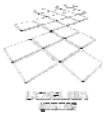
Mechanizmy rozgłaszania niezawodnego



Rozgłaszanie niezawodne – definicja nieformalna

Nieformalnie, przez rozgłaszanie rozumiemy mechanizm (abstrakcję) komunikacyjny, za pomocą którego proces może wysłać wiadomość do grupy procesów.

W mechanizmach rozgłaszania niezawodnego gwarantowane są ponadto pewne własności pomimo występowania awarii.

Przykładowe zastosowania mechanizmów niezawodnego rozgłaszania

- Systemy z wieloma uczestnikami
- Zwielokrotnianie (replikacja)

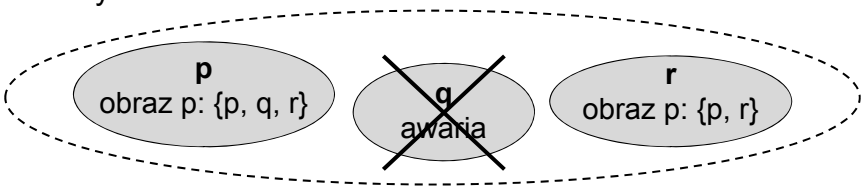
Podstawowe definicje (1)

- Komunikacja grupowa (ang. *group communication*) to mechanizm umożliwiający rozsyłanie (ang. *multicast*) poprzez organizowanie procesów (szerzej – obiektów) w grupy
- Obejmuje dwa aspekty: zarządzanie grupami procesów (usługa członkostwa, ang. *membership service*) oraz algorytmy (niezawodnego) rozsyłania wiadomości w grupie
- Pozwala modelować niezawodną komunikację procesów przy założeniu, że zbiór procesów dynamicznie się zmienia

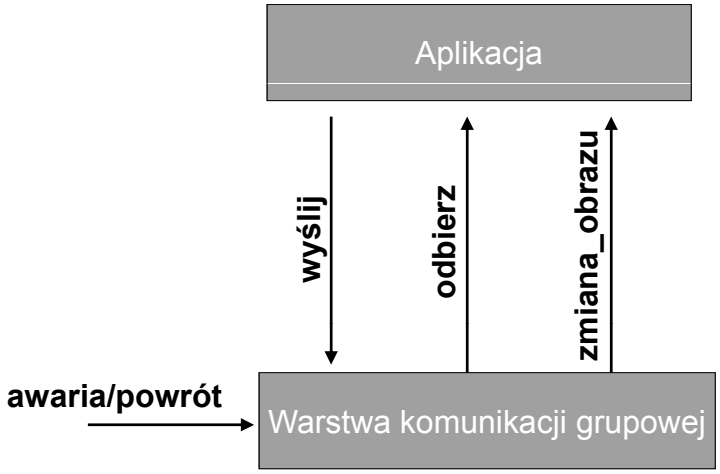
Podstawowe definicje (2)

Grupa – rzeczywisty zbiór procesów uczestniczących we wspólnym przetwarzaniu i komunikujących się poprzez przekazywanie wiadomości

Obraz grupy (ang. *view*) – grupa, widziana w danej chwili czasu rzeczywistego w pojedynczym procesie; obraz grupy jest generowany przez usługę członkostwa; różne procesy mogą mieć różne obrazy tej samej grupy w tym samym momencie

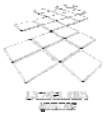


Architektura systemu



Przykładowe systemy

- ISIS – pionierski system GCS
  http://www.cs.cornell.edu/Info/Projects/Isis/
- Horus (Ensemble) – nowoczesna wersja ISIS
  http://www.cs.cornell.edu/Info/Projects/HORUS/
- Jgroups – system GCS dla języka Java
  http://www.jgroups.org
- Transis
  http://www.cs.huji.ac.il/labs/transis/



Klasy mechanizmów rozgłaszania niezawodnego

- Podstawowe rozgłaszanie niezawodne
  (ang. *best-effort broadcast* - BEB)
- Zgodne rozgłaszanie niezawodne
  (ang. *regular reliable broadcast* - RB)
- Jednolite rozgłaszanie niezawodne
  (ang. *uniform reliable broadcast - URB*)
- Zgodne rozgłaszanie niezawodne z uporządkowainiem FIFO wiadomości
  (ang. *FIFO reliable broadcast*)
- Zgodne rozgłaszanie niezawodne z przyczynowym uporządkowaniem wiadomości
  (ang. *causal reliable broadcast*)
- Zgodne rozgłaszanie niezawodne z globalnym uporządkowaniem wiadomości
  (ang. *total order reliable broadcast*)

Podstawowe rozgłaszanie niezawodne BEB : specyfikacja

Mechanizm podstawowego rozgłaszania niezawodnego (ang. *best-effort broadcast*) ma następujące własności:

- **Ważność** (ang. *best-effort validity*): Jeżeli procesy P_i oraz P_j są poprawne, to każda wiadomość rozgłaszana przez P_i jest ostatecznie dostarczona do P_j
- **Brak powielania** (ang. *no duplication*): Jeżeli wiadomość jest dostarczona, to jest dostarczona co najwyżej raz
- **Brak samogeneracji** (ang. *no creation*): Jeżeli jakaś wiadomość jest dostarczona do procesu P_j, to została wcześniej rozgłoszona przez jakiś proces P_i

Algorytm podstawowego rozgłaszania niezawodnego: operacje komunikacyjne

Operację rozgłaszania gwarantującą własności podstawowego rozgłaszania niezawodnego oznaczymy przez:

$$\boldsymbol{send}^{\boldsymbol{BRB}}(P_i, \mathcal{P}, M) \qquad (12.1)$$

Zdarzenie odpowiadające tej operacji zapisywać będziemy jako $e\_send^{BRB}(P_i, \mathcal{P}, M)$.

Analogicznie, przez

$$\boldsymbol{deliver}^{\boldsymbol{BRB}}(P_j, P_i, M) \qquad (12.2)$$

oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{BRB}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M do procesu aplikacyjnego P_i.

Algorytm podstawowego rozgłaszania niezawodnego: założenia

Algorytm podstawowego rozgłaszanie niezawodnego (ang. *best-effort broadcast*) zakłada, że dostępny jest mechanizm kanałów niezawodnych (ang. *perfect point-to-point links*).

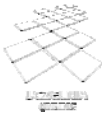
Algorytm nie wymaga detektorów awarii, a więc przyjmuje model przetwarzania z ukrytymi awariami (ang. *fail-silent*).

Algorytm podstawowego rozgłaszania niezawodnego (1)

```
type PACKET extends FRAME is record of
      data        : MESSAGE
    end record


pcktOut  : PACKET
msgIn    : MESSAGE
```

Algorytm podstawowego rozgłaszanie niezawodnego (2)

```
1.  when e_send^BRB(P_i, 𝒫, msgOut: MESSAGE) do
2.    pcktOut.data := msgOut
3.    for all P_j ∈ 𝒫 do
4.        send^PL(Q_i, Q_j, pckOut);
5.    end for
6.  end when

7.  when e_receive^PL(Q_j, Q_i, pcktIn: PACKET) do
8.      msgIn := pckIn.data
9.      deliver^BRB(P_j, P_i, msgIn)
10. end when
```

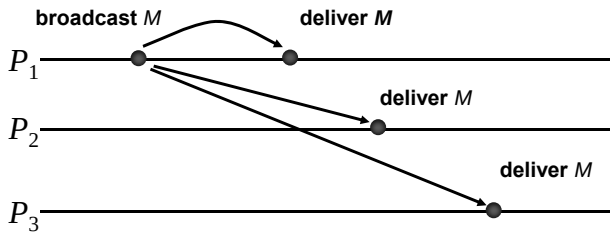


Algorytm podstawowego rozgłaszania niezawodnego: złożoność

Pomijając czas przetwarzania lokalnego, nadawca wysyła wiadomość do wszystkich procesów w 1 kroku. Zatem:

- złożoność czasowa wynosi 1,
- złożoność komunikacyjna wynosi *n*

Ilustracja podstawowego rozgłaszania niezawodnego



Legenda:

broadcast M : operacja **send**$^{BRB}(P_i, \mathcal{P}, M)$

deliver M : operacja **deliver**$^{BRB}(P_j, P_i, M)$

Zgodne rozgłaszanie niezawodne RB : specyfikacja

Mechanizm zgodnego rozgłaszania niezawodnego (ang. *regular reliable broadcast*) ma następujące własności:

- **Ważność** (ang. *validity*): Jeżeli proces P_i jest poprawny, to każda wiadomość rozgłaszana przez ten proces jest ostatecznie dostarczona do P_i
- **Brak powielania** (ang. *no duplication*): Jeżeli wiadomość jest dostarczona, to jest dostarczona co najwyżej raz
- **Brak samogeneracji** (ang. *no creation*): Jeżeli jakaś wiadomość została dostarczona do procesu P_i, to została wcześniej rozgłoszona przez jakiś proces P_i
- **Zgodność** (ang. *agreement*): Jeżeli jakaś wiadomość została odebrana przez pewien poprawny proces P_i, to ostatecznie wszystkie poprawne procesy odbiorą tę wiadomość

Zgodne rozgłaszanie niezawodne: operacje komunikacyjne

Operację rozgłaszania gwarantującą własności zgodnego rozgłaszania niezawodnego oznaczymy przez:

$$\boldsymbol{send}^{\boldsymbol{RRB}}(P_i, \mathcal{P}, M) \qquad (12.3)$$

Zdarzenie odpowiadające tej operacji zapisywać będziemy jako $e\_send^{RRB}(P_i, \mathcal{P}, M)$.

Analogicznie, przez

$$\boldsymbol{deliver}^{\boldsymbol{RRB}}(P_j, P_i, M) \qquad (12.4)$$

oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{RRB}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M do procesu aplikacyjnego P_i.

Pasywny algorytm zgodnego rozgłaszania niezawodnego: założenia

Pasywny algorytm zgodnego rozgłaszania niezawodnego (ang. *lazy reliable broadcast*) zakłada, że dostępne są następujące mechanizmy:

- doskonały detektor awarii
- podstawowe rozgłaszanie niezawodne

Pasywny algorytm zgodnego rozgłaszania niezawodnego (1)

```
type PACKET extends FRAME is record of
      origin  : PROCESS_ID
      data        : MESSAGE
    end record
```

Pasywny algorytm zgodnego rozgłaszania niezawodnego (2)

```
pcktIn      : PACKET
msgOut      : MESSAGE
correct_i   : set of PROCESS_ID := 𝒫
delivered_i : set of MESSAGE := ∅
from_i      : array [1..n] of set of
                  pair of ⟨PROCESS_ID, MESSAGE⟩ := ∅
pId         : PROCESS_ID
msg         : MESSAGE
```

Pasywny algorytm zgodnego rozgłaszania niezawodnego (3)

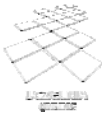
```
1.  when e_send^RRB(P_i, 𝒫, msgOut: MESSAGE) do
2.    pcktOut.origin := P_i
3.    pcktOut.data := msgOut
4.    send^BRB(Q_i, 𝒬, pcktOut)
5.  end when
```

Pasywny algorytm zgodnego rozgłaszania niezawodnego (4)

```
6.  when e_receive^BRB(Q_j, Q_i, pcktIn: PACKET) do
7.    msgIn := pckIn.data
8.    if msgIn ∉ delivered_i then
9.      delivered_i := delivered_i ∪ {msgIn}
10.     deliver^RRB(pcktIn.origin, P_i, msgIn)
11.     from_i[j] := from_i[j] ∪ ⟨pcktIn.origin, msgIn⟩
12.     if P_j ∉ correct_i then
13.       send^BRB(Q_i, 𝒬, pcktIn)
14.     end if
15.   end if
16. end when
```

Pasywny algorytm zgodnego rozgłaszania niezawodnego (5)

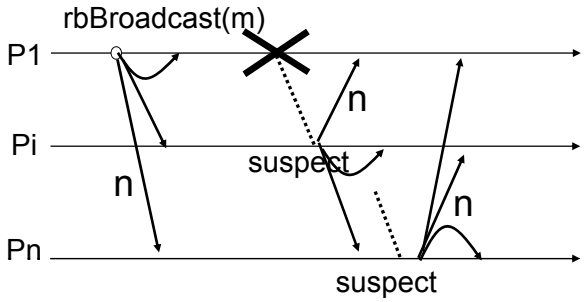
```
17.  when e_crash(P_j) do
18.    correct_i := correct_i \ {P_j}
19.    for all ⟨pId, msg⟩ ∈ from_i[j] do
20.          pcktOut.origin := pId
21.          pcktOut.data := msg
22.          send^BRB(Q_i, 𝒬, pcktOut)
23.    end for
24.  end when
```



Pasywny algorytm zgodnego rozgłaszania niezawodnego: złożoność

- Przypadek optymistyczny:
  - złożoność czasowa wynosi 1,
  - złożoność komunikacyjna wynosi n
- Przypadek pesymistyczny:
  - złożoność czasowa wynosi n,
  - złożoność komunikacyjna wynosi n^2

Pasywny algorytm zgodnego rozgłaszania niezawodnego: złożoność



Aktywny algorytm zgodnego rozgłaszania niezawodnego: założenia

Aktywny algorytm zgodnego rozgłaszanie niezawodnego (ang. *eager reliable broadcast*) zakłada, że dostępny jest mechanizm podstawowego rozgłaszania niezawodnego.

Aktywny algorytm zgodnego rozgłaszania niezawodnego (1)

```
type PACKET extends FRAME is record of
      origin  : PROCESS_ID
      data    : MESSAGE
    end record


pcktIn       : PACKET
msgOut       : MESSAGE
delivered_i  : set of MESSAGE := ∅
```

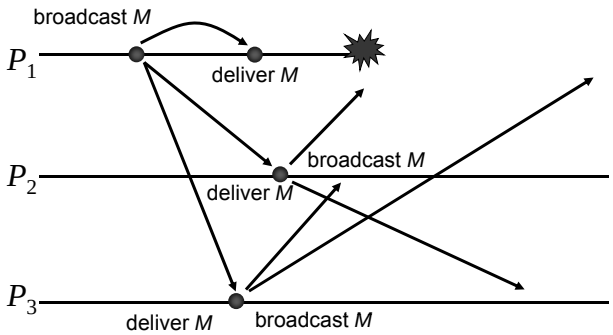
Aktywny algorytm zgodnego rozgłaszania niezawodnego (2)

```
1.  when e_send^RRB(P_i, 𝒫, msgOut: MESSAGE) do
2.    pcktOut.data := msgOut
3.    pcktOut.origin := P_i
4.    deliver^RRB(P_i, P_i, msgOut)
5.    delivered_i := delivered_i ∪ {msgOut}
6.    send^BRB(Q_i, 𝒬, pcktOut)
7.  end when
```

Aktywny algorytm zgodnego rozgłaszania niezawodnego (3)

```
8.   when e_receive^BRB(Q_j, Q_i, pcktIn: PACKET) do
9.     msgIn := pckIn.data
10.    if msgIn ∉ delivered_i then
11.      delivered_i := delivered_i ∪ {msgIn}
12.      deliver^RRB(pcktOut.origin, P_i, msgIn)
13.      send^BRB(Q_i, 𝒬 , pcktOut)
14.    end if
15.  end when
```

Aktywny algorytm zgodnego rozgłaszania niezawodnego: przykład (1)

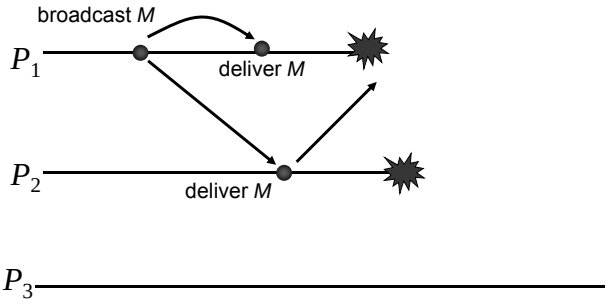


Legenda:

broadcast M : operacja **send**$^{RRB}(P_i, \mathcal{P}, M)$

deliver M : operacja **deliver**$^{RRB}(P_j, P_i, M)$

Aktywny algorytm zgodnego rozgłaszania niezawodnego: przykład (2)



Legenda:

broadcast M : operacja $\boldsymbol{send}^{RRB}(P_i, \mathcal{P}, M)$

deliver M : operacja $\boldsymbol{deliver}^{RRB}(P_j, P_i, M)$

Aktywny algorytm zgodnego rozgłaszania niezawodnego: złożoność

- Przypadek optymistyczny:
  - złożoność czasowa wynosi 1,
  - złożoność komunikacyjna wynosi n^2
- Przypadek pesymistyczny:
  - złożoność czasowa wynosi n,
  - złożoność komunikacyjna wynosi n^2

Jednolite rozgłaszanie niezawodne URB: specyfikacja

Mechanizm jednolitego rozgłaszania niezawodnego (ang. *uniform reliable broadcast*) ma następujące własności:

- **Ważność** (ang. *validity*): Jeżeli proces P_i jest poprawny, to każda wiadomość rozgłaszana przez ten proces jest ostatecznie dostarczona do P_i
- **Brak powielania** (ang. *no duplication*): Jeżeli wiadomość jest dostarczona, to jest dostarczona co najwyżej raz
- **Brak samogeneracji** (ang. *no creation*): Jeżeli jakaś wiadomość jest dostarczona do procesu P_j, to została wcześniej rozgłoszona przez jakiś proces P_i
- **Jednolita zgodność** (ang. *uniform agreement*): Jeżeli jakaś wiadomość została odebrana przez pewien proces P_i (poprawny bądź niepoprawny), to ostatecznie wszystkie poprawne procesy odbiorą tę wiadomość

Jednolite rozgłaszanie niezawodne: operacje komunikacyjne

Operację rozgłaszania gwarantującą własności jednolitego rozgłaszania niezawodnego oznaczymy przez:

$$\mathbf{send}^{\mathbf{URB}}(P_i, \mathcal{P}, M) \qquad (12.5)$$

Zdarzenie odpowiadające tej operacji zapisywać będziemy jako $e\_send^{URB}(P_i, \mathcal{P}, M)$.

Analogicznie, przez

$$\mathbf{deliver}^{\mathbf{URB}}(P_j, P_i, M) \qquad (12.6)$$

oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{URB}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M do procesu aplikacyjnego P_i.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich: założenia

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (ang. *all-ack uniform reliable broadcast*) zakłada, że dostępne są następujące mechanizmy:

- podstawowe rozgłaszania niezawodnego
- niezawodne kanały komunikacyjne
- doskonały detektor awarii

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (1)

```
type PACKET extends FRAME is record of
      origin  : PROCESS_ID
      data    : MESSAGE
    end record
```

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (2)

```
pcktIn      : PACKET
msgOut      : MESSAGE
correct_i   : set of PROCESS_ID := 𝒫
delivered_i : set of MESSAGE := ∅
pending_i   : set of PACKET := ∅
ack_i       : set of pair of
                  ⟨PACKET, PROCESS_ID⟩ := ∅
result      : set of PROCESS_ID
pcktAck     : PACKET
pId         : PROCESS_ID
pckt        : PACKET
```

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (3)

```
1.  function SELECT(pcktIn: PACKET)
2.    result := ∅
3.    for all ⟨pcktAck, pId⟩ ∈ ack_i ∧
4.                        pcktAck = pcktIn do
5.        result := result ∪ {pId}
6.    end for
7.    return result
8.  end function
```

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (4)

```
9.  when e_send^URB(P_i, 𝒫, msgOut: MESSAGE) do
10.   pcktOut.origin := P_i
11.   pcktOut.data := msgOut
12.   pending_i := pending_i ∪ {pcktOut}
13.   send^BRB(Q_i, 𝒬, pcktOut)
14. end when

15. when e_crash(P_j) do
16.   correct_i := correct_i \ {P_j}
17. end when
```

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (5)

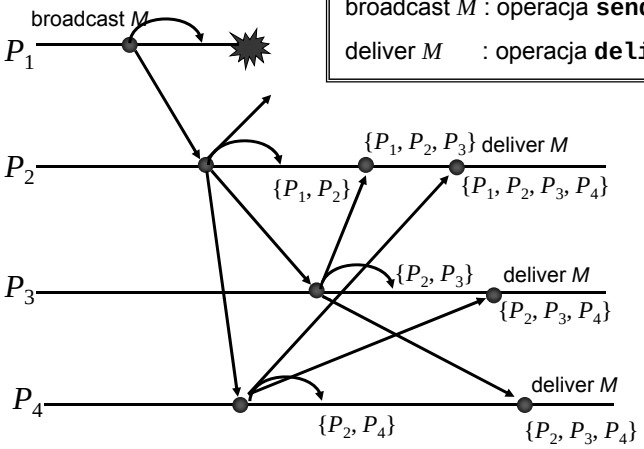
```
18. when e_receive^BRB(Q_j, Q_i, pcktIn: PACKET) do
19.   ack_i := ack_i ∪ ⟨pcktIn, P_j⟩
20.   if pcktIn ∉ pending_i then
21.     pending_i := pending_i ∪ {pcktIn}
22.     send^BRB(Q_i, 𝒬, pcktIn)
23.   end if
24. end when

25. when ∃pckt :: pckt ∈ pending_i ::
      pckt.data ∉ delivered_i ∧ correct_i ⊆ SELECT(pckt) do
26.     delivered_i := delivered_i ∪ {pckt.data}
27.     deliver^URB(pckt.origin, P_i, pckt.data)
28. end when
```

Ilustracja algorytmu jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich

Legenda:

broadcast M : operacja **send**$^{\text{URB}}(P_i, \mathcal{P}, M)$

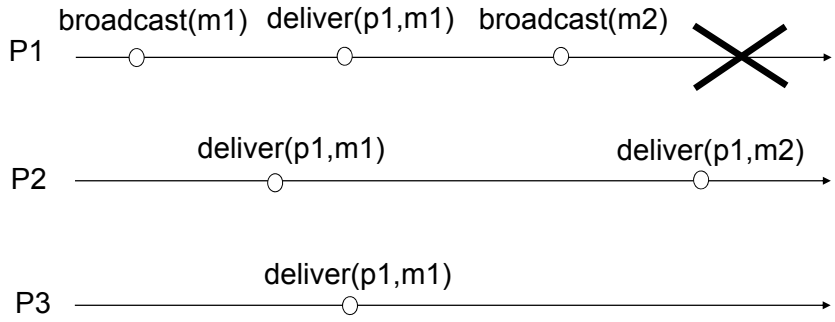
deliver M : operacja **deliver**$^{\text{URB}}(P_j, P_i, M)$



Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich: złożoność

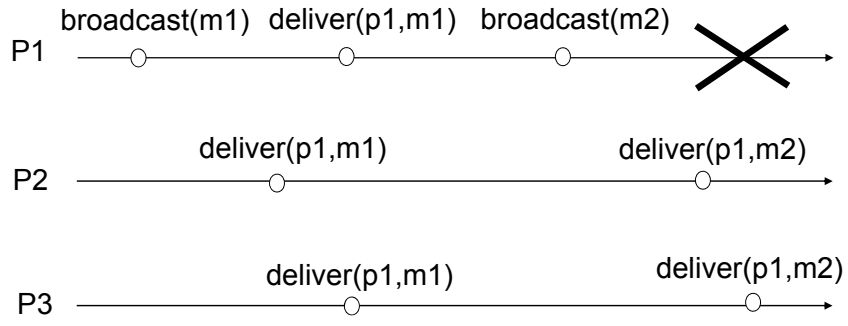
- Przypadek optymistyczny:
  - złożoność czasowa wynosi 2
  - złożoność komunikacyjna wynosi n^2
- Przypadek pesymistyczny:
  - złożoność czasowa wynosi $n + 1$
  - złożoność komunikacyjna wynosi n^2

Jaki rodzaj rozgłaszania przedstawiony jest na rys?



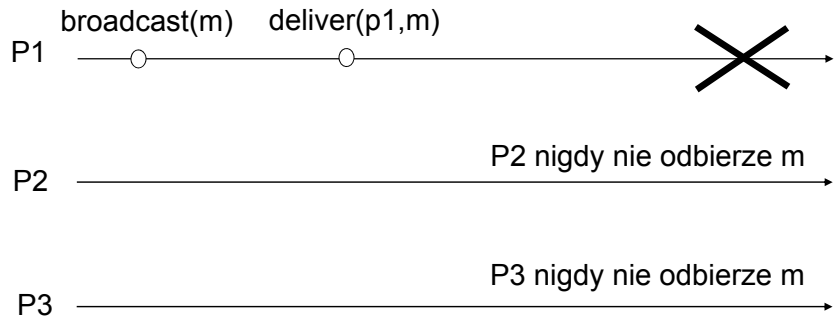
P1 i P3 nigdy nie odbiorą m2

Jaki rodzaj rozgłaszania przedstawiony jest na rys?



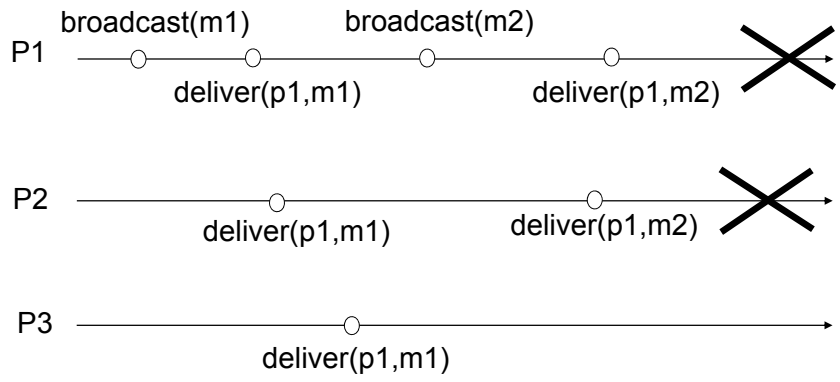
Wszystkie poprawne procesy odbiorą m1 i m2 – BEB i RB i URB

Jaki rodzaj rozgłaszania przedstawiony jest na rys?



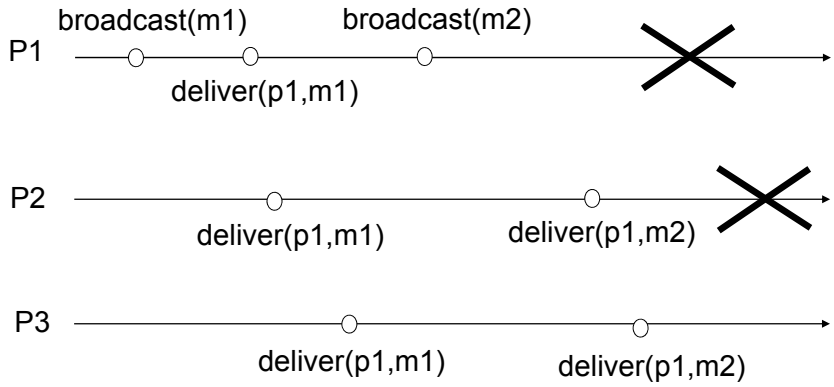
Żaden poprawny proces nie odbierze m – BEB i RB

Jaki rodzaj rozgłaszania przedstawiony jest na rys?



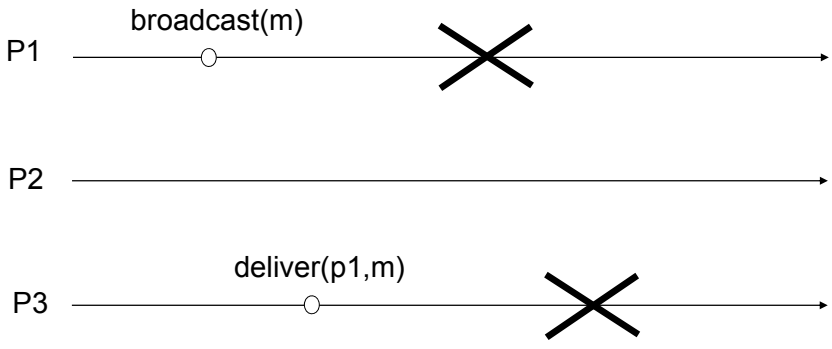
Żaden poprawny proces nie odbierze m2, ale wszystkie niepoprawne odbiorą m2 (we don't care) – BEB i RB

Jaki rodzaj rozgłaszania przedstawiony jest na rys?



BEB, RB i URB

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



Jest spełniona własność zgodności, ale nie jednolitej zgodności

Casual 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 m2
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)

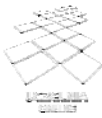
RB1, RB2, RB3, RB4

FIFO Order: Jeżeli proces Pi rozgłosił wiadomość m1 przed m2, to każdy proces Pj nie odbierze m2 jeśli nie odebrał wcześniej m1

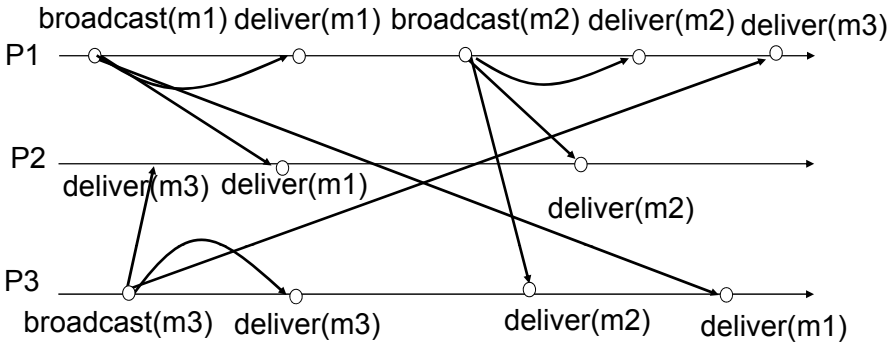
Reliable Casual Broadcast (RCB)

RB1, RB2, RB3, RB4

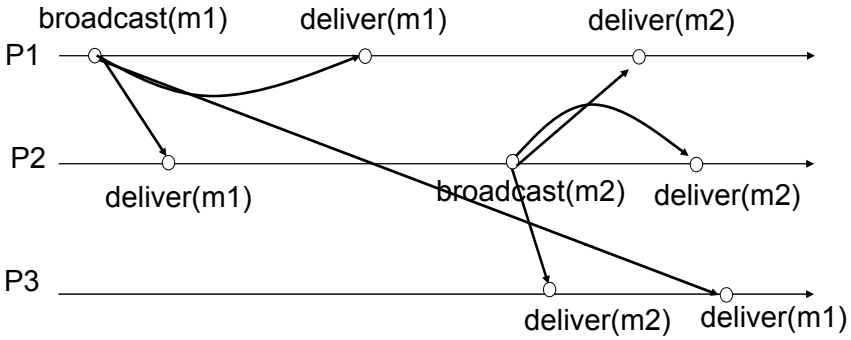
Casual order: Proces Pi nie odbierze wiadomości m2, dopóki nie odebrał wszystkich wiadomości m1, takich, że $m1 \rightarrow m2$



Zaproponuj przetwarzanie spełniające RB, ale nie spełniające FIFO



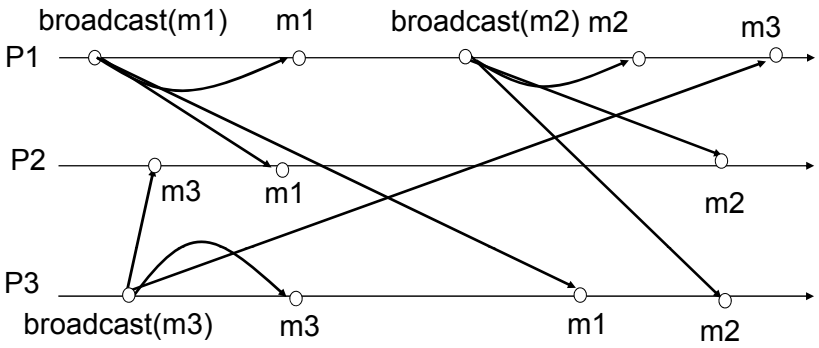
Podaj przykład przetwarzania, które spełnia warunki RFB i nie spełnia warunków RCB



$m1 \rightarrow m2$

m2 odebrane przed m1

Zaproponuj przetwarzanie spełniające RB, RFB i RCB



Total Order Broadcast (TO)

RB1, RB2, RB3, RB4

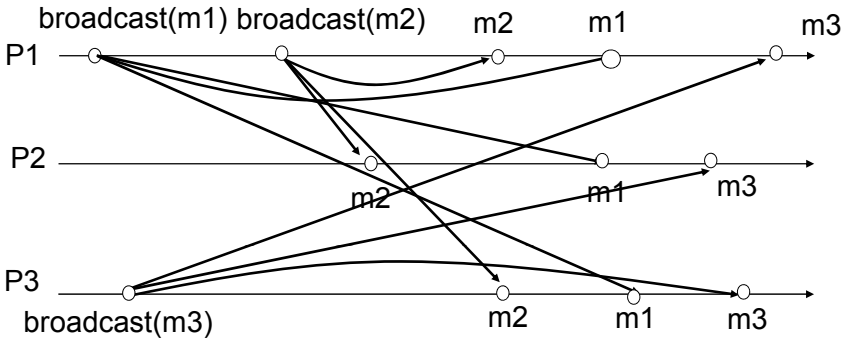
Globalne uporządkowanie wiadomości (total order):

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)

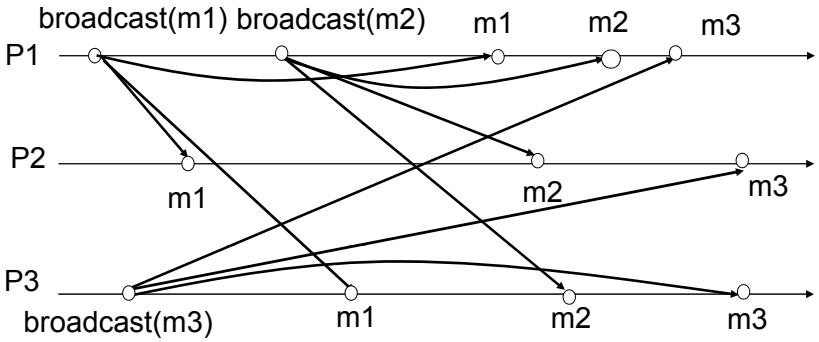
jeśli Pi i Pj odbierają wiadomość m i jeśli Pi odbiera m' przed m, to także Pj odbiera m' przed m.

Czy na poniższym rys. jest zapewniony TO?



TO – tak ; FIFO – nie

Czy na poniższym rys. jest zapewniony TO?



TO – tak; FIFO – tak; casual – tak