Maciej Leszczyk 136759, L6 maciej.leszczyk@student.put.poznan.pl
Kamil Kowalczyk 136742, L6 kamil.m.kowalczyk@student.put.poznan.pl

Sprawozdanie

Wskrzeszanie Smoków

Przetwarzanie Rozproszone - laboratorium

- **Opis problemu**

Ubić smoka może byle idiota. Za to wskrzesić... do tego potrzeba profesjonalisty. Są dwa rodzaje procesów: jeden generuje co pewien czas zlecenie. O zlecenie ubiegają się profesjonaliści o jednej z trzech możliwych specjalizacji (głowa, ogon, tułów). Do realizacji zlecenia potrzeba trzech profesjonalistów o różnych specjalizacjach - należy zapewnić, by nie doszło do zakleszczeń! Dodatkowo, profesjonaliści muszą wypełnić robotę papierkową (robi to jeden z nich) przy jednym z *b* biurek w gildii Wskrzeszania Smoków. Następnie profesjonaliści zdobywają dostęp do jednego z *s* szkieletów smoków i rozpoczynają wskrzeszanie. Należy zapewnić, by profesjonaliści dzielili się w miarę równo pracą.

Wersja uproszczona: : Brak specjalizacji.

W podanym problemie zostały użyte następujące założenia:

- **n + 1** procesów (gdzie 1 proces jest procesem zlecającym zlecenia, a **n** procesów pełni rolę specjalistów do wskrzeszania smoków). Liczba procesów jest sparametryzowana i stała w obrębie całego programu
- **b** biurek
- **s** szkieletów
- Kanały są FIFO oraz niezawodne. Procesy nie ulegają awarii

- **Struktury danych**

  - **Specjalista**
    - własne id
    - lista zleceń (id zleceń)
    - liczba wykonanych zleceń
    - zegar Lamporta
    - lista wszystkich specjalistów (id specjalisty, specjalizacja i liczba wykonanych zleceń)
    - lista specjalistów od ogonów będących w stanie **READY**
    - lista specjalistów od tułowi będących w stanie **READY**
    - lista specjalistów od głów będących w stanie **READY**
    - lista specjalistów (id) z którymi wykonuje się zlecenie
    - licznik odpowiedzi *ACK*, na początku 0

        - lista id procesów oczekujących na *ACK* (używana w momencie gdy proces zwalnia sekcję krytyczną)

- # Stany

    - **READY** - stan w którym znajduje się proces w momencie gdy jest gotowy przyjąć nowe zlecenie
    - **TEAM** - stan, w którym procesy mają już swoją własną drużynę. Każdy proces zna ID procesów współpracowników
    - **PREPARING-PAPERWORK** - stan w którym znajduje się specjalista, który spośród trójki procesów w swojej drużynie został wybrany, by zaczynać ubiegać się o wstęp do gildii magów
    - **PAPERWORK** - stan w którym proces rozpoczął pracę papierkową
    - **PREPARING-SKELETON** - stan w którym znajduje się specjalista, który wypełnił robotę papierkową i zaczyna ubiegać się o jeden ze szkieletów dla swojej drużyny
    - **SKELETON-GUARDING** - stan w którym znajduje się prcoes o najniższym id w drużynie, gdy dostał się on do sekcji krytycznej szkieletów. Będzie on w tym stanie bronić miejsca w sekcji krytycznej.
    - **JOINING** - stan po ukończeniu pracy w którym proces chce dołączyć do grupy procesów ze statusami **READY**

- # Komunikaty

    - ***BROADCAST*** - komunikat powiązany ze zleceniem rozsyłanym przez proces zlecający. Zawiera ID zlecenia.
    - ***INFO*** - komunikat wysyłany przez wszystkie procesy w momencie uruchomienia programu. Zawiera w sobie:
        - id procesu
        - specjalizację procesu
        - liczba wykonanych dotychczas zleceń (jako wartość priorytetu)
    - ***REQ-JOIN*** - komunikat wysyłany przez proces specjalistę, który zakończył pracę i ubiega się o dołączenie do grupy procesów w stanie **READY**
    - ***ACK-JOIN*** - komunikat wysyłany w odpowiedzi na wiadomość *REQ-JOIN*, w momencie w którym proces nie jest w stanie **JOINING**. Zawiera id procesu wysyłającego *REQ-JOIN*
    - ***REQ-PAPERWORK*** - komunikat wysyłany przez proces specjalisty pytającego o możliwość wstępu do gildii
    - ***REQ-SKELETON*** - komunikat wysyłany przez proces specjalisty pytającego o możliwość zajęcia jednego ze szkieletów
    - ***ACK-PAPERWORK*** - komunikat wysyłany jako odpowiedź na komunikat *REQ-PAPERWORK* w przypadku zgody na wstęp do gildii magów
    - ***ACK-SKELETON*** - komunikat wysyłany jako odpowiedź na komunikat *REQ-SKELETON* w przypadku zgody na zajęcie jednego ze szkieletów
    - ***START-WORK*** - komunikat wysyłany przez proces w stanie **SKELETON-GUARDING** w celu poinformowanie procesu z drużyny o tym, że może rozpocząć pracę.

- **Opis reakcji na wiadomości w konkretnych stanach**

  - **WSZYSTKIE STANY**
    - *BROADCAST*
      - Proces tworzy nową instancję zlecenia w pamięci lokalnej.
    - *REQ-JOIN*
      - Wysyłany jest w odpowiedzi komunikat *ACK-JOIN* . W tym momencie proces, który odsyła zgodę, może założyć że prędzej czy później proces proszący o pozwolenie dostanie się do grupy procesów w stanie **READY**, więc dopisuje go do swojej lokalnej listy gotowych procesów.
    - *REQ-SKELETON / REQ-PAPERWORK*
      - Odpowiada ACK.
    - *ACK-SKELETON / ACK-PAPERWORK / ACK-JOIN*
      - Ignoruje
  - **TEAM**
    - *REQ-JOIN*
      - Gdy proces jest specjalistą o „środkowym” id w swojej drużynie i otrzyma wiadomość *REQ-JOIN* od procesu, który posiada najwyższe id w drużynie, proces wnioskuje wtedy, że jego kolega zakończył pracę, więc odsyła mu *ACK-JOIN* i rozpoczyna pracę. W przeciwnym przypadku odsyłane jest samo *ACK-JOIN*.
  - **PREPARING-PAPERWORK**
    - *REQ-PAPERWORK*
      - jeżeli priorytet procesu odbiorcy jest niższy od priorytetu procesu nadawcy, wysyła *ACK-PAPERWORK.* W przeciwnym przypadku proces nie wysyła *ACK-PAPERWORK* i zapisuje id procesu nadawcy w lokalnej liście procesów oczekujących na zgodę
    - *ACK-PAPERWORK*
      - zwiększamy licznik otrzymanych wiadomości *ACK*
  - **PAPERWORK**
    - *REQ-PAPERWORK*
      - proces nie wysyła *ACK-PAPERWORK* i zapisuje id procesu nadawcy w lokalnej liście procesów oczekujących na zgodę
  - **PREPARING-SKELETON**
    - *REQ- SKELETON*
      - Jeżeli priorytet procesu odbiorcy jest niższy od priorytetu procesu nadawcy, wysyła *ACK-SKELETON*. W przeciwnym przypadku proces nie wysyła *ACK-SKELETON* i zapisuje id procesu nadawcy w lokalnej liście procesów oczekujących na zgodę
    - *ACK-SKELETON*
      - zwiększamy licznik otrzymanych wiadomości *ACK*
  - **SKELETON-GUARDING**
    - *REQ-SKELETON*
      - proces nie wysyła *ACK-SKELETON* i zapisuje id procesu nadawcy w lokalnej liście procesów oczekujących na zgodę
  - **JOINING**
    - *REQ-JOIN*
      - Gdy proces odbiorcy ma niższy priorytet odsyłane jest *ACK-JOIN*. W przypadku w którym odbiorca ma wyższy priorytet nie wysyła akceptacji, a id procesu proszącego o zgodę zapisuje w

lokalnej liście, by po przejściu do grupy gotowych procesów odesłać zgodę – tak jak w standardowym algorytmie Lamporta.

- *ACK-JOIN*
  - Inkrementowana jest wartość licznika ACK.

Algorytm (przydział zleceń)

- Wszystkie procesy są w stanie **READY**
- Na samym początku rozsyłają między sobą wiadomości *INFO*, informujące o tym jaką posiadają specjalizację. Wiadomość ta wysyłana jest tylko raz na jedno uruchomienie programu. Następnie liczba wykonanych zleceń, oraz zegar Lamporta każdego z procesów są ustawiane na 0.
- Zegar Lamporta stanowi wartość priotytetu przy roztrzyganiu konfliktów żądań o sekcje krytyczne w programie
- Każdy proces po wysłaniu do wszystkich procesów wiadomość *INFO* następnie odczytuje te same wiadomości od każdego procesu, dodając każdy proces do lokalnej listy wszystkich procesów, oraz do lokalnej listy procesów w stanie **READY** dla konkretnej specjalizacji
- Po odczytaniu wszystkich wiadomości, proces ma komplet informacji o wszystkich procesach i jest w stanie w przyszłości wywnioskować czy dostanie dostęp do któregoś ze zleceń (jeżeli takowe istnieją), oraz które zlecenia zostaną zajęte przez inne procesy.
- Z biegiem czasu będą pojawiały się nowe zlecenia, które (jeżeli nie będzie deficytu którejś ze specjalizacji) będą automatycznie przydzielane do procesów posiadających najwyższy priorytet w swojej specjalizacji. Priorytetem jest tu wartość zegara Lamporta, a w przypadku remisu wygrywa mniejsze id. Wszystkie procesy mają świadomość kto otrzymał przydział po pojawieniu się zlecenia, ze względu na dane które przechowują lokalnie. Procesy które otrzymały przydział do zlecenia usuwane są z list procesów w stanie **READY**. Zlecenie usuwane jest z lokalnych list zleceń.
- W pewnym momencie będzie musiało dojść do sytuacji, że z którejś, przynajmniej jednej specjalizacji zabraknie specjalisty do przyjęcia zlecenia. W takim przypadku zlecenie nie jest przyjmowane, a procesy w stanie **READY** czekają.
- Proces, który ukończył pracę, zmienia stan na **JOINING**.
- Proces wysyła komunikat *REQ-JOIN* do wszystkich procesów. Inkrementuje następnie wartość własnego licznika ACK z 0 na 1 i oczekuje na zgody od innych procesów by móc dołączyć do procesów ze stanem **READY**.
- W przypadku gdy otrzyma wiadomości *ACK-JOIN* od wszystkich procesów, oznacza to, że może przejść w stan **READY**, o czym wiedzą wszystkie inne procesy.
- W przypadku, gdy proces nie otrzymał od wszystkich zgody, oznacza to, że inny proces, o wyższym priorytecie także ubiega się o dołączenie do procesów w stanie **READY**. Otrzyma on zgodę dopiero, gdy tamty przejdzie do grupy procesów gotowych i odeśle mu wiadomność *ACK-JOIN*.
- Po przejściu pula procesów w stanie **READY** zwiększa się i (jeżeli to możliwe) automatycznie przydzielane są zaległe zlecenia.

Algorytm (gildia i smoki)

- Od razu po stworzeniu pełnej drużyny specjalista o najmniejszym identyfikatorze udaje się do gildii, by wykonać pracę papierkową. Przechodzi wtedy w stan **PREPARING-PAPERWORK**. Pozostałe dwa procesy przechodzą w stan **TEAM**.

- Zanim specjalista dostanie się do gildii, musi otrzymać zezwolenie wstępu do tego obiektu i zasiądnięcia przy jednym z B biurek.
- W celu dostania się do sekcji krytycznej proces:
  - ustawia wartość licznika wiadomości *ACK* na 0
  - wysyła do każdego procesu specjalisty (nie licząc siebie i tych procesów z którymi jest w drużynie) zapytanie *REQ-PAPERWORK*, które zawiera znacznik czasowy zegara Lamporta.
- Proces dostaje się do sekcji krytycznej, jeżeli liczba otrzymanych wiadomości *ACK-PAPERWORK* zrówna się z wartością progu minimalnej ilości zgód

POJĘCIE:
Minimalną ilość zgód MIN_ACK (wiadomości ACK), jakie musi otrzymać proces, by móc wejść do danej sekcji krytycznej można wyznaczyć ze wzoru:

$$MIN\_ACK = p - size - 2$$

gdzie:
p - liczba procesów specjalistów
size - rozmiar danej sekcji krytycznej

Liczba *2* pojawiła się tu ze względu na to, że wybrany proces specjalista z drużyny 3-osobowej nie musi wysyłać wiadomości o REQ do swoich

- Dostajac się do gildii proces przehodzi w stan **PAPERWORK**.
- Po skończeniu roboty papierkowej i wyjściu z gildii, specjalista rozsyła do wszystkich procesów oczekujących na zgodę wiadomość *ACK-PAPERWORK*, po czym przechodzi w stan **PREPARING-SKELETON**.
- Dostęp do sekcji przebiega tak samo, analogicznie jak w przypadku domagania się wstępu do gildii. Szkielety także są sekcją krytyczną o rozmiarze S. Jedyną różnicą jest typ zapytania, który w tym przypadku jest *REQ-SKELETON*, oraz typ odpowiedzi, którym jest *ACK-SKELETON***.**
- Proces, który ubiegał się o szkielet smoka, w momencie dostania się do sekcji przechodzi w stan **SKELETON-GUARD**. Od tej chwili na żadne zapytanie o wstęp do sekcji krytycznej szkieletów nie będzie odpowiadać zgodą, a identyfikatory procesów nadawców takich wiadomości będzie zapisywać w lokalnej liście procesów oczekujacych na zgodę.
- Następnie ten sam proces wysyła wiadomość *START-WORK* do procesu w drużynie o najwyższym id, by powiadomić go o tym, że może zaczynać pracę.
- Po zakończeniu pracy, proces przechodzi w stan **JOINING** i rozsyła wiadomości *REQ-JOIN* do wszystkich procesów. Dzięki temu proces o środowym id otrzymując taką wiadomość wie już, że jego kolega z najwyższym id skończył pracę i teraz on może zabrać się do roboty. Następnie, po tym gdy zakończy swoją część pracy, także rozsyła wiadomości *REQ-JOIN*, tym samym powiadamiając proces o najniżym id w drużynie, że może on dokończyć pracę wykonując swoją, ostatnią część zadania.
- Po zakończeniu pracy przez proces o najniższym id, wysyła on wpierw *ACK-SKELETON*, do wszystkich oczekujących procesów, a dopiero po tym przechodzi w stan **JOINING**.
- Warto w tym momencie zaznaczyć pewną cechę algorytmu. Całość brudnej roboty wykonuje w każdej drużynie tylko jeden proces – ten o najniższym id (tak naprawdę to mógłby być którykolwiek z trzech procesów). To on ubiega się o sekcje krytyczne i przechodzi tym samym przez różne stany w programie. Jest on także odpowiedzialny za „bronienie” szklieletu smoka, w momencie dostania się do tej sekcji. Z uwagi na to, jego koledzy z drużyny

przechodzą tylko przez 3 stany w całym programie: **READY**, **TEAM** I **JOINING**. W celu ułatwienia zrozumienia niżej rozpisano punktowy przebieg programu od stanu **READY** do stanu **JOINING** dla jednej drużyny złożonej z procesów o identyfikatorach 1, 2 i 3:

- Procesy **1**, **2** i **3** dobierają się w drużynę
- 2 i 3 przechodzą w stan **TEAM**
- 1 przechodzi w stan **PREPARING-PAPERWORK** i zaczyna ubiegać się o wstęp do gildii
- 1 dostaje się do gildii i przechodzi w stan **PAPERWORK**
- 1 kończy pracę w gildii i zaczyna ubiegać się o szkielet smoka, przechodząc w stan **PREPARING-SKELETON**
- 1 dostaje się do sekcji krytycznej, rezerwuje smoka przechodząc w stan **SKELETON-GUARDING**, a następnie wysyła wiadomość do procesu 3 z informacją że może zacząć pracę
- 3 zaczyna pracę
- 3 kończy pracę, rozsyła o tym informację do wszystkich procesów
- 2 zaczyna pracę
- 2 kończy pracę, rozsyła o tym informację do wszystkich procesów
- 1 zaczyna pracę
- 1 kończy pracę

• Złożoności

○ Komunikacyjna

- Wysłanie wiadomości *INFO* do wszysktich procesów: n
- Wysyłanie próśb o dostęp do sekcji krytycznej biurek: n
- Wysyłanie próśb o dostęp do sekcji krytycznej szkieletów: n
- Wysłanie *START-INFO* do kolegi specialisty: 1
- Wysłanie próśb o zmianę stanu na READY: n

SUMA: 4n + 1

○ Czasowa

- Wysłanie wiadomości *INFO* do wszysktich procesów: 1
- Wysyłanie próśb o dostęp do sekcji krytycznej biurek: 1
- Wysyłanie próśb o dostęp do sekcji krytycznej szkieletów: 1
- Wysłanie *START-INFO* do kolegi specialisty: 1
- Wysłanie próśb o zmianę stanu na READY: 1

SUMA: 5