КРЪГОВ АЛГОРИТЪМ ЗА РАЗПРЕДЕЛЕН ИЗБОР

Ring-Based Distributed Election  
(Formalized Specification)

**Pi::E**

**{SYNOPSIS}**

**Basic variant of the algorithm for distributed election over logical communication ring.**

**{ASSUMPTIONS}**

**No failures of any kind (neither processes nor channels) are allowed, by condition.**

**However, the algorithm is not sensitive to failures - in the event of a failure and subsequent recovery, the algorithm restarts and the coordinator is established again. Moreover, both requirements for this class of distributed algorithms are observed - E1 (safety) and E2 (feasibility, liveness).**

**{SYSTEM CONSTANTS}**

String MSG\_ELECTION // type of message “election”

String MSG\_ELECTED // type of message “elected”

**{MESSAGES}**

<msg\_election, i>

<msg\_elected, i>

**{SET OF STATES}**

<State> := {E\_NON\_PARTICIPANT, E\_PARTICIPANT}

**{INTERNAL STATE SPACE}**

State eState // current process state

Id i // process **Pi** identifier

Id idCoordinator // “coordinator” process identifier

**{EVENTS}**

**OnInit:**

eStatus := E\_NON\_PARTICIPANT

idCoordinator := ⊥

**OnStartElection:**

eStatus := E\_PARTICIPANT

idCoordinator := ⊥

Send <msg\_election, i>

**OnReceiptOf <msg\_election, j>:**

**If** j > i

eStatus := E\_PARTICIPANT

idCoordinator := ⊥

Send <msg\_election, j>

**Else If** j < i

**If** eStatus = E\_NON\_PARTICIPANT

eStatus := E\_PARTICIPANT

idCoordinator := ⊥

Send <msg\_election, i>

**EndIf**

**Else**

eStatus := E\_NON\_PARTICIPANT

idCoordinator := i

Send <msg\_elected, i>

**EndIf**

**OnReceiptOf <msg\_elected, j>:**

**If** j ≠ i

eStatus := E\_NON\_PARTICIPANT

idCoordinator := j

Send <msg\_elected, j>

{Local End}

ME::OnAfterElection()

**Else**

{Global End}

ME::OnAfterElection()

**EndIf**