Distributed algorithms

The Leader Election Protocol (IEEE 1394)

J.R. Abrial and Dominique Cansell and Dominique Méry

26th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems

September 26-29 2006, Paris, France

This Session

- Background :-)

- An informal presentation of the protocol :-)

- Step by step formal design :-|

- Short Conclusion. :-)

IEEE 1394 High Performance Serial Bus (FireWire)

- It is an international standard

- There exists a widespread commercial interest in its correctness

- Sun, Apple, Philips, Microsoft, Sony, etc involved in its development

- Made of three layers (physical, link, transaction)

- The protocol under study is the Tree Identify Protocol

- Situated in the Bus Reset phase of the physical layer

The Problem (1)

- The bus is used to transport digitized video and audio signals

- It is “hot-pluggable”

- Devices and peripherals can be added and removed at any time

- Such changes are followed by a bus reset

- The leader election takes place after a bus reset in the network

- A leader needs to be chosen to act as the manager of the bus

The Problem (2)

- After a bus reset: all nodes in the network have equal status

- A node only knows to which nodes it is directly connected

- The network is connected

- The network is acyclic

References (1)

BASIC

- IEEE. *IEEE Standard for a High Performance Serial Bus. Std 1394-1995*. 1995

- IEEE. *IEEE Standard for a High Performance Serial Bus (supplement). Std 1394a-2000*. 2000

References (2)

GENERAL

- N. Lynch. *Distributed Algorithms*. Morgan Kaufmann. 1996

- R. G. Gallager et al. *A Distributed Algorithm for Minimum Weight Spanning Trees*. IEEE Trans. on Prog. Lang. and Systems. 1983.

References (3)

MODEL CHECKING

- D.P.L. Simons et al. *Mechanical Verification of the IEE 1394a Root Contention Protocol using Uppaal2* Springer International Journal of Software Tools for Technology Transfer. 2001

- H. Toetenel et al. *Parametric verification of the IEEE 1394a Root Contention Protocol using LPMC* Proceedings of the 7th International Conference on Real-time Computing Systems and Applications. IEEE Computer Society Press. 2000

References (4)

THEOREM PROVING

- M. Devillers et al. *Verification of the Leader Election: Formal Method Applied to IEEE 1394*. Formal Methods in System Design. 2000

- J.R. Abrial et al. *A Mechanically Proved and Incremental Development of IEEE 1394*. To be published 2002

Informal Absract Properties of the Protocol

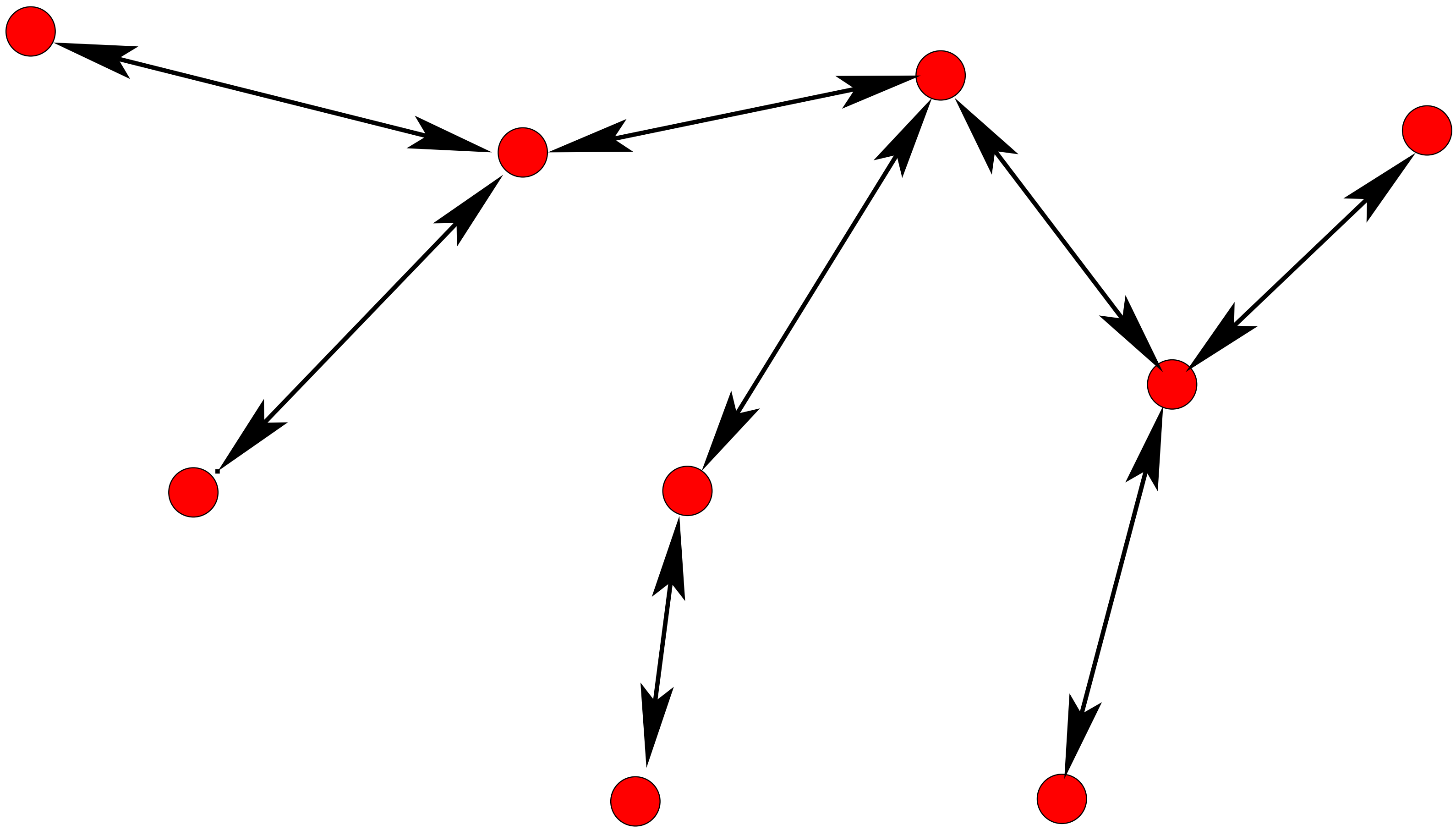
- We are given a connected and acyclic network of nodes

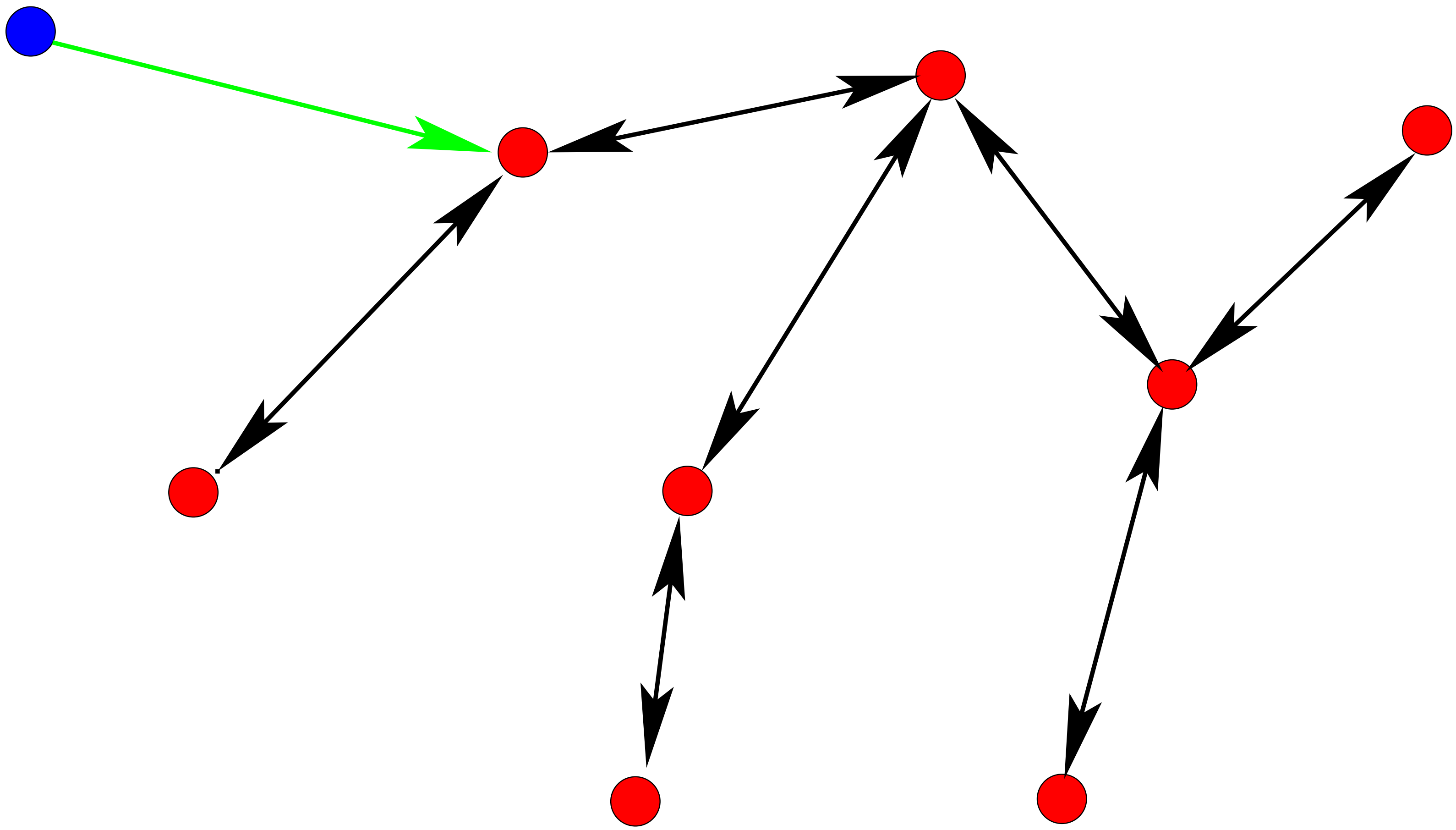
- Nodes are linked by bidirectional channels

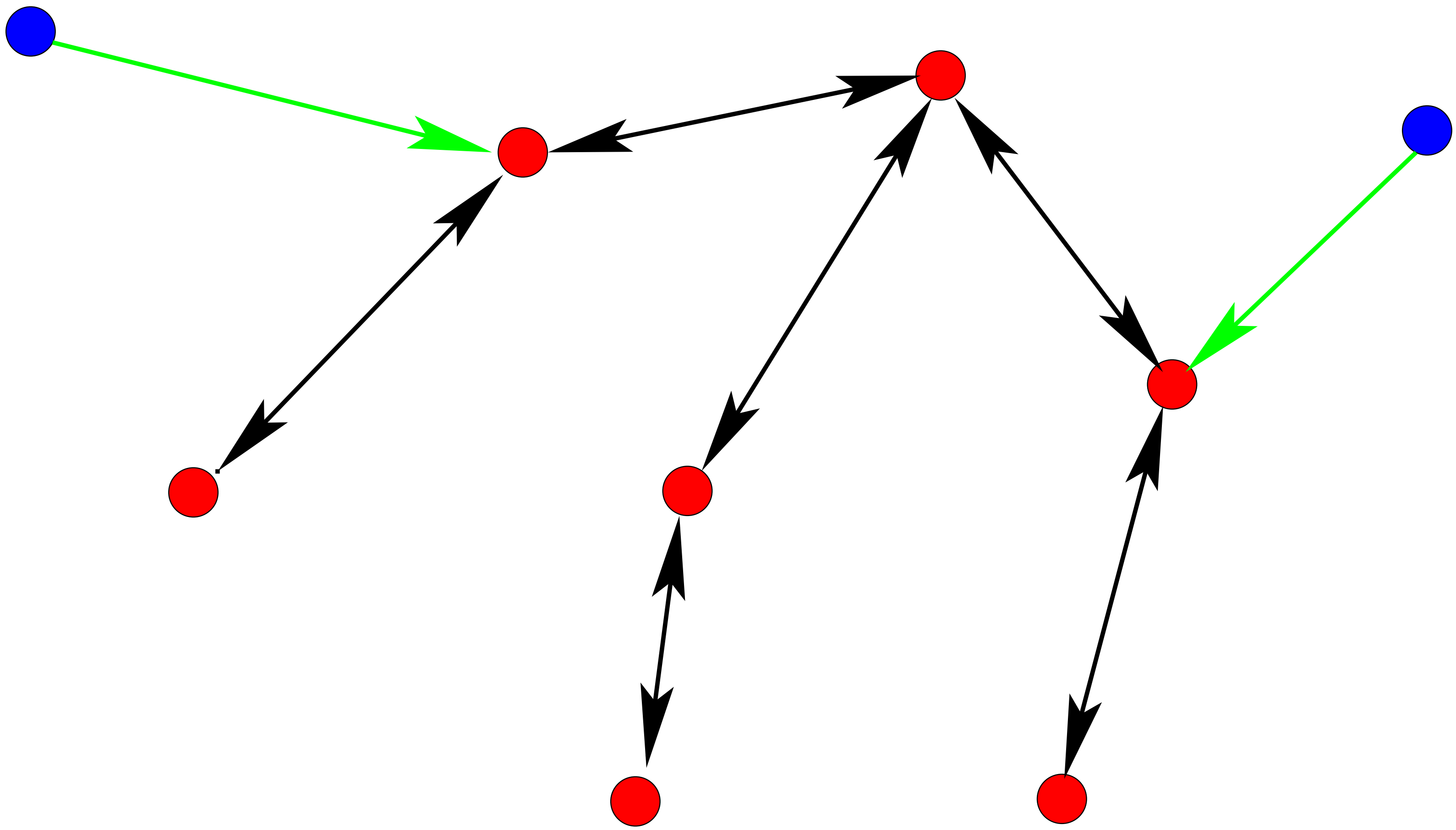
- We want to have one node being elected the leader in a finite time

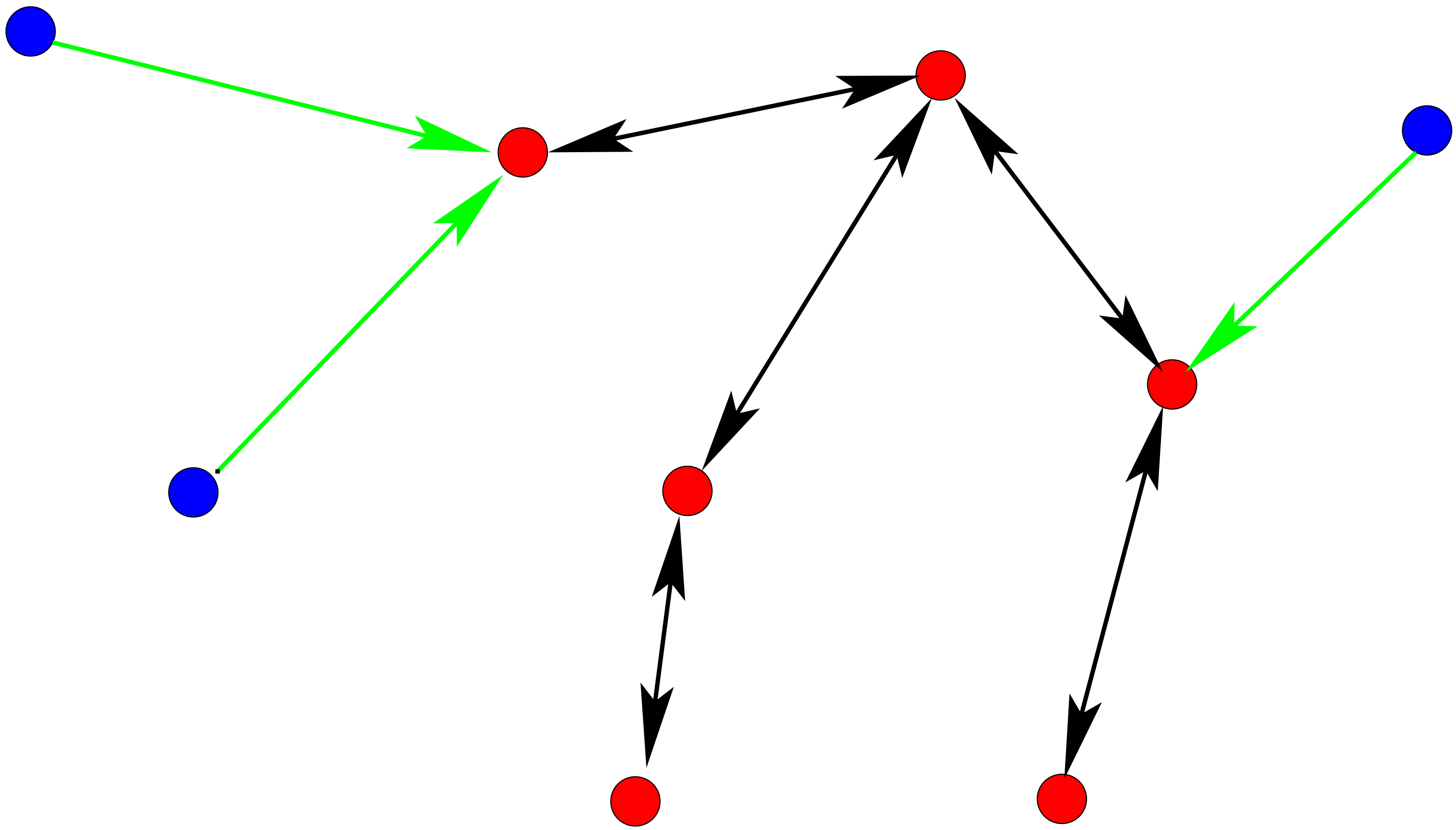
- This is to be done in a distributed and non-deterministic way

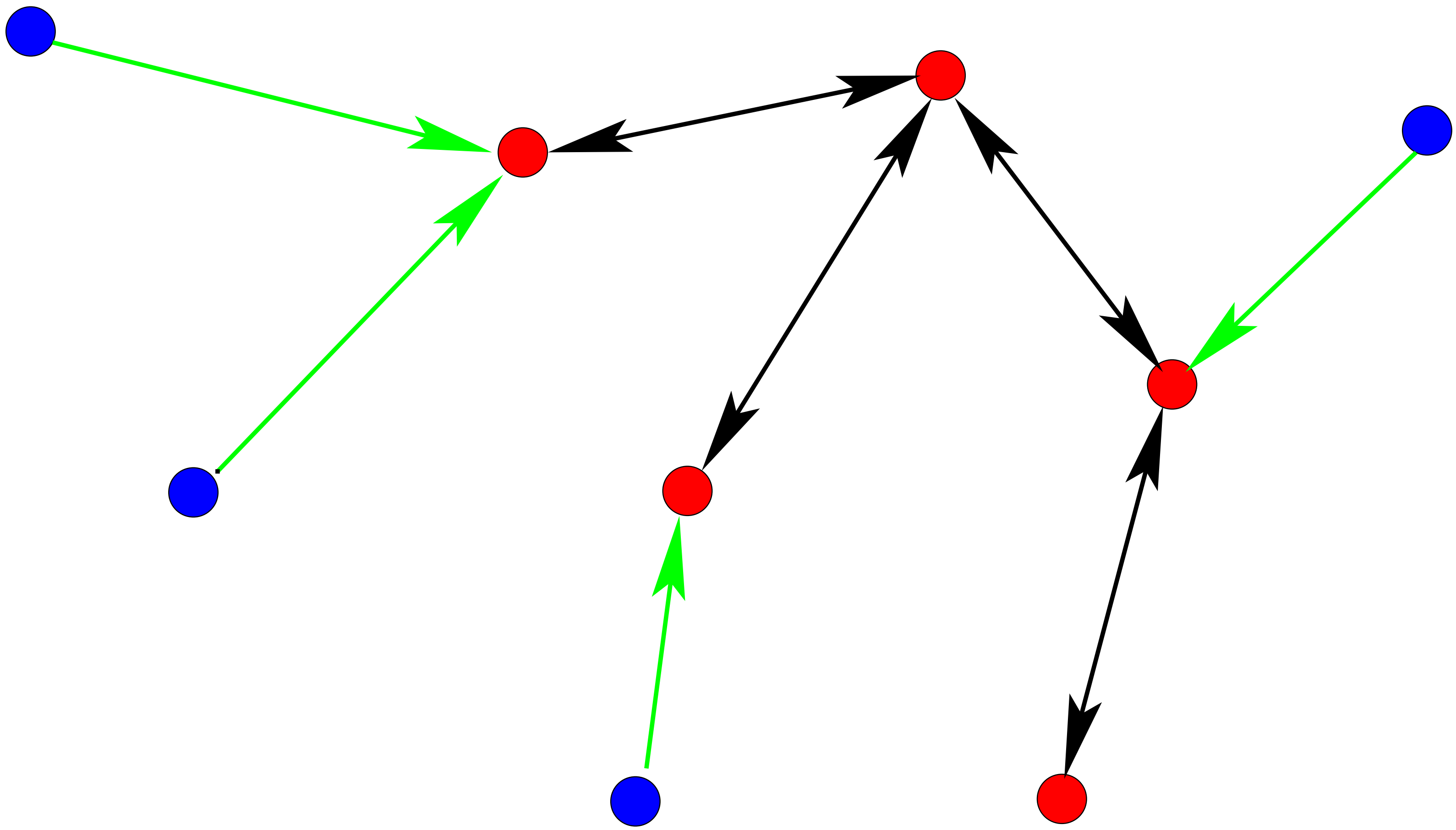
- Next are two distinct abstract animations of the protocol

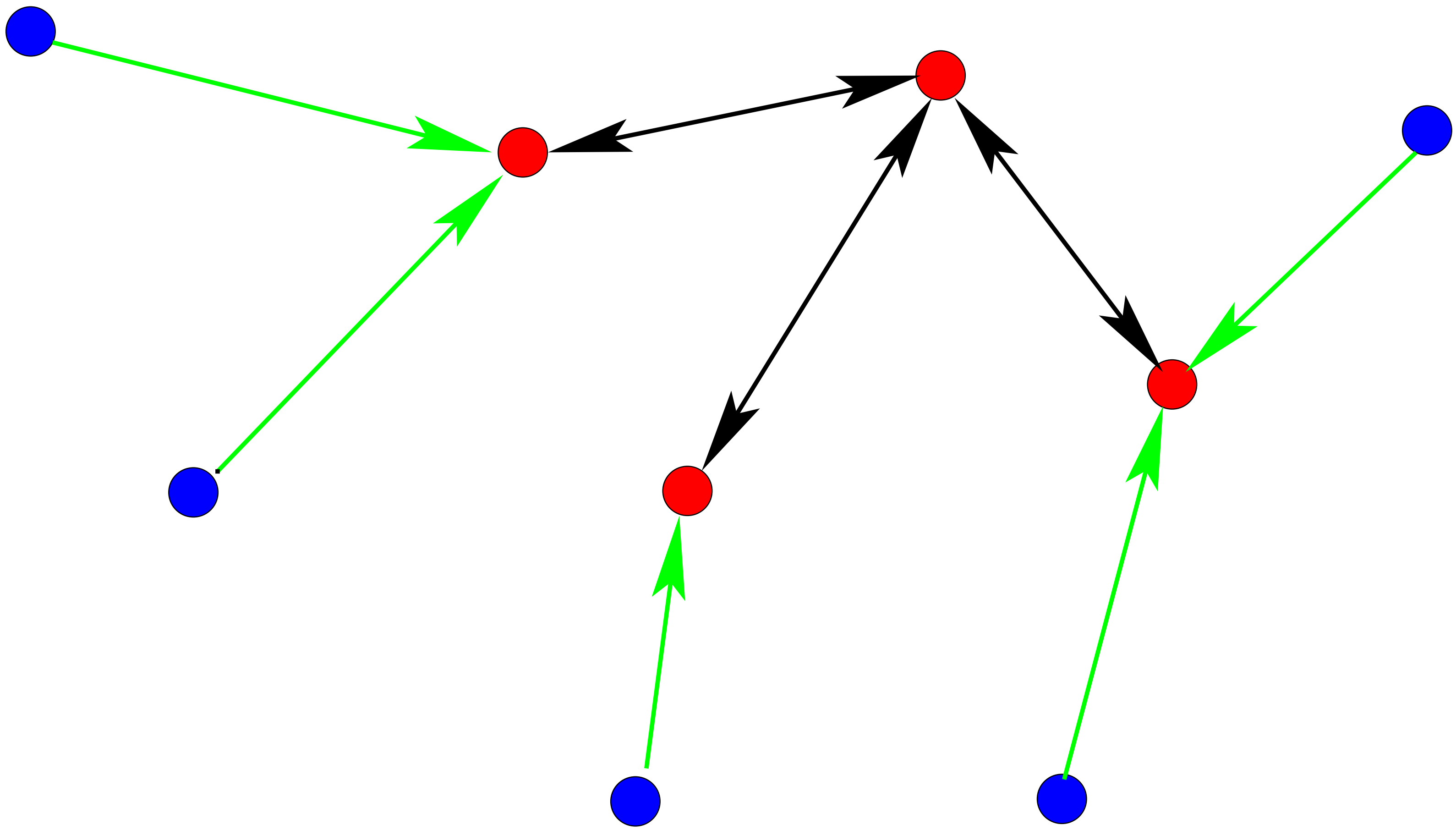


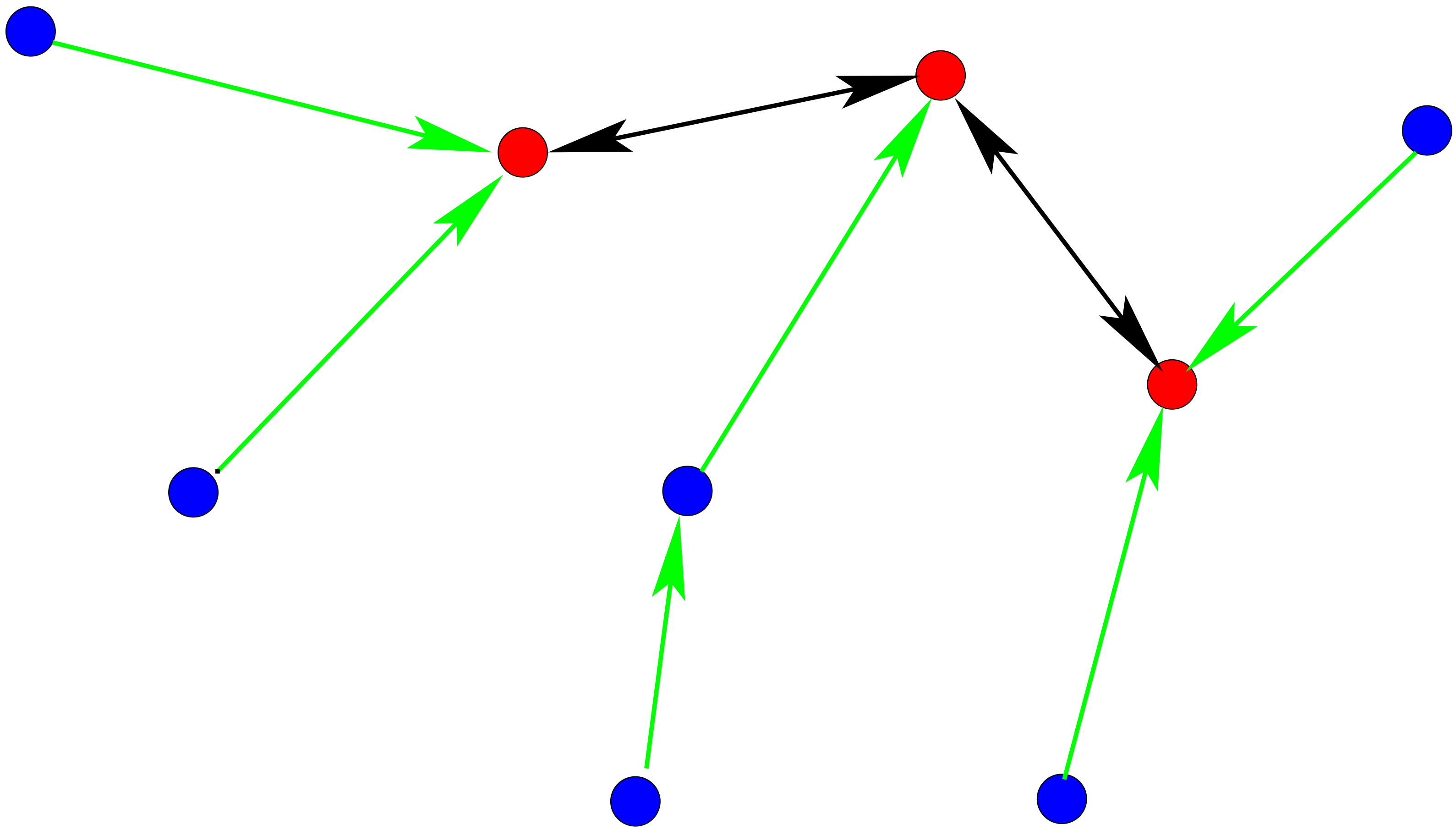


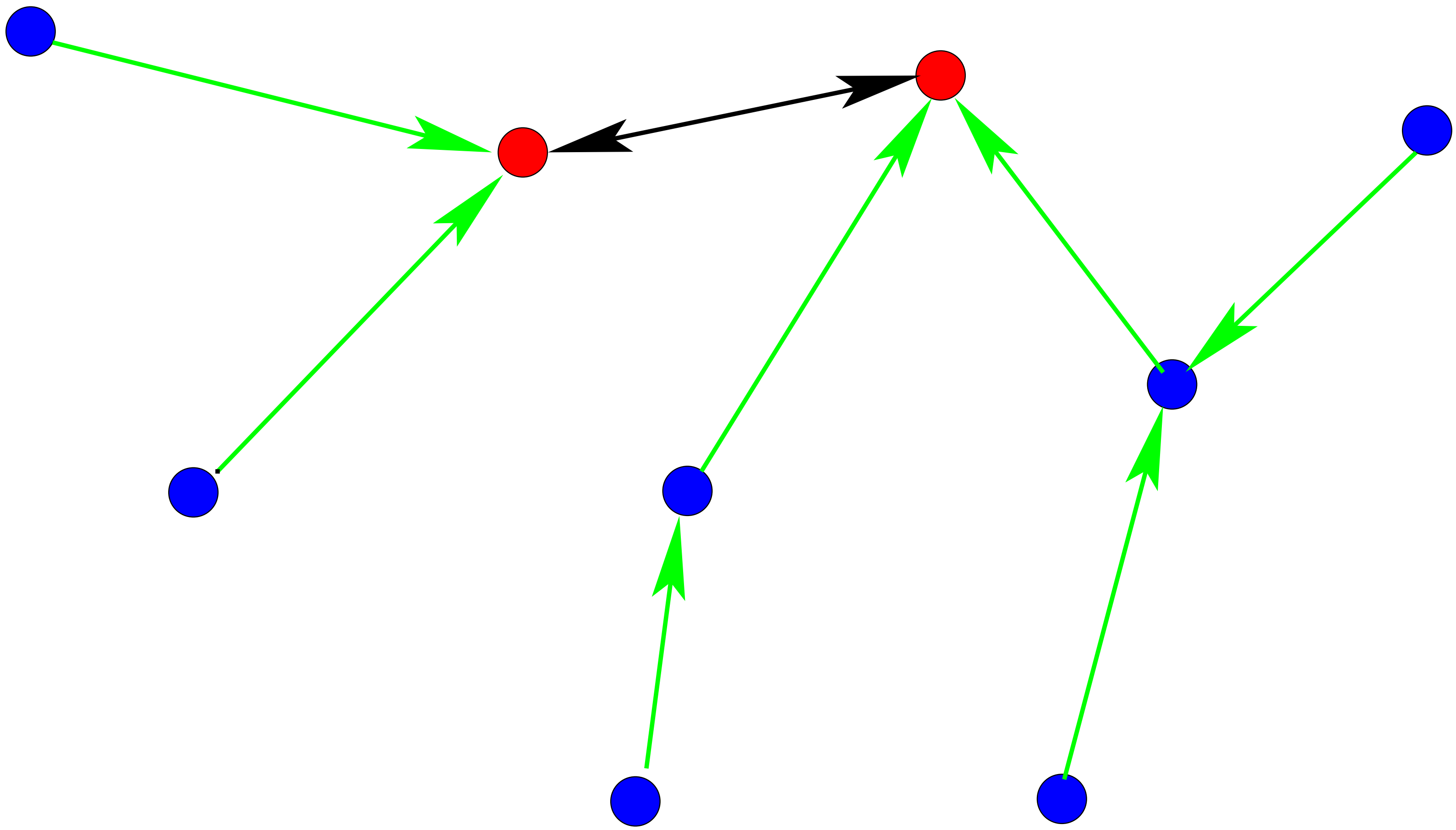


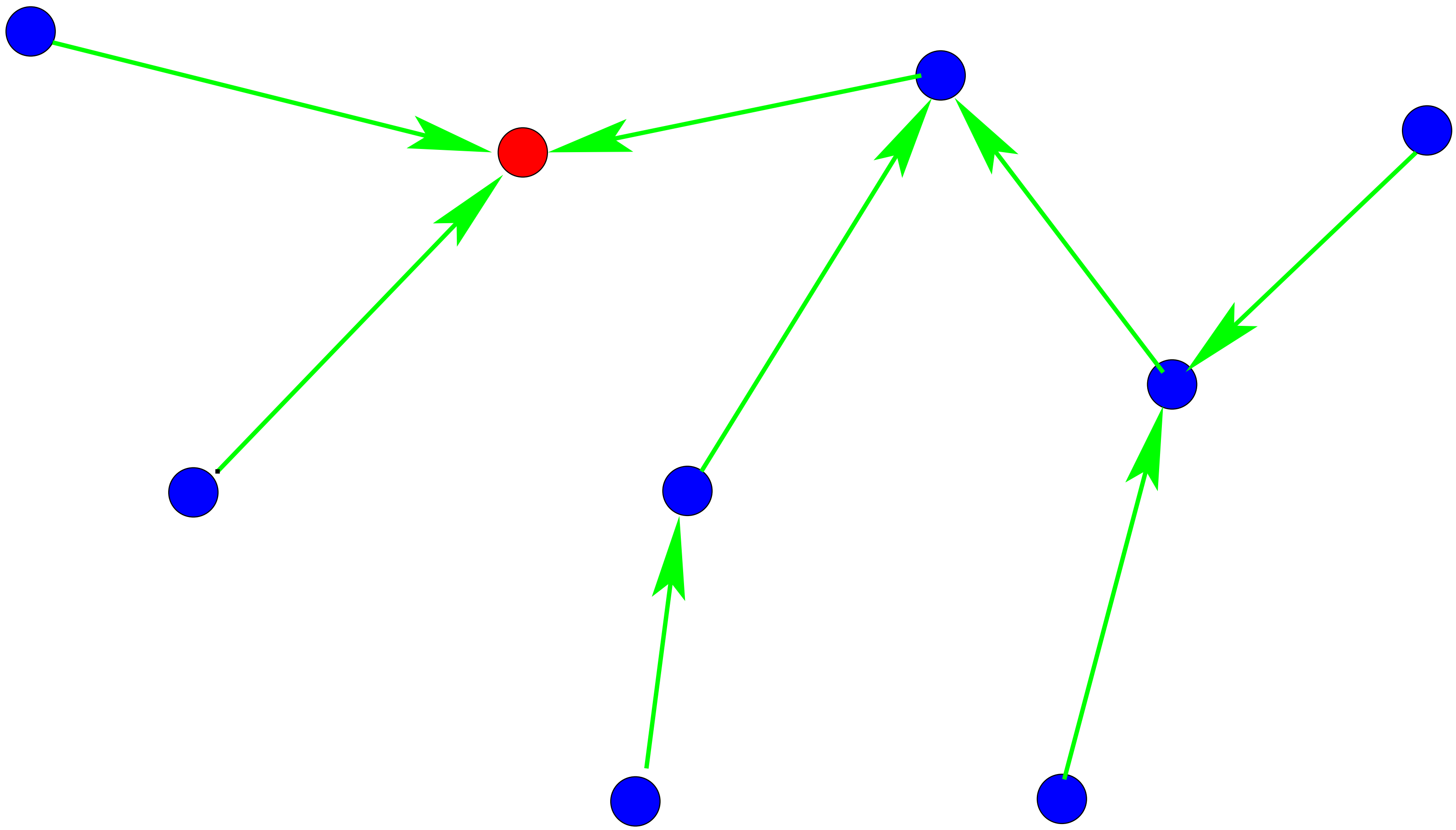


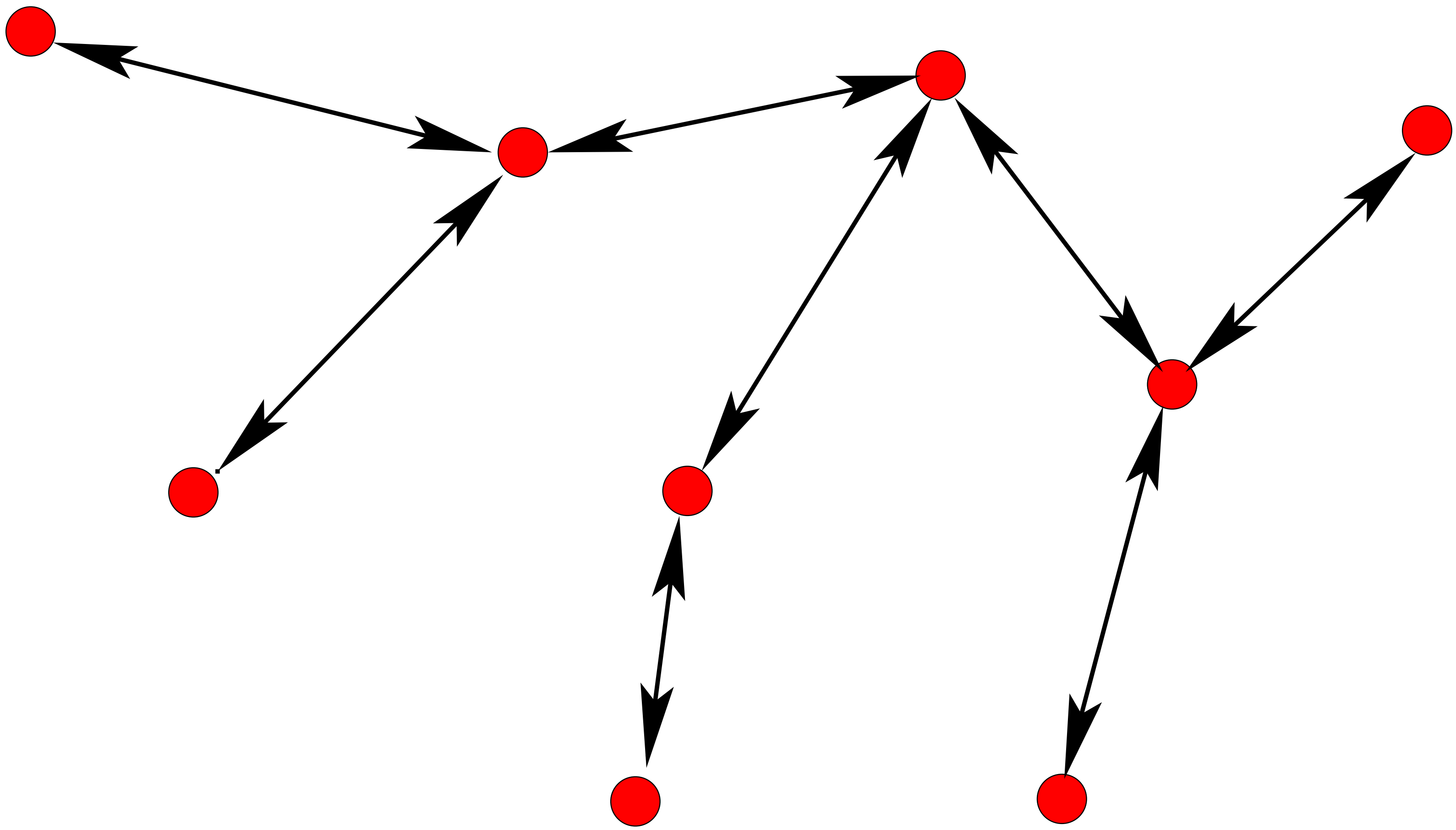


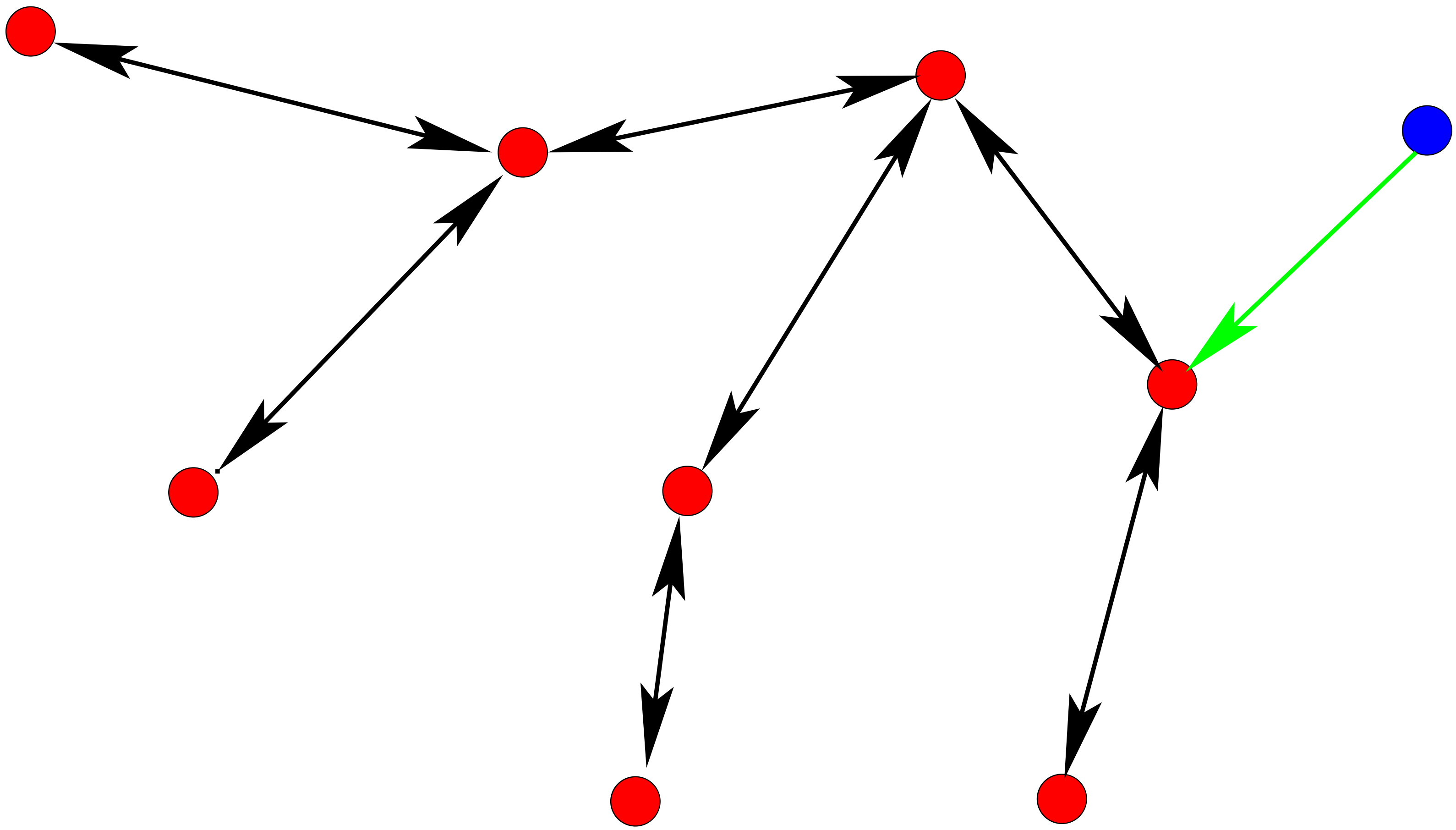


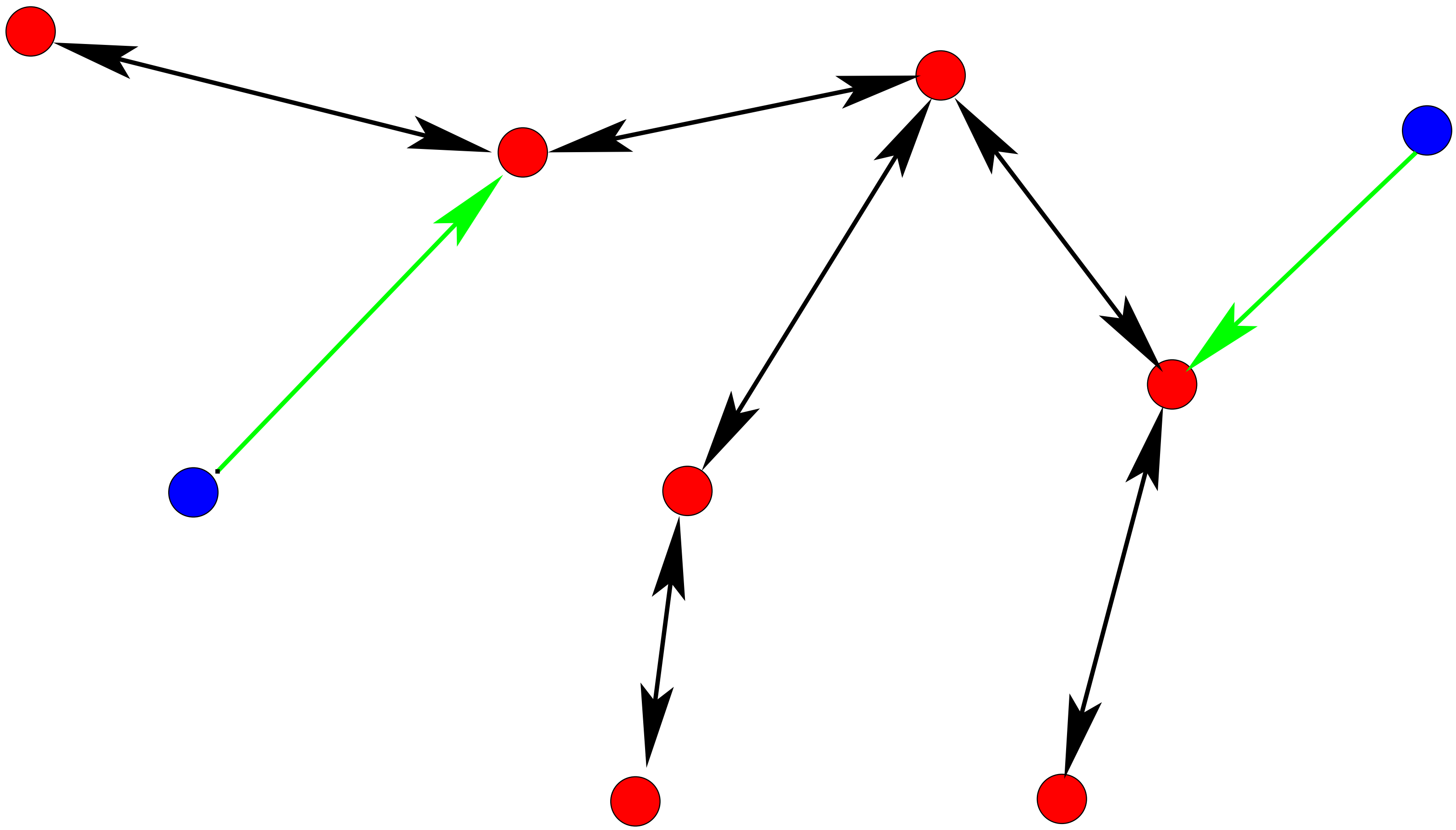


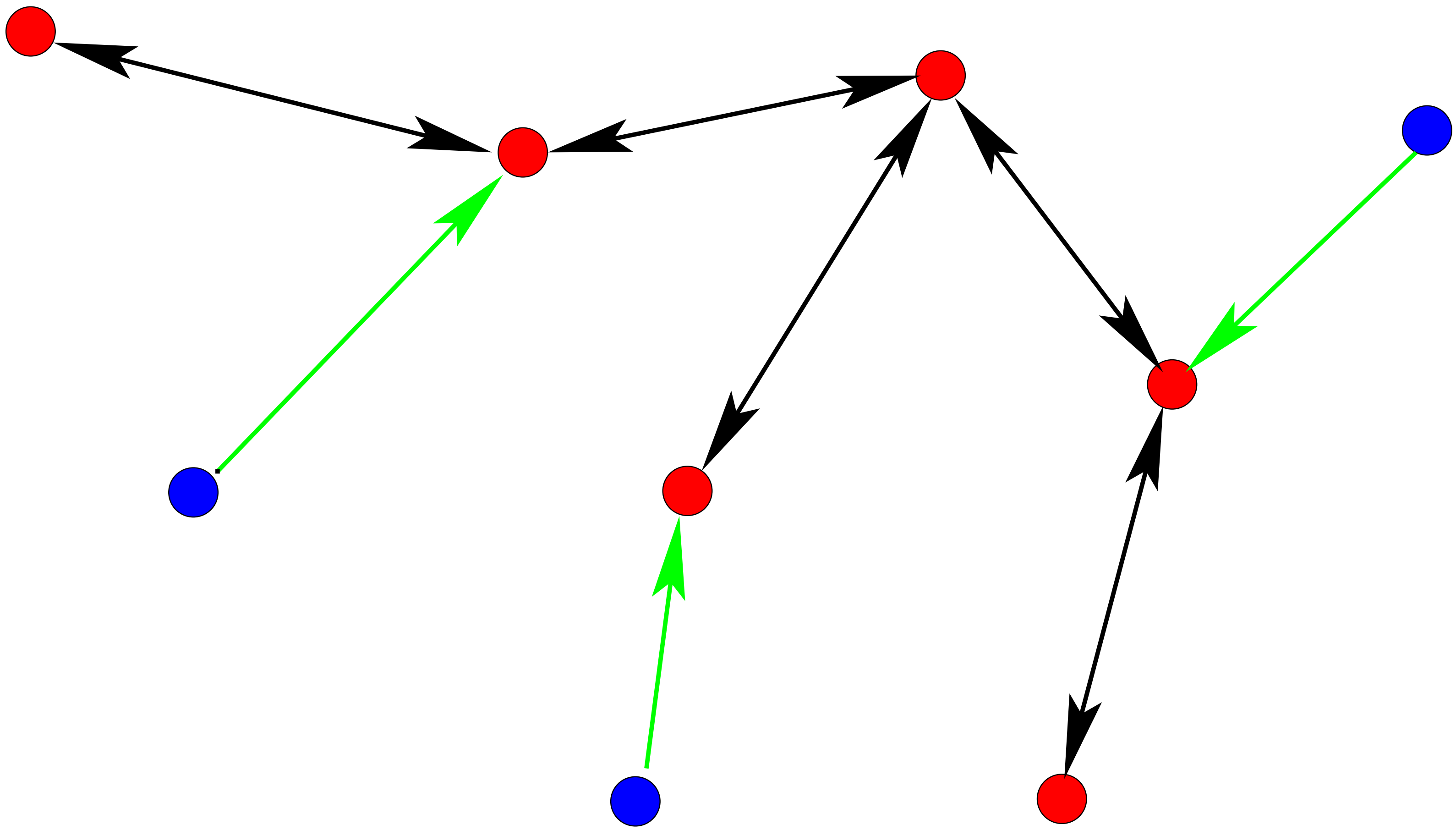


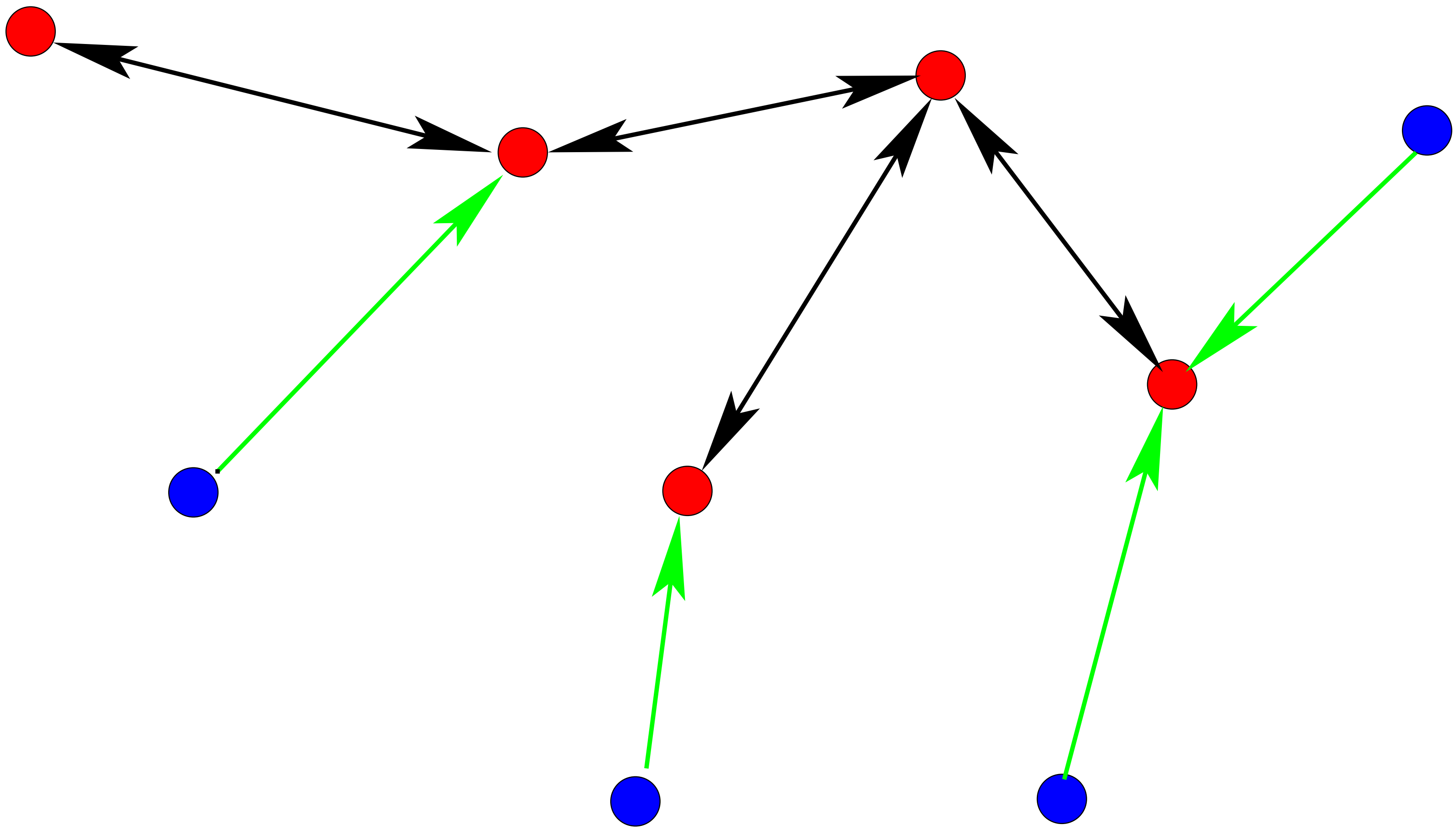


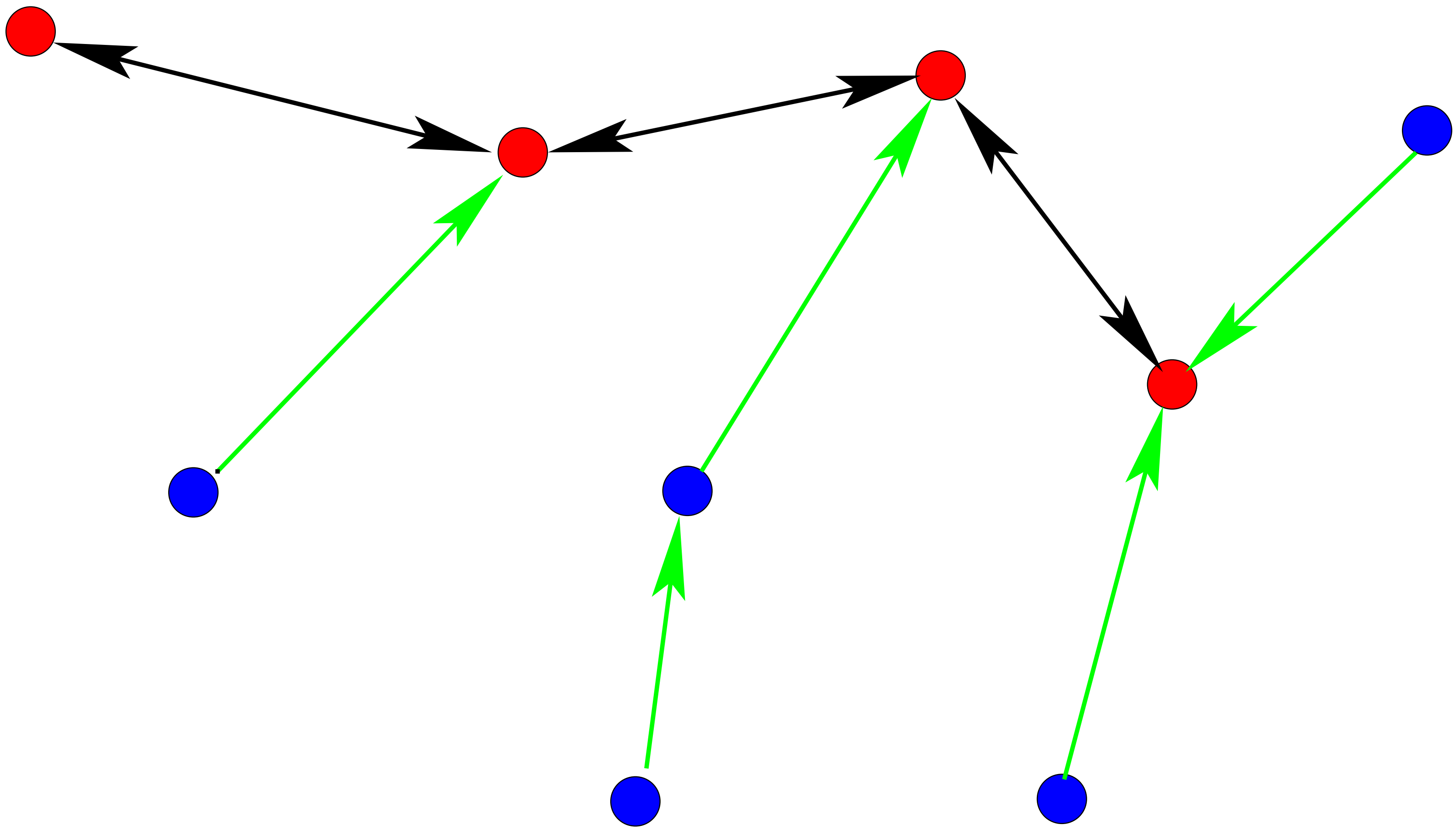


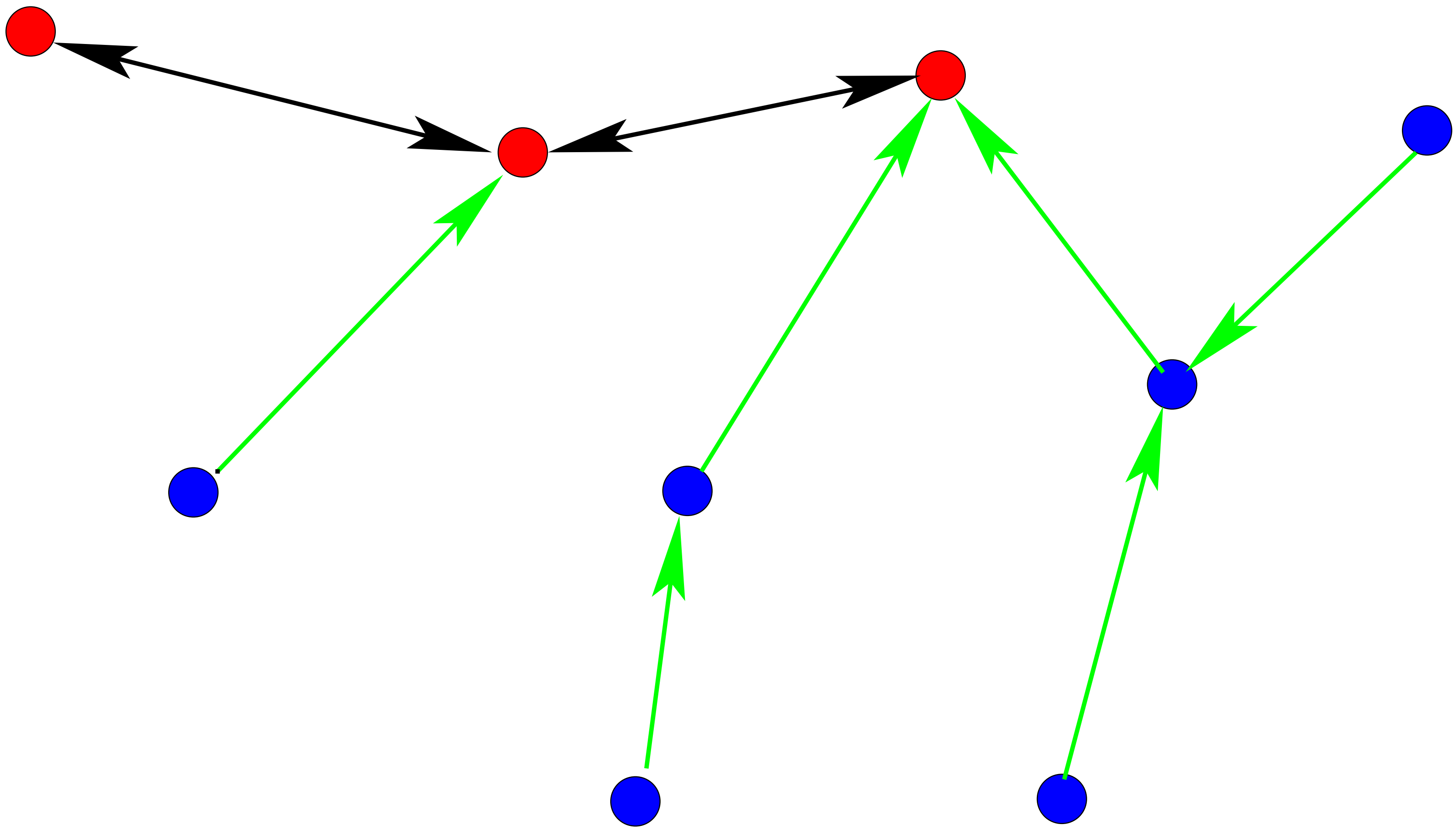


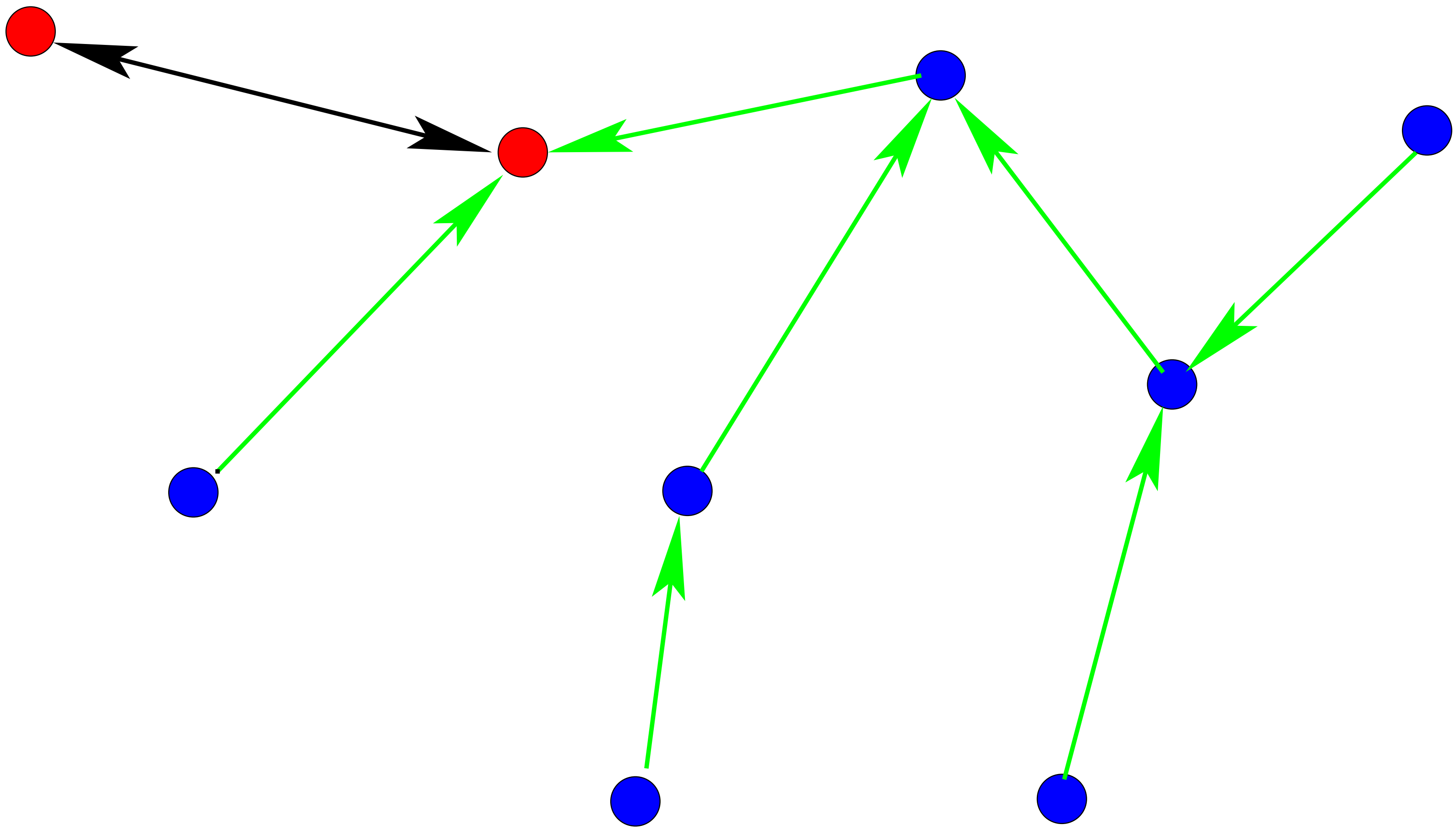


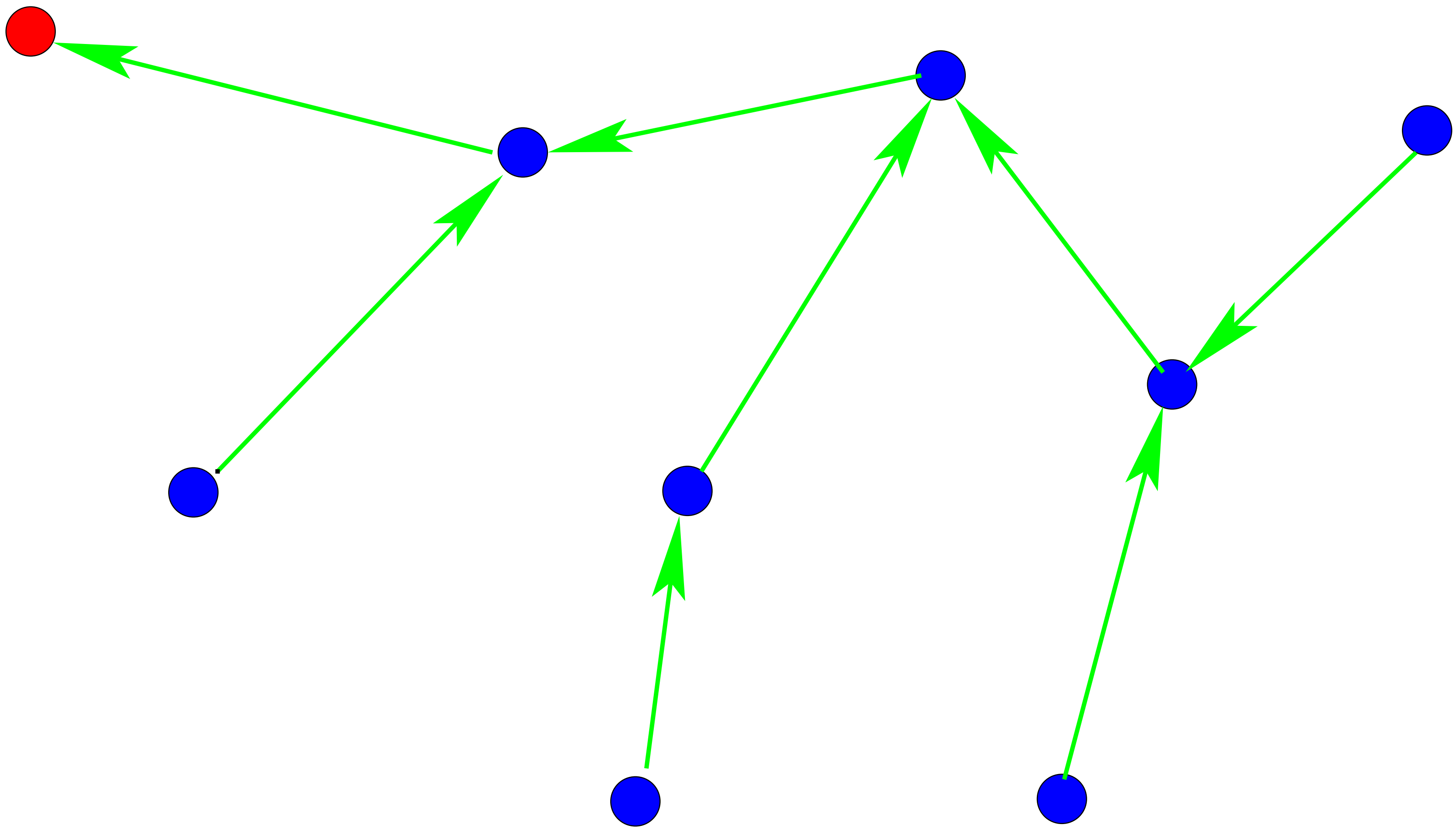












Summary of Development Process

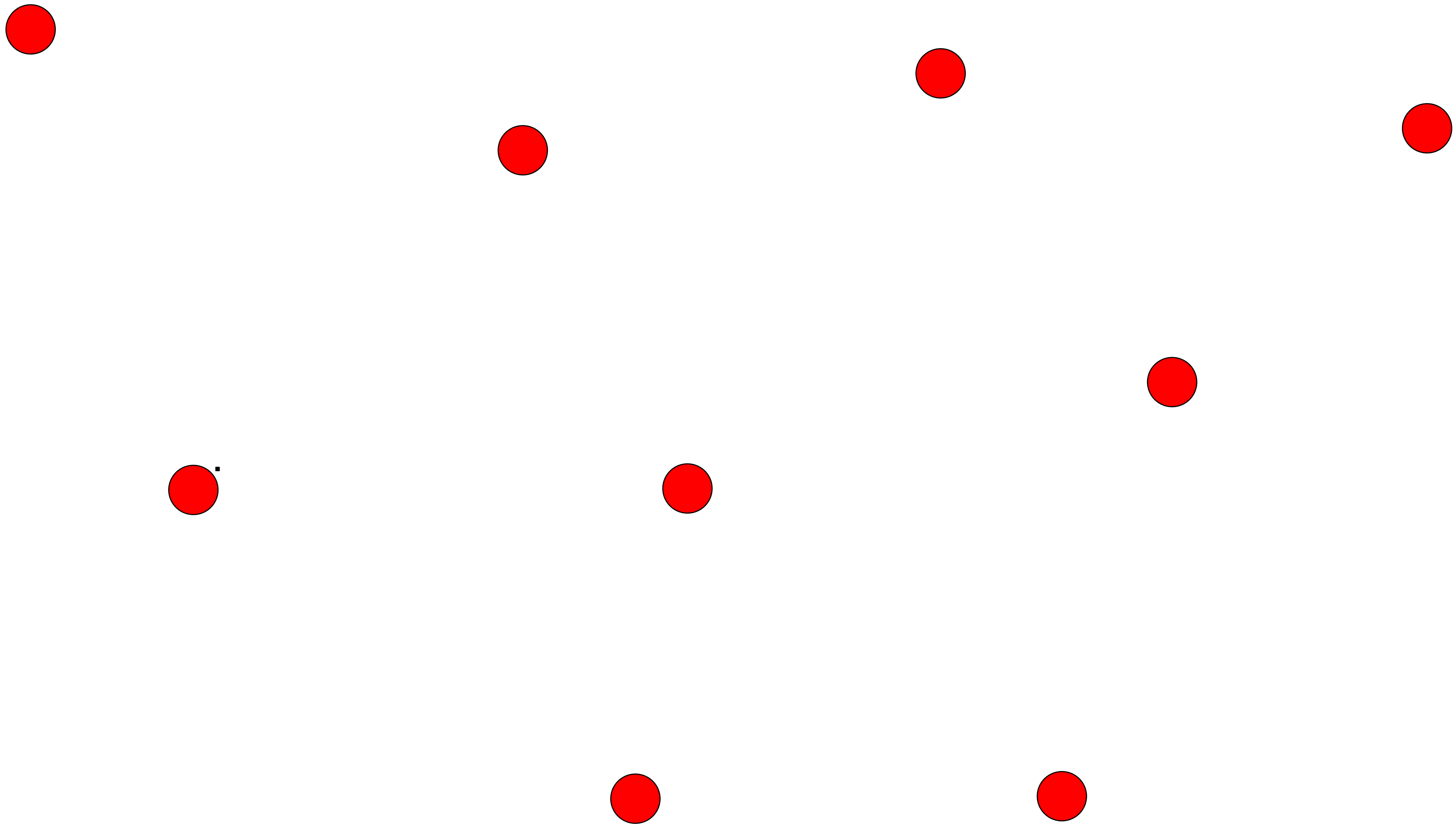
- Formal definition and properties of the network

- A one-shot abstract model of the protocol

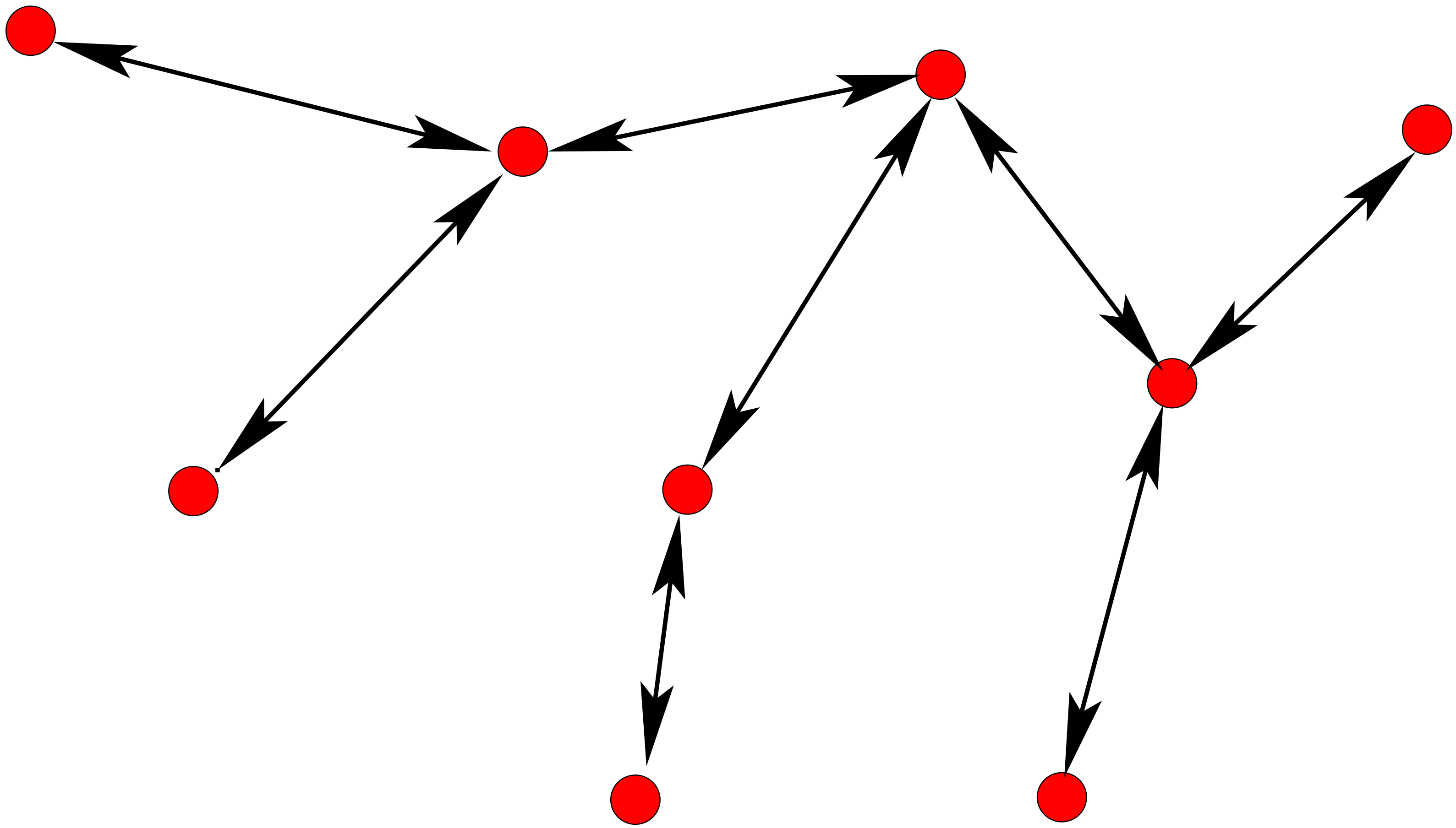
- Presenting a (still abstract) loop-like centralized solution

- Introducing message passing between the nodes (delays)

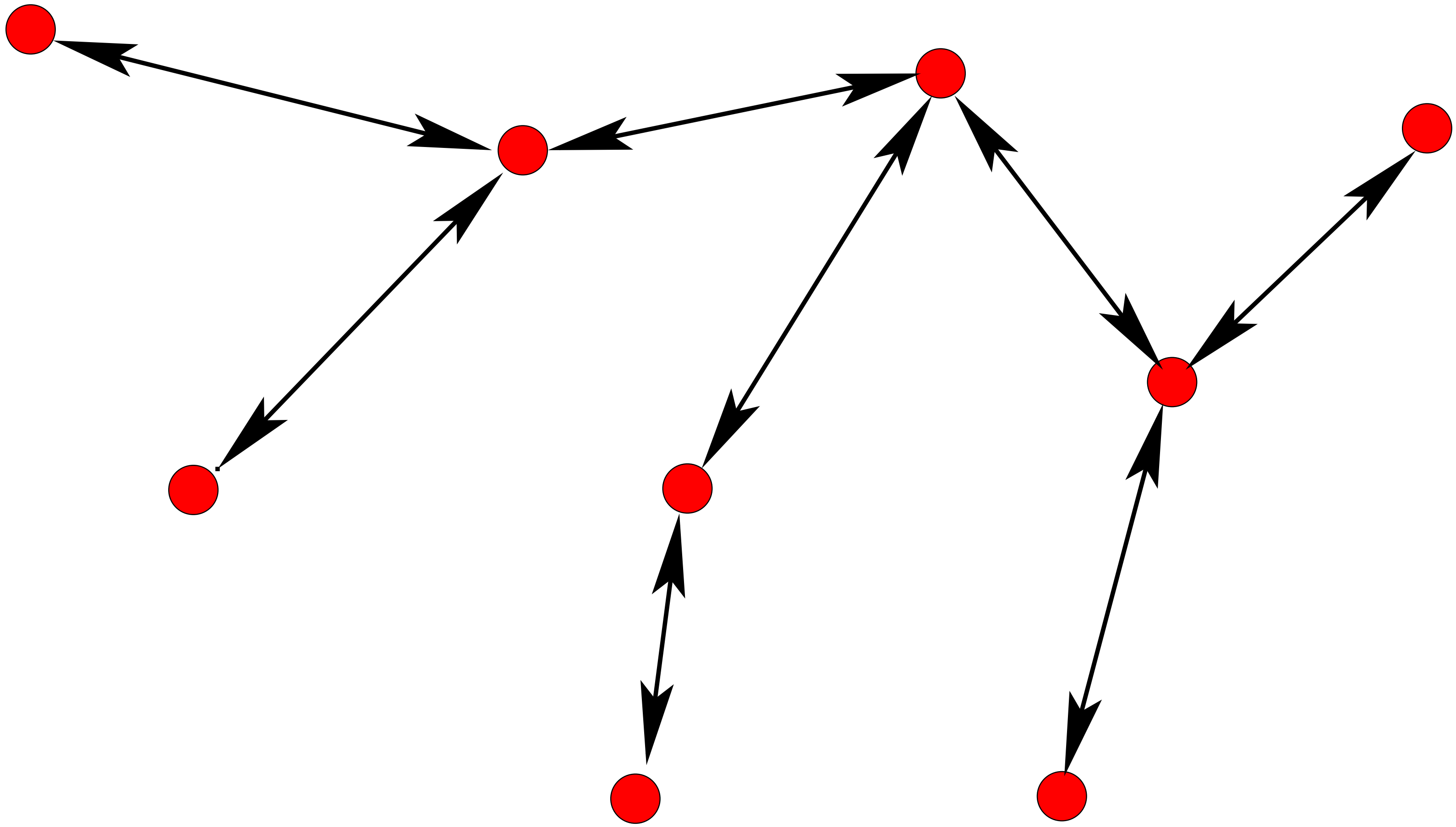
- Modifying the data structure in order to distribute the protocol



Let ND be a set of nodes (with at least 2 nodes)



Let gr be a graph built and defined on ND



gr is a symmetric and irreflexive graph

gr is a graph built on ND $\quad gr \subseteq ND \times ND$

gr is a graph built on ND $\qquad gr \subseteq ND \times ND$

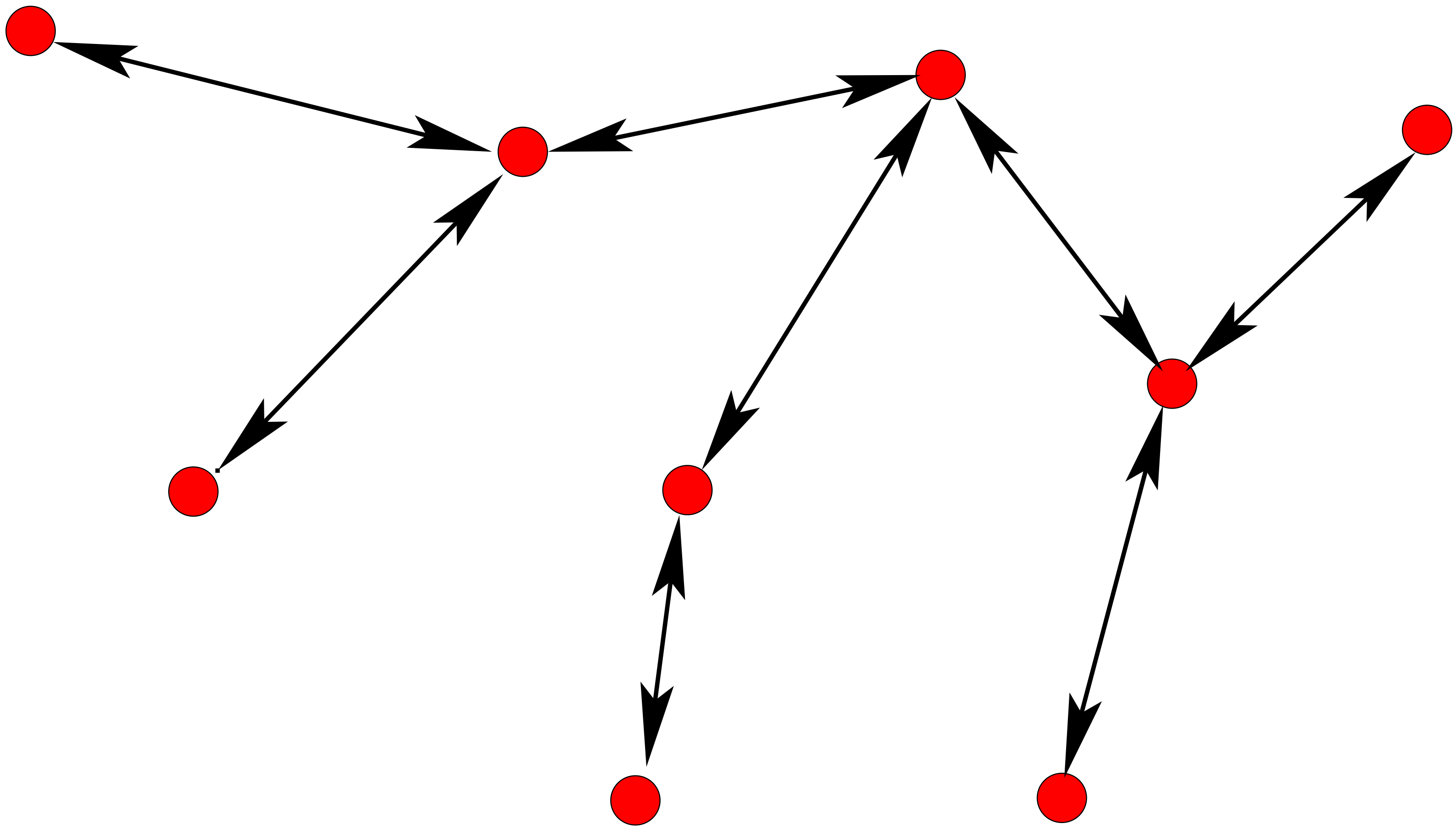
gr is defined on ND $\qquad \mathsf{dom}\,(gr) = ND$

gr is a graph built on ND $\qquad gr \subseteq ND \times ND$

gr is defined on ND $\qquad \mathsf{dom}\,(gr) = ND$

gr is symmetric $\qquad gr = gr^{-1}$

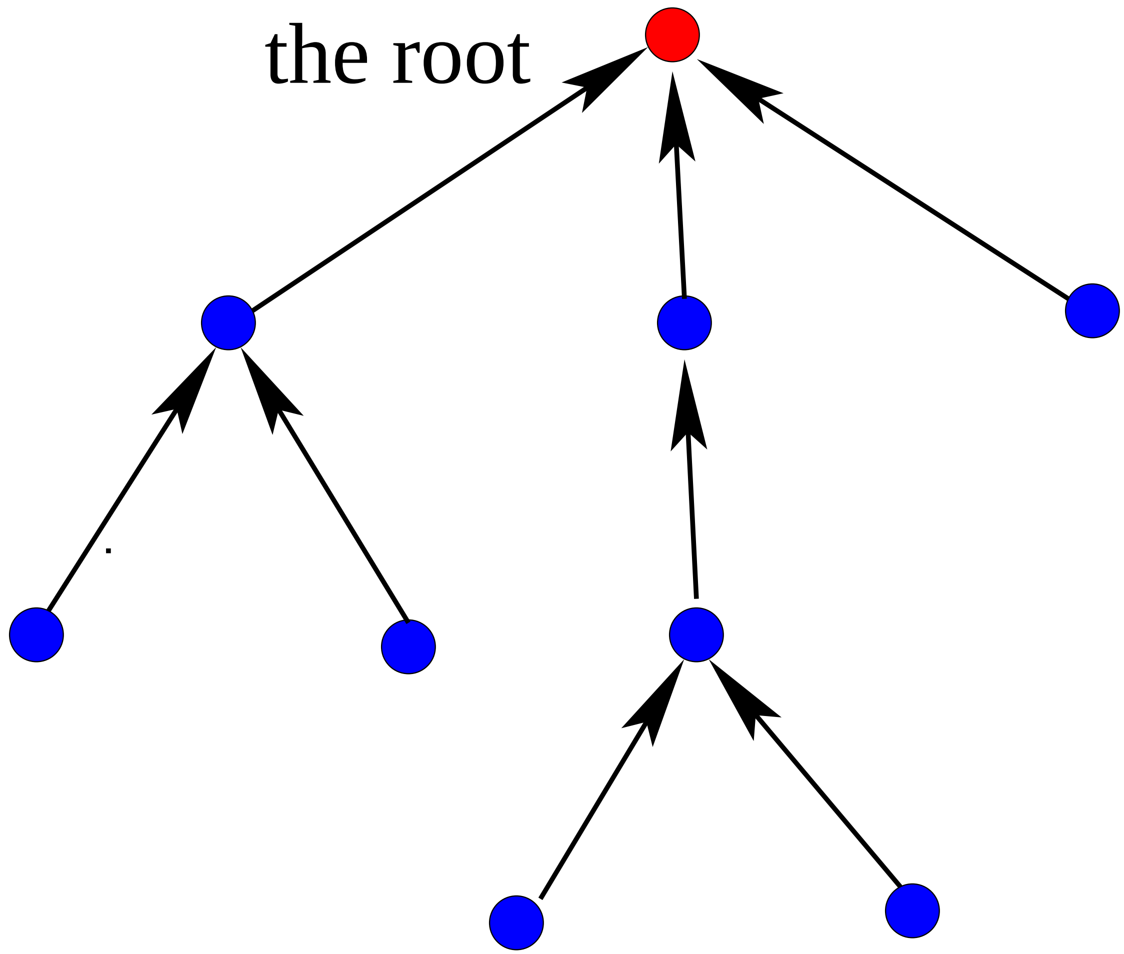
gr is a graph built on ND	$gr \subseteq ND \times ND$
gr is defined on ND	$\mathsf{dom}\,(gr) = ND$
gr is symmetric	$gr = gr^{-1}$
gr is irreflexive	$\mathsf{id}\,(ND) \cap gr = \emptyset$



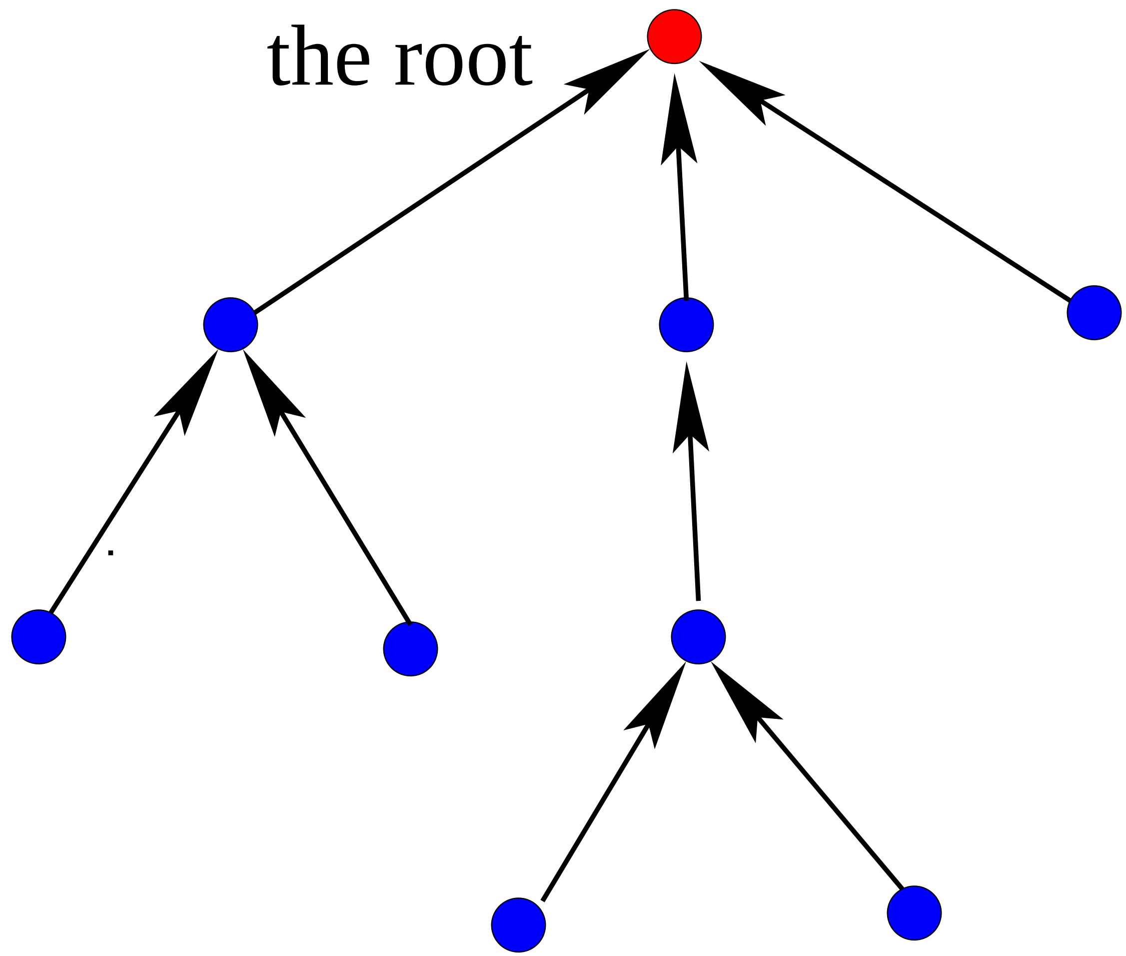
gr is connected and acyclic

A Little Detour Through Trees

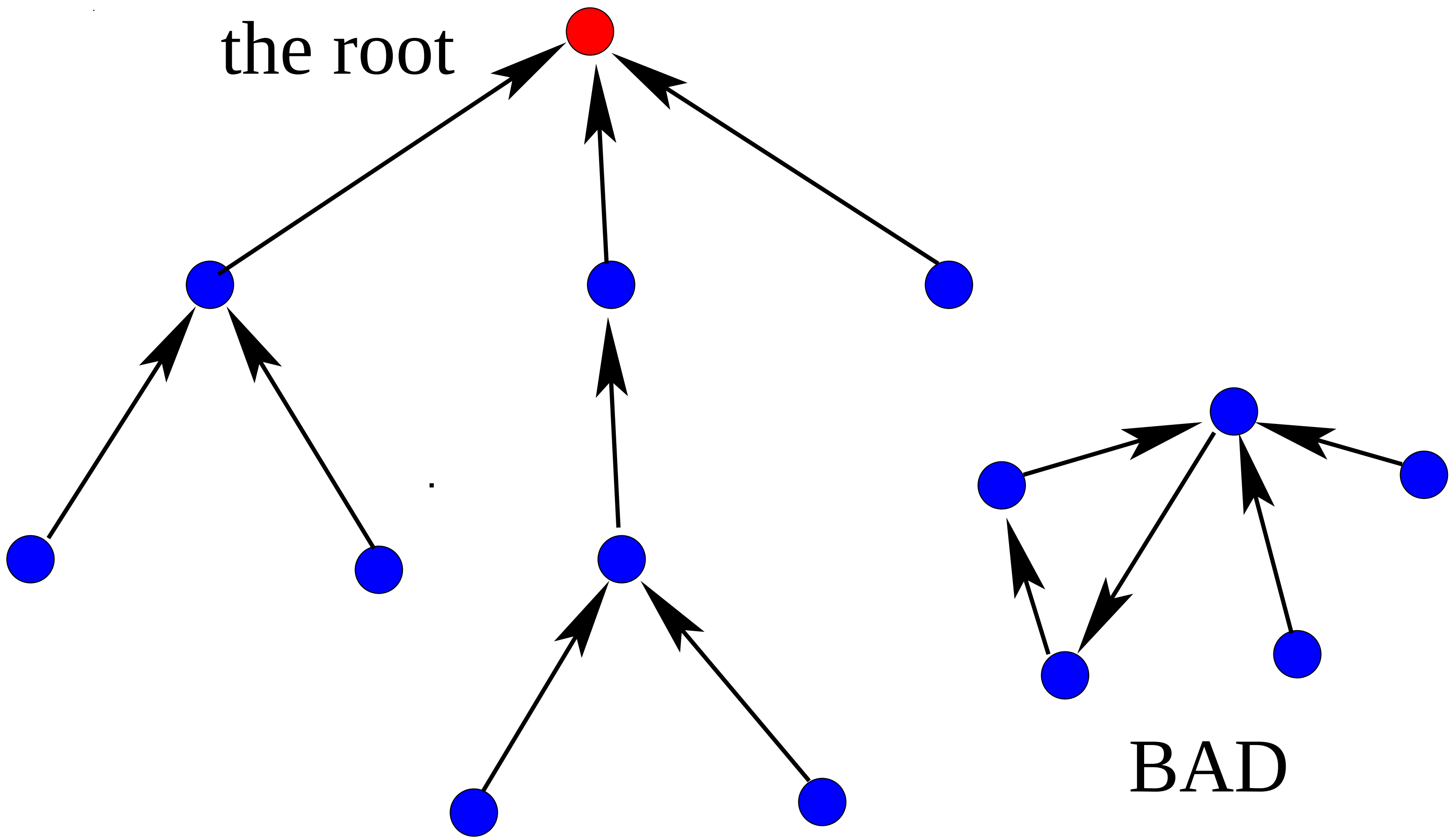
- A tree is a special graph
- A tree has a root
- A tree has a, so-called, father function
- A tree is acyclic
- A tree is connected from the root



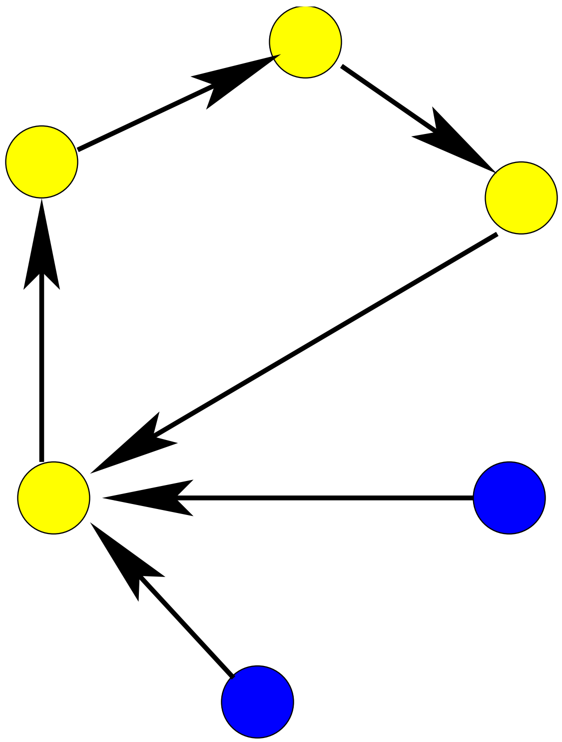
A tree t built on a set of nodes



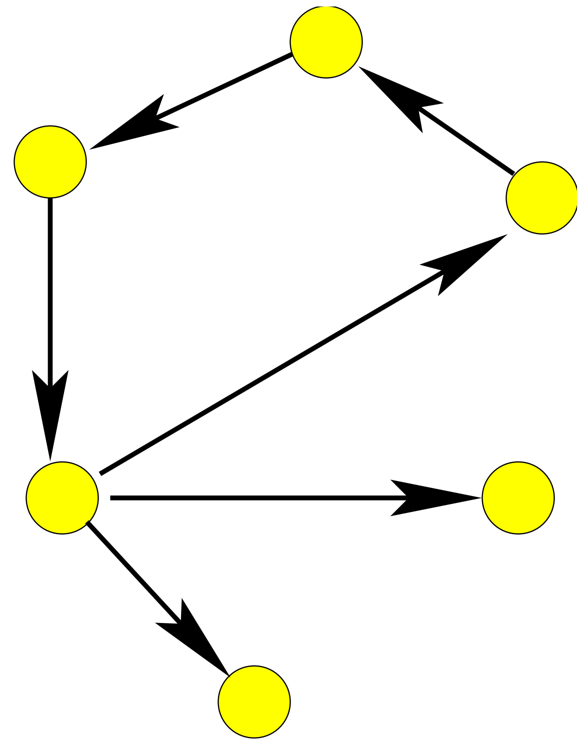
t is a function defined on ND except at the root



Avoidind cycles



A cycle



Its inverse image

The nodes of a cycle are included in their inverse image

- Given
    - a set ND
    - a subset p of ND
    - a binary relation t built on ND
- The inverse image of p under t is denoted by $t^{-1}[p]$

$$t^{-1}[p] \;\widehat{=}\; \{\, x \mid x \in ND \;\wedge\; \exists y \cdot (\, y \in p \;\wedge\; (x, y) \in t\,)\,\}$$

- When t is a partial function, this reduces to

$$\{\, x \mid x \in \mathsf{dom}\,(t) \;\wedge\; t(x) \in p \,\}$$

- If p is included in its inverse image, we have then:

$$\forall x \cdot (x \in p \Rightarrow x \in \mathsf{dom}\,(t) \ \wedge \ t(x) \in p)$$

- Notice that the empty set enjoys this property

$$\emptyset \subseteq t^{-1}[\emptyset]$$

- The property of having no cycle is thus equivalent to:

The only subset p of ND s.t. $p \subseteq t^{-1}[p]$ is EMPTY

$$\forall p \cdot \begin{pmatrix} \quad p \subseteq ND \;\; \wedge \\ \quad p \subseteq t^{-1}[p] \\ \Rightarrow \\ \quad p = \emptyset \end{pmatrix}$$

The predicate tree (r, t)

The predicate tree (r, t)

r is a member of ND $\quad r \in ND$

The predicate tree (r, t)

r is a member of ND $\quad r \in ND$

t is a function $\quad t \in ND - \{r\} \longrightarrow ND$

The predicate tree (r, t)

r is a member of ND $\quad r \in ND$

t is a function $\quad t \in ND - \{r\} \rightarrow ND$

t is acyclic
$$\forall p \cdot \left(\begin{array}{l} \quad p \subseteq ND \;\; \wedge \\ \quad p \subseteq t^{-1}[p] \\ \Rightarrow \\ \quad p = \emptyset \end{array} \right)$$

t is acyclic: equivalent formulations

$$\forall p \cdot \begin{pmatrix} \quad p \subseteq ND \;\; \wedge \\ \quad p \subseteq t^{-1}[p] \\ \Rightarrow \\ \quad p = \emptyset \end{pmatrix} \quad \Leftrightarrow \quad \forall q \cdot \begin{pmatrix} \quad q \subseteq ND \;\; \wedge \\ \quad r \in q \;\; \wedge \\ \quad t^{-1}[q] \subseteq q \\ \Rightarrow \\ \quad ND \subseteq q \end{pmatrix}$$

This gives an Induction Rule

$$\forall q \cdot \left(\begin{array}{l} \quad q \subseteq ND \;\; \wedge \\ \quad r \in q \;\; \wedge \\ \quad \forall x \cdot (\, x \in ND - \{r\} \;\wedge\; t(x) \in q \Rightarrow x \in q \,) \\ \Rightarrow \\ \quad ND \subseteq q \end{array} \right)$$

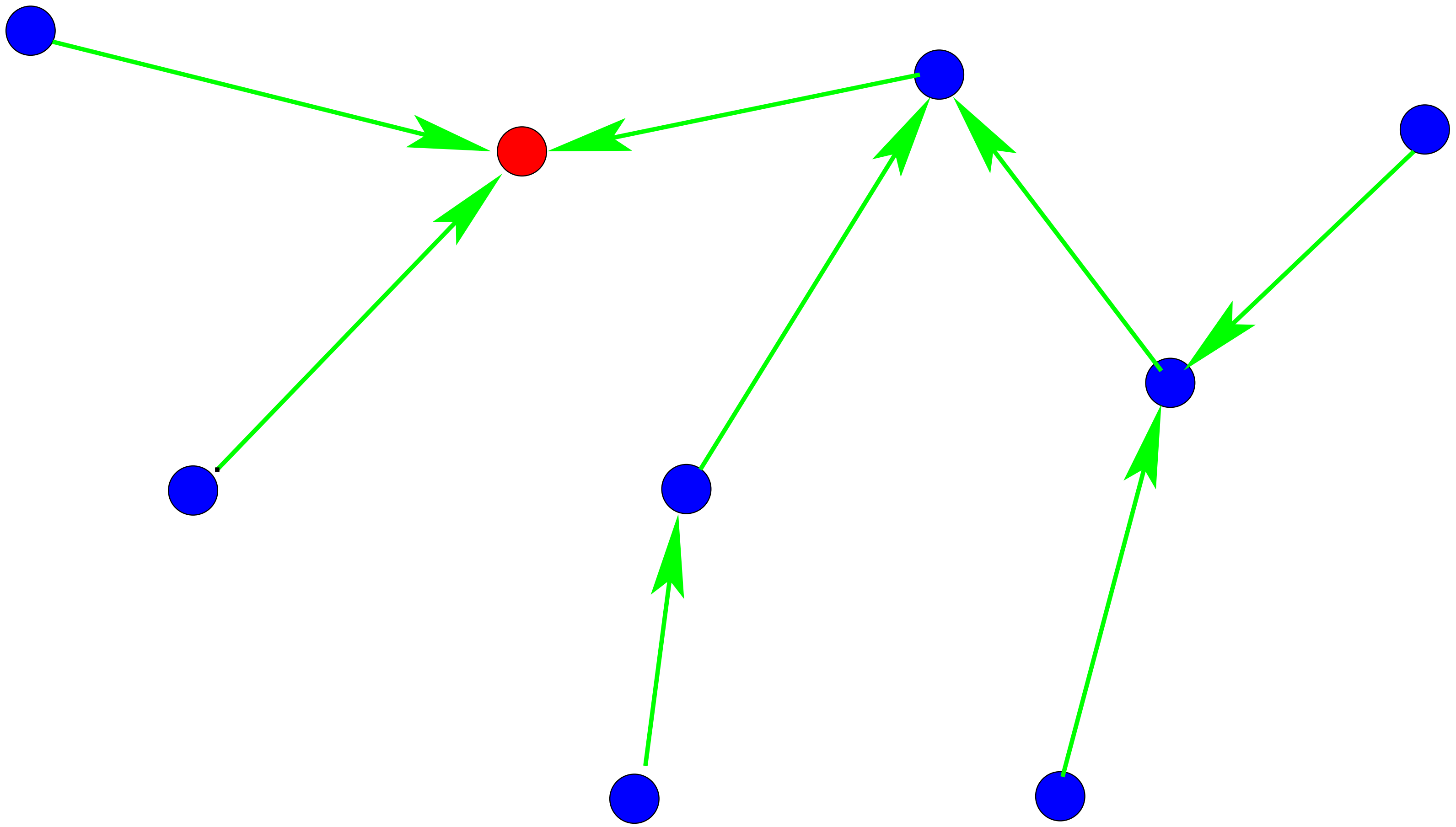
The predicate tree (r, t)

r is a member of ND $\quad r \in ND$

t is a function $\quad t \in ND - \{r\} \rightarrow ND$

t is acyclic

$$\forall q \cdot \begin{pmatrix} q \subseteq ND \;\wedge \\ r \in q \;\wedge \\ t^{-1}[q] \subseteq q \\ \Rightarrow \\ ND \subseteq q \end{pmatrix}$$



A spanning tree t of the graph gr

The predicate spanning (r, t, gr)

r, t is a tree tree (r, t)

t is included in gr $t \subseteq gr$

The graph gr is connected and acyclic (1)

- Defining a relation fn linking a node to the possible spanning trees of gr having that node as a root:

$$fn \;\subseteq\; ND \times (ND \mathrel{+\!\!\!\rightarrow} ND)$$

$$\forall (r,t) \cdot \left(\begin{array}{l} \quad r \in ND \;\; \land \\ \quad t \in ND \mathrel{+\!\!\!\rightarrow} ND \\ \Rightarrow \\ \quad (r,t) \in fn \;\Leftrightarrow\; \text{spanning}\,(r,t,gr) \end{array} \right)$$

The graph gr is connected and acyclic (2)

Totality of relation $fn \Rightarrow$ Connectivity of gr

Functionality of relation $fn \Rightarrow$ Acyclicity of gr

Summary of constants gr and fn

$$gr \subseteq ND \times ND$$

$$\mathrm{dom}\,(gr) = ND$$

$$gr = gr^{-1}$$

$$\mathrm{id}\,(ND) \cap gr = \emptyset$$

$$fn \in ND \to (ND \twoheadrightarrow ND)$$

$$\forall (r,t) \cdot \begin{pmatrix} \quad r \in ND \;\; \land \\ \quad t \in ND \twoheadrightarrow ND \\ \Rightarrow \\ \quad t = fn(r) \;\Leftrightarrow\; \mathsf{spanning}\,(r, t, gr) \end{pmatrix}$$

Election in One Shot: Building a Spanning Tree

- Variables rt and ts

$$rt \in ND$$
$$ts \in ND \leftrightarrow ND$$

elect $\widehat{=}$

BEGIN

rt, ts : spanning (rt, ts, gr)

END

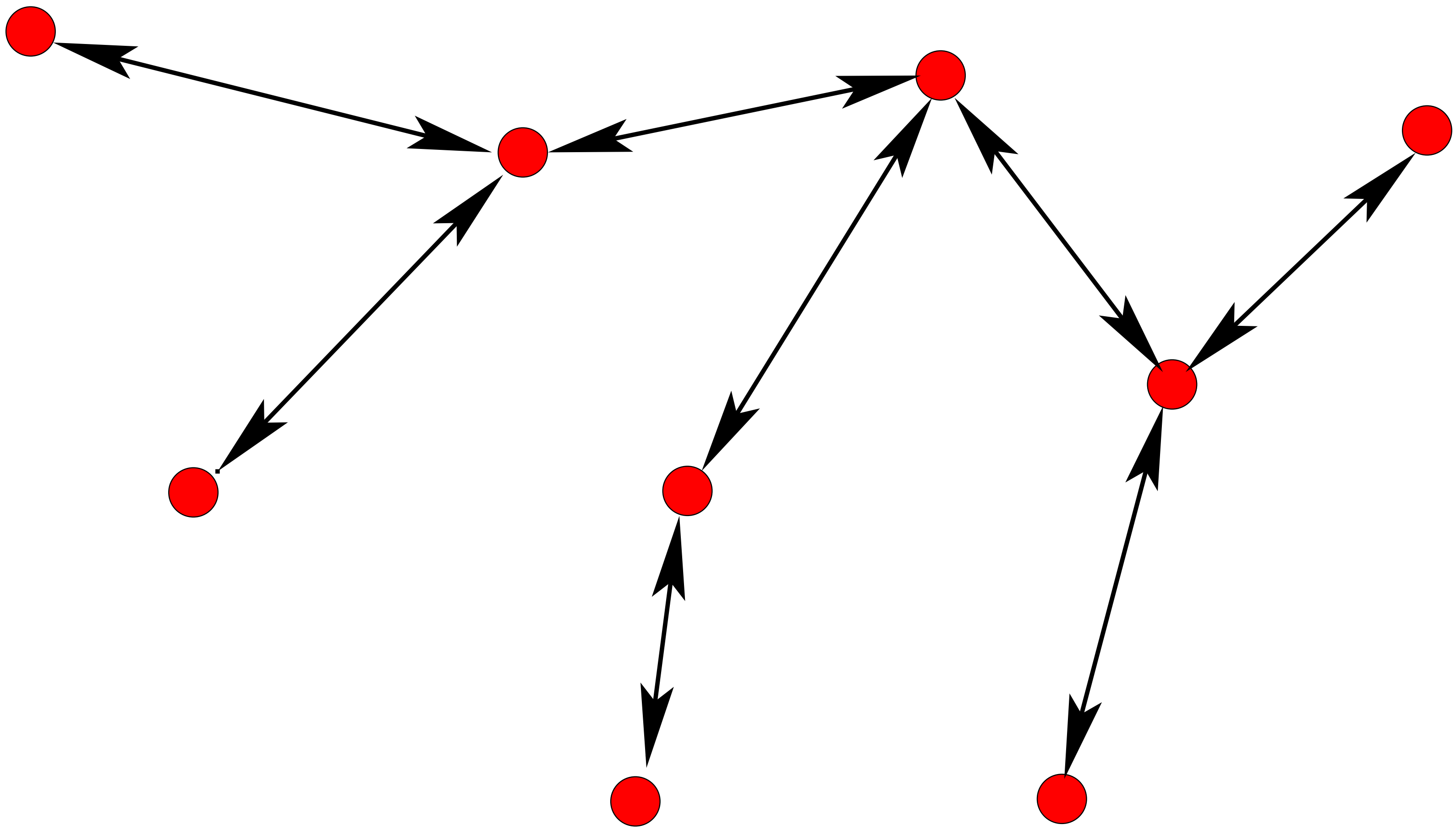
First Refinement (1)

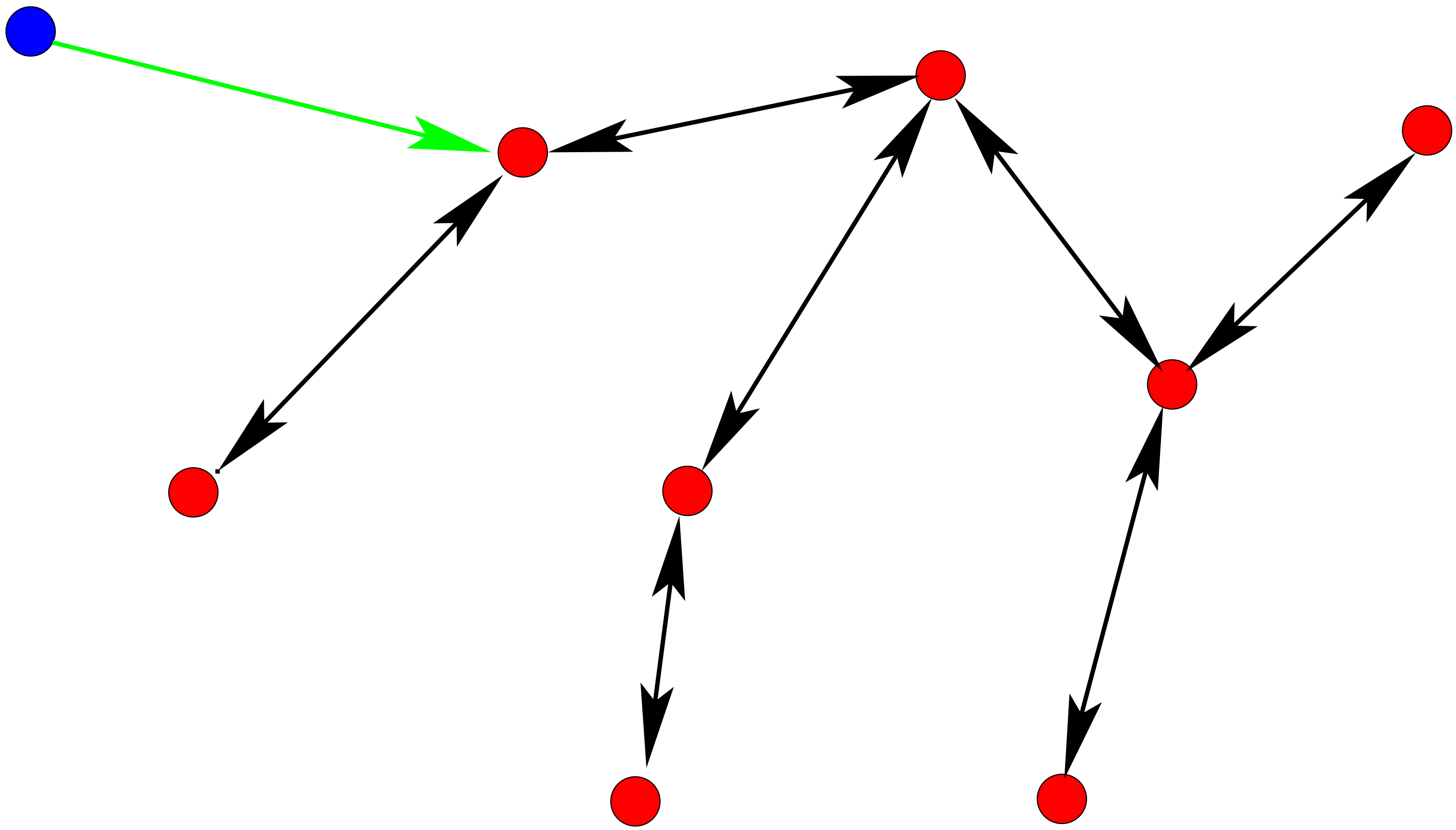
- Introducing a new variable, tr, corresponding to the "tree" in construction

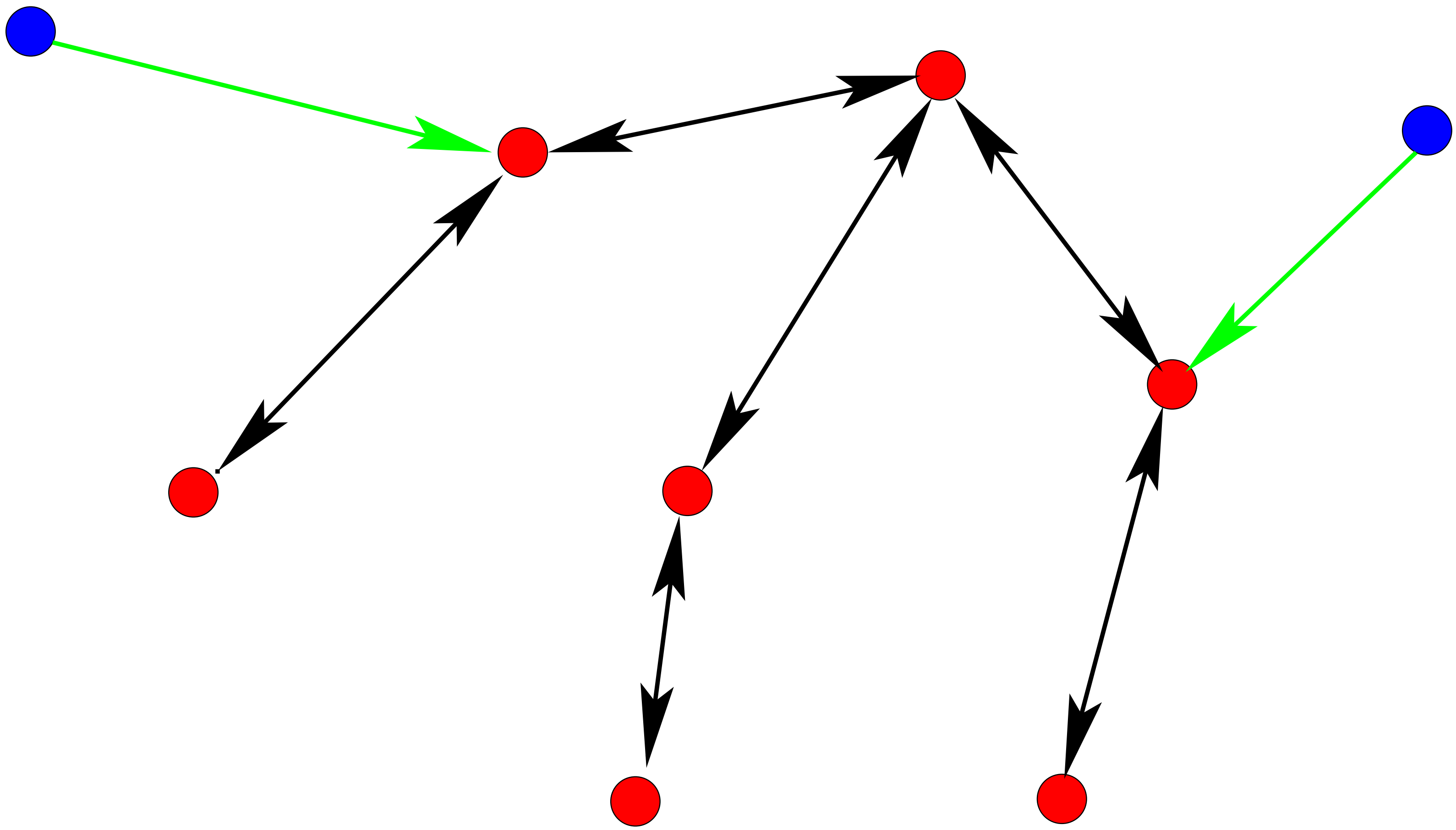
- Introducing a new event: the progression event

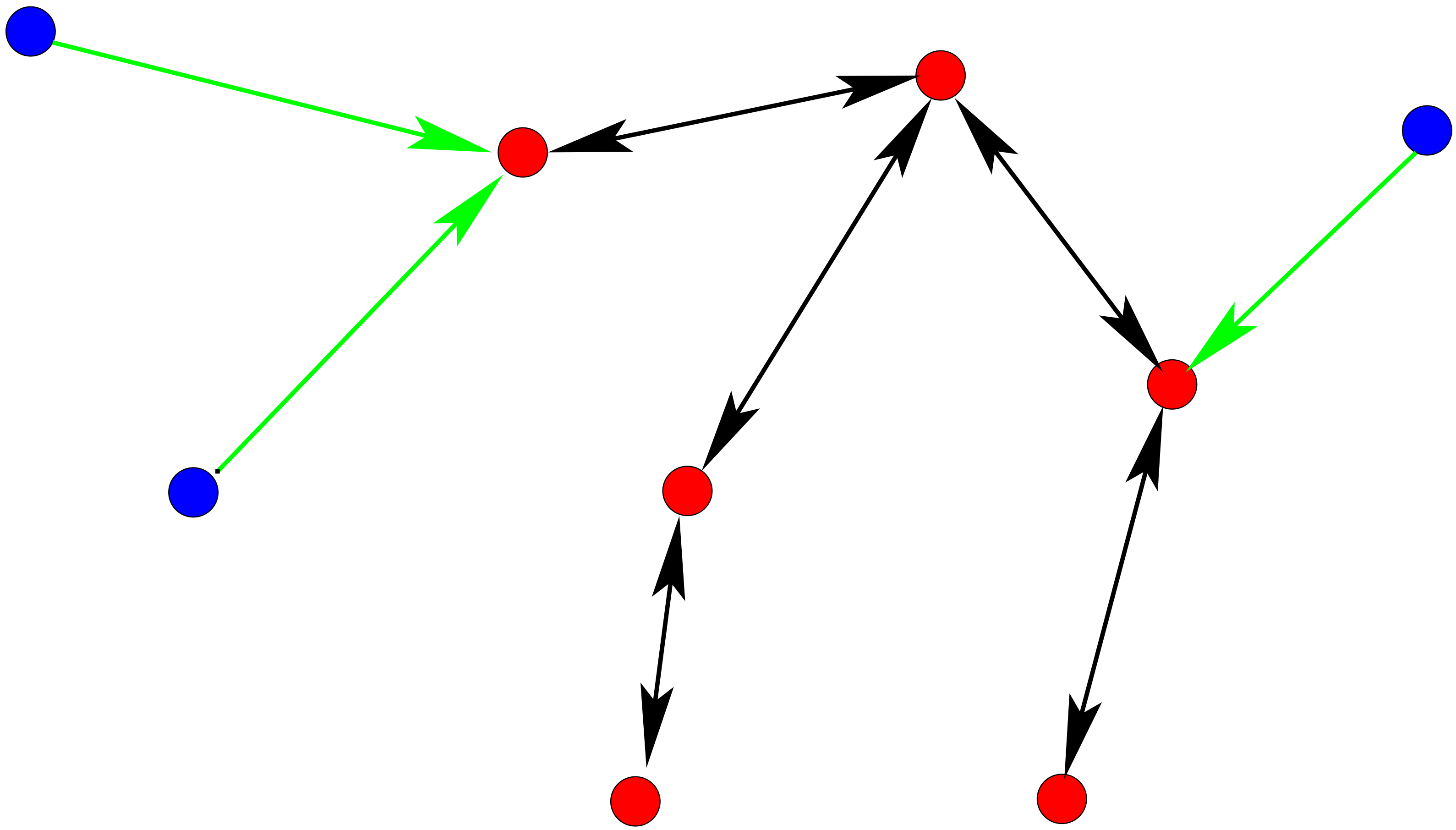
- Defining the invariant

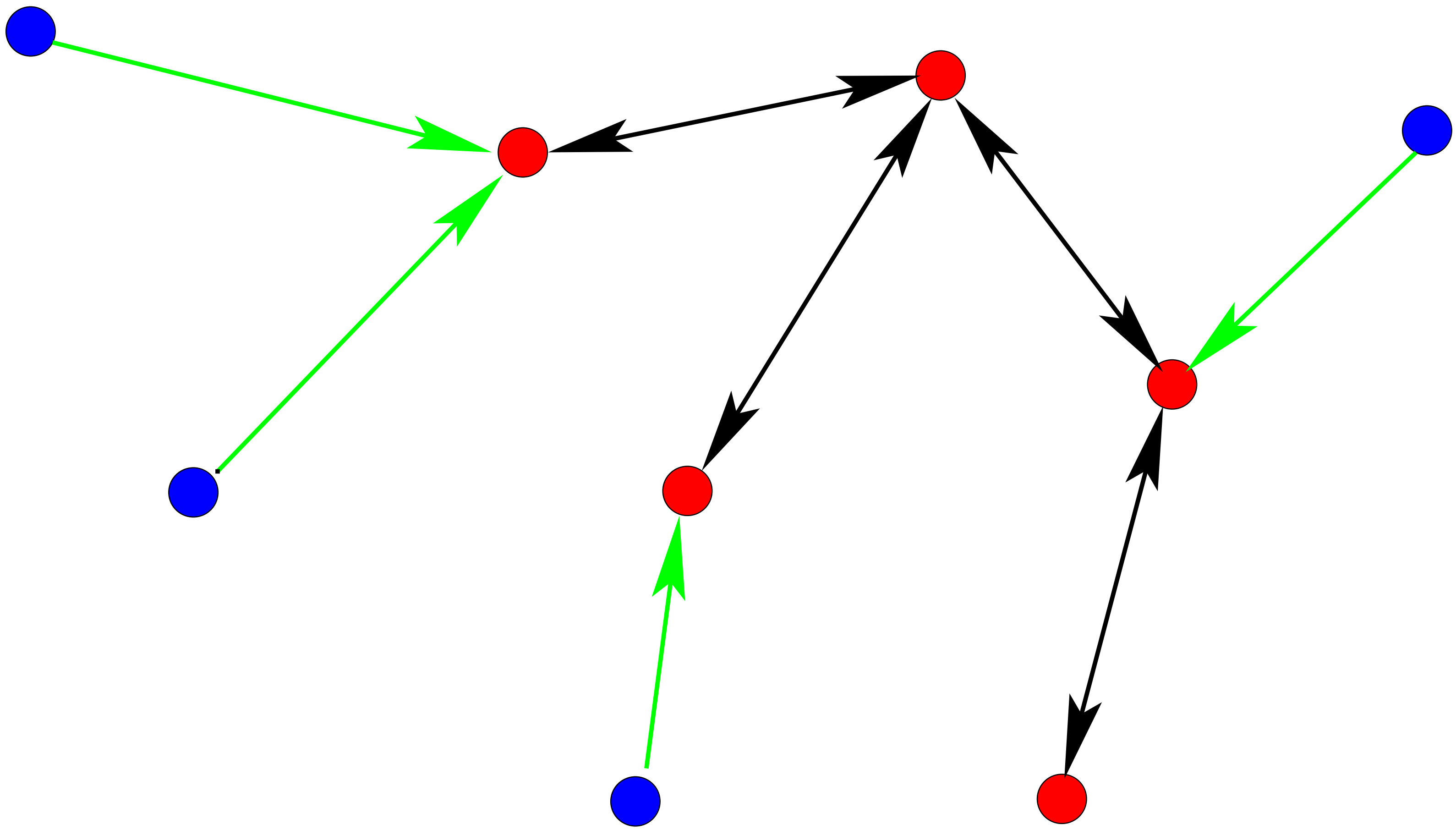
- Back to the animation : Observe the construction of the tree

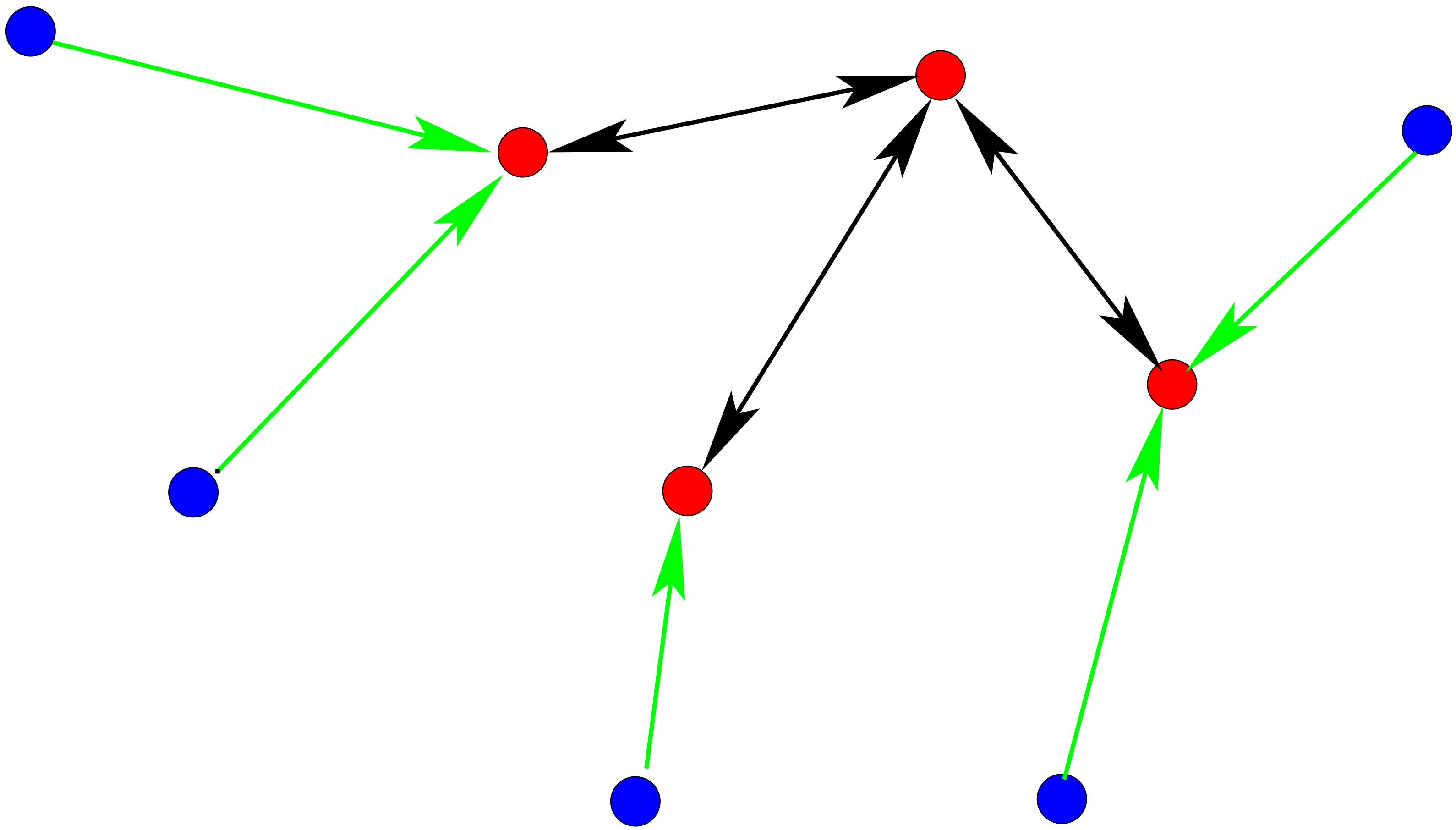


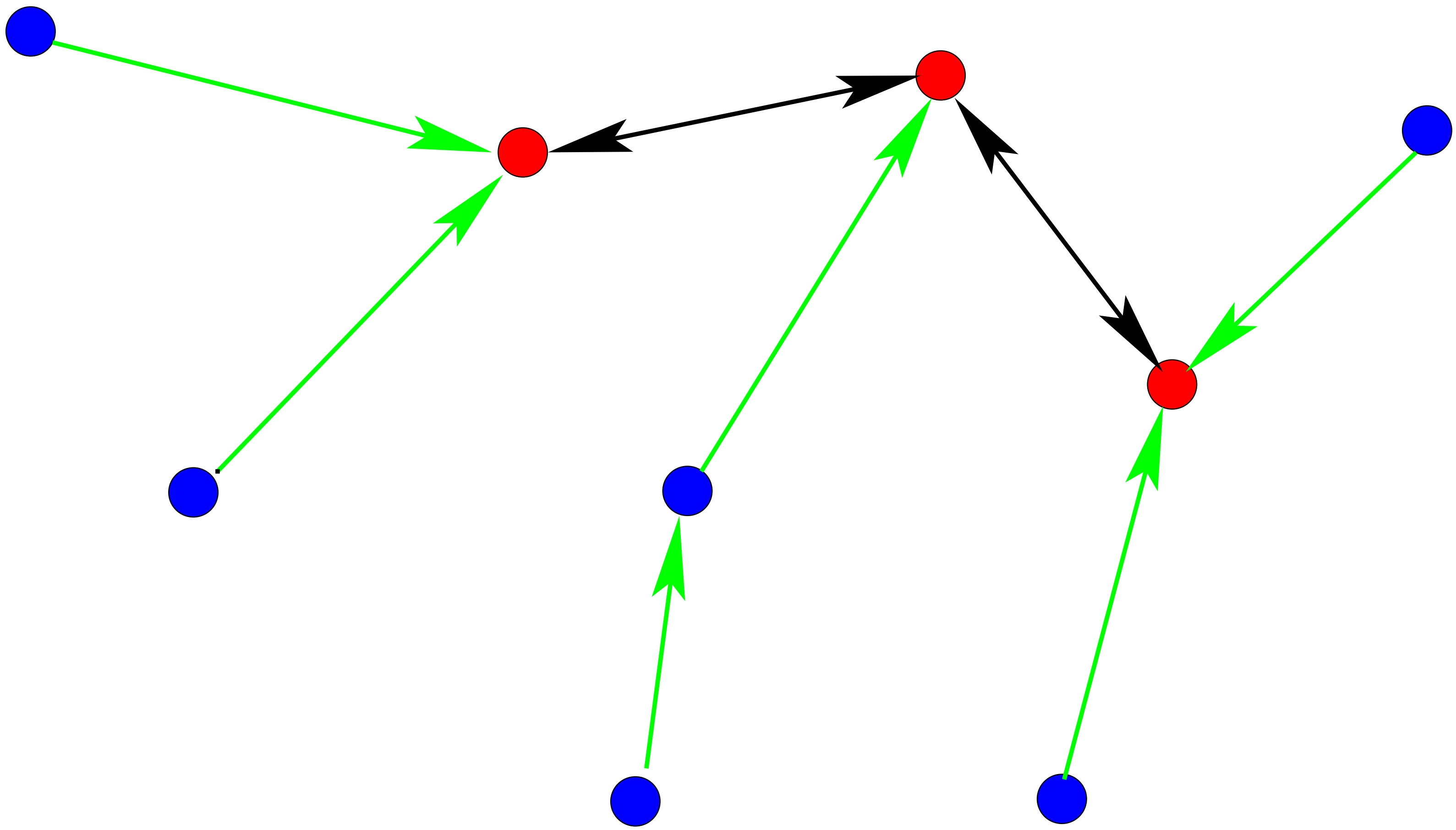


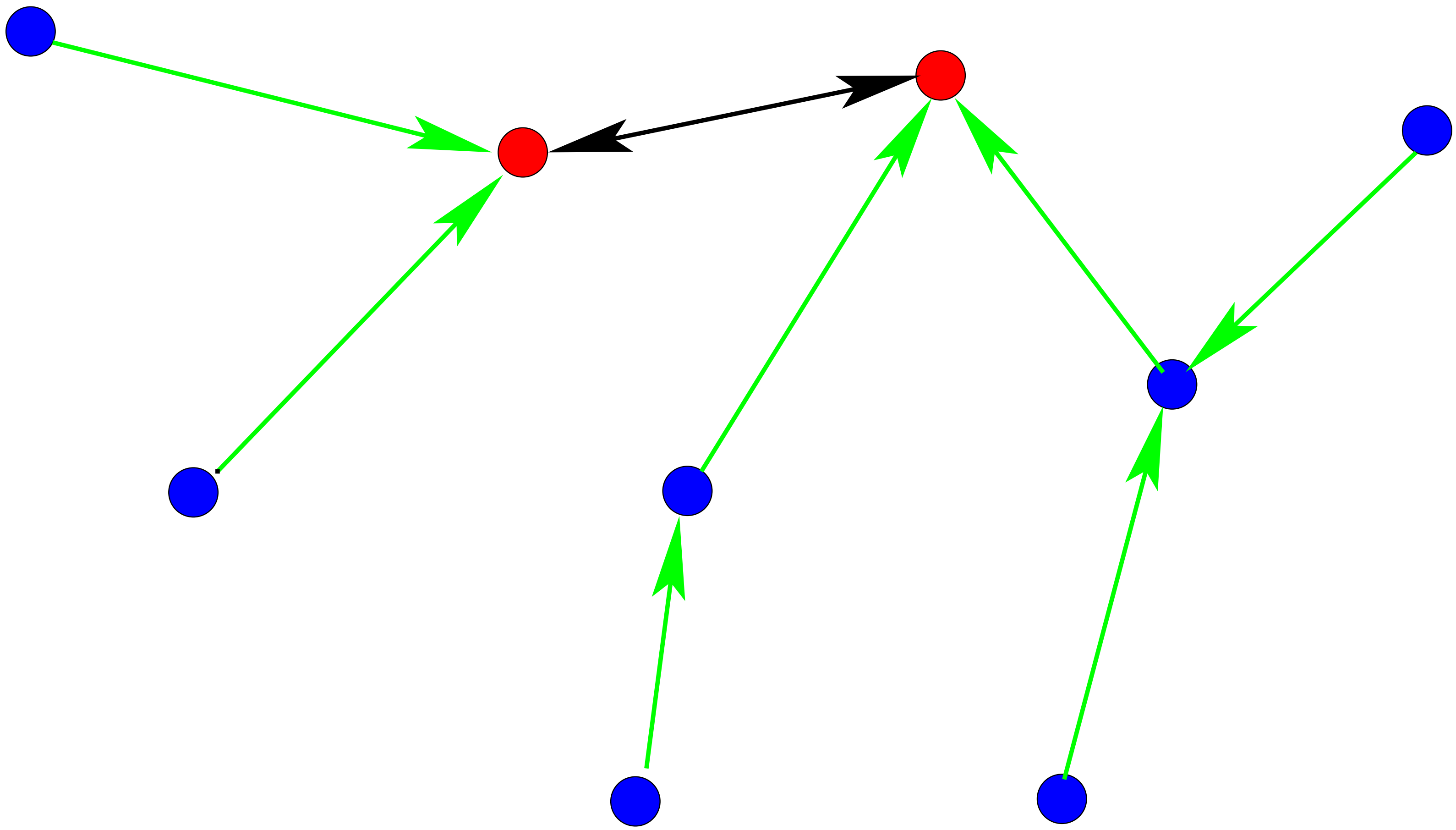


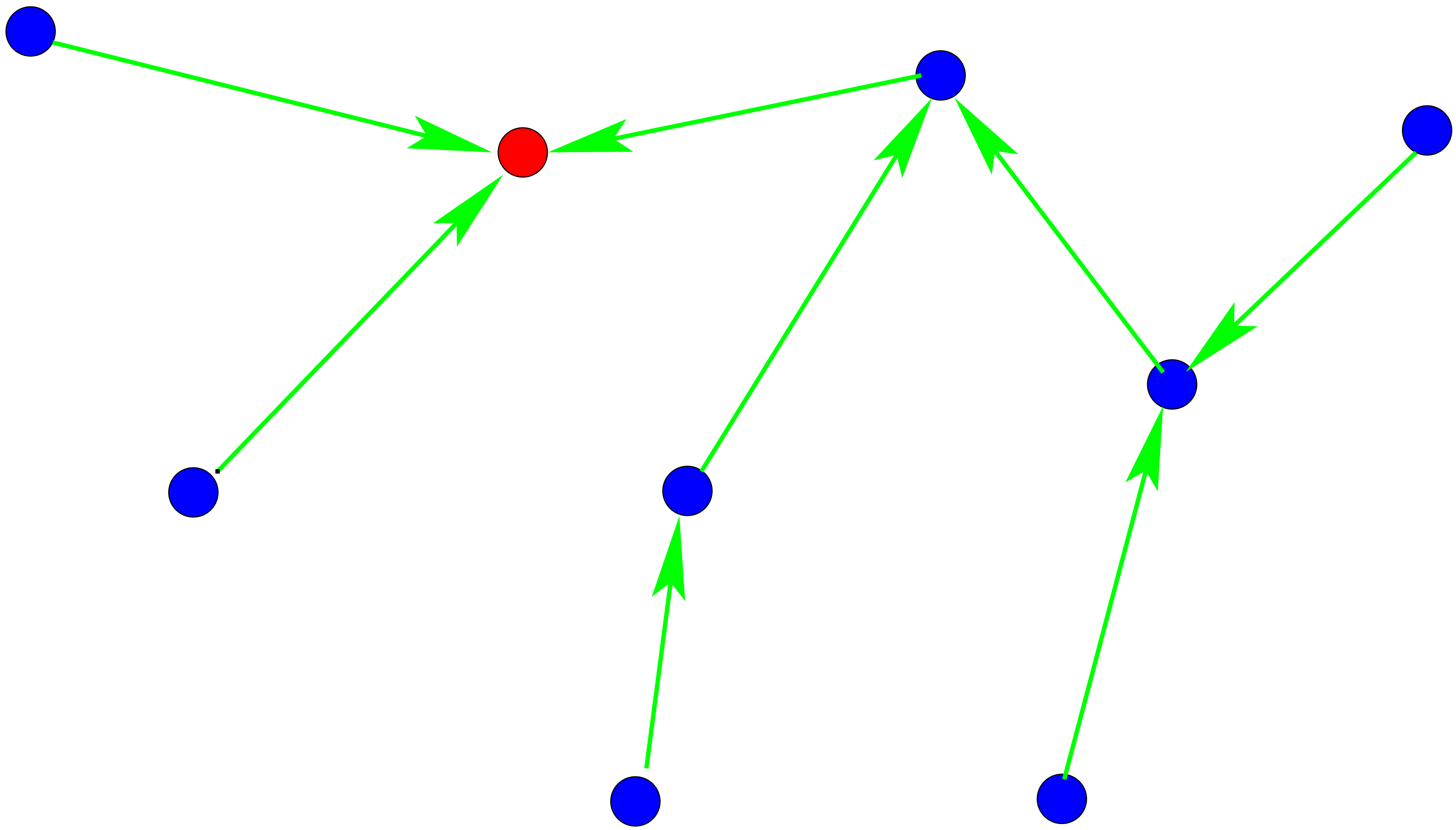












- The green arrows correspond to the tr function

- The blue nodes are the domain of tr

- The function tr is a forest (multi-tree) on nodes

- The red nodes are the roots of these trees

The predicate invariant (tr)

$$tr \in ND \mathrel{\mapstochar\rightarrow} ND$$

The predicate invariant (tr)

$$tr \in ND \leftrightarrow ND$$

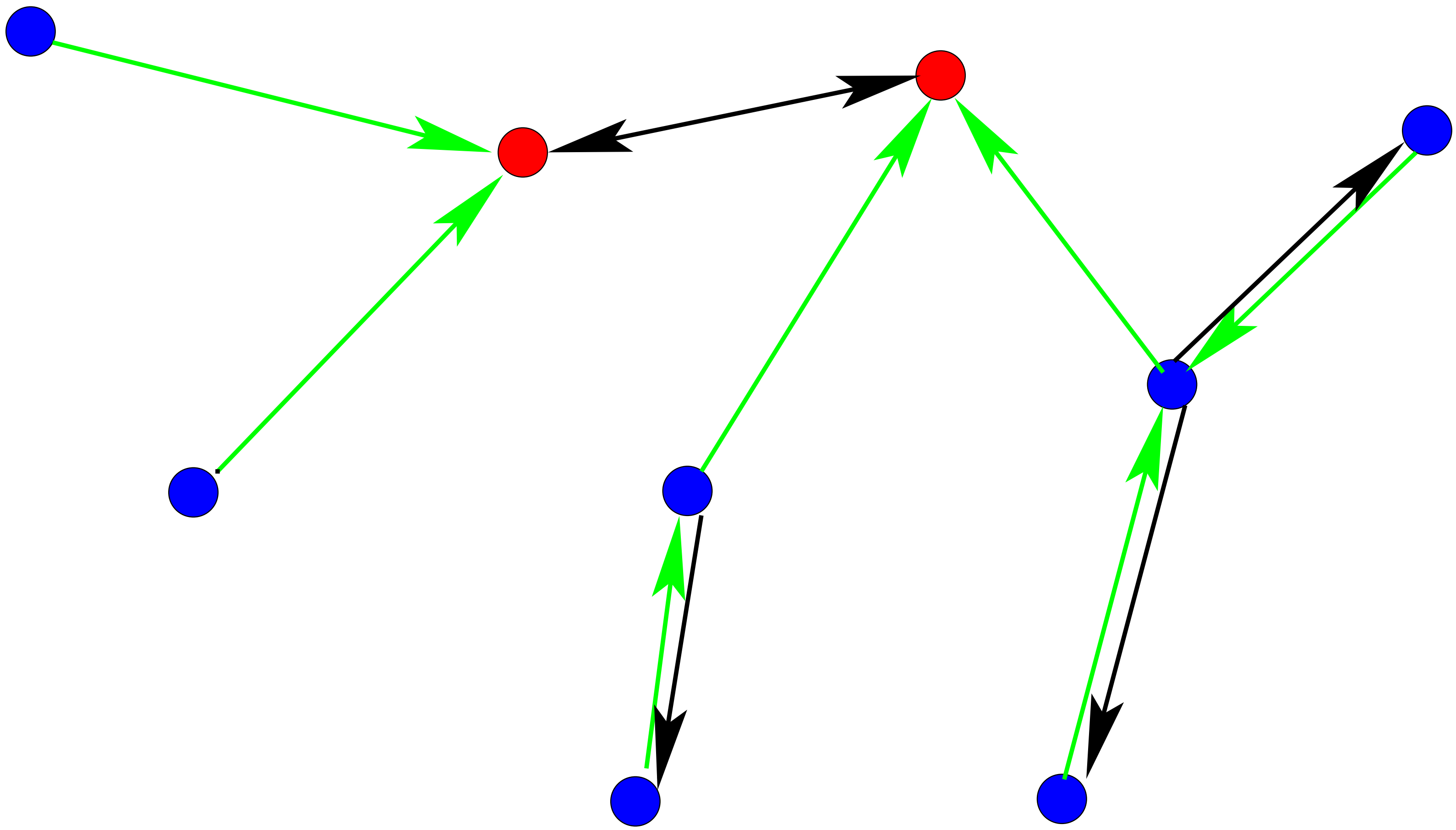
$$\forall p \cdot \left(\begin{array}{l} \quad p \subseteq ND \qquad \wedge \\ \quad ND - \mathsf{dom}\,(tr) \subseteq p \qquad \wedge \\ \quad tr^{-1}\,[p] \subseteq p \\ \Rightarrow \\ \quad ND \subseteq p \end{array} \right)$$

The predicate invariant (tr)

$$tr \in ND \leftrightarrow ND$$

$$\forall p \cdot \left(\begin{array}{l} \quad p \subseteq ND \qquad \wedge \\ \quad ND - \mathrm{dom}\,(tr) \subseteq p \qquad \wedge \\ \quad tr^{-1}\,[p] \subseteq p \\ \Rightarrow \\ \quad ND \subseteq p \end{array} \right)$$

$$\mathrm{dom}\,(tr) \lhd (tr \cup tr^{-1}) \;=\; \mathrm{dom}\,(tr) \lhd gr$$

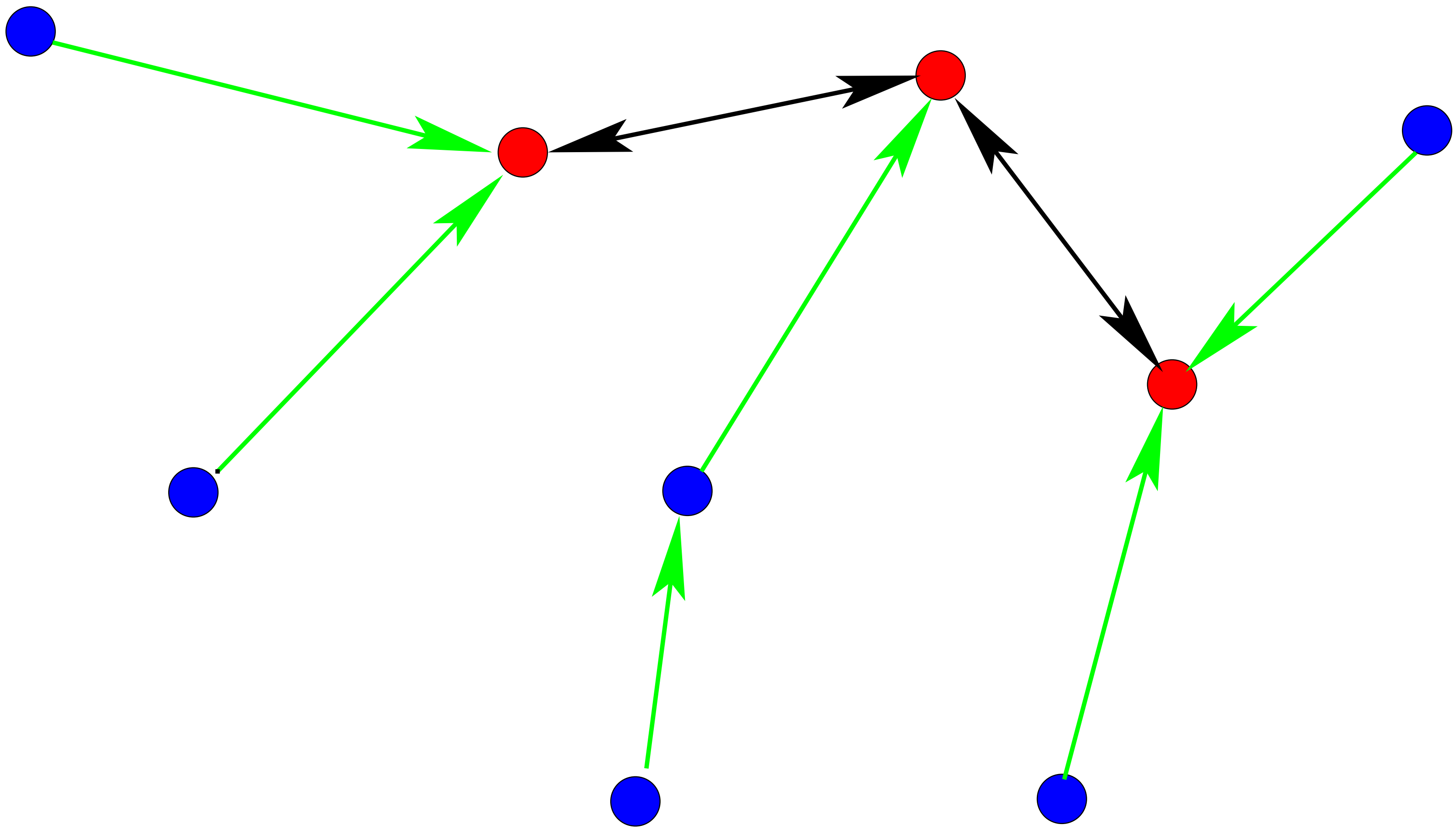


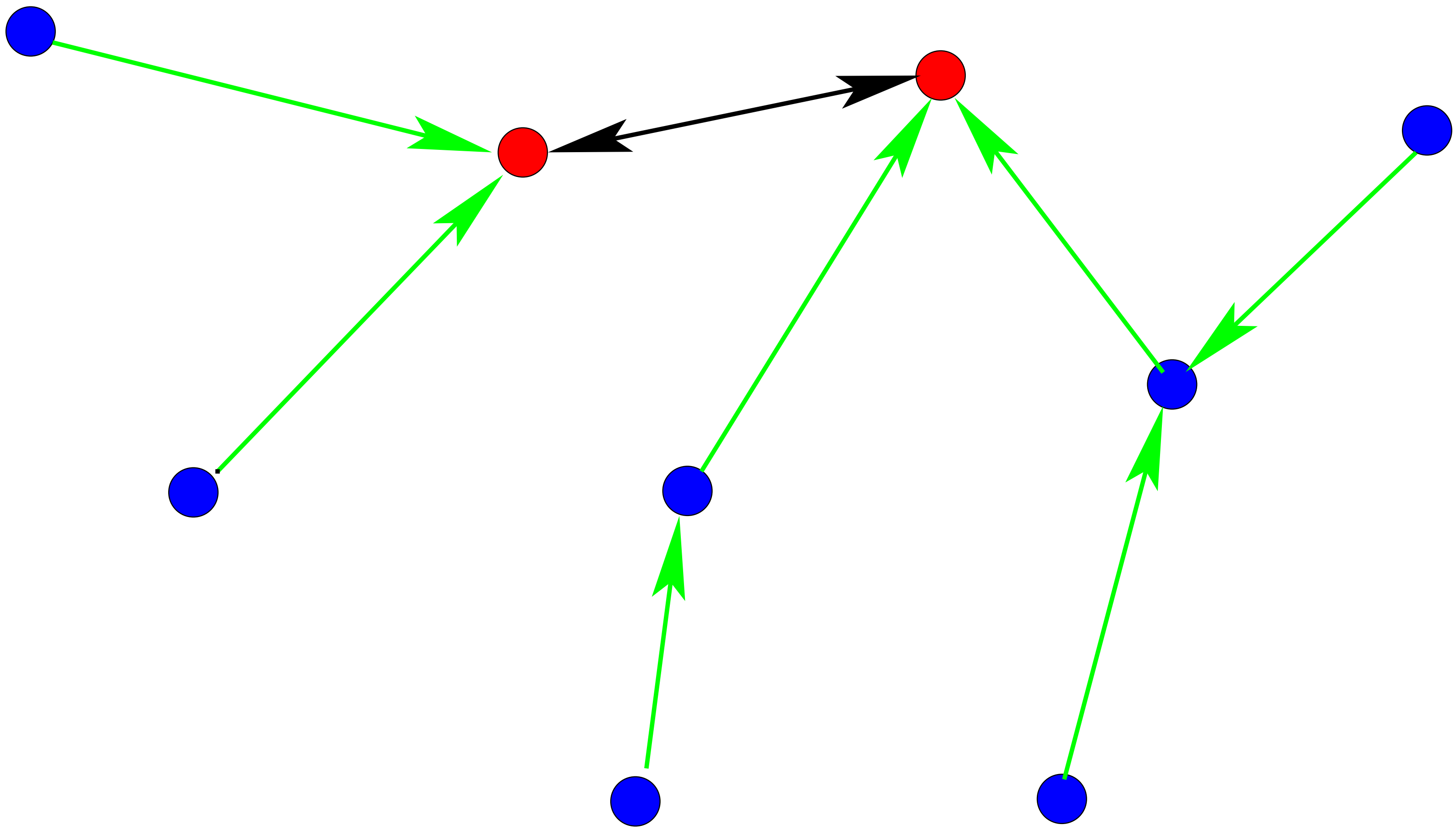
First Refinement (2)

- Introducing the new event "progress"

- Refining the abstract event "elect"

- Back to the animation : Observe the "guard" of progress





When a red node x is connected to AT MOST one other red node y then event "progress" can take place

progress $\widehat{=}$

ANY x, y **WHERE**

$x, y \in gr \ \wedge$

$x \notin \mathrm{dom}\,(tr) \ \wedge$

$y \notin \mathrm{dom}\,(tr) \ \wedge$

$gr[\{x\}] = tr^{-1}[\{x\}] \cup \{y\}$

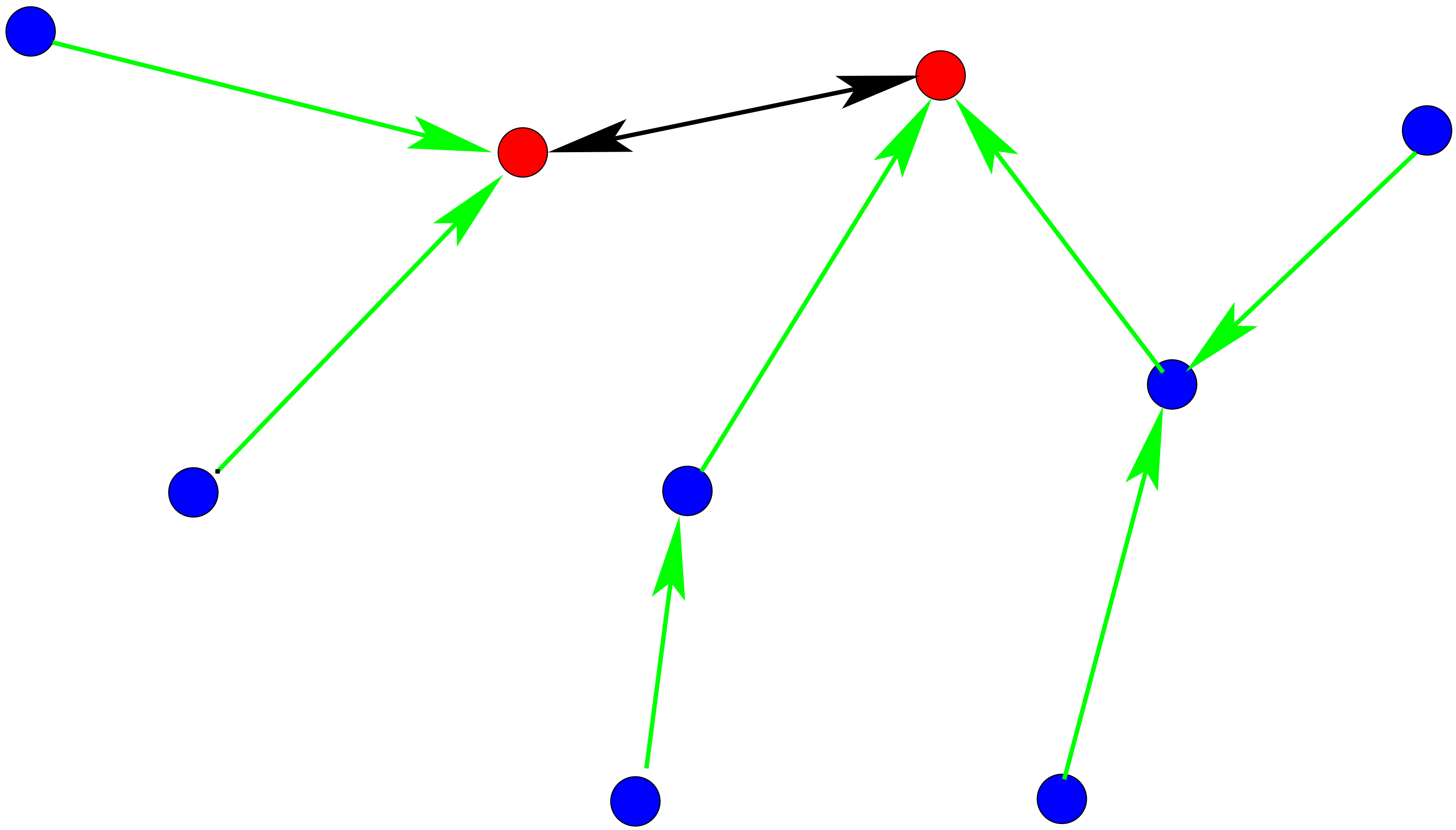
THEN

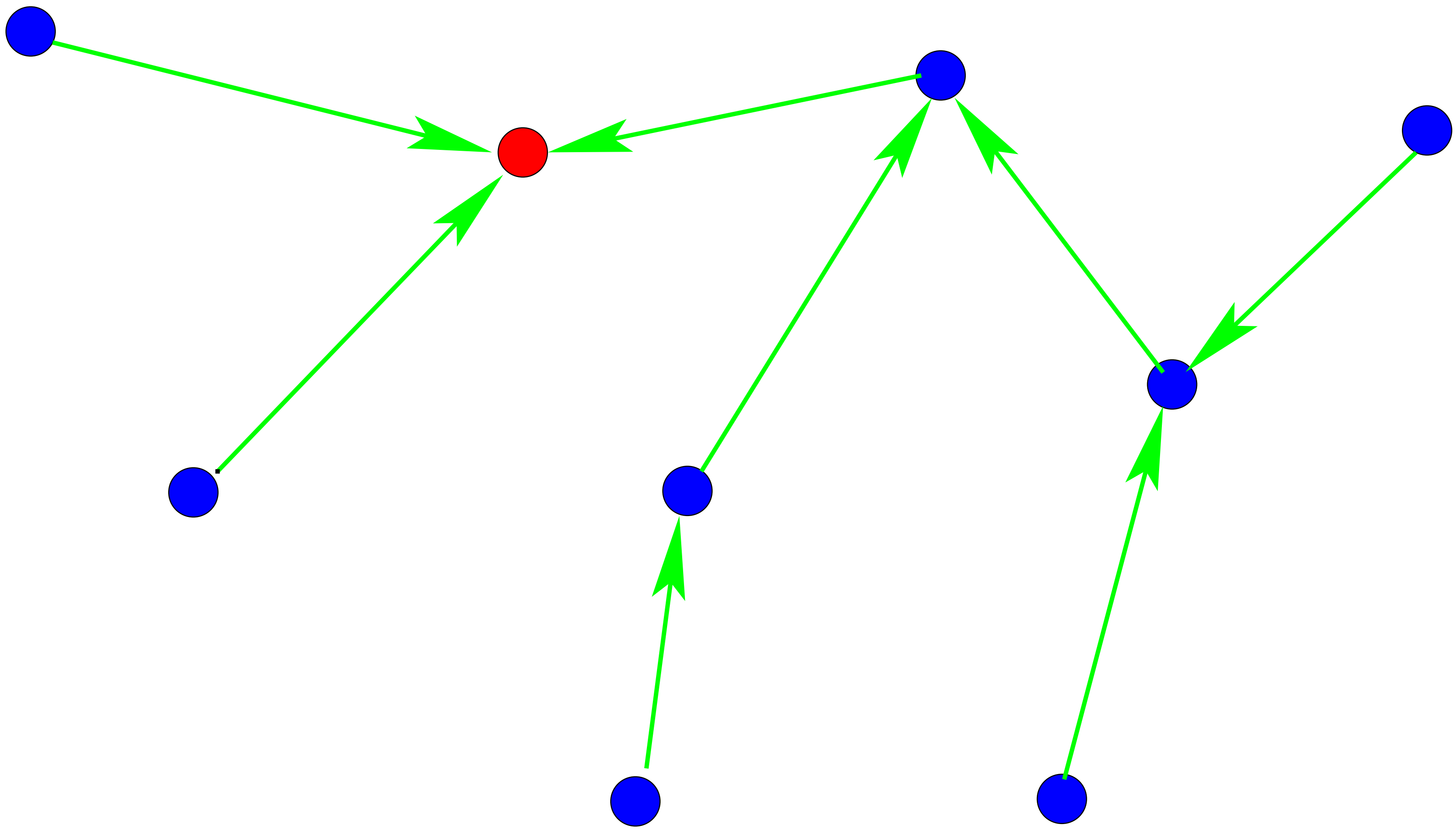
$tr := tr \cup \{x \mapsto y\}$

END

To be proved

$$
\begin{array}{l}
\quad \mathsf{invariant}(tr) \quad \wedge \\
\quad x, y \in gr \quad \wedge \\
\quad x \notin tr \quad \wedge \\
\quad y \notin tr \quad \wedge \\
\quad gr[\{x\}] = tr^{-1}[\{x\}] \cup \{y\} \\
\Rightarrow \\
\quad \mathsf{invariant}(tr \cup \{x \mapsto y\})
\end{array}
$$





When a red node x is ONLY connected to blue nodes then event "elect" can take place

$$
\begin{array}{l}
\text{elect} \quad \widehat{=} \\
\quad \textbf{ANY } x \textbf{ WHERE} \\
\qquad x \in ND \ \wedge \\
\qquad gr[\{x\}] = tr^{-1}[\{x\}] \\
\quad \textbf{THEN} \\
\qquad rt, ts := x, tr \\
\quad \textbf{END}
\end{array}
$$

elect $\widehat{=}$

BEGIN

rt, ts : spanning (rt, ts, gr)

END

elect $\widehat{=}$

ANY x **WHERE**

$x \in ND \;\wedge$

$gr[\{x\}] = tr^{-1}[\{x\}]$

THEN

$rt, ts := x, tr$

END

To be proved

$$
\begin{array}{l}
\quad \text{invariant}(tr) \quad \wedge \\
\quad x \in ND \quad \wedge \\
\quad gr[\{x\}] = tr^{-1}[\{x\}] \\
\quad ts = tr \\
\Rightarrow \\
\quad \text{spanning}(x, ts, gr)
\end{array}
$$

Summary of First Refinement

- 15 proofs

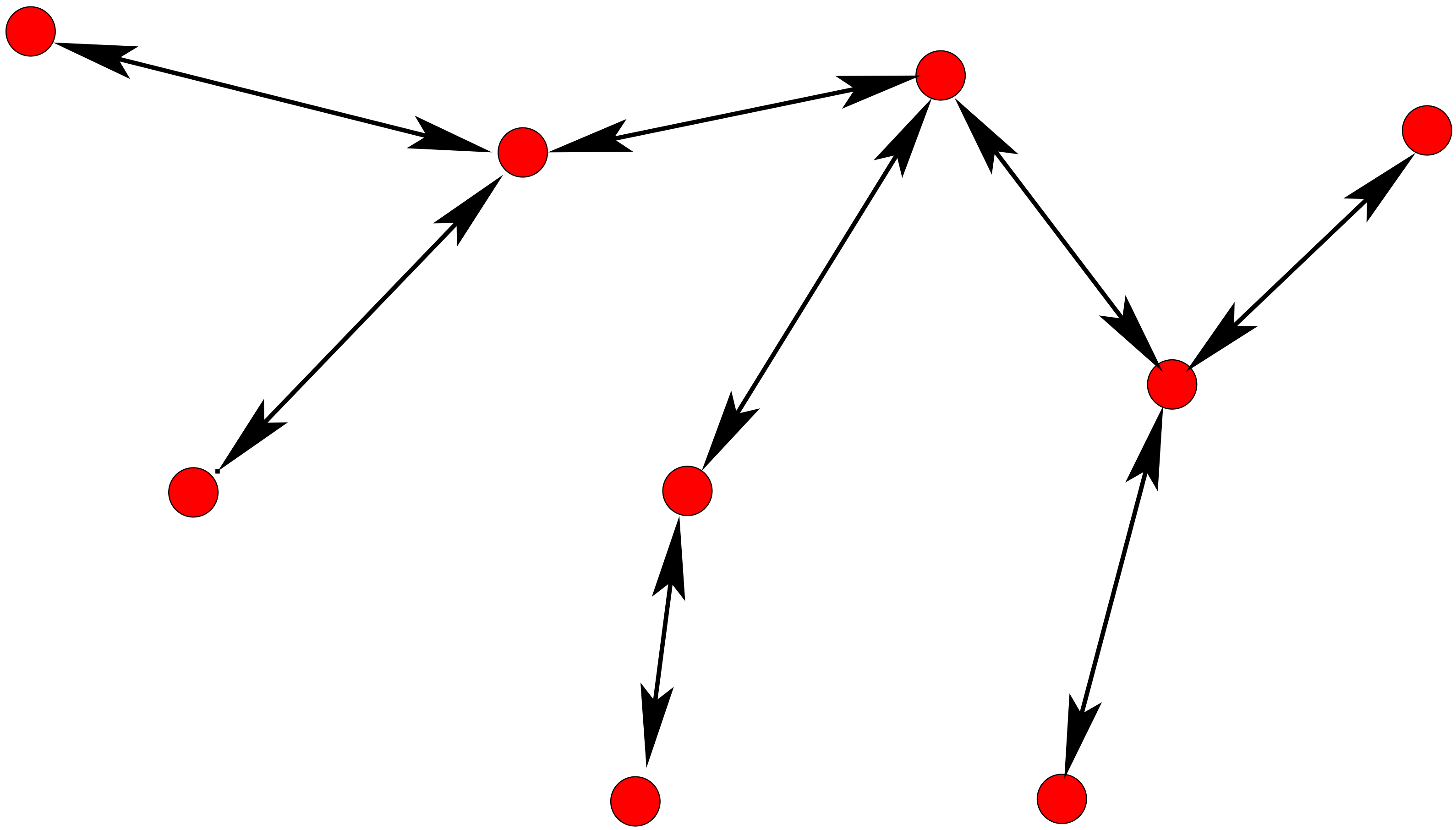
- Among which 9 were interactive (one is a bit difficult !)

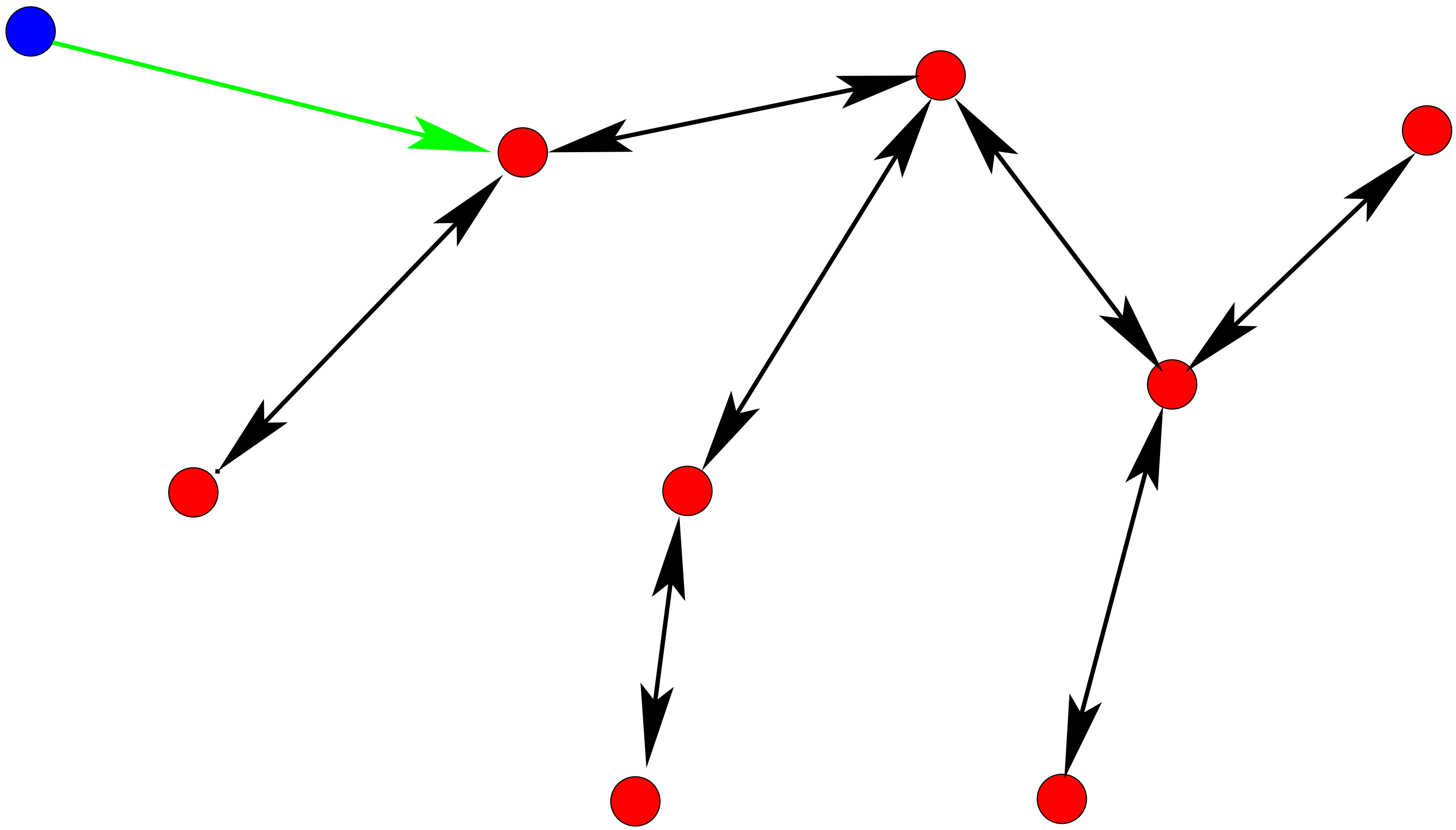
Current state of the model

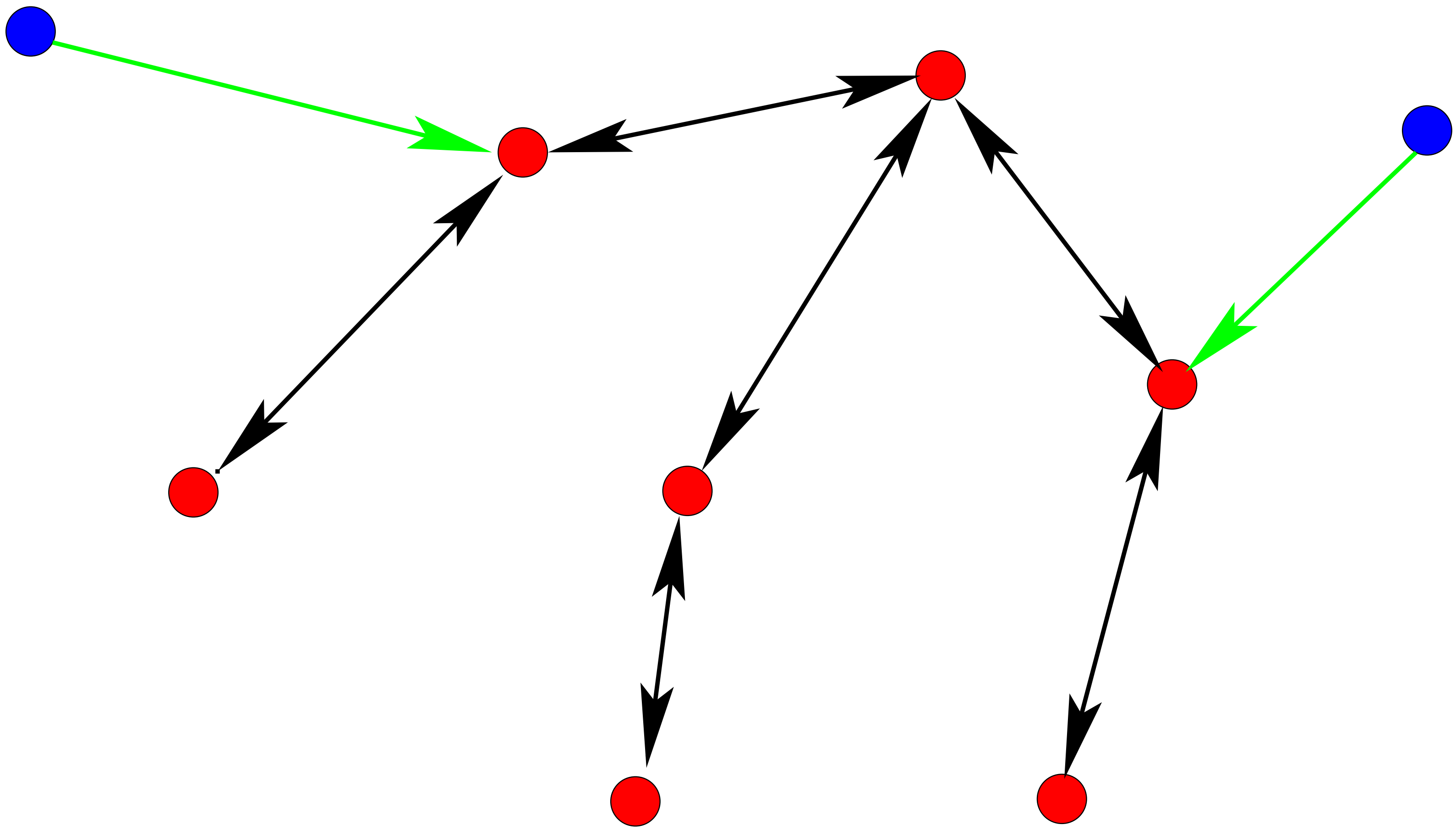
- 12 proofs

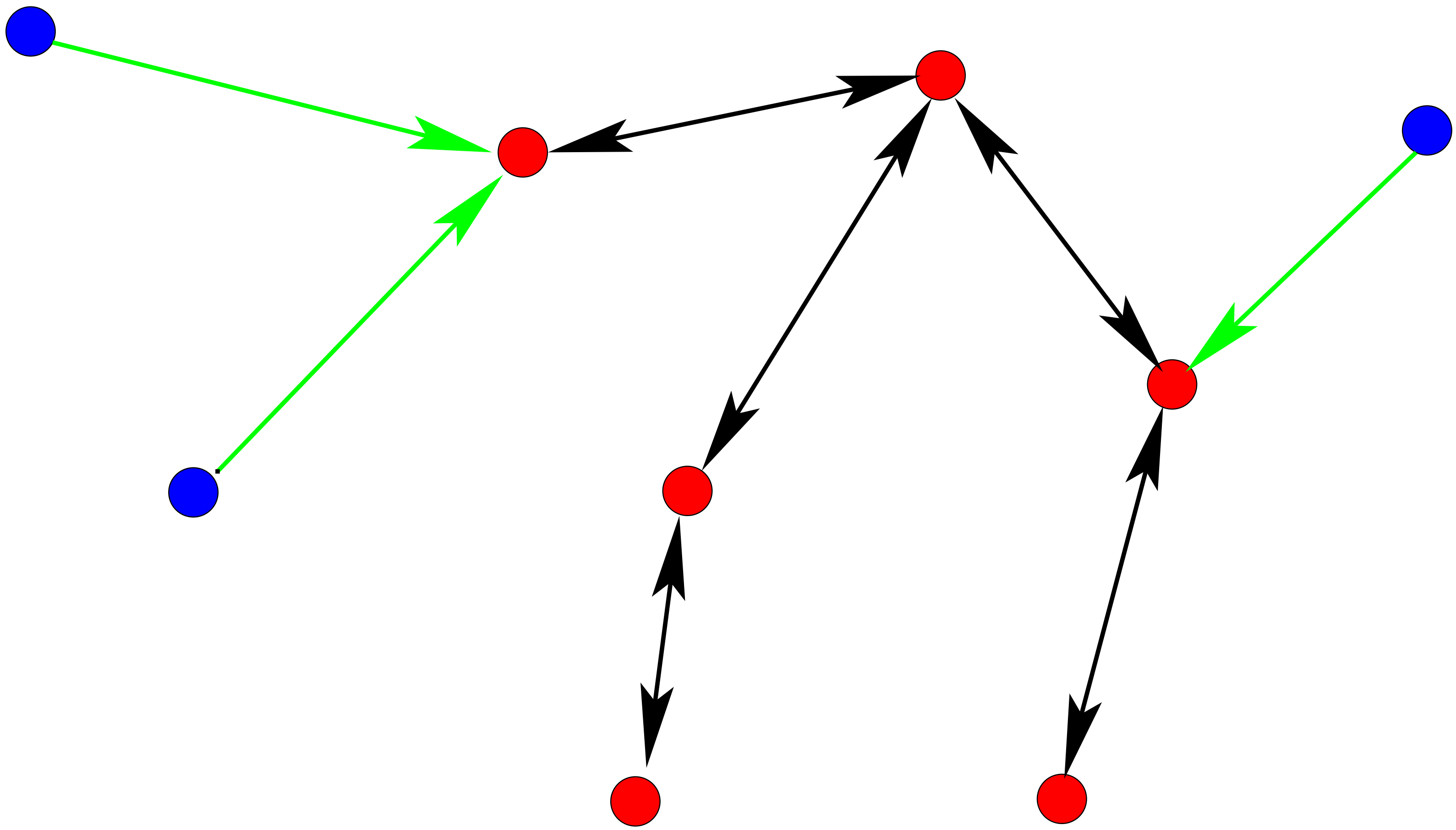
- Among which 5 were interactive (one is a bit difficult !)

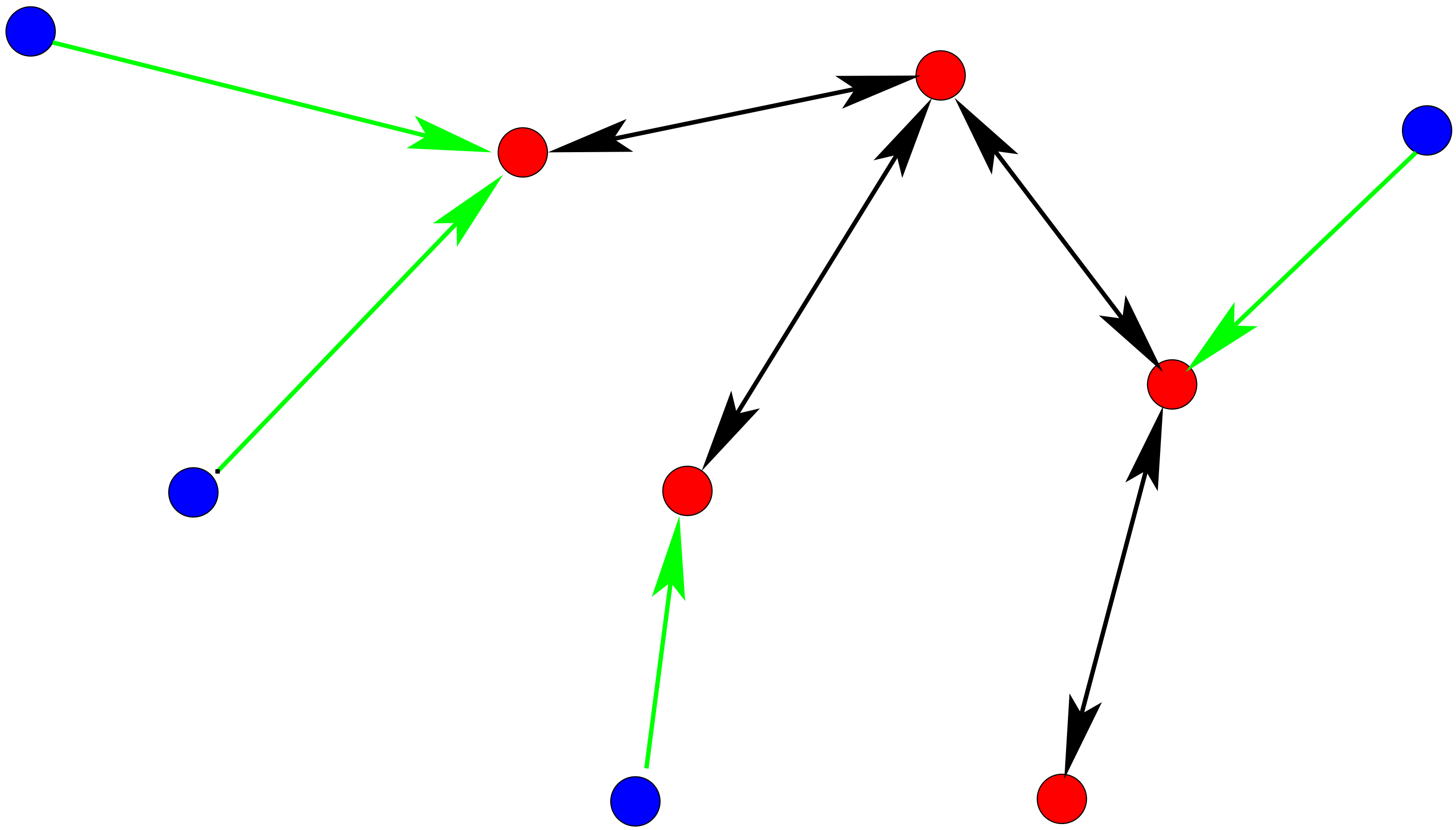
- Animation of the current model

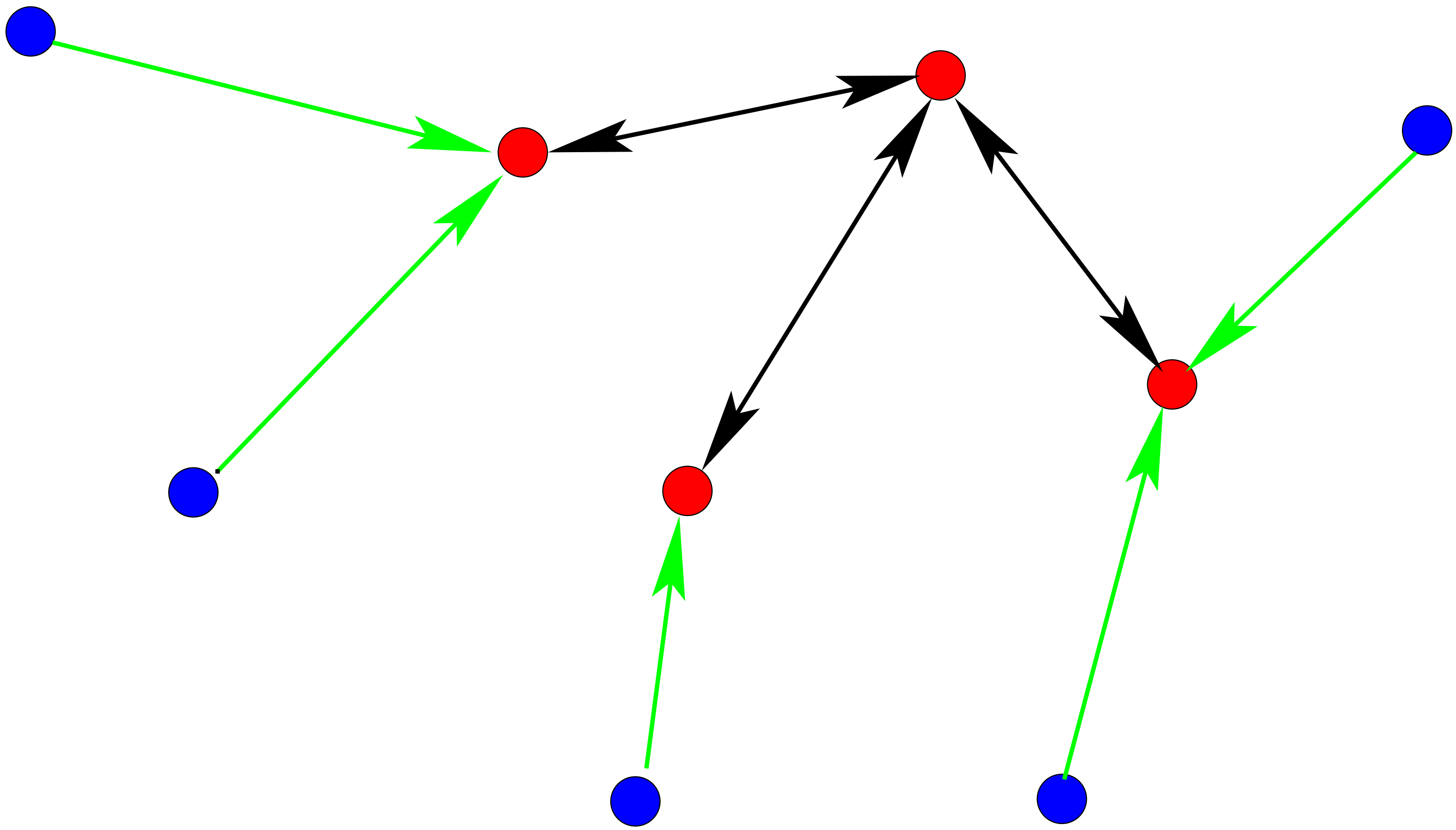


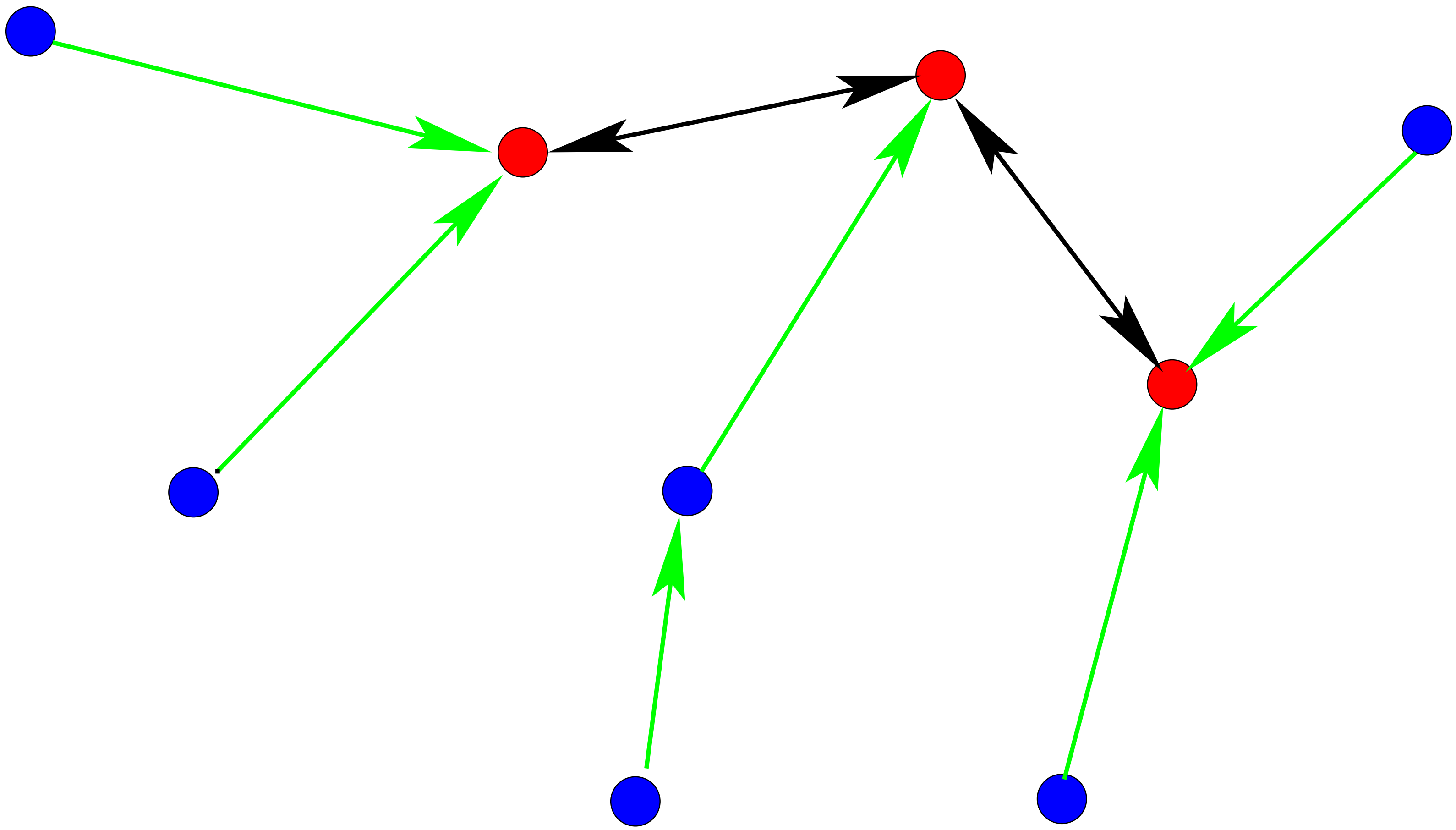


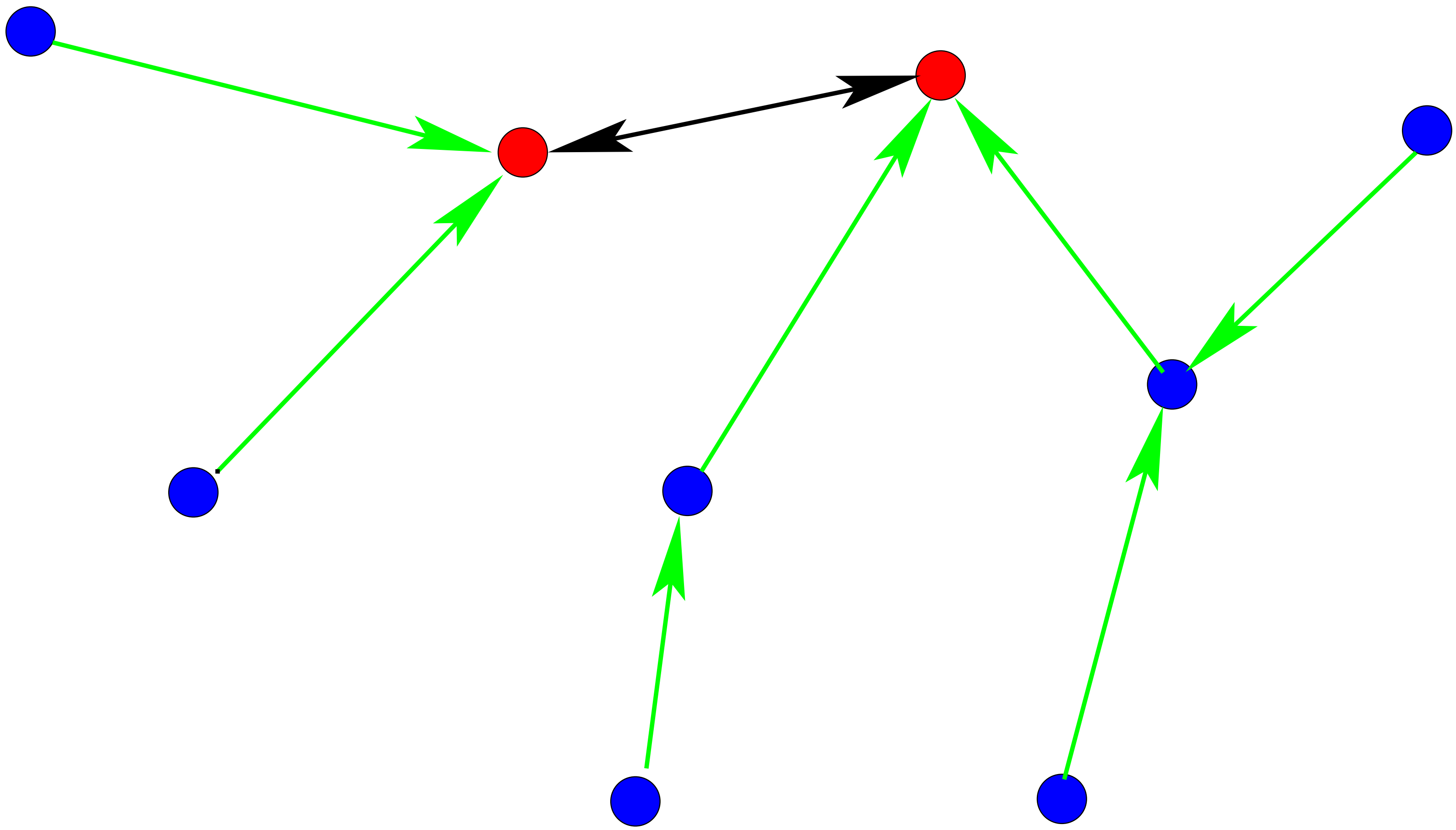


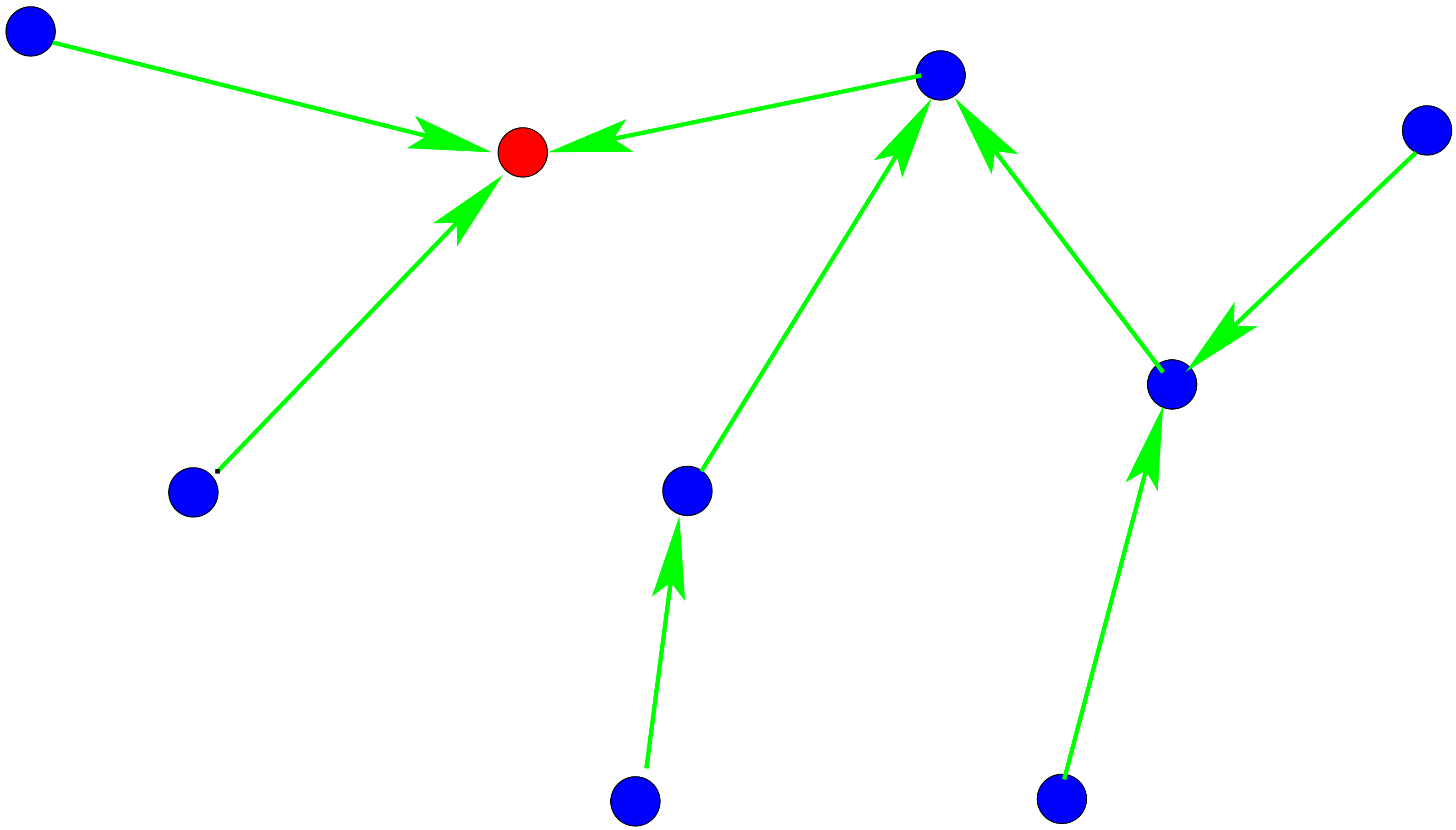


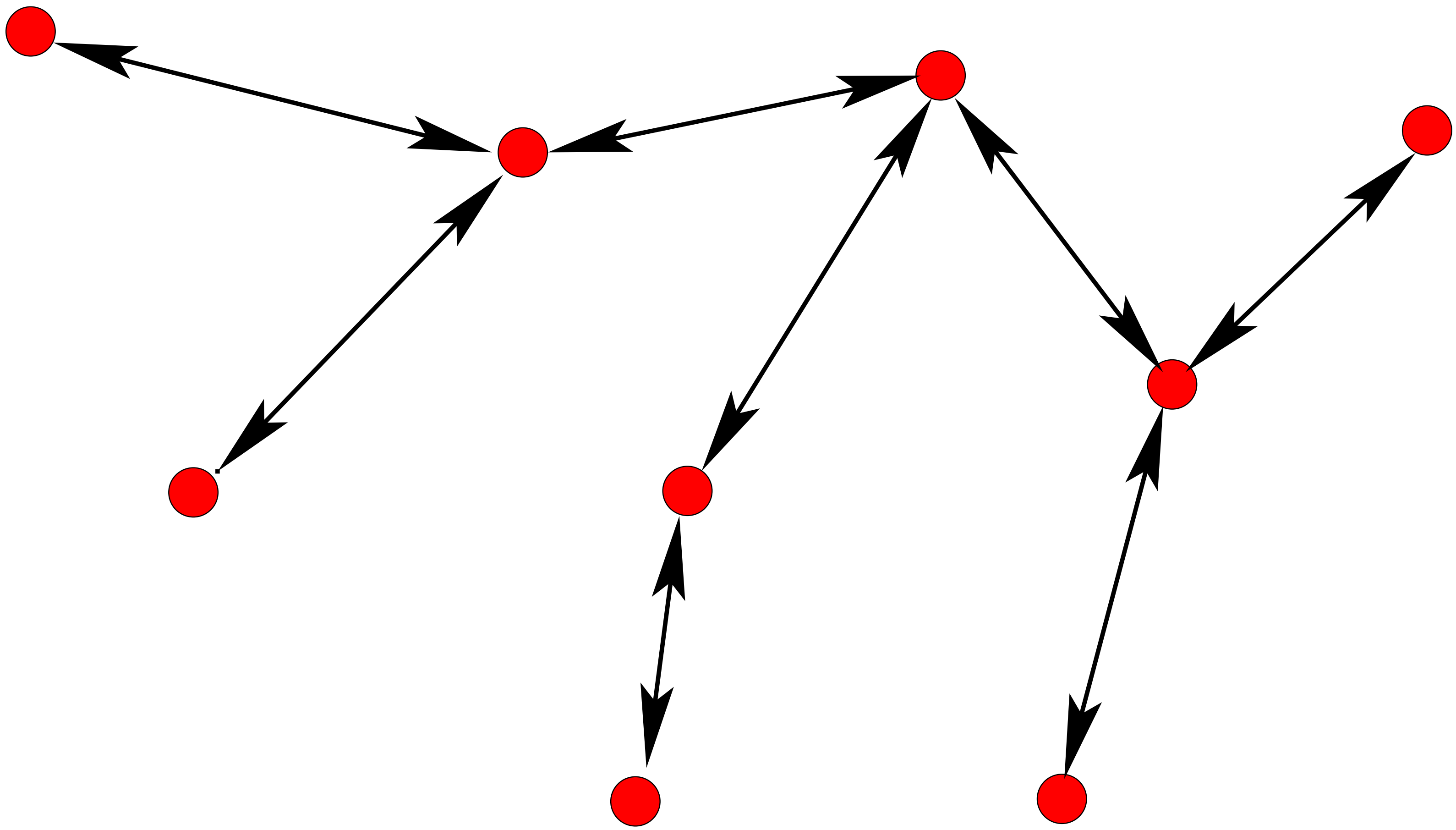


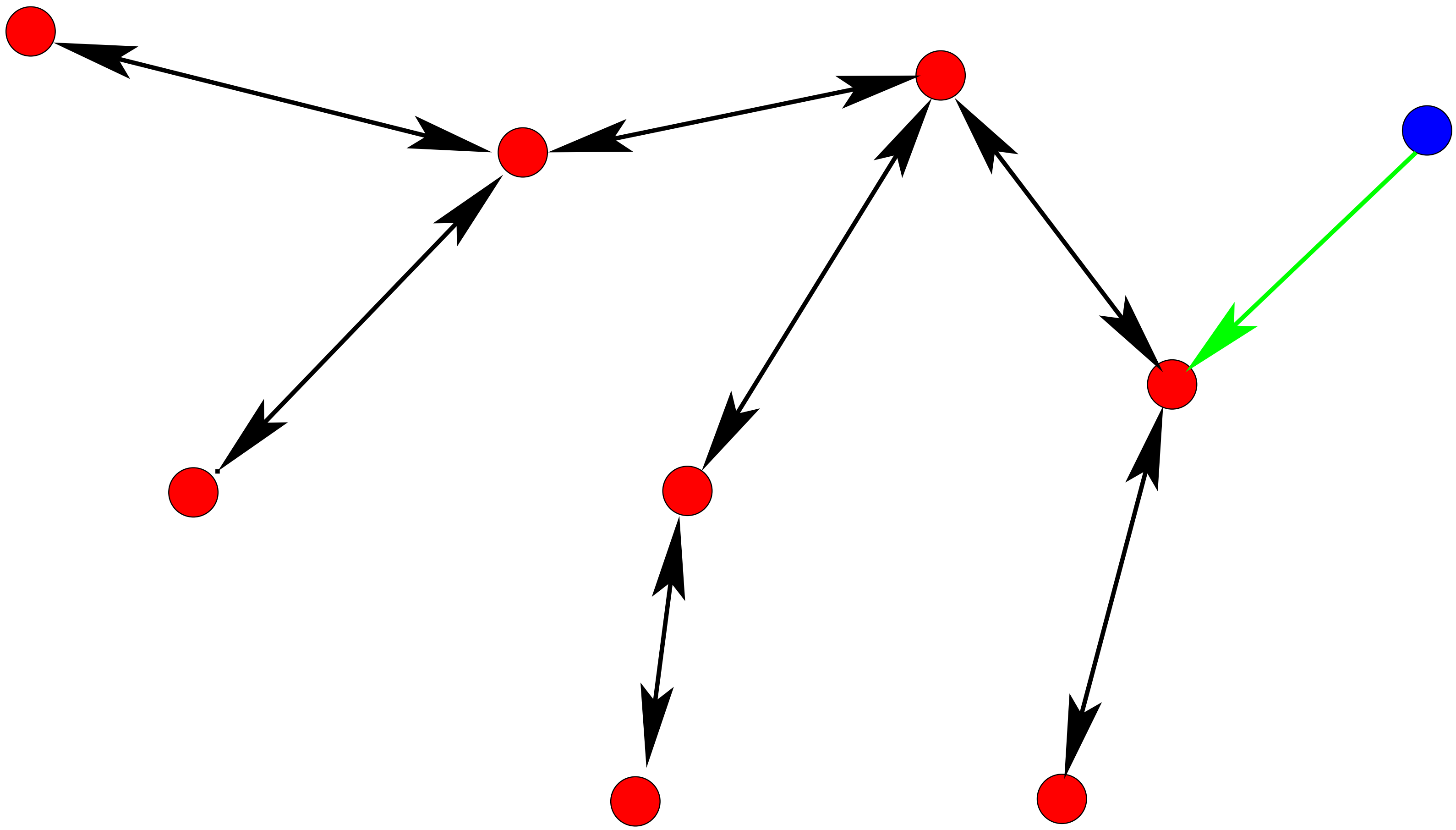


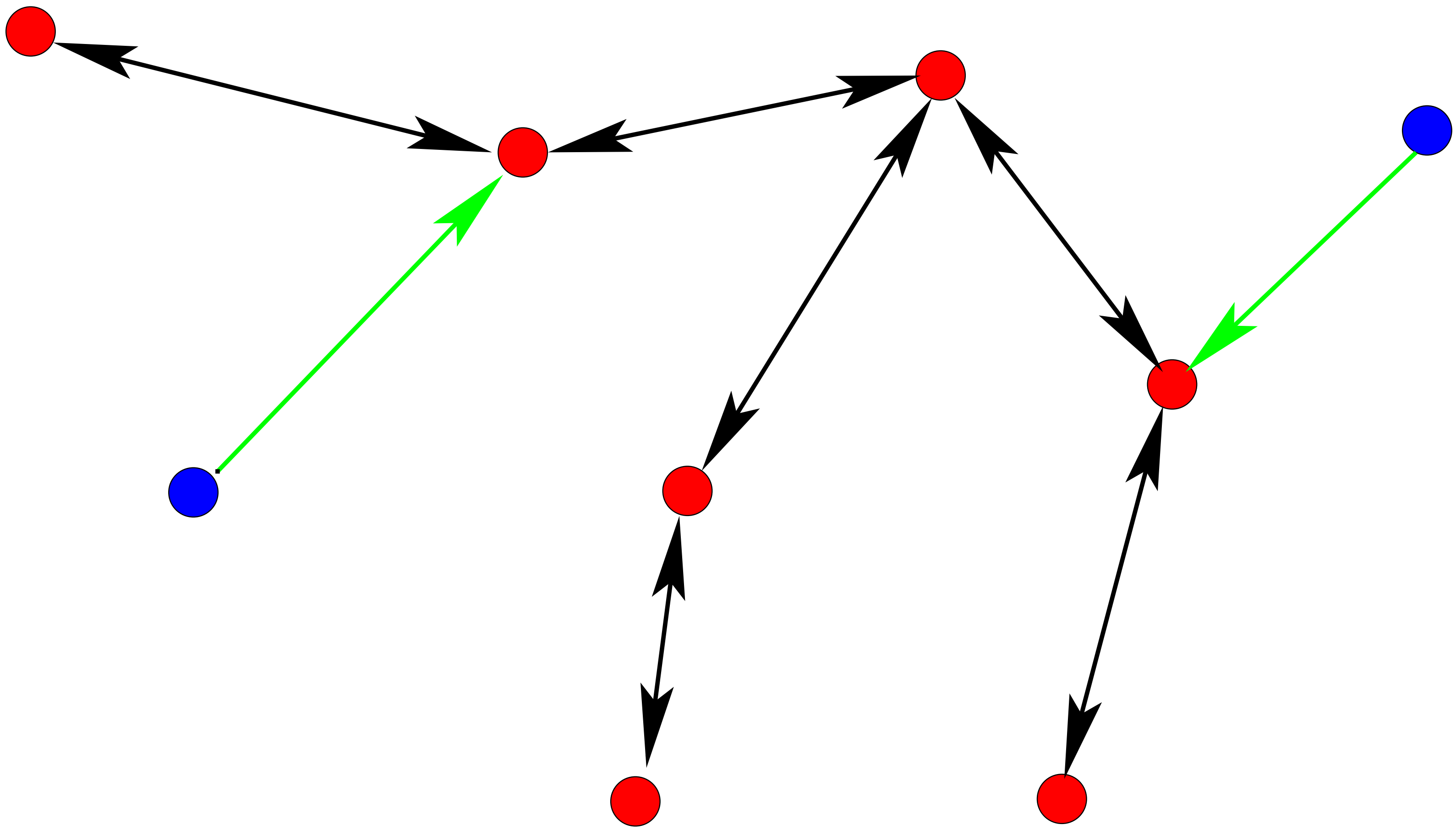


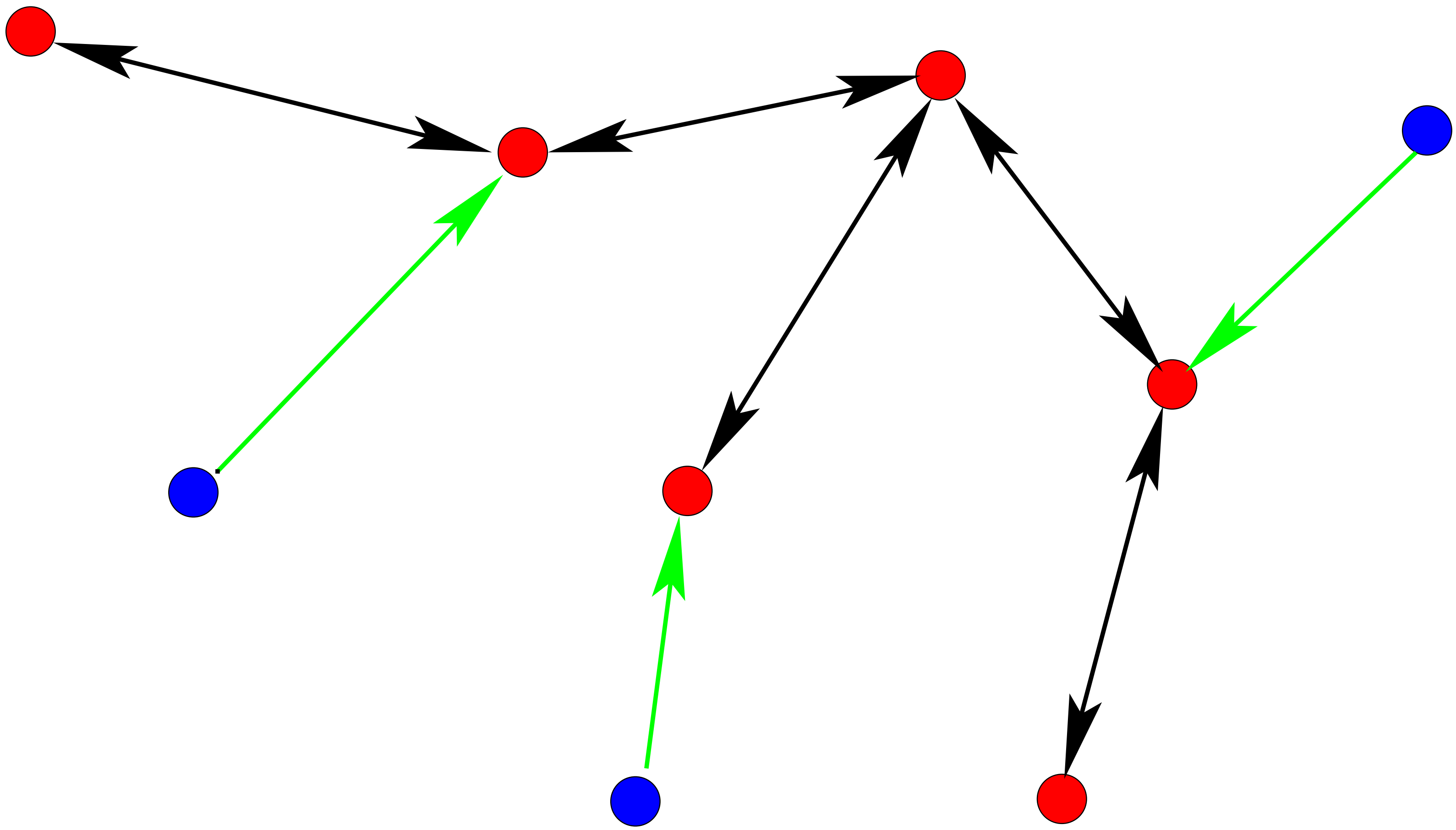


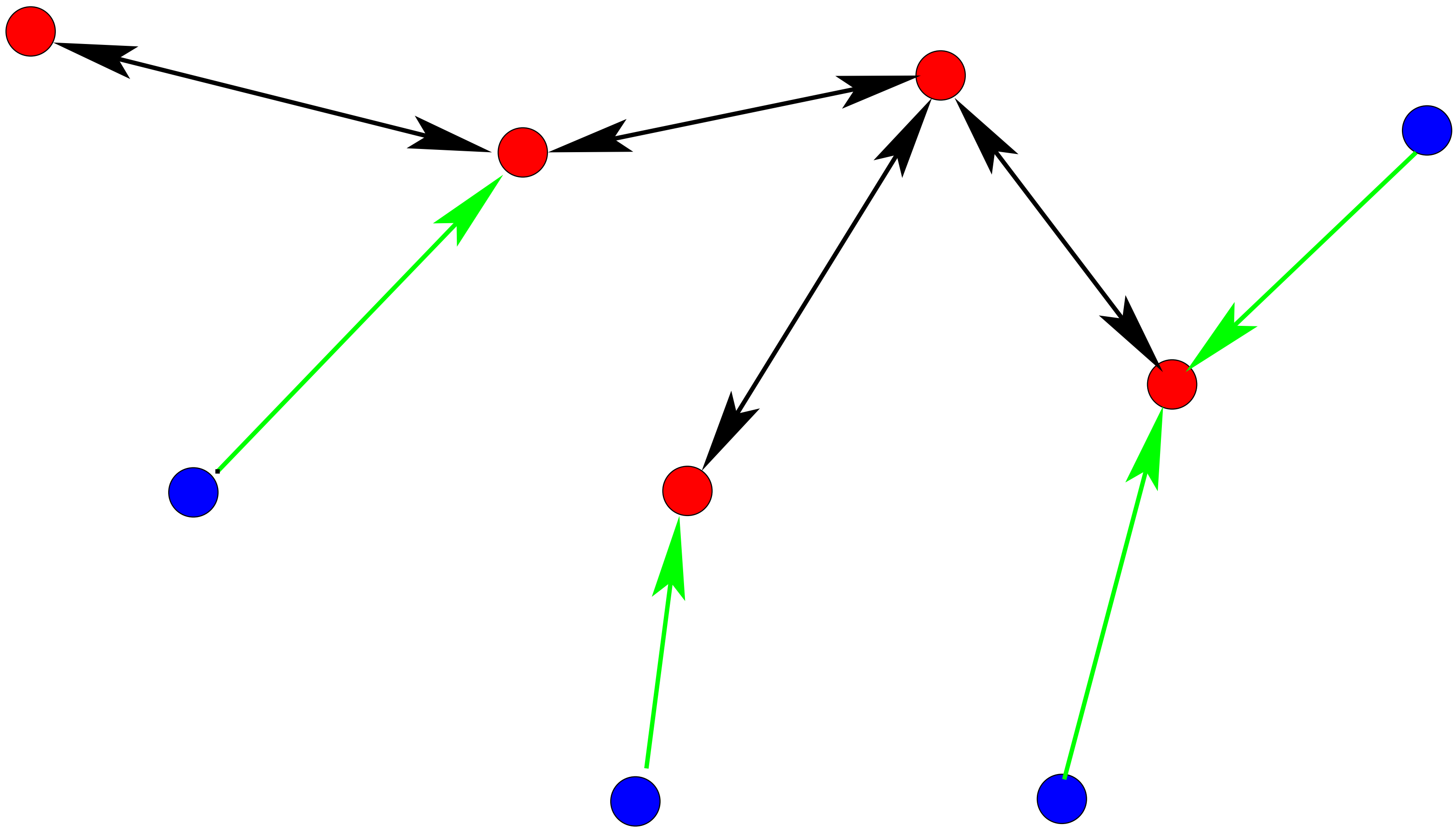


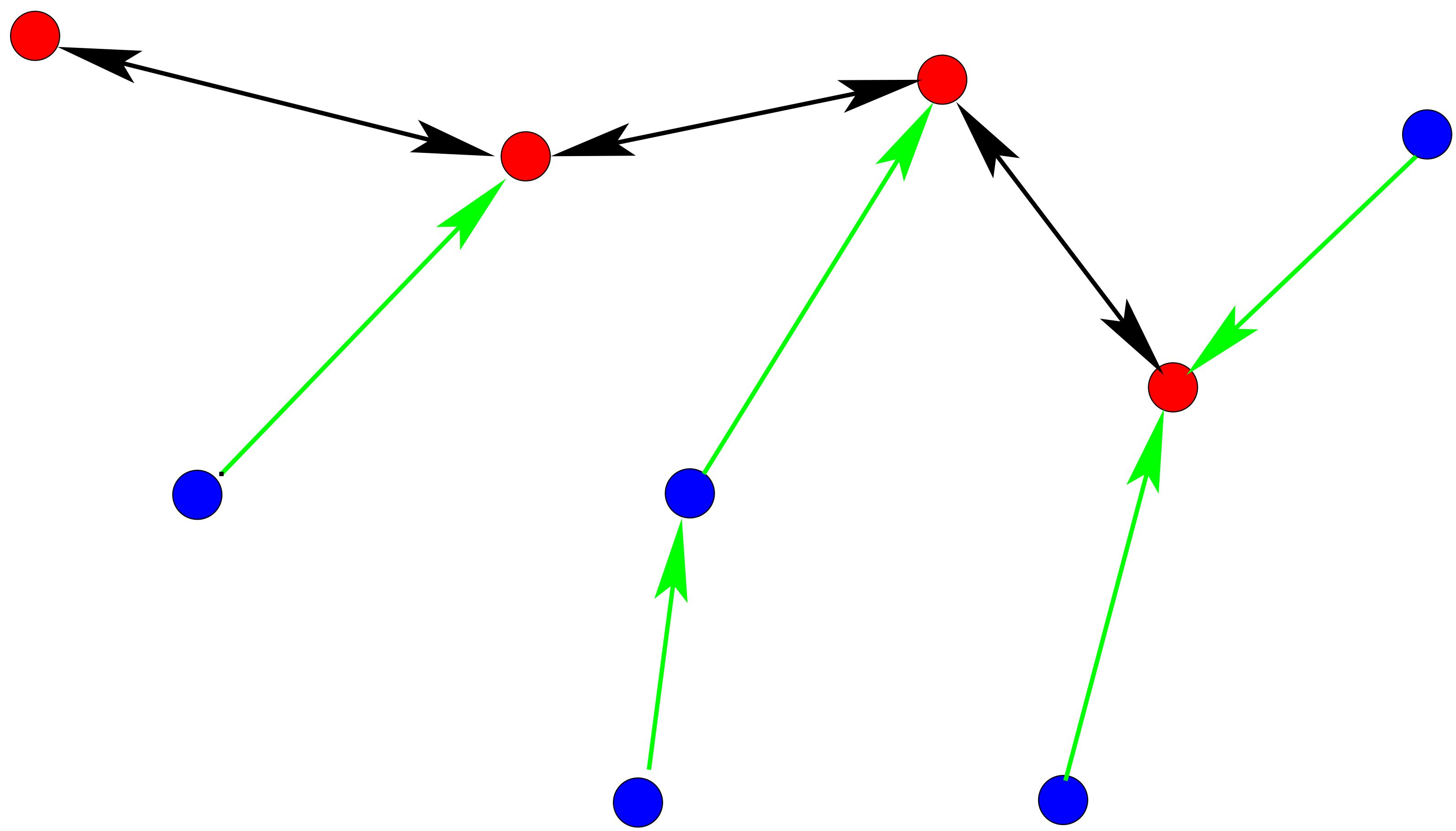


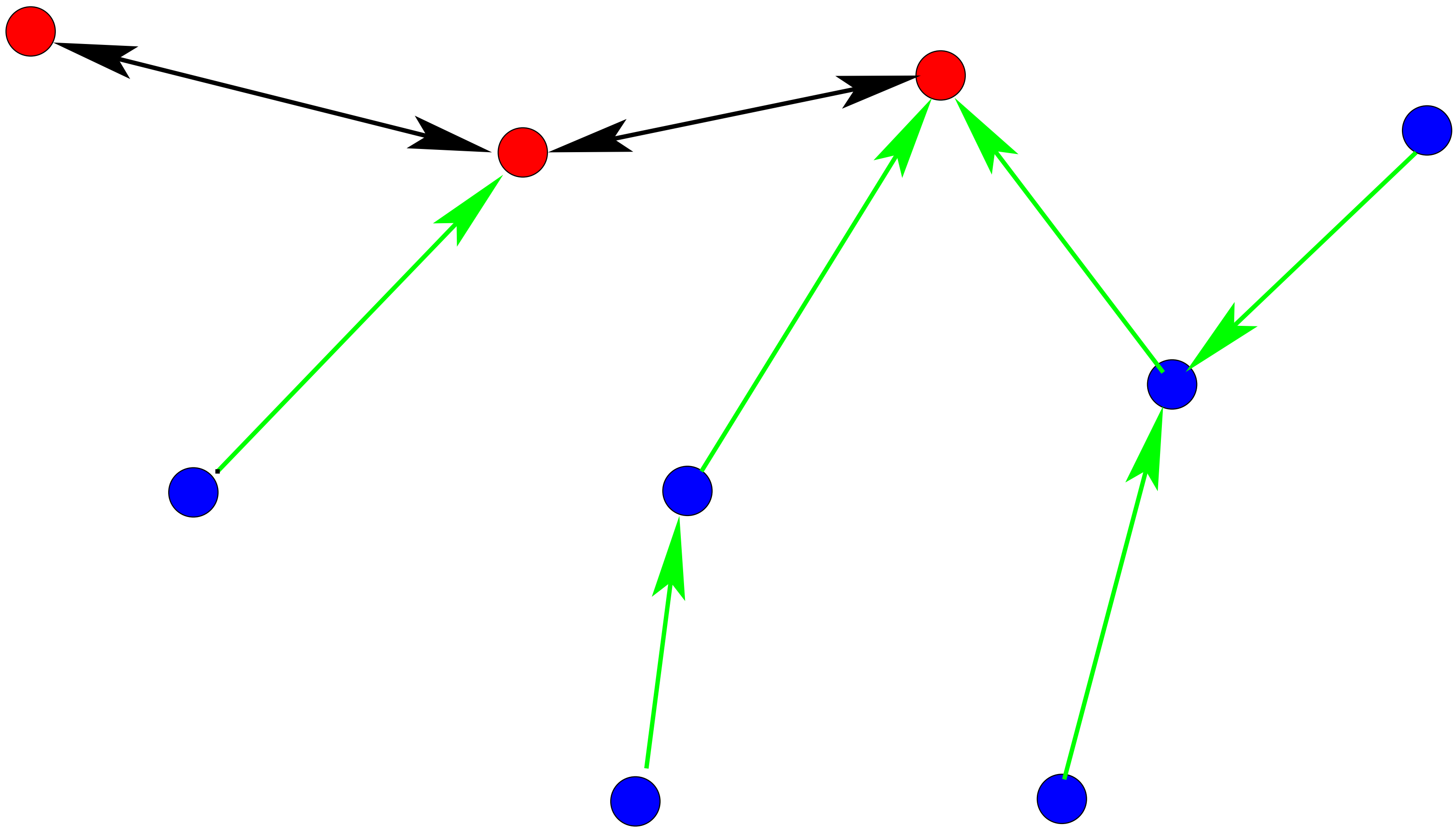


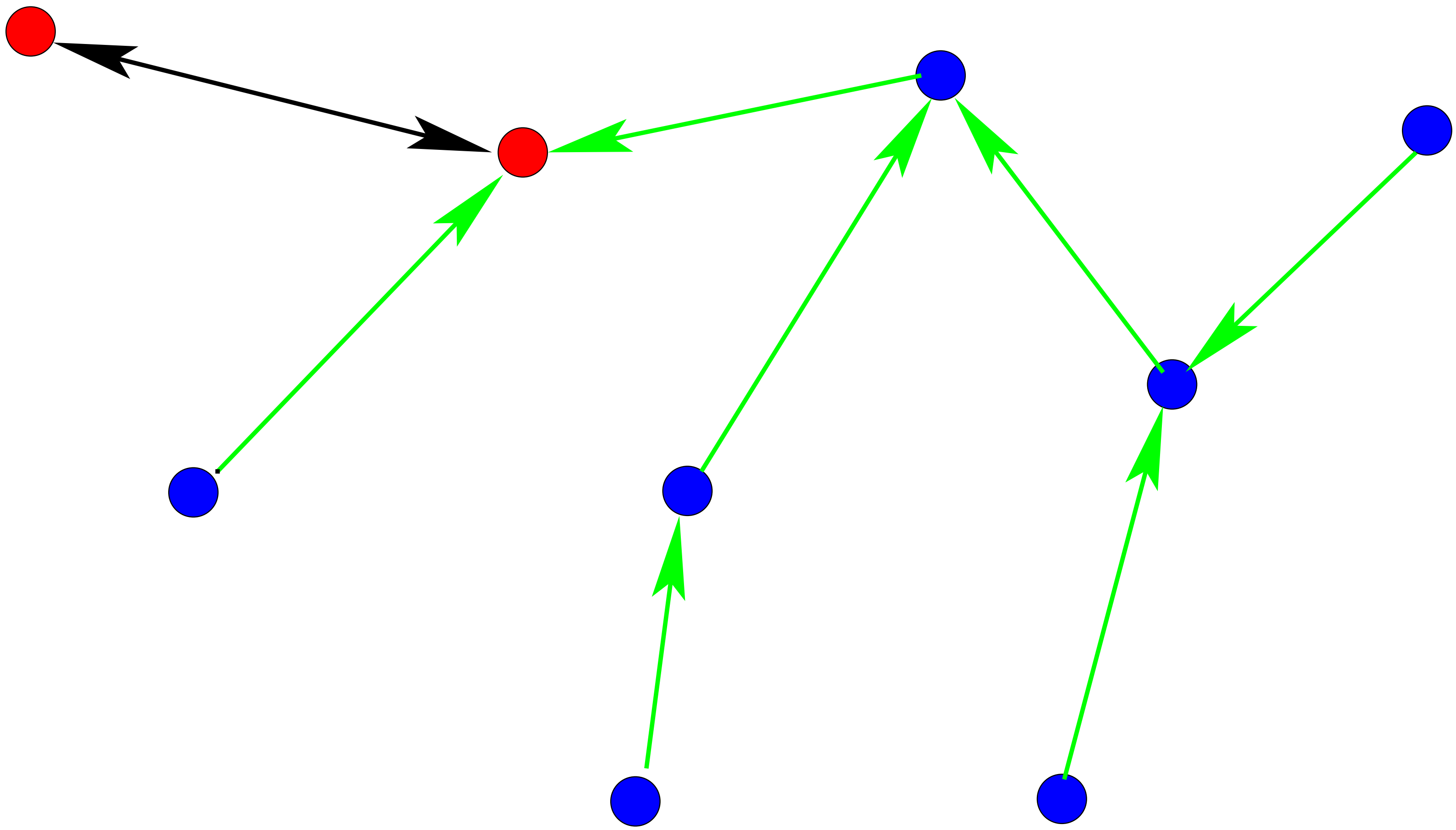


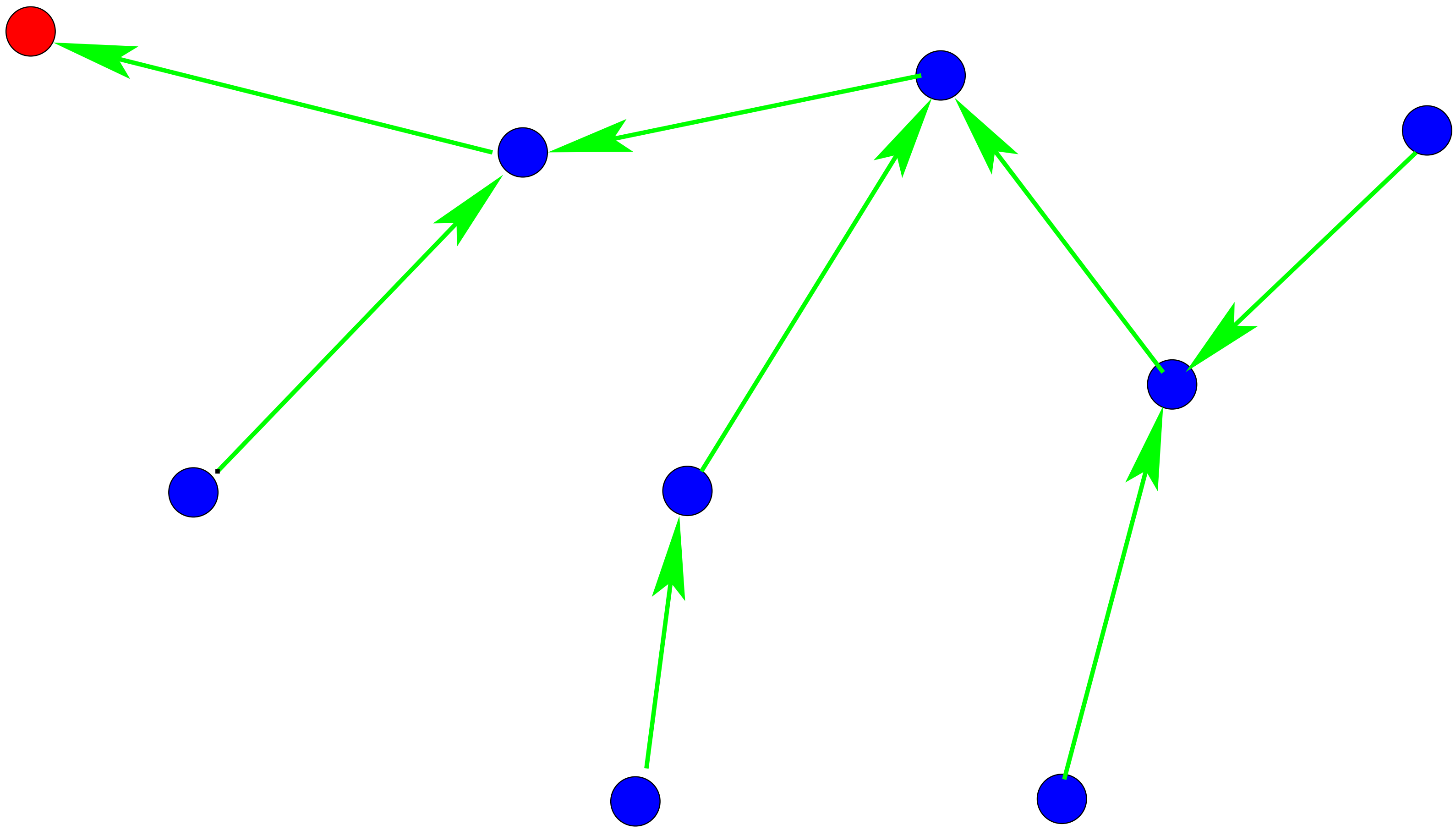












Second Refinement

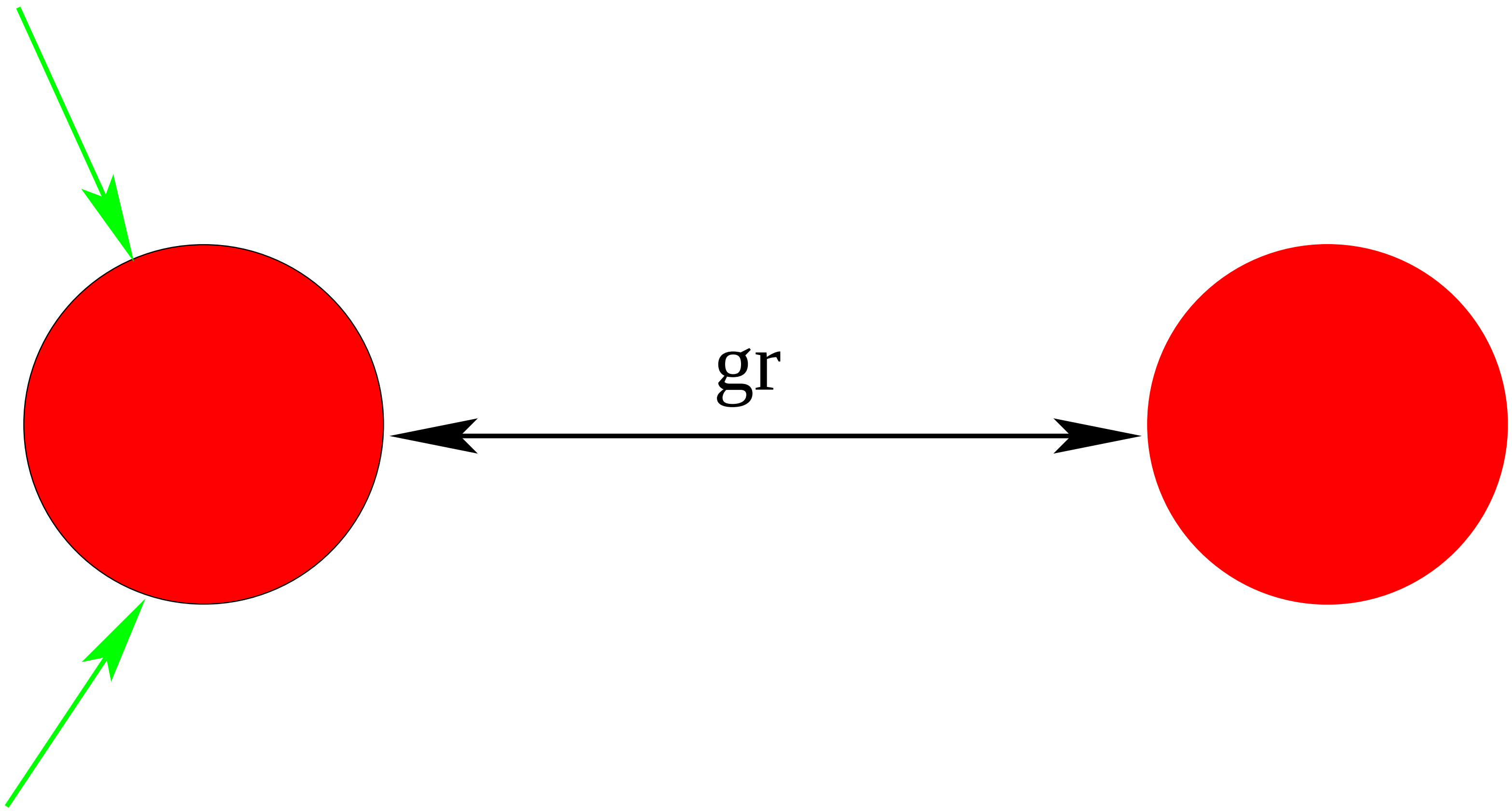
- Nodes are communicating with their neighbors

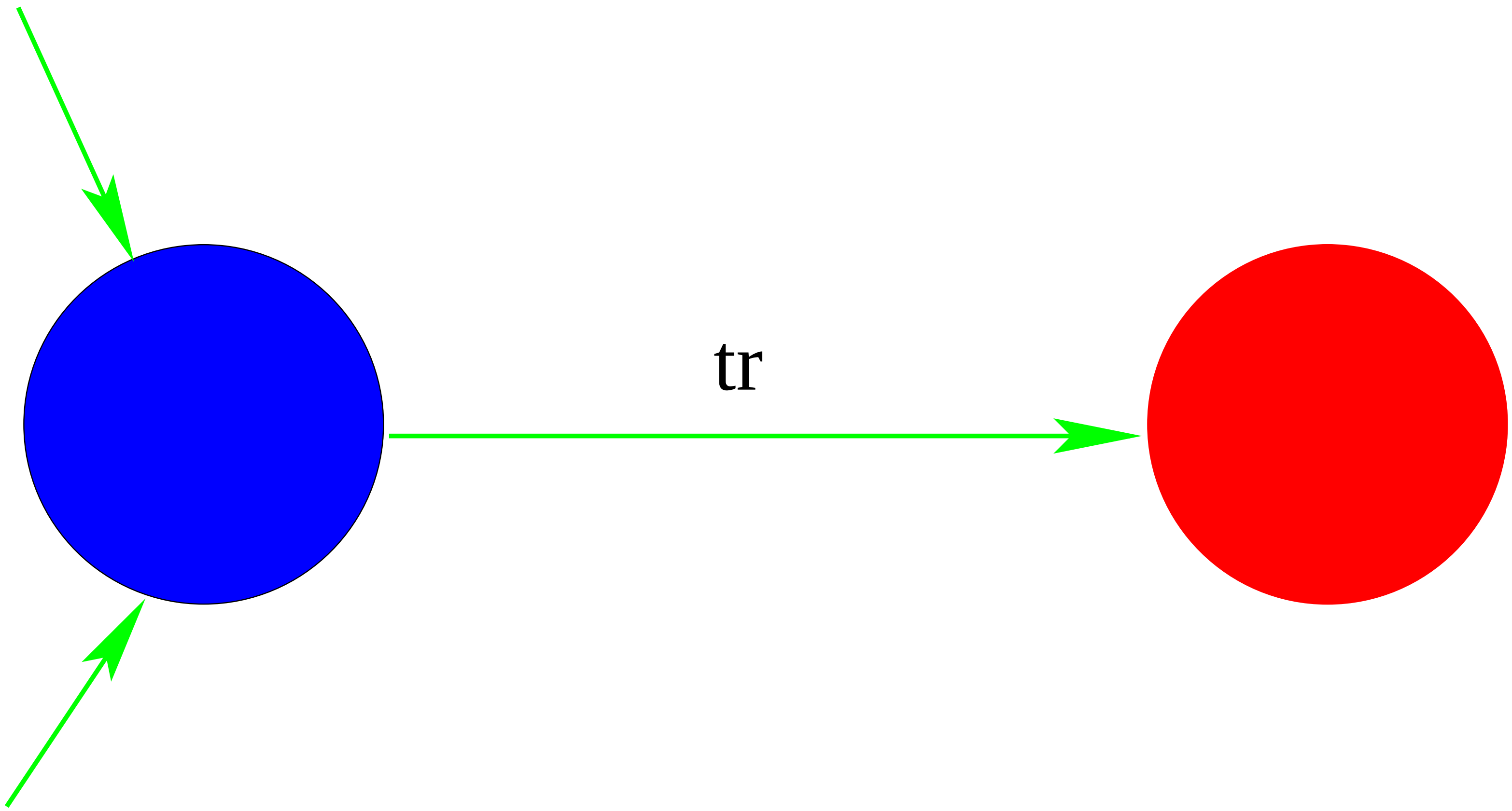
- This is done by means of messages

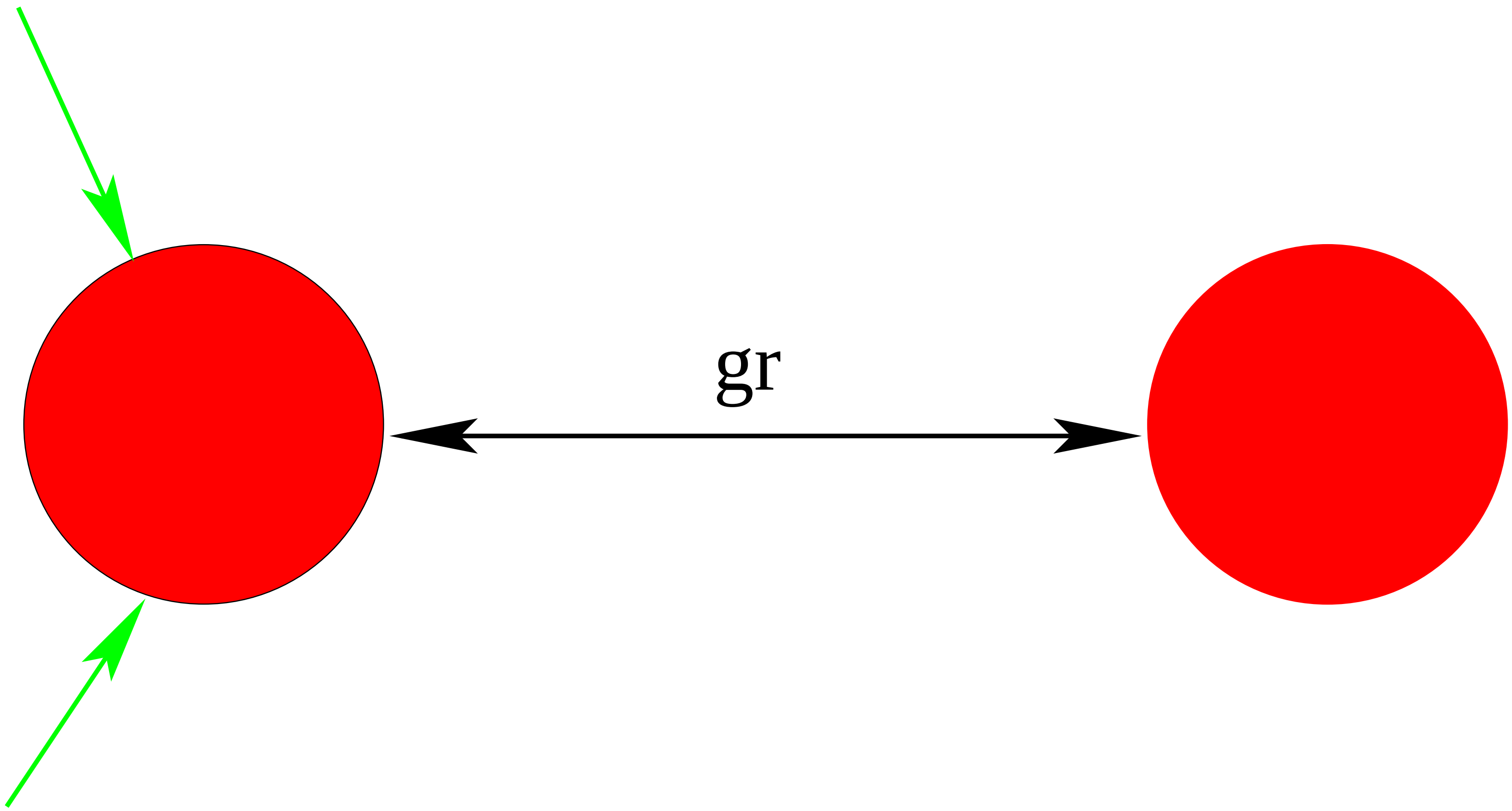
- Messages are acknowledged

- Acknowledgements are confirmed

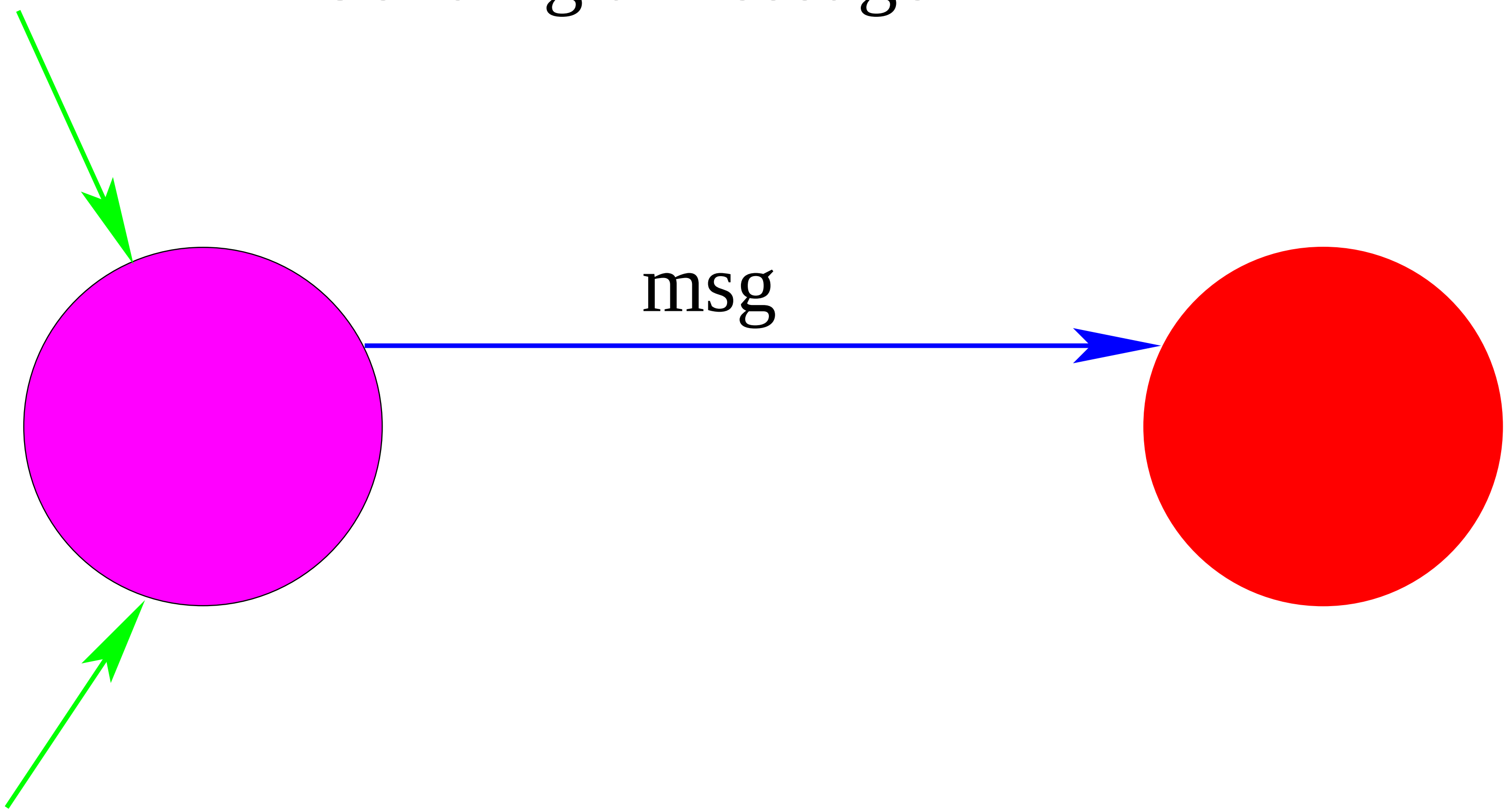
- Next is a local animation


gr


tr

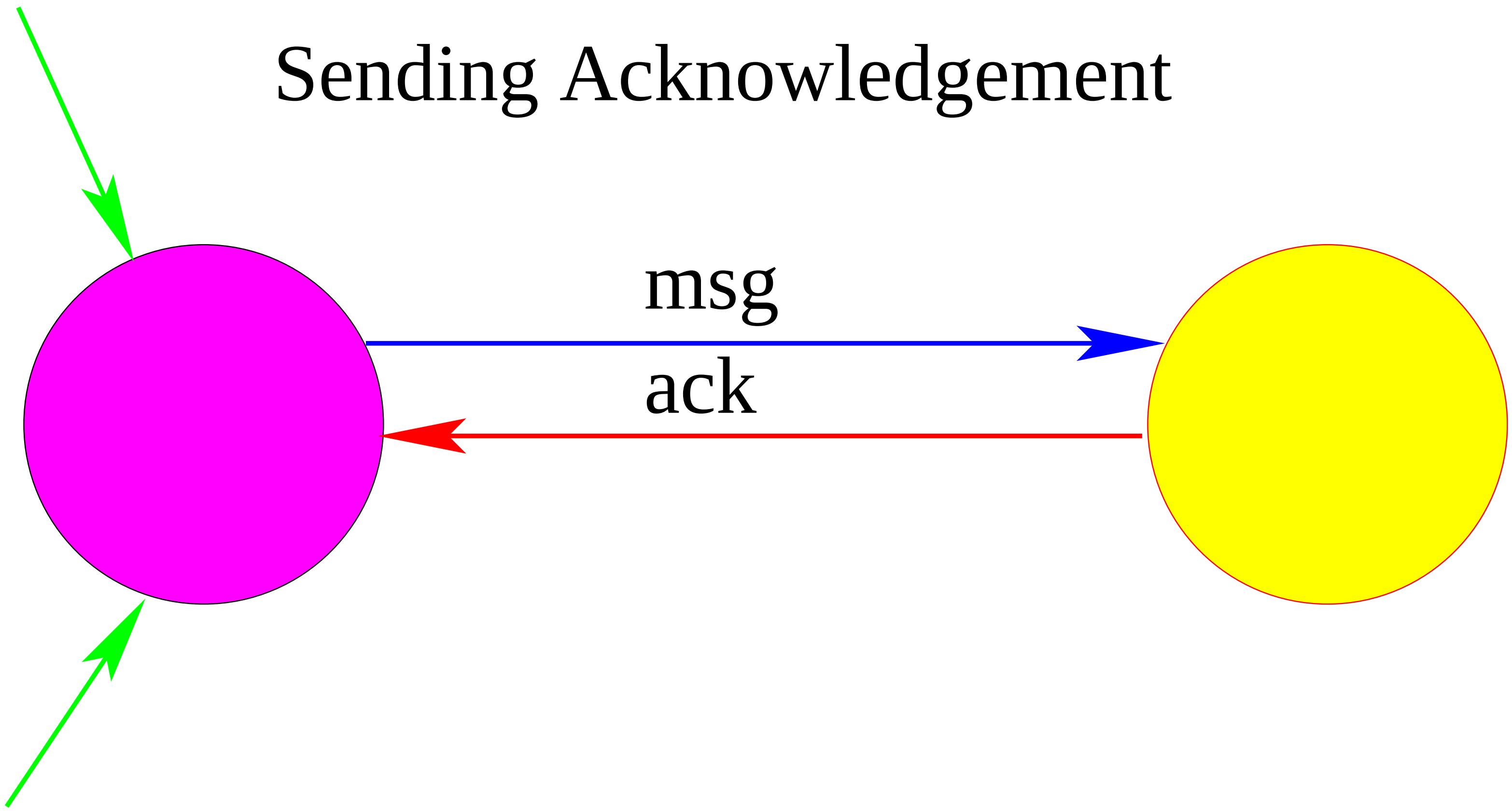

gr

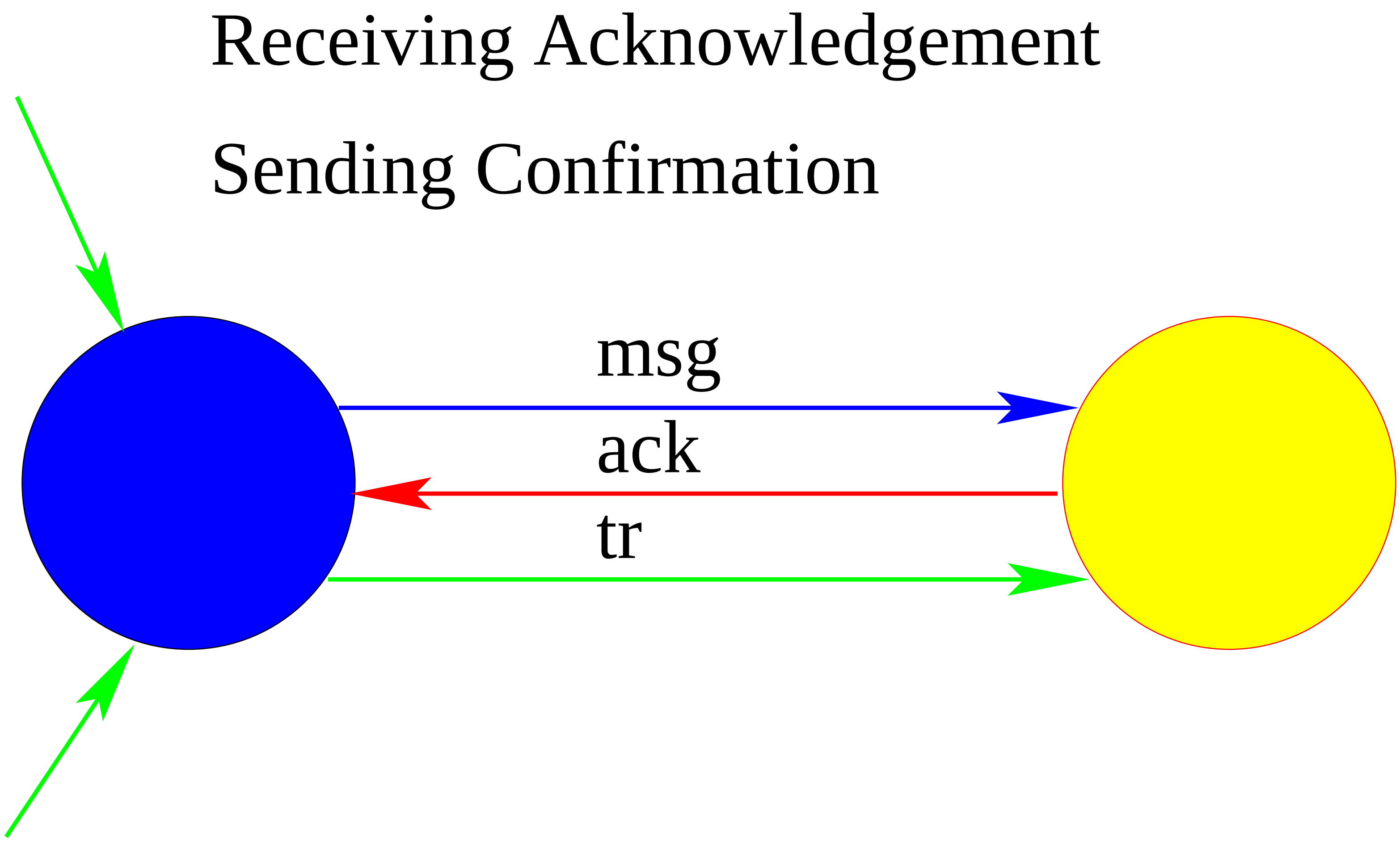
Sending a message



Receiving a message

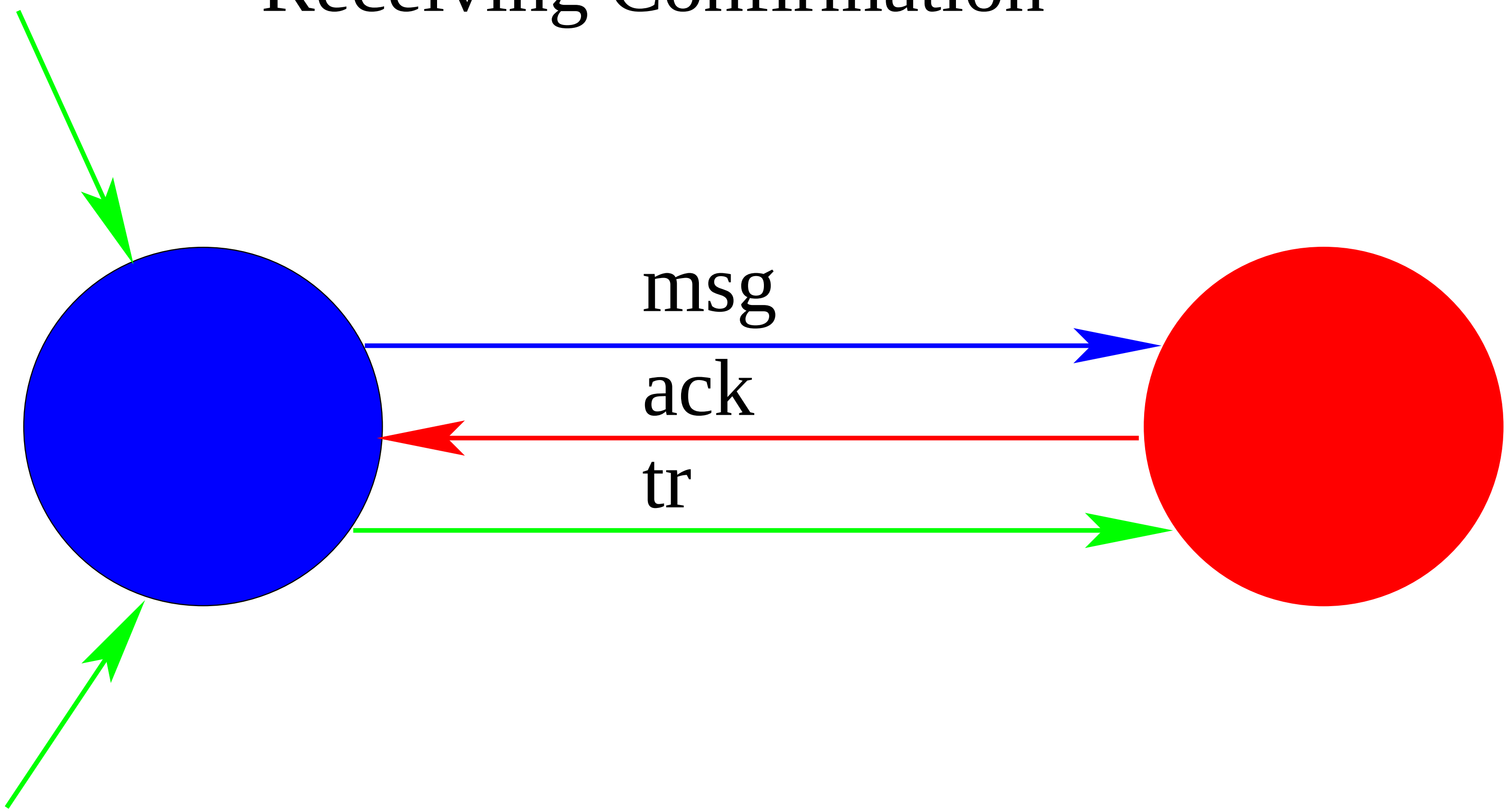
Sending Acknowledgement





Receiving Acknowledgement
Sending Confirmation
msg
ack
tr

Receiving Confirmation



Invariant (1)

$$msg \in ND \leftrightarrow ND$$

$$ack \in ND \leftrightarrow ND$$

$$tr \ \subseteq \ ack \ \subseteq \ msg \ \subseteq \ gr$$

Node x sends a message to node y

send_msg $\widehat{=}$

ANY x, y **WHERE**

$x, y \in gr \ \wedge$

$x \notin \text{dom}\,(tr) \ \wedge$

$y, x \notin tr \ \wedge$

$gr[\{x\}] = tr^{-1}[\{x\}] \cup \{y\} \ \wedge$

$y, x \notin ack \ \wedge$

$x \notin \text{dom}\,(msg)$

THEN

$msg := msg \cup \{x \mapsto y\}$

END

Node y sends an acknowledgement to node x

send_ack $\;\widehat{=}$

ANY x, y **WHERE**

$x, y \in msg - ack \;\; \wedge$

$y \notin \mathrm{dom}\,(msg)$

THEN

$ack := ack \cup \{x \mapsto y\}$

END

Node x sends a confirmation to node y

progress $\widehat{=}$

ANY x, y **WHERE**

$x, y \in ack \ \wedge$

$x \notin \mathrm{dom}\,(tr)$

THEN

$tr := tr \cup \{x \mapsto y\}$

END

Invariant (2)

$$\forall\,(x,y)\cdot\left(\begin{array}{l}\quad x,y \in msg{-}ack \\ \Rightarrow \\ \quad x,y \in gr \qquad \wedge \\ \quad x \notin \mathsf{dom}\,(tr) \;\;\wedge\;\; y \notin \mathsf{dom}\,(tr) \;\;\wedge \\ \quad gr[\{x\}] \;=\; tr^{-1}[\{x\}]\cup\{y\}\end{array}\right)$$

$$\forall\,(x,y)\cdot\left(\begin{array}{l}\quad x,y \in ack \qquad \wedge \\ \quad x \notin \mathsf{dom}\,(tr) \\ \Rightarrow \\ \quad x,y \in gr \qquad \wedge \\ \quad y \notin \mathsf{dom}\,(tr) \;\;\wedge \\ \quad gr[\{x\}] \;=\; tr^{-1}[\{x\}]\cup\{y\}\end{array}\right)$$

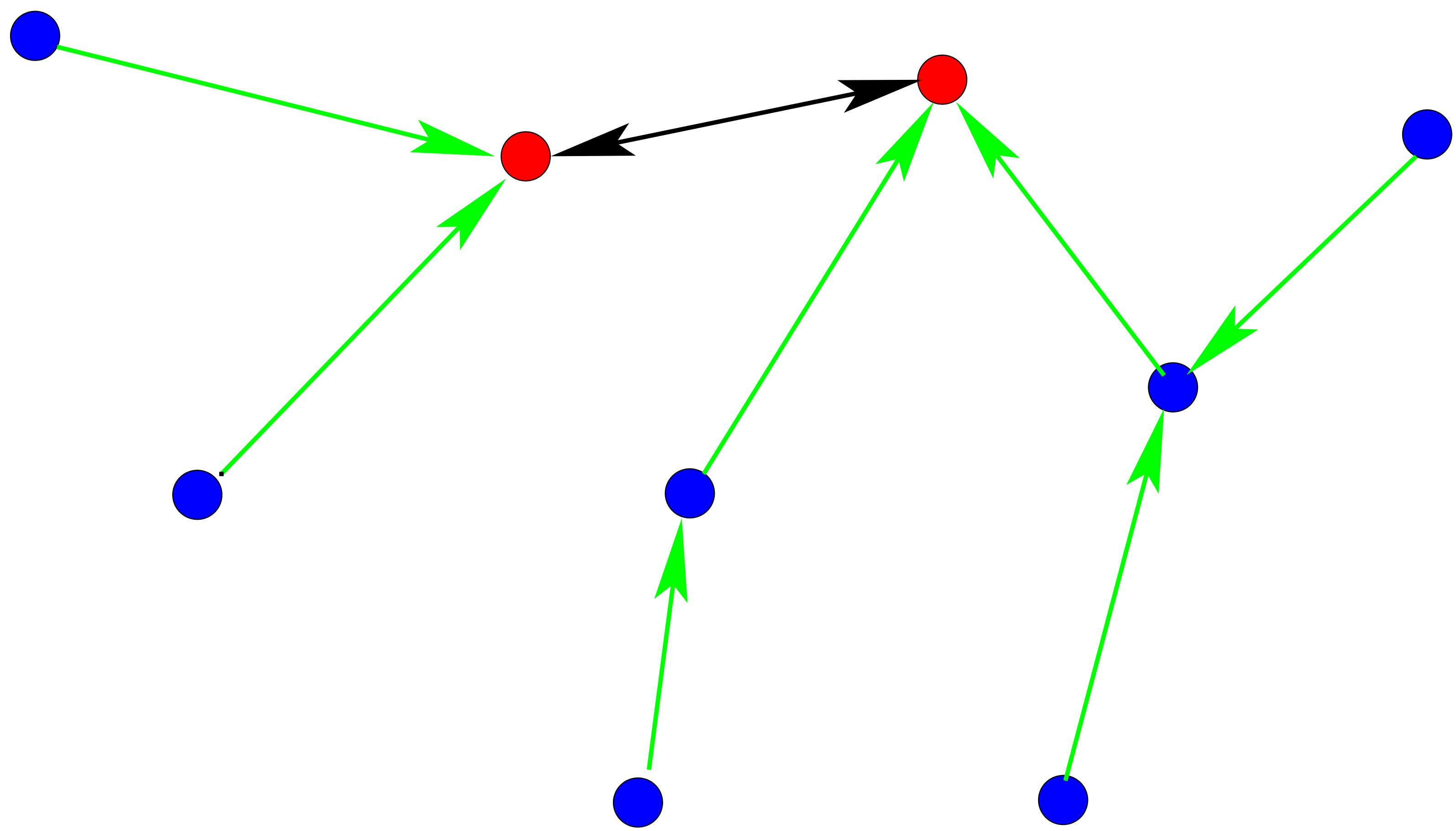
Second Refinement: The problem of contention

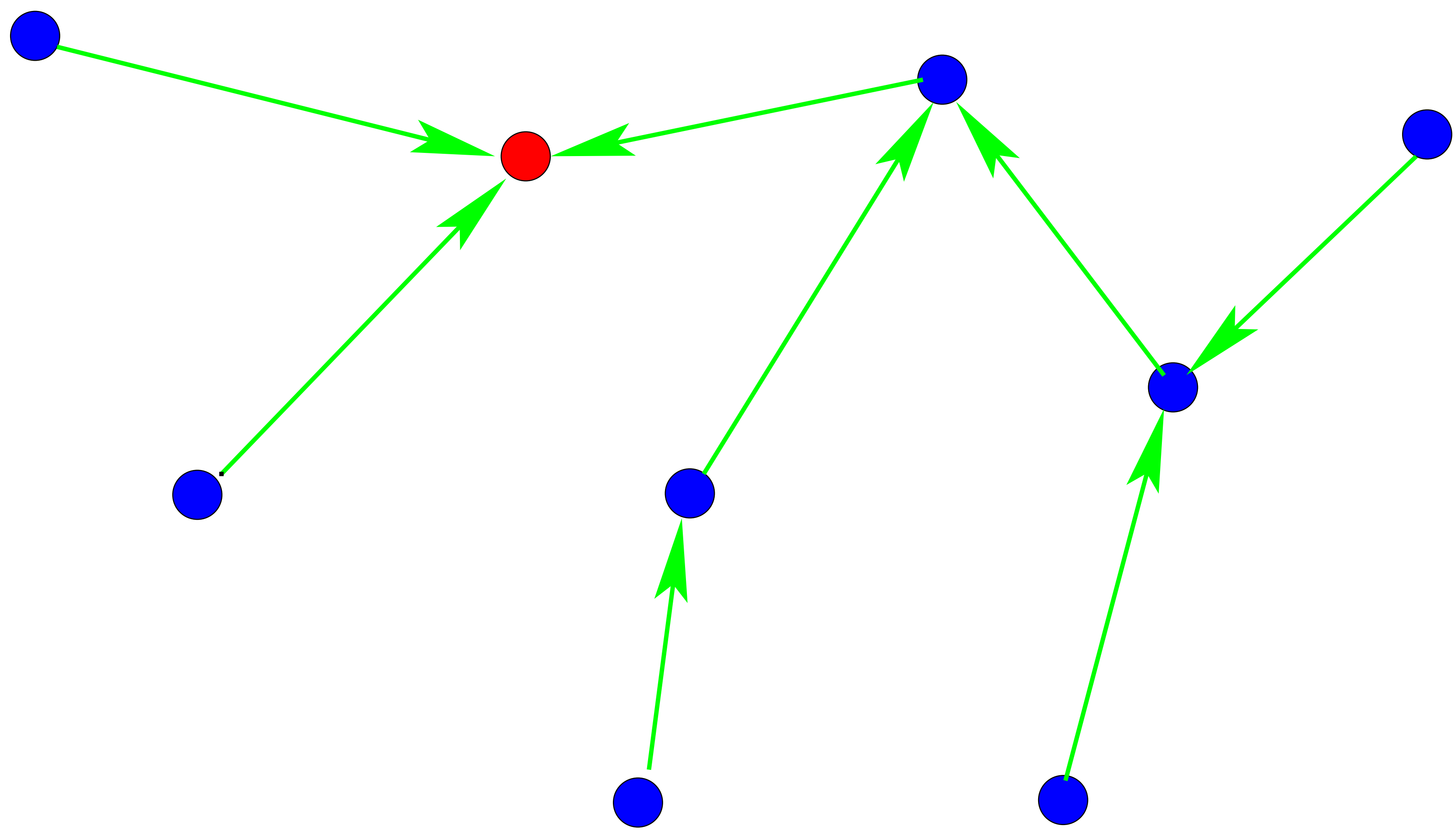
- Explaining the problem

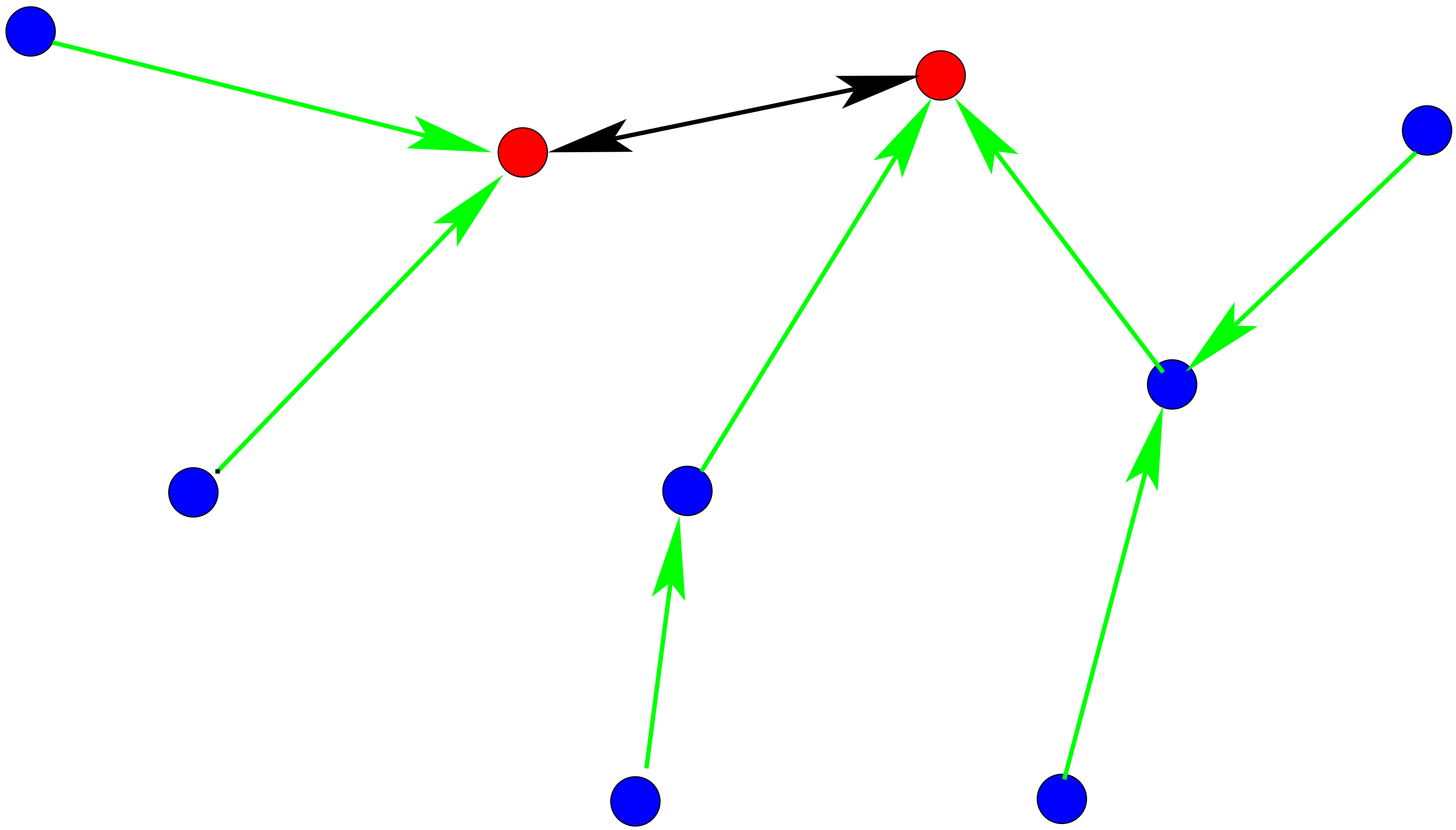
- Proposing a partial solution

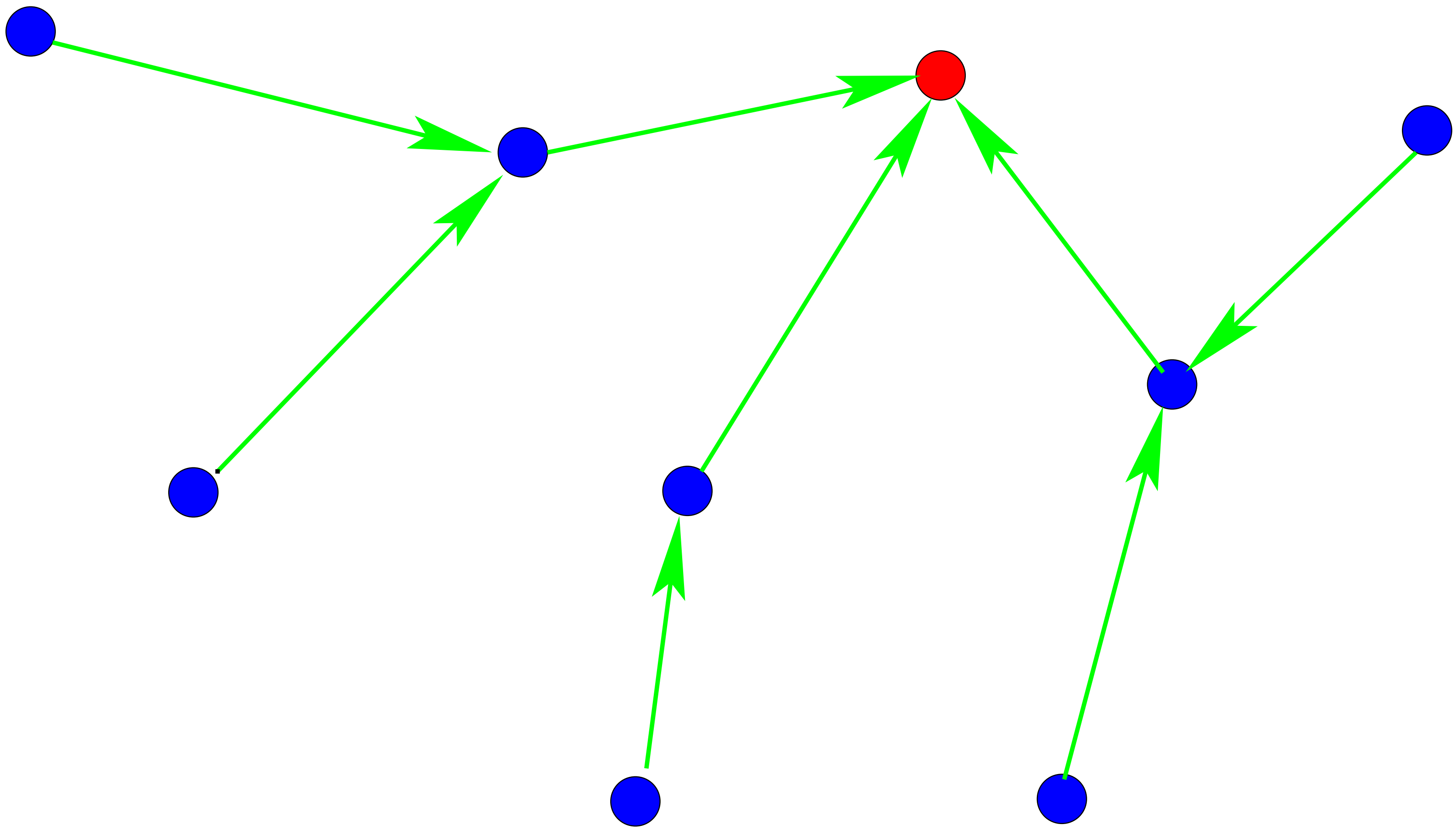
- Towards a better treatment

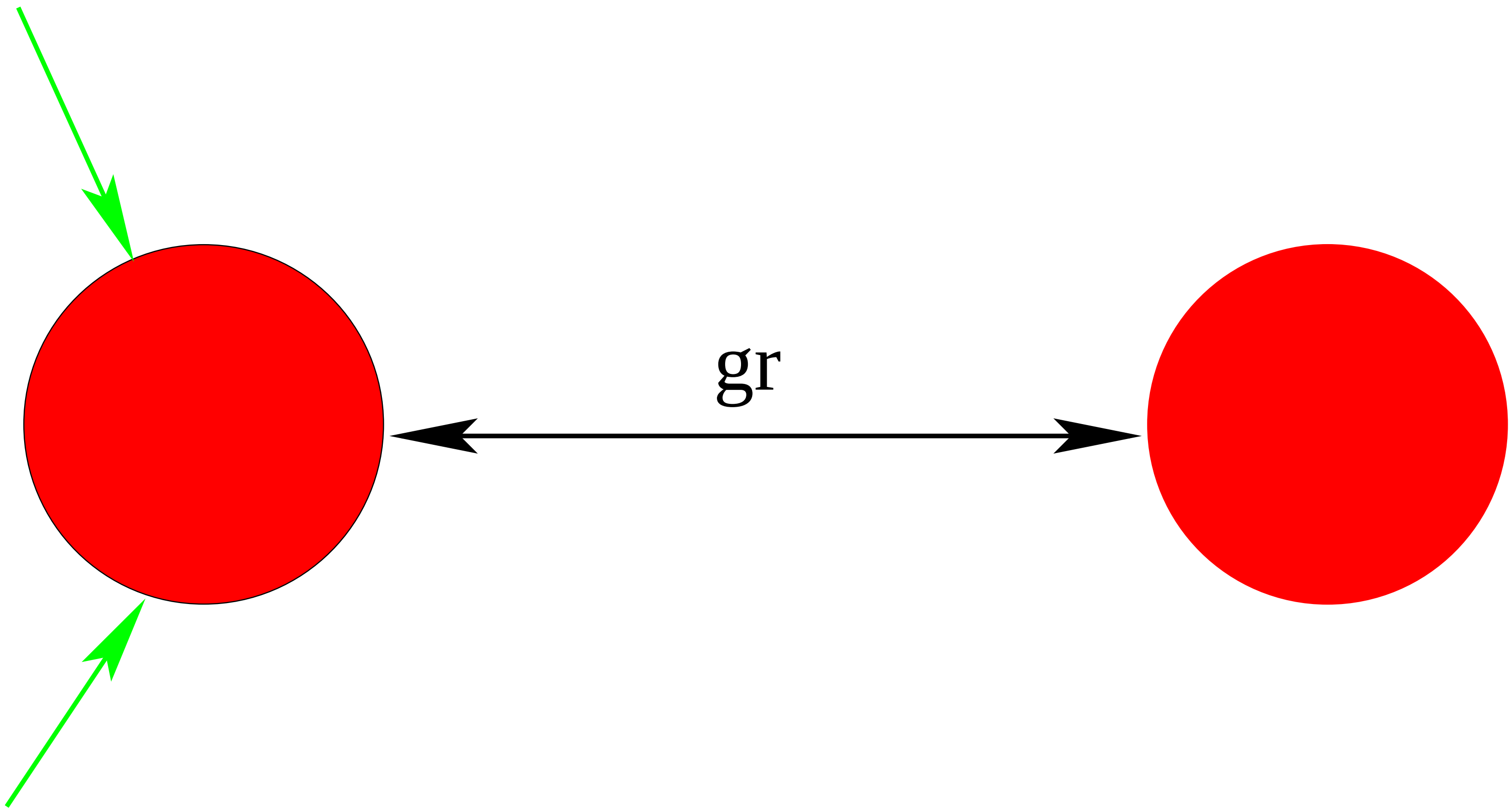
- Back to the local animation



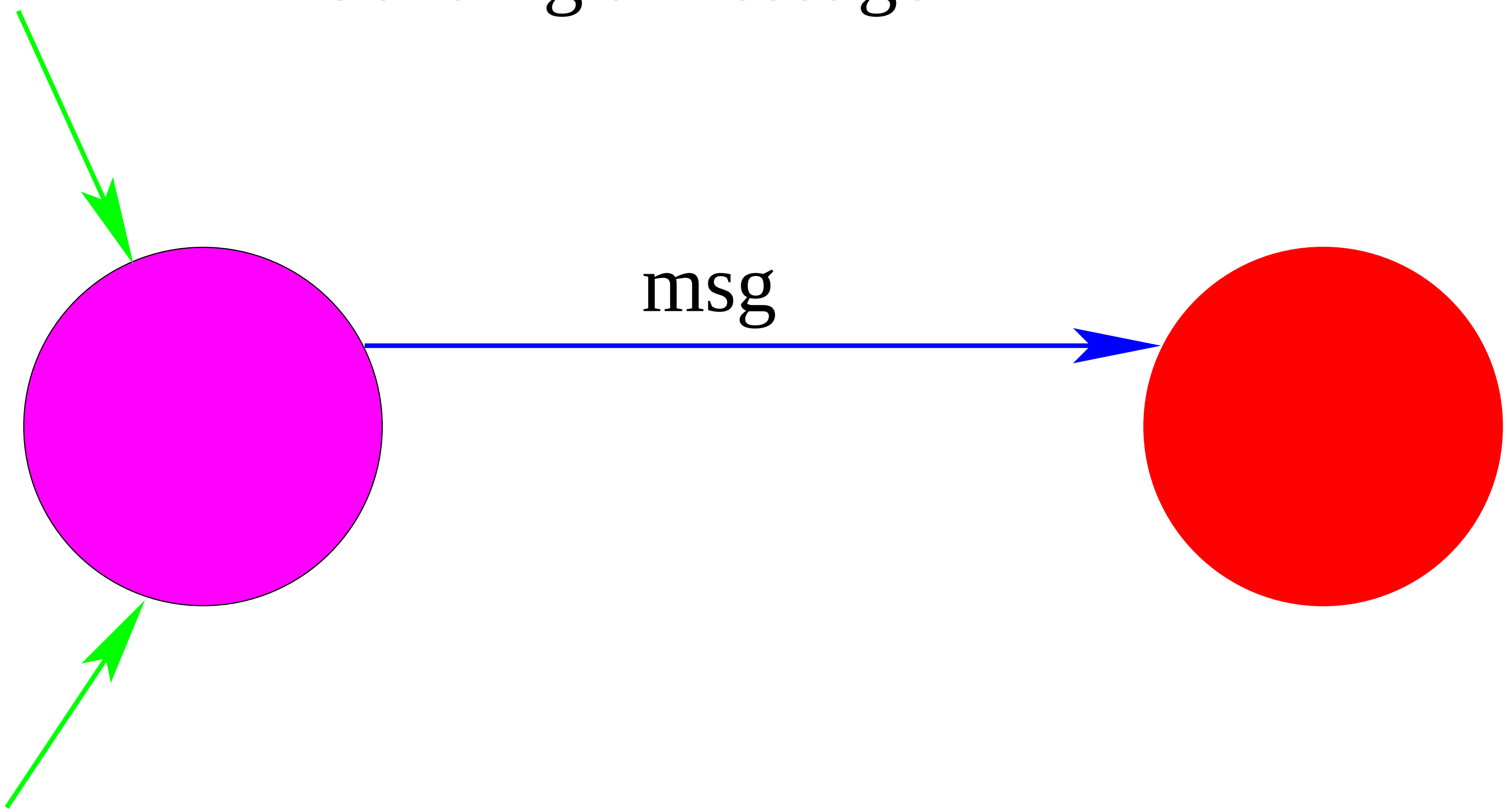




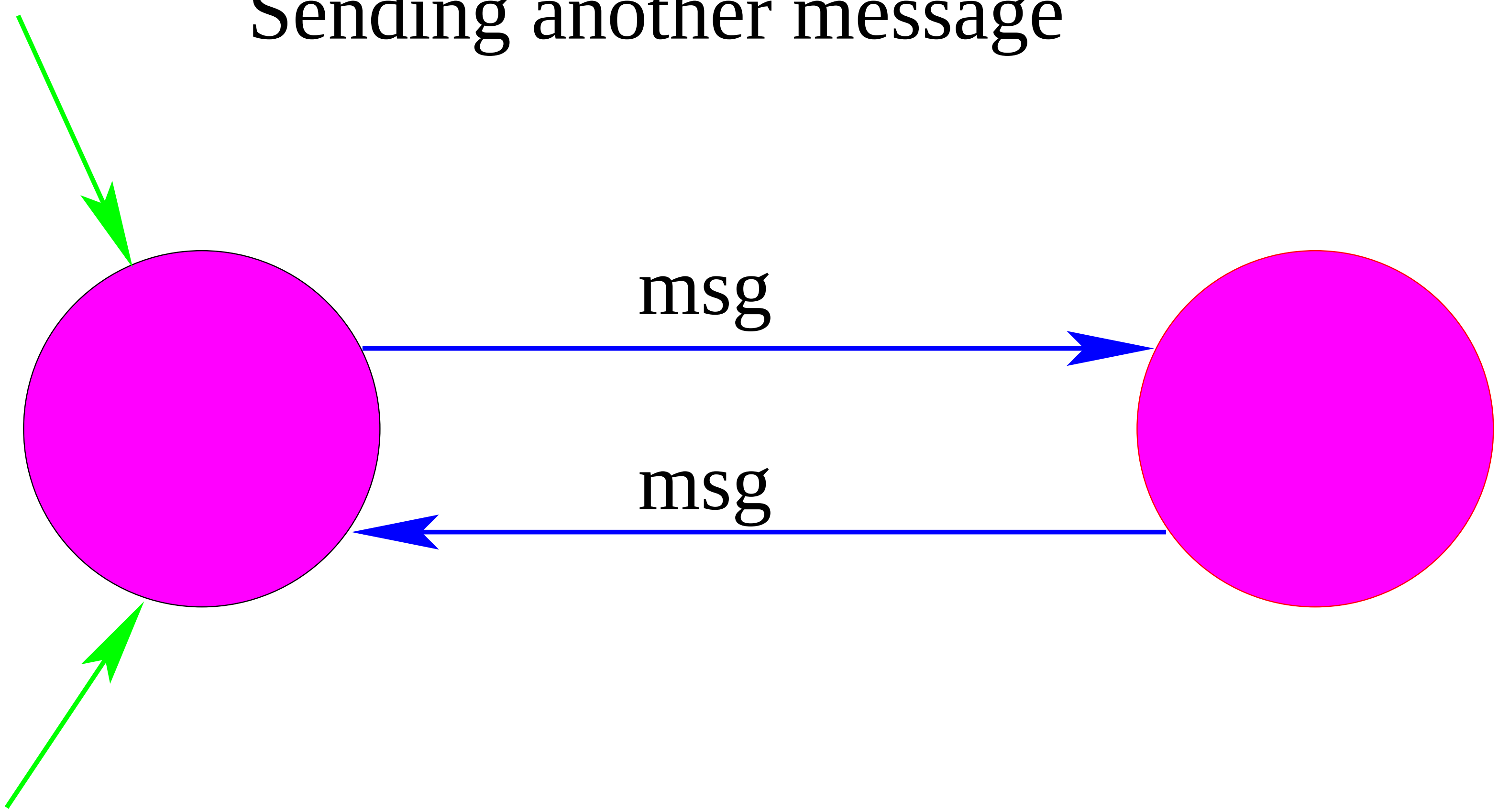



gr

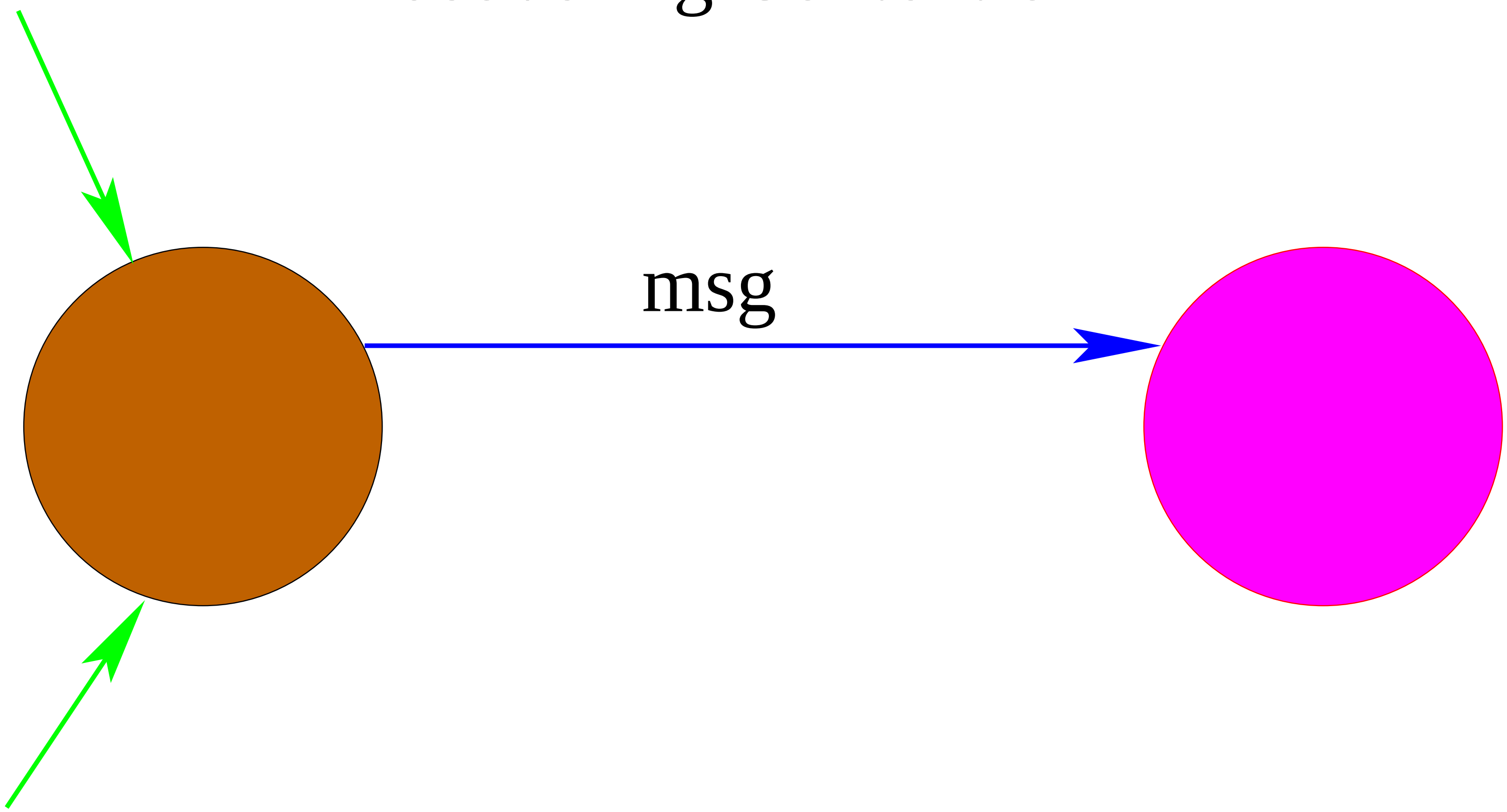
Sending a message



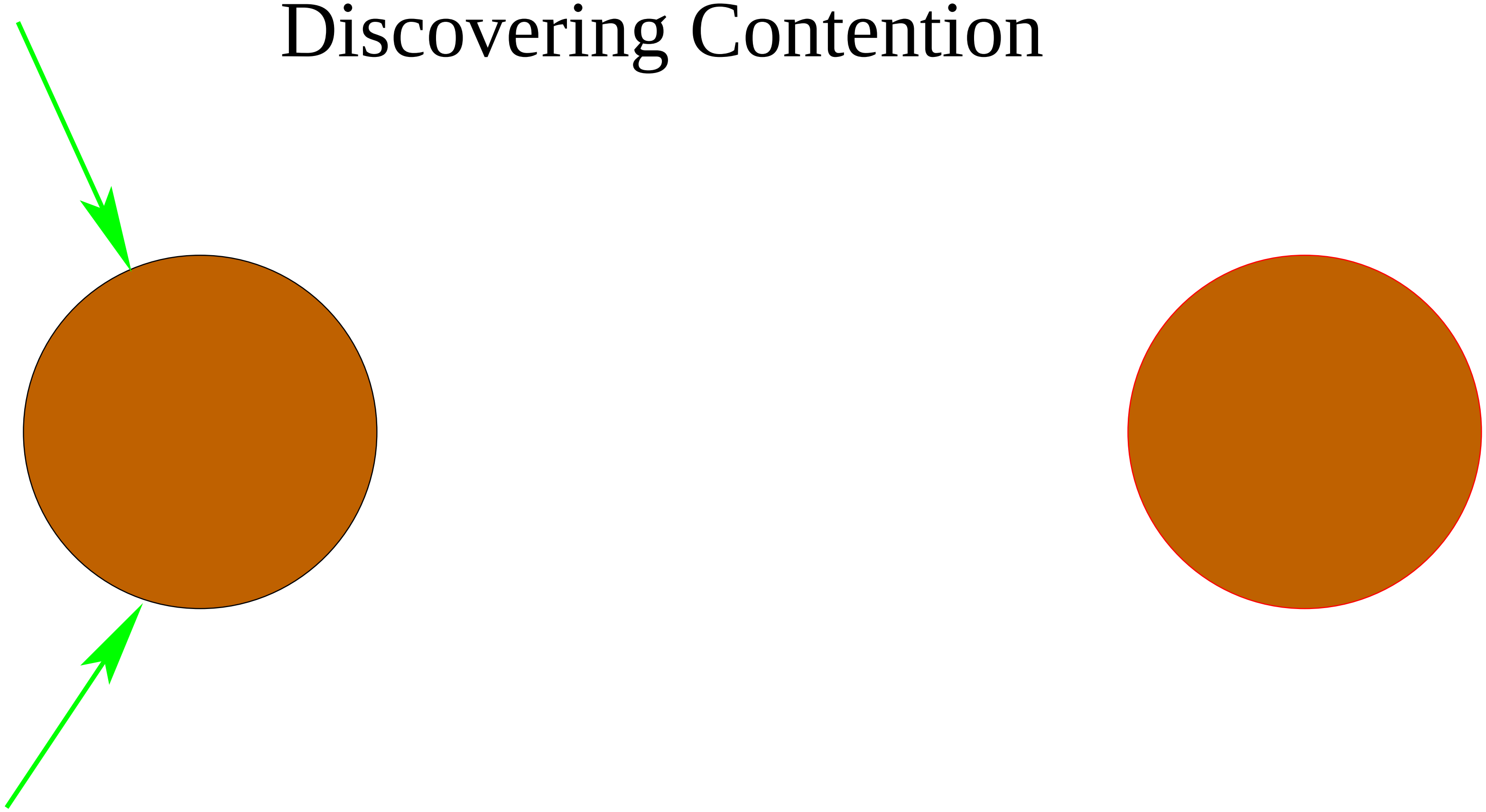
Sending another message



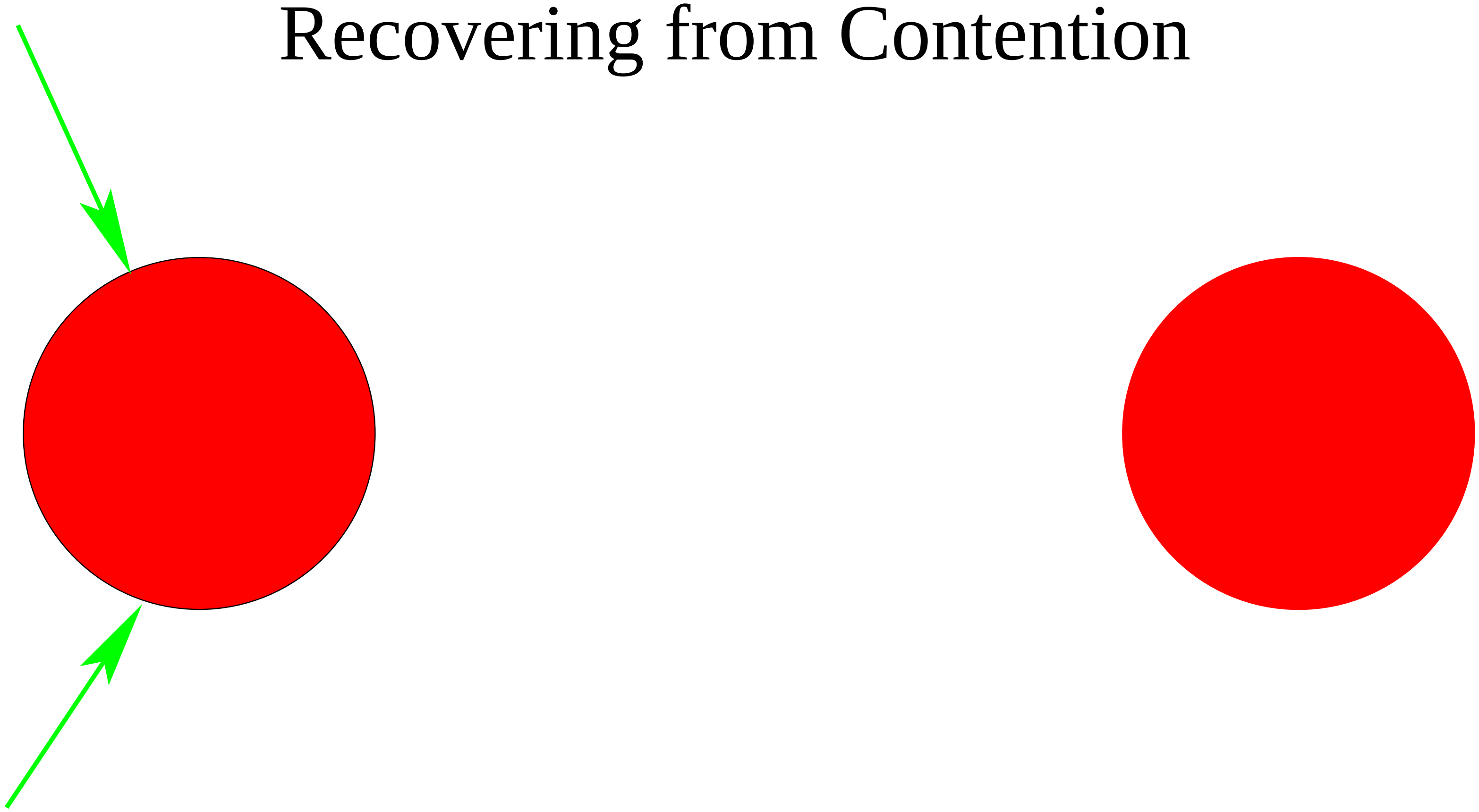
Discovering Contention



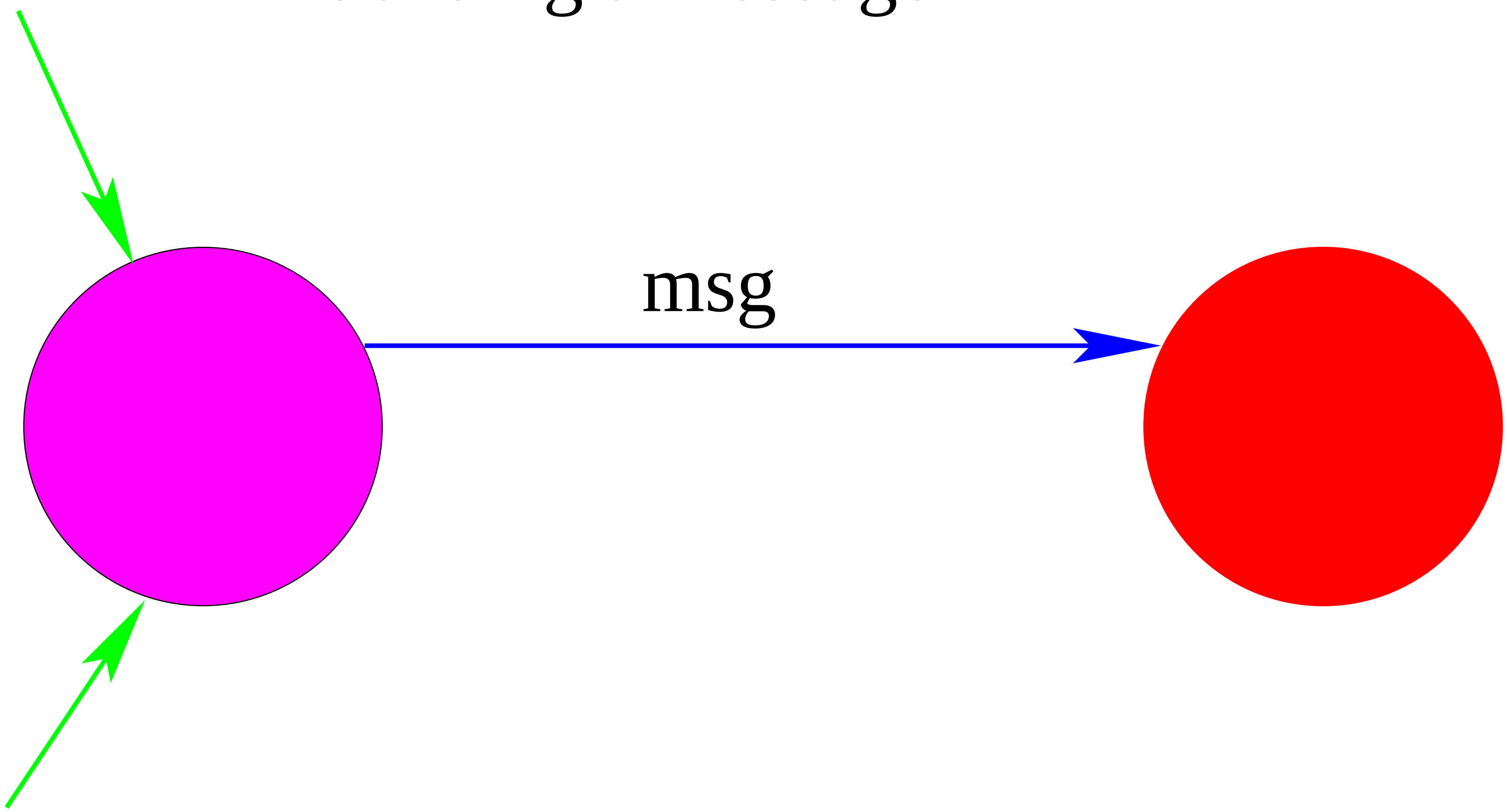
Discovering Contention



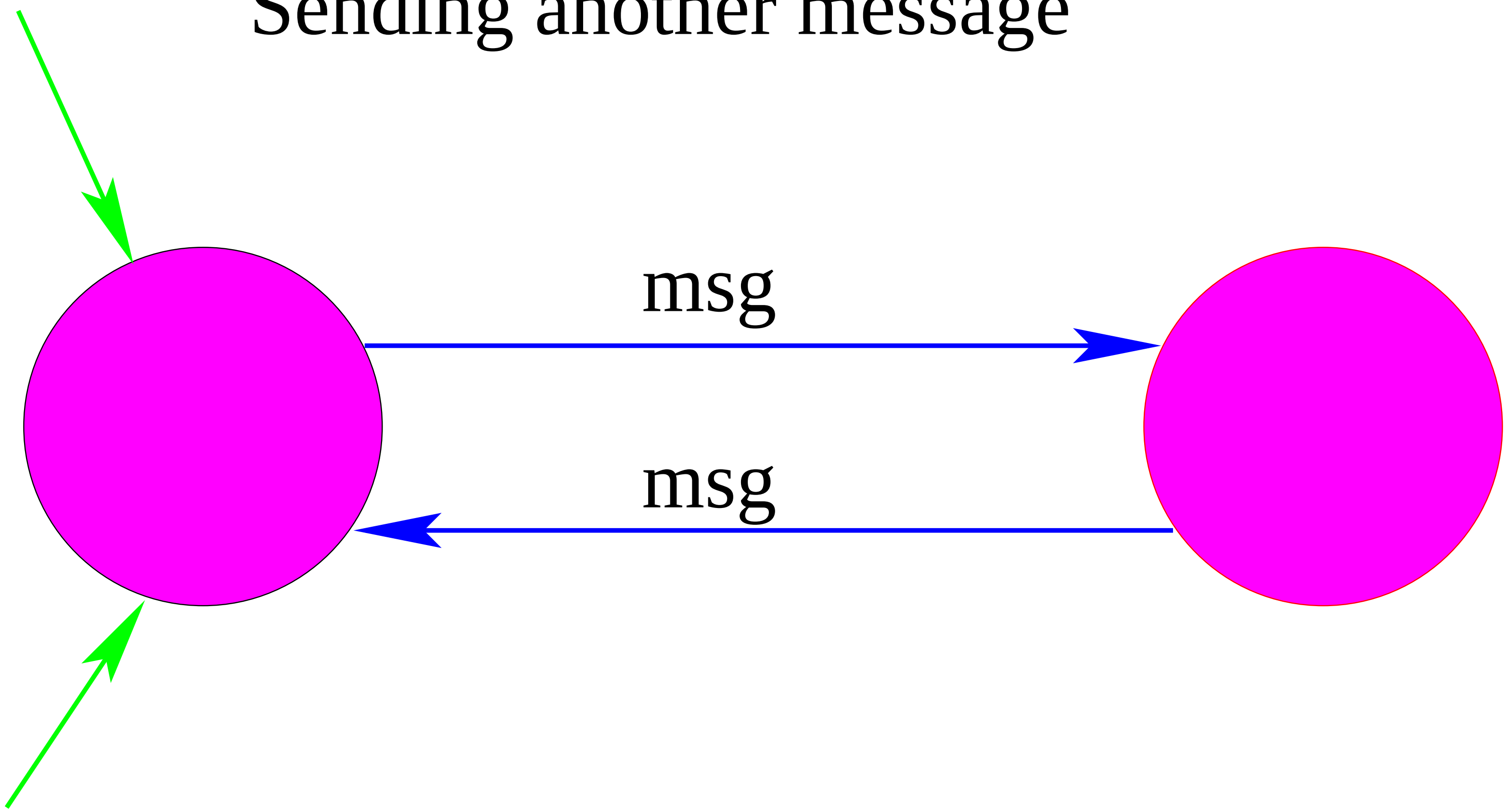
Recovering from Contention



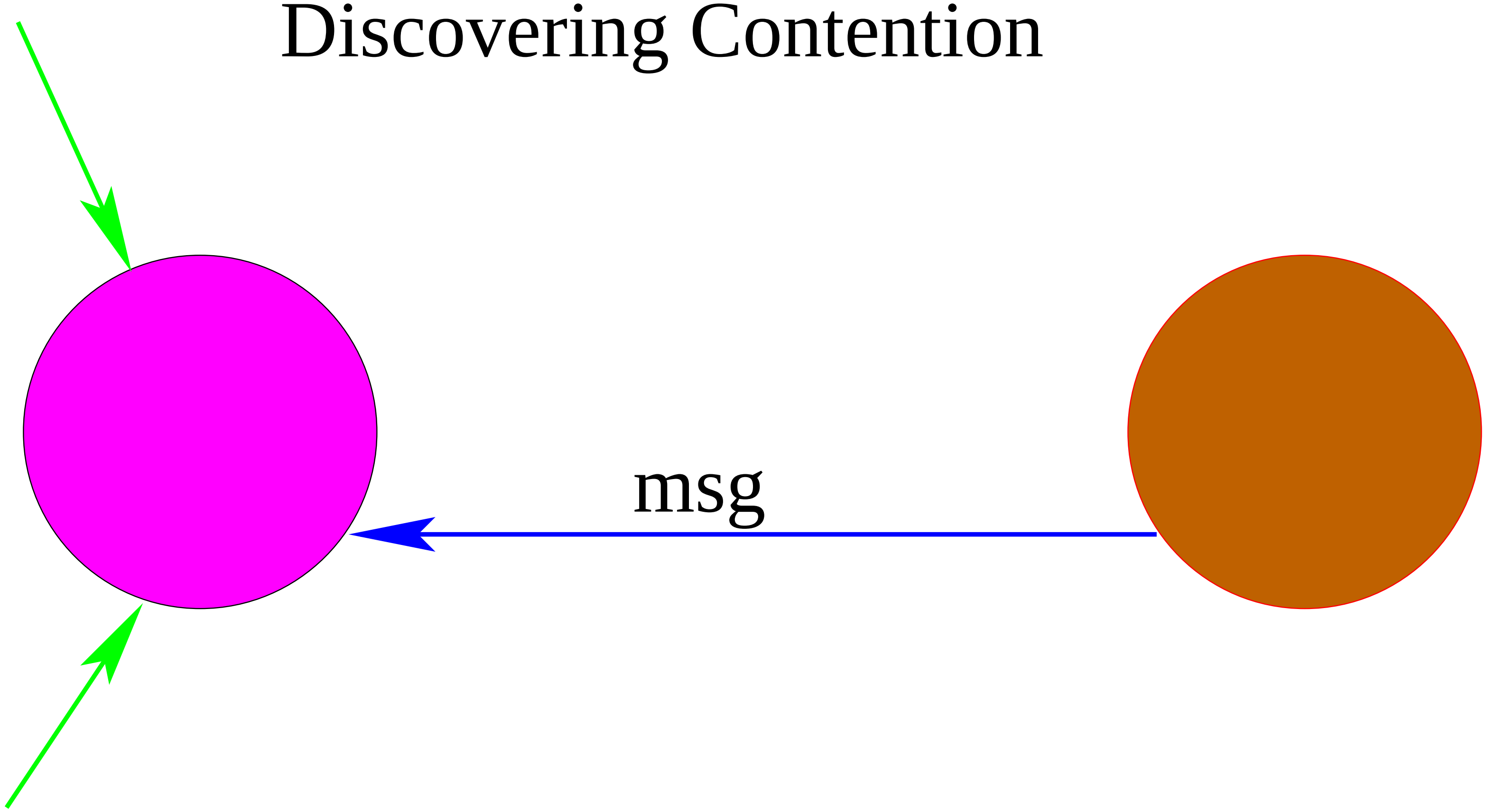
Sending a message



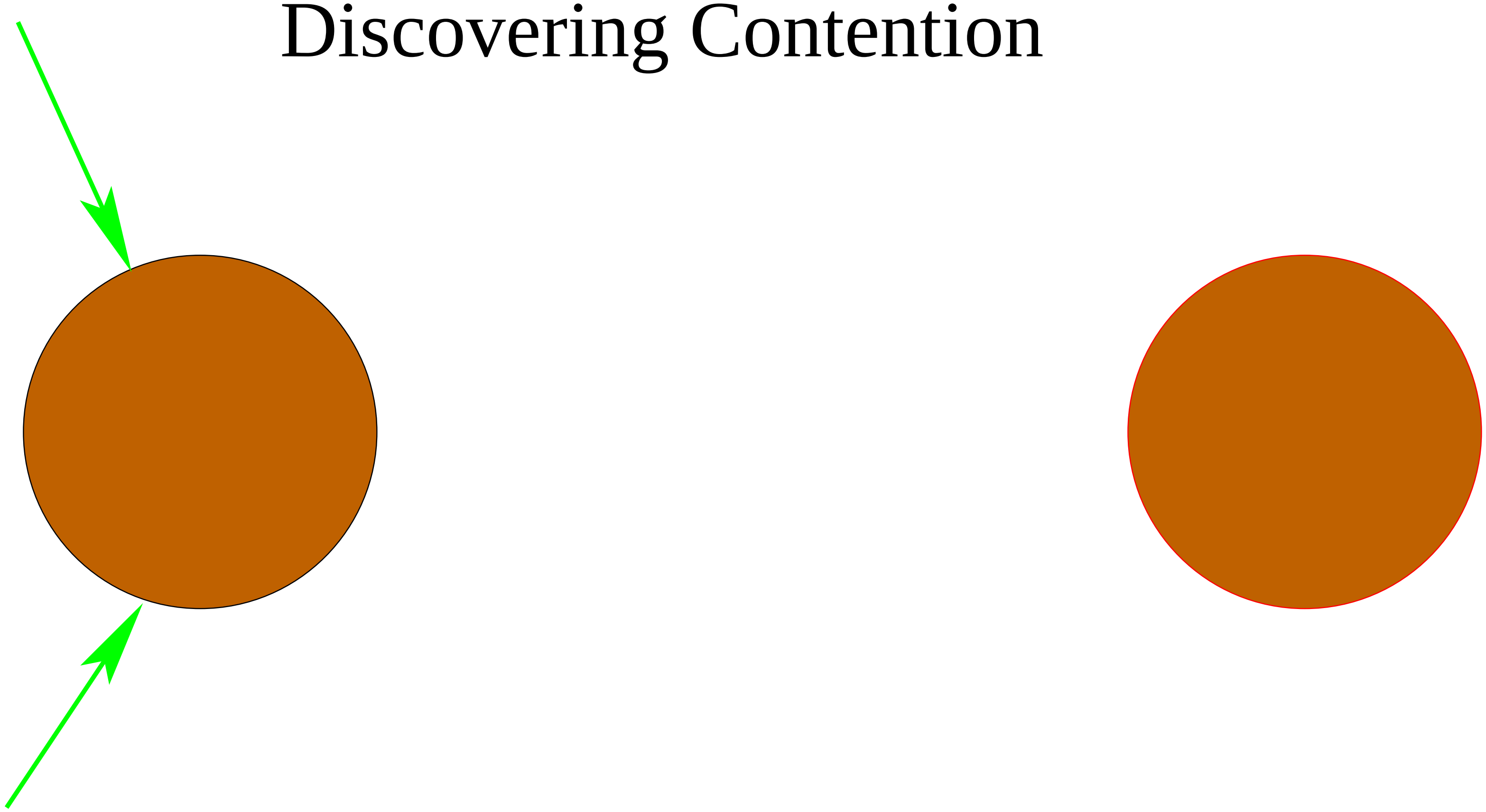
Sending another message



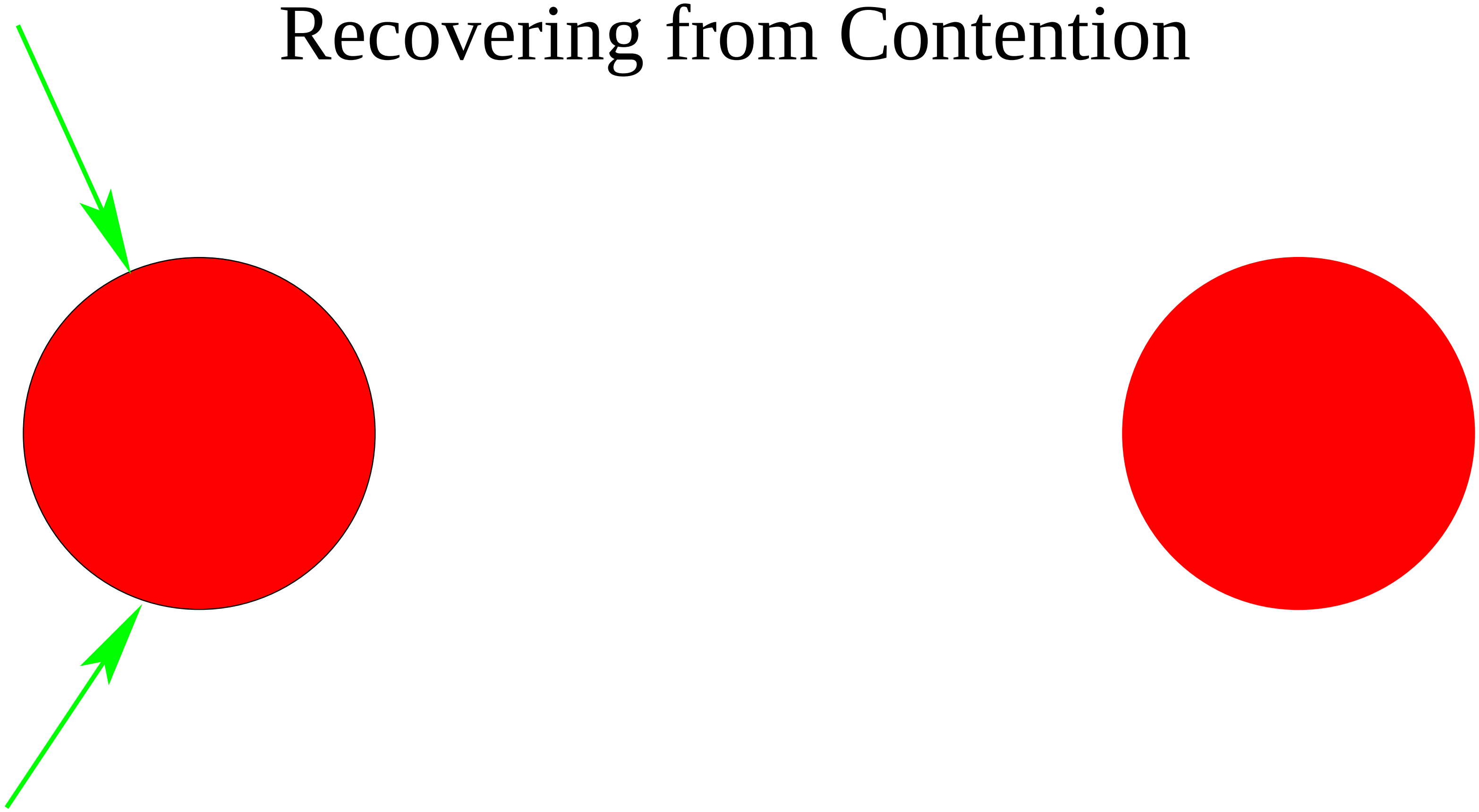
Discovering Contention



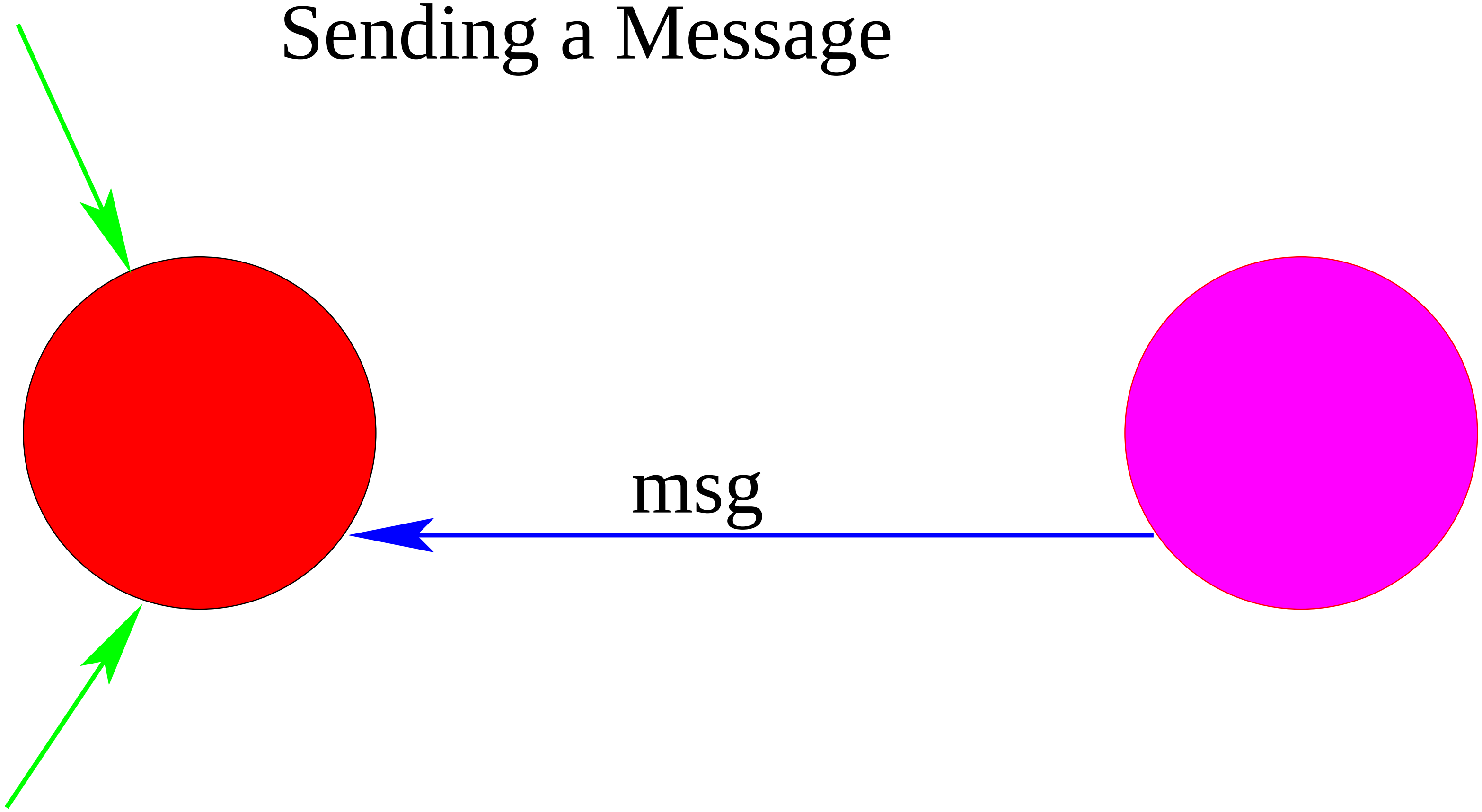
Discovering Contention



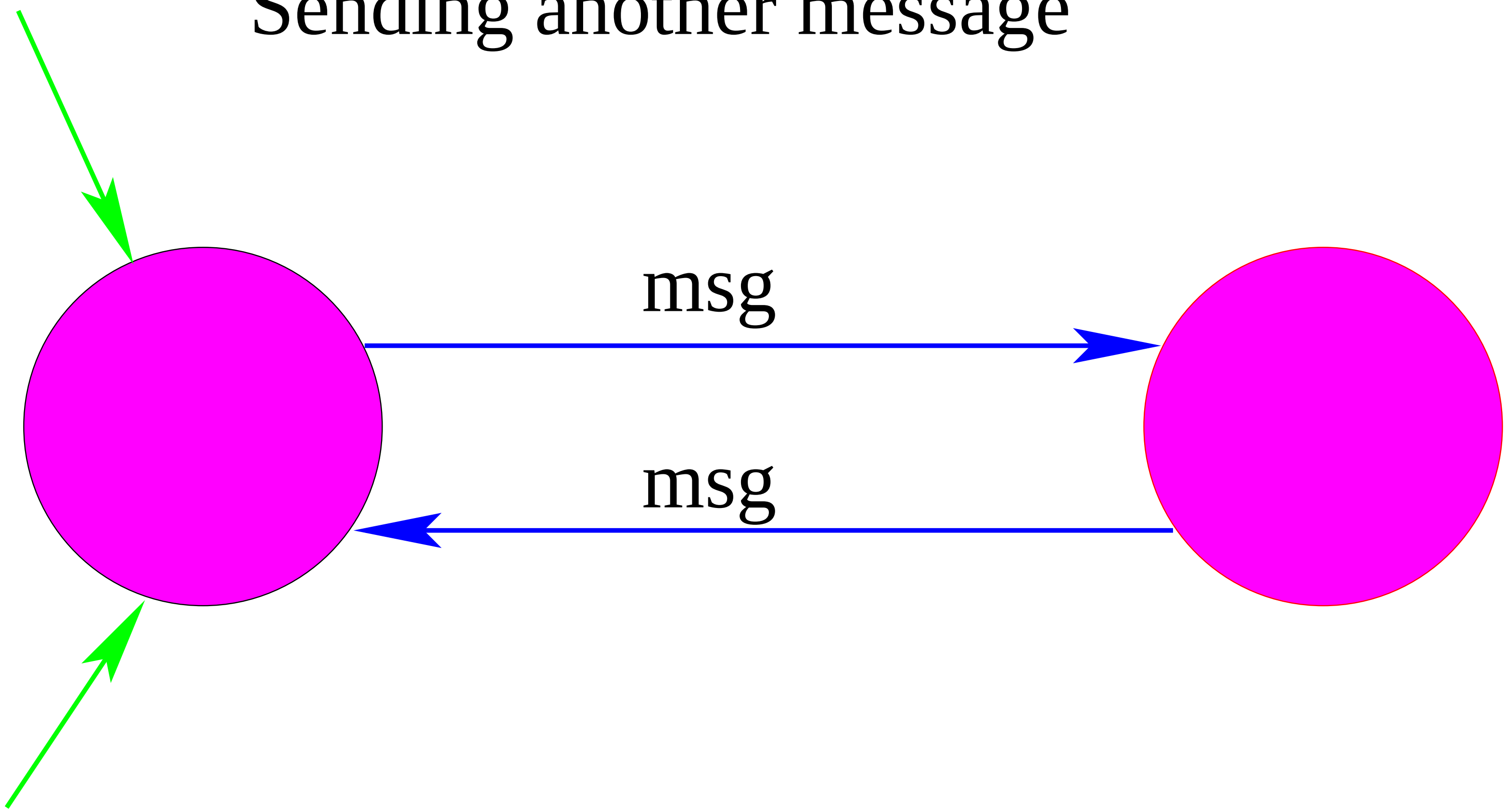
Recovering from Contention



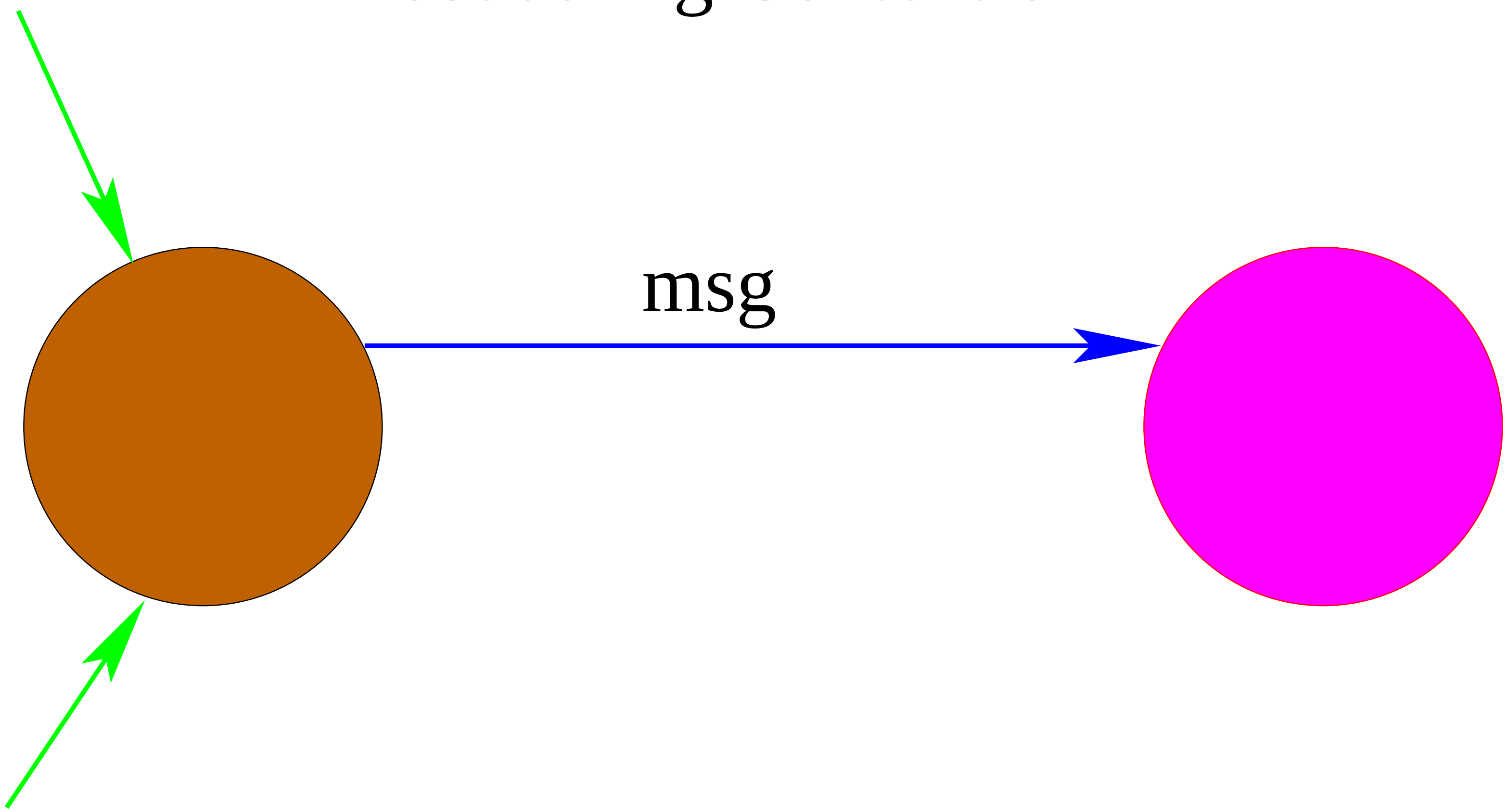
Sending a Message



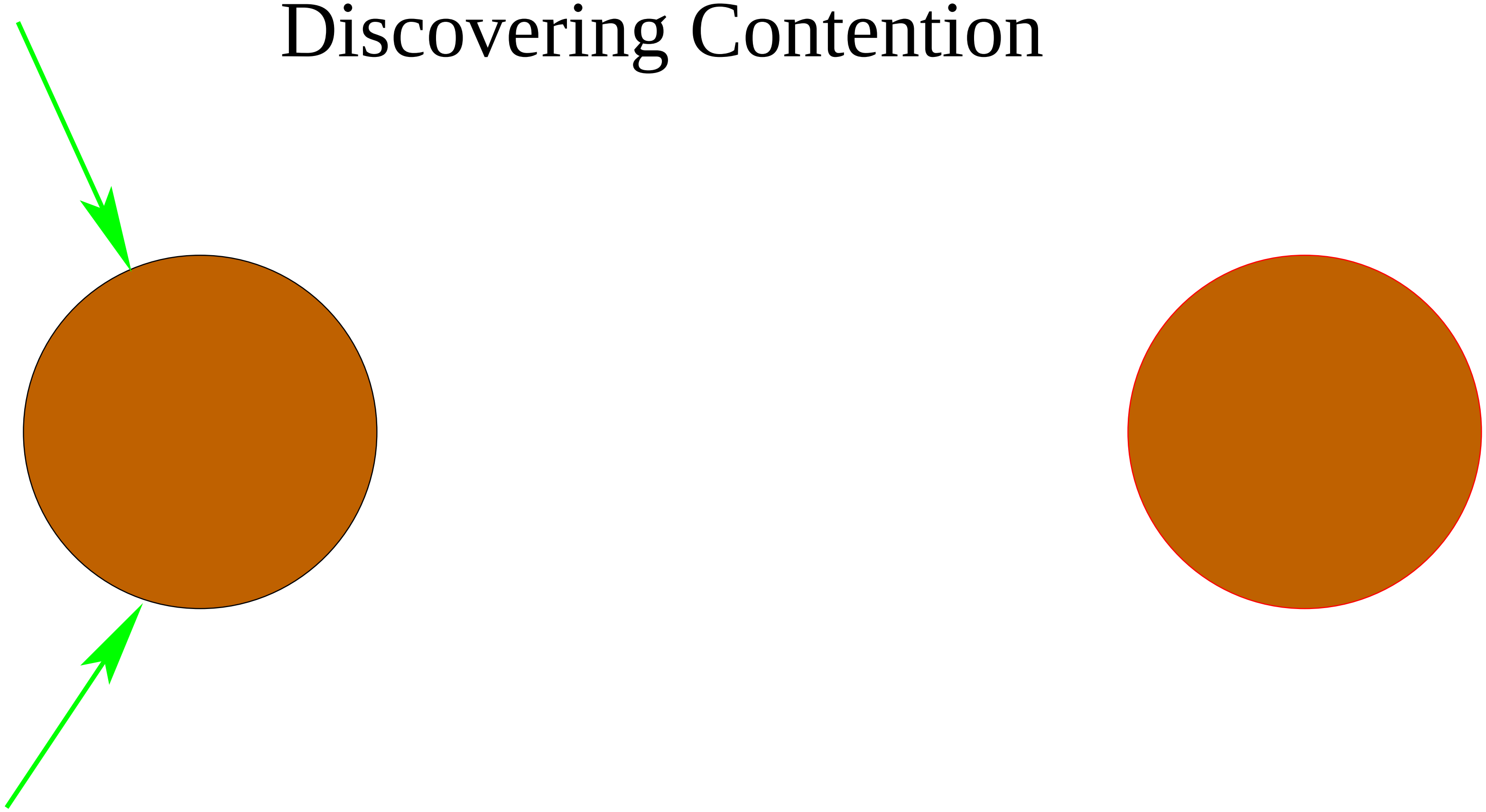
Sending another message



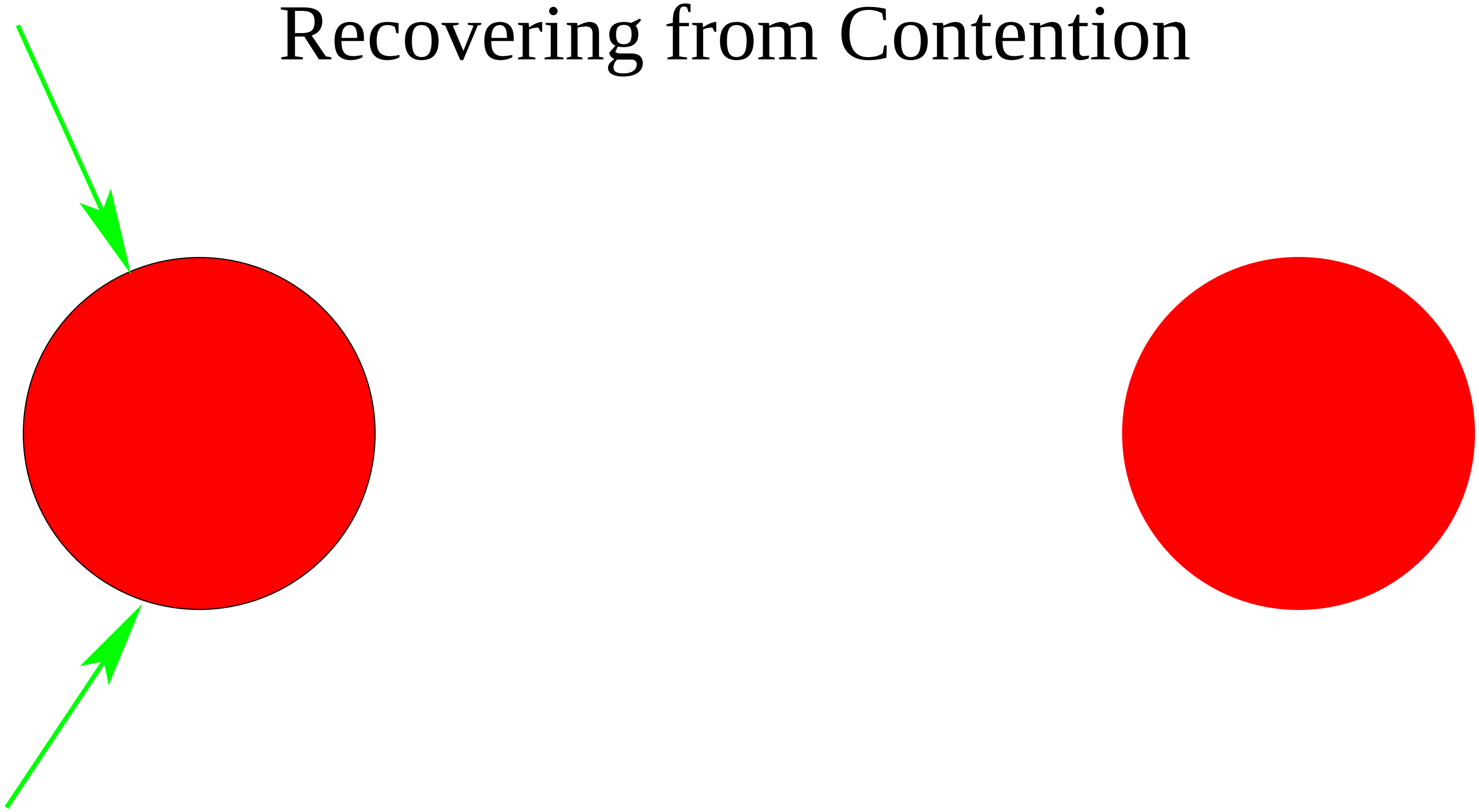
Discovering Contention



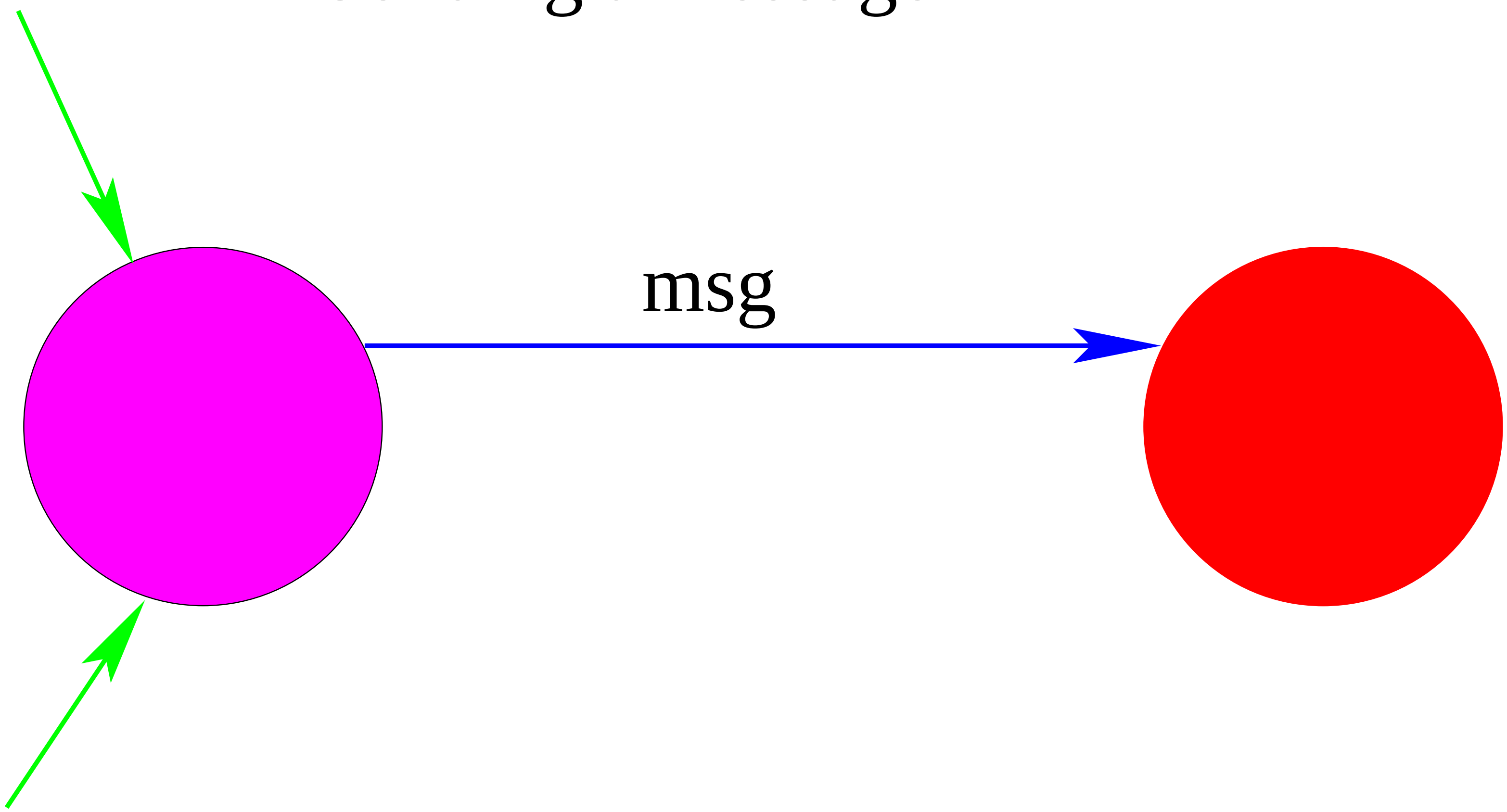
Discovering Contention



Recovering from Contention

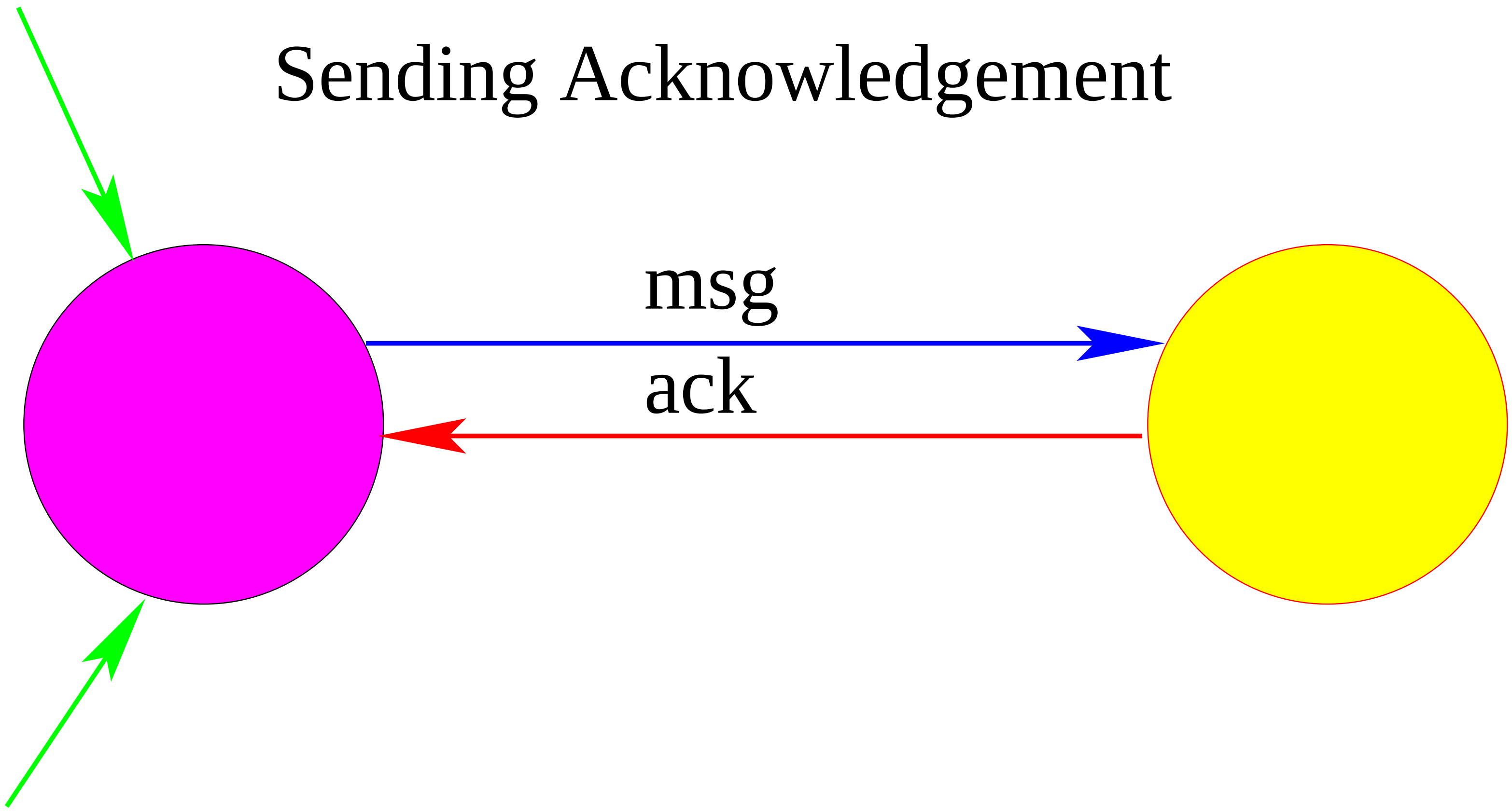


Sending a message



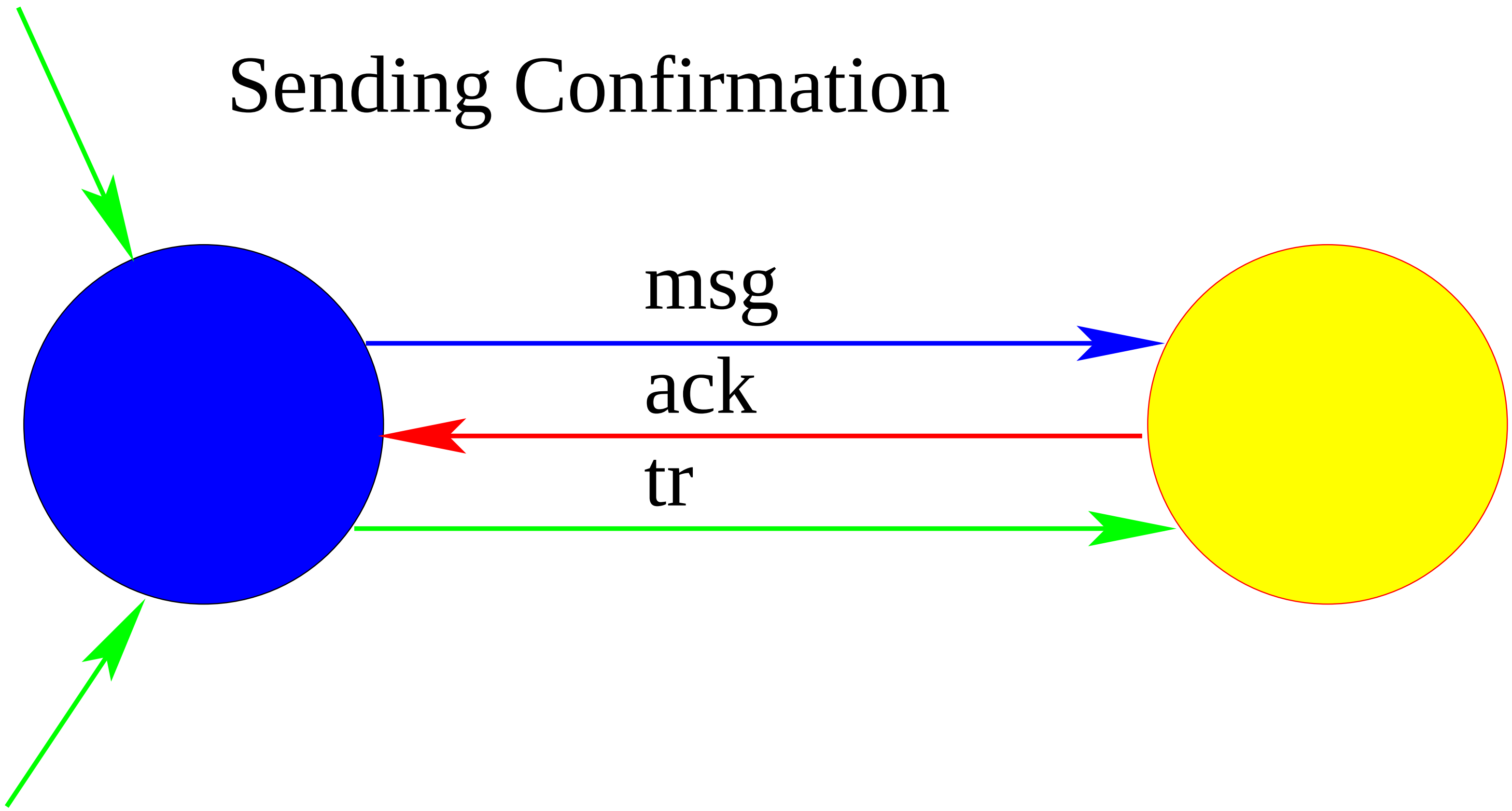
Receiving a message

Sending Acknowledgement

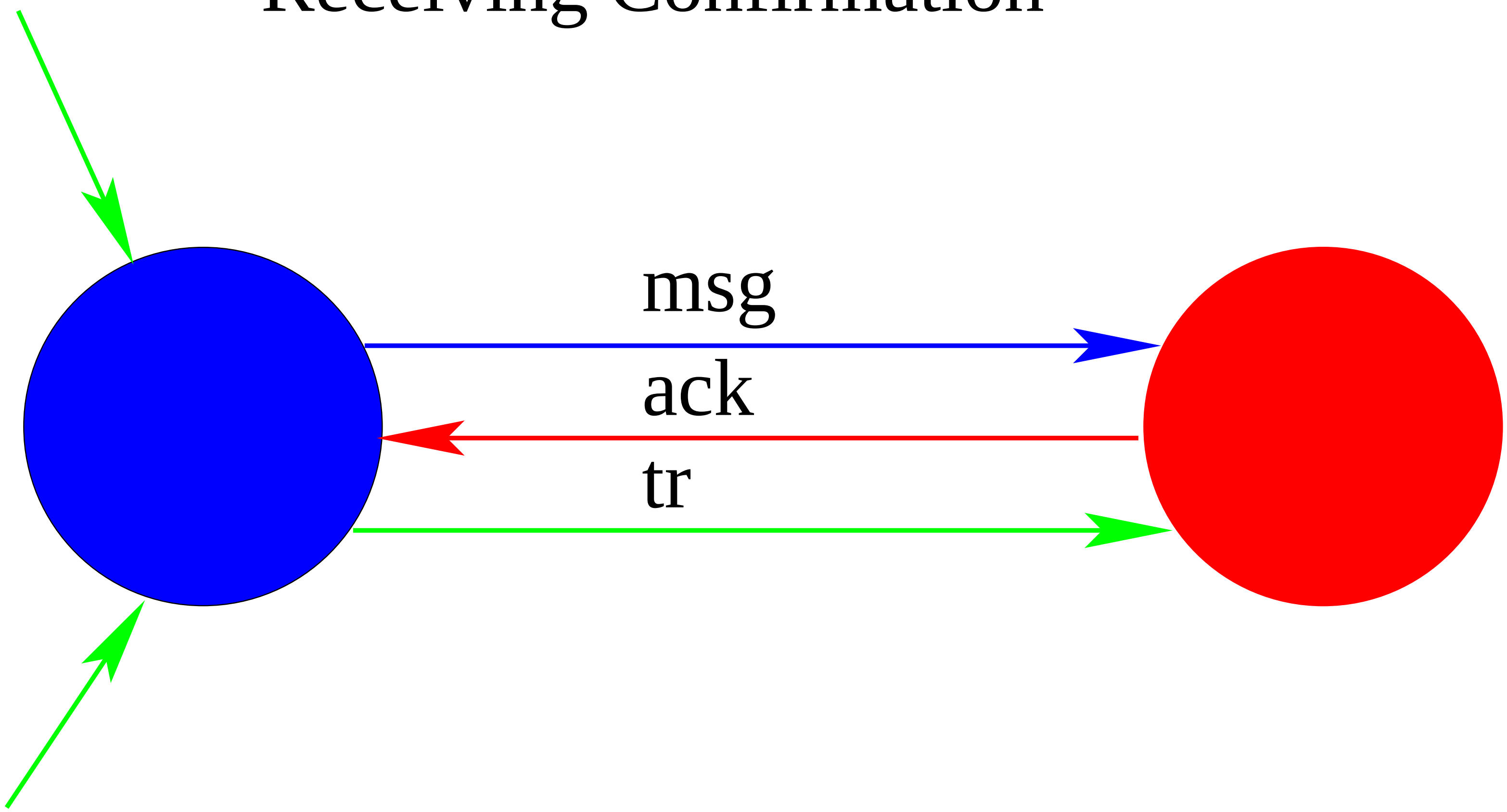


Receiving Acknowledgement

Sending Confirmation



Receiving Confirmation



Discovering the Contention (1)

- Node y discovers the contention with node x because:
  - It has sent a message to node x
  - It has not yet received a response from node x
  - It receives instead a message from node x

Discovering the Contention (2)

- Node x also discovers the contention with node y

- Assumption: The time between both discoveries

IS SUPPOSED TO BE BOUNDED

BY τ ms

- The time τ is the maximum transmission time

between 2 connected nodes

A Partial Solution

- Each node waits for τ ms after its own discovery

- After this, each node thus knows that the other has also discovered the contention

- Each node then retries immediately

- PROBLEM: This may continue for ever

A Better Solution (1)

- Each node waits for τ ms after its own discovery

- Each node then choses with equal probability:
    - either to wait for a short delay
    - or to wait for a large delay

- Each node then retries

A Better Solution (2)

- Question: Does this solves the problem ?

- Are we sure to eventually have one node winning ?

- Answer: Listen carefully to Caroll Morgan’s lectures

Node y discovers a contention with node x

$$
\begin{array}{l}
\text{send\_ack} \quad \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\quad\quad x, y \in msg - ack \;\; \wedge \\
\quad\quad y \notin \text{dom}\,(msg) \\
\quad \textbf{THEN} \\
\quad\quad ack := ack \cup \{x \mapsto y\} \\
\quad \textbf{END}
\end{array}
$$

$$
\begin{array}{l}
\text{contention} \quad \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\quad\quad x, y \in msg - ack \;\; \wedge \\
\quad\quad y \in \text{dom}\,(msg) \\
\quad \textbf{THEN} \\
\quad\quad cnt := cnt \cup \{x \mapsto y\} \\
\quad \textbf{END}
\end{array}
$$

- Introducing a dummy contention channel: cnt

$$cnt \in ND \mathrel{+\!\!\!\rightarrow} ND$$

$$cnt \;\subseteq\; msg$$

$$ack \cap cnt \;=\; \emptyset$$

Solving the contention (simulating the τ delay)

solve_contention $\widehat{=}$

ANY x, y **WHERE**

$x, y \in cnt \cup cnt^{-1}$

THEN

$msg := msg - cnt \quad \|$

$cnt := \emptyset$

END

Summary of Second Refinement

- 73 proofs

- Among which 34 were interactive

Third Refinement: Localization

- The representation of the graph gr is modified

- The representation of the tree tr is modified

- Other data structures are localized

Localization (1)

The graph gr and the tree tr are now localized

$$nb \in ND \rightarrow \mathbb{P}(ND)$$

$$\forall x \cdot (\, x \in ND \Rightarrow nb(x) = gr[\{x\}] \,)$$

$$sn \in ND \rightarrow \mathbb{P}(ND)$$

$$\forall x \cdot (\, x \in ND \Rightarrow sn(x) \subseteq tr^{-1}[\{x\}] \,)$$

Localization (2)

$$bm \subseteq ND$$

$$bm = \mathsf{dom}\,(msg)$$

$$bt \subseteq ND$$

$$bt = \mathsf{dom}\,(tr)$$

$$ba \in ND \rightarrow \mathbb{P}(ND)$$

$$\forall x \cdot (\, x \in ND \Rightarrow ba(x) = ack^{-1}[\{x\}]\,)$$

- Node x is elected the leader

elect $\widehat{=}$

ANY x **WHERE**

$x \in ND \;\wedge$

$nb(x) = sn(x)$

THEN

$rt := x$

END

- Node x sends a message to node y (y is unique)

send_msg $\widehat{=}$

ANY x, y **WHERE**

$x \in ND - bm \ \wedge$

$y \in ND - (ba(x) \cup sn(x)) \ \wedge$

$nb(x) = sn(x) \cup \{y\}$

THEN

$msg := msg \cup \{x \mapsto y\} \quad \|$

$bm := bm \cup \{x\}$

END

- Node y sends an acknowledgement to node x

send_ack $\widehat{=}$

ANY x, y **WHERE**

$x, y \in msg \;\wedge$

$x \notin ba(y) \;\wedge$

$y \notin bm$

THEN

$ack := ack \cup \{x \mapsto y\} \quad \|$

$ba(y) := ba(y) \cup \{x\}$

END

- Node x sends a confirmation to node y

progress $\mathrel{\widehat{=}}$

ANY x, y **WHERE**

$x, y \in ack \ \wedge$

$x \notin bt$

THEN

$tr := tr \cup \{x \mapsto y\} \quad \|$

$bt := bt \cup \{x\}$

END

- Node y receives confirmation from node x

rcv_cnf $\widehat{=}$

ANY x, y **WHERE**

$x, y \in tr \;\wedge$

$x \notin sn(y)$

THEN

$sn(y) := sn(y) \cup \{x\}$

END

contention $\widehat{=}$

ANY x, y **WHERE**

$x, y \in cnt \cup cnt^{-1} \ \wedge$

$x \notin ba(y) \ \wedge$

$y \in bm$

THEN

$cnt := cnt \cup \{x \mapsto y\}$

END

solve_contention $\widehat{=}$

ANY x, y **WHERE**

$x, y \in cnt \cup cnt^{-1}$

THEN

$msg := msg - cnt \quad \|$

$bm := bm - \mathrm{dom}\,(cnt) \quad \|$

$cnt := \emptyset$

END

Summary of Third Refinement

- 29 proofs

- Among which 19 were interactive

Main Summary

- 119 proofs

- Among which 63 were interactive

Conclusion: a Systematic Approach to Distribution

- Establishing the mathematical framework

Conclusion: a Systematic Approach to Distribution

- Establishing the mathematical framework

- Resolving the mathematical problem in one shot

Conclusion: a Systematic Approach to Distribution

- Establishing the mathematical framework

- Resolving the mathematical problem in one shot

- Resolving the same problem on a step by step basis

Conclusion: a Systematic Approach to Distribution

- Establishing the mathematical framework

- Resolving the mathematical problem in one shot

- Resolving the same problem on a step by step basis

- Involving communication by means of messages

Conclusion: a Systematic Approach to Distribution

- Establishing the mathematical framework

- Resolving the mathematical problem in one shot

- Resolving the same problem on a step by step basis

- Involving communication by means of messages

- Towards the localization of data structures