The Leader Election Protocol (IEEE 1394)

J.R. Abrial, D. Cansell, D. Méry

July 2002

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 Abstract Properties of the Protocol

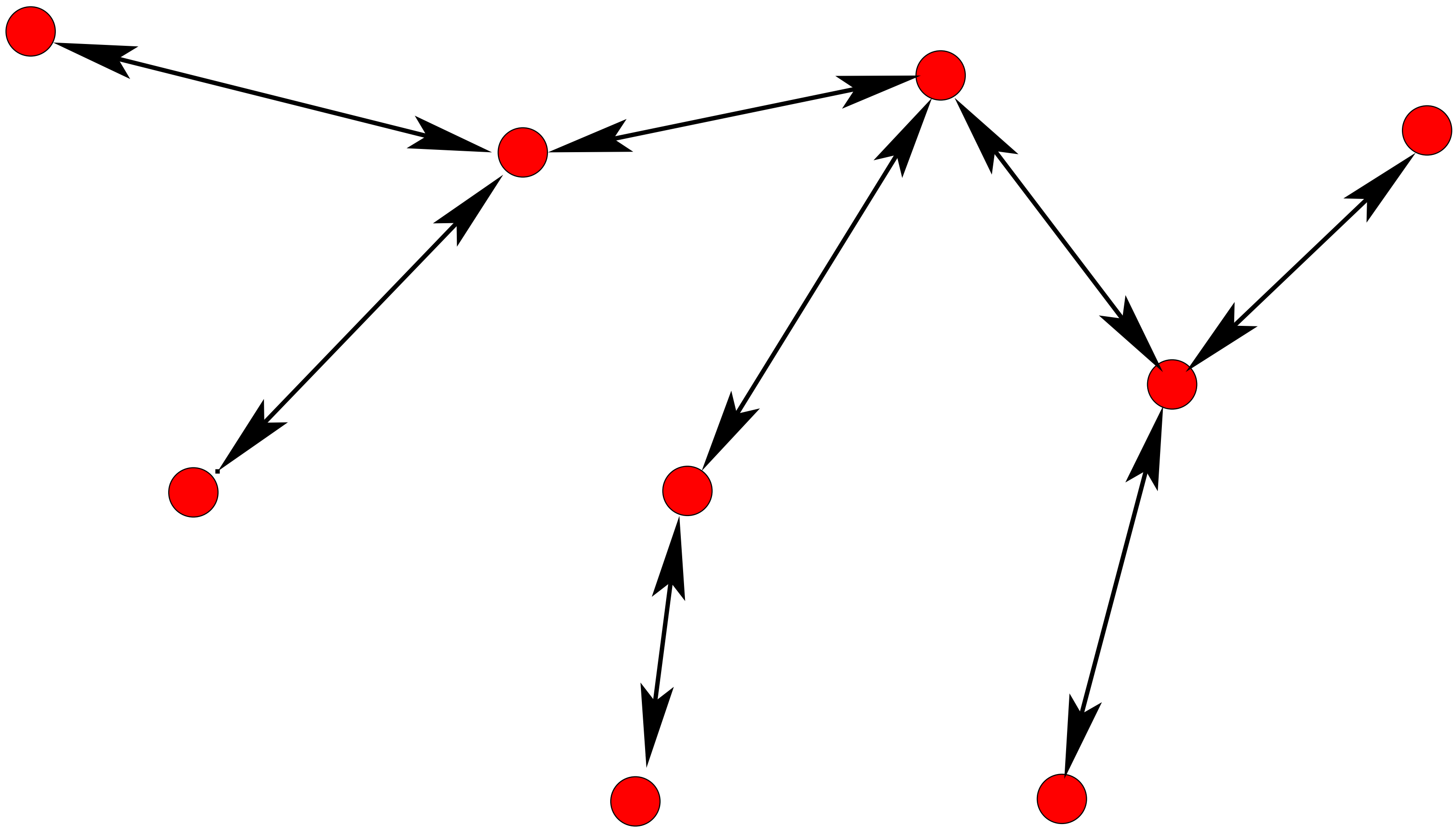
- We are given a connected and acyclic network of nodes

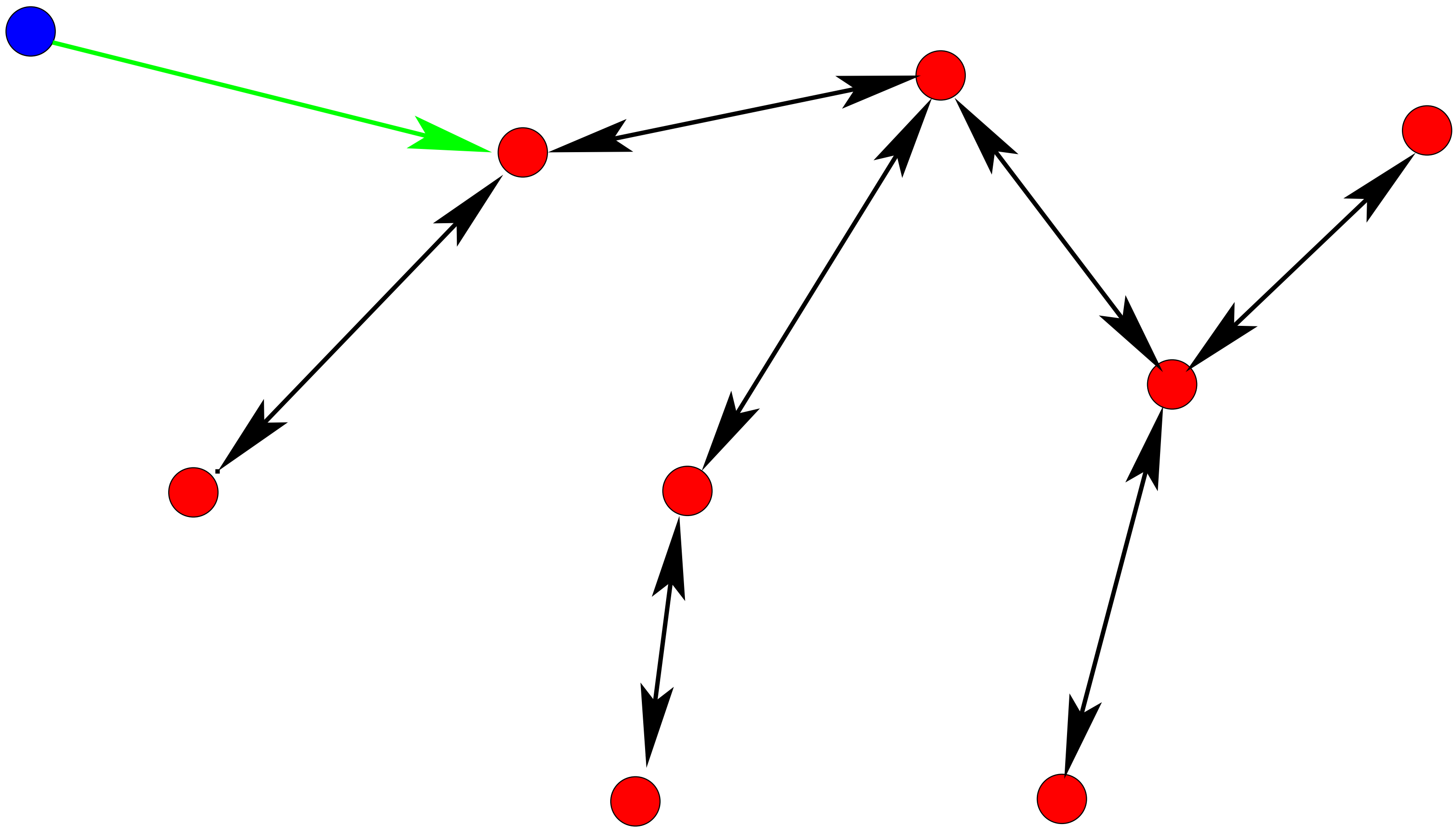
- Nodes are linked by bidirectional channels

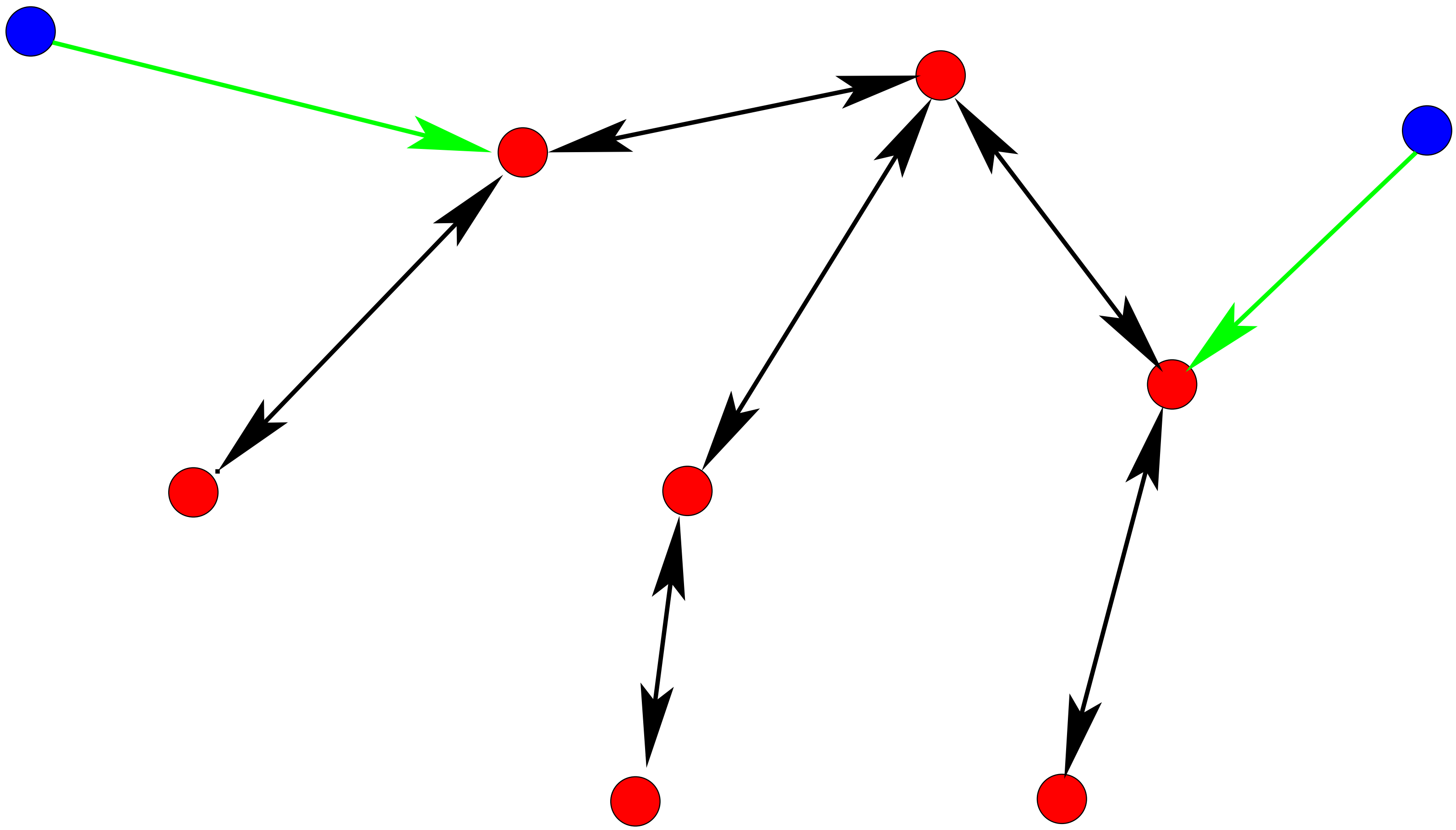
- We want to have one node being elected the leader in a finite time

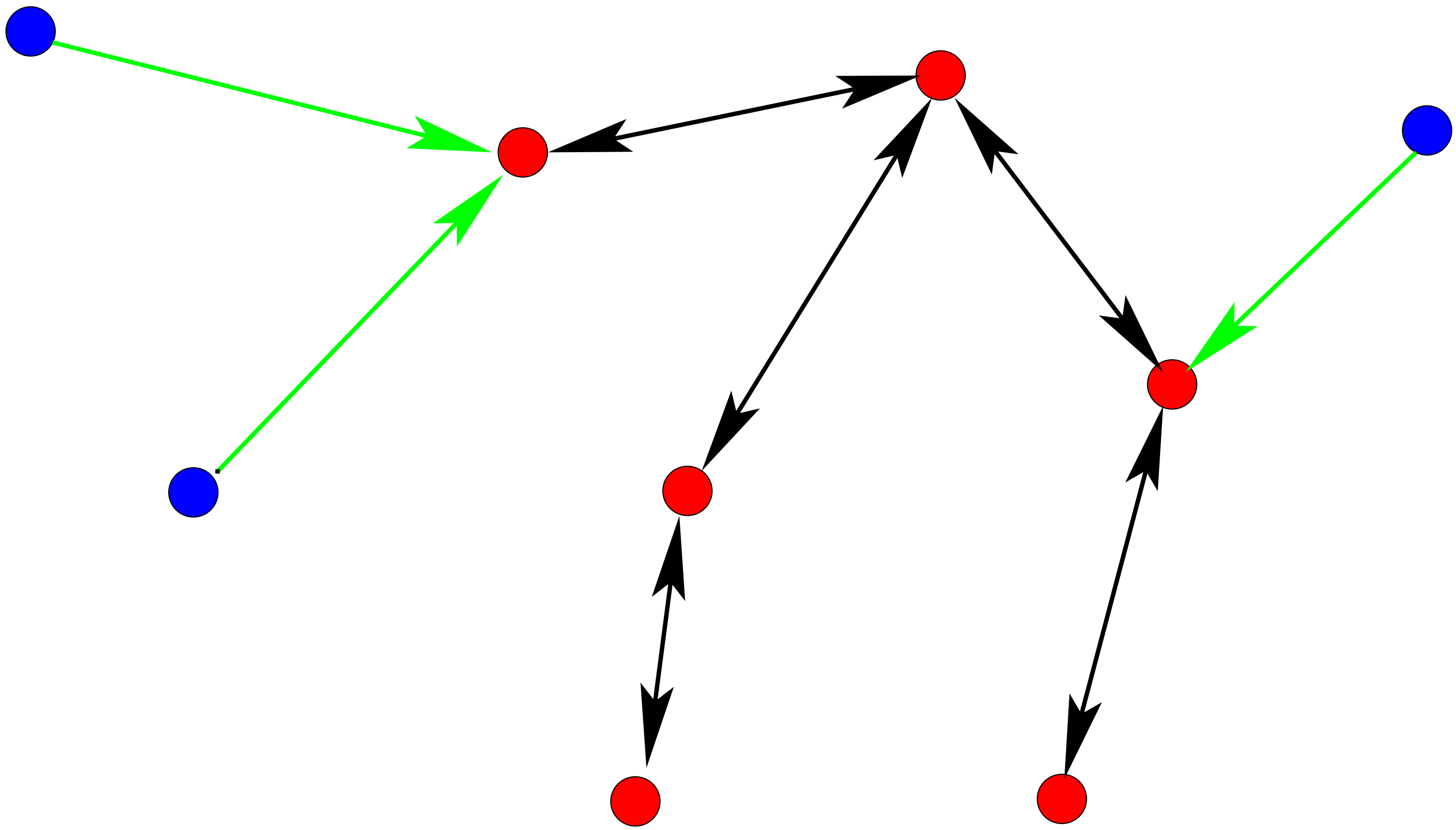
- This is to be done in a distributed and non-deterministic way

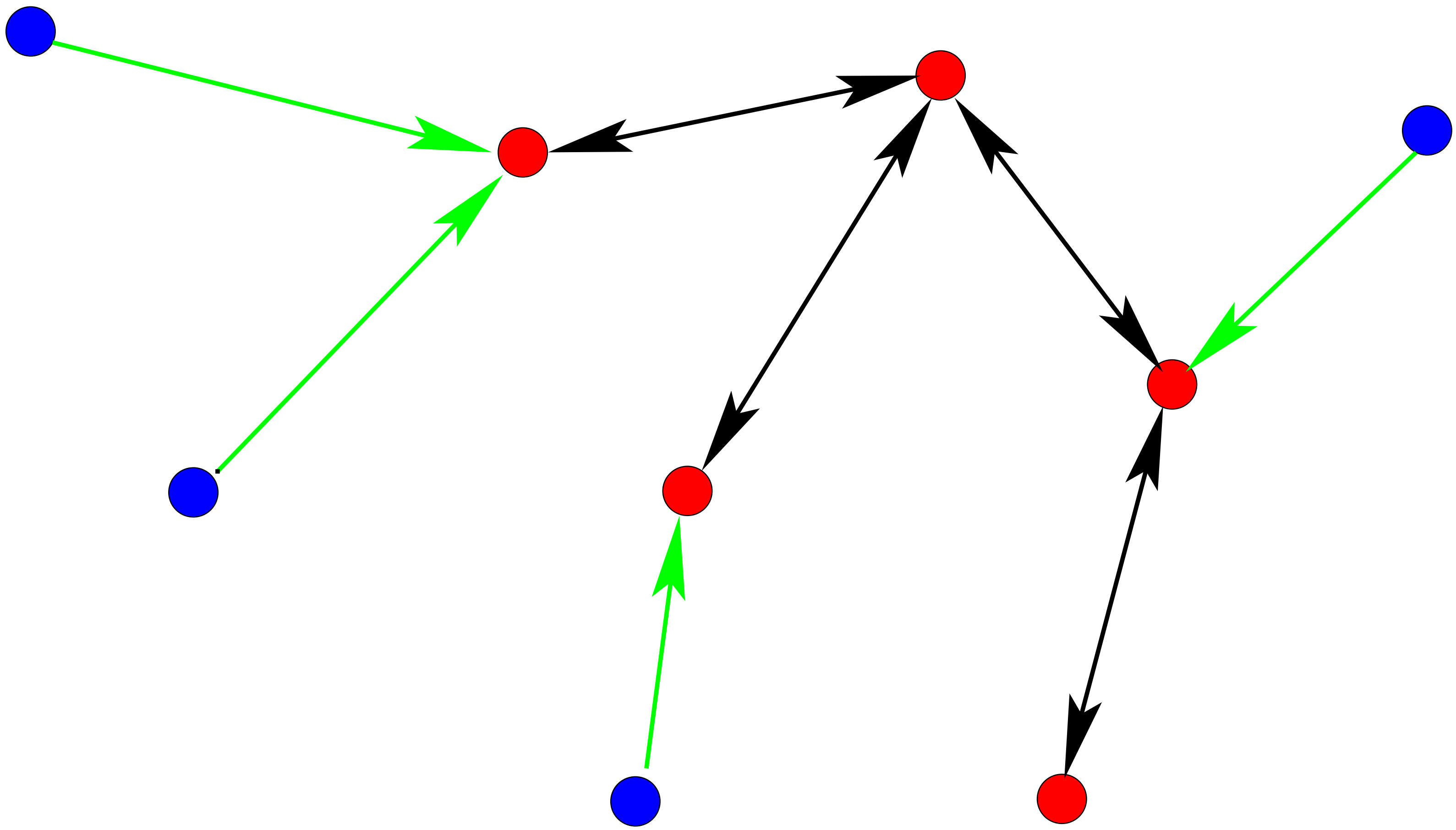
- Next are two distinct abstract animations of the protocol

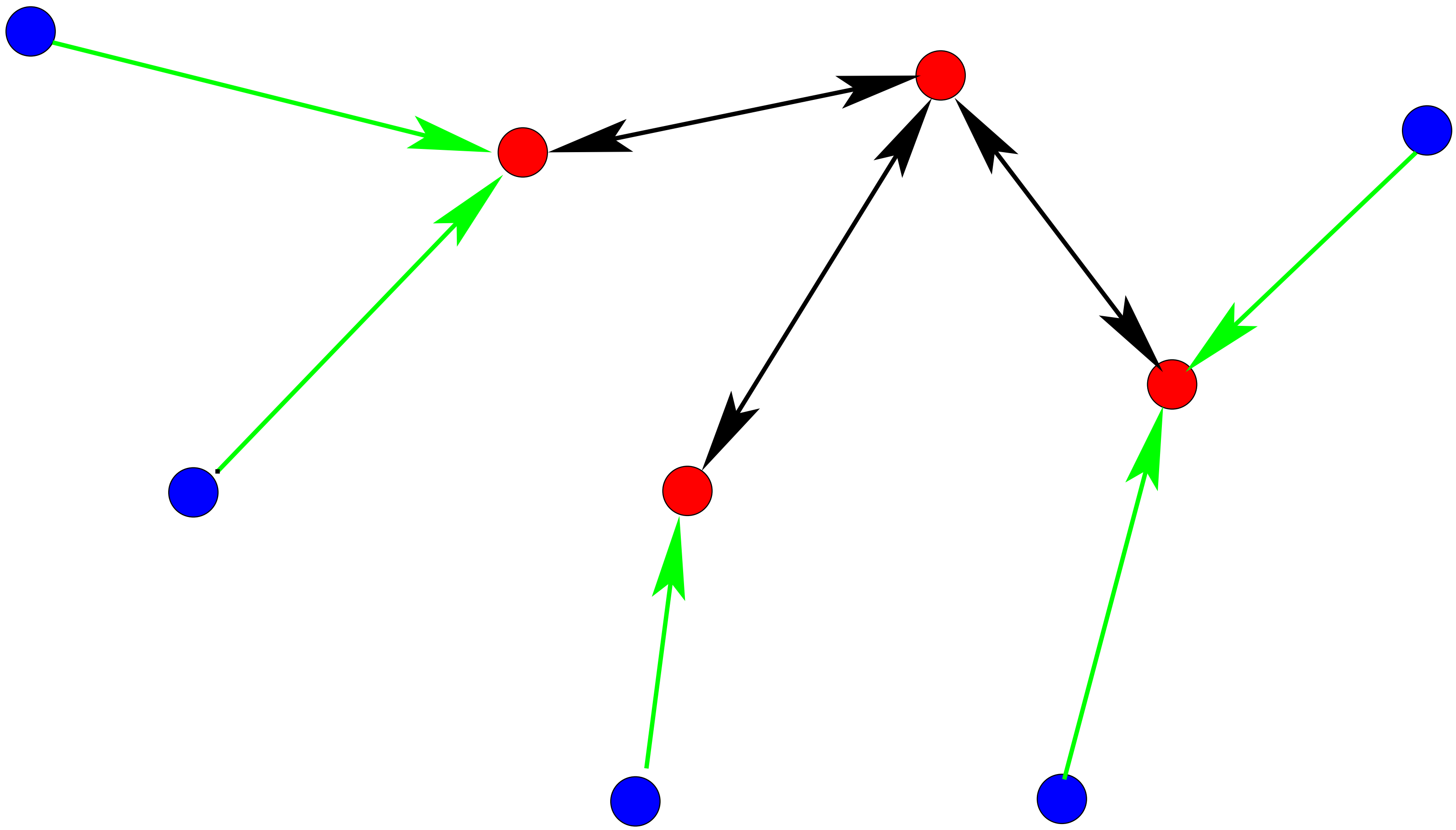


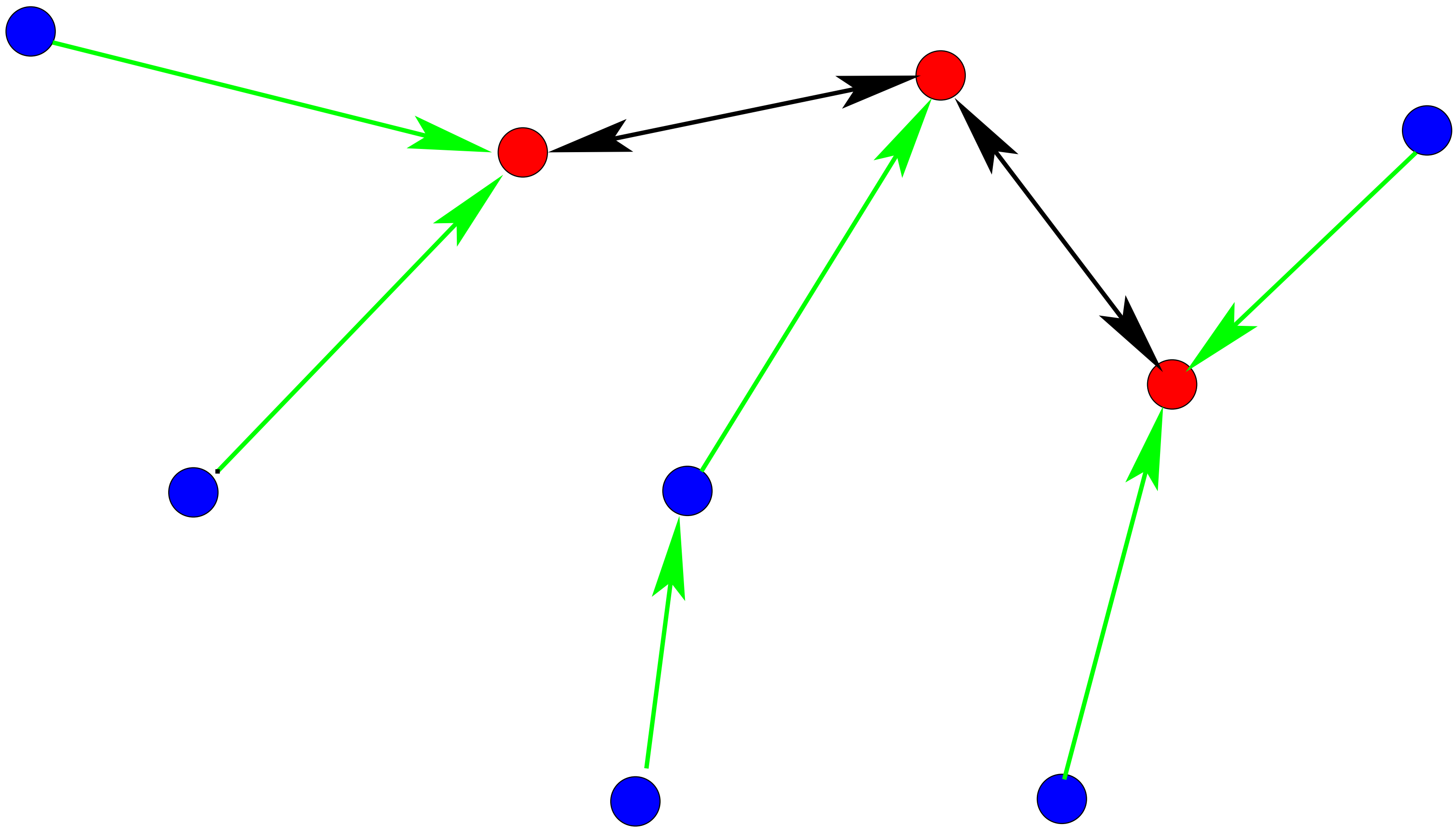


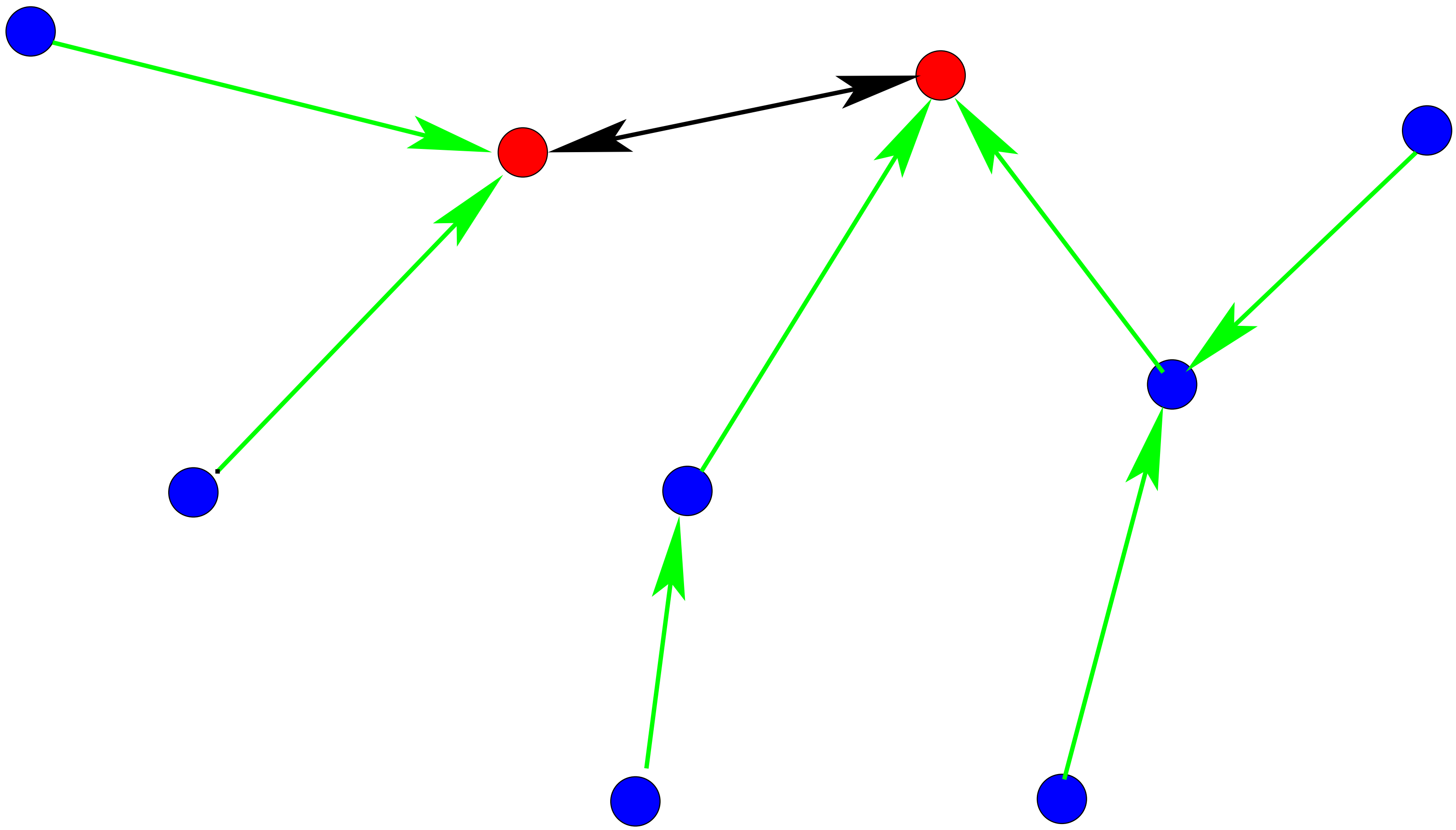


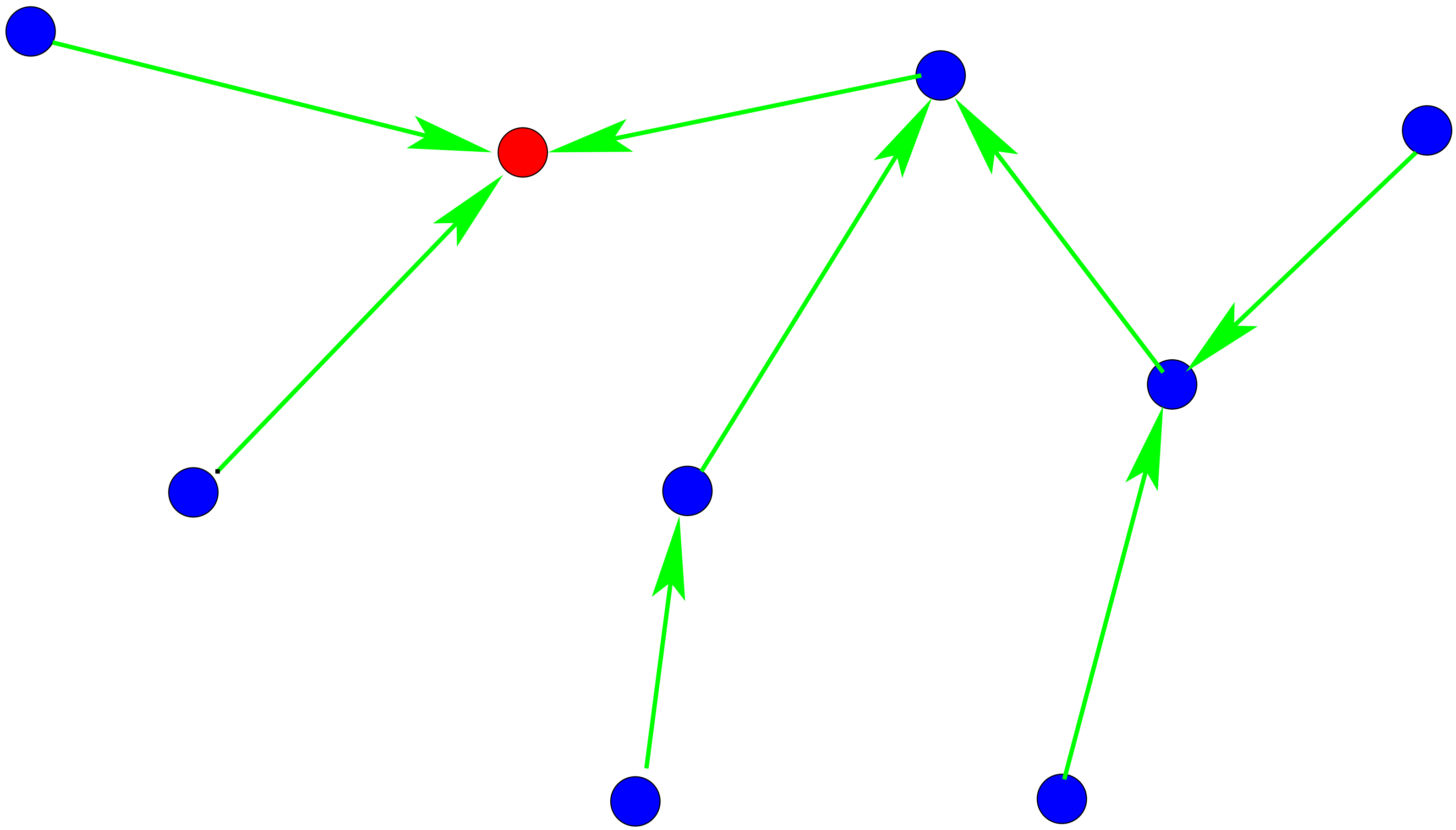


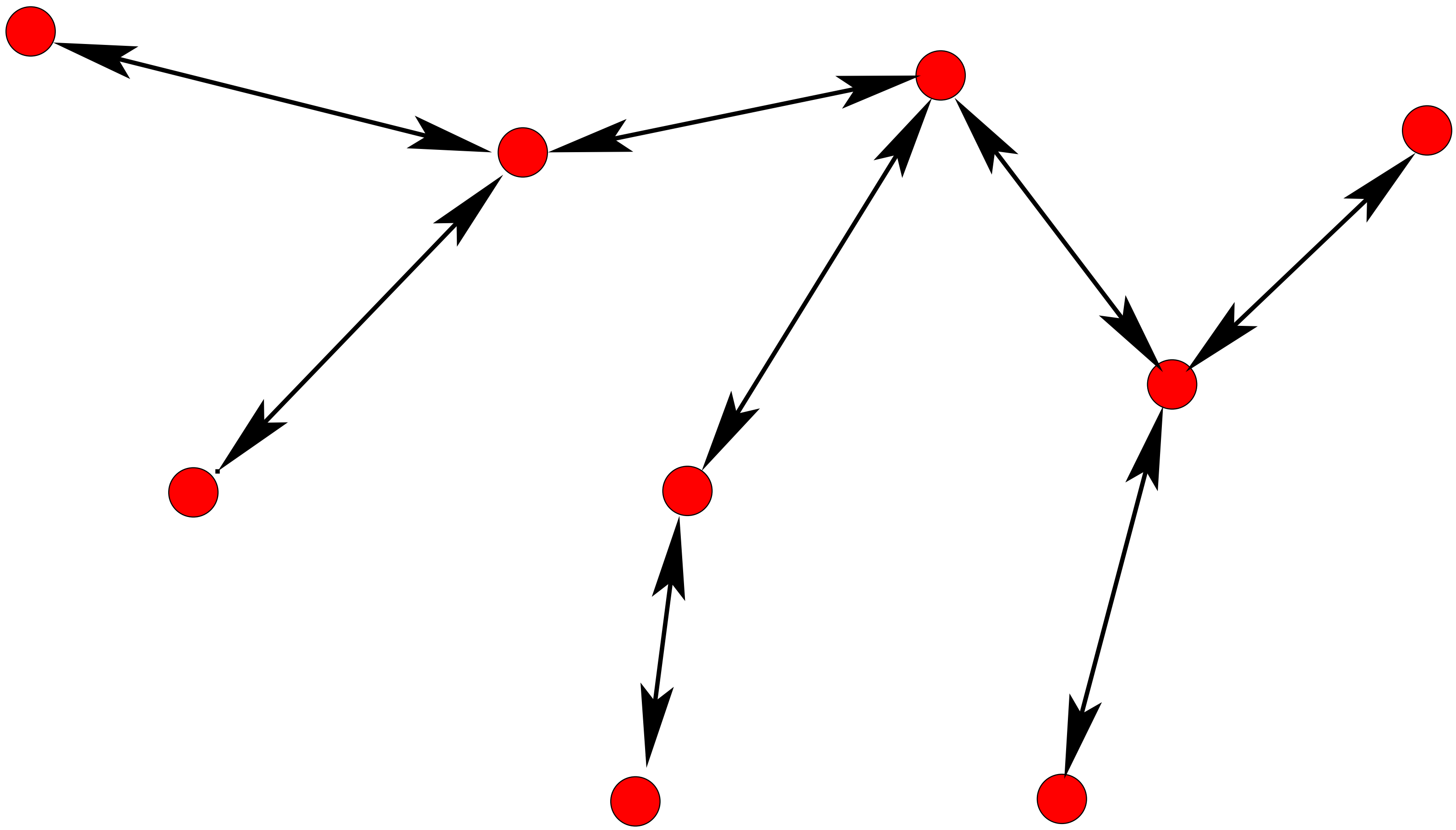


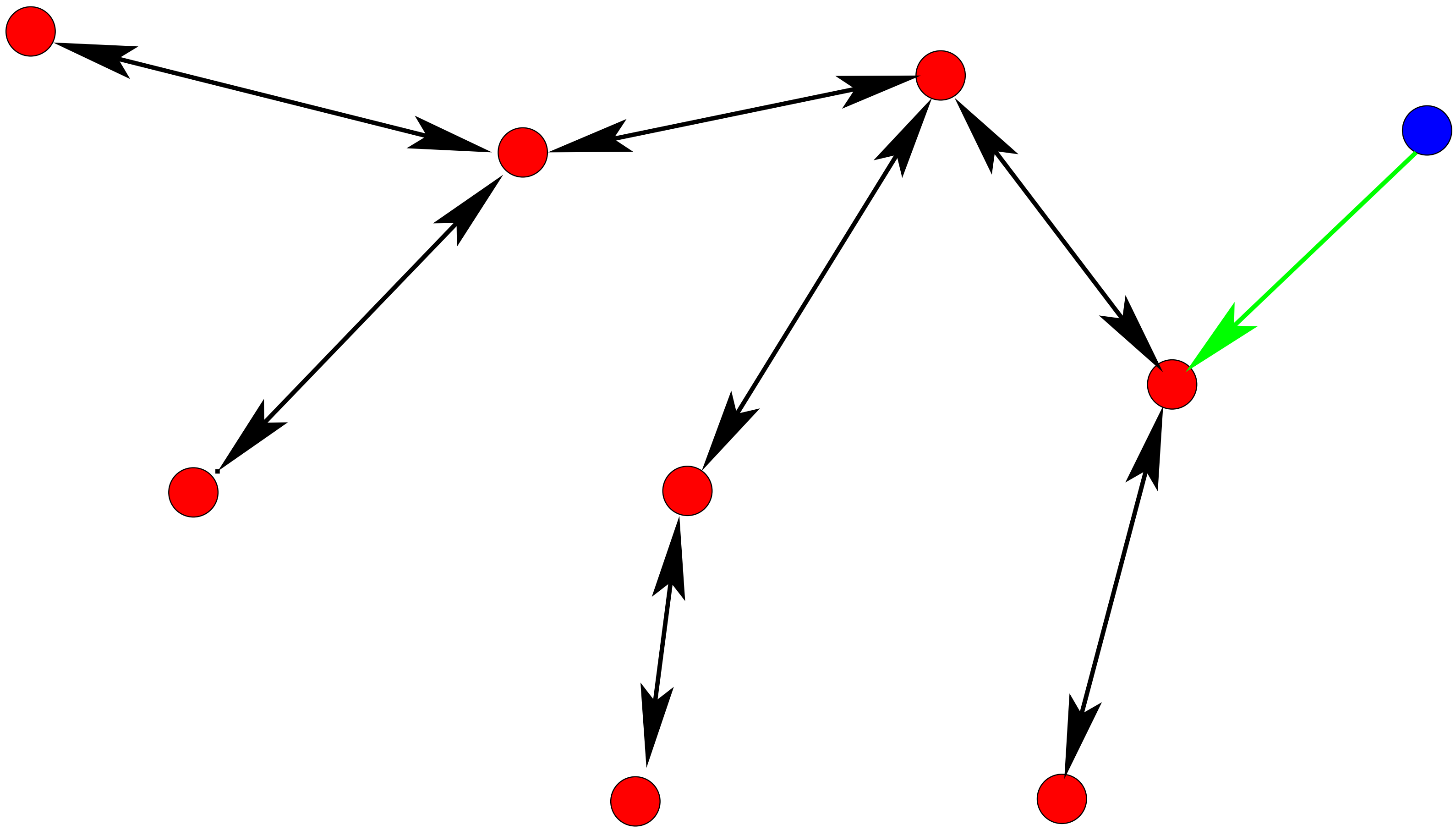


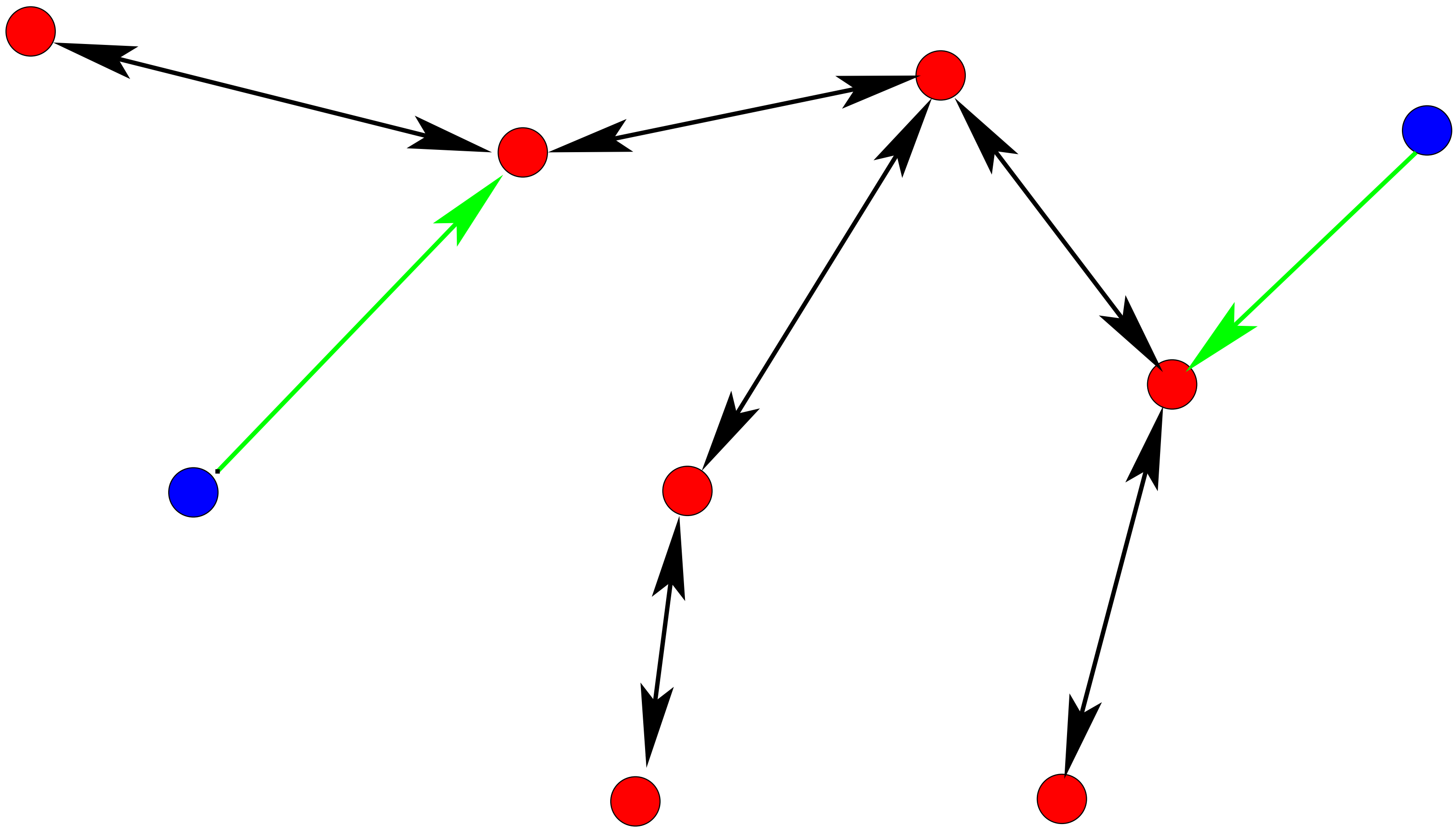


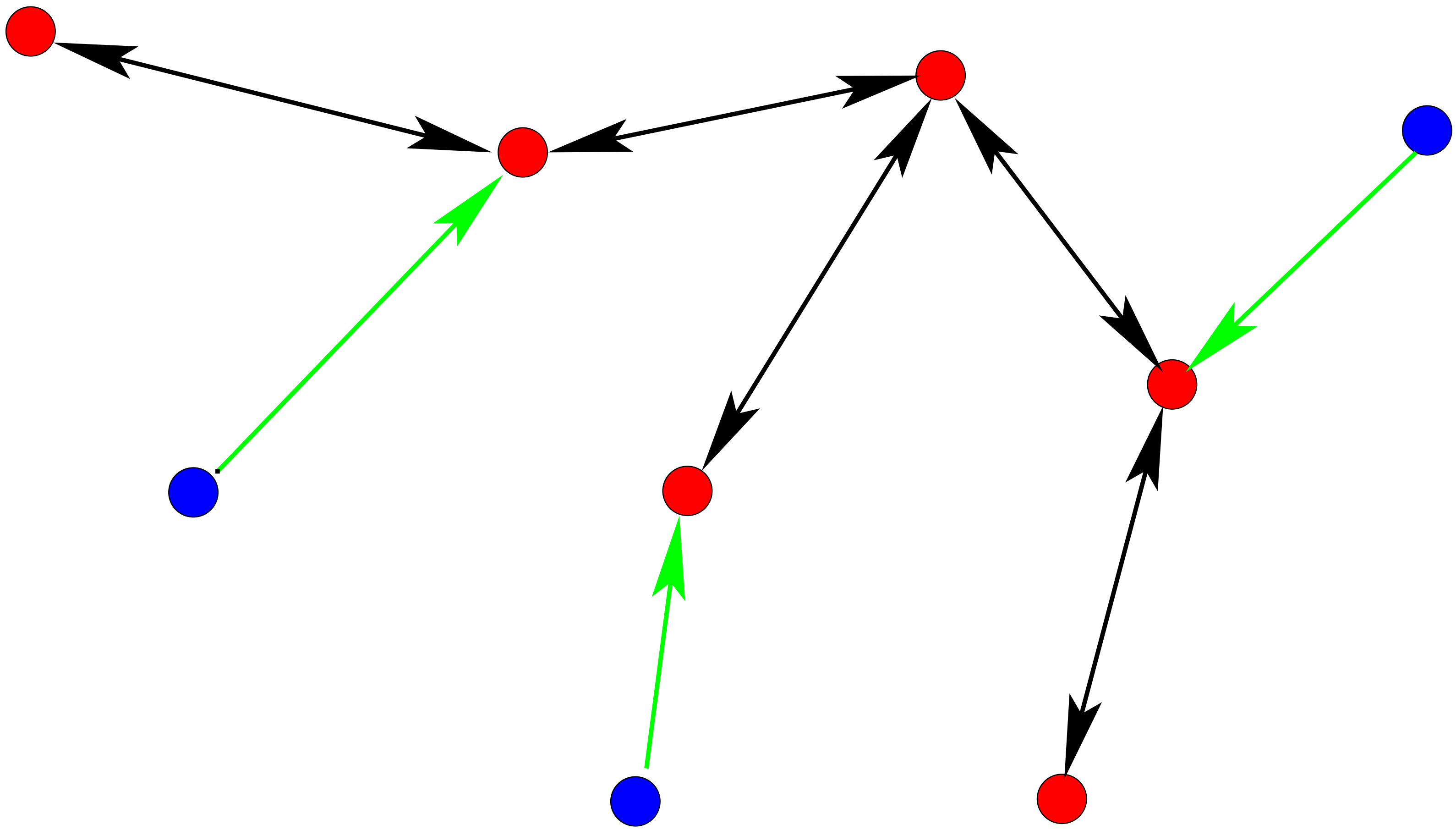


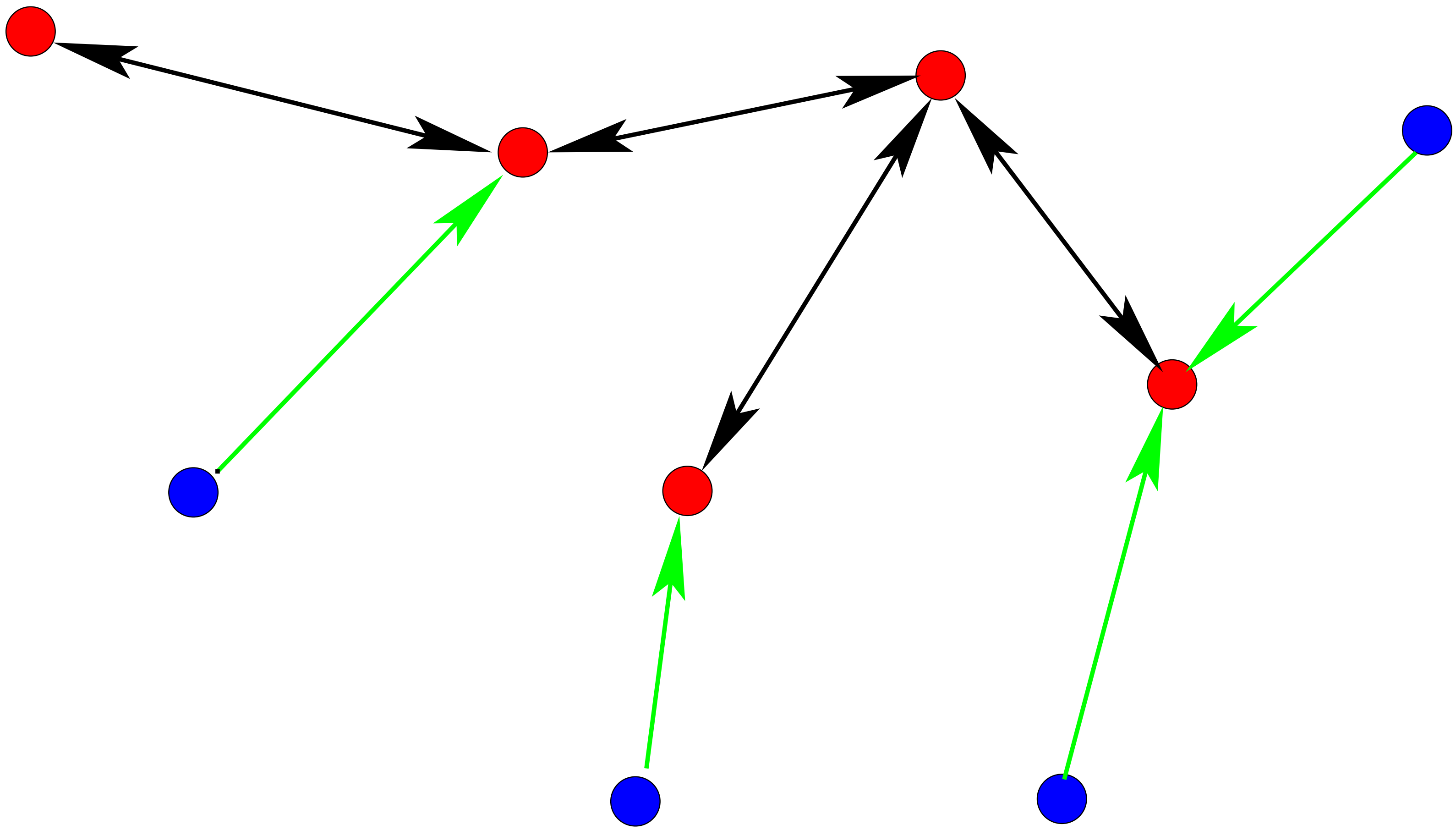


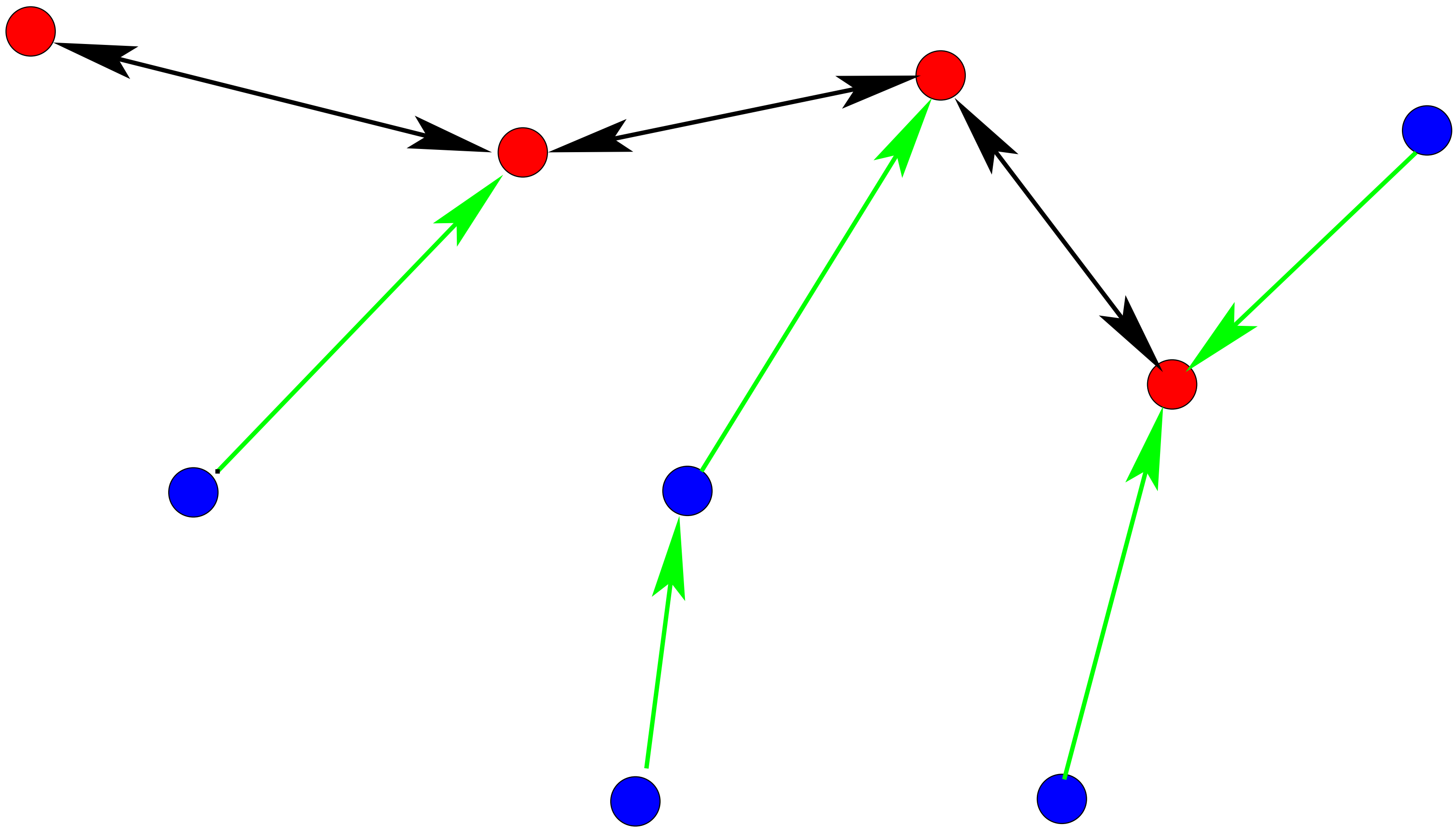


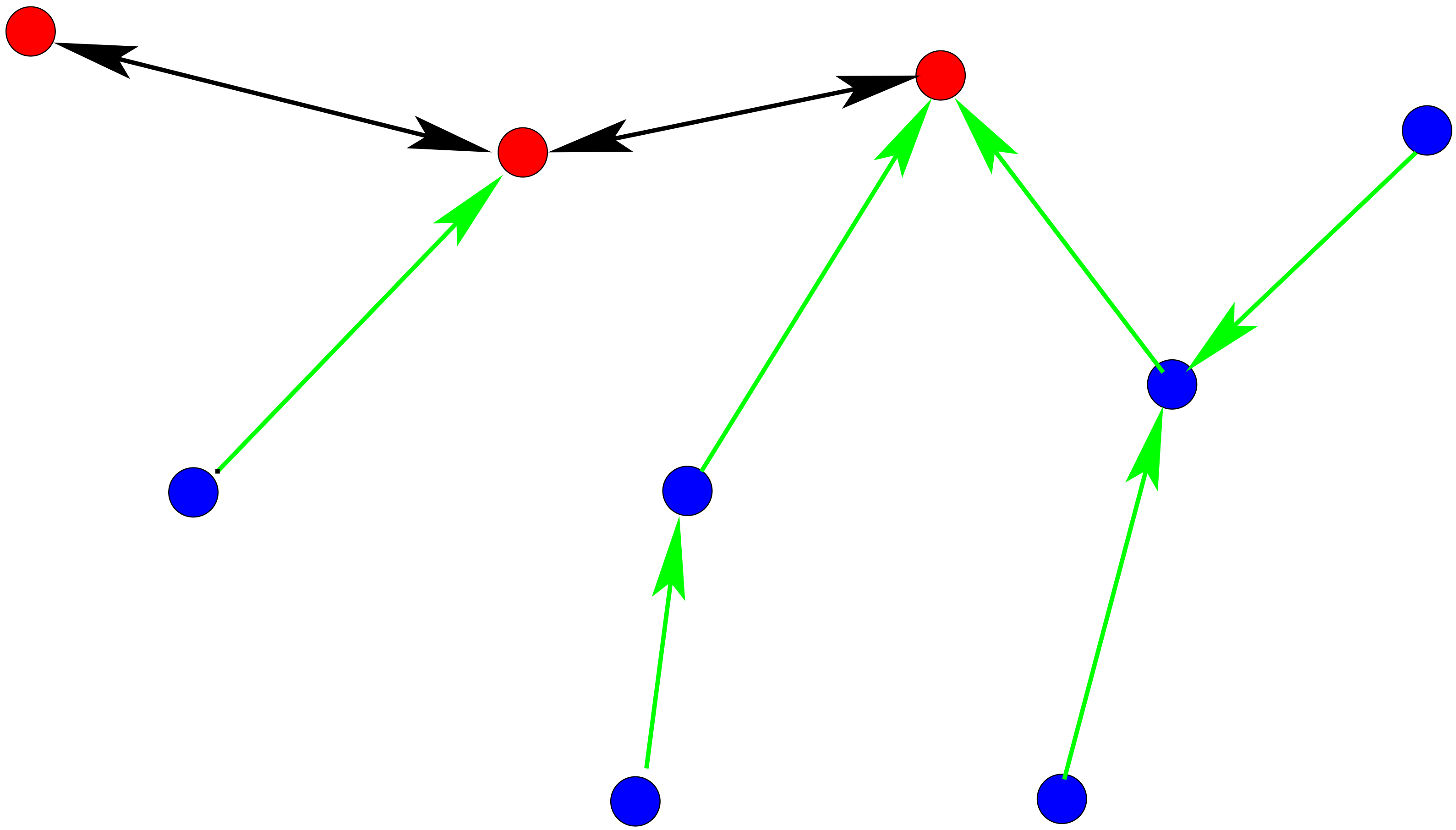


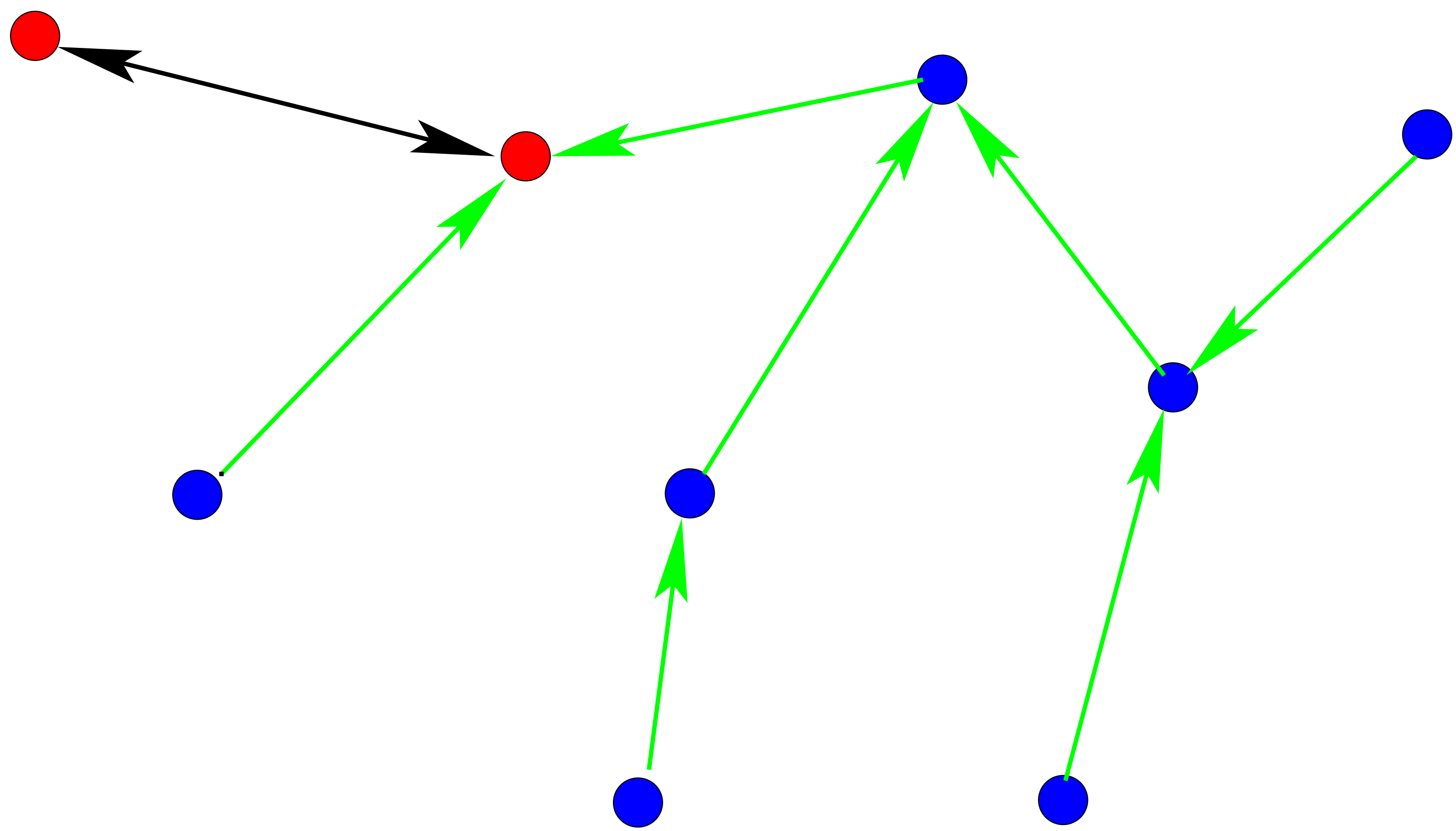


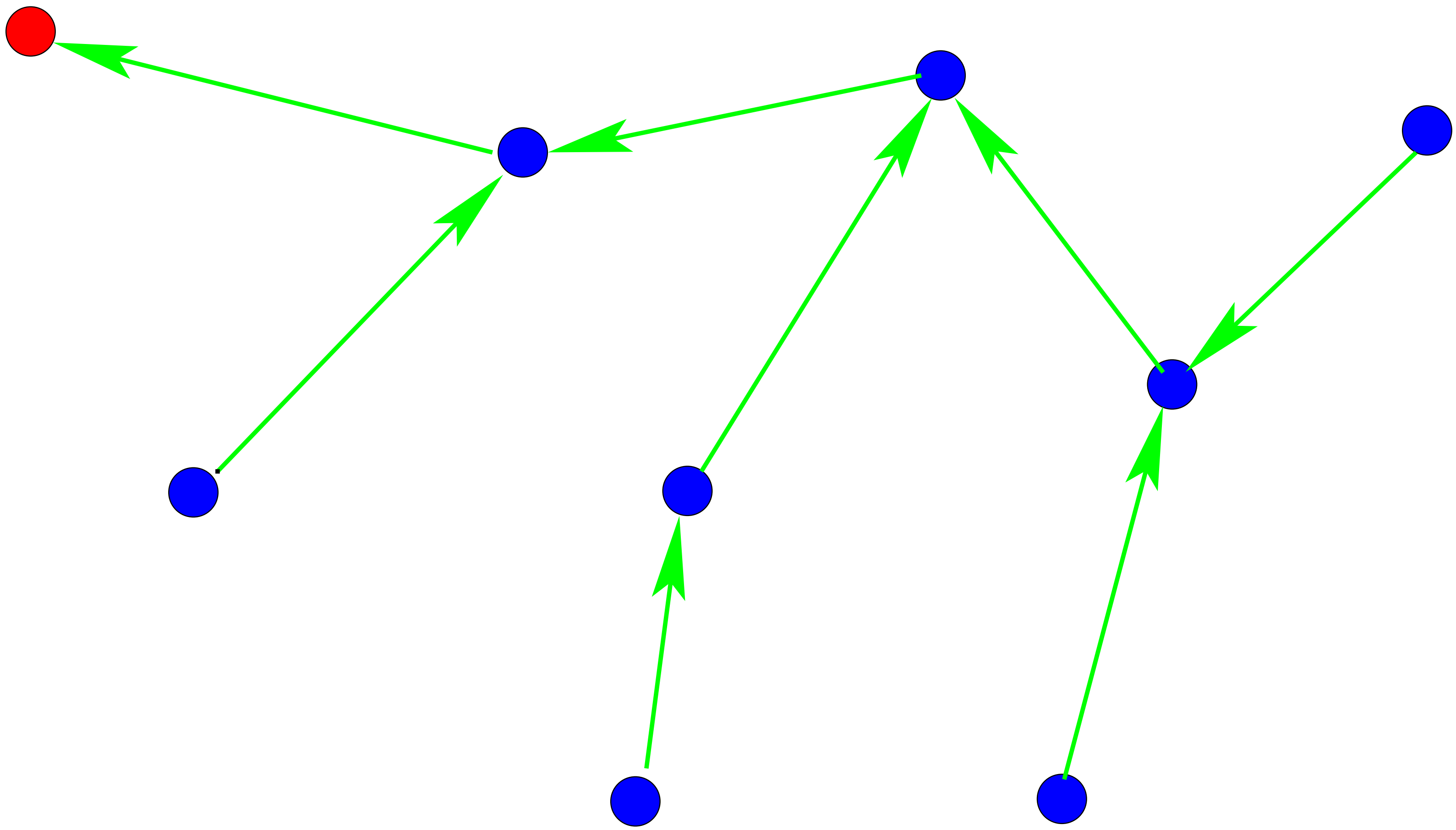












Summary of Development Process

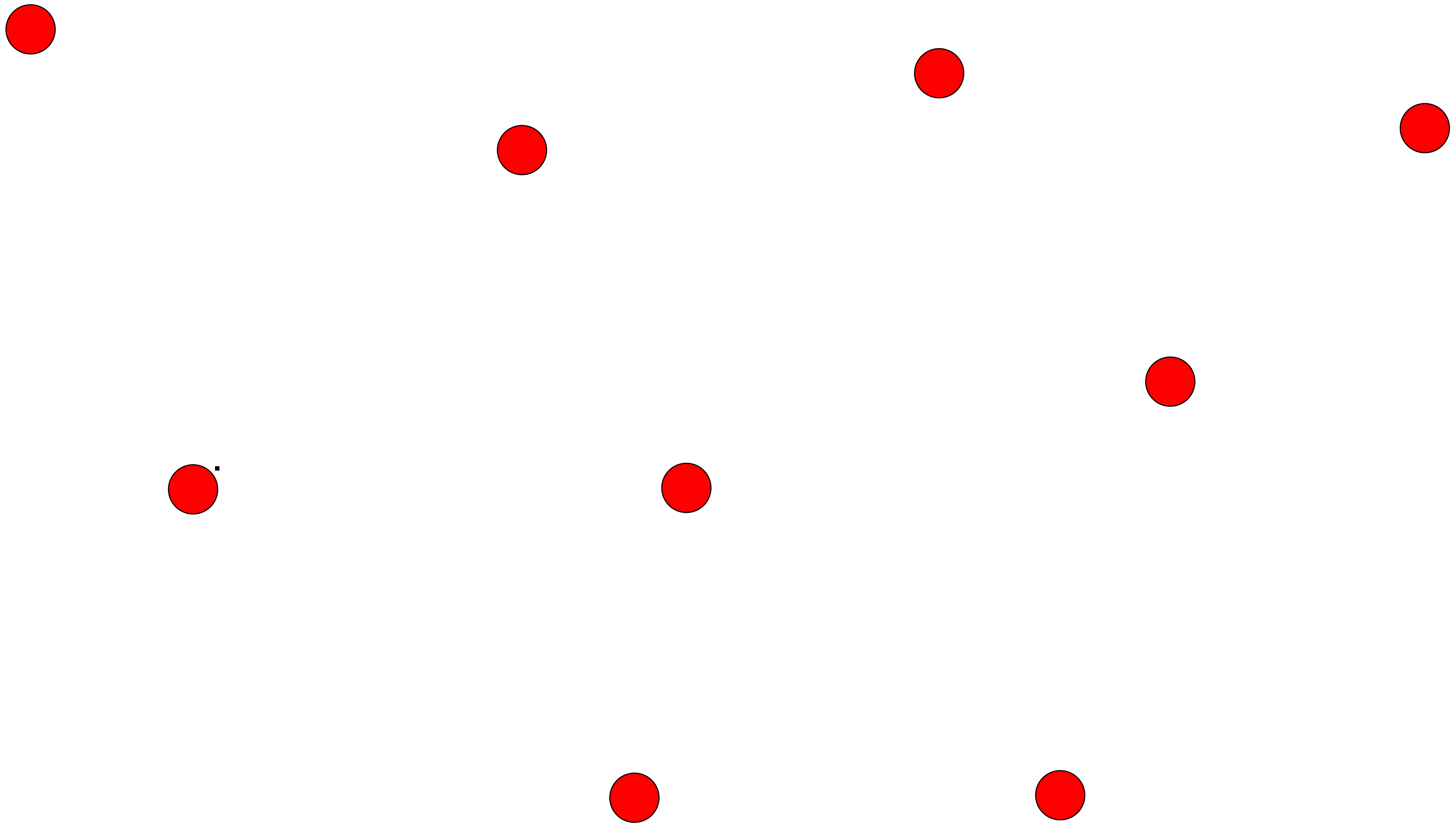
- Formal definition and properties of the network

- A one-shot abstract model of the protocol

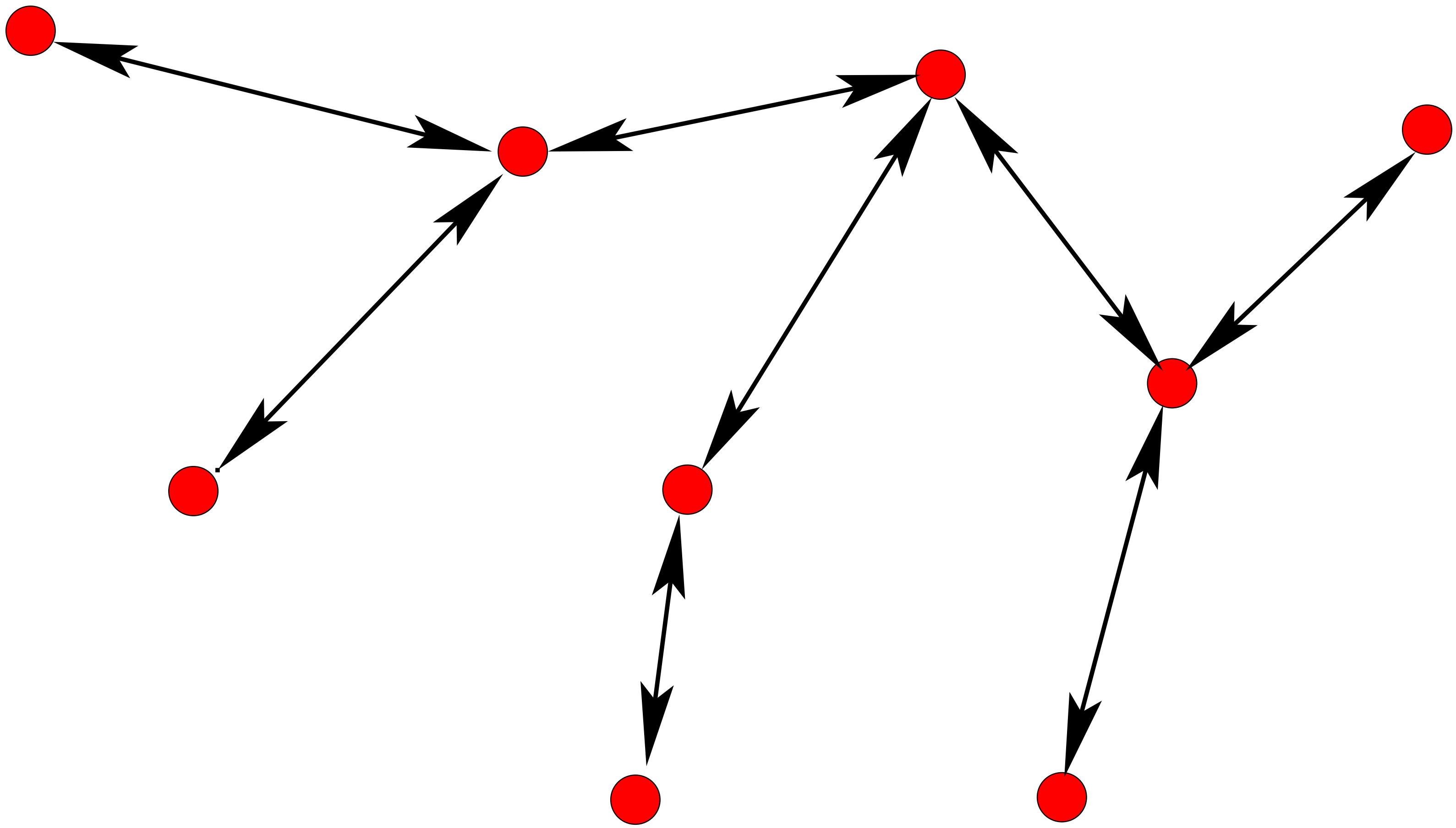
- Presenting a (still abstract) loop-like centralized solution

- Introducing message passing between the nodes (delays)

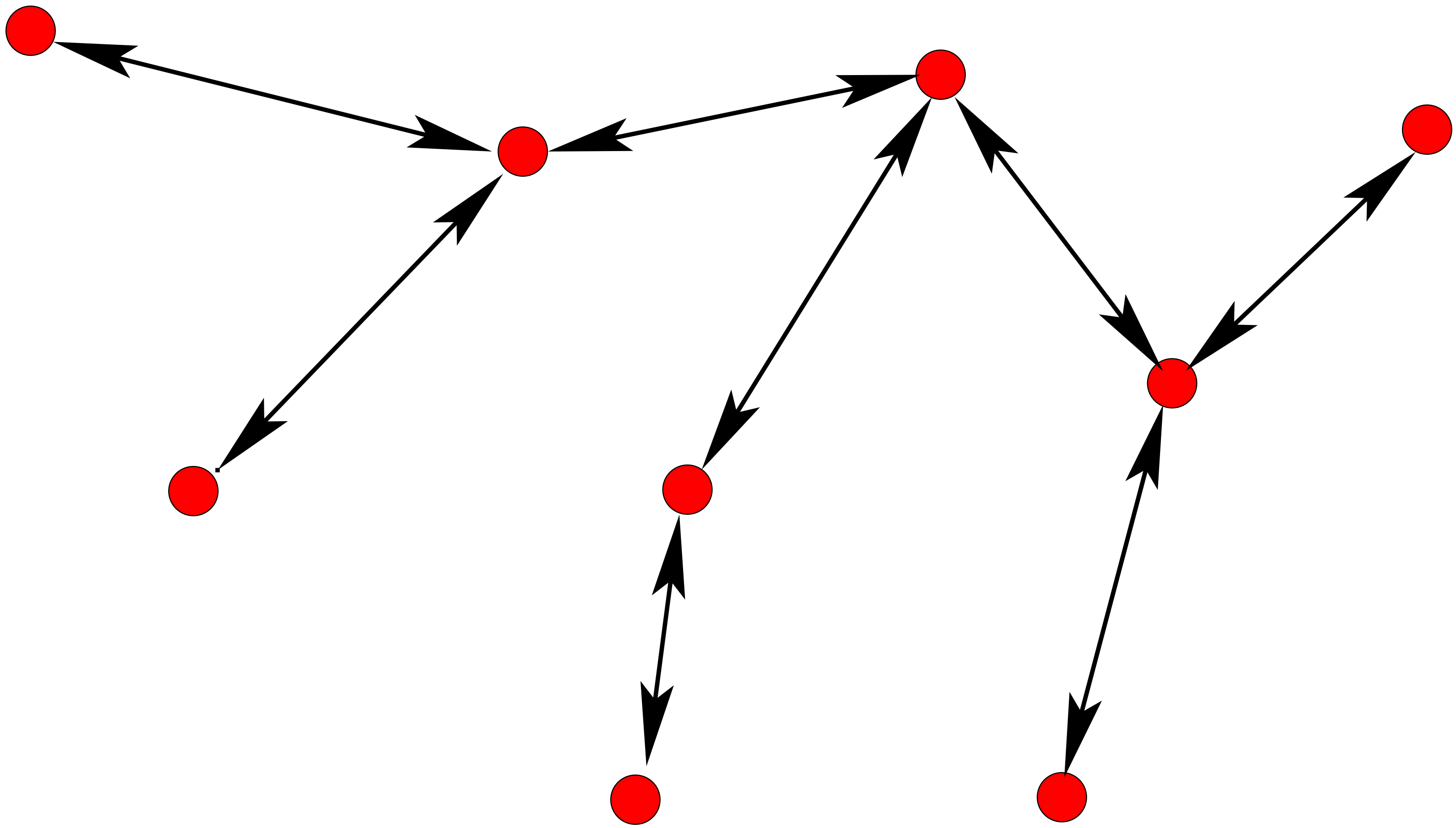
- Modifying the data structure in order to distribute the protocol



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 $\qquad gr \subseteq ND \times ND$

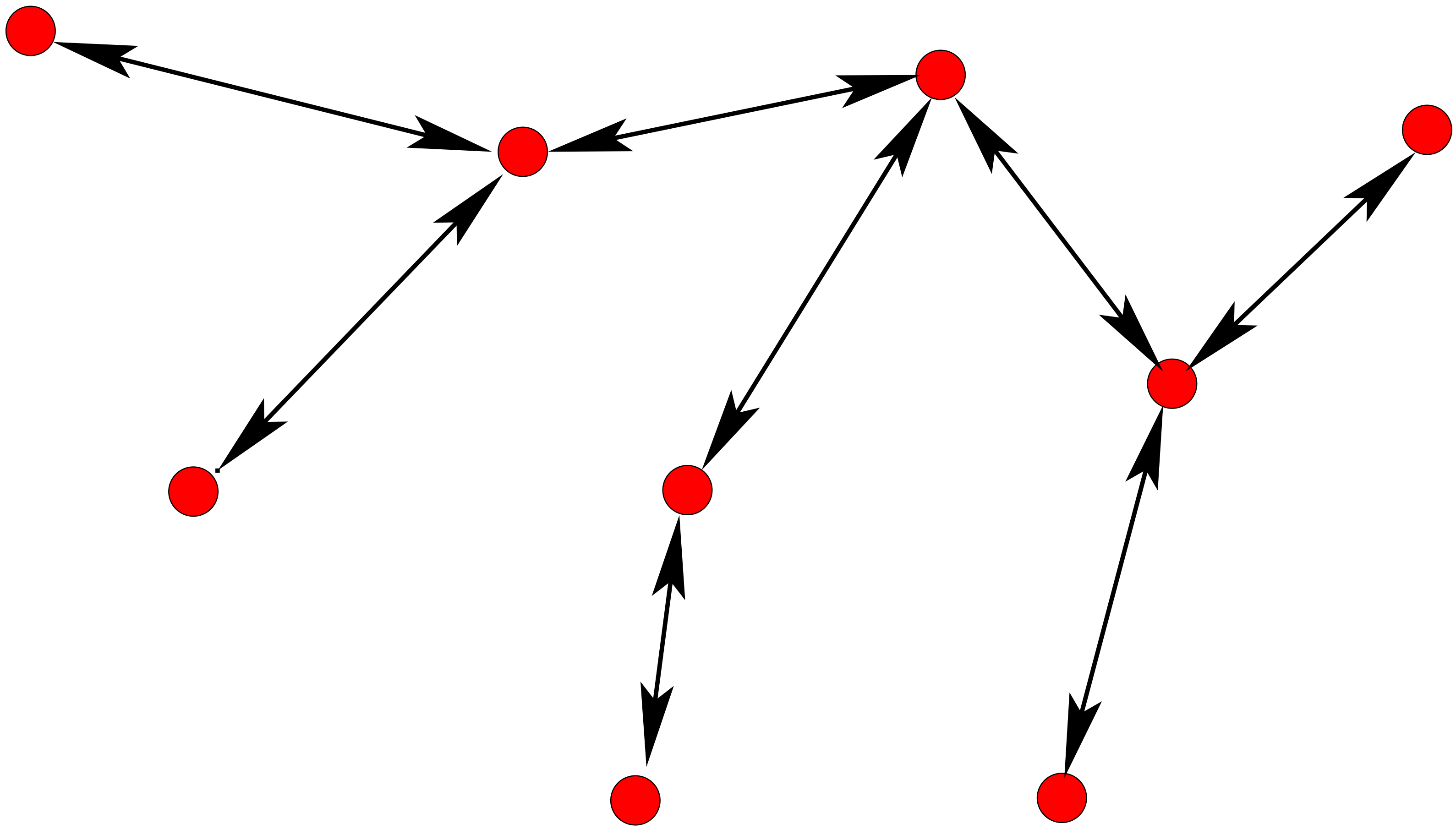
gr is defined on ND $\qquad \mathsf{dom}\,(gr) = ND$

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

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

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

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

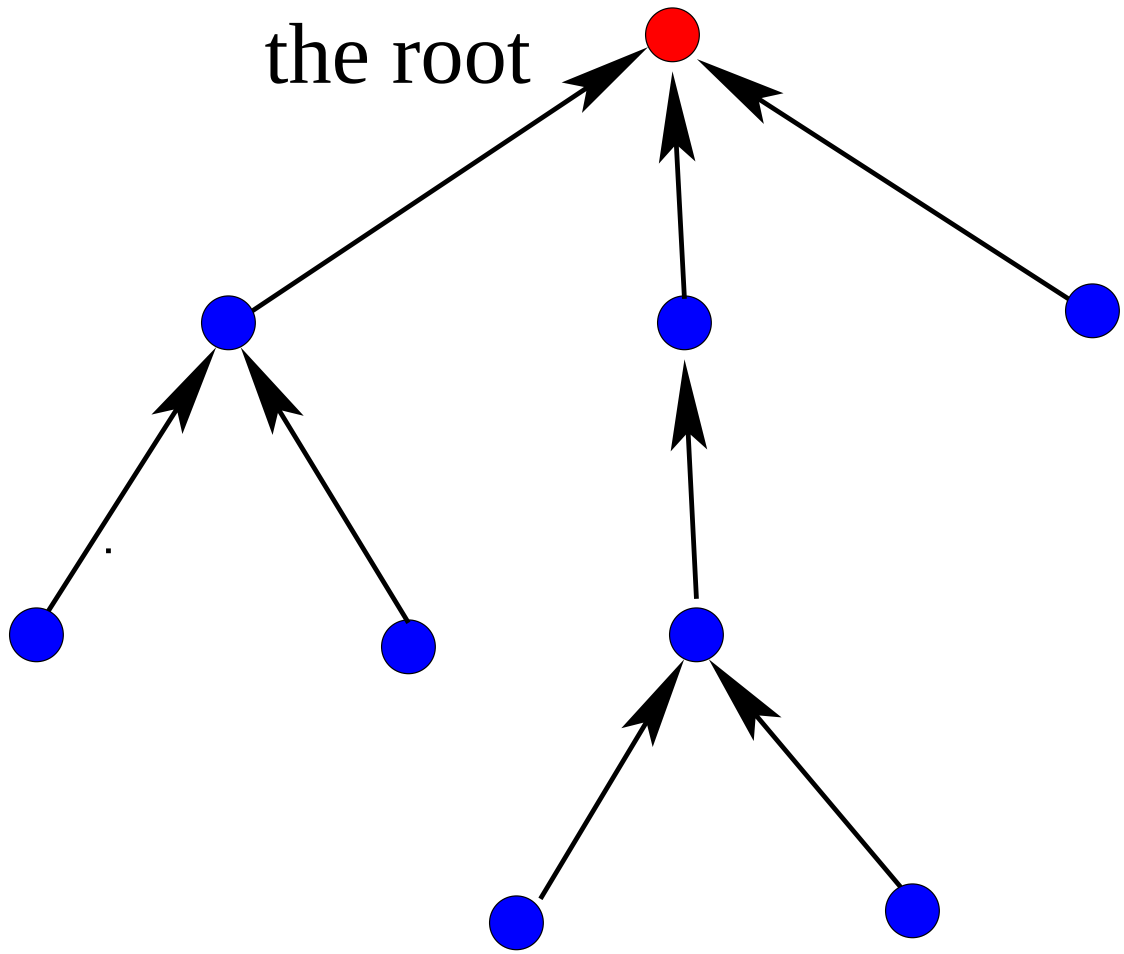
- A tree is a special graph

- A tree has a root

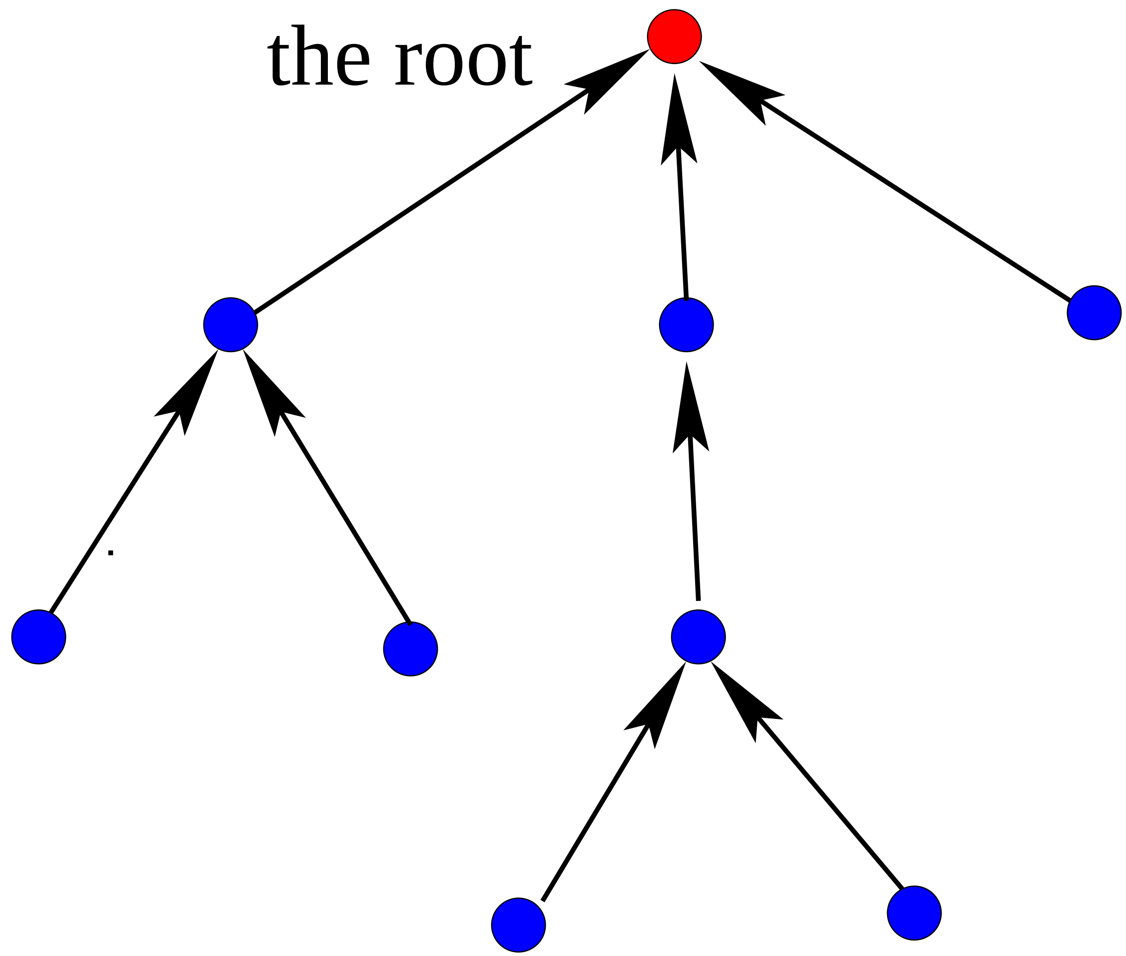
- A tree has a, so-called, father function

- A tree is acyclic

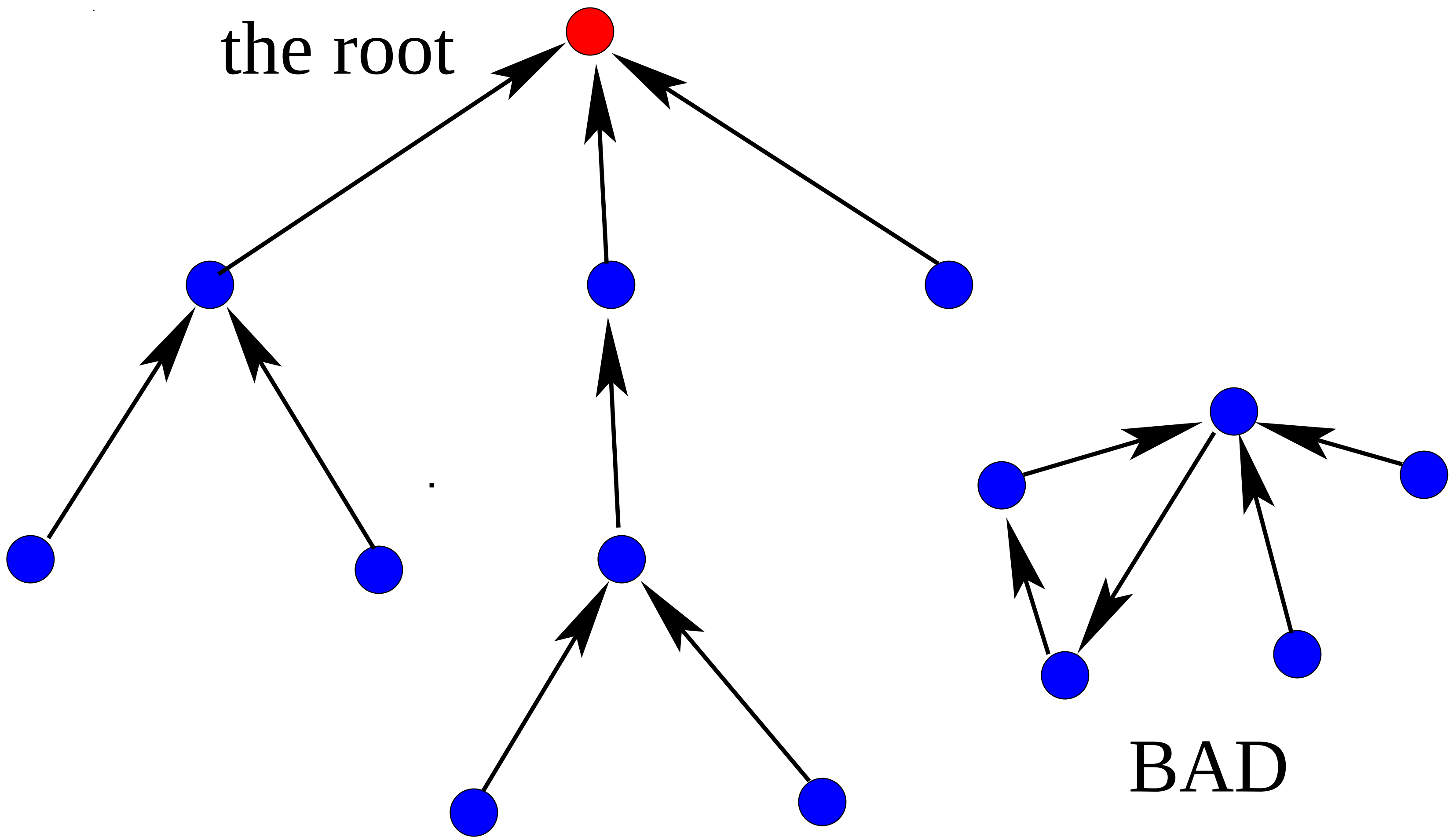
- A tree is connected from the root



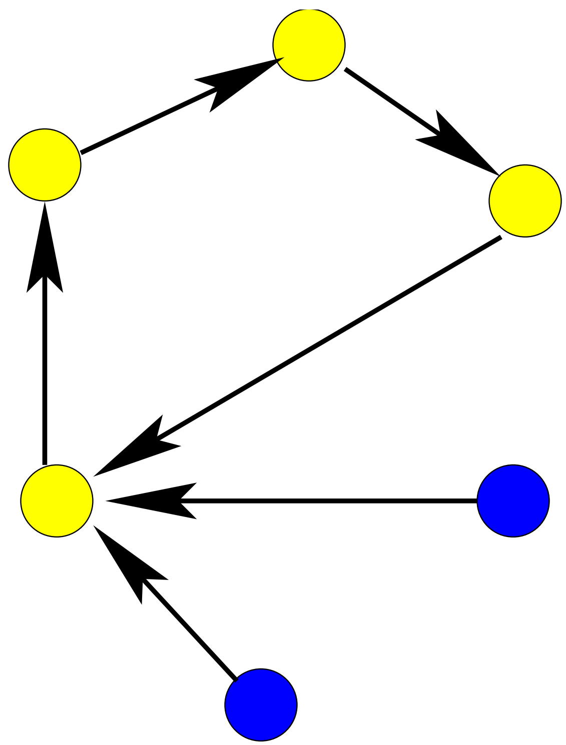
A tree t built on a set of nodes



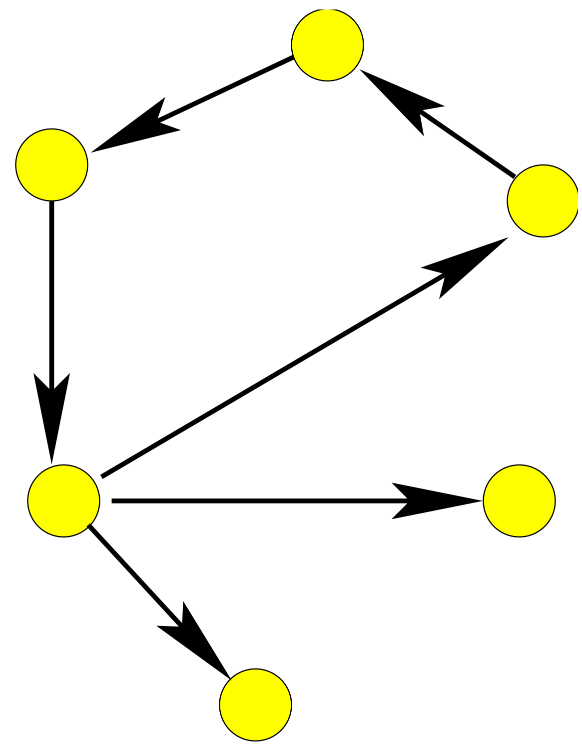
t is a function defined on ND except at the root



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

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

- 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 $\quad t \in ND - \{r\} \to ND$

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

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

This gives an Induction Rule

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

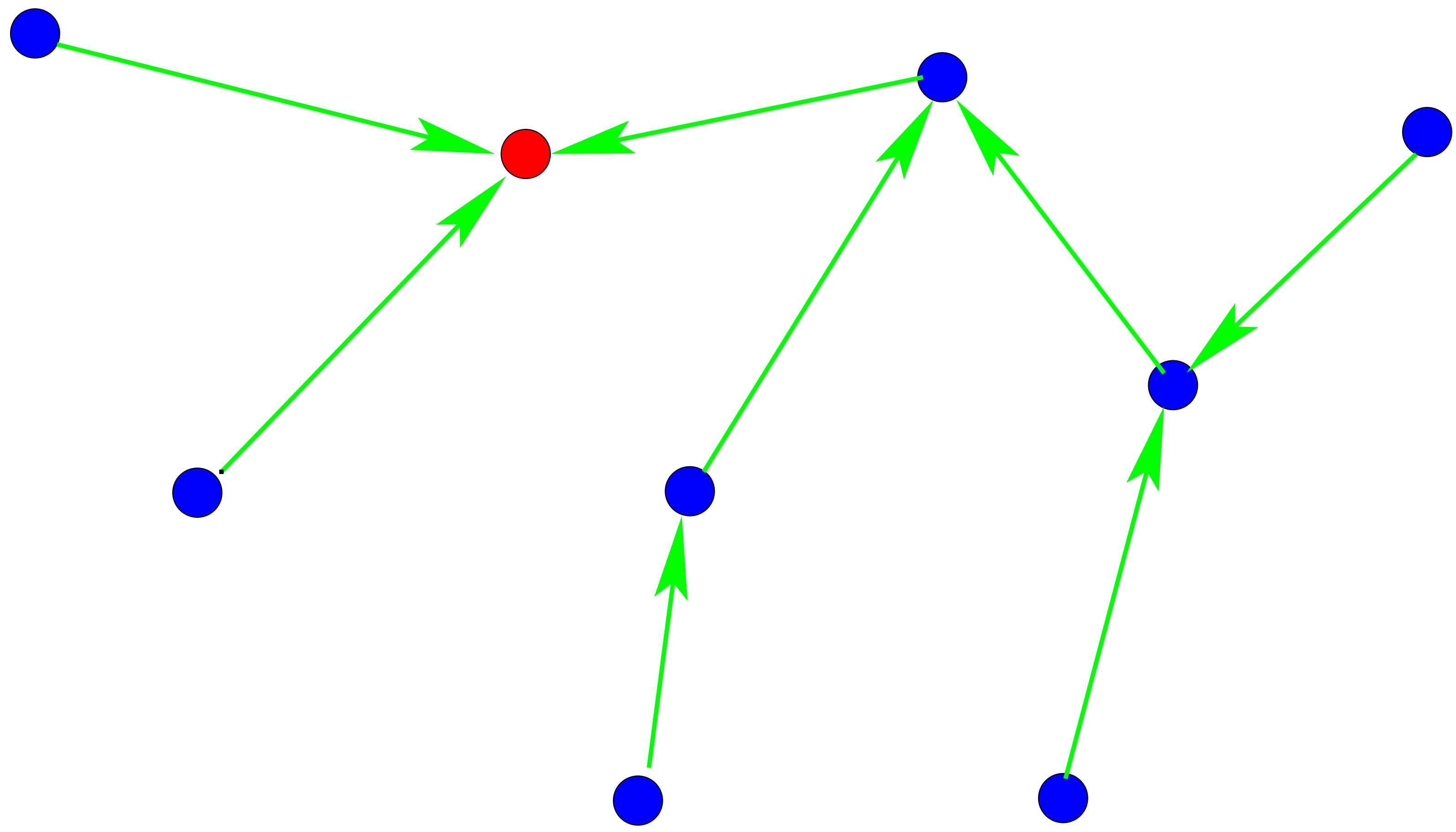
The predicate tree (r, t)

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:

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

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

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

$$gr \subseteq ND \times ND$$

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

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

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

$$fn \in ND \rightarrow (ND \nrightarrow ND)$$

$$\forall (r, t) \cdot \left(\begin{array}{l} \quad r \in ND \;\; \wedge \\ \quad t \in ND \nrightarrow ND \\ \Rightarrow \\ \quad t = fn(r) \;\Leftrightarrow\; \text{spanning}\,(r, t, gr) \end{array} \right)$$

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

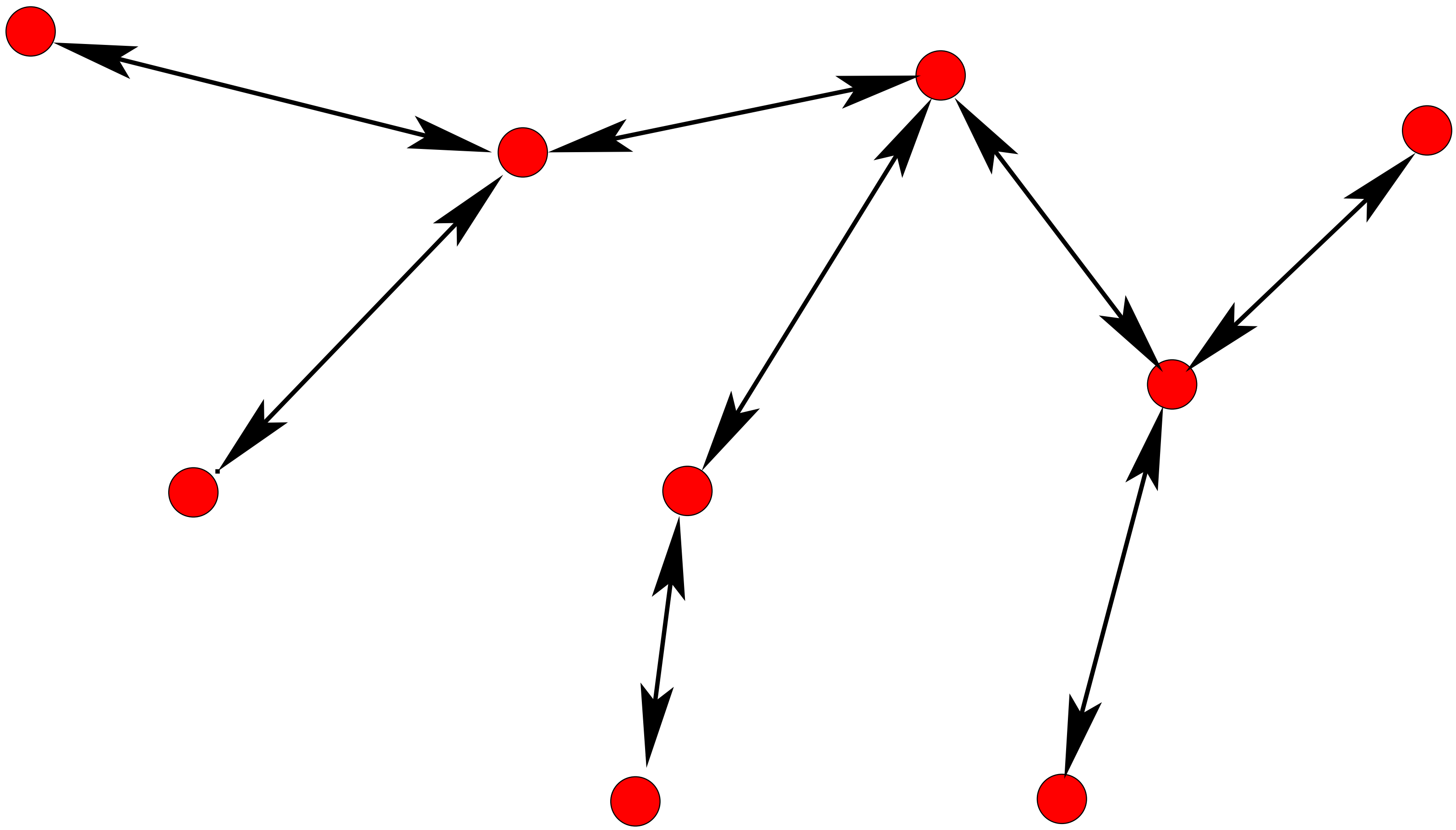
First Refinement (1)

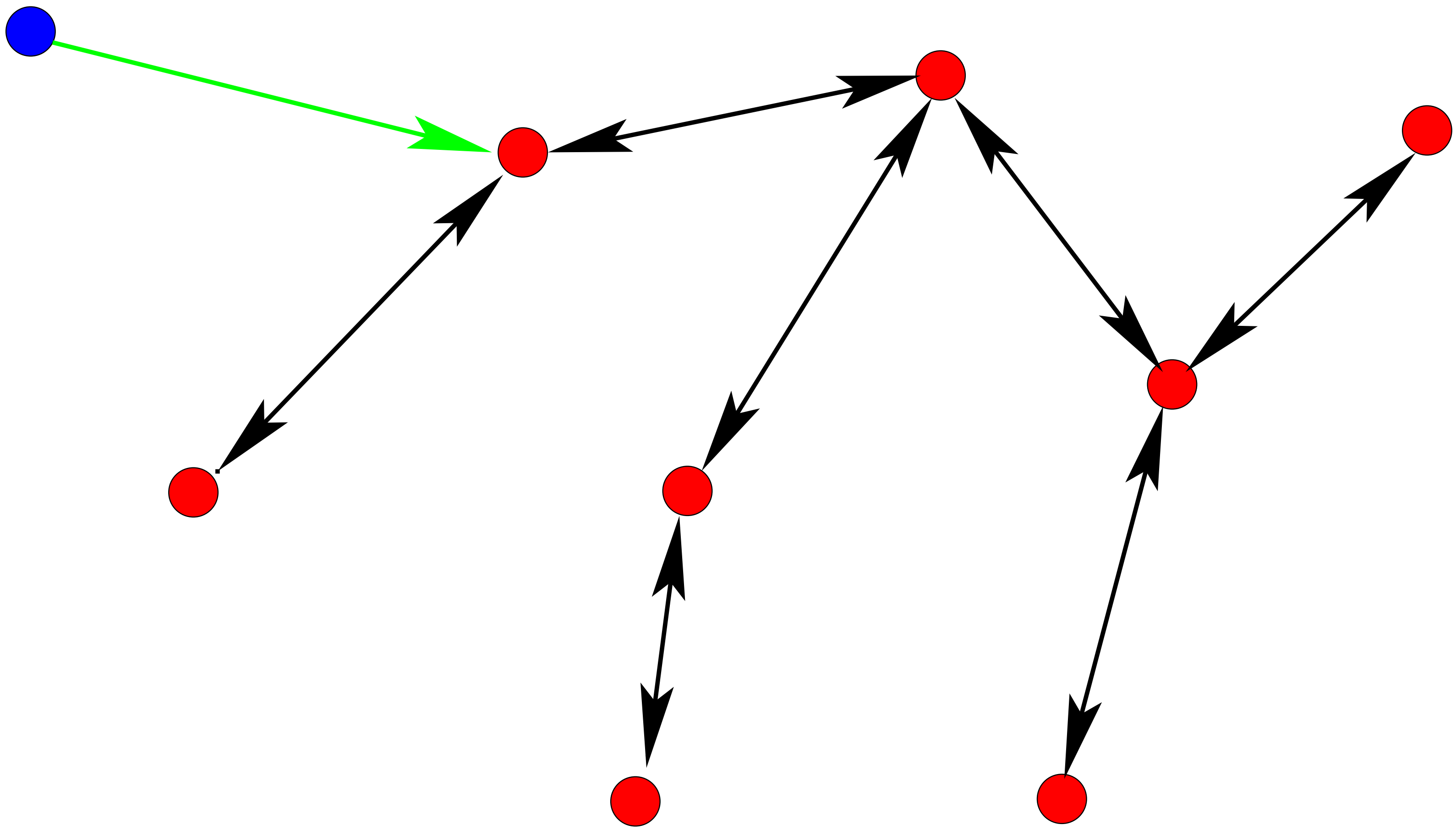
- Introducing a new variable, tr, corresponding to the "tree" in construction

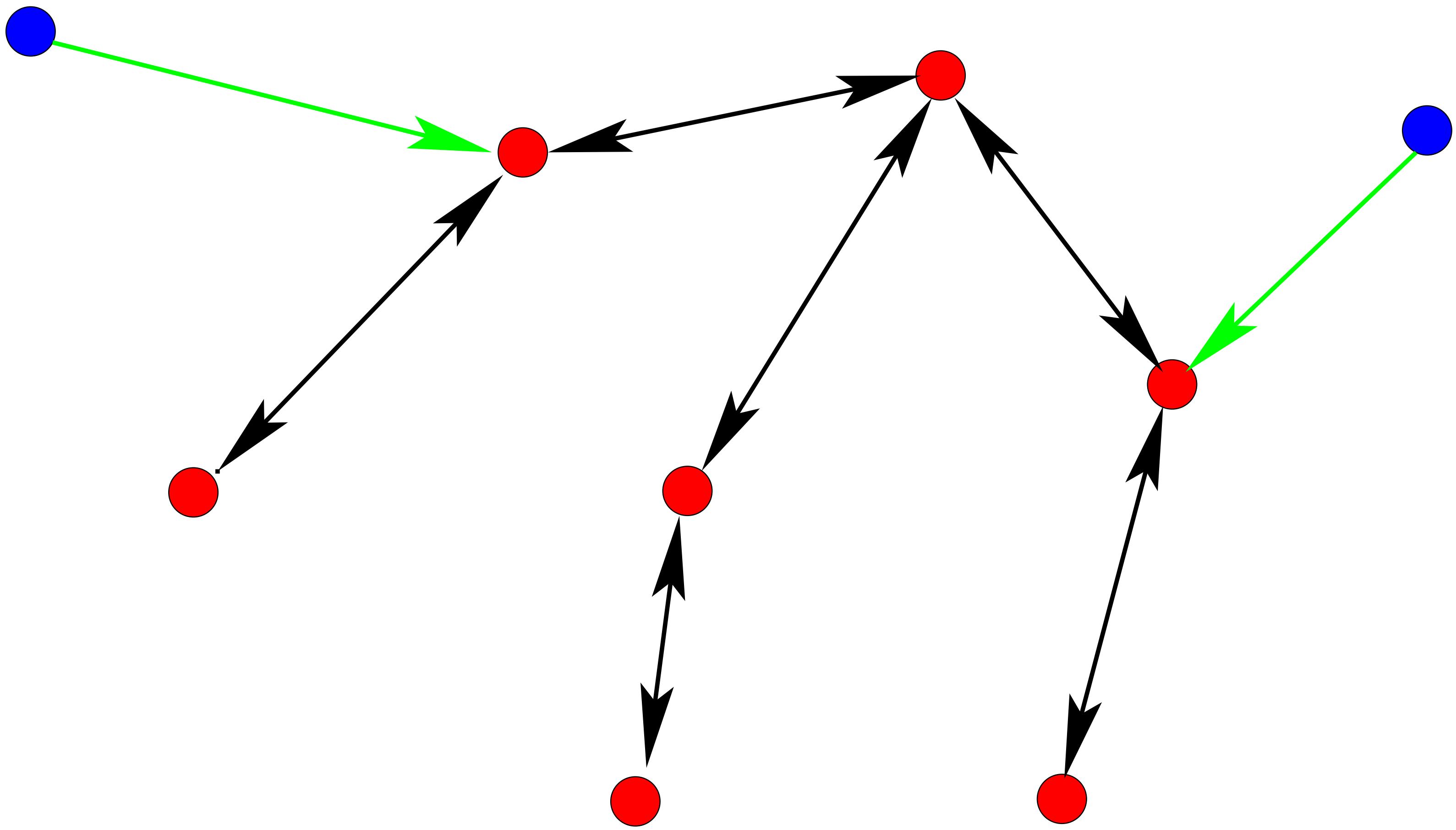
- Introducing a new event: the progression event

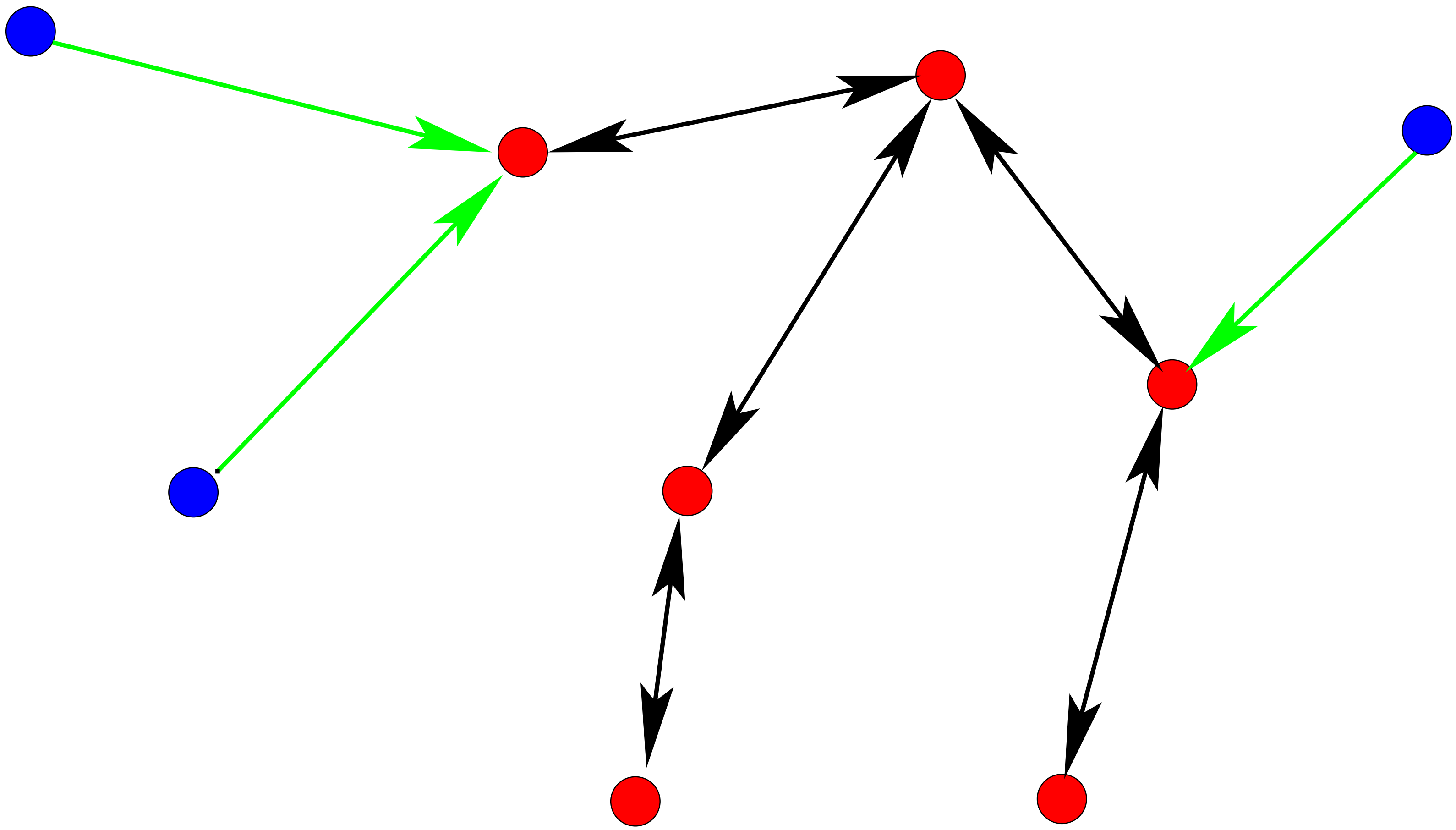
- Defining the invariant

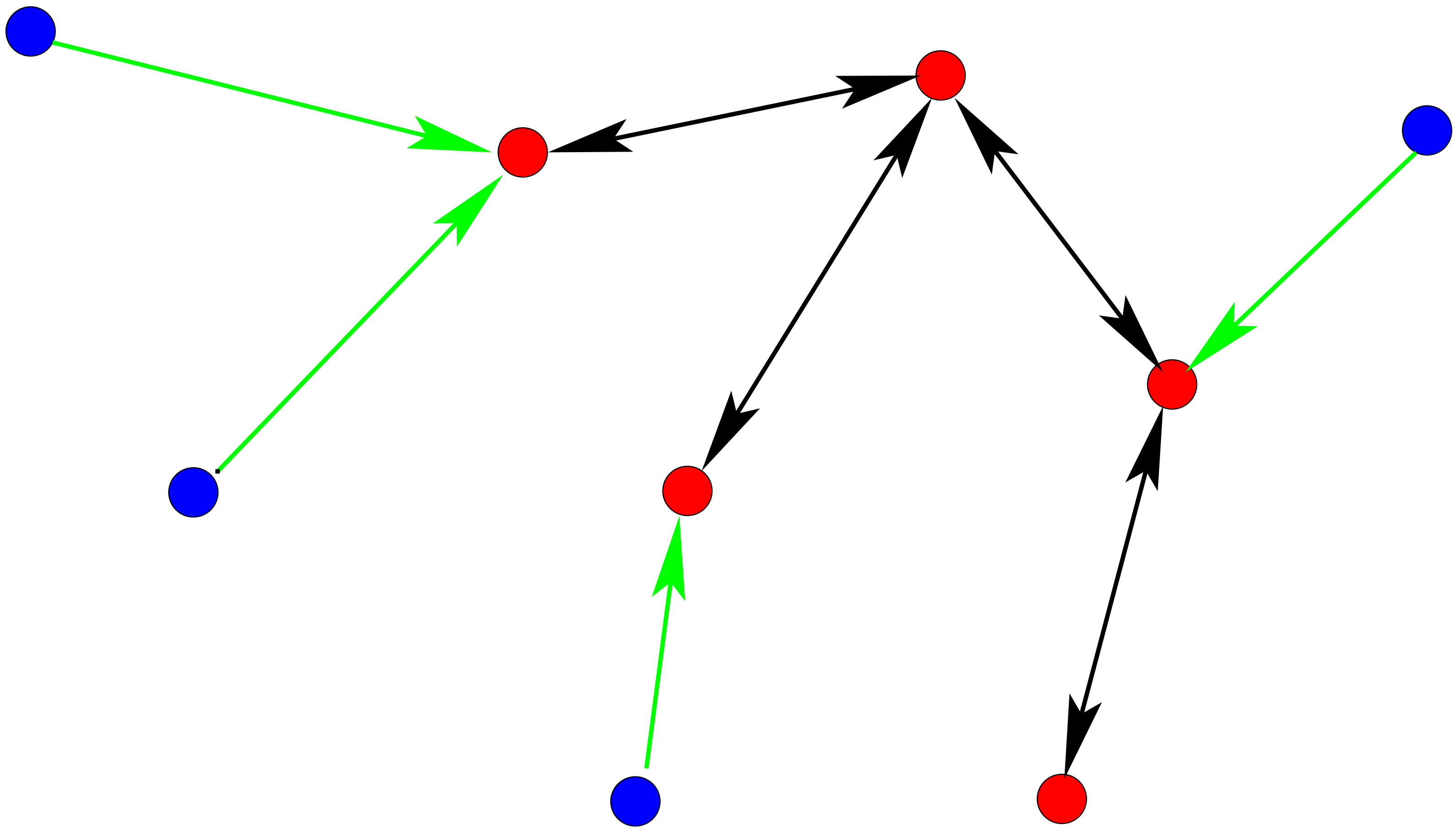
- Back to the animation : Observe the construction of the tree

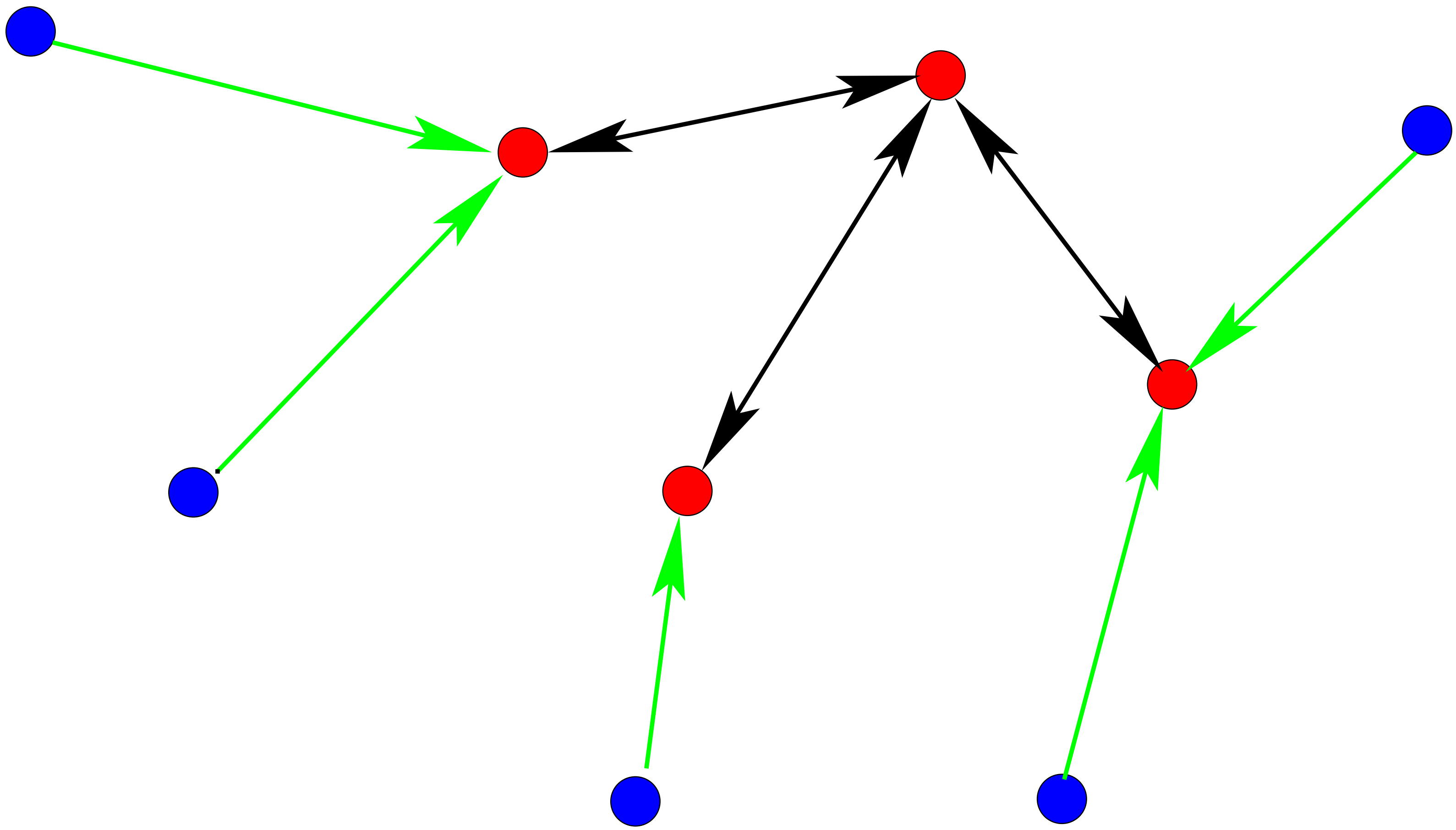


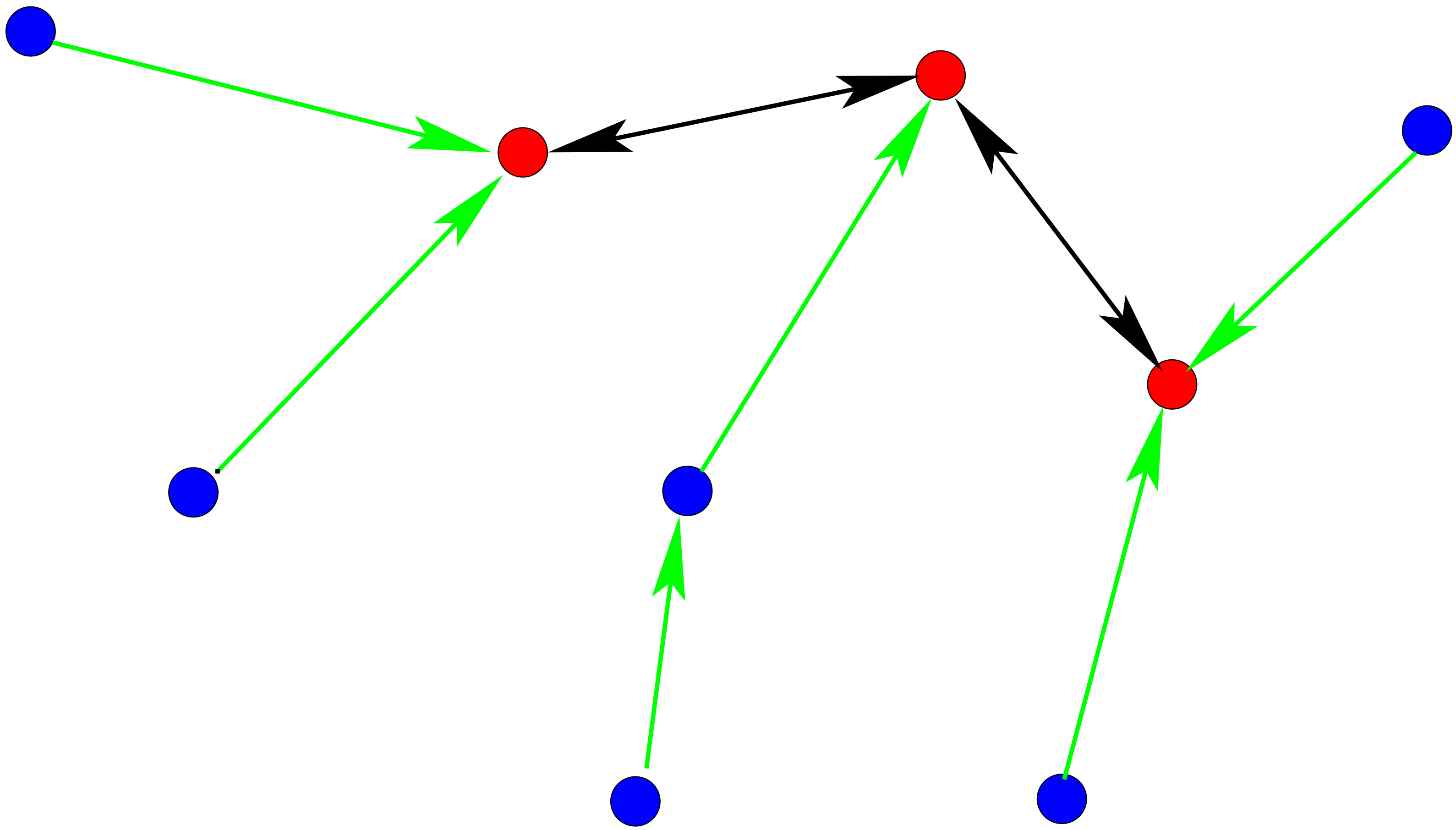


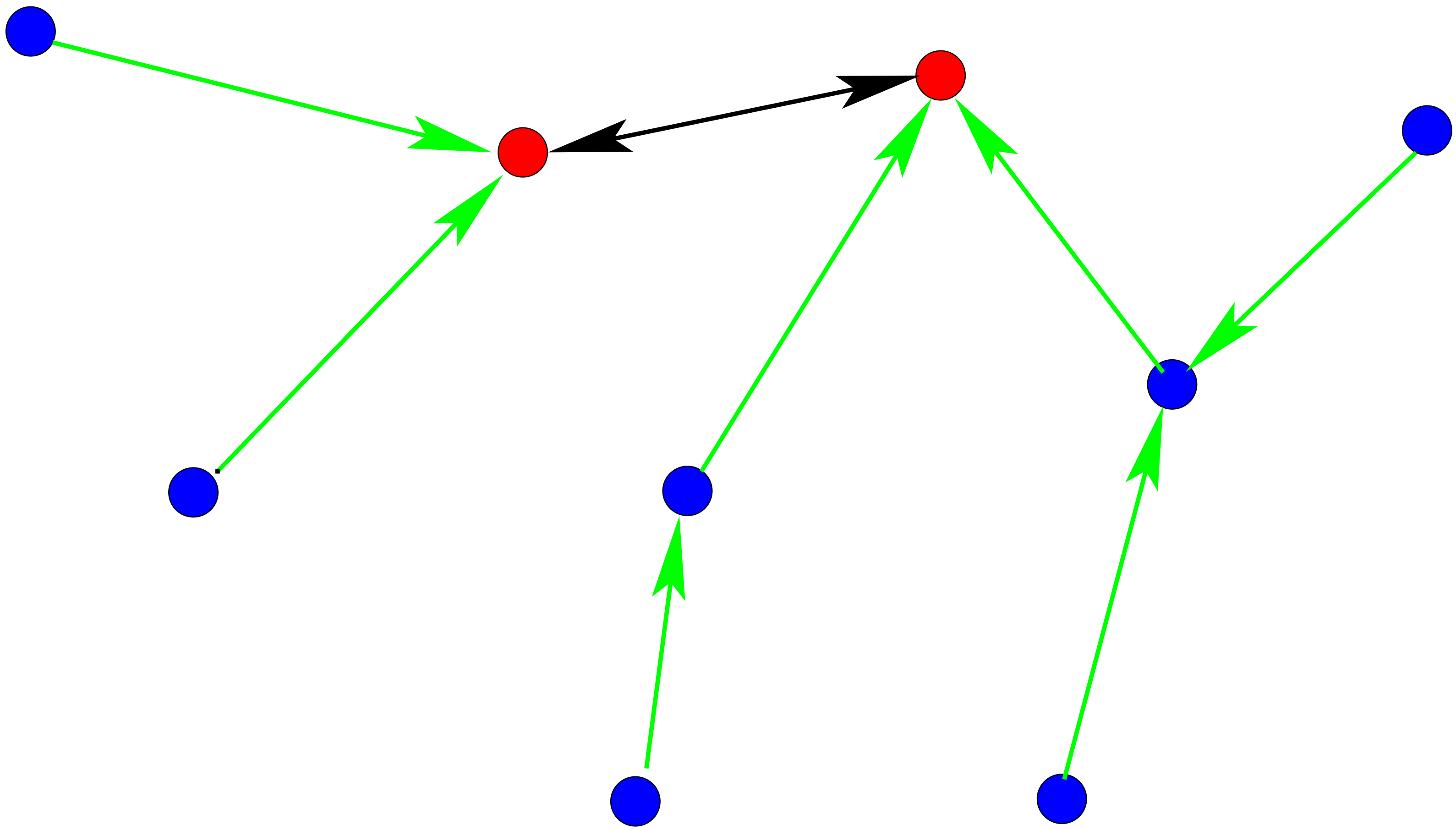


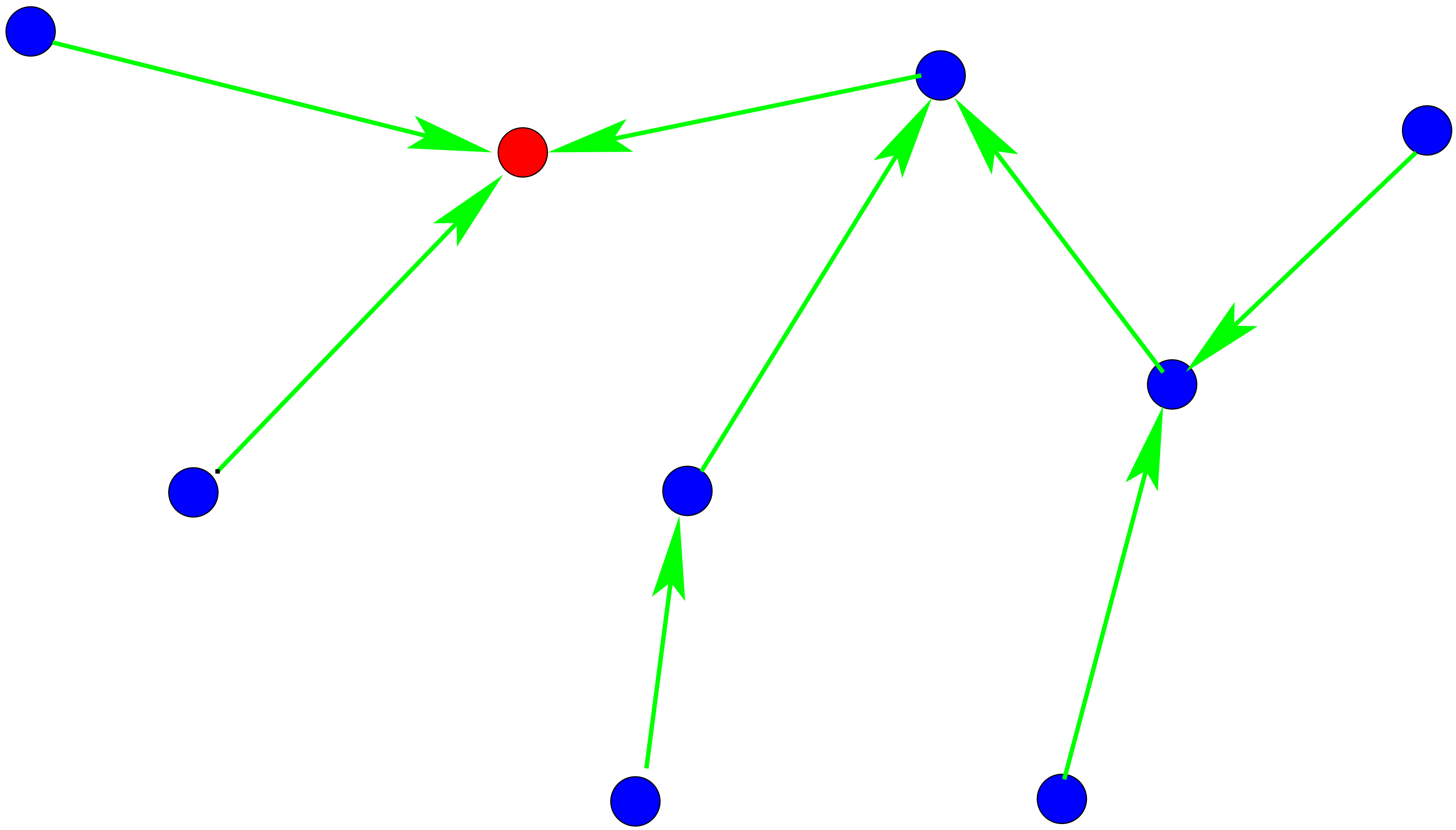












- The green arrows correspond to the tr function

- 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)

$$tr \in ND \nrightarrow ND$$

The predicate invariant (tr)

$$tr \in ND \leftrightarrow ND$$

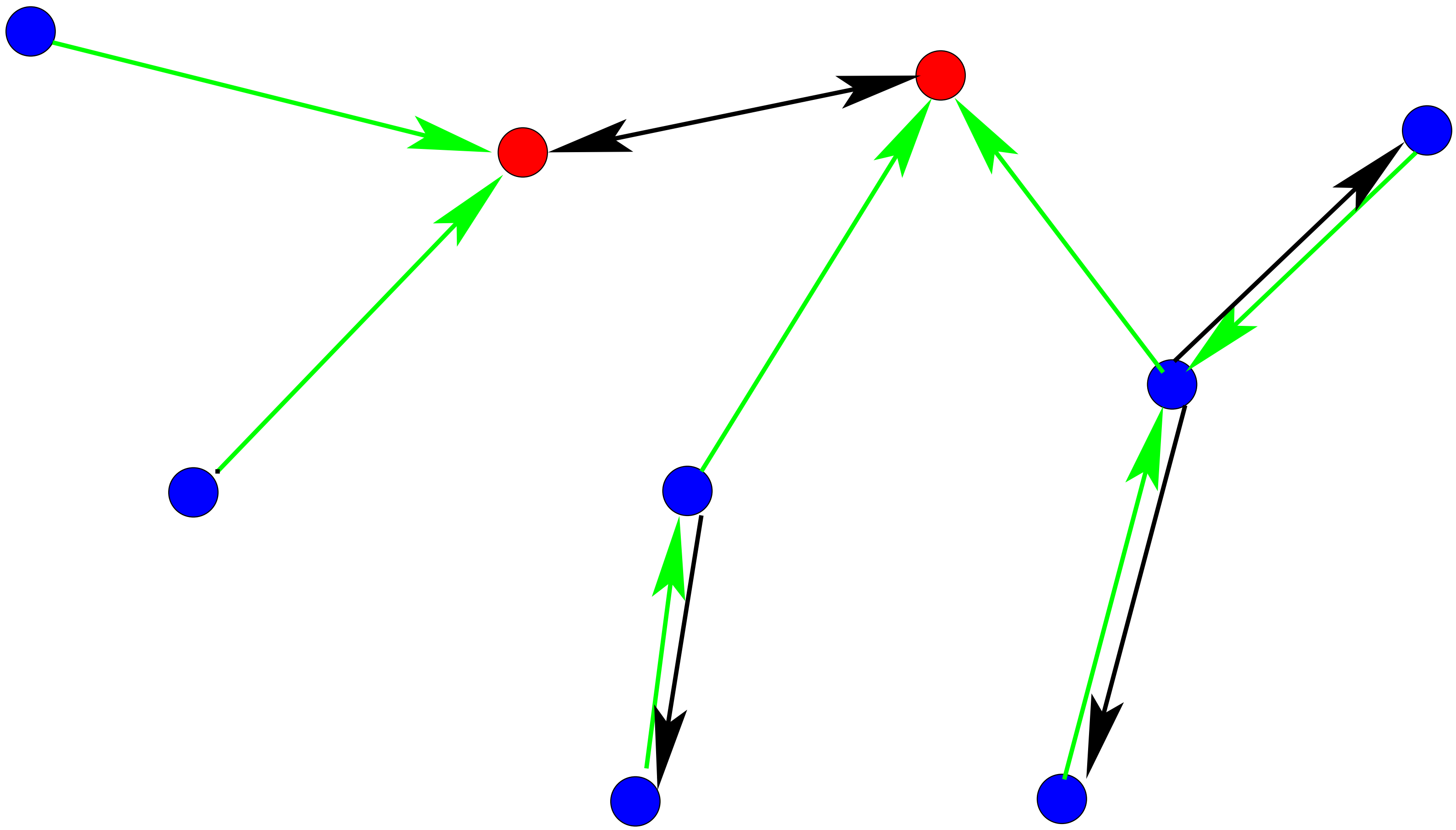
$$\forall p \cdot \left(\begin{array}{l} \quad p \subseteq ND \qquad \wedge \\ \quad ND - \mathsf{dom}\,(tr) \subseteq p \qquad \wedge \\ \quad tr^{-1}[p] \subseteq p \\ \Rightarrow \\ \quad ND \subseteq p \end{array} \right)$$

The predicate invariant (tr)

$$tr \in ND \leftrightarrow ND$$

$$\forall p \cdot \begin{pmatrix} p \subseteq ND \quad \land \\ ND - \mathsf{dom}\,(tr) \subseteq p \quad \land \\ tr^{-1}\,[p] \subseteq p \\ \Rightarrow \\ ND \subseteq p \end{pmatrix}$$

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

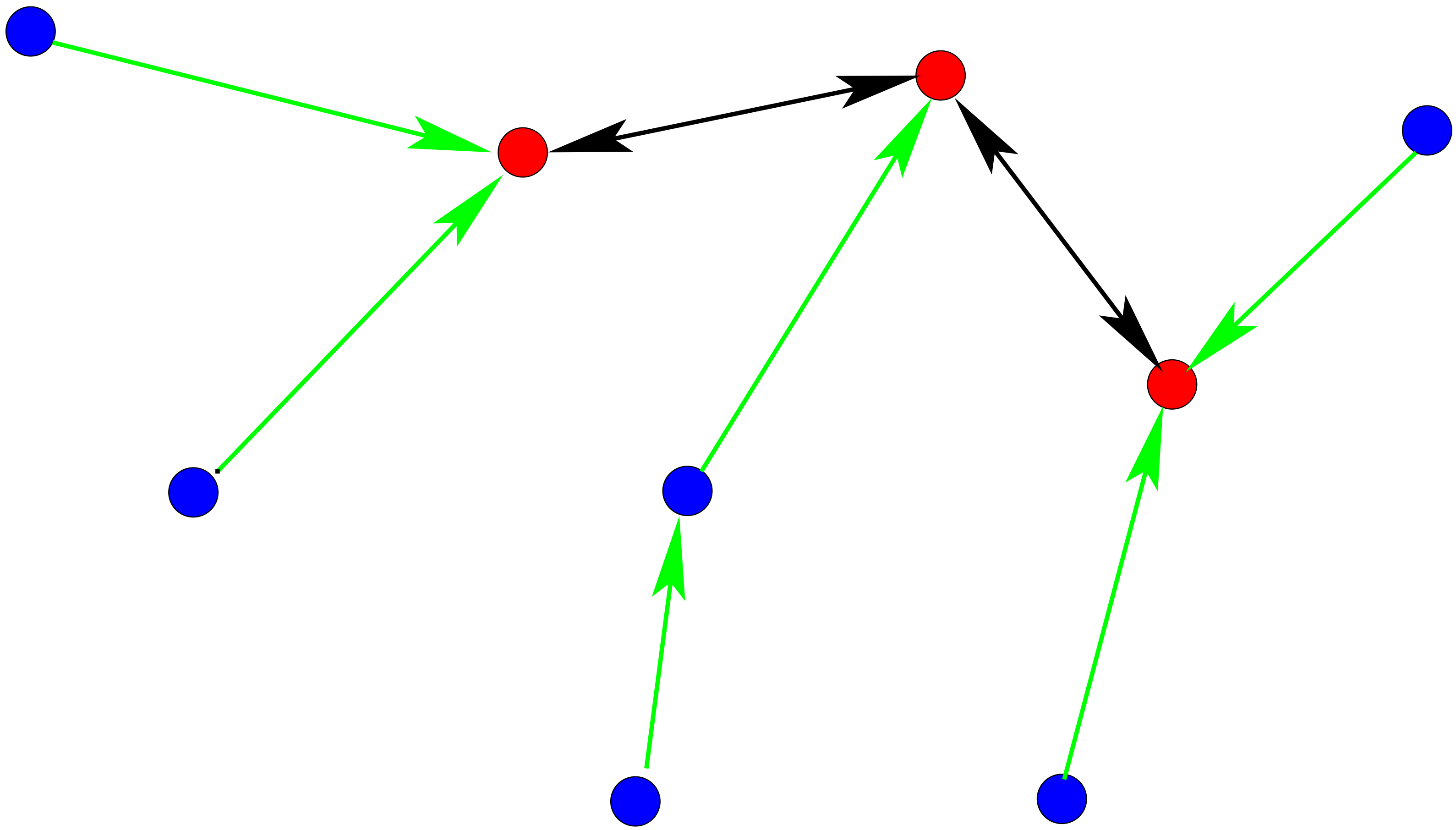


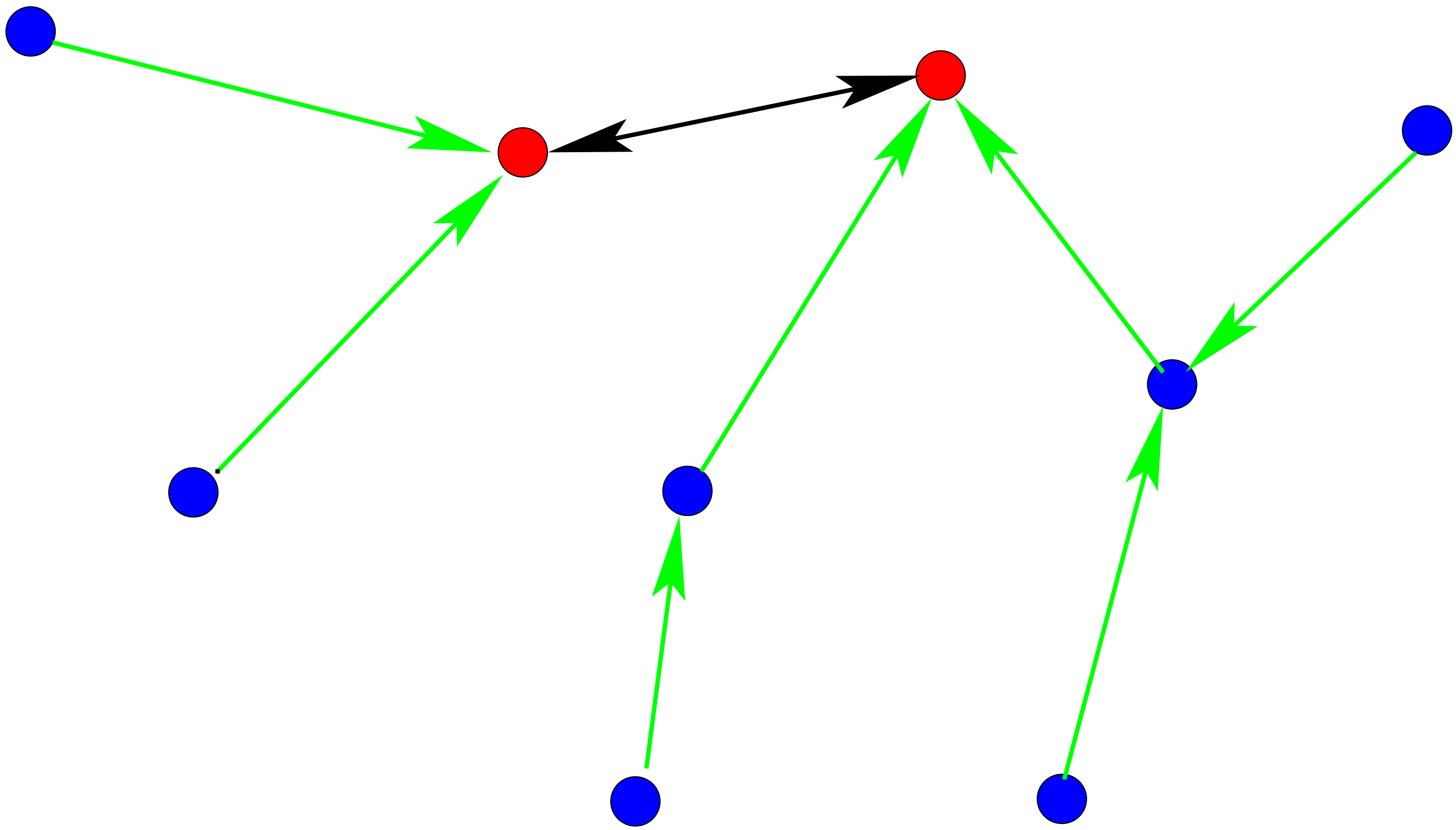
First Refinement (2)

- 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**

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

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

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

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

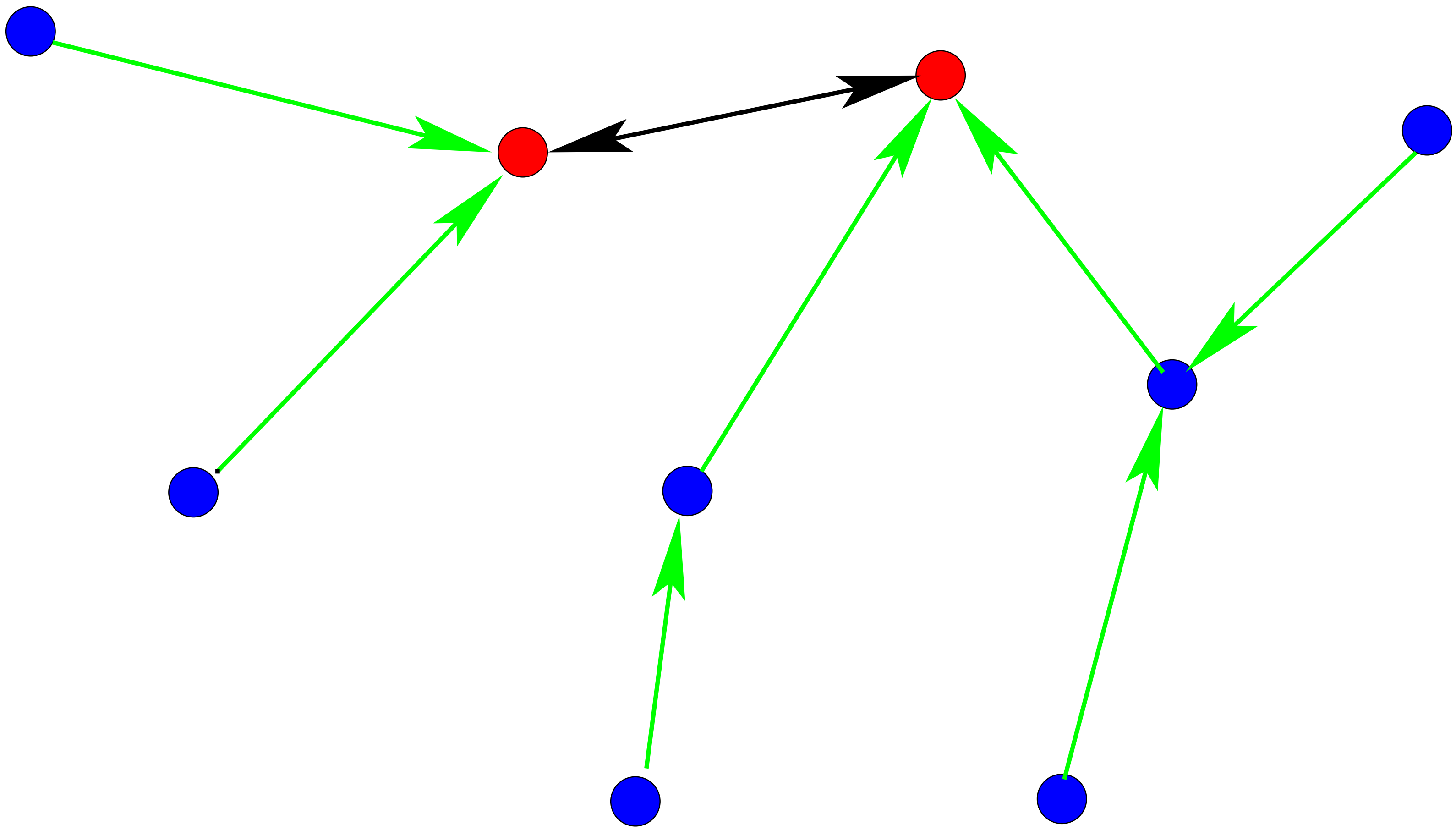
then

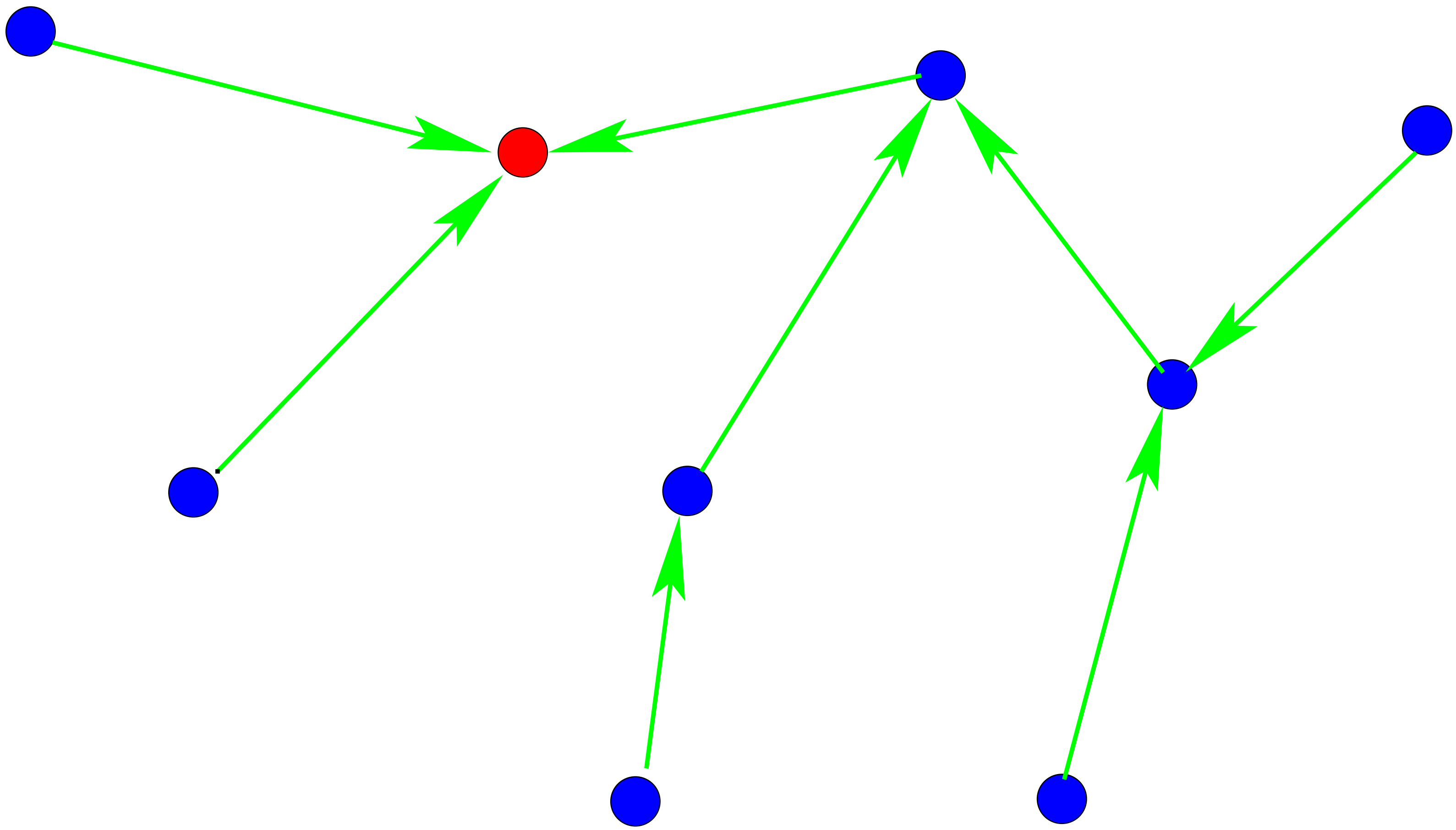
$tr := tr \cup \{x \mapsto y\}$

end

To be proved

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





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

invariant(tr) $\wedge$

$x \in ND$ $\wedge$

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

$ts = tr$

$\Rightarrow$

spanning(x, ts, gr)

Summary of First Refinement

- 15 proofs

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

Second Refinement

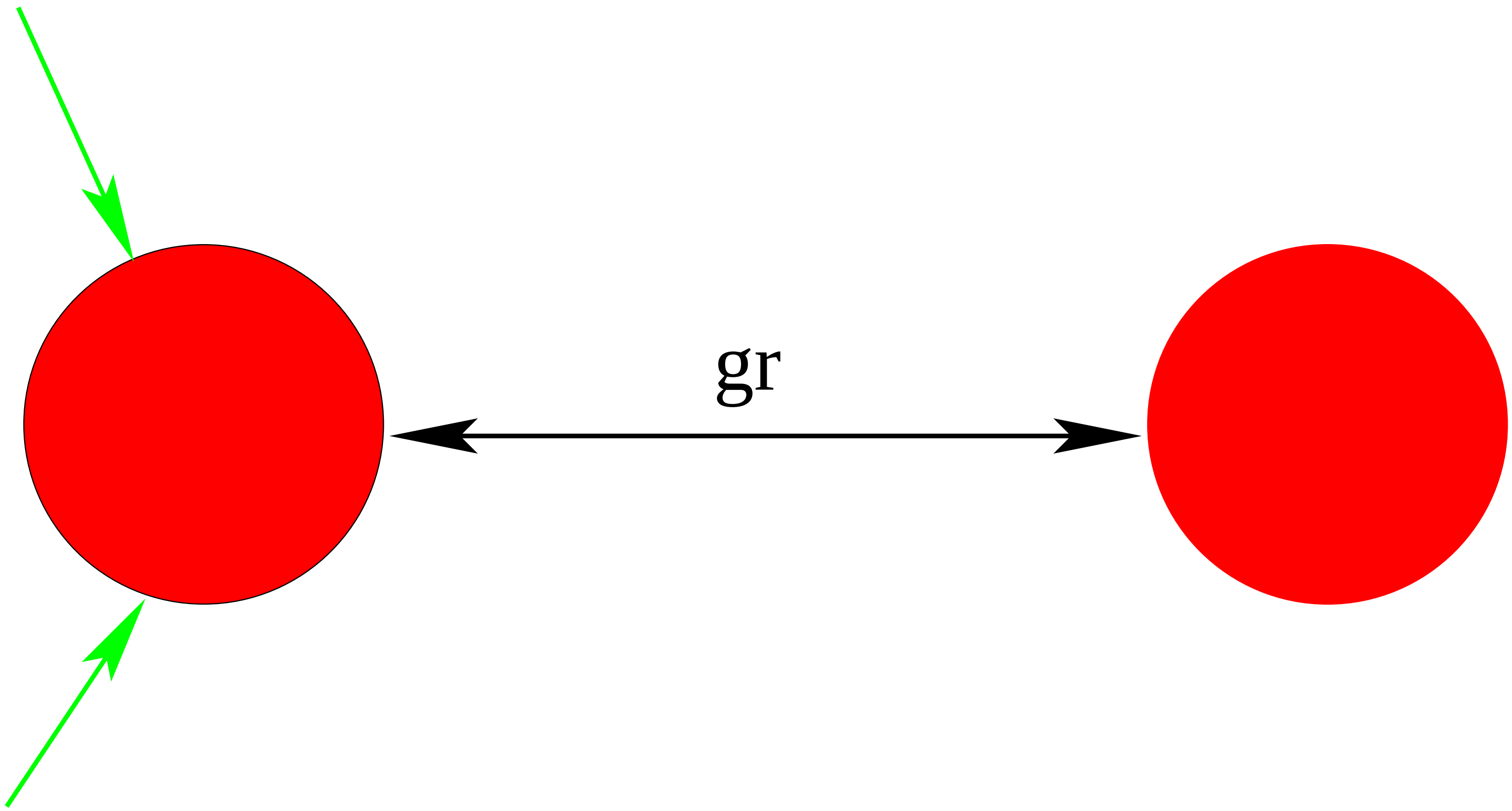
- Nodes are communicating with their neighbors

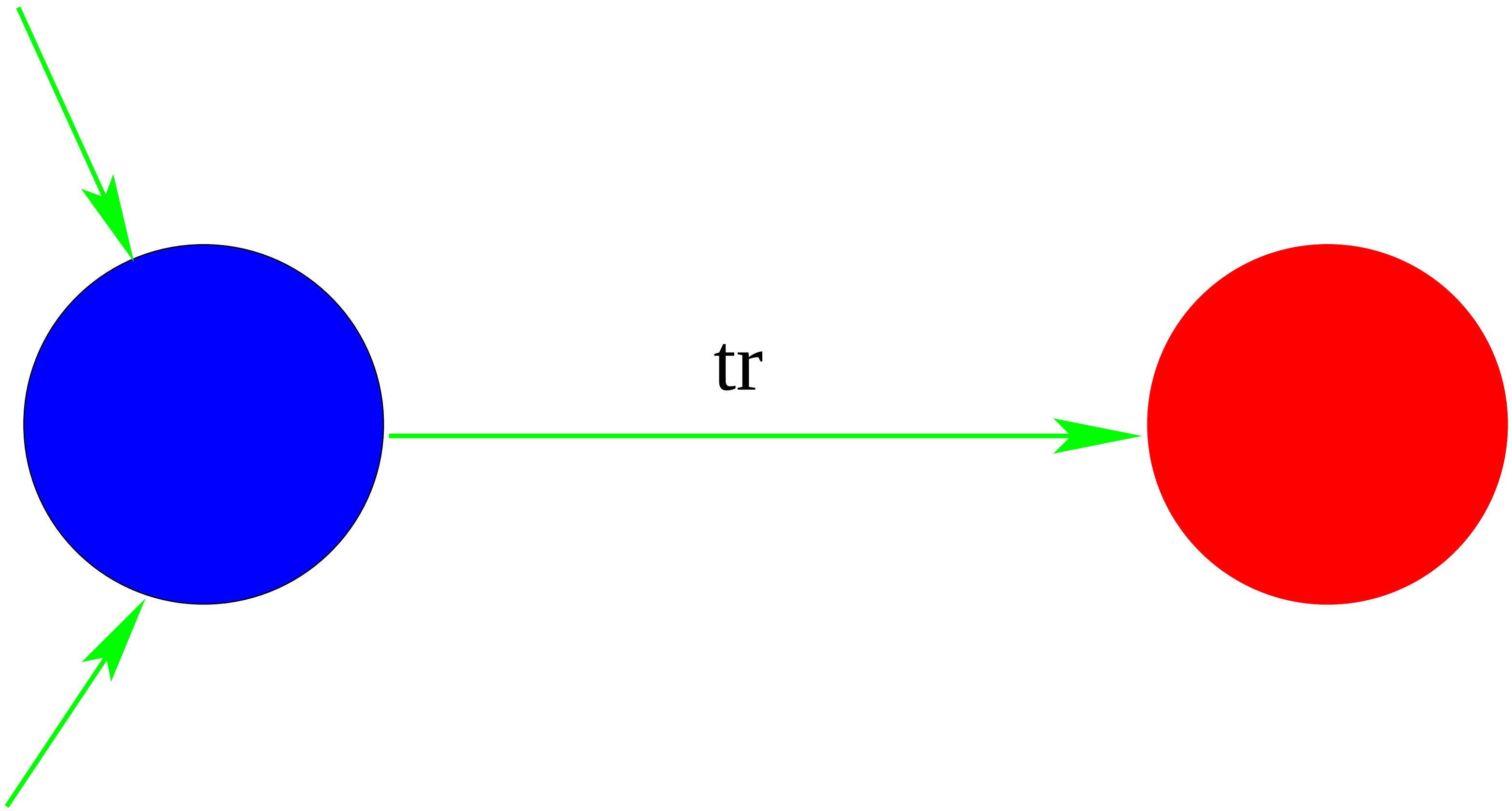
- This is done by means of messages

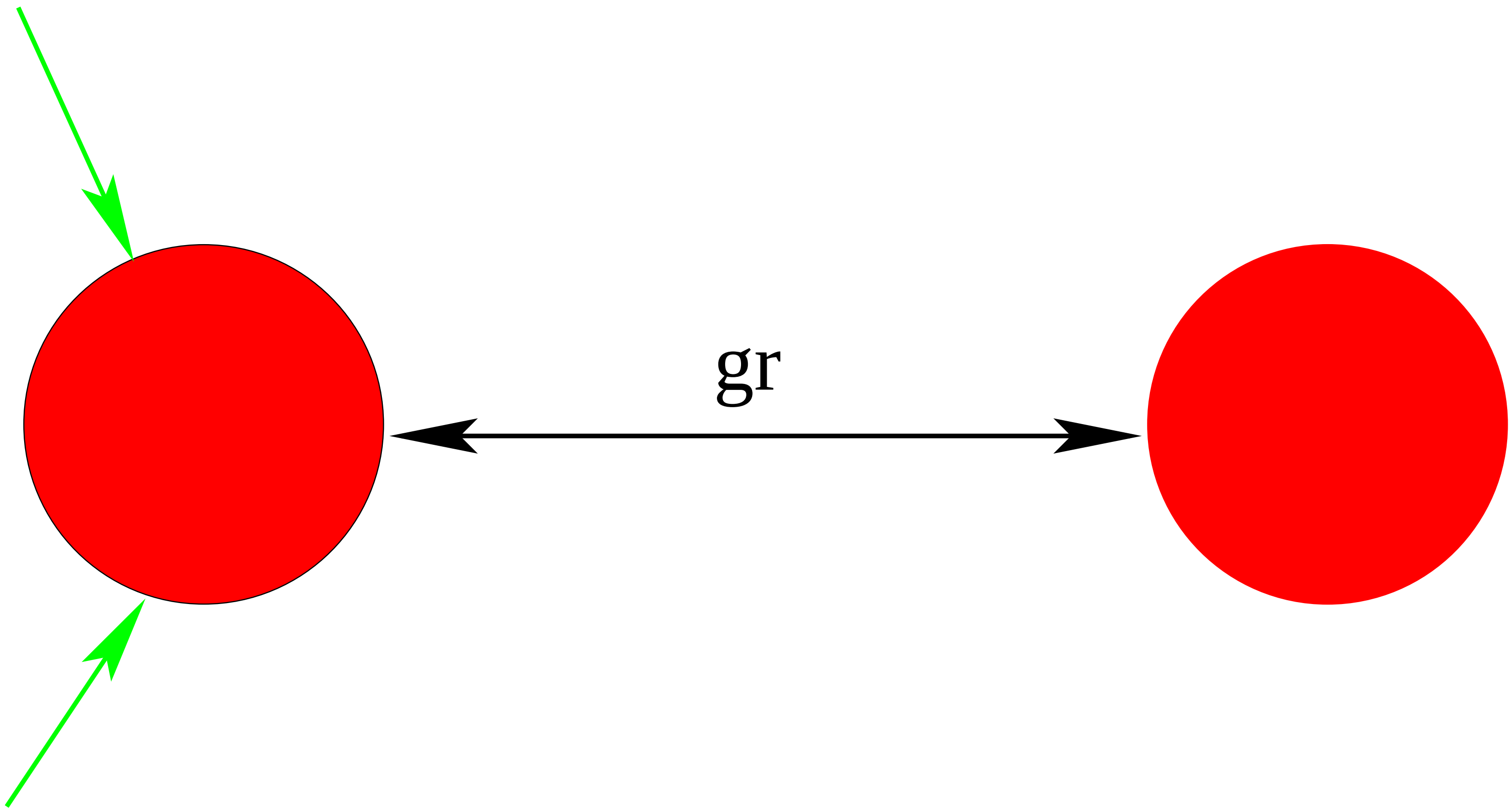
- Messages are acknowledged

- Acknowledgements are confirmed

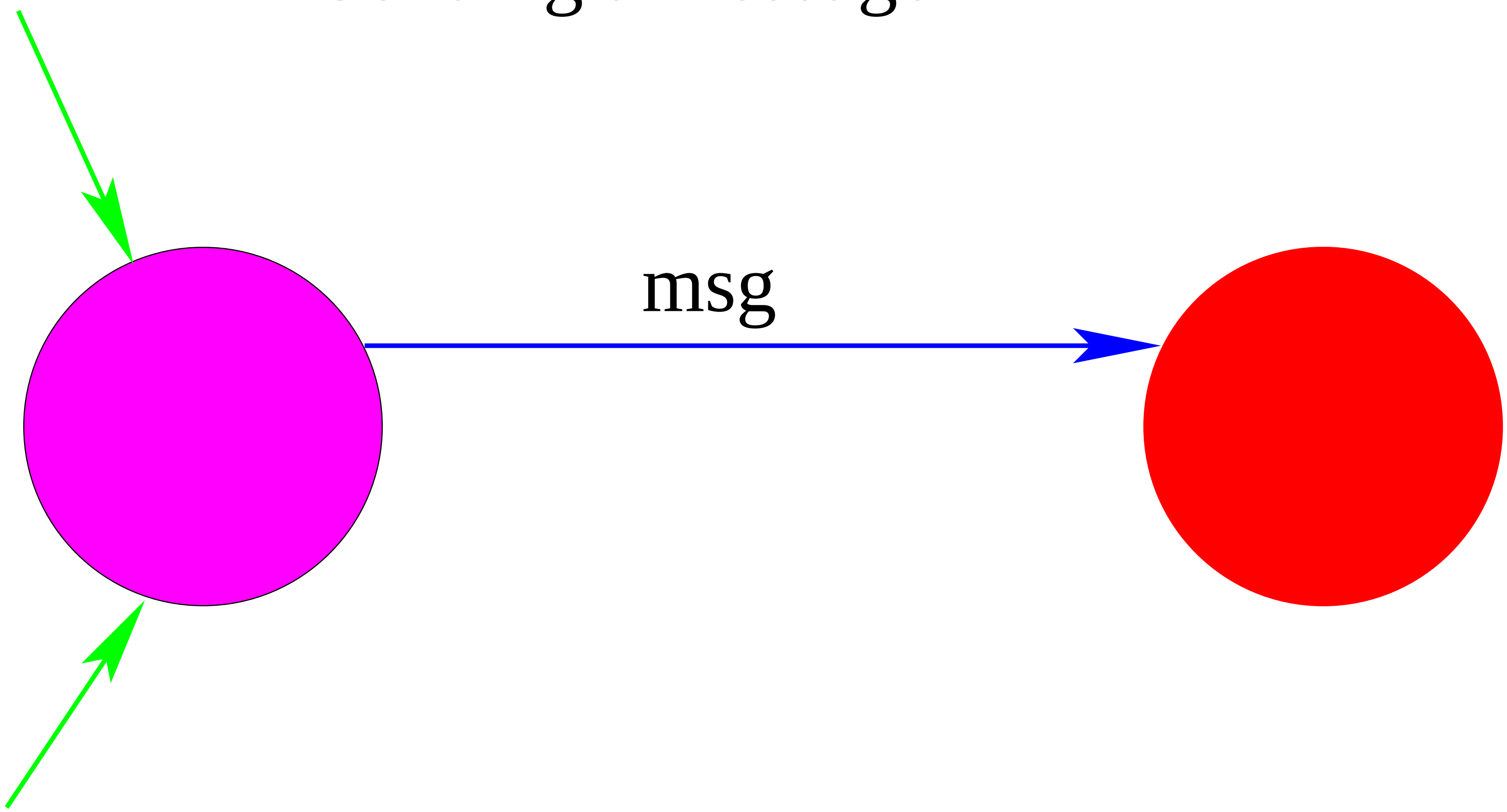
- Next is a local animation


gr


tr

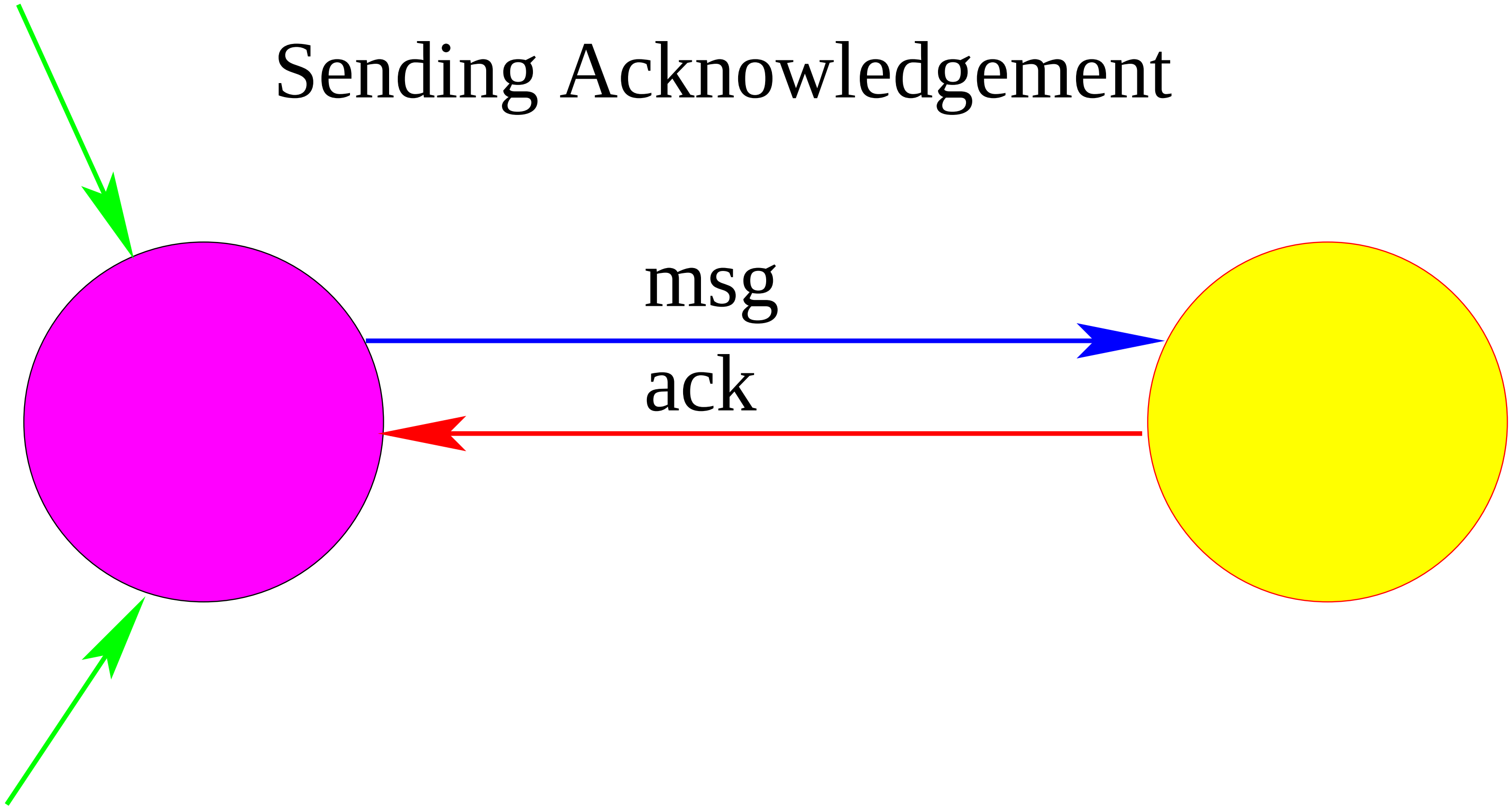

gr

Sending a message



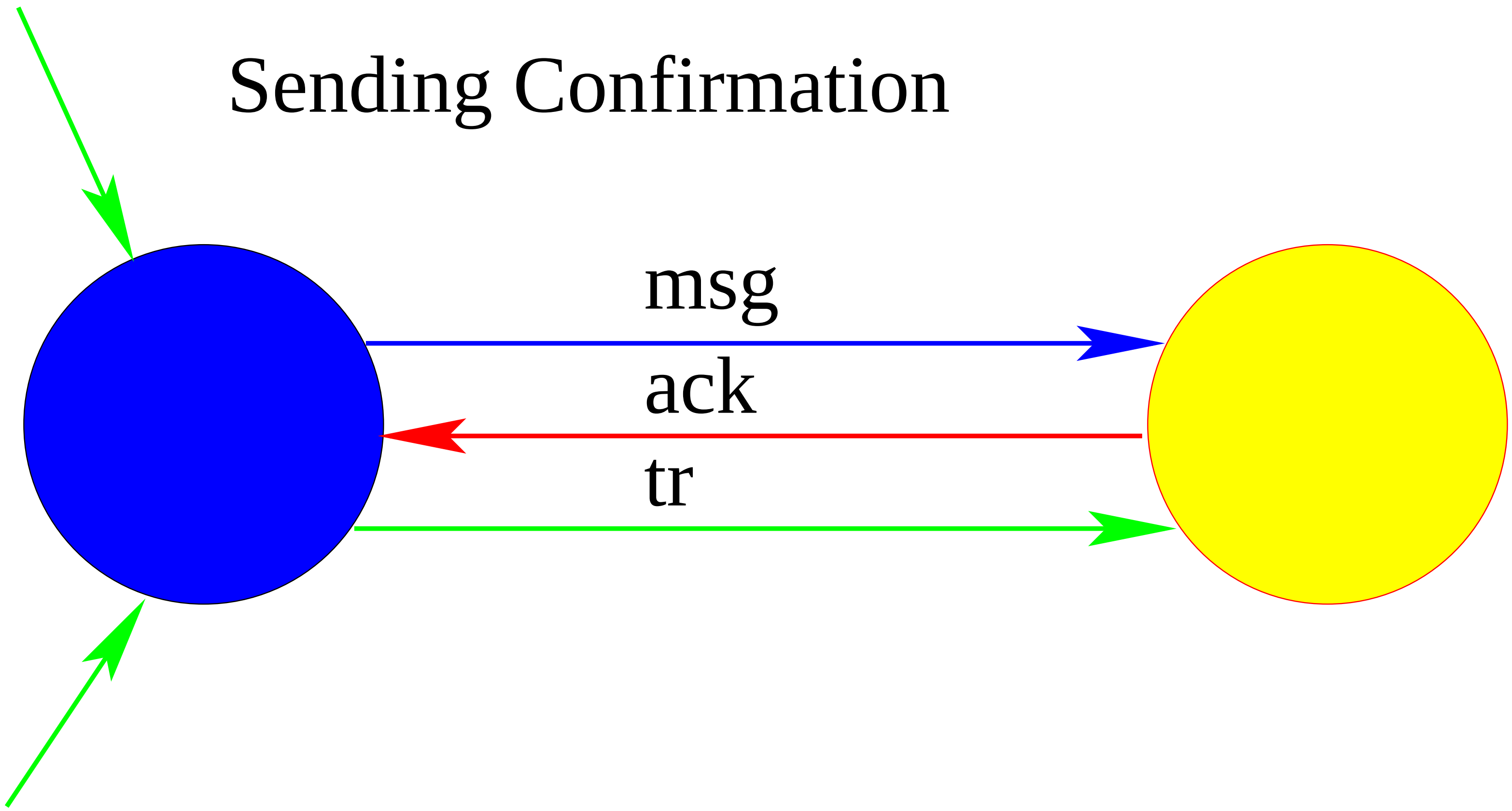
Receiving a message

Sending Acknowledgement

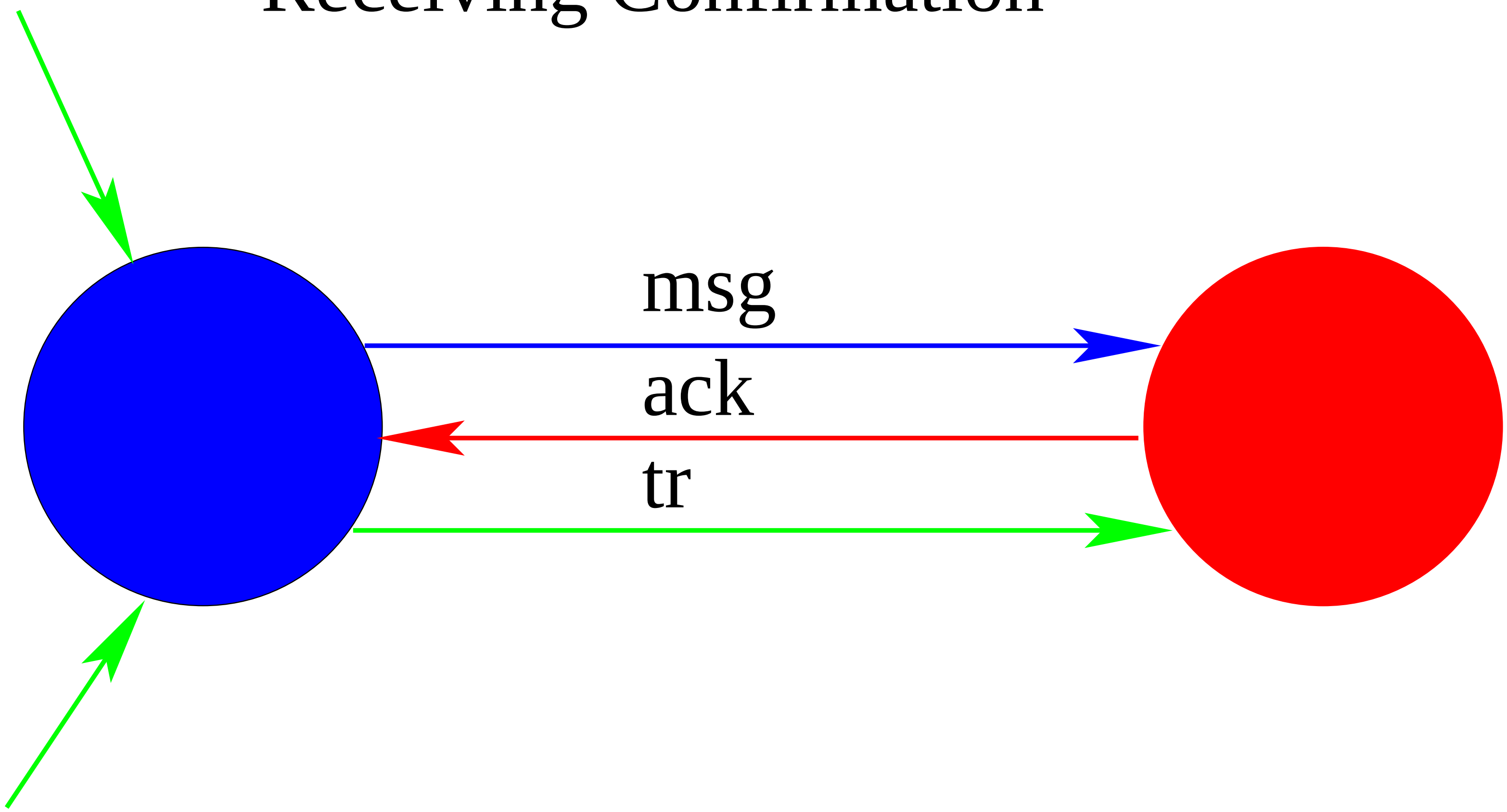


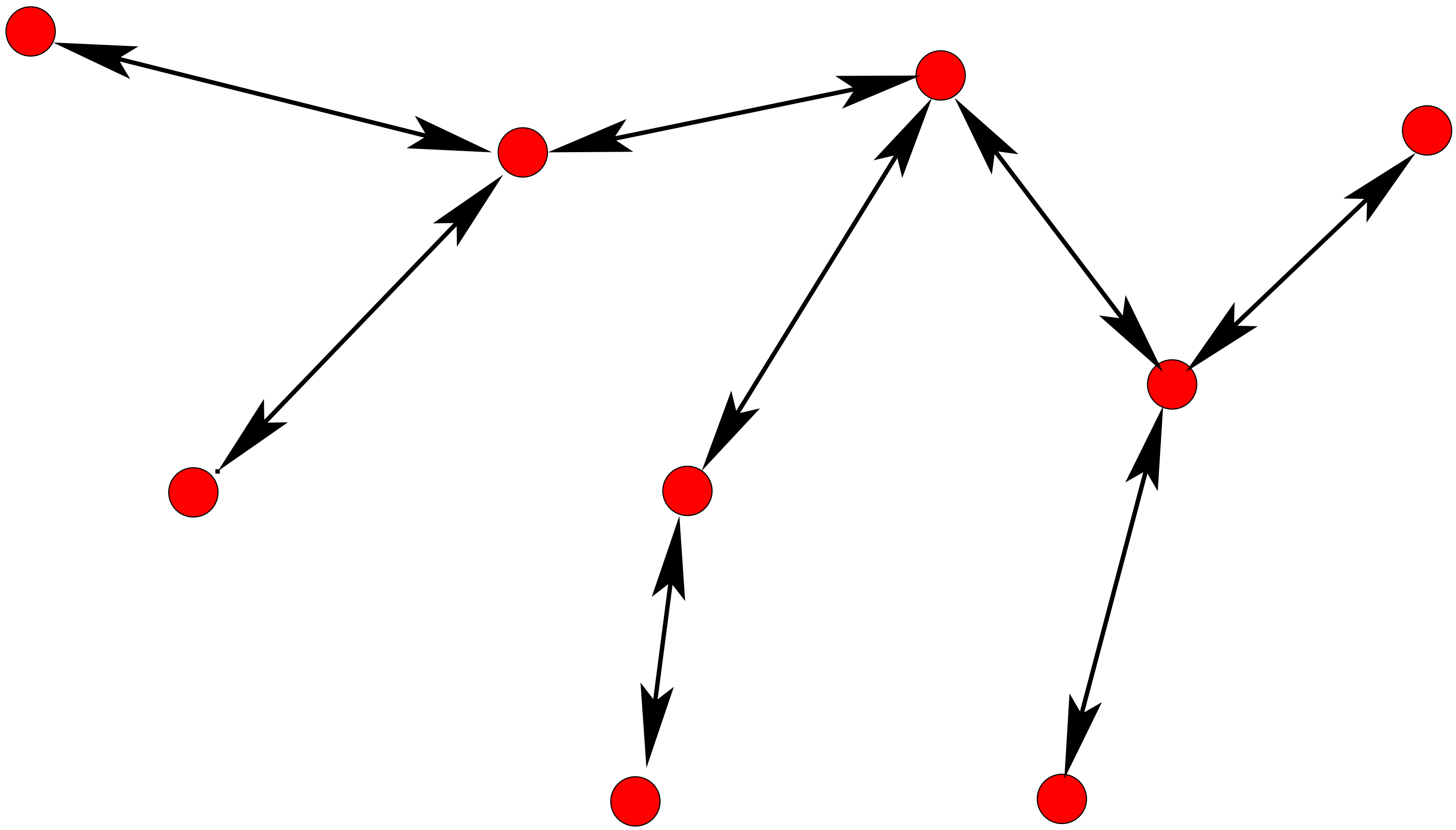
Receiving Acknowledgement

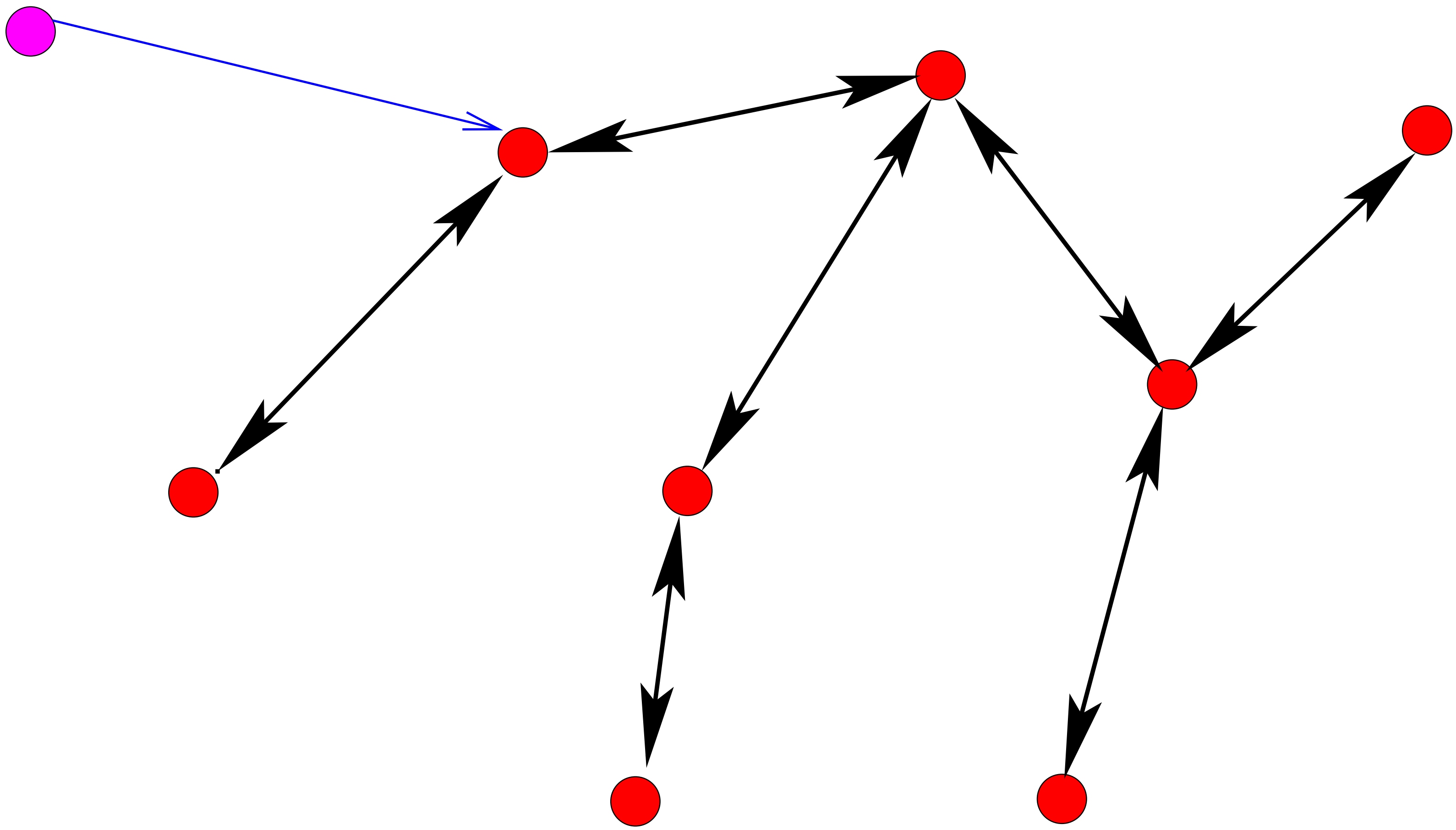
Sending Confirmation

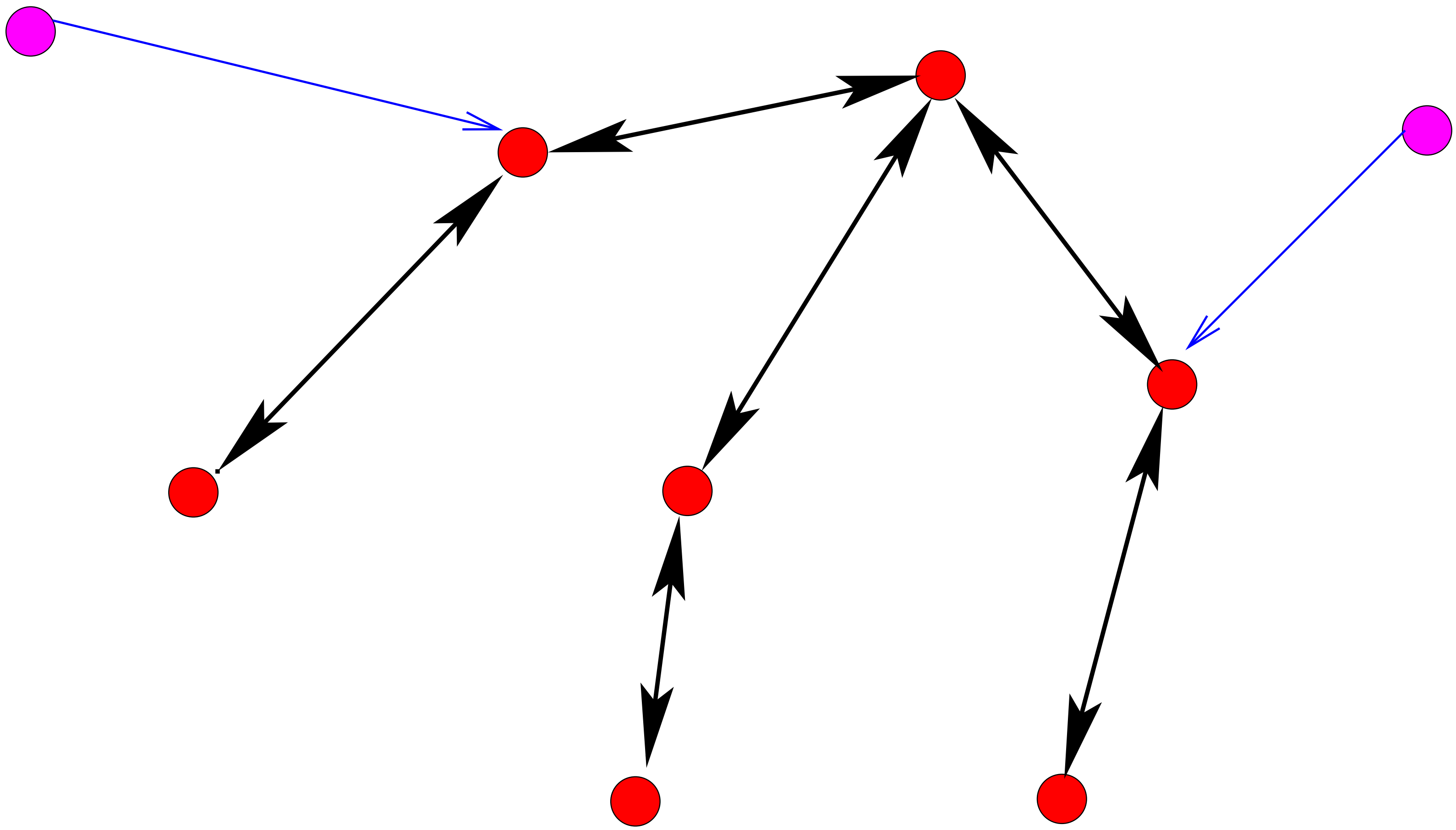


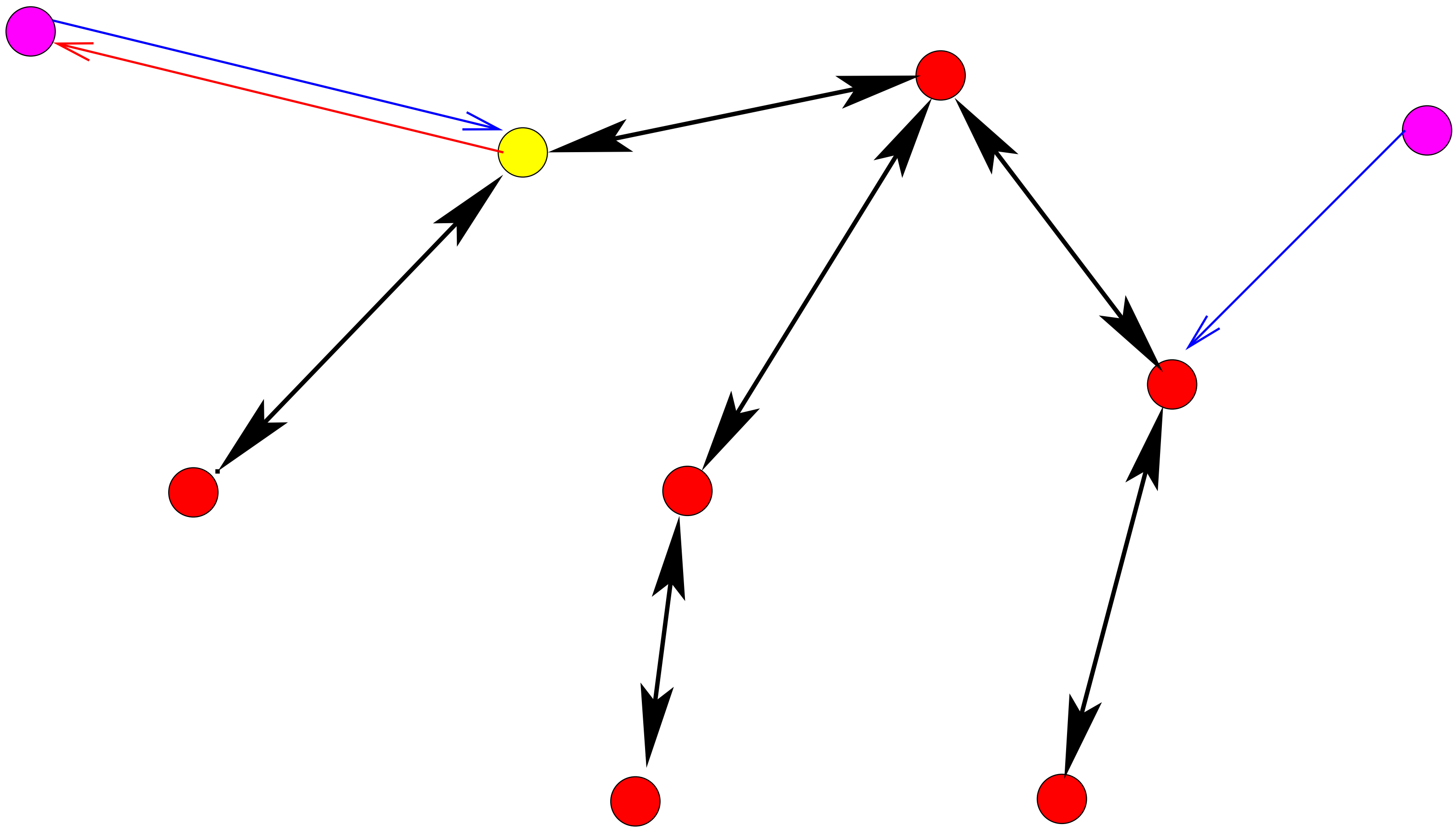
Receiving Confirmation

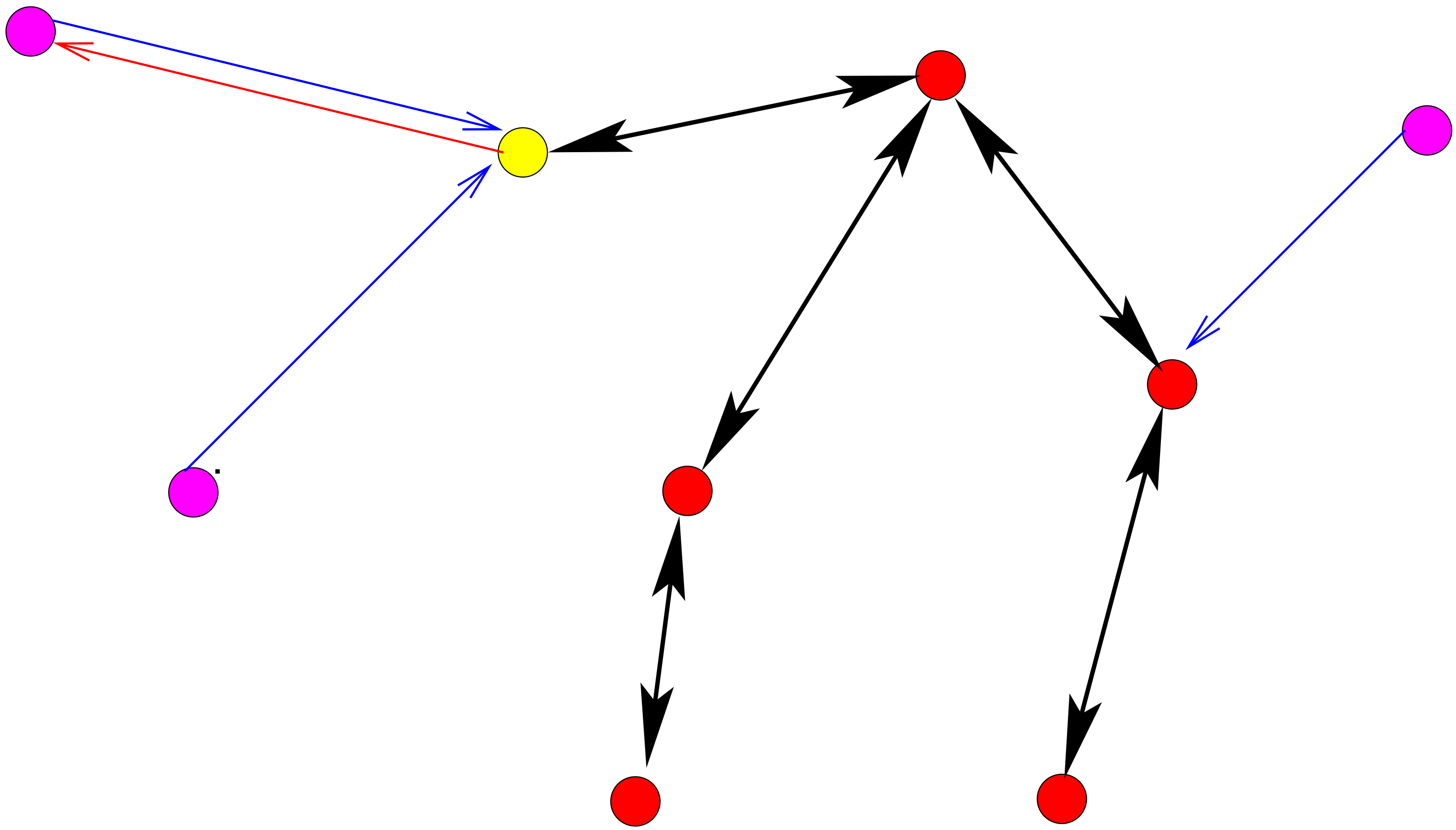


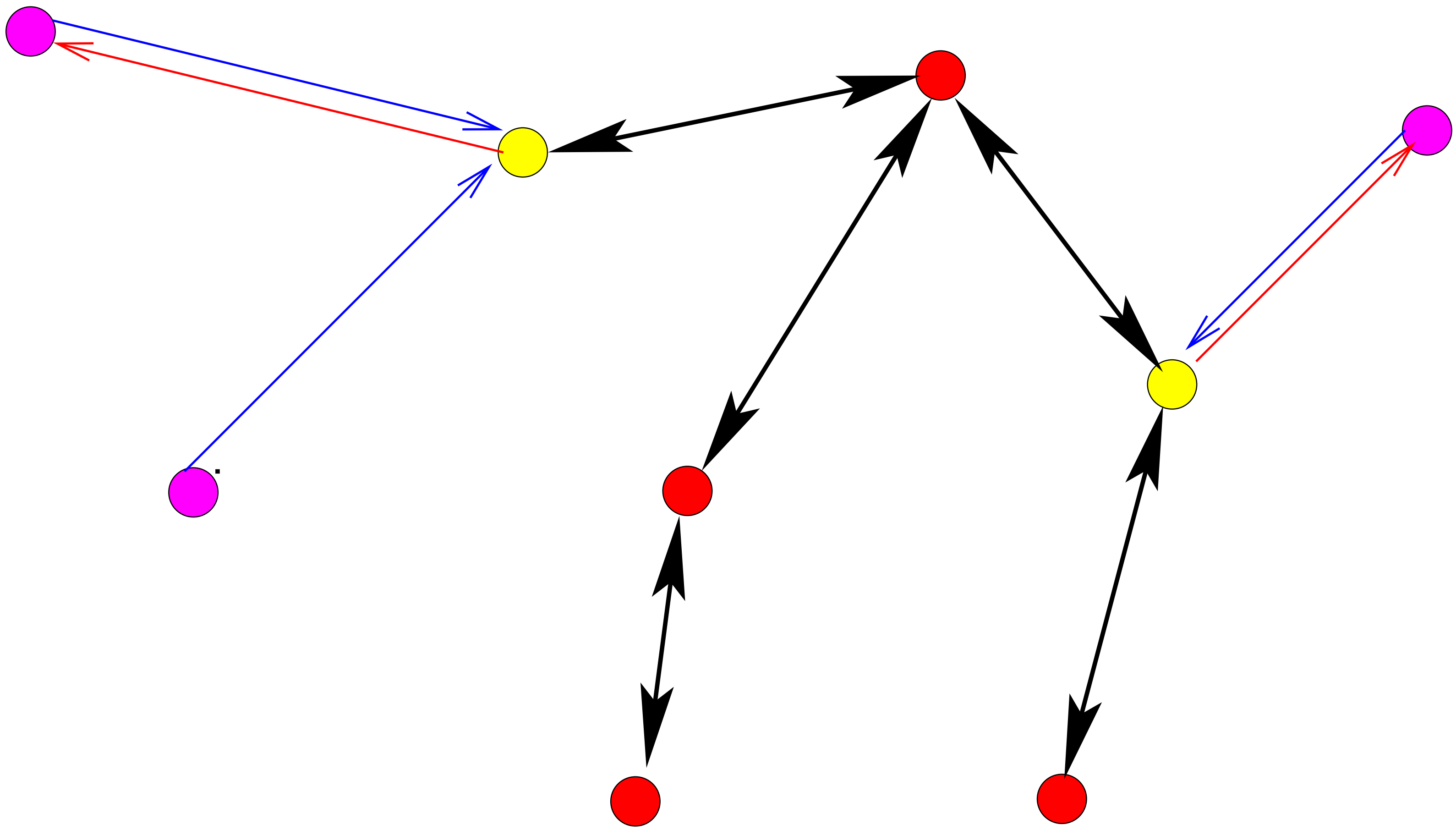


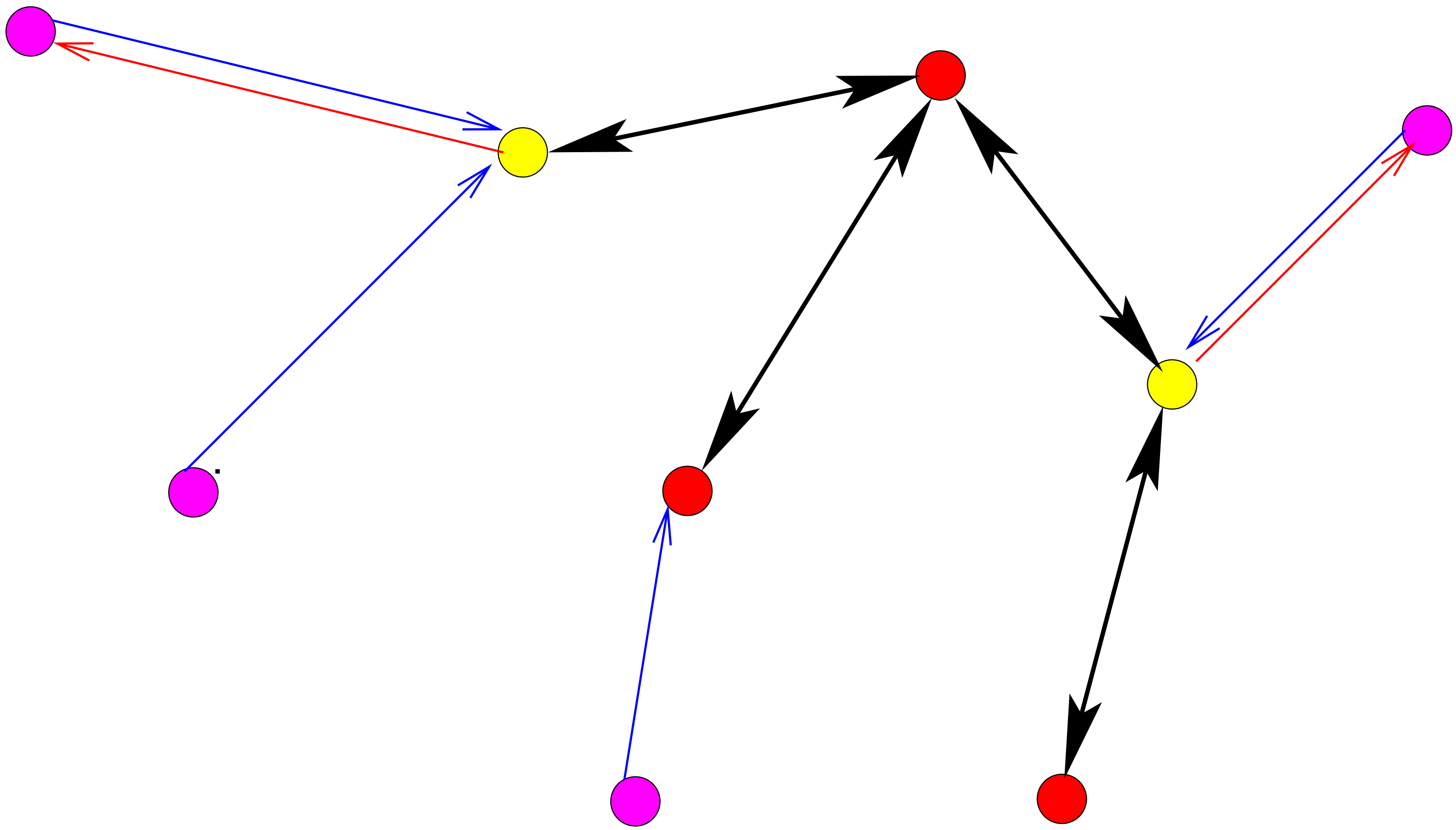


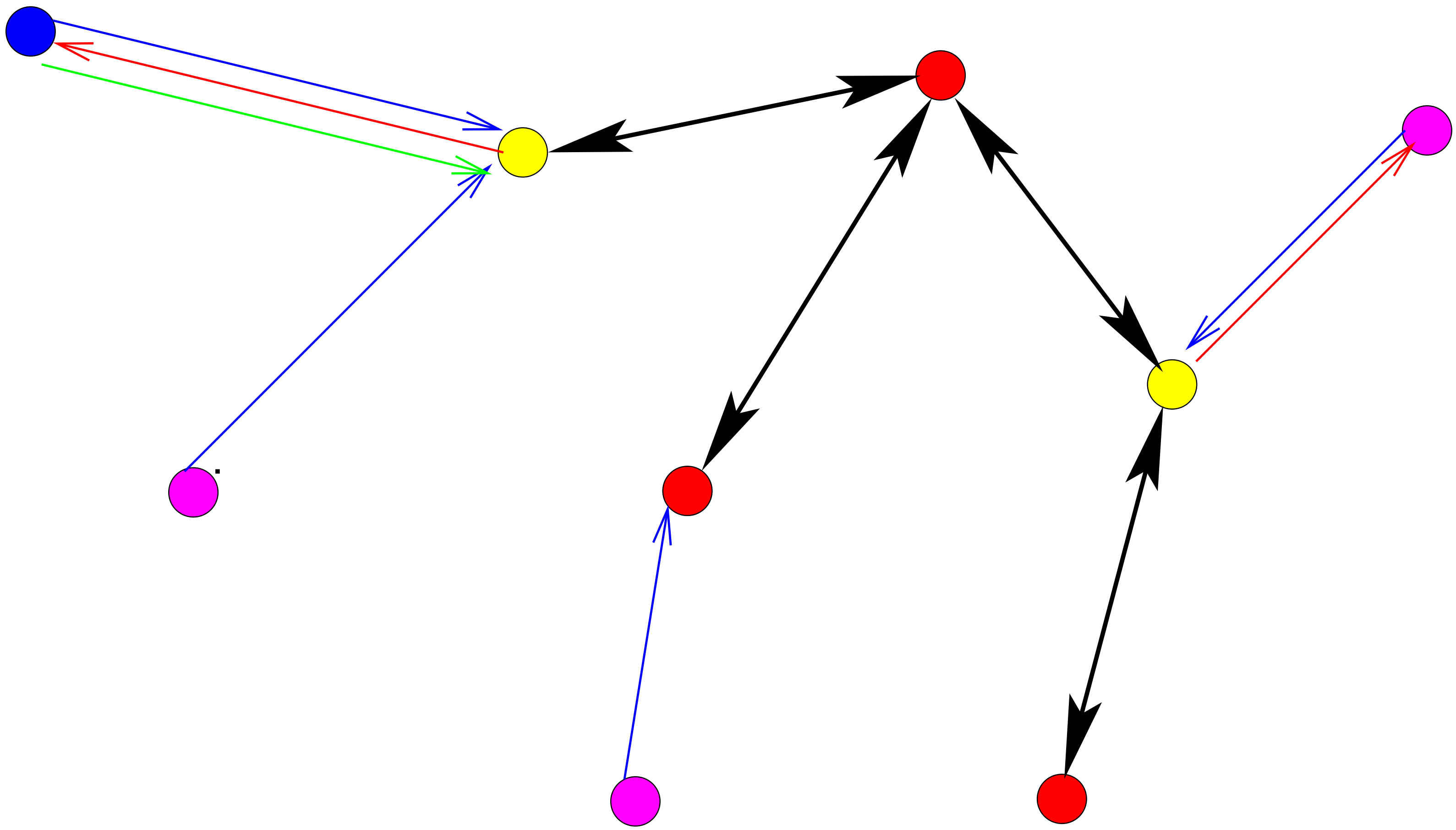


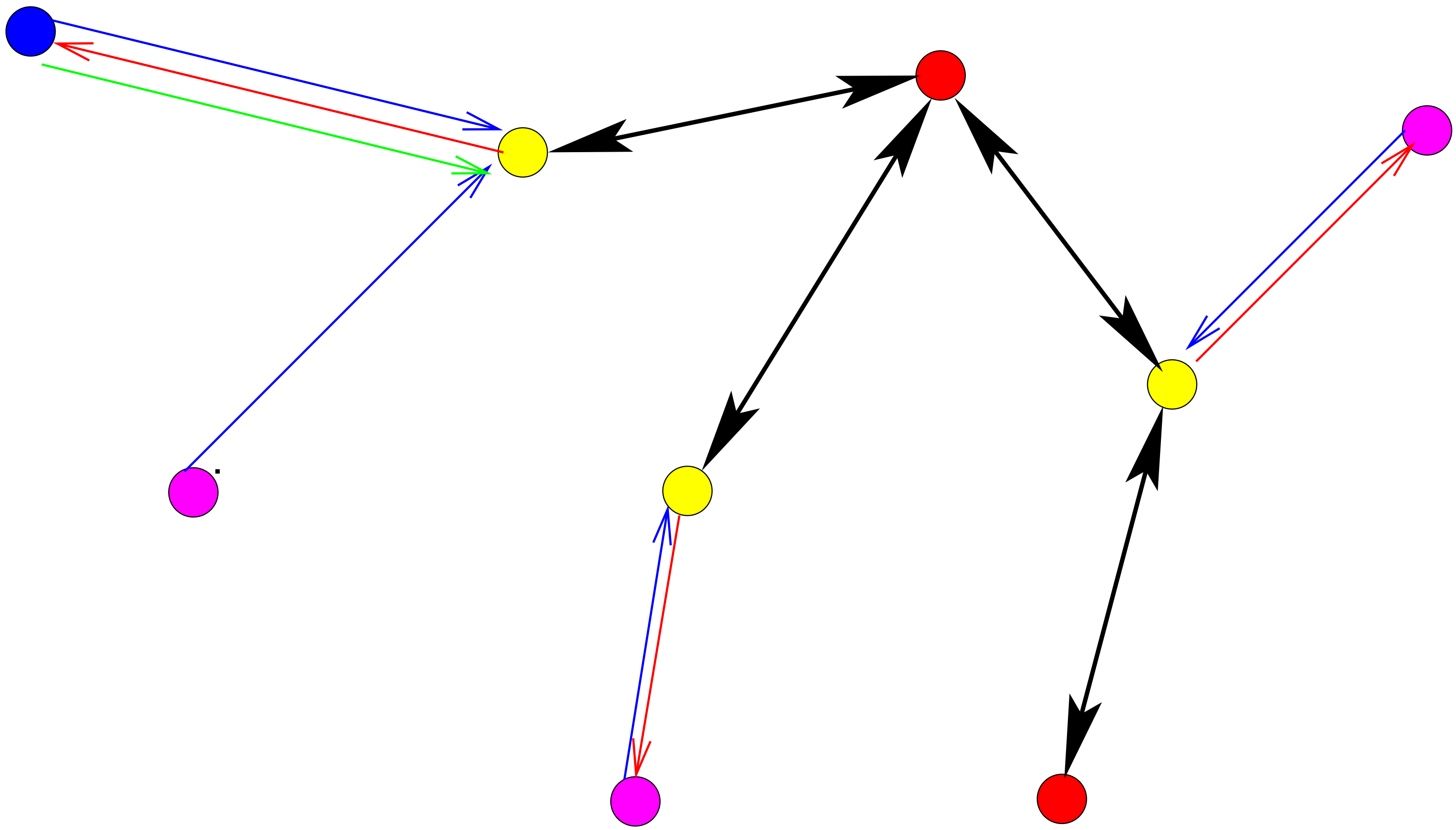


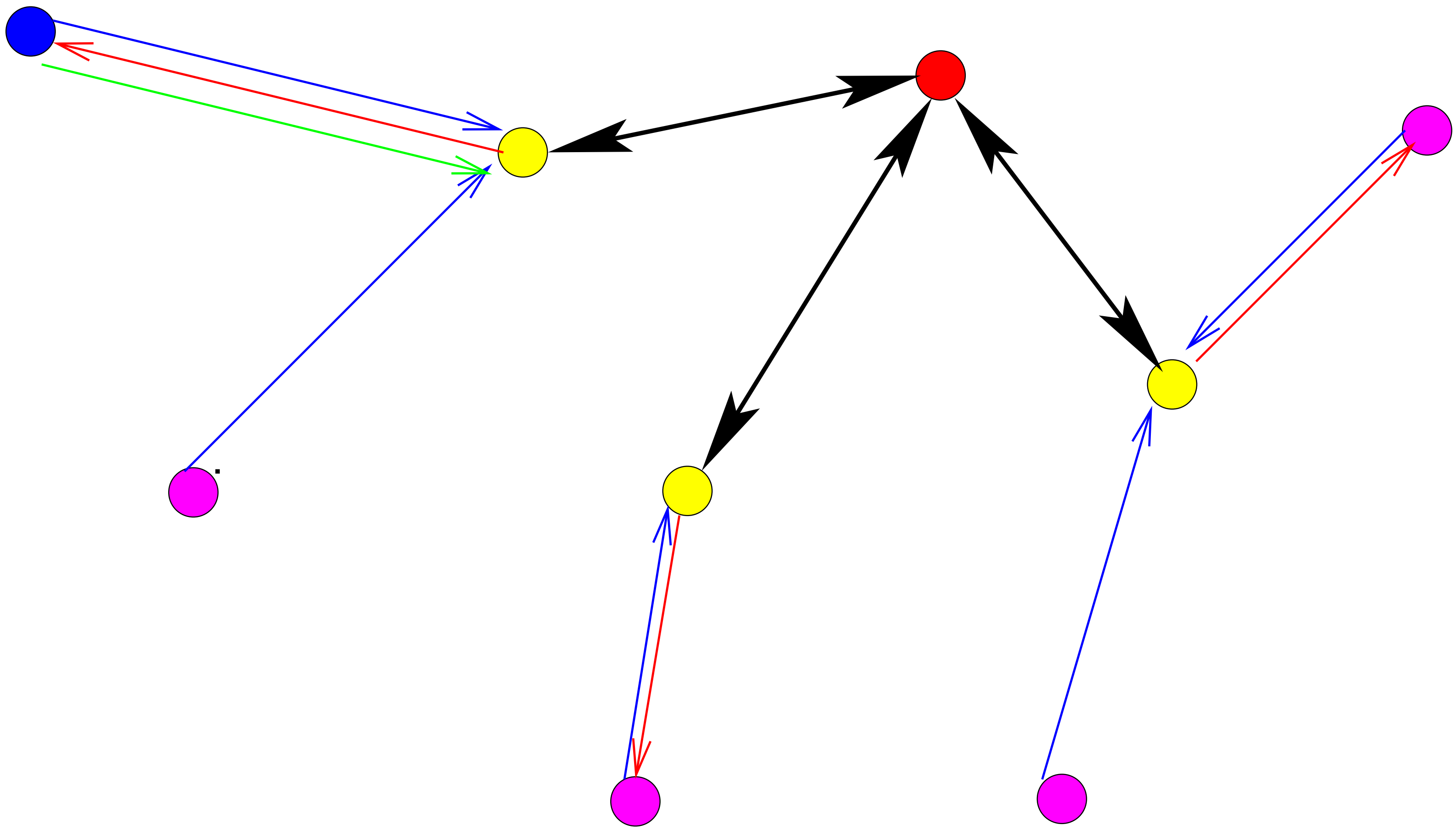


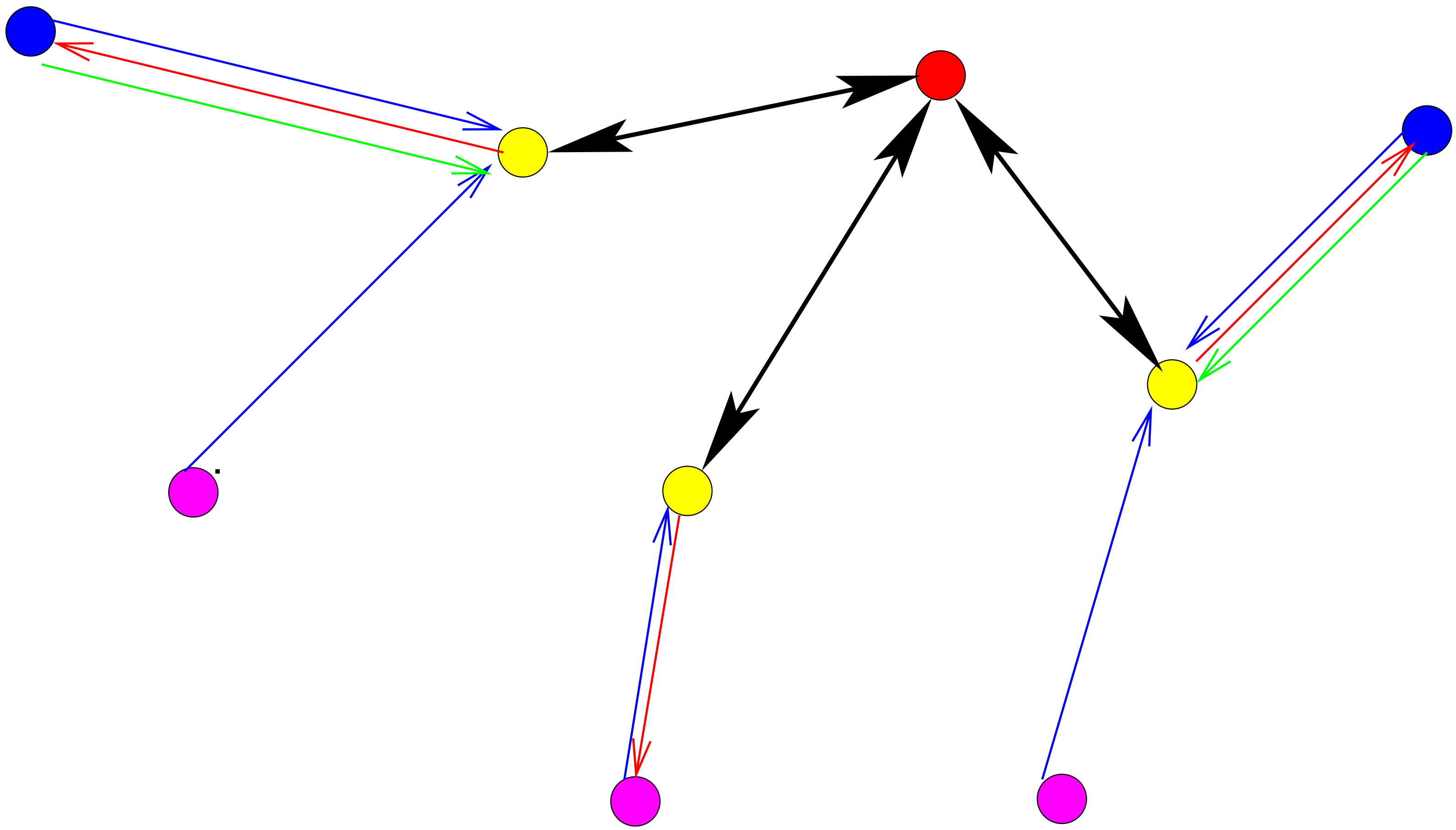


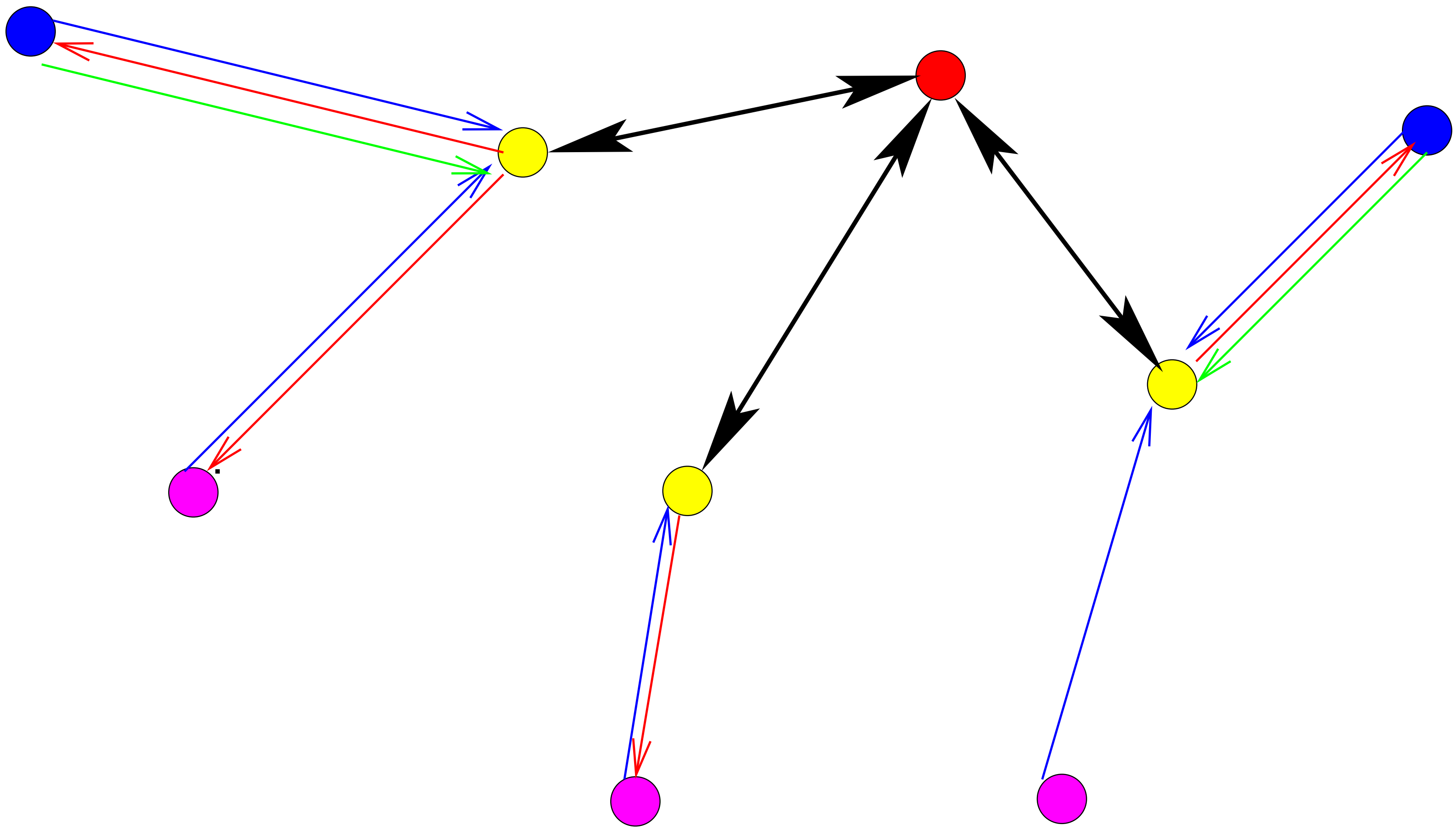


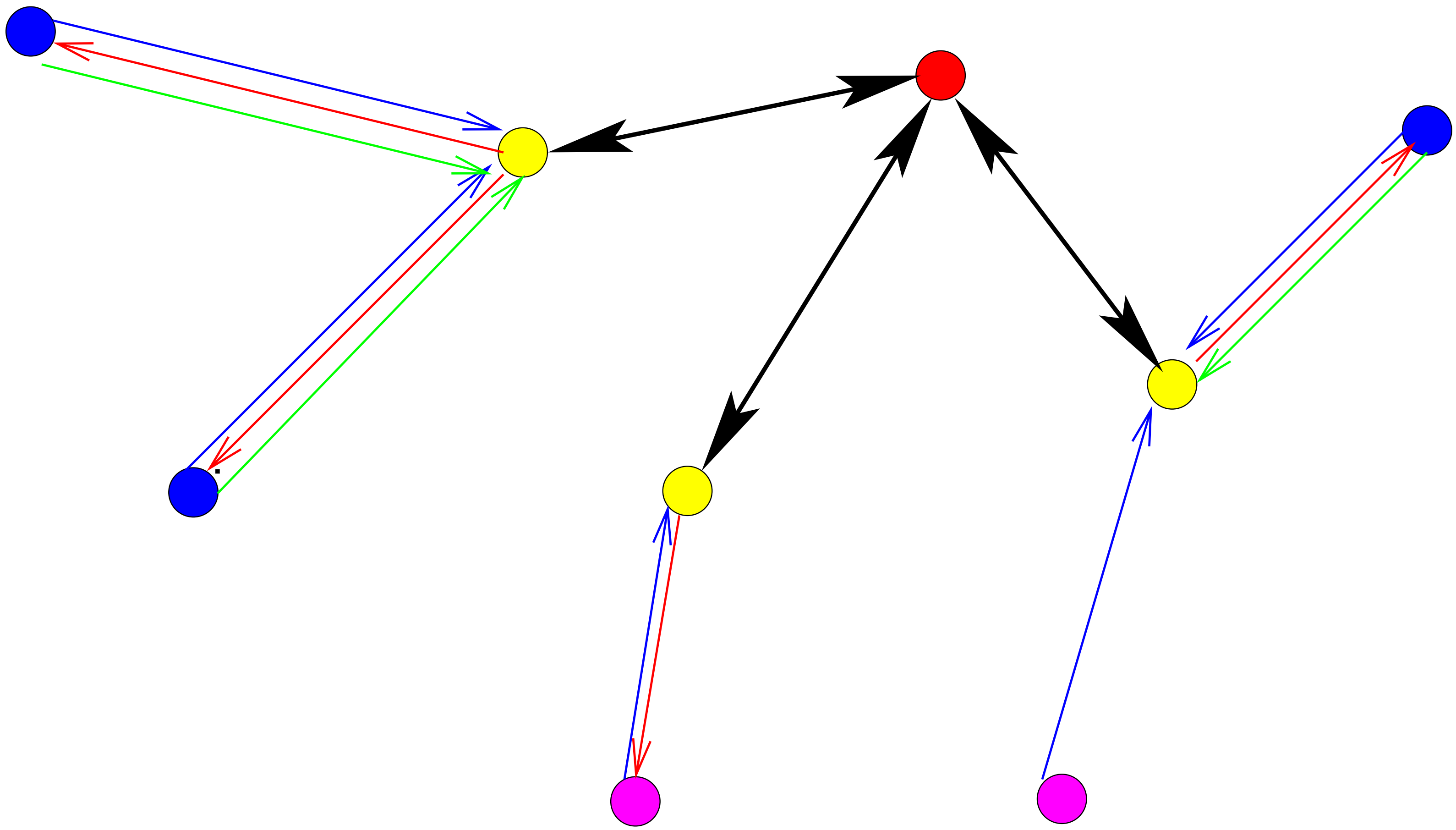


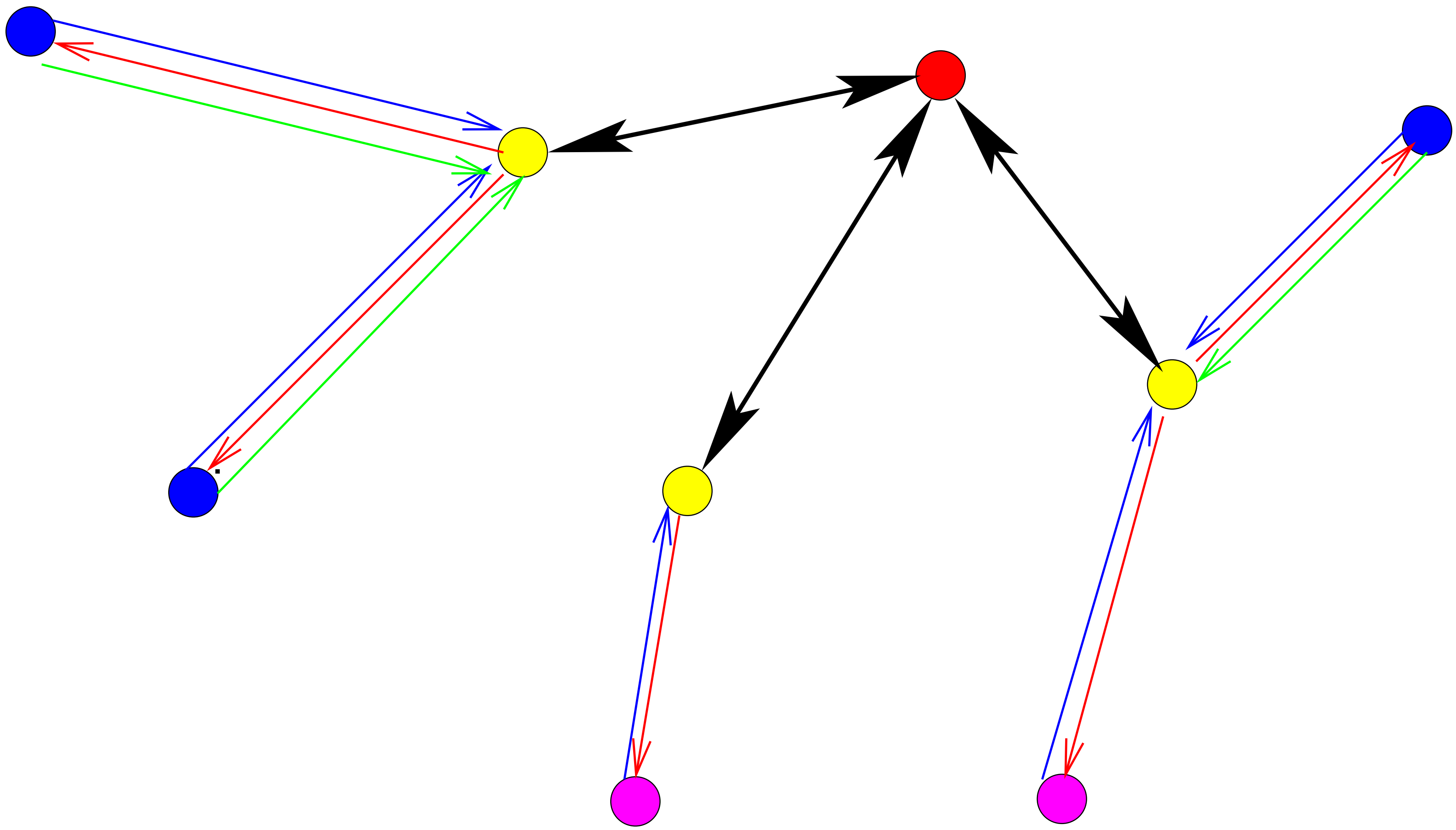


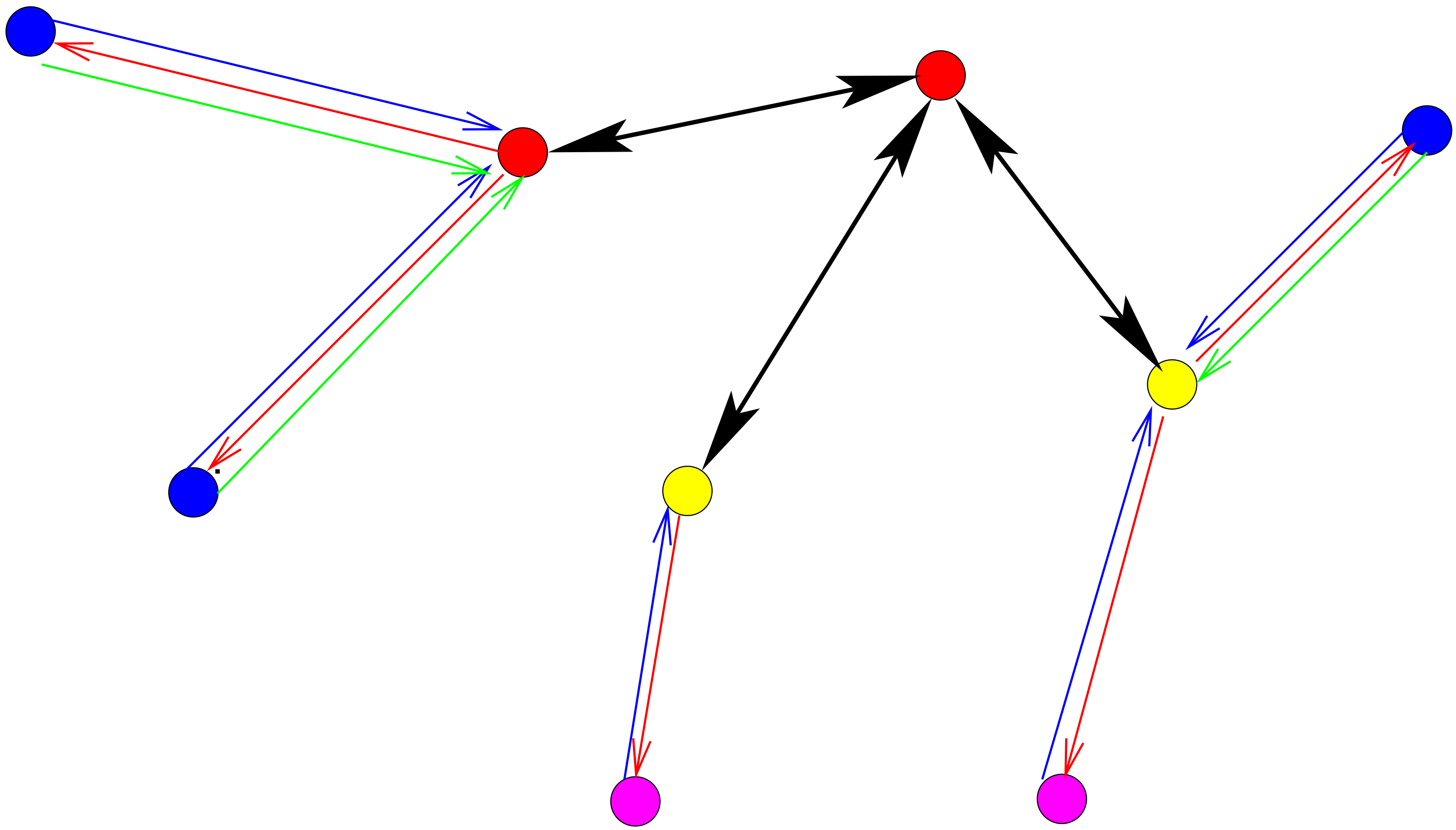


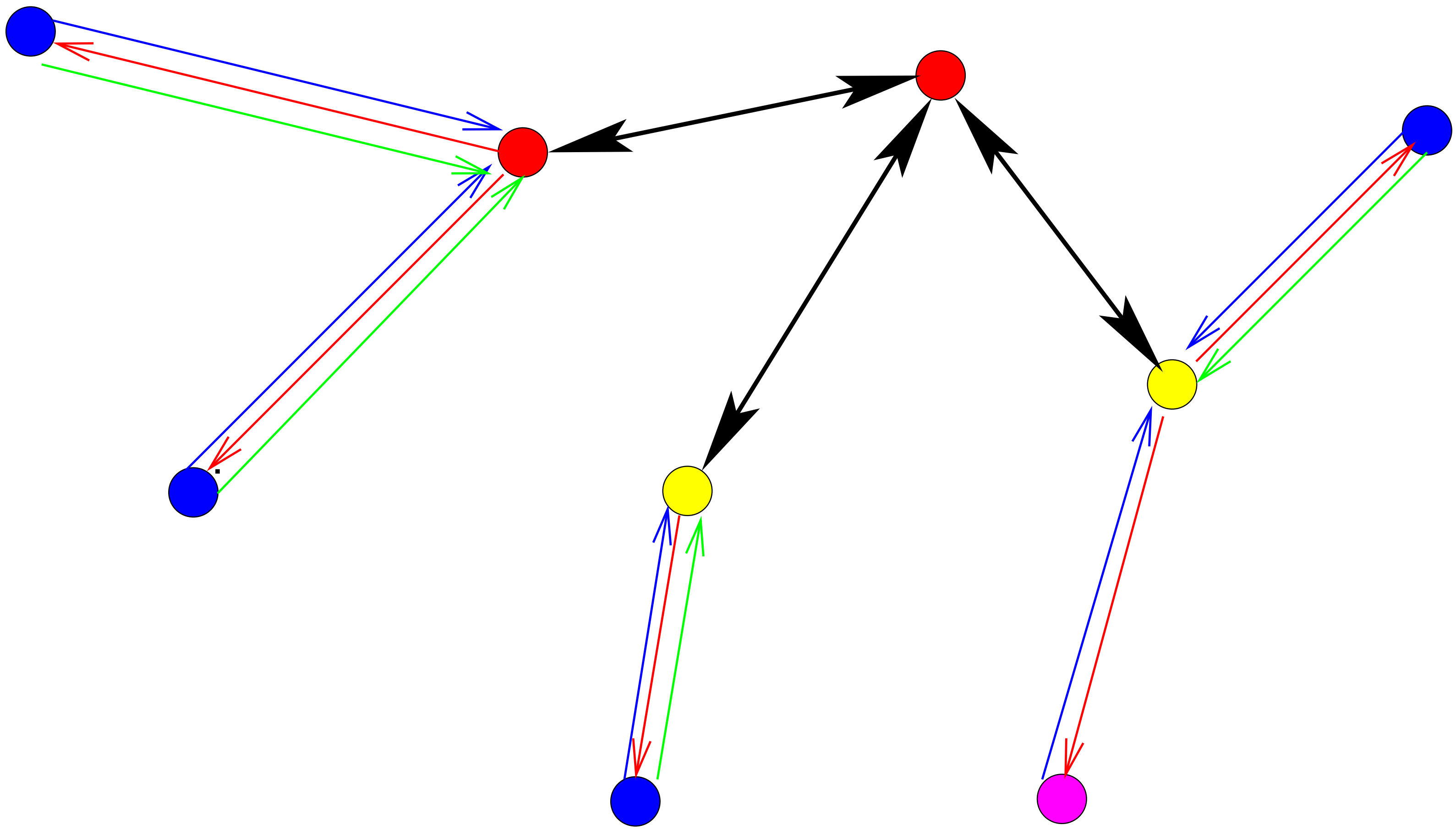


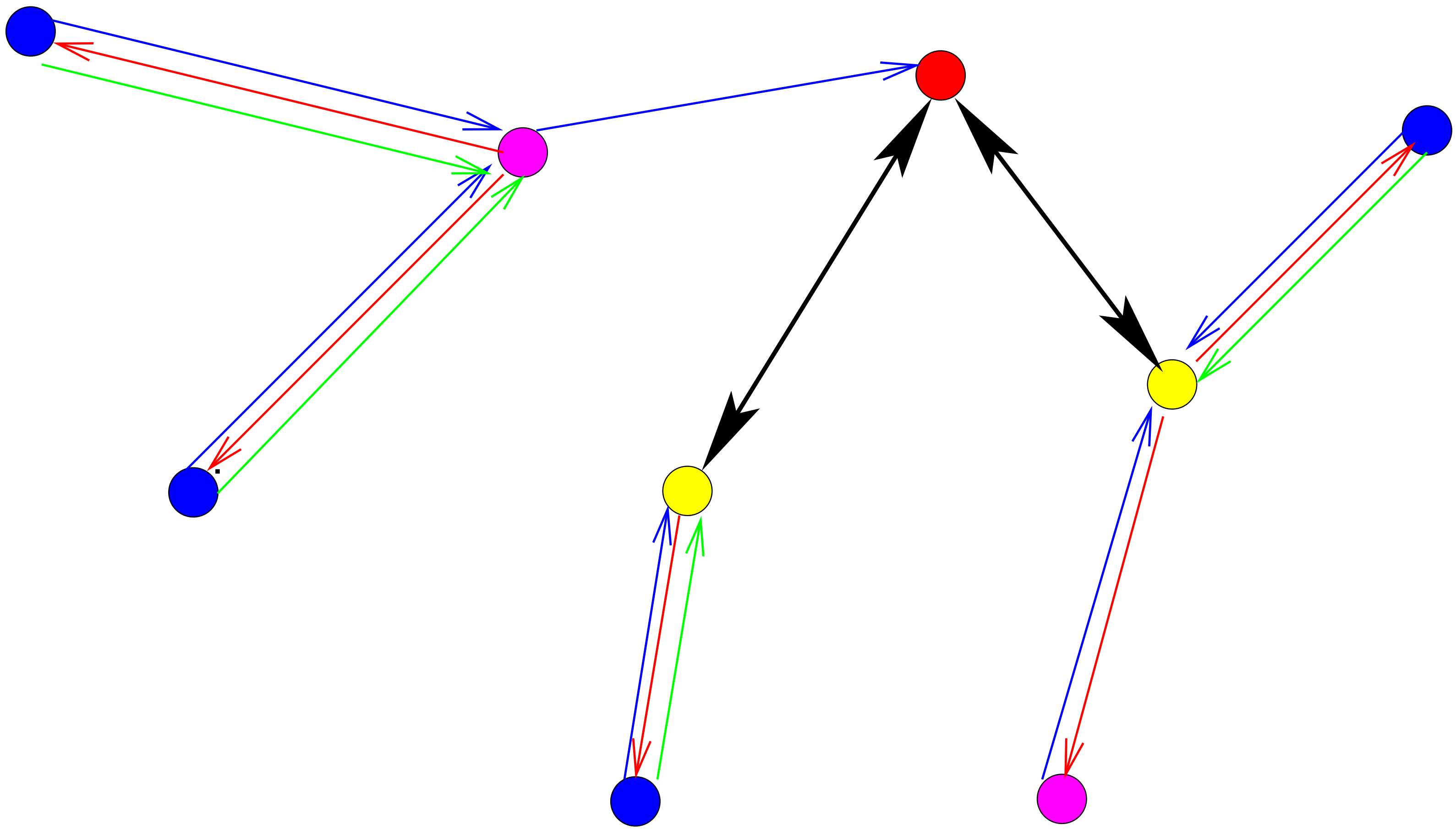


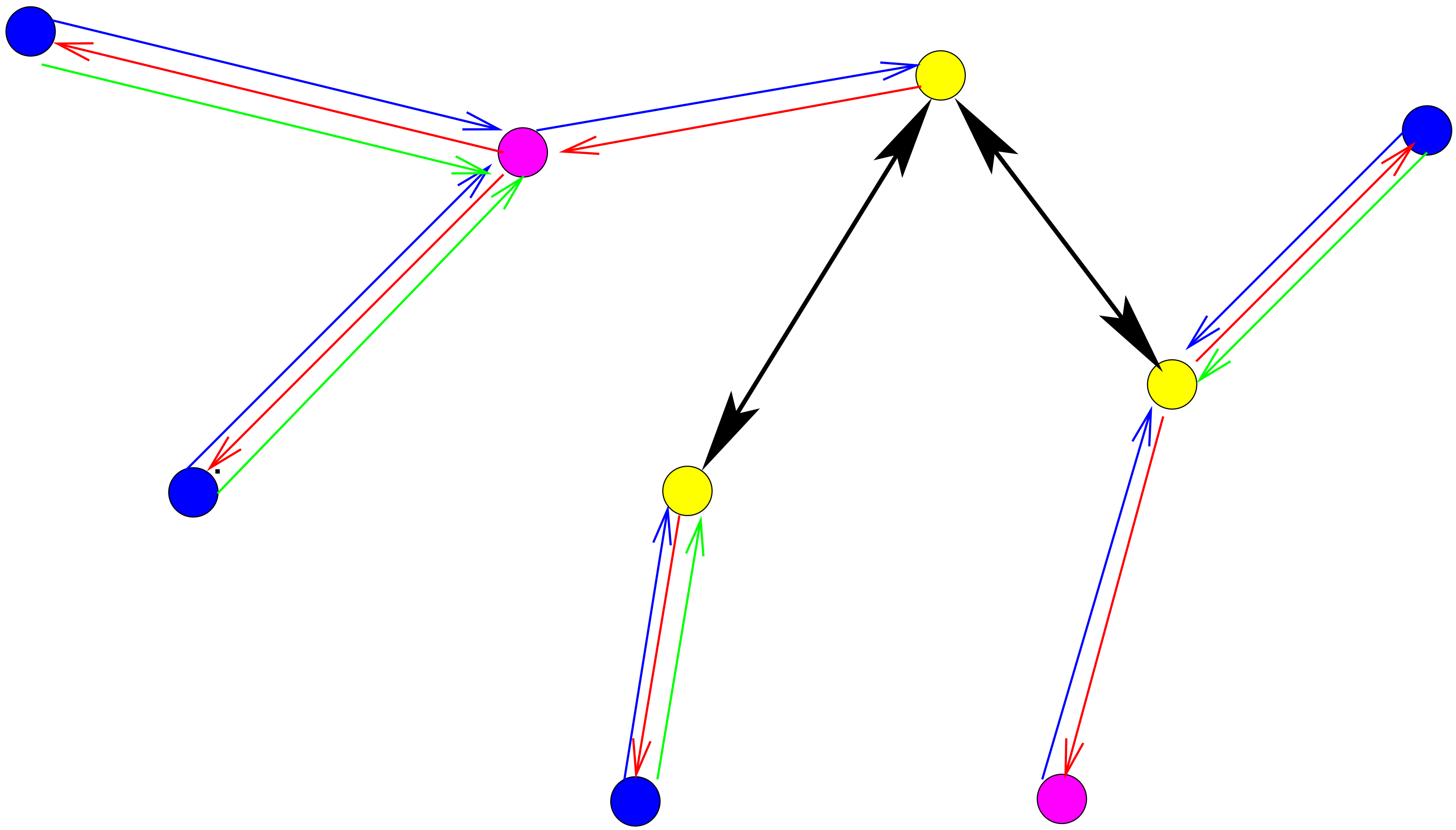


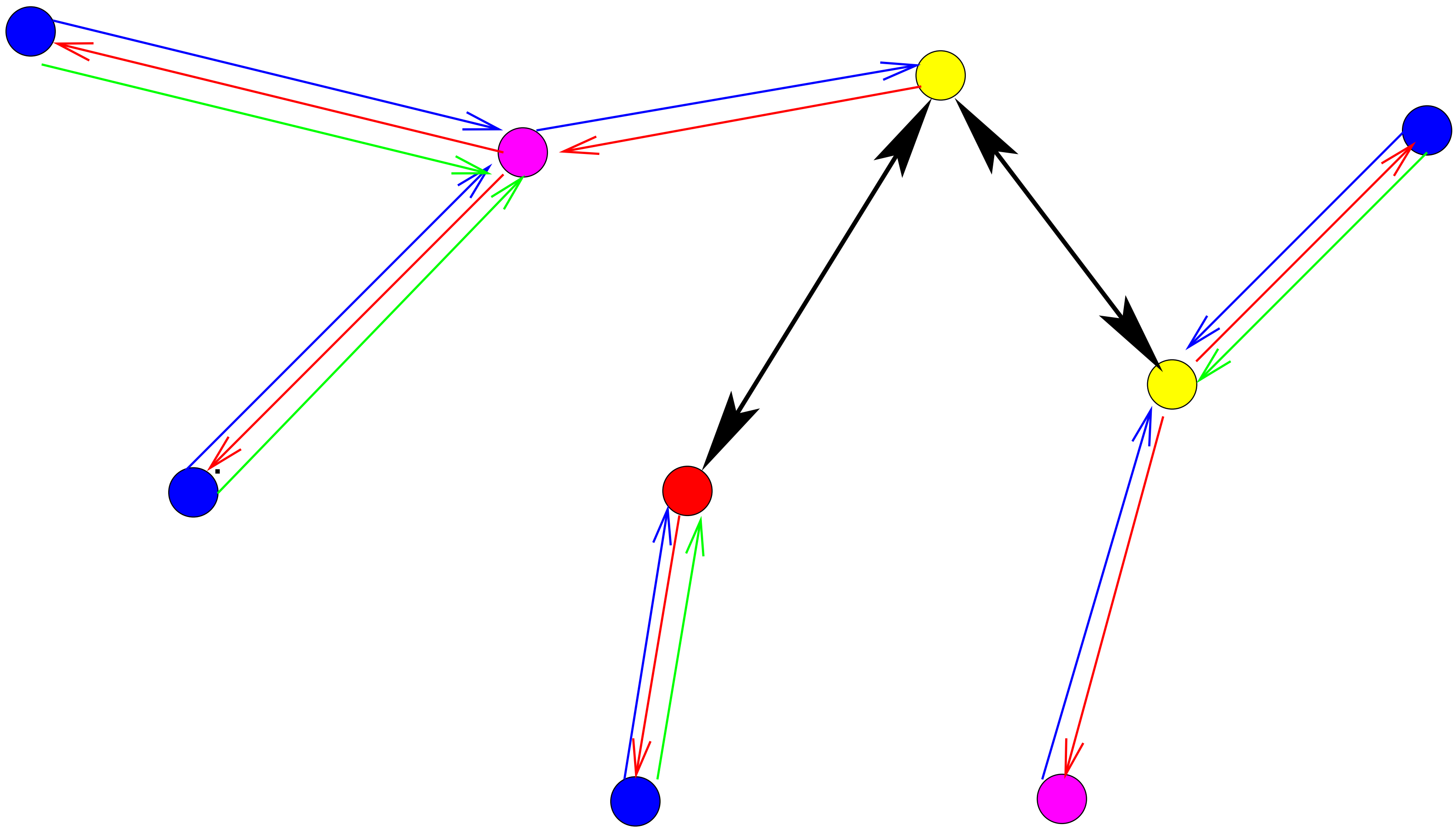


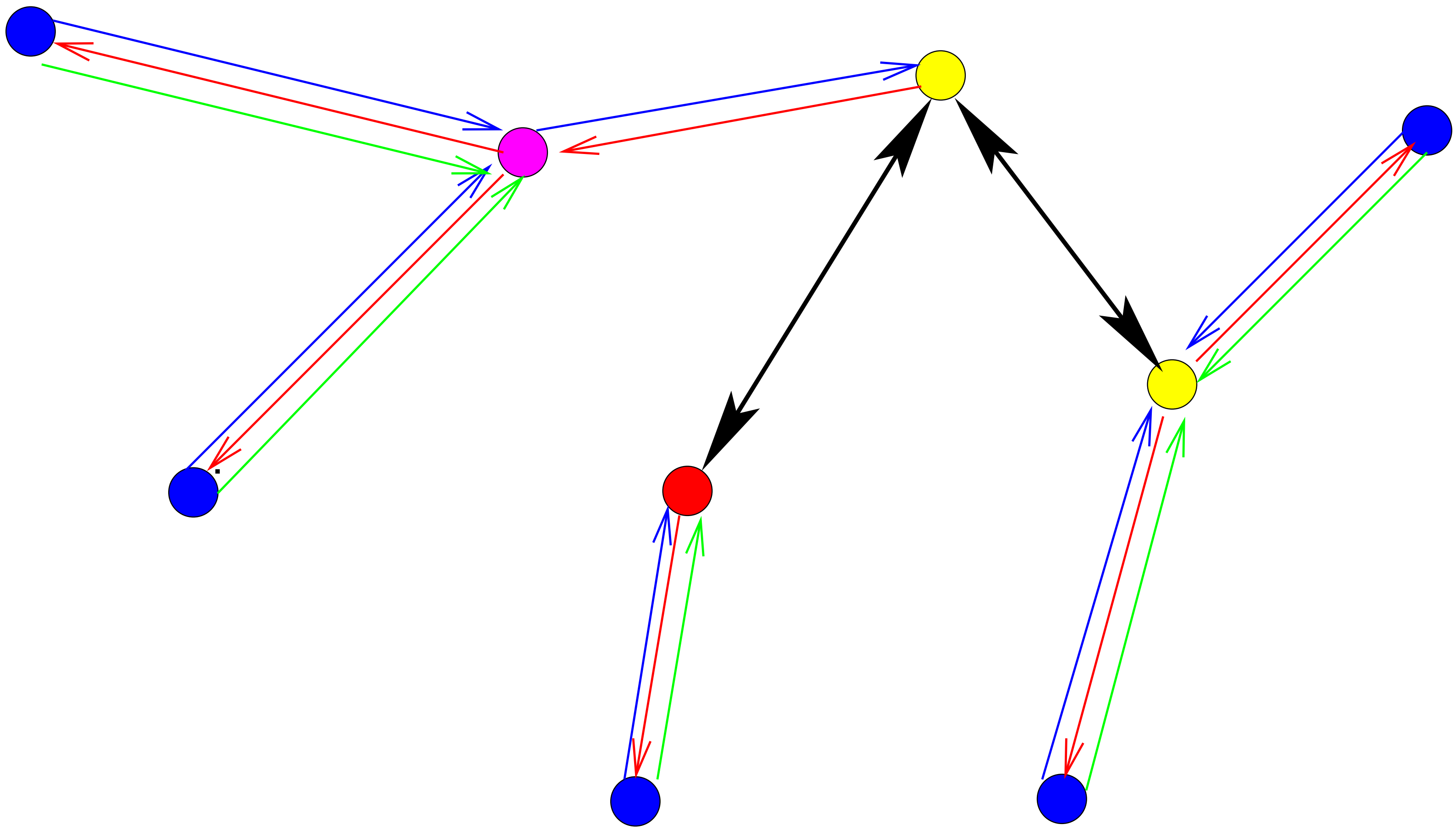


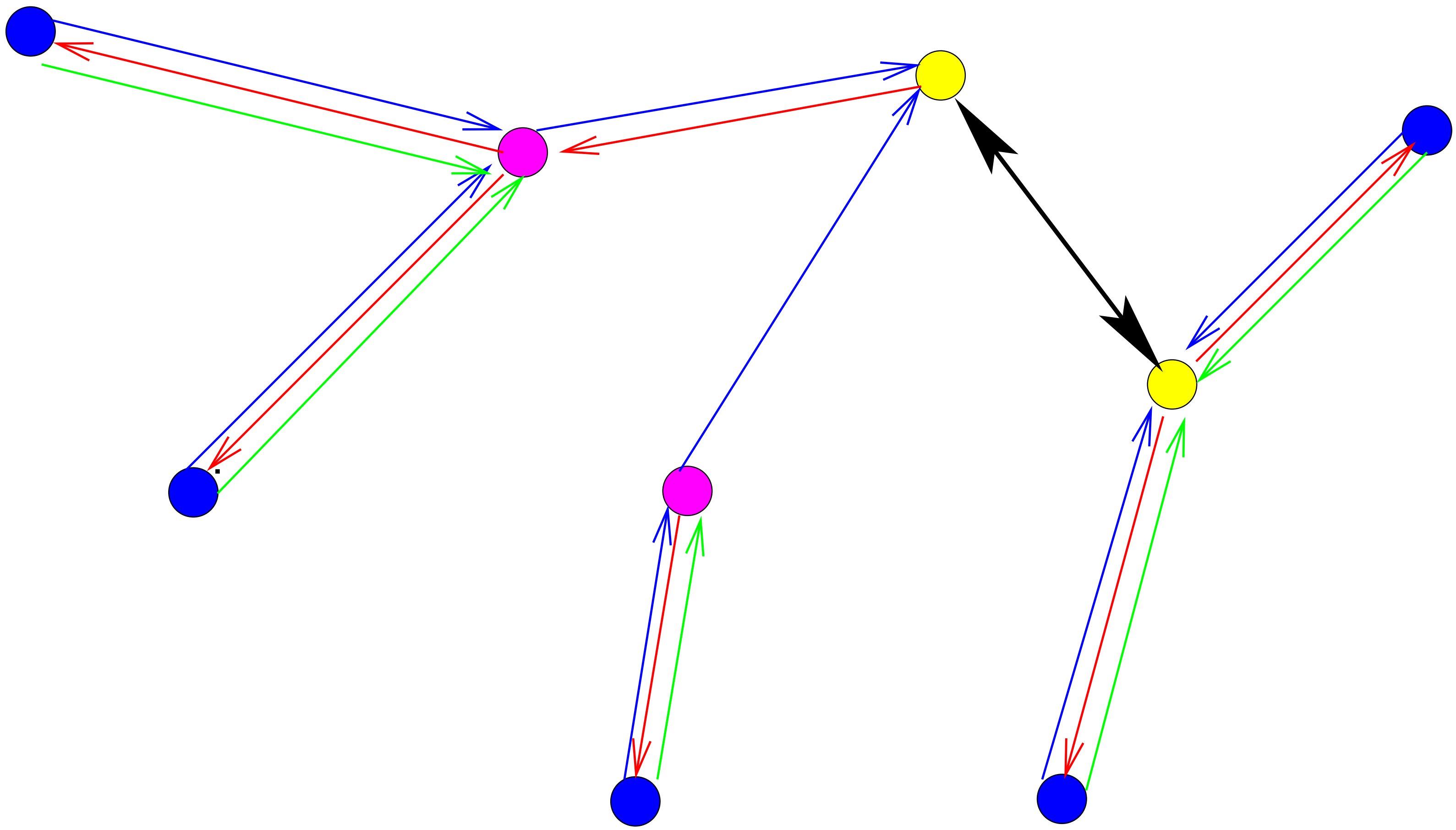


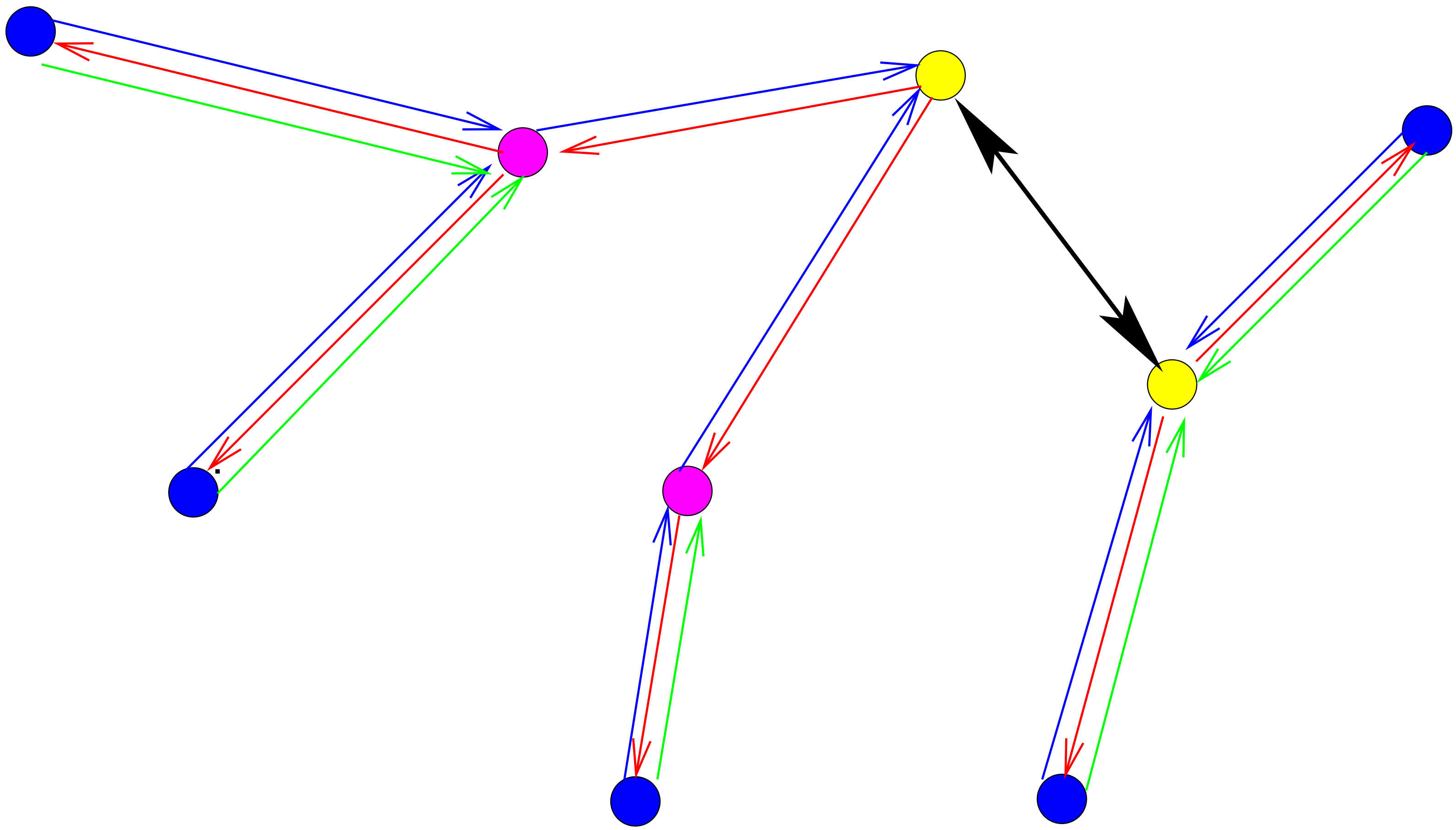


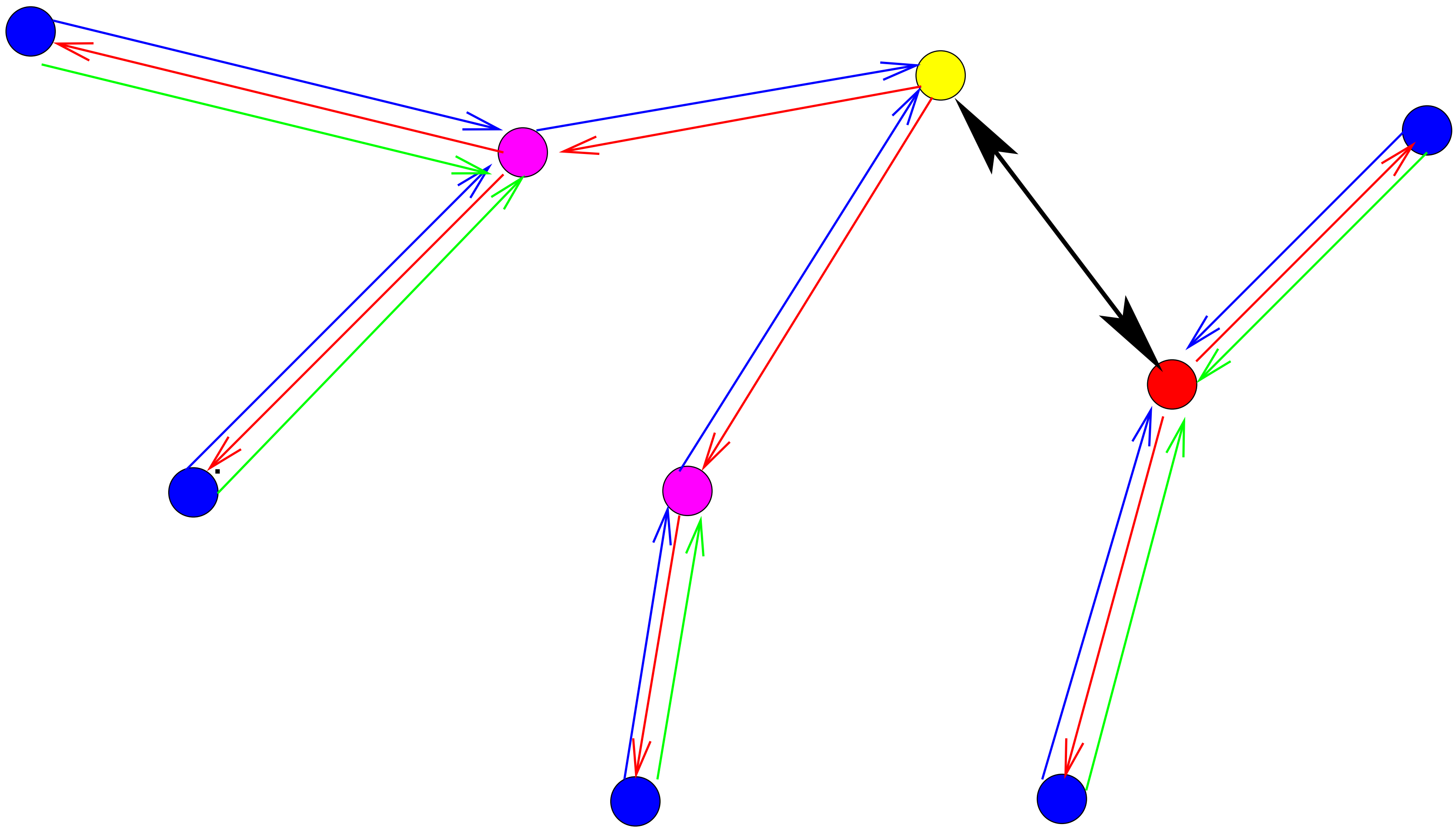


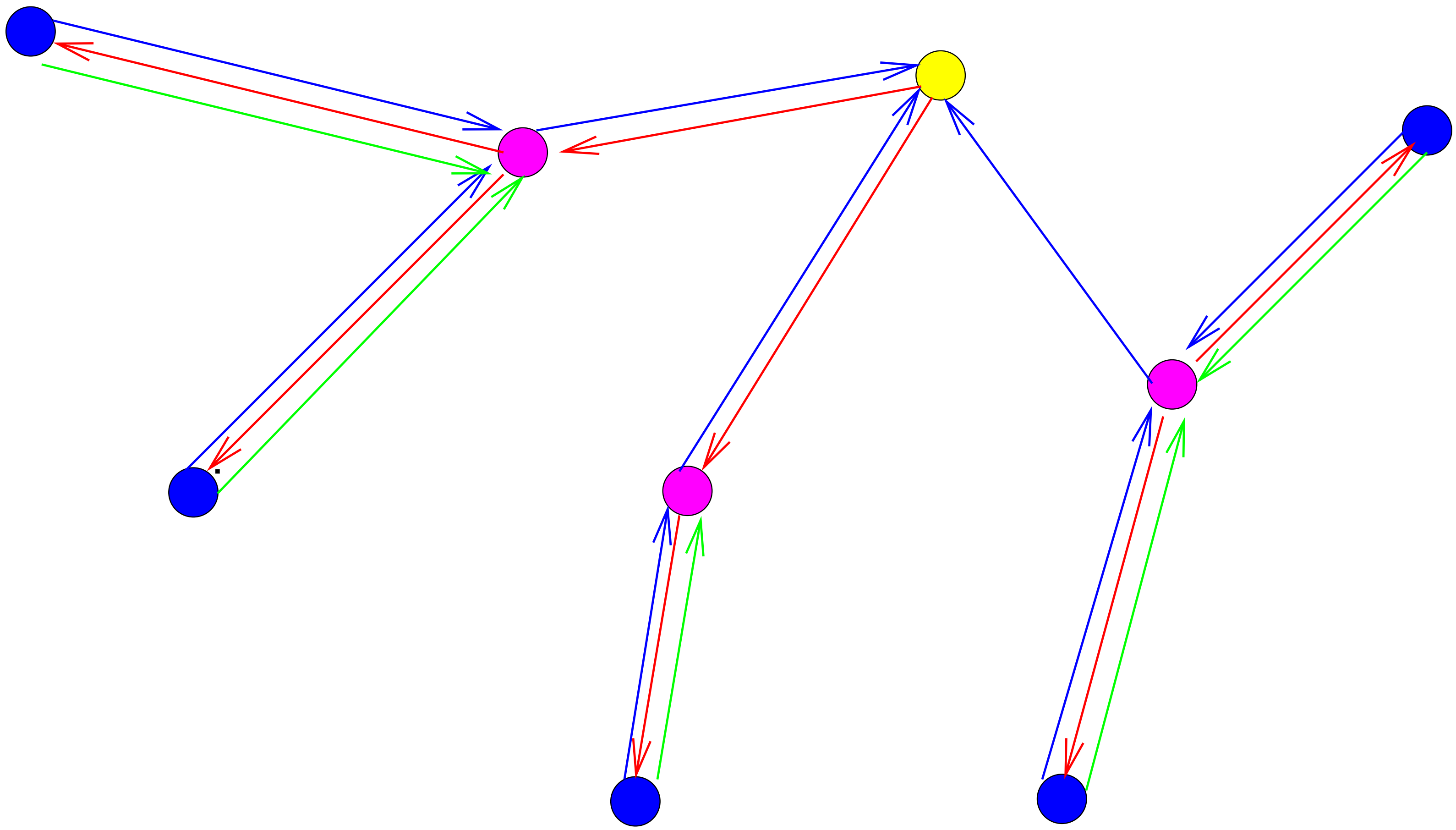


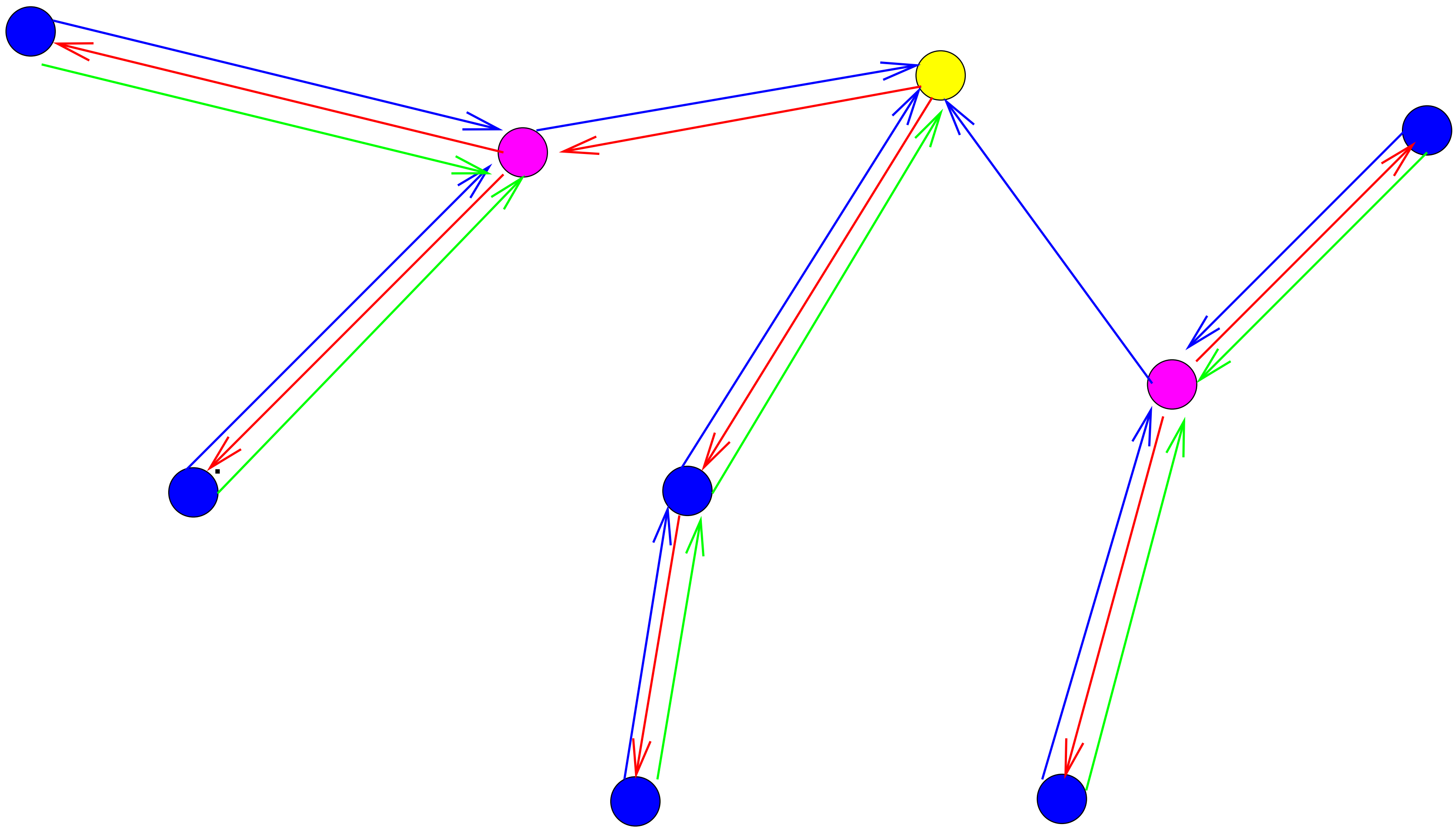


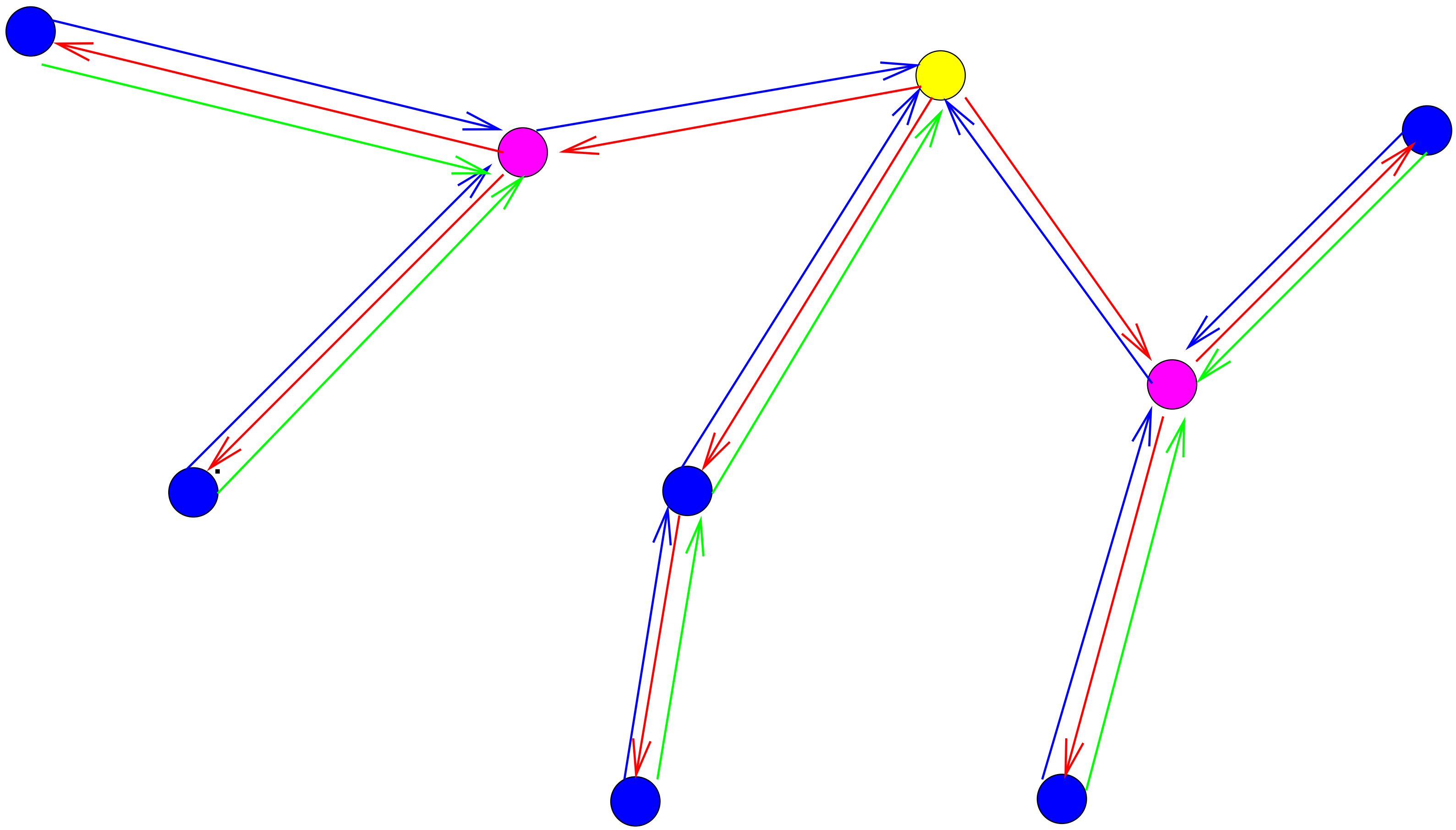


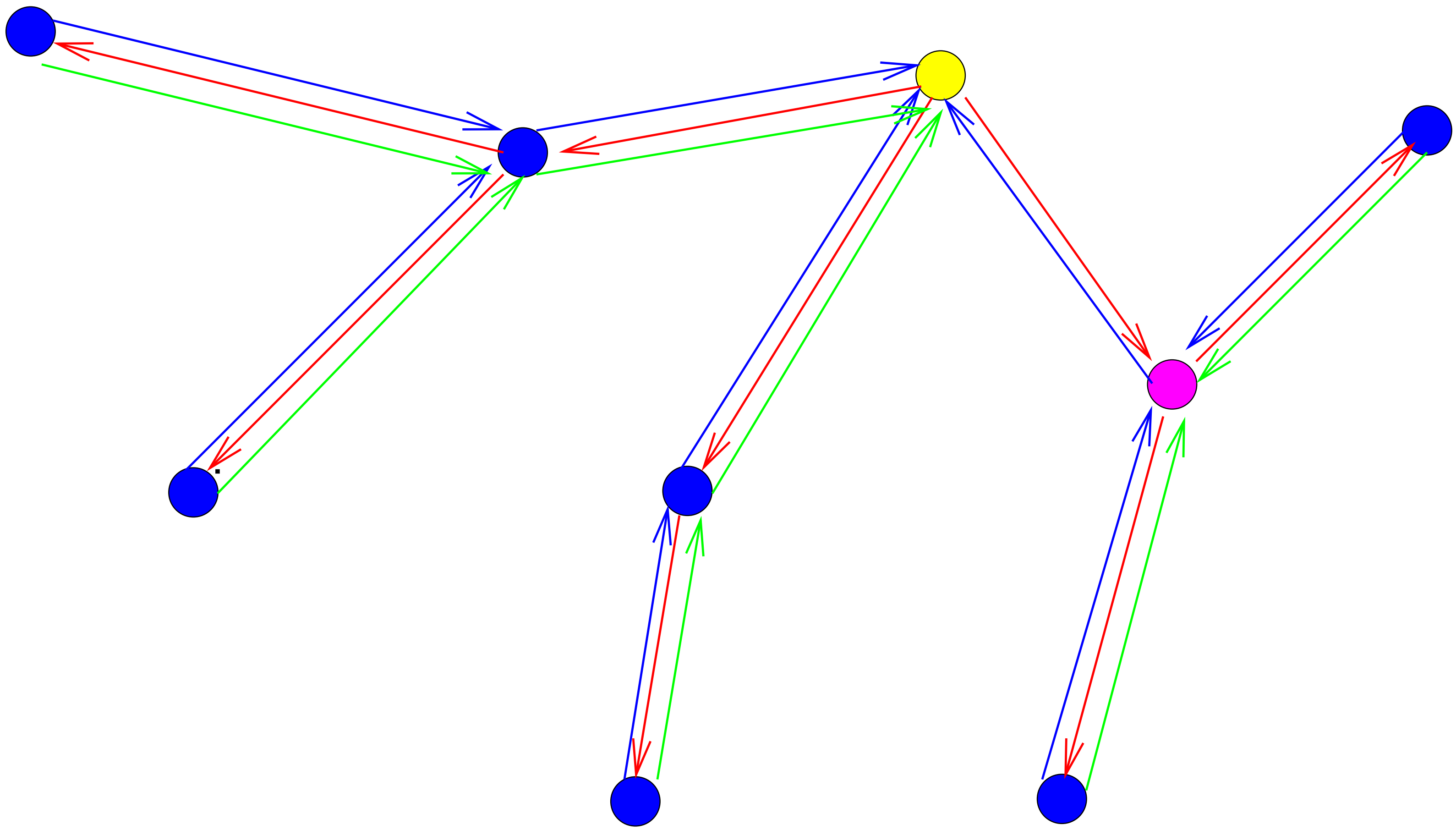


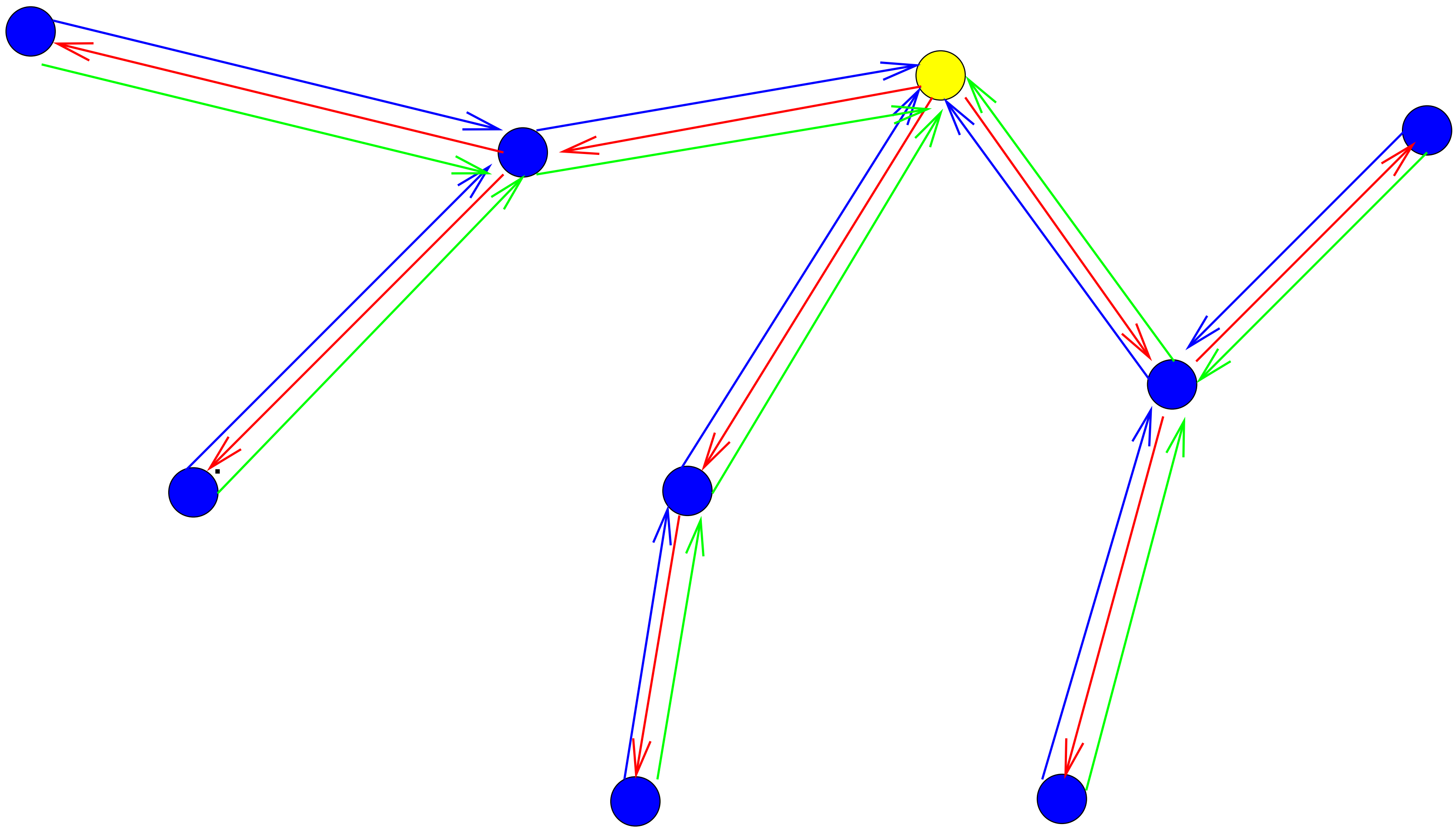


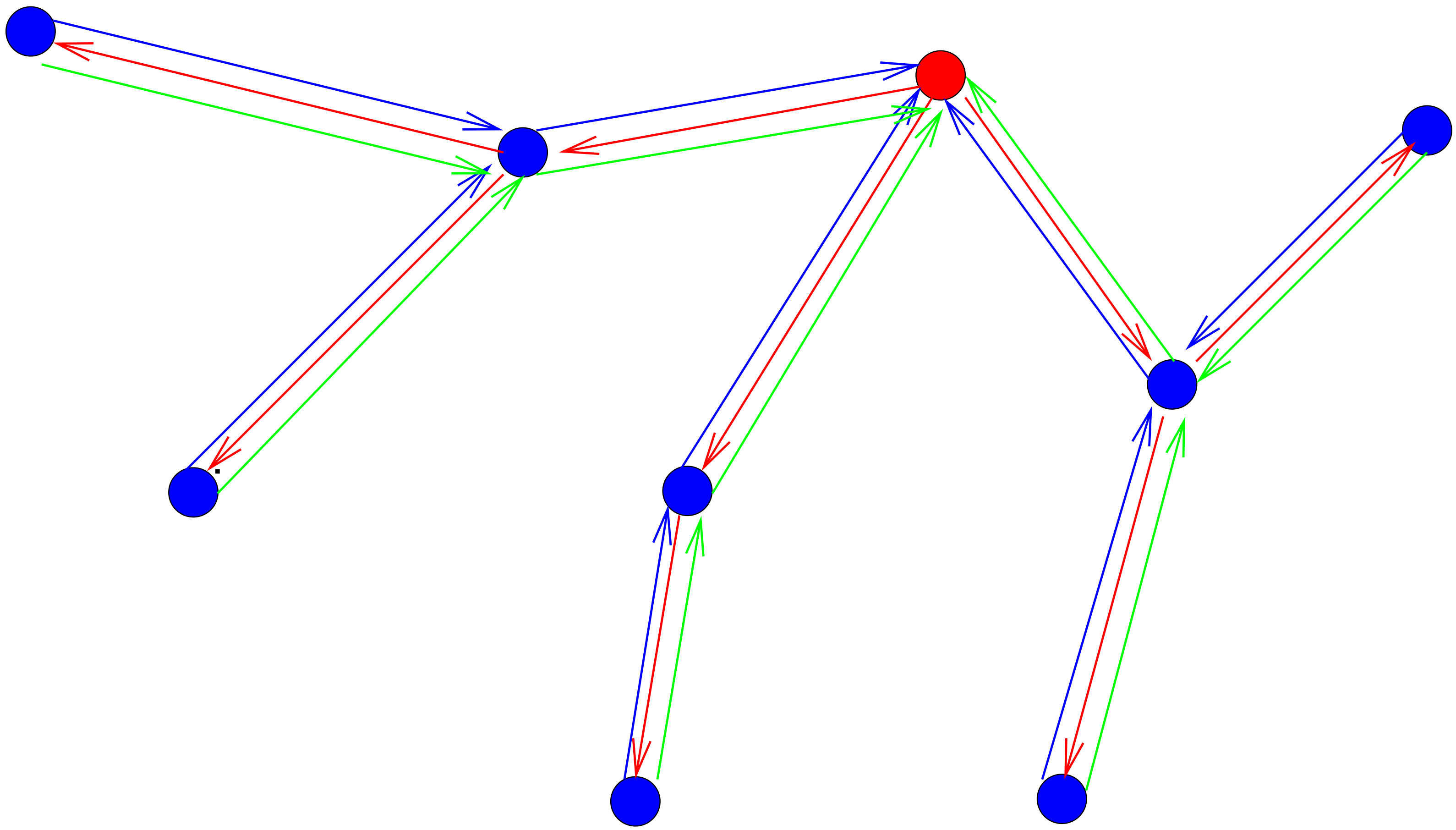












Invariant (1)

- Each node sends AT MOST one message
- Each node receives AT MOST one acknowledgment
- Each node sends AT MOST one confirmation

$$msg \in ND \rightarrowtail ND$$

$$ack \in ND \rightarrowtail ND$$

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

Node x sends a message to node y

send_msg $\widehat{=}$

any x, y **where**

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

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

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

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

$y, x \notin ack \;\; \wedge$

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

then

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

end

Node y sends an acknowledgement to node x

send_ack $\widehat{=}$

any x, y **where**

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

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

then

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

end

Node x sends a confirmation to node y

progress $\widehat{=}$

any x, y **where**

$x, y \in ack \ \wedge$

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

then

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

end

Invariant (2)

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

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

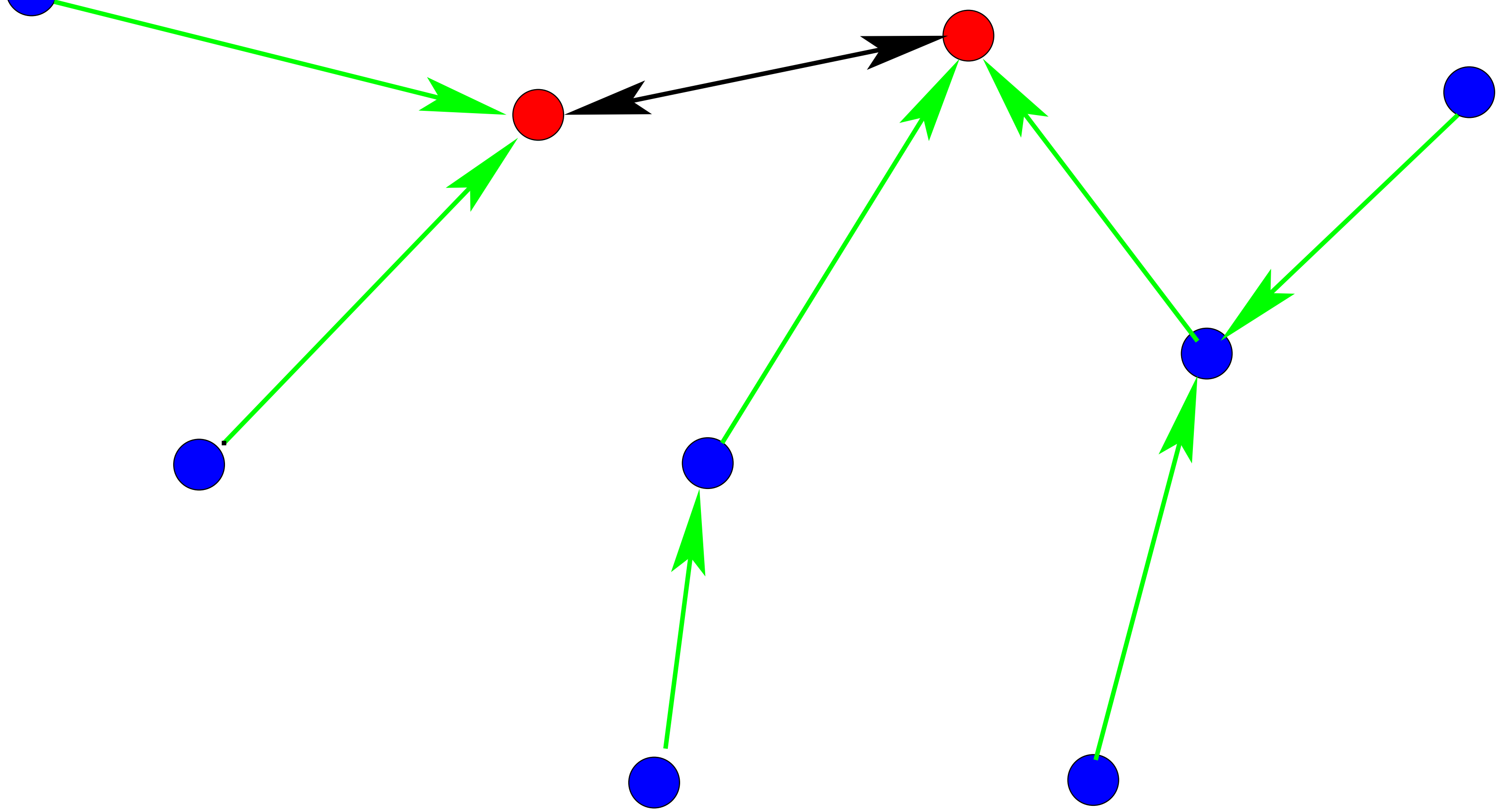
Second Refinement: The problem of contention

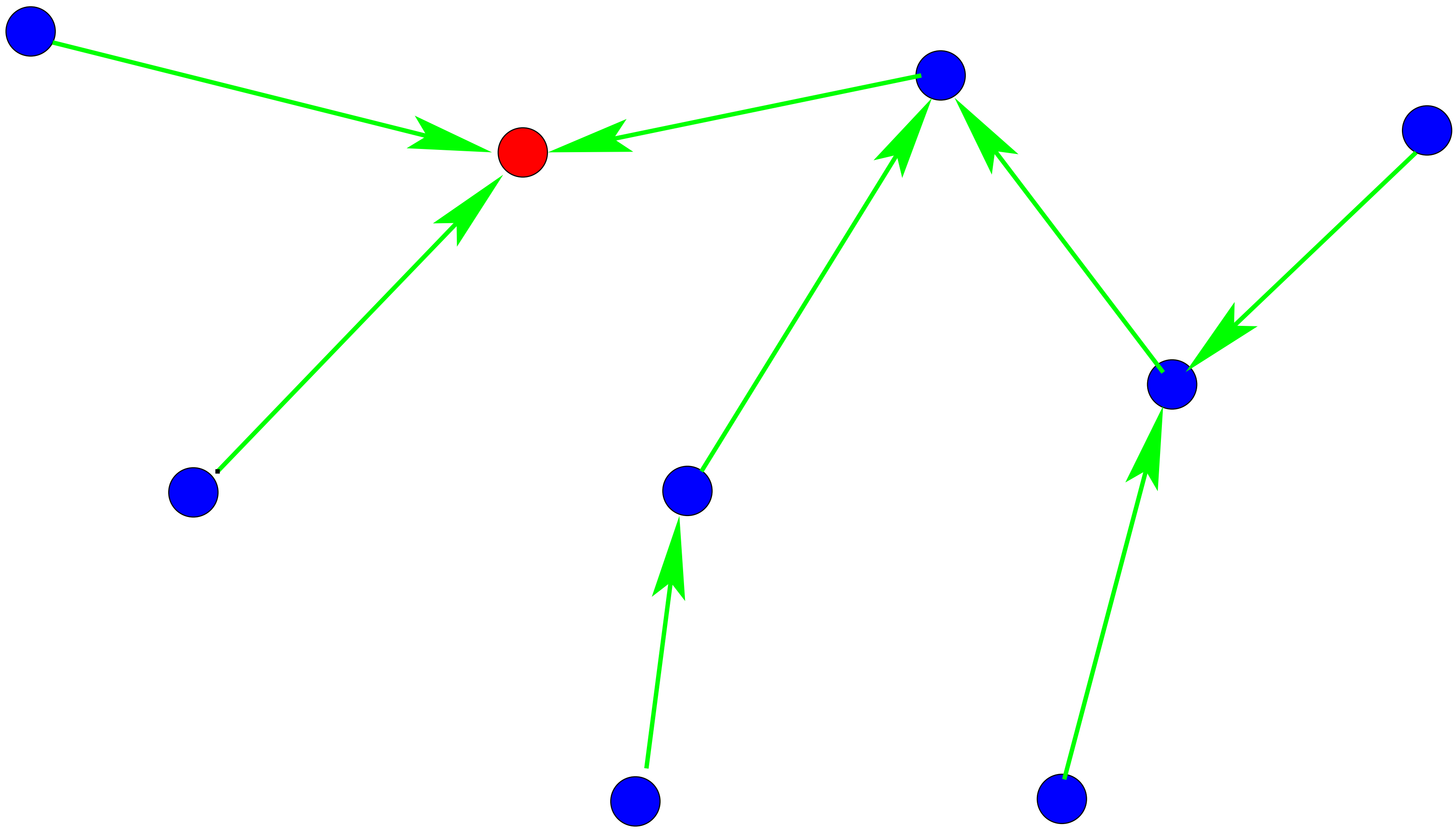
- Explaining the problem

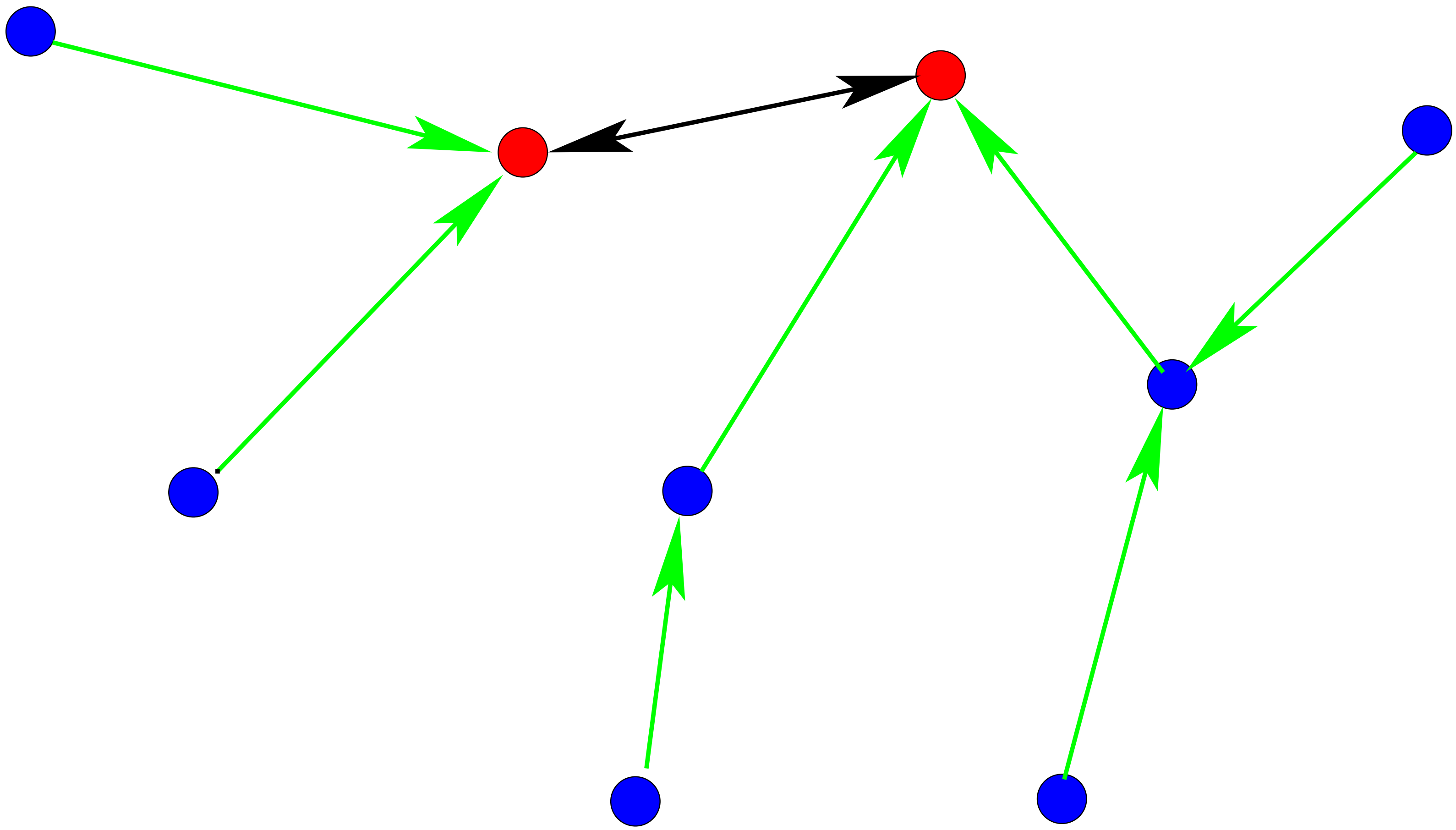
- Proposing a partial solution

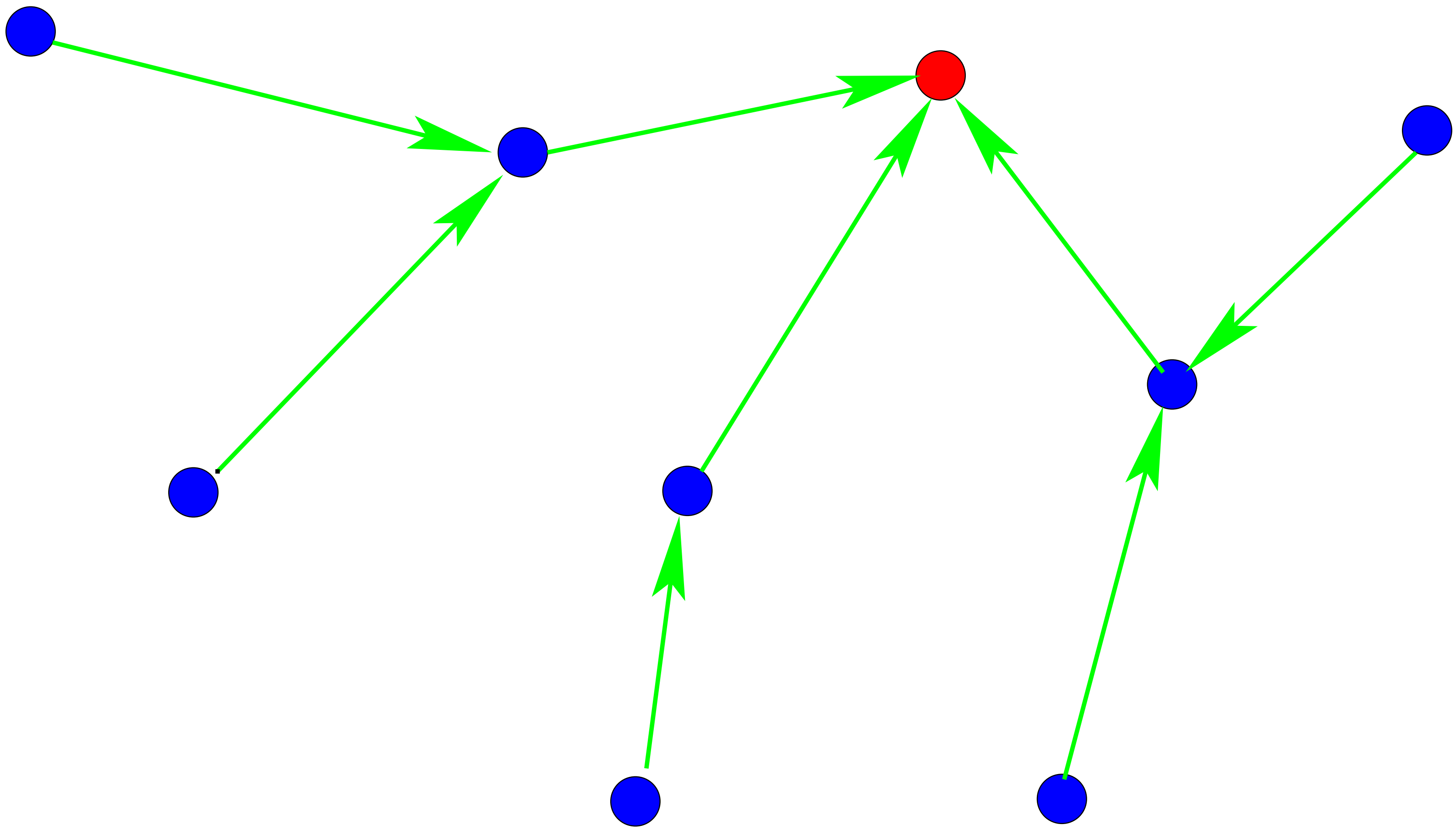
- Towards a better treatment

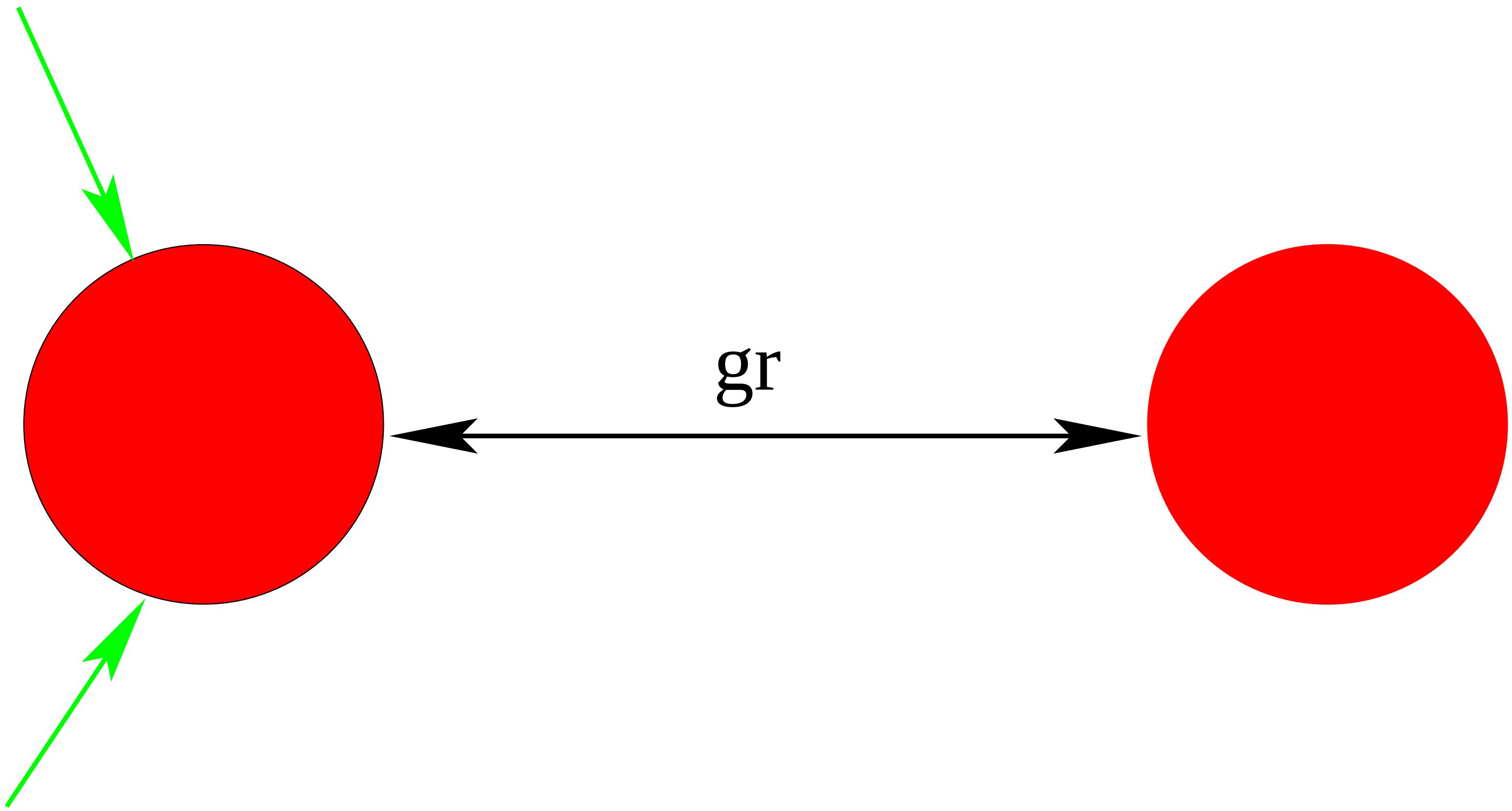
- Back to the local animation



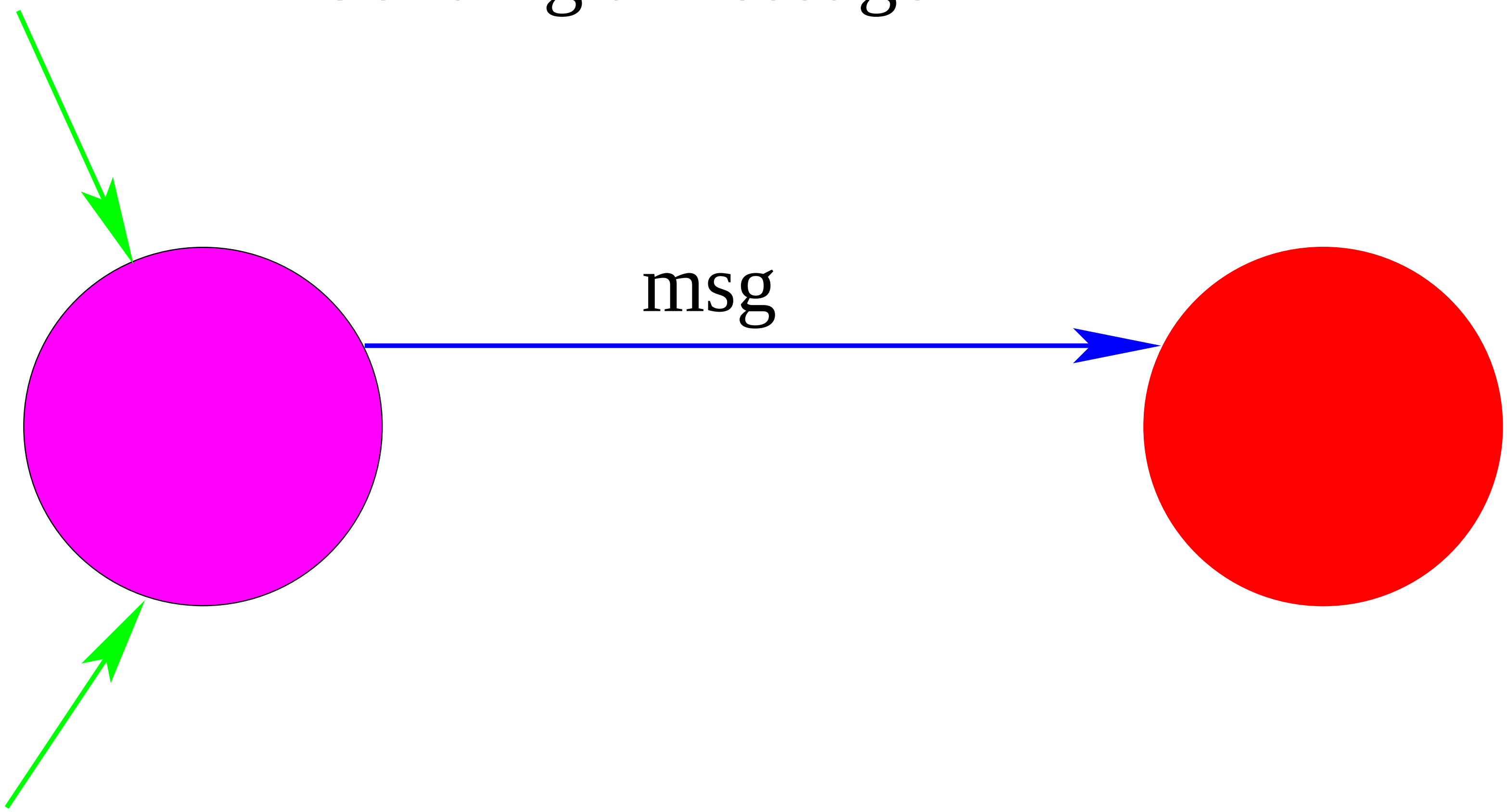




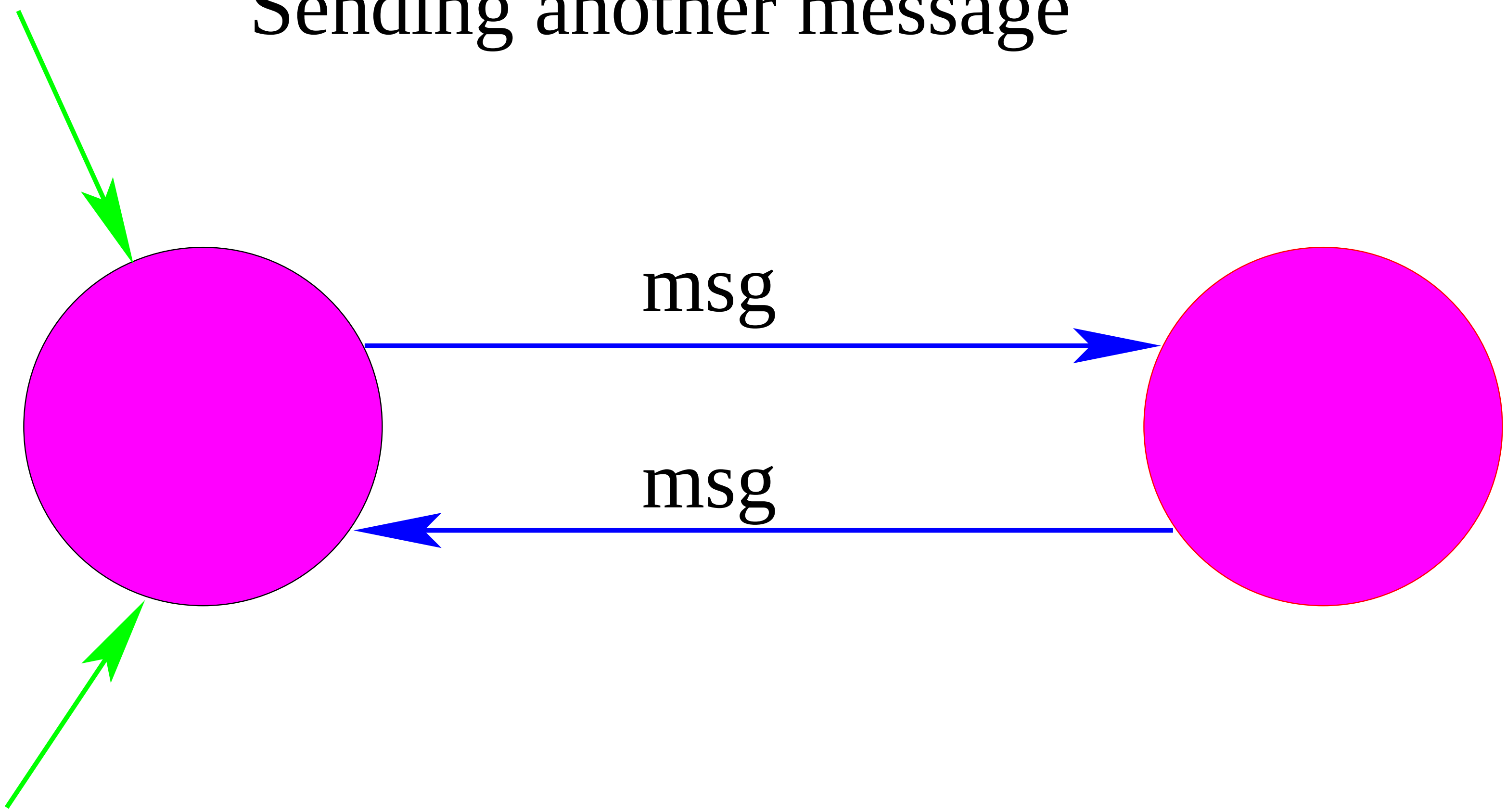



gr

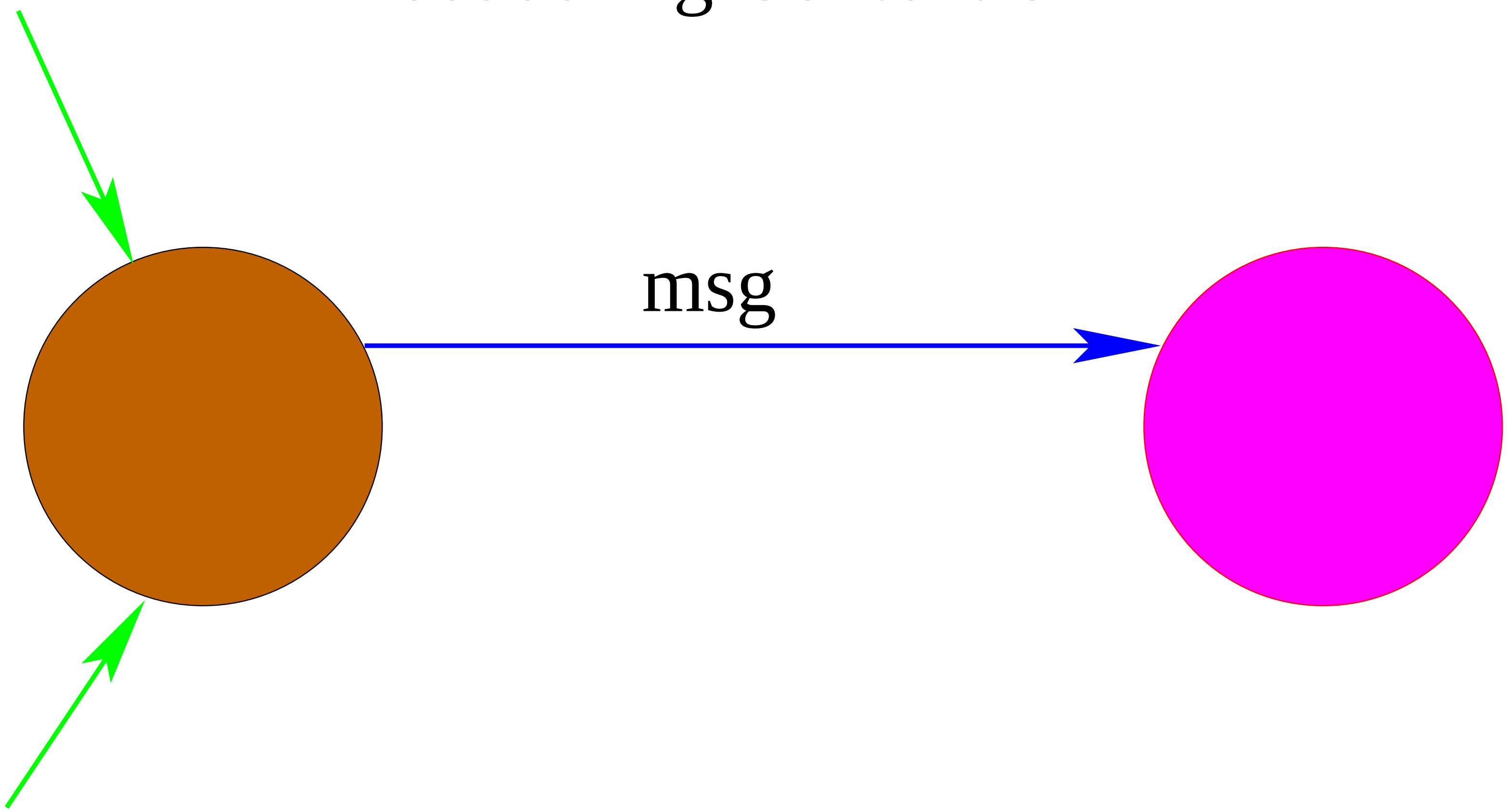
Sending a message



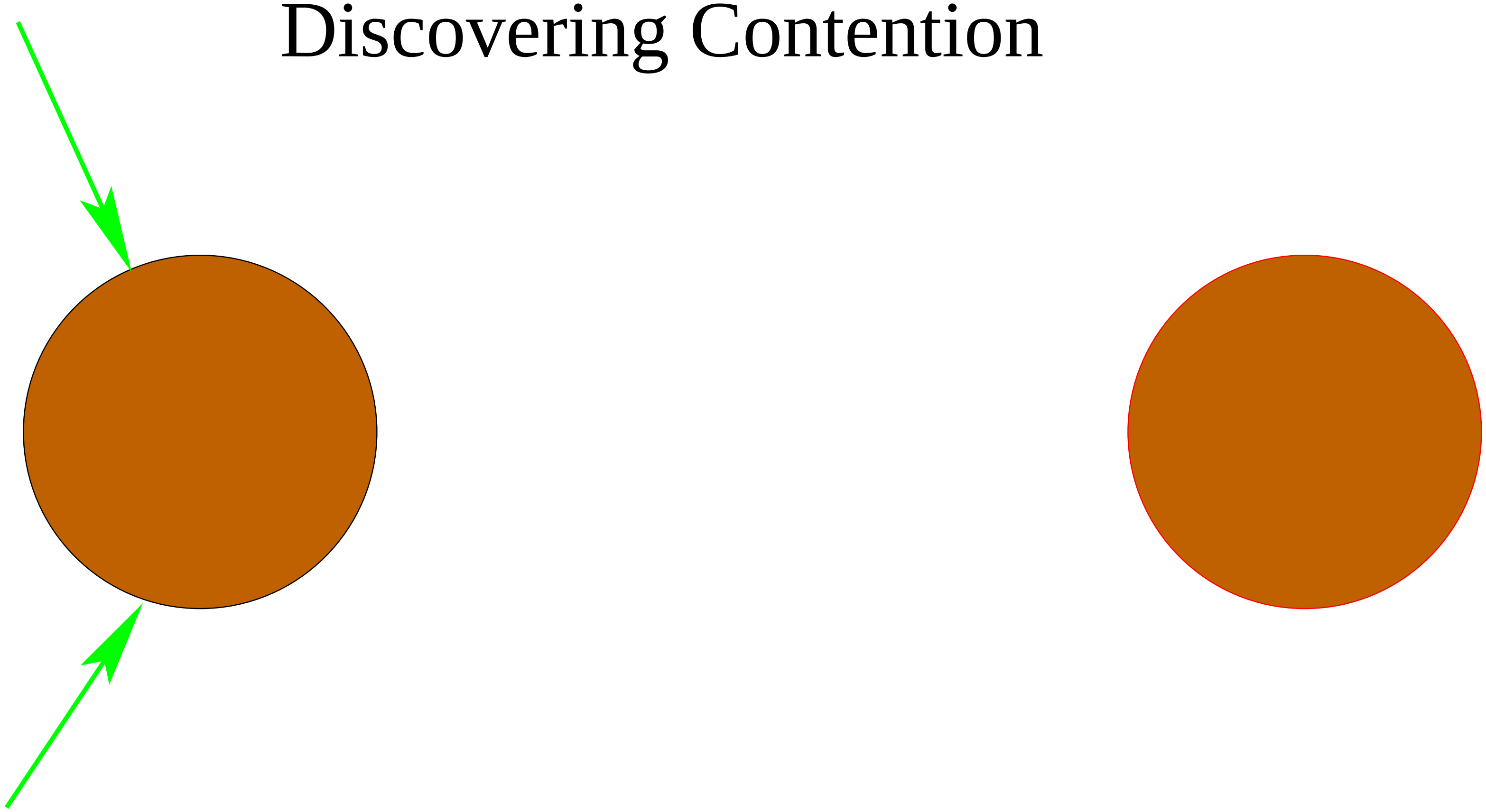
Sending another message



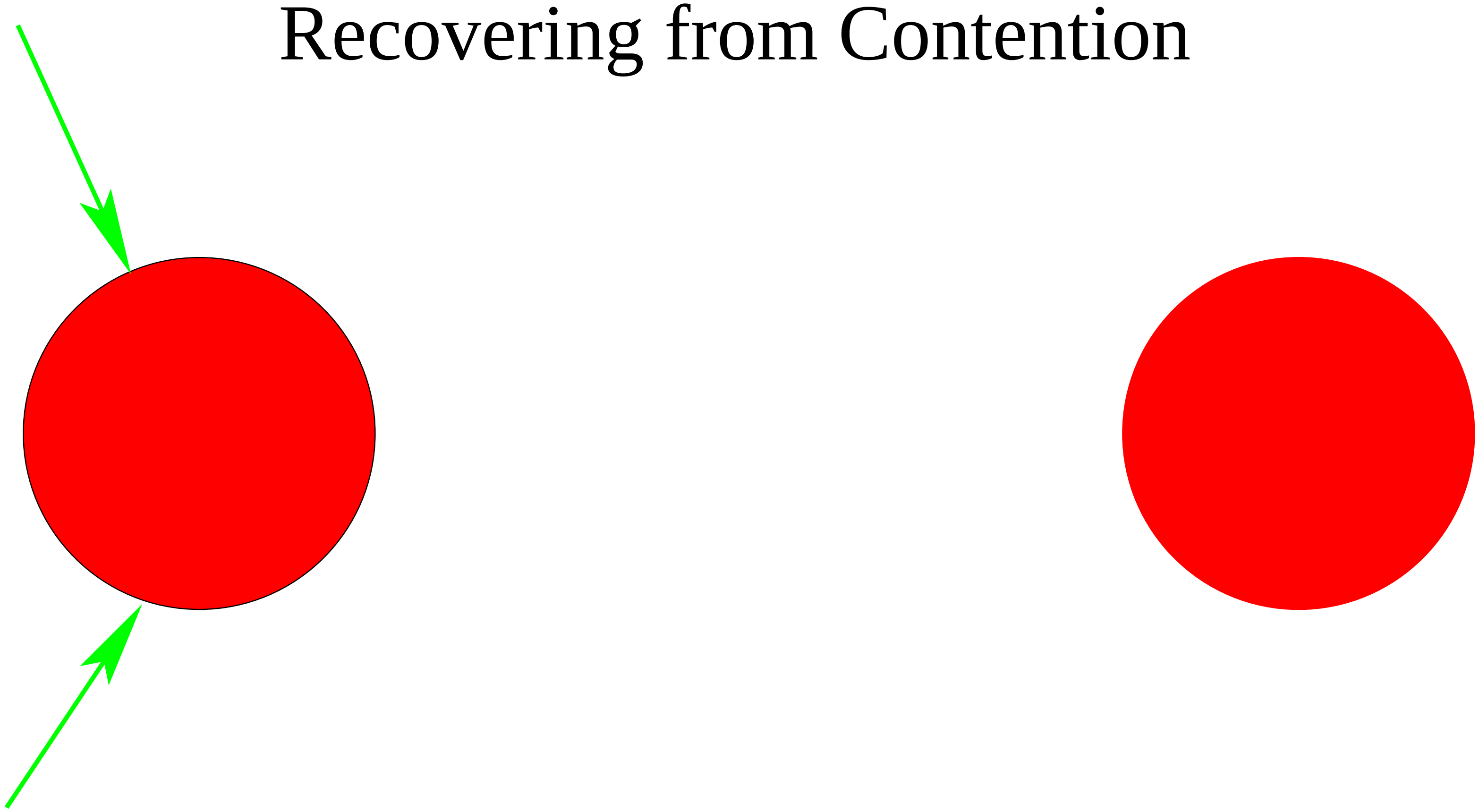
Discovering Contention



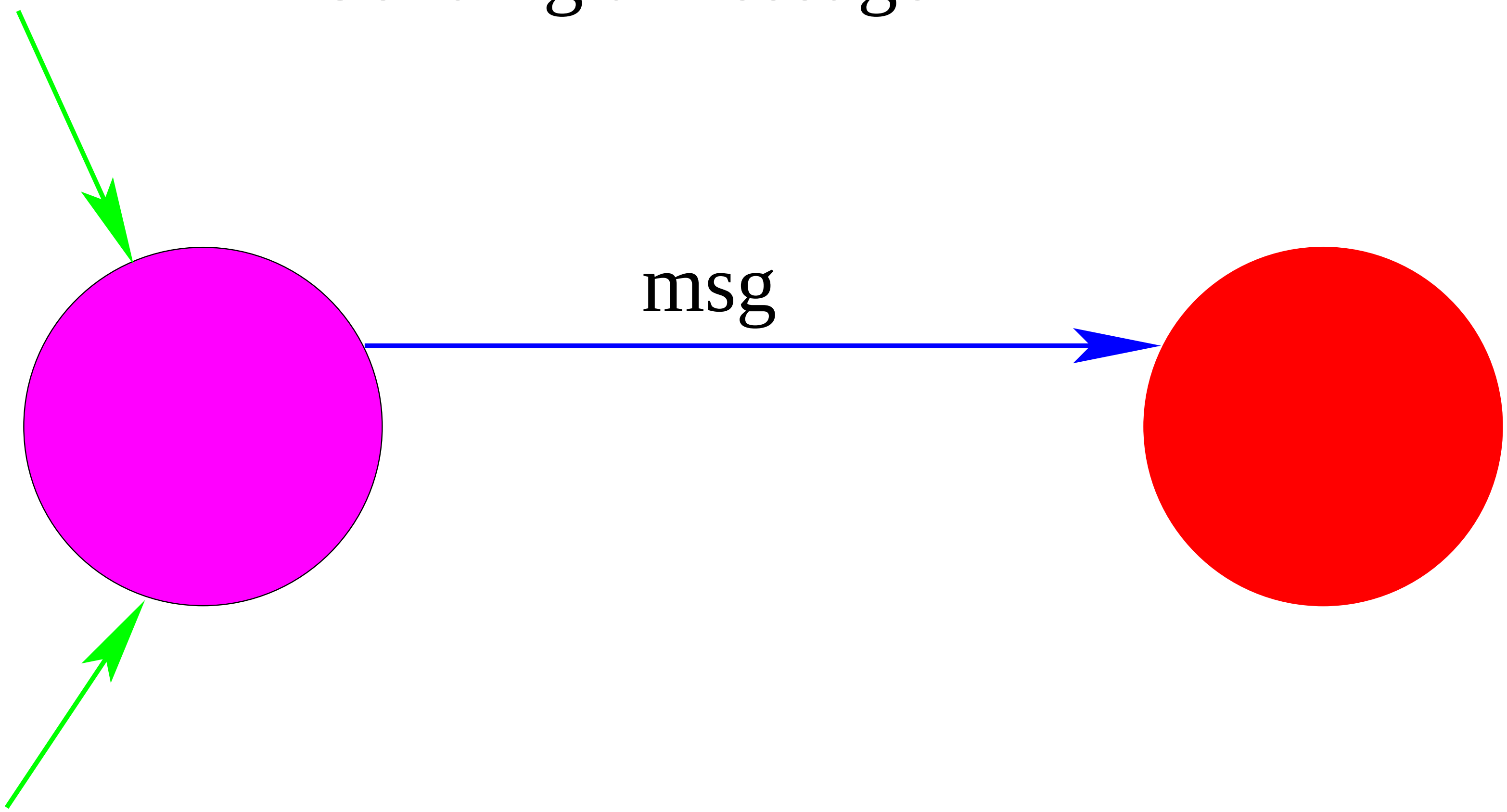
Discovering Contention



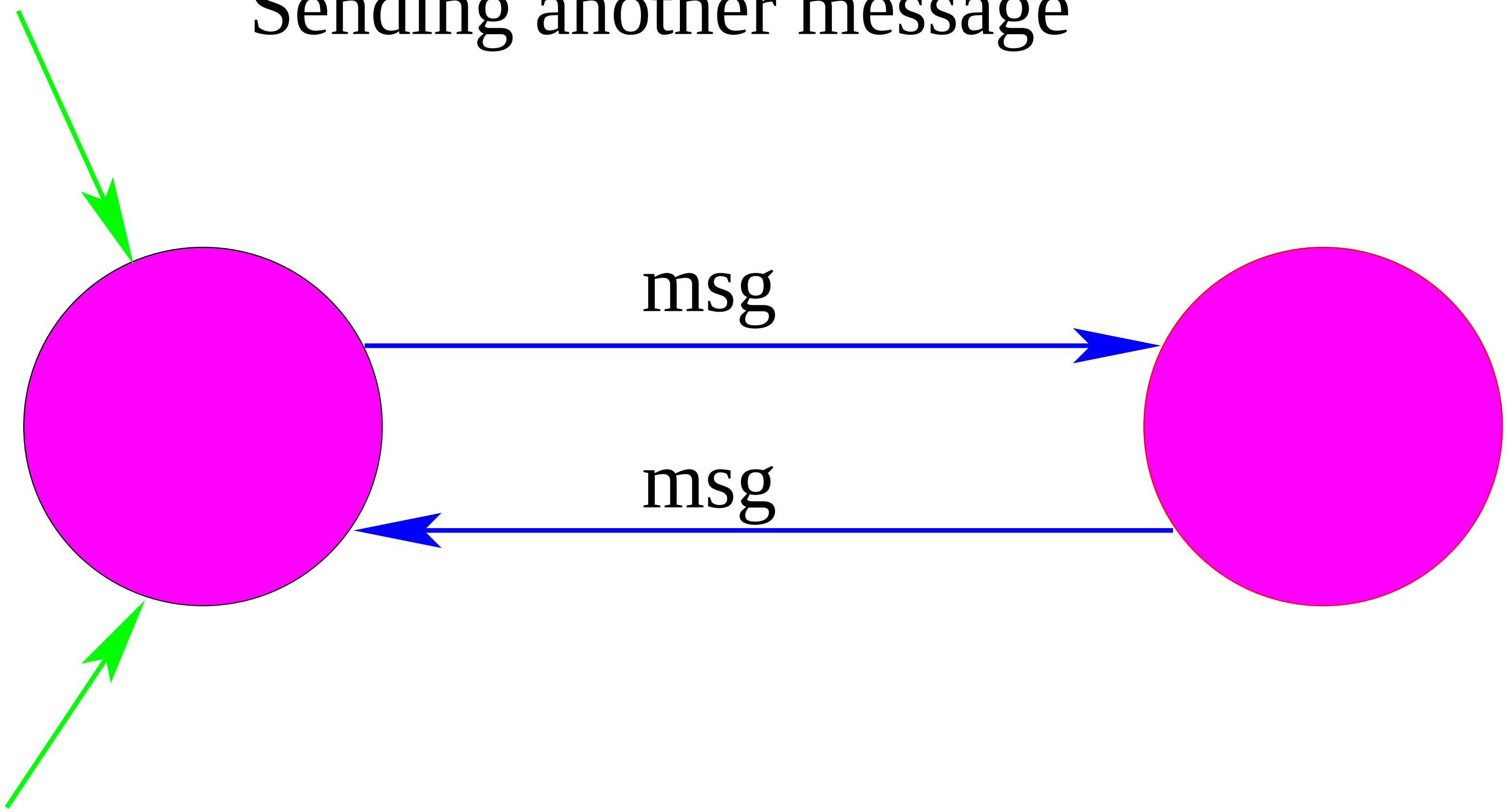
Recovering from Contention



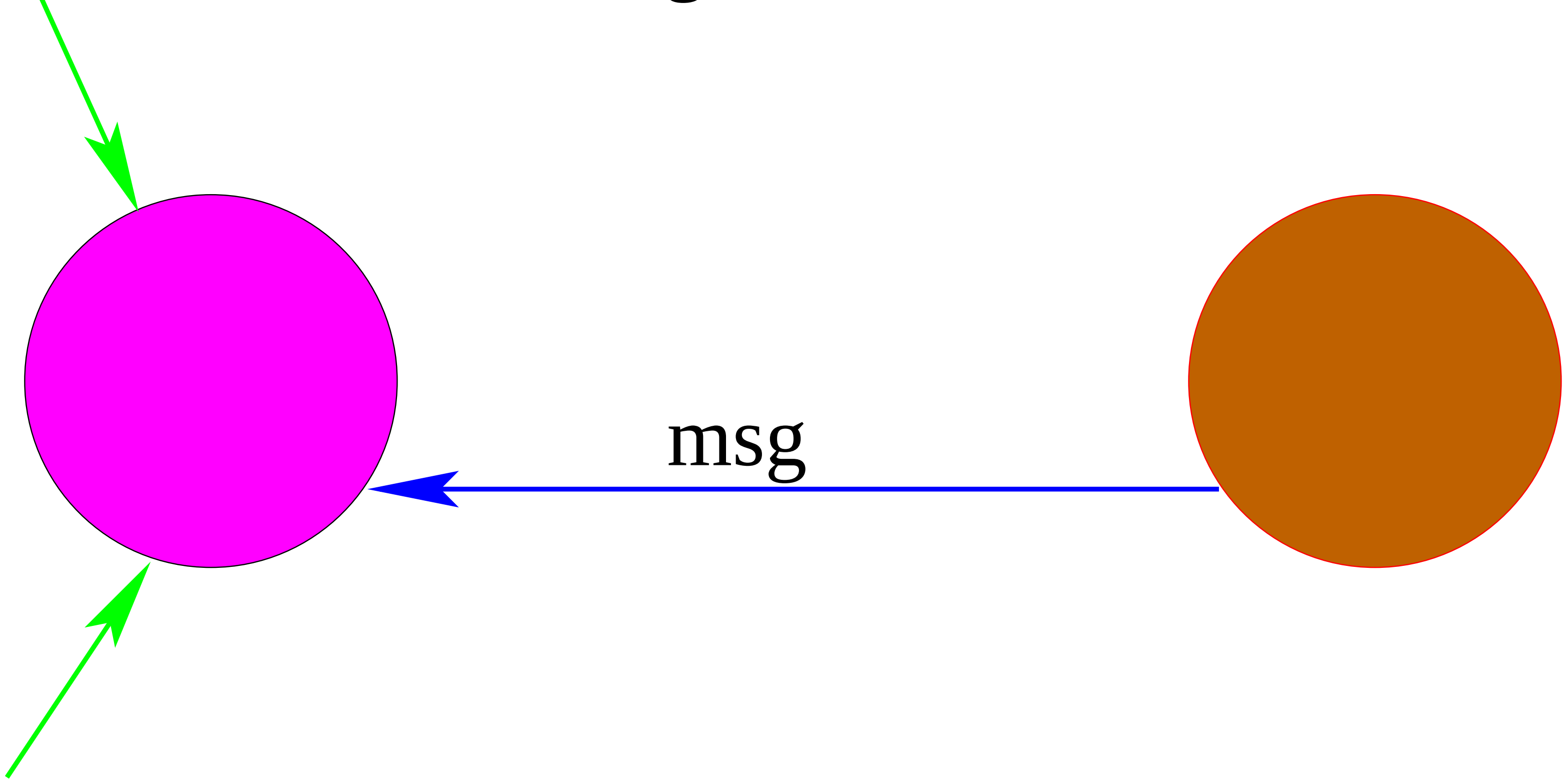
Sending a message



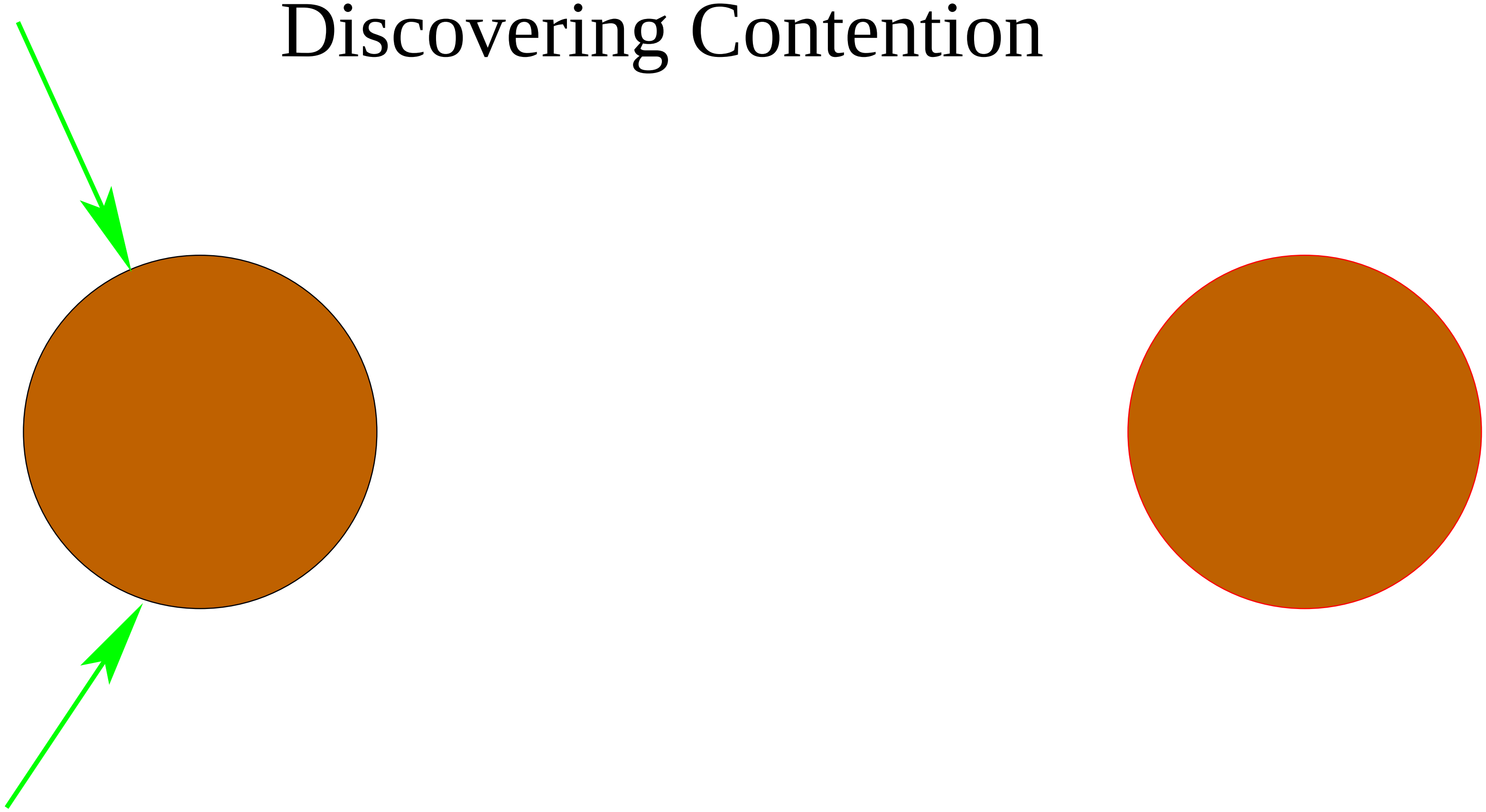
Sending another message



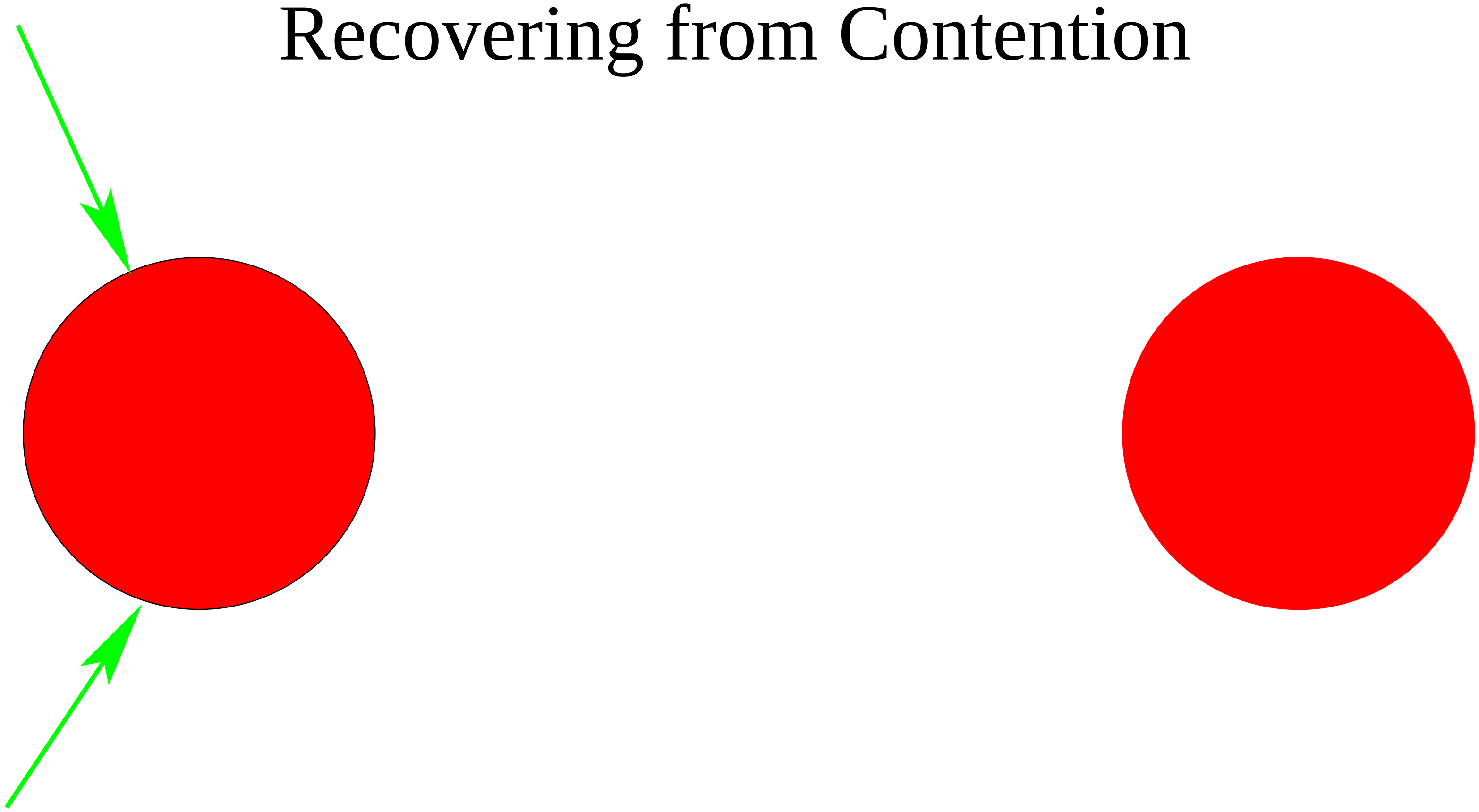
Discovering Contention



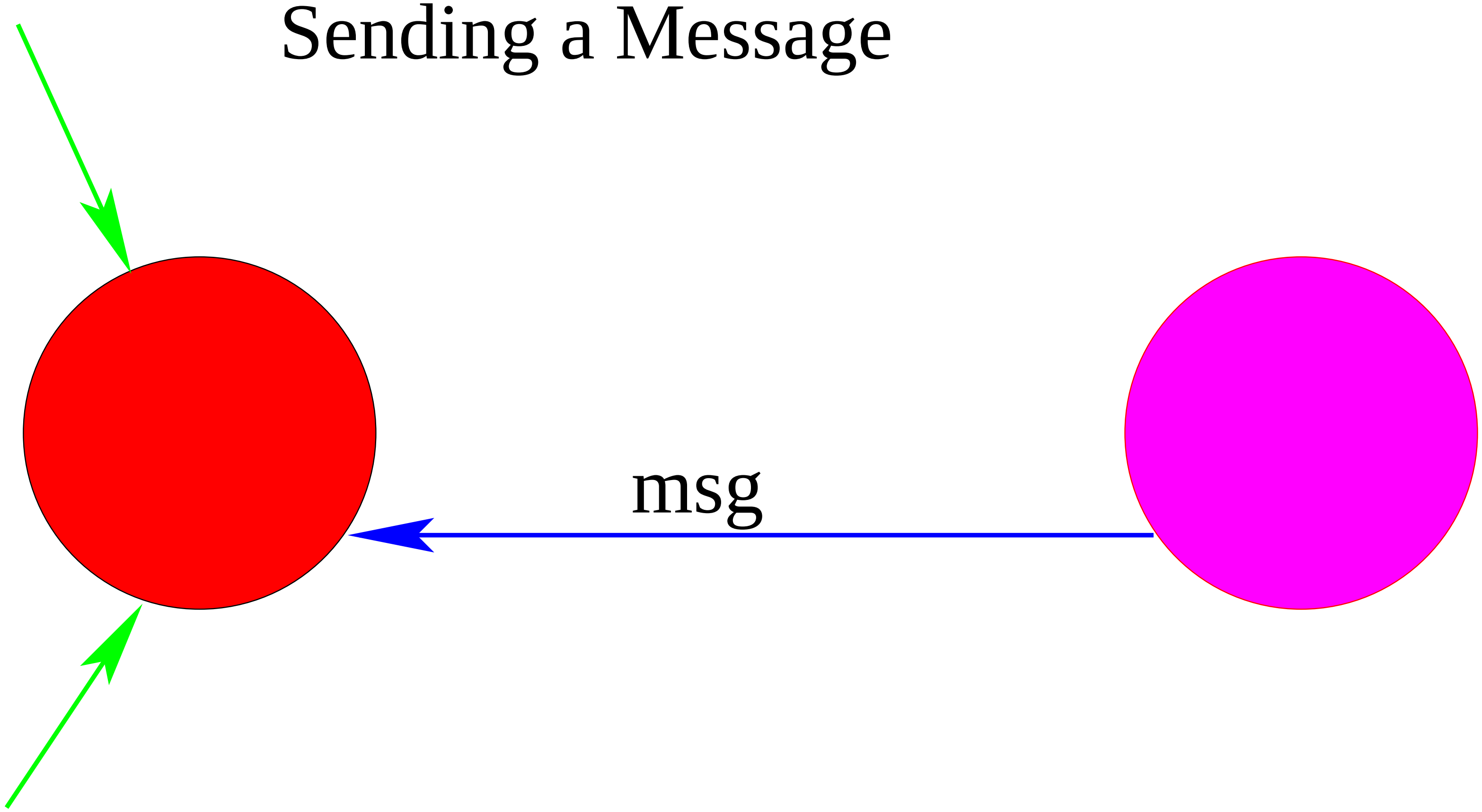
Discovering Contention



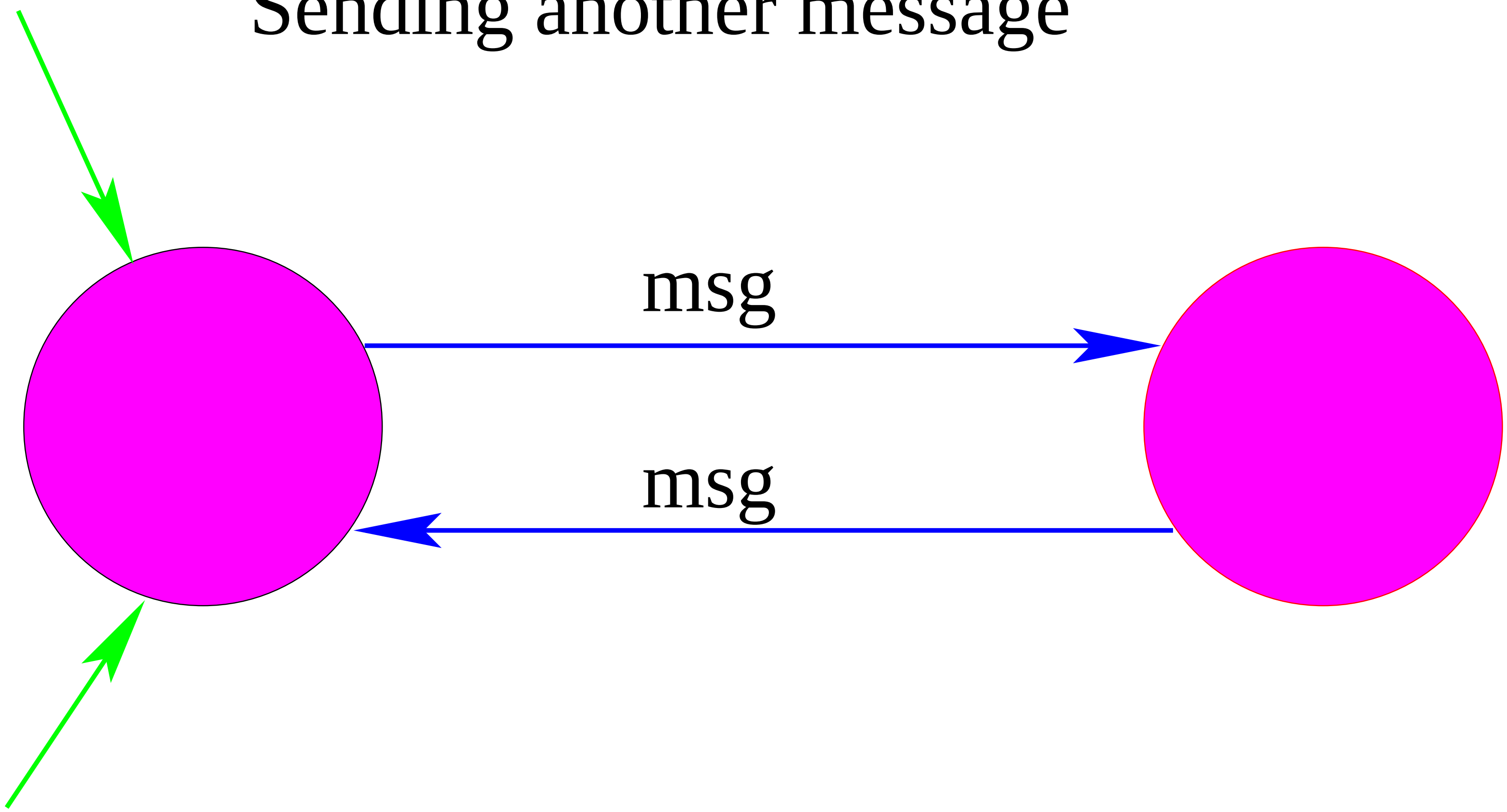
Recovering from Contention



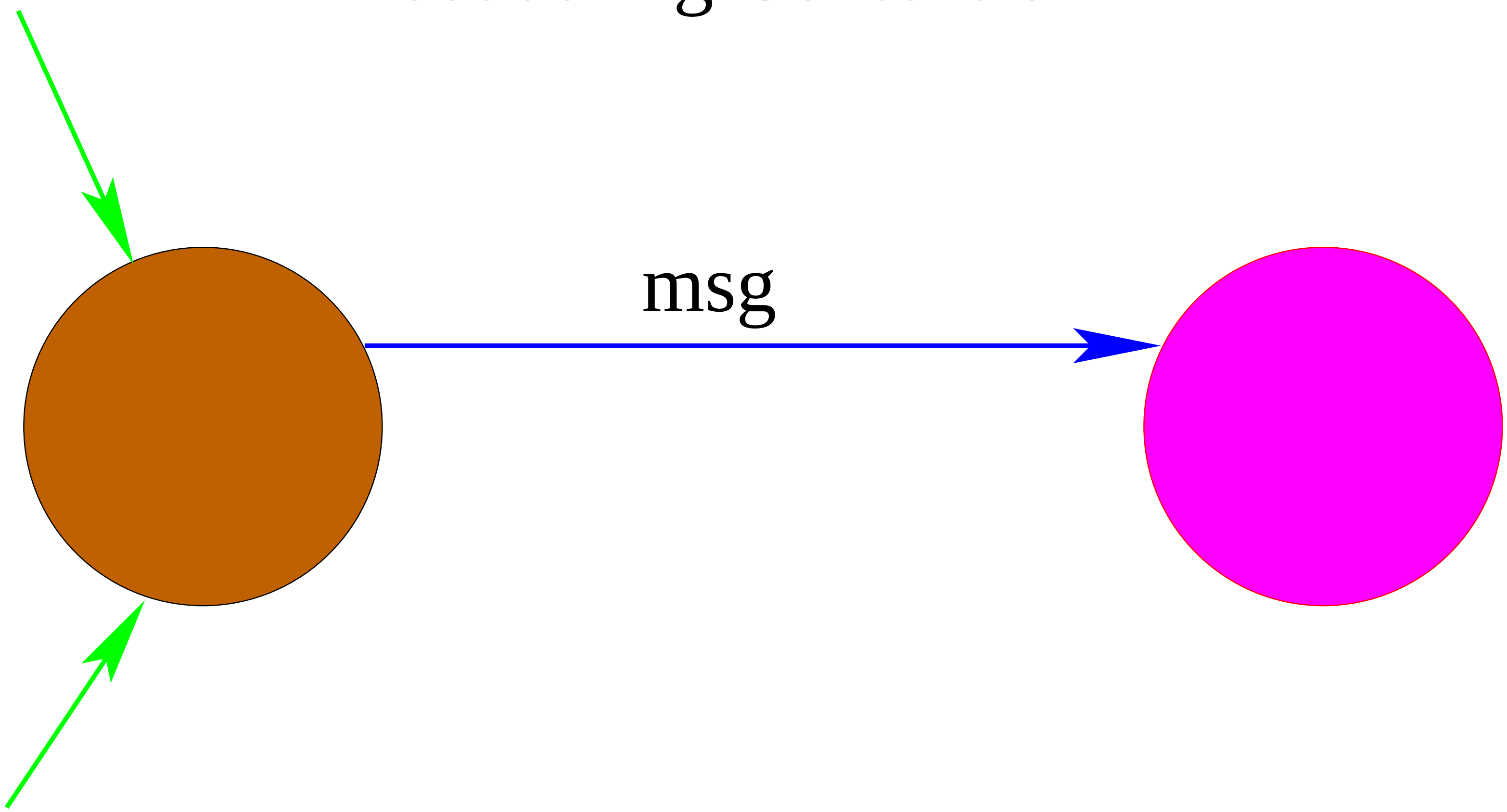
Sending a Message



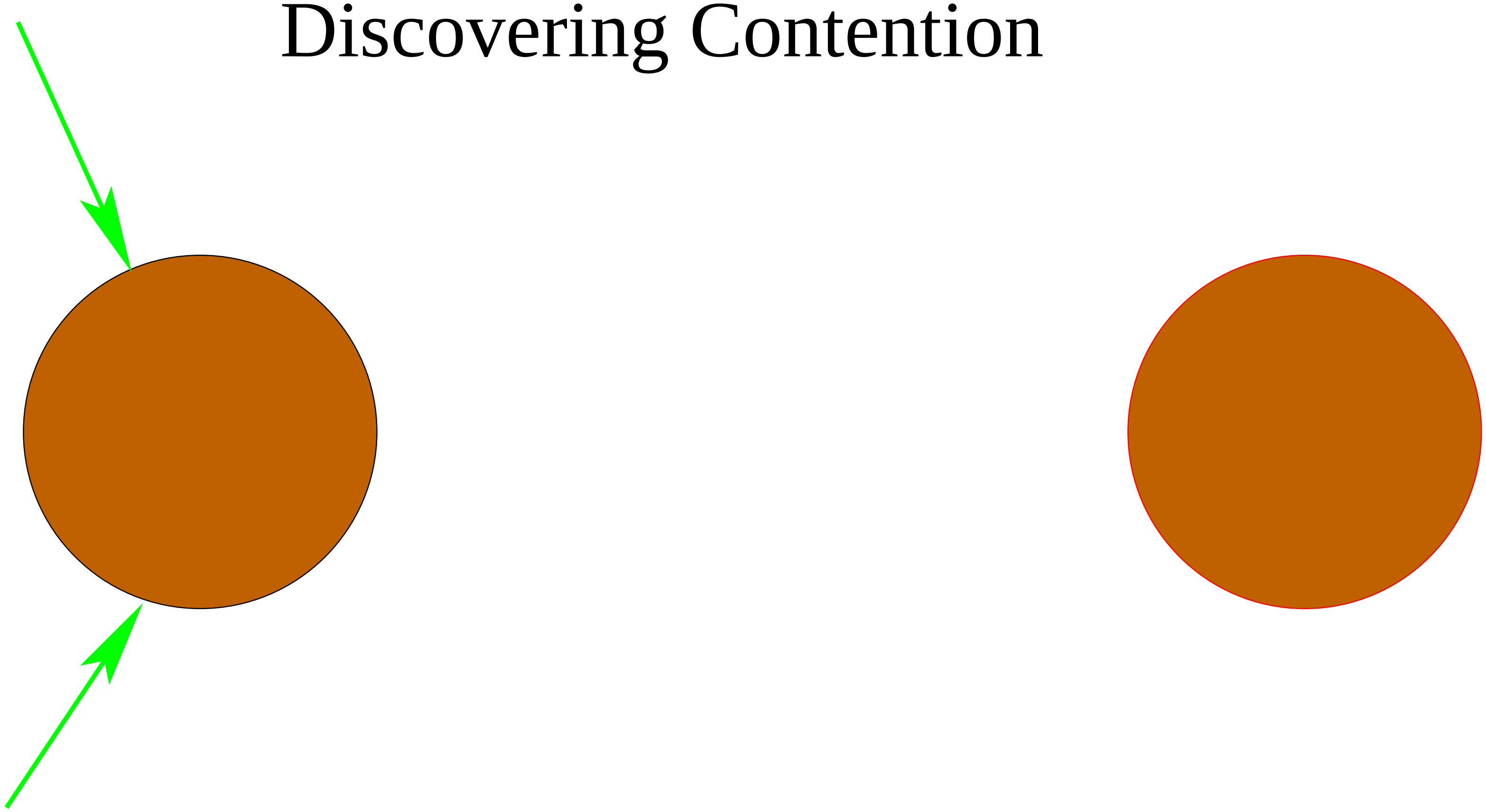
Sending another message



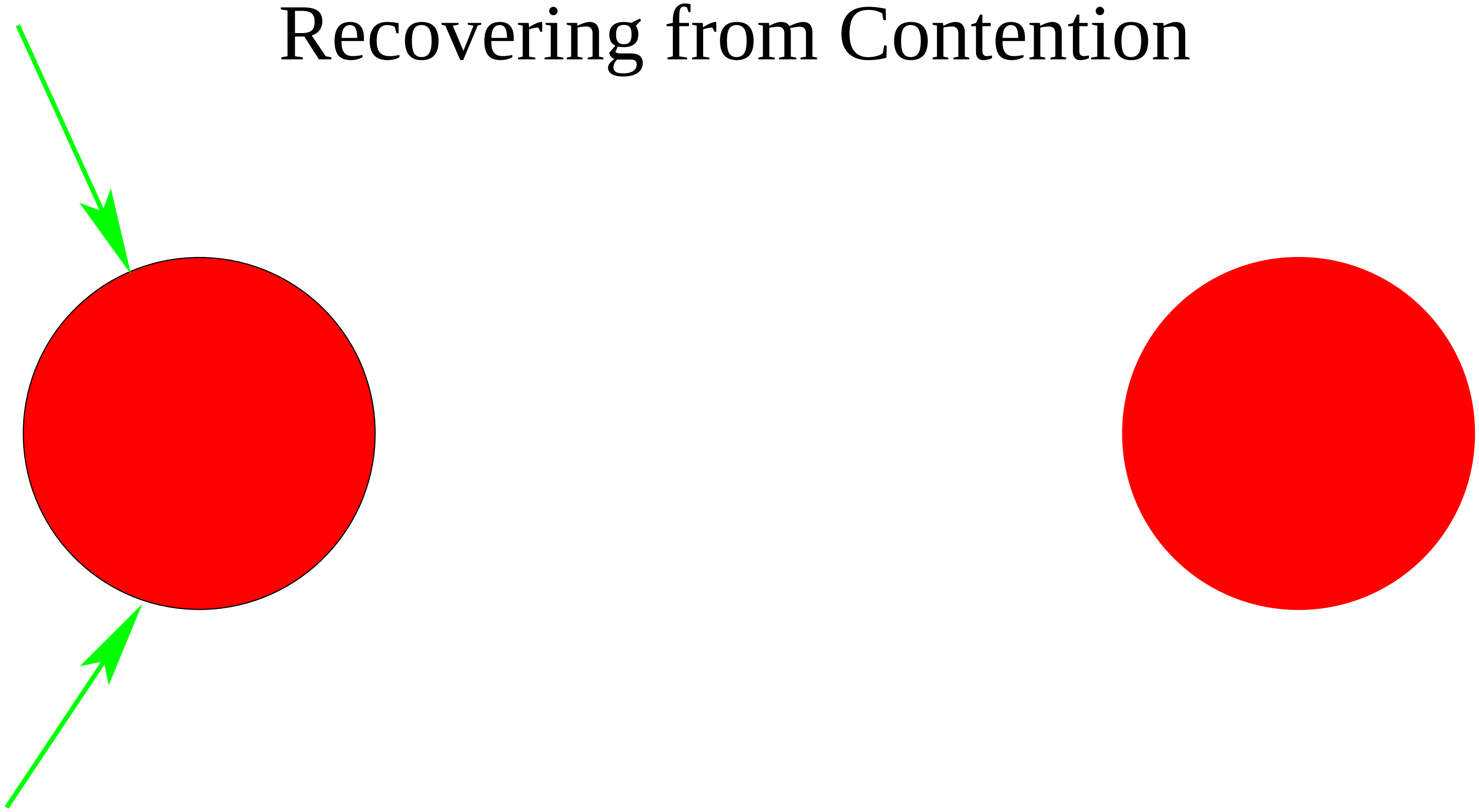
Discovering Contention



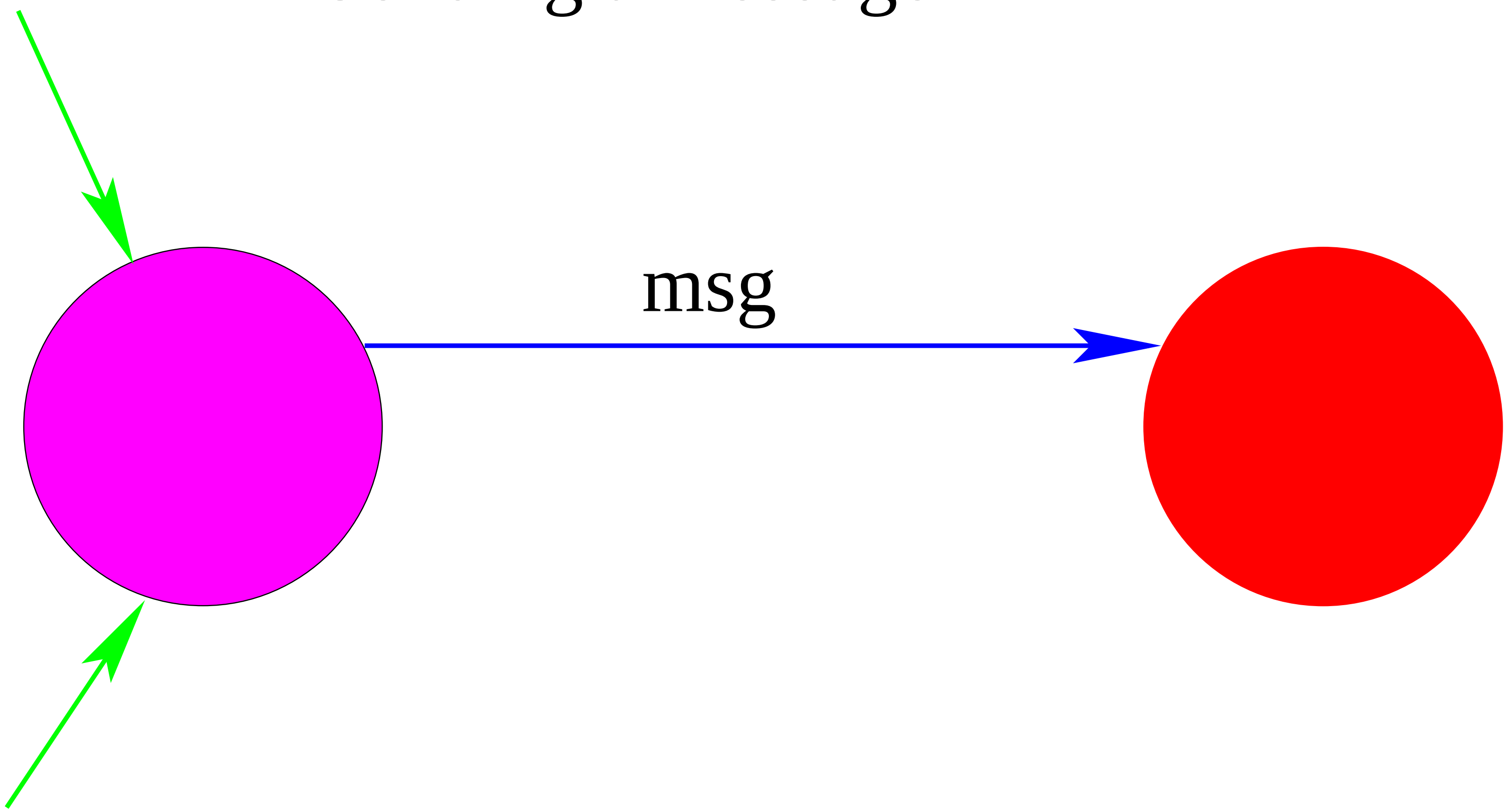
Discovering Contention



Recovering from Contention

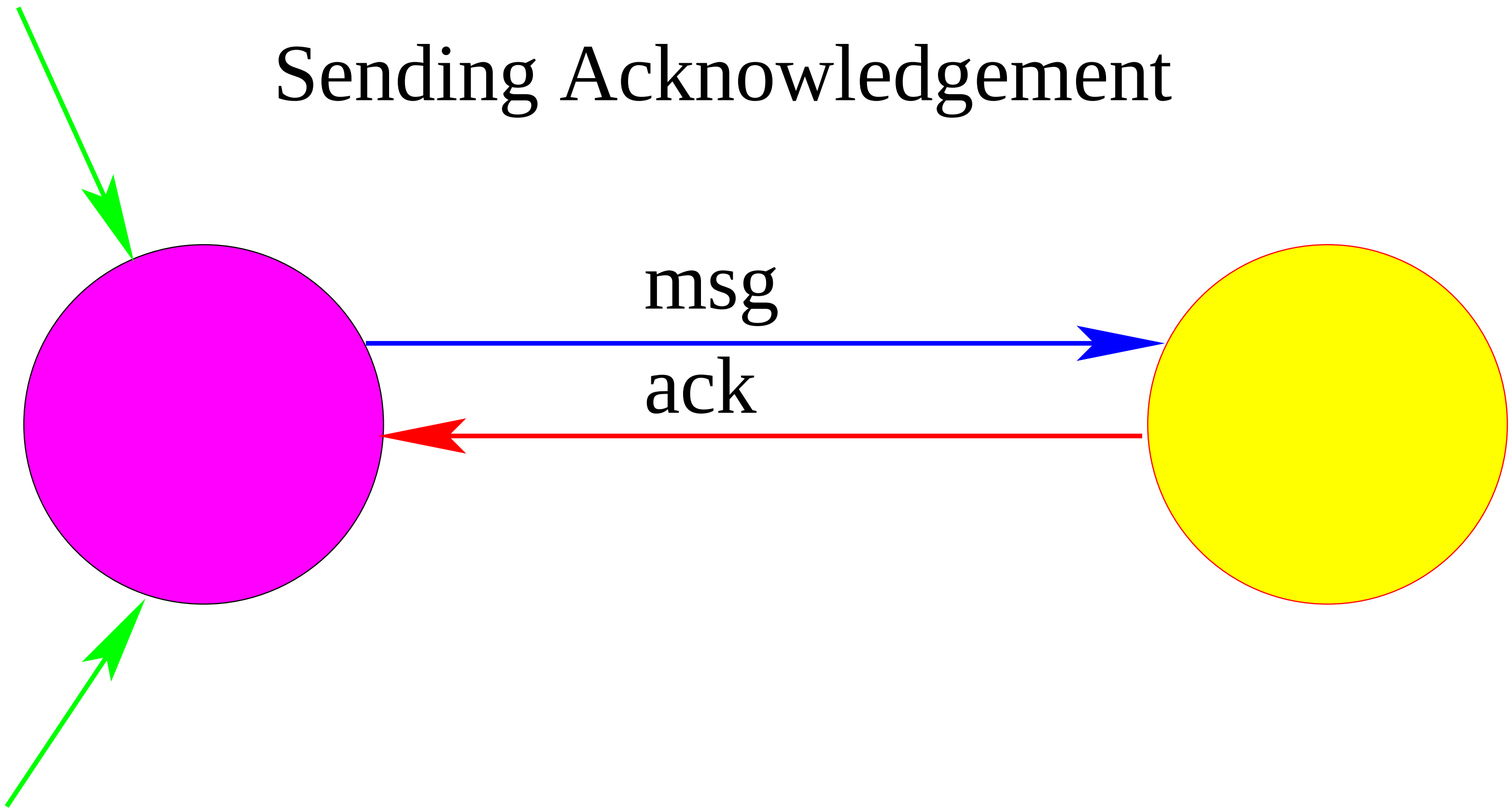


Sending a message



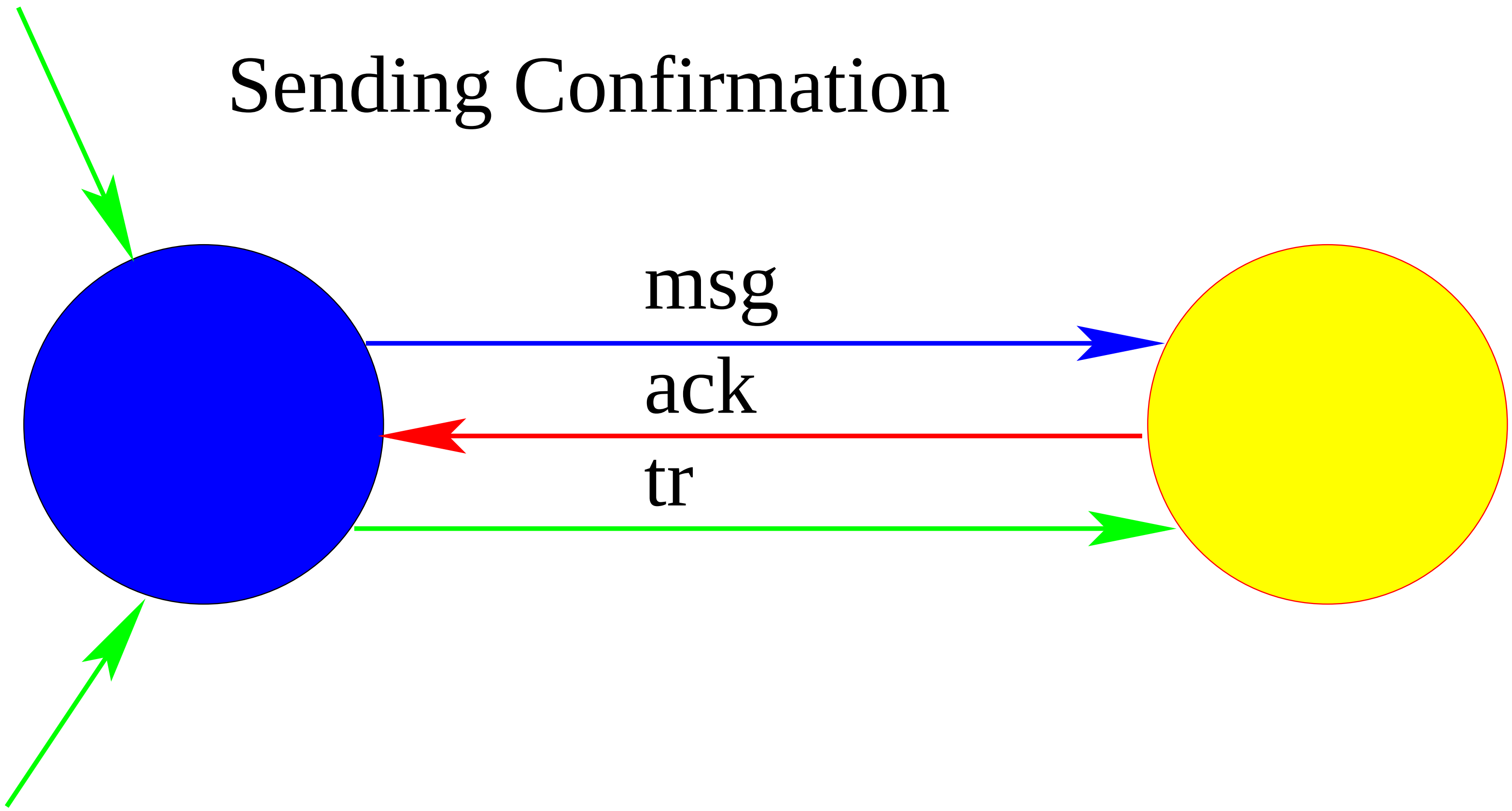
Receiving a message

Sending Acknowledgement

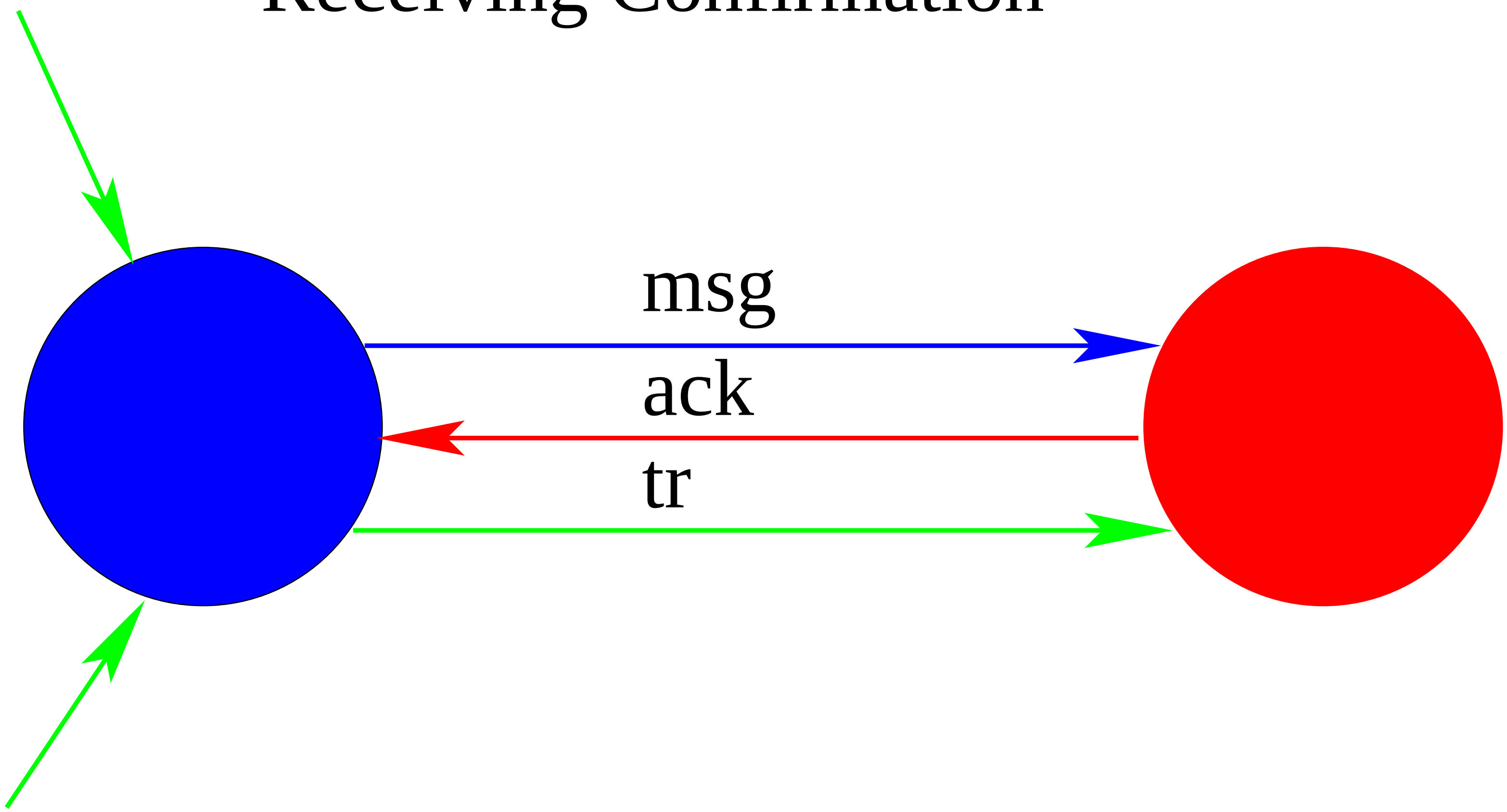


Receiving Acknowledgement

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 acknowledgment 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

$$\begin{array}{l}\text{send\_ack} \;\;\widehat{=}\\ \quad \textbf{any}\ x, y\ \textbf{where}\\ \qquad x, y \in msg - ack \;\;\wedge\\ \qquad y \notin \mathrm{dom}\,(msg)\\ \quad \textbf{then}\\ \qquad ack := ack \cup \{x \mapsto y\}\\ \quad \textbf{end}\end{array}$$

$$\begin{array}{l}\text{contention} \;\;\widehat{=}\\ \quad \textbf{any}\ x, y\ \textbf{where}\\ \qquad x, y \in msg - ack \;\;\wedge\\ \qquad y \in \mathrm{dom}\,(msg)\\ \quad \textbf{then}\\ \qquad cnt := cnt \cup \{x \mapsto y\}\\ \quad \textbf{end}\end{array}$$

- Introducing a dummy contention channel: cnt

$$cnt \in ND \nrightarrow ND$$

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

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

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**

$x \in ND - bm \;\; \land$

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

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

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 $\widehat{=}$

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

$$
\begin{array}{l}
\textsf{solve\_contention} \quad \widehat{=} \\
\quad \textbf{any } x, y \textbf{ where} \\
\quad\quad x, y \in cnt \cup cnt^{-1} \\
\quad \textbf{then} \\
\quad\quad msg := msg - cnt \quad \parallel \\
\quad\quad bm := bm - \mathsf{dom}\,(cnt) \quad \parallel \\
\quad\quad cnt := \emptyset \\
\quad \textbf{end}
\end{array}
$$

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