Nanozombie - opis algorytmu

Łukasz Duhr, 136700
Mateusz Ksok, 136751

1. **Struktury:**
    a. Proces (turysta) - zawiera wszelkie informacje potrzebne do działania programu, np.
        1) Stan obecnego procesu
        2) ID obecnego procesu
        3) Listę ID wszystkich procesów
        4) Liczbę dostępnych strojów
        5) Stan posiadania stroju (posiada / nie posiada)
        6) Listę łodzi
        7) Wstrzymane żądania
        8) Stan zegara Lamporta

    b. Łódź - zawiera:
        1) numer na liście
        2) pojemność (losowo wybierane n)
        3) stan łodzi (oczekuje / wypłynęła)
        4) listę pasażerów

    c. Request - struktura opisująca wysłaną/odebraną wiadomość
        1) typ i zawartość wiadomości
        2) numer id obiektu wysyłającego

2. Typy wiadomości:

a. LAUNCH - wysyłana jedynie przez wątek główny, służy do uruchomienia wątków i inicjalizacji danych
b. SREQ/BREQ - wiadomość wysyłana przez proces aktualnie ubiegający się o strój / miejsce w łodzi.
c. SACK - wiadomość wysyłana jako odpowiedź przez proces akceptujący żądanie innego procesu o strój (aktualnie nie korzysta z żadnego)
d. BACK - wiadomość wysyłana jako odpowiedź na żądanie dostępu do łodzi - na danych uzyskanych z tych odpowiedzi proces chcący wejść do łodzi aktualizuje swoją lokalną tablicę łodzi, aby zadecydować o wejściu na którąś z nich.
e. CRUISE - wiadomość wysyłana przez proces, który nie mógł się zmieścić na konkretnej łodzi będącej jeszcze w porcie - wówczas wyrusza ona w rejs.
f. CRUISE_END - wiadomość wysyłana przez proces zarządzający rejsem (jednego, arbitralnie wybranego pasażera) oznaczająca zakończenie rejsu.

3. Stany procesów:

Każdy ze stanów (poza INIT) reaguje na wiadomości CRUISE i CRUISE_END oznaczeniem łodzi w swojej lokalnej kopii danych o ID podanym w wiadomości jako wypłyniętej w rejs, bądź będącej w porcie.

1. INIT - każdy proces zaczyna pracę w tym stanie. Dane są inicjalizowane w tym miejscu.
   Reakcje na wiadomości:

    a. **LAUNCH - po pierwszym otrzymaniu proces inicjalizuje struktury danych i przechodzi w stan PENDING.**
    b. Pozostałe wiadomości w tym stanie są ignorowane, gdyż nie ma możliwości wysłania wiadomości typu innego niż LAUNCH - każdy proces znajduje się w stanie INIT.

2. PENDING - w tym stanie proces oczekuje losowy kwant czasu na “zgłoszenie się turysty” - po czym przechodzi do stanu SUIT_CRITICAL
   Reakcje na wiadomości:
    a. SREQ - odpowie SACK
    b. BREQ - odpowie BACK z wartościami -1, -1 (nie korzysta z żadnej łodzi)
    c. Pozostałe sygnały są ignorowane.

3. SUIT_CRITICAL - w tym stanie proces ubiega się o otrzymanie stroju.
   Reakcje na wiadomości:
    a. SACK - jeśli wysyłający proces nie chce stroju bądź chce strój, ale ma wyższą wartość sygnału Lamporta (niższy priorytet - sygnalizowane odpowiednią wartością) - dodaj 1 do pomyślnie otrzymanych odpowiedzi. Po uzbieraniu ilości pomyślnych odpowiedzi równej przynajmniej [liczba procesów - liczba strojów] przejdź do stanu BOAT_CRITICAL.
    b. SREQ - odpowiedz SACK jeśli proces wysyłający ma wyższy priorytet, w przeciwnym razie dodaj do listy wstrzymanych żądań (odpowiedź zostanie wysłana po zwolnienu zasobu)
    c. BREQ - odpowiedz BACK z wartościami -1, -1 (nie korzysta z żadnej łodzi)
    d. Sygnał REJECT usunięto
    e. Pozostałe sygnały są ignorowane.

4. BOAT_CRITICAL - w tym stanie proces próbuje otrzymać miejsce w łodzi.
   Reakcje na wiadomości:
    a. SREQ - wstrzymaj żądanie do momentu zwolnienia stroju
    b. BREQ - wstrzymaj żądanie jeżeli proces wysyłający ma niższy priorytet, w przeciwnym wypadku - dodaj do tablicy
    c. BACK - zaktualizuj tablicę stanu łodzi na podstawie otrzymanych danych.
    d. Pozostałe sygnały są ignorowane.

5. ON_BOAT - w tym stanie proces otrzymał dostęp do obu zasobów krytycznych i jest gotów wypłynąć w rejs.
   Reakcje na wiadomości:
    a. SREQ - wstrzymaj żądanie do momentu zwolnienia stroju
    b. BREQ - wyślij BACK z wartościami równymi id łodzi i ilością zajętego miejsca przez siebie
    c. CRUISE_END - dodatkowo, poza zmianą stanu odpowiedniej łodzi, proces zmienia stan na PENDING. Po otrzymaniu tego sygnału cykl się zapętla.
    d. Pozostałe sygnały są ignorowane.

4. Algorytm:

1) Inicjalizacja:

otrzymanie pierwszego sygnału *LAUNCH*

a. ustawienie liczby turystów, strojów i łodzi
b. przypisanie łodzi do przeznaczonej do tego kolejki

otrzymanie drugiego sygnału *LAUNCH* - przejście do stanu PENDING

2) Odczekanie przez proces losowo wybranego kwantu czasu, po czym proces przechodzi do stanu SUIT_CRITICAL, w którym czeka na strój

3) Proces wysyła wiadomość SREQ do wszystkich procesów o żądanie stroju.

4) Proces dostaje wiadomość SACK - łączna liczba wiadomości powinna wynieść minimum [liczba procesów - liczba strojów]:

a. OK, jeżeli proces nie potrzebuje stroju lub ma mniejszy priorytet (większy licznik zegara Lamporta)
b. Nie otrzymuje wiadomości, gdy proces używa stroju - żądanie jest wstrzymywane do momentu zwolnienia zasobu.

Jeśli proces dostanie strój, przechodzi do stanu BOAT_CRITICAL.

5) Wysyła wiadomość BREQ o żądanie miejsca w łodzi.

6) Dostaje wiadomość od pozostałych procesów:

a. Jeśli pytany proces już zajął miejsce w łodzi, wysyła BACK z ID łodzi i ilością miejsca, które zajął
b. Jeśli pytany proces nie zajął łodzi, bądź ma mniejszy priorytet, wysyła BACK z -1, -1.
c. **Proces czeka na wiadomość od pozostałych procesów. Jeśli wszystkie odpowiedzi, które otrzymał wskazują na to, że każdy z pozostałych procesów jest na łodzi, to automatycznie uruchamia rejs dla każdej z łodzi niezależnie od tego, czy uda mu się na którąś z łodzi wejść, czy nie.**

7) Przeiteruj przez wszystkie łodzie i na podstawie zebranych danych zajmij miejsce w pierwszej łodzi mającej wystarczająco miejsca, która nie wypłynęła (dodaj do listy id turysty).

a. Jeżeli dana łódź na liście ma mniej miejsca, niż pasażer potrzebuje, wyślij wszystkim sygnał *CRUISE* o wypłynięciu danej łodzi, a samemu zajmij miejsce w następnej. Jeśli się nie uda - czekaj na molo.
b. Przy zajęciu miejsca w łodzi proces zmienia status na ON_BOAT.
c. W momencie przejścia do statusu ON_BOAT, proces wysyła odpowiedzi na wszystkie wstrzymane żądania typu BREQ.
d. **W razie niepowodzenia w dostępie do łodzi proces czeka na wiadomość typu CRUISE_END od dowolnego procesu (które oznacza dobicie łodzi do portu i zwolnienie sekcji krytycznych). Wówczas proces ponownie wysyła żądanie dostępu do łodzi.**

8) Rejs - jeden z pasażerów (np. ten o najmniejszym ID na liście pasażerów) losuje czas, przez który łódź jest na rejsie, po czym odsyła (i sam też sobie zmienia) wszystkim wiadomość *CRUISE_END*.

9) Zwolnienie łodzi:

a. Kiedy wyprawa dobiegnie końca, wysłana do wszystkich zostaje informacja *CRUISE_END* o powrocie łodzi oraz zwolnieniu jej.
b. Dana łódź zostaje oznaczona jako będąca w porcie.

10) Turysta po zakończeniu rejsu zwalnia strój i wraca do początku algorytmu (wraca do stanu PENDING).