MODIFIED RING-BASED MUTUAL EXCLUSION ALGORITHM WITH FAILOVER RECOVERY[[1]](#footnote-1)

**Eng. Ivaylo Rashkov, BSc – MSc Student**

Department of Computer Systems & Technologies,

University of Ruse “Angel Kanchev”

Tel.: +358 84 661663

E-mail: ivorashkov@abv.bg

**Assoc. Prof. Milen Lukanchevski, PhD**

Department of Computer Systems & Technologies,

University of Ruse “Angel Kanchev”

Рhone: 0877 303 850

Е-mail: mil@ieee.org

**Abstract:** The ring-based aka token-ring mutual exclusion algorithm is the simplest decentralized mutual exclusion algorithm. The only preliminary information any constituent system processes should know are its own identifier and the identifier of its immediate neighbor. The main drawback of this attractive algorithm is strong presumption of full system reliability which is impractical.

The modified version of algorithm proposed in this paper eliminates presumption of full system reliability and guarantees failover recovery from any kind of multiple faults. Two strategies exists – without or with communicational ring reconfiguration.

Main part of the modification is new algorithm for mutual exclusion token control/recovery. It introduces one new initialization message <mrk\_me\_clr> and extension of mutual exclusion token <mrk\_me> with time marker Ti. The time marker is used to control validity of mutual exclusion token and initialization message serves as separator between mutual exclusion token before and after failover recovery.

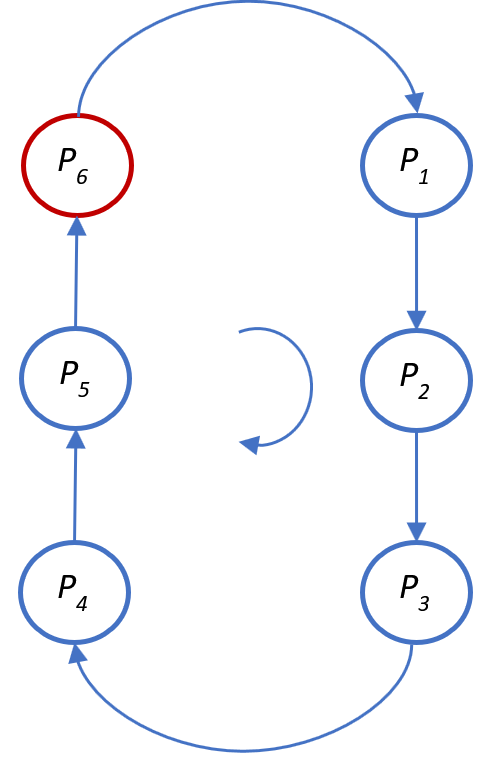
***Keywords:*** *Distributed Systems, Failover Recovery, Distributed Mutual Exclusion, Token-Ring.*

ВЪВЕДЕНИЕ

Под *„взаимно изключване“*се разбира изключване на възможността повече от един активен обект *(процес, нишка)* да се обръщат в даден момент към поделен ресурс (Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G., 2011, Kshemkalyani, A. & M. Singhal. 2008, Lukanchevski, M., 2014). Докато при многозадачните операционни системи тази задача се решава чрез глобални променливи под контрола на ядрото, при разпределените системи решението се основава на идеята за критичната секция. Под критична секция се разбира участък от кода на активния обект, в който се достъпва поделения ресурс.

По отношение на алгоритмите за разпределено взаимно изключване се поставят три условия: безопасност *(safety, ME1)*, осъществимост *(liveness, ME2)* и справедливост *(fairness, ME3)*. Абсолютно задължителни от тях са първите две.

Алгоритмите за разпределено взаимно изключване се делят на две големи фамилии: разрешителни *(permission-based)* и маркерни *(token-based)* (Raynal, M. 1991). Един от популярните маркерни алгоритми е кръговият. При него системата е хомогенна и се състои от n идентични процеса *P1, P2, …, Pn,* свързани в логическа кръгова топология. На фиг. 1 е представена подобна система при *n = 6*. Съществува един единствен маркер *ME* (*token, служебно съобщение*), който се разпространява в кръга *(комуникационния пръстен)* по посока на часовниковата стрелка. Тъй като само процеса, притежаващ маркера *ME*, може да се намира в критичната секция, се удовлетворява условието *ME1*. Условието *ME2* се удовлетворява от кръговата топология. Маркерът *ME* се генерира еднократно от координатора *P6* – процеса с най-голям идентификатор. Координаторът се определя при инициализацията на системата чрез алгоритъм за разпределен избор, като самата подредба на процесите в пръстена е без значение.

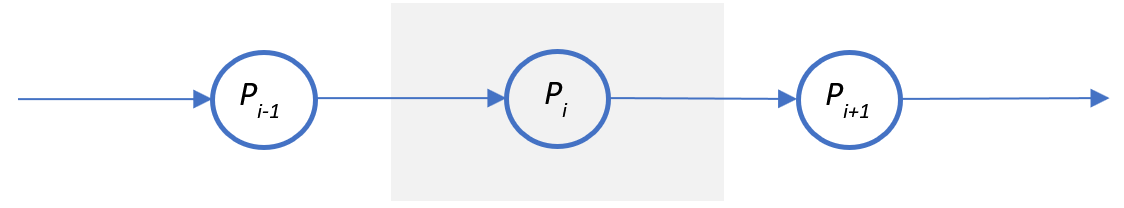


Фиг. 1. Канонично представяне

Основен недостатък на този базов вариант на кръговия алгоритъм за разпределено взаимно изключване е предположението за недопустимостта на откази от никакъв вид - нито на процеси, нито на канали.

Поставена е задачата да се намери подходяща модификация на базовия вариант на кръговия алгоритъм, която да допуска физическо отпадане/възстановяване на който и да е от процесите по всяко време. След възстановяването на отказалия процес, всички останали процеси в кръга продължават нормалното си изпълнение от работното състояние, в което са били в момента на отказа, а маркерът *ME* се възстановява. При това трябва да се гарантира строгото съблюдаване на условията *ME1* и *ME2*. Комуникационният пръстен не се реконфигурира.

В представения доклад се разглеждат някои възлови моменти от модифицирания кръгов алгоритъм *Mx1ME* за разпределено взаимно изключване с възстановяване от отказ във варианта без реконфигуриране на комуникационния пръстен. Алгоритъмът е оригинален и позволява възстановяването от множествени откази.



Фиг. 2. Графична дефиниция на понятието „отказ на процес“

Понятието *„отказ на процес“* се дефинира като отказ на самия процес *Pi*, отказ на участъка от входния канал на Pi или отказ на участъка от изходния канал, за който *Pi* отговаря (фиг. 2).

ИЗЛОЖЕНИЕ

**Алгоритъм за управление/възстановяване на маркера**

Маркерът *ME* е критичен за алгоритъма ресурс. За да се гарантира строгото съблюдаване на условията *ME1* и *ME2* този маркер не трябва да се губи или дублира. Това предполага разширяването на базовия алгоритъм за разпределено взаимно изключване с алгоритъма *MrkME* за управление/възстановяване на маркер *ME*.

Алгоритъмът *MrkME* използва две съобщения: инициализиращ маркер *<mrk\_me\_clr>* и модифициран маркер *ME <mrk\_me, Ti>*. Модификацията на маркер *ME*, спрямо базовия алгоритъм, се заключава в добавянето на характеристичното време *Ti*, което отговаря на момента на създаването на маркер *ME* от координатора.

Използват се следните собствени вътрешни променливи:

- *променливата на състоянието boolMrkMe*;

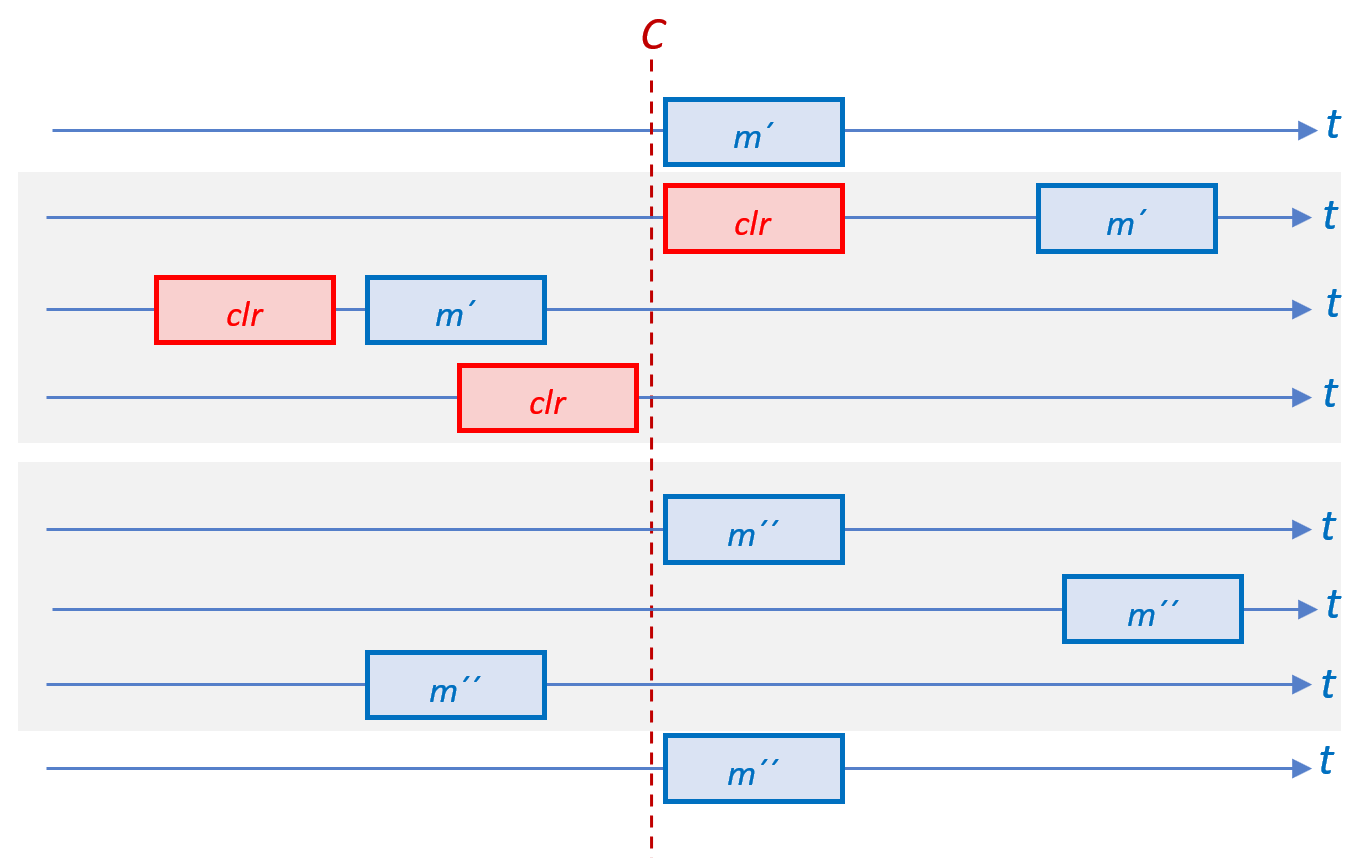
-променливата *timeMrkME*, съдържаща *характеристичното време Ti* на текущия за процеса маркер*;*

- признака *strClrPending* за задържан *инициализиращ маркер*.

Освен това се използва *идентификатора на координатора* *idCoordinator*, дефиниран в алгоритъма за разпределен избор и променливата на състоянието *meState* на алгоритъма *Mx1ME*.

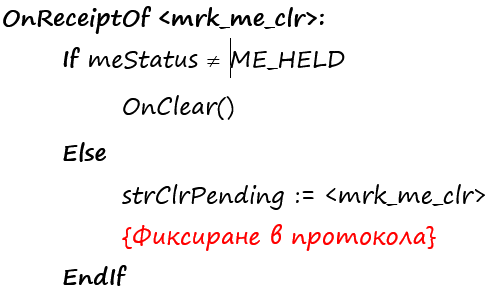
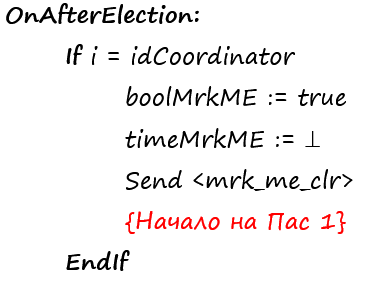
Алгоритъмът *MrkME* се стартира от координатора след приключване на избора и протича в два паса. По време на първия пас координаторът формира инициализиращото съобщение *<mrk\_me\_clr>*. Вторият пас се стартира също от координатора като се формира новия маркер *ME <mrk\_me, Ti>*.

Принципът на работа на алгоритъма *MrkME* във времето е представен на фиг. 3. Преди отказа в пръстена се разпространява маркера *m’*. След отказа и възстановяването на отказалия процес, координаторът C стартира първия пас като формира инициализиращото съобщение *clr*. Това съобщение се разпространява след маркера *m’* и анулира неговото по-нататъшно действие. Щом маркер *m’* постъпи отново в координатора бива погълнат от него. В някакъв следващ момент в координатора постъпва и инициализиращото съобщение *clr*, което е признак за анулиране действието на маркера *m’* в целия кръг и първият пас приключва. Координаторът C стартира втория пас като формира новия маркер *ME m’’*. Преминавайки последователно по кръга, този маркер се фиксира от всеки от процесите като нов маркер *ME*. При първото последващо постъпване на маркер *ME m’’* в координатора завършва и вторият пас. Така завършва възстановяването на маркер *ME* в цялата система и се възстановява работоспособността на разпределения кръгов алгоритъм за взаимно изключване.



Фиг. 3. Условно представяне на принципа на работа

Формалната спецификация на алгоритъма *MrkME* съдържа и псевдокода на множеството събития: *OnInit, OnAfterElection, OnReceiptOf <mrk\_me\_clr>, OnClear, OnReceiptOf <mrk\_me, Ti>*, например



Събитието *OnInit* сработва при първоначалната инициализация на процеса, докато събитието *OnAfterElection* – при всяко приключване на разпределения избор. Разпределеният избор от своя страна се стартира не само след първоначалната инициализация на процеса, но и след възстановяването на отказалия процес, а това води до последователно включване на всички системни процеси в избора. Това гарантира, че независимо дали отказалият процес е бил координатора или не, алгоритъмът *MrkME* ще се стартира.

При получаването на инициализиращото съобщение сработва събитието *OnReceiptOf <mrk\_me\_clr>*. Интересна негова особеност е задържането на инициализиращото съобщение *<mrk\_me\_clr>* в процеса, намиращ се в критичната секция *(meStatus = ME\_HELD)*. Едва след като процесът излезе от секцията *(meStatus = ME\_RELEASED)* и освободи текущия маркер (m’ от фиг. 3), инициализиращото съобщение се препредава по кръга. Така, това съобщение формира граница между стария и новия маркер *ME* и предотвратява дублирането му.

**Модифициран кръгов алгоритъм за разпределено взаимно изключване**

Базовият вариант на кръговия алгоритъм за разпределено взаимно изключване, описан в литературата (Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G., 2011, Kshemkalyani, A. & M. Singhal. 2008, Lukanchevski, M., 2014) е модифициран с оглед интеграцията му с алгоритъма *MrkME*. С това се преодолява основният недостатък на базовия вариант - предположението за недопустимостта на отказите.

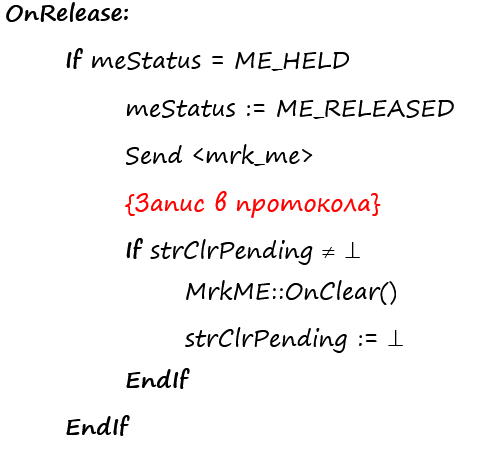
Интеграцията с алгоритъма *MrkME* предполага използването на двете съобщения: инициализиращ маркер *<mrk\_me\_clr>* и модифициран маркер *ME <mrk\_me, Ti>*. Освен собствената вътрешна променлива на състоянието meState се използва идентификатора на координатора *idCoordinator*, дефиниран в алгоритъма за разпределен избор, както и вътрешните променливи на алгоритъма *MrkME* - *boolMrkMe*, *timeMrkME* и *strClrPending*.

Запазено е множеството събития на базовия кръгов алгоритъм: *OnInit*, *OnAfterElection*, *OnEnter*, *OnReceiptOf <mrk\_me>*, *OnRelease*. При *OnEnter* и *OnReceiptOf* *<mrk\_me>* обаче се забранява смяната на работното състояние при невалиден маркер *ME*.

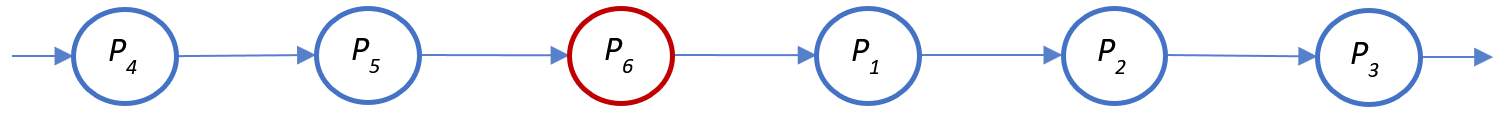
Добавено е събитието *OnRingUp*, което отговаря на момента на откриване изграждането на комуникационния пръстен от процеса *Pi*. Това е възлов момент, тъй като от него се стартира алгоритъма за разпределен избор.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Преработката на събитието *OnRelease* е по-различна, заради необходимостта от задържане на инициализиращото съобщение *<mrk\_me\_clr>* в процеса, намиращ се в критичната секция *(meStatus = ME\_HELD)*. В случая на процеса винаги се разрешава да излезе от секцията като освободи текущия маркер (m’ от фиг. 3). След това се проверява за наличието на задържано инициализиращо съобщение и ако има такова, то се препредава по кръга.

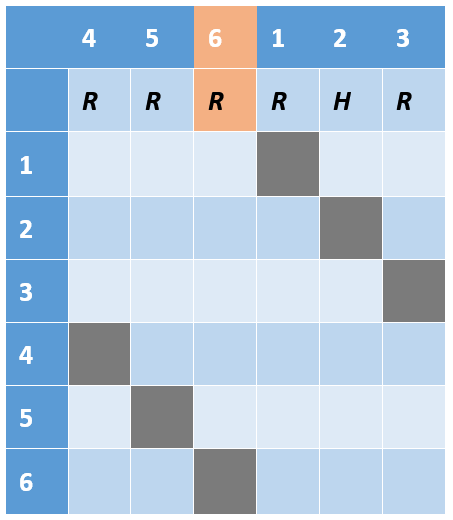
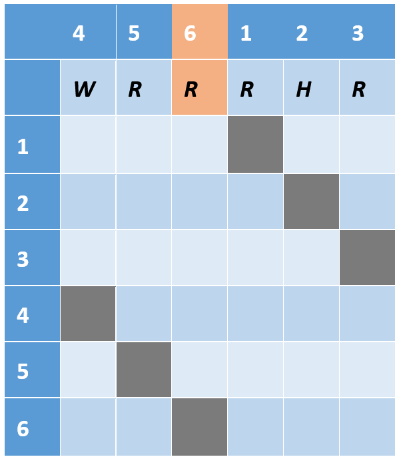
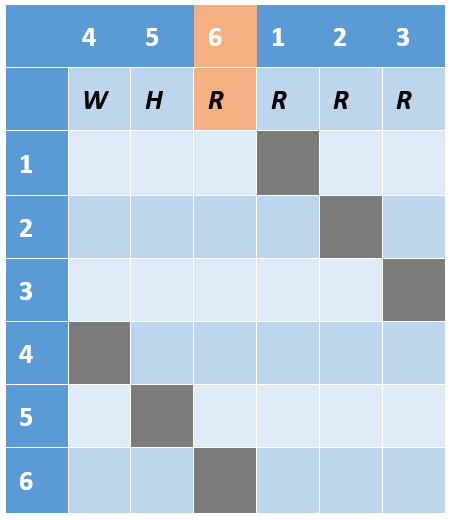


За да се обхванат възможните сценарии на отказите се извършва развиване на кръговата топология в линейна, като стремежът е координаторът да е в средата на последователността от процеси (фиг. 4).



Фиг. 4. Разгънато представяне

Това дава възможност за компактно кодиране на обобщеното състояние на всички процеси в кръга в момента на възникване на отказа, например: *RRRRRR* (всички процеси са в *ME\_RELEASED)*, *RRRRHR* *(P2* е в *ME\_HELD*), *WRRRHR* (*P4* е в *ME\_WANTED, P2* е в *ME\_HELD*), *WRRRHW (P4* и *P3* са в *ME\_WANTED, P2* е в *ME\_HELD), WHRRRR (P4* е в *ME\_WANTED, P5* е в *ME\_HELD).*

Фиг. 5.Таблично представяне на сценариите RRRRHR, WRRRHR и WHRRRR

При предположението за единични откази се получават табличните представяния на сценариите от фиг. 5, където по колони са номерата на процесите, а по редове е маркиран отказалия процес. Предложената таблична форма може да се разшири и за случаите на множествени откази.

ИЗВОДИ

Представеният модифициран кръгов алгоритъм за разпределено взаимно изключване с възстановяване от отказ е реализиран със средствата на средата Embarcadero C++ Builder® 10.2. Реализацията е ориентирана както към демонстрация на работата на алгоритъма, така и към евентуалното му практическо използване. Извършени са тестове на всички възможни сценарии на единични и двойни откази като са фиксирани диагностическите протоколи на отделните процеси.

REFERENCES

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011). *Distributed Systems: Concepts and Design*. 5th Ed. – Boston: Addison-Wesley, p. 1008.

Kshemkalyani, A. & M. Singhal. (2008) Distributed Computing: *Principles, Algorithms, and Systems.* – Cambridge: Cambridge University Press, p. 736.

Lukanchevski, M. (2014). *Razpredeleni sistemi i algoritmi: teoria i praktika*. Ruse: Izdatelski tsentar na RU „A. Kanchev”, ISBN 978-619-7071-35-1, p. 213.

Raynal, M. (1991) *A simple taxonomy for distributed mutual exclusion algorithms.* // ACM SIGOPS Operating Systems Review Volume 25 Issue 2, April, p. 47-50.

1. Докладът е представен на студентската научна сесия на 27.04.2018 в секция“Комуникационна и компютърна техника и технологии“ с оригинално заглавие на български език: МОДИФИЦИРАН КРЪГОВ АЛГОРИТЪМ ЗА ВЗАИМНО ИЗКЛЮЧВАНЕ С ВЪЗСТАНОВЯВАНЕ ПРИ ОТКАЗ. [↑](#footnote-ref-1)