A CHARACTERIZATION OF THE DYNAMICAL
ASCENDING ROUTING

A. BOUABDALLAH  M. TREHEL

Laboratoire d'Informatique de Besançon, Université de Franche-Comté
16 route de Gray, 25030, Besançon Cedex
FRANCE

Abstract - We present a characterization of a routing
defined on a dynamical logical structure. By
uniting the structure with the messages
destabilizing it, we get an invariant which allows a
simple proof of the properties of the routing. We
exploit the results by designing and structuring the
proof of a mutual exclusion algorithm, which uses
this routing. The specifications are done in an
axiomatic manner and the proof is given in a
temporal framework.

Index Terms - distributed algorithm, routing techni-
que, formal specification, safety and liveness
properties, linear temporal logic, graph theory.

This research was financed by the University of
Franche-Comté, the C.N.R.S., associated unit 040822 and by
the C3 co-ordinated research program.

1. INTRODUCTION.

In this last decade, a significant work in the
distributed algorithms area led to a large class of
solutions to problems originating in the design of
distributed systems. Solutions to election, mutual
exclusion, stable properties detection ...[17] [18] give
a basis to a methodology by identifying the different
kinds of problems and forging some specific tools to
solve them [19]. Concerning the routing of
informations [3] [22], a simple but fruitful approach
consists in exploiting the topological properties of
the underlying network to define a distributed
schedule of the communication: a token on a ring
[11][12] or a diffusing computation on a tree [2] are
such approaches.

In this paper we study the properties of the
dynamical ascending routing (d.a.r). Initially this
technique has been used to solve the mutual exclusion
problem on a complete network [14] [13], then
adapted to an arbitrary network [15] [13] [5]. In fact,
one of its major interests [4] is to allow the
concurrent schedule of a waiting queue, distributed
differently from the classical approach of the
duplication [6], by partitioning it. The routing is
defined on a rooted tree, which allows every process
to transmit a message to the root; by sending a
message a process becomes a root. The transit of
messages disturbs globally the structure, which
becomes disconnected. We give an invariant, built on
the routing structure and on the messages in
transit, strong enough to characterize this routing.
Indeed it allows a simple proof of its safety and
liveness properties, namely:

- when messages are in transit, the routing
  structure is a forest of rooted trees
- when messages are moving no more, the rooted
  tree is restored again
- a sent message is received in a finite time by a
  root different from the producer and then stopped

In the second paragraph we give the principle,
the specification and the properties of the routing in
a complete network. We sketch a proof of a mutual
exclusion algorithm using this routing in section
three and we conclude by section four.

2. THE DYNAMICAL ASCENDING ROUTING
IN A COMPLETE NETWORK.

2.1. Principle.

A rooted tree is initially defined (fig.1).

A rooted tree can't send a message
if a node, neither being a root or waiting on an
acknowledgement, sends a message to its father, it
becomes then a root (fig.2) and waits on the
acknowledgement

\[ \text{fig.1} \]

\[ \text{fig.2} \]

The flow of messages is as following:
- a root can't send a message
- if a node, neither being a root or waiting on an
acknowledgement, sends a message to its father, it
becomes then a root (fig.2) and waits on the
acknowledgement
When receiving a message, a node:
- which is a root stops it and sends an acknowledgement to its producer, which becomes its father (fig.3, fig.5)

which is not a root, transmits it to its father. The producer of the message becomes its father (fig.4, fig.5)

When receiving an acknowledgement, a node waits no longer (fig.6)

2.2. Specification.

The technique we use, is essentially based on Lamport's works on formal specification methods [7] [8] [9]. The algorithm is described by a triple (B, S, A), where S is a set of global states, A a set of indivisible local actions and B a set of interleavings of the form $s_0 \rightarrow \alpha_1 \rightarrow s_1 \rightarrow \alpha_2 \rightarrow s_2 \rightarrow \ldots$, where $s_i$ belongs to S and $\alpha_i$ belongs to A. S is characterized by elementary state functions, which allow to extract from each state the relevant informations. A is exhaustively defined. The accepted behaviors of the algorithm are those, which verify a set of logical assertions, namely:
- the initial axiom giving the initial values of the state functions
- the transition axioms stating precisely how the actions can change the state functions
- the liveness axioms pointing out what changes must occur in each behavior of the algorithm

Lamport uses state and action predicates, which are boolean-valued functions defined respectively on S and on A. We use a slightly different definition of these predicates, being characteristic functions of subsets respectively of S and of A. The state predicates are built on the elementary state functions. Concerning the action predicates, we will use the following convention:
given an action $\alpha$ of A, the associate predicate is noted $\alpha$. Therefore a predicate P is true in x and we write $x = P$ or $P(x)$ if and only if x belongs to P. According to the finiteness of the ranges of the elementary state functions and the set A used in our study, we express the initial axioms and the transition axioms in a propositional calculus, by using the ordinary logical connectives $\neg$, $\land$, $\lor$, $\Rightarrow$, $\equiv$. The linear temporal operators $[I$ (always), $\diamond$ (eventually) and $\bigcirc$ (next time) are needed to express the liveness axioms [10][16]. We define inductively the meaning of formula P on a behavior $\sigma = s_0 \rightarrow \alpha_1 \rightarrow s_1 \rightarrow \alpha_2 \rightarrow s_2 \rightarrow \ldots$ $\sigma \vdash P$ by:

We suppose given:
- an integer $n > 1$
- a special symbol nil
- a set of informations $(m, \text{ack})$
We denote:

\[ n^* = \{ \text{nlu (nil), } 4 \} \]

\[ \mathcal{V} = \{ \text{m, ack} \} \times [n]^2 \] the set of messages

\[ \forall x \in \mathcal{V} : \text{pr}_1(x) = \text{info}(x) \]
\[ \text{pr}_2(x) = \text{prod}(x) \) (producer of x)
\[ \text{pr}_3(x) = \text{adr}(x) \) (addressee of x)

The set of actions \( A \) is:

\[ A = \{ \text{null, } \text{prod} \}_{i \in [n]} \]

Intuitively \( \text{prod}_i \) is the production and the sending of a message, the information of which is \( m \), by process \( i \); \( \text{cons}_i \) is the receipt and the consummation of a message by process \( i \) with a possible sending of a message.

The set of states \( S \) is characterized by the following elementary state functions:

\[ \forall i \in [n] : \text{father}_i \text{ of range } [n]^* \]

\[ \text{waiting}_i \text{ boolean} \]

\[ \forall j \in [n,i] : \text{C}(i,j) \text{ a subset of } \mathcal{V} \text{ ordered by a relation } \tau, \text{ with a greatest lower bound (g.l.b) and a least upper bound (l.u.b)} \]

\[ \text{C}(i,j) \text{ gives the state of the channel from } i \text{ to } j. \tau \text{ represents the order of the messages passed by the channel.} \]

\[ \inf(C(i,j)) = \text{g.l.b}(\tau) \]
\[ \sup(C(i,j)) = \text{l.u.b}(\tau) \]

**Initial axiom:**

\[ \text{Init}(s) = (\text{father}_1 = \text{nil}) \land \land_{i \in [n,1]}(\text{father}_i = 1) \]
\[ \land_{i \in [n]} \neg \text{waiting}_i \]
\[ \land_{i,j \in [n], i \neq j}(\text{C}(i,j) = \emptyset) \]

**Transition axioms:**

A set of triples (\( \alpha, \text{enabled}(\alpha), \text{R}(\alpha) \))\( \alpha \in A \) where:

\( \text{enabled}(\alpha) \) is the set of states of \( S \) where \( \alpha \) is enabled

\( \text{R}(\alpha) \) is a predicate defined on \( S \times S \)

\( (\text{null, enabled(null)}, \text{R(null)}) \)

\( \text{enabled(null)} = \text{true} \)
\( \text{R(null)}(s,t) = s = t \)

\[ \forall i \in [n] : \]
\[ \text{R} \text{(Prod}_i, \text{R} \text{(Prod}_i) \}

\[ \text{enabled(Prod}_i(s) = \text{father}_i(s) = \text{nil} \land \neg \text{waiting}_i(s) \]
\[ \land \text{waiting}_i(t) \land \#(i, \text{father}_i(s)) + 1 = \#(i, \text{father}_i(s)) \]
\[ \land \text{inf}(i, \text{father}_i(s)) \]
\[ \land \text{inf}(i, \text{father}_i(s)) \]
$X/\cup_1U$ is the set of connected parts of $U$. $U$ is connected (resp. empty) if and only if $X/\cup_1U = \{ X \}$ (resp. $X/\cup_1U = \emptyset$). The cyclomatical number of $U$, $Q(U)$ is defined by:

$$Q(U) = \#(U) + \#(X/\cup_1U) - \#(X)$$

$U$ is without cycle if and only if $Q(U) = 0$. $U$ is a tree if and only if it is connected without cycle. If $U$ is oriented the external half degree (resp. internal half degree) of $U$ is the function

$$\partial^+_U \ (resp. \partial^-_U): X \rightarrow \mathbb{N}$$

such that

$$\partial^+_U(i) = \#(\{(i)x \cup \{x\} \cap U\})$$

(resp. $\partial^-_U(i) = \#(\{(X\cup\{i\})x \cap \{i\} \cap U\}$)

$U$ is a rooted tree if and only if:

- it is a tree
- it exists a node $i$, called the root of $U$, such that $\partial^+_U(i) = 0$
- $\forall j \in X \setminus \{i\}: \partial^+_U(j) = 1$

$U$ is a forest of rooted trees if and only if:

- $X/\cup_1U = \{ X_1, \ldots, X_k \}$ k > 1
- if $U X_i$ is the restriction of $U$ to $X_i$ then
  - $U$ is a tree
  - $\forall X_i \in X/\cup_1U, U X_i$ is a rooted tree

We define the following state functions:

- $\text{FATHER}(s)(i) =$ father$\text{FATHER}(s)(i)$
- $\text{TRANSIT}(s)(i,j) =$ C(i,j)(s) n 2(m)x[n]x[n]

We associate to:

- $\text{FATHER}(s)$ an oriented graph $\text{FATHER}(s)$ defined on $[n]$ such that the edge from $i$ to $j$, noted $(i,j)$ belongs to $\text{FATHER}(s)$ if and only if $\text{FATHER}(s)(i) = j$
- $\text{TRANSIT}(s)$ a labelled oriented graph $\text{TRANSIT}(s)$ defined on $[n]$ such that, each element of $\text{TRANSIT}(s)(i,j)$, when $\text{TRANSIT}(s)(i,j) \neq \emptyset$, is the label of an oriented edge from $i$ to $j$.

Let the following state function and predicates:

- $\text{ROOT}(\text{FATHER})(s) =$ (i $\in [n]$ \& $\text{FATHER}(s)(i) = \text{nil}$)
- $\text{ROOTED TREE}(\text{FATHER}) =$ $\text{FATHER}$ is a rooted tree
- $\text{ROOTED TREES FOREST}(\text{FATHER}) =$ $\text{FATHER}$ is a forest of rooted trees
- $\text{EMPTY}(\text{TRANSIT}) =$ $\text{TRANSIT}$ is empty
- $\text{TREE}(\text{FATHER}, \text{TRANSIT}) =$ $\text{FATHER}$ U $\text{TRANSIT}$ is a tree

Safety properties of the d.a.r

We use the induction principle [10][16] given by the following rule IP:

let be a behavior of $B \sigma = s_0 \rightarrow \alpha_1 \rightarrow s_1 \rightarrow \alpha_2 \rightarrow s_2 \ldots$ and a state predicate $P$ then

$$(s_0 \models P) \land (\forall i \geq 0: s_i \models P \Rightarrow s_{i+1} \models P)$$

\[ IP: \] $\sigma \models [\] P

$\text{lemma1:}$ $\models \text{TREE}(\text{FATHER}, \text{TRANSIT})$

proof from $\models \text{Init}$ we get $\models \text{ROOTED TREE}(\text{FATHER})$ and $\models \text{EMPTY}(\text{TRANSIT})$

$\text{theorem1:}$ $\models [\{ \text{TREE}(\text{FATHER}, \text{TRANSIT}) \}$

proof by lemma1 and IP it suffices to show that:

$\forall \sigma \in B \sigma = s_0 \rightarrow \alpha_1 \rightarrow \cdots \rightarrow s_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow s_2 \ldots$

$\forall i \geq 0: s_i \models \text{TREE}(\text{FATHER}, \text{TRANSIT}) \supset s_{i+1} \models \text{TREE}(\text{FATHER}, \text{TRANSIT})$

We get it by structural induction on $A$, for example if we have $R(\text{Prod}_1(s_i(s_{i+1}))$ then

$\text{FATHER}(s_i) U \text{TRANSIT}(s_{i+1}) =$

$(\text{FATHER}(s_i) - (i, \text{father}(s_i))) U \text{TRANSIT}(s_i)$

$U \{ (i, \text{father}(s_i))$ labelled by $(m,j, \text{father}(s_j))$ 

So $\text{FATHER}(s_{i+1}) U \text{TRANSIT}(s_{i+1})$ is connected and

$\#(\text{FATHER}(s_i) U \text{TRANSIT}(s_i)) =$ $\#(\text{FATHER}(s_{i+1}) U \text{TRANSIT}(s_{i+1}))$

$\Omega(\text{FATHER}(s_i) U \text{TRANSIT}(s_i)) = 0$, from which we deduce $s_{i+1} \models \text{TREE}(\text{FATHER}, \text{TRANSIT})$. The other cases are proved in the same manner.

$\text{corollary1:}$ $\models [\{ \#(\text{ROOT}(\text{FATHER})) = 1 + \#(\text{TRANSIT}) = \#([n]/\cup \text{FATHER}) \}$

proof from theorem1 we get $\models [\{ \#(\text{FATHER} U \text{TRANSIT}) = 0 \land \#([n]/\cup \text{FATHER} U \text{TRANSIT}) = 1 \}$ the result follows by applying the obvious properties

$\models [\{ \#(\text{ROOT}(\text{FATHER})) = n - \#(\text{FATHER}) \}$ and

$\models [\{ \#(\text{FATHER} U \text{TRANSIT}) = 0 \Rightarrow \Omega(\text{FATHER}) = \Omega(\text{TRANSIT}) = 0 \}$

$\text{theorem2:}$

$\ast \models [\{ \neg \text{EMPTY}(\text{TRANSIT}) \Rightarrow \text{ROOTED TREE}(\text{FATHER}) \}$

$\ast \ast \models [\{ \neg \text{EMPTY}(\text{TRANSIT}) \Rightarrow \text{ROOTED TREES FOREST}(\text{FATHER}) \}$

proof obvious by applying corollary1
liveness properties of the d.a.r

**Theorem 1:**

\[ \forall i \in [n] \left[ \prod_i \Rightarrow \emptyset \lor \exists j \in [n, i] \right] \]

(\(\text{father}_j = \text{nil} \land \text{Cons}_j \land \)

\((m, i) \in \left\{ \sup(C(u, j), u \in [n, j]) \right\} \land \)

\(\Theta((m, i) \lor \left( \sup(C(u, j), u \in [n, j]) \right)) \) )

**Proof:** (outline)

The theorem tells us that a produced message is consummated by a root different from the producer in a finite time and then stopped. If it is not the case, it will be consummated infinitely often, so at least twice by a same node. From theorem 1 an edge \((i, j)\) in \(\text{TRANSIT}\) determines two subtrees of \(\text{FATHER} \cup \text{TRANSIT}\) \(([n_1], U_1)\) and \(([n_2], U_2)\) such that \(i\) belongs to \([n_1]\), \(j\) belongs to \([n_2]\), \([n_1]\) and \([n_2]\) are disjoint and \(\text{FATHER} \cup \text{TRANSIT}\) is the union of \(U_1, U_2\) and \((i, j)\). The consummation by \(j\) of the message labelling \((i, j)\) creates a new edge \((p, j)\) in \(\text{TRANSIT}\) labelled by a message having the same information and producer as the previous one and \(k\) as addressee. The edge \((p, j)\) determines two other subtrees \(([n_1'], U_1')\) and \(([n_2'], U_2')\) such that \(i\) and \(j\) belong to \([n_1']\), \(k\) belongs to \([n_2']\), \([n_1']\) contains \([n_1]\) and \([n_2']\) contains \([n_2]\). Just before \(j\) consummates the message a second time, the edge \((p, j)\) labelled by the message determines two subtrees \(([n_1''], U_1'')\) and \(([n_2''], U_2'')\) such that \(j\) belongs to the intersection of \([n_1'']\) and \([n_2'']\) therefore \(U_1'' \cup U_2''\) is connected; on the other hand \(U_1'' \cup U_2'' = \text{FATHER} \cup \text{TRANSIT} \cdot ((p, j))\) is not connected by theorem 1 leading to a contradiction. Therefore the message is necessarily stopped within a finite time.

**Corollary 2:**

\[ \exists \exists j \in [n] \left[ \text{waiting}_i \Rightarrow \emptyset \lor \text{waiting}_j \right] \]

**Proof:** from theorem 3 and \(R(\text{Cons}_j)\)

3. **ANALYSIS OF A MUTUAL EXCLUSION ALGORITHM USING THE D.A.R.**

3.1. **Principle.**

We study the Naimi-Trehel mutual exclusion algorithm [14], the main techniques of which are:

- A dynamical ascending routing of messages requesting the critical section
- A moving token, which unifies the control of the critical section [11] [21]
- A waiting queue distributed by partitioning, to schedule the requesters

We recall briefly the principle. Initially the processes are structured by a rooted tree, the root holds the token and the waiting queue is empty. The head of the queue is associated to a node holding the token and requesting the critical section, the rear to a requester which is a root. When requesting messages are in transit no more, the queue is totally organized in a way that each requester knows only the identity of the following process in the queue.

3.2. **Specification.**

We enrich the specification of the d.a.r with some modifications. The set of informations is \(\{\text{req,token}\}\).

The set of actions is \(A = \{ \text{null}, (\text{REQ-CS}_i)_i \in [n], (\text{REL-CS}_i)_i \in [n], (\text{Cons}_i)_i \in [n] \} \). \(\text{REQ-CS}_i\) and \(\text{REL-CS}_i\) are respectively the requesting and the releasing of the critical section by process \(i\). The elementary state functions are:

\[ \forall i \in [n] : \text{father}_i, \text{next}_i \text{ of range } [n] \]

\(\text{request}_i, \text{present-token}_i : \text{boolean} \)

\(\forall j \in [n, i] : (C(i, j)) \text{ a subset of } V \text{ ordered by a relation } \tau, \text{ with a g.l.b and a l.u.b.} \)

\(\inf(C(i, j)) = \text{g.l.b}(\tau) \)

\(\sup(C(i, j)) = \text{l.u.b}(\tau) \)

**Initial axiom:**

\(\text{Init}(s) \equiv (\text{father}_1 = \text{nil} \land \text{token-present}_1) \)

\(\wedge \quad \exists i \in [n] (\text{father}_i = 1 \land \text{~token-present}_i) \)

\(\wedge \quad \exists i \in [n] (\text{~request}_i \land \text{next}_i = \text{nil}) \)

\(\wedge \quad \exists i, j \in [n, i] (C(i, j) = \emptyset) \)

**Transition axioms:**

\(\forall i \in [n] : \)

\(\text{REL-CS}_i \equiv (\text{REL-CS}_i)_i \in [n] \)

\(. \text{enabled}(\text{REL-CS}_i)(s) \equiv \text{~request}_i(s) \)

\(. \text{R}(\text{REL-CS}_i)(s, t) = \text{enabled}(\text{REL-CS}_i)(s) \land \text{request}_i(t) \land \)

\[ \left[ (\text{father}_i(s) = \text{father}_i(t) = \text{nil}) \right. \)

\(\left. \lor (\text{father}_i(s) \neq \text{nil} \land \text{father}_i(t) = \text{nil}) \right. \)

\(\wedge \quad \#C(i, \text{father}_i(s))(s) + 1 = \#C(i, \text{father}_i(s))(t) \)

\(\wedge \quad \text{inf}[C(i, \text{father}_i(s))(t)] = (\text{req}, \text{father}_i(s)) \)

\]
3.3. Correctness of the algorithm.

Stating the correctness of a mutual exclusion algorithm consists in proving the mutual exclusion property itself and the absence of starvation [17]. Starvation or absence of fairness means that a process can wait indefinitely to enter a critical section, although others processes are entering and leaving them [20]. When starvation covers all the network, we get a deadlock. Fairness implies the deadlock freedom.

We use the previous notions and define the following state functions:

\[ \text{NEXT}(s) = \text{next}_i(s) \]

\[ \text{TRANSIT-req}(s)(i,j) = \text{C}(i,j)(s) \]

\[ \text{TRANSIT-token}(s)(i,j) = \text{C}(i,j)(s) \]

We associate to:

- NEXT(s) an oriented graph NEXT(s) defined on \([n]\) such that the edge from \(i\) to \(j\), noted \((i,j)\), belongs to \(\text{NEXT}(s)(i)\)

- TRANSIT-req(s) (resp. TRANSIT-token(s)) a labelled oriented graph TRANSIT-req(s) (resp. TRANSIT-token(s)) defined on \([n]\) such that each element of TRANSIT-req(s)(i,j) (resp. TRANSIT-token(s)(i,j)) is the label of an oriented edge from \(i\) to \(j\).

Let be the following state functions and predicates:

- \(\text{REQ-N-TOK}(s) = \{ i \in [n] / \text{request}_i(s) \land \text{present-token}_i(s) \} \)

- \(\text{TOK}(s) = \{ i \in [n] / \text{present-token}_i(s) \} \)

- \(\text{M-EX}(s) = \bigwedge_{i=1}^{n-1} \text{(present-token}_i \land \text{present-token}_{i+1}) \)

\(\text{theorem} \quad \text{(mutual exclusion)}\)

\[ \text{I} = \bigwedge_{i=1}^{n} \text{M-EX}(s) \]

\[ \text{proof} \quad \text{by using the induction rule IP} \]

\[ \text{lemma} \quad \text{2} \]

\[ a = [\leq (\Omega, \text{NEXT}) = 0] \]

\[ b = [\leq (\text{#(NEXT)} = \text{#(REQ-N-TOK)} + \text{#(TOK)}) \land \text{#(ROOT)}(\text{FATHER})] \]

\[ c = [\leq (\text{#(NEXT)} \land (0 \leq (\text{NEXT}) \leq 1) \land (0 \leq (\text{NEXT}) \leq 1)) \]

\[ d = [\leq (\text{#(NEXT)} + \text{#(TRANSIT-req)} + \text{TRANSIT-token}) = \text{#(REQ-N-TOK)}] \]

\[ e = [\leq (\text{next}_i \neq \text{nil} \land \text{request}_i \land \text{father}_i \neq \text{nil})] \]

\[ f = [\leq (\text{next}_i \neq \text{nil} \land \text{present-token}_i \land \text{request}_i \land \text{next}_i = \text{nil})] \]

242
Droof we get:
\[ a, b, c \]
using the rule IP
d from corollary1, theorem4 and lemma2
\[ e, f \] and \[ g \] from the specification

\[ \text{lemma 3:} \]
\[ \exists a, b : X \in \{n\} \wedge \#(X) > 1 \] \[ \Omega(\text{NEXT } \setminus X) = 0 \wedge \]
\[ \exists a, b : X : \# \text{NEXT}(a) = \# \text{NEXT}(b) = 0 \wedge \]
\[ \text{next}_i = \text{j} \] \[ \text{request}_i \wedge \neg \text{present-token}_j \] \[ \text{proof:} \]
follows from lemma2 a c e f

\[ \text{theorem 5:} \]
\[ \exists a, b : X : X = \{n\} \wedge \text{father} = \text{nil} \]
\[ \text{next}_i = \text{nil} \] \[ \text{proof (outline)} \]
if such a \( j \) doesn't exist, the state of process \( i \) won't change and \( \text{REQ-N-TOK} \) won't decrease. By theorem3, the requesting messages of the processes belonging to \( \text{REQ-N-TOK} \) will be consummated within a finite time by roots different from \( i \).

By:
\[ \text{lemma 2} \]
\[ \text{f ROOT} \]
\[ \text{(FATHER)} \] - \( i \) is included in \( \text{REQ-N-TOK} \).

\[ \text{lemma 2} \]
\[ \text{e next}_i = \text{nil} \]

\[ \text{lemma 2} \] \[ \text{g} \] for \( j \) belonging to \( [n,i] \), \( \text{next}_j \) \( \neq i \)

\[ \text{Therefore NEXT} \]
\[ \text{if} \] - \( \text{REQ-N-TOK} \) \[ \text{and} \]
\[ \#(\text{NEXT}) = 0 \] \[ \text{and} \]
\[ \#(\text{NEXT} \setminus \text{REQ-N-TOK}) = 0 \] \[ \text{by lemma 2} \]
\[ \text{d} \]
\[ \text{Next} \]
\[ \text{if} \] \[ \text{REQ-N-TOK} \] \[ \text{and} \]
\[ \text{Proof (outline)} \]
by theorem3 and lemma3, a starvation would cover within a finite time all the network leading to a deadlock

4. CONCLUSION.

By uniting the logical rooted tree with the messages destabilizing it, we were led to formulate an invariant, which allows a simple proof of the properties of the dynamical ascending routing. We have studied the routing on a complete network, however when adapting it to an arbitrary topology, our results remain valid (in a preparing paper). It seems to be a reasonable characterization of this routing. Moreover it gives a basis to derive different classes of d.a.r by means of the invariants they involve.

REFERENCES