonanie, gdzie poprawny proces P odbiera wiadomość m (coDeliver), której nie odbierze (coDeliver) nigdy proces Q
4. Ponieważ algorytm spełnia własności BEB, jest to możliwe tylko wtedy, gdy oryginalny nadawca wiadomości R nie działa poprawnie.
5. Załóżmy teraz, że po otrzymaniu m, proces P rozgłasza (coBroadcast) wiadomość m’.
6. Jeśli P jest poprawny i rozgłaszanie spełnia warunki BEB, to to wszystkie poprawne procesy (włączając Q) otrzymają (coDeliver) m’.

Zadanie 24 – dowód cd.

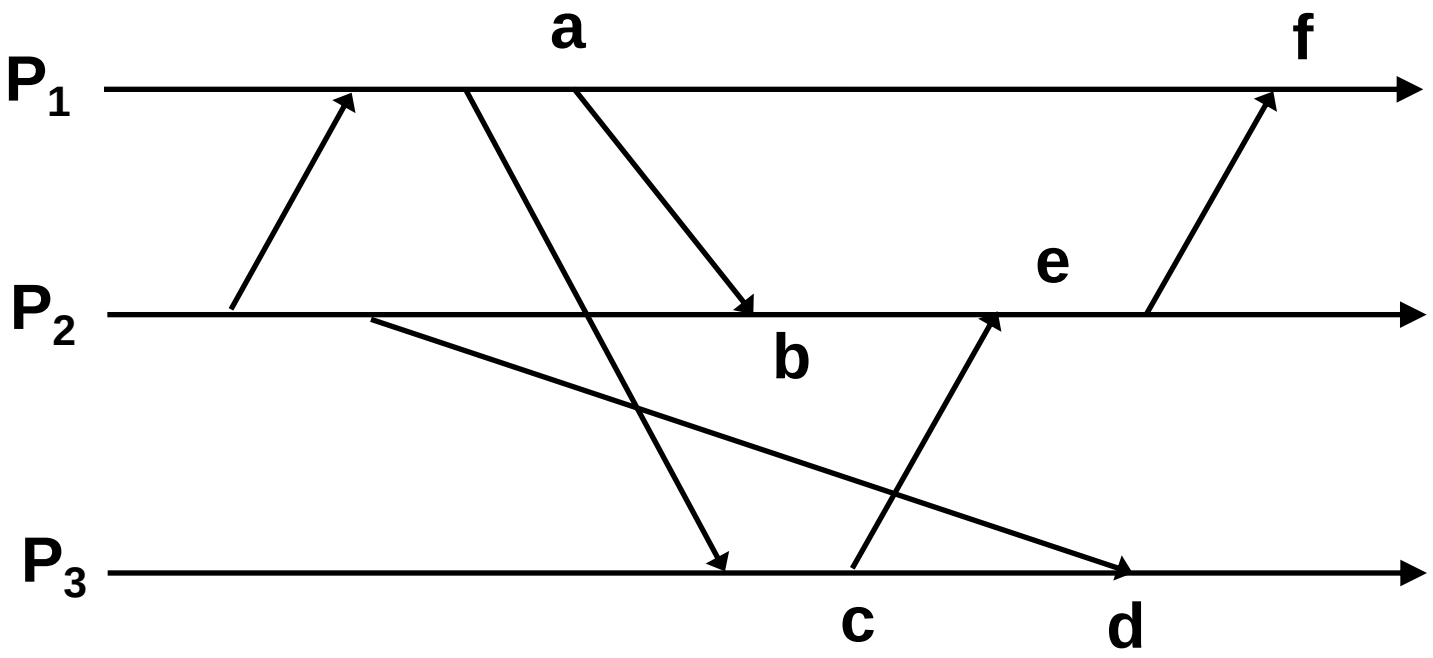
7. Zakładając, że m poprzedza m', proces Q musiał otrzymać (coDeliver) m – sprzeczność z 3a)

Stąd, każdy BEB który spełnia przyczynowy porządek dostarczania wiadomości także spełnia własność zgodności, więc jest niezawodny.

Zadania z zegarów logicznych / wektorowych

Zadanie 1

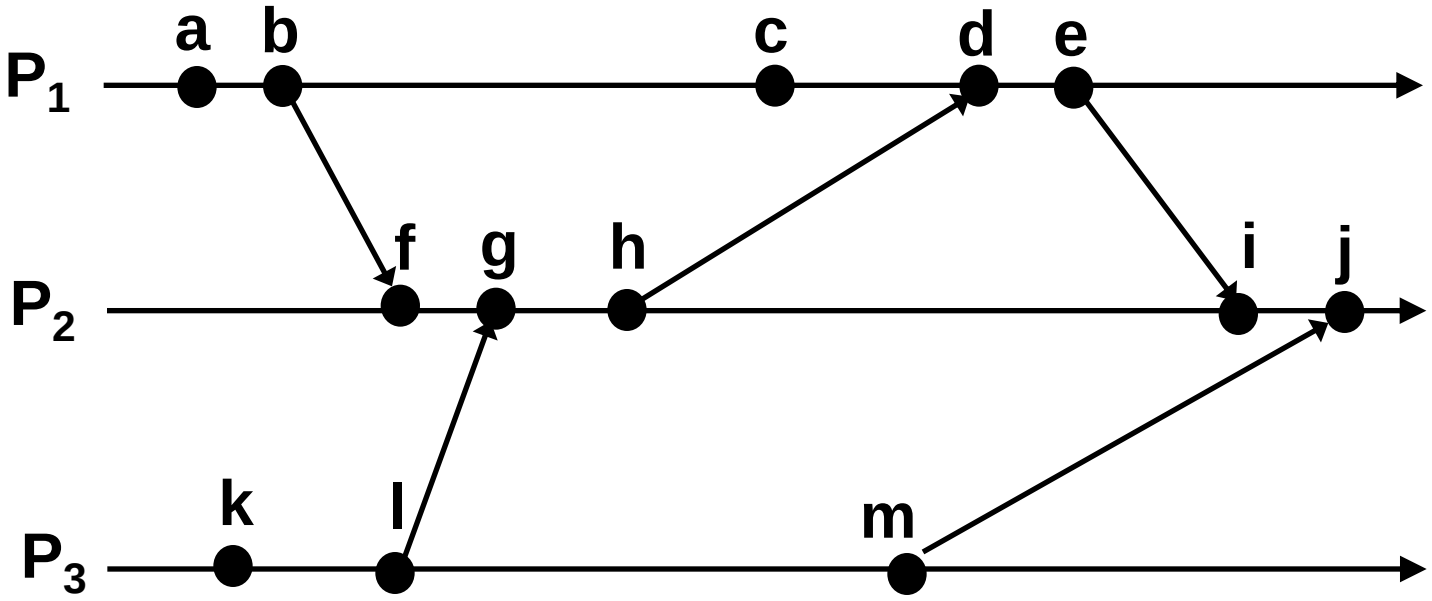
- Jakie są wartości zegarów wektorowych?
- Które zdarzenia są współbieżne, a które przyczynowo zależne?



Zadanie 2

a) Jakie są wartości zegarów wektorowych?

b) Jaka jest zależność pomiędzy zdarzeniami: f - j, a - m, c - j, k - e, j - k?



Zadanie 3

Udowodnij, że w każdym momencie czasu dla znaczników wektorowych zachodzi $V_j[i] \leq V_i[i]$

- Wartość znacznika czasowego Vi[i] procesu Pi = liczba zdarzeń, które zaszły do tej pory w procesie Pi
- Wartość znacznika czasowego Vj[i] procesu Pj = wartość zegara procesu Pi, którą zna proces Pj.
- Vi[i] jest zwiększana tylko przez proces Pi
- Przed wysłaniem wiadomości przez Pi Vi[i] jest inkrementowane i dołączane do wysyłanej wiadomości
- Pj otrzymując wiadomość od Pi aktualizuje wektor Vj, ustawiając każdy wpis Vj[k] na wartość maksimum{Vi[k], Vj[k]}.

Zadanie 4

Udowodnij, że jeśli $e \rightarrow e' \Rightarrow V(e) < V(e')$

1. Jeśli zdarzenia e i e' są kolejnymi zdarzeniami występującymi w tym samym procesie, to nierówność jest prawdziwa na podst. pkt.1 dotyczącego zegarów wektorowych
2. Jeśli istnieje wiadomość m taka, że $e=send(m)$, a $e'=rcv(m)$, to nierówność jest prawdziwa na podst. pkt.3 dotyczącego zegarów wektorowych
3. Załóżmy, że nierówność jest prawdziwa dla wszystkich par zdarzeń tworzących ciąg: $e_1, e_2, \ldots, e_{N+1}$ (dla każdej pary zdarzeń zachodzi pkt. 1 lub 2)

Zadanie 4

Udowodnij, że jeśli $e \rightarrow e' \Rightarrow V(e) < V(e')$

4. Załóżmy, że zdarzenia e i e' należą do rozważanego ciągu zdarzeń $e_1, e_2, \ldots, e_{N+1}$, przy czym $e=e_1$ i $e'=e_{N+1}$
5. Dla zadanego ciągu $e \rightarrow e_N$ z pkt. 1 i 2, czyli $V(e)<V(e_N)$
6. Ale $e_N \rightarrow e_{N+1}$, zatem na podstawie indukcji matematycznej $e \rightarrow e_{N+1}$, stąd $e \rightarrow e'$ oraz $V(e)<V(e')$.

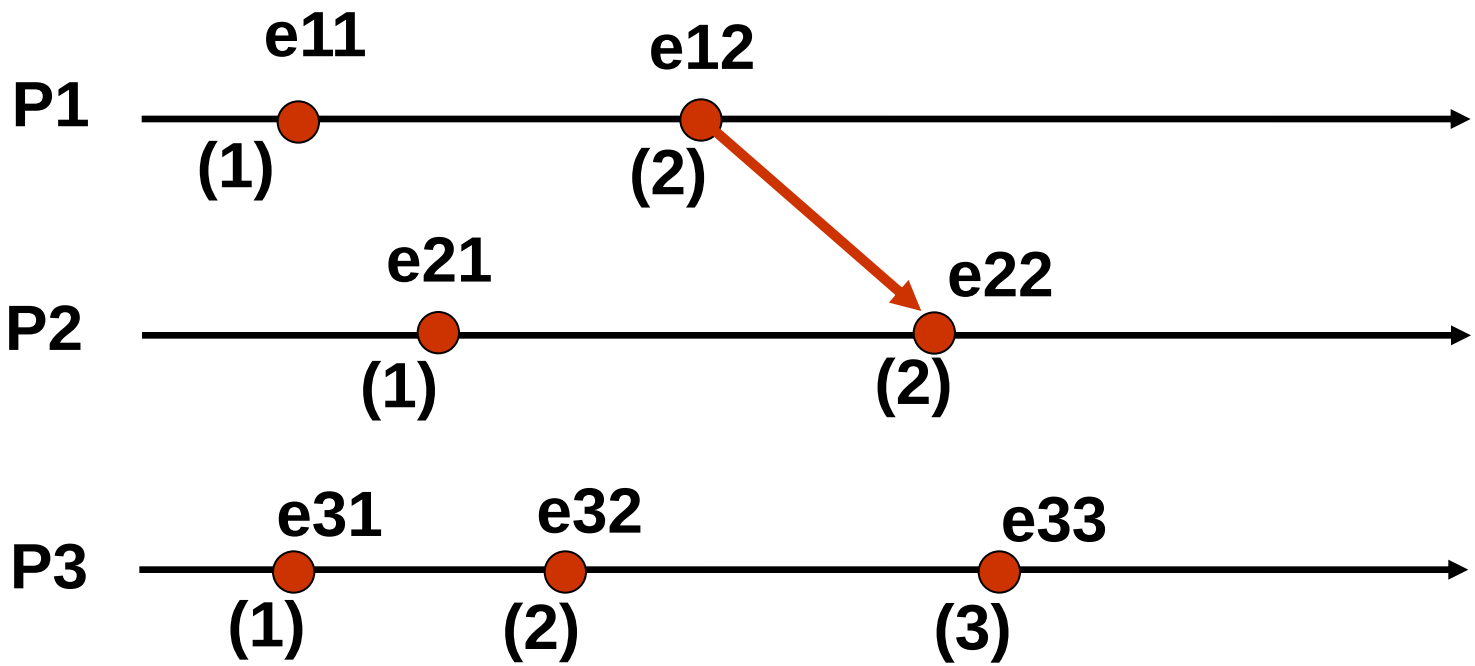
Zadanie 5

Udowodnij, że jeżeli V(e)< V(e') to e→e'

1. Załóżmy, że V(e)< V(e') oraz, że zdarzenia e i e' są współbieżne.
2. Niech e będzie zdarzeniem, które zaszło w procesie P_i oraz e' – zdarzeniem w procesie P_j
3. Oznacza to, że $V_i[i] > V_j[i]$, oraz $V_j[j] > V_i[j]$
4. Stąd, V(e) i V(e') są nieporównywalne.
5. Sprzeczność z założeniem, zatem e→e'.

Zadanie 6

Dla zegarów Lamporta spełniona jest zależność: e→e' ⇒ L(e)<L(e'). Czy prawdziwa jest implikacja odwrotna?



L(e11)<L(e12) i L(e11)<L(e32), e11 i e32 nie są powiązane przyczynowo.

Zadanie 6

Dla zegarów Lamporta spełniona jest zależność: $e \rightarrow e' \Rightarrow L(e)<L(e')$. Czy prawdziwa jest implikacja odwrotna?

Zegary Lamporta mogą tylko zagwarantować, że jeśli

$$L(a)<L(b), \text{ to } b \mid a$$

(przyszłość nie wpływa na przeszłość),

ale nie potrafimy powiedzieć, czy a i b są przyczynowo powiązane czy nie.