**Modified Distributed Ring-based Mutual Exclusion Algorithm Mx1ME**

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

**Pi::Mx1ME**

**{SYNOPSIS}**

The basic distributed ring-based mutual exclusion algorithm described in the literature is modified to integrate it with the Token Management/Recovery Algorithm MrkME. This overcomes the main drawback of the basic algorithm - the assumption of inadmissibility of failures.

**{ASSUMPTIONS}**

The Mx1ME algorithm is valid if the following conditions are met:

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

Physical failure/recovery of any of the system processes is allowed at any time.

After recovery of the failed process, all other processes in the ring continue their normal execution from the operating state they were in at the time of the failure, and only one instance of ME marker is restored.

**ALGORITHM 1**: Declarative Part of *Pi::Mx1ME*

**{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> := {INIT, RELEASED, WANTED, HELD}

**{INTERNAL STATE SPACE}**

State state // current process ME state

E::pidCoordinator // “coordinator” process identifier

MrkME::state // current process MrkME state

MrkME::timeMrkME // characteristic token timeTi

MrkME::strClrPending // hold message <mrk\_me\_clr>

**ALGORITHM 2**: Event Handlers of *Pi::Mx1ME*

**OnInit:**

state := INIT

E::pidCoordinator := NULL

**OnRingUp:**

E::OnStart()

**OnAfterElection:**

**If** i = E::pidCoordinator

StartMarker()

**End If**

state := RELEASED

**OnEnter:**

**If** state = RELEASED

**If** MrkME::timeMrkME ≠ NULL

state := WANTED

**End If**

**End If**

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

**If** state = WANTED

**If** MrkME::timeMrkME ≠ NULL

state := HELD

*{Entrance into the Critical Section}*

*{access()}*

**End If**

**Else If** state = RELEASED

Send <mrk\_me, Tj>

**End If**

**OnRelease:**

**If** state = HELD

state := RELEASED

**If** MrkME::strClrPending ≠ NULL

MrkME::OnClear()

MrkME::strClrPending := NULL

**Else**

Send <mrk\_me, Tj>

**End If**

**End If**

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