MODULE *DoRiS*

Naturals, *Reals*, *Sequences* and *TLC* are TLA+ modules (libraries) imported by the specification.

EXTENDS *Naturals*, *Reals*, *Sequences*, *TLC*

VARIABLES

Shared: distributed global state variable shared by the servers made up of six fields :
– *soft*: the group membership of the soft ring.
– *chipTimer*: an increasing and cyclic timer that range from 0 to *deltaChip*.
– *chipCount*: an increasing and cyclic counter that range from 0 to *nServ*.
– *cycleCount*: an increasing and cyclic counter that range from 0 to horizon.
– *medium*: If no message is being transmitted, medium equals {}. Else, medium stores the message being transmitted.
– *macTimer*: a decreasing timer that specifies the time remaining in order to complete an on-going message transmission. As a special case, $macTimer = 0 \Rightarrow medium = \{\}$

HardState: local state variable (tuple) associated to the hard ring made up of four fields : – *msg*: list of hard messages stored in local buffers after their
reception by the network device.
– *execTimer*: decreasing timer that specifies the time remaining in order to complete the processing of a hard message to *nServ*.
– *res*: reservation list for the *nServ* next chips.
– *cons*: counter that represents the number of elementary messages received in a complete *DoRiS* cycle.

SoftState: local state variable (tuple) associated to the soft ring made up of three fields :
– *token*: counter used to rule the token circulation during a soft window.
– *list*: list of soft messages waiting to be transmitted.
– *count*: the number of soft messages received during a soft window .

History: an observer global state variable made up of two fields, used to check specific temporal properties :
– *elem*: the number of elementary messages sent in a cycle.
– *rese*: the number of reservation messages sent in a cycle.

VARIABLES *Shared*, *HardState*, *SoftState*, *History*

CONSTANTS (Their values are defined in the file *DoRiS.cfg*)

– *nServ*: number of *DoRiS* servers.
– *deltaChip*: duration of a chip.
– *delta*: transmission time of a hard message (smallest *Ethernet* frame – 64 bytes).
– *pi*: processing time of a hard message by the slowest device.
– *maxTxTime*: transmission time of the largest *Ethernet* frame (1500 bytes).
– *horizon*: Number of total cycles used to check a *DoRiS* model.

CONSTANTS *nServ*, *deltaChip*, *delta*, *pi*, *maxTxTime*, *horizon*

(*Continue*)

Miscellaneous definitions
– $min(S)$: function that returns the minimum value of S.
– $next(i, S)$: function that returns the clothest value from i in ring S.
– *ServID*: set of servers identifiers.
– *HardServ* and *SoftServ*: server local threads dealing with the hard and soft communication.
– *HardRing*: hard ring.
– $SoftRing(S)$: function that, given a set S of identifiers, returns the soft ring membership.
– $hardID(s)$: function that, given a hard server thread s, returns the identifier of that thread.
– $softID(s, S)$: function that, given a soft server thread s and a soft membership S, returns the identifier of that thread.
– *vars*: the set of all variables of the specification

$$min(S) \triangleq \text{CHOOSE } m \in S : \forall\, y \in S : m \leq y$$

$$next(i, R) \triangleq$$
$$\text{IF } \forall\, j \in R : j \leq i \text{ THEN } min(R) \text{ ELSE } min(\{j \in R : j > i\})$$

$$ServID \triangleq 1 \,..\, nServ$$
$$HardServ \triangleq [i \in ServID \mapsto \langle \text{“Hard”}, i \rangle]$$
$$SoftServ \triangleq [j \in ServID \mapsto \langle \text{“Soft”}, j \rangle]$$
$$HardRing \triangleq \{HardServ[i] : i \in ServID\}$$
$$SoftRing(S) \triangleq \{SoftServ[j] : j \in S\}$$

$$hardID(s) \triangleq \text{CHOOSE } i \in ServID : HardServ[i] = s$$
$$softID(s, S) \triangleq \text{CHOOSE } j \in S : SoftServ[j] = s$$

$$vars \triangleq \langle Shared, SoftState, HardState, History \rangle$$

Four alternative soft message list fabric (prefixed definition are not used when model-checking).

$$list(j, cycle) \triangleq$$
$$\text{CASE } j \in \{cycle\} \to [i \in 1 \,..\, 4 \mapsto [txTime \mapsto maxTxTime]]$$
$$\square\ j \in \{2, 3\} \setminus \{cycle\} \to \langle\rangle$$
$$\square\ j \in \{4, 7\} \setminus \{cycle\} \to [i \in 1 \,..\, 2 \mapsto [txTime \mapsto maxTxTime]]$$
$$\square\ \text{OTHER} \to 1 :> [txTime \mapsto maxTxTime]$$

$$A\_list(j) \triangleq 1 :> [txTime \mapsto maxTxTime]$$

$$B\_list(j) \triangleq \text{IF } j = 1$$
$$\text{THEN } 1 :> [txTime \mapsto maxTxTime] \,@@\, 2 :> [txTime \mapsto maxTxTime] \,@@$$
$$3 :> [txTime \mapsto maxTxTime]$$
$$\text{ELSE IF } j = 2 \text{ THEN } \langle\rangle \text{ ELSE } 1 :> [txTime \mapsto maxTxTime]$$

$$C\_list(j, cycle) \triangleq$$
$$\text{CASE } j \in \{cycle\} \to [i \in 1 \,..\, 4 \mapsto [txTime \mapsto maxTxTime]]$$
$$\square\ \text{OTHER} \to \langle\rangle$$

(*Continue*)

Initializations of the variables. The *Shared* variable represents the common knowledge of each server.

$$
\begin{array}{l}
Init \triangleq \\
\quad \wedge Shared = [soft \mapsto 1 \,..\, nServ, \\
\qquad\qquad chipTimer \mapsto 0, \\
\qquad\qquad chipCount \mapsto 1, \\
\qquad\qquad cycleCount \mapsto 1, \\
\qquad\qquad macTimer \mapsto 0, \\
\qquad\qquad medium \mapsto \{\}] \\
\quad \wedge HardState = [i \in ServID \mapsto [msg \mapsto \langle\rangle, \\
\qquad\qquad\qquad\qquad\qquad\qquad execTimer \mapsto Infinity, \\
\qquad\qquad\qquad\qquad\qquad\qquad res \mapsto [j \in ServID \mapsto -1], \\
\qquad\qquad\qquad\qquad\qquad\qquad cons \mapsto [j \in ServID \mapsto 1]]] \\
\quad \wedge SoftState \;\; = [j \in Shared.soft \mapsto [token \mapsto 1, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad list \mapsto list(j,\, 1), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad count \mapsto 0]] \,@@ \\
\qquad\qquad\qquad\quad [j \in ServID \setminus Shared.soft \mapsto [token \mapsto -1, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad list \mapsto list(j,\, 1), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad count \mapsto 0]] \\
\quad \wedge History = [elem \mapsto 0, \\
\qquad\qquad\qquad\quad rese \;\mapsto 0]
\end{array}
$$

The elementary sending action. *reservation*(i) is an arbitrary reservation function that states that task "i" wants to reserve as much slots as possible.

$$
\begin{array}{l}
reservation(i) \triangleq \\
\quad \text{IF } \forall\, j \in ServID \quad\;\; : HardState[i].cons[j] = 1 \\
\quad \text{THEN } \{j \in ServID : HardState[i].res[j] = -1\} \\
\quad \text{ELSE } \{(((i-1) + (nServ-1)) \;\%\; nServ) + 1\}
\end{array}
$$

(*Continue*)

$$
\begin{array}{l}
\mathit{ElemSend}(s) \triangleq \\
\quad \land\ \mathit{Shared.medium} = \{\} \\
\quad \land\ \mathit{Shared.chipTimer} = 0 \\
\quad \land\ \text{LET } i \triangleq \mathit{hardID}(s) \\
\qquad\quad \mathit{flag} \triangleq \text{IF } \mathit{SoftState}[i].\mathit{list} \neq \langle\rangle \text{ THEN } 1 \text{ ELSE } 0 \\
\qquad \text{IN} \quad \land\ \mathit{Shared.chipCount} = i \\
\qquad\quad \land\ \text{LET } \mathit{resSet} \triangleq \mathit{reservation}(i) \\
\qquad\qquad \text{IN} \quad \land\ \mathit{Shared}' = [\mathit{Shared} \text{ EXCEPT} \\
\qquad\qquad\qquad\qquad !.\mathit{macTimer} = \mathit{delta}, \\
\qquad\qquad\qquad\qquad !.\mathit{medium} = \{[\mathit{id} \mapsto i,\ \mathit{type} \mapsto \text{“hard”}, \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{res} \mapsto \mathit{resSet},\ \mathit{softFlag} \mapsto \mathit{flag}]\}] \\
\qquad\qquad\qquad\quad \land\ \mathit{HardState}' = [\mathit{HardState} \text{ EXCEPT} \\
\qquad\qquad\qquad\qquad ![i].\mathit{cons}[i] = 1, \\
\qquad\qquad\qquad\qquad ![i].\mathit{res} = [j \in \mathit{ServID} \mapsto \text{IF } j \in \mathit{resSet} \text{ THEN } i \text{ ELSE } @[j]]] \\
\qquad\qquad\qquad\quad \land\ \mathit{SoftState}' = [\mathit{SoftState} \text{ EXCEPT} \\
\qquad\qquad\qquad\qquad ![i].\mathit{token} = \text{IF } \mathit{flag} = 0 \text{ THEN } -1 \text{ ELSE } @] \\
\qquad\qquad\qquad\quad \land\ \mathit{History}' = [\mathit{History} \text{ EXCEPT } !.\mathit{elem} = @ + 1]
\end{array}
$$

The reservation sending action.

$$
\begin{array}{l}
\mathit{ReseSend}(s) \triangleq \mathit{Shared.medium} = \{\} \\
\quad \land\ \mathit{Shared.chipTimer} = \mathit{delta} \\
\quad \land\ \text{LET } i \triangleq \mathit{hardID}(s) \\
\qquad \text{IN} \quad \land\ \mathit{HardState}[i].\mathit{res}[\mathit{Shared.chipCount}] = i \\
\qquad\quad \land\ \mathit{Shared}' = [\mathit{Shared} \text{ EXCEPT} \\
\qquad\qquad\qquad !.\mathit{macTimer} = \mathit{delta}, \\
\qquad\qquad\qquad !.\mathit{medium} = \{[\mathit{id} \mapsto i,\ \mathit{type} \mapsto \text{“hard”},\ \mathit{res} \mapsto \{-1\}]\}] \\
\qquad\quad \land\ \mathit{HardState}' = [j \in \mathit{ServID} \mapsto [\mathit{HardState}[j] \text{ EXCEPT} \\
\qquad\qquad\qquad !.\mathit{res}[\mathit{Shared.chipCount}] = -1]] \\
\qquad\quad \land\ \mathit{History}' = [\mathit{History} \text{ EXCEPT } !.\mathit{rese} = @ + 1] \\
\qquad\quad \land\ \text{UNCHANGED } \mathit{SoftState}
\end{array}
$$

The soft window action.
“Failed” specifies the set of omission failures that take place as a function of “*Shared.chipCount*”. *tailList*(*msgList*): function that remove the first message of *msgList* (if *msgList* is not empty). *lenMsg*(i): function that returns the time needed to transmit the first message of server “i” list of pending soft messages, if this list is not empty. Else, it returns “delta”.

$$
\begin{array}{l}
\mathit{Failed} \triangleq \text{CASE } \mathit{Shared.chipCount} = 2 \to \{3\} \\
\qquad\qquad\quad \square\ \mathit{Shared.chipCount} \in \{3, 4\} \to \{3, 5\} \\
\qquad\qquad\quad \square\ \mathit{Shared.chipCount} = 5 \to \{3, 5\} \\
\qquad\qquad\quad \square\ \text{OTHER } \to \{\}
\end{array}
$$

$$\mathit{A\_Failed} \triangleq \{\}$$

(*Continue*)

$tailList(msgList) \triangleq$
IF $msgList = \langle\rangle$ THEN $msgList$ ELSE $Tail(msgList)$

$lenMsg(i) \triangleq$
IF $SoftState[i].list \neq \langle\rangle$ THEN $Head(SoftState[i].list).txTime$ ELSE $delta$

$SoftSend(s) \triangleq$
$\land\ Shared.medium = \{\}$
$\land\ 2 * delta \leq Shared.chipTimer \land Shared.chipTimer \leq deltaChip$
$\land$ LET $i \triangleq softID(s, Shared.soft)$
$lenTX \triangleq lenMsg(i)$
$d \triangleq Shared.chipTimer + lenTX$
$consis \triangleq \forall j \in ServID : HardState[i].cons[j] = 1$
$wait \triangleq (d > deltaChip) \lor (\neg consis)$
$noMsg \triangleq (i \in Failed) \lor wait$
IN $\land\ i = SoftState[i].token$
$\land\ Shared' = [Shared$ EXCEPT
$!.macTimer =$ IF $noMsg$ THEN $Infinity$ ELSE $lenTX$,
$!.medium =$ IF $noMsg$ THEN @ ELSE $\{[id \mapsto i, type \mapsto$ "soft"$]\}]$
$\land\ SoftState' = [SoftState$ EXCEPT
$![i].list =$ IF $wait$ THEN @ ELSE $tailList(@)$,
$![i].token =$ CASE $wait \rightarrow @$
$\square\neg consis \rightarrow -1$
$\square$OTHER $\rightarrow next(i, Shared.soft)$,
$![i].count =$ IF $noMsg$ THEN @ ELSE $@ + 1]$
$\land$ UNCHANGED $\langle HardState, History\rangle$

The hard and soft messages receiving actions. The hard and soft messages receiving actions. $NoRecvSet(m)$: function that defines omission failures of the "m" reception action. $tokenUpdate(m)$: function that updates the "token" counter.

$NoRecvSet(m) \triangleq$ IF $Shared.chipCount \in \{2, 4\}$ THEN $\{m.id, 2\}$ ELSE $\{m.id\}$

$C\_NoRecvSet(m) \triangleq$ IF $Shared.chipCount \in \{2, 4\}$ THEN $\{m.id, 3\}$ ELSE $\{m.id\}$

$A\_NoRecvSet(m) \triangleq$ LET $i \triangleq$ CHOOSE $j \in ServID :$ TRUEIN $\{m.id, i\}$

$B\_NoRecvSet(m) \triangleq \{m.id\}$

$tokenUpdate(m) \triangleq SoftState' =$
$[i \in (ServID \setminus Shared.soft) \cup NoRecvSet(m) \mapsto SoftState[i]]$ @@
$[i \in Shared.soft \setminus NoRecvSet(m) \mapsto [SoftState[i]$ EXCEPT
$!.token =$ IF $@ = m.id$ THEN $next(m.id, Shared'.soft)$ ELSE $@]]$

(*Continue*)

$HardRecv(m) \triangleq$
  $\land$ CASE $m.res \neq \{-1\} \land m.softFlag = 1$
      $\rightarrow$ $\land Shared' = [Shared$ EXCEPT $!.medium = \{\},\ !.soft = @ \cup \{m.id\}]$
        $\land$ UNCHANGED $SoftState$
    $\square$ $m.res \neq \{-1\} \land m.softFlag = 0$
    $\rightarrow$ $\land Shared' = [Shared$ EXCEPT $!.medium = \{\},\ !.soft = @ \setminus \{m.id\}]$
      $\land\ tokenUpdate(m)$
    $\square$ $m.res = \{-1\}$
    $\rightarrow$ $\land Shared' = [Shared$ EXCEPT $!.medium = \{\}]$
      $\land$ UNCHANGED $SoftState$
  $\land\ HardState' =$
      $[i \in NoRecvSet(m) \mapsto HardState[i]]\ @@$
      $[i \in ServID \setminus NoRecvSet(m) \mapsto [HardState[i]$ EXCEPT
        $!.execTimer =$ IF $Len(HardState[i].msg) = 0$ THEN pi ELSE $@,$
        $!.cons[m.id] =$ IF $m.res \neq \{-1\}$ THEN 1 ELSE $@,$
        $!.msg = Append(@,\ m),\ !.res =$
            IF $m.res = \{-1\}$
            THEN $[j \in ServID \mapsto$ IF $j = m.id$ THEN -1 ELSE $@[j]]$
            ELSE $[j \in ServID \mapsto$ IF $j \in m.res$ THEN $m.id$ ELSE $@[j]]]]$
  $\land$ UNCHANGED $History$

$SoftRecv(m) \triangleq$
  $\land$ LET $lastSoft \triangleq$ $\land\ 2 * delta \leq Shared.chipTimer$
      $\land\ Shared.chipTimer \leq deltaChip$
      $\land \forall\, j \in ServID : SoftState[j].token = -1$
    IN $Shared' = [Shared$ EXCEPT
        $!.medium = \{\},$
        $!.macTimer =$ IF $lastSoft$ THEN $Infinity$ ELSE $@,$
        $!.soft =$ IF $lastSoft$ THEN $\{\}$ ELSE $@]$
  $\land\ SoftState' = [j \in (ServID \setminus Shared.soft) \cup \{m.id\} \mapsto SoftState[j]]\ @@$
      $[j \in Shared.soft \setminus \{m.id\} \mapsto [SoftState[j]$ EXCEPT
        $!.token = next(@,\ Shared.soft),$
        $!.count = @ + 1]]$
  $\land$ UNCHANGED $\langle HardState,\ History\rangle$

$Receive(m) \triangleq$
  $\land$ $Shared.macTimer = 0$
  $\land$ CASE $m.type =$ "hard" $\rightarrow HardRecv(m)$
    $\square$ $m.type =$ "soft" $\rightarrow SoftRecv(m)$

(*Continue*)

NextTick is the only actions that update timers according to passage of time. *timerUpdate*(): function that updates all timers

$$
\begin{array}{l}
timerUpdate(d,\ noRese,\ noSoft) \triangleq \\
\quad \land\ Shared' = [Shared\ \text{EXCEPT} \\
\qquad !.chipTimer = @ + d, \\
\qquad !.macTimer = \text{CASE}\ \ noRese \to @ \ \square\ noSoft \to Infinity \\
\qquad\qquad\qquad\qquad \square\ @ = Infinity \to Infinity\ \square\ \text{OTHER}\ \to @ - d] \\
\quad \land\ HardState' = [i \in ServID \mapsto [HardState[i]\ \text{EXCEPT} \\
\qquad !.msg = \text{IF}\ HardState[i].execTimer - d = 0\ \ \text{THEN}\ Tail(@)\ \text{ELSE}\ \ @, \\
\qquad !.execTimer = \text{IF}\ @ - d = 0 \\
\qquad\qquad\qquad\qquad \text{THEN IF}\ Len(HardState[i].msg) > 1\ \text{THEN}\ pi\ \text{ELSE}\ \ Infinity \\
\qquad\qquad\qquad\qquad \text{ELSE}\ \ \text{IF}\ @ = Infinity\ \text{THEN}\ @\ \text{ELSE}\ \ @ - d]]
\end{array}
$$

$$
\begin{array}{l}
minIncVal(tmp,\ noRese,\ noSoft) \triangleq \\
\quad \text{CASE}\ \ noRese\ \ \to min(\{delta\} \cup tmp) \\
\qquad \square\ noSoft\ \ \ \to min(tmp) \\
\qquad \square\ \text{OTHER}\ \ \ \to min(\{Shared.macTimer\} \cup tmp)
\end{array}
$$

$$
\begin{array}{l}
NextTick \triangleq \\
\quad \text{LET}\ noRese \triangleq \land\ Shared.medium = \{\} \\
\qquad\qquad\qquad\qquad \land\ Shared.chipTimer = delta \\
\qquad\qquad\qquad\qquad \land\ \forall\, i \in ServID : HardState[i].res[Shared.chipCount] \neq i \\
\qquad\quad noSoft \triangleq \land\ 2 * delta \leq Shared.chipTimer \\
\qquad\qquad\qquad\qquad \land\ Shared.chipTimer \leq deltaChip \\
\qquad\qquad\qquad\qquad \land\ Shared.medium = \{\} \\
\qquad\qquad\qquad\qquad \land\ \forall\, j \in Shared.soft : SoftState[j].token \neq j \\
\qquad\quad tmp \triangleq \{HardState[i].execTimer : i \in ServID\} \cup \\
\qquad\qquad\qquad\quad \{deltaChip - Shared.chipTimer\} \\
\qquad\quad d \triangleq minIncVal(tmp,\ noRese,\ noSoft) \\
\quad \text{IN}\ \ \land\ d > 0 \\
\qquad\quad \land\ timerUpdate(d,\ noRese,\ noSoft) \\
\qquad\quad \land\ \text{UNCHANGED}\ \langle SoftState,\ History \rangle
\end{array}
$$

NextChip increment *ChipCount* modulo the number of task. It sets the arbitrary soft sequence of message of each process during every cycle. Alltogether, *NextTick* and *NextChip* implements the time circularity of the model. The "count" counter allows to check if a soft message has been sent in the previous chip. If it is not the case, all processes increment the token value by 1 at the end of the chip.

(*Continue*)

$$
\begin{array}{l}
\mathit{NextChip} \triangleq \\
\quad \land \mathit{Shared.medium} = \{\} \\
\quad \land \mathit{Shared.chipTimer} = \mathit{deltaChip} \\
\quad \land \text{LET } \mathit{Overflow} \triangleq \exists\, j \in \mathit{Shared.soft} : \mathit{Len}(\mathit{SoftState}[j].\mathit{list}) > 14 \\
\qquad\quad \mathit{TimeCircle} \triangleq \mathit{Shared.cycleCount} = \mathit{horizon} \\
\qquad\quad \mathit{NextCycle} \triangleq \mathit{Shared.chipCount} = \mathit{nServ} \\
\qquad \text{IN} \quad \land \mathit{Shared}' = [\mathit{Shared} \text{ EXCEPT} \\
\qquad\qquad\quad !.\mathit{macTimer} = 0, \\
\qquad\qquad\quad !.\mathit{chipCount} = (@ \,\%\, \mathit{nServ}) + 1, \\
\qquad\qquad\quad !.\mathit{chipTimer} = \text{IF } \mathit{Overflow} \text{ THEN } -1 \text{ ELSE } 0, \\
\qquad\qquad\quad !.\mathit{cycleCount} = \text{IF } \mathit{TimeCircle} \\
\qquad\qquad\qquad\qquad\qquad \text{THEN } 1 \\
\qquad\qquad\qquad\qquad\qquad \text{ELSE IF } \mathit{NextCycle} \text{ THEN } @ + 1 \text{ ELSE } @] \\
\qquad\qquad \land \mathit{SoftState}' = [j \in (\mathit{ServID} \setminus \mathit{Shared.soft}) \mapsto \mathit{SoftState}[j]]\, @@ \\
\qquad\qquad\qquad\qquad\quad [j \in \mathit{Shared.soft} \mapsto [\mathit{SoftState}[j] \text{ EXCEPT} \\
\qquad\qquad\quad !.\mathit{count} = 0, \\
\qquad\qquad\quad !.\mathit{token} = \text{IF } \mathit{SoftState}[j].\mathit{count} = 0 \text{ THEN } \mathit{next}(@, \mathit{Shared.soft}) \text{ ELSE } @, \\
\qquad\qquad\quad !.\mathit{list} = \text{IF } \mathit{NextCycle} \text{ THEN } @ \circ \mathit{list}(j, \mathit{Shared.cycleCount}) \text{ ELSE } @]] \\
\qquad\qquad \land \mathit{HardState}' = [i \in \mathit{ServID} \mapsto [\mathit{HardState}[i] \text{ EXCEPT} \\
\qquad\qquad\quad !.\mathit{cons}[\mathit{Shared}'.\mathit{chipCount}] = 0]] \\
\qquad\qquad \land \text{IF } \mathit{NextCycle} \\
\qquad\qquad\quad \text{THEN } \mathit{History}' = [\mathit{elem} \mapsto 0,\, \mathit{rese} \mapsto 0] \\
\qquad\qquad\quad \text{ELSE UNCHANGED } \mathit{History}
\end{array}
$$

Next is the disjunct of steps choices that doesn't alterate the timers

$$
\begin{array}{lll}
\mathit{Next} & \triangleq & \lor \exists\, s \in \mathit{HardRing} : \mathit{ElemSend}(s) \lor \mathit{ReseSend}(s) \\
& & \lor \exists\, t \in \mathit{SoftRing}(\mathit{Shared.soft}) : \mathit{SoftSend}(t) \\
& & \lor \exists\, \mathit{msg} \in \mathit{Shared.medium} : \mathit{Receive}(\mathit{msg})
\end{array}
$$

Tick defines the passage of time.

$$\mathit{Tick} \triangleq \mathit{NextTick} \lor \mathit{NextChip}$$

Liveness is a condition that warrants the eventual flow of time.

$$\mathit{Liveness} \triangleq \Box\Diamond \mathit{Tick}$$

The main formula of the specification

$$\mathit{Spec} \triangleq \mathit{Init} \land \Box[\mathit{Next} \lor \mathit{Tick}]_{\mathit{vars}} \land \mathit{Liveness}$$

(*Continue*)

The following temporal properties are checked when specified in the *DoRiS.cfg* configuration file.
"*TypeInvariance*" checks the variables type invariance.
"*CollisionAvoidance*" checks that the token can be hold by only one task or process at once.
"*HardRingCorrectness*" makes use of the "History" variable to check that all mandatory elementary message are sent in a *DoRiS* cycle. "*ReservationSafety*" guaranty that if a task holds the token in a reservation slot, then all other tasks are aware of its reservation.
"*SoftRingFairness*" guaranty that each process will eventually receive the token.
"Omission" states that some task omission failure takes place.
"Failure" states that some process failure takes place.
"*NoReservationSafety*", "*NoCollisionAvoidance*" and "*NoOmission*" are contradiction of the respective properties, used to generate handfull conter-examples.

$$\begin{aligned}
&HardMsg \triangleq Seq([id : ServID,\ type : \{\text{“hard”}\},\ res : \{\{-1\}\}] \cup \\
&\quad [id : ServID,\ type : \{\text{“hard”}\},\ res : \text{SUBSET } (\{-1\} \cup ServID),\ softFlag : 0\,..\,1])
\end{aligned}$$

$$\begin{aligned}
&MediumMsg \triangleq \{m : m \in [id : ServID,\ type : \{\text{“soft”}\}] \cup \\
&\quad [id : ServID,\ type : \{\text{“hard”}\},\ res : \text{SUBSET } (\{-1\})] \cup \\
&\quad [id : ServID,\ type : \{\text{“hard”}\},\ res : \text{SUBSET } (\{-1\} \cup ServID),\ softFlag : 0\,..\,1\ ]\}
\end{aligned}$$

$$\begin{aligned}
&TypeInvariance \triangleq \\
&\quad \land\ Shared.chipCount \in 1\,..\,nServ \\
&\quad \land\ Shared.cycleCount \in 1\,..\,horizon \\
&\quad \land\ Shared.chipTimer \in 0\,..\,deltaChip \\
&\quad \land\ Shared.macTimer \in 0\,..\,maxTxTime \cup \{Infinity\} \\
&\quad \land\ \forall\, m \in Shared.medium : m \in MediumMsg \\
&\quad \land\ SoftState \in [ServID \rightarrow \\
&\qquad [token : ServID \cup \{-1\}, \\
&\qquad\ count : 0\,..\,50, \\
&\qquad\ list : \{\langle\rangle\} \cup Seq([txTime : \{maxTxTime\}])]] \\
&\quad \land\ HardState \in [ServID \rightarrow \\
&\qquad [msg : \{\langle\rangle\} \cup HardMsg, \\
&\qquad\ res\ \ : [ServID \rightarrow \{-1\} \cup ServID\ ], \\
&\qquad\ execTimer : 0\,..\,pi \cup \{Infinity\}, \\
&\qquad\ cons : [ServID \rightarrow 0\,..\,1]\ ]]
\end{aligned}$$

$$\begin{aligned}
&Send(s) \triangleq \ \lor\ \land\ s \in HardRing \\
&\qquad\qquad\ \land\ \lor \text{ENABLED } ElemSend(s) \\
&\qquad\qquad\qquad\ \lor \text{ENABLED } ReseSend(s) \\
&\qquad\qquad\ \lor\ \land\ s \in SoftRing(Shared.soft) \\
&\qquad\qquad\qquad\ \land \text{ENABLED } SoftSend(s)
\end{aligned}$$

(*Continue*)

$$
\begin{array}{l}
\mathit{CollisionAvoidance} \triangleq \\
\quad \forall\, s,\, t \in \mathit{HardRing} \cup \mathit{SoftRing}(\mathit{ServID}) : \\
\qquad \Box(\text{ENABLED } (\mathit{Send}(s) \land \mathit{Send}(t)) \Rightarrow (s = t)) \\
\\
\mathit{NoCollisionAvoidance} \triangleq \\
\quad \exists\, s,\, t \in \mathit{HardRing} \cup \mathit{SoftRing}(\mathit{ServID}) : \\
\qquad \Diamond((s \neq t) \land \text{ENABLED } (\mathit{Send}(s) \land \mathit{Send}(t))) \\
\\
\mathit{HardRingCorrectness} \triangleq \\
\quad \land \forall\, s \in \mathit{HardRing} : \Box(\mathit{Len}(\mathit{HardState}[\mathit{hardID}(s)].\mathit{msg}) \leq 3) \\
\quad \land \Box(\text{ENABLED } \mathit{NextChip} \Rightarrow \mathit{History}.\mathit{elem} = \mathit{Shared}.\mathit{chipCount}) \\
\\
\mathit{ReservationSafety} \triangleq \\
\quad \Box\forall\, \mathit{chip},\, j \in \mathit{ServID} : \land \text{ENABLED } \mathit{ReseSend}(\mathit{HardServ}[j]) \\
\qquad\qquad\qquad\qquad\qquad \land \mathit{Shared}.\mathit{chipCount} = \mathit{chip} \\
\qquad \Rightarrow \quad \land \mathit{HardState}[j].\mathit{res}[\mathit{chip}] = j \\
\qquad\qquad\quad \land \forall\, i \in \mathit{ServID} \setminus \{j\} : \mathit{HardState}[i].\mathit{res}[\mathit{chip}] \in \{j,\ -1\} \\
\\
\mathit{SoftRingFairness} \triangleq \\
\quad \land \forall\, i \in \mathit{ServID} : \Box(i \in \mathit{Shared}.\mathit{soft} \\
\qquad \Rightarrow (\mathit{SoftState}[i].\mathit{list} \neq \langle\rangle \Rightarrow \Diamond(i = \mathit{SoftState}[i].\mathit{token}))) \\
\quad \land \Box\Diamond(\forall\, i \in \mathit{ServID} \setminus \mathit{Failed} : i \in \mathit{Shared}.\mathit{soft} \Rightarrow \mathit{Len}(\mathit{SoftState}[i].\mathit{list}) = 0) \\
\\
\mathit{NoReservationSafety} \triangleq \\
\quad \Box\Diamond\ \exists\, \mathit{chip} \in \mathit{ServID} : \exists\, j \in \mathit{ServID} : \\
\qquad \land \mathit{HardState}[j].\mathit{res}[\mathit{chip}] \neq -1 \\
\qquad \land \text{ENABLED } \mathit{ReseSend}(\mathit{HardServ}[j]) \\
\qquad \land \mathit{Shared}.\mathit{chipCount} = \mathit{chip} \\
\qquad \land \exists\, i \in \mathit{ServID} \setminus \{j\} : \neg \mathit{HardState}[i].\mathit{res}[\mathit{chip}] \in \{j,\ -1\} \\
\\
\mathit{Omission} \triangleq \exists\, s \in \mathit{HardRing} : \\
\quad \Diamond(\text{ENABLED } \mathit{ElemSend}(s)\ \land \exists\, i \in \mathit{ServID} : \mathit{HardState}[\mathit{hardID}(s)].\mathit{cons}[i] = 0) \\
\\
\mathit{NoOmission} \triangleq \forall\, s \in \mathit{HardRing} : \\
\quad \Box(\text{ENABLED } \mathit{ElemSend}(s)\ \Rightarrow (\forall\, i \in \mathit{ServID} : \mathit{HardState}[\mathit{hardID}(s)].\mathit{cons} = 1)) \\
\\
\mathit{Failure} \triangleq \Box\Diamond(\exists\, p \in \mathit{SoftRing}(\mathit{ServID}) : \mathit{softID}(p,\ \mathit{ServID}) \in \mathit{Failed})
\end{array}
$$