Gorące pyrczoki

Jacek Pospychała, Katarzyna Rafalska

4 czerwca 2006

1 Treść zadania

Dane jest N procesów, każdy z nich posiada koszyk. K spośród nich posiada ziemniaka, którego pragnie się pozbyć. Proces posiadający ziemniak wybiera więc pusty koszyk i stara się do niego wrzucić ziemniak. Sytuacja, gdy ziemniak jest wrzucany do koszyka, który w miedzyczasie się zapełnił jest niedopuszczalna.
Założenia: $N > K$

2 Omówienie algorytmu

2.1 Typy komunikatów

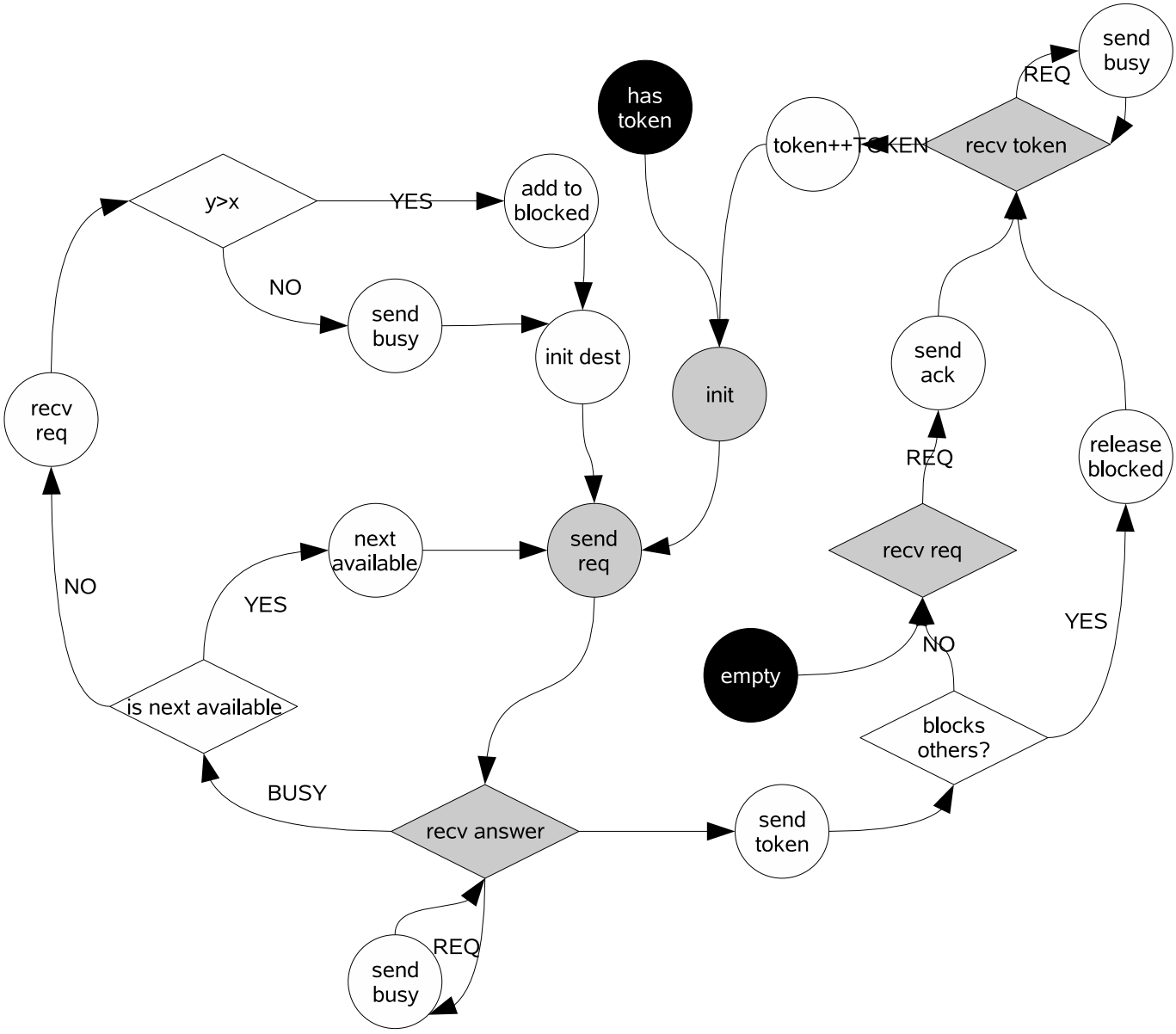
REQ(X)	Zapytanie, czy token (ziemniak) o wartości X zostanie przyjęty.
ACK	Potwierdzenie chęci przyjęcia tokenu.
BUSY	Odrzucenie zapytania o przyjęcie tokenu. Proces jest zajęty przetwarzaniem innego.
TOKEN(X)	Gorący pyrczok. O wartości X, tj. dotychczas przekazany X razy.

2.2 Scenariusz

1. Proces i otrzymał pyrczoka ($token(x)$).
2. i wysyła zapytanie do procesu $j = i + 1$, czy jest wolny.
3. i oczekuje na odpowiedź
   (a) proces j jest wolny, procej i wysyła mu pyrczoka, zwiększając jego liczbę rzuceń i rozpoczyna oczekiwanie na zapytania od innych. Jeżeli jednak wstrzymywał odpowiedź innym procesom, których pyrczoki miały większe wartości (były więcej razy rzucane), teraz przyjmuje pyrczoka o najmniejszej liczbie rzuceń, a pozostałym przekazuje informację że jest zajęty. Do punktu 2.
   (b) proces i otrzymuje zapytanie od dowolnego procesu k z pyrczokiem. Odpowiada że jest zajęty i pozostaje w punkcie 3.
   (c) proces j jest zajęty, proces i wysyła zapytanie do $j + 1$ i wraca do kroku 3. Jeżeli kontaktował się już z wszystkimi, przechodzi do punktu 4.
4. Jeżeli w punkcie 3. proces nie odpowiedział innym procesom (tj. był w pkt. 3.b), dokonuje ponownej próby znalezienia biorcy dla swego pyrczoka. Przechodzi do punktu 2.
5. Proces nie znalazł biorcy dla swego pyrczoka. Oczekuje więc na nowe zapytanie od procesu z pyrczokiem, który ciągle zajmuje potencjalnych biorców, tak by móc go zablokować i przejść do kolejnego punktu.
6. Proces i wznawia odpytywanie, wraca do punktu 2. Z zastrzeżeniem, że będzie odpytywał tylko te procesy, których już w międzyczasie nie zablokował.

2.3 Graf przejść procesu

Graf na rysunku 1 przedstawia czynności, jakie wykonuje program, zależnie od tego w jakim jest stanie, lub jaką otrzyma wiadomość. Odzwierciedla główną część kodu programu. Czarne stany, to stany startowe, stany wyszczególnione w głównej pętli programu oznaczono kolorem szarym. Oznaczenia: x-wartość tokena, y-wart.tokena przychodzącego, *recv*-odebranie wiadomości. Na łuku oznaczono typ otrzym. wiadomości lub wynik operacji logicznej.



Rysunek 1: Graf przejść procesu.

3 Złożoność czasowa i komunikacyjna

Pesymistyczny wariant zakłada, że wśród N procesów, tylko jeden jest wolny tj. $K = N - 1$ posiada pyrczoka. Złożoność czasowa wynika z takiego toku postępowania: Proces i odpytuje pozostałe $N - 1$ procesy, nie znajduje wolnego i zatrzymuje się. Do czasu aż otrzymuje zapytanie od innego procesu, blokuje go i odpytuje pozostałe $N - 2$ procesy (bez siebie i zablokowanego). Nie znajduje biorcy dla pyrczoka i tak powtarza cykle, kolejno zablokowując pozostałe procesy. Ostatecznie zablokowane są wszystkie procesy poza jednym, wolnym. Token zostaje przesłany, a zablokowane procesy zwolnione. Cykli odpytywania jest $K = N-1$, czyli pierwszy cykl, oraz $K-1$ cykli wynikających z zablokowania pozostałych pyrczoków. Każda komunikacja składa się z zapytania i odpowiedzi. Ostatnia dodatkowo

z przesłania tokenu (+1). Matematyczny zapis odzwierciedlający taki przypadek:

$$2(N-1)+2(N-2)+\ldots+2(N-N+2)+2(N-N+1)+1=$$
$$2(N-1+N-2+\ldots+2+1)+1= \quad (1)$$
$$N(N-1)+1 \quad (2)$$

Przejście z (1) do (2) z własności liczb trójkątnych. Ponieważ każdy krok algorytmu de facto odpowiada wysłaniu dokładnie jednej wiadomości, zatem złożoność komunikacyjna, a więc liczona w liczbie wysłanych wiadomości jest równa złożoności czasowej. Wyznaczając bitową złożoność komunikacyjną, tj. uwzględniając rozmiar pakietów zakładamy że typ wiadomości zajmuje A bitów (minimalnie 2 bity, co koduje 4 typy wiadomości), a licznik zawarty w wiadomościach TOKEN i REQ, ma rozmiar B bitów (tak by $2^B > N$), należy tak naprawdę określić, jaką część wszystkich wiadomości stanowią komunikaty TOKEN i REQ. Przesłanie TOKEN jest tylko jedno, natomiast przesłań REQ jest dokładnie połowa. Zatem jest to:

$$A(N(N-1)+1)+\tfrac{B}{2}(N(N-1)+1)$$

Pamiętając, że $N(N-1)$ jest zawsze podzielne przez 2, wynik będzie zawsze liczbą całkowitą, a zakładając jednakowy rozmiar wszystkich wiadomości ($A=1$, $B=0$), równanie sprowadza się do poprzedniego. Jest więc ogólniejsze.