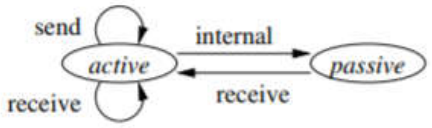
Kapitel 6: Termination Detection

21. december 2017 13:26

Fire forskellige:
Dijkstra-Scholten algoritm
Weight-throwing algoritm
Ranas algortim
Safras algoritm

Side 92-108 i http://www.cs.vu.nl/~tcs/da/daslides.pdf

Grundalgoritmen afsluttes, hvis (1) hver proces er passiv, og
(2) ingen grundlæggende meddelelser er i transit.



Kontrolalgoritmen vedrører termination detection og announcement.
announcement enkel; vi fokuserer på detection.
Termination detection bør ikke påvirke basale beregninger.

Dijkstra-Scholten algoritm:
Kræver en centraliseret grundlæggende algoritme og et uindrettet netværk.

Et træ T opretholdes, som har initiatoren p0 som rod og
omfatter alle aktive processer.
Indledningsvis består T af p0.

ccp estimater (fra oven) antallet af children af proces p i T.

- Når p sender en grundlæggende besked, ccp ← ccp + 1.
- Lad denne besked modtages af q.
  - Hvis q endnu ikke er i T, slutter det T med parent p og ccq ← 0.
  - Hvis q allerede er i T, sender den en kontrolmeddelelse til p, da den ikke er et nyt child på s. Efter modtagelse af denne meddelelse, ccp ← ccp - 1.

- Når en noninitiator p er passiv og ccp = 0, afslutter den T og informerer sin parent om, at det ikke længere er et child.
- Når initiatoren p0 er passiv og ccp0 = 0, kaldes det Announce.

Ranas algorithm
Tillader en decentraliseret grundlæggende algoritme; kræver et uindrettet netværk.
Hver grundlæggende besked er anerkendt.
Et logisk ur giver (grundlæggende og kontrol) hændelser med et tidsstempel.
Tidsstempel for en proces er det højeste tidsstempel for dets begivenheder hidtil (i første omgang er det 0).

Hvis det på t tid bliver en proces stille, dvs. (1) er den blevet passiv,
og (2) alle basale meddelelser, der er sendt, er blevet anerkendt.
det starter en wave (af kontrol meddelelser), mærket med t (og dens id).
Kun processer, der har været stille fra en tid ≤ t på, deltager i
wave.
Hvis en wave afslutter, ringer initiativtageren Announce.

Antag en wave, mærket med noget t, fuldfører ikke.
Derefter deltager nogle processer p ikke i denne wave.
På grund af denne wave bliver p's logiske tid større end t.
Når p bliver stille, starter den en ny wave, mærket med en t
0> t.

Antag en stille proces q deltager i en wave, og bliver senere gjort aktiv af en grundlæggende besked fra en proces s det var endnu ikke besøgt af denne wave.
Så vil denne bølge ikke fuldføre.
Lad bølgen nemlig mærkes med t.
Når q deltager i wave, bliver dens logiske ur> t.
Ved ack fra q til p, som svar på den grundlæggende besked fra p,
det logiske ur af p bliver> t.
Så p vil ikke deltage i wave(fordi den er mærket med t).

Weight-throwing algoritm:
Kræver en centraliseret grundlæggende algoritme; tillader et rettet netværk.
Initiatoren har vægt 1, alle noninitiators har vægt 0.

Når en proces sender en grundlæggende besked, overfører den en del af dens vægt på denne meddelelse.
Når en proces modtager en grundlæggende besked, tilføjer den vægten af denne besked til sin egen vægt.
Når en noninitiator bliver passiv, vender den tilbage til initiativtageren.
Når initiatoren bliver passiv og har genvundet 1, det kaldes Announce.

Underflow: Vægten af en proces kan blive for lille at blive delt yderligere.

Løsning 1: Processen giver sig ekstra vægt og informerer initiativtageren at der er ekstra vægt i systemet.
En ack fra initiativtager er nødvendig før ekstravægt kan bruges til at undgå raceforhold.

Løsning 2: Processen initierer en Weight-throwing termination påvisning underopkald, og returnerer kun sin vægt til initiativtager når det er blevet passivt og dette underopkald er afsluttet.

Safras algortihm:
Tillader en decentraliseret grundlæggende algoritme og et rettet netværk.
Hver proces opretholder en tæller af type Z; i første omgang er det 0.
Ved hver udgående / indgående grundlæggende besked er tælleren øges / formindskes.
På ethvert tidspunkt er summen af alle tællere i netværket ≥ 0, og det er 0 hvis og kun hvis ingen grundlæggende meddelelser er i transit.
På hver rundrejse bærer token summen af tællerne af de processer, den har gennemgået.

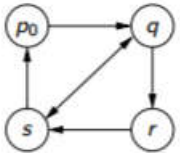
Komplikation: Symbolet kan afslutte en rundrejse med en negativ sum, når en besøgt passiv proces bliver aktiv med en grundlæggende besked, og sender basale beskeder, der modtages af en uvisteret proces.

Processer er farvede hvid eller sort. De er oprindeligt hvide, og en proces, der modtager en grundlæggende besked, bliver sort.

- Når p0 er passiv, sender den et hvidt tegn med tæller 0.
- En noninitiator videresender kun token, når den er passiv.

- Når en sort proces modtager token, bliver processen
  hvid og symbolsk sort.
  Token vil forblive sort for resten af rundrejsen.
- Til sidst vender token tilbage til p0, hvem venter, indtil den er passiv:
  - Hvis symbolet er hvidt og summen af alle tællere er nul,
  p0 opkald Announce.
  - Ellers sender p0 et hvidt tegn igen.

Eksempel:
Symbolet er ved p0; kun s er aktiv; ingen meddelelser er i transit
alle processer er hvide med tæller 0.



s sender en grundlæggende besked m til q, hvor tælleren s er sat til 1.
s bliver passiv.
Token bevæger sig rundt i netværket, hvid med sum 1.
Token går videre til r, hvid med summen 0.
m rejser til q og tilbage til s, hvilket gør dem aktive, sorte, med tæller 0.
s bliver passiv.
Token rejser fra r til p0, sort med summen 0.
q bliver passiv
Efter to mere runde ture af tokenen kalder p0 Announce.

Når systemet er afsluttet,

- token vil farve alle processer hvidt og
- processernes tællere udgør op til nul.

Så token vender så tilbage til initiator hvid med tæller 0.
Antag, at et token vender tilbage til initiator hvid med tæller 0.
Da token er hvid: hvis modtagelse af en besked er inkluderet i tælleren, så sender denne besked også med i tælleren.
Så da tælleren er 0:

- ingen proces blev gjort aktiv efter token besøg, og
- ingen meddelelser er i transit.

Eventuelle forslag til optimering af Safras algoritme?

(Hint: Kan vi gøre væk med sorte tokens?)
Svar: Når en sort proces får token, afviser den token (og bliver hvid).
Når processen bliver passiv, sender den et nyt token, mærket med dens id.