АЛГОРИТЪМ ЗА ОТКРИВАНЕ НА МОМЕНТА НА ИЗГРАЖДАНЕ НА КОМУНИКАЦИОННИЯ ПРЪСТЕН

Communication Ring Check Up Algorithm *RUP.AutoList.GD*

(Formalized Specification)

**Pi::RUP.AutoList.GD**

**{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.**

**The “AutoList” variant of this algorithm automatically fills in the list of processes’ ids. It does not require this list to be known in advance.**

**{ASSUMPTIONS}**

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

* **The distributed system is synchronous.**
* **The type of process failures is strongly “*fail-stop*”, but** not **all failures are allowed if AutoList mode is selected: only failures of other processes (j ≠ i) and only after finishing of the algorithm for *Pi* are allowed in this mode.**

**{SYSTEM CONSTANTS}**

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

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

String MRK\_RUP2 // message type „RUP AutoList Token“

CEH::i // process Pi identifier

CEH::j // default neighbor process Pj

CEH::AutoList // <true> if ListPIds is filled during RUP

// <false> if ListPIds is known in advance

**{MESSAGES}**

<mrk\_rup, i, list>

<mrk\_rup2, i, list>

**{SET OF STATES}**

<State> := {RUP\_INIT, RUP\_DOWN, RUP\_UP}

**{INTERNAL STATE SPACE}**

State state // current process RUP state

Timer TimerRUP // timer

CEH::ListPIds // list of process identifiers

CEH::PIdNext // current neighbor process identifier

**{EVENTS}**

**OnInit:**

state := RUP\_INIT

TimerRUP.Interval := MAX\_RUP\_PERIOD

**OnOutputConnect:**

{*Ring Check Up First Attempt*}

state := RUP\_DOWN

**If** CEH::AutoList = true

Send <mrk\_rup2, i, list.Clear()>

**Else**

Send <mrk\_rup, i>

**EndIf**

TimerRUP.Start()

**OnOutputDisconnect:**

state := RUP\_DOWN

**OnOutputError:**

state := RUP\_DOWN

**OnReceiptOf <mrk\_rup2, j, list>:**

**If** j = i

TimerRUP.Stop()

state := RUP\_UP

ListPIds := list

{*Distributed Election Entry Point*}

E::OnStartElection()

**Else**

**If** CEH::AutoList = true

Send <mrk\_rup2, j, list.Add(i)>

**Else**

Send <mrk\_rup, i>

**EndIf**

**EndIf**

**OnTimer:**

{*Ring Check Up Next Attempt*}

TimerRUP.Stop()

**If** CEH::AutoList = true

Send <mrk\_rup2, i, list.Clear()>

**Else**

Send <mrk\_rup, i>

**EndIf**

TimerRUP.Start()