Nanozombie - Sprawozdanie

Marcin Pastwa 136779
Piotr Tomaszewski 136821

1 Opis problemu i słowny opis rozwiązania

Zadanie polegało na zrealizowaniu dostępu do sekcji krytycznej w systemie rozproszonym. W naszym rozwiązaniu wyróżniliśmy dwie sekcje krytyczne. Pierwsza - pobranie stroju kucyka, druga - zajęcie miejsca na łodzi.

Współbieżny dostęp do strojów kucyka zrealizowany został bazując na algorytmie Ricarta - Agravali. Różnica, w stosunku do podstawowej wersji algorytmu polega na zwiększeniu rozmiaru sekcji krytycznej do liczby procesów równej liczbie strojów kucyka. Proces może zabrać strój po otrzymaniu co najmniej takiej liczby zgód, że nawet gdyby wszystkie procesy, od których jeszcze nie dostał zgody znajdowały się w sekcji krytycznej (miały strój), to i tak zostałby co najmniej jeden strój wolny.

Po pobraniu stroju kucyka, proces wybiera łódź podwodną. Wyboru dokonuje stosując następującą heurystykę: Spośród łodzi, które jednocześnie uznawane są za dostępne oraz w powiązanej z nimi kolejce znajduje się jakiś proces, wybierana jest taka, która cechuje się najmniejszym ilorazem zajętej pojemności do całkowitej pojemności. Jeśli taka łódź nie istnieje, wybierana jest pusta łódź o najniższym id. Jeśli pustych łodzi również nie ma, wybierana jest dowolna łódź.

Po wybraniu łodzi proces zaczyna ubiegać się o dostęp do niej. Przyjęty algorytm bazuje na algorytmie Lamporta. Ponownie, została jednak rozszerzona sekcja krytyczna - proces może wejść do sekcji krytycznej, jeśli suma rozmiarów jego i procesów znajdujących się przed nim w kolejce nie przekracza maksymalnej pojemności łodzi.

Proces, który znajdzie się w sekcji krytycznej sprawdza czy może mianować się kapitanem łodzi. Kapitanem jest proces, który wygrałby dostęp do sekcji krytycznej w podstawowej wersji alg. Lamporta - czyli jest pierwszy w kolejce.

Sygnał do odpłynięcia łodzi wydaje kapitan. Dokonuje tego, gdy ustali, że nikt więcej nie wsiądzie na łódź, tj. gdy otrzyma wiadomość od jakiegoś procesu, że ten nie może zmieścić się na daną łódź albo wykryje, że nikt więcej nie będzie mógł wsiąść, gdyż łącznie na łodziach przebywa maksymalna dopuszczalna liczba turystów (tj. $\min\{$*liczba strojów kucyka*, *liczba turystów*$\}$). Kapitan wylicza ile procesów z kolejki zmieści się na łodzi, po czym wysyła do nich zapytanie mające na celu celu upewnienie się, że wszystkie z nich są już gotowe do podróży. Po otrzymaniu wszystkich potwierdzeń wysyła do nich informację o rozpoczęciu podróży. Po ponownym zebraniu potwierdzeń rozpoczyna podróż.

Po zakończeniu podróży informuje pasażerów na swojej łodzi o tym fakcie, dając im czas na opuszczenie łodzi. Wiadomość ta jest odpowiednikiem informacji o zwolnieniu sekcji krytycznej w algorytmie Lamporta, jest ona jednak zgrupowana - kapitan informuje o zwolnieniu sekcji krytycznej przez wszystkich pasażerów. Procesy odsyłają kapitanowi potwierdzenie opuszczenia i wracają na brzeg. Po zebraniu potwierdzeń, kapitan wysyła tę wiadomość ponownie, tym razem do procesów, które nie przebywały z nim na pokładzie, nie oczekuje już jednak potwierdzenia, tylko od razu wraca na brzeg.

Na brzegu następuje zwolnienie sekcji krytycznej związanej ze strojem kucyka. Zgodnie z zastosowanym alg. Ricarta - Agravali, następuje wysłanie zgody na pobranie stroju do wszystkich procesów, które wysłały prośbę w czasie, gdy omawiany proces znajdował się w sekcji krytycznej.

2 Założenia

Przyjmujemy, że środowisko jest w pełni asynchroniczne, kanały komunikacyjne są FIFO i niezawodne. Procesy nie ulegają awarii.
Ponaddto, przyjmujemy, że każdy turysta może zmieścić się w każdej pustej łodzi podwodnej, to jest $max\{S_{tourist}\} \leq min\{C_{submarine}\}$, gdzie $S_{toursit}$ to zbiór rozmiarów turystów, $C_{submarine}$ to zbiór pojemności łodzi podwodnych.
Gdyby zrezygnować z tego założenia, w szczególnym wypadku mógłby istnieć turysta, który nie zmieści się na żadnej łodzi. Proces taki nie mógłby brać udziału w przetwarzaniu, pozostawałby więc w uśpieniu, nigdy nie ubiegając się o dostęp do sekcji krytycznej.

2.1 Struktury i zmienne

received_ack_no Liczba otrzymanych wiadomości typu ACK. Jest ustawiana na 1 przed rozesłaniem wiadomości wymagającej zebrania potwierdzeń.

my_req_pony_timestamp Zmienna przechowująca wartość zegara Lamporta w momencie wysłania wiadomości REQ_PONY.

queue_pony Kolejka, na której umieszczane są id nadawców, od których proces otrzymał REQ_PONY, gdy był w sekcji krytycznej. Uwaga, kolejka ta nie zawiera duplikatów - jeśli dane id już widnieje w kolejce, nie zostanie do niej ponownie dodane.

available_submarine_list Lista łodzi podwodnych, w której proces przechowuje informację, czy jego zdaniem łodzie mają jeszcze wolne miejsca, czy też nie. Jej głównym zastosowaniem jest uniknięcie sytuacji, w której proces po wycofaniu się z kolejki powiązanej z łodzią i w trakcie wyboru kolejnej łodzi ponownie wybrałby tę samą. Początkowo, wszystkie łodzie są oznaczone jako dostępne.

queue_submar{id} Lista kolejek (po jednej dla każdej łodzi podwodnej). Proces przechowuje na niej pary (znacznik czasowy, id procesu) dla każdego z procesów, które wysłały do niego REQ_SUBMAR z parametrem stanowiącym id łodzi. Uwaga, procesy w ramach każdej kolejki są posortowane malejąco według priorytetu. Im niższy znacznik czasowy, tym wyższy priorytet. W przypadku równych znaczników, wyższy priorytet ma proces o niższym id. Początkowo, wszystkie kolejki są puste.

try_no Liczba niepowodzeń przy zajmowaniu miejsca w łodzi podwodnej. Początkowo równa 0.

my_submarine_id Id łodzi podwodnej wybranej przez proces. Początkowo może mieć dowolną wartość - pierwszy odczyt wartości tej zmiennej zawsze występuje po pierwszym zapisie.

boarded_on_my_submarine Lista tworzona przez kapitana, w której przechowuje id procesów, które znajdują się na pokładzie jego łodzi podwodnej. Początkowo pusta.

lamport_clock Zmienna, w której proces przechowuje aktualną wartość zegara Lamporta. Początkowo, wartość ta równa jest 0. Jest zwiększana o 1 przy wysyłaniu wiadomości (niezależnie od tego, do ilu procesów wiadomość ma dotrzeć), nowa wartość dołączana jest

do wiadomości. Po otrzymaniu wiadomości jest ustawiana na max{*lamport_ clock*, znacznik z otrzymanej wiadomości}+1.

Packet Struktura reprezentująca wiadomość. Zawiera następujące pola: typ wiadomości, wartość zegara Lamporta, id łodzi podwodnej, liczba pasażerów. Pierwsze dwa parametry są zawsze wymagane, kolejne dwa, w zależności od typu wiadomości, są wymagane albo mogą mieć dowolną wartość.

2.2 Stałe

Tourist_ no Całkowita liczba turystów w systemie.

Pony_ no Całkowita liczba strojów kucyka.

Submar_ no Całkowita liczba łodzi podwodnych.

Dict_ tourist_ sizes Słownik, w którym kluczem jest id turysty, wartością jest jego rozmiar.

Dict_ submar_ capacity Słownik, w którym kluczem jest id łodzi podwodnej, wartością jest jej całkowita pojemność.

Max_ try_ no Próg decydujący o tym, ile prób wejścia na łódź może podjąć proces, nim postanowi zostać w kolejce do łodzi, w której aktualnie nie może się zmieścić. Ma na celu zapewnienie warunku postępu. Może mieć wartość równą 0, wtedy proces nigdy nie będzie się wycofywał z kolejki.

2.3 Wiadomości

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

ACK_PONY Zgoda na pobranie stroju kucyka.

REQ_SUBMAR{*submarine_ id*} Żądanie dostępu do drugiej sekcji krytycznej (żądanie dostępu do wskazanej łodzi podwodnej).

ACK_SUBMAR Potwierdzenie wpisania nadawcy do odpowiedniej kolejki *queue_ submar*.

FULL_SUBMAR_RETREAT{*submarine_ id*} Wysyłana przez proces, który nie zmieścił się na łodzi podwodnej, nie przekroczył jeszcze progu *Max_ try_ no*, więc się wycofuje.

FULL_SUBMAR_STAY{*submarine_ id*} Wysyłana przez proces, który nie zmieścił się na łodzi podwodnej, przekroczył jednak próg *Max_ try_ no*, informuje więc, że pozostaje w kolejce do łodzi.

RETURN_SUBMAR{*submarine_ id, passenger_ no*} Wysyłana przez kapitana w celu poinformowania o powrocie łodzi, którą ten kapitan płynął.

TRAVEL_READY Wysyłana przez kapitana w celu weryfikacji, czy wszystkie procesy, które mogą wsiąść na łódź podwodną przeszły już do stanu **BOARDED**.

ACK_TRAVEL Potwierdzenie odsyłane kapitanowi zgodnie z zasadami podanymi w sekcji szczegółowego opisu.

DEPART_SUBMAR Wysyłana przez kapitana w celu poinformowania, że łódź, na której zarówno kapitan, jak i odbiorca przebywają właśnie odpływa z portu.

DEPART_SUBMAR_NOT_FULL Wysyłana przez kapitana w celu poinformowania, że łódź, na której zarówno kapitan, jak i odbiorca przebywają właśnie odpływa z portu oraz, że wykryte zostało zakleszczenie.

2.4 Stany

Początkowym stanem procesu jest **RESTING**.

RESTING Nie ubiega się o dostęp do żadnej sekcji krytycznej. (Turysta odpoczywa).

WAIT_PONY Ubiega się o dostęp do pierwszej sekcji krytycznej - o pobranie stroju kucyka.

CHOOSE_SUBMAR Pobrał strój kucyka, wybiera sobie łódź podwodną.

WAIT_SUBMAR Ubiega się o dostęp do drugiej sekcji krytycznej - miejsce na łodzi podwodnej.

BOARDED Oczekuje na rozpoczęcie podróży.

TRAVEL Podróżuje łodzią podwodną.

ON_SHORE Zakończył podróż, opuszcza obie sekcje krytyczne.

2.5 Opis szczegółowy

2.5.1 RESTING

Stan początkowy.
Proces przebywa w stanie **RESTING** do czasu, aż podejmie decyzję o rozpoczęciu wyprawy. Gdy postanowi wyruszyć, zaczyna ubiegać się o dostęp do pierwszej sekcji krytycznej.
Ze stanu **RESTING** następuje przejście do **WAIT_PONY** po uprzednim ustawieniu zmiennej *received_ack_no* na 1 (przyjmujemy, że proces ma już swoje pozwolenie) oraz wysłaniu REQ_PONY do wszystkich pozostałych procesów. Wartość zegara Lamporta wysłanej wiadomości jest zapisywana do zmiennej *my_req_pony_timestamp*.

Reakcja na wiadomości

REQ_PONY Odpowiada ACK_PONY.

ACK_PONY Ignoruje. Otrzymanie tej wiadomości jest możliwe tylko, jeśli proces był już chociaż raz w sekcji krytycznej.

REQ_SUBMAR{*submarine_id*} Dodaje nadawcę do kolejki *queue_submar{id}* powiązanej ze wskazaną przez parametr *submarine_id* łodzią podwodną i odpowiada ACK_SUBMAR.

ACK_SUBMAR Niemożliwe, ignoruje.

FULL_SUBMAR_STAY{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna.

FULL_SUBMAR_RETREAT{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna oraz usuwa nadawcę z kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id*.

RETURN_SUBMAR{*submarine_id, passenger_no*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest dostępna oraz usuwa z początku kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id passenger_no* pozycji.

TRAVEL_READY Niemożliwe, ignoruje.

ACK_TRAVEL Niemożliwe, ignoruje.

DEPART_SUBMAR Niemożliwe, ignoruje.

DEPART_SUBMAR_NOT_FULL Niemożliwe, ignoruje.

2.5.2 WAIT_PONY

Proces przechodzi ze stanu **WAIT_PONY** do **CHOOSE_SUBMAR** po otrzymaniu wystarczającej liczby odpowiedzi ACK_PONY, tak aby mieć pewność, że może bezpiecznie zająć sekcję krytyczną (tj. zabrać strój kucyka). Pewność tę uzyskuje, gdy jego zmienna *received_ack_no* ma wartość co najmniej $Tourist\_no - Pony\_no + 1$. W szczególnym przypadku wymagana liczba potwierdzeń może być mniejsza od 1. W takiej sytuacji, proces nie musi czekać na żadne potwierdzenie, może od razu przejść do następnego stanu.

Reakcja na wiadomości

REQ_PONY Porównuje znacznik czasowy otrzymanej wiadomości z *my_req_pony_timestamp*. Jeśli otrzymana wiadomość ma niższy priorytet (jej znacznik czasowy jest większy od *my_req_pony_timestamp* albo, jeśli znaczniki są równe, id nadawcy jest większe od id omawianego procesu), dodaje nadawcę do kolejki *queue_pony* i nic nie odpowiada. W przeciwnym razie odpowiada ACK_PONY.

ACK_PONY Inkrementuje zmienną *received_ack_no*.

REQ_SUBMAR{*submarine_id*} Dodaje nadawcę do kolejki *queue_submar{id}* powiązanej ze wskazaną przez parametr *submarine_id* łodzią podwodną i odpowiada ACK_SUBMAR.

ACK_SUBMAR Niemożliwe, ignoruje.

FULL_SUBMAR_STAY{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna.

FULL_SUBMAR_RETREAT{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna oraz usuwa nadawcę z kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id*.

RETURN_SUBMAR{*submarine_id, passenger_no*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest dostępna oraz usuwa z początku kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id passenger_no* pozycji.

TRAVEL_READY Niemożliwe, ignoruje.

ACK_TRAVEL Niemożliwe, ignoruje.

DEPART_SUBMAR Niemożliwe, ignoruje.

DEPART_SUBMAR_NOT_FULL Niemożliwe, ignoruje.

2.5.3 CHOOSE_SUBMAR

Aby przejść ze stanu **CHOOSE_SUBMAR** do **WAIT_SUBMAR** proces musi najpierw wybrać, o miejsce na której łodzi będzie się ubiegał. Wyboru dokonuje według następującego algorytmu: Proces wybiera spośród łodzi, w których kolejce *queue_submar{id}* jest przynajmniej jeden proces i jednocześnie, na liście *available_submarine_list* są oznaczone jako 'available'. Jeśli istnieją łodzie spełniające te warunki, to wybrana zostaje taka, która jest w najmniejszym stopniu zajęta (iloraz zajętego miejsca do całkowitej pojemności - wartość ta jest szacowana na podstawie listy *queue_submar* i słownika *Dict_tourist_sizes*). Jeśli taka łódź nie istnieje, wybierana jest pusta łódź o najniższym id. Jeśli takiej łodzi też nie ma, wybierana jest dowolna łódź. Id wybranej łodzi zostaje zapisane w zmiennej *my_submarine_id*
Po wyborze łodzi, proces ustawia zmienną *received_ack_no* na 1, wysyła do wszystkich pozostałych procesów wiadomość REQ_SUBMAR{*my_submarine_id*}. Może teraz przejść do stanu **WAIT_SUBMAR**.

Reakcja na wiadomości

REQ_PONY Proces jest w sekcji krytycznej, więc dodaje nadawcę do kolejki *queue_pony* i nic nie odpowiada.

ACK_PONY Ignoruje.

REQ_SUBMAR{*submarine_id*} Dodaje nadawcę do kolejki *queue_submar{id}* powiązanej ze wskazaną przez parametr *submarine_id* łodzią podwodną i odpowiada ACK_SUBMAR.

ACK_SUBMAR Niemożliwe, ignoruje.

FULL_SUBMAR_STAY{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna.

FULL_SUBMAR_RETREAT{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna oraz usuwa nadawcę z kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id*.

RETURN_SUBMAR{*submarine_id, passenger_no*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest dostępna oraz usuwa z początku kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id* *passenger_no* pozycji.

TRAVEL_READY Niemożliwe, ignoruje.

ACK_TRAVEL Niemożliwe, ignoruje.

DEPART_SUBMAR Niemożliwe, ignoruje.

DEPART_SUBMAR_NOT_FULL Niemożliwe, ignoruje.

2.5.4 WAIT_SUBMAR

Z tego stanu proces może przejść do **BOARDED** albo wrócić do **CHOOSE_SUBMAR**. Proces czeka na otrzymanie ACK_SUBMAR od każdego innego procesu, następnie na podstawie listy *queue_submar{my_submarine_id}* sprawdza czy zmieści się na łodzi. Jeśli tak, to przechodzi do stanu **BOARDED**.

W przeciwnym razie, inkrementuje licznik niepowodzeń *try_ no*. Jeśli nowa wartość licznika nie przekracza limitu *Max_ try_ no*, proces usuwa się z kolejki *queue_ submar{my_ submarine_ id}*, oznacza łódź jako niedostępną w *available_ submarine_ list*, wysyła do wszystkich pozostałych procesów wiadomość FULL_SUBMAR_RETREAT{*my_ submarine_ id*}, po czym wraca do stanu **CHOOSE_SUBMAR**.
Jeśli jednak próg licznika został przekroczony, proces wysyła do pozostałych FULL_SUBMAR_STAY{*my_ submarine_ id*}, po czym zawiesza się, aż otrzyma RETURN_SUBMAR{*my_ submarine_ id, ...*}. Po przebudzeniu i stosownym obsłużeniu wiadomości ponownie sprawdza czy może się zmieścić. Jeśli tak, przechodzi do stanu **BOARDED**, jeśli nie, czeka ponownie.

Reakcja na wiadomości

REQ_PONY Proces jest w sekcji krytycznej, więc dodaje nadawcę do kolejki *queue_ pony* i nic nie odpowiada.

ACK_PONY Ignoruje.

REQ_SUBMAR{*submarine_ id*} Dodaje nadawcę do kolejki *queue_ submar{id}* powiązanej ze wskazaną przez parametr *submarine_ id* łodzią podwodną i odpowiada ACK_SUBMAR.

ACK_SUBMAR Inkrementuje licznik *received_ ack_ no*.

FULL_SUBMAR_STAY{*submarine_ id*} Oznacza na liście *available_ submarine_ list*, że łódź wskazywana przez parametr *submarine_ id* jest niedostępna.

FULL_SUBMAR_RETREAT{*submarine_ id*} Oznacza na liście *available_ submarine_ list*, że łódź wskazywana przez parametr *submarine_ id* jest niedostępna oraz usuwa nadawcę z kolejki *queue_ submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_ id*.

RETURN_SUBMAR{*submarine_ id, passenger_ no*} Oznacza na liście *available_ submarine_ list*, że łódź wskazywana przez parametr *submarine_ id* jest dostępna oraz usuwa z początku kolejki *queue_ submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_ id* *passenger_ no* pozycji. Jeśli proces oczekuje na powrót tej łodzi, budzi się i wykonuje kroki opisane powyżej.

TRAVEL_READY Kolejkuje odpowiedź. Odpowiedzi udzieli po przejściu do **BOARDED**, o ile do tego czasu nie zmieni łodzi.

ACK_TRAVEL Niemożliwe, ignoruje.

DEPART_SUBMAR Niemożliwe, ignoruje.

DEPART_SUBMAR_NOT_FULL Niemożliwe, ignoruje.

2.5.5 BOARDED

Na początku proces zeruje wartość *try_ no*. Z tego stanu proces może przejść tylko do **TRAVEL**. Jednak, moment przjścia jest zależny od tego czy proces został kapitanem łodzi podwodnej, czy też nie. Kapitanem mianowany jest proces, który znajduje się na pierwszej pozycji w kolejce powiązanej z łodzią, na której przebywa.
Jest kapitanem

Oczekuje na wiadomość, że łódź jest pełna: FULL_SUBMAR_STAY albo FULL_SUBMAR_RETREAT z parametrem równym *my_submarine_id*. Może się jednak zdarzyć, że taka wiadomość nigdy nie nadejdzie - dojdzie do zakleszczenia. Kapitan stara się wykryć taką sytuację poprzez sprawdzanie, po każdej otrzymanej i obsłużonej wiadomości REQ_SUBMAR, czy łączna liczba procesów w kolejkach *queue_submar* jest równa min{*Pony_no*, *Tourist_no*}.
Niezależnie od przyczyny, po wykryciu konieczności wypłynięcia proces, na podstawie właściwiej *queue_submar* wyznacza listę procesów znajdujących się na łodzi i umieszcza ją w uprzednio wyczyszczonej zmiennej *boarded_on_my_submarine*. Proces ustawia *received_ack_no* na 1. Do każdego procesu z tej listy (oprócz siebie) wysyła zapytanie TRAVEL_READY i oczekuje na otrzymanie od każdego z nich ACK_TRAVEL. Mając już pewność, że pozostałe procesy są gotowe do wypłynięcia, ustawia *received_ack_no* na 1, po czym wysyła do tych samych procesów: jeśli nie wykrył zakleszczenia wiadomość DEPART_SUBMAR, w przeciwnym razie DEPART_SUBMAR_NOT_FULL. Ponownie czeka na potwierdzenie ACK_TRAVEL, po czym przechodzi do stanu **TRAVEL**.

Nie jest kapitanem

Jeśli w poprzednim stanie otrzymał zapytanie TRAVEL_READY i nie zmienił łodzi, odpowiada ACK_TRAVEL, w przeciwnym razie, oczekuje na TRAVEL_READY, na który odpowiada ACK_TRAVEL. Następnie oczekuje na DEPART_SUBMAR albo DEPART_SUBMAR_NOT_FULL. Ponownie odpowiada ACK_TRAVEL i przechodzi do stanu **TRAVEL**.
Czasami może się zdarzyć (np. w przypadku wykrycia zakleszczenia, gdyż wtedy łodzie odpływają mając wolne miejsca), że proces znajdzie się w stanie **BOARDED**, gdy wybrana przez niego łódź jest w podróży. Dlatego, śpiący proces, oprócz obu wersji DEPART_SUBMAR może być obudzony przez RETURN_SUBMAR{*my_submarine_id, ...*}. Taki proces zachowuje się tak, jak gdyby dopiero wszedł do stanu **BOARDED**.

Reakcja na wiadomości

REQ_PONY Proces jest w sekcji krytycznej, więc dodaje nadawcę do kolejki *queue_pony* i nic nie odpowiada.

ACK_PONY Ignoruje.

REQ_SUBMAR{*submarine_id*} Dodaje nadawcę do kolejki *queue_submar{id}* powiązanej ze wskazaną przez parametr *submarine_id* łodzią podwodną i odpowiada ACK_SUBMAR. Jeśli jest kapitanem, sprawdza czy doszło do zakleszczenia.

ACK_SUBMAR Niemożliwe, ignoruje.

FULL_SUBMAR_STAY{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna. Jeśli jest kapitanem, rozpoczyna procedurę wypłynięcia.

FULL_SUBMAR_RETREAT{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna oraz usuwa nadawcę z kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id*. Jeśli jest kapitanem, rozpoczyna procedurę wypłynięcia oraz usuwa proces z listy *boarded_on_my_submarine*, dodatkowo przestaje oczekiwać na ACK_TRAVEL od tego procesu.

RETURN_SUBMAR{*submarine_id, passenger_no*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest dostępna oraz usuwa z początku

kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id* *passenger_no* pozycji. Jeśli *submarine_id* = *my_submarine_id*, proces budzi się.

TRAVEL_READY Jeśli nim jest - niemożliwe, ignoruje, jeśli nim nie jest, odpowiada ACK_TRAVEL.

ACK_TRAVEL Jeśli jest kapitanem inkrementuje licznik otrzymanych potwierdzeń, jeśli nim nie jest, niemożliwe, ignoruje.

DEPART_SUBMAR Jeśli jest kapitanem - niemożliwe, ignoruje. Jeśli nim nie jest, odpowiada ACK_TRAVEL i przechodzi do stanu TRAVEL.

DEPART_SUBMAR_NOT_FULL Jeśli jest kapitanem - niemożliwe, ignoruje. Jeśli nim nie jest, postępuje tak jak w przypadku DEPART_SUBMAR oraz oznacza wszystkie łodzie, w których kolejce *queue_submar* znajduje się jakiś proces jako niedostępne (lista *available_submarines*).

2.5.6 TRAVEL

Ponownie, przetwarzanie jest zależne od tego czy proces jest kapitanem, zawsze jednak przejście następuje do **ON_SHORE**.

Jest kapitanem

Czeka, do czasu, aż uzna, że podróż się skończyła i można dać pozostałym pasażerom sygnał do opuszczenia łodzi.

W tym celu, ustawia *received_ack_no* na 1, wysyła do każdego procesu z listy *boarded_on_my_submarine* (oprócz siebie samego) wiadomość RETURN_SUBMAR dołączając do niej id łodzi oraz liczbę procesów na liście *boarded_on_my_submarine*. Usuwa z odpowiedniej kolejki *queue_submar* liczbę procesów równą liczbie pasażerów i czeka, na odpowiedzi ACK_TRAVEL.

Następnie wysyła tę samą wiadomość do wszystkich procesów, które nie znajdowały się z nim na pokładzie, tym razem nie oczekując już jednak potwierdzenia. Po czym przechodzi do stanu **ON_SHORE**.

Nie jest kapitanem

Czeka na wiadomość RETURN_SUBMAR od kapitana, usuwa wskazaną liczbę procesów z kolejki, odpowiada ACK_TRAVEL i przechodzi do stanu **ON_SHORE**.

Reakcja na wiadomości

REQ_PONY Proces jest w sekcji krytycznej, więc dodaje nadawcę do kolejki *queue_pony* i nic nie odpowiada.

ACK_PONY Ignoruje.

REQ_SUBMAR{*submarine_id*} Dodaje nadawcę do kolejki *queue_submar{id}* powiązanej ze wskazaną przez parametr *submarine_id* łodzią podwodną i odpowiada ACK_SUBMAR.

ACK_SUBMAR Niemożliwe, ignoruje.

FULL_SUBMAR_STAY{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna.

FULL_SUBMAR_RETREAT{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna oraz usuwa nadawcę z kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id*.

RETURN_SUBMAR{*submarine_id, passenger_no*} Jeśli wiadomość dotyczy łodzi, na której się znajduje postępuje według kroków opisanych powyżej. W przeciwnym razie, oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest dostępna oraz usuwa z początku kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id passenger_no* pozycji.

TRAVEL_READY Niemożliwe, ignoruje.

ACK_TRAVEL Jeśli jest kapitanem inkrementuje licznik otrzymanych potwierdzeń, jeśli nim nie jest, niemożliwe, ignoruje.

DEPART_SUBMAR Niemożliwe, ignoruje.

DEPART_SUBMAR_NOT_FULL Niemożliwe, ignoruje.

2.6 ON_SHORE

Przechodzi do stanu **RESTING** po wysłaniu ACK_PONY do wszystkich procesów z kolejki *queue_pony* czyszcząc tę listę.

Reakcja na wiadomości

REQ_PONY Odpowiada ACK_PONY.

ACK_PONY Ignoruje.

REQ_SUBMAR{*submarine_id*} Dodaje nadawcę do kolejki *queue_submar{id}* powiązanej ze wskazaną przez parametr *submarine_id* łodzią podwodną i odpowiada ACK_SUBMAR.

ACK_SUBMAR Niemożliwe, ignoruje.

FULL_SUBMAR_STAY{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna.

FULL_SUBMAR_RETREAT{*submarine_id*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest niedostępna oraz usuwa nadawcę z kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id*.

RETURN_SUBMAR{*submarine_id, passenger_no*} Oznacza na liście *available_submarine_list*, że łódź wskazywana przez parametr *submarine_id* jest dostępna oraz usuwa z początku kolejki *queue_submar{id}* powiązanej z łodzią wskazywaną przez parametr *submarine_id passenger_no* pozycji.

TRAVEL_READY Niemożliwe, ignoruje.

ACK_TRAVEL Niemożliwe, ignoruje.

DEPART_SUBMAR Niemożliwe, ignoruje.

DEPART_SUBMAR_NOT_FULL Niemożliwe, ignoruje.