Nanozombie - algorytm

Marcin Pastwa, Piotr Tomaszewski

QUEUE_PONY

Kolejka procesów oczekujących na ACK_PONY.
Kolejka jest początkowo pusta.
Do kolejki trafiają procesy, które ubiegają się o dostęp do sekcji krytycznej, gdy ten proces się w niej znajduje. Czyli turyści, którzy proszą go o zgodę na pobranie stroju kucyka, gdy ma on przy sobie taki strój.
Kolejka nie zawiera duplikatów - proces wpisany do niej kilkukrotnie będzie widniał na niej tylko raz. Po wyjściu z sekcji krytycznej (po zwrocie stroju) do wszystkich procesów z kolejki wysyłane jest ACK_PONY, kolejka jest następnie czyszczona.

QUEUE_SUBMAR{id_łodzi}

Każdy proces posiada po jednej kolejce dla każdej łodzi podwodnej. W kolejce znajdują się procesy ubiegające się o miejsce na niej.

Wartość ujęta w nawiasy klamrowe oznacza, że mówimy o kolejce powiązanej z łodzią o danym indentyfikatorze.

Dzięki kolejce można wyznaczyć, które procesy mogą zająć miejsce na danej łodzi podwodnej (znaleźć się w strefie krytycznej).

Kolejka wypełniana jest zgodnie z algorytmem Lamporta, z tą różnicą, że w sekcji krytycznej na raz może znajdować się >1 proces. Dokładniej, proces, który otrzymał wszystkie potwierdzenia i w kolejce ma pozycję "*i*", może zająć miejsce (wejść do strefy krytycznej), jeśli suma rozmiarów procesów na pozycjach <="*i*" jest <= maksymalnej pojemności danej łodzi.

TRY_NO i MAX_TRY_NO

Aby proces, który czeka w kolejce do łodzi, ale nie starczyło dla niego miejsca nie musiał oczekiwać, aż ta łódź odbędzie podróż i ponownie przybije do brzegu, wycofuje się i próbuje wsiąść do innej łodzi. Takie zachowanie stwarza jednak ryzyko, że proces nigdy nie wyruszy na wyprawę. W celu rozwiązania tego problemu proces zlicza w zmiennej TRY_NO ile razy musiał się wycofać. Zmienna ta jest na początku równa 0 i jest inkrementowana przy każdym wycofaniu. Gdy procesowi uda się zająć miejsce na łodzi zmienna ta jest zerowana. Jeśli jednak liczba prób przekroczy próg MAX_TRY_NO, wtedy proces poddaje się i nie próbuje już zmieniać łodzi, tylko zostaje w kolejce do obecnej.

MAX_TRY_NO jest parametrem, jego wartość jest pewną liczbą $>= 0$. Dokładna wartość tej stałej powinna zostać dobrana eksperymentalnie.

LIST_SUBMAR

Lista, w której proces przechowuje informację o każdej łodzi podwodnej, czy jego zdaniem znajduje się ona teraz w porcie, niezapełniona, czy też nie.
Proces preferuje wybór łodzi o najmniejszym stopniu zapełnienia. Wyznaczanie zapełnienia jest złożone obliczeniowo, dlatego jako pewien rodzaj heurystyki przyjmujemy, że proces rozważa tylko te łodzie, które (według jego obecnej wiedzy) stoją w porcie i są niezapełnione. Jak zostało to już podkreślone, zawartość tej listy może być nieaktualna. Nie spowoduje to jednak błędów. W najgorszym wypadku, proces ustawi się w kolejce do łodzi, która odpłynęła i będzie musiał się z niej wycofać i wybrać inną. Jednak, ten problem występuje niezależnie od tego, czy wykorzystamy tę listę, czy nie. Próba uniknięcia tego zjawiska wiązałaby się z ograniczeniem współbieżności.

DICT_TOURIST_SIZES

Tablica poglądowa lub w ogólności słownik, w którym kluczem jest identyfikator procesu, natomiast wartością jest rozmiar turysty (ile miejsc na łodzi zajmuje).

Wartości te są stałe, więc na potrzeby algorytmu przyjmujemy, że są już każdemu procesowi znane.

Jeśli jednak założyć, że wartości te nie są znane z góry, procesy musiałyby przesłać swój rozmiar pozostałym przed rozpoczęciem pętli głównej.

DICT_SUBMAR_CAPACITY

Tablica poglądowa lub w ogólności słownik, w którym kluczem jest identyfikator łodzi podwodnej, natomiast wartością jest jej maksymalna pojemność.

Wartości te są stałe, więc na potrzeby algorytmu przyjmujemy, że są już każdemu procesowi znane.

Pozostałe stałe i zmienne

PONY_NO Łączna liczba dostępnych strojów kucyka (stała).

TOURIST_NO Łączna liczba turystów (stała).

SUBMAR_NO Łączna liczba łodzi podwodnych (stała).

REC_ACK_NO Zmienna przechowująca liczbę zebranych ACK. Jest zerowana w momencie wysłania żądania, które wymaga zebrania potwierdzeń.

Znacznik Lamporta (Timestamp)

1. Do każdej wiadomości dołączany jest znacznik czasowy (timestamp)
2. modyfikowany zgodnie z zasadami zegara logicznego Lamporta.

Wiadomości

W nawiasach klamrowych znajdują się parametry dołączane do wiadomości.

REQ_PONY żądanie dostępu do sekcji krytycznej (żądanie dostępu do stroju kucyka).

ACK_PONY potwierdzenie dostępu do stroju kucyka.

REQ_SUBMAR{id_łodzi} żądanie dostępu do sekcji krytycznej (do miejsca na łodzi podwodnej).

ACK_SUBMAR{id_łodzi} potwierdzenie wpisania do kolejki do wskazanej łodzi.

FULL_SUBMAR_RETREAT{id_łodzi} wysyłane przez proces, któremu nie udało się wsiąść do łodzi, więc się z niej wycofuje.

Wiadomości c.d.

FULL_SUBMAR_STAY{id_łodzi} wysyłane przez proces, któremu nie udało się wsiąść do łodzi, ale poddaje się i zostaje w kolejce.

RETURN_SUBMAR{id_łodzi, liczba_pasażerów} informacja, że łódź o podanym id i wioząca podaną liczbę pasażerów wróciła do portu. Dla procesów, które płynęły tą łodzią oznacza to rozkaz zwolnienia zasobów (opuszczenia jej), dla ogółu procesów, że można usunąć z kolejki podaną drugim parametrem liczbę procesów.

TRAVEL_READY weryfikacja, czy można rozpocząć podróż łodzi podwodnej.

ACK_TRAVEL odpowiedź na zapytania kapitana.

DEPART_SUBMAR informacja, że łódź, w której dany proces przebywa właśnie wypływa z portu.

Stany

RESTING	Jest to stan początkowy. Symuluje odpoczynek turysty między zakończeniem jednej wycieczki, a rozpoczęciem kolejnej.
WAIT_PONY	Proces ubiega się o dostęp do sekcji krytycznej - o pobranie stroju kucyka.
CHOOSE_SUBMAR	Proces ma już strój i wybiera sobie łódź.
WAIT_SUBMAR	Proces ubiega się o dostęp do sekcji krytycznej - o miejsce na łodzi podwodnej.
BOARDED	Proces zajął miejsce na łodzi podwodnej i czeka na rozpoczęcie wyprawy.
TRAVEL	Proces podróżuje łodzią podwodną.
ON_SHORE	Proes zakończył podróż, wysiadł z łodzi podwodnej (zwolnił zasób) i zamierza oddać (zwolnić) strój kucyka.

RESTING

1 Proces przebywa w stanie RESTING do czasu, aż podejmie decyzję
2 o rozpoczęciu wyprawy. W tym celu potrzebuje stroju kucyka,
3 zaczyna więc się o niego ubiegać.

4 Proces wysyła do wszytkich pozostałych procesów żądanie
5 REQ_PONY, po czym przechodzi do stanu WAIT_PONY.

RESTING - odpowiedzi

REQ_PONY odpowiada ACK_PONY.

ACK_PONY niemożliwe, ignoruje.

REQ_SUBMAR{id_łodzi} dodaje nadawcę do kolejki powiązanej z tą łodzią (QUEUE_SUBMAR) i odpowiada ACK_SUBMAR{id_łodzi}.

ACK_SUBMAR{id_łodzi} niemożliwe, ignoruje.

FULL_SUBMAR_STAY{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta.

FULL_SUBMAR_RETREAT{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta oraz usuwa nadawcę z kolejki powiązanej z daną łodzią QUEUE_SUBMAR{id_łodzi}.

RESTING - odpowiedzi c.d.

13 RETURN_SUBMAR{id_łodzi, liczba_pasażerów} oznacza łódź
14 jako dostępną (LIST_SUBMAR) oraz usuwa pierwsze
15 liczba_pasażerów pozycji z kolejki
16 (QUEUE_SUBMAR{id_łodzi}).

17 TRAVEL_READY niemożliwe, ignoruje.

18 ACK_TRAVEL niemożliwe, ignoruje.

19 DEPART_SUBMAR niemożliwe, ignoruje.

WAIT_PONY

Proces w tym stanie ubiega się o możliwość zabrania stroju kucyka, czyli na dostęp do sekcji krytycznej.

Ubieganie się o dostęp do sekcji krytycznej bazuje na algorytmie Ricarta - Agravali. Różnica polega na poszerzeniu sekcji krytycznej do liczby równej liczbie strojów kucyka (PONY_NO). Zatem, proces może wejść do sekcji krytycznej (zabrać strój) po zebraniu co najmniej TOURIST_NO - PONY_NO odpowiedzi ACK_PONY (różnica między liczbą procesów, a liczbą strojów). Zakładamy, że proces ma od razu jedno pozwolenie - swoje własne.
Po zebraniu minimalnej liczby pozwoleń, proces zajmuje zasoby (strój) i przechodzi do stanu GOT_PONY.

WAIT_PONY - odpowiedzi

REQ_PONY jeśli otrzymana wiadomość ma niższy priorytet dodaje nadawcę do kolejki QUEUE_PONY i nic nie odpowiada. W przeciwnym razie, uznaje pierwszeństwo i odpowiada ACK_PONY.

ACK_PONY inkrementuje REC_ACK_NO, po zdobyciu wymaganej liczby przechodzi do stanu GOT_PONY.

REQ_SUBMAR{id_łodzi} dodaje nadawcę do kolejki powiązanej z tą łodzią (QUEUE_SUBMAR) i odpowiada ACK_SUBMAR{id_łodzi}.

ACK_SUBMAR{id_łodzi} niemożliwe, ignoruje.

FULL_SUBMAR_STAY{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta.

WAIT_PONY - odpowiedzi c.d.

13 FULL_SUBMAR_RETREAT{id_łodzi} oznacza na liście
14 LIST_SUBMAR, że dana łódź jest zajęta oraz usuwa
15 nadawcę z kolejki powiązanej z daną łodzią
16 (QUEUE_SUBMAR{id_łodzi}).

17 RETURN_SUBMAR{id_łodzi, liczba_pasażerów} oznacza łódź
18 jako dostępną (LIST_SUBMAR) oraz usuwa pierwsze
19 liczba_pasażerów pozycji z kolejki
20 QUEUE_SUBMAR{id_łodzi}.

21 TRAVEL_READY niemożliwe, ignoruje.

22 ACK_TRAVEL niemożliwe, ignoruje.

23 DEPART_SUBMAR niemożliwe, ignoruje.

CHOOSE_SUBMAR

Proces znajduje się w tym stanie, gdy ma już strój kucyka, ale nie wybrał sobie jeszcze łodzi.

Proces wybiera łódź podwodną. Będzie starał się wybrać taką, która według jego obecnej wiedzy jest w najmniejszym stopniu zajęta. Ponieważ wyznaczanie zajętości wymaga kilku obliczeń, proces ogranicza sobie zbiór kandydatów do tych, które, (również według obecnej, niekoniecznie aktualnej wiedzy), znajdują się w porcie i mają jeszcze wolne miejsca (lista LIST_SUBMAR). Jeśli takich łodzi nie ma, czeka na sygnał RETURN_SUBMAR.

Po wybraniu łodzi podwodnej proces wysyła do pozostałych wiadomość REQ_SUBMAR{id_wybranej_łodzi}, dodaje się do kolejki QUEUE_SUBMAR{id_wybranej_łodzi} i przechodzi do stanu WAIT_SUBMAR.

CHOOSE_SUBMAR - odpowiedzi

REQ_PONY proces jest w sekcji krytycznej, więc dodaje nadawcę do kolejki QUEUE_PONY i nic nie odpowiada.

ACK_PONY jest już w sekcji krytycznej, więc ignoruje.

REQ_SUBMAR{id_łodzi} dodaje nadawcę do kolejki powiązanej z tą łodzią (QUEUE_SUBMAR) i odpowiada ACK_SUBMAR{id_łodzi}.

ACK_SUBMAR{id_łodzi} niemożliwe, ignoruje.

FULL_SUBMAR_STAY{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta.

CHOOSE_SUBMAR - odpowiedzi c.d.

10 FULL_SUBMAR_RETREAT{id_łodzi} oznacza na liście
11 LIST_SUBMAR, że dana łódź jest zajęta oraz usuwa
12 nadawcę z kolejki powiązanej z daną łodzią
13 (QUEUE_SUBMAR{id_łodzi}).

14 RETURN_SUBMAR{id_łodzi, liczba_pasażerów} oznacza łódź
15 jako dostępną (LIST_SUBMAR) oraz usuwa pierwsze
16 liczba_pasażerów pozycji z kolejki
17 QUEUE_SUBMAR{id_łodzi}. Jeśli zawiesił się, bo
18 nie znalazł łodzi, to ta wiadomość go budzi.

19 TRAVEL_READY niemożliwe, ignoruje.

20 ACK_TRAVEL niemożliwe, ignoruje.

21 DEPART_SUBMAR niemożliwe, ignoruje.

Uwaga do algorytmu

Uwaga, w dalszej części algorytmu potrzebny będzie proces, który wyda sygnał do odpłynięcia i potem powrotu do portu łodzi podwodnej - będzie pełnił rolę kapitana. Turyści znajdujący się na łodzi mogliby, co prawda, wspólnie podejmować te decyzje. Jednak, wprowadziłoby to tylko konieczność przesłania dodatkowych wiadomości, bez poprawy współbieżności działania - procesy w tym czasie nic nie robią, czekają jedynie na zakończenie podróży. Dlatego, postanowiliśmy ponownie wykorzystać już zbudowaną kolejkę QUEUE_SUBMAR{id_łodzi}. Sygnał do odpłynięcia i powrotu wyda zatem proces, który zdobyłby dostęp do sekcji krytycznej w algorytmie Lamporta - pierwszy w kolejce.

WAIT_SUBMAR

Proces w tym stanie oczekuje na dostęp do kolejnej sekcji krytycznej - zajęcie miejsca na wybranej łodzi podwodnej.

Tym razem, ubieganie się o dostęp do sekcji krytycznej bazuje na algorytmie Lamporta. Proces czeka, aż otrzyma potwierdzenia od wszystkich pozostałych procesów.
Następnie, sprawdza czy może zająć miejsce na łodzi. Zająć miejsce na łodzi, może, jeśli suma rozmiarów jego i procesów znajdujących się przed nim w kolejce nie przekracza maksymalnej pojemności łodzi. Jeśli może, to zajmuje miejsca, przechodzi do stanu BOARDED.

WAIT_SUBMAR - c.d.

Jeśli jednak się nie zmieści, inkrementuje licznik niepowodzeń (TRY_NO) i sprawdza, czy nie przekroczył limitu (MAX_TRY_NO). Jeśli mieści się w limicie, usuwa się z kolejki, wysyła do pozostałych wiadomość FULL_SUBMAR_RETREAT{id_łodzi} i wraca do stanu CHOOSE_SUBMAR. Jeśli jednak przekroczył limit, poddaje się i postanawia czekać, aż łódź wróci z wyprawy. Wysyła więc do pozostałych wiadomość FULL_SUBMAR_STAY{id_łodzi} i zawiesza się, do czasu, aż łódź wróci.

WAIT_SUBMAR - odpowiedzi

REQ_PONY proces jest w sekcji krytycznej, więc dodaje nadawcę do kolejki QUEUE_PONY i nic nie odpowiada.

ACK_PONY jest już w sekcji krytycznej, więc ignoruje.

REQ_SUBMAR{id_łodzi} dodaje nadawcę do kolejki powiązanej z tą łodzią (QUEUE_SUBMAR) i odpowiada ACK_SUBMAR{id_łodzi}.

ACK_SUBMAR{id_łodzi} inkrementuje licznik zebranych potwierdzeń. Po zebraniu wszystkich sprawdza, czy może wejść do łodzi.

FULL_SUBMAR_STAY{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta.

WAIT_SUBMAR - odpowiedzi c.d.

FULL_SUBMAR_RETREAT{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta oraz usuwa nadawcę z kolejki powiązanej z daną łodzią (QUEUE_SUBMAR{id_łodzi}).

RETURN_SUBMAR{id_łodzi, liczba_pasażerów} oznacza łódź jako dostępną (LIST_SUBMAR) oraz usuwa pierwsze liczba_pasażerów pozycji z kolejki QUEUE_SUBMAR{id_łodzi}.

TRAVEL_READY kolejkuje odpowiedź. Odpowiedzi udzieli po przejściu do stanu STATE_BOARDED, o ile nie zmieni łodzi.

ACK_TRAVEL niemożliwe, ignoruje.

DEPART_SUBMAR niemożliwe, ignoruje.

BOARDED

Proces w tym stanie zajął miejsca w łodzi podwodnej i czeka na rozpoczęcie wyprawy.

Jeśli proces został kapitanem czeka na otrzymanie wiadomości, że łódź jest pełna. Po otrzymaniu tej informacji, wyznacza na podstawie kolejki, które proesy z nim płyną i wysyła do każdego z nich zapytanie TRAVEL_READY (ma ono na celu zweryfikowanie czy wszyscy już wsiedli), po czym czeka, aż każdy z nich mu odpowie ACK_TRAVEL. Po otrzymaniu zgód wysyła do nich wiadomość DEPART_SUBMAR, infomrując je, że wszyscy są gotowi i można wyruszać i ponownie czeka na potwierdzenie ACK_TRAVEL. Po ich otrzymaniu przechodzi do stanu TRAVEL.

BOARDED - c.d.

12 Pozostałe procesy czekają na pytanie TRAVEL_READY.
13 Odpowiadają na nie ACK_TRAVEL. Jeżeli to zapytanie uzyskał
14 jeszcze w poprzednim stanie i nie zmienił w tym czasie łodzi, to
15 teraz na nie odpowiada.
16 Następnie proces czeka na DEPART_SUBMAR. Po otrzymaniu
17 wiadomości odpowiada ACK_TRAVEL i przechodzi do stanu
18 TRAVEL.

BOARDED - odpowiedzi

REQ_PONY proces jest w sekcji krytycznej, więc dodaje nadawcę do kolejki QUEUE_PONY i nic nie odpowiada.

ACK_PONY jest już w sekcji krytycznej, więc ignoruje.

REQ_SUBMAR{id_łodzi} dodaje nadawcę do kolejki powiązanej z tą łodzią (QUEUE_SUBMAR) i odpowiada ACK_SUBMAR{id_łodzi}.

ACK_SUBMAR{id_łodzi} niemożliwe, ignoruje.

FULL_SUBMAR_STAY{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta. Jeśli jest kapitanem tej łodzi rozpoczyna procedurę wypłynięcia.

BOARDED - odpowiedzi c.d.

FULL_SUBMAR_RETREAT{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta oraz usuwa nadawcę z kolejki powiązanej z daną łodzią (QUEUE_SUBMAR{id_łodzi}). Jeśli jest kapitanem tej łodzi rozpoczyna procedurę wypłynięcia.

RETURN_SUBMAR{id_łodzi, liczba_pasażerów} oznacza łódź jako dostępną (LIST_SUBMAR) oraz usuwa pierwsze liczba_pasażerów pozycji z kolejki QUEUE_SUBMAR{id_łodzi}.

TRAVEL_READY odpowiada ACK_TRAVEL.

ACK_TRAVEL jeśli jest kapitanem: postępuje według kroków opisanych wcześniej. Jeśli nie jest kapitanem, wiadomość niemożliwa, ignoruje.

DEPART_SUBMAR Odpowiada ACK_TRAVEL i przechodzi do stanu TRAVEL.

TRAVEL

Ten stan symuluje podróż i zwiedzanie wyspy przez turystów.

Jeśli proces został kapitanem czeka przez pewną, losową ilość czasu. Po tym czasie, uznajemy, że łódź zakończyła podróż i wróciła do brzegu. Chcemy, aby turyści, którzy przebywają w łodzi mogli z niej wysiąść, nim wsiądą do niej kolejni. Dlatego, kapitan najpierw wysyła wiadomość RETURN_SUBMAR{id_łodzi, liczba_pasażerów} do procesów, które z nim płynęły. (Liczba pasażerów jest podana, ponieważ wyznaczanie, kto płynął łodzią wymaga obliczeń. Dzięki dodaniu tego parametru pozostałe procesy nie muszą już tego wyznaczać).
Kapitan zwalnia zasoby (miejsce na łodzi), usuwa z kolejki tyle procesów, ile było pasażerów i czeka, aż pozostali pasażerowie udzielą odpowiedzi ACK_TRAVEL.

TRAVEL - c.d.

Po otrzymaniu potwierdzeń od współpasażerów kapitan wysyła RETURN_SUBMAR{id_łodzi, liczba_pasażerów} do pozostałych procesesów (tym razem nie oczekując już na potwierdzenia). Po czym przechodzi do stanu ON_SHORE.

Pozostałe procesy czekają na zakończenie podróży. Po otrzymaniu wiadomości RETURN_SUBMAR zwalniają swoje miejsca, usuwają z kolejki odpowiednią liczbę procesów, odpowiadają ACK_TRAVEL i przechodzą do stanu ON_SHORE.

TRAVEL - odpowiedzi

REQ_PONY proces jest w sekcji krytycznej, więc dodaje nadawcę do kolejki QUEUE_PONY i nic nie odpowiada.

ACK_PONY jest już w sekcji krytycznej, więc ignoruje.

REQ_SUBMAR{id_łodzi} dodaje nadawcę do kolejki powiązanej z tą łodzią (QUEUE_SUBMAR) i odpowiada ACK_SUBMAR{id_łodzi}.

ACK_SUBMAR{id_łodzi} niemożliwe, ignoruje.

FULL_SUBMAR_STAY{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta.

TRAVEL - odpowiedzi c.d.

FULL_SUBMAR_RETREAT{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta oraz usuwa nadawcę z kolejki powiązanej z daną łodzią (QUEUE_SUBMAR{id_łodzi}).

RETURN_SUBMAR{id_łodzi, liczba_pasażerów} oznacza łódź jako dostępną (LIST_SUBMAR) oraz usuwa pierwsze liczba_pasażerów pozycji z kolejki QUEUE_SUBMAR{id_łodzi}. Jeśli płynie tą łodzią, to dodatkowo zwalnia miejsce na łodzi i wysyła kapitanowi ACK_TRAVEL.

TRAVEL_READY niemożliwe, ignoruje.

ACK_TRAVEL jeśli jest kapitanem: postępuje według kroków opisanych wcześniej. Jeśli nie jest kapitanem, wiadomość niemożliwa, ignoruje.

DEPART_SUBMAR niemożliwe, ignoruje.

ON_SHORE

Proces w tym stanie zakończył właśnie podróż, zwolnił jedną sekcję krytyczną (łódź) i zamierza zwolnić kolejną - oddać strój kucyka.

Proces zwalnia strój kucyka, wysyła ACK_PONY do wszystkich procesów z QUEUE_PONY oraz czyści tę listę. Następnie przechodzi do stanu RESTING.

ON_SHORE - odpowiedzi

1. REQ_PONY odpowiada ACK_PONY.
2. ACK_PONY jest jeszcze w sekcji krytycznej, więc ignoruje.
3. REQ_SUBMAR{id_łodzi} dodaje nadawcę do kolejki powiązanej z tą łodzią (QUEUE_SUBMAR) i odpowiada ACK_SUBMAR{id_łodzi}.
4. ACK_SUBMAR{id_łodzi} niemożliwe, ignoruje.
5. FULL_SUBMAR_STAY{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta.

ON_SHORE - odpowiedzi c.d.

FULL_SUBMAR_RETREAT{id_łodzi} oznacza na liście LIST_SUBMAR, że dana łódź jest zajęta oraz usuwa nadawcę z kolejki powiązanej z daną łodzią (QUEUE_SUBMAR{id_łodzi}).

RETURN_SUBMAR{id_łodzi, liczba_pasażerów} oznacza łódź jako dostępną (LIST_SUBMAR) oraz usuwa pierwsze liczba_pasażerów pozycji z kolejki QUEUE_SUBMAR{id_łodzi}.

TRAVEL_READY niemożliwe, ignoruje.

ACK_TRAVEL niemożliwe, ignoruje.

DEPART_SUBMAR niemożliwe, ignoruje.