**Communication Ring Check Up Algorithm RUP**

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

**Pi::RUP**

**{SYNOPSIS}**

Distributed algorithm for communication ring integrity check up.

It is started just after opening of the output channel of the process which connected it with the immediate neighbor.

**{ASSUMPTIONS}**

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

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

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

**{SYSTEM CONSTANTS}**

Int MAX\_RUP\_PERIOD // period to next check

String MRK\_RUP // message type „RUP Token“

PId i // process Pi identifier

**{MESSAGES}**

<mrk\_rup, i>

**{SET OF STATES}**

<State> := {INIT, DOWN, UP}

**{INTERNAL STATE SPACE}**

State state // current process RUP state

Timer TimerRUP // timer

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

**OnInit:**

state := INIT

TimerRUP.Interval := MAX\_RUP\_PERIOD

**OnOutputConnect:**

*{Ring Check Up First Attempt}*

state := DOWN

Send <mrk\_rup, i>

TimerRUP.Start()

**OnOutputDisconnect:**

state := DOWN

**OnOutputError:**

state := DOWN

**OnReceiptOf <mrk\_rup, j>:**

**If** j = i

TimerRUP.Stop()

state := UP

*{Distributed Election Entry Point}*

E::OnStartElection()

**Else**

Send <mrk\_rup, j>

**End If**

**OnTimer:**

*{Ring Check Up Next Attempt}*

TimerRUP.Stop()

Send <mrk\_rup, i>

TimerRUP.Start()

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