**Mutual Exclusion Token Management/Recovery Algorithm MrkME**

(Generalized Formal Specification[[1]](#footnote-1))

**Pi::MrkME**

**{SYNOPSIS}**

The MrkME algorithm is started by the coordinator after the selection is completed and runs in two passes.

During the first pass, the coordinator generates the initialization message <mrk\_me\_clr>. This pass completes when the coordinator receive back the same message.

The second pass is also started by the coordinator immediately after the end of the first pass, forming the new ME token <mrk\_me, Ti>. when the coordinator receives back the same message.

In the rest (conventional) processes the MrkME algorithm begins with the receipt of <mrk\_me\_clr> message and continues until <mrk\_me, Tj> message reception under condition timeMrkME=NULL.

In this period the processes should not change their current state meState, which is controlled in the modified version of the distributed ring-based ME algorithm Mx1ME.

The exception is the process that probably is in ME\_HELD. When such a process receives the <mrk\_me\_clr> message that process holds the message.

When exiting the section the conventional process first releases the current token and right after that processes the <mrk\_me\_clr> message and releases it. Otherwise, if the process is the coordinator when exiting the section processes only the hold <mrk\_me\_clr> message.

**{ASSUMPTIONS}**

The MrkME algorithm is valid if the following conditions are met:

* The type of process failures is strongly “fail-stop”.

Multiple concurrent initiations are allowed.

Failures are allowed during the execution.

**ALGORITHM 1**: Declarative Part of *Pi::MrkME*

**{SYSTEM CONSTANTS}**

String MRK\_ME\_CLR // type of message “ME Clear” (Pass 1)

String MRK\_ME // type of message “ME token” (Pass 2)

PId i // process Pi identifier

**{MESSAGES}**

<mrk\_me\_clr>

<mrk\_me, Ti>

**{SET OF STATES}**

<State> := {OFF, ON}

**{INTERNAL STATE SPACE}**

State state // current process MrkME state

Time timeMrkME // characteristic token timeTi

String strClrPending // retained token <mrk\_me\_clr>

ME::state // current process ME state

E::pidCoordinator // “coordinator” process identifier

**ALGORITHM 2**: Event Handlers of *Pi::MrkME*

**OnInit:**

state := OFF

timeMrkME := NULL

strClrPending := NULL

**OnAfterElection:**

**If** i = E::pidCoordinator

state := ON

timeMrkME := NULL

Send <mrk\_me\_clr>

*{Start of Pass 1}*

**End If**

**OnReceiptOf <mrk\_me\_clr>:**

**If** Mx1ME::meStatе ≠ ME\_HELD

OnClear()

**Else**

strClrPending := <mrk\_me\_clr>

**End If**

**OnClear:**

**If** i = Е::pidCoordinator

*{Global End of Pass 1}*

timeMrkME := Ti

Send <mrk\_me, Ti>

*{Start of Pass 2}*

**Else**

state := ON

timeMrkME := NULL

Send <mrk\_me\_clr>

*{Local Begin of the MrkME Algorithm}*

**End If**

**OnReceiptOf <mrk\_me, Tj> ∩ (MrkME::state = ON):**

**If** i = Е::pidCoordinator

**If** timeMrkME = Tj

Send <mrk\_me, Tj>

*{Global End of Pass 2}*

**EndIf**

**Else**

**If** timeMrkME = NULL

state := OFF

timeMrkME := Tj

Send <mrk\_me, Tj>

*{Local End of Pass 2}*

**Else If** timeMrkME = Tj

Send <mrk\_me, Tj>

**EndIf**

**EndIf**

1. Implementation <https://github.com/milphaser/XME.Ring> [↑](#footnote-ref-1)