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: $N > K$

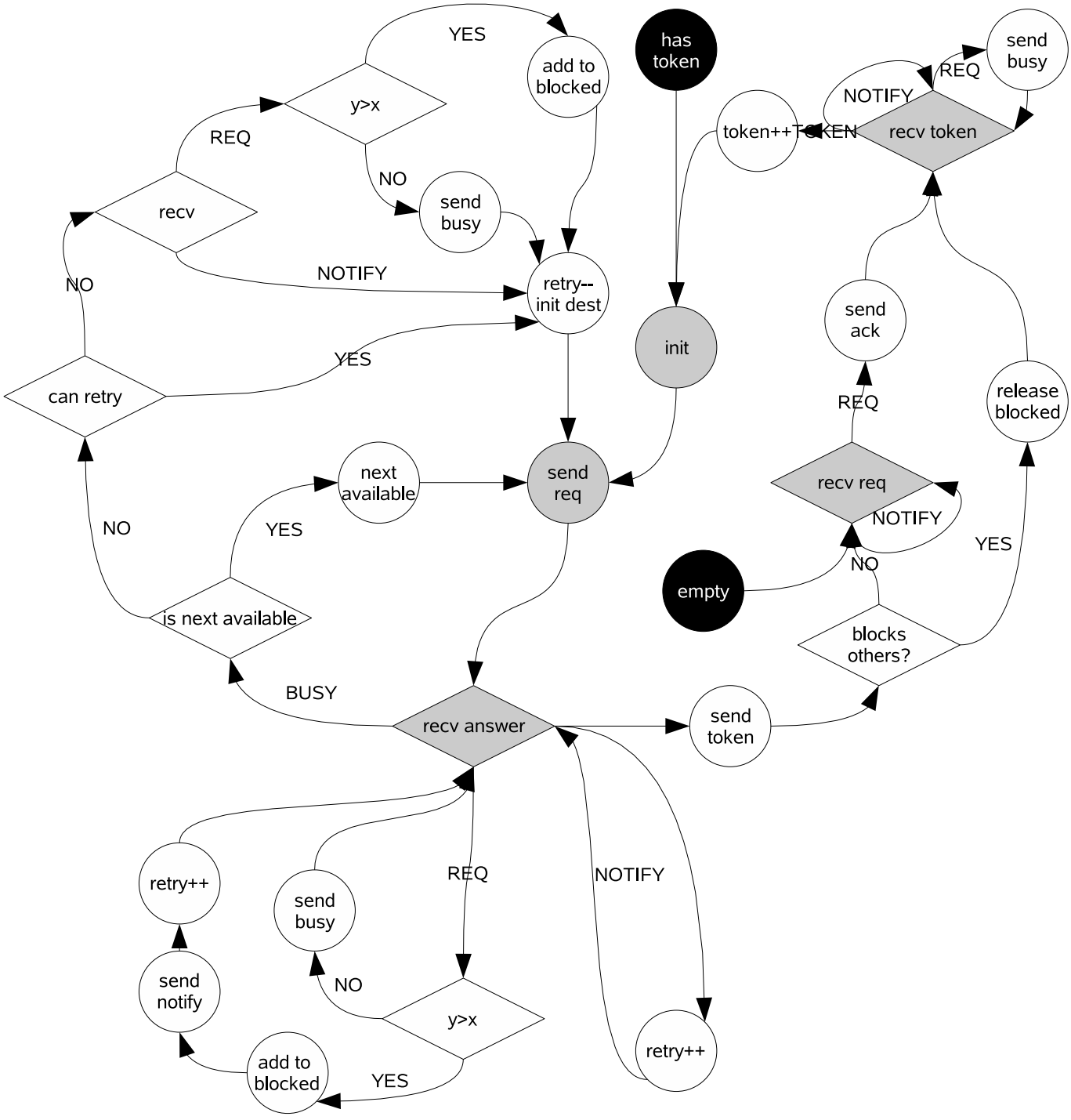
2 Omówienie algorytmu

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.
NOTIFY	Przypomnienie.

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. Jeżeli pyrczok procesu k był przekazywany więcej razy, niż procesu i, proces i zapamiętuje liczbę rzuceń k i nie odpowiada mu, dopóki sam się nie pozbędzie swego pyrczoka. W przeciwnym razie odpowiada że jest zajęty. 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, prawdopodobnie zatrzymał inne procesy chcące wysłać swoje (mniej pilne) pyrczoki i w kolejnych próbach także nie znalazł biorcy. Oczekuje więc na nowe zapytanie od procesu z pyrczokiem, który ciągle zajmuje potencjalnych biorców, by móc go zablokować i przejść do działania. Jeżeli któryś z procesów powstrzymywanych przez i (lub któryś z procesów przez nie powstrzymywanych), otrzyma takie zapytanie, prześle procesowi i przypomnienie ($notify$).
6. Proces i wznawia odpytywanie, wraca do punktu 2. Z zastrzeżeniem, że będzie odpytywał tylko te procesy, których już w międzyczasie nie zablokował.



Rysunek 1: Graf 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.

2.3 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 $N - 2$ procesy (poza sobą, oraz pechowo tym, który jest wolny) aż zostaje zablokowany przez j. Odpytanie to 2 wiadomości - ACK i BUSY. j odpytuje $N - 3$ (poza sobą, zablokowanym i wolnym) i zostaje zablokowany przez k, itd. aż zostaną tylko dwa procesy z których jeden blokuje wszystkie, a drugi jest wolny. Następuje przesłanie tokena. Proces który pozbył się tokena, odpowiada poprzednio zablokowanemu (ACK), otrzymuje TOKEN. itd. czyli $K-$krotnie następuje wymiana 2 wiadomości (ACK i TOKEN). Podsumowując, pierwszy proces odpytuje $N - 2$ procesów, otrzymuje $K - 1$ odrzuceń i 1 potwierdzenie ($N - 2$ odpowiedzi), wysyła 1 token. kolejne $N - 3$ procesy (poza pierwszym, wolnym i ostatnim) odpytują $N - 3$ procesów, otrzymują $K - 2$ odrzuceń i 1 potwierdzenie ($N - 3$ odpowiedzi), wysyłają 1 token, ostatni odpytuje $N - 2$ procesy, otrzymuje $K-1$ odrzuceń i 1 potwierdzenie, wysyła 1 token. Łączną liczbę wysłań odpowiada więc wyrażenie:

$$(N-2+N-2+1)+(N-3)(N-3+N-3+1)+(N-2+N-2+1)=$$
$$2(2N-4+1)+(N-3)(2N-6+1)=4N-6+2N^2-5N-6N+15=$$
$$2N^2-7N+9$$

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 3 bity), 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. W scenariuszu opisanym przed chwilą następuje $K = N - 1$ przesłań pyrczoków (wiadomości typu TOKEN). Natomiast wiadomości REQ jest $(N-2)+(N-3)(N-3)+(N-2) = N^2-4N+5$. Drugie tyle to wiadomości ACK lub BUSY. Wyrażenie określające bitową złożoność komunikacyjną:

$$A((N-2)+(N-3)(N-3)+(N-2)+(N-1))+(A+B)((N-2)+(N-3)(N-3)+(N-2))=$$
$$A(N^2-3N+4)+(A+B)(N^2-4N+5)$$
$$A(2N^2-7N+9)+B(N^2-4N+5)$$