12. Routing Algorithm for Mobile Agent

Jean-Raymond Abrial

2009

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:

L. Moreau. *Distributed Directory Service and Message Routing for Mobile Agent*. Science of Computer Programming 2001.

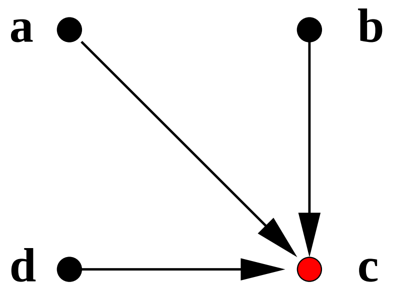
- A mobile agent $\mathcal{M}$ is supposed to travel between sites

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

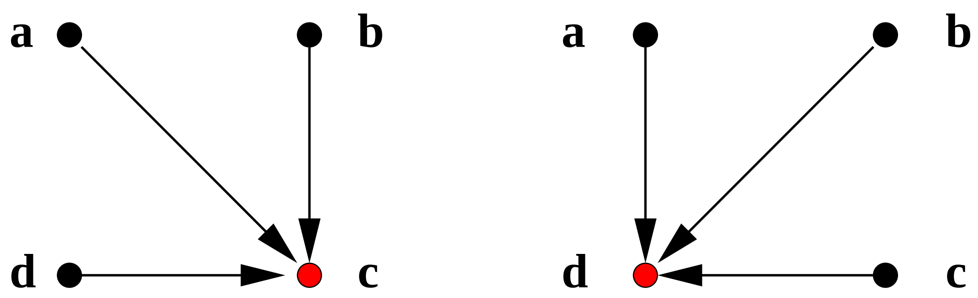
- In an abstract world:
    - the moves of $\mathcal{M}$ are instantaneous
    - the traveling of messages between sites takes no time
    - the knowledge of the moves of $\mathcal{M}$ is also instantaneous

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

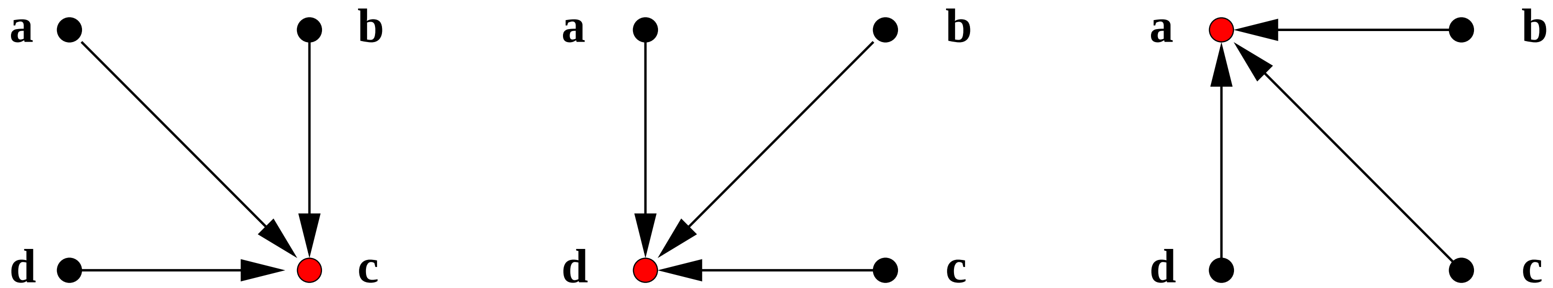
Initial Situation

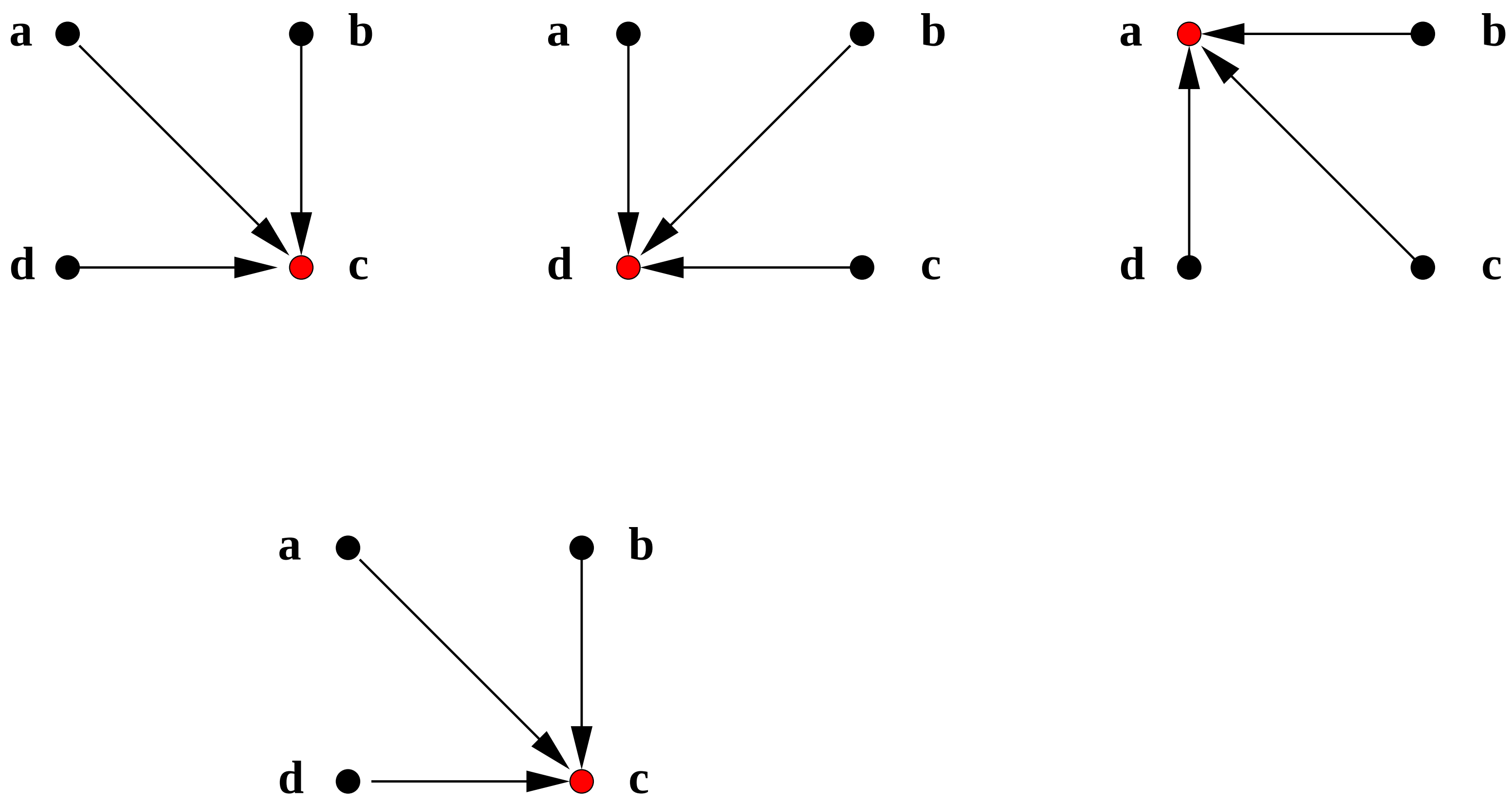


$\mathcal{M}$ moves from c to d

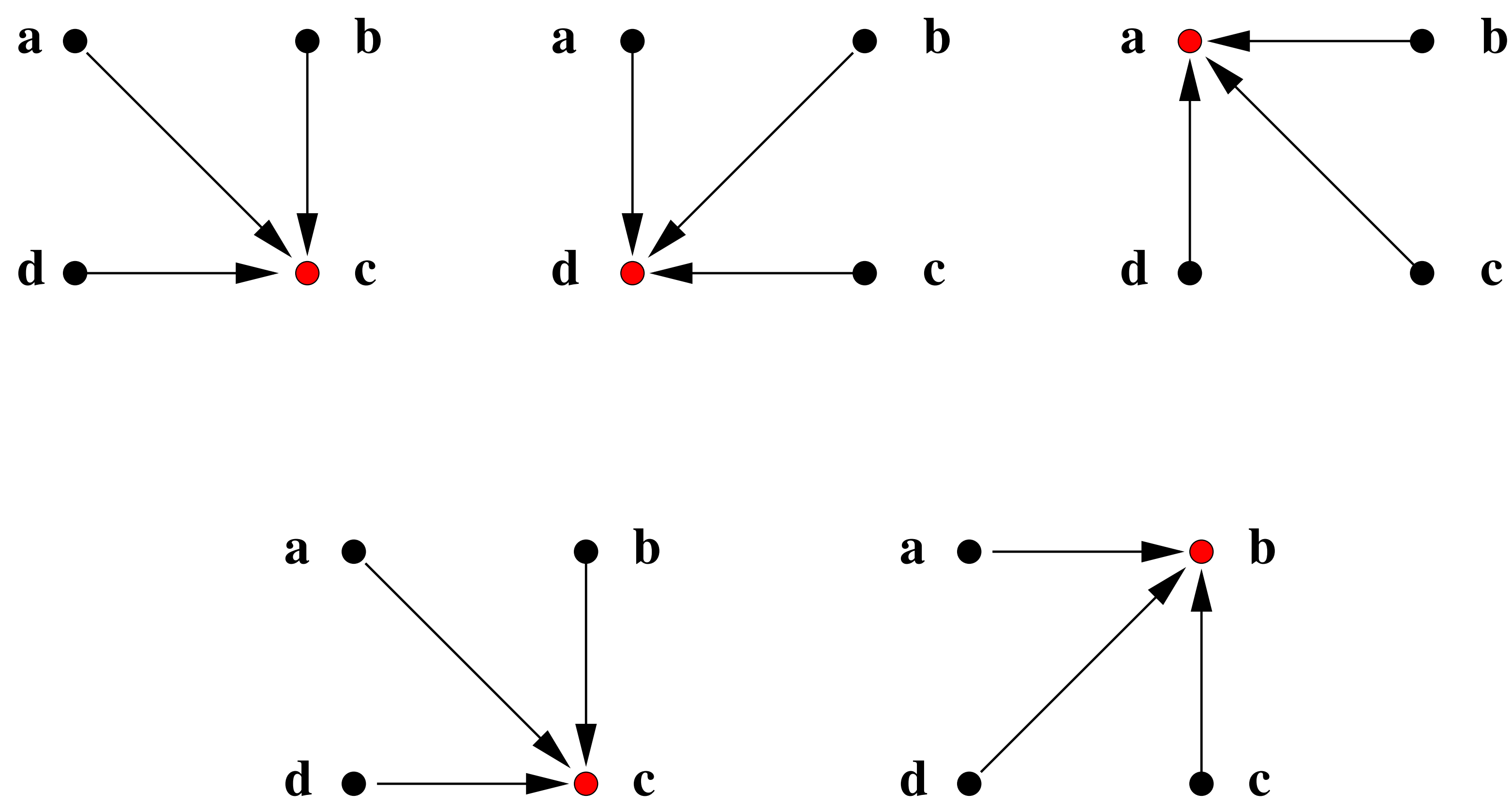


$\mathcal{M}$ moves from d to a




a
b
d
c
a
b
d
c
a
b
d
c
a
b
d
c

$\mathcal{M}$ moves from c to b



A More Concrete Situation

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

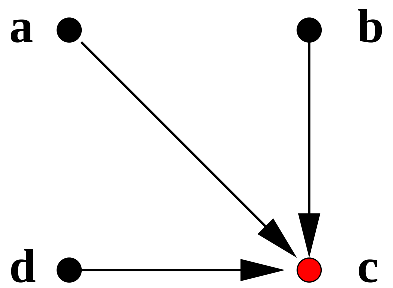
- The traveling of messages between sites still takes no time

- $\boxed{\text{The knowledge of the moves of } \mathcal{M} \text{ is not instantaneous any more}}$

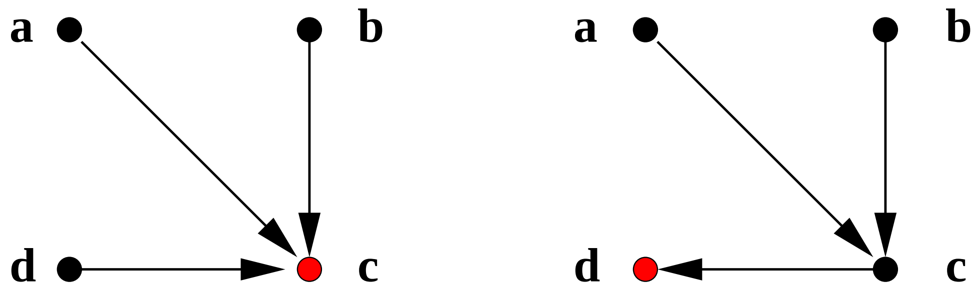
A More Concrete Situation (cont'd)

- 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

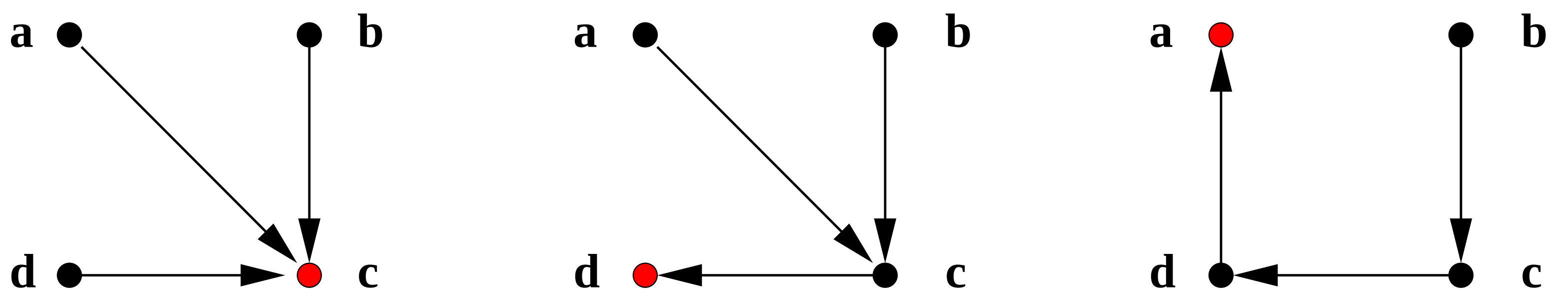
Initial Situation



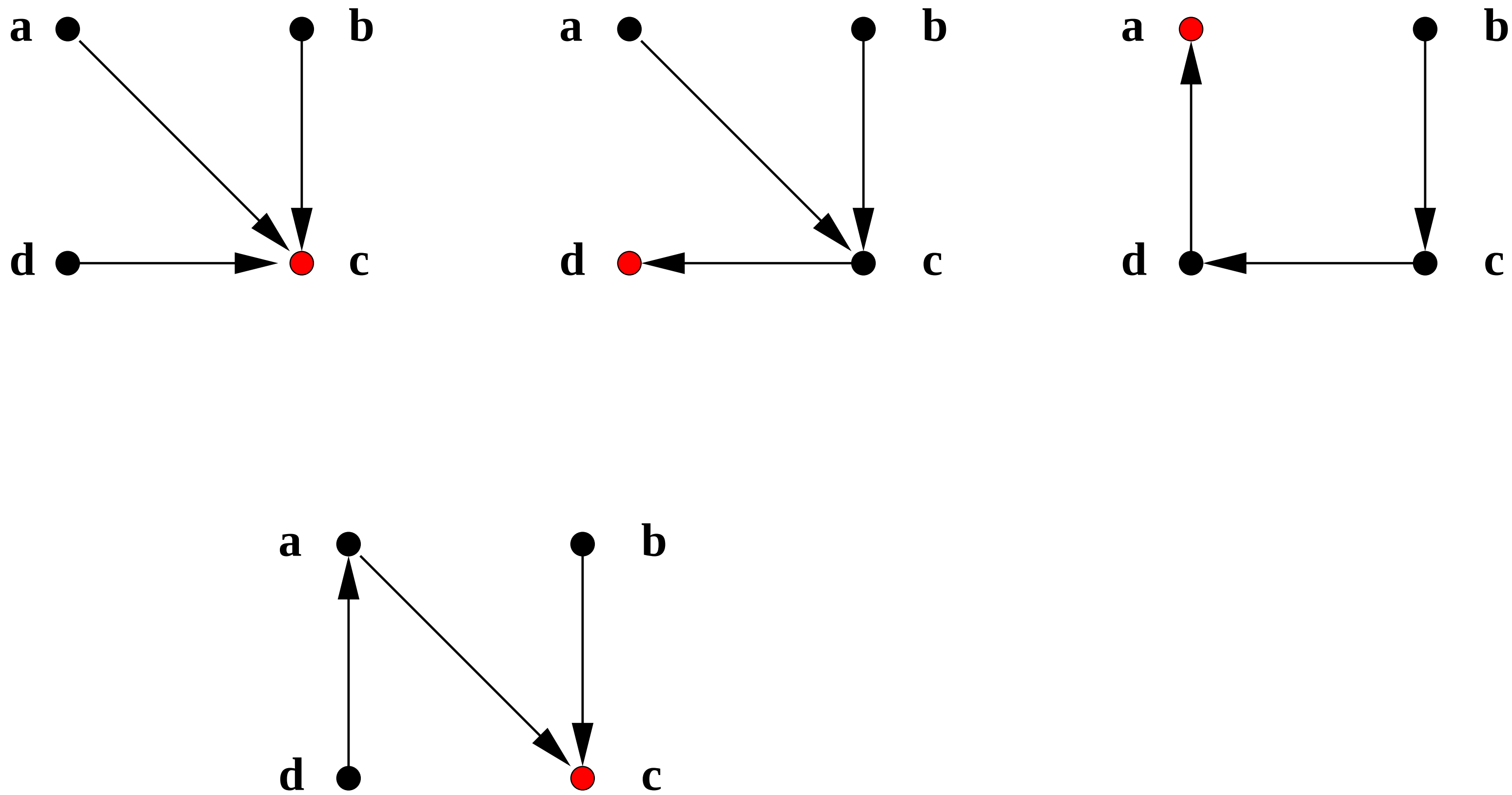
$\mathcal{M}$ moves from c to d



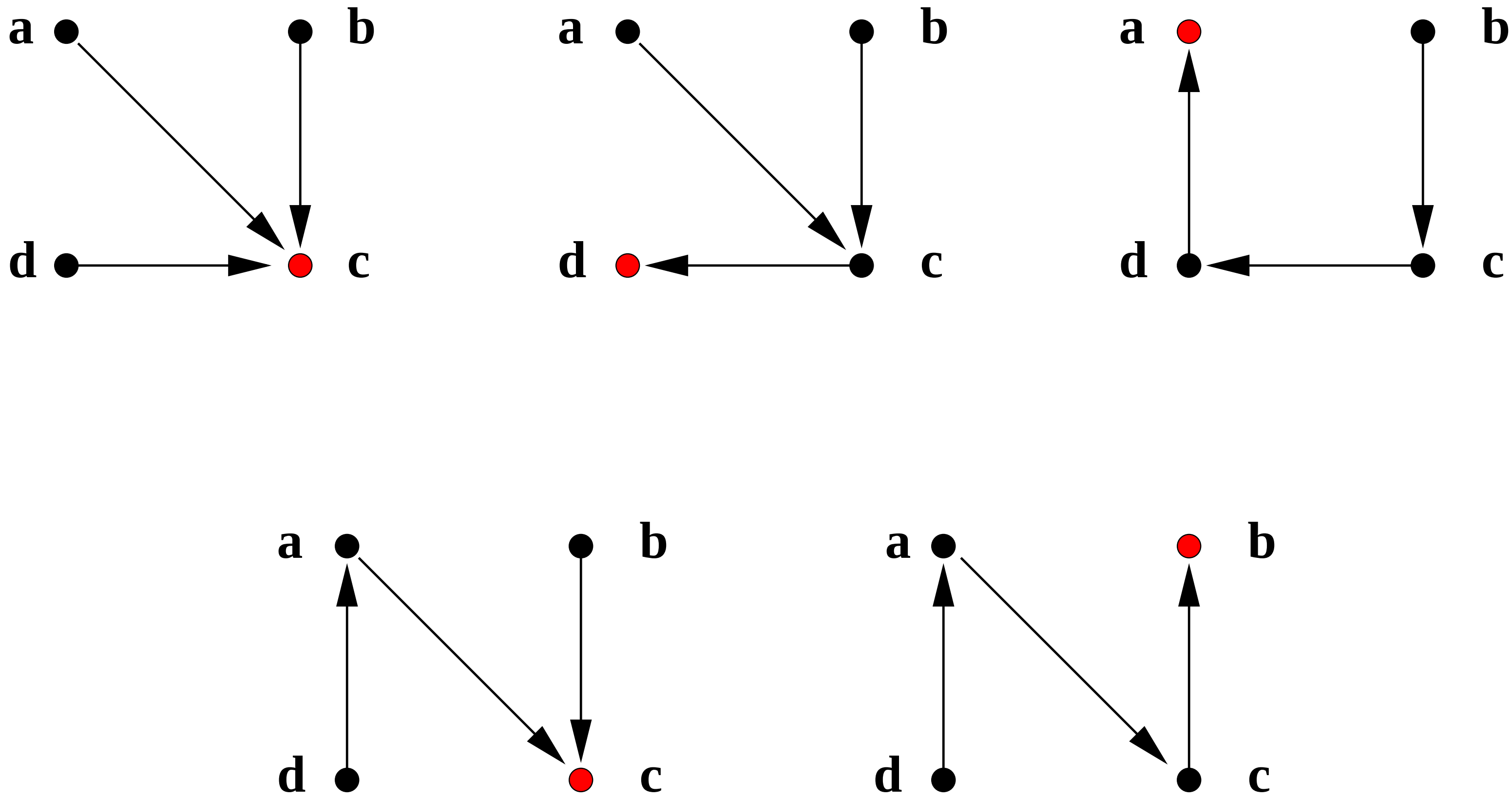
$\mathcal{M}$ moves from d to a



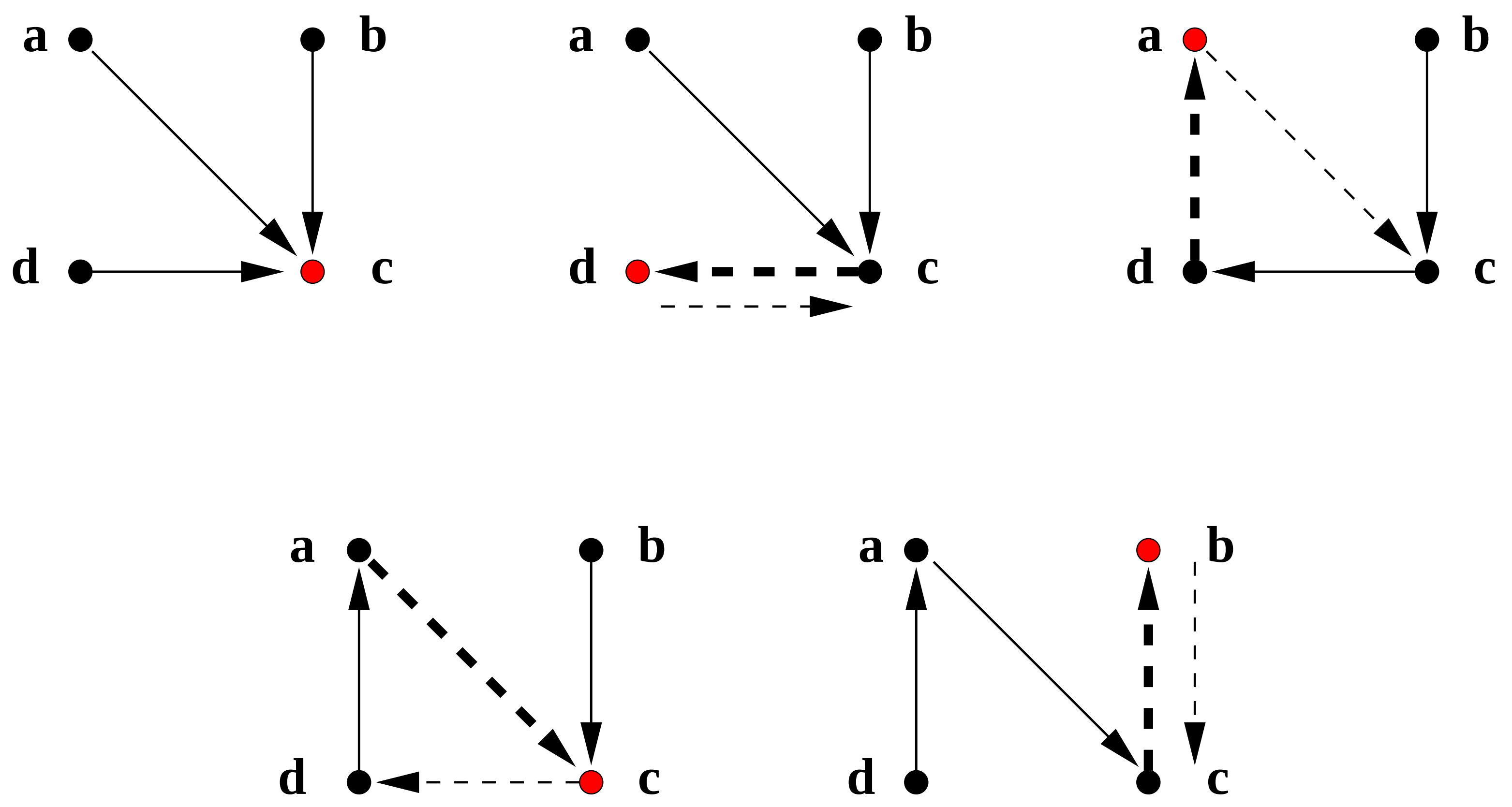
$\mathcal{M}$ moves from a to c

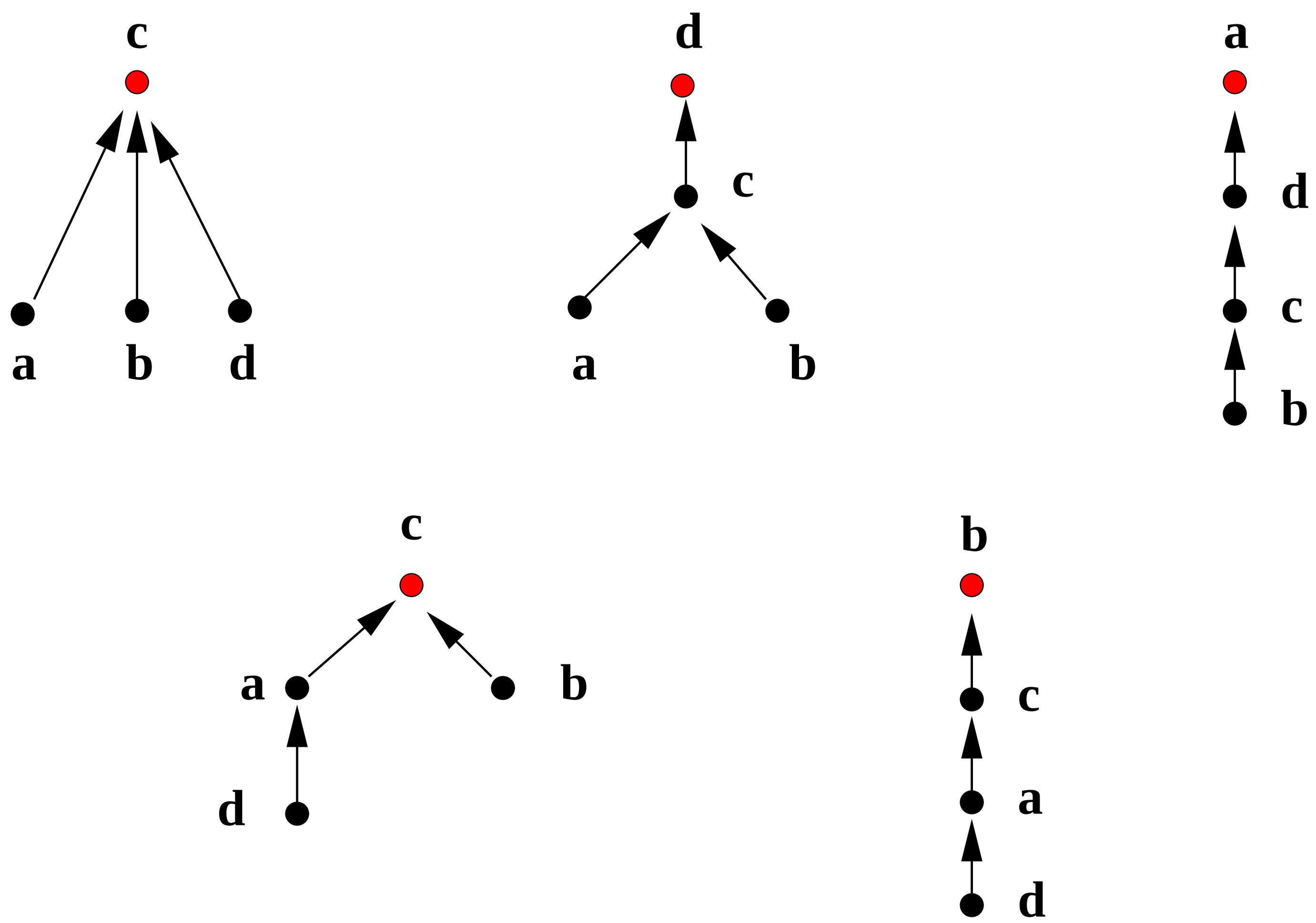


$\mathcal{M}$ moves from c to b

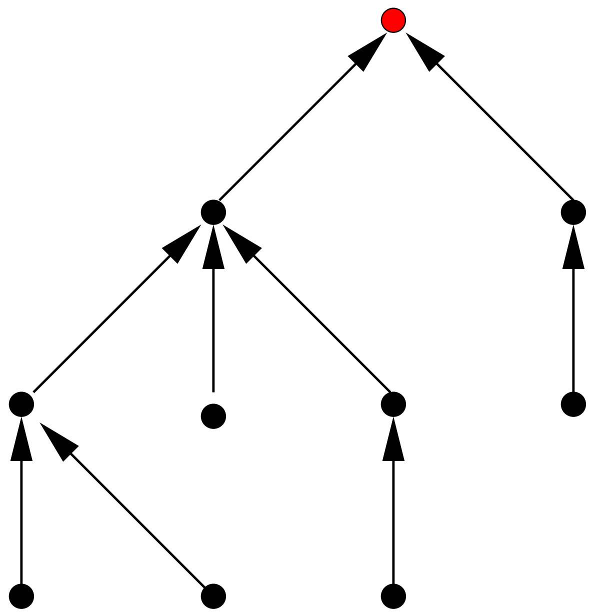


Showing the Structural Modifications



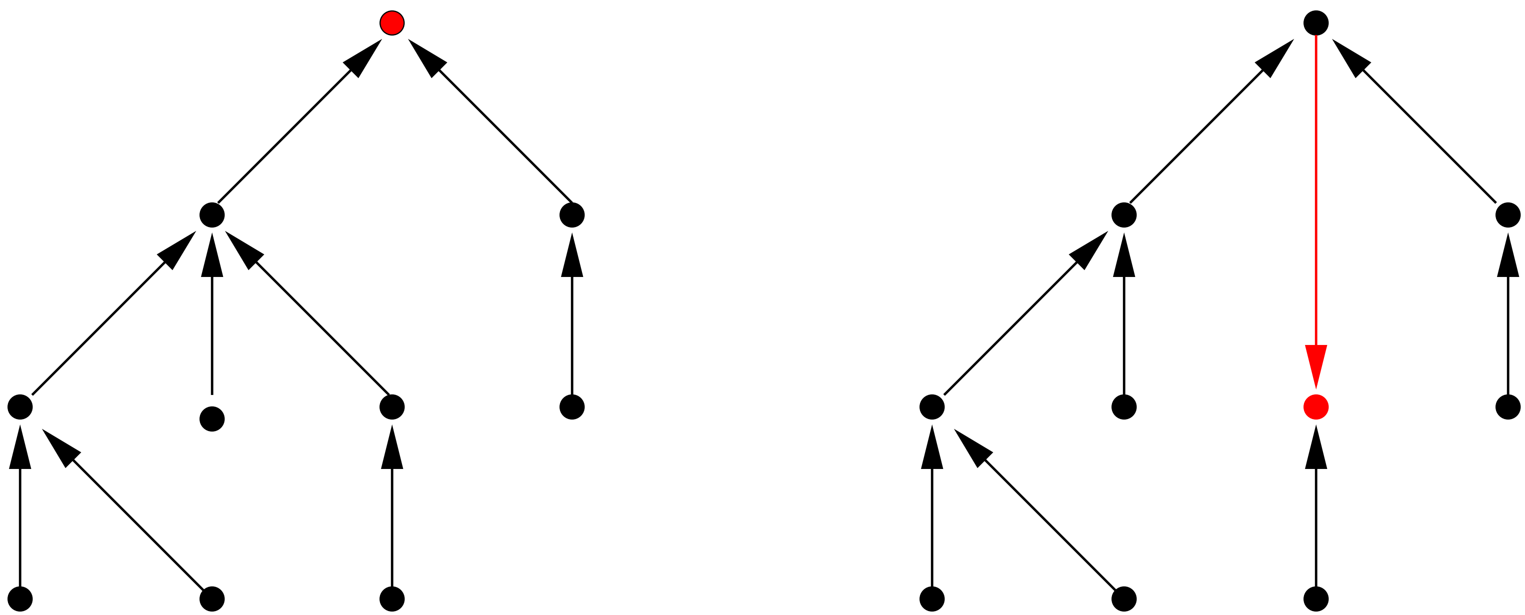

c
a
b
d
d
c
a
b
a
d
c
b
c
a
b
d
b
c
a
d

A Tree of Communications

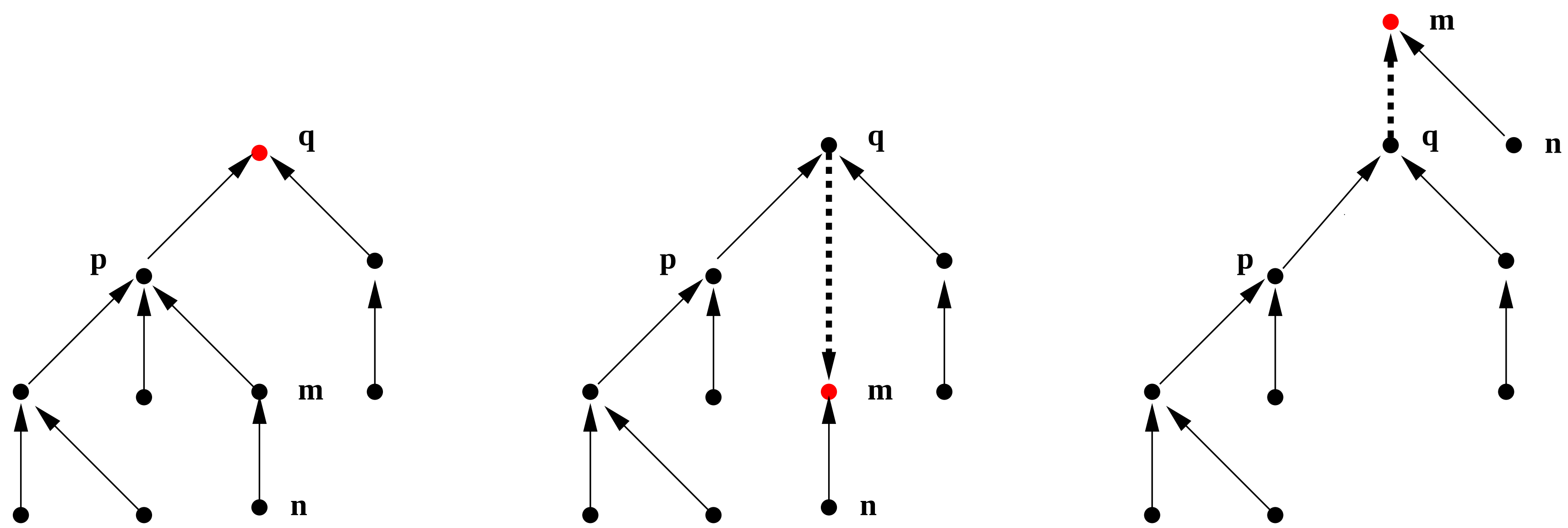


- The mobile $\mathcal{M}$ is at the root of a tree

Modification of the Tree



Comparing the two Situations



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

- S denotes the set of sites

- M denotes the set of messages

carrier sets: S
M

constant: il

axm0_1: $il \in S$

axm0_2: $\text{finite}(S)$

- Constant il denotes the initial location of the mobile

Initial Model: the Variables (1)

variables: l
c
p

inv0_1: $l \in S$

inv0_2: $c \in S \setminus \{l\} \rightarrow S$

inv0_3: $p \in M \leftrightarrow S$

- Variable l denotes the actual location of the mobile

- Variable c denotes the dynamic channel structure

- Variable p denotes the position of each message

This invariant states that the channel structure is a tree with:

- root: l
- parent function: c

inv0_4: $\forall T \cdot T \subseteq S \;\wedge\; T \subseteq c^{-1}[T] \;\Rightarrow\; T = \varnothing$

Initial Model: the Events (1)

init
$l := il$
$c := (S \setminus \{il\}) \times \{il\}$
$p := \varnothing$

rcv_agt
any s **where**
$s \neq l$
then
$l := s$
$c := (\{s\} \lhd c) \cup \{l \mapsto s\}$
end

- This event describes the move of the mobile from l to s

- The move of the mobile from l to s is supposed to be instantaneous

- Node s sends a message to the Mobile

- This message is stored locally

snd_msg
  any s, m **where**
    $s \in S$
    $m \in M \setminus \mathrm{dom}(p)$
  then
    $p(m) := s$
  end

Initial Model: the Events (3)

- Messages are either delivered or forwarded

dlv_msg
  any m **where**
    $m \in \mathrm{dom}(p)$
    $p(m) = l$
  then
    $p := \{m\} \lhd p$
  end

fwd_msg
  any m **where**
    $m \in \mathrm{dom}(p)$
    $p(m) \neq l$
  then
    $p(m) := c(p(m))$
  end

- When delivered, a message is removed

rcv_agt
  any s **where**
    $s \in S \setminus \{l\}$
  then
    $l := s$
    $c := (\{s\} \lhd c) \cup \{l \mapsto s\}$
  end

...	$\ldots$
Invariant **inv0_4**	$\forall T \cdot \begin{pmatrix} T \subseteq S \\ T \subseteq c^{-1}[T] \\ \Rightarrow \\ T = \varnothing \end{pmatrix}$
Guard of rcv_agt	$s \in S \setminus \{l\}$
$\vdash$	$\vdash$
Modified Invariant **inv0_4**	$\forall T \cdot \begin{pmatrix} T \subseteq S \\ T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\ \Rightarrow \\ T = \varnothing \end{pmatrix}$

$$
\begin{array}{l}
\quad \ldots \\
\quad \forall T \cdot \left(\begin{array}{l} T \subseteq S \\ T \subseteq c^{-1}[T] \\ \Rightarrow \\ T = \varnothing \end{array} \right) \\
\quad s \in S \setminus \{l\} \\
\quad T \subseteq S \\
\quad T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\
\vdash \\
\quad T = \varnothing
\end{array}
\quad \text{ALL\_L} \quad
\begin{array}{l}
\quad \ldots \\
\quad \left(\begin{array}{l} T \subseteq S \\ T \subseteq c^{-1}[T] \\ \Rightarrow \\ T = \varnothing \end{array} \right) \\
\quad s \in S \setminus \{l\} \\
\quad T \subseteq S \\
\quad T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\
\vdash \\
\quad T = \varnothing
\end{array}
\quad \textbf{SET} \ \ldots
$$

$$
\ldots \quad
\begin{array}{l}
\quad \ldots \\
\quad \left(\begin{array}{l} T \subseteq S \\ T \subseteq c^{-1}[T] \\ \Rightarrow \\ T = \varnothing \end{array} \right) \\
\quad s \in S \setminus \{l\} \\
\quad T \subseteq S \\
\quad T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\
\quad \color{red}{T \subseteq c^{-1}[T]} \\
\vdash \\
\quad T = \varnothing
\end{array}
\quad \text{IMP\_L} \quad
\begin{array}{l}
\quad \ldots \\
\quad T = \varnothing \\
\quad s \in S \setminus \{l\} \\
\quad T \subseteq S \\
\quad T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\
\quad T \subseteq c^{-1}[T] \\
\vdash \\
\quad T = \varnothing
\end{array}
\quad \text{HYP}
$$

- The key to this proof is the following lemma:

$$
\begin{array}{l}
\quad \ldots \\
\quad s \in S \setminus \{l\} \\
\quad T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\
\vdash \\
\quad T \subseteq c^{-1}[T]
\end{array}
$$

Hint: Consider two cases successively, $s \in T$ and $s \notin T$.

A More Concrete Situation

- $\boxed{\text{The moves of } \mathcal{M} \text{ are not completely instantaneous any more}}$

- The traveling of messages between sites still takes no time

- $\boxed{\text{The knowledge of the moves of } \mathcal{M} \text{ is not instantaneous any more}}$

When $\mathcal{M}$ Departs from Site l

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

- Agents of other sites are not aware of the move

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

- Messages at other sites can be forwarded (in general)

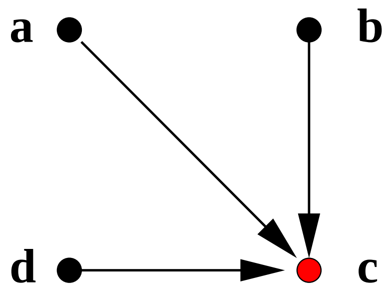
When $\mathcal{M}$ Arrives at its destination s (coming from l)

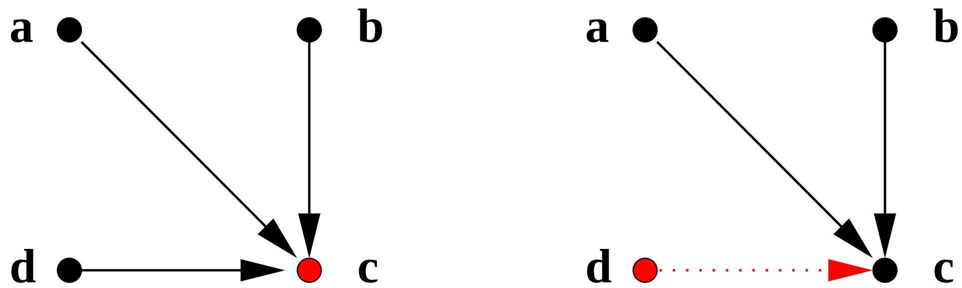
- It sends a "service message" to l to inform it about its new position

- Once l has received the "service message" it can forward again communication messages which were pending

- From now on, we have to distinguish:
  - communication messages (still instantaneous)
  - service messages (which take some time)

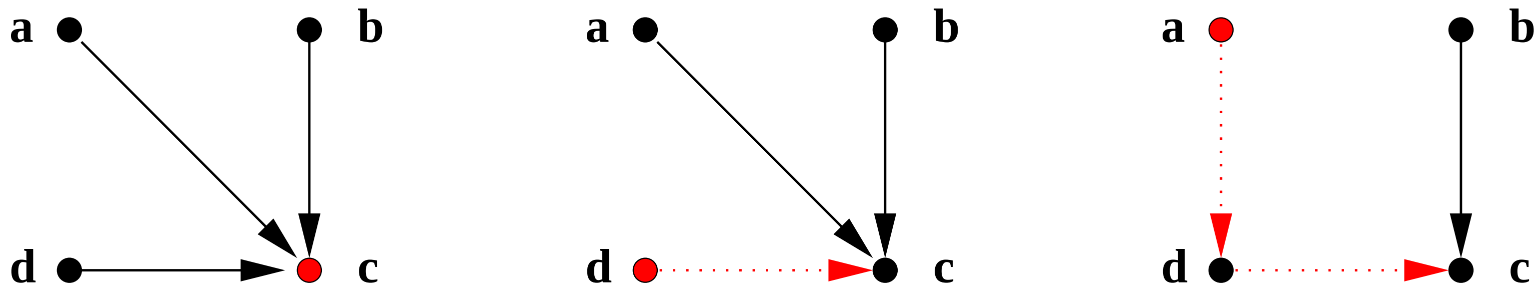
Initial Situation





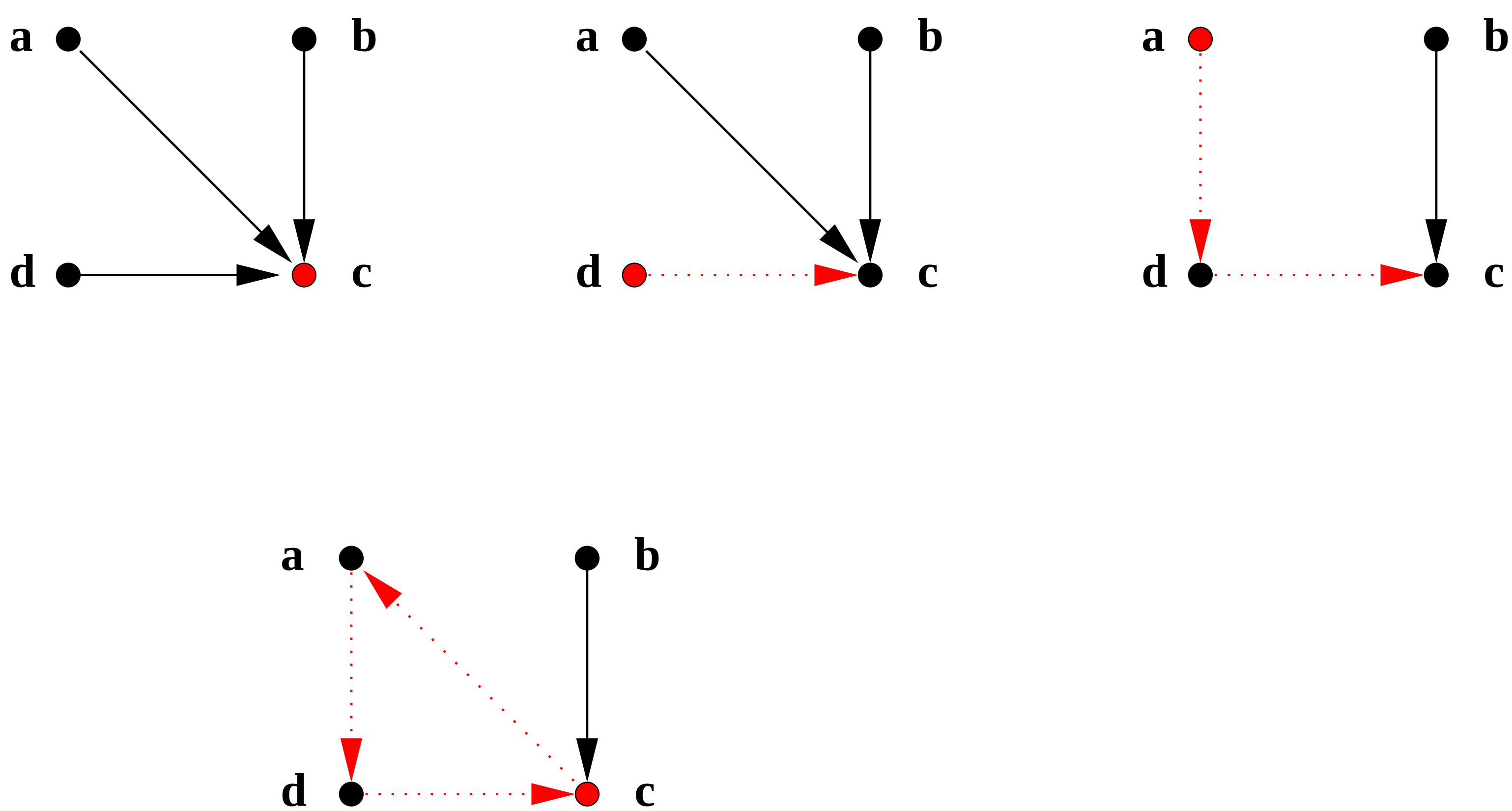
$\mathcal{M}$ sends a service message to c: "I am now in d"

Site c suspend sending com. msg. until it knows where $\mathcal{M}$ is



$\mathcal{M}$ sends a service message to d: "I am now in a"

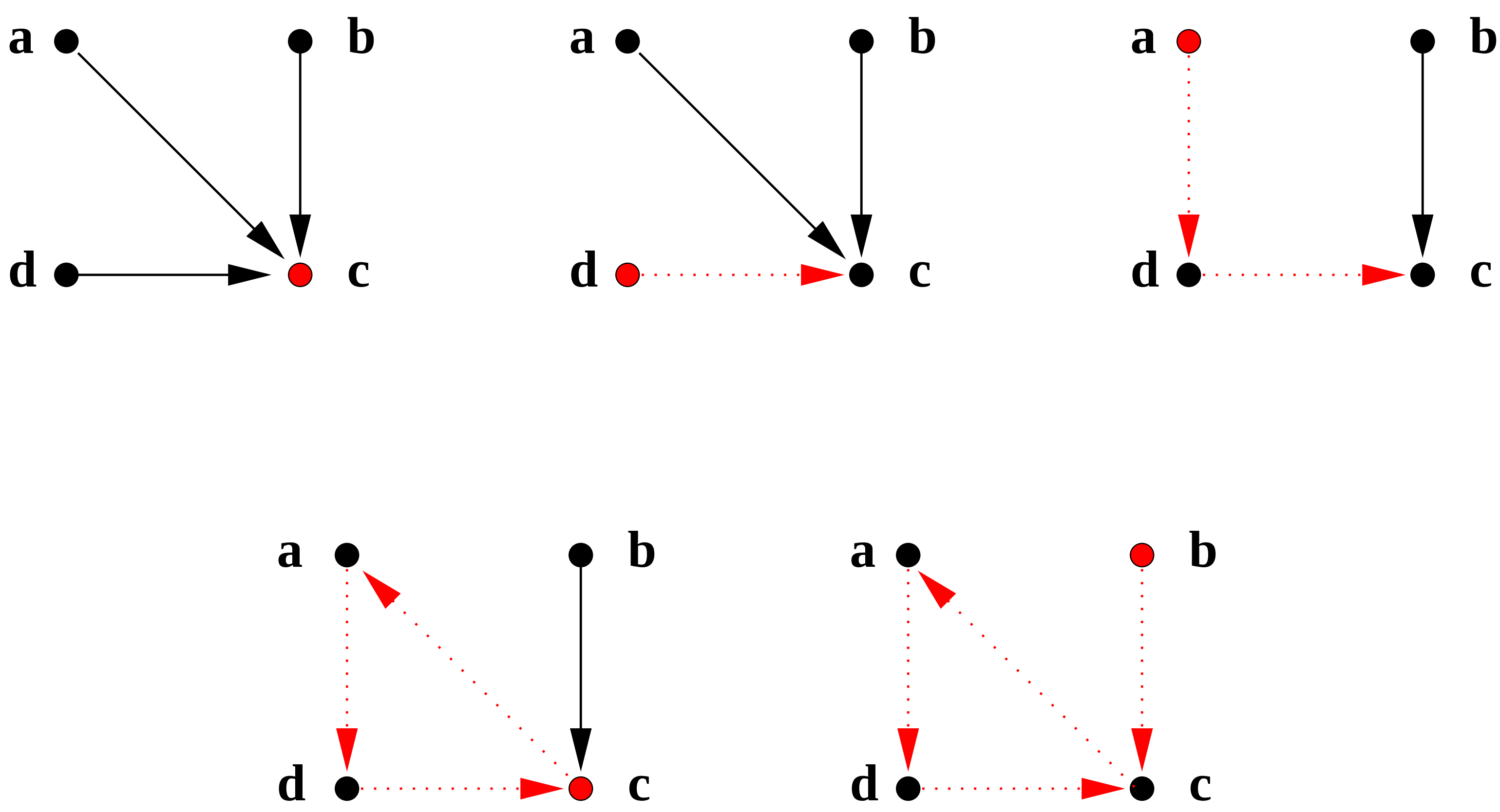
Site d suspend sending com. msg. until it knows where $\mathcal{M}$ is



$\mathcal{M}$ sends a service message to a: "I am now in c"

Site a suspend sending com. msg. until it knows where $\mathcal{M}$ is

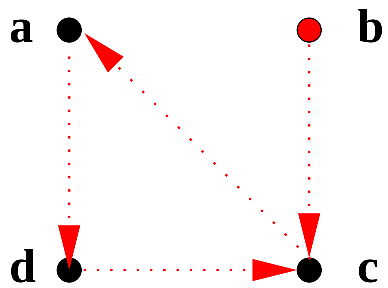
$\mathcal{M}$ moves from c to b



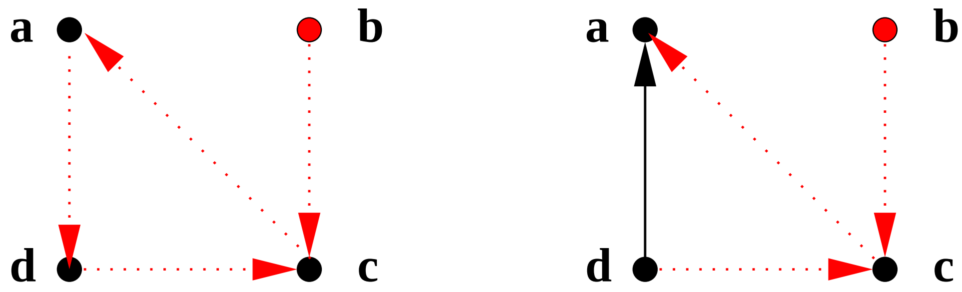
$\mathcal{M}$ sends a service message to c: "I am now in b"

Site c suspend sending com. msg. until it knows where $\mathcal{M}$ is

No Service Message has Arrived yet

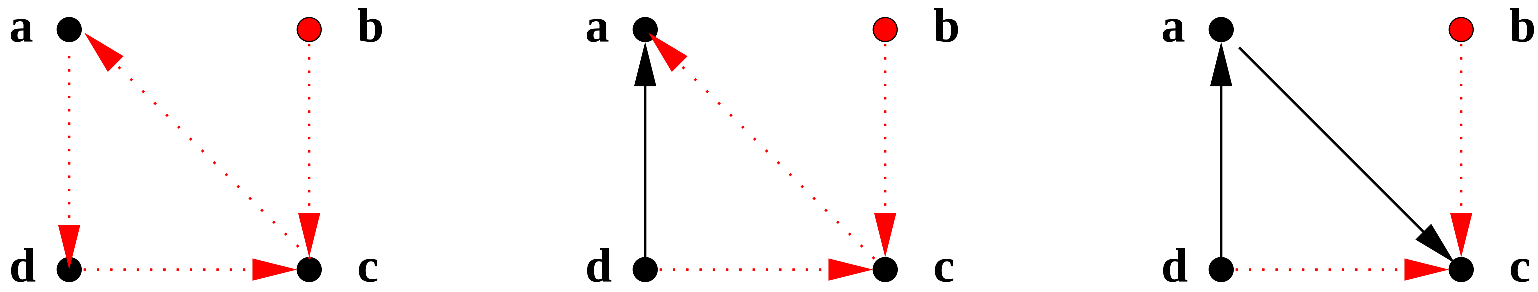


Service Message from a to d Arrives



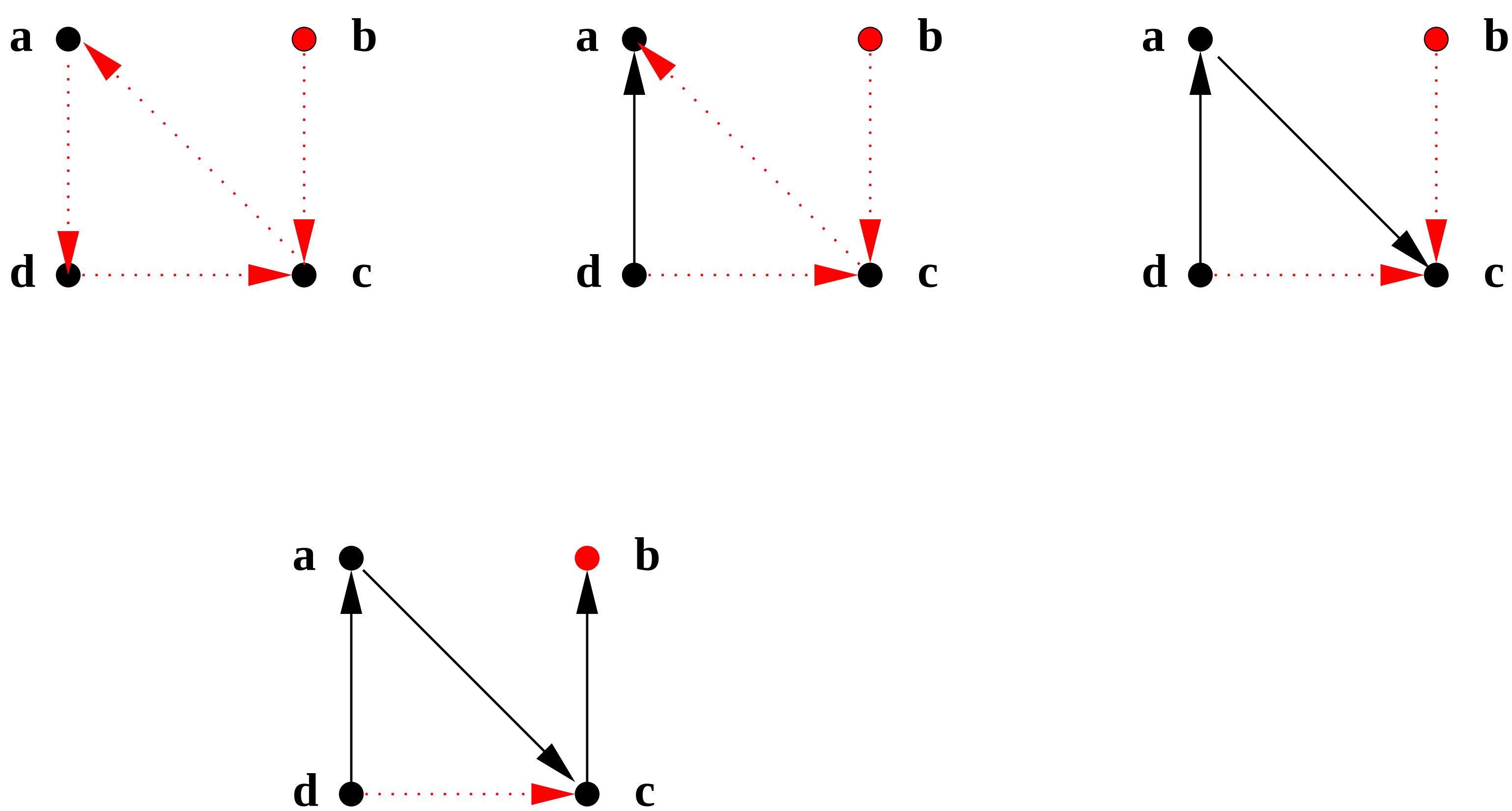
Site d believes $\mathcal{M}$ is in a. It now forwards pending com. msg. to a

Service Message from c to a Arrives



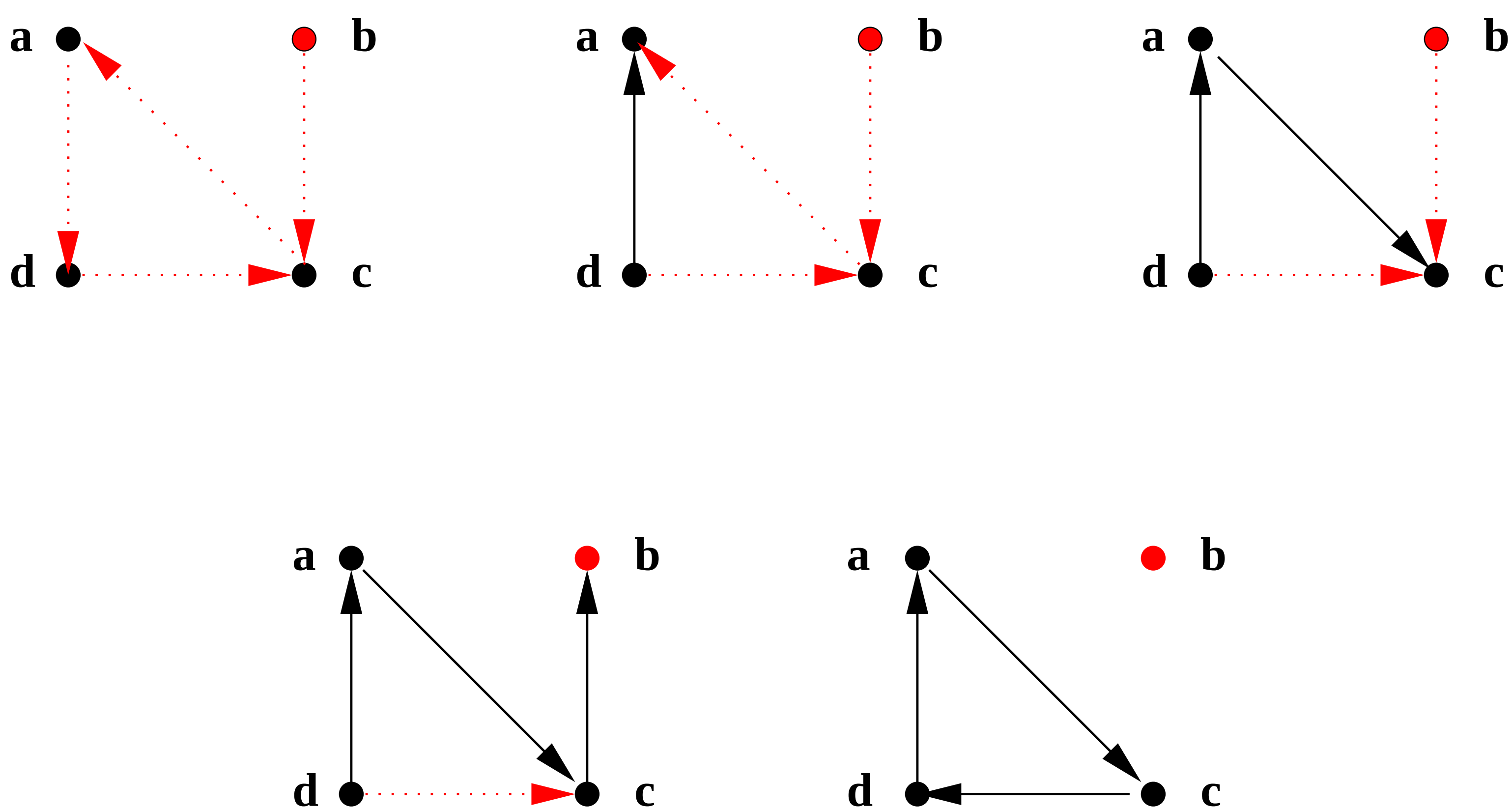
Site a believes $\mathcal{M}$ is in c. It now forwards pending com. msg. to c

Service Message from b to c Arrives



Site c believes $\mathcal{M}$ is in b. It now forwards pending com. msg. to b

Service Message from d to c Arrives. FAILURE



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

The tree structure is destroyed: we have a CYCLE.

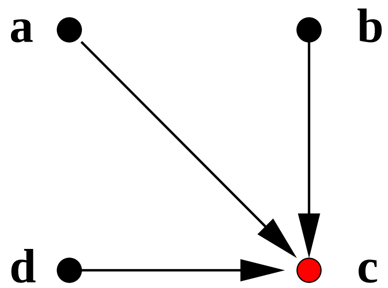
Analysis of Failure and "magic" Solution

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

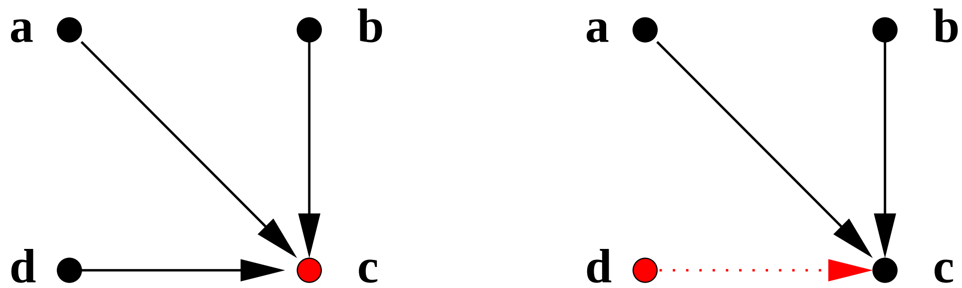
- We must preclude this to happen

- We shall suppose that we have the following "magic" behavior
    - When $\mathcal{M}$ sends a service message to site x
    - It is able to remove all other pending service messages whose destination is also x

Initial Situation



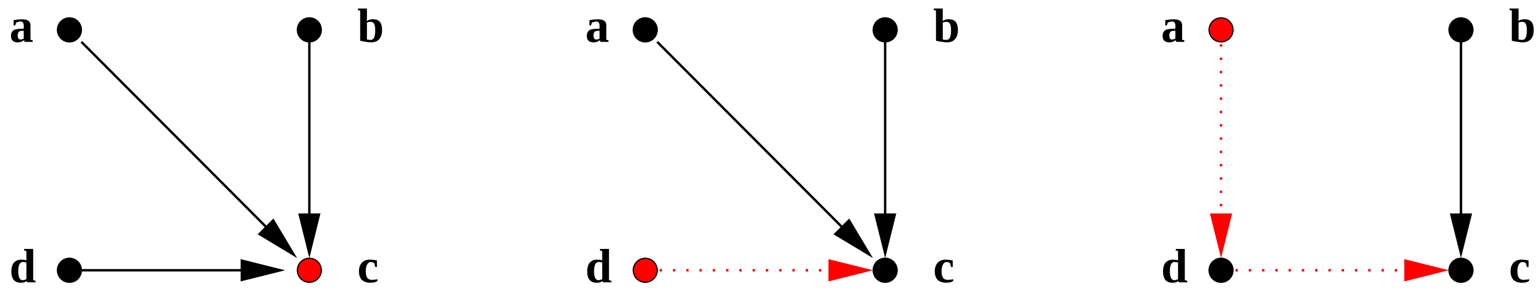
$\mathcal{M}$ moves from c to d



$\mathcal{M}$ sends a service message to c: "I am now in d"

Site c suspend forwarding com. msg. until it believes where $\mathcal{M}$ is

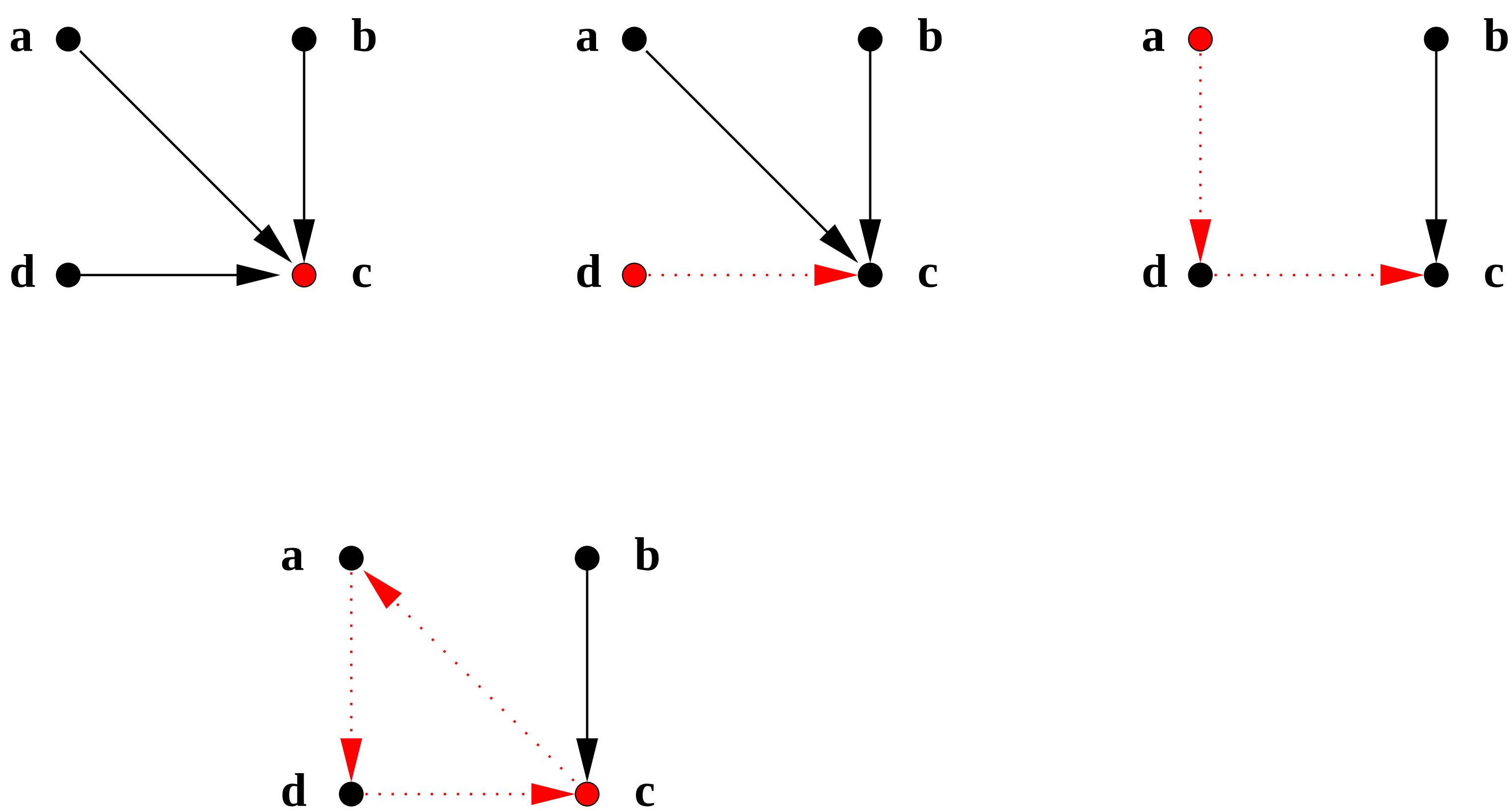
$\mathcal{M}$ moves from d to a



$\mathcal{M}$ sends a service message to d: "I am now in a"

Site d suspend forwarding com. msg. until it believes where $\mathcal{M}$ is

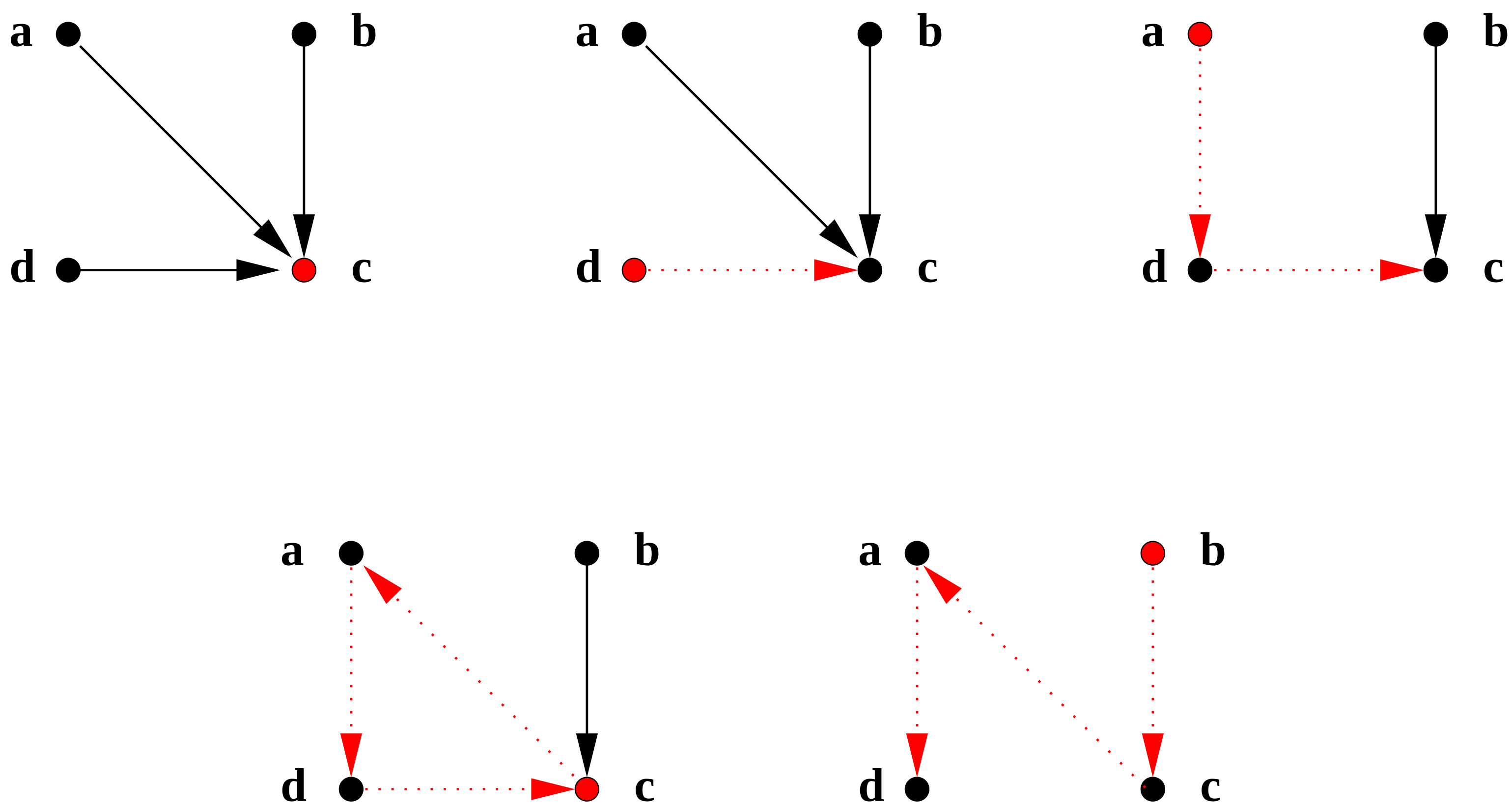
$\mathcal{M}$ moves from a to c



$\mathcal{M}$ sends a service message to a: "I am now in c"

Site a suspend forwarding com. msg. until it believes where $\mathcal{M}$ is

$\mathcal{M}$ moves from c to b



$\mathcal{M}$ sends a service message to c: "I am now in b"

$\mathcal{M}$ “magically” removes the other service message arriving to c

First Refinement: the State (1)

variables: l, p, d, a, da

inv1_1: $d \in S \setminus \{l\} \rightarrowtail\!\!\!\!\!+ S$

inv1_2: $a \in S \setminus \{l\} \rightarrowtail\!\!\!\!\!+ S$

inv1_3: $c = d \Leftarrow\!\!\!+ a$

- Variable d denotes the new dynamic tree structure

- Variable a denotes the service message channel.

- **inv1_3** denotes the link between c and the concrete d and a

$$\textbf{inv1\_2:} \quad a \setminus \{l\} \;\in\; S \rightarrowtail\!\!\!\!\twoheadrightarrow S$$

- $s1 \mapsto s2$ in a means a message from $s2$ (new site) to $s1$ (old site)

- Notice that the new site cannot be l

- At most one service message is in transit to site $s1$ (a is a function)

- This magic behavior is fundamental

First Refinement: the State (2)

inv1_4: $da \subseteq S$

inv1_5: $\mathrm{dom}(a) = da \setminus \{l\}$

- Variable da denotes the set of sites expecting a service message

- Such nodes cannot forward a message

First Refinement: the Events (1)

dlv_msg
any m **where**
$m \in \mathrm{dom}(p)$
$p(m) \notin da$
$p(m) = l$
then
$p := \{m\} \lhd\!\!\!- p$
end

fwd_msg
any m **where**
$m \in \mathrm{dom}(p)$
$p(m) \notin da$
$p(m) \neq l$
then
$p(m) := d(p(m))$
end

- The guards are now local

- We can later data-refine da with a local boolean variable

First Refinement: the Events (3)

leave_agt
  when
    $l \notin da$
  then
    $da := da \cup \{l\}$
  end

rcv_agt
  any s **where**
    $s \in S \setminus \{l\}$
    $l \in da$
  then
    $l := s$
    $a := (s \lhd a) \mathbin{\lhd\!\!\!-} (l \mapsto s)$
    $d := \{s\} \lhd d$
    $da := da \setminus \{s\}$
  end

- Event leave_agt is a new event where the set da is extended

- In event rcv_agt, the new site location s is removed from da

- A previous service message to l is removed.

rcv_srv

any s **where**

$s \in \mathrm{dom}(a)$

$s \neq l$

then

$d(s) := a(s)$

$a := \{s\} \lhd\!\!\!- a$

$da := da \setminus \{s\}$

end

- This is a new event

- It corresponds to the arrival of the service message

Second Refinement: Implementing the “magic” srv. channel

- Magic behavior when sending a new service message to x:
    - Pending service messages to x are removed

- The mobile $\mathcal{M}$ travels with a logical clock

- Each site has a last time counter

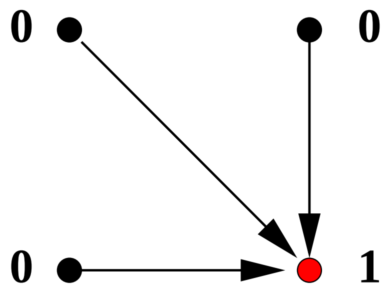
- This counter records the “time” of the last visit of $\mathcal{M}$

- When $\mathcal{M}$ arrives at a site y
    - it increments its logical clock
    - it stores its incremented clock in the last time counter of y
    - it sends a new service message to its previous location x

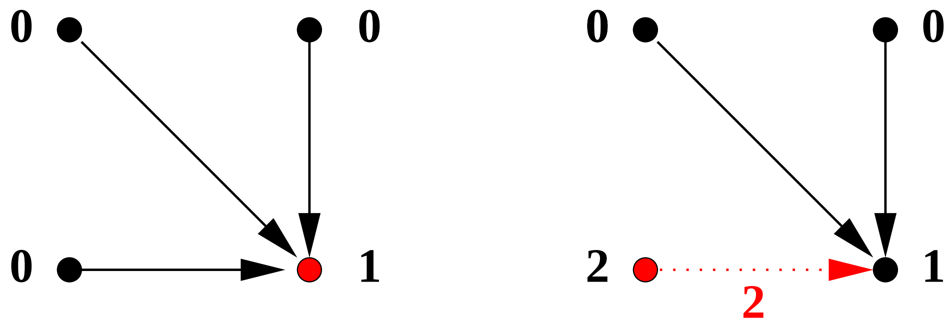
- The srv. msg. from y to x is stamped with the new clock value

- When a service message arrives at a site x, it is accepted
    - only if its stamp value is greater than the time counter of x
    - the last time counter takes the value of the stamp

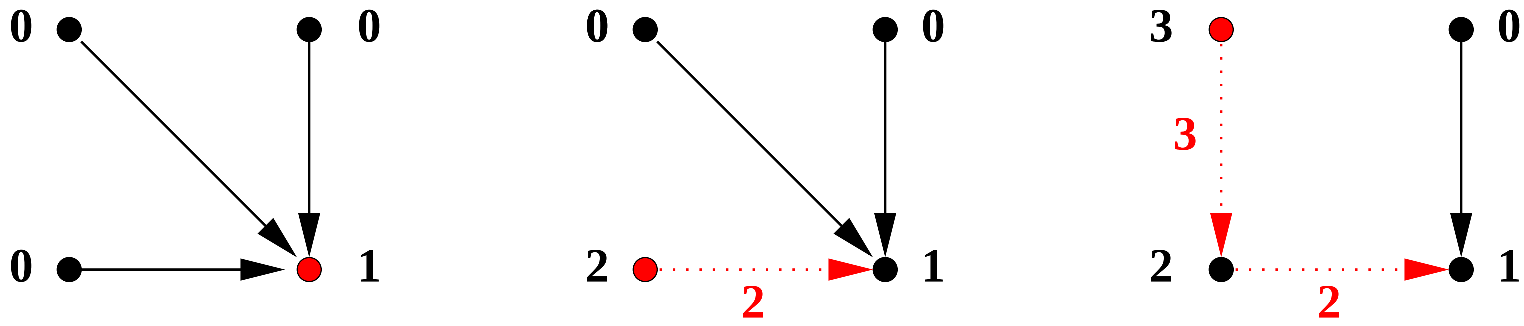
Initial Situation

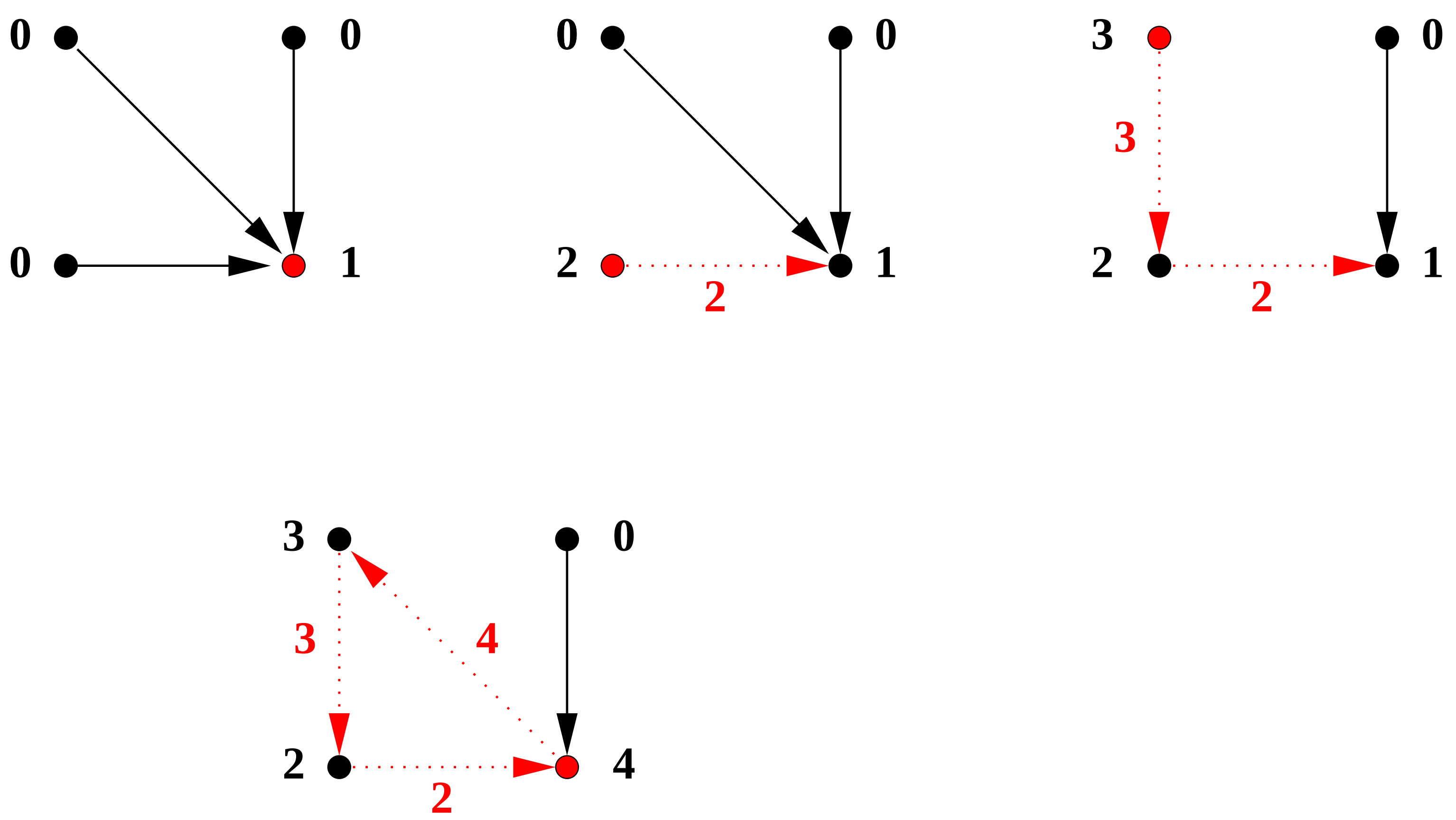


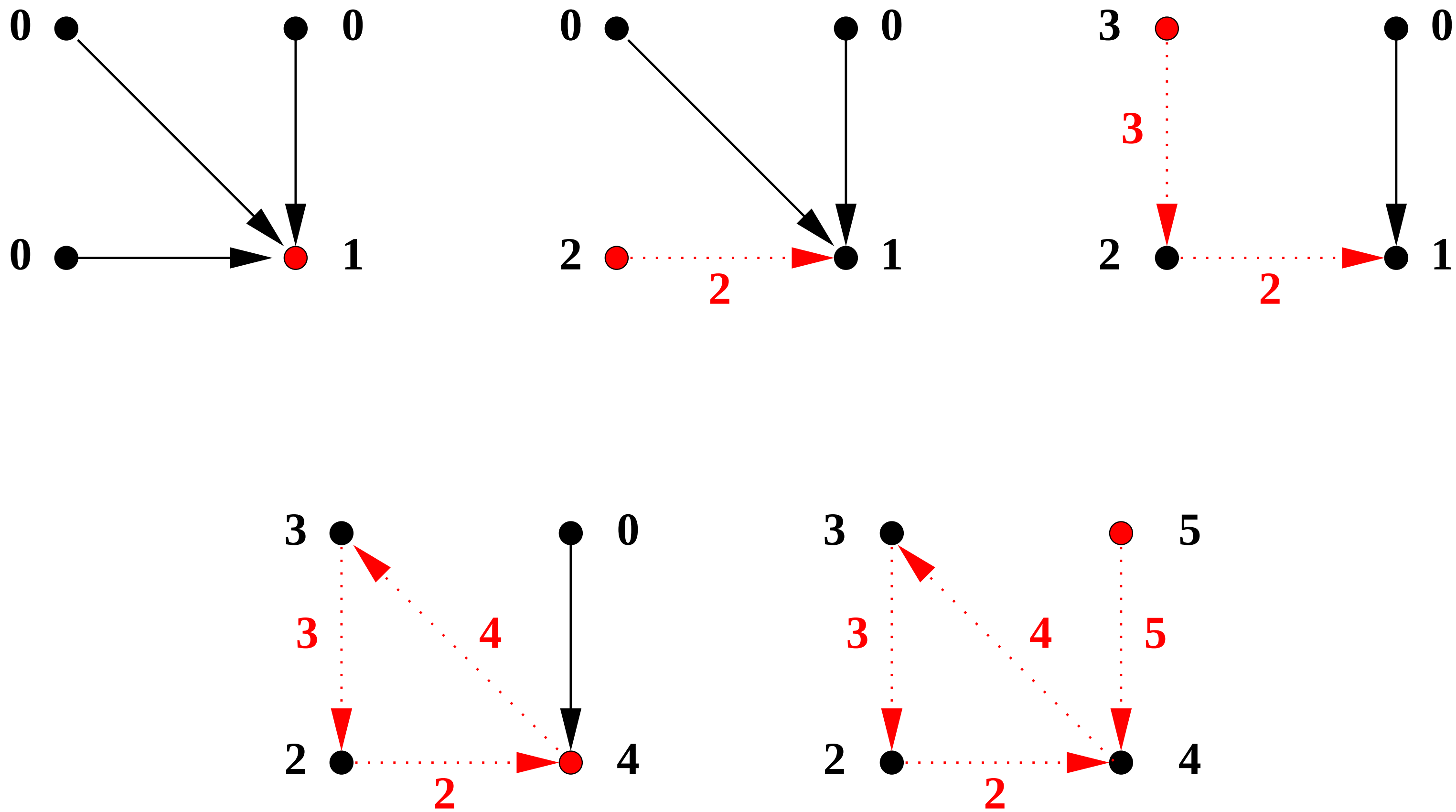
$\mathcal{M}$ moves from c to d



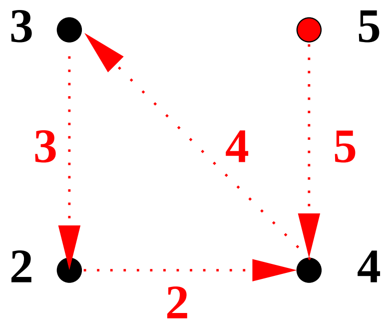
$\mathcal{M}$ moves from d to a



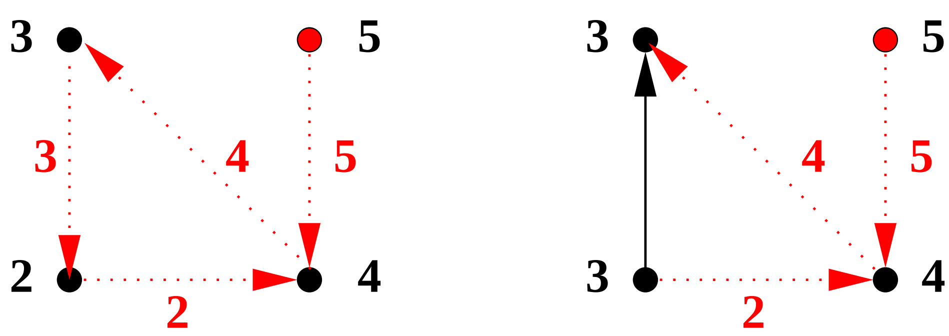

0
0
0
1
0
0
2
1
2
3
0
3
2
1
2
3
0
3
4
2
4
2


0
0
0
1
0
0
2
1
2
3
0
3
2
2
1
3
0
3
4
2
2
4
3
5
3
4
5
2
2
4

No Service Message has yet Arrived

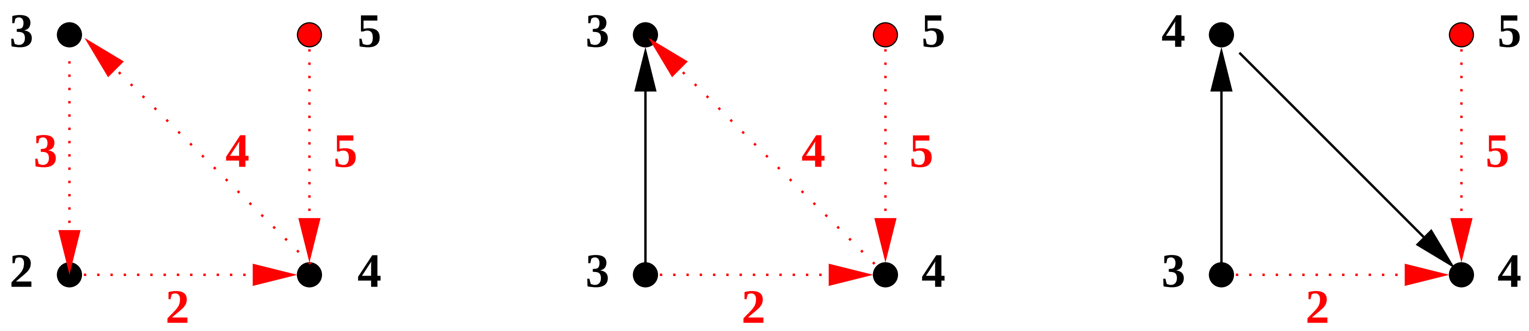


Service Message from a to d Arrives



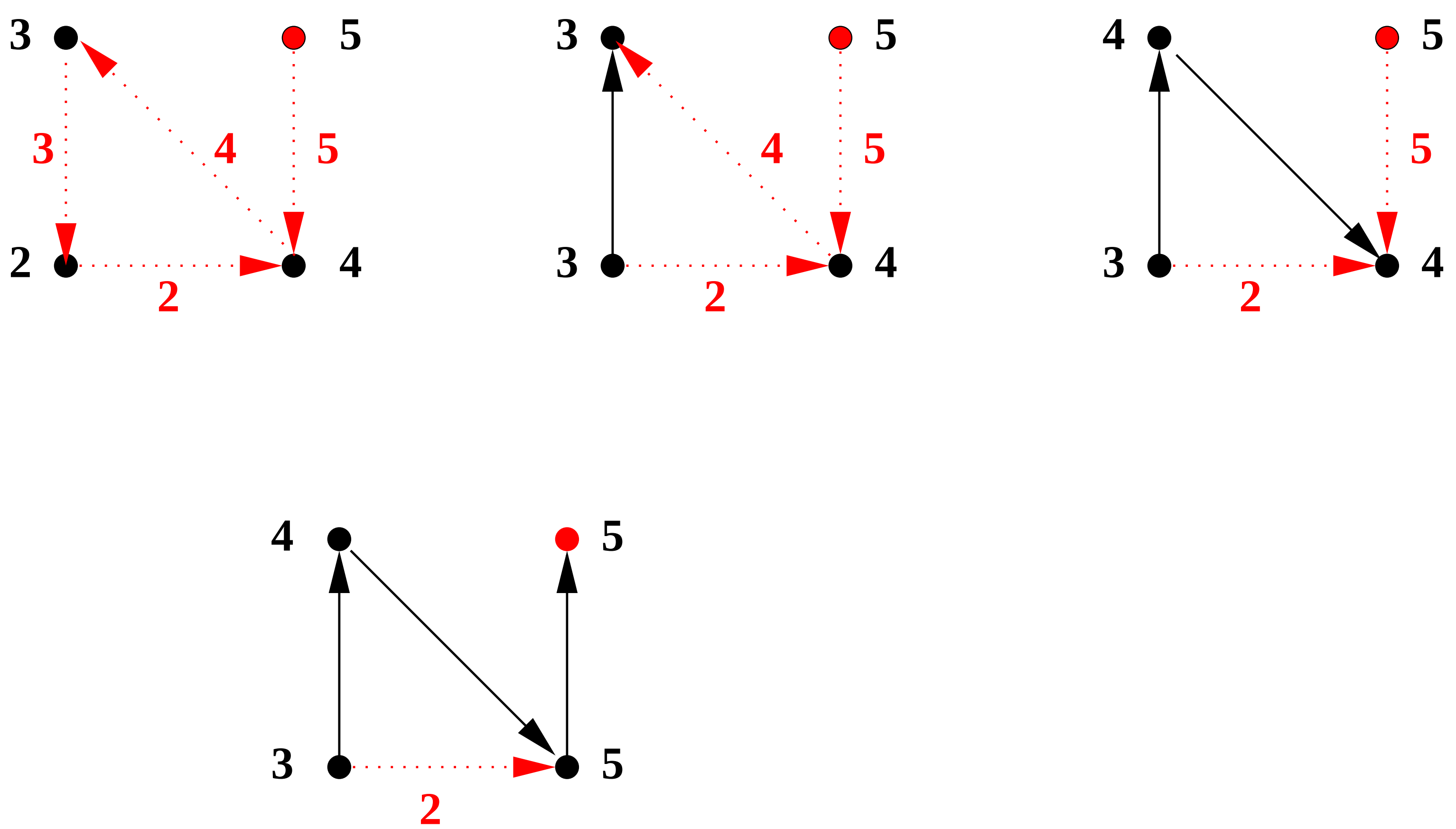
- It is accepted

Service Message from c to a Arrives

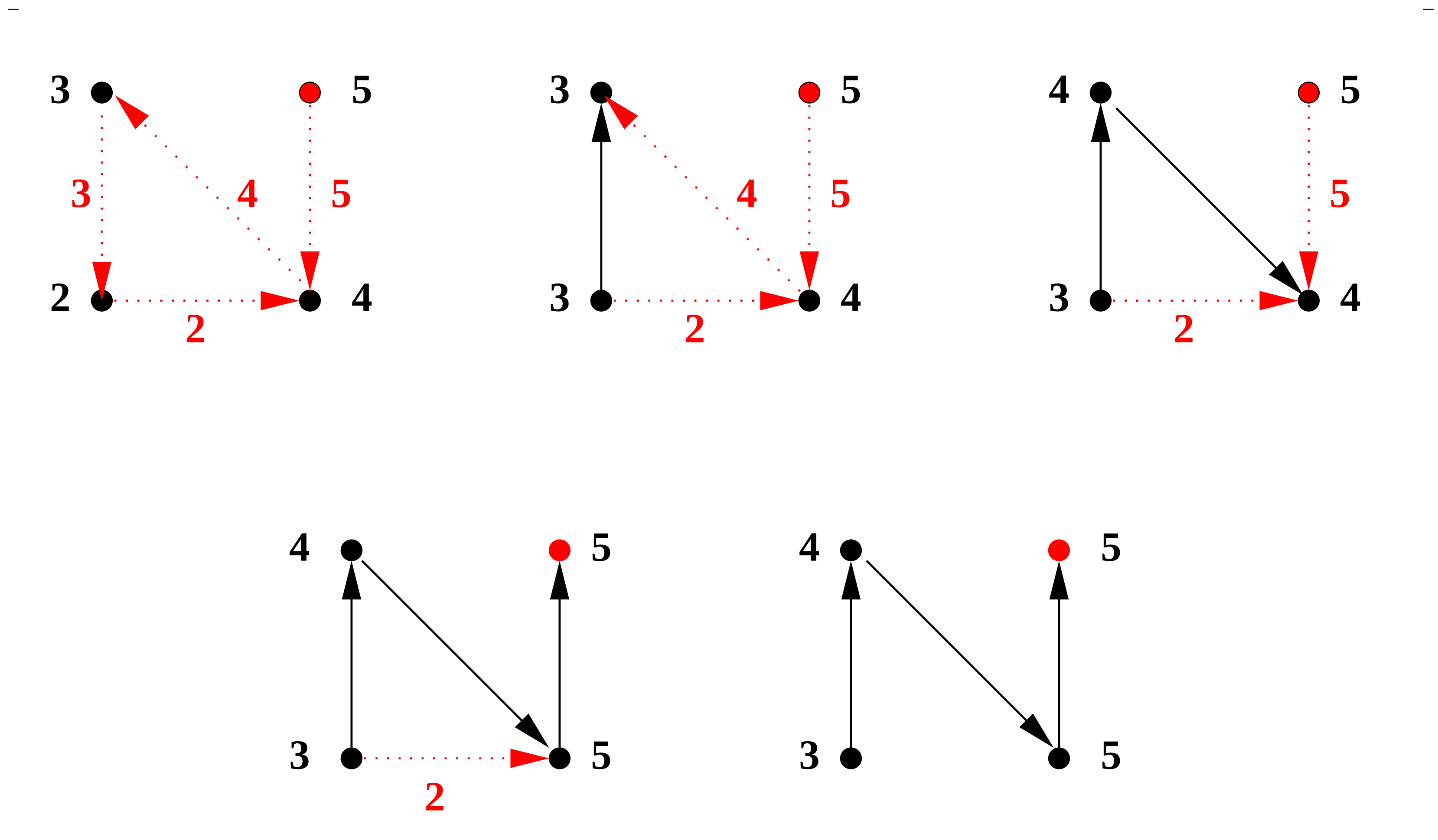


- It is accepted

Service Message from b to c Arrives



- It is accepted



- It is rejected

- Suppose:
    - $s1$ has emitted a service msg. to s at time 3
    - $s2$ has emitted a service msg. to s at time 5
    - $s3$ has emitted a service msg. to s at time 9

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

$$s \mapsto \{3 \mapsto s1, 5 \mapsto s2, 9 \mapsto s3\}$$

- In the abstract service channel we had: $s \mapsto s3$

Second Refinement: the State

variables: $l, p, d, da,$
k, t, b

inv2_1: $k \in \mathbb{N}$

inv2_2: $t \in S \rightarrow \mathbb{N}$

inv2_3: $b \in S \rightarrow (\mathbb{N} \leftrightarrow S)$

- Variable k is the clock taken by the Mobile when it travels

- Variable t denotes the time of the last visit of the Mobile to a site

- Variable b is the new service channel, it data-refines variable a

Connecting the Abstract a and the Concrete b (1)

- An abstract service message is the most recent concrete one

inv2_4: $$\forall s \cdot \left(\begin{array}{l} s \in \mathrm{dom}(a) \\ \Rightarrow \\ \mathrm{dom}(b(s)) \neq \varnothing \\ a(s) = b(s)(\max(\mathrm{dom}(b(s)))) \end{array}\right)$$

$$\textbf{inv2\_5:} \quad \forall \boldsymbol{s} \cdot \left(\begin{array}{l} \boldsymbol{s} \in \boldsymbol{S} \\ \mathbf{dom}(\boldsymbol{b}(\boldsymbol{s})) \neq \varnothing \\ \boldsymbol{t}(\boldsymbol{s}) < \mathbf{max}(\mathbf{dom}(\boldsymbol{b}(\boldsymbol{s}))) \\ \Rightarrow \\ \boldsymbol{s} \in \mathbf{dom}(\boldsymbol{a}) \end{array} \right)$$

- This technical invariant will help us proving guard strengthening for event rcv_srv

More Invariants Between k and t

inv2_6: $\forall s \cdot s \in S \;\Rightarrow\; \mathrm{dom}(b(s)) \subseteq 0 \,..\, k$

inv2_7: $t(l) = k$

inv2_8: $\forall s \cdot s \in S \setminus \{l\} \;\Rightarrow\; t(s) \leq k$

The Key Invariant

- The only service message stamp to a site s which is strictly greater than the time of last visit to that site s is the maximum one.

inv2_9: $$\forall\, s, n \cdot \begin{pmatrix} s \in S \\ n \in \mathrm{dom}(b(s)) \\ t(s) < n \\ \Rightarrow \\ n = \max(\mathrm{dom}(b(s))) \end{pmatrix}$$

Second Refinement: the Events (1)

- Sending the service message with the time stamp $k + 1$

(abstract-)rcv_agt
any s **where**
$s \in S \setminus \{l\}$
$l \in da$
then
$l := s$
$a := (s \lhd a) \lhd\!\!- (l \mapsto s)$
$d := \{s\} \lhd\!\!\!- d$
$da := da \setminus \{s\}$
end

(concrete-)rcv_agt
any s **where**
$s \in S \setminus \{l\}$
$l \in da$
then
$l := s$
$t(s) := k + 1$
$k := k + 1$
$b(l) := b(l) \lhd\!\!- \{k + 1 \mapsto s\}$
$d := \{s\} \lhd\!\!\!- d$
$da := da \setminus \{s\}$
end

(abstract-)rcv_srv
any s **where**
$s \in \mathrm{dom}(a)$
$s \neq l$
then
$d(s) := a(s)$
$a := \{s\} \lhd a$
$da := da \setminus \{s\}$
end

(concrete-)rcv_srv
any s, n **where**
$s \in S$
$n \in \mathrm{dom}(b(s))$
$t(s) < n$
then
$d(s) := b(s)(n)$
$t(s) := n$
$da := da \setminus \{s\}$
$b(s) := \{n\} \lhd b(s)$
end

$$\forall s \cdot \left(\begin{array}{l} s \in S \\ \mathrm{dom}(b(s)) \neq \varnothing \\ t(s) < \max(\mathrm{dom}(b(s))) \\ \Rightarrow \\ s \in \mathrm{dom}(a) \end{array} \right)$$

$$\forall s, n \cdot \left(\begin{array}{l} s \in S \\ n \in \mathrm{dom}(b(s)) \\ t(s) < n \\ \Rightarrow \\ n = \max(\mathrm{dom}(b(s))) \end{array} \right)$$

variables: $\boldsymbol{l, p, d, b,}$
$\boldsymbol{dab, k, t}$

inv3_1: $dab \in S \rightarrow \mathrm{BOOL}$

inv3_3: $\forall x \cdot x \in S \Rightarrow (x \in da \Leftrightarrow dab(x) = \mathrm{TRUE})$

Third Refinement: The Events (1)

init
$l := il$
$p := \varnothing$
$d := (S \setminus \{il\}) \times \{il\}$
$b := S \times \{\varnothing\}$
$dab := S \times \{\mathrm{FALSE}\}$
$k := 1$
$t := S \times \{0\} \;\lhd\!\!\!- \{il \mapsto 1\}$

leave_agt
when
$dab(l) = \mathrm{FALSE}$
then
$dab(l) := \mathrm{TRUE}$
end

Third Refinement: The Events (2)

rcv_agt
any s **where**
$s \in S \setminus \{l\}$
$dab(l) = \mathrm{TRUE}$
then
$l := s$
$t(s) := k + 1$
$k := k + 1$
$b(l) := b(l) \mathbin{\lhd\!\!-} \{k + 1 \mapsto s\}$
$d := \{s\} \mathbin{\lhd\!\!\!-} d$
$dab(s) := \mathrm{FALSE}$
end

rcv_srv
any s, n **where**
$s \in S$
$n \in \mathrm{dom}(b(s))$
$t(s) < n$
then
$d(s) := b(s)(n)$
$t(s) := n$
$dab(s) := \mathrm{FALSE}$
end

dlv_msg
any m **where**
$m \in \mathrm{dom}(p)$
$dab(p(m)) = \mathrm{FALSE}$
$p(m) = l$
then
$p := \{m\} \lhd p$
end

fwd_msg
any m **where**
$m \in \mathrm{dom}(p)$
$dab(p(m)) = \mathrm{FALSE}$
$p(m) \neq l$
then
$p(m) := d(p(m))$
end

Initial Model	11	0
1st Reft.	23	2
2nd Reft.	70	14
3rd Reft.	25	0
Total	129	16