TANCERZE EMERYTI

Błażej Krzyżanek *136749* github.com/BlazejKrzyzanek
Maciej A. Czyżewski *136698* github.com/maciejczyzewski

PART 1: idea

Zadanie

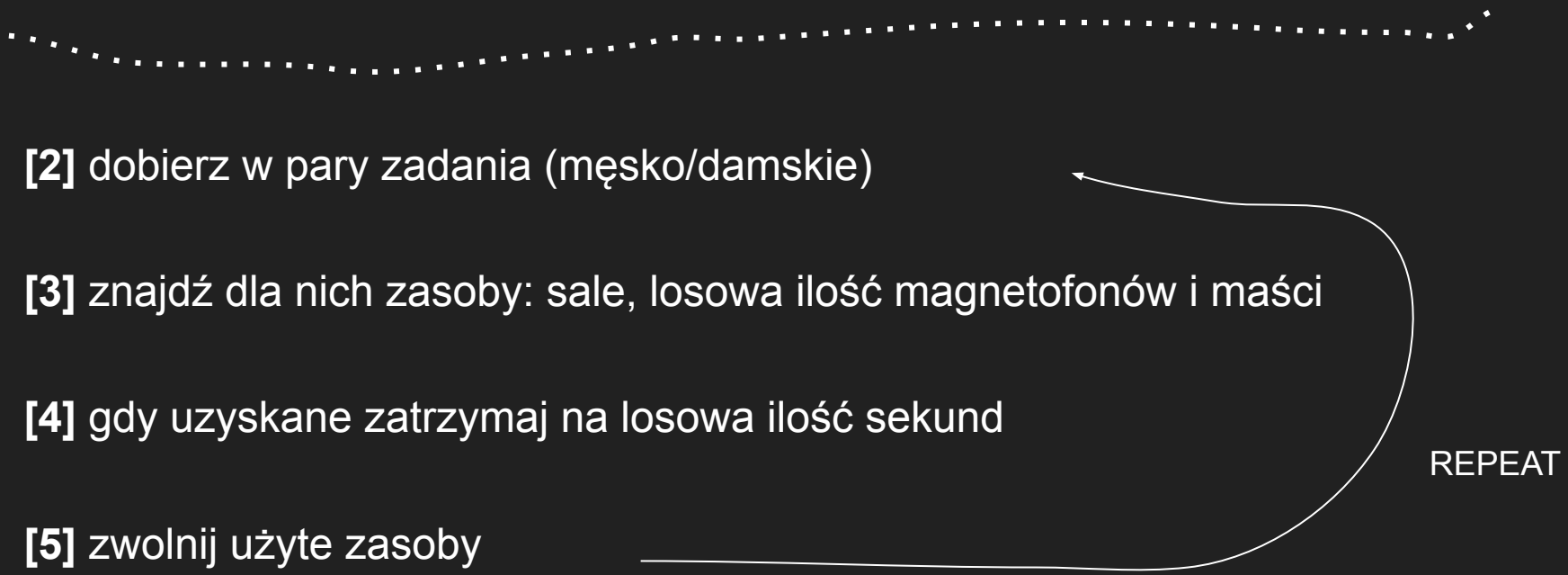
[1] eksploruj dostępne procesy i zasoby w systemie

[2] dobierz w pary zadania (męsko/damskie)

[3] znajdź dla nich zasoby: sale, losowa ilość magnetofonów i maści

[4] gdy uzyskane zatrzymaj na losowa ilość sekund

[5] zwolnij użyte zasoby



Dobieranie w pary

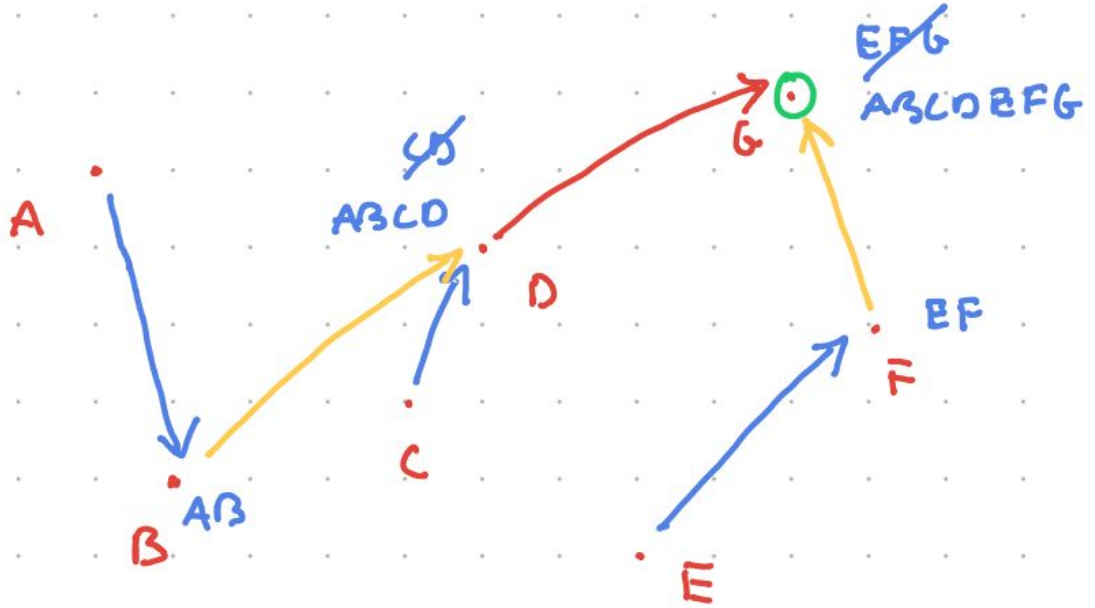
procesy dlatego nie są głodzone (sprawiedliwy podział)

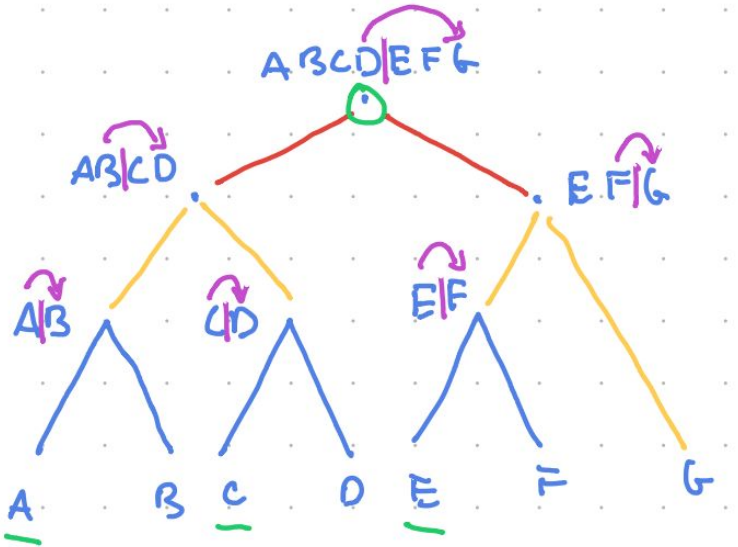
[1] wylosuj liczbę R (wygeneruj z niej drzewo MST)

[2] dla liści wyślij rozkazy i liczbę R (aby każdy lokalnie znal topologie)

[3] kolejne wierzchołki przekazują informacje

[4] glowna funkcja wykonuje się w root-cie (zielony kolor) on podejmuje decyzje na bazie aktualnego snapshot-u





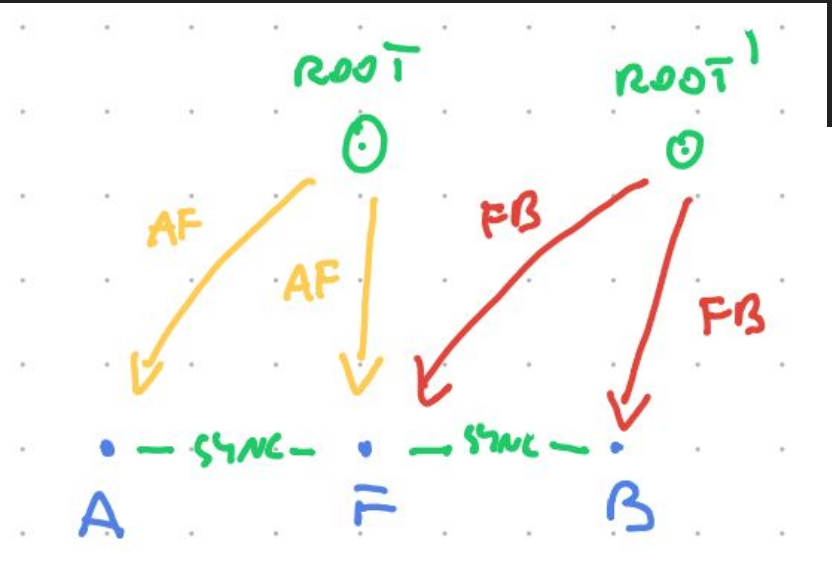
Root np. wybrał parę (AF) ale współbieżnie inny root innej decyzji wybrał (FB), czyli kolizja

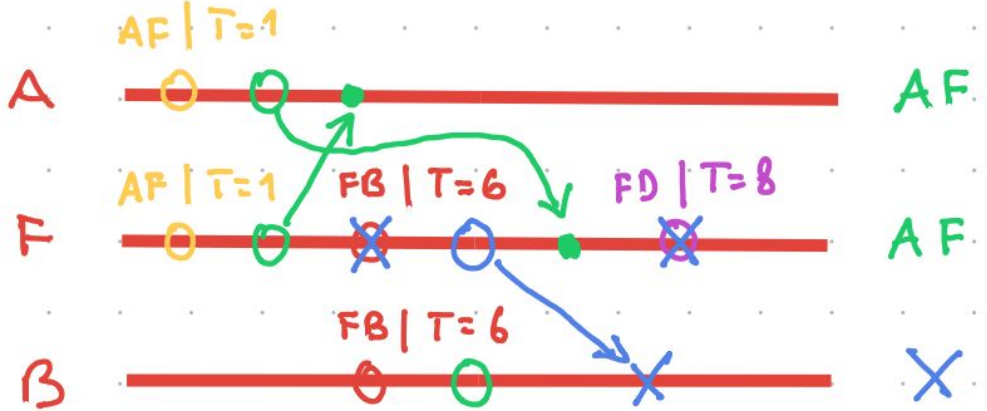
root mający prawie aktualne snapshot-y podejmuje decyzje a rozwiązanie wysyła do przeznaczonych odbiorców (w naszym wypadku członków pary)

[1] protokół potwierdzenia pomiędzy parami (wygrywa ten kto ma niższy *T*)

[2] zapisujemy lokalnie w tablicy aktualnie potwierdzaną parę

[3] dla większych T niz aktualnie wysyłamy anulowanie (niebieskie)

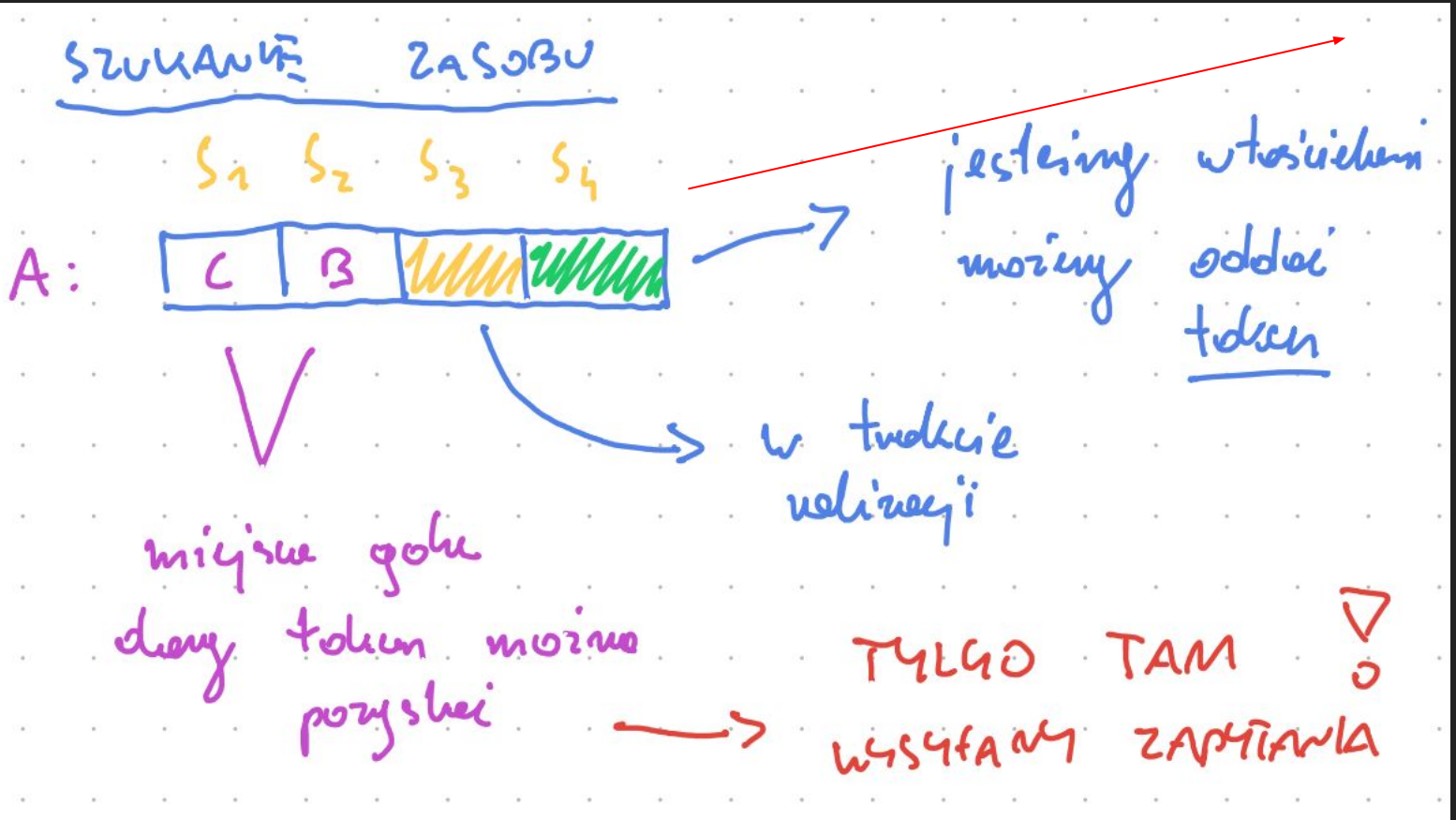




Zasoby (czyli sale, magnetofony, maści)

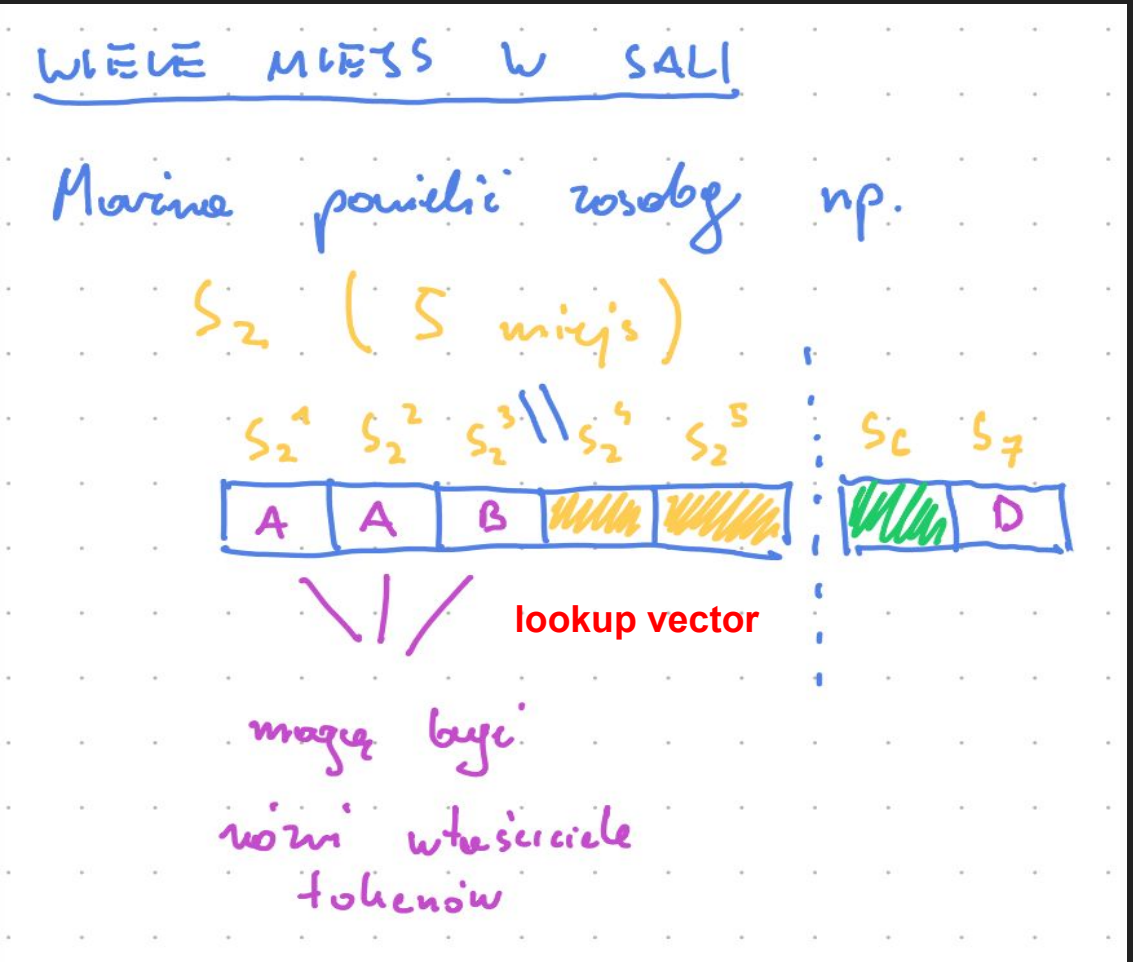
każdy process (emeryt) posiada tablice zasobów - może być albo **aktualnym właścicielem** (posiadaczem tokena), lub **wiedzieć kto go ma**

możliwość dodatkowych informacji: średni czas przetwarzania, czy zajęty i kiedy ostatnio (dlatego służy też jako **lookup vector**)



dzięki temu mamy wysoka ziarnistość, blokady są zakładane tylko w procesie wlasciciela - wiec można kontynuować pozyskiwanie innych zasobów bez potrzeby angażowania całej sieci procesów

Okay, ale sale mają swoja pojemnosc, tak?

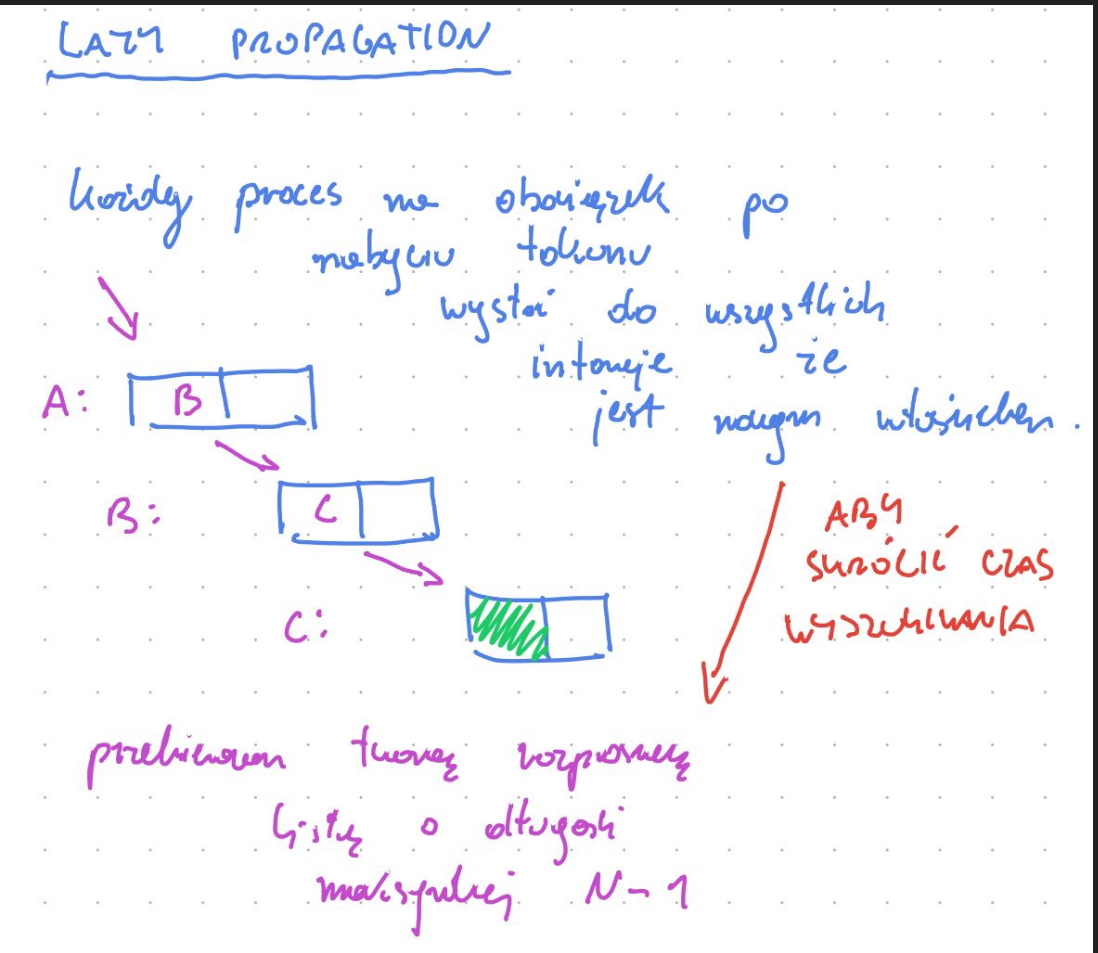


np. S1 - masc; S2 - masc; S3 - magnetofon
S4.1 - 1 miejsce w sali 4
S4.2 - 2 miejsce w sali 4
S4.3 - 3 miejsce w sali 4

pojedyncze miejsce można potraktować jako zasób - para aby tańczyć musi zarezerwować 2 miejsca - jest wiele możliwych strategii, przykład:

1) para komunikuje się ustalając numer sali (może wybrać tą salę która w **lookup vector** ma wolne przynajmniej 2 miejsca) - próbuje zająć asynchronicznie (osobno) - dopóki oba nie mają - czekają na siebie

Co gdy wskazywany wlasciciel juz nie posiada tokena?

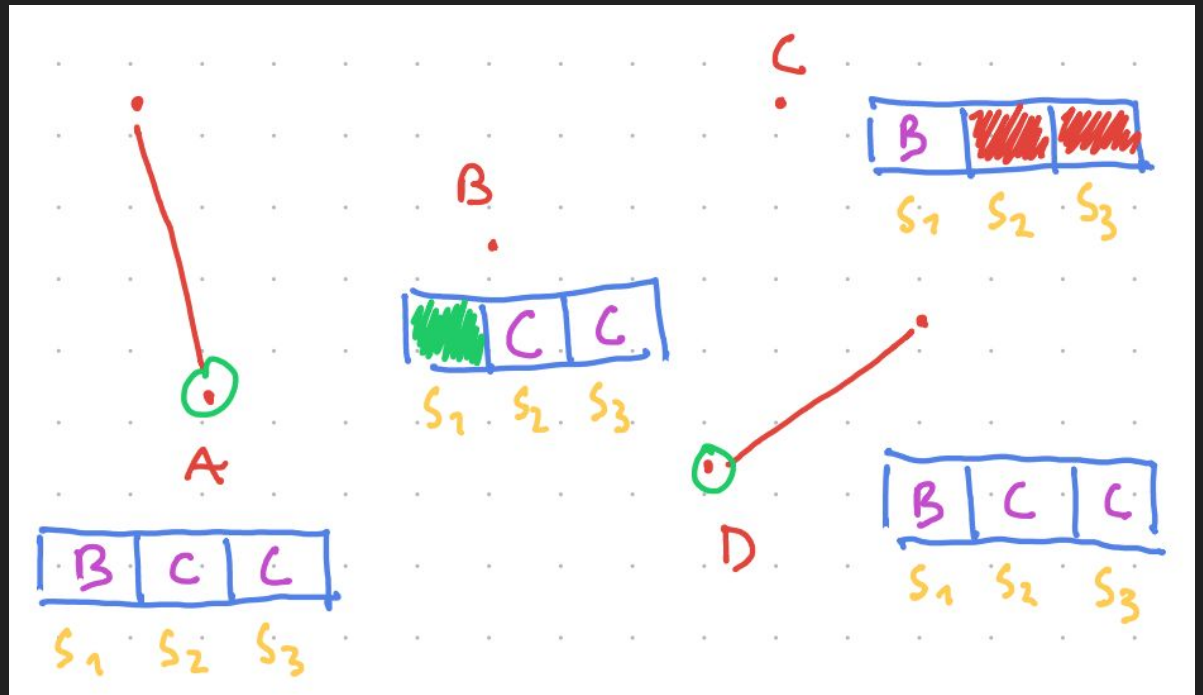


spodziewana średnia złożoność znalezienia wolnego zasobu to O(1) - jednak w najgorszym przypadku może wynosić O(N - 1), nie psuje to jednak poprawności

wszystko zależy od czasu propagacji **broadcast**-a procesu informującego że: jest nowym właścicielem, zwalnia zasoby (kolor zielony) / wtedy powstają skróty

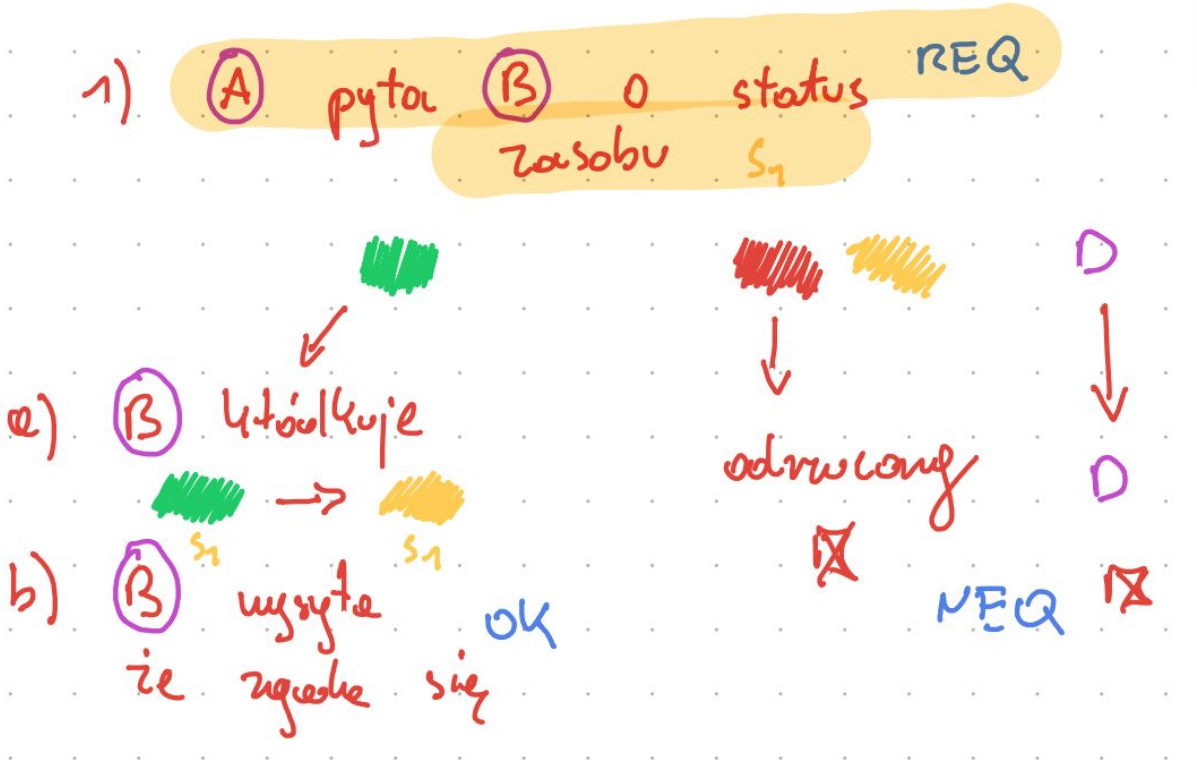
Przykładowy świat emerytow (3 zasoby, 2 pary, 6 osób)

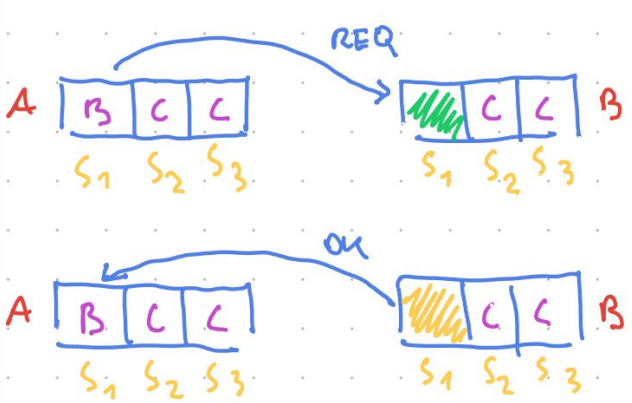
załóżmy że process **A** zamierza pozyskać zasob **S1**, patrzy na swój w miarę aktualny **lookup vector**, widzi ze aktualnym właścicielem jest **B**, więc zaczyna z nim konsultowanie przekazania tokenu (inne procesy nie uczestniczą)



[FAZA 1]: pytanie o aktualny status / chęć oddania

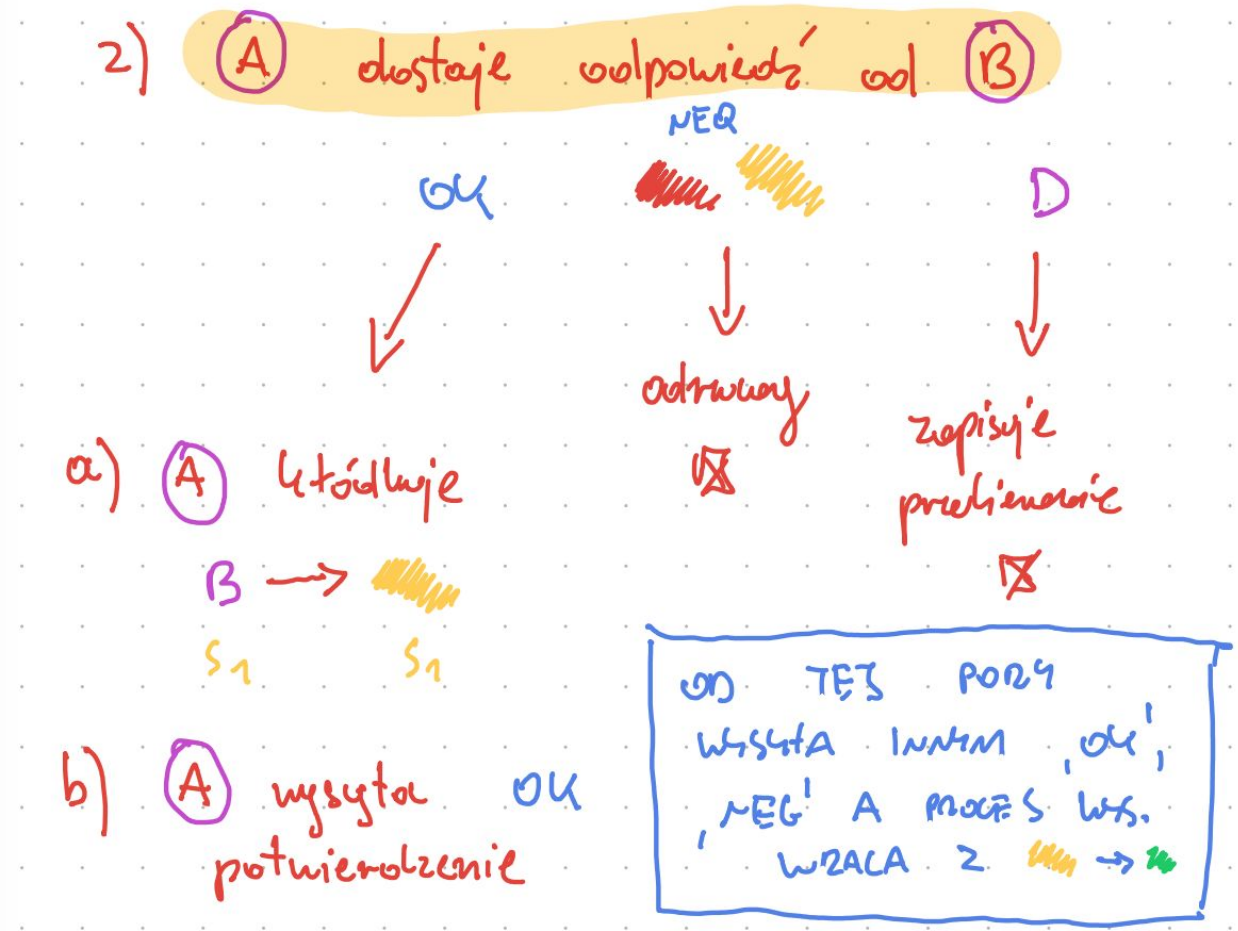
proces A wysyła do B prośbę - wstępnie może już z dodatkowym informacji ocenić czy sie oplaca pytać

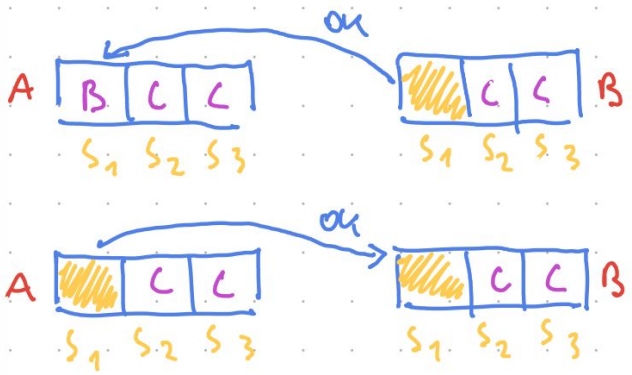




np. gdy w wektorze ma informacje że S1 należy do B; a z różnicy czasu widać że najprawdopodobniej ten zasób jest już wolny (czerwony na zielony) ale brak potwierdzenia; lub doszedł broadcast który o tym informuje (zapisujemy na później takie informacje)

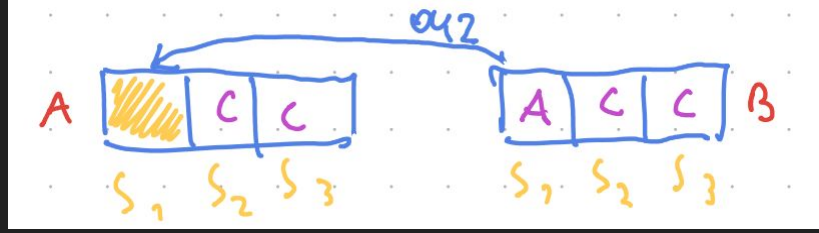
[FAZA 2]: zgoda na przekazanie & potwierdzenie



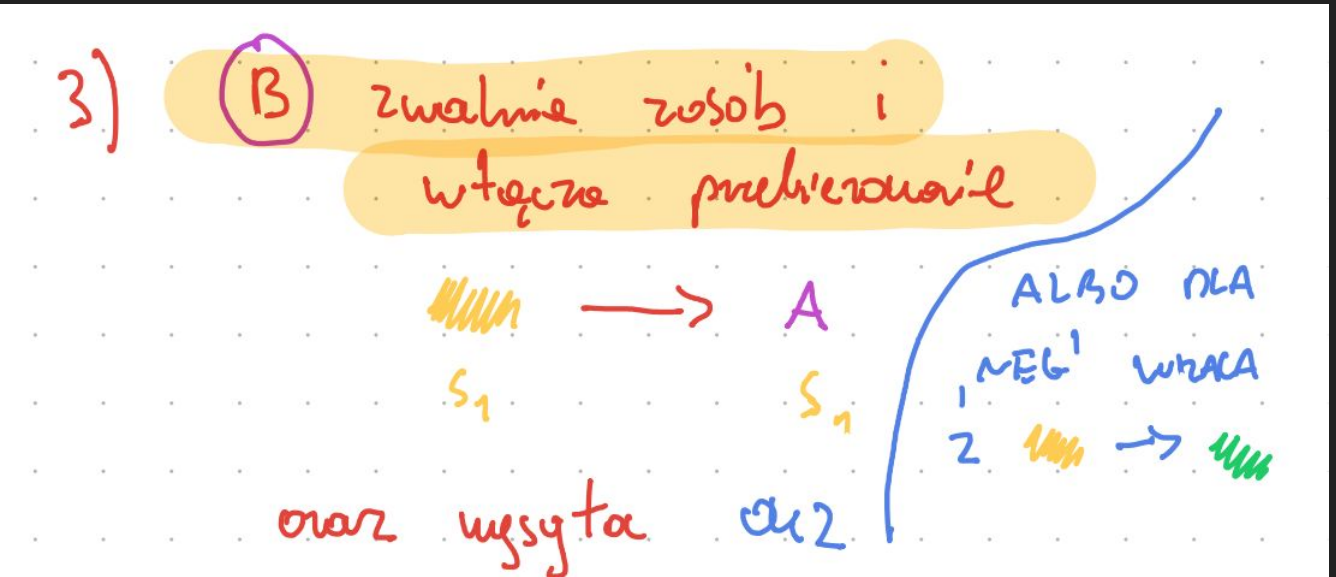


proces A może pytać o wszystkie wolne np. maści, więc może do niego spływać wiele odpowiedzi - dlatego zmienia swój status na żółty, i kontynuuje komunikacje z pierwszym zaakceptowanym - a wszystkim innym odsyła NEQ

[FAZA 3]: przekierowanie / lub rezygnacja

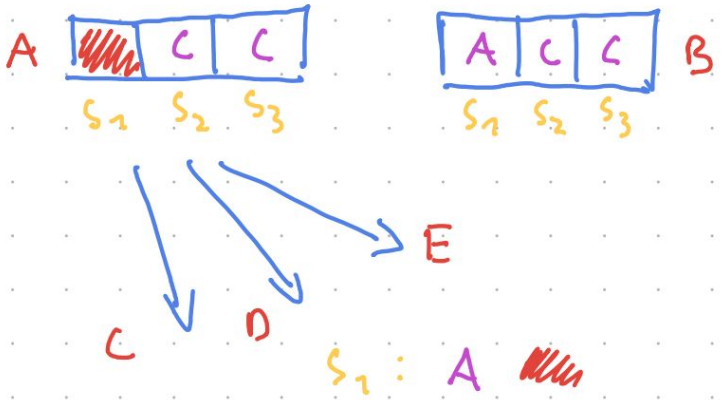


faza potrzebna jest gdy inny proces (nie A), probuje uzyskac dostep do S1, a jeszcze nie wiemy czy przejmie (no i w razie awarii procesu)

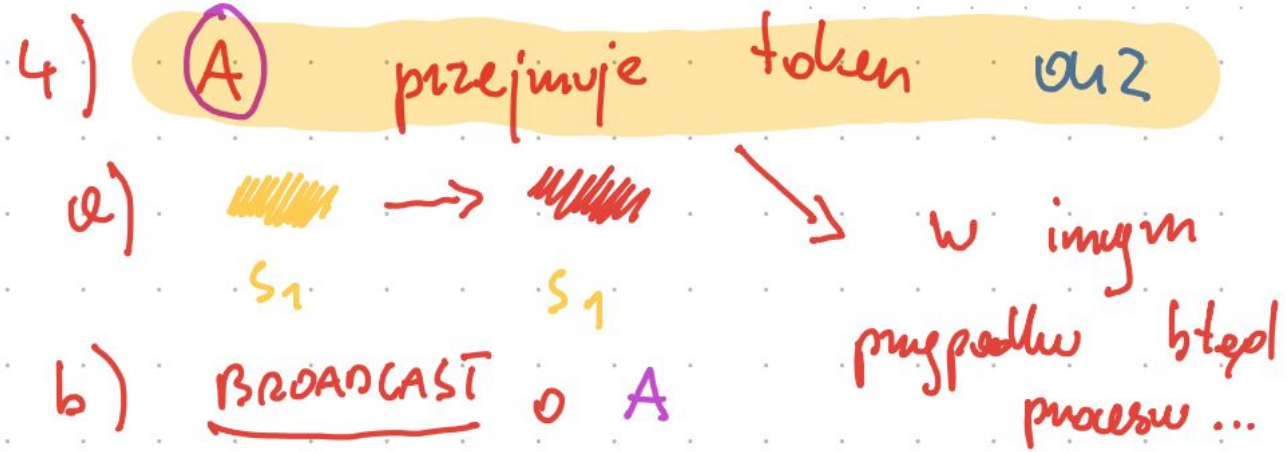


[FAZA 4]: nowy właściciel chwali się całej sieci że jest nowym właścicielem i ze zajal zasob na najprawdobniej czas T

[1-raz] broadcast gdy jest nowym właścicielem (kolor czerwony)



BONUS: broadcast może zawierać dodatkowe informacje (spadkowy czas zegara)



[2-raz] broadcast gdy zwalnia zasoby oraz jest właścicielem (kolor zielony)

PART 2: struktura/ramki

Struktury wspólne dla procesu mężczyzny i kobiety:

1. słownik <S_id, P_id> : przechowujący informacje o tym który proces P zarządza salą S.
2. słownik <G_id, P_id> : przechowujący informacje o tym który proces P zarządza magnetofonem G.
3. słownik <M_id, P_id> : przechowujący informacje o tym który proces P zarządza maścią M.

Struktury w procesie mężczyzny

1. id procesu kobiety K_id: numer identyfikujący proces kobiety z którą mężczyzna jest w parze lub wartość pusta jeśli nie ma pary.
2. lista S: zarządzanych przez proces sal
3. lista G: zarządzanych przez proces magnetofonów
4. lista M: zarządzanych przez proces maści

Stany mężczyzny: FREE

1. mężczyzna w stanie wolnym, tzn. aktywnie poszukujący partnerki do tańca oraz zasobów.
2. Wiadomości możliwe do wysłania, wszystkie zapytania zawierają id procesu który wysłał wiadomość.
   a. REQ_K: zapytanie o wolną kobietę wysyłane do losowego innego procesu
   b. REQ_S: zapytanie o salę z wylosowanym id, wysyłane do procesu znalezionego w słowniku <S_id, P_id> lub losowego innego procesu, jeśli nie znaleziono odpowiedniego.
   c. REQ_G: zapytanie o magnetofon z wylosowanym id, wysyłane do procesu znalezionego w słowniku <G_id, P_id> lub losowego innego procesu, jeśli nie znaleziono odpowiedniego.
   d. REQ_M: zapytanie o maść z wylosowanym id, wysyłane do procesu znalezionego w słowniku <M_id, P_id> lub losowego innego procesu, jeśli nie znaleziono odpowiedniego.
   e. RESP_EMPTY_S, RESP_AV_G, RESP_AV_M, GIVE_ME_S, GIVE_ME_G, GIVE_ME_M, YOUR_S, YOUR_G, YOUR_M, MY_S, MY_G, MY_M, DANCE_WITH_ME, LETS_DANCE, TOO_LATE - opisane na drugiej stronie
   f. LETS_DANCE: wysyłany do procesu K_id, jeżeli zdobyte są wszystkie potrzebne zasoby

Stany mężczyzny: FREE

3. Reakcje na otrzymane wiadomości, na przykładzie sal S, jednak dla magnetofonów i maści dokładnie ta sama zasada:
   a. REQ_S:
      - jeżeli zarządza daną salą to sprawdza czy jest wolna:
        1. jeżeli jest wolna to wysyła RESP_EMPTY_S - odpowiedź o wolnej sali do procesu podanego w zapytaniu
        2. jeżeli jest zajęta to ignoruje wiadomość
      - jeżeli nie zarządza daną salą to przekazuje informację do procesu znalezionego w słowniku <S_id, P_id> lub losowego innego procesu, jeżeli nie znalazł tej informacji.
   b. RESP_EMPTY_S: jeżeli nie ma przypisanej sali to wysyła wiadomość GIVE_ME_S do procesu od którego otrzymał tą wiadomość, w przeciwnym przypadku ignoruje wiadomość
   c. GIVE_ME_S: jeżeli zarządza salą i jest ona wolna to zaznacza salę jako zajętą oraz odsyła YOUR_S, zawierająca salę S_id.
   d. YOUR_S: dodaje salę do listy S, jeśli nie ma sali do tańca to zajmuje tę którą otrzymał, następnie wysyła do wszystkich procesów wiadomość MY_S, zawierającą id sali, jeśli ma już wszystkie zasoby to wysyła też LETS_DANCE do procesu K_id
   e. MY_S: jeżeli miał to usuwa salę z listy swoich sal S, następnie nadpisuje informację o procesie który zarządza salą w słowniku <S_id, P_id>.

Stany mężczyzny: FREE

3. Reakcje na otrzymane wiadomości cd.
    a. REQ_K: przekazuje wiadomość do losowego innego procesu.
    b. RESP_K: jeżeli nie ma pary to odsyła wiadomość DANCE_WITH_ME do procesu kobiety który wysłał wiadomość.
    c. READY_TO_DANCE: jeżeli nie ma pary, to zapisuje id procesu który wysłał wiadomość i odsyła READY_TO_DANCE, jeśli ma już wszystkie zasoby to wysyła też LETS_DANCE, w przeciwnym przypadku odsyła TOO_LATE
4. Przejście do innego stanu:
    a. Po wysłaniu wiadomości LETS_DANCE przechodzi w stan DANCING

Stany mężczyzny: DANCING

1. Mężczyzna w stanie tanecznym, tzn. aktywnie tańczący.
2. Wiadomości wysyłane przez proces, sytuacje użycia opisane na kolejnym slajdzie:
    a. RESP_EMPTY_S
    b. YOUR_S
    c. MY_S
    d. REQ_K
    e. TOO_LATE
    f. OK_REST

Stany mężczyzny: DANCING

3. Wiadomości odbierane przez proces:
    a. REQ_S:
        i. jeżeli zarządza daną salą to sprawdza czy jest wolna:
            1. jeżeli jest wolna to wysyła RESP_EMPTY_S - odpowiedź o wolnej sali do procesu podanego w zapytaniu
            2. jeżeli jest zajęta to ignoruje wiadomość
        ii. jeżeli nie zarządza daną salą to przekazuje informację do procesu znalezionego w słowniku <S_id, P_id> lub losowego innego procesu, jeżeli nie znalazł tej informacji.
    b. RESP_EMPTY_S: ignoruje
    c. GIVE_ME_S: jeżeli zarządza salą i jest ona wolna to zaznacza salę jako zajętą oraz odsyła YOUR_S, zawierająca salę S_id.
    d. YOUR_S: dodaje salę do listy S, oraz wysyła do wszystkich procesów wiadomość MY_S, zawierającą id sali
    e. MY_S: jeżeli miał to usuwa salę z listy swoich sal S, następnie nadpisuje informację o procesie który zarządza salą w słowniku <S_id, P_id>.
    f. REQ_K: przekazuje wiadomość do losowego innego procesu.
    g. RESP_K: ignoruje
    h. READY_TO_DANCE: odsyła TOO_LATE
    i. I_NEED_REST - odsyła OK_REST

Stany mężczyzny: DANCING

4. Przejście do następnego stanu:
    a. Po otrzymaniu wiadomości I_NEED_REST od procesu K_id, mężczyzna odsyła wiadomość OK_REST i przechodzi do stanu RESTING.

Stany mężczyzny: RESTING

1. Mężczyzna w fazie odpoczynku, tzn. ignorujący wszystko.
2. Wiadomości wysyłane przez proces: brak
3. Wiadomości odbierane przez proces: wszystkie możliwe są ignorowane
4. Przejście do następnego stanu:
    a. Po upływie losowego czasu odpoczynku, proces przechodzi w stan FREE

Struktury w procesie kobiety

1. id procesu mężczyzny M_id: numer identyfikujący proces mężczyzny z którym kobieta jest w parze lub wartość pusta jeśli nie ma pary.

Stany kobiety: FREE

1. Kobieta wolna, tzn. czekająca na zaproszenie do tańca
2. Wiadomości wysyłane przez proces:
    a. RESP_K
    b. REQ_S, REQ_G, REQ_M
    c. READY_TO_DANCE
3. Wiadomości odbierane przez proces:
    a. REQ_K: jeżeli M_id jest puste to odsyła RESP_K na id procesu z zapytania
    b. REQ_S, REQ_G, REQ_M: przesyła dalej jeśli ma id procesu ze słownika lub ignoruje.
    c. MY_S, MY_G, MY_M: aktualizuje odpowiedni słownik
    d. DANCE_WITH_ME: Blokuje zmienną M_id i odsyła wiadomość READY_TO_DANCE
    e. TOO_LATE: Odblokowuje zmienną M_id
    f. LETS_DANCE: zapisuje otrzymane id w zmiennej M_id
4. Przejście do kolejnego stanu:
    a. Po otrzymaniu wiadomości LETS_DANCE przechodzi w stan DANCING

Stany kobiety: DANCING

1. Kobieta tańcząca
2. Wiadomości wysyłane przez proces:
   a. REQ_S, REQ_G, REQ_M
   b. I_NEED_REST
3. Wiadomości odbierane przez proces:
   a. OK_REST - przechodzi w stan RESTING
   b. resztę ignoruje
4. Przejście do innego stanu
   a. Po upływie pewnego czasu wysyła wiadomość I_NEED_REST do procesu M_id, gdy otrzyma w odpowiedzi OK_REST to przechodzi w stan RESTING i usuwa M_id.

Stany kobiety: RESTING

1. Kobietaw fazie odpoczynku, tzn. ignorująca wszystko.
2. Wiadomości wysyłane przez proces: brak
3. Wiadomości odbierane przez proces: wszystkie możliwe są ignorowane
4. Przejście do następnego stanu:
    a. Po upływie losowego czasu odpoczynku, proces przechodzi w stan FREE

Struktury w strukturach Sala, Magnetofon, Maść

1. liczba ID: numer identyfikacyjny odpowiedniej struktury
2. wartość logiczna used: reprezentuje stan danej struktury, czy jest aktualnie używana (przy czym używana może być tylko przez proces który nią zarządza)