A TLA+ Formal Specification and Verification of a New Real-Time Communication Protocol

P. Regnier, G. Lima, A. Andrade

LaSiD
Master of Mechatronic
Computer Science Department
Institute of Mathematics - UFBA

Funding FAPESB - TIC 2630-2006

SBMF'08

Summary

Summary

Model

Distributed System

- System: A set of station sharing a same communication bus (connected graph).
- Capabilities: messages, clocks and/or timers and processing capabilities

Restrictions

- No shared-memory
- No global clock

Model

Distributed System

- System: A set of station sharing a same communication bus (connected graph).
- Capabilities: messages, clocks and/or timers and processing capabilities

Restrictions

- No shared-memory
- No global clock

Goals

Deterministic communication

- Timeliness: Temporal requirements of applications
- Reliability: Fault-tolerance

↪ Hard Messages

High Throughput

- Optimization and flexibility
- Diversity of the devices
- Reconfiguration of the system

↪ Soft Messages

Goals

Deterministic communication

- Timeliness: Temporal requirements of applications
- Reliability: Fault-tolerance

➥ Hard Messages

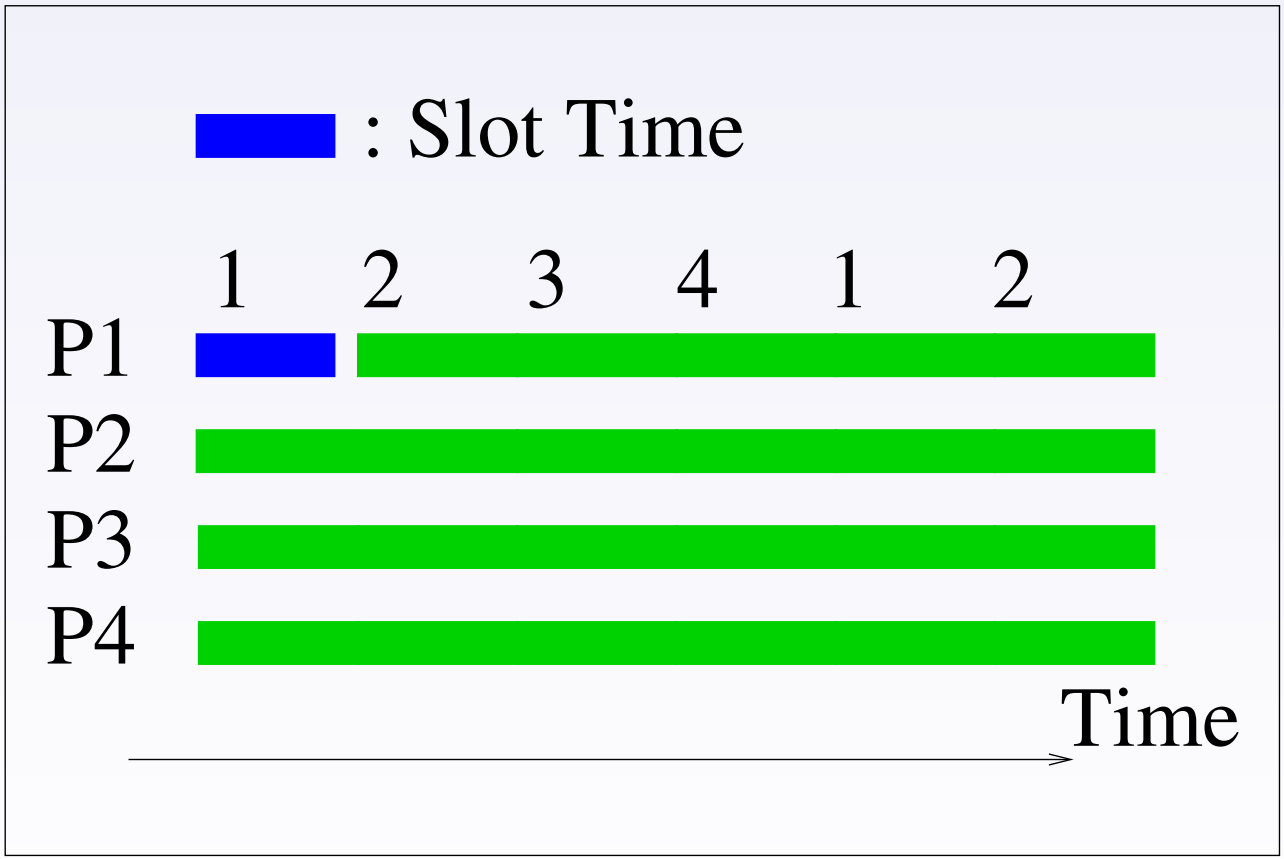
High Throughput

- Optimization and flexibility
- Diversity of the devices
- Reconfiguration of the system

➥ Soft Messages

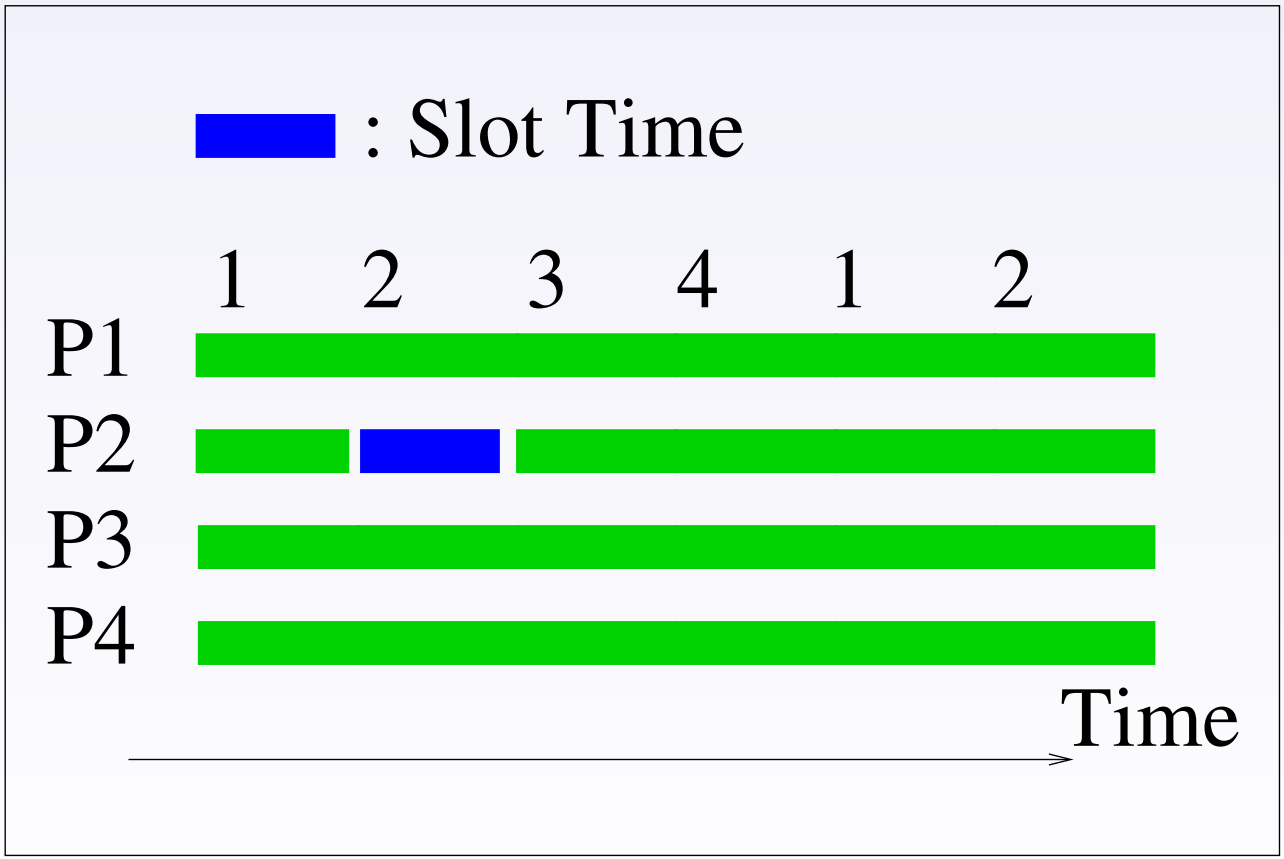
TDMA Approach - Time Division Multiple Access

- Periodic slots allocated to each process
- Limitation: Clocks synchronization, bandwidth waste



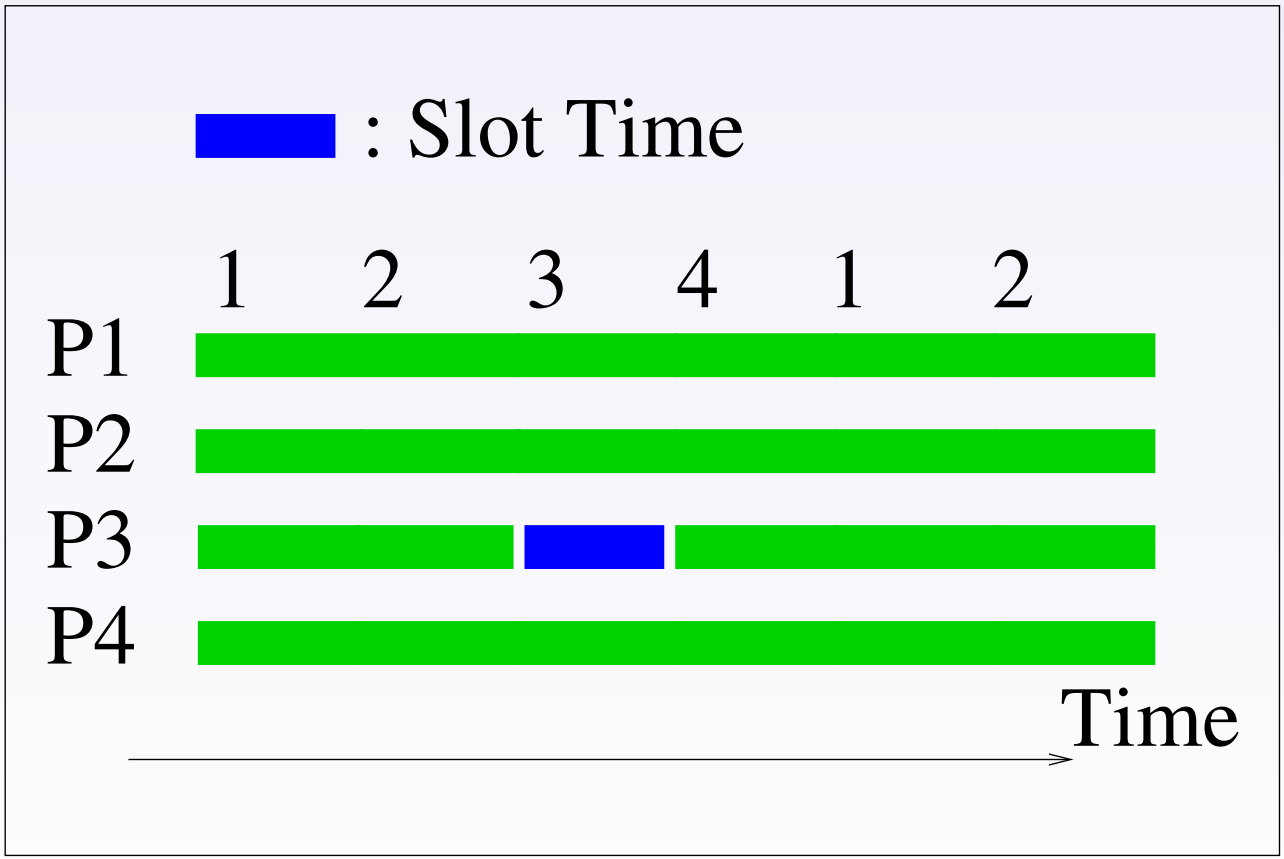
TDMA Approach - Time Division Multiple Access

- Periodic slots allocated to each process
- Limitation: Clocks synchronization, bandwidth waste



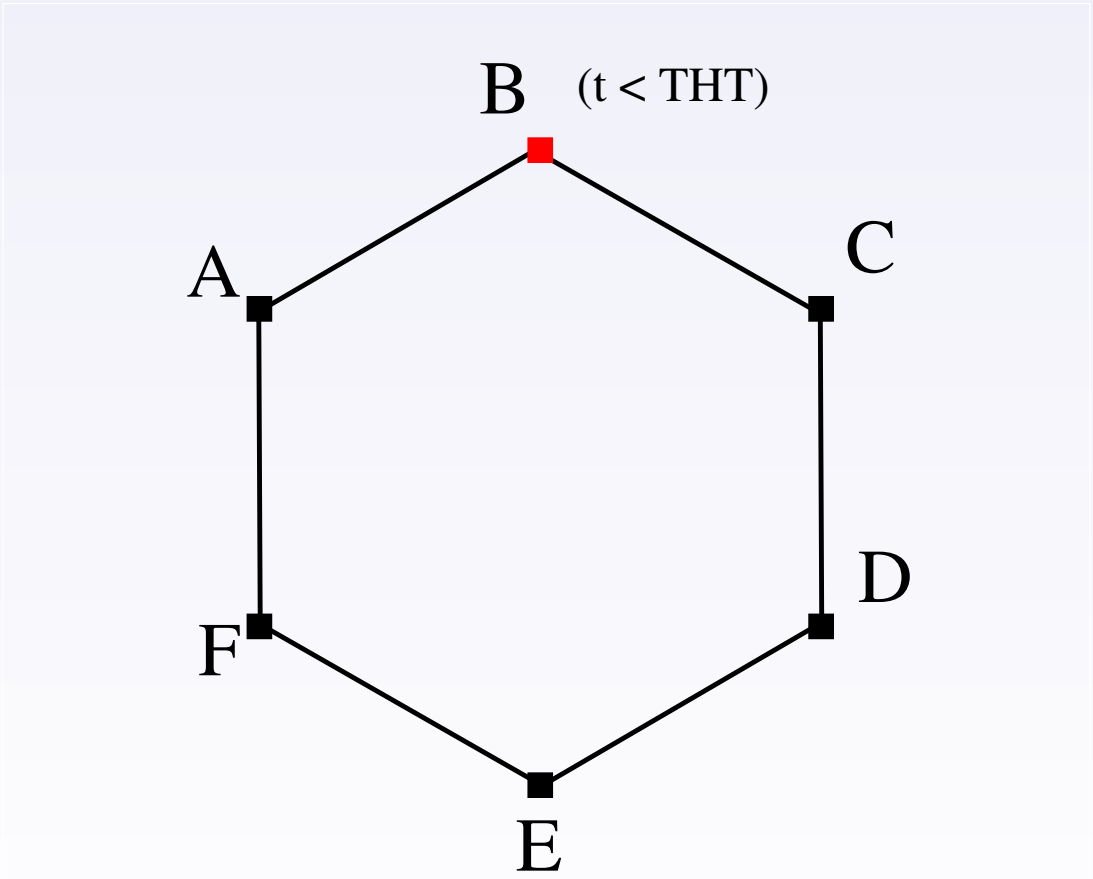
TDMA Approach - Time Division Multiple Access

- Periodic slots allocated to each process
- Limitation: Clocks synchronization, bandwidth waste



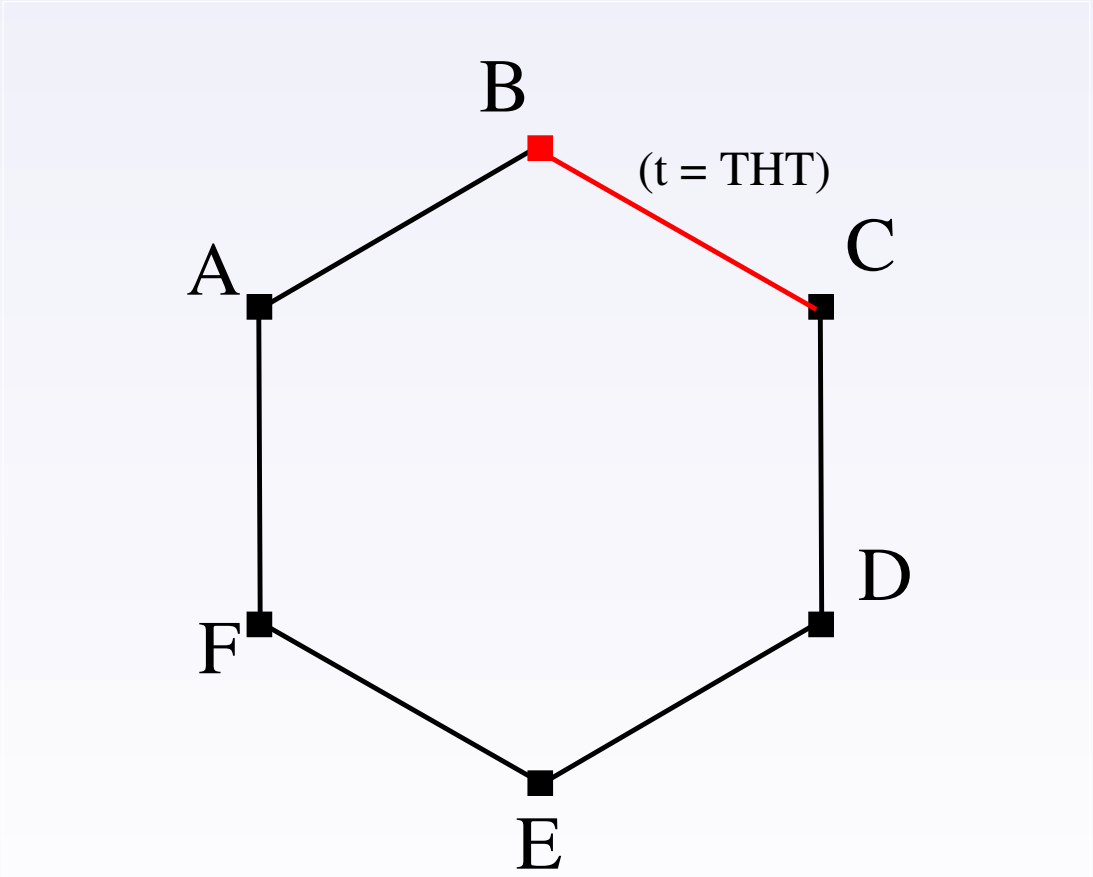
Token Ring Approach

- Token (explicit or not) $\Rightarrow$ gives the right to transmit
- Drawbacks: Overload, token loss, variability



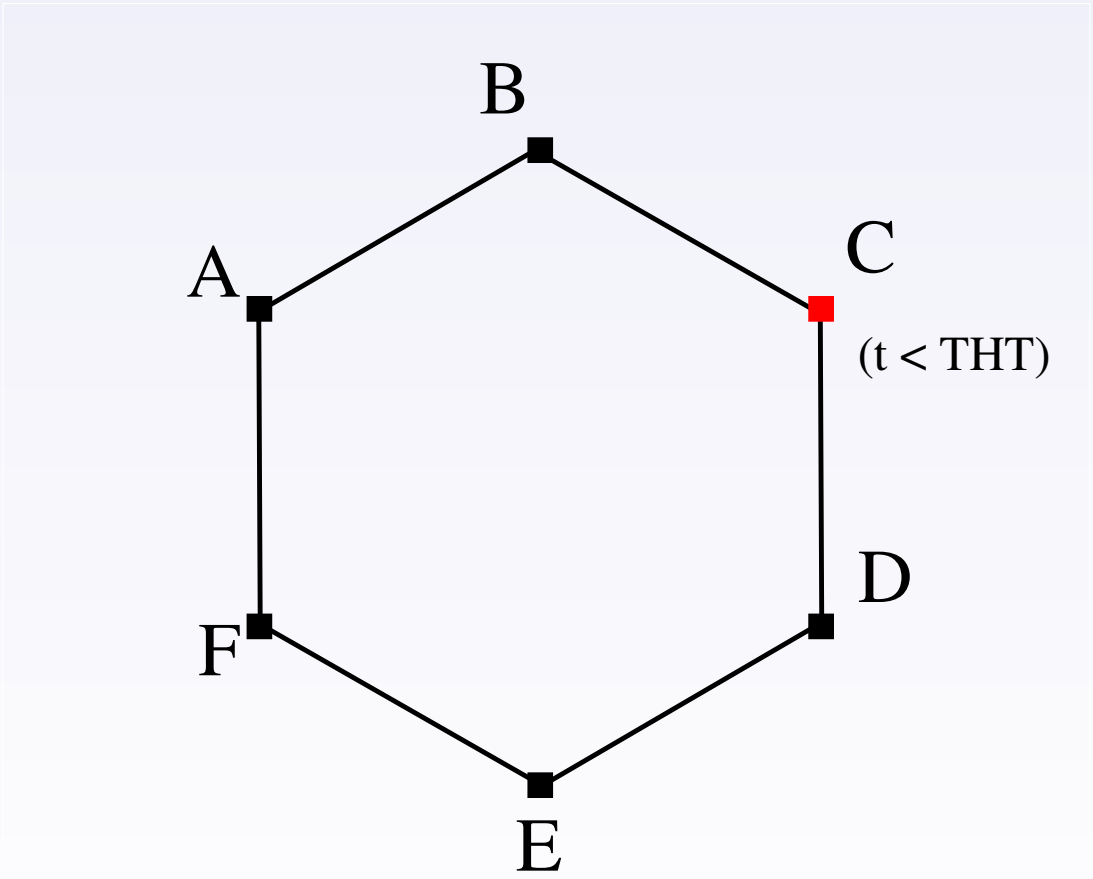
Token Ring Approach

- Token (explicit or not) $\Rightarrow$ gives the right to transmit
- Drawbacks: Overload, token loss, variability

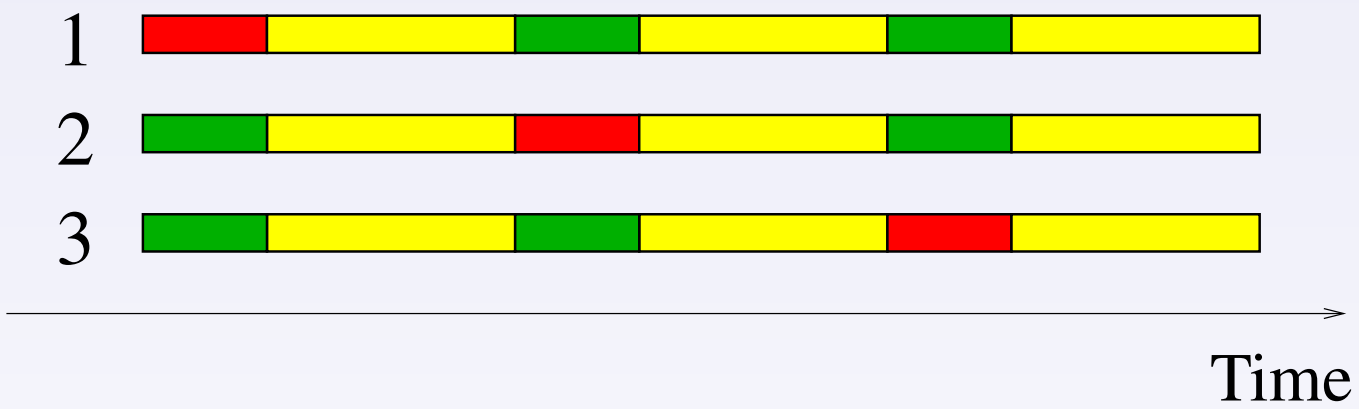


Token Ring Approach

- Token (explicit or not) $\Rightarrow$ gives the right to transmit
- Drawbacks: Overload, token loss, variability



PROCESSING DYNAMICS: SLOW VS FAST



: Hard messages

: Slow nodes processing

: Busy medium

FUNDAMENTAL PROPERTY

While slow nodes process hard messages
fast nodes use the available bandwidth for soft communication

Summary

DoRiS: Computational Model - I

Servers

- *DoRiS* segment: *nServ* nodes and one communication medium
- Each node hosts one *DoRiS* server, uniquely identified, responsible for hard and soft messages transmission
- *HardServ*[*i*] and *SoftServ*[*i*]: threads of server *i* dealing with the hard and soft queues, respectively

Synchronous distributed system

Regular and predictable points of synchronism

⇛ Node clocks are synchronized

⇛ Global clock emulation (despite our system model)

Failure

Crash failure and message omission failure

DoRiS: Computational Model - I

Servers

- *DoRiS* segment: *nServ* nodes and one communication medium
- Each node hosts one *DoRiS* server, uniquely identified, responsible for hard and soft messages transmission
- *HardServ*[*i*] and *SoftServ*[*i*]: threads of server *i* dealing with the hard and soft queues, respectively

Synchronous distributed system

Regular and predictable points of synchronism

➠ Node clocks are synchronized

➠ Global clock emulation (despite our system model)

Failure

Crash failure and message omission failure

DoRiS: Computational Model - I

Servers

- *DoRiS* segment: *nServ* nodes and one communication medium
- Each node hosts one *DoRiS* server, uniquely identified, responsible for hard and soft messages transmission
- *HardServ*[*i*] and *SoftServ*[*i*]: threads of server *i* dealing with the hard and soft queues, respectively

Synchronous distributed system

Regular and predictable points of synchronism

⇛ Node clocks are synchronized

⇛ Global clock emulation (despite our system model)

Failure

Crash failure and message omission failure

DoRiS: Computational Model - II

Messages

- δ : hard message transmission time
  (e.g. 64B $\Rightarrow \approx 5\mu s$ over Fast-Ethernet)
- π : worst-case processing time of slow nodes

Assumption: $\delta \ll \pi$

Communication model

- Broadcast and publish-subscribe
- Soft messages of variable length
  (e.g. $64B \leqslant L \leqslant 1500B$ over Ethernet).

DoRiS: Computational Model - II

Messages

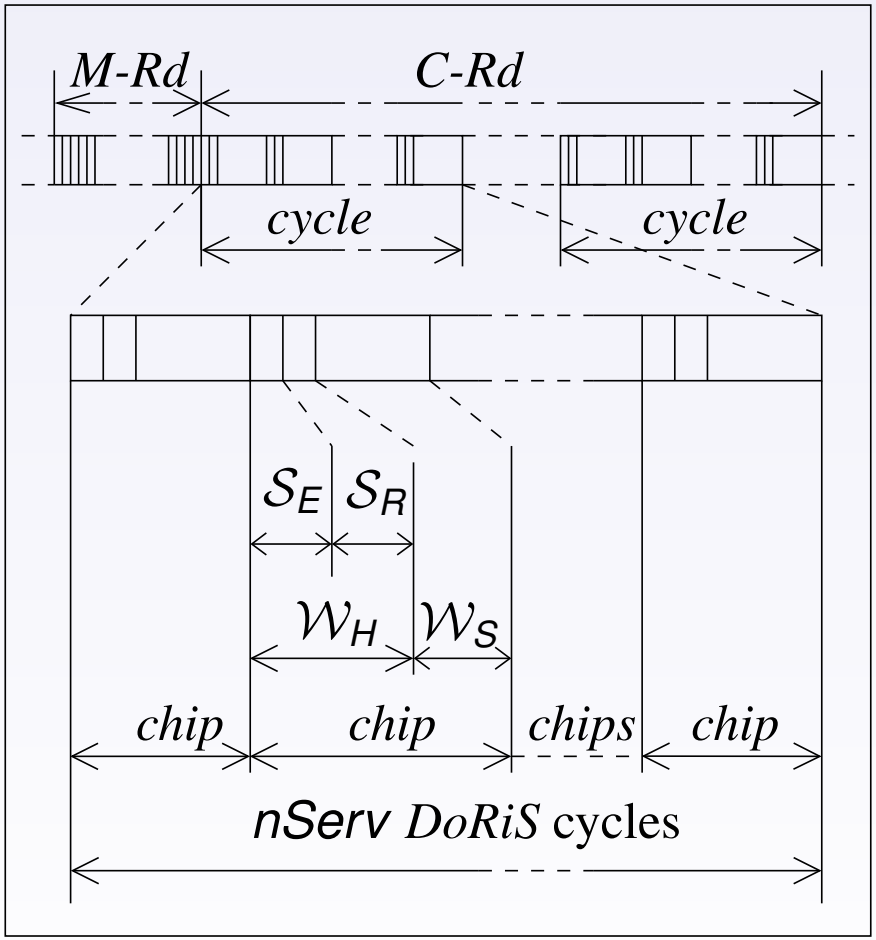
- δ : hard message transmission time
  (e.g. 64B $\Rightarrow\approx 5\mu s$ over Fast-Ethernet)
- π : worst-case processing time of slow nodes

Assumption: $\delta \ll \pi$

Communication model

- Broadcast and publish-subscribe
- Soft messages of variable length
  (e.g. $64B \leqslant L \leqslant 1500B$ over Ethernet).

DoRiS: Time Structure



Time

Membership Rounds (M-Rd) and Communication Rounds (C-Rd)

Cycle

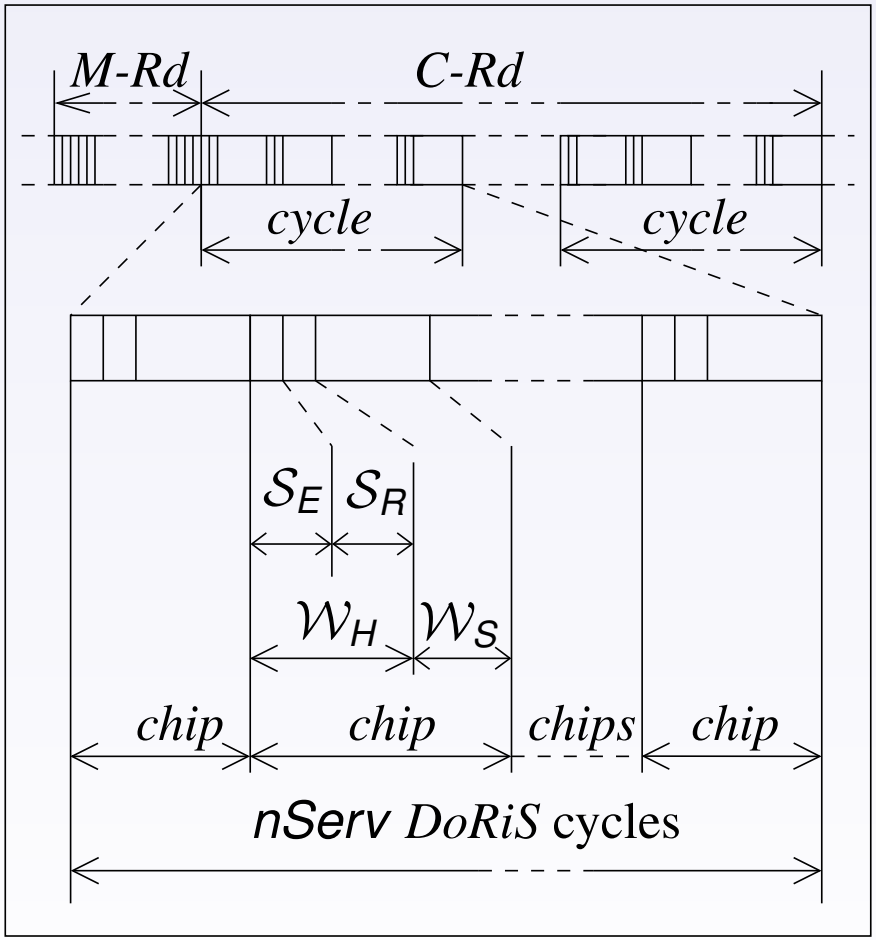
Sequence of *nServ* chips (Δ_C)

Chip

Hard ($\mathcal{W}_H$) and soft window ($\mathcal{W}_S$)

- $\mathcal{W}_H$ ($2\Delta_E$): elementary ($\mathcal{S}_E$) and reservation slot ($\mathcal{S}_R$)
- $\mathcal{W}_S$: contains at least two soft messages

DoRiS: Time Structure



Time

Membership Rounds (M-Rd) and Communication Rounds (C-Rd)

Cycle

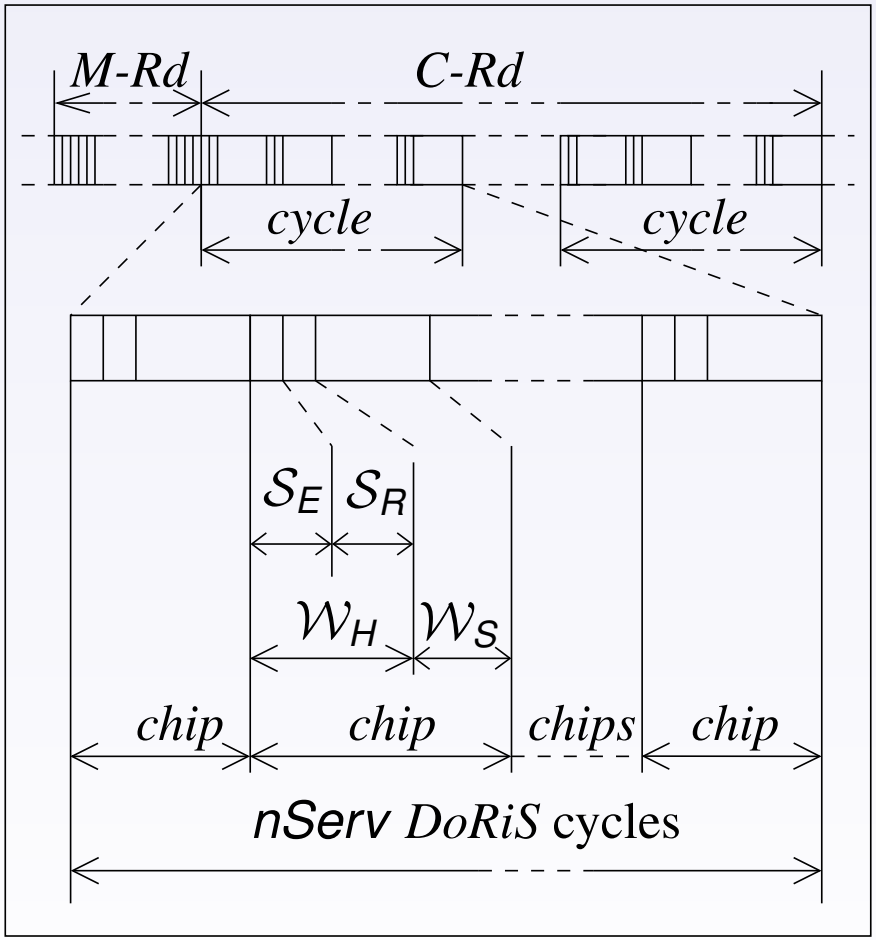
Sequence of *nServ* chips (Δ_C)

Chip

Hard ($\mathcal{W}_H$) and soft window ($\mathcal{W}_S$)

- $\mathcal{W}_H$ ($2\Delta_E$): elementary ($\mathcal{S}_E$) and reservation slot ($\mathcal{S}_R$)
- $\mathcal{W}_S$: contains at least two soft messages

DoRiS: Time Structure



Time

Membership Rounds (M-Rd) and Communication Rounds (C-Rd)

Cycle

Sequence of *nServ* chips (Δ_C)

Chip

Hard ($\mathcal{W}_H$) and soft window ($\mathcal{W}_S$)

- $\mathcal{W}_H$ ($2\Delta_E$): elementary ($\mathcal{S}_E$) and reservation slot ($\mathcal{S}_R$)
- $\mathcal{W}_S$: contains at least two soft messages

DoRiS: Fundamental Properties

Elementary Messages

Mandatory ➠ Each server sends a unique elementary message per cycle

Consequence

Periodicity ➠ **Determinism**, **Synchrony** (Global clock) and **Fault Tolerance**

Informations piggybacked on elementary messages

- Reservation mechanism gives hard ring flexibility ▸ Res. Mech.
- Dynamic management of process group membership ▸ Dyn. Group

DoRiS: Fundamental Properties

Elementary Messages

Mandatory ➠ Each server sends a unique elementary message per cycle

Consequence

Periodicity ➠ **Determinism**, **Synchrony** (Global clock) and **Fault Tolerance**

Informations piggybacked on elementary messages

- Reservation mechanism gives hard ring flexibility ▸ Res. Mech.
- Dynamic management of process group membership ▸ Dyn. Group

Summary

Formal specification language

- Predicate logic, temporal Logic and Zermelo-Fraenkel set theory
- Allow for the specification of concurrent and distributed systems
- TLC: Temporal Logic Checker

Characteristics

- Modular structure: allows for an incremental process of specification, according to the abstraction level required
- Concrete specification, close to the code level

TLA+ : Temporal Logic of Actions

Formal specification language

- Predicate logic, temporal Logic and Zermelo-Fraenkel set theory
- Allow for the specification of concurrent and distributed systems
- TLC: Temporal Logic Checker

Characteristics

- Modular structure: allows for an incremental process of specification, according to the abstraction level required
- Concrete specification, close to the code level

TIME MODEL

- Time: one more variable $\Rightarrow$ State number increase
- Representation by naturals $\Leftrightarrow$ Discrete increments
- Synchronous system (timers without jitter nor drift) $\Leftrightarrow$ global clock

MAIN FORMULA

$$Spec \triangleq Init \wedge \Box[Next \vee Tick]_{vars} \wedge Liveness$$

- *Init*: the initial protocol states
- $\Box[Next \vee Tick]_{vars}$: the next-state relation
- $Liveness \triangleq \Box\Diamond Tick$: condition that ensures the progress of the behavior

TIME MODEL

- Time: one more variable $\Rightarrow$ State number increase
- Representation by naturals $\Leftrightarrow$ Discrete increments
- Synchronous system (timers without jitter nor drift) $\Leftrightarrow$ global clock

MAIN FORMULA

$$Spec \triangleq Init \wedge \Box[Next \vee Tick]_{vars} \wedge Liveness$$

- *Init*: the initial protocol states
- $\Box[Next \vee Tick]_{vars}$: the next-state relation
- $Liveness \triangleq \Box\Diamond Tick$: condition that ensures the progress of the behavior

MAIN ACTIONS

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

ElemSend, *ReseSend* and *SoftSend* — Message emission actions

Receive (*HardRecv* and *SoftRecv*) — Message reception action

NextTick and *NextChip* — Time passing actions

The complete specification is Here

MAIN ACTIONS

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

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

ElemSend, *ReseSend* and *SoftSend* — Message emission actions

Receive (*HardRecv* and *SoftRecv*) — Message reception action

NextTick and *NextChip* — Time passing actions

The complete specification is Here

DoRiS: TLC Verification Performance - I

Set-Up

2 Ghz Intel Core Duo processor, JVM with a 512M heap size

3 Metrics:
- CPU user time (U)
- Total number of generated distinct states (N)
- Diameter (D) of the reachability graph

Results

	# Servers						
	2	4	6	8	10	12	14
U (*seconds*)	3	21	59	153	329	833	4,117
N (*#states*)	853	2,529	3,565	5,032	6,430	8,148	9,540
D (*#states*)	850	2,516	3,486	5,003	6,392	8,099	9,482

DoRiS: TLC Verification Performance - I

Set-Up

2 Ghz Intel Core Duo processor, JVM with a 512M heap size

3 Metrics:
- CPU user time (U)
- Total number of generated distinct states (N)
- Diameter (D) of the reachability graph

Results

	# Servers						
	2	4	6	8	10	12	14
U (*seconds*)	3	21	59	153	329	833	4,117
N (#*states*)	853	2,529	3,565	5,032	6,430	8,148	9,540
D (#*states*)	850	2,516	3,486	5,003	6,392	8,099	9,482

DoRiS: TLC Verification Performance - II

Reduced state number

- Temporal guards of actions
- Communication and fault scenarios fixed
- Time increment strategy (*NextTick*)

Comparison with naive strategy: Unit time increments

- 6 servers: $U = 453s(59s)$, $N = 21,604(3,565)$ and $D = 11,261(3,486)$
- 14 servers: TLC did not finished after 20h (state explosion)

DoRiS: TLC Verification Performance - II

Reduced state number

- Temporal guards of actions
- Communication and fault scenarios fixed
- Time increment strategy (*NextTick*)

Comparison with naive strategy: Unit time increments

- 6 servers: $U = 453s(59s)$, $N = 21,604(3,565)$ and $D = 11,261(3,486)$
- 14 servers: TLC did not finished after 20h (state explosion)

COLLISION AVOIDANCE

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

$$
\begin{aligned}
&\mathit{CollisionAvoidance} \triangleq \\
&\qquad \forall\, s,\, t \in \mathit{HardRing} \cup \mathit{SoftRing}(\mathit{ServID}) : \\
&\qquad\qquad \Box(\text{ENABLED}\ (\mathit{Send}(s) \land \mathit{Send}(t)) \Rightarrow (s = t))
\end{aligned}
$$

ESSENTIAL *DoRiS* TEMPORAL PROPERTIES

▸ Temp. Prop.

- Correctness of the Hard Ring
- Correctness of the Reservation Mechanism
- Fairness of the Soft Ring

DoRiS: Temporal properties - I

Collision Avoidance

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

$$
\begin{aligned}
& CollisionAvoidance \triangleq \\
& \quad \forall s, t \in HardRing \cup SoftRing(ServID) : \\
& \qquad \Box(\text{ENABLED } (Send(s) \land Send(t)) \Rightarrow (s = t))
\end{aligned}
$$

Essential *DoRiS* Temporal Properties

▸ Temp. Prop.

- Correctness of the Hard Ring
- Correctness of the Reservation Mechanism
- Fairness of the Soft Ring

Discussion

Protocol Conception

- Specification and conception: interactive process
- Protocol drawbacks detected by model-checking the specification

Fault Tolerance

- Fault occurrences were specified in incremental steps
- Straightforward process in TLA+

Implementation

- Specification: gives insights to carry out the implementation
- But, complexity of the Linux-based real-time operating system

Discussion

Protocol Conception

- Specification and conception: interactive process
- Protocol drawbacks detected by model-checking the specification

Fault Tolerance

- Fault occurrences were specified in incremental steps
- Straightforward process in TLA+

Implementation

- Specification: gives insights to carry out the implementation
- But, complexity of the Linux-based real-time operating system

Discussion

Protocol Conception

- Specification and conception: interactive process
- Protocol drawbacks detected by model-checking the specification

Fault Tolerance

- Fault occurrences were specified in incremental steps
- Straightforward process in TLA+

Implementation

- Specification: gives insights to carry out the implementation
- But, complexity of the Linux-based real-time operating system

Summary

Conclusion

Specification of a new real-time communication protocol

- TLA+ specification of the *DoRiS* protocol
- Verification of *DoRiS* and its temporal properties by TLC

Software Engineering Perspective

- Interactive design methodology: specification vs conception
- Implementation has been greatly improved

Questions ?

Conclusion

Specification of a new real-time communication protocol

- TLA+ specification of the *DoRiS* protocol
- Verification of *DoRiS* and its temporal properties by TLC

Software Engineering Perspective

- Interactive design methodology: specification vs conception
- Implementation has been greatly improved

Questions ?

Conclusion

Specification of a new real-time communication protocol

- TLA+ specification of the *DoRiS* protocol
- Verification of *DoRiS* and its temporal properties by TLC

Software Engineering Perspective

- Interactive design methodology: specification vs conception
- Implementation has been greatly improved

Questions ?

Summary

Brief Bibliography I

L. Lamport
Specifying Systems: The TLA+ language and tools for hardware and software engineers
Addison Wesley, 2002

F. B. Carreiro, J. A. Fonseca and P. Pedreiras
Virtual Token-Passing Ethernet - VTPE
Proc. FeT2003 5th IFAC Int. Conf. on Fieldbus Systems and their Applications, 2003

DoRiS: Reservation Mechanism

Back

- Each server holds a reservation vector
- Elementary messages piggyback a reservation list
- Each server can reserve all free slots of the next cycle (its horizon)
- There always is a reservation slot available in a server horizon

Fault Tolerance Mechanism

- A server is consistent if it has received the previous *nServ* elementary messages
- A server can reserve more than one reservation slot of its horizon iff it is consistent

DoRiS: Reservation Mechanism

Back

- Each server holds a reservation vector
- Elementary messages piggyback a reservation list
- Each server can reserve all free slots of the next cycle (its horizon)
- There always is a reservation slot available in a server horizon

Fault Tolerance Mechanism

- A server is consistent if it has received the previous *nServ* elementary messages
- A server can reserve more than one reservation slot of its horizon iff it is consistent

DoRiS: Dynamic management of process group membership

Back

- Elementary messages piggyback a flag bit
- Each server holds a list with its view of the process group membership
- Upon receiving an elementary message, each server updates the process group membership

Fault Tolerance Mechanism

- Elementary message omission failure $\Rightarrow$ The server loose its right to send soft messages
- After an empty window $\mathcal{W}_S$, all servers update the process group membership to the empty set

DoRiS: Dynamic management of process group membership

Back

- Elementary messages piggyback a flag bit
- Each server holds a list with its view of the process group membership
- Upon receiving an elementary message, each server updates the process group membership

Fault Tolerance Mechanism

- Elementary message omission failure $\Rightarrow$ The server loose its right to send soft messages
- After an empty window $\mathcal{W}_S$, all servers update the process group membership to the empty set

Summary

TLA+ Concepts I

Back

A computation of a system is represented by a sequence of states.

Definitions

State	$\triangleq$	Assignment of constant values to variables
Behavior	$\triangleq$	Sequence of states
Predicate	$\triangleq$	Formula which contains variables of only one state
Step $S : i \rightarrow f$	$\triangleq$	pair of consecutive states i and f
Transition function	$\triangleq$	Formula which contains variables of both state i e f
Action	$\triangleq$	Boolean-valued transition function on steps

TLA+ Concepts II

The prime ($'$) operator is used to distinguish the values of variables on steps. Let $S : i \rightarrow f$ be a step:

- v : value of v in state i
- v': value of v in state f

Example

- $v = 0$ in i and $v = 1$ in f ($v' = 1$)
- $[v' - v]$ is a transition function
- $[v' - v]$ equals 1 on S
- $[v' = v + 1]$ is an action true on S

Next-state relation

The next-state relation associated to step $S : i \longrightarrow f$ is the set of actions defined on S.

The $\Box$ operator

In a behavior Σ, an action A is true, denoted $\Box[A]_{vars}$ $\Leftrightarrow$ $\forall S : i \longrightarrow f$ of Σ that alters the set *vars* of all variables, A is true on S

Summary

DoRiS : The Specification Constants

Constants	Description
nServ	Number of *DoRiS* servers of a *DoRiS* segment
deltaChip	Duration (Δ_C) of a chip
delta	Transmission time (δ) of a hard message
pi	Processing time (π) of a hard message by the slowest device
maxTxTime	Transmission time of the largest message
horiz	Number of total cycles used to check a *DoRiS* model

SHARED

Field	Description
soft	set of servers participating of the soft communication.
chipTimer	increasing timer, ranging from 0 to Δ_C, that represents the flow of time during a chip.
chipCount	counter that identifies each chip, modulo *nTask*.
cycleCount	counter that identifies each cycle, modulo *nCycle*.
medium	if no message is being transmitted, *medium* equals $\{\}$. Else, *medium* stores the message being transmitted.
macTimer	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

Field	Description
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.
res	reservation list for the *nServ* next chips.
cons	counter that represents the number of elementary messages received in a complete *DoRiS* cycle.

SOFTSTATE

Field	Description
token	counter used to rule the token circulation during $\mathcal{W}_S$.
list	list of soft messages waiting to be transmitted.
count	the number of soft messages received during a $\mathcal{W}_S$ window.

HISTORY

Field	Description
elem	the number of elementary messages sent in a cycle.
rese	the number of reservation messages sent in a cycle.

Summary

Elementary Message Sending Action

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

Reservation Message Sending Action

Volta

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

Soft Message Sending Action

Volta

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

HARD MESSAGE RECEPTION - I

$$
\begin{array}{l}
HardRecv(m) \triangleq \\
\quad \wedge \ \text{CASE} \ \ m.res \neq \{-1\} \wedge m.softFlag = 1 \\
\qquad \rightarrow \quad \wedge Shared' = [Shared \ \text{EXCEPT} \ !.medium = \{\}, \ !.soft = @ \cup \{m.id\}] \\
\qquad\qquad \wedge \ \text{UNCHANGED} \ SoftState \\
\qquad \Box \ m.res \neq \{-1\} \wedge m.softFlag = 0 \\
\qquad \rightarrow \ \wedge Shared' = [Shared \ \text{EXCEPT} \ !.medium = \{\}, \ !.soft = @ \setminus \{m.id\}] \\
\qquad\qquad \wedge \ tokenUpdate(m) \\
\qquad \Box \ m.res = \{-1\} \\
\qquad \rightarrow \ \wedge Shared' = [Shared \ \text{EXCEPT} \ !.medium = \{\}] \\
\qquad\qquad \wedge \ \text{UNCHANGED} \ SoftState
\end{array}
$$

Continue

$$
\begin{array}{l}
\land\ HardState' = \\
\quad [i \in NoRecvSet(m) \mapsto HardState[i]]\ @@ \\
\quad [i \in ServID \setminus NoRecvSet(m) \mapsto [HardState[i] \text{ EXCEPT} \\
\qquad !.execTimer = \text{IF } Len(HardState[i].msg) = 0 \text{ THEN } pi \text{ ELSE } @, \\
\qquad !.cons[m.id] = \text{IF } m.res \neq \{-1\} \text{ THEN } 1 \text{ ELSE } @, \\
\qquad !.msg = Append(@, m), !.res = \\
\qquad\qquad \text{IF } m.res = \{-1\} \\
\qquad\qquad\quad \text{THEN } [j \in ServID \mapsto \text{IF } j = m.id \text{ THEN } -1 \text{ ELSE } @[j]] \\
\qquad\qquad\quad \text{ELSE } [j \in ServID \mapsto \text{IF } j \in m.res \text{ THEN } m.id \text{ ELSE } @[j]]]] \\
\land\ \text{UNCHANGED } History
\end{array}
$$

Soft Message Reception

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

Message Reception

Volta

$$
\begin{aligned}
&Receive(m) \triangleq \\
&\quad \wedge \quad Shared.macTimer = 0 \\
&\quad \wedge \quad \text{CASE} \quad m.type = \text{“hard”} \rightarrow HardRecv(m) \\
&\qquad\quad \square \; m.type \quad = \text{“soft”} \rightarrow SoftRecv(m)
\end{aligned}
$$

TIME PASSING - I

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

TIME PASSING - II

Volta

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

CORRECTNESS OF THE HARD RING

$$HardRingCorrectness \triangleq$$
$$\wedge \forall s \in HardRing : \Box(Len(HardState[hardID(s)].msg) \leq 3)$$
$$\wedge \Box(\text{ENABLED } NextChip \Rightarrow History.elem = Shared.chipCount)$$

CORRECTNESS OF THE RESERVATION MECHANISM

$$ReservationSafety \triangleq$$
$$\Box\forall chip, j \in ServID : \wedge \text{ENABLED } ReseSend(HardServ[j])$$
$$\wedge Shared.chipCount = chip$$
$$\Rightarrow \wedge HardState[j].res[chip] = j$$
$$\wedge \forall i \in ServID \setminus \{j\} : HardState[i].res[chip] \in \{j, -1\}$$

FAIRNESS OF THE SOFT RING

$$
\begin{aligned}
&\mathit{SoftRingFairness} \triangleq \\
&\quad \land \forall\, i \in \mathit{ServID} : \Box(i \in \mathit{Shared.soft} \\
&\qquad \Rightarrow (\mathit{SoftState}[i].\mathit{list} \neq \langle\rangle \Rightarrow \Diamond(i = \mathit{SoftState}[i].\mathit{token}))) \\
&\quad \land \Box\Diamond(\forall\, i \in \mathit{ServID} \setminus \mathit{Failed} : i \in \mathit{Shared.soft} \\
&\qquad\qquad \Rightarrow \mathit{Len}(\mathit{SoftState}[i].\mathit{list}) = 0)
\end{aligned}
$$