Simple and Efficient Local Codes for Distributed Stable Network Construction

Othon Michail

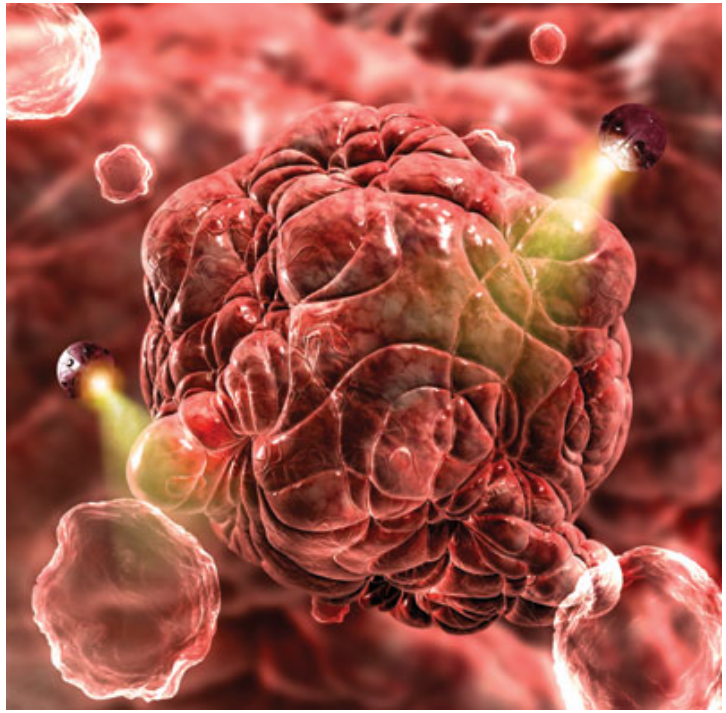
joint work with
Paul G. Spirakis

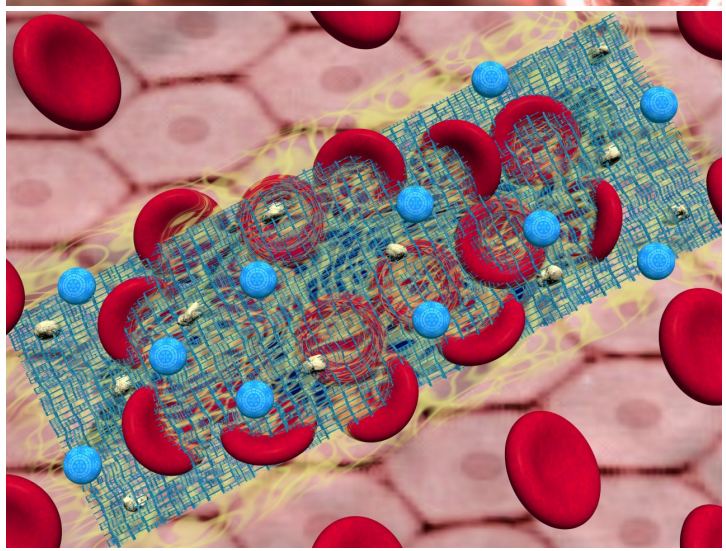
Computer Technology Institute & Press "Diophantus" (CTI)
Department of Computer Science, University of Liverpool, UK

33rd Annual ACM Symposium on Principles of Distributed Computing (PODC)
July 15-18, 2014
Paris, France

A Motivating Example

- n tiny computational devices
- Injected into a human circulatory system for monitoring/treating
- Move and interact passively (blood flow)
- Cooperation: can create bonds with each other
- Self-assemble into a desired global structure/network
- The artificial population evolves greater complexity, better storage capacity, and adapts and optimizes its performance to the specific task to be accomplished





Distributed Network Construction

Fundamental problem: Algorithmic distributed construction of an actual communication topology

- Processes can form/delete connections between them
- Physical or virtual connections depending on the application
- on/off case: a connection either exists (active) or does not exist (inactive)
- Initially all connections are inactive
- Goal: End up with a desired stable network

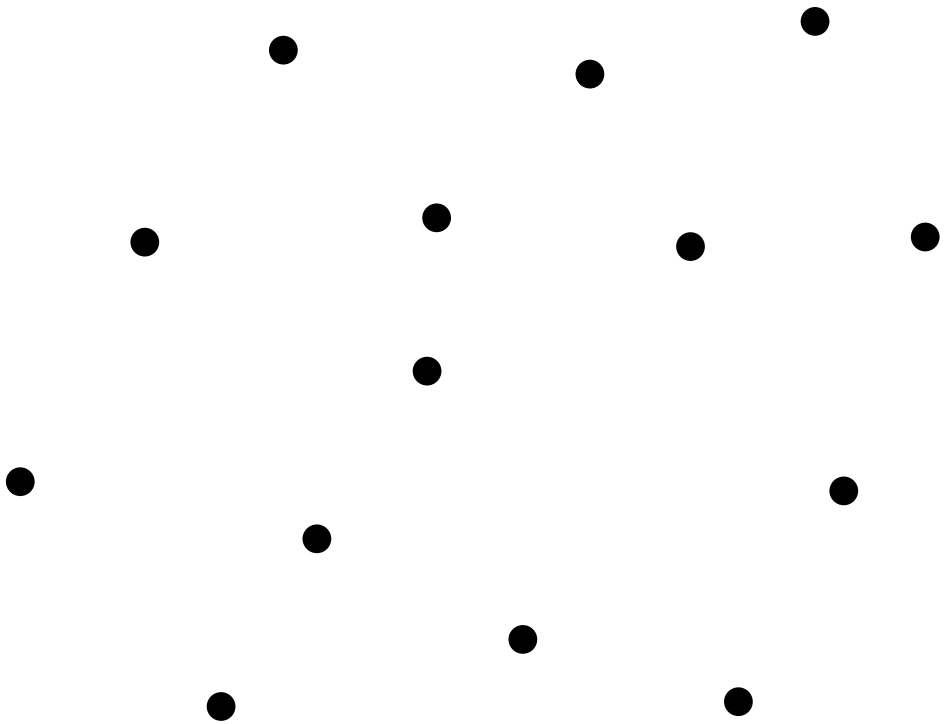
A Minimal Model

- Processes are finite automata
- Homogeneity: All begin from the same initial state and execute the same finite program
- Adversarial environment: Fair adversary scheduler choosing pairwise interactions (a la population protocols)
- Uniform random scheduler to estimate efficiency
- Complex global behaviour via simple, uniform, anonymous, homogeneous, and cooperative entities

An Illustration: Spanning Star

- 2 states: black and red
- Initially all black
- Construct a global star
- Program:

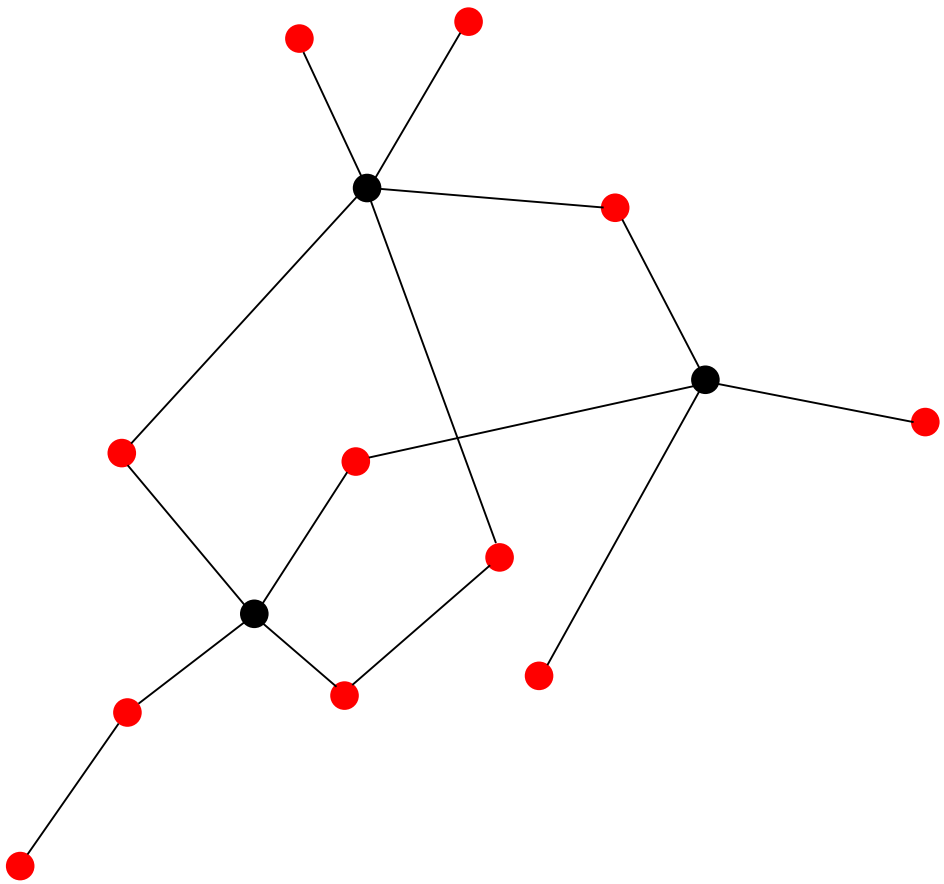
$$(b, b, 0) \to (b, r, 1)$$
$$(r, r, 1) \to (r, r, 0)$$
$$(b, r, 0) \to (b, r, 1)$$



An Illustration: Spanning Star

- 2 states: black and red
- Initially all black
- Construct a global star
- Program:

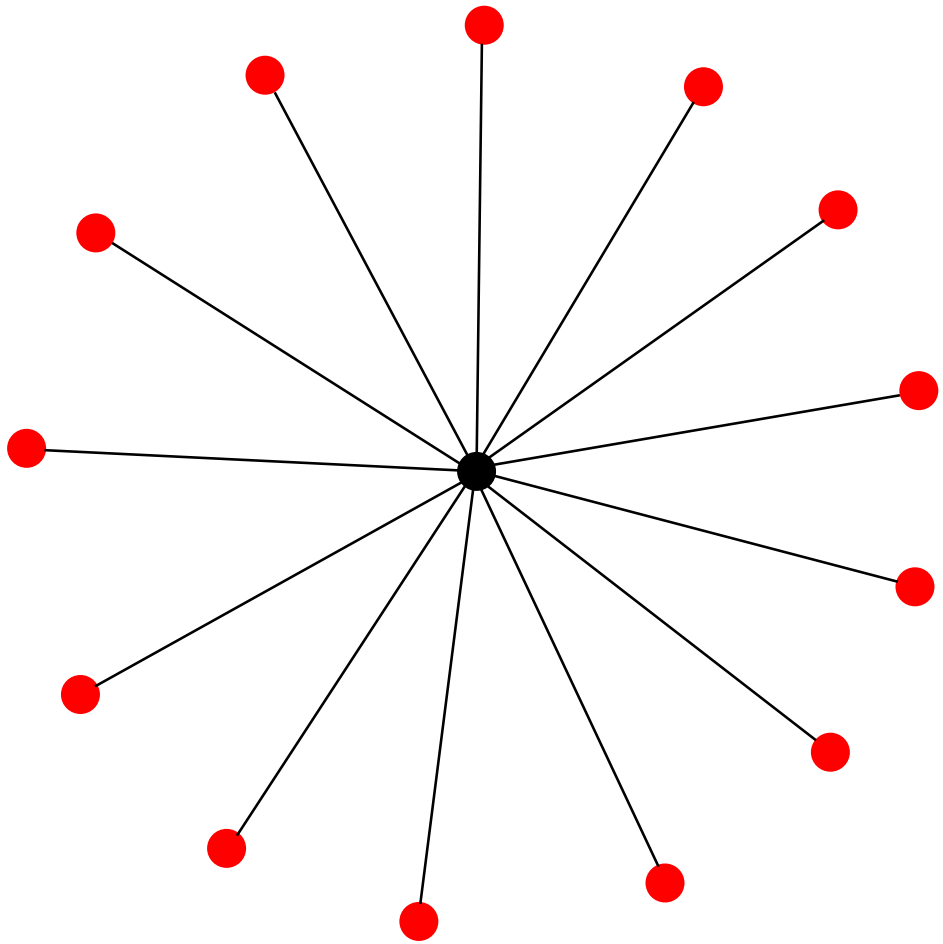
$$(b, b, 0) \to (b, r, 1)$$
$$(r, r, 1) \to (r, r, 0)$$
$$(b, r, 0) \to (b, r, 1)$$



An Illustration: Spanning Star

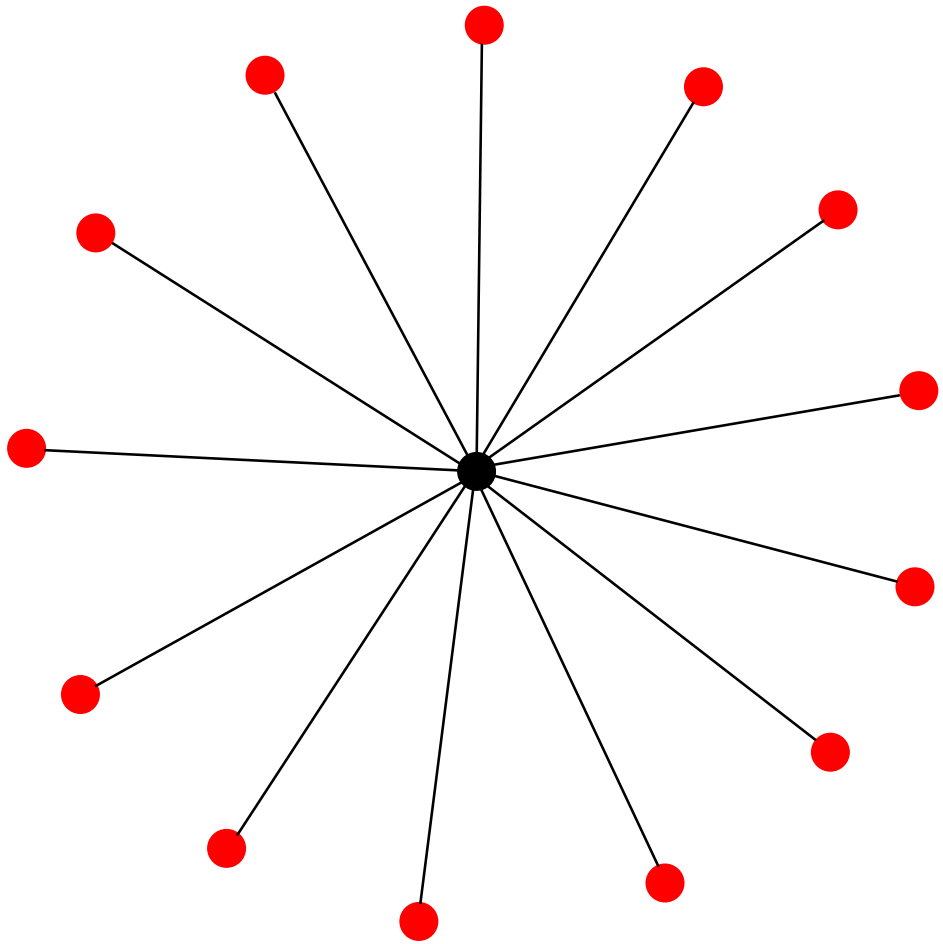
- 2 states: black and red
- Initially all black
- Construct a global star
- Program:

$$(b, b, 0) \to (b, r, 1)$$
$$(r, r, 1) \to (r, r, 0)$$
$$(b, r, 0) \to (b, r, 1)$$



An Illustration: Spanning Star

- Size: 2 states
- Time: $O(n^2 \log n)$
- Optimal w.r.t. both
- Time = # interactions (sequencial)
  - Parallel time is on the average $\frac{1}{n} \cdot$ sequential time



Related Work

- Population Protocol model [AADFP, Distr. Comp., '06]
- Mediated Population Protocol model [MCS, Theor. Comp. Sci., '11]
  - The focus was on the computation of functions on input values
  - In contrast to MPPs we allow only two possible edge-states
- PPs formally equivalent to Chemical Reaction Networks under uniform random scheduler [Doty, SODA, '14]
  - PPs/CRNs can only capture the dynamics of molecular counts and not of structure formation
  - We aim to capture the stable structures that may occur in a well-mixed solution

Our Goal: Determine what stable structures can result in such systems, how fast, and under what conditions (e.g. by what underlying codes/ reaction-rules)

Overview-Contribution

- Simple and efficient protocols for fundamental networks: spanning lines, spanning rings, cycle-covers, partitioning into cliques, and regular networks
- $\Omega(n \log n)$ generic lower bound for all spanning networks
- Spanning line: important for universal constructors
  - $\Omega(n^2)$ lower bound
  - 3 protocols, each improving on the running time by using more states
  - Simplest: $\Omega(n^4)$ and $O(n^5)$ using only 5 states
  - Fastest: $O(n^3)$ expected time using 9 states
- Question: What is in principle constructible by our model?
  - We give several satisfactory characterizations (universality results) by (i) simulating TMs and (ii) organizing the population into a distributed system with names and logarithmic local memories

The Model of Network Constructors

1. Q: finite set of node-states,
2. $q_0 \in Q$: initial node-state,
3. $Q_{out} \subseteq Q$: set of output node-states, and
4. $\delta : Q \times Q \times \{0,1\} \to Q \times Q \times \{0,1\}$: the transition function

- In every step, a pair uv is selected by the scheduler and u, v interact according to δ
- Fair scheduler: If a configuration is reachable infinitely often then it is eventually reached
- Output network: nodes that are in output states and edges between them that are active
- Stability: The output network cannot change in future steps

Global Line

Global Line: For all n, the n processes must construct a spanning line

Theorem (Generic Lower Bound)

The expected time to convergence (always under the uniform random scheduler) of any protocol that constructs a spanning network is $\Omega(n \log n)$.

Proof.

Consider the time at which the last edge is activated. By that time, all nodes have some active edge incident to them, thus every node has interacted at least once. The latter can be shown to require an expected number of $\Theta(n \log n)$ steps. □

Theorem (Line Lower Bound)

The expected time to convergence of any protocol that constructs a spanning line is $\Omega(n^2)$.

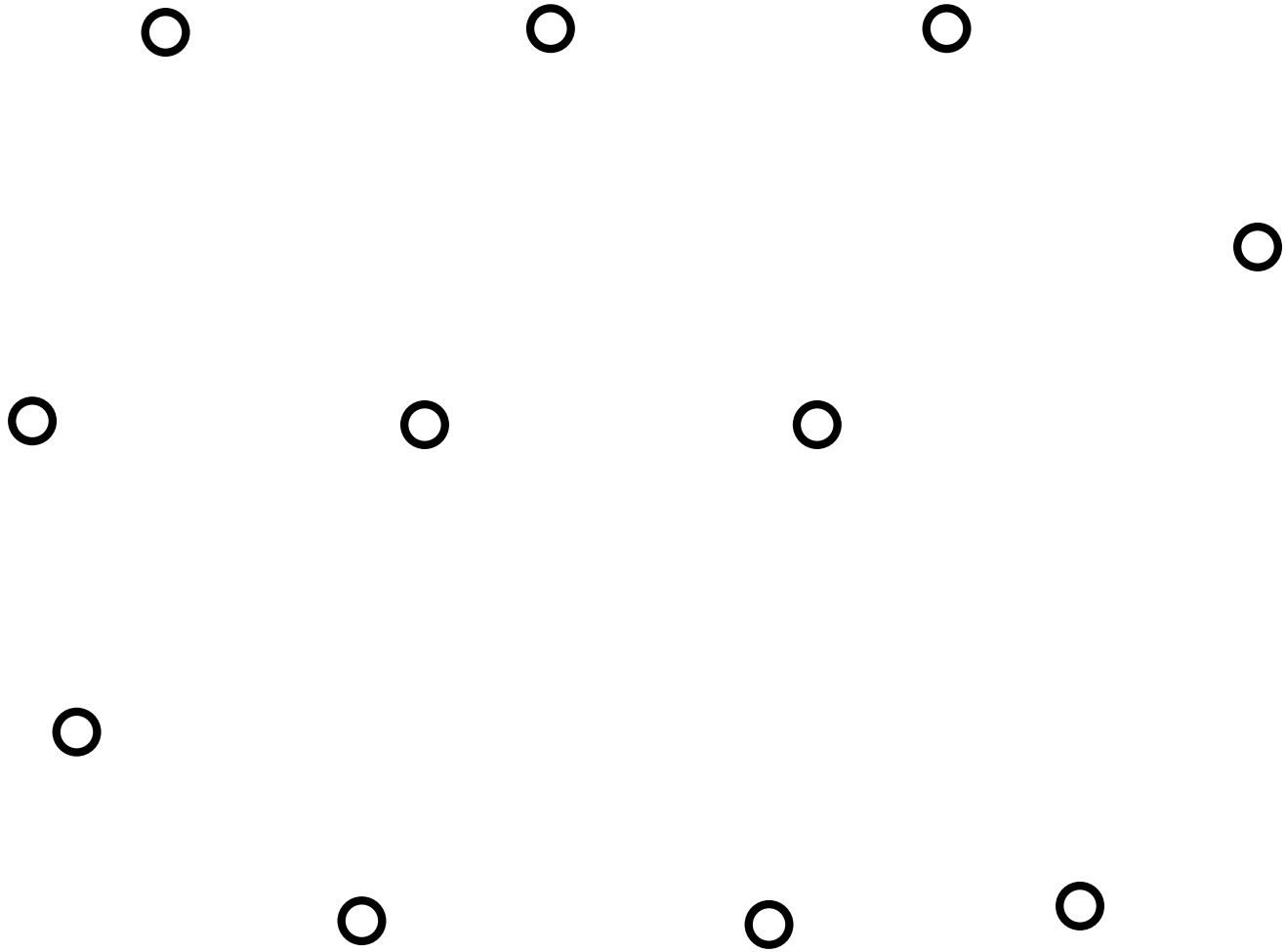
Simple-Global-Line Protocol

- Protocol *Simple-Global-Line*:

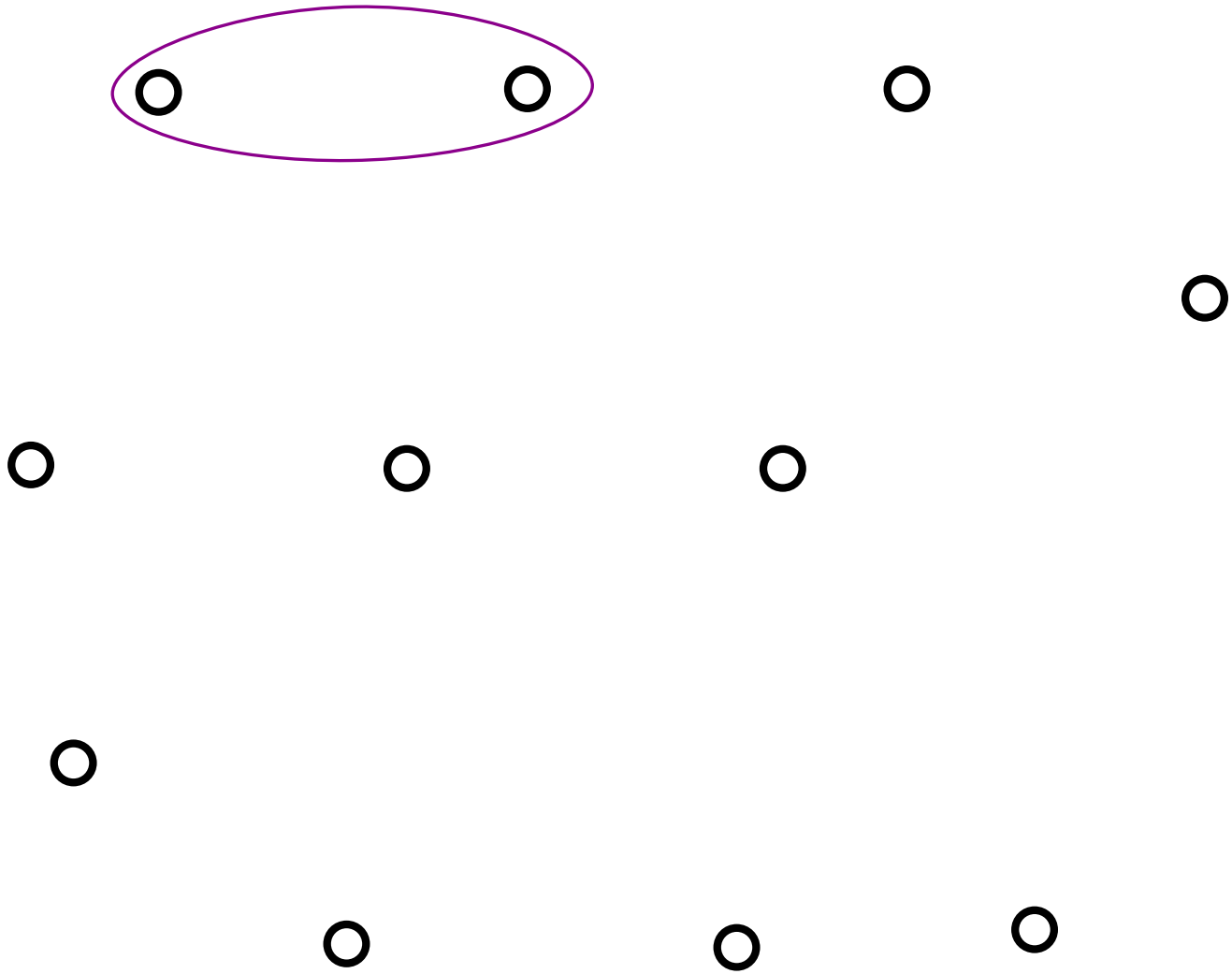
$$
\begin{aligned}
(q_0, q_0, 0) &\to (q_1, l, 1) \\
(l, q_0, 0) &\to (q_2, l, 1) \\
(l, l, 0) &\to (q_2, w, 1) \\
(w, q_2, 1) &\to (q_2, w, 1) \\
(w, q_1, 1) &\to (q_2, l, 1)
\end{aligned}
$$

  - Every node remembers its degree (q_0, q_1, q_2)
  - Every line has a unique leader (endpoint: l, internal: w)
  - Lines expand towards isolated nodes and merge to other lines always via their l leaders
  - After merging, the new line has a single internal w leader that performs a random walk until it reaches an endpoint and become l

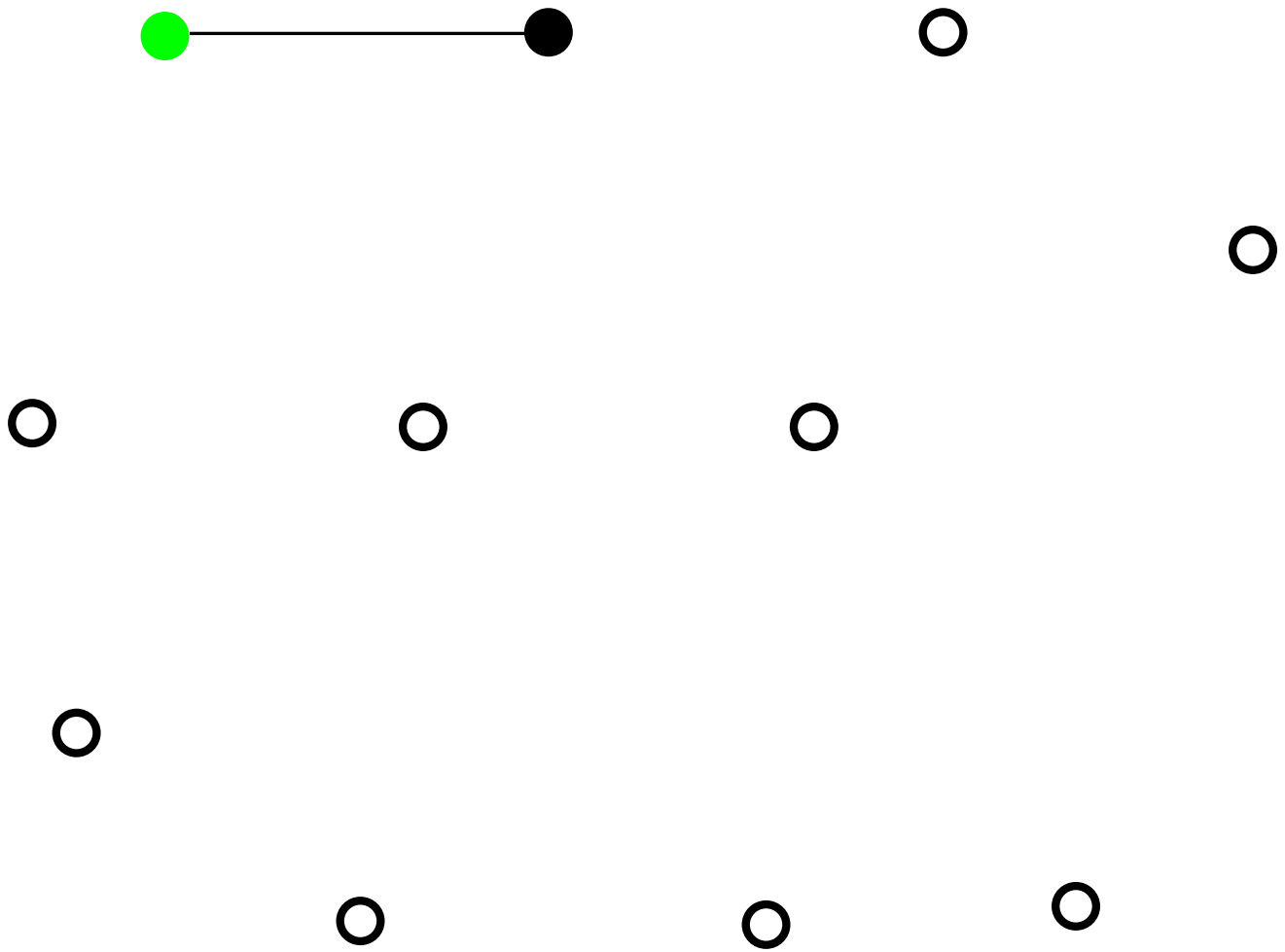
Simple-Global-Line Protocol



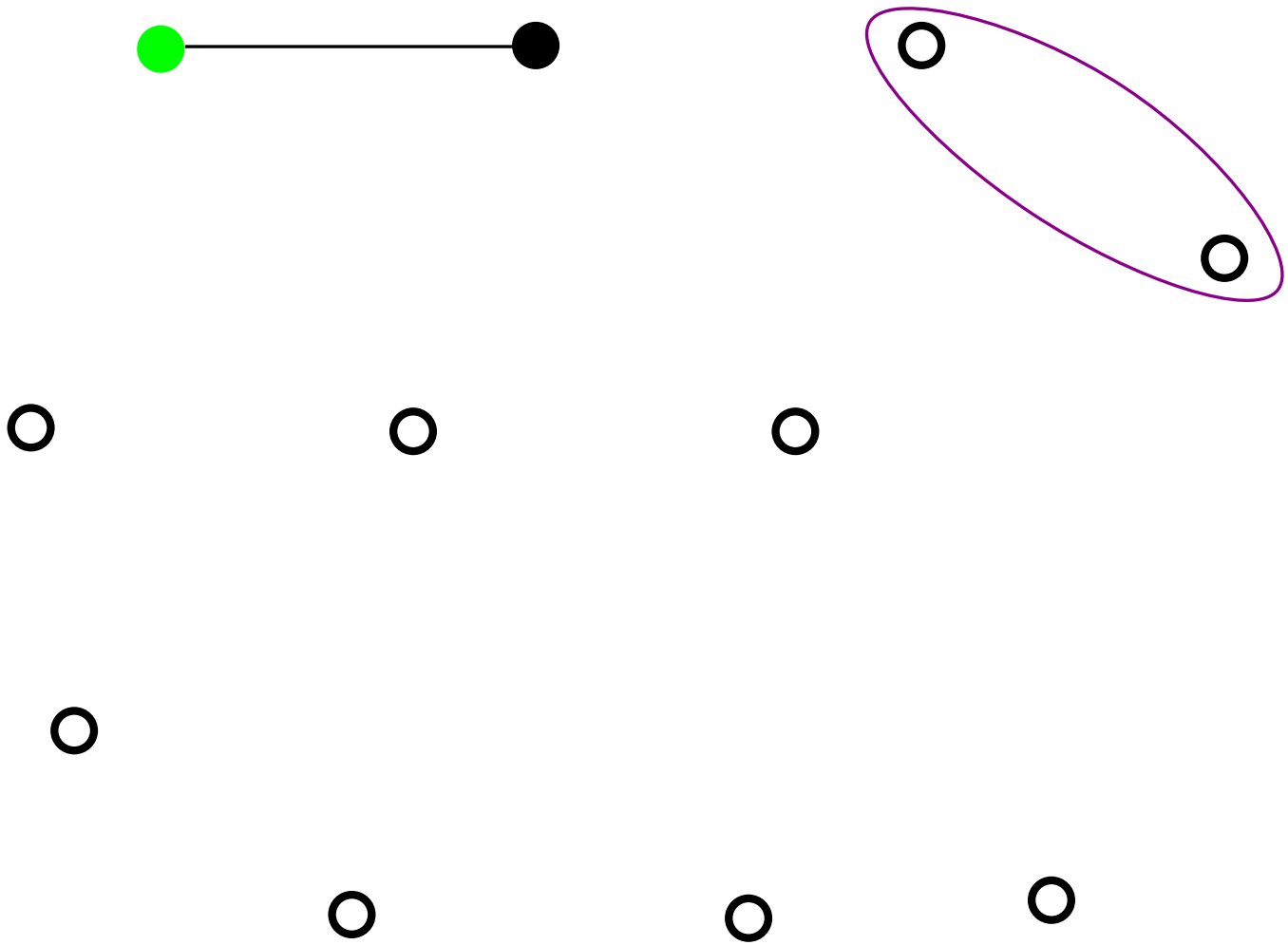
Simple-Global-Line Protocol



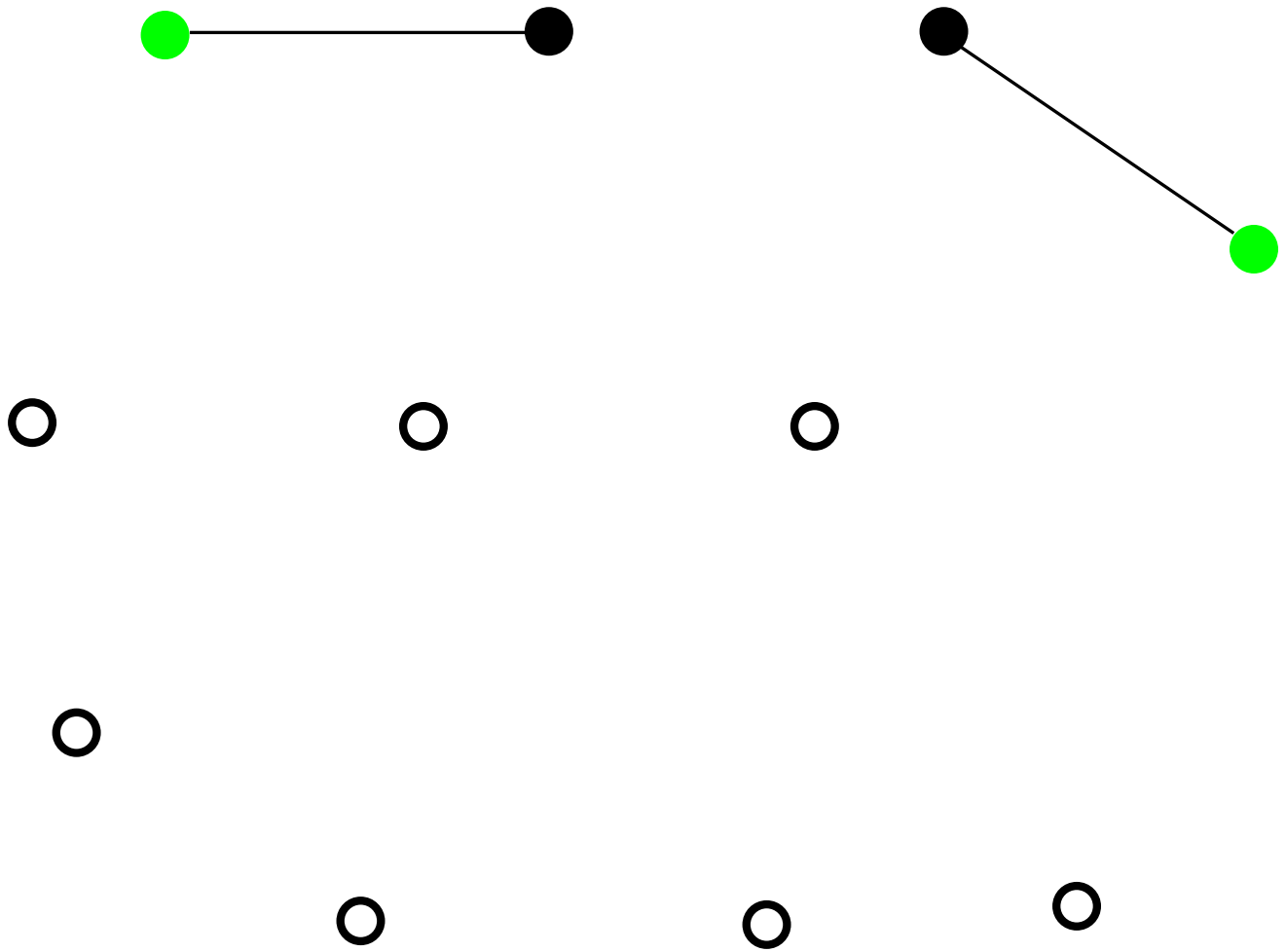
Simple-Global-Line Protocol



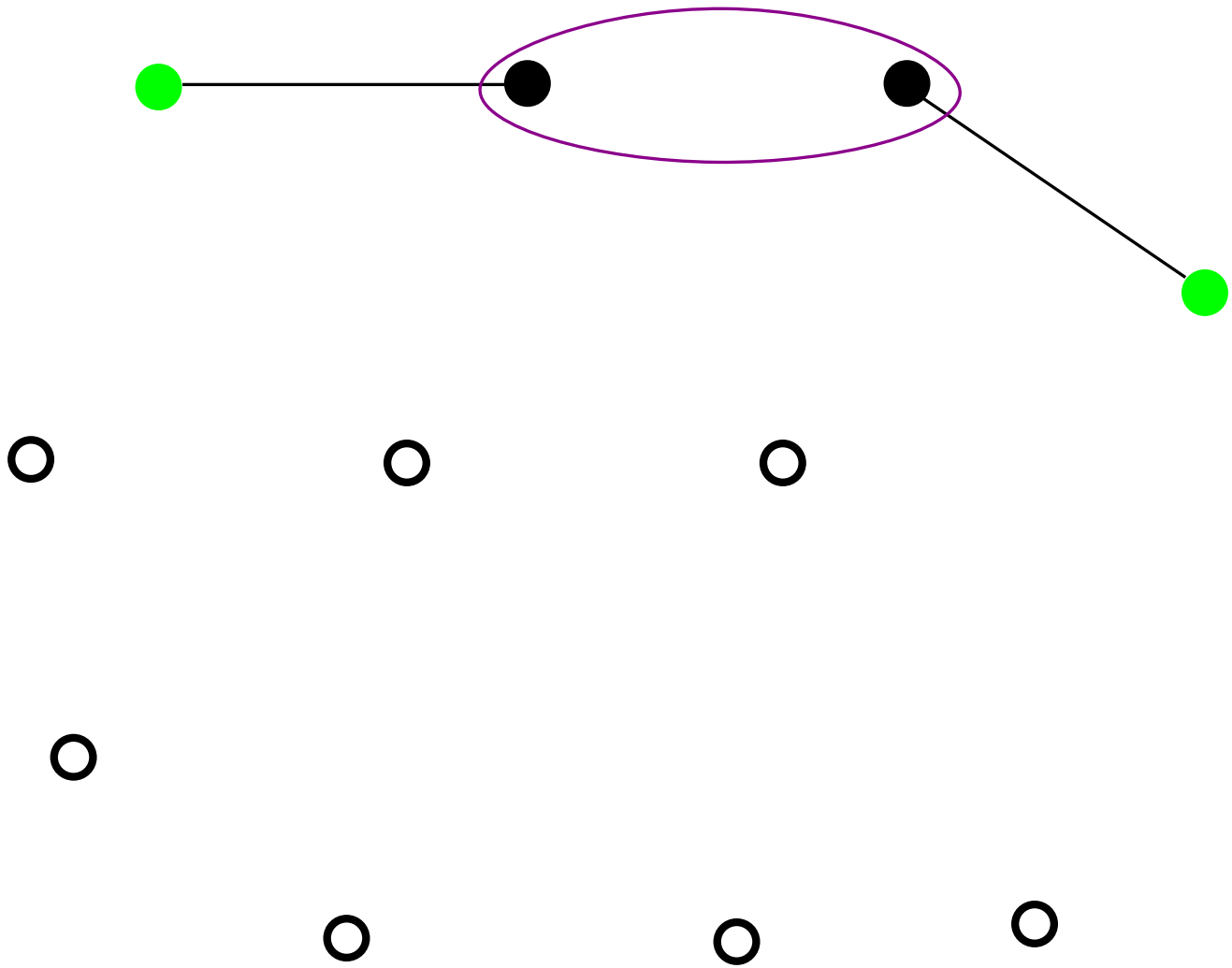
Simple-Global-Line Protocol



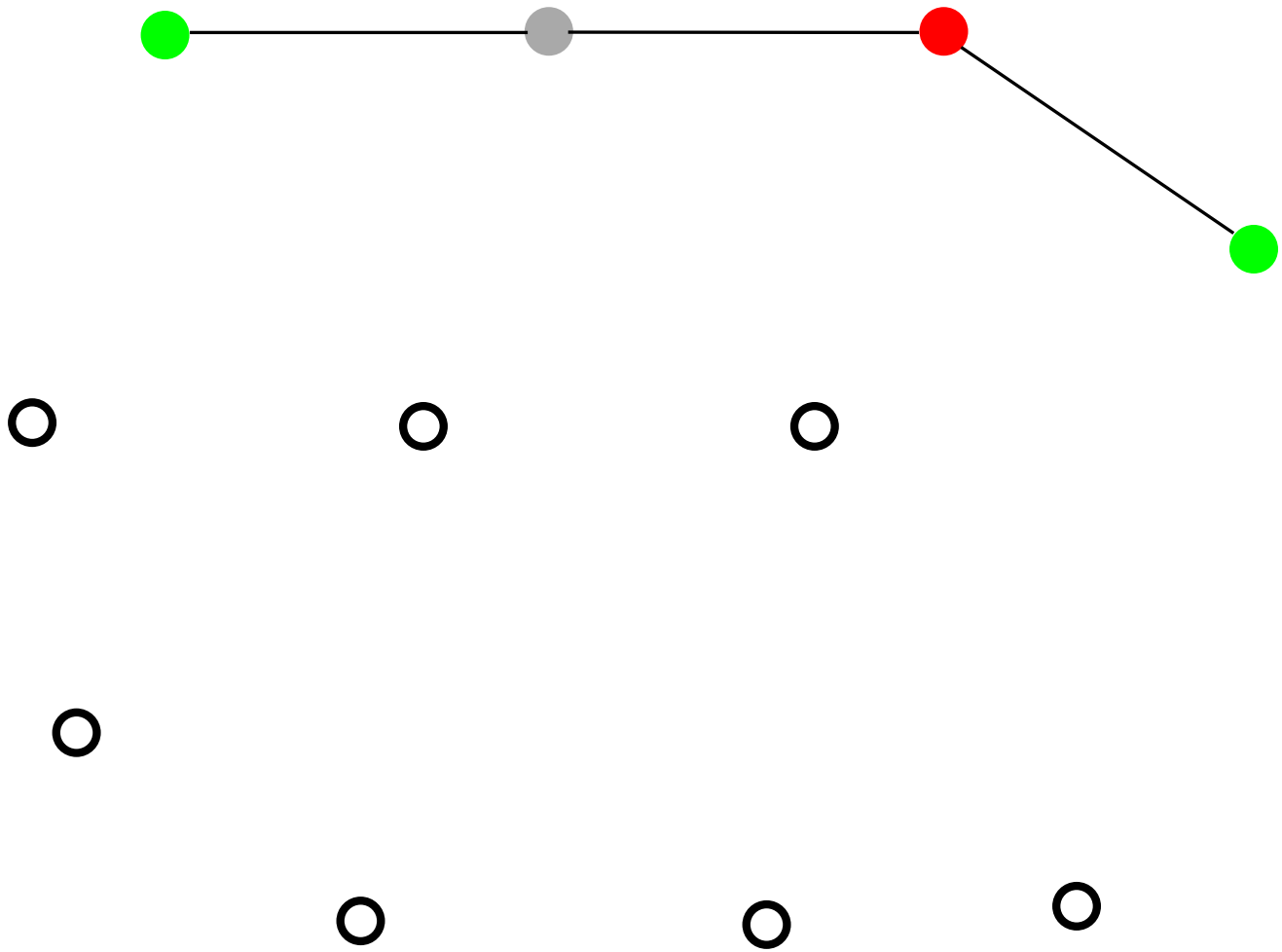
Simple-Global-Line Protocol



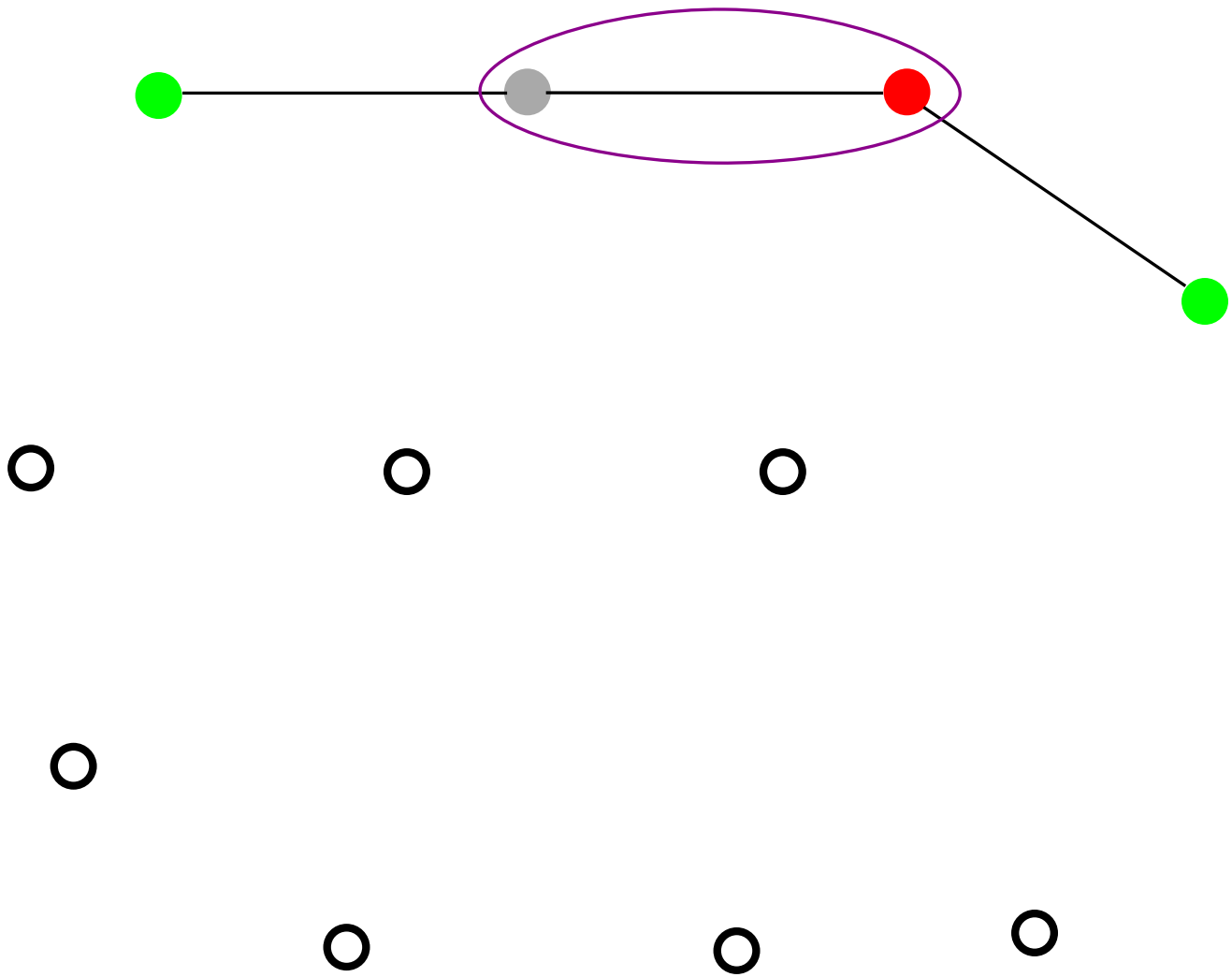
Simple-Global-Line Protocol



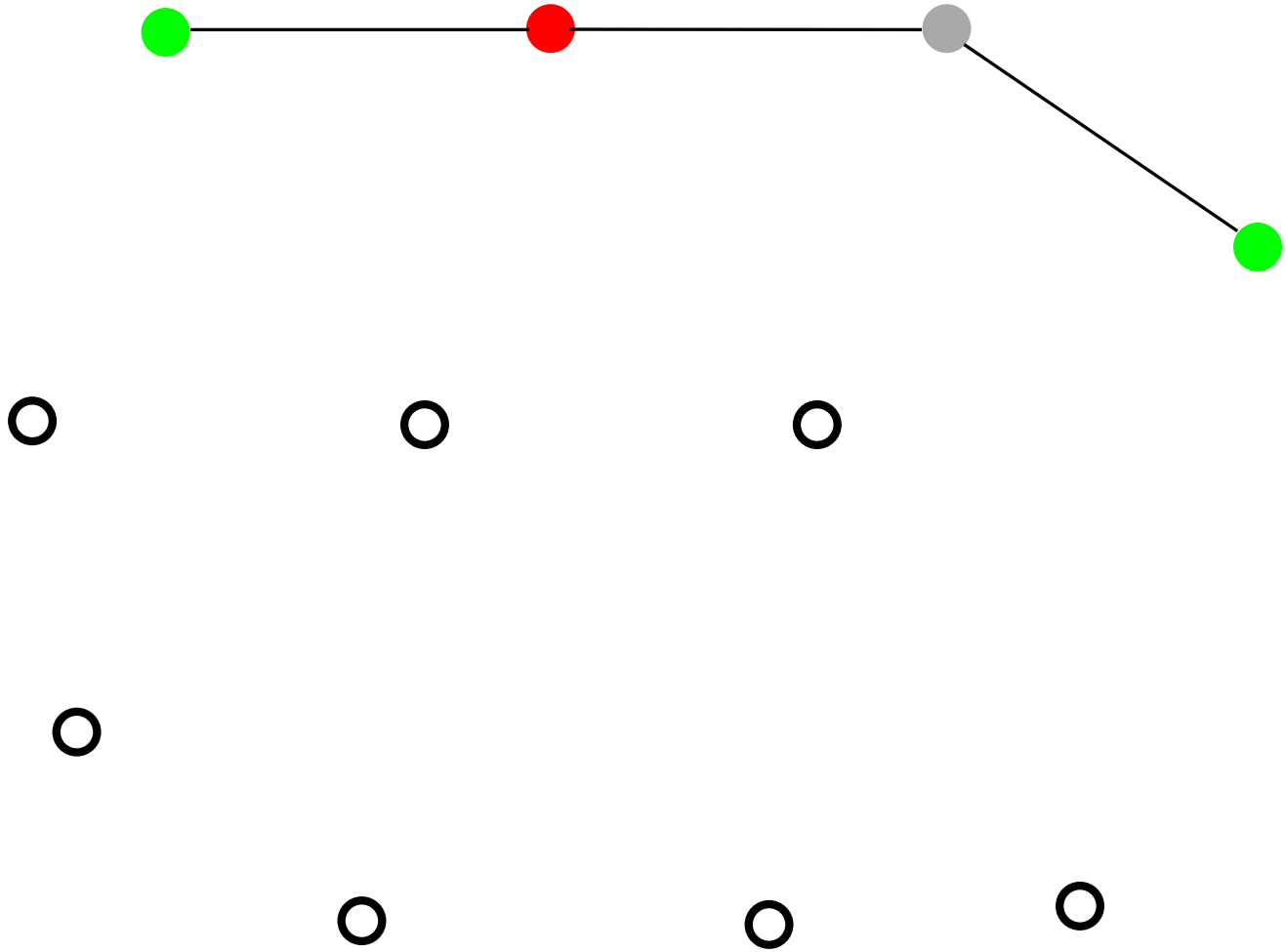
Simple-Global-Line Protocol



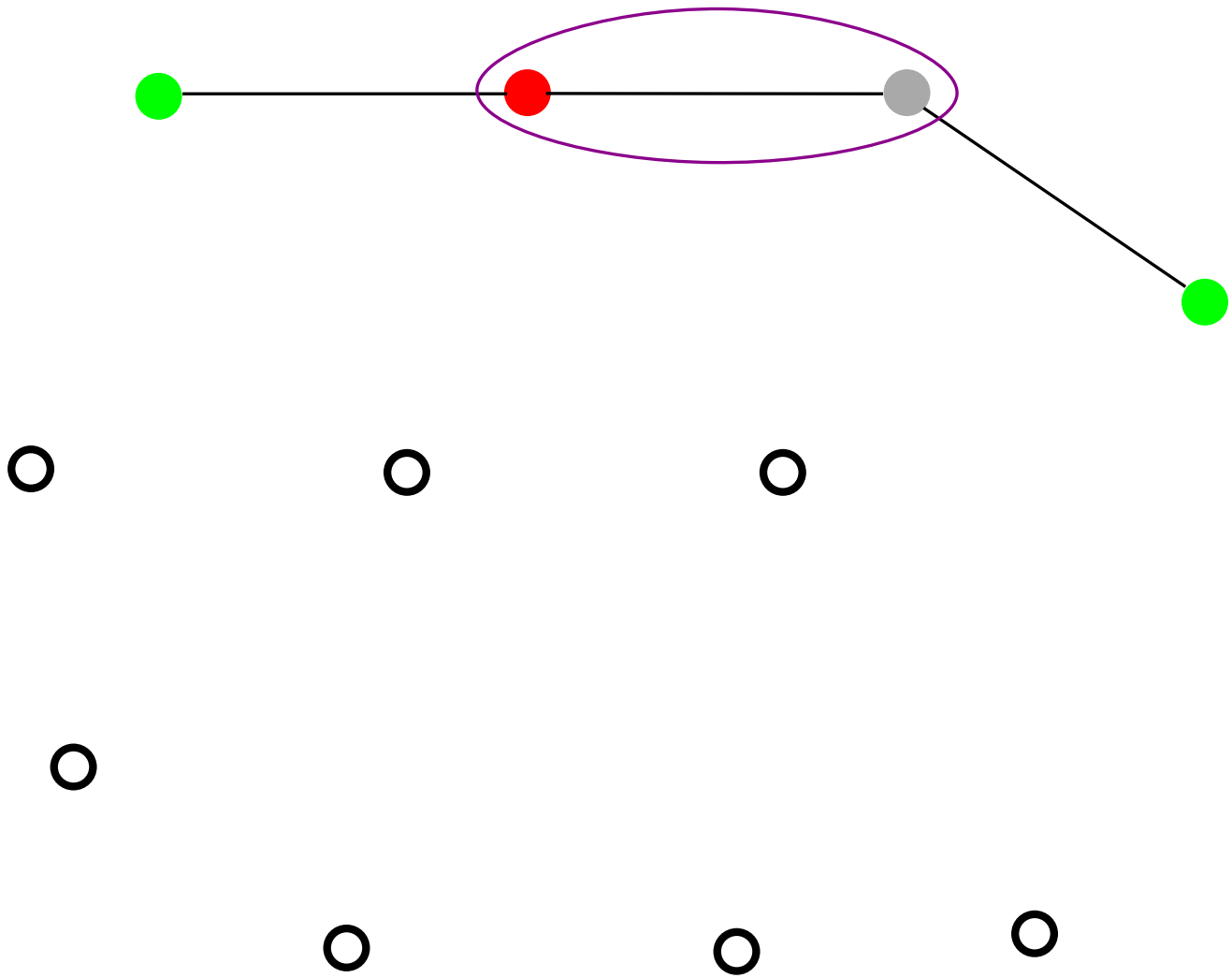
Simple-Global-Line Protocol



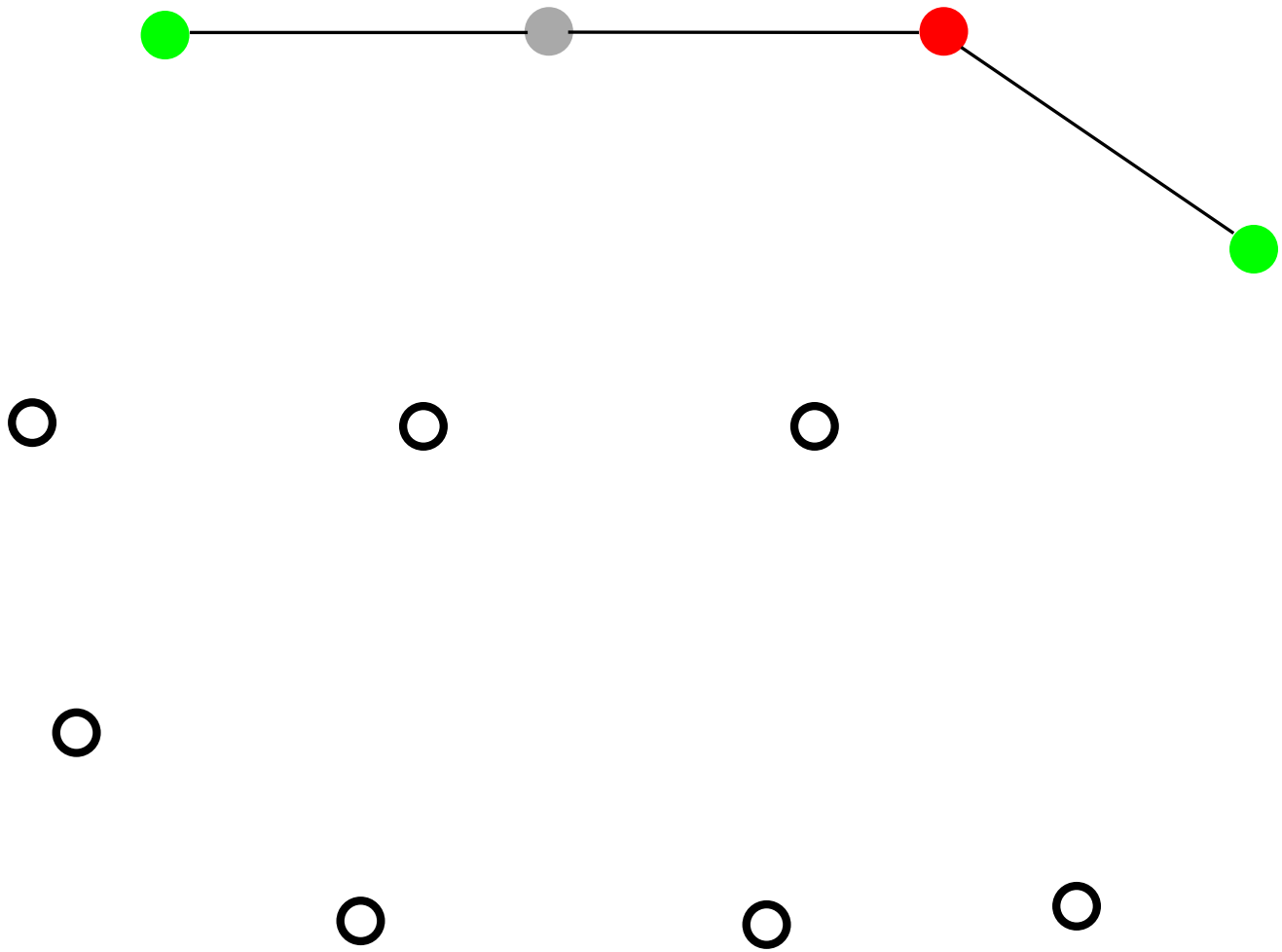
Simple-Global-Line Protocol



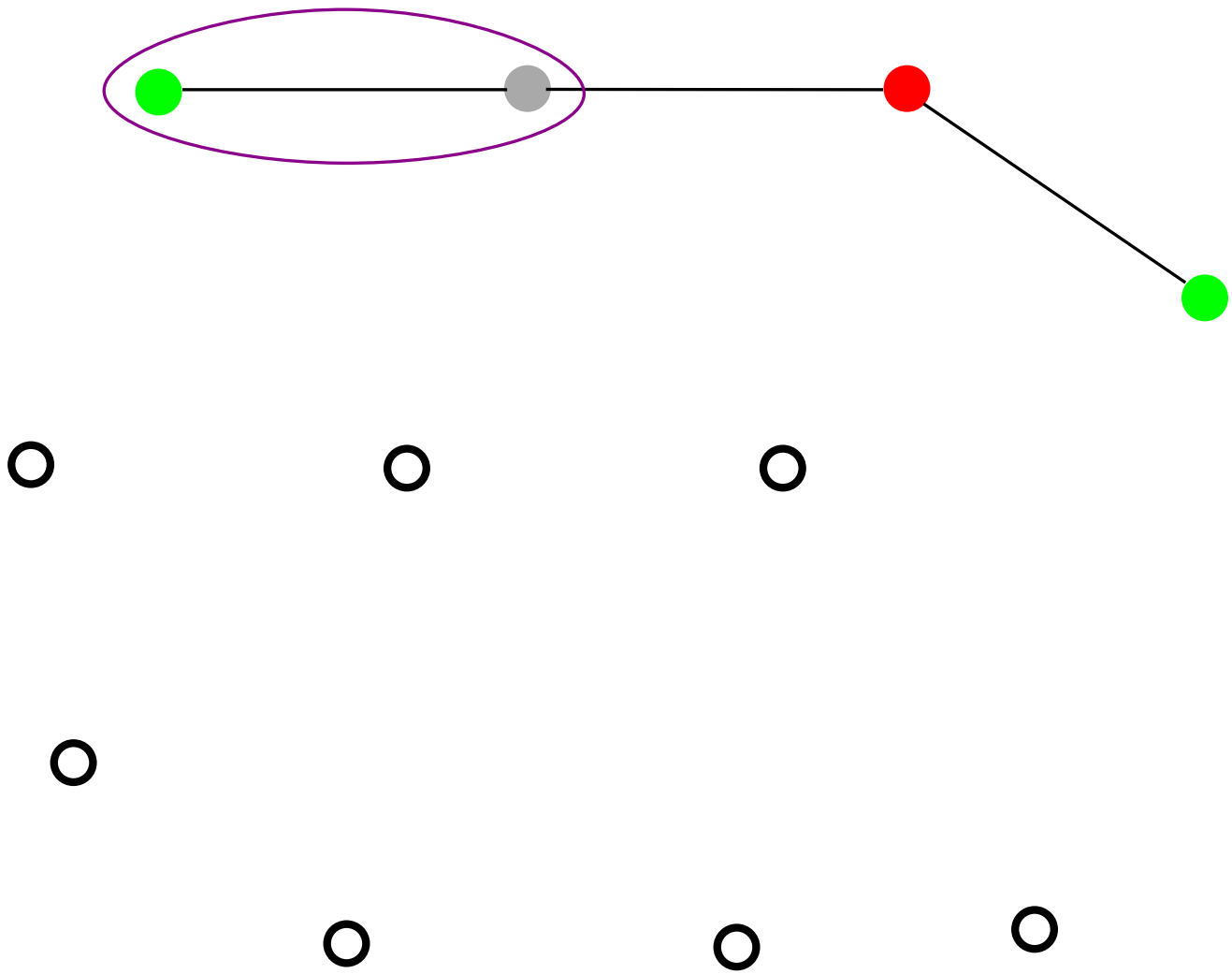
Simple-Global-Line Protocol



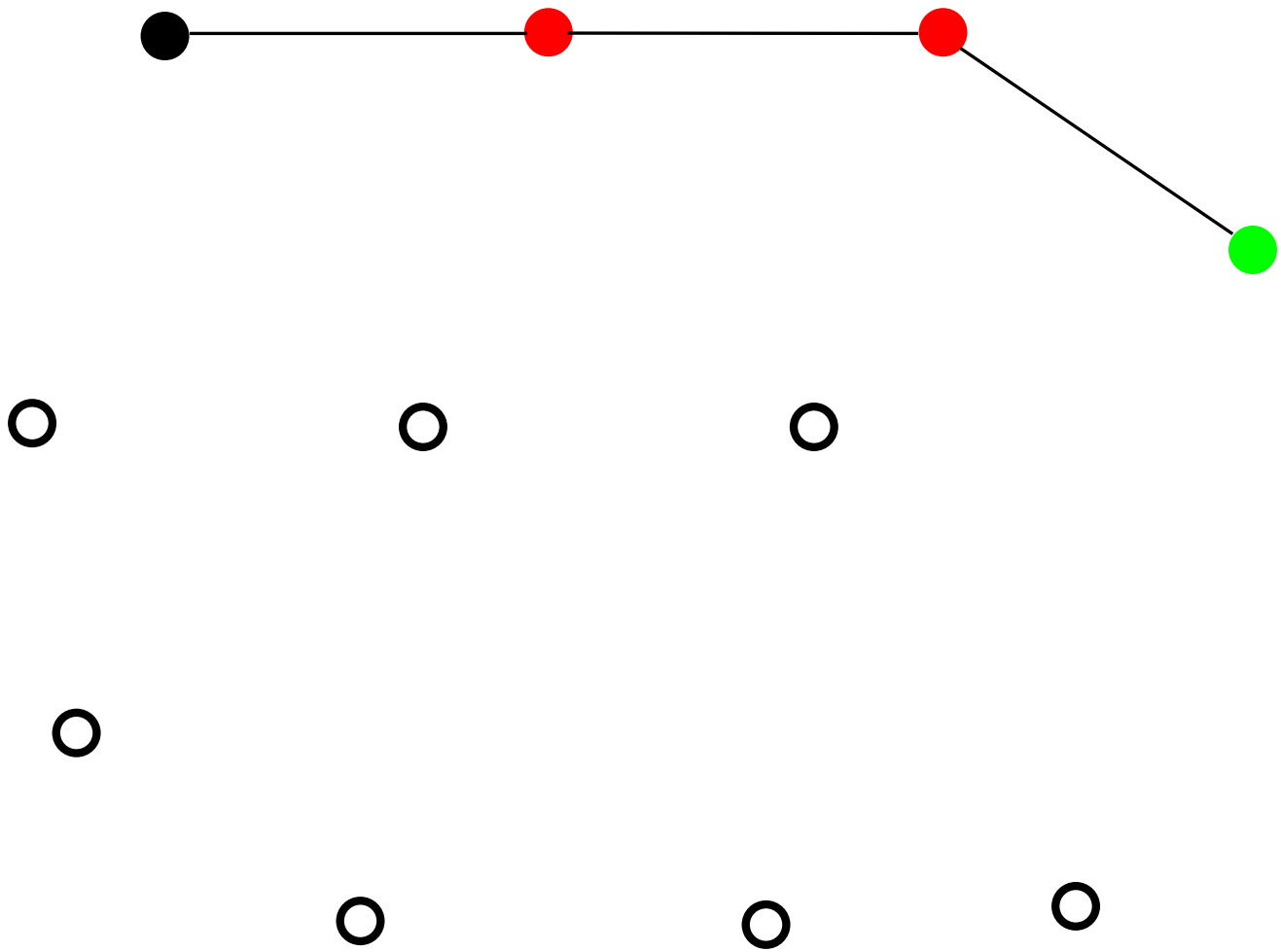
Simple-Global-Line Protocol



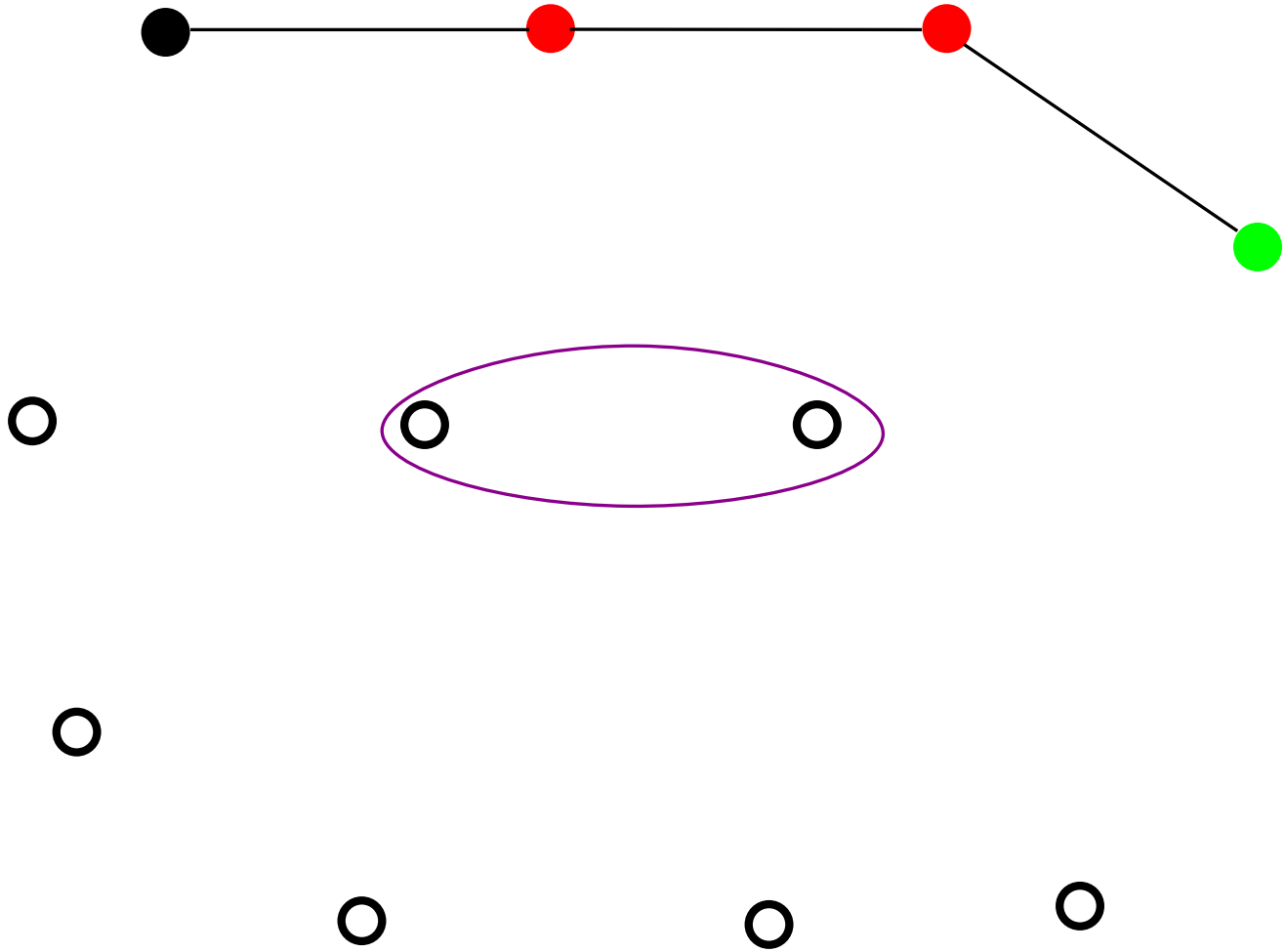
Simple-Global-Line Protocol



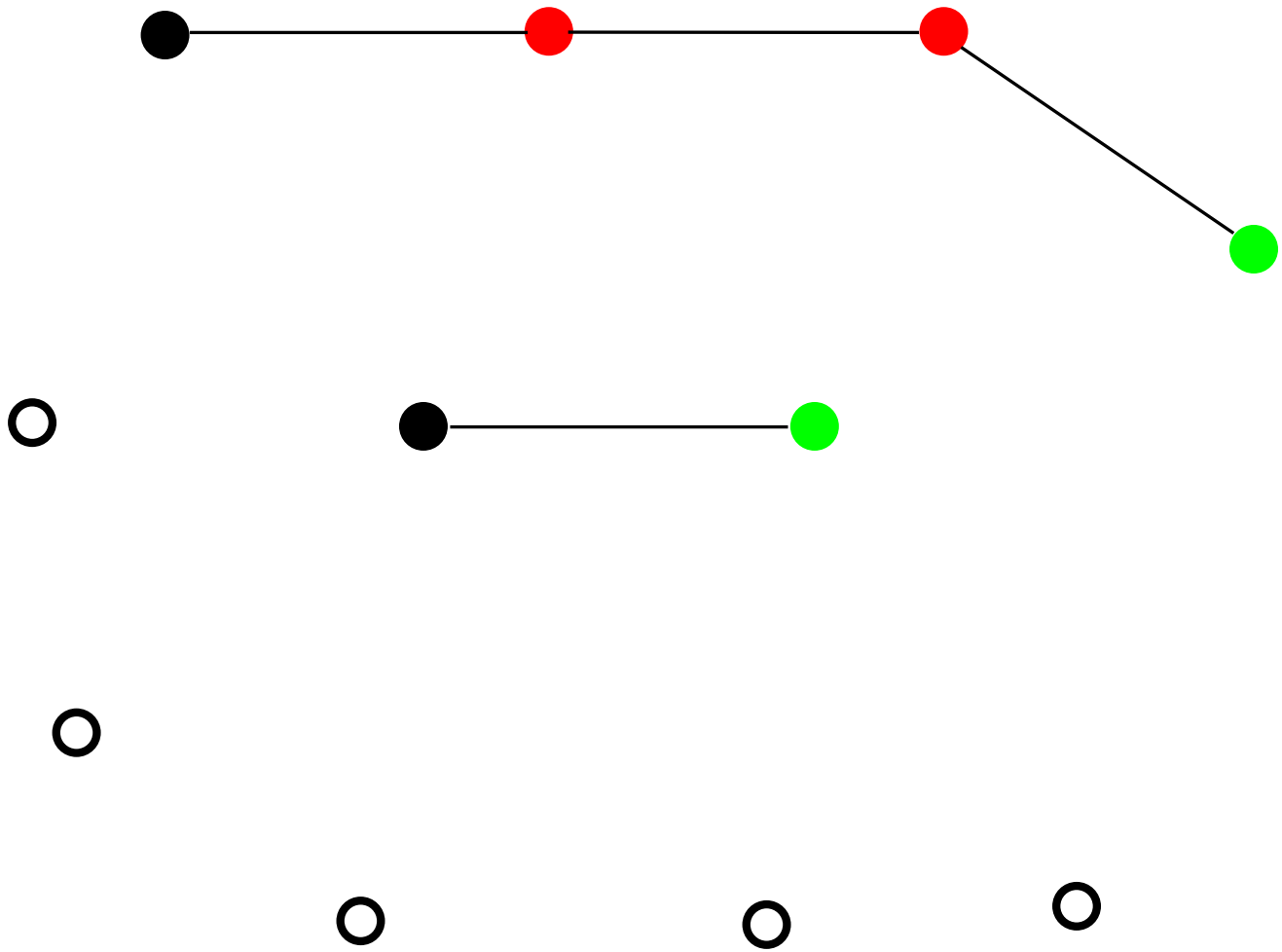
Simple-Global-Line Protocol



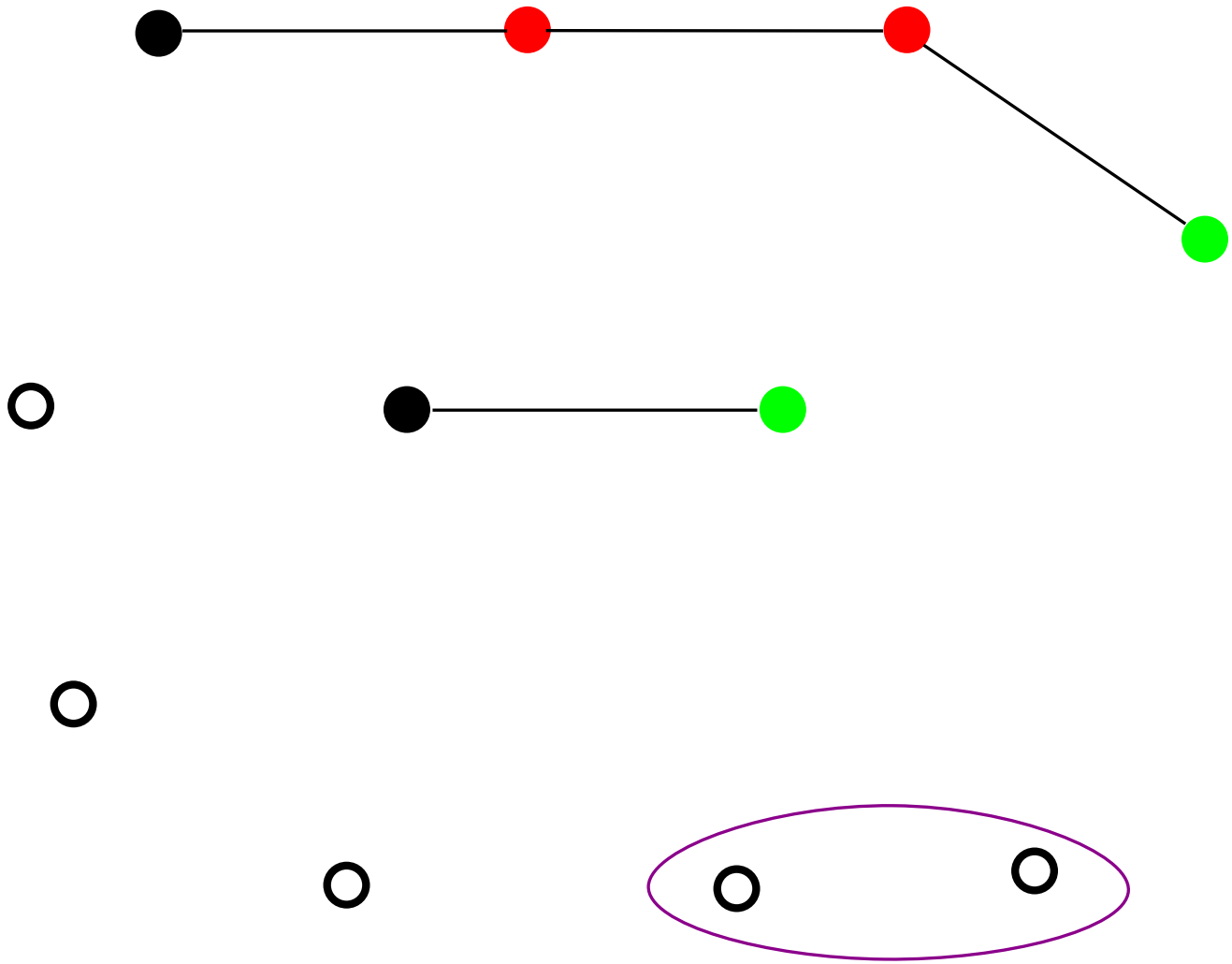
Simple-Global-Line Protocol



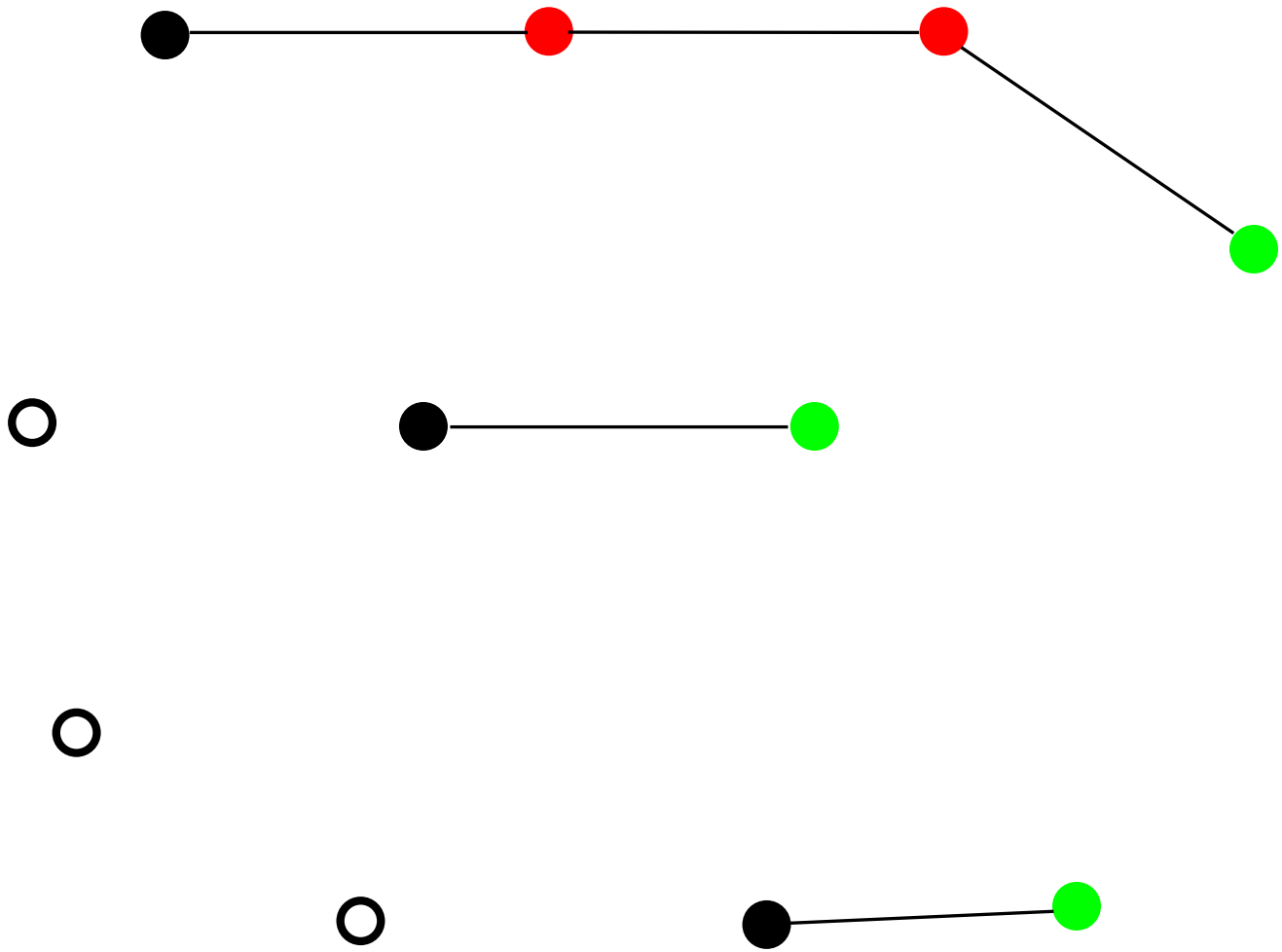
Simple-Global-Line Protocol



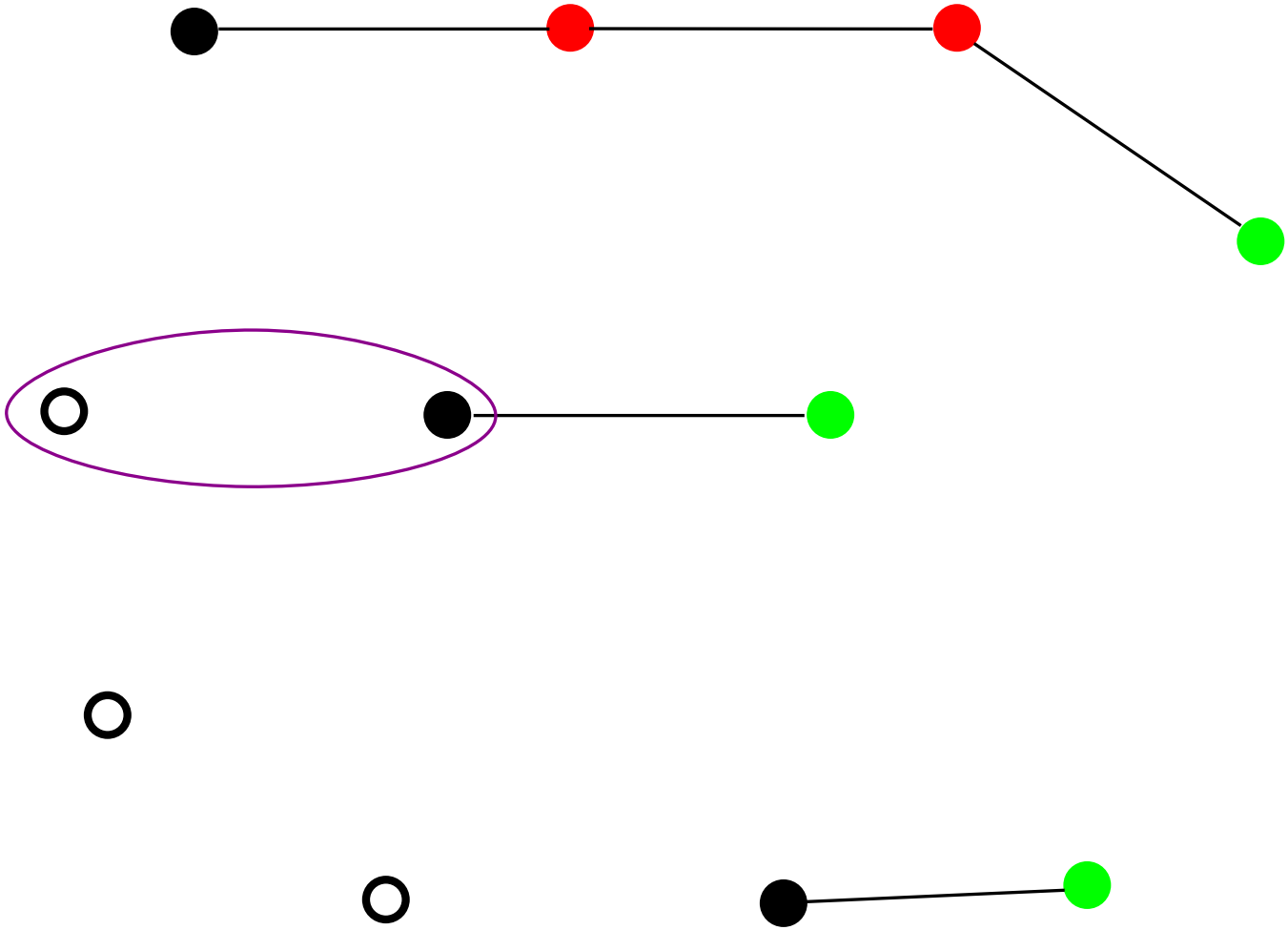
Simple-Global-Line Protocol



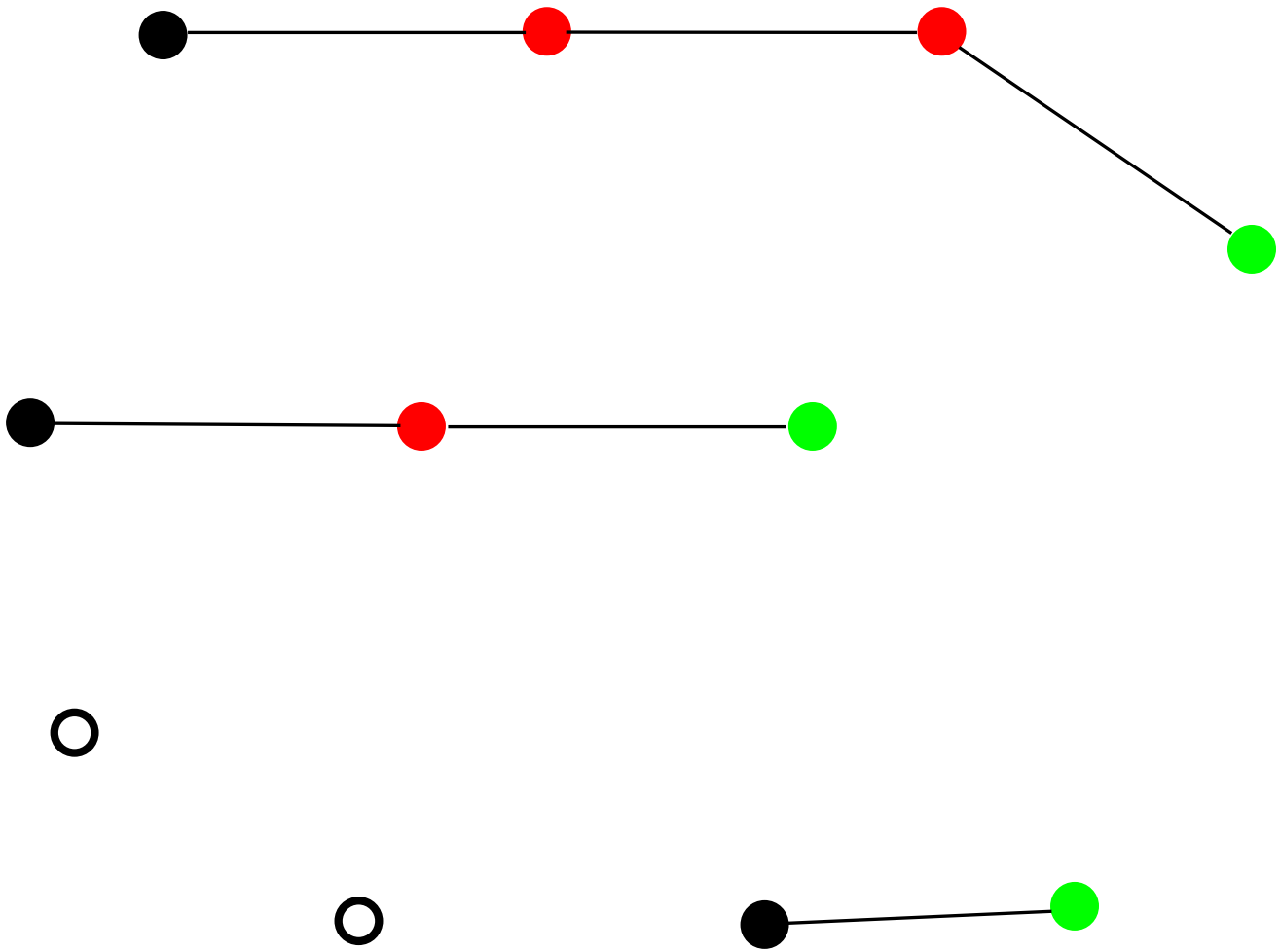
Simple-Global-Line Protocol



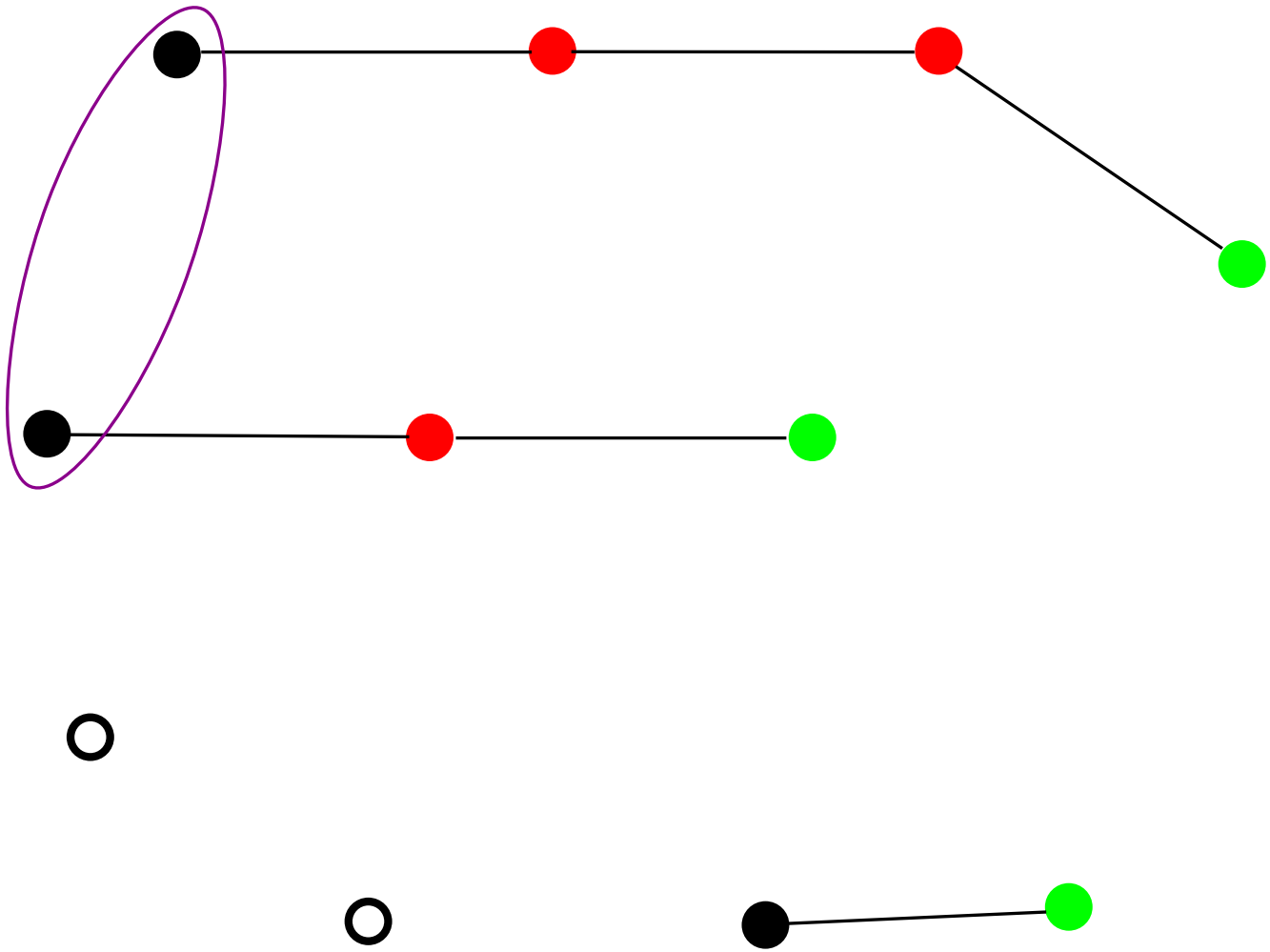
Simple-Global-Line Protocol



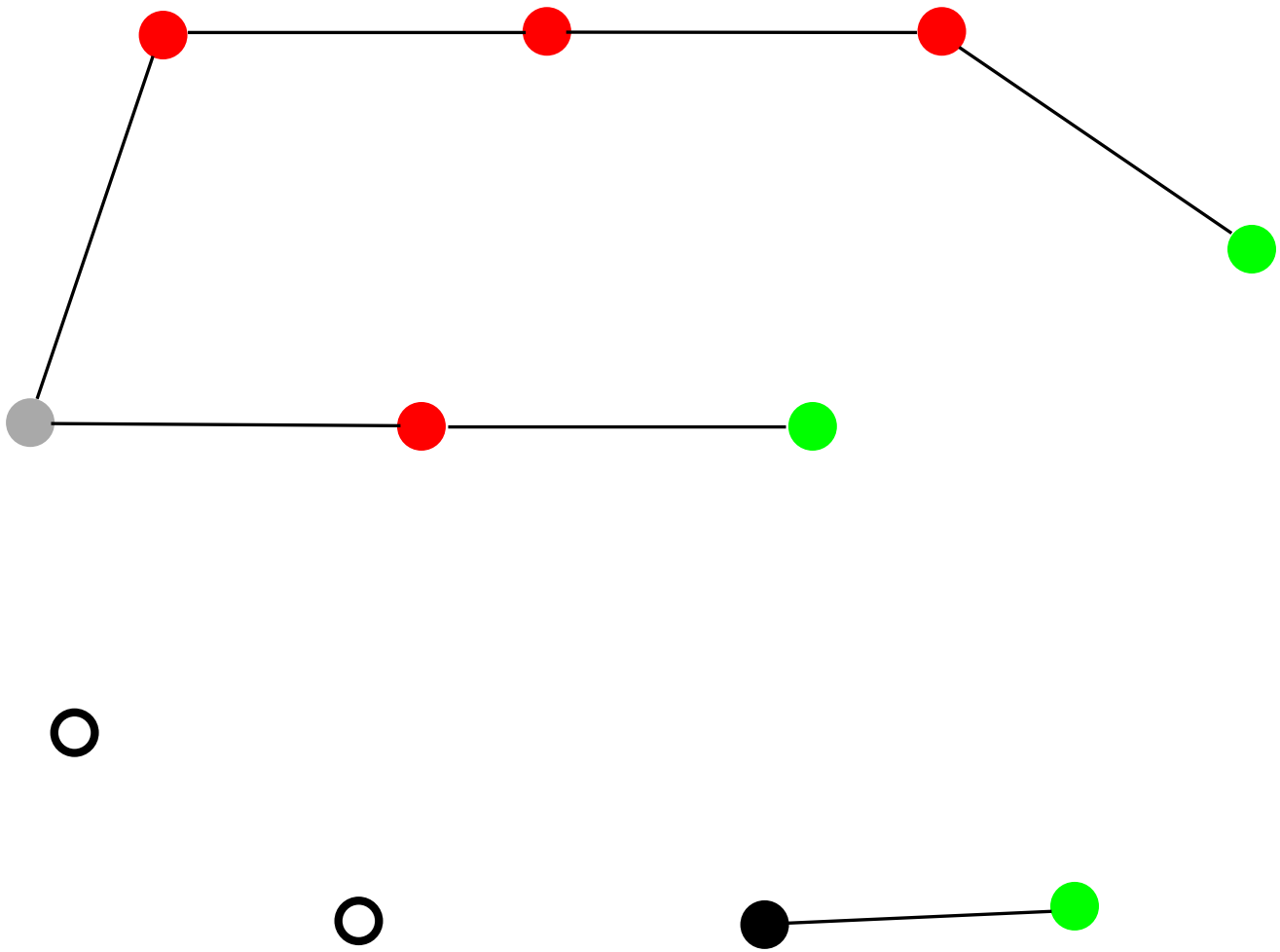
Simple-Global-Line Protocol



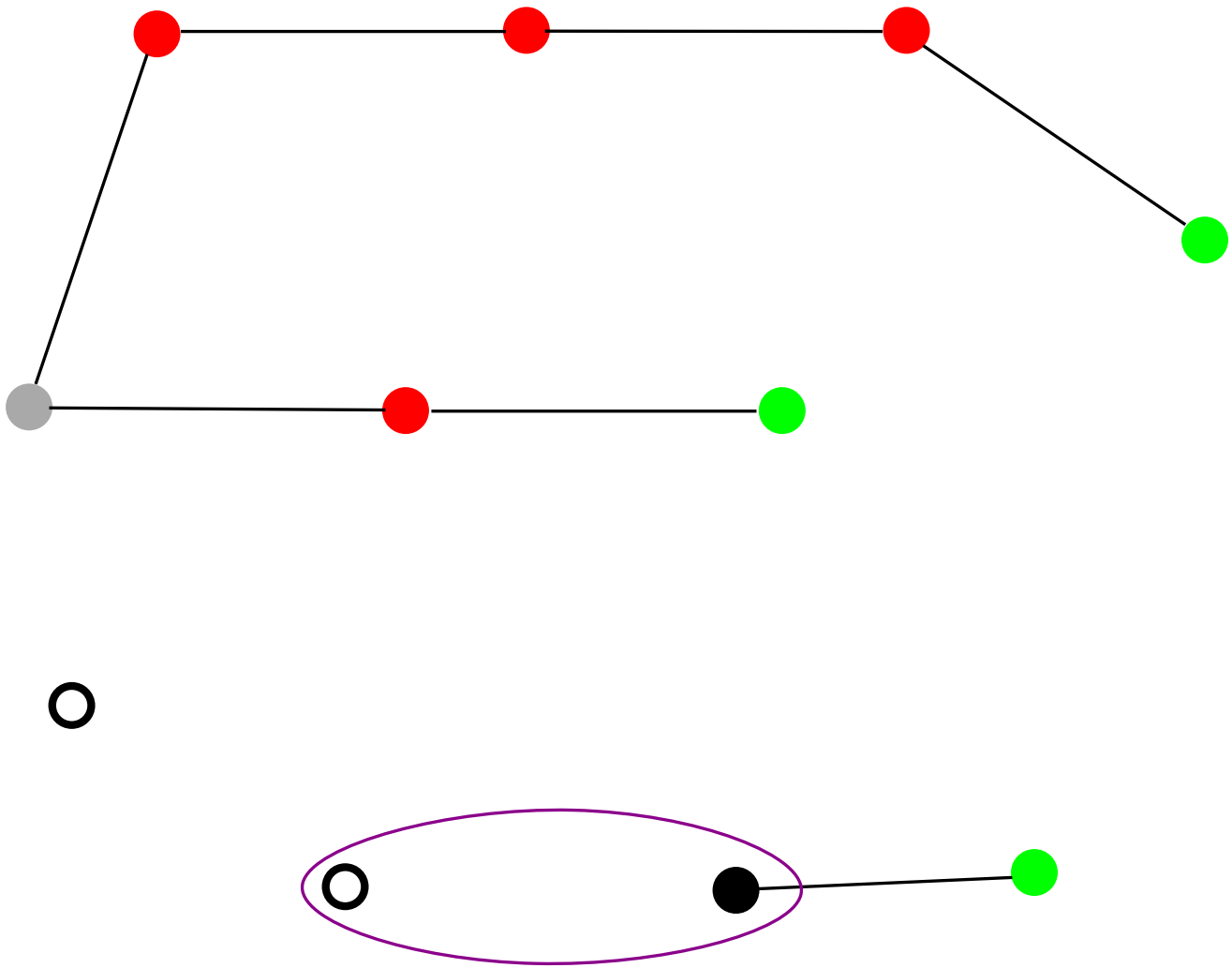
Simple-Global-Line Protocol



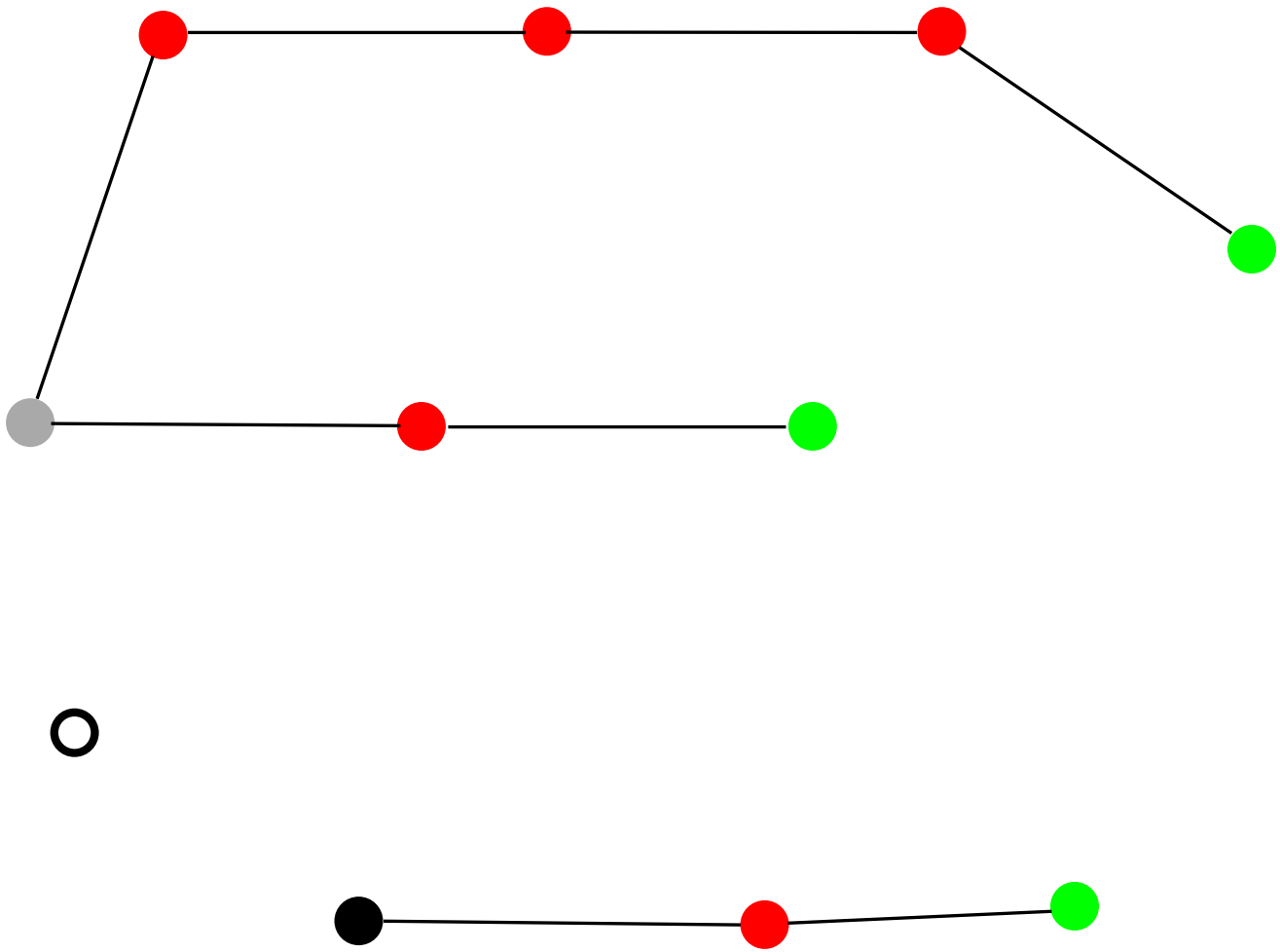
Simple-Global-Line Protocol



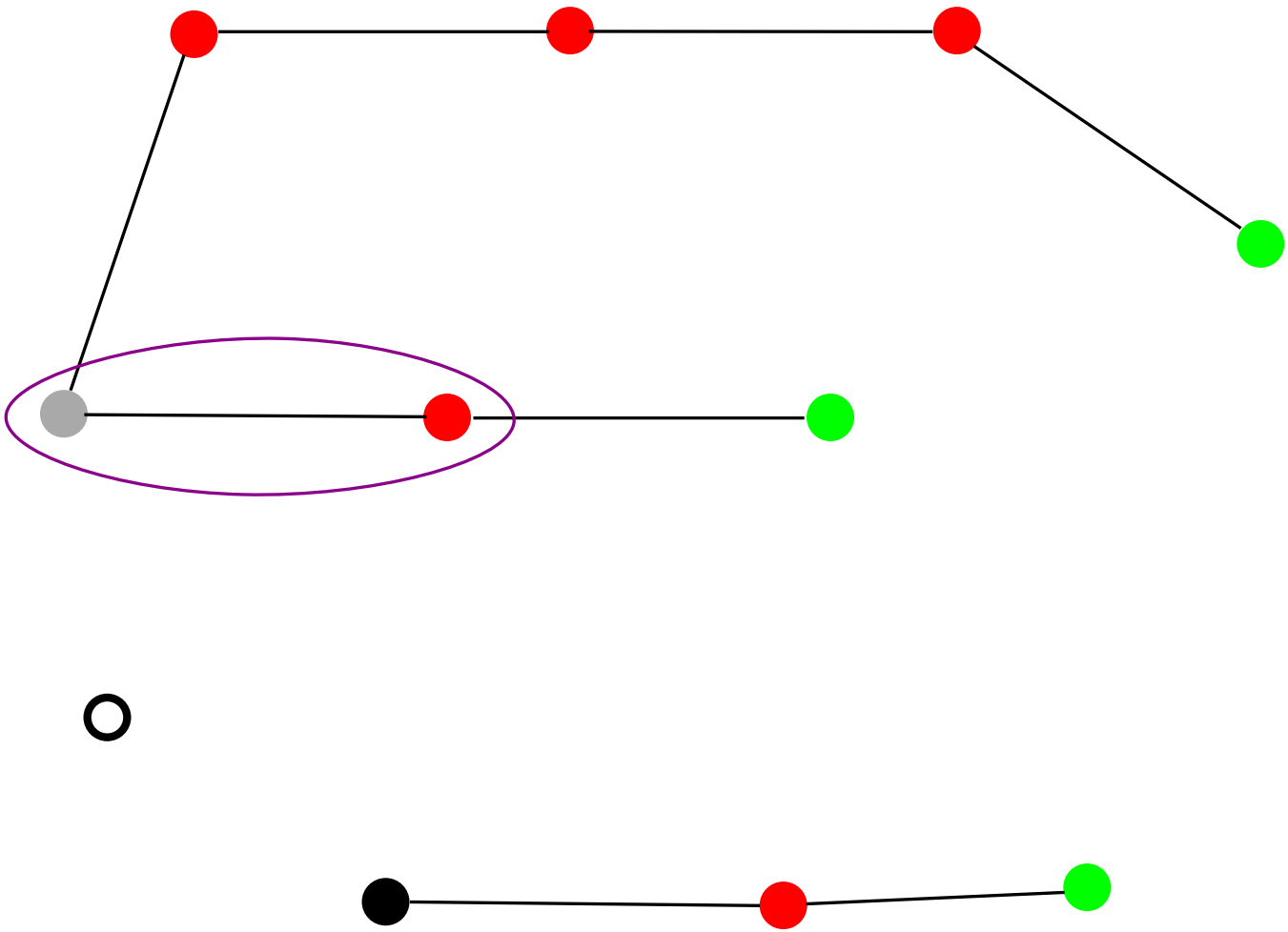
Simple-Global-Line Protocol



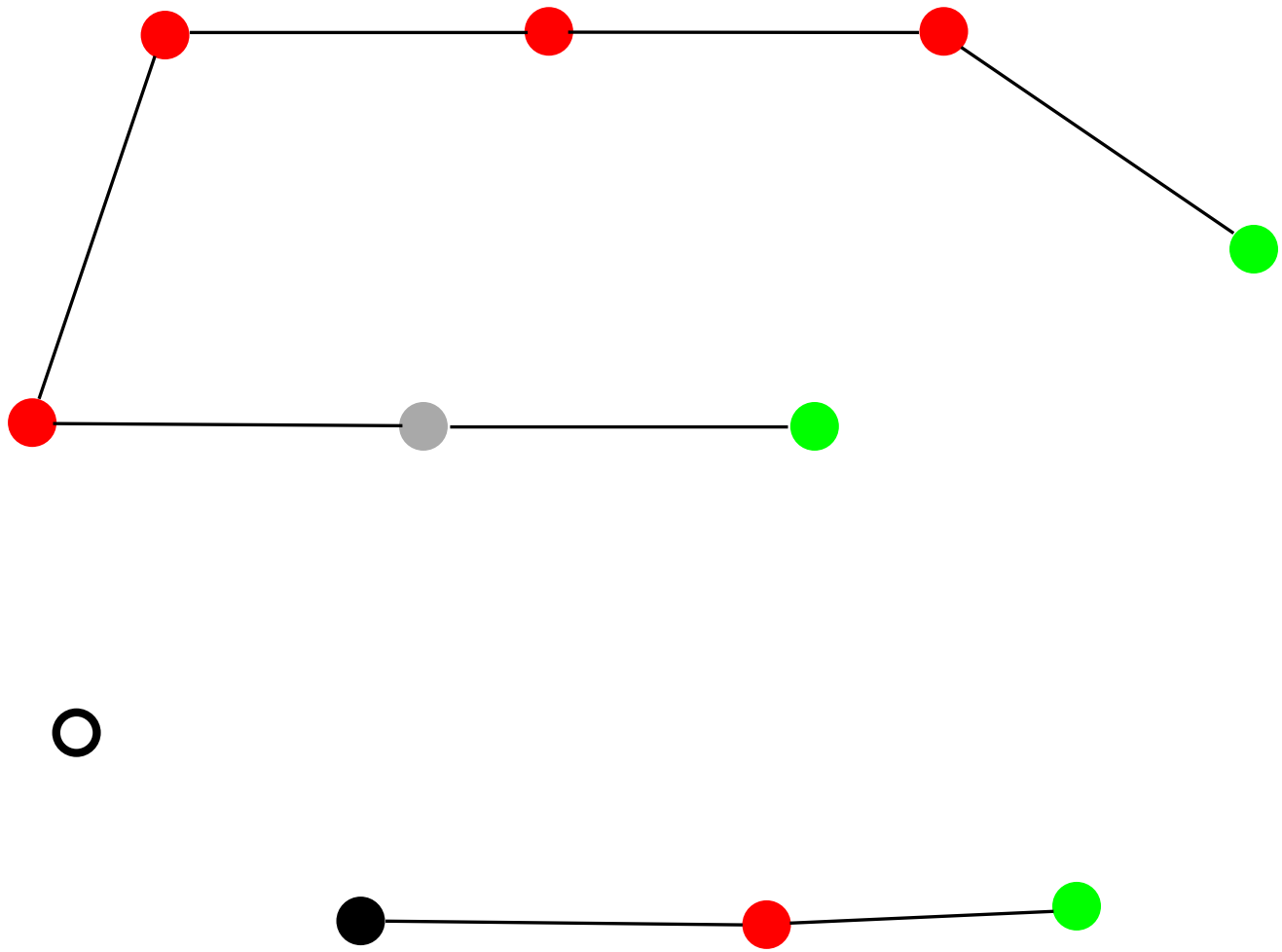
Simple-Global-Line Protocol



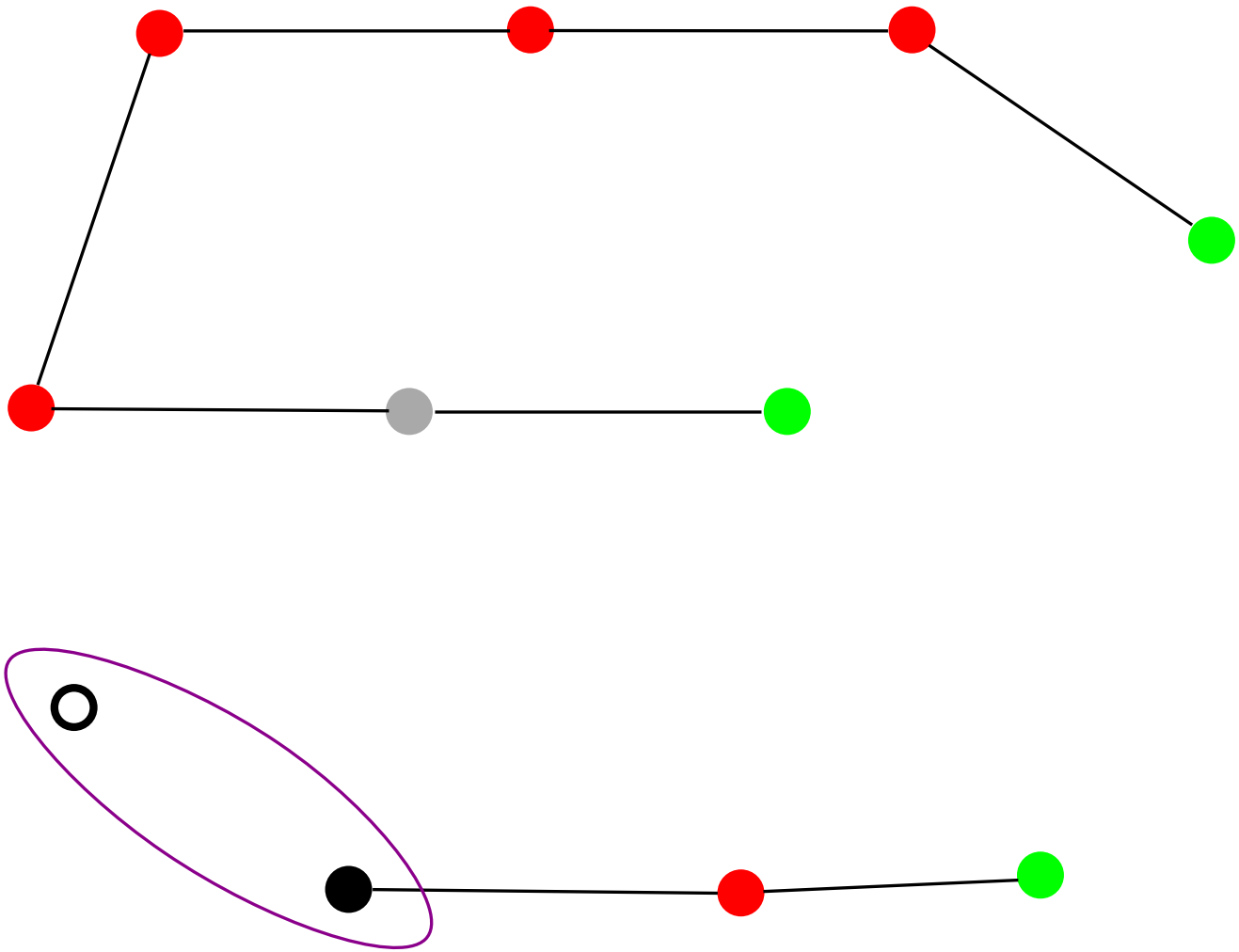
Simple-Global-Line Protocol



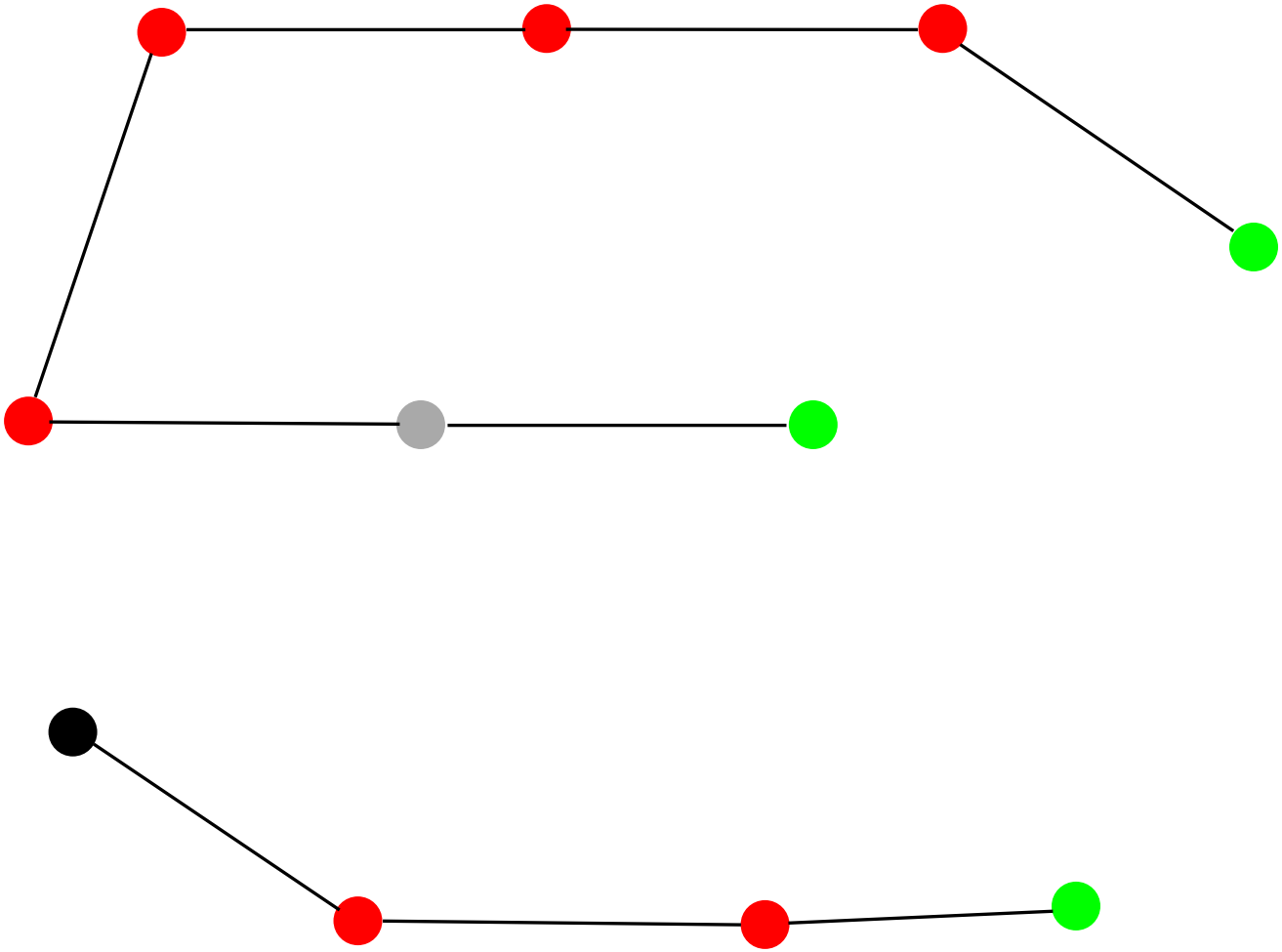
Simple-Global-Line Protocol



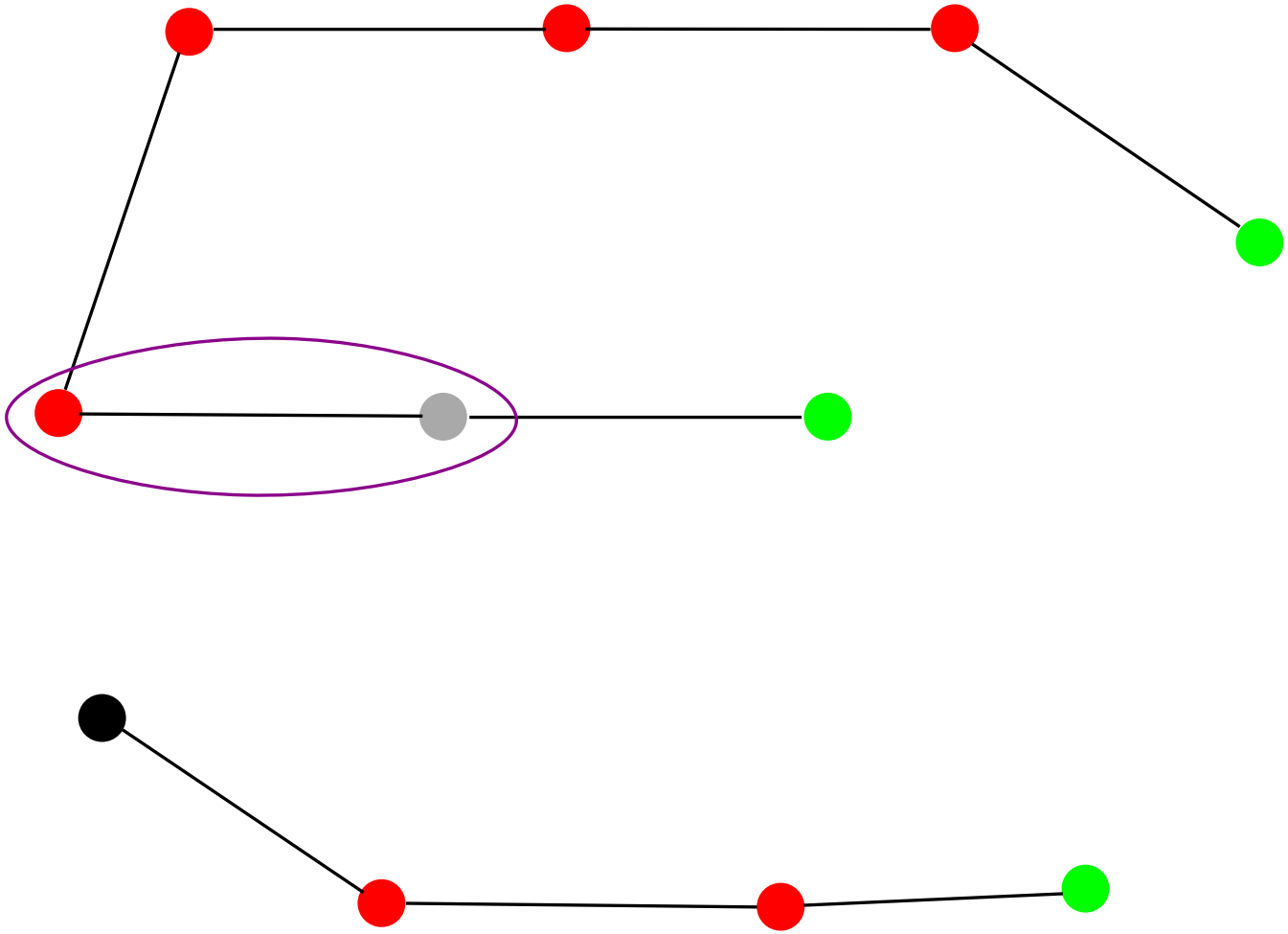
Simple-Global-Line Protocol



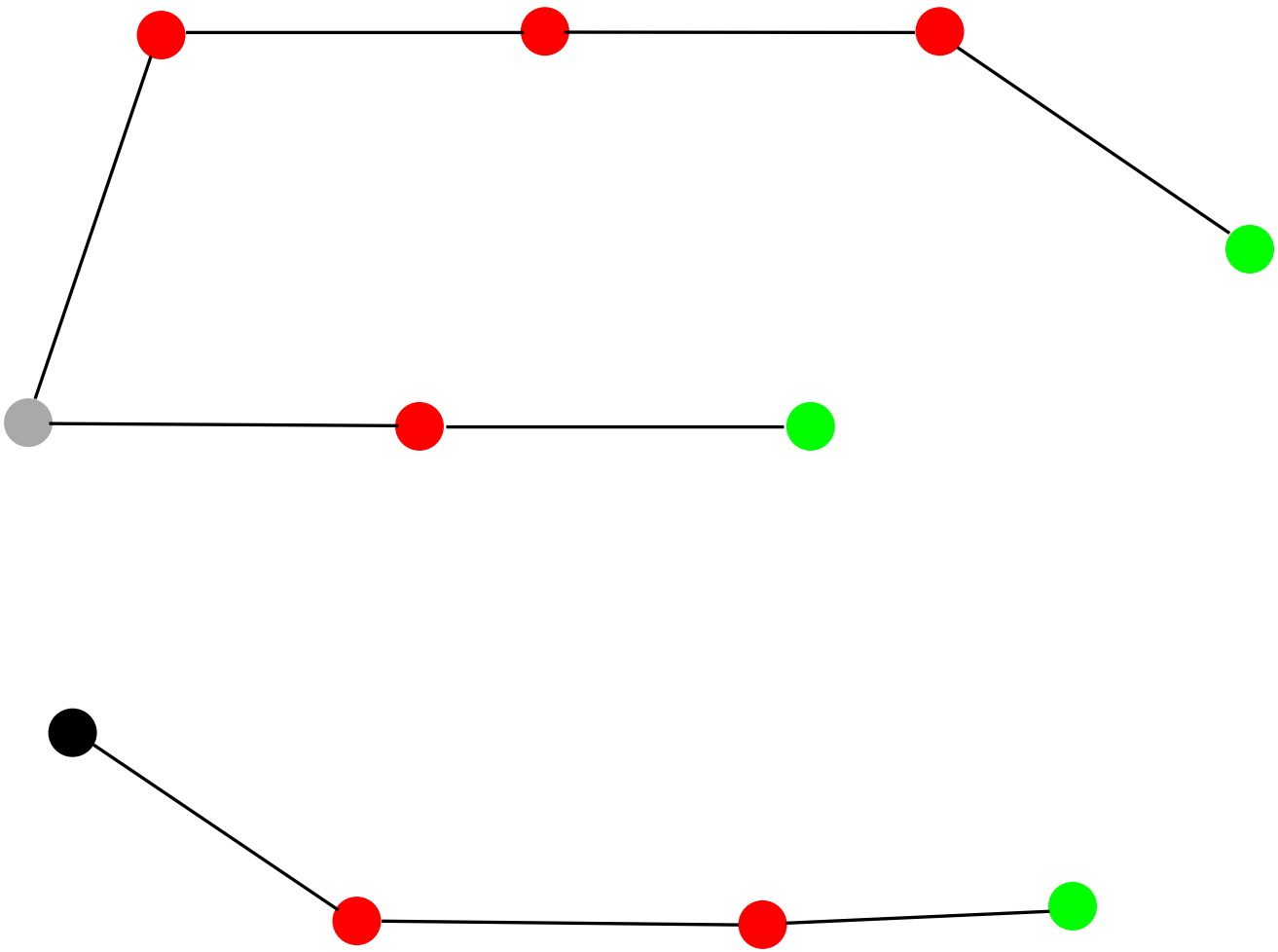
Simple-Global-Line Protocol



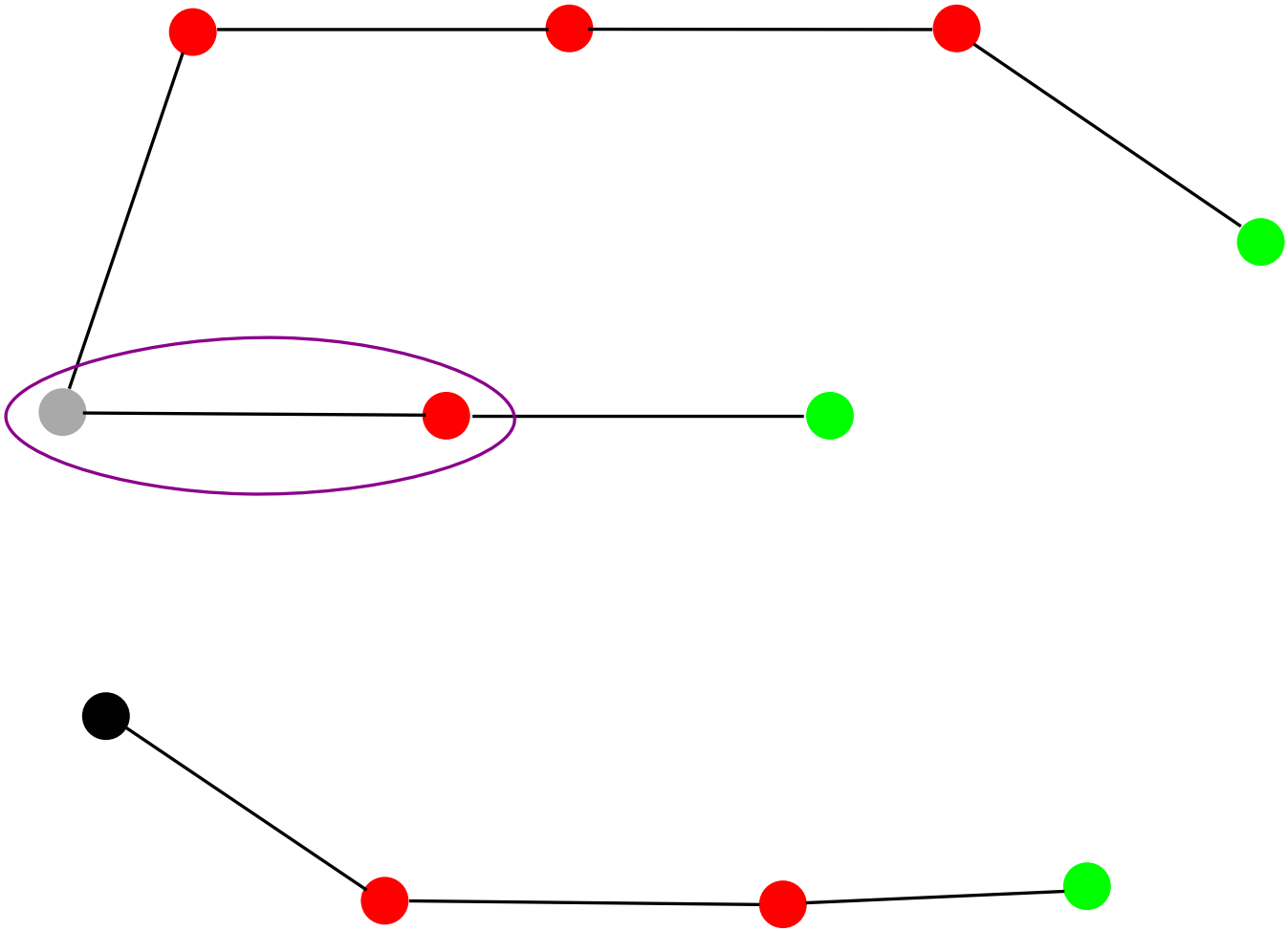
Simple-Global-Line Protocol



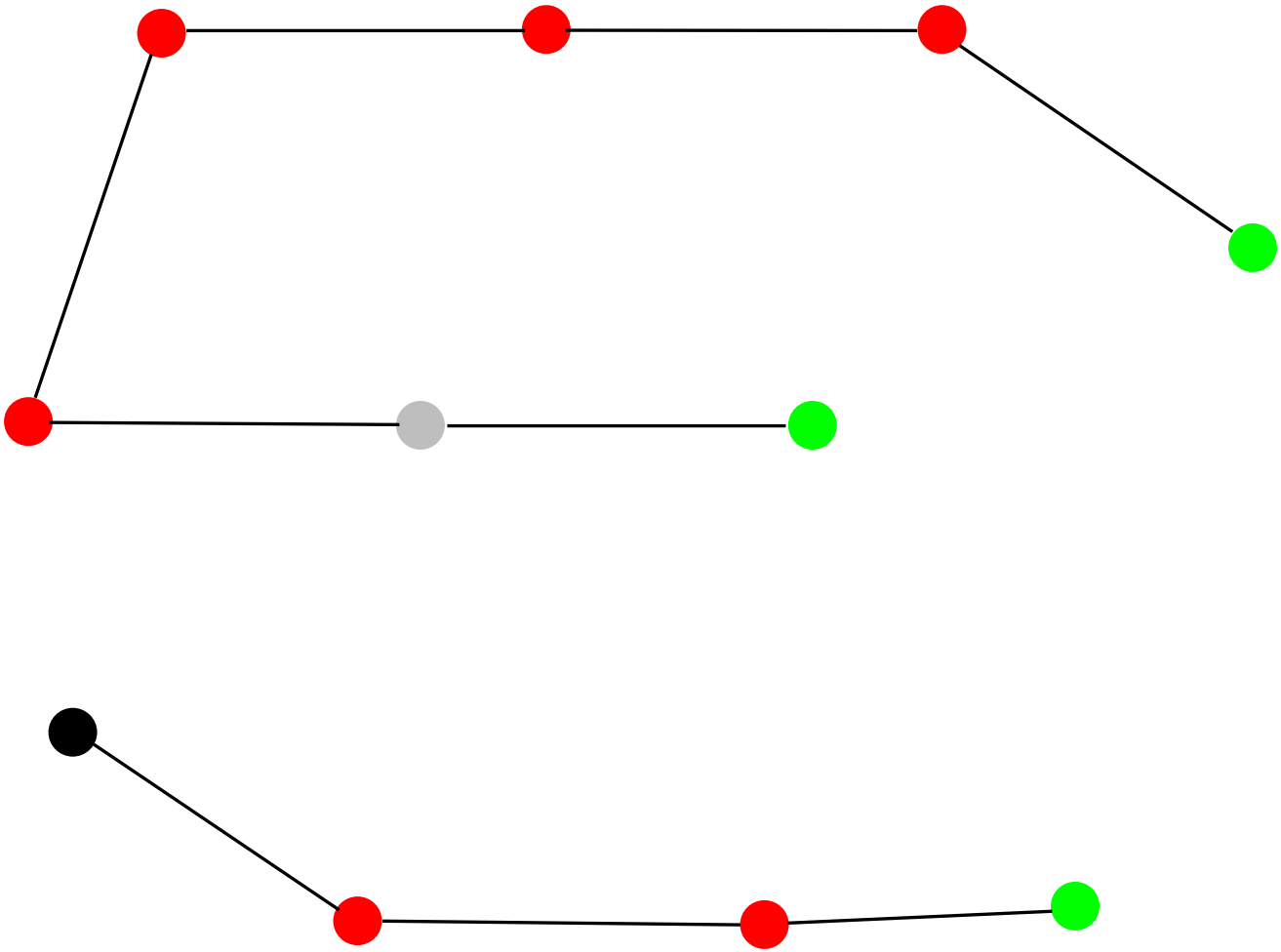
Simple-Global-Line Protocol



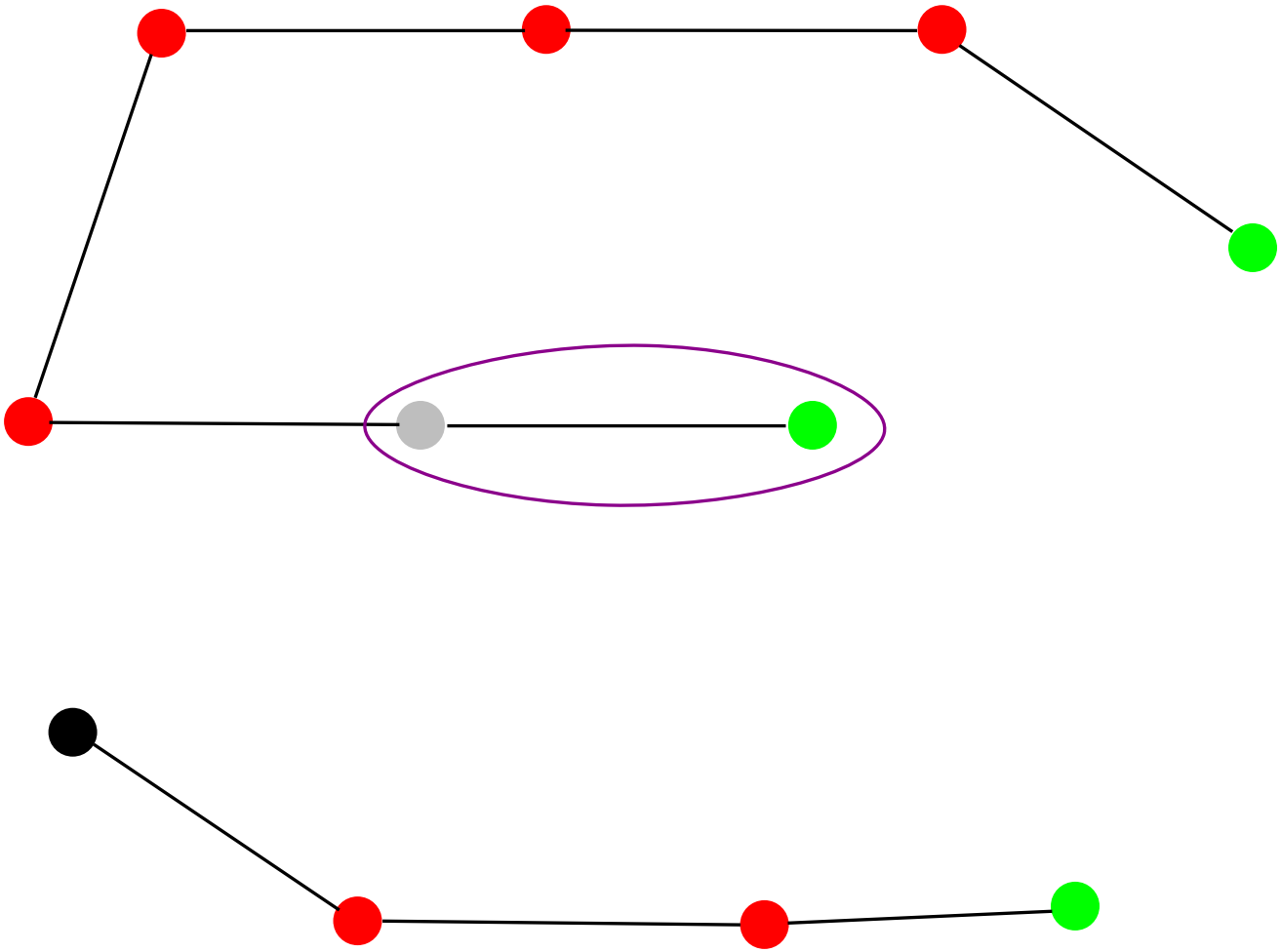
Simple-Global-Line Protocol



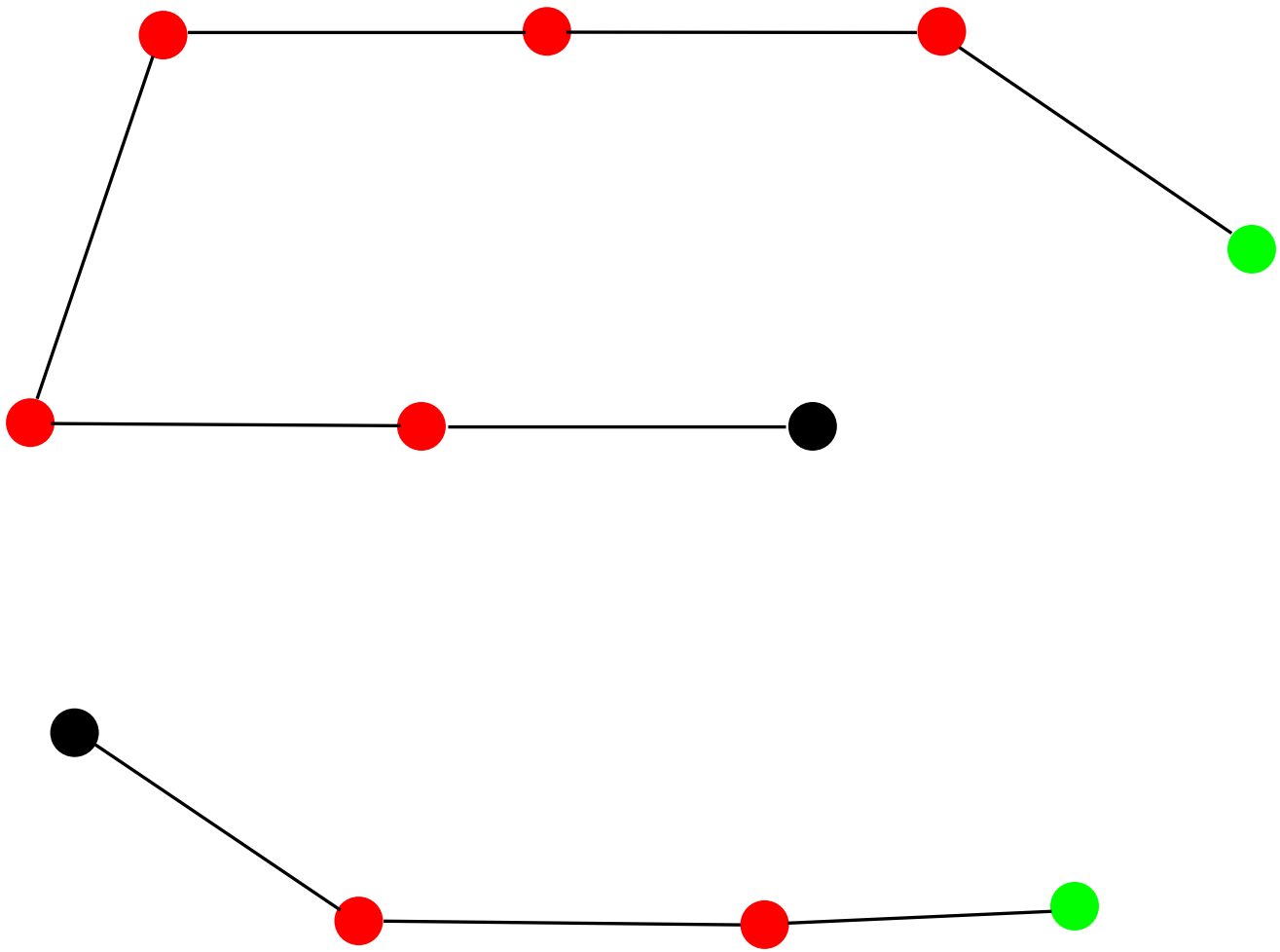
Simple-Global-Line Protocol



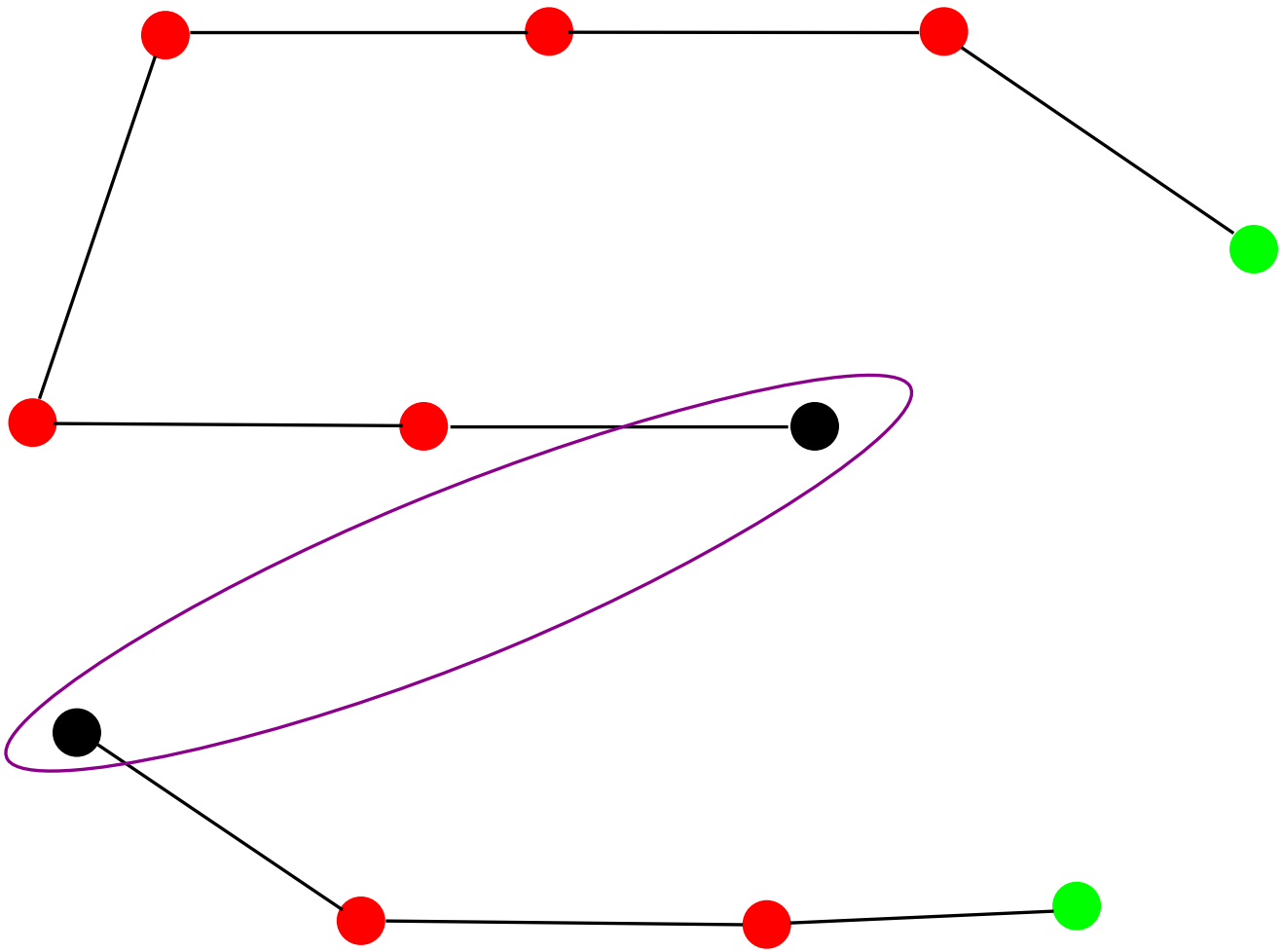
Simple-Global-Line Protocol



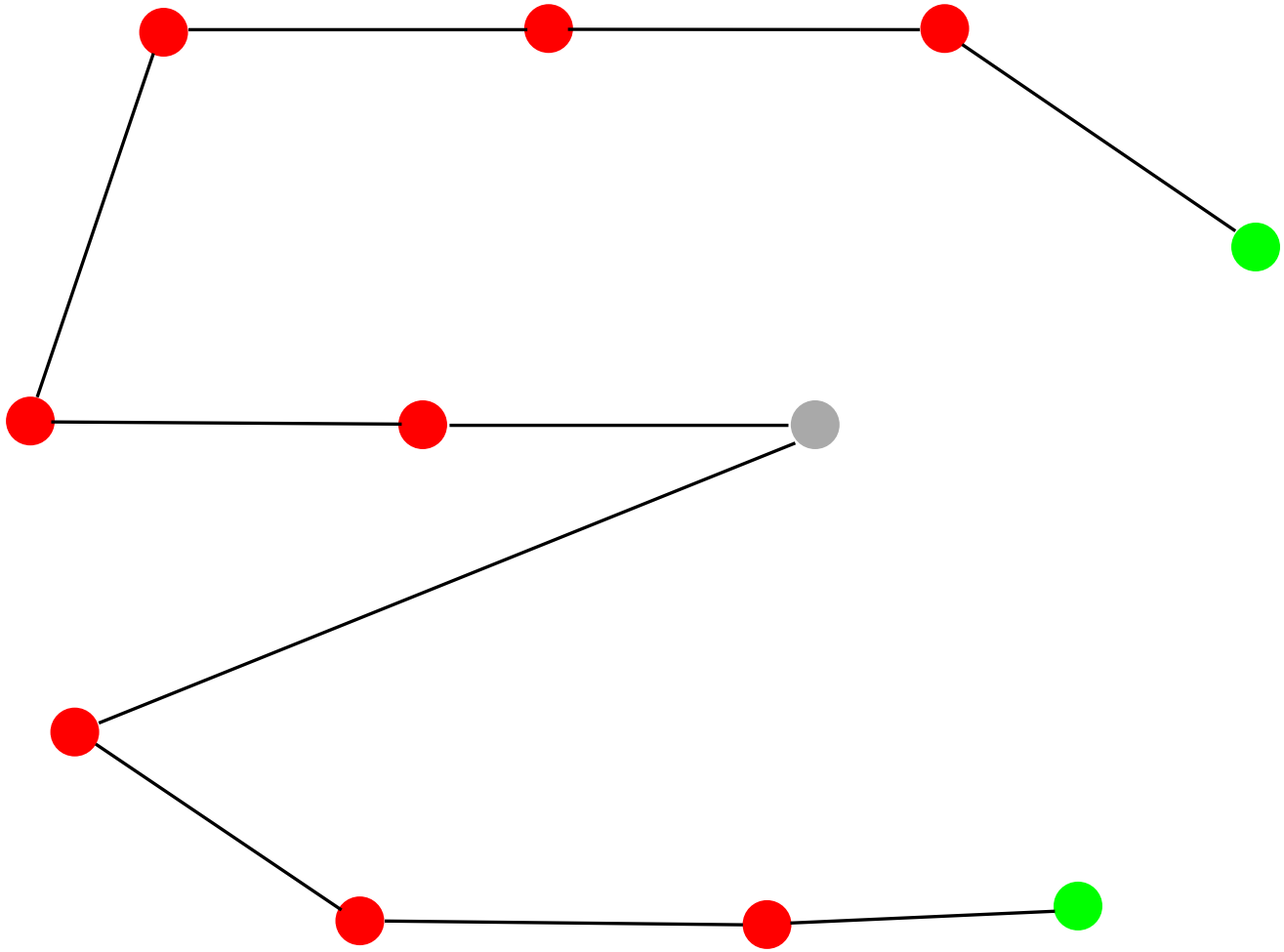
Simple-Global-Line Protocol



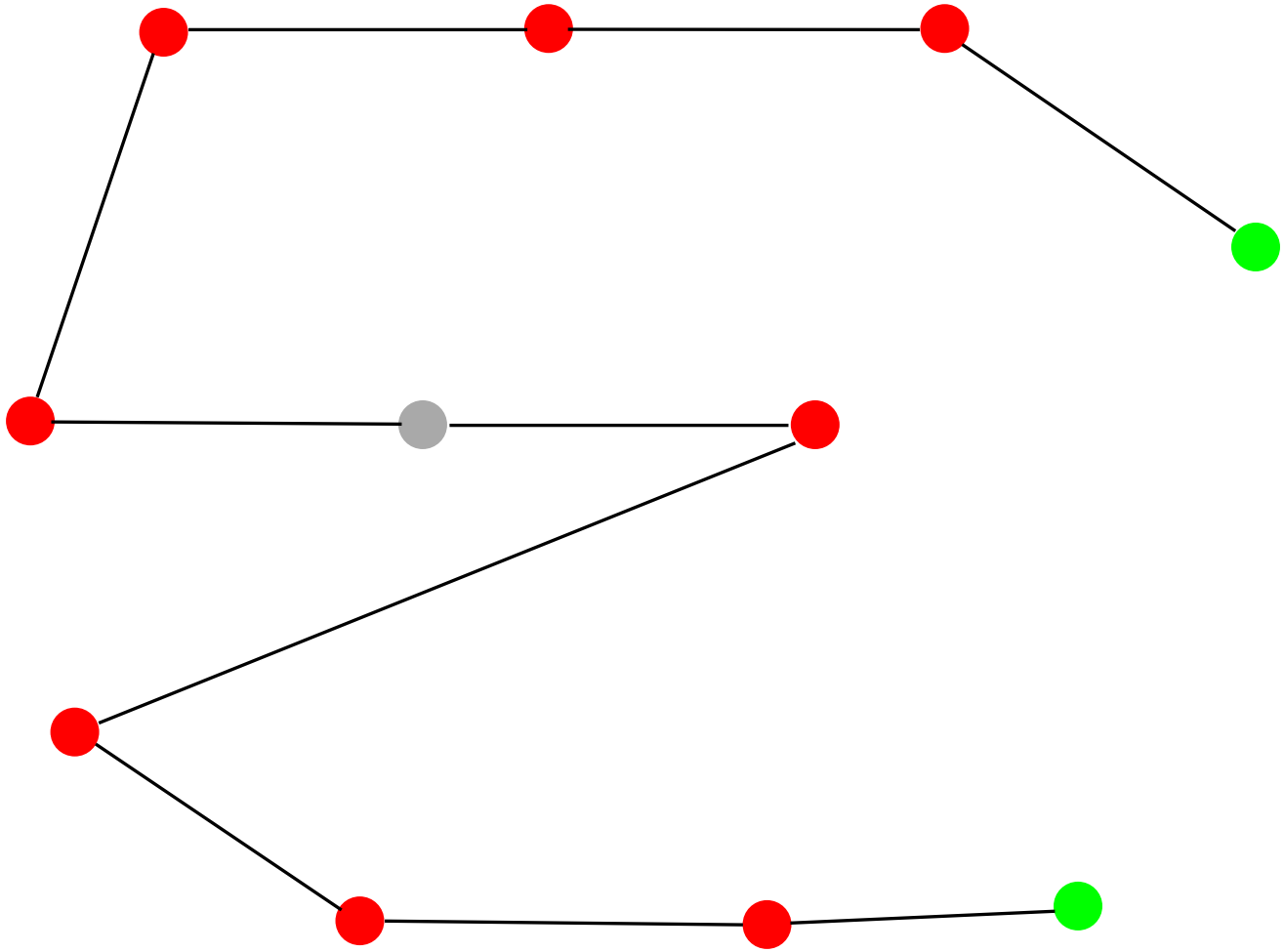
Simple-Global-Line Protocol



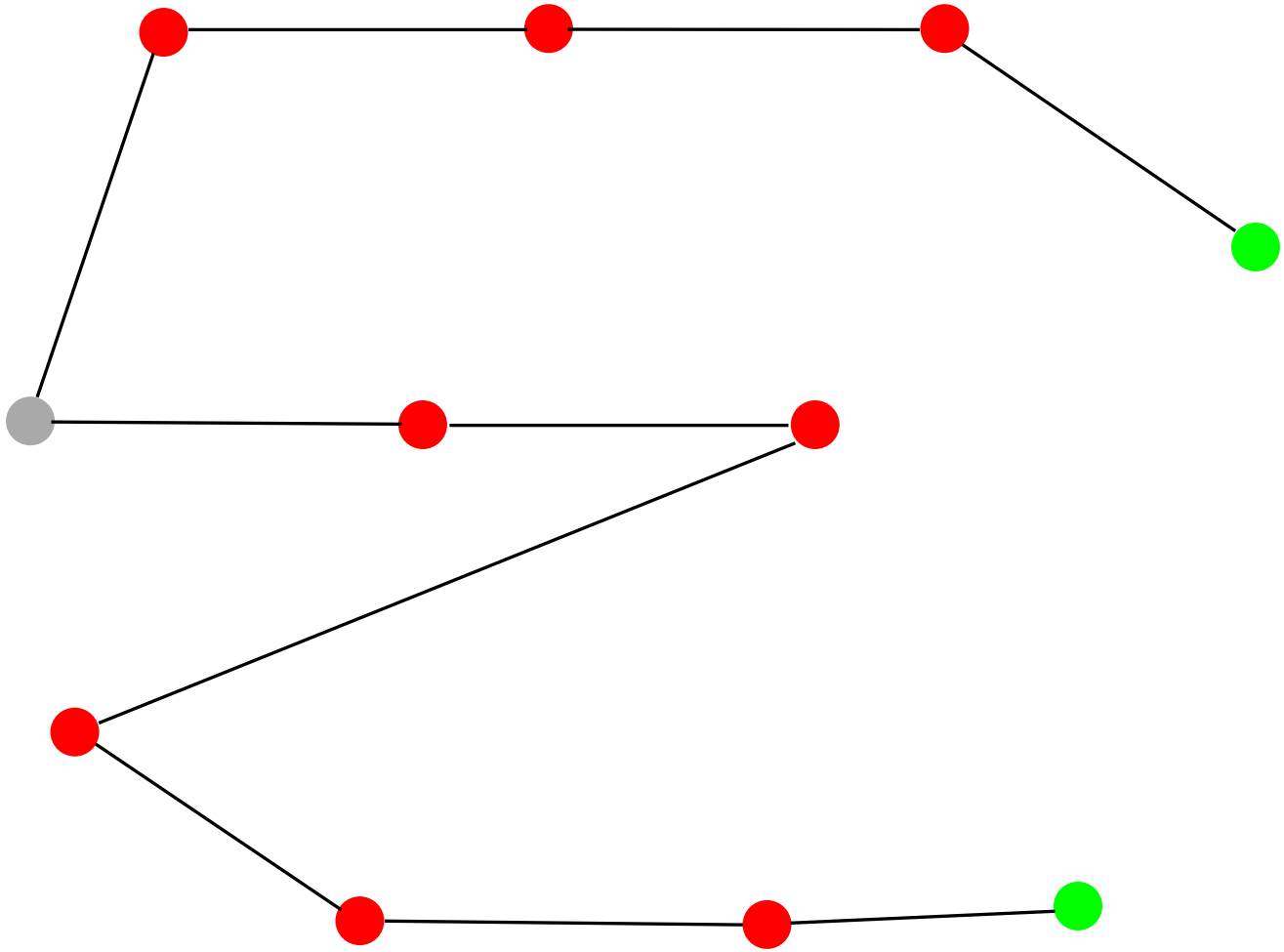
Simple-Global-Line Protocol



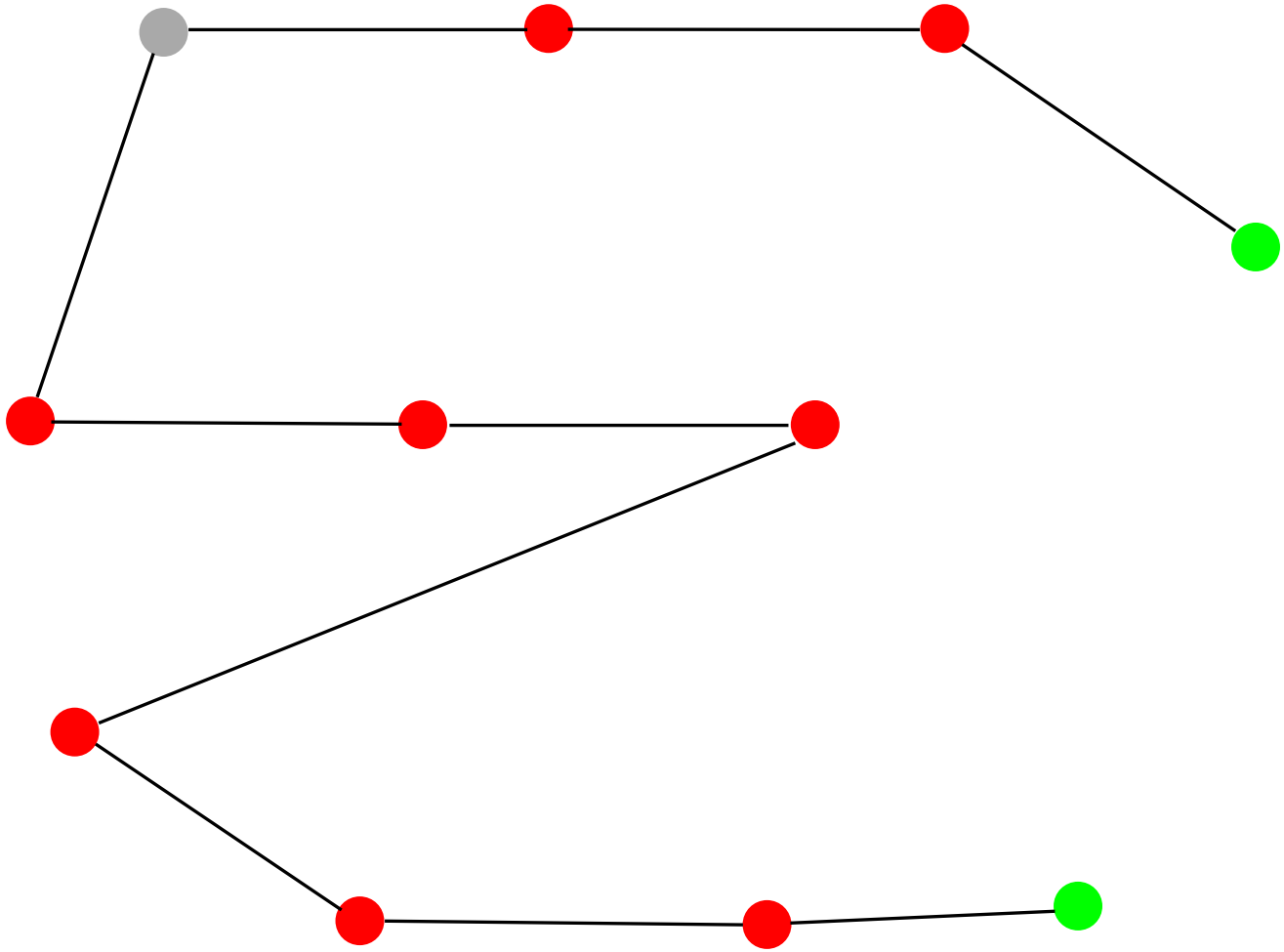
Simple-Global-Line Protocol



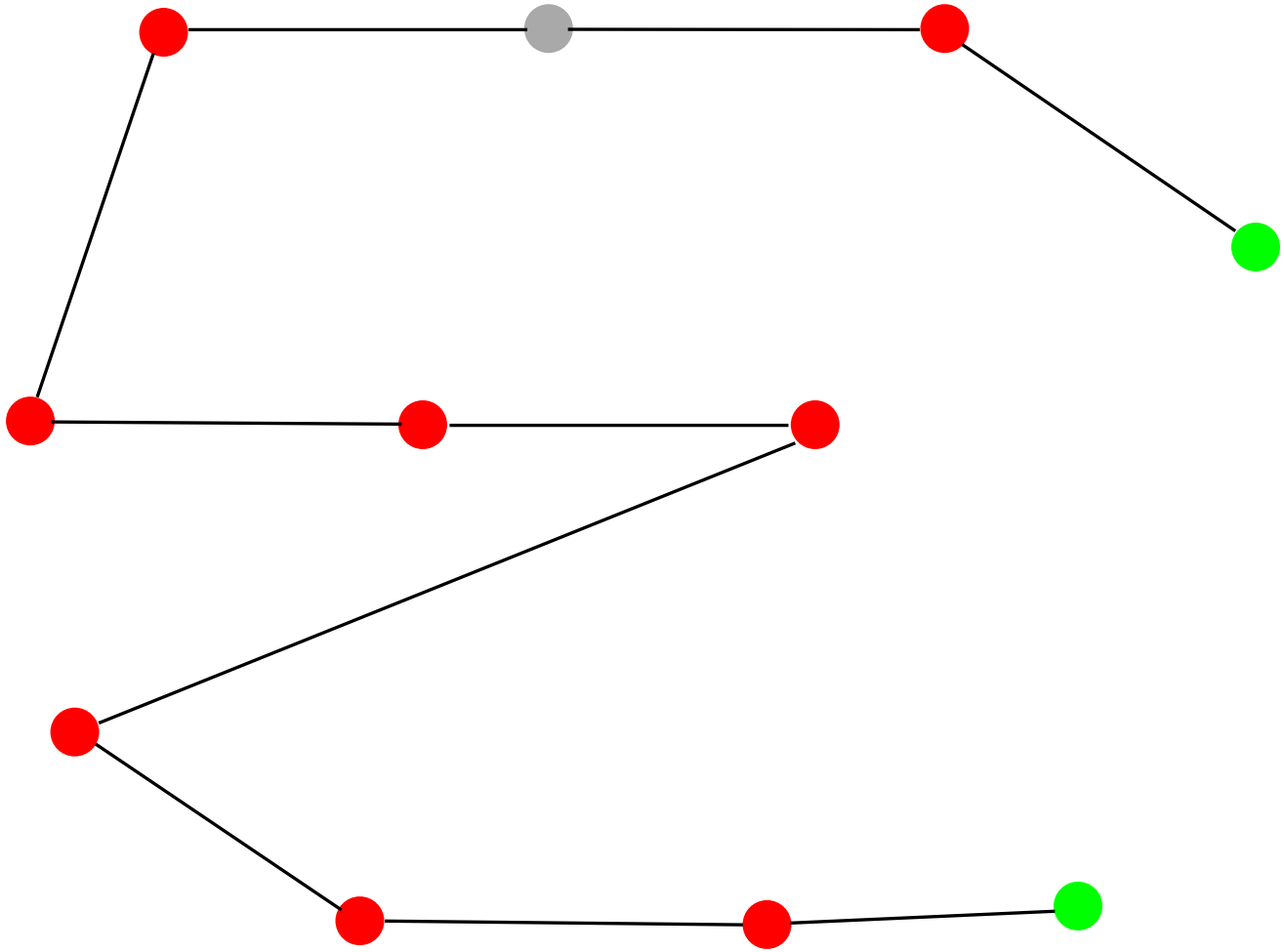
Simple-Global-Line Protocol



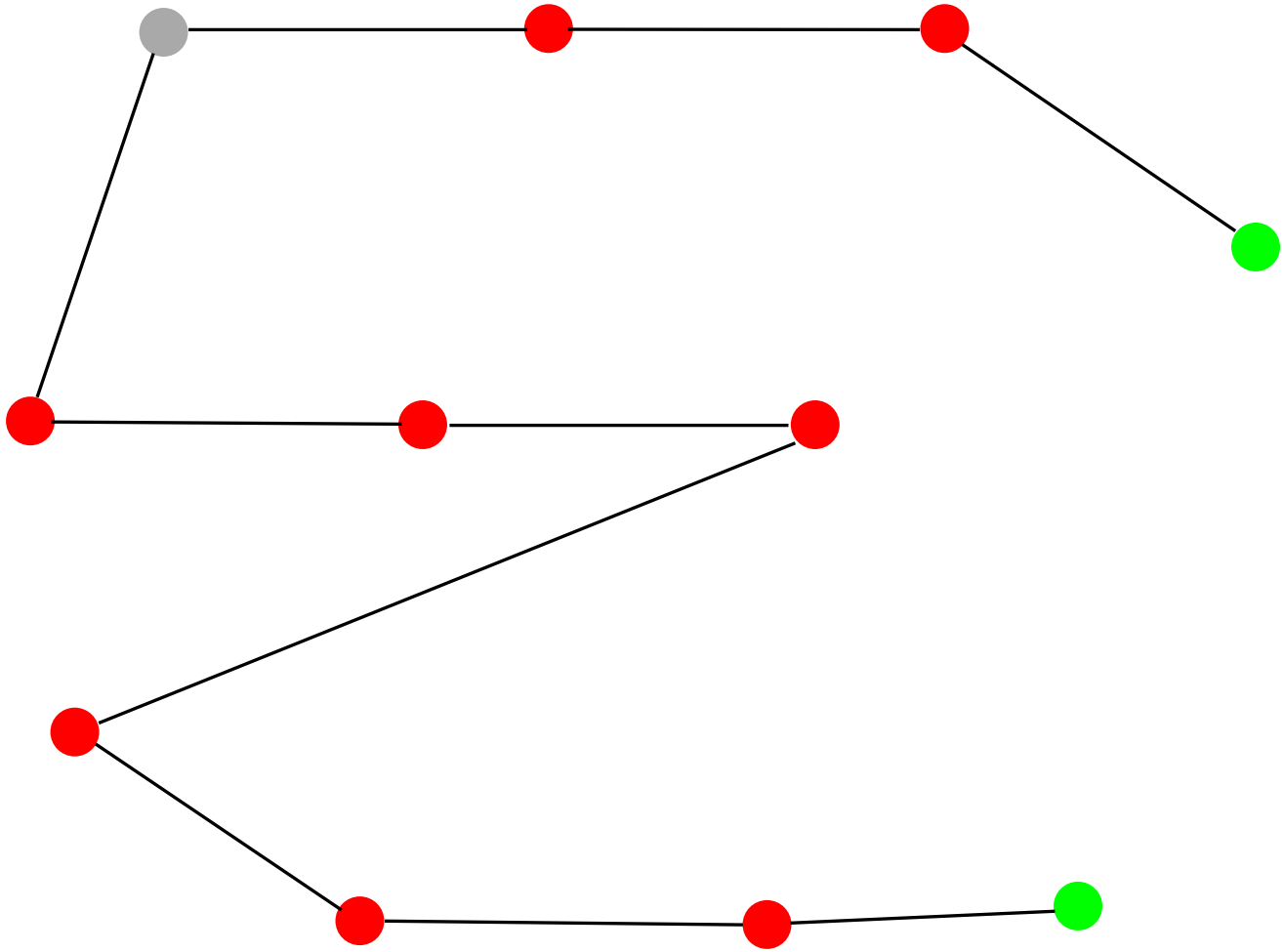
Simple-Global-Line Protocol



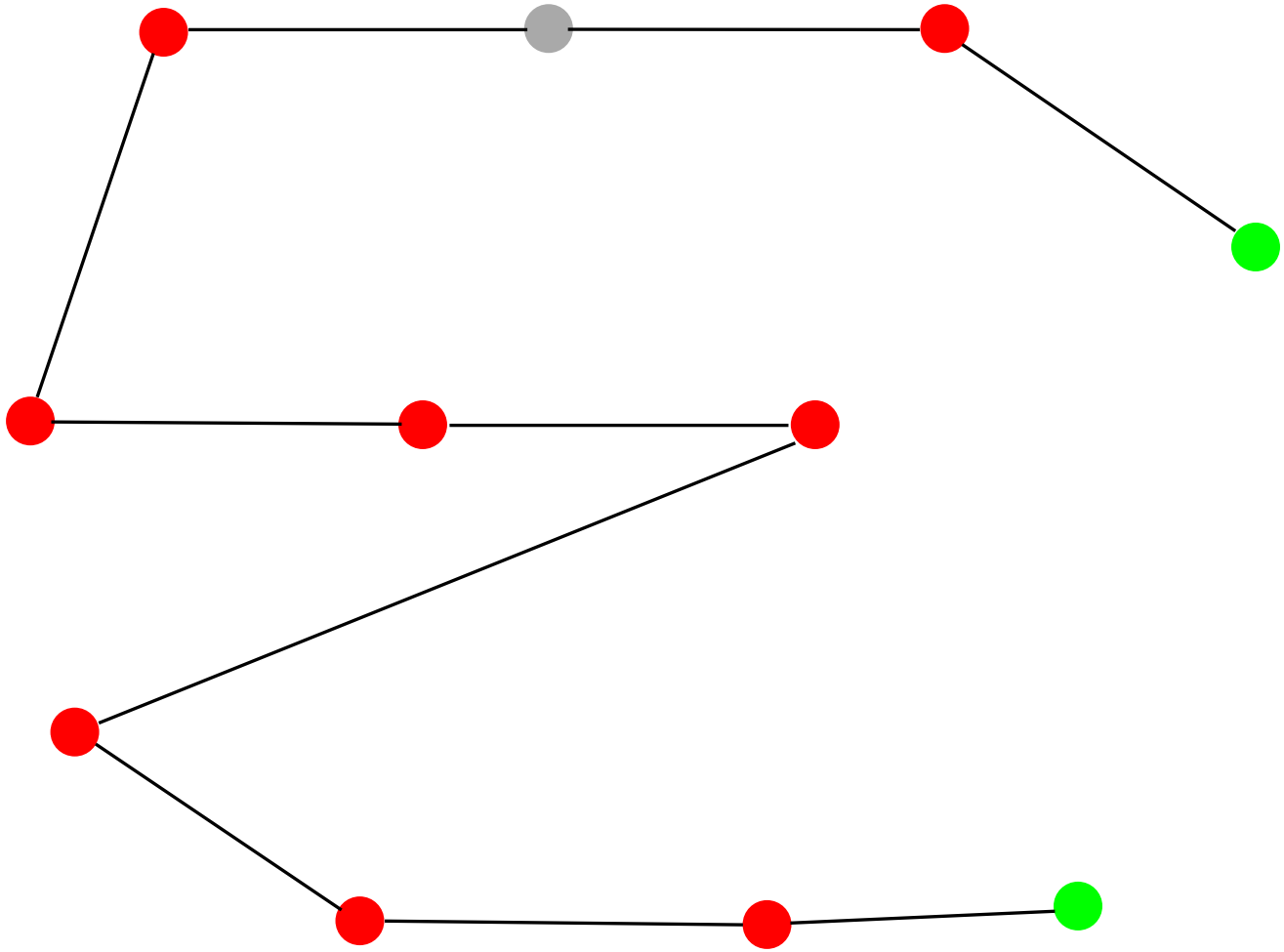
Simple-Global-Line Protocol



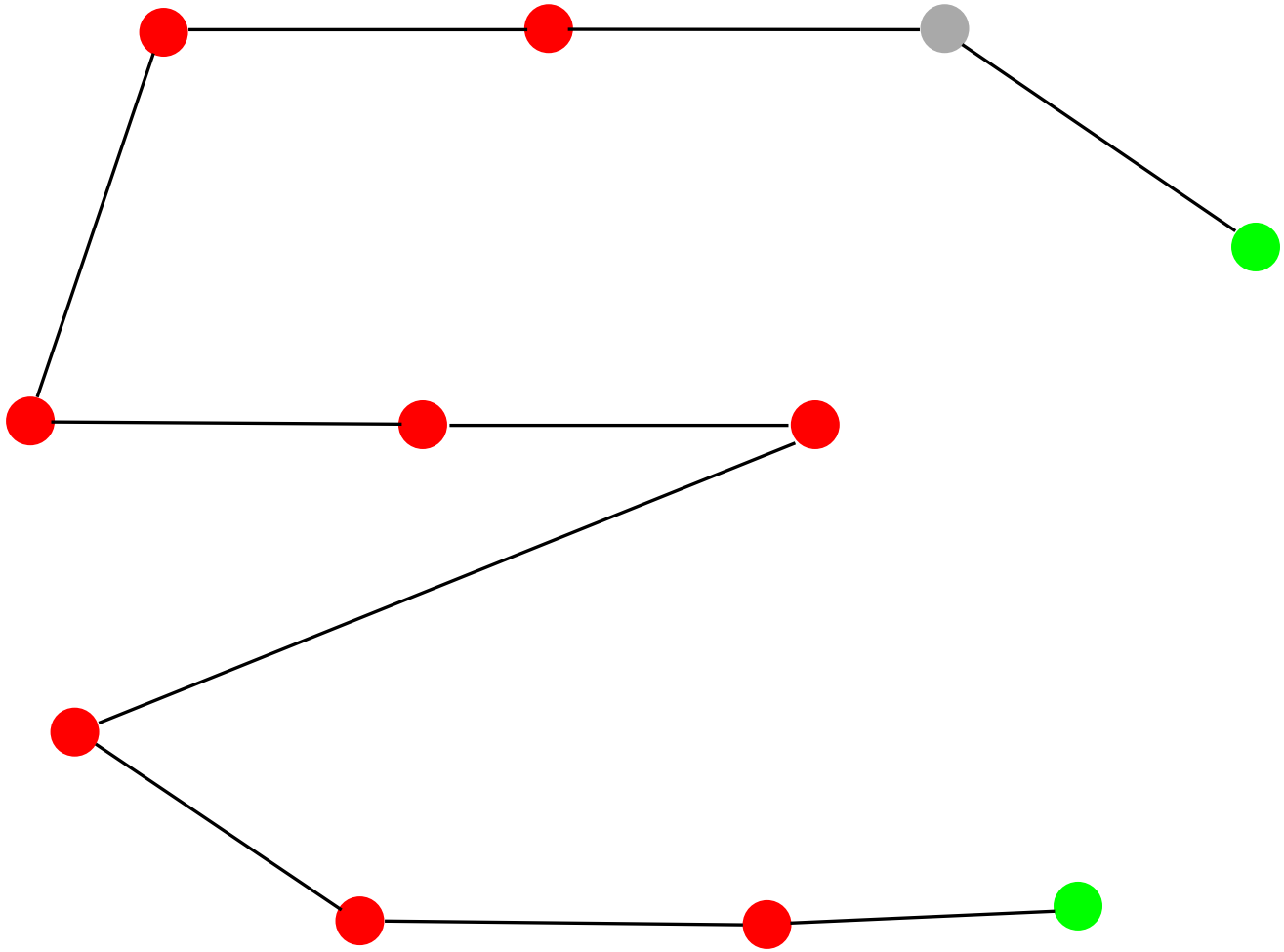
Simple-Global-Line Protocol



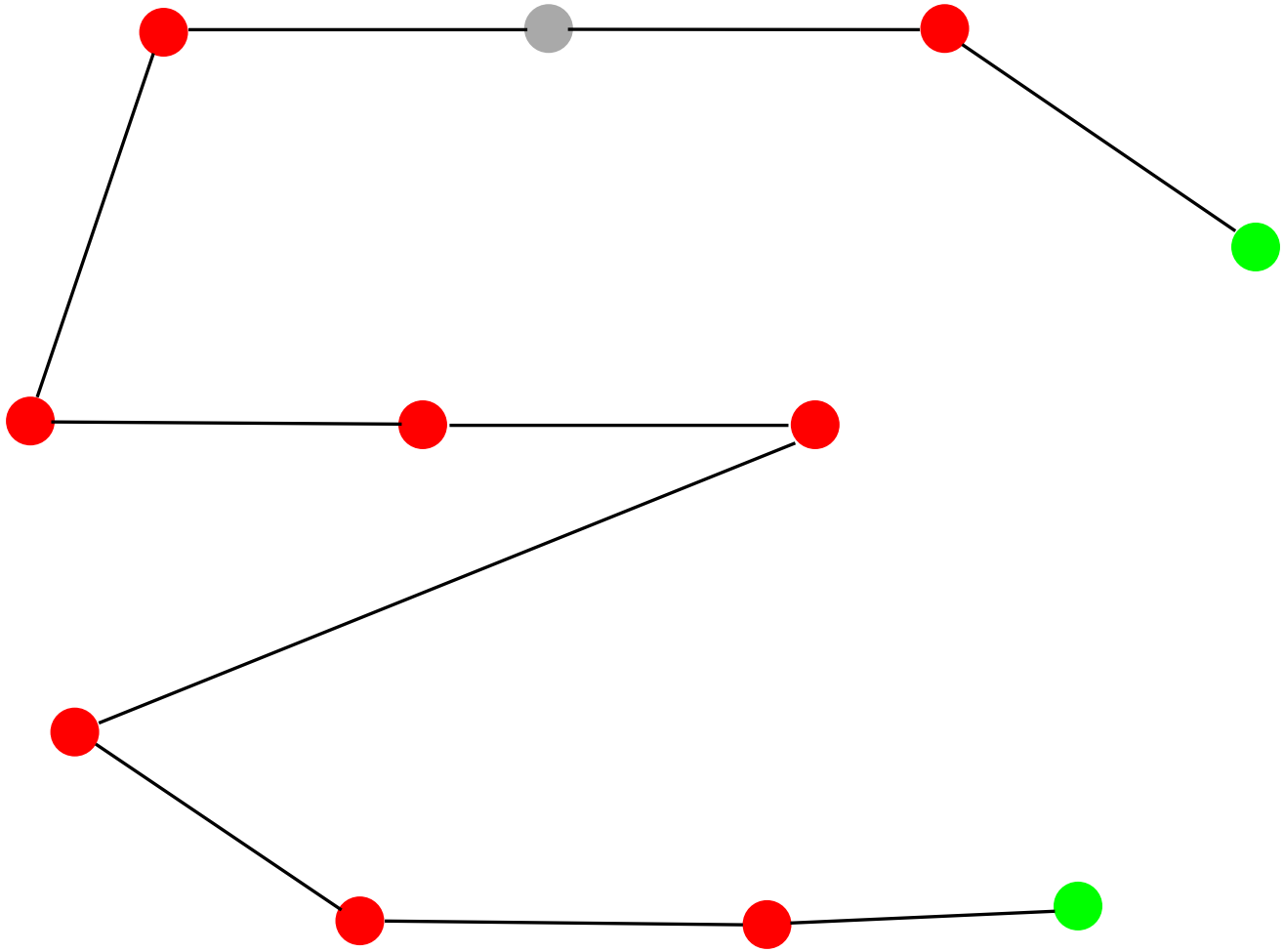
Simple-Global-Line Protocol



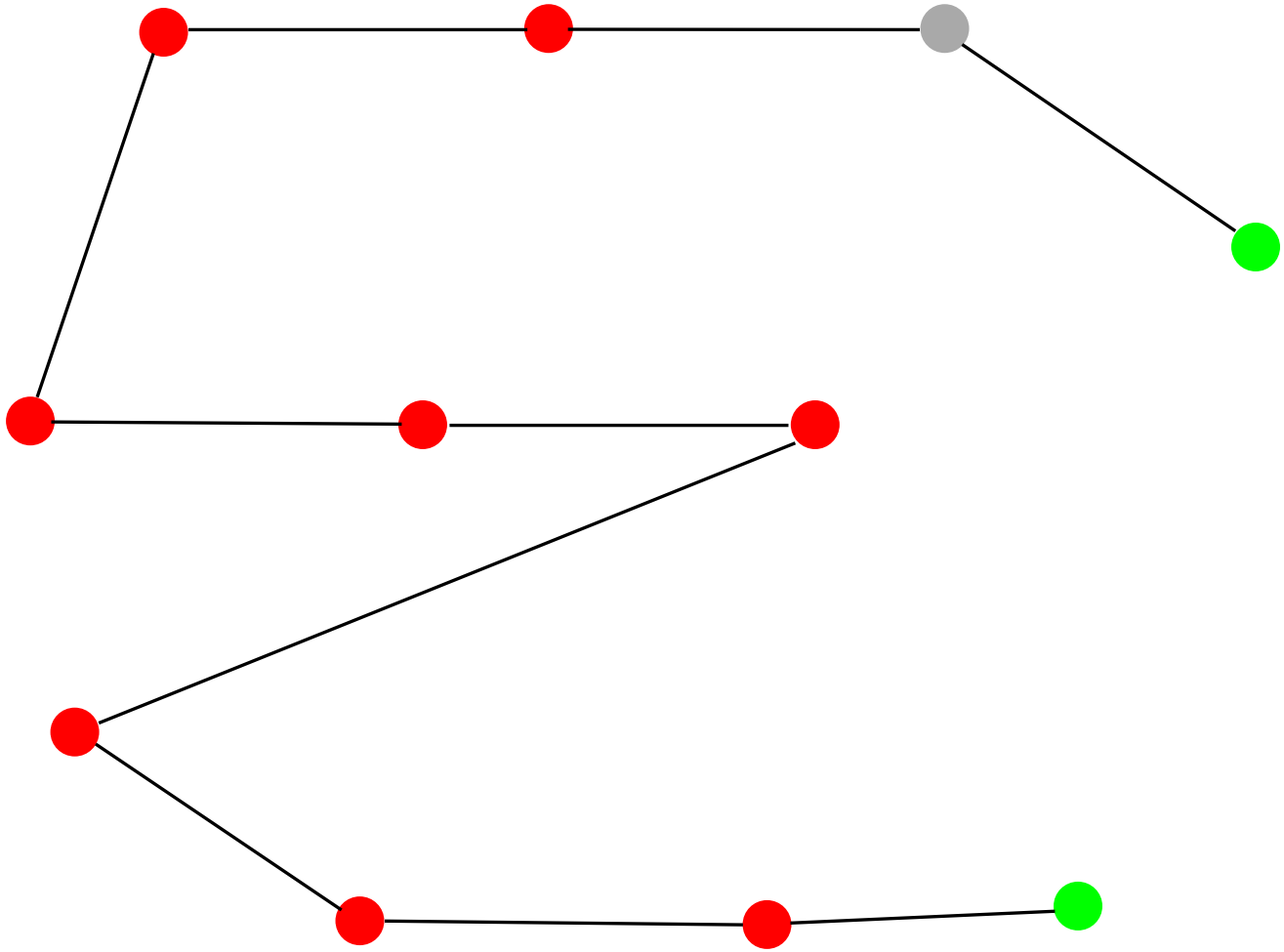
Simple-Global-Line Protocol



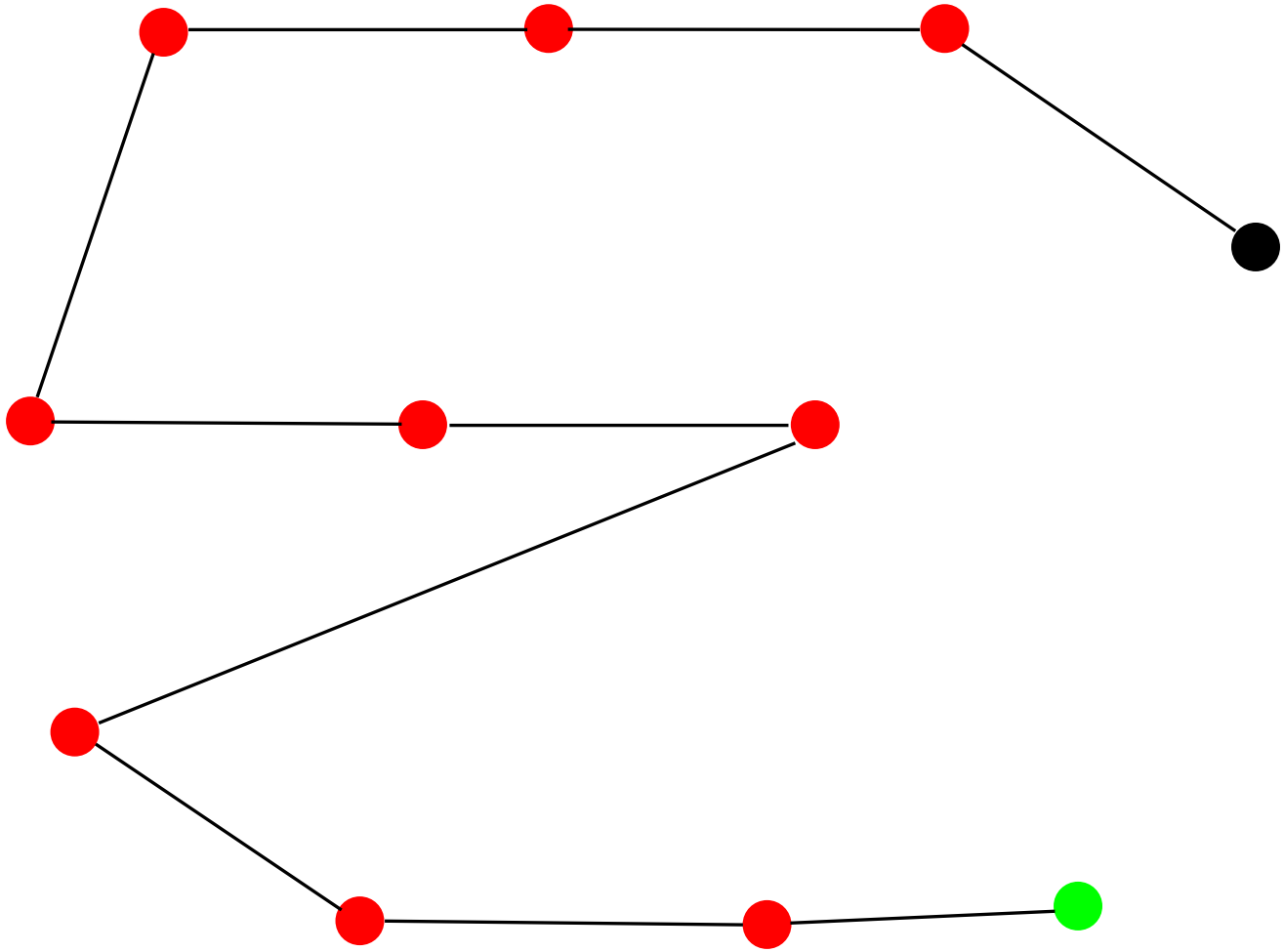
Simple-Global-Line Protocol



Simple-Global-Line Protocol



Simple-Global-Line Protocol



Theorem

Protocol Simple-Global-Line has expected running time $\Omega(n^4)$ *and* $O(n^5)$.

Proof.

- Upper bound: $(n-2)[O(n^2)+O(n^4)] = O(n^5)$
- Lower Bound:
  - w.h.p. constructs $\Theta(n)$ different lines of length 1 during its course
  - $k = \Theta(n)$ disjoint lines $\Rightarrow$ $k = \Theta(n)$ distinct merging processes
    1. 2 lines both of length $\Theta(n)$ get merged $\Rightarrow$ $\Omega(n^4)$
    2. In every merging, at least one of the two lines has length at most $d_{max} = o(n)$ $\Rightarrow$ only 1 line becomes $h = k/2$ and can only grow sequentially by at most d_{max} $\Rightarrow$ $j = \Theta(n)/d_{max}$ distinct random walks:

$$\mathrm{E}[Y] = \mathrm{E}[\textstyle\sum_{i=1}^{j} Y_i] = \sum_{i=1}^{j} \mathrm{E}[Y_i] = \sum_{i=1}^{j} n^2(h + d_1 + \ldots + d_{i-1})d_i$$
$$\geq n^2 \textstyle\sum_{i=1}^{j} h d_i = n^2 h \sum_{i=1}^{j} d_i = n^2\Theta(n)(k-h) = \Theta(n^4)$$

□

Fast-Global-Line Protocol

- *Fast-Global-Line*: $(q_0, q_0, 0) \to (q_1, l, 1)$, $(l, q_0, 0) \to (q_2, l, 1)$, $(l, l, 0) \to (q_2', l', 1)$, $(l', q_2, 1) \to (l'', f_1, 0)$, $(l', q_1, 1) \to (l'', f_0, 0)$, $(l'', q_2', 1) \to (l, q_2, 1)$, $(l, f_0, 0) \to (q_2, l, 1)$, $(l, f_1, 0) \to (q_2', l', 1)$
  - Avoid mergings as they seem to consume much time
    - even deterministic merging takes time $\Omega(n^3)$ and $O(n^4)$
  - When leaders of lines interact, they play a pairwise game
  - The winner grows by one towards the other line and the loser sleeps
  - A sleeping line cannot increase any more and only loses nodes by lines that are still awake
  - A single leader is guaranteed to always win and eventually remain unique and this occurs quite fast
  - The leader makes progress (by one) in most interactions and every such progress is in turn quite fast
  - Running-time: $O(n^3)$

Other Basic Constructors

- Cycle cover: Every process must eventually have degree 2 (collection of node-disjoint cycles spanning V_l)
- Global ring: The processes must construct a spanning ring, i.e. a connected cycle cover
- k-regular connected: Generalization of global ring in which every node has degree $k \geq 2$ (constant)
- c-cliques: The processes must partition themselves into $\lfloor n/c \rfloor$ cliques of order c each
- Replication: The protocol is given an input graph G on a subset V_1 of the processes. The goal is to create a replica of G on $V_2 = V_l \setminus V_1$, provided that $|V_2| \geq |V_1|$.

All Established Bounds

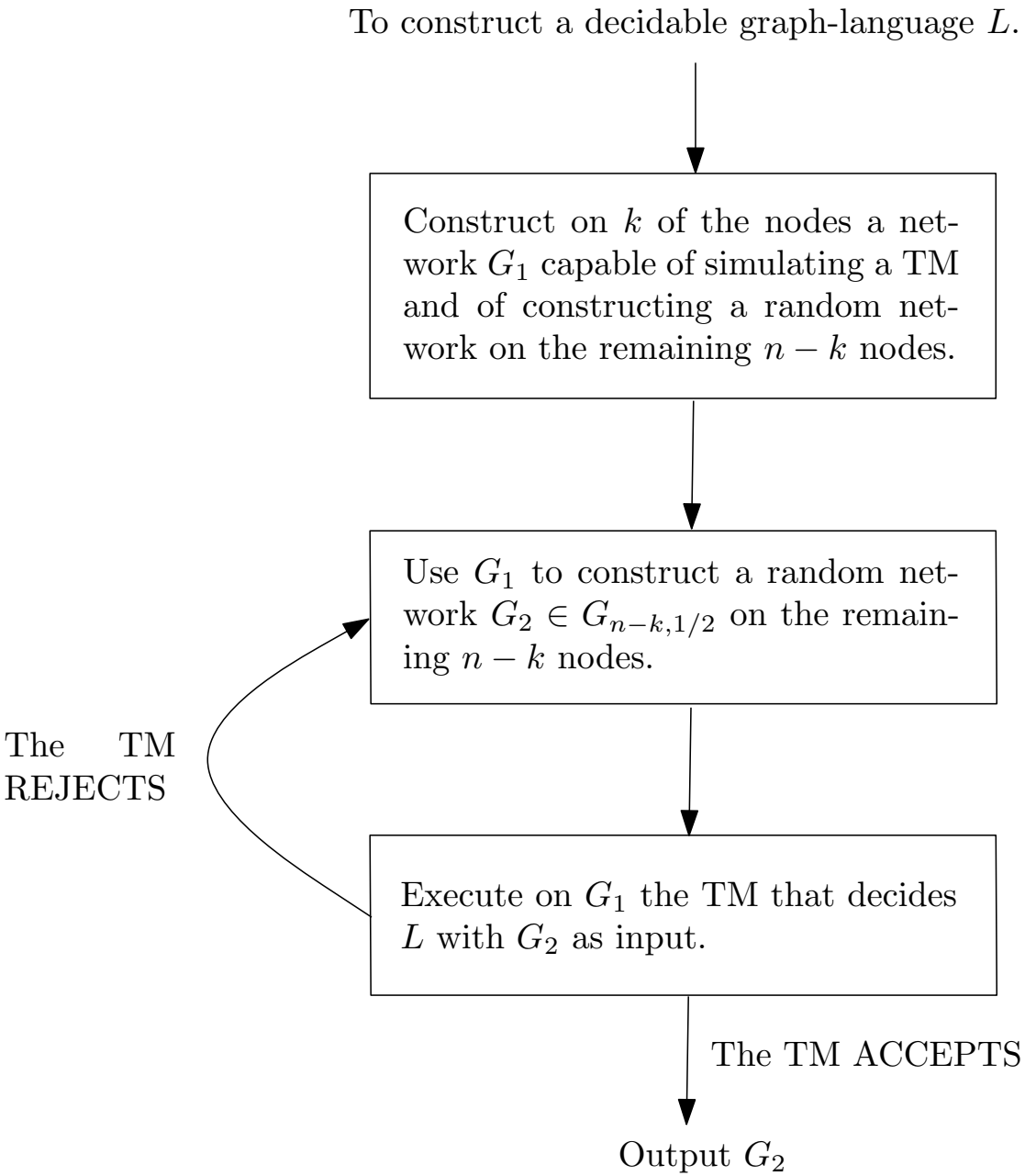
Protocol	# states	Expected Time	Lower Bound
Simple-Global-Line	5	$\Omega(n^4)$ and $O(n^5)$	$\Omega(n^2)$
Intermediate-Global-Line	8	$\Omega(n^3)$ and $O(n^4)$	$\Omega(n^2)$
Fast-Global-Line	9	$O(n^3)$	$\Omega(n^2)$
Cycle-Cover	3	$\Theta(n^2)$ (optimal)	$\Omega(n^2)$
Global-Star	2 (optimal)	$\Theta(n^2 \log n)$ (optimal)	$\Omega(n^2 \log n)$
Global-Ring	9		$\Omega(n^2)$
2RC	6		$\Omega(n \log n)$
kRC	$2(k+1)$		$\Omega(n \log n)$
c-Cliques	$5c-3$		$\Omega(n \log n)$
Leader-Replication	12	$\Theta(n^4 \log n)$	

Generic Constructors

Question: Is there a generic constructor capable of constructing a large class of networks?

1. constructors that simulate a Turing Machine
2. a constructor that simulates a distributed system with names and logarithmic local memories
   - l: the binary length of the input of a TM
     - In what follows, $l = \Theta(n^2)$
   - **DGS**$(f(l))$: the class of graph languages decidable by a TM of (binary) space $f(l)$ (input graph in adjacency matrix encoding)
   - **REL**$(g(n))$: the class of graph languages constructible with useful space $g(n)$ (relation or on/off class)
   - **PREL**$(g(n))$: (i) allow transitions that with probability $1/2$ give one outcome and with probability $1/2$ another (ii) all graphs must be constructed equiprobably

Simulating a TM: Main Mechanism



Linear Waste-Half

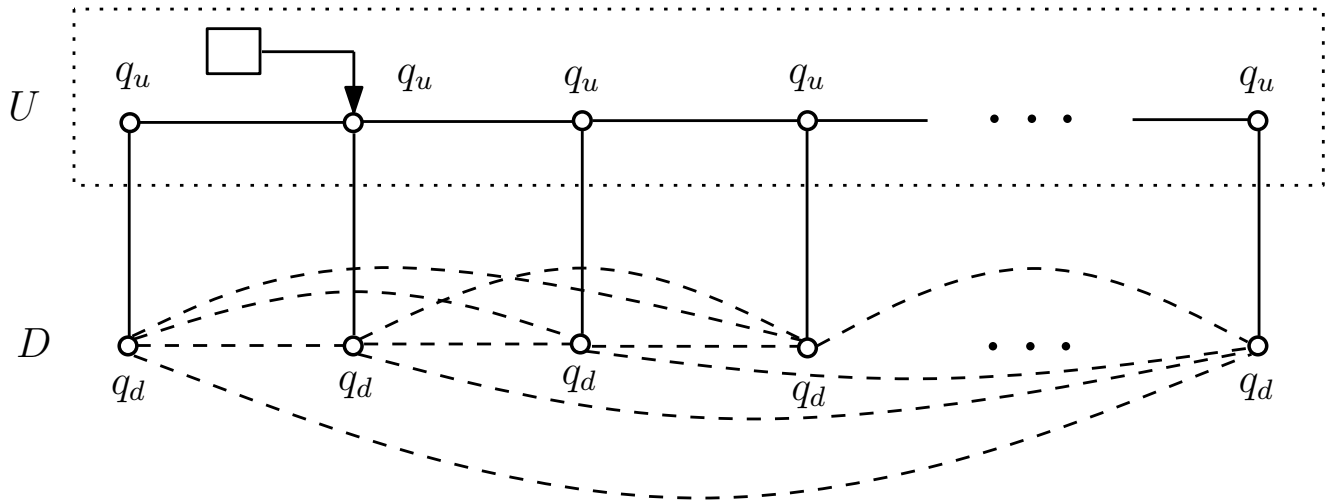
Theorem (Linear Waste-Half)

DGS$(O(\sqrt{l})) \subseteq$ **PREL**$(\lfloor n/2 \rfloor)$. *In words, for every graph language L that is decidable by a $O(\sqrt{l})$-space TM, there is a protocol that constructs L equiprobably with useful space $\lfloor n/2 \rfloor$.*

Proof

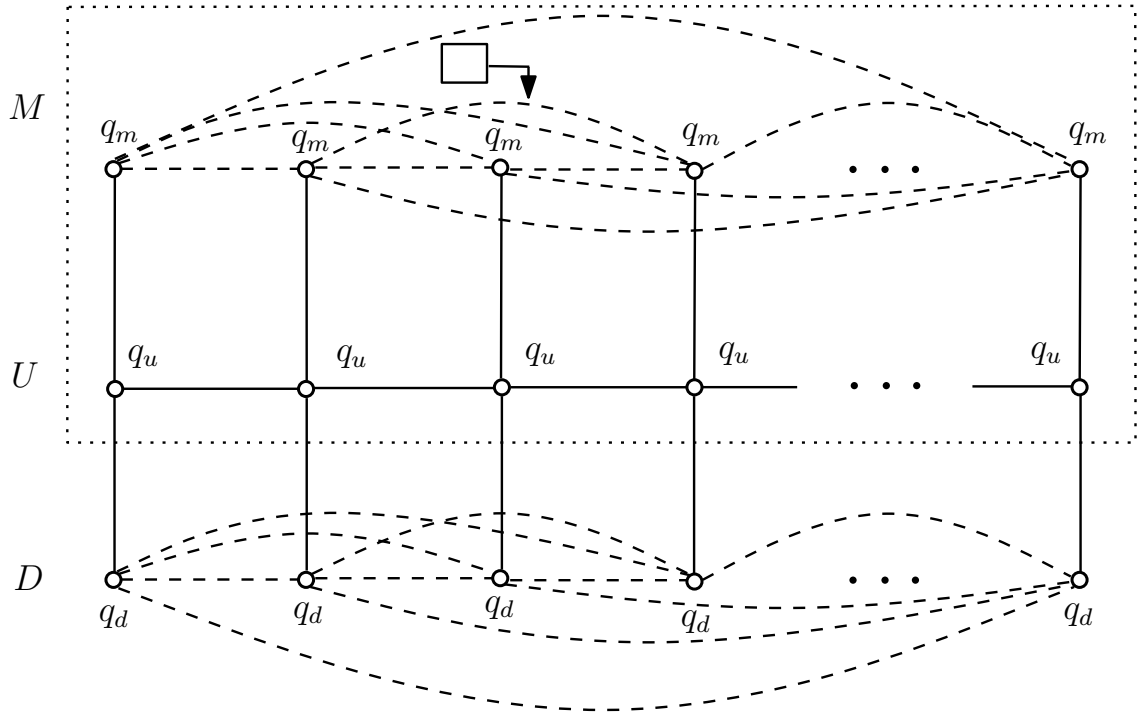
- *Partition the population into equal sets U and D and construct an active perfect matching between U and D*
- *Construct a spanning line in U (e.g. Fast-Global-Line)*
- *Organize the line into a TM M*
- *M must compute a graph from L and construct it on D*
  - *uniquely identify the nodes of D by their distance from one endpoint*
  - *to modify edge (i, j) mark appropriately the D-nodes at distances i and j from one endpoint*

Linear Waste-Half



- *M computes a graph G from L equiprobably: activate/deactivate each edge of D equiprobably and independently of other edges*
- *Then it simulates on input G the TM that decides L in $\sqrt{l}$ space to determine whether $G \in L$*
- *The TM rejects: M repeats the random experiment to produce a new random graph G'*
- *The TM accepts: release the network, set D-nodes to q_{out}*
- *Reinitialize whenever the global line protocol makes progress* □

Linear Waste-Two Thirds



Theorem (Linear Waste-Two Thirds)

DGS$(l + O(\sqrt{l})) \subseteq$ **PREL**$(\lfloor n/3 \rfloor)$.

Proof.

- Three equal sets now: U, D, M
- M is now an additonal (external) $\Theta(n^2)$ memory for the TM

□

Logarithmic Waste

Theorem (Logarithmic Waste)

$\mathbf{DGS}(O(\log l)) \subseteq \mathbf{PREL}(n - \log n)$.

Proof.

- Construct a line to count n in binary, i.e. to occupy $\log n$ cells
- Release the constructed counter and reinitialize the other (free) nodes
- So, there is a logarithmic memory with a leader, knowing a good estimate of $n - \log n$
- Construct a random graph on the free nodes:
  - For every free node, let it toss a coin on one after the other its edges to other free nodes
  - To know when to stop, count on the line up to $n - \log n$ (known) □

No Waste-Supernodes

Theorem (No Waste)

Let L be a graph language such that: (i) there exists a natural number d s.t. for all $G \in L$ there is a subgraph G' of G of logarithmic order s.t. either G' or its complement is connected and has degree upper bounded by d and (ii) L is decidable in logarithmic space. Then $L \in \mathbf{PREL}(n)$, i.e. there is a protocol that constructs L equiprobably with useful space n.

Theorem (Partitioning into Supernodes)

For every network G that can be constructed by k nodes having local memories $\lceil \log k \rceil$ and unique names there is a NET that constructs G on $n = k\lceil \log k \rceil$ nodes.

Open Problems

- Direct constructors for other basic networks, like grids, planar graphs,...
- Probabilistic analysis of protocols and establishment of tight bounds
- Global Line: Haven't succeeded to give a protocol faster than $O(n^3)$ (best lower bound is $\Omega(n^2)$)
  - Is there a $\Theta(n^2 \log n)$ constructor?
- Lower bounds on the size of network constructors. Formalize the relationship between the size and the running time of a protocol

Open Problems

- Characterize the class **REL** (no access to internal randomness)
- The constructive power increases as a function of the available waste
  - Complete characterization?
- Allow the connections to have more than two states
- Other probabilistic scheduling models or even more adversarial
- Draw connections to physical and chemical (possibly biological) processes

Thank You!