Development of a Network Topology Discovery Algorithm

Thai Son Hoang

David Basin

Hironobu Kuruma

Jean-Raymond Abrial

Reference

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

Developing Topology Discovery in Event-B

Science of Computer Programming (2009), 74 (11-12).

What is presented here is another development

of the same problem

Why it is an Interesting Subject

- The final distributed algorithm is rather simple

- But the corresponding (formal) development is not that simple

- This example proposes some interesting techniques that can be used elsewhere

- We shall also insist on the importance of requirement analysis (requirements: what for?)

Outline

- Prologue

- Requirements

- Difficulties and Strategy

- Formal Development

- Conclusion

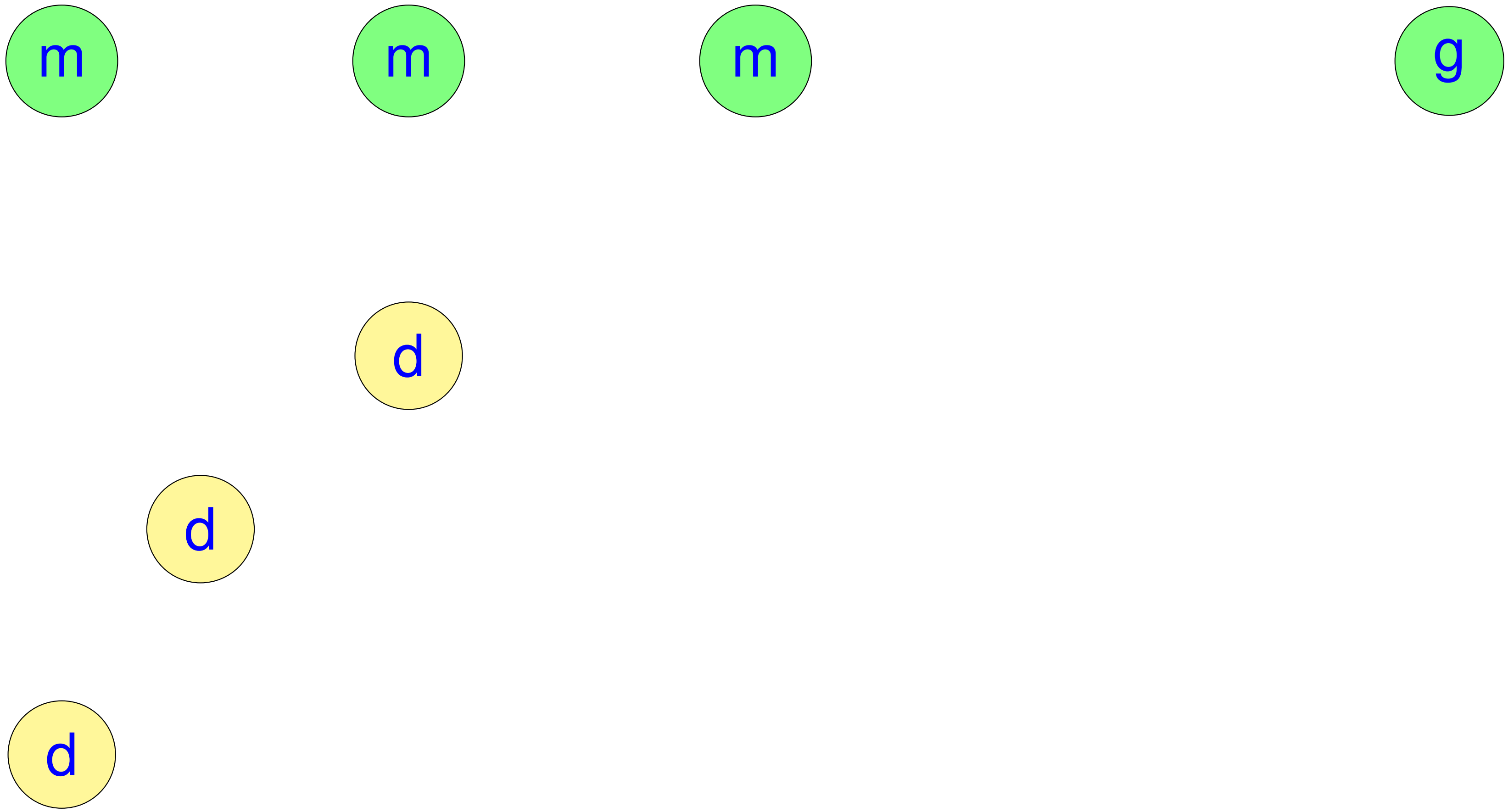
The Master and Dog paradigm

The Master rides a bike

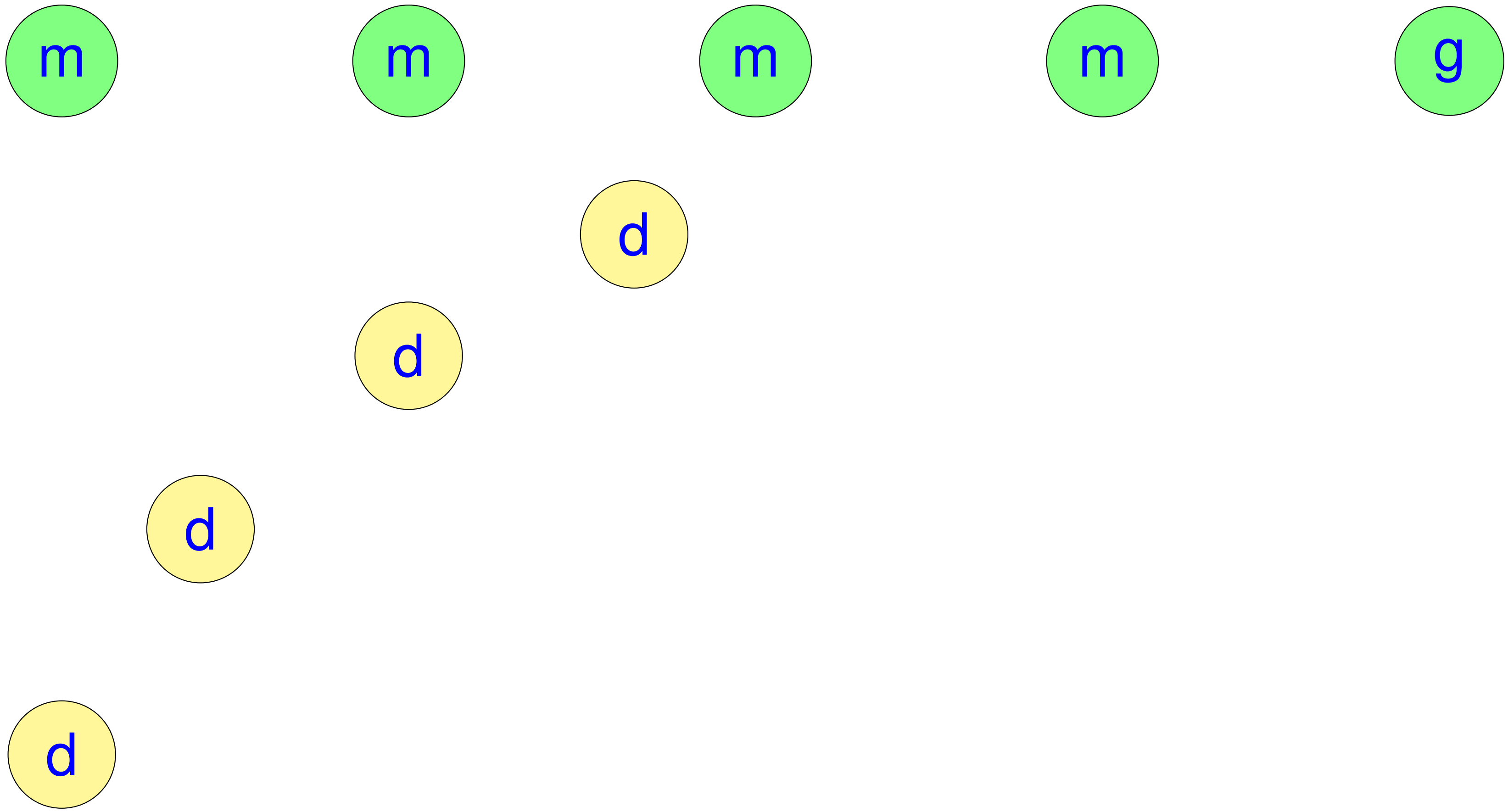
The Dog tries to get to his Master

Master m, Dog d, and Goal g

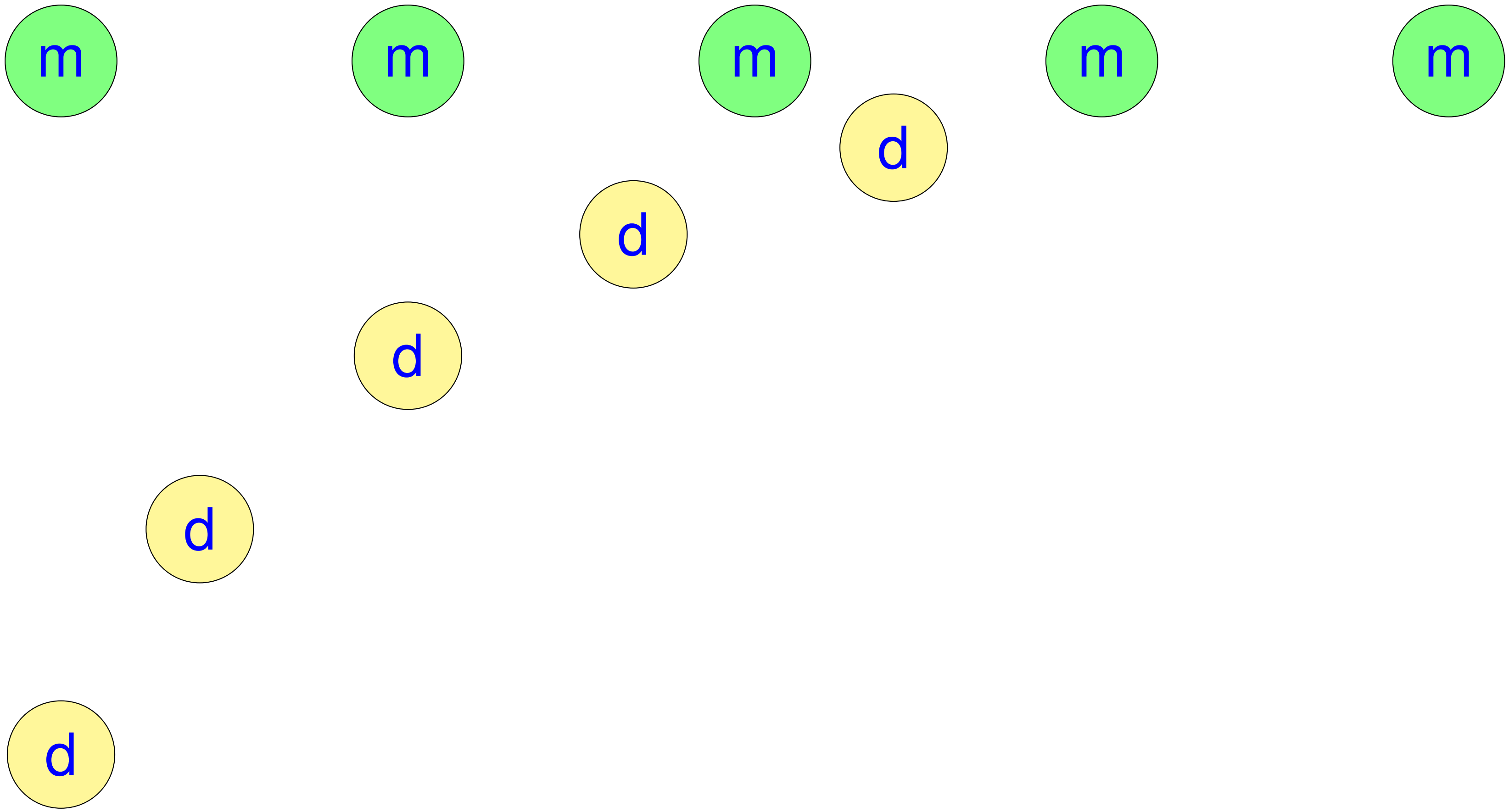
Master and Dog


m
m
m
g
d
d
d

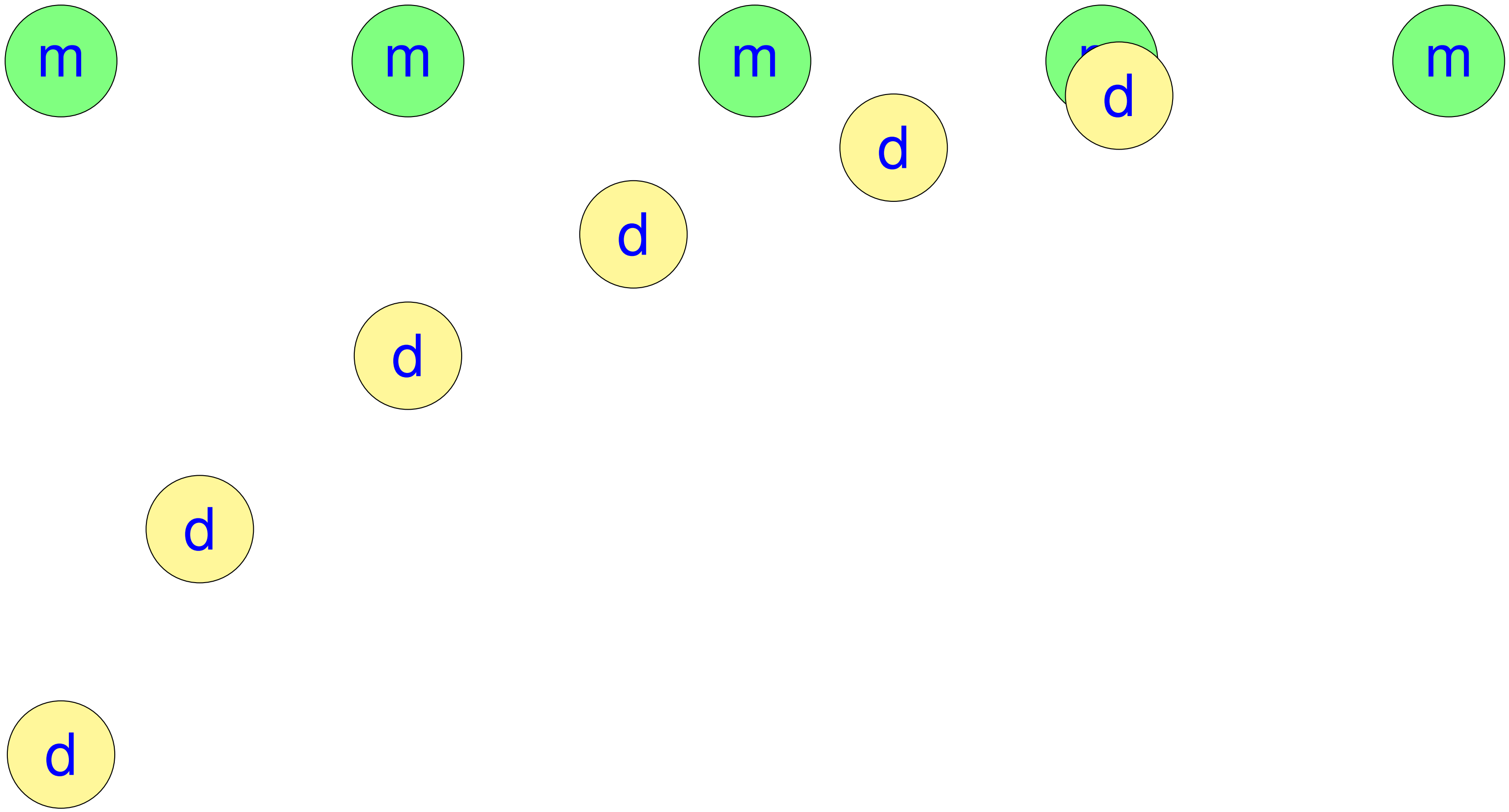
Master and Dog



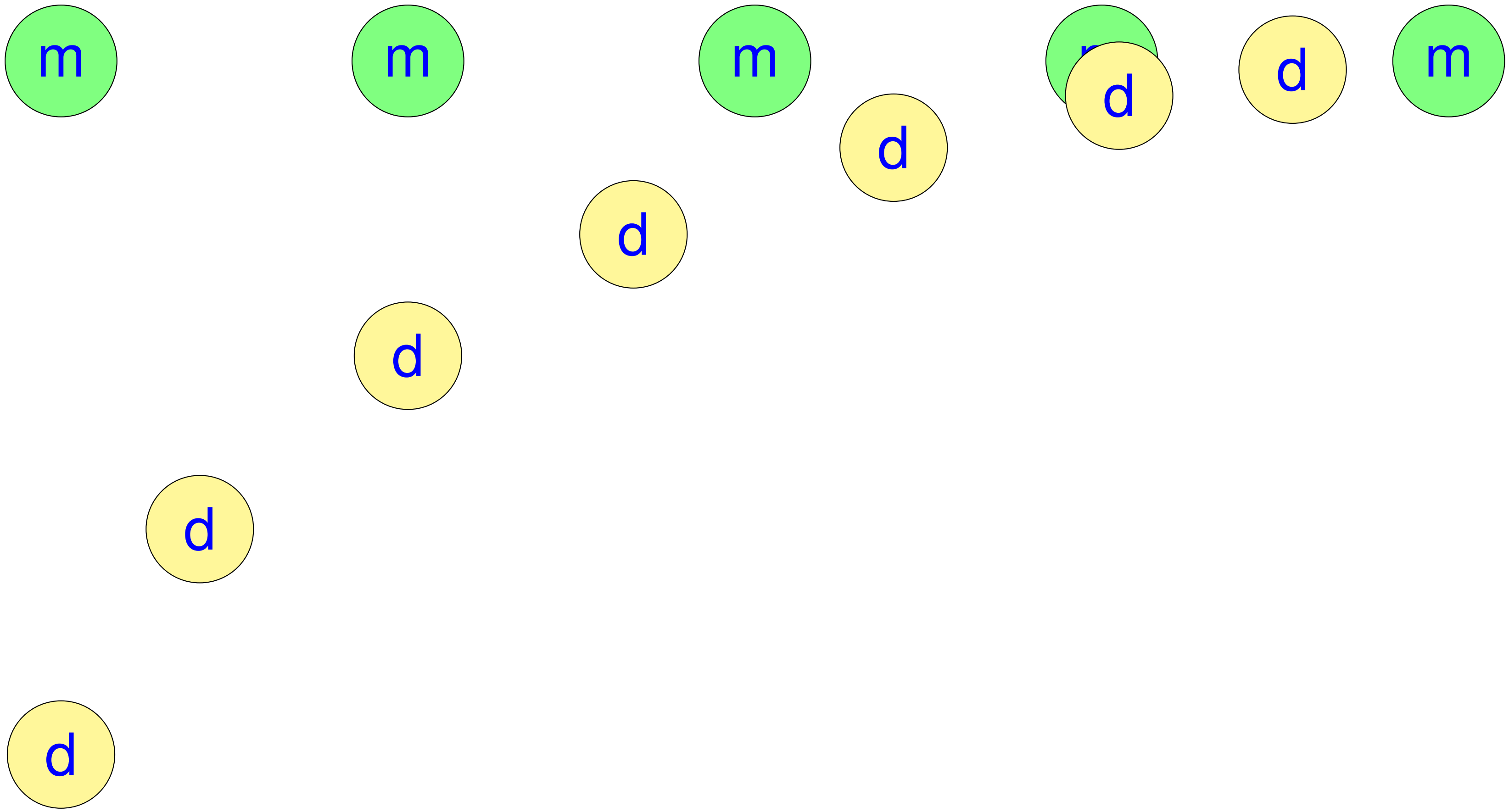
Master and Dog: the Master Stops



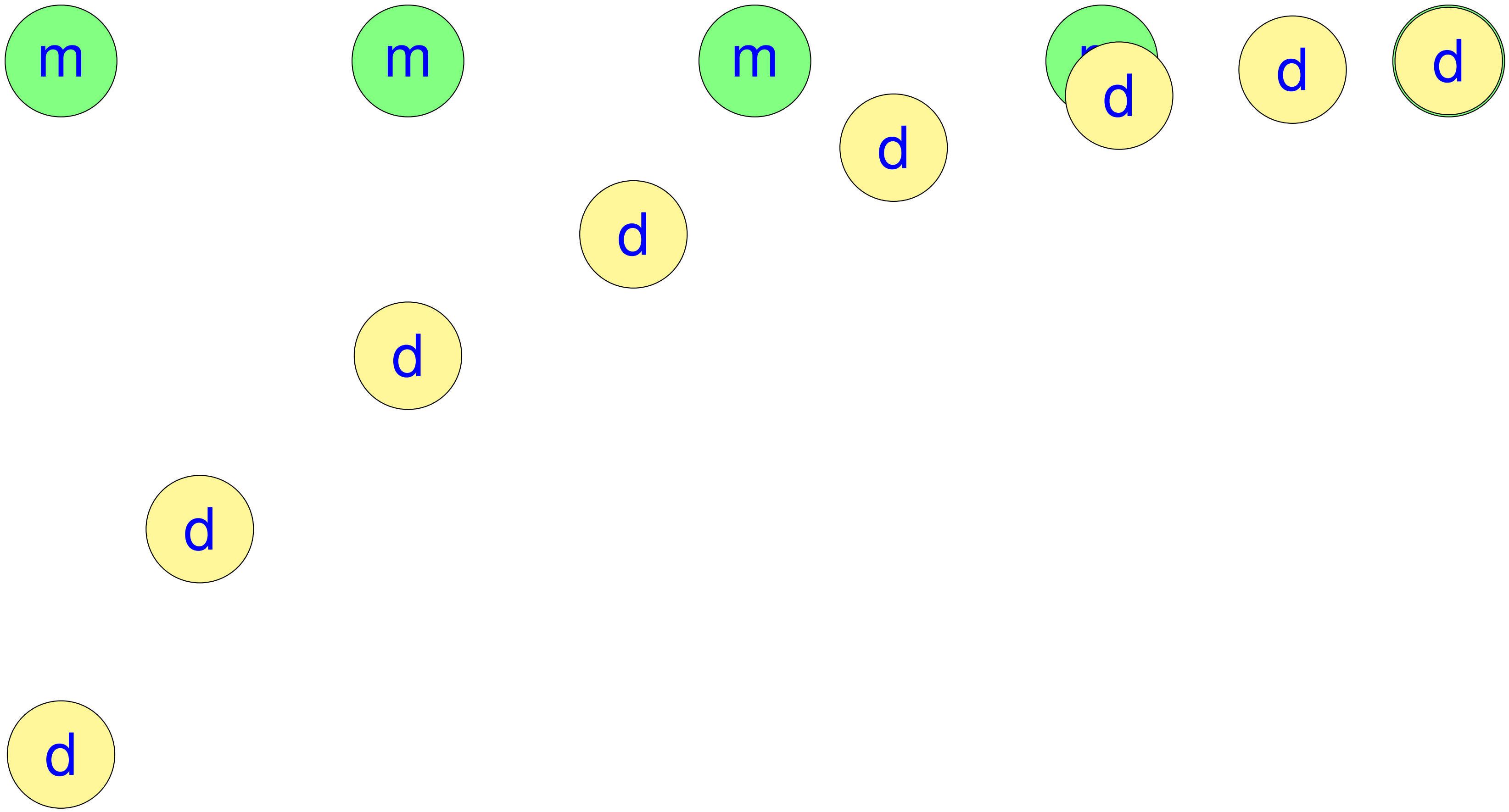
Master and Dog



Master and Dog



Master and Dog: The Dog Reaches the Master



- The Dog builds a local mental image of his Master's position.

- The Dog reaches the Master if he stops for a sufficiently long time.

Another Dog, Just for the Fun ;-)

I like it

Requirements

Roles of Requirements

- Requirements are very important

- Usually they are missing or poorly defined

- Requirements are informal statements

- They define the correctness criteria of a final implementation

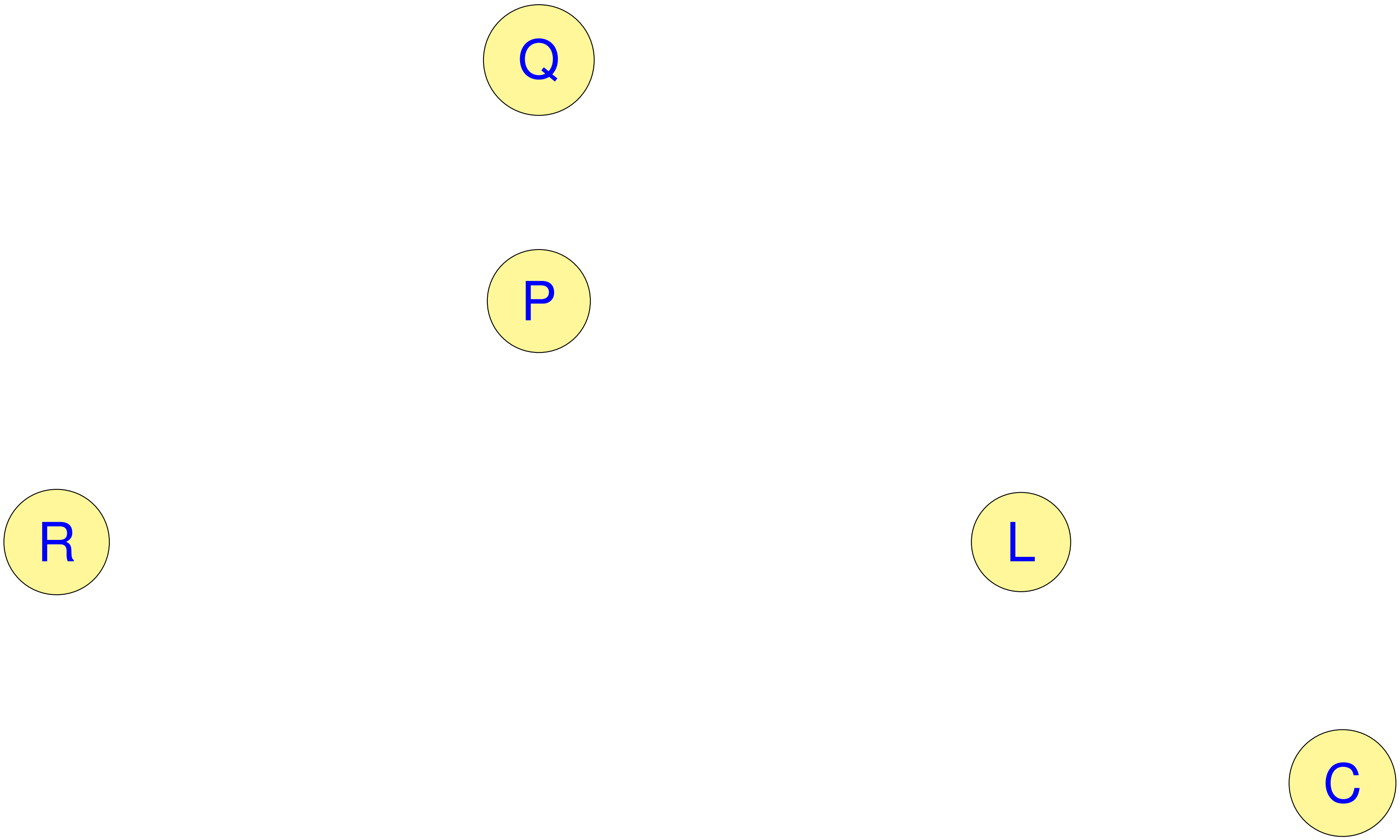
Requirement 1: the Master

- In our case, the master is a network

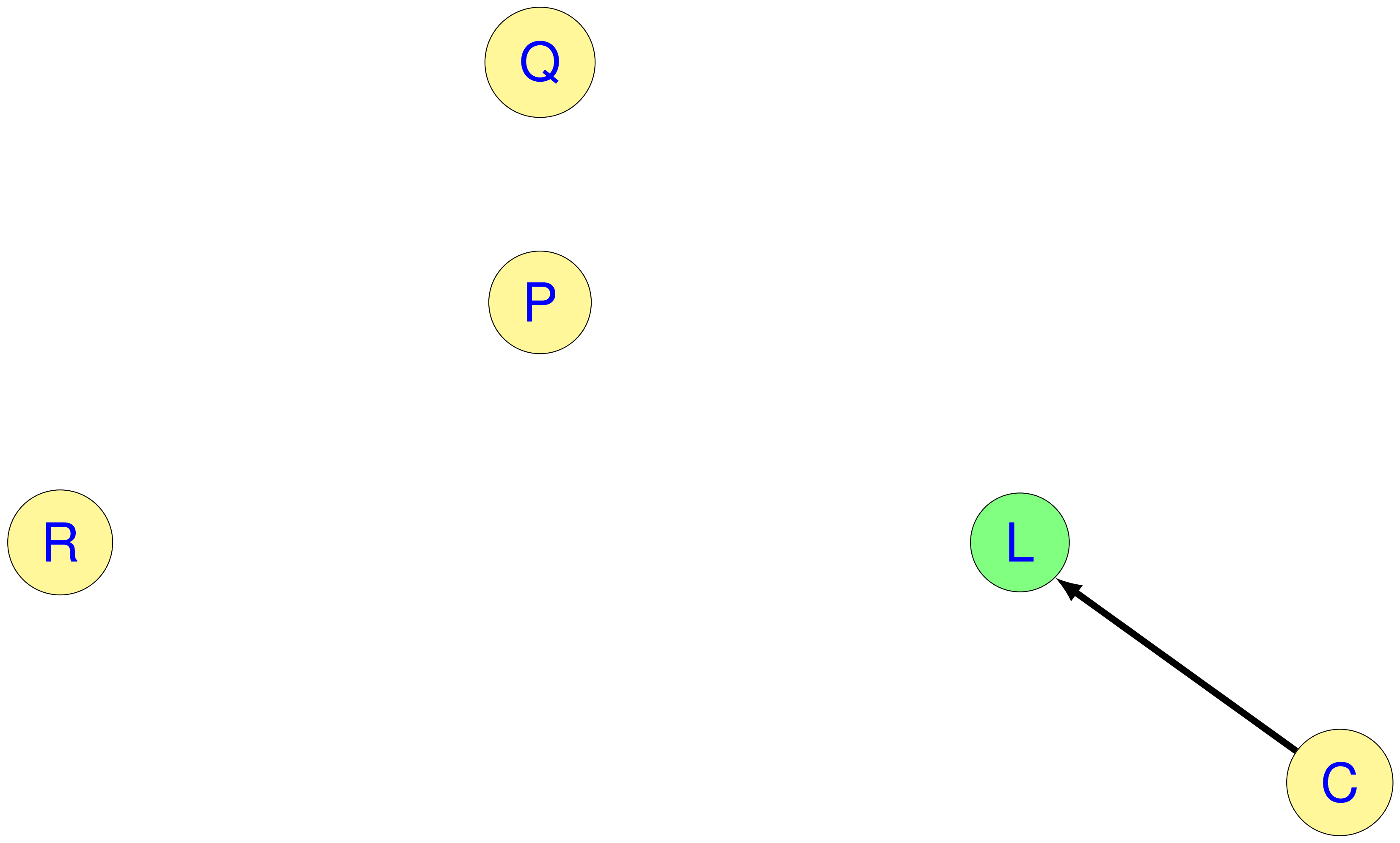
- This network may evolve as time goes

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

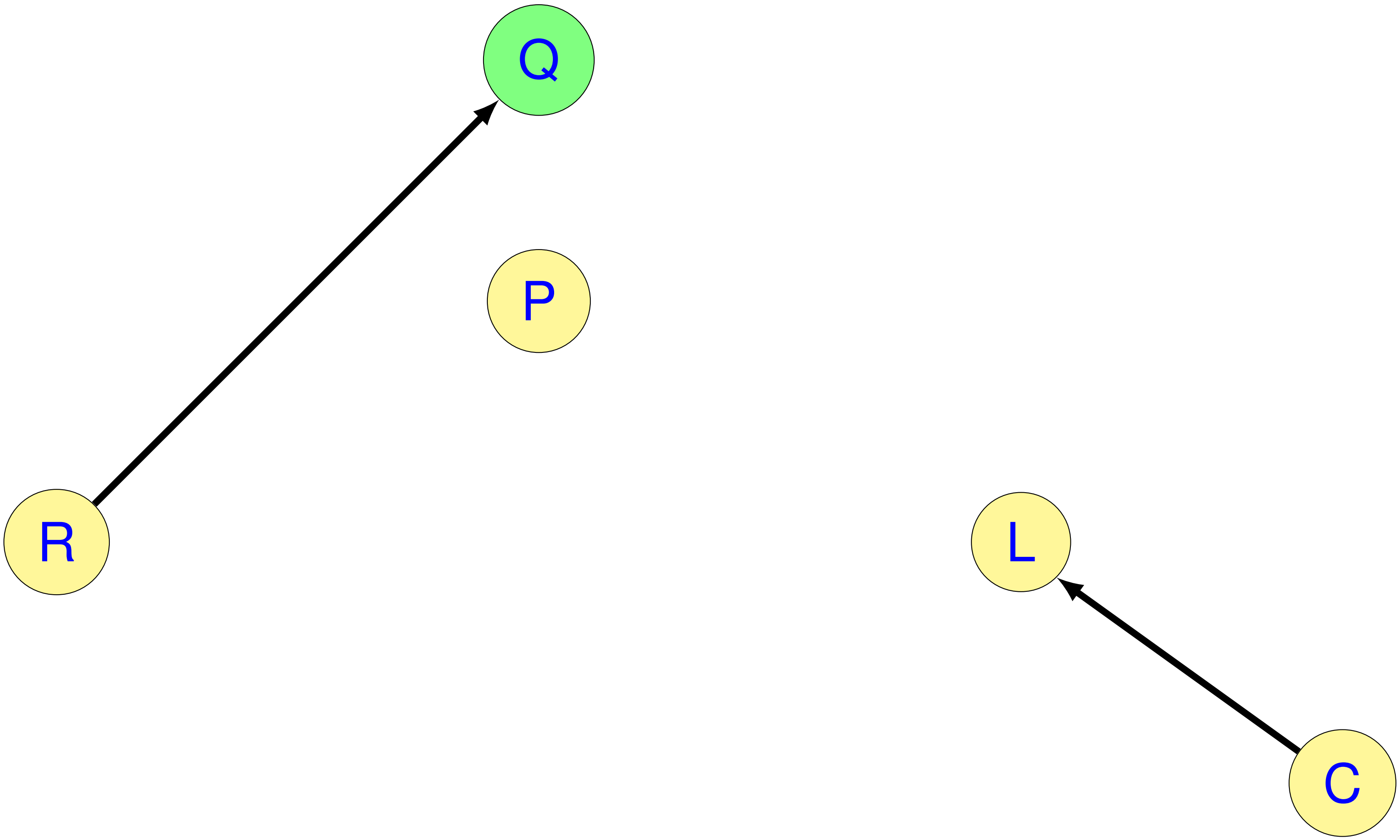
The Network



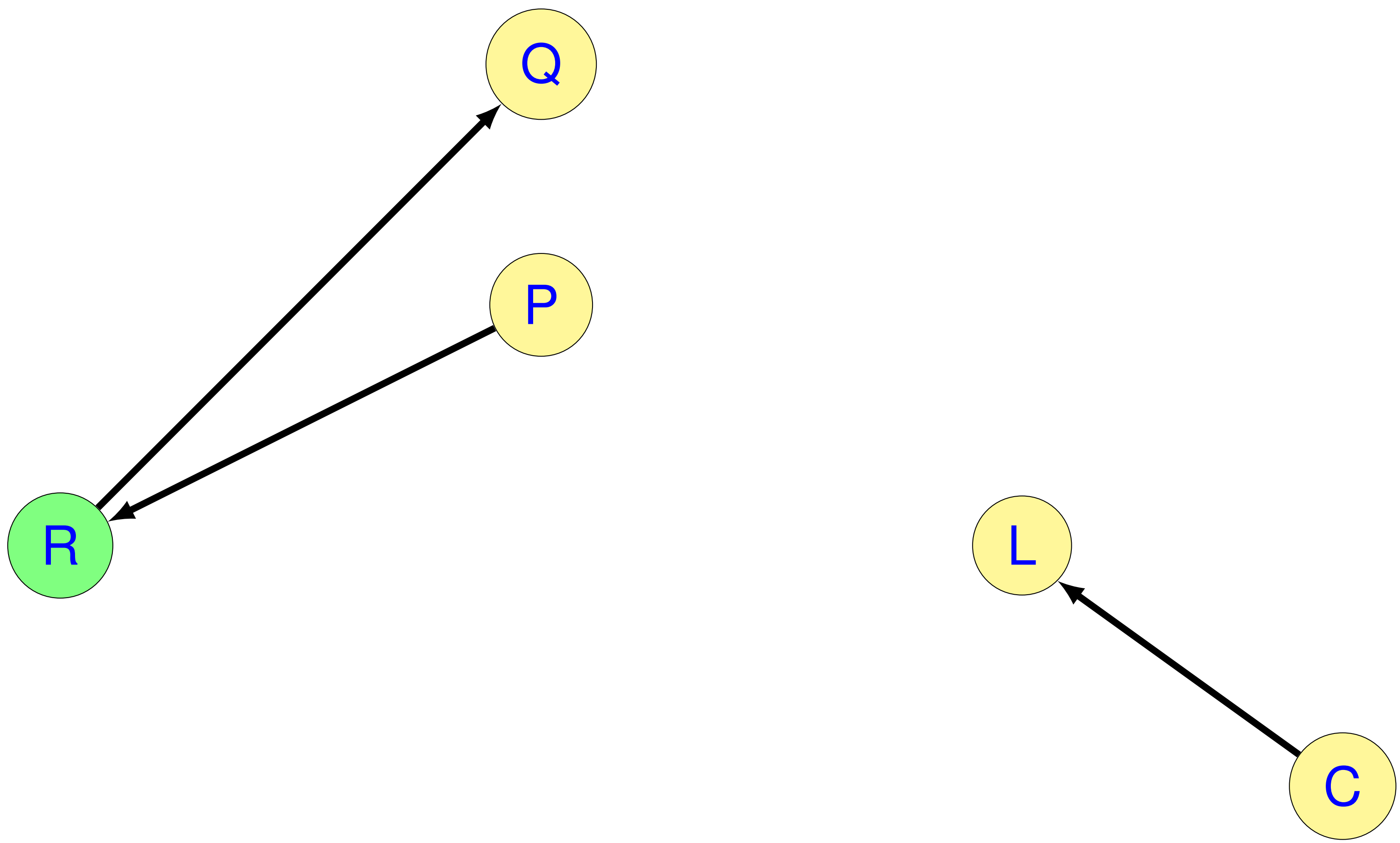
The Network



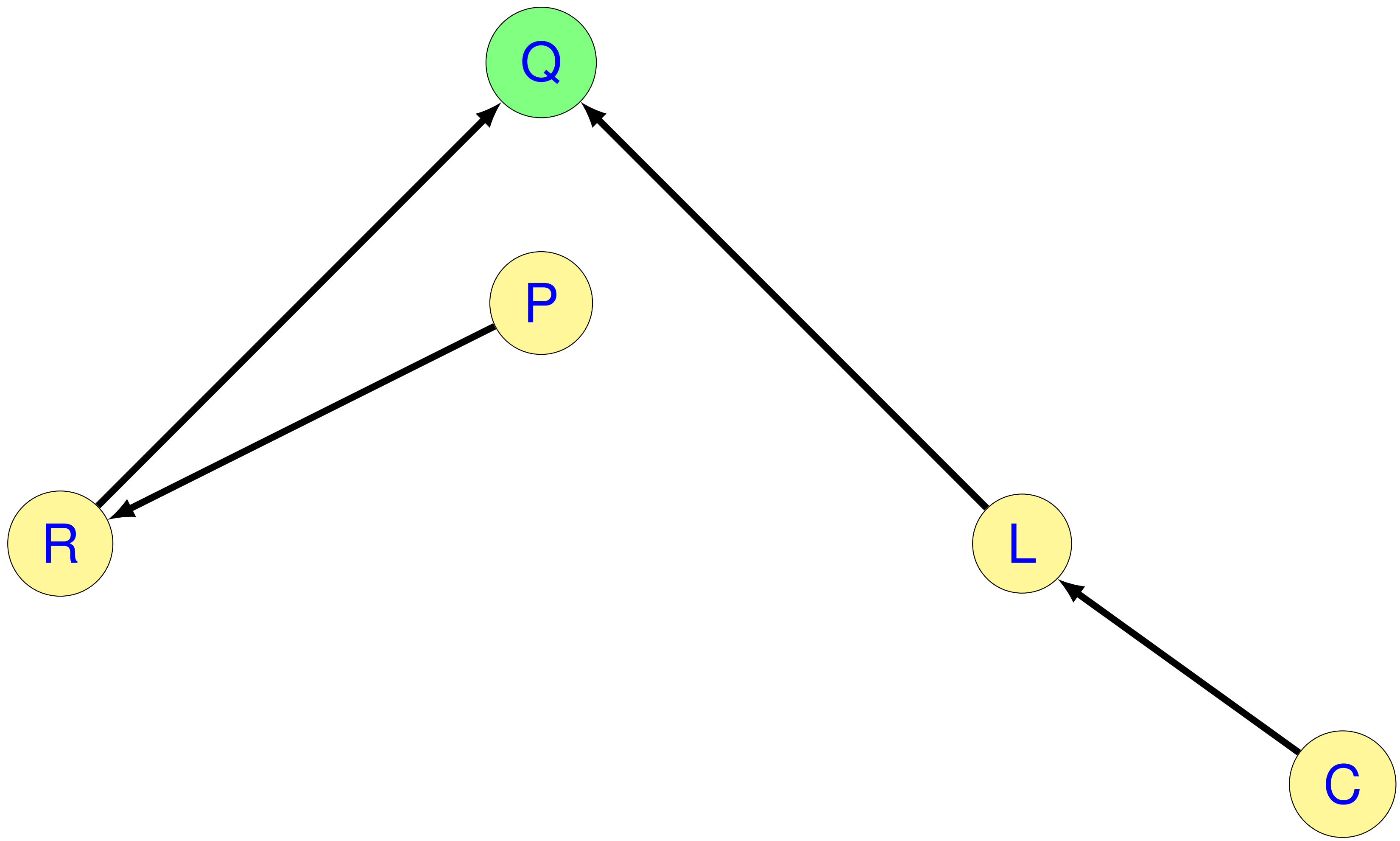
The Network

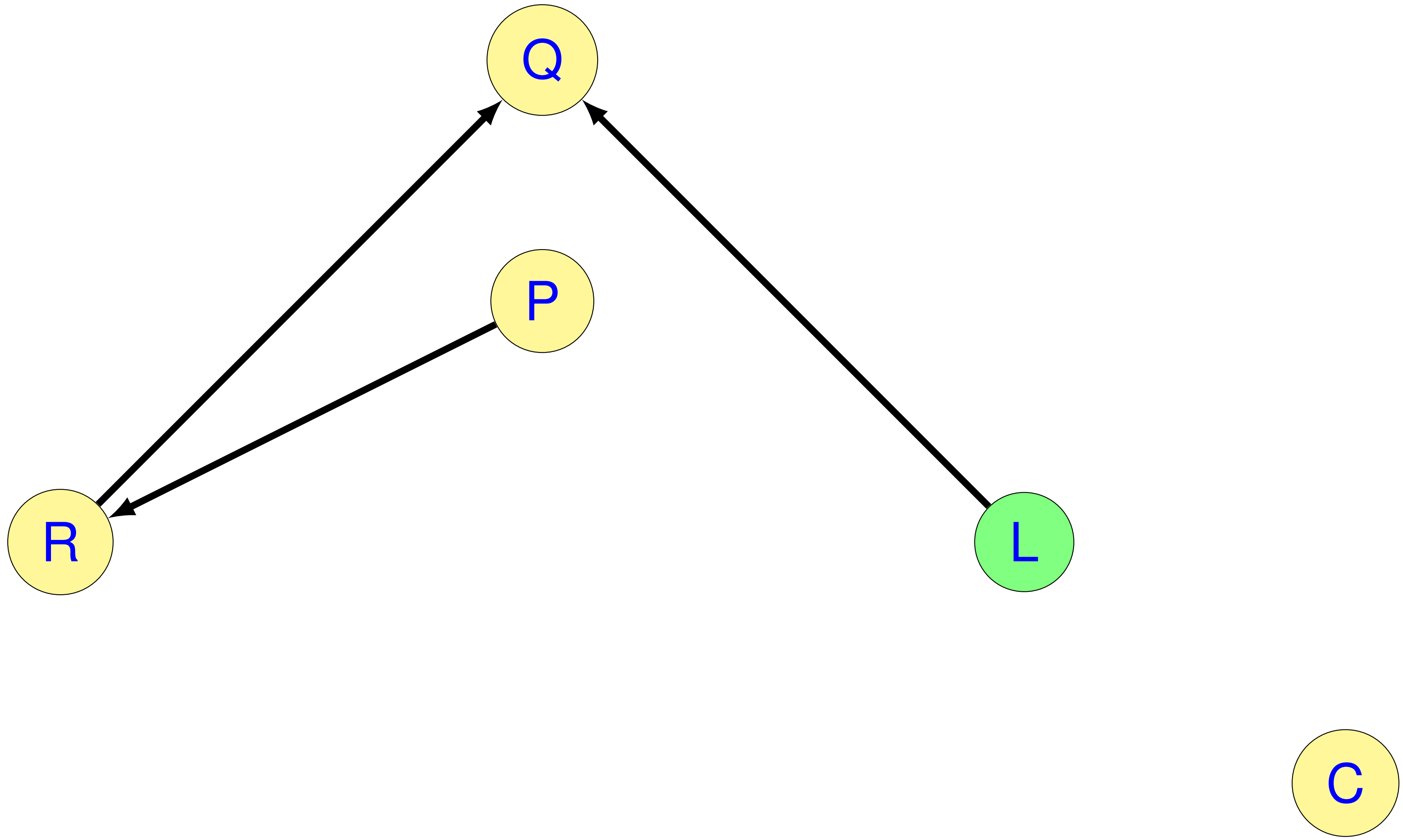


The Network

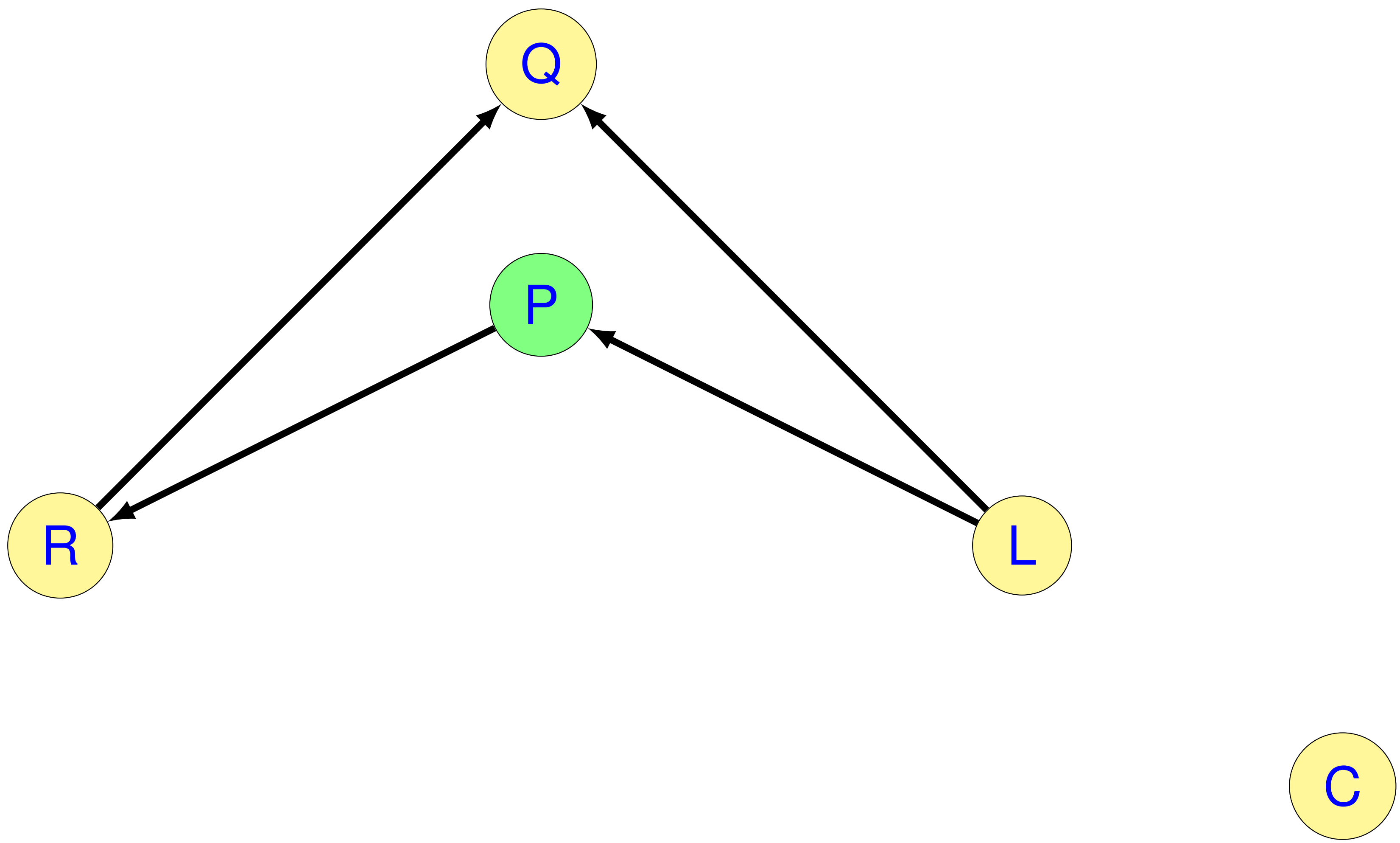


The Network

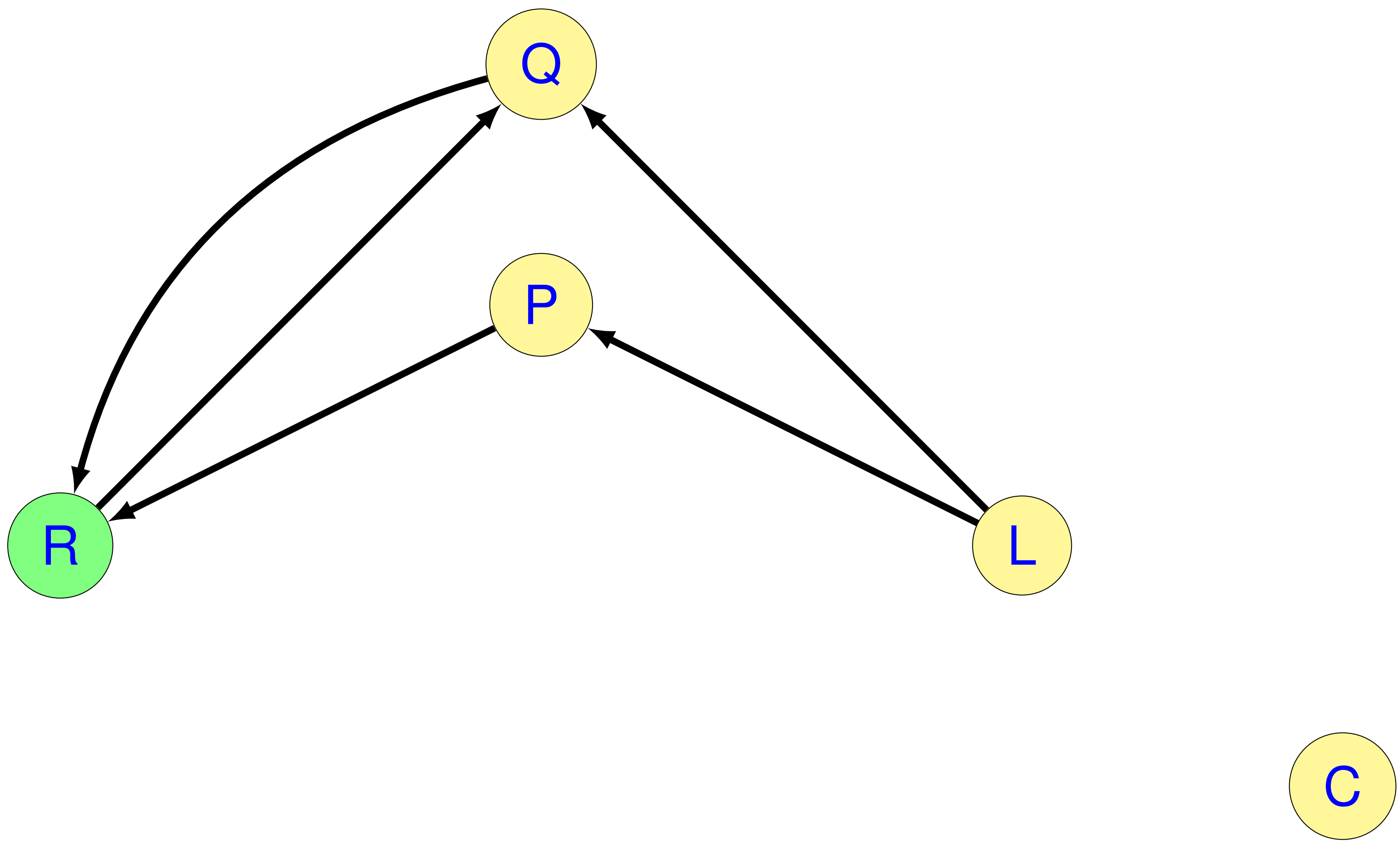



Q
P
R
L
C

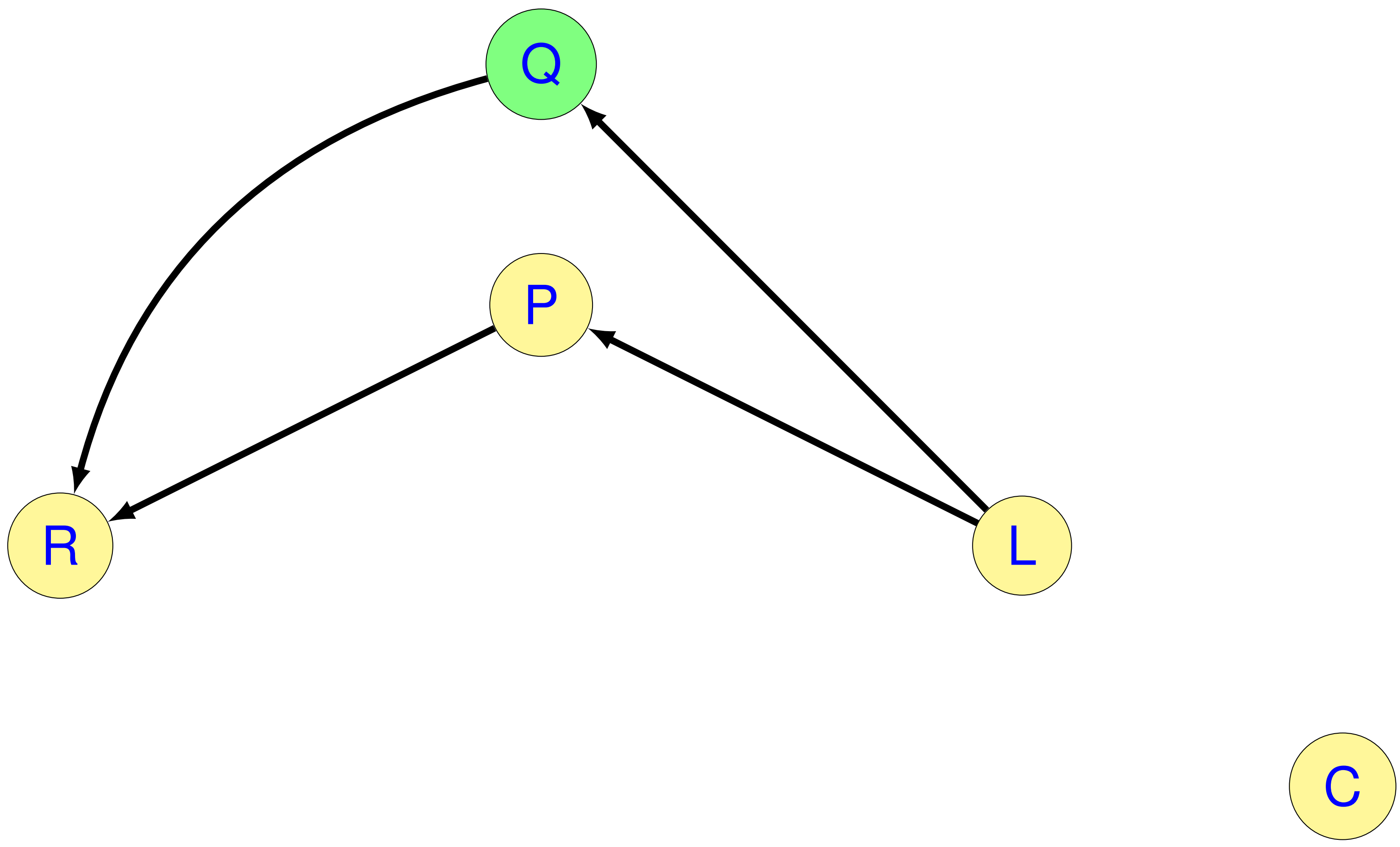
The Network

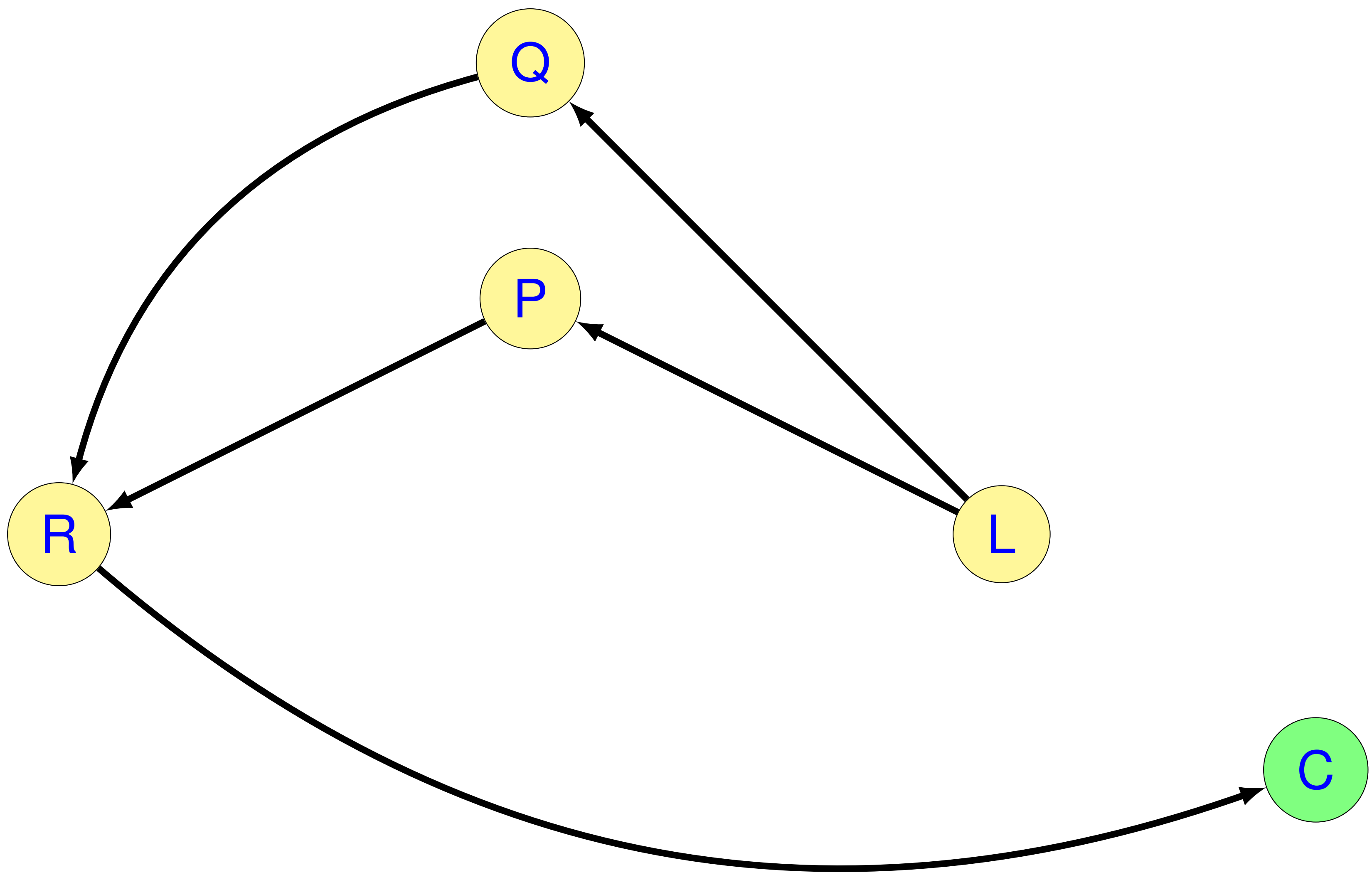


The Network

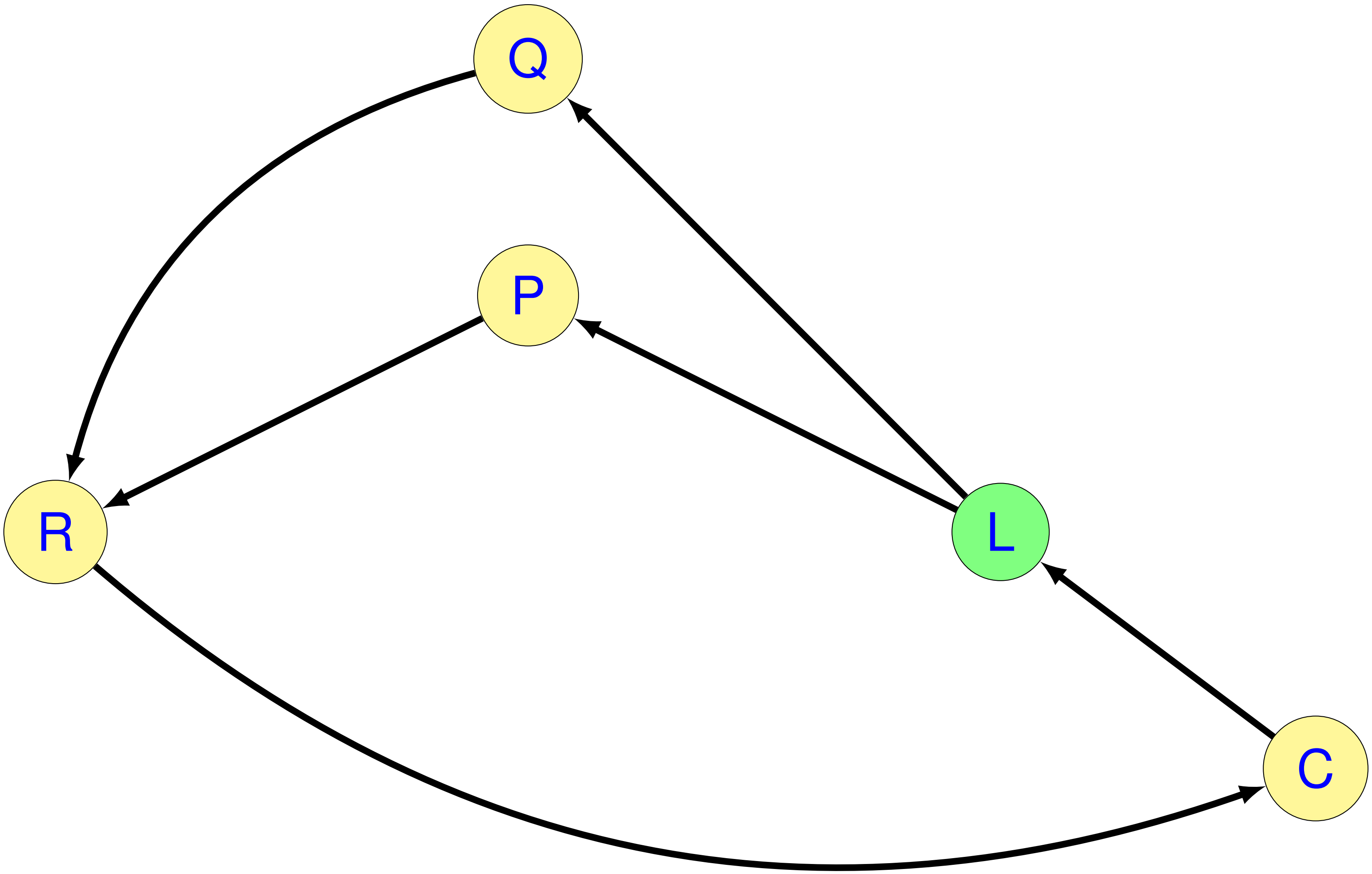


The Network




Q
P
R
L
C

The Network



Requirements 2: the Dogs

- In our case, we have several dogs.

- Each node in the network is a dog.

- Each dog follow the master by forming a local image of the network.

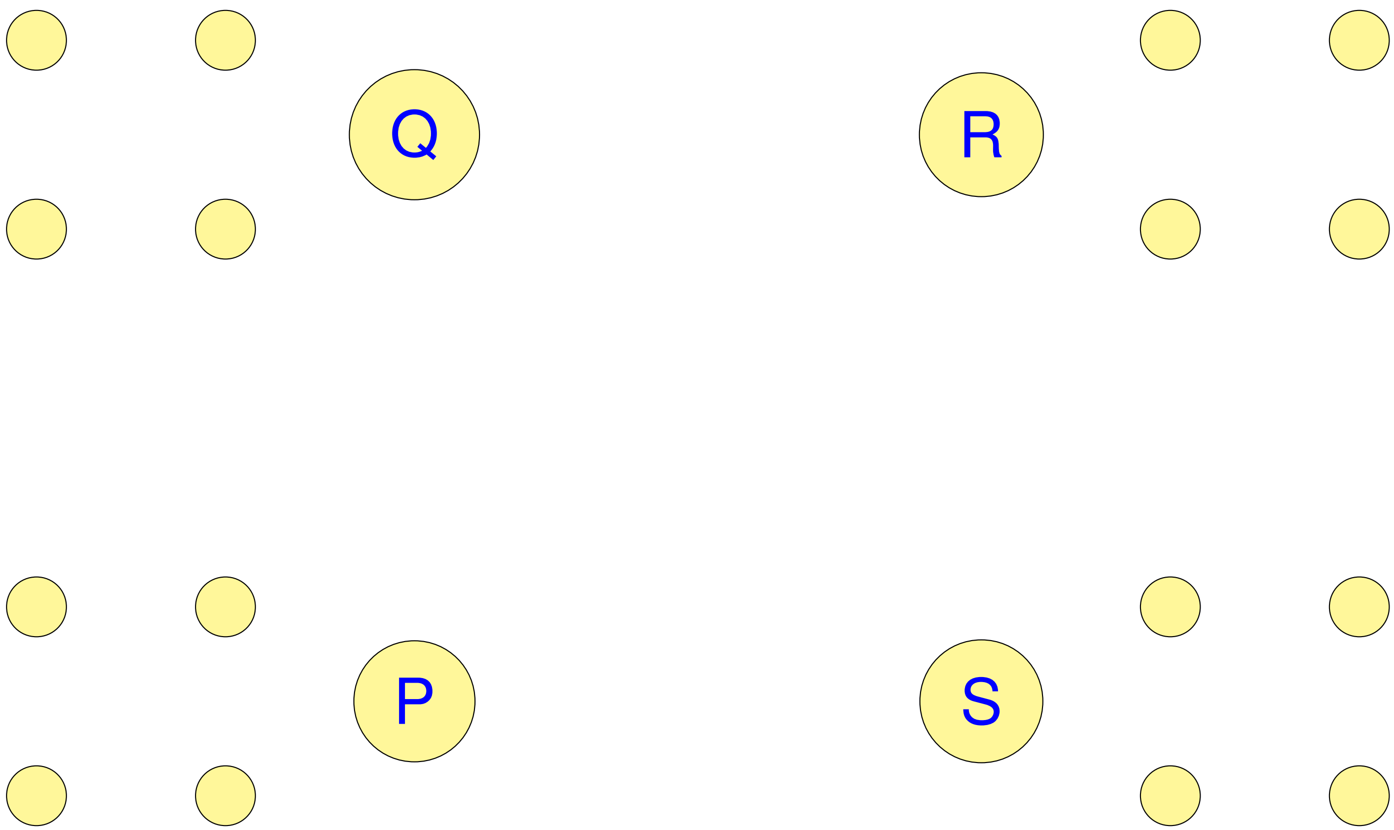
- A node builds the network image by two different approaches (next slide)

- A dog reaches the master when its local image is the same as the network.

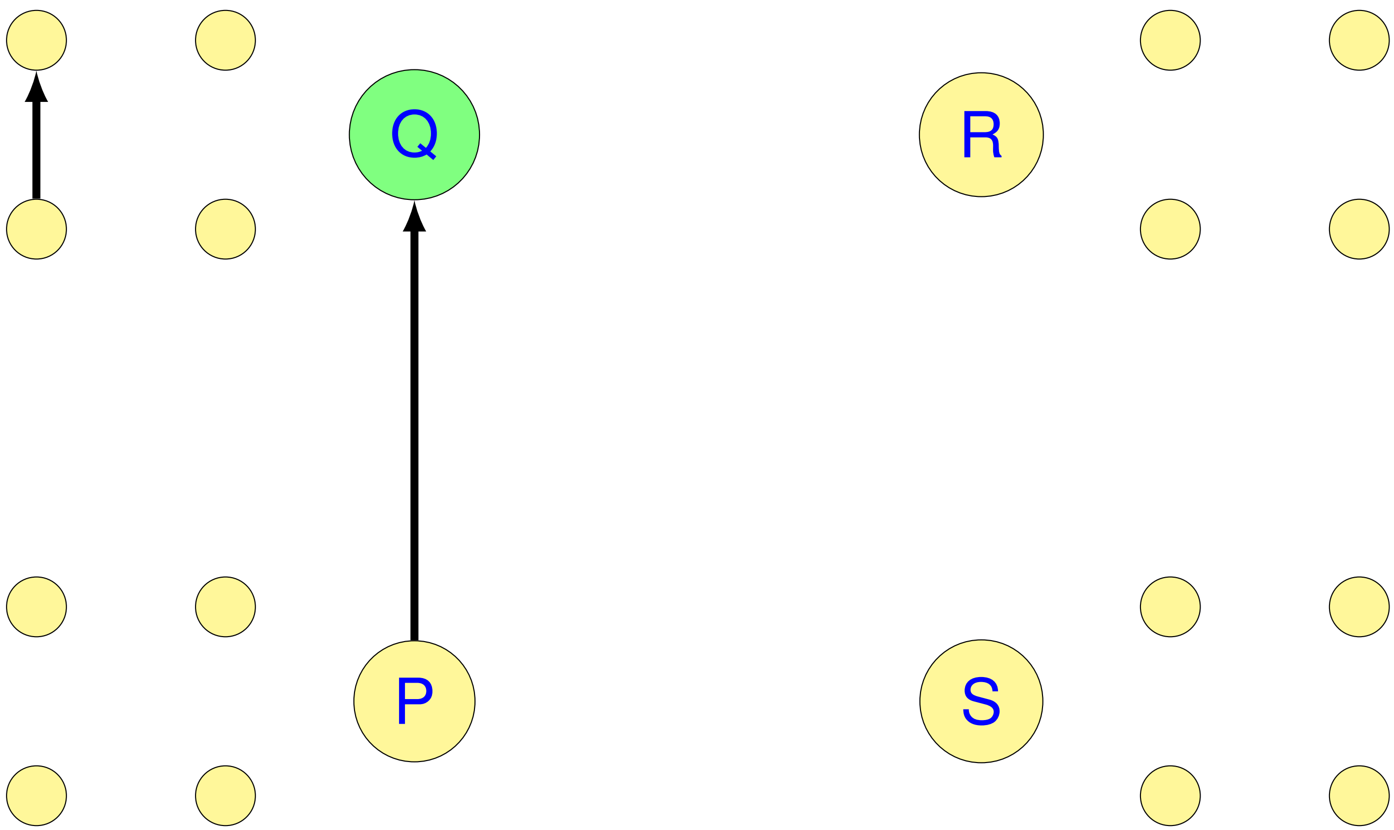
When a link from node $\boldsymbol{a}$ to node $\boldsymbol{b}$ is added to or removed from the network then node $\boldsymbol{b}$ is DIRECTLY made aware of it	REQ-2

The neighbors of a node $\boldsymbol{b}$ are the nodes that are connected to $\boldsymbol{b}$ by a link $\boldsymbol{l}$ entering in $\boldsymbol{b}$.	REQ-3

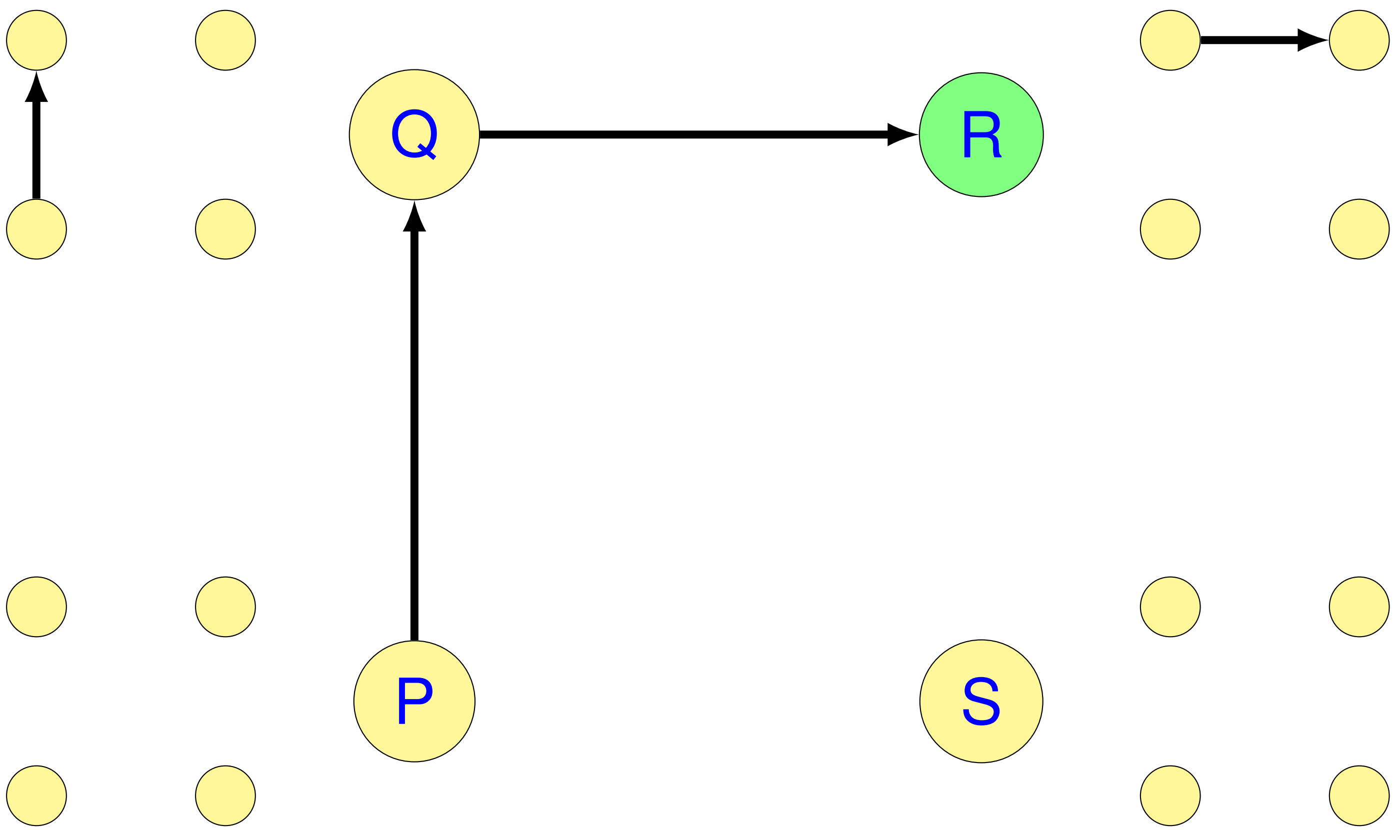
Neighbors of $\boldsymbol{b}$ send to $\boldsymbol{b}$ their local networks with MESSAGES sent to it through the connecting link	REQ-4


Q
R
P
S

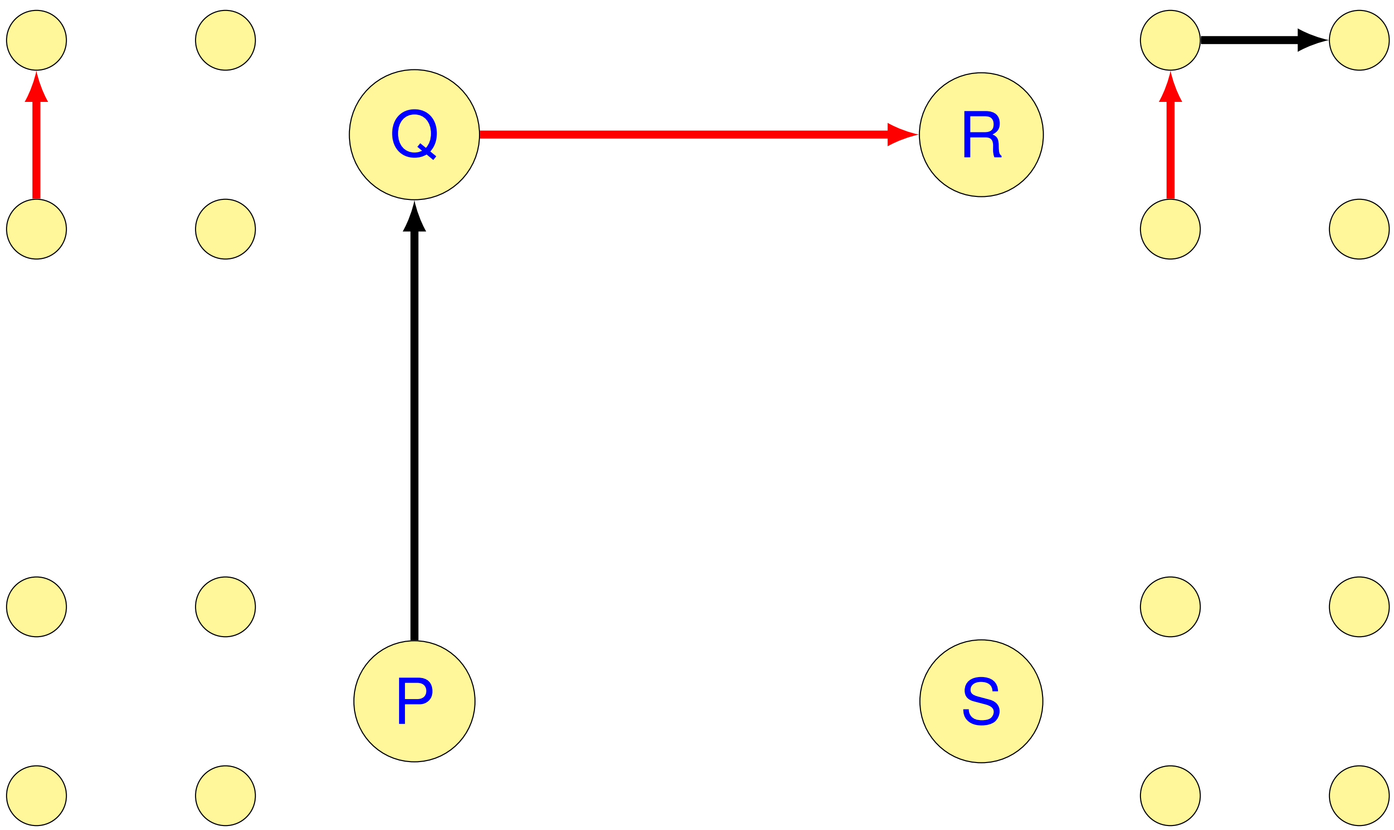
The Network: Node Q is Made Aware of New Connection PQ



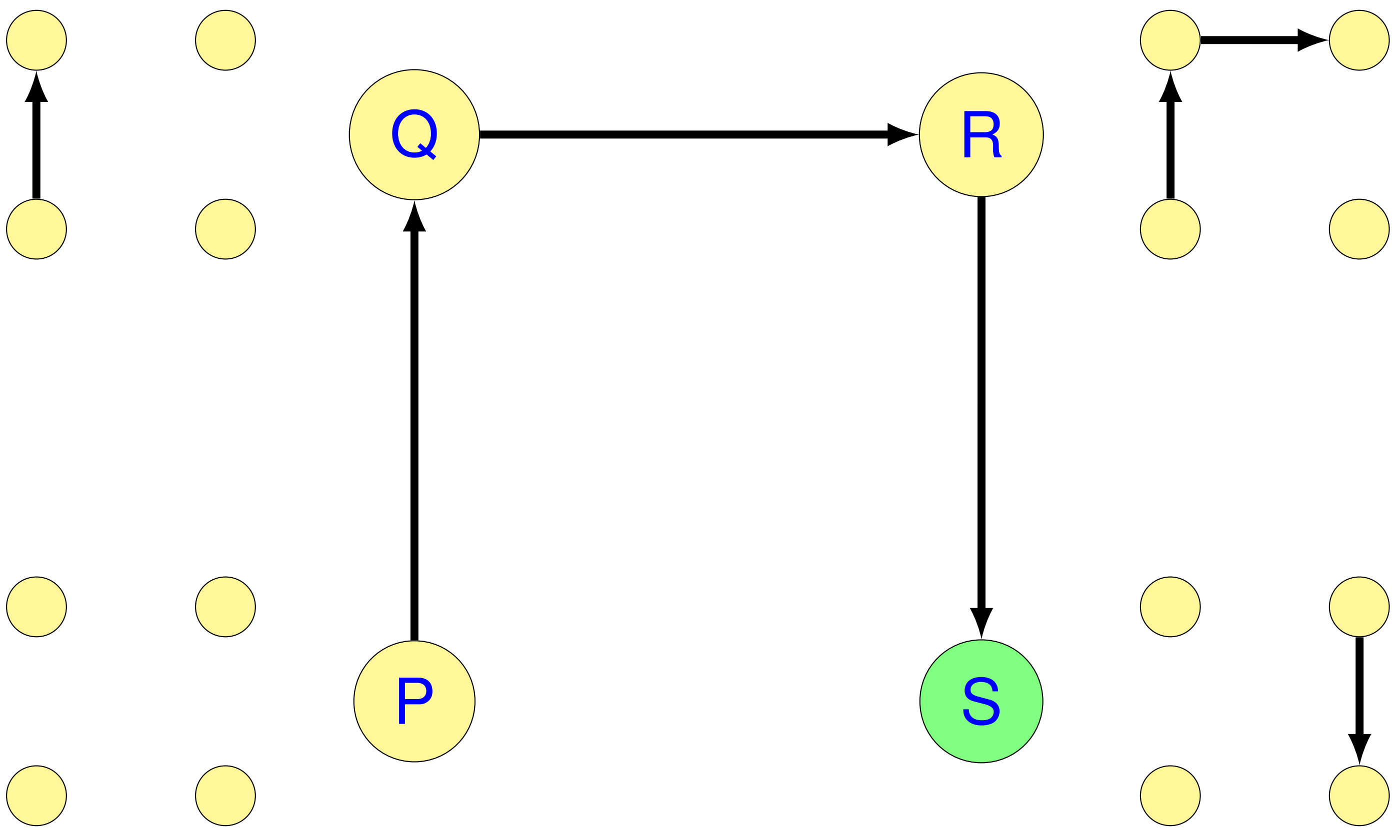
The Network: Node R is Made Aware of New Connection QR



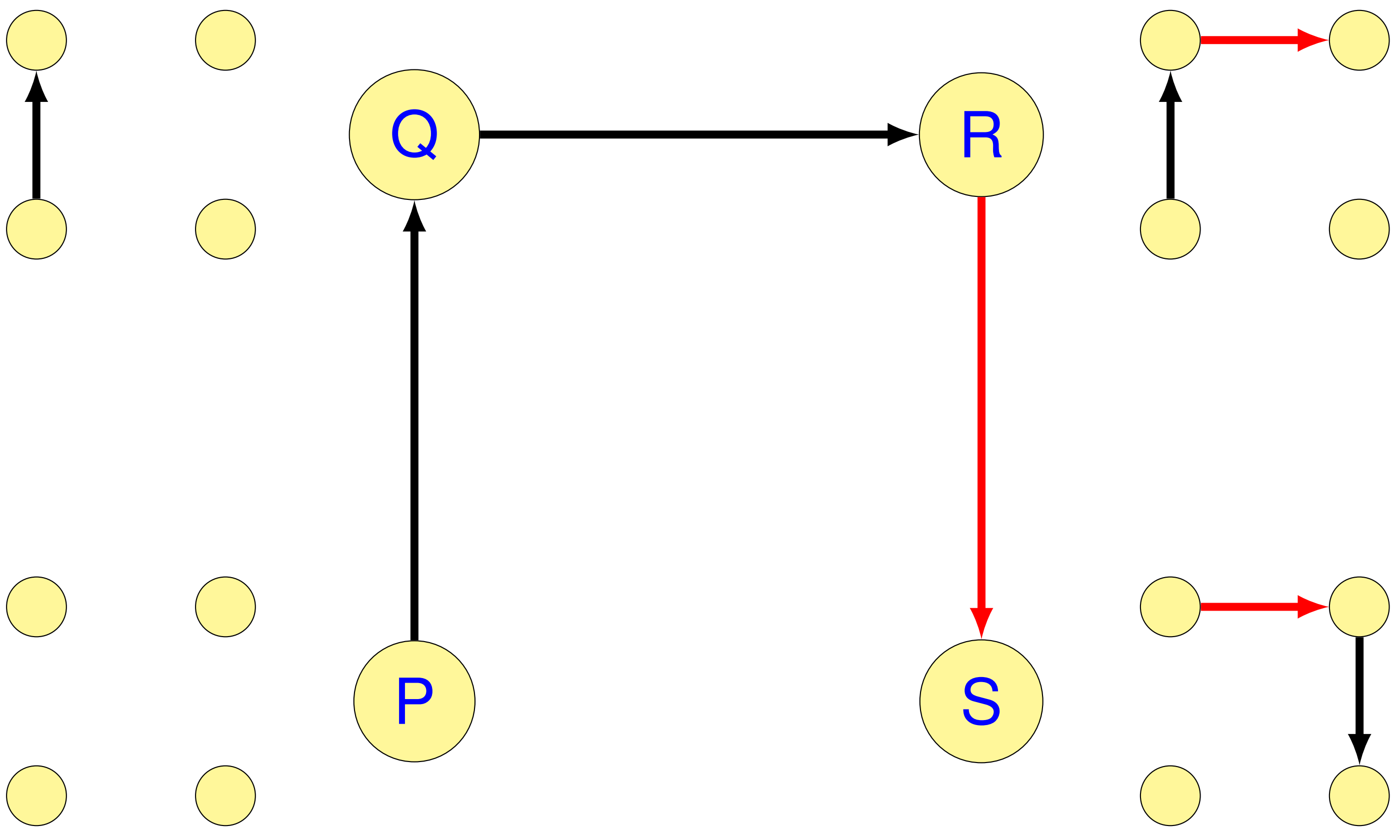
The Network: Local Connection PQ in Q is sent from Q to R

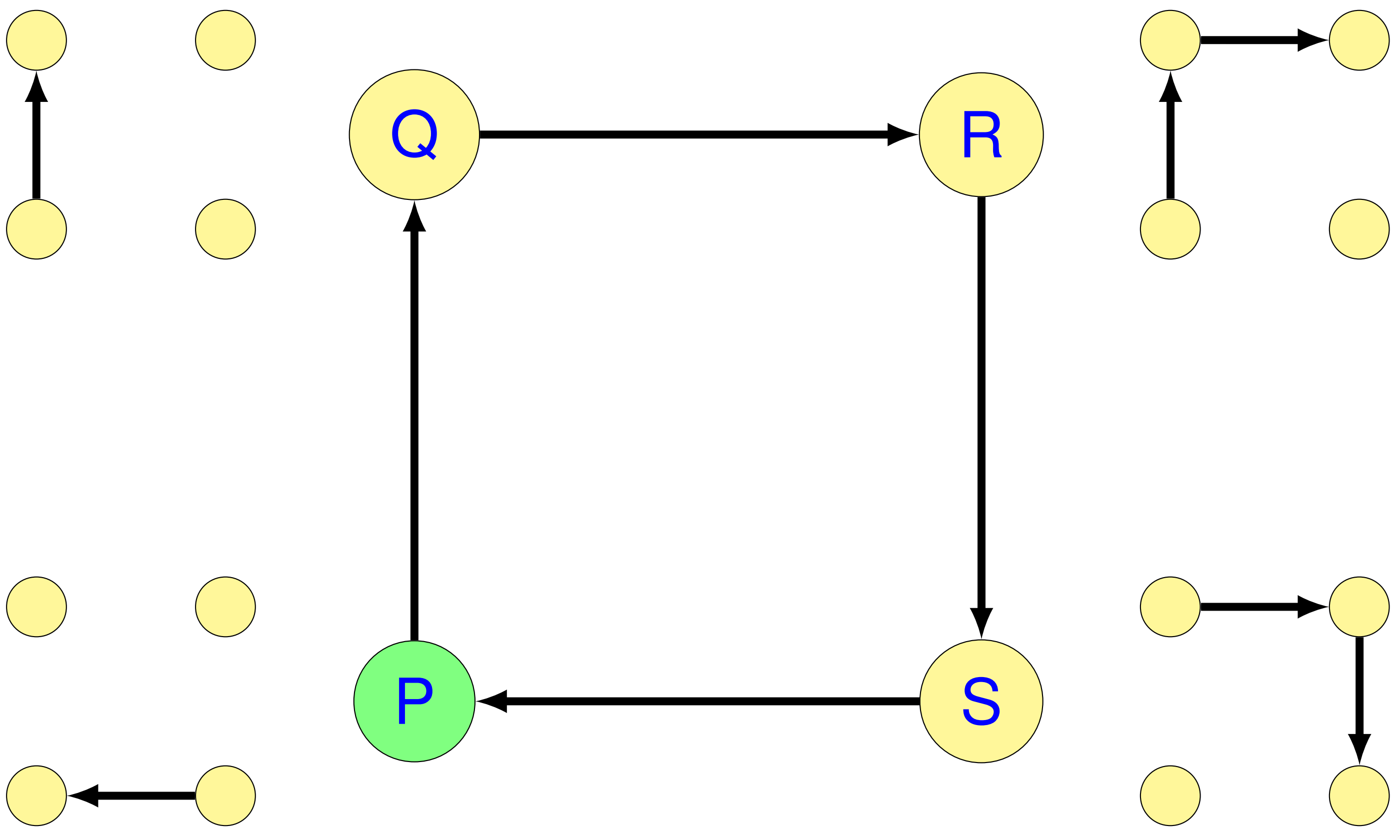


The Network: Node S is Made Aware of New Connection RS

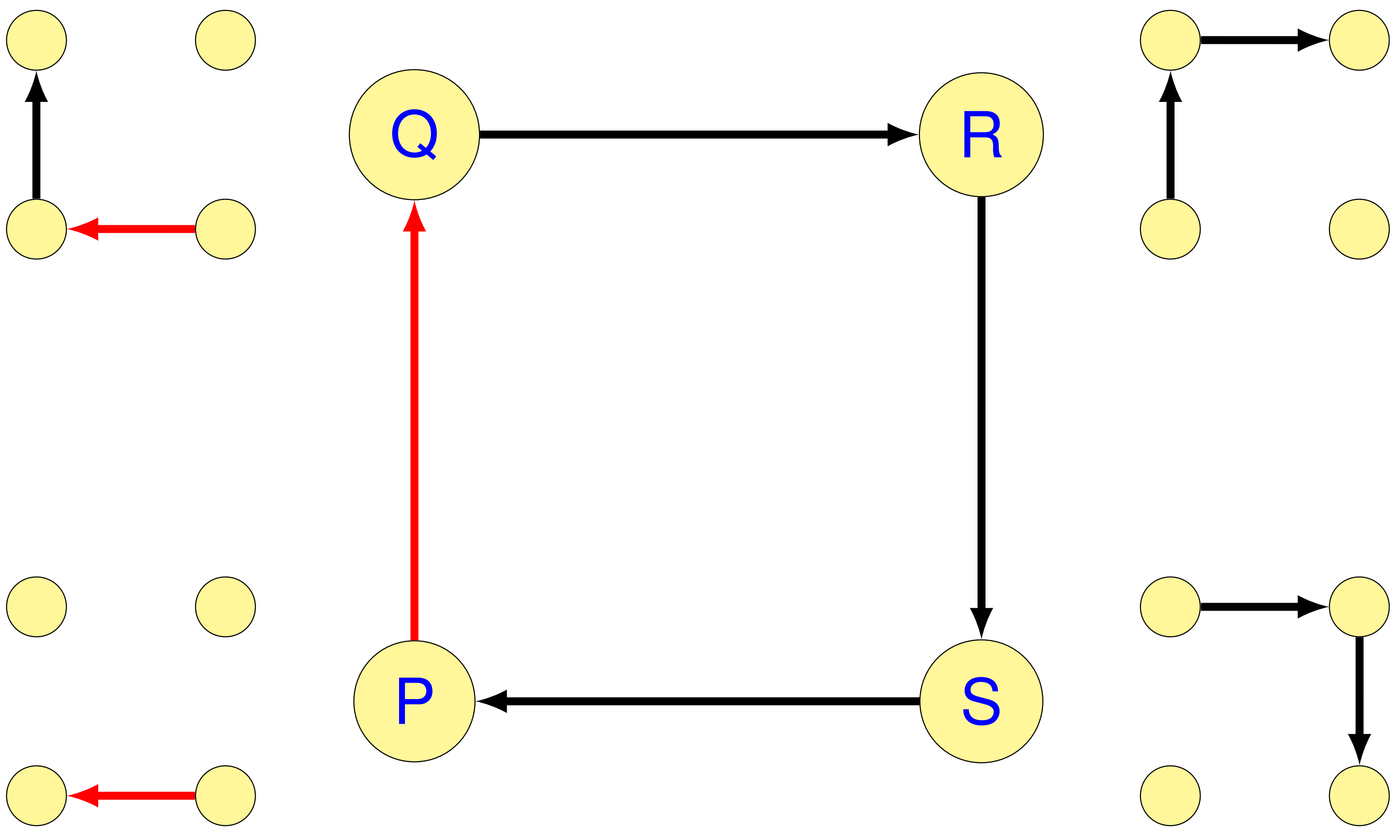


The Network: Local Connection QR in R is sent from R to S

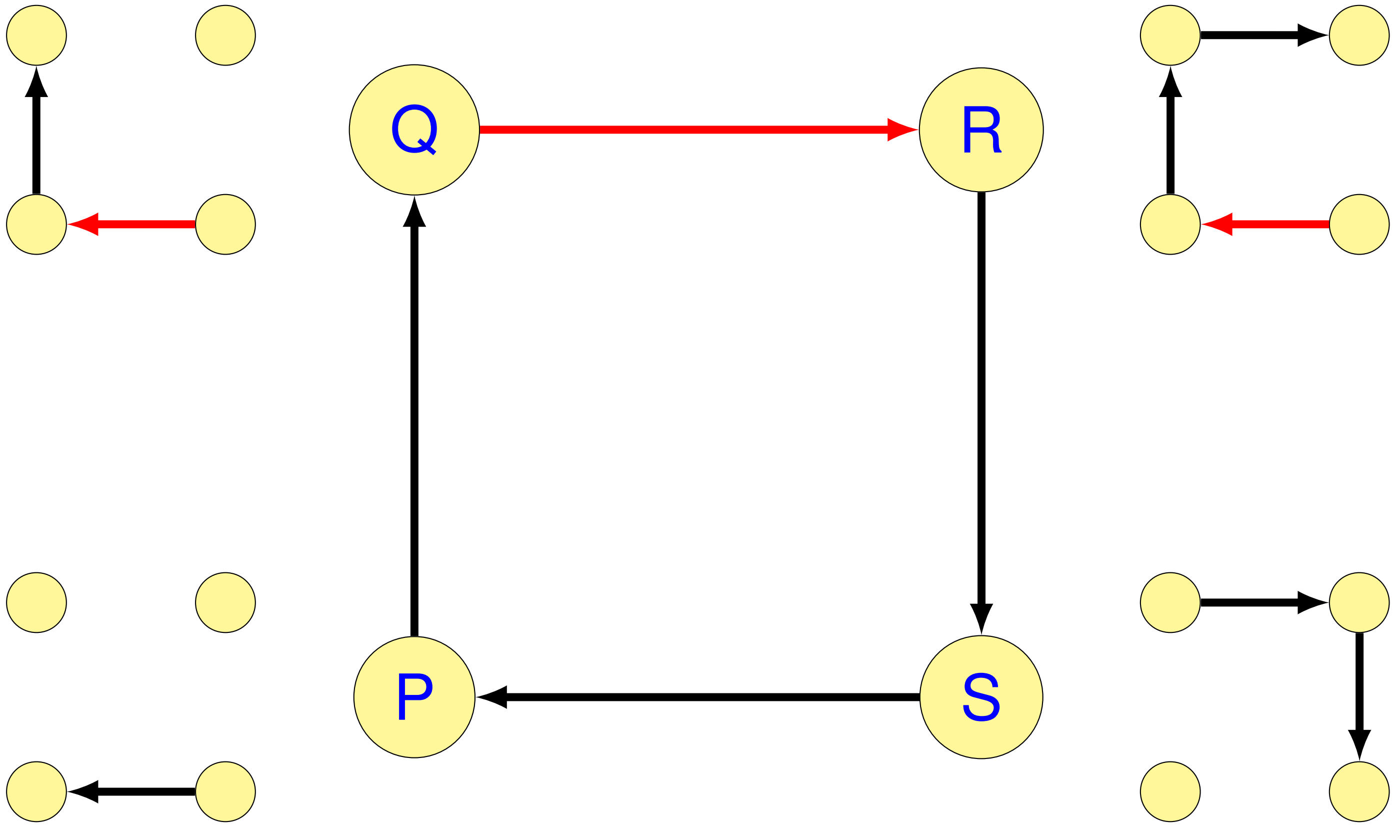


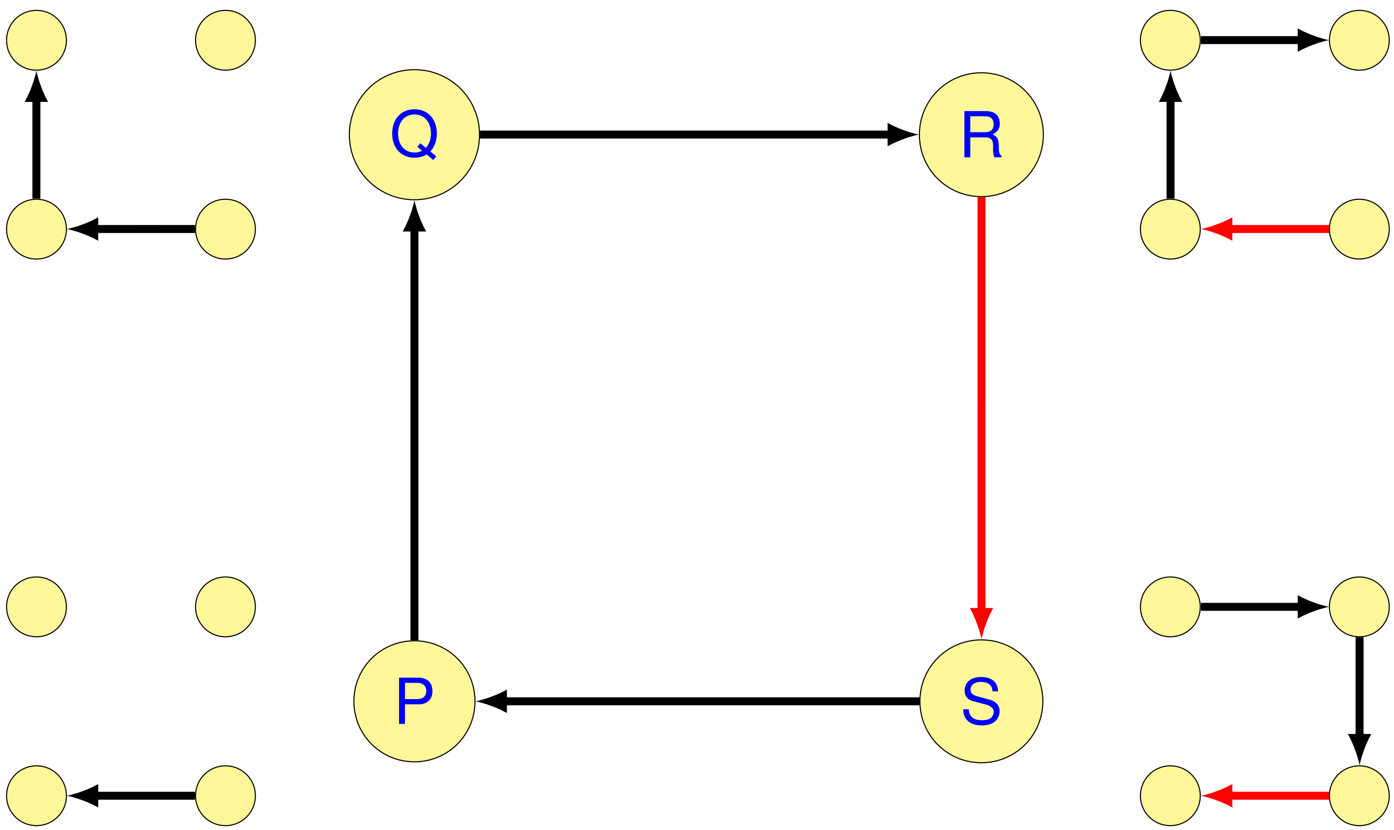

Q
R
P
S

The Network



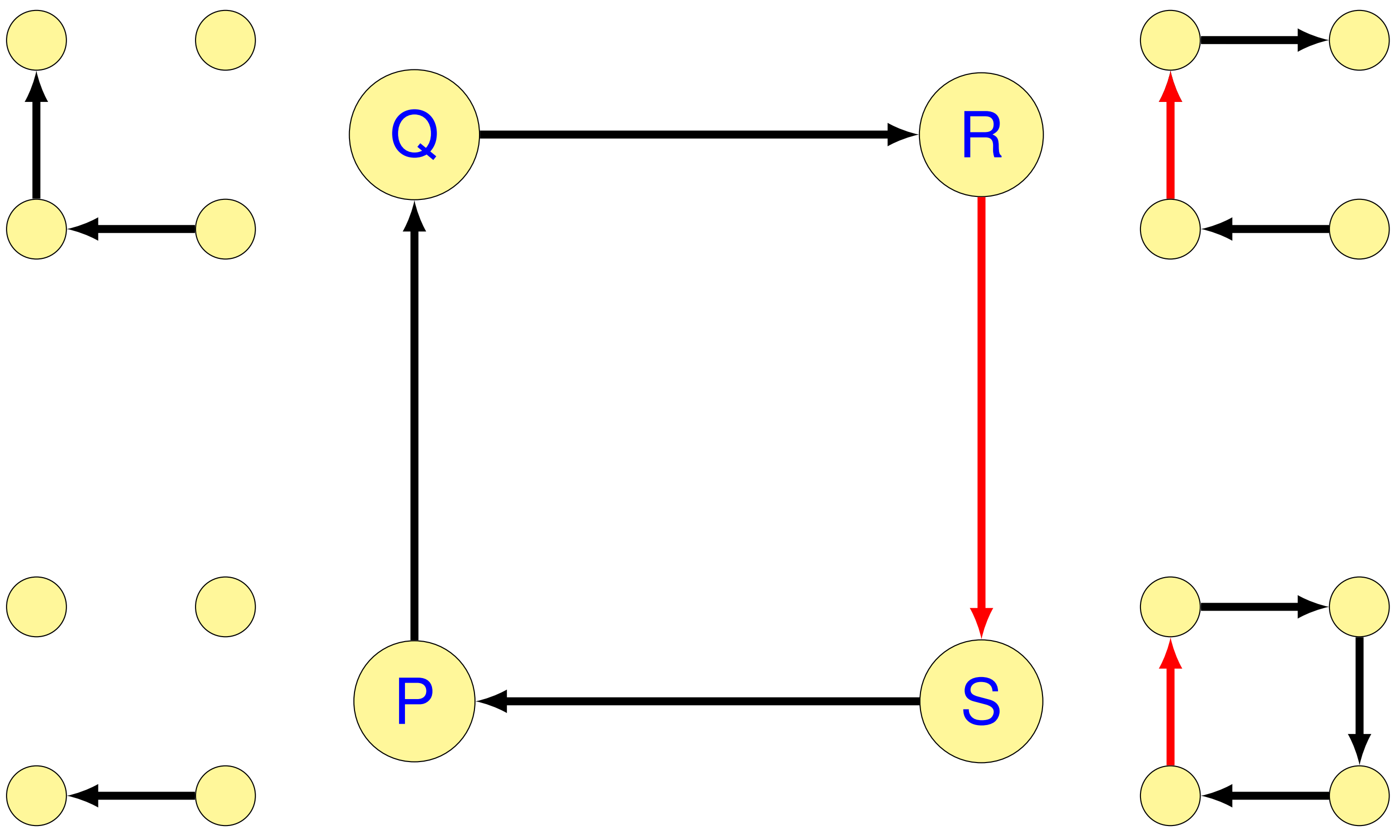
The Network



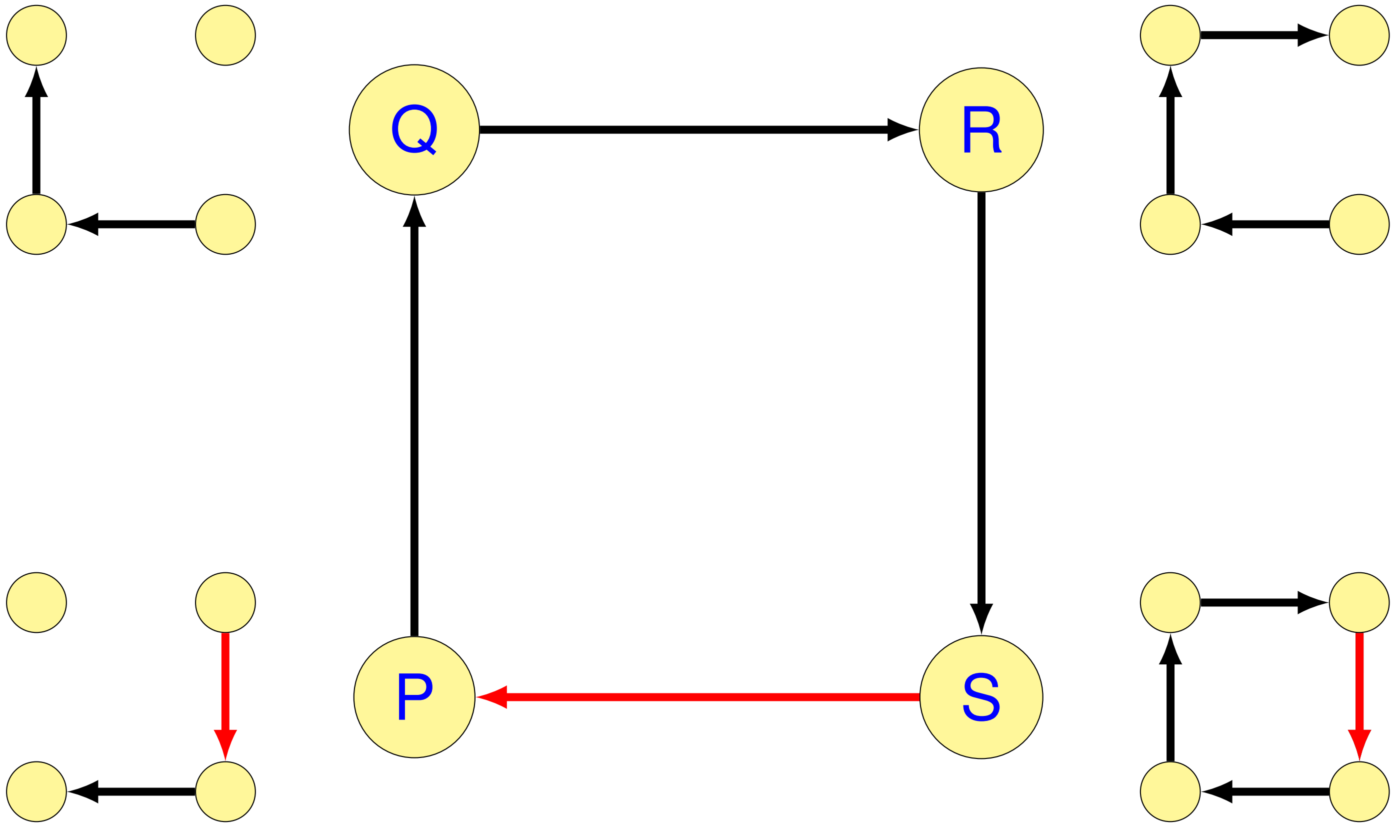


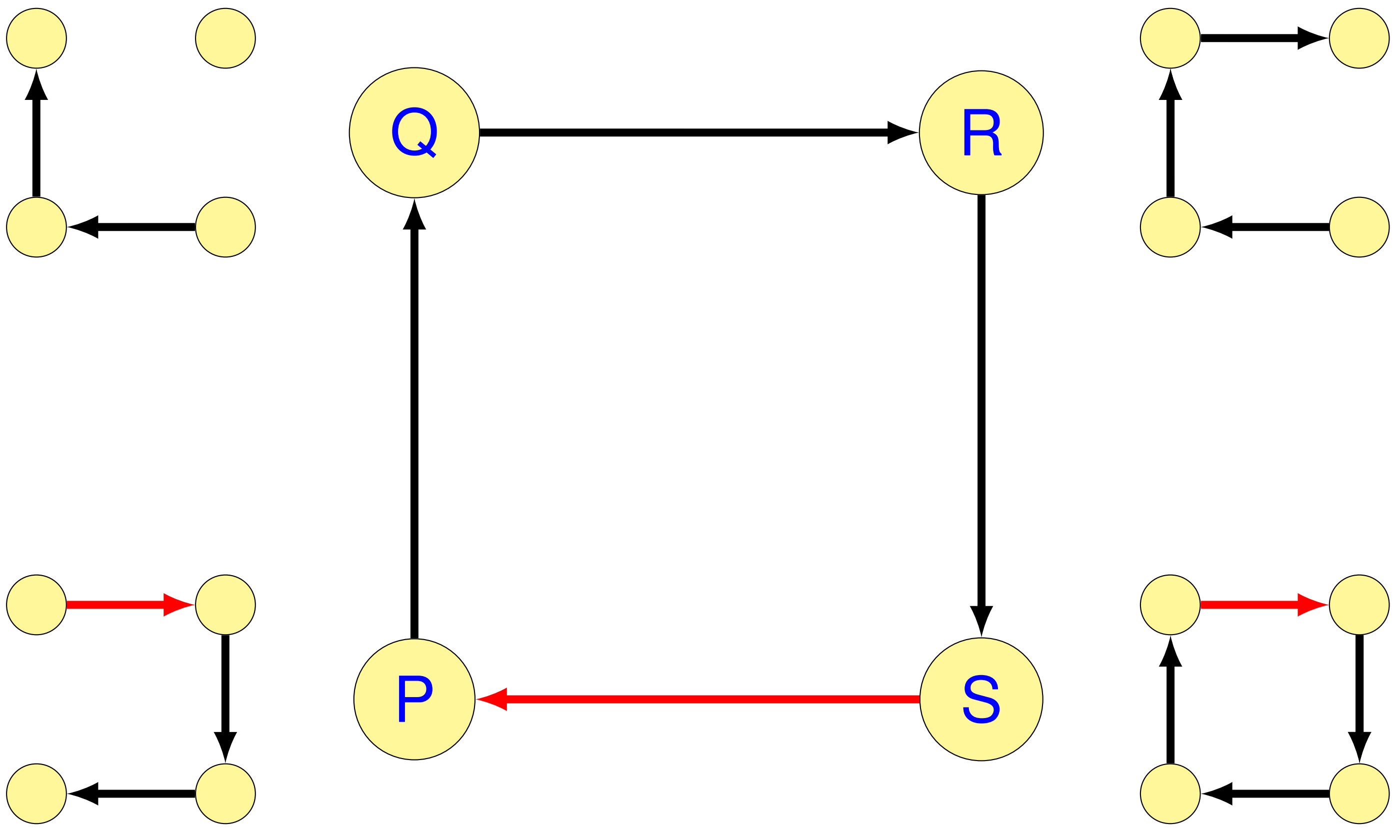
Q
R
P
S

The Network

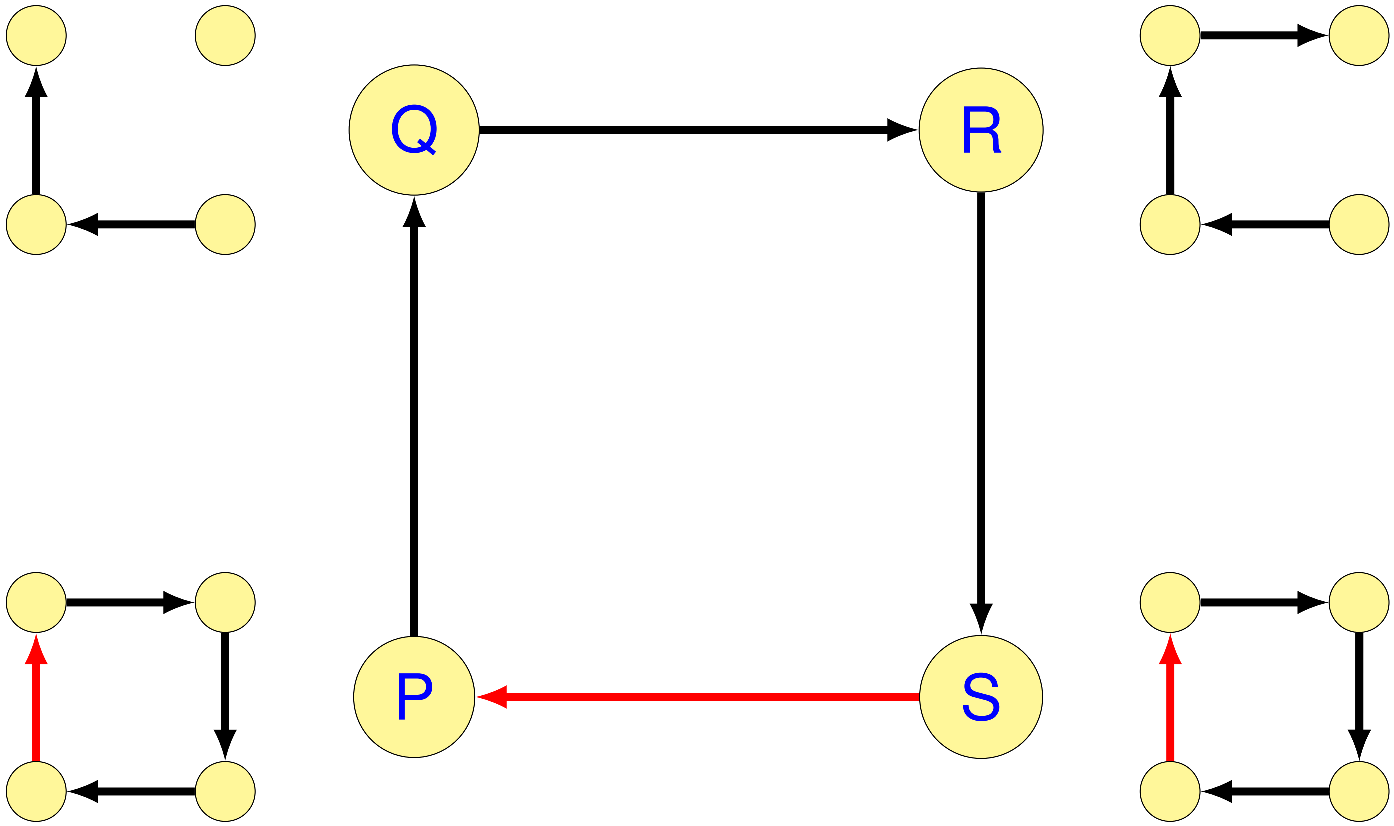


The Network

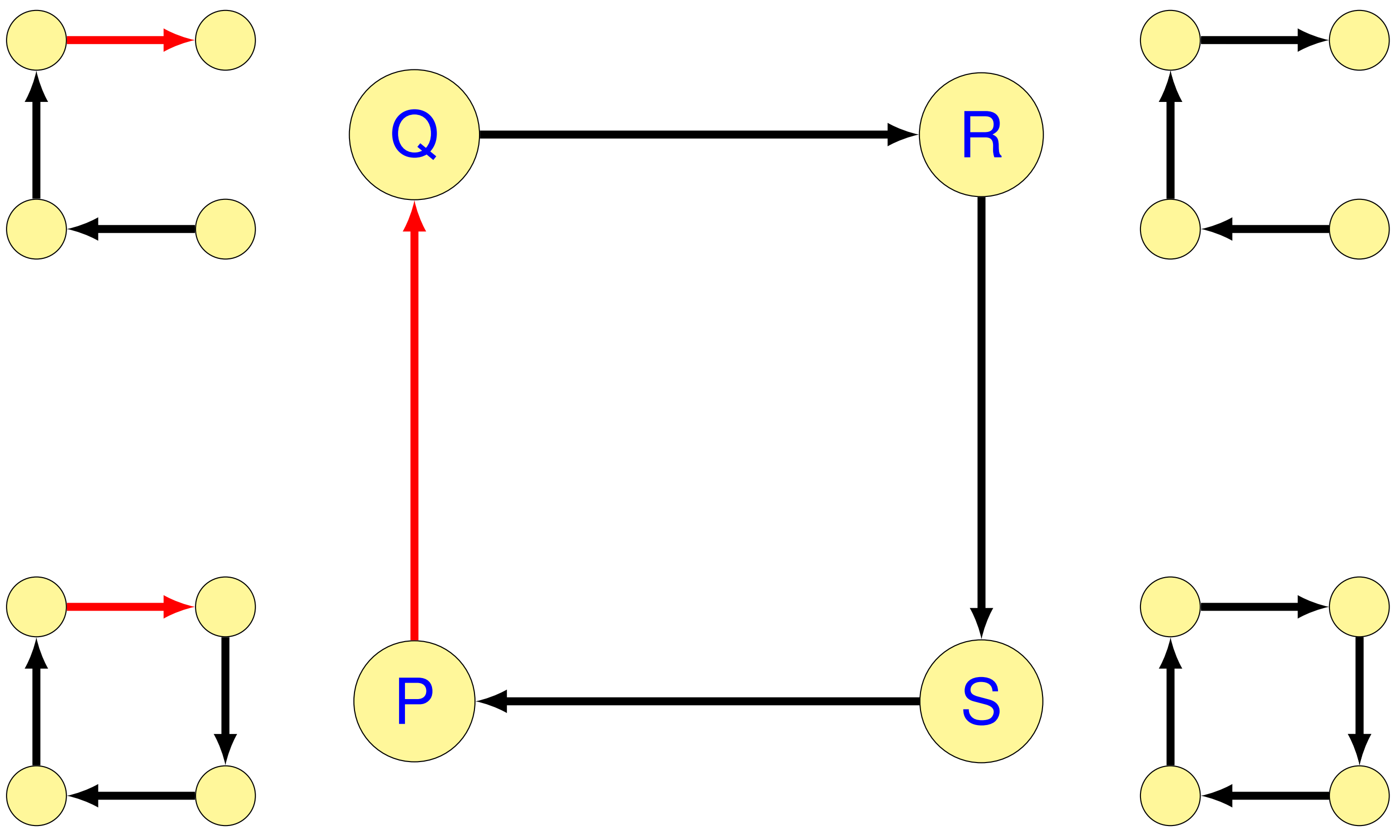



Q
R
P
S

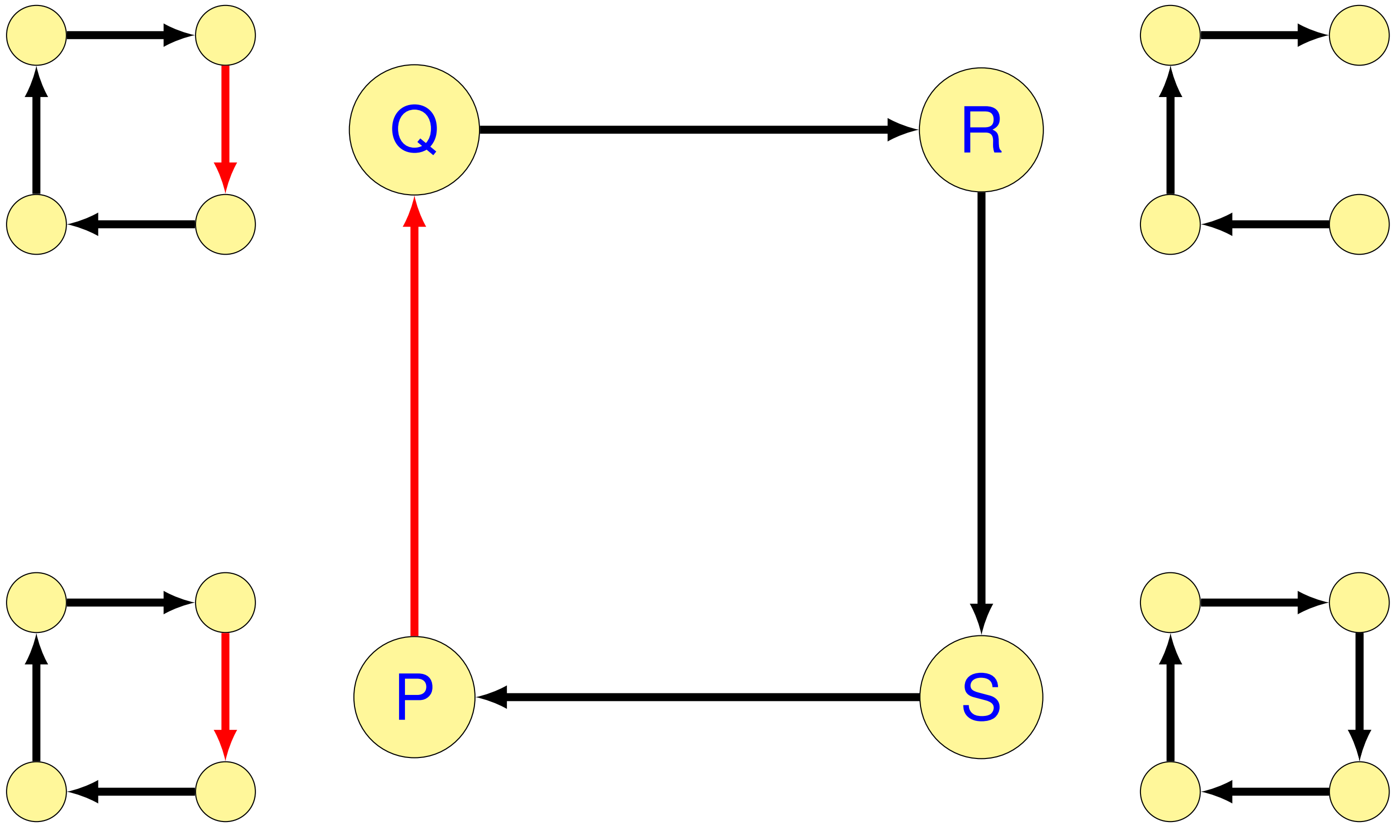
The Network

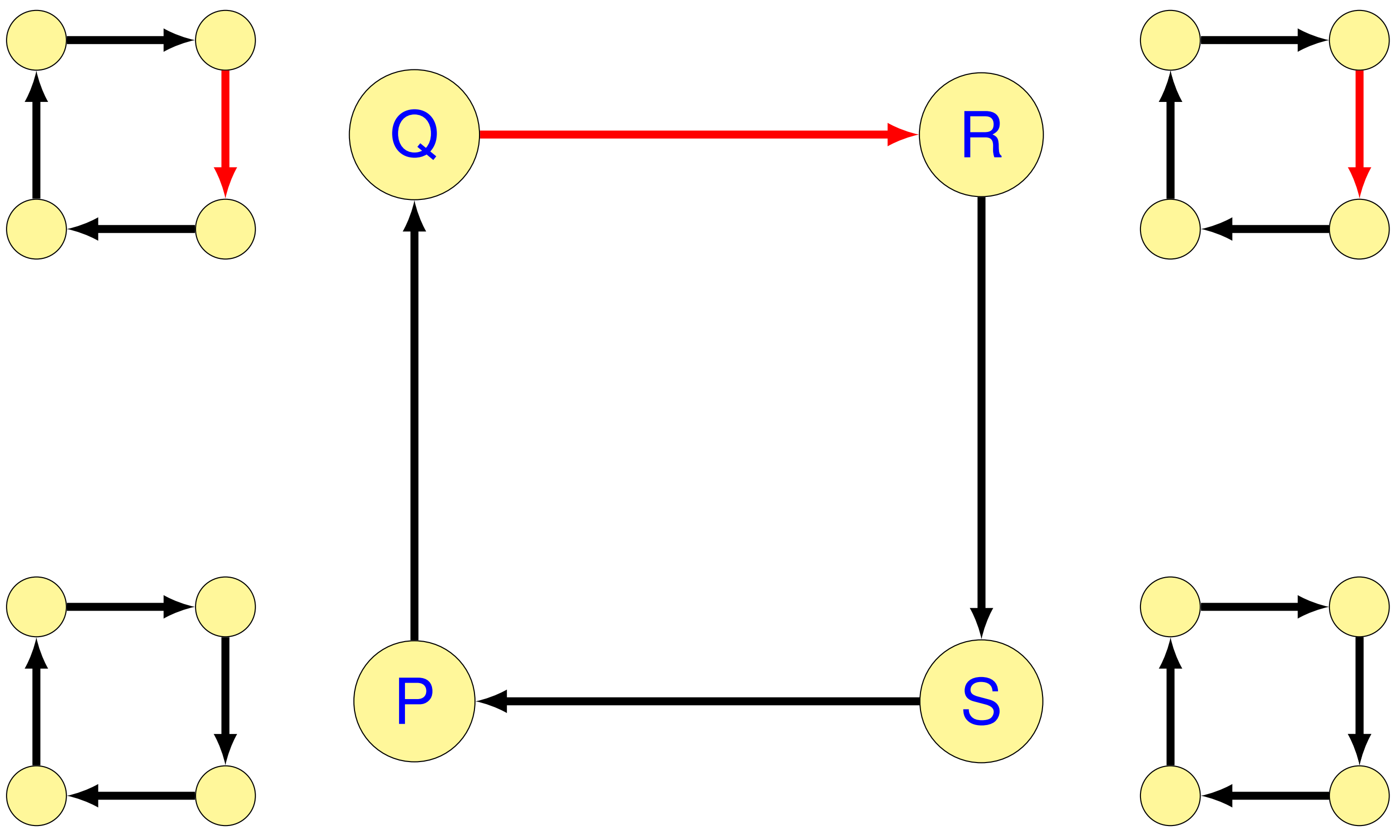


The Network



The Network




Q
R
P
S

Remarks About the Previous Animation

- We have not shown any link removal.

- Messages from one node to the other travelled instantaneously.

- As a consequence, we had no loss of messages.

- But unfortunately:
    - links can be removed
    - messages do not travel instantaneously

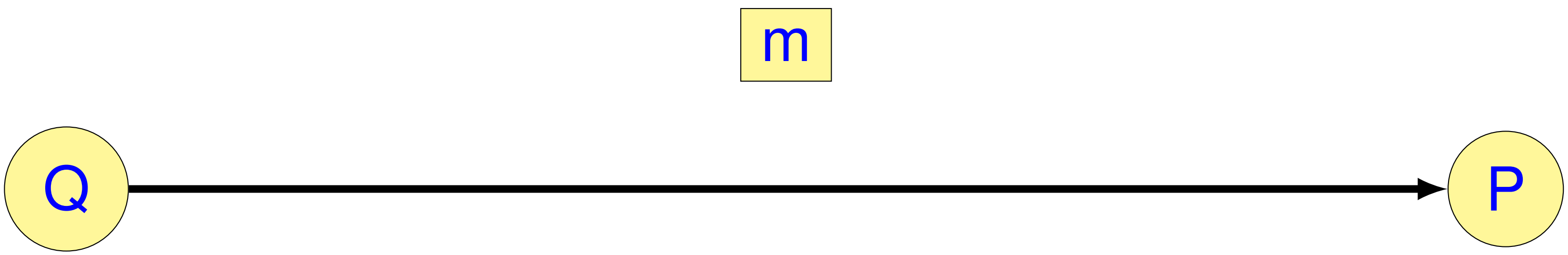
- Hence messages between nodes can be lost.

Requirement 3: Loss of Messages

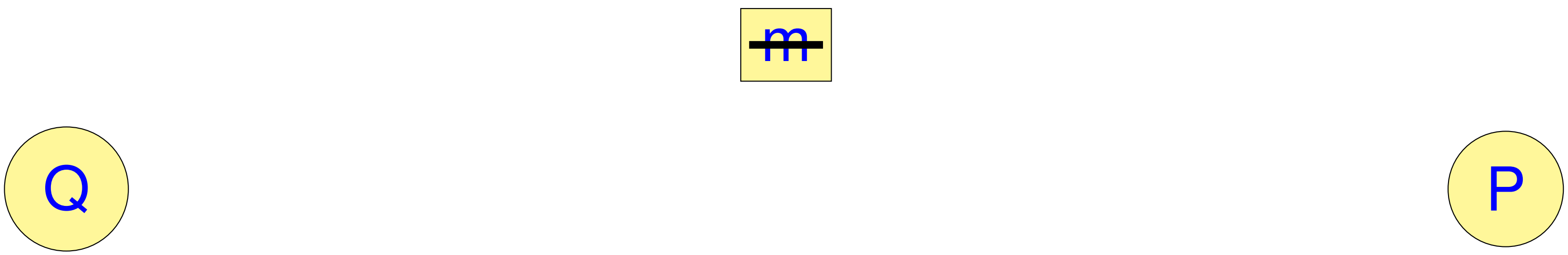
A message sent from a to b is lost if the link from a to b, which existed when the message was sent, is broken before the message reaches b	REQ-5

A Message m to be Sent from Node Q to Node P

The Message is Travelling



The Link is broken: the Message is Lost



The Link is not broken: the Message Reaches its Destination

Requirement 4: the Final Touch

- The dog-nodes can all reach together the master-network.

  (under certain conditions)

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-6

- The conditions are the following:
  - the network is not modified for a certain time,
  - the network is strongly connected when not modified

Summary of Requirements (1)

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

When a link from node a to node b is added to or removed from the network then node b is DIRECTLY made aware of it	REQ-2

The neighbors of a node b are the nodes that are connected to b by a link l entering in b.	REQ-3

Summary of Requirements (2)

Neighbors of b send to b their local networks with MESSAGES sent to it through the connecting link	REQ-4

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-5

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-6

Difficulties and Strategy

Modeling Difficulties

- Two ways of detecting link modifications: direct or with messages.

- Hence contradictory messages may circulate on the network.

- Which one reflects the real situation of the physical graph?

- Messages can be lost.

- What is the final condition (where all nodes know the graph)?

- How to express it?

- Where and how to start the modeling?

Guidelines

- Use abstraction

- Proceed by successive refinements

- Make proofs at each refinement level

Design Strategy: a Proposal

- Initial Model: Introducing the physical network.

- Refinement 1: Introducing a global logical network (detected).

- Refinement 2: Introducing local networks and message reservoirs.

- Refinement 3: Introducing ages of network changes.

- Refinement 4: Merging events.

- Refinement 5: Removing message reservoirs.

- Refinement 6: Finding the limit condition.

- Refinement 7: Handling loss of messages.

Formal Development

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

sets: L

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

variables: NET

inv0_1: $NET \subseteq L$

- The physical network, NET, is represented by a set of links L

init
$\quad NET := \varnothing$

Modify_up
$\quad$ **any**
$\quad\quad l$
$\quad$ **where**
$\quad\quad l \notin NET$
$\quad$ **then**
$\quad\quad NET := NET \cup \{l\}$
$\quad$ **end**

Modify_dn
$\quad$ **any**
$\quad\quad l$
$\quad$ **where**
$\quad\quad l \in NET$
$\quad$ **then**
$\quad\quad NET := NET \setminus \{l\}$
$\quad$ **end**

- These events denote changes of the physical network NET

First Refinement: Introducing a Global Logical Network

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-2

variables: NET net

inv1_1: $net \subseteq L$

- We do not introduce the nodes yet (nor the local networks).

-The variable net denotes the global logical image directly detected.

- It is an abstraction representing the direct detection of the network.

First Refinement: Initialisation

init

$$NET := \varnothing$$
$$net := \varnothing$$

- Events modify_up and modify_dn are not modified.

```
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
```

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

- Mind the convergence (Master and Dog)

Second Refinement: Introducing Local Networks

sets: N

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

variables: NET net l_net

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

- l_net denotes the local relation between nodes and links

- $l\_net[\{n\}]$ is the set of links recorded in node n

- The local networks are modified in two distinct ways:

  - either directly when detected in the physical network

  - or indirectly by neighbors

variables: $\ldots$
m_net_up
m_net_dn

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

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

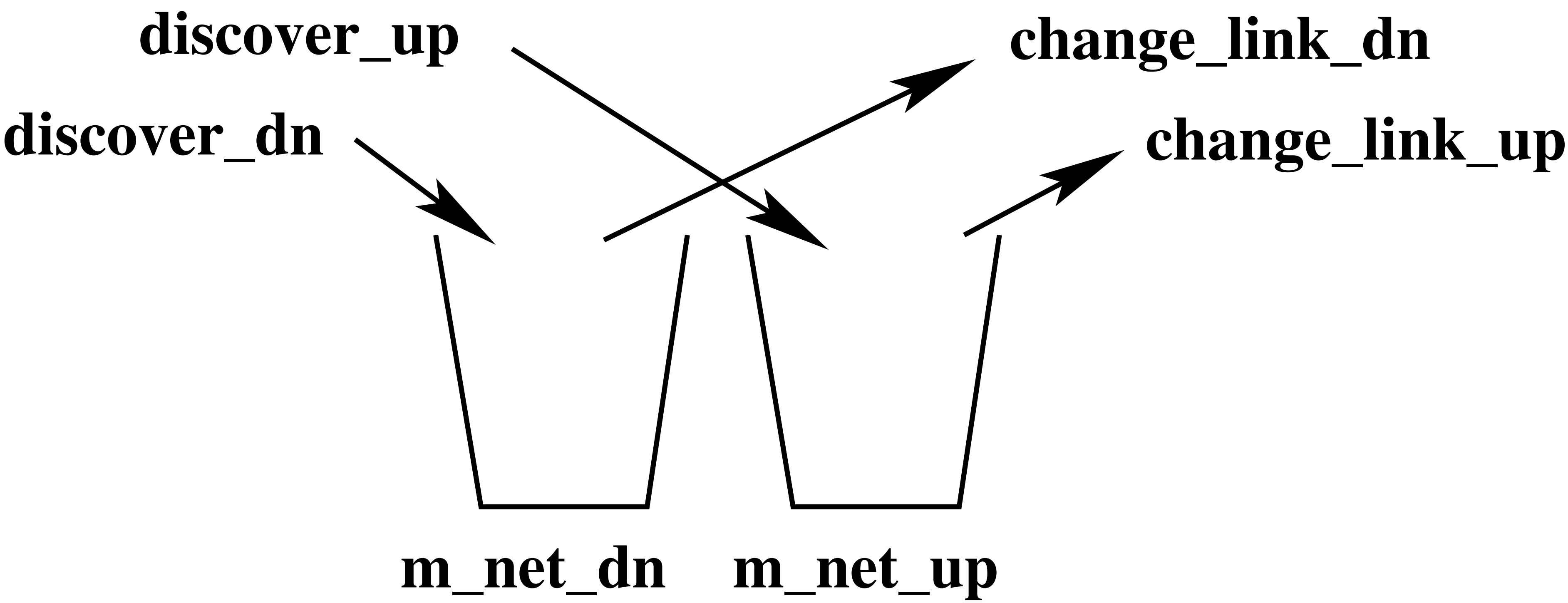
inv2_4: $m\_net\_up \cap m\_net\_dn = \varnothing$

- $n \mapsto l \in m\_net\_up$ means there is a message for node n about up-link l.

- $n \mapsto l \in m\_net\_dn$ means there is a message for node n about dn-link l.

- m_net_up and m_net_dn are abstractions with messages from neighbors

- Such messages are supposed to be in transit in m_net_up and m_net_dn


discover_up
discover_dn
change_link_dn
change_link_up
m_net_dn
m_net_up

init

$$
\begin{aligned}
NET &:= \varnothing \\
net &:= \varnothing \\
l\_net &:= \varnothing \\
m\_net\_up &:= \varnothing \\
m\_net\_dn &:= \varnothing
\end{aligned}
$$

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

- Node n, concerned by the change discovery, is chosen arbitrarily

- Local network of node n is updated directly ($l\_net := \ldots$)

- New messages about link l are prepared for nodes ($m\_net\_up := \ldots$)

- Old messages about link l are MAGICALLY discarded ($m\_net\_dn := \ldots$)

Second Refinement: Refining Event discover_dn

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

- Node n, concerned by the change discovery, is chosen arbitrarily

- Local network of node n is updated directly ($l\_net := \ldots$)

- New messages about link l are prepared for nodes ($m\_net\_dn := \ldots$)

- Old messages about link l are MAGICALLY discarded ($m\_net\_up := \ldots$)

Second Refinement: Updating l_net from "neighbors" (1)

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

- **anticipated** means the convergence is proved later.

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

Second Refinement: Updating l_net from "neighbors" (3)

change_link_2
  status
    anticipated
  any
    ln
  where
    $ln \in N \leftrightarrow L$
  then
    $l\_net := ln$
  end

- To be explained in next refinement.

Third Refinement: Introducing ages

- Each modification is "decorated" with a natural number: the age.

- An even number for an addition.

- An odd number for a removal.

- Such numbers are always increasing.

- Consequence, removal of:
  - net
  - m_net_up and m_net_dn
  - l_net

constants: $parity$

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

axm3_2: $parity(0) = 1$

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

Third Refinement: The age of a Link

variables: $NET \quad age$

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

- The age is recorded when events discover_up or discover_dn occur

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

- Thanks to **inv3_2**, variable net can be removed (replaced by age).

Third Refinement: Locally Recorded Age (1)

variables: NET age l_age

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

- Each local network contains the age (when recorded) of each link

- This locally recorded age is at most equal to the "real" age (**inv3_4**)

inv3_4: $\forall n, l \cdot l\_age(n \mapsto l) \leq age(l)$

Third Refinement: Locally Recorded Age (2)

inv3_5: $\forall n, l \cdot n \mapsto l \in l\_net \Leftrightarrow parity(l\_age(n \mapsto l) = 1)$

- Thanks to **inv3_5**, variable l_net can be removed (replaced by l_age).

Third Refinement: Messages with Ages (1)

variables: $\ldots\ m\_net$

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

- Each message (when prepared) contains the age of the link

- Messages with the same link and different ages might exist

inv3_7: $\forall\, n, l, a \cdot n \mapsto l \mapsto a \in m\_net \;\Rightarrow\; a \leq age(l)$

- The recorded age in a message is at most equal to the real age

inv3_8: $\forall\, n, l, a \cdot l\_age(n \mapsto l) < a \;\wedge\; a \leq age(l) \;\Rightarrow\; n \mapsto l \mapsto a \in m\_net$

- Invariant **inv3_8** will be used in refinement 5

Third Refinement: Messages with Ages (2)

inv3_9: $\forall n, l \cdot n \mapsto l \in m\_net\_up \Leftrightarrow n \mapsto l \mapsto age(l) \in m\_net \wedge parity(age(l)) = 1$

inv3_10: $\forall n, l \cdot n \mapsto l \in m\_net\_dn \Leftrightarrow n \mapsto l \mapsto age(l) \in m\_net \wedge parity(age(l)) = 0$

- m_net_up and m_net_dn can be removed (replaced by m_net).

Third Refinement: Initialisation

init

$$
\begin{aligned}
NET &:= \varnothing \\
age &:= L \times \{0\} \\
l\_age &:= N \times L \times \{0\} \\
m\_net &:= \varnothing
\end{aligned}
$$

Third Refinement: Events (1)

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

discover_up
any
l
n
where
$l \in NET$
$parity(age(l)) = 0 \quad /* \ l \notin net \ \Leftrightarrow \ parity(age(l)) = 0$ **(inv3_2)** $*/$
$n \in N$
then
$age(l) := age(l) + 1$
$l\_age(n \mapsto l) := age(l) + 1$
$m\_net := m\_net \ \cup \ ((N \setminus \{n\}) \times \{l\} \times \{age(l) + 1\})$
end

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

discover_dn
any
l
n
where
$l \notin NET$
$parity(age(l)) = 1$ $/* \ l \in net \ \Leftrightarrow \ parity(age(l)) = 1$ **(inv3_2)** $*/$
$n \in N$
then
$age(l) := age(l) + 1$
$l\_age(n \mapsto l) := age(l) + 1$
$m\_net := m\_net \ \cup \ ((N \setminus \{n\}) \times \{l\} \times \{age(l) + 1\})$
end

Third Refinement: Events (3)

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

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

- Events discover_up and discover_dn will be merged (same actions).

- The merged event will be convergent

Third Refinement: Events (4)

(abstract-)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

(abstract-)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

change_link_up
status
convergent
any
n
l
x
where
$x = age(l)$
$n \mapsto l \mapsto x \in m\_net$ /* **inv3_9** */
$parity(x) = 1$
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
$x = age(l)$
$n \mapsto l \mapsto x \in m\_net$ /* **inv3_10** */
$parity(x) = 0$
then
$l\_age(n \mapsto l) := x$
$m\_net := m\_net \setminus \{n \mapsto l \mapsto x\}$
end

Third Refinement: Events (5)

- The age x in the message is different from $age(l)$ but node n cannot know that
- It can only observe that it is greater than $l\_age(n \mapsto l)$

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

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

variant3: m_net

- Events change_link_up, change_link_dn and change_link_2 will be merged (same actions).
- The merged event will be convergent too.

Third Refinement: New Events

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 events will be merged (same actions)

- But the merged event will not be convergent

  (because event discard_2 is not convergent)

Fourth Refinement: Merging Events (1)

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

- The guard does not involve the variable m_net.

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

- In the next refinement, the guard will be made independent from m_net.

discard
  refines
    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

- The guard does not involve the variable m_net.

Fifth Refinement: Representing links with nodes

constants: $fst \quad snd \quad link$

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

Fifth Refinement: Removing variable m_net (1)

change_link
  any
    l
    x
    k
  where
    $k \in NET$
    $l\_age(snd(k) \mapsto l) < x$
    $x \leq l\_age(fst(k) \mapsto l) \qquad /* \; n \mapsto l \mapsto x \in m\_net \; */$
  with
    $n = snd(k)$
  then
    $l\_age(snd(k) \mapsto l) := x$
    $/* \; m\_net := m\_net \setminus \{snd(k) \mapsto l \mapsto x\} \; */$
  end

- The proof of guard strengthening uses invariant **inv3_8**

$$\forall\, n, l, x \cdot l\_age(n \mapsto l) < x \;\wedge\; x \leq age(l) \;\Rightarrow\; n \mapsto l \mapsto x \in m\_net$$

- and invariant **inv3_4**

$$\forall n, l \cdot l\_age(n \mapsto l) \leq age(l)$$

Fifth Refinement: Removing variable m_net (2)

- m_net is removed since it does not appear in any guard

discover
any
l
where
$l \in NET \Leftrightarrow parity(age(l)) = 0$
with
$n = snd(l)$
then
$age(l) := age(l) + 1$
$l\_age(snd(l) \mapsto l) := age(l) + 1$
end

- m_net is removed since it does not appear in any guard

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

variables: age $\quad l\_age$ $\quad G$

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

- Invariant **inv6_2** allows us to remove the variable NET.

- Invariant **inv6_3** will allow us to remove the variable age (in next refinement)

init

$$
\begin{aligned}
G &:= \varnothing \\
age &:= L \times \{0\} \\
l\_age &:= N \times L \times \{0\}
\end{aligned}
$$

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

Neighbors of b send to b their local networks with MESSAGES sent to it through the connecting link	REQ-4

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

- Node $fst(k)$ is a neighbor of $snd(k)$ since $fst(k) \mapsto snd(k) \in G$

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-6

- To be proved

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

- There is nothing to discover (the graph is still)
- There is nothing significative to transmitt.
- Note that these events are both convergent.

- Guard of the event discover:

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

- Negation of the guard of the event discover (deadlock condition):

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

- Guard of the event change_link:

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

- Negation of the guard of the event change_link (deadlock condition):

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

Sixth Refinement: Finding the Limit Condition (4)

- Observe strong connectivity of graph G

thm6_1:

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

- Hint: instantiate s with $\{n \mid l\_age(n \mapsto l) = age(l)\}$.

thm6_2:

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

thm6_2:

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

thm6_3:

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

- Hint: Use invariant **inv3_5**

inv3_5: $\forall n, l \cdot l \in l\_net[\{n\}] \;\Leftrightarrow\; parity(l\_age(n \mapsto l)) = 1$

thm6_3:

$$\forall l \cdot l \in NET \Leftrightarrow parity(age(l)) = 1$$
$$\forall\, l, a, b \cdot a \mapsto b \in G \;\Rightarrow\; l\_age(a \mapsto l) \leq l\_age(b \mapsto l)$$
$$\forall s \cdot s \neq \varnothing \;\wedge\; G[s] \subseteq s \;\Rightarrow\; N \subseteq s$$
$$\Rightarrow$$
$$\forall n \cdot l\_net[\{n\}] = NET$$

Event discover deadlocks

Event change_link deadlocks

Physical graph is strongly connected

$\Rightarrow$

Local networks are all equal to physical network

Sixth Refinement: Non-convergent Events(1)

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
$G := G \setminus \{fst(l) \mapsto snd(l)\}$
end

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

Seventh Refinement: Introducing Message Channels m

variables: G $\quad l\_age$ $\quad m$

inv7_1: $m \subseteq L \times L \times \mathbb{N}$

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

- $k \mapsto l \mapsto x \in m$ means a message dealing with link l and age x is travelling on link k.

- **inv7_2** says that ages at the origin of a message, $l\_age(fst(k) \mapsto l)$, can only increase while message is travelling

- **inv7_3** says that a message is always travelling on an existing link

Seventh Refinement: Initialisation

init

$G := \varnothing$

$m := \varnothing$

$l\_age := N \times L \times \{0\}$

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-5

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

- All messages travelling on link l are lost when l is broken.

(abstract-)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

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

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

Neighbors of b send to b their local networks with MESSAGES sent to it through the connecting link	REQ-4

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

- Observe the non-determinacy on l

- This event is not convergent: a message can always be sent.

(abstract-)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

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

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

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

Conclusion

- The final result is particularly simple:
  - Three variables:
    - G, l_age, m
  - Simple events:
    - Modify_up, Modify_dn
    - discover
    - send_message
    - accept_message
    - discard_message
- However, the development is not that simple

Conclusion: Proof Statistics

model	proofs	auto.	manual	% auto.
Ref. 1	3	3	0	100
Ref. 2	19	19	0	100
Ref. 3	115	79	36	68
Ref. 4	6	5	1	83
Ref. 5	18	17	1	94
Ref. 6	13	10	3	76
Ref. 7	38	32	6	84
Total	212	165	47	77

THAT'S ALL