Małe, wesołe pszczółki murarki budzą się wczesną wiosną i zaczynają zbierać nektar. Jest P pszczółek, które konkurują w dostępie do K nierozróżnialnych kwiatków oraz na samym początku działania do jednej z T rozróżnialnych trzcin. Po zebraniu nektaru pszczółka składa jajo w kokonie. Po złożeniu 5 jaj pszczółka umiera i trzcinę może zająć inna pszczółka. Maksymalnie w trzcinie zmieści się 15 kokonów. Początkowe priorytety pszczółek (np zegary lamporta) powinny być losowe.

P pszczółek – każda pszczółka musi na początku zebrać nektar, a potem złożyć jajo w kokonie. Po złożeniu 5 jaj pszczółka umiera.

K nierozróżnialnych kwiatów – z kwiatów zbierany jest nektar, każdy kwiat odpowiada jednej pszczółce.

T rozróżnialnych trzcin – każda trzcina może pomieścić 15 jaj (kokonów)

Struktury i zmienne:

p - początkowa liczba pszczół (opcjonalnie zmniejsza z każdą martwą pszczółką)

k - liczba kwiatów (nierozróżnialny zasób)

t - liczba trzcin (rozróżnialny zasób) zmniejszana o liczbę trzcin, na których znajduje się już 15 kokonów

request - struktura zawierająca dwa pola: pid, timestamp

WaitQueueFlowers - kolejka procesów oczekujących na ACKflower

AckNumFlower - liczba uzyskanych ACKflower od innych pszczół, początkowo 0

WaitQueueReeds - lokalna kolejka żądań dostępu do trzcin posortowana po ich znacznikach czasowych (lamport) oraz pid (zawiera struktury request)

AckNumReed - liczba uzyskanych ACKreed on innych pszczół, początkowo 0

layedEggs - liczba złożonych jaj dlla danej pszczóły, początkowo 0

reeds - tablica int'ów, która na danym indeksie przechowuje liczbę złożonych jaj dla danej trzciny (początkowo wypełniona zerami)

req\_ack\_message - struktura wiadomości zawierająca pid oraz timestamp

release\_message - struktura wiadomości zawierająca następujące pola: pid, FULL/AVAILABLE w zależności od wypełnienia trzicny

Wiadmości:

Wszystkie wiadomości są podbite znacznikiem czasowym (timestampem), modyfikowanym zgodnie z zasadami skalarnego zegara logicznego Lamporta.

a. REQflower - żądanie o dostęp do kwiata (sekcji krytycznej). (req\_ack\_message)

b. ACKflower - potwierdzenie dostępu do kwiata (sekcji krytycznej). (req\_ack\_message)

c. REQreed - żądanie o dostęp do trzciny (sekcji krytycznej). Zawiera priorytet żądania. (req\_ack\_message)

d. ACKreed - potwierdzenie otrzymania REQreed odsyłane do nadawcy tego requesta. (req\_ack\_message)

e. RELEASEreed - wiadomość zawierająca strukturę release\_message, wysyłana przy opuszczaniu trzciny.

Stany:

Początkowym stanem jest REST

a. REST - nie ubiega się o dostęp do żadnego zasobu

b. WAIT\_REED - czeka na dostęp do trzciny (sekcji krytycznej)

c. ON\_REED - w sekcji krytycznej, składa jaja

d. WAIT\_FLOWER - czeka na dostęp do kwiata (sekcji krytycznej) (proces cały czas jest w sekcji krytycznej REED)

e. ON\_FLOWER - w sekcji krytycznej, zbiera nektar (proces cały czas jest w sekcji krytycznej REED)

f. DEAD - pszczółka nie żyje (opcjonalne)

Szkic algorytmu:

Pszczółka i ubiega się o dostęp do jednej z rozróżnialnych trzcin (sekcji krytycznej) poprzez wysłanie REQreed (podbite lokalnym zegarem lamporta) do wszystkich pozostałych pszczółek oraz umieszcza się na kolejce WaitQueueReeds. Pozostałe pszczółki umieszczają zegar lamporta oraz pid z otrzymanego requesta REQreed na kolejce WaitQueueReeds, która jest posortowana względem zegarów lamporta (rosnąco) oraz po pid (rosnąco). Następnie pszczółki odsyłają ACKreed (podbite lokalnym zegarem lamporta). Gdy do pszczółki i dotrze wiadomość ACKreed to zwiększa ona licznik zgód AckNumReed. Gdy pszczółka i otrzyma od wszystkich ACKreed (AckNumReed == p) oraz jej indeks w kolejce WaitQueueReeds < t to wchodzi ona do sekcji krytycznej.

Gdy pszczółka znajduje się w sekcji krytycznej trzcin, przechodzi do ubiegania się o dostęp do kwiatka (sekcji krytycznej) wysyłając do wszystkich pozostałych pszczółek prośbe REQflower. Pozostałe pszczółki odsyłają ACKflower do pszczółki i, o ile same się nie ubiegają o dostęp albo jeżeli priorytet (timestamp) ich żądania REQflower jest mniejszy od priorytetu pszczółki i. W przeciwnym przypadku zapamiętują otrzymany REQflower w kolejce WaitQueueFlowers i odsyłają ACKflower po wyjściu z sekcji krytycznej. Pszczółka wchodzi do sekcji po zebraniu ACKflower od p - k pszczółek.

Szczegółowy opis algorytmu dla procesu i:

1. REST: stan początkowy

1.1. Proces i przebywa w stanie REST do czasu, aż podejmie decyzję o ubieganie się o sekcję krytyczną (trzcinę). Ze stanu REST następuje przejście do stanu WAIT\_REED po umieszczeniu siebie w kolejce WaitQueueReeds, wysłaniu wiadomości REQreed do wszystkich innych procesów oraz ustawieniu ACKreed na zero. Wszystkie wiadomości REQreed są opisane tym samym priorytetem, równym zegarowi Lamporta w chwili wysłania pierwszej wiadomości REQreed.

1.2. Reakcje na wiadomości:

a. REQflower: odsyła ACKflower

b. REQreed: odsyła ACKreed

c. ACKflower: ignoruje (sytuacja niemożliwa)

d. ACKreed: ignoruje (sytuacja niemożliwa)

e. RELEASEreed: proces aktualizuje lokalną kolejkę żądań WaitQueueReeds. Z WaitQueueReeds usuwane jest żądanie procesu, którego pid jest zawarty w wiadomości, a reszta żądań jest przesuwana o jedną pozycję do przodu.

2. WAIT\_REED: ubieganie się o sekcję krytyczną.

2.1 Ze stanu WAIT\_REED następuje przejście do stanu ON\_REED pod warunkiem, że proces otrzyma ACKreed od wszystkich innych procesów oraz jeżeli jego indeks i w kolejce WaitQueueReeds spełnia warunek i < t.

2.2 Reakcje na wiadomości:

a. REQflower: odsyła ACKflower

b. REQreed: Po otrzymaniu REQreed od procesu j-tego, proces i-ty wstawia żądanie do lokalnej kolejki żądań (posortowanej po ich znacznikach czasowych/priorytetach). Nastepnie proces i-ty odsyła do procesu j-tego ACKreed.

c. ACKflower: ignoruje (sytuacja niemożliwa)

d. ACKreed: zwiększa licznik otrzymanych ACKreed (AckNumReed++)

e. RELEASEreed: proces aktualizuje lokalną kolejkę żądań WaitQueueReeds. Z WaitQueueReeds usuwane jest żądanie procesu, którego pid jest zawarty w wiadomości, a reszta żądań jest przesuwana o jedną pozycję do przodu.

3. ON\_REED: przebywanie w sekcji krytycznej.

3.1 Po wejściu do sekcji krytycznej, do pszczółki musi zostać przydzielona trzcina. Przydział jest dokonywany w sposób statyczny w momencie wypełnienia kolejki WaitQueueReeds. Do określenia do której trzicny ma otrzymać dostęp pszczółka używana jest operacja modulo. Przykładowo (dla 2 trzcin) 1 pszczoła w kolejce WaitQueueReeds otrzyma dostęp do pierwszej trzciny jako pierwsza, 2 pszczoła w kolejce otrzyma dostęp do drugiej trzciny jako pierwsza, 3 pszczoła w kolejce otrzyma dostęp do pierwszej trzicny jako druga, 4 pszczoła otrzyma dostęp do drugiej trzicny jako druga.

3.2 Po wybraniu trzciny proces będzie pięciokrotnie przechodził cykl stanów ON\_REED -> WAIT\_FLOWER -> ON\_FLOWER -> ON\_REED

3.3 Przejście do stanu WAIT\_FLOWER następuje w dowolnym momencie, w którym proces się na to zdecyduje. W momencie decyzji do wszystkich procesów wysyłane są wiadomości REQflower.

3.4 Po przejściu ze stanu ON\_FLOWER na ON\_REED proces zwiększa prywatną zmienną layedEggs o 1. W momencie osiągnięcia wartości 5 następnuje przejście do stanu REST lub DEAD (opcjonalne).

3.5 Po podjęciu decyzji o opuszczeniu sekcji lub śmierci (opcjonalne) proces wysyła wiadomość RELEASEreed, która zawiera informację o tym, którą trzcinę zwalnia oraz czy trzcina jest już przepełniona.

3.6 Reakcje na wiadomości:

a. REQflower: odsyła ACKflower

b. REQreed: Po otrzymaniu REQreed od procesu j-tego, proces i-ty wstawia żądanie do lokalnej kolejki żądań (posortowanej po ich znacznikach czasowych/priorytetach). Nastepnie proces i-ty odsyła do procesu j-tego ACKreed.

c. ACKflower: ignoruje (sytuacja niemożliwa)

d. ACKreed: ignoruje (sytuacja niemożliwa)

e. RELEASEreed: proces aktualizuje lokalną kolejkę żądań WaitQueueReeds. Z WaitQueueReeds usuwane jest żądanie procesu, którego pid jest zawarty w wiadomości, a reszta żądań jest przesuwana o jedną pozycję do przodu.

4. WAIT\_FLOWER: ubieganie się o sekcję krytyczną.

4.1 Ze stanu WAIT\_FLOWER następuje przejście do stanu ON\_FLOWER pod warunkiem, że proces otrzyma ACKflower od wszystkich innych procesów pomniejszonych o liczbę kwiatów (AckNumFlower == p - k).

4.2 Reakcje na wiadomości:

a. REQflower: od procesu j: jeżeli priorytet zawarty w REQflower jest wiekszy od priorytetu i (pamiętamy: większe wartości oznaczają mniejszy priorytet), odsyła ACKflower. W przeciwnym wypadku REQflower zapamiętywany jest w kolejce WaitQueueFlowers

b. REQreed: Po otrzymaniu REQreed od procesu j-tego, proces i-ty wstawia żądanie do lokalnej kolejki żądań (posortowanej po ich znacznikach czasowych/priorytetach). Nastepnie proces i-ty odsyła do procesu j-tego ACKreed.

c. ACKflower: zwiększa licznik otrzymanych ACKflower (AckNumFlower++). Tak, jak opisano to wyżej, gdy otrzymano ACKflower od wszystkich pozostałych procesów pomniejszonych o liczbę kwiatów, proces i przechodzi do stanu ON\_FLOWER

d. ACKreed: ignoruje (sytuacja niemożliwa)

e. RELEASEreed: proces aktualizuje lokalną kolejkę żądań WaitQueueReeds. Z WaitQueueReeds usuwane jest żądanie procesu, którego pid jest zawarty w wiadomości, a reszta żądań jest przesuwana o jedną pozycję do przodu.

5. ON\_FLOWER: przebywanie w sekcji krytycznej.

5.1 Proces przebywa w sekcji krytycznej do czasu podjęcia decyzji o jej opuszczeniu. Po podjęciu decyzji o opuszczeniu sekcji, proces wysyła ACKflower w reakcji na wszystkie REQflower znajdujące się w WaitQueueFlowers, a następnie przechodzi do stanu ON\_REED

5.2. Reakcje na wiadomości:

a. REQflower: dodaje żądanie do kolejki WaitQueueFlowers

b. REQreed: Po otrzymaniu REQreed od procesu j-tego, proces i-ty wstawia żądanie do lokalnej kolejki żądań (posortowanej po ich znacznikach czasowych/priorytetach). Nastepnie proces i-ty odsyła do procesu j-tego ACKreed.

c. ACKflower: ignoruje (sytuacja niemożliwa)

d. ACKreed: ignoruje (sytuacja niemożliwa)

e. RELEASEreed: proces aktualizuje lokalną kolejkę żądań WaitQueueReeds. Z WaitQueueReeds usuwane jest żądanie procesu, którego pid jest zawarty w wiadomości, a reszta żądań jest przesuwana o jedną pozycję do przodu.