Zabawa w zabijanie

Struktury i zmienne:

1) WaitQueue - Kolejka procesów oczekujących na ACK, początkowo pusta
2) AckNum - liczba otrzymanych potwierdzeń ACK (początkowo 0)
3) NackNum - liczba otrzymanych potwierdzeń NACK (początkowo 0)
4) NackList - lista identyfikatorów procesów, które wysłały procesowi NACK przy próbie dostępu do sekcji krytycznej (używane przy sekcji krytycznej pistoletów)
5) n - liczba procesów
6) p - liczba pistoletów
7) winAmount - liczba wygranych potyczek w parze (początkowo 0)
8) cyclesNum - liczba cykli w grze (parametr >=1)
9) currentCycle - aktualny odbywany cykl (początkowo 0)

Wiadomości:

1) Wszystkie wiadomości są podbite znacznikiem czasowym (timestampem), modyfikowanym zgodnie z zasadami skalarnego zegara logicznego Lamporta.
    A) REQ - żądanie o dostęp do sekcji krytycznej. Zawiera priorytet żądania
    B) ACK - potwierdzenie dostępu do sekcji krytycznej
    C) NACK - informacja o byciu niżej w kolejce priorytetów ubiegania się o sekcję krytyczną niż inny proces (odpowiedź na REQ)
    D) RELEASE - informacja o zwolnieniu miejsca w sekcji krytycznej przez proces

Stany:

1) Początkowym stanem procesu jest INIT
    A) INIT - stan przyjmowany na początku rundy
    B) WAIT_ROLE - oczekiwanie na dostęp do sekcji krytycznej przy wyborze roli
    C) ROLE_PICKED - stan pasywny po wyborze roli, oczekiwanie na wystarczające wypełnienie się kolejki oraz, w wypadku ofiar, oczekiwanie na REQ w sprawie par
    D) WAIT_PAIR - oczekiwanie na odpowiedź przy dobieraniu się w pary
    E) WAIT_GUN - oczekiwanie na dostęp do sekcji krytycznej przyznawającej pistolet
    F) ROLLING - rzut przeciwstawny kośćmi o zwycięzcę potyczki
    G) WAIT_END - oczekiwanie na dostęp do sekcji krytycznej końca rundy
    H) FINISHED - koniec gry w rundzie i oczekiwanie na detekcję zakończenia

Opis algorytmu:

Dostęp do sekcji krytycznej

1) Im większy zegar Lamporta, tym mniejszy priorytet.
2) Proces i ubiegający się o wejście do sekcji krytycznej zabójców wysyła do wszystkich pozostałych prośby REQ o dostęp. Proces wchodzi do sekcji po otrzymaniu ACK od przynajmniej połowy procesów oraz NACK od reszty.
   AckNum >= n/2 oraz <= n/2-1 NACK. (NackNum + AckNum == n/2-1)
3) Proces i ubiegający się o wejście do sekcji krytycznej pistoletów wysyła prośby REQ do innych zabójców. Proces wchodzi do sekcji po otrzymaniu ACK bądź RELEASE od n/2-p procesów oraz NACK od reszty.
   n/2-p >= ACK || RELEASE oraz <= p-1 NACK.
4) Procesy otrzymujące REQ zapamiętują żądanie w kolejce WaitQueue. Odsyłają ACK do procesu i, o ile same się nie ubiegają o dostęp albo jeżeli priorytet ich żądania jest mniejszy od priorytetu procesu i. W przeciwnym wypadku odsyłają NACK, po wyjściu z sekcji krytycznej odsyłają RELEASE.
5) W momencie otrzymania RELEASE od innego procesu, wcześniejsza odpowiedź NACK od niego jest usuwana.

Dobieranie w pary

1) Na podstawie priorytetów żądań w WaitQueue. Procesy które nie dostały dostępu do sekcji krytycznej zabójców zostają ofiarami. Pierwszy proces poza sekcją krytyczną zostanie sparowany z pierwszym procesem z sekcji krytycznej itd.

Bariera zakończenia cyklu

1) Proces który dostał rolę zabójcy z najwyższym możliwym priorytetem wysyła do sąsiedniego procesu w topologii pierścienia token o z wartością 0.
2) Proces otrzymujący token przesyła go kolejnemu jeśli skończył już cykl, w przeciwnym wypadku zmienia wartość na 1 i odsyła go do procesu który zainicjalizował koniec cyklu.

Przebieg cyklu

1) Wszystkie procesy ubiegają się o dostęp do sekcji krytycznej zabójców.
2) Dobranie w pary.
3) Zabójcy ubiegają się o dostęp do sekcji krytycznej z pistoletami. Po otrzymaniu dostępu, wysyłają do swojej ofiary losową liczbę z wiadomością ROLLING. Ofiara po otrzymaniu żądania odsyła ROLLING z inną losową liczbą. Wśród pary, jeśli otrzymana liczba z żądaniem ROLLING jest niższa od wysłanej do przeciwnika, proces inkrementuje swój winAmount.
4) Bariera zakończenia cyklu.
5) Rozpoczęcie następnego cyklu w przypadku gdy currentCycle < cyclesNum-1 i inkrementacja zmiennej lokalnej currentCycle. W przeciwnym wypadku rozstrzygnięcie zwycięzcy.

Rozstrzyganie zwycięzcy i detekcja zakończenia

1) Po skończeniu wszystkich swoich cykli, proces który dostał rolę zabójcy z najwyższym możliwym priorytetem wysyła do sąsiedniego procesu w topologii pierścienia token z licznikiem zainicjowanym wartością 0 i zmienną zainicjowaną swoim identyfikatorem i wynikiem.
2) Każdy proces w ten sposób przesyła dalej ten token inkrementując zawarty w nim licznik; jeżeli wynik zdobyty przez proces jest wyższy od przechowywanego w tokenie, jest on aktualizowany, o ile skończył już on rozgrywkę. W przeciwnym wypadku licznik przed przesłaniem jest zerowany.
3) Jeżeli licznik = n, to oznacza że wszyscy skończyli rozgrywkę, więc proces resetuje się do stanu początkowego i zaczyna uczestnictwo w kolejnej rundzie. Zebrany najwyższy wynik jest wypisywany. O ile pozostały rundy gry, to token wciąż jest przekazywany, by poinformować resztę procesów o rozpoczęciu następnej rundy. Jeżeli wszystkie rundy zostały zagrane, program się kończy.