Development of a Network Topology Discovery Algorithm

by Thai Son Hoang, David Basin, Hironobu Kuruma, and Jean-Raymond Abrial

1 The Master and Dog Paradigm

Topology discovery is a distributed algorithm that is at the core of several routing algorithms, such as link-state routing. It is the problem of each node in the network discovering and maintening information on the network topology. The problem is challenging as the network can change rapidly, indeed more rapidly than the nodes can track and account for the changes.

The topology discovery algorithm we develop generalizes of the "master and dog" paradigm. Here is the story. A master rides a motorbike while his dog, running behind him, tries to get to him. If the master reduces the speed of the motorbike, then the dog come a bit closer to him. However, if the master accelerates, then the distance between the two increases. But, certainly enough, if the master stops for a sufficiently long period of time then the dog will reach eventually his master.

In our problem, the master is a graph representing the communication structure of a network. A move of the master corresponds to a modification of this graph. What makes our case different from the basic paradigm is that we have many dogs, each of them being a node in the graph. The distance between each dog and the master is represented by the difference between the physical graph and the image of it that each node constructs as soon as it gets some information about the "position" of the master. By definition, we say that a dog-node has reached the master-graph when the local image of the graph in this node corresponds exactly to the real situation of the graph.

In the case of the simple "master and dog" story, the dog reacts to the moves of the master because he can see the master. In our case, the situation is a bit more complicated. Only certain nodes can "see" certain moves of the graph. In other words, the nodes as a community get all information about the moves of the graph, but each individual node has only a partial direct access to these moves. More precisely, a node n gets direct information about the modification of a link l of the physical network if n is the arriving node of link l.

Besides the limited information that it can get directly on the graph, each node can acquire more information about the graph from its neighbors (note that these neighbors changes over times as the graph evolves), which themselves either get some direct information on the graph or more information from their neighbors, and so on.

We would like to prove a result analogous to the one we have between the master and the dog, namely that the dog can reach his master if the latter stays still for a while. In our case, we shall prove that, *under certain conditions*, all dog-nodes can reach the master-graph if the latter stays still long enough.

2 Requirements for the Topology Discovery Problem

In this section, we define the requirements of our problem with great care. Note that we do not state any solution, rather we express the constraints that must be guaranteed by a potential solution for it to be considered correct. In the formal development of our solution in subsequent sections, we shall record the place where we take accounts (totally or partially) of these requirements.

We are given a finite set of nodes connected by oriented links which can be added or removed as time goes, thus forming a *dynamic graph*	REQ-1

Each node builds an image of the graph either by getting direct information from it (cf. REQ-3) or indirect information from its neighbors (cf. REQ-4)	REQ-2

When a link from node a to node b is added to or removed from the graph then node b is directly made aware of it	REQ-3

The neighbors of a node b are the nodes that are connected to b by a link entering in b. Neighbors of b send to b their local networks	REQ-4

Nodes communicates by means of messages sent on links of the graph	REQ-5

A message sent from a to b is lost if the link from a to b, which existed when the message was sent, is broken before the message reaches b	REQ-6

We must prove that under certain conditions the images of the graph built by nodes are all identical and equal to the graph itself	REQ-7

3 Development Strategy

Our development comprises an initial model followed by seven refinements. Here is a brief description of each of them:

- The initial model only takes care of the first requirement defining the physical network.
- In refinement 1, we introduce the logical network. It corresponds to the idea that the physical network is transmitted to all nodes as a community.
- In refinement 2, we introduce explicitly the local networks of each node. This is done together with a very abstract way of gradually transmitting the contents of the logical network to the local networks.
- In refinement 3, we introduce modification times of the logical network as well as modification times of the local networks. This allows us to simplify the formalization.
- Refinement 4 is technical: it concerns the merging of various events.
- In refinement 5, we formally define the neighbors of a node and thus take full account of the fourth requirement.
- In refinement 6, we prove our main result, corresponding to the seventh requirement.
- In the last refinement, we take account of the message handling, corresponding to requirements five and six.

4 Formal Development

4.1 Initial Machine

This initial machine takes account of requirement REQ-1.

We are given a finite set of nodes connected by oriented links that can be added or removed as time goes, thus forming a *dynamic graph*	REQ-1 r

Communications channels between nodes of the *physical* network form a graph, which can be represented by a finite binary relation. At this stage however, we shall not formalize our graph by means of a binary relation. In fact, it is more convenient to represent the graph by a subset of the set L of all possible *links* between two different nodes:

sets: L

axm0_1: $\mathrm{finite}(L)$

The physical network NET is thus a variable defined as a simple subset of L:

variables: NET

inv0_1: $NET \subseteq L$

At this stage, we have two events: adding or removing a link from the physical NET. This corresponds to the dynamic evolution of the connections between the nodes. Our topology discovery algorithm has no influence on these external evolutions.

```
init
  NET := ∅
```

```
Modify_up
  any
    l
  where
    l ∉ NET
  then
    NET := NET ∪ {l}
  end
```

```
Modify_dn
  any
    l
  where
    l ∈ NET
  then
    NET := NET \ {l}
  end
```

4.2 First Refinement

In this refinement, we take account of requirements REQ-2 and REQ-3 in a very abstract way. More will be said on these requirements in the next refinement.

Each node builds an image of the graph either by getting direct information from it (cf. REQ-3) or indirect information from its neighbors (cf. REQ-4)	REQ-2

When a link from node a to node b is added to or removed from the graph then node b is directly made aware of it	REQ-3

To this end, we introduce the *logical* network net. It represents the abstract knowledge that the nodes as a community may obtain "somehow" on the overall situation of the physical network. The variable net is thus a subset of L similar to NET. Notice again that the logical network net is a convenient abstraction which will be given later a concrete definition in refinement 5.

variables: net

inv1_1: $net \subseteq L$

Besides the two previous events, we formalize two more events adding or removing a link from the logical net following the evolution of NET.

```
init
  NET := ∅
  net := ∅
```

```
discover_up
  status
    convergent
  any
    l
  where
    l ∈ NET \ net
  then
    net := net ∪ {l}
  end
```

```
discover_dn
  status
    convergent
  any
    l
  where
    l ∈ net \ NET
  then
    net := net \ {l}
  end
```

Note that these events do not make any node aware of the link that is added to or removed from the graph as is mentioned explicitly in requirement REQ-3. This will be done in the next refinement. For the moment, they only update the logical network net. These events are convergent: this is proved by means of the following variant denoting the symmetric difference between the sets NET and net:

variant1: $(NET \setminus net) \cup (net \setminus NET)$

This convergence means that if the physical network NET is not modified for a certain time then the logical network net will eventually be the same as the physical network NET (master and dog).

4.3 Second Refinement

In this refinement, we take further account of requirement REQ-3: an arbitrary node is made aware of the change when a modification in the graph has occurred. Moreover, we also take further account of requirement REQ-2 dealing with the image of the graph that is built by each node.

To this end, we define the *local* networks. Each of them represents the individual knowledge that each node acquires about the physical network. It corresponds to the gradual spreading of the logical network net to each node of the physical network. In order to formalize this, we introduce first the finite set of nodes N, which we do not yet relate to the links L.

sets: N

axm2_1: $\text{finite}(N)$

Here is the definition of the local networks l_net: it is a binary relation from N to L. More precisely, the set of links associated with node n is the image $l\_net[\{n\}]$ of the singleton set $\{n\}$ under the relation l_net.

variables: l_net

inv2_1: $l\_net \in N \leftrightarrow L$

When a link modification is detected by the events discover_up or discover_dn, then some *messages* to all nodes (except the node that is directly made aware of the modification) are put in two big reservoirs called m_net_up (for establishing a new link) and m_net_dn (for removing a link). Notice that the message handling done in these reservoirs does not correspond at all to the final message handling that we have in the real algorithm (presented in refinement 7). This is only a convenient abstraction which we use for the moment.

variables: m_net_up
m_net_dn

inv2_2: $m\_net\_up \in N \leftrightarrow L$

inv2_3: $m\_net\_dn \in N \leftrightarrow L$

Moreover at this stage, a modification of one reservoir is done "magically" by removing old messages that can be still waiting in the other (see the updates of the events discover_up and discover_up below). As a consequence, m_net_up and m_net_dn are disjoint:

inv2_4: $m\_net\_up \;\cap\; m\_net\_dn = \varnothing$

Besides the event init, the events discover_up and discover_dn are updated as follows:

init
  $NET := \varnothing$
  $net := \varnothing$
  $l\_net := \varnothing$
  $m\_net\_up := \varnothing$
  $m\_net\_dn := \varnothing$

discover_up
  any
    l
    n
  where
    $l \in NET \setminus net$
    $n \in N$
  then
    $net := net \cup \{l\}$
    $m\_net\_up := m\_net\_up \;\cup\; ((N \setminus \{n\}) \times \{l\})$
    $m\_net\_dn := m\_net\_dn \;\setminus\; (N \times \{l\})$
    $l\_net := l\_net \cup \{n \mapsto l\}$
  end

As can be seen, a node n is chosen (arbitrarily for the moment) and the local network in that node is updated. This arbitrary node will be made precise in refinement 5. A similar behavior can be observed for the next event:

discover_dn
any
l
n
where
$l \in net \setminus NET$
$n \in N$
then
$net := net \setminus \{l\}$
$m\_net\_dn := m\_net\_dn \ \cup \ ((N \setminus \{n\}) \times \{l\})$
$m\_net\_up := m\_net\_up \ \setminus \ (N \times \{l\})$
$l\_net := l\_net \setminus \{n \mapsto l\}$
end

In this refinement, which is still very abstract, nodes have free access to the reservoirs m_net_up and m_net_dn. Thus they can update their local knowledge. This is done by two additional events named change_link_up and change_link_dn. Notice that these events have the status anticipated, meaning that their convergence must be proved in further refinements.

change_link_up
status
anticipated
any
n
l
where
$n \mapsto l \in m\_net\_up$
then
$l\_net := l\_net \ \cup \ \{n \mapsto l\}$
$m\_net\_up := m\_net\_up \ \setminus \ \{n \mapsto l\}$
end

change_link_dn
status
anticipated
any
n
l
where
$n \mapsto l \in m\_net\_dn$
then
$l\_net := l\_net \ \setminus \ \{n \mapsto l\}$
$m\_net\_dn := m\_net\_dn \ \setminus \ \{n \mapsto l\}$
end

What has been presented so far is illustrated in figure 1.

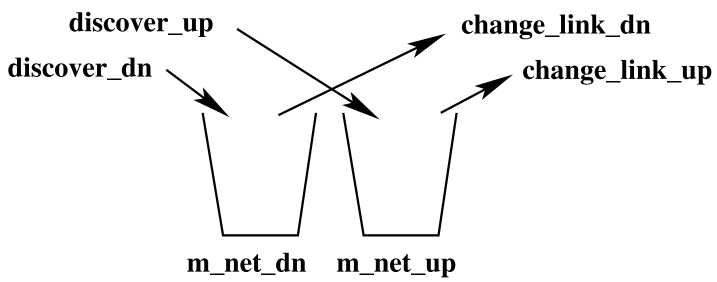

Fig. 1. Handling Abstract Messages

All four events of this refinement maintain the following invariants between net, l_net, m_net_up and m_net_dn:

inv2_5:	$\forall n, l \cdot l \in net \;\wedge\; n \mapsto l \notin l\_net \;\Rightarrow\; n \mapsto l \in m\_net\_up$
inv2_6:	$\forall n, l \cdot l \in net \;\wedge\; n \mapsto l \in l\_net \;\Rightarrow\; n \mapsto l \notin m\_net\_dn$
inv2_7:	$\forall n, l \cdot l \notin net \;\wedge\; n \mapsto l \in l\_net \;\Rightarrow\; n \mapsto l \in m\_net\_dn$
inv2_8:	$\forall n, l \cdot l \notin net \;\wedge\; n \mapsto l \notin l\_net \;\Rightarrow\; n \mapsto l \notin m\_net\_up$

Figure 2 gives an informal explanation of refinement **inv2_5**. When a link l is present in net and absent from l_net for node n, then it must be present in the message reservoir m_net_up for node n.

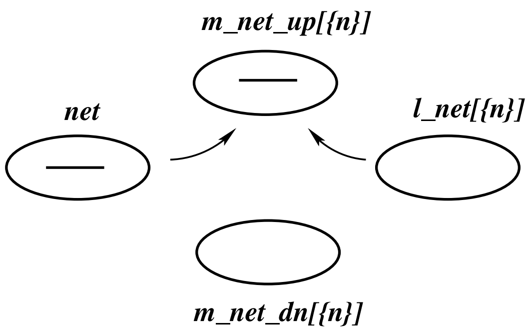

Fig. 2. Informal Explanation of Invariant **inv2_5**

Figure 3 gives a similar explanation for invariant **inv2_6**. When a link l is present in both net and l_net for node n, then it must be absent from the message reservoir m_net_dn for node n.

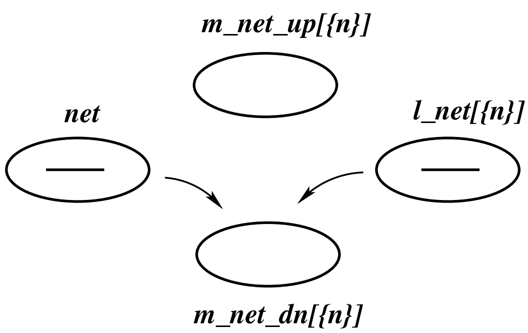

Fig. 3. Informal Explanation of Invariant **inv2_6**

We have similar explanations for invariants **inv2_7** and **inv2_8**. Finally, we add the following new anticipated event, which will be explained in the next refinement. At this stage, we just ensure that this event maintains invariants non-deterministically:

change_link_2
status
anticipated
any
ln
where
$ln \in N \leftrightarrow L$
$\forall n, l \cdot l \in net \;\wedge\; n \mapsto l \notin ln \;\Rightarrow\; n \mapsto l \in m\_net\_up$
$\forall n, l \cdot l \in net \;\wedge\; n \mapsto l \in ln \;\Rightarrow\; n \mapsto l \notin m\_net\_dn$
$\forall n, l \cdot l \notin net \;\wedge\; n \mapsto l \in ln \;\Rightarrow\; n \mapsto l \in m\_net\_dn$
$\forall n, l \cdot l \notin net \;\wedge\; n \mapsto l \notin ln \;\Rightarrow\; n \mapsto l \notin m\_net\_up$
then
$l\_net := ln$
end

4.4 Third Refinement

This refinement is not concerned with any requirement; it is instead a data-refinement. First we formally define the concept of $parity$ for natural numbers:

constants: $parity$

axm3_1: $parity \in \mathbb{N} \rightarrow \{0, 1\}$

axm3_2: $parity(0) = 1$

axm3_3: $\forall x \cdot parity(x + 1) = 1 - parity(x)$

Now the magic abstract behavior of the variables m_net_up and m_net_dn of previous refinement is refined. For this, we add a variable age, denoting the "date" when the logical network net has been modified:

variables: age

inv3_1: $age \in L \rightarrow \mathbb{N}$

inv3_2: $\forall l \cdot l \in L \;\Rightarrow\; (l \in net \;\Leftrightarrow\; parity(age(l)) = 1)$

Invariant **inv3_2** might seem a bit strange. It is due to the fact that the presence or absence of a link in net changes simultaneously when the age variable is incremented (see the updates of events discover_up and discover_dn below). The parity of the variable age thus determine the absence or presence of a link in net: this allows us to remove the variable net since we can *easily reconstruct it* from the variable age. Similarly, we introduce a variable l_age, recording the local age of a link in the variable l_net:

variables: l_age

inv3_3: $l\_age \in N \times L \rightarrow \mathbb{N}$

inv3_4: $\forall l \cdot l \in L \Rightarrow l\_age[N \times \{l\}] \subseteq 0 \,..\, age(l)$

inv3_5: $\forall n, l \cdot \; n \in N \wedge l \in L \Rightarrow (n \mapsto l \in l\_net \Leftrightarrow parity(l\_age(n \mapsto l)) = 1)$

Invariant **inv3_4** states that local ages are less than or equal to ages. In other words, $age(l)$ is the maximum of all local ages concerning link l. Moreover, invariant **inv3_5** expresses a property for local ages and local networks similar to the one that we expressed for the variables age and net in invariant **inv3_2**. As a consequence and for the same reason as for variable net, we can remove variable l_net. Finally, we have also to introduce ages in the messages. For this we define a variable m_net. As we shall see below, it will replace the abstract reservoirs m_net_up and m_net_dn:

variables: m_net

Variable m_net is subject to the following invariants:

inv3_6: $m\_net \in N \times L \leftrightarrow \mathbb{N}$

inv3_7: $\forall l \cdot l \in L \Rightarrow m\_net[N \times \{l\}] \subseteq 0 \,..\, age(l)$

inv3_8: $\forall n, l \cdot \; n \in N \wedge l \in L \Rightarrow (n \mapsto l \in m\_net\_up \Leftrightarrow n \mapsto l \mapsto age(l) \in m\_net \wedge parity(age(n \mapsto l)) = 1)$

inv3_9: $\forall n, lr \cdot \; n \in N \wedge l \in L \Rightarrow (n \mapsto l \in m\_net\_dn \Leftrightarrow n \mapsto l \mapsto age(l) \in m\_net \wedge parity(age(n \mapsto l)) = 0)$

Invariants **inv3_8** and **inv3_9** establish the connection between m_net and both m_net_up and m_net_dn. They explain that a link in m_net_up or m_net_dn corresponds to the most recent link in m_net together with the right parity. As a consequence, variables m_net_up and m_net_dn can be removed. The next two invariants say that $age(l)$ is either in l_age or in m_net, but not in both:

inv3_10: $\forall n, l \cdot age(l) = l\_age(n \mapsto l) \vee n \mapsto l \mapsto age(l) \in m\_net$

inv3_11: $\forall n, l \cdot n \mapsto l \mapsto l\_age(n \mapsto l) \notin m\_net$

The next invariant states that if a local age is strictly less than an age, then intermediate ages are waiting in m_net:

$$\textbf{inv3\_12:}\quad \forall n,l\cdot\ age(n \mapsto l) < age(l) \Rightarrow l\_age(n \mapsto l)+1\,..\,age(l) \in m\_net[\{n \mapsto l\}]$$

Next are the updates of the events:

init
$NET := \varnothing$
$age := L \times \{0\}$
$l\_age := N \times L \times \{0\}$
$m\_net := \varnothing$

discover_up
any
l
n
where
$l \in NET$
$parity(age(l)) = 0$
$n \in N$
then
$age(l) := age(l) + 1$
$m\_net := m\_net \cup$
$((N \setminus \{n\}) \times \{l\} \times \{age(l)+1\})$
$l\_age(n \mapsto l) := age(l) + 1$
end

discover_dn
any
l
n
where
$l \notin NET$
$parity(age(l)) = 1$
$n \in N$
then
$age(l) := age(l) + 1$
$m\_net := m\_net \cup$
$((N \setminus \{n\}) \times \{l\} \times \{age(l)+1\})$
$l\_age(n \mapsto l) := age(l) + 1$
end

We notice that the actions of these events are identical. Hence we shall be able to merge them in the next refinement.

change_link_up
status
convergent
any
n
l
x
where
$x = age(l)$
$n \mapsto l \mapsto x \in m\_net$
$parity(x) = 1$
then
$l\_age(n \mapsto l) := x$
$m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
end

change_link_dn
status
convergent
any
n
l
x
where
$x = age(l)$
$n \mapsto l \mapsto x \in m\_net$
$parity(x) = 0$
then
$l\_age(n \mapsto l) := x$
$m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
end

Likewise, we notice that the actions of these events are identical and the same as the actions of the next event. Hence we shall merge the three of them in the next refinement. The next event gives an explanation to abstract event change_link_2: it corresponds to old messages being stored locally, although they are not up-to-date but still with their "age" greater than the local ages (node n cannot know that they are not the most recent ones).

change_link_2
  status
    convergent
  any
    n
    l
    x
  where
    $x \neq age(l)$
    $n \mapsto l \mapsto x \in m\_net$
    $x > l\_age(n \mapsto l)$
  with
    $(parity(x) = 0 \;\Rightarrow\; ln = l\_net \setminus \{n \mapsto l\}) \wedge$
    $(parity(x) = 1 \;\Rightarrow\; ln = l\_net \cup \{n \mapsto l\})$
  then
    $l\_age(n \mapsto l) := x$
    $m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
  end

Notice the implicative definition of the witness for abstract parameter ln: its definition depends on the $parity$ of parameter x.

Event discard_1 below is new. It corresponds to messages that should not be stored as they are definitely too old. This event is also convergent.

The last event discard_2, also new, corresponds to dummy messages already received but still sent and thus received again. Notice that this event is *not convergent*. This is the only event that is not convergent. As a result, our fundamental theorem (proved in refinement 5) concerning the eventual identity between the local networks and the real network will depend on the *fairness* of this event. In other words, it must allow other events to eventually "execute" and thus converge.

discard_1
  status
    convergent
  any
    n
    l
    x
  where
    $n \mapsto l \mapsto x \in m\_net$
    $x \leq l\_age(n \mapsto l)$
  then
    $m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
  end

discard_2
  any
    n
    l
    x
  where
    $n \mapsto l \mapsto x \notin m\_net$
    $x \leq l\_age(n \mapsto l)$
  then
    $m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
  end

These two events have identical actions. Thus they will be merged in the next refinement.

Events which are said to be convergent in this refinement use the following obvious variant following our last invariant, which says that variable m_net is finite.

inv3_13: $\text{finite}(m\_net)$

variant3: m_net

4.5 Fourth Refinement

In this refinement, we just merge events, as announced in the previous section:

init
$NET := \varnothing$
$age := L \times \{0\}$
$l\_age := N \times L \times \{0\}$
$m\_net := \varnothing$

discover
refines
discover_up
discover_dn
any
l
n
where
$l \in NET \Leftrightarrow parity(age(l)) = 0$
$n \in N$
then
$age(l) := age(l) + 1$
$m\_net := m\_net \cup ((N \setminus \{n\}) \times \{l\} \times \{age(l) + 1\})$
$l\_age(n \mapsto l) := age(l) + 1$
end

change_link
refines
change_link_up
change_link_dn
change_link_2
any
n
l
x
where
$n \mapsto l \mapsto x \in m\_net$
$x > l\_age(n \mapsto l)$
then
$l\_age(n \mapsto l) := x$
$m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
end

discard
discard_1
discard_2
any
n
l
x
where
$x \leq l\_age(n \mapsto l)$
then
$m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
end

Variable m_net still occurs in the guard of the event change_link. In the next refinement however, we will eliminate the variable m_net from this guard and thus we will be able to remove the variable m_net.

4.6 Fifth Refinement

In this refinement, we finally establish the connection between the set L of links and the set N of nodes. For this, we define two constant functions fst and snd projecting a link on its first and second node respectively. We also define a constant bijection $link$ between pairs of nodes and links. Axioms **axm5_4**, **axm5_5**, and **axm5_6** state the obvious properties relating these three constants.

constants:	fst snd $link$

axm5_1: $fst \in L \twoheadrightarrow N$

axm5_2: $snd \in L \twoheadrightarrow N$

axm5_3: $link \in N \times N \rightarrowtail\!\!\!\!\twoheadrightarrow L$

axm5_4: $\forall n, m \cdot fst(link(n \mapsto m)) = n$

axm5_5: $\forall n, m \cdot snd(link(n \mapsto m)) = m$

axm5_6: $\forall l \cdot link(fst(l) \mapsto snd(l)) = l$

Notice that when a link l belongs to the physical network (that is, $l \in NET$) then node $fst(l)$ is a neighbor of node $snd(l)$. This takes account of requirement REQ-4:

The neighbors of a node b are the nodes that are connected to b by a link arriving in b. Neighbors of b send to it their local networks	REQ-4

Now we are able to remove the message container m_net. This is due to the fact that a link between two neighbors in the graph can play its role. More precisely, from invariant **inv3_12** established in the third refinement, which we copy below,

inv3_12: $\forall n, l \cdot \; age(n \mapsto l) < age(l) \;\Rightarrow\; l\_age(n \mapsto l) + 1 \,..\, age(l) \in m\_net[\{n \mapsto l\}]$

we can deduce the following theorem (with the help of invariant **inv3_4**):

thm5_1: $\forall n, l \cdot \; l\_age(n \mapsto l) < l\_age(m \mapsto l) \;\Rightarrow\; n \mapsto l \mapsto l\_age(m \mapsto l) \in m\_net$

This theorem says that when the local age of link l at node n is strictly less than that at node m then the local age at node m is in a message for node n. In other words, the difference in local ages induces a presence in m_net. This will help us remove m_net from the guard of abstract event change_link, which we copy below:

change_link
any
n
l
x
where
$n \mapsto l \mapsto x \in m\_net$
$x > l\_age(n \mapsto l)$
then
$l\_age(n \mapsto l) := x$
$m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
end

In fact, we replace now abstract parameter n by node $snd(k)$, where k is a link arriving in n (thus, as said above, $fst(k)$ is a neighbor of $snd(k)$). Moreover, we suppose that parameter x is less than or equal to the local age at node $fst(k)$. This additional guard might seem strange: why have not we chosen $fst(k)$ for x? The answer will be given in refinement 7. From all this, we obtain the following concrete version of the event change_link:

```
change_link
  any
    l
    x
    k
  where
    k ∈ NET
    x > l_age(snd(k) ↦ l)
    x ≤ l_age(fst(k) ↦ l)
  with
    n = snd(k)
  then
    l_age(snd(k) ↦ l) := x
    m_net := m_net \ {snd(k) ↦ l ↦ x}
  end
```

But now we see that variable m_net does not occur any more in any guard. As a consequence, we can remove this variable which does not influence the behavior of our events. Next are the updates of the events. In the event discover we make now the last touch, by replacing the random choice of parameter n by the precise node $snd(l)$. In doing this, we take full account of requirement REQ-3.

When a link from node a to node b is set into or removed from the graph then node b is directly made aware of it	REQ-3

```
init
  NET := ∅
  age := L × {0}
  l_age := N × L × {0}
```

```
discover
  any
    l
  where
    l ∈ NET ⇔ parity(age(l)) = 0
  with
    n = snd(l)
  then
    age(l) := age(l) + 1
    l_age(snd(l) ↦ l) := age(l) + 1
  end
```

Notice that in the event discard, we perform a similar modification as the one we have done in the event change_link, namely replacing parameter n by node $snd(k)$, where k is a link belonging to NET.

change_link
any
l
x
k
where
$k \in NET$
$x > l\_age(snd(k) \mapsto l)$
$x \leq l\_age(fst(k) \mapsto l)$
with
$n = snd(k)$
then
$l\_age(snd(k) \mapsto l) := x$
end

discard
any
l
x
k
where
$k \in NET$
$x \leq l\_age(snd(k) \mapsto l)$
with
$n = snd(k)$
then
skip
end

4.7 Sixth Refinement

In this refinement, we take account of requirement REQ-7:

We want to prove that under certain conditions the images of the graph built by nodes are all identical and equal to the graph itself	REQ-7

In particular, we replace the variable NET by the variable G representing the physical graph. Here is the definition of this new variable and its connection to abstract variable NET:

variables: G

inv6_1: $G \in N \leftrightarrow N$

inv6_2: $\forall l \cdot l \in NET \Leftrightarrow fst(l) \mapsto snd(l) \in G$

inv6_3: $\forall l \cdot l\_age(snd(l) \mapsto l) = age(l)$

Invariant **inv6_3** will allow us to remove the variable age in the next refinement. We can now state the main result. Our two convergent events are discover and change_link. If these events cannot be "executed" any more (deadlock) and if the physical network is strongly connected at that time, then all local networks are identical to the physical network. Here are the new versions of these events:

discover
any
l
where
$fst(l) \mapsto snd(l) \in G \;\Leftrightarrow\; parity(age(l)) = 0$
then
$age(l) := age(l) + 1$
$l\_age(snd(l) \mapsto l) := age(l) + 1$
end

change_link
any
l
x
k
where
$fst(k) \mapsto snd(k) \in G$
$x > l\_age(snd(k) \mapsto l)$
$x \leq l\_age(fst(k) \mapsto l)$
with
$n = snd(k)$
then
$l\_age(snd(k) \mapsto l) := x$
end

Since we assume that they cannot be "executed" then their guards are false. Here are these guards:

$$\exists\, l \cdot fst(l) \mapsto snd(l) \in G \;\Leftrightarrow\; parity(age(l)) = 0$$

$$\begin{array}{l} \exists\, l, x, k \cdot fst(k) \mapsto snd(k) \in G \;\wedge \\ \qquad x > l\_age(snd(k) \mapsto l) \;\wedge \\ \qquad x \leq l\_age(fst(k) \mapsto l) \end{array}$$

The negation of the first one can be simplified as follows:

$$\begin{array}{l} \neg\, \exists\, l \cdot fst(l) \mapsto snd(l) \in G \;\Leftrightarrow\; parity(age(l)) = 0 \\ \Leftrightarrow \\ \quad \forall\, l \cdot fst(l) \mapsto snd(l) \in G \;\Leftrightarrow\; parity(age(l)) = 1 \\ \Leftrightarrow \\ \quad \forall\, l \cdot l \in NET \;\Leftrightarrow\; parity(age(l)) = 1 \end{array}$$

The negation of the second one can be simplified as follows:

$$\begin{array}{l} \neg\, \exists\, l, x, k \cdot fst(k) \mapsto snd(k) \in G \;\wedge \\ \qquad\qquad x > l\_age(snd(k) \mapsto l) \;\wedge \\ \qquad\qquad x \leq l\_age(fst(k) \mapsto l) \\ \Leftrightarrow \\ \quad \forall\, l, x, k \cdot \; fst(k) \mapsto snd(k) \in G \\ \qquad\qquad \Rightarrow \\ \qquad\qquad x \leq l\_age(snd(k) \mapsto l) \;\;\vee\;\; x > l\_age(fst(k) \mapsto l) \\ \Leftrightarrow \\ \quad \forall\, l, x, a, b \cdot \; a \mapsto b \in G \;\Rightarrow\; x \leq l\_age(b \mapsto l) \;\;\vee\;\; x > l\_age(a \mapsto l) \\ \Leftrightarrow \\ \quad \forall\, l, a, b \cdot \; a \mapsto b \in G \;\Rightarrow\; l\_age(a \mapsto l) \leq l\_age(b \mapsto l) \end{array}$$

The strong connectivity of the graph G is expressed by means of the following statement (justified in the Appendix):

$$\forall s \cdot s \neq \varnothing \;\wedge\; G[s] \subseteq s \;\Rightarrow\; N \subseteq s$$

We first prove the following theorem

$$
\begin{array}{ll}
\textbf{thm6\_1:} & \forall\, l,a,b \cdot a \mapsto b \in G \;\Rightarrow\; l\_age(a \mapsto l) \leq l\_age(b \mapsto l) \\
& \forall s \cdot s \neq \varnothing \;\wedge\; G[s] \subseteq s \;\Rightarrow\; N \subseteq s \\
& \Rightarrow \\
& \forall\, l, v \cdot l\_age(n \mapsto l) = age(l)
\end{array}
$$

To prove this theorem, we instantiate the quantified variable s in the second hypothesis with the set $\{n \mid l\_age(n \mapsto l)\}$. Proving that this set is not empty is easy: $snd(l)$ belongs to it according to invariant **inv6_3**. $G[s] \subseteq s$ is a consequence of the first hypothesis. The result then follows easily. The following theorem can be deduced from the previous one:

$$
\begin{array}{ll}
\textbf{thm6\_2:} & \forall l \cdot l \in NET \Leftrightarrow parity(age(l)) = 1 \\
& \forall\, l,a,b \cdot a \mapsto b \in G \;\Rightarrow\; l\_age(a \mapsto l) \leq l\_age(b \mapsto l) \\
& \forall s \cdot s \neq \varnothing \;\wedge\; G[s] \subseteq s \;\Rightarrow\; N \subseteq s \\
& \Rightarrow \\
& \forall l, n \cdot l \in NET \Leftrightarrow parity(l\_age(n \mapsto l)) = 1
\end{array}
$$

Here is our final result (with the help of invariant **inv3_5**).

$$
\begin{array}{ll}
\textbf{thm6\_3:} & \forall l \cdot l \in NET \Leftrightarrow parity(age(l)) = 1 \\
& \forall\, l,a,b \cdot a \mapsto b \in G \;\Rightarrow\; l\_age(a \mapsto l) \leq l\_age(b \mapsto l) \\
& \forall s \cdot s \neq \varnothing \;\wedge\; G[s] \subseteq s \;\Rightarrow\; N \subseteq s \\
& \Rightarrow \\
& \forall n \cdot l\_net[\{n\}] = NET
\end{array}
$$

This theorem can be restated in a less formal way as follows:

Event discover deadlocks and
Event change_link deadlocks and
Physical graph is strongly connected
$\Rightarrow$
Local networks are all equal to physical network

Next are the updates of the three remaining non-convergent events:

Modify_up	Modify_dn
any l **where** $fst(l) \mapsto snd(l) \notin G$ **then** $G := G \cup \{fst(l) \mapsto snd(l)\}$ **end**	**any** l **where** $fst(l) \mapsto snd(l) \in G$ **then** $G := G \setminus \{fst(l) \mapsto snd(l)\}$ **end**

init
$G := \varnothing$
$age := L \times \{0\}$
$l\_age := N \times L \times \{0\}$

discard
any
l
x
k
where
$fst(k) \mapsto snd(k) \in G$
$x \leq l\_age(snd(k) \mapsto l)$
then
skip
end

4.8 Seventh Refinement

In this last refinement, we take account of requirements REQ-5 and REQ-6:

Nodes communicates by means of messages sent on links of the graph	REQ-5

A message sent from a to b can be lost if the link from a to b, which existed when the message was sent, is broken before the message reaches b	REQ-6

To this end, we have an additional variable m defined as follows (notice that variable age has been removed):

variables: G
l_age
m

inv7_1: $m \subseteq L \times L \times N$

inv7_2: $\forall k, l, x \cdot k \mapsto l \mapsto x \in m \;\Rightarrow\; x \leq l\_age(fst(k) \mapsto l)$

inv7_3: $\forall k, l, x \cdot k \mapsto l \mapsto x \in m \;\Rightarrow\; fst(k) \mapsto snd(k) \in G$

Invariant **inv7_2** states that an age x recorded in a message is always less than or equal to the local age at the sending point of the message. This is because the local age at the sending point can only be incremented after the message has been sent. This allows us to remove a guard from the event change_link, renamed below accept_message.

Invariant **inv7_3** states that a message arriving at its destination $snd(k)$ is always such that the link from $fst(k)$ to $snd(k)$ is in the graph. This allows us to remove guards from the events accept_message and discard, renamed below to discard_message.

Here are all our final concrete events:

Modify_up
any
l
where
$fst(l) \mapsto snd(l) \notin G$
then
$G := G \cup \{fst(l) \mapsto snd(l)\}$
end

Modify_dn
any
l
where
$fst(l) \mapsto snd(l) \in G$
then
$G := G \setminus \{fst(l) \mapsto snd(l)\}$
$m := m \setminus (\{l\} \times L \times \mathbb{N})$
end

In the event Modify_dn, messages sent along the removed link l are lost as stipulated by requirement REQ-6:

A message sent from a to b is lost if the link from a to b, which existed when the message was sent, is broken before the message reaches b	REQ-6

init
$G := \varnothing$
$m := \varnothing$
$l\_age := N \times L \times \{0\}$

discover
any
l
where
$fst(l) \mapsto snd(l) \in G \;\Leftrightarrow\; parity(l\_age(snd(l) \mapsto l)) = 0$
then
$l\_age(snd(l) \mapsto l) := l\_age(snd(l) \mapsto l) + 1$
end

The next event is new. It corresponds to sending a message. This can be done if a link indeed exists between the corresponding nodes as mentioned in requirement REQ-5. Note that this event is *not convergent*: a message can always be sent.

send_message
any
k
l
where
$fst(k) \mapsto snd(k) \in G$
$l \in L$
then
$m := m \cup \{k \mapsto l \mapsto l\_age(fst(k) \mapsto l)\}$
end

The last events are renaming of abstract events change_link and discard:

accept_message
refines
change_link
any
l
x
k
where
$k \mapsto l \mapsto x \in m$
$x > l\_age(snd(k) \mapsto l)$
then
$l\_age(snd(k) \mapsto l) := x$
$m := m \setminus \{k \mapsto l \mapsto x\}$
end

discard_message
refines
discard
any
l
x
k
where
$k \mapsto l \mapsto x \in m$
$x \leq l\_age(snd(k) \mapsto l)$
then
$m := m \setminus \{k \mapsto l \mapsto x\}$
end

5 Concluding Remarks

The formal development was conducted using the Rodin Platform. Here is a summary of the proof statistics together with the corresponding percentage of automatic proofs:

model	proofs	auto.	manual	% auto.
Ref. 1	3	3	0	100
Ref. 2	47	47	0	100
Ref. 3	130	93	37	72
Ref. 4	6	4	2	67
Ref. 5	18	16	2	89
Ref. 6	13	9	4	69
Ref. 7	38	25	13	66
Total	255	197	58	77

As can be seen, Refinement 3 required by far the largest number of proofs: it is due to the large number of invariants (13), together with the number of events (10). It should be noted that the manual proofs were not difficult. It seems that many of them could be done automatically, which suggests that the automatic provers have to be improved!

APPENDIX: Strongly Connected Graph

Given a set V of vertices and a binary relation r from V to itself, the graph represented by this relation is said to be *strongly connected* if any two nodes in V are connected by a path built on r. This is illustrated in figure 4.

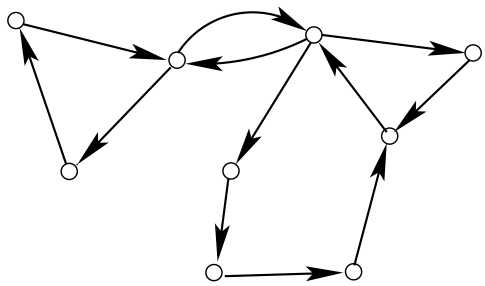

Fig. 4. A Strongly Connected Graph

This can be formalized as follows, where r^* is the reflexive transitive closure of r:

$$\boxed{\mathbf{def\_1}: \quad r^* = V \times V}$$

This definition is easy to understand: it simply says that every two points of V are related through the reflexive transitive closure r^* of r. But this definition is not very convenient to use in practice. Here is an *equivalent one* which is more convenient:

$$\boxed{\mathbf{def\_2}: \quad \forall s \cdot s \neq \varnothing \;\wedge\; r[s] \subseteq s \;\Rightarrow\; V \subseteq s}$$

This definition is convenient because it gives us an *induction rule* to prove properties of the vertices of a strongly connected graph, which is the one we used in section 4.7. In order to prove the equivalence between these definitions, we need to have a formal representation of r^*:

$$\boxed{\begin{array}{ll}
\mathbf{axm_1}: & r^* \in V \leftrightarrow V \\[1ex]
\mathbf{axm_2}: & \mathrm{id} \subseteq r^* \\[1ex]
\mathbf{axm_3}: & r^* \,;\, r \subseteq r^* \\[1ex]
\mathbf{axm_4}: & \forall p \cdot \mathrm{id} \subseteq p \;\wedge\; p \,;\, r \subseteq p \;\Rightarrow\; r^* \subseteq p
\end{array}}$$

Proof of:

$$\begin{array}{l} \forall s \cdot s \neq \varnothing \ \wedge \ r[s] \subseteq s \ \Rightarrow \ V \subseteq s \\ \Rightarrow \\ r^* = V \times V \end{array}$$

We prove the equivalent lemma

$$\begin{array}{l} \forall s \cdot s \neq \varnothing \ \wedge \ r[s] \subseteq s \ \Rightarrow \ V \subseteq s \\ \Rightarrow \\ \forall x \cdot V \subseteq r^*[\{x\}] \end{array}$$

To prove this, we instantiate s in the hypothesis with $r^*[\{x\}]$. The result follows easily according to **axm_2** and **axm_3**.

Proof of:

$$\begin{array}{l} r^* = V \times V \\ \Rightarrow \\ \forall s \cdot s \neq \varnothing \ \wedge \ r[s] \subseteq s \ \Rightarrow \ V \subseteq s \end{array}$$

First we replace r^* by $V \times V$ in **axm_2** to **axm_4**. Then we simplify the goal. Finally, we instantiate the quantified variable p of **axm_4** with $(s \times s) \cup ((N \setminus s) \times N)$. The result follows easily.