Routing Protocol Model

J-R. Abrial

September 2004
Purpose of this Lecture

- No more learning about refinement and abstraction (practicing)

- No more learning about modeling conventions (practicing)

- Re-using dynamically the small tree theory we already developed

- Study a practical problem in distributed computing communication

- The example comes from the following paper:

The Abstract Communication Situation with a Mobile Agent

- A mobile agent $\mathcal{M}$ is supposed to travel between sites

- Some fixed agents at sites want to send messages to $\mathcal{M}$

- In an abstract world:
  - the moves of $\mathcal{M}$ are instantaneous
  - the traveling of messages between sites takes no time
  - the knowledge of the moves of $\mathcal{M}$ is also instantaneous

- Thus fixed agents always send messages where $\mathcal{M}$ is
Initial Situation

\begin{itemize}
  \item a \rightarrow b
  \item d \rightarrow c
\end{itemize}
M moves from c to d
\( \mathcal{M} \) moves from \( d \) to \( a \)
$\mathcal{M}$ moves from a to c
M moves from c to b
A More Concrete Situation

- The moves of $\mathcal{M}$ are still instantaneous

- The traveling of messages between sites still takes no time

- The knowledge of the moves of $\mathcal{M}$ is not instantaneous any more
- When $\mathcal{M}$ moves from site $x$ to site $y$ then
  - Agents of $x$ and $y$ knows it immediately
  - Agents of other sites are not aware of the move
  - They still sent their messages where they believe $\mathcal{M}$ is

- A message arriving at a site which $\mathcal{M}$ has left can be forwarded
Initial Situation
M moves from c to d
$M$ moves from $d$ to $a$
$M$ moves from $a$ to $c$
\( M \) moves from \( c \) to \( b \)
Showing the Structural Modifications

Diagram showing structural modifications with nodes labeled a, b, c, and d, and arrows indicating changes in connections.
- The mobile $\mathcal{M}$ is at the root of a tree
Comparing the two Situations

The mobile $\mathcal{M}$ remains at the root of a tree (to be proved however)
A Dynamic Tree Network

Diagram of a network showing nodes and connections.
Formal Description. The State

prp_1: \( il \in S \)

inv_1: \( l \in S \)

inv_2: \( c \in S \rightarrow S \)

inv_3: \( p \in M \rightarrow S \)

inv_4: \( \forall T \cdot \left( \begin{array}{l} T \subseteq S \\ l \in T \\ \{l\} \triangleleft c)^{-1}[T] \subseteq T \\ \Rightarrow \\ S \subseteq T \end{array} \right) \)

Sets:
- \( S \) set of sites
- \( M \) set of com. msgs.

Constants:
- \( il \) initial location of \( M \)

Variables:
- \( l \) location of \( M \)
- \( c \) dynamic channels
- \( p \) sites of com. msgs.
Formal Description. The Events

\[
\text{init} \equiv \begin{align*}
\text{begin} \\
l &:= il \\
c &:= S \times \{il\} \\
p &:= \emptyset \\
\text{end}
\end{align*}
\]

\[
\text{rcv\_agt} \equiv \begin{align*}
\text{any} & \ s \text{ where} \\
& s \in S \setminus \{l\} \\
\text{then} \\
l &:= s \\
c(l) &:= s \\
\text{end}
\end{align*}
\]

Exercise:
Define events for sending, forwarding, and delivering messages
Proving the preservation of inv_4 by Event rcv_agt

Assumption

\[ \forall T \cdot \begin{cases} T \subseteq S \\ l \in T \\ (\{l\} \perp c)^{-1}[T] \subseteq T \\ \Rightarrow S \subseteq T \end{cases} \]

rcv_agt \ \ \ \ \equiv

any \ s \ where \ s \in S \setminus \{l\}

then

l := s

c := \{l\} \perp c \cup \{l \mapsto s\}

end

To be proved

(since \ s \neq l)

\[ \forall T \cdot \begin{cases} T \subseteq S \\ s \in T \\ (\{s, l\} \perp c \cup \{l \mapsto s\})^{-1}[T] \subseteq T \\ \Rightarrow S \subseteq T \end{cases} \]
Proving the preservation of inv_4 by Event rcv_agt (cont’d)

- It is sufficient to prove:

\[ T \subseteq S \]
\[ s \in T \]
\[ \left( \{s, l\} \equiv c \cup \{l \mapsto s\} \right)^{-1}[T] \subseteq T \]
\[ \Rightarrow \]
\[ T \subseteq S \]
\[ l \in T \]
\[ \left( \{l\} \equiv c \right)^{-1}[T] \subseteq T \]

But we have (since \( s \in T \)):

\[ \left( \{s, l\} \equiv c \cup \{l \mapsto s\} \right)^{-1}[T] = \left( \left( \{l\} \equiv c \right)^{-1}[T] \setminus \{s\} \right) \cup \{l\} \]

Proof by cases:  
\( s \notin \left( \{l\} \equiv c \right)^{-1}[T] \)  or  \( s \in \left( \{l\} \equiv c \right)^{-1}[T] \)
An Even More Concrete Situation

- The moves of $M$ are not completely instantaneous any more

- The traveling of messages between sites still takes no time

- The knowledge of the moves of $M$ is not instantaneous any more
When $\mathcal{M}$ Departs from Site $x$

- Agents of $x$ do not know where $\mathcal{M}$ is going

- Agents of other sites are not aware of the move

- Messages at $x$ cannot be forwarded until $x$ knows where $\mathcal{M}$ is

- Messages at other sites can be forwarded (in general)
When $M$ Arrives at its destination $y$ (coming from $x$)

- It sends a “acknowledgment message” to site $x$ to inform of its position.

- Once $x$ has received the “acknowledgment message” it can forward again communication messages which were pending.

- From now on, we have to distinguish:
  - communication messages (still instantaneous)
  - acknowledgment messages (which take some time)
Initial Situation
\( M \) moves from \( c \) to \( d \)

\( M \) sends an acknowledgment message to \( c \): "I am now in \( d \)"

Site \( c \) suspend sending com. msg. until it knows where \( M \) is
\( M \) moves from \( d \) to \( a \)

\( M \) sends an acknowledgment message to \( d \): "I am now in \( a \)"

Site \( d \) suspend sending com. msg. until it knows where \( M \) is
\( M \) moves from \( a \) to \( c \)

\( M \) sends an acknowledgment message to \( a \): "I am now in \( c \)"

Site \( a \) suspend sending com. msg. until it knows where \( M \) is
\( \mathcal{M} \) moves from \( c \) to \( b \)

\( \mathcal{M} \) sends an acknowledgment message to \( c \): "I am now in \( b \)"

Site \( c \) suspend sending com. msg. until it knows where \( \mathcal{M} \) is
No Acknowledgment Message has Arrived yet

\[ \text{Diagram with nodes } a, b, c, \text{ and connections.} \]
Acknowledgment Message from $a$ to $d$ Arrives

Site $d$ believes $M$ is in $a$. It now forwards pending com. msg. to $a$. 
Acknowledgment Message from c to a Arrives

Site a believes M is in c. It now forwards pending com. msg. to c
Site $c$ believes $\mathcal{M}$ is in $b$. It now forwards pending com. msg. to $b$. 
Acknowledgment Message from $d$ to $c$ Arrives. FAILURE

Site $c$ believes $M$ is in $d$. It now forwards pending com. msg. to $d$

The tree structure is destroyed: we have a CYCLE.
Analysis of Failure and “magic” Solution

- The failure comes from the two ack. msg. arriving in the same place

- We must preclude this to happen

- We shall suppose that we have the following “magic” behavior
  - When $M$ sends an acknowledgment msg. to site $x$
  - It is able to remove pending ack. msgs. arriving to $x$
Initial Situation
\( \mathcal{M} \) moves from \( c \) to \( d \)

\( \mathcal{M} \) sends an acknowledgment message to \( c \): "I am now in \( d \"")

Site \( c \) suspend sending com. msg. until it knows where \( \mathcal{M} \) is
\(\mathcal{M}\) moves from d to a

\(\mathcal{M}\) sends an acknowledgment message to \(d\): "I am now in a"

Site \(d\) suspend sending com. msg. until it knows where \(\mathcal{M}\) is
$M$ moves from $a$ to $c$

$M$ sends an acknowledgment message to $a$: "I am now in $c"

Site $a$ suspend sending com. msg. until it knows where $M$ is
$M$ moves from $c$ to $b$

$M$ sends an acknowledgment message to $c$: "I am now in $b"

$M$ "magically" removes the other ack. message arriving to $c$
First Refinement. The State

\[
\begin{align*}
\text{inv}_5 : & \quad d \in S \rightarrow S \\
\text{inv}_6 : & \quad a \in S \rightarrow S \\
\text{inv}_7 : & \quad c = d \leftrightarrow a
\end{align*}
\]

d \quad \text{new channel structure}

\[a\quad \text{acknowledgment messages}\]

- Invariant \text{inv}_7 is the gluing invariant with the abstraction

- Caution: An ack. msg. from \(y\) to \(x\) is in \(a\) as: \(\{x \mapsto y\}\)

- It corresponds to \(M\) moving from \(x\) to \(y\)
First Refinement. The Events

\[
\begin{align*}
\text{rcv_agt} & \equiv \\
\text{any } s \text{ where } s \in S \setminus \{l\} \text{ then } \\
l & := s \\
a(l) & := s \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{rcv_ack} & \equiv \\
\text{any } s \text{ where } s \in \text{dom}(a) \text{ then } \\
d(s) & := a(s) \\
a & := \{s\} \triangleleft a \\
\text{end}
\end{align*}
\]

- Observe that the pending ack. msg. to \(l\) (if any) is removed since

\[
a(l) \ := \ s \quad \text{is the same as} \quad a \ := \ \{l\} \triangleleft a \cup \{l \mapsto s\}
\]

- Observe that the acknowledgment channel is cleaned \((a \ := \ \{s\} \triangleleft a)\)
Comparing with Previous Version

(abs\_\_rcv\_agt) \equiv \forall s \, \text{where} \ s \in S \setminus \{l\} then 
\begin{align*}
l &:= s \\
c(l) &:= s
\end{align*}
end

(ref\_\_rcv\_agt) \equiv \forall s \, \text{where} \ s \in S \setminus \{l\} then 
\begin{align*}
l &:= s \\
a(l) &:= s
\end{align*}
end

(rcv\_\_ack) \equiv \forall s \, \text{where} \ s \in \text{dom}(a) then 
\begin{align*}
d(s) &:= a(s) \\
a &:= \{s\} \triangleleft a
\end{align*}
end
Second Refinement: Implementing the “magic” ack. channel

- Magic behavior when sending a new ack. msg. to $x$:
  - Pending ack. msg. to $x$ are removed

- The mobile $M$ travels with a logical clock

- Each site has a mobility counter

- The mobility counter records the “time” of the last visit of $M$
- When $M$ arrives at a site $y$
  - it increments its logical clock
  - it stores its incremented clock in the mobility counter of $y$
  - it sends a new ack. msg. to its previous location $x$

- The ack. msg. from $y$ to $x$ is stamped with the new clock value

- When an ack. msg. arrives at a site $x$, it is accepted
  - only if its stamp value is greater than the mobility counter of $x$
  - the mobility counter takes the value of the stamp
Initial Situation
\( M \) moves from \( c \) to \( d \)
$\mathcal{M}$ moves from $d$ to $a$
$M$ moves from $a$ to $c$
$M$ moves from c to b
No Acknowledgment Message has yet Arrived
Acknowledgment Message from $\alpha$ to $d$ Arrives

- It is accepted
Acknowledgment Message from c to a Arrives

- It is accepted
Acknowledgment Message from $b$ to $c$ Arrives

- It is accepted
Acknowledgment Message from $d$ to $c$ Arrives. NO FAILURE

- It is rejected
- Suppose:
  - \( s_1 \) has emitted an ack. msg. to \( s \) at time 3
  - \( s_2 \) has emitted an ack. msg. to \( s \) at time 5
  - \( s_3 \) has emitted an ack. msg. to \( s \) at time 9

- This will be “recorded” in the refined ack. channel as follows:

\[
s \leftrightarrow \{3 \leftrightarrow s_1, 5 \leftrightarrow s_2, 9 \leftrightarrow s_3\}
\]

- In the abstract ack. channel we simply had: \( s \leftrightarrow s_3 \)
Second Refinement. The State

Variables: $l, d, p, b, m$

\begin{align*}
\text{inv}_8 & : \quad m \in S \rightarrow \mathbb{N} \\
\text{inv}_9 & : \quad b \in S \rightarrow (\mathbb{N} \rightarrow S) \\
\text{inv}_{10} & : \quad \forall s \cdot (s \in S \Rightarrow m(s) \leq m(l))
\end{align*}

- The mobility counter at $l$ is the maximum
(abs)rcv_agt  ≜
any  s  where
s ∈ S \ {l}
then
l := s
a(l) := s
end

(ref)rcv_agt  ≜
any  s  where
s ∈ S \ {l}
then
l := s
b(l)(m(l) + 1) := s
m(s) := m(l) + 1
end
Second Refinement. The Events

\[ \text{thm}_1 : \quad \forall s, n \cdot \begin{cases} s \in S \\ n \in \text{dom} \ (b(s)) \\ m(s) < n \Rightarrow \\ s \in \text{dom} \ (a) \\ a(s) = b(s)(n) \end{cases} \]

(abs_)rcv_ack \quad \equiv \quad \begin{cases} \text{any} \ s, n \ \text{where} \\ s \in \text{dom} \ (a) \\ \text{then} \\ d(s) := a(s) \\ a := \{s\} \lhd a \end{cases}

(ref_)rcv_ack \quad \equiv \quad \begin{cases} \text{any} \ s, n \ \text{where} \\ s \in S \\ n \in \text{dom} \ (b(s)) \\ m(s) < n \ \text{then} \\ d(s) := b(s)(n) \\ a(s) := b(s)(n) \end{cases}

end
Second Refinement. Gluing Invariant

\[\text{inv}_{11} : \forall s \cdot \left( \begin{array}{l} s \in S \\ \text{dom} \ (b(s)) \neq \emptyset \\ \Rightarrow \\ \text{max} \ (\text{dom} \ (b(s))) \leq m(l) \end{array} \right)\]

\[\text{inv}_{12} : \forall s \cdot \left( \begin{array}{l} s \in \text{dom} \ (a) \\ \text{dom} \ (b(s)) \neq \emptyset \\ \Rightarrow \\ a(s) = b(s)(\text{max} \ (\text{dom} \ (b(s)))) \end{array} \right)\]

\[\text{inv}_{13} : \forall s \cdot \left( \begin{array}{l} s \in S \\ \text{dom} \ (b(s)) \neq \emptyset \\ m(s) < \text{max} \ (\text{dom} \ (b(s))) \\ \Rightarrow \\ s \in \text{dom} \ (a) \end{array} \right)\]
Second Refinement. Main Invariant and Theorem

inv\_14 : \forall s, n \cdot \begin{cases} 
    s \in S \\
    n \in \text{dom}(b(s)) \\
    m(s) < n \\
    \Rightarrow \\
    n = \max(\text{dom}(b(s))) 
\end{cases}

thm\_1 : \forall s, n \cdot \begin{cases} 
    s \in S \\
    n \in \text{dom}(b(s)) \\
    m(s) < n \\
    \Rightarrow \\
    s \in \text{dom}(a) \\
    a(s) = b(s)(n) 
\end{cases}
More Refinements

- The move of the mobile $\mathcal{M}$ is not instantaneous any more

- The communication messages are not sent instantaneously
What we Have Learned in this Lecture

- No more mathematical conventions

- Re-using an already introduced small theory (trees)

- How to gradually introduce constraints in refinements (again)

- How things are “magically” possible in an abstraction
# Proof Summary for all examples

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>Ring</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Tree</td>
<td>52</td>
<td>19</td>
</tr>
<tr>
<td>Mobile</td>
<td>59</td>
<td>20</td>
</tr>
</tbody>
</table>

- Summary of the proofs with the tool (**total**, **interactive**)

- Tool URL: [www.B4free.com](http://www.B4free.com)
What we Have Learned in this Course

- By the end of the course you should be more comfortable with:
  
- Modeling (versus programming)
  
- Abstraction and Refinement
  
- Some mathematical techniques (for data structures)
  
- The idea of proving (what to prove)
  
- Is it the case?