АЛГОРИТЪМ ЗА ОТКРИВАНЕ НА МОМЕНТА НА ИЗГРАЖДАНЕ НА КОМУНИКАЦИОННИЯ ПРЪСТЕН

Communication Ring Check Up Algorithm *RUP*

(Formalized Specification)

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

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

**{SYSTEM CONSTANTS}**

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

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

CEH.PId i // process Pi identifier

**{MESSAGES}**

<mrk\_rup, i>

**{SET OF STATES}**

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

**{INTERNAL STATE SPACE}**

State state // current process state

Timer TimerRUP // timer

**{EVENTS}**

**OnInit:**

state := RUP\_INIT

TimerRUP.Interval := MAX\_RUP\_PERIOD

**OnOutputConnect:**

{*Ring Check Up First Attempt*}

state := RUP\_DOWN

Send <mrk\_rup, i>

TimerRUP.Start()

**OnOutputDisconnect:**

state := RUP\_DOWN

**OnOutputError:**

state := RUP\_DOWN

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

**If** j = i

TimerRUP.Stop()

state := RUP\_UP

{*Distributed Election Entry Point*}

E::OnStartElection()

**Else**

Send <mrk\_rup, j>

**EndIf**

**OnTimer:**

{*Ring Check Up Next Attempt*}

TimerRUP.Stop()

Send <mrk\_rup, i>

TimerRUP.Start()