Seminar 4 – Anonymous Networks: Advanced Computation Techniques

Distributed Computing in Anonymous Dynamic Systems

Giovanni Viglietta

Rome – March 11, 2024

Distributed Computing in Anonymous Dynamic Systems

Syllabus

- Anonymous Networks
  - Introduction and basic algorithms for static networks
  - Dynamicity and history trees
  - Optimal computation in networks with and without leaders
  - Computation in dynamic congested networks
- Population Protocols
  - Introduction and basic algorithmic techniques
  - Leader election in Mediated Population Protocols
- Mobile Robots
  - Gathering and Pattern Formation in the plane
  - Meeting in a polygon by oblivious robots

Exam

Pre-recorded 10-minute presentation video on one of the papers that will be suggested at the end of the course.

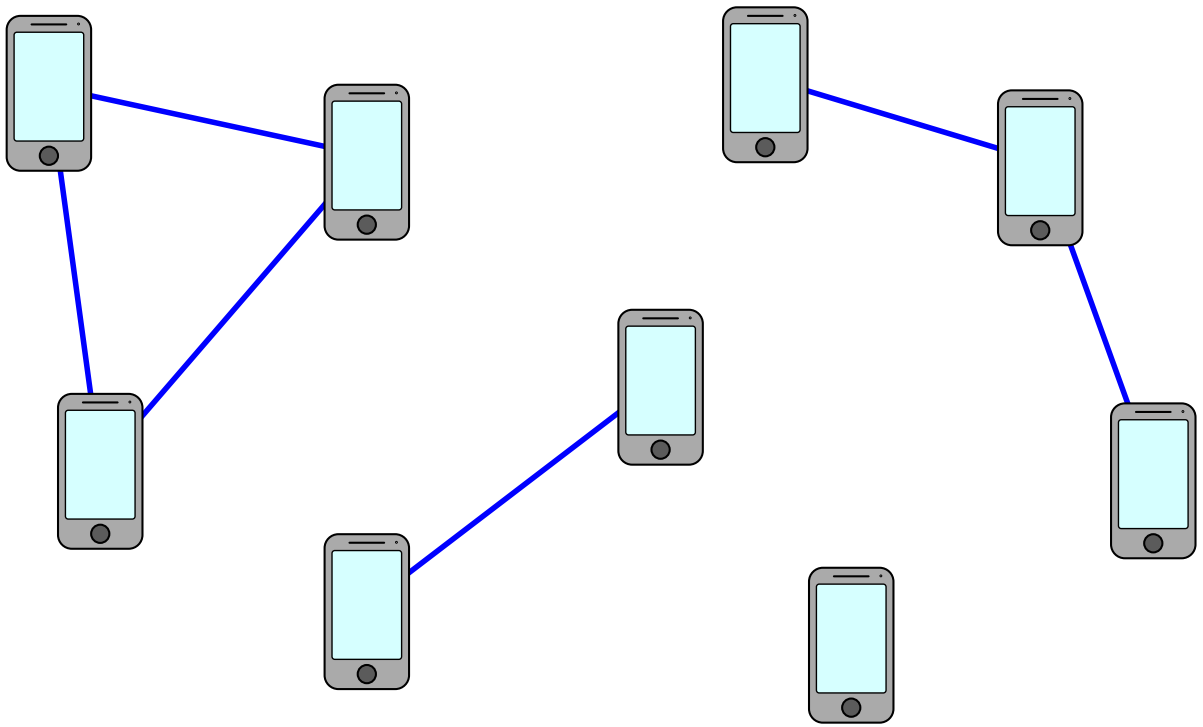
Today's seminar

- Disconnected networks
- Asynchronous communications
- Self-stabilization
- Networks with multiple leaders
- Directed networks
- Congested networks

Disconnected Networks

Disconnected networks

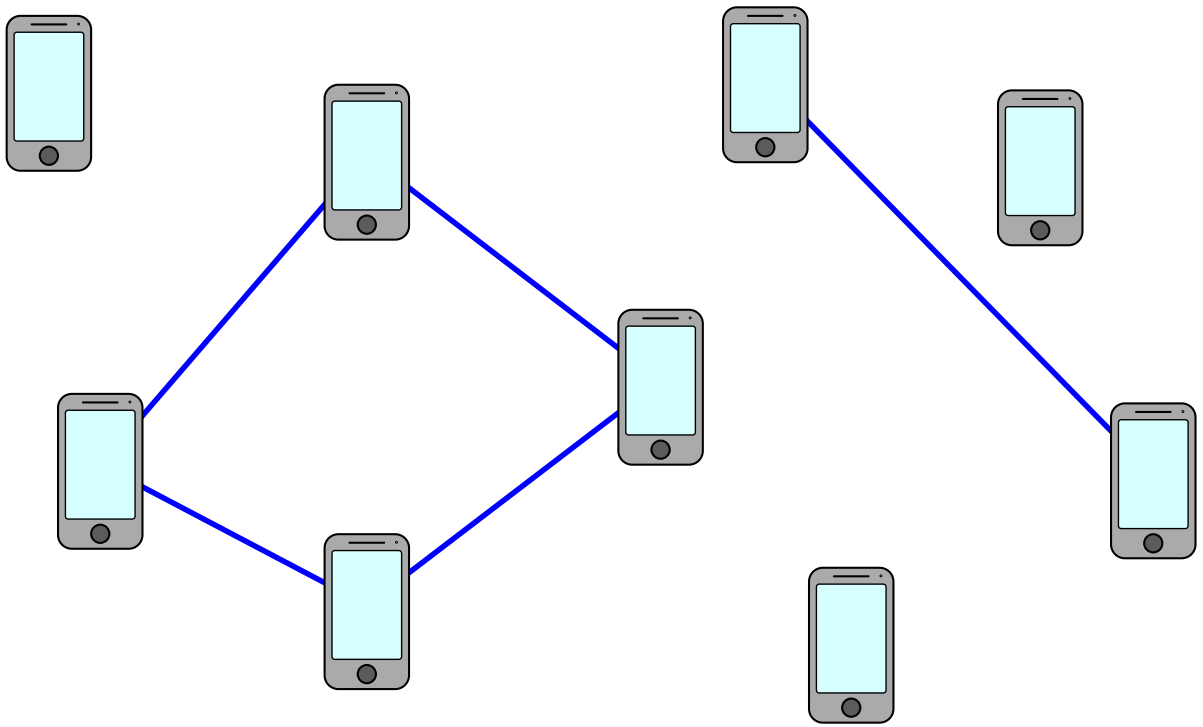
While networks are typically assumed to be (strongly) connected at every round, we may remove this assumption.



In a τ-*union-connected network*, the union of the network graphs at τ consecutive rounds is a (strongly) connected multi-graph.

Disconnected networks

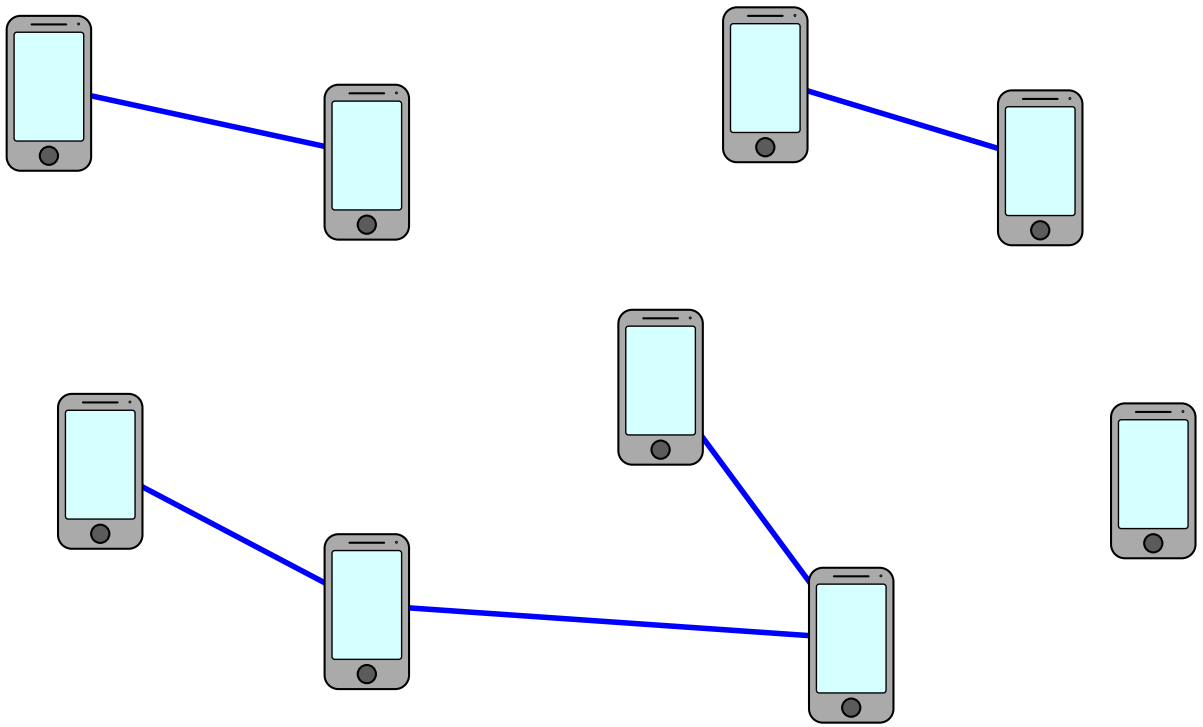
While networks are typically assumed to be (strongly) connected at every round, we may remove this assumption.



In a τ-*union-connected network*, the union of the network graphs at τ consecutive rounds is a (strongly) connected multi-graph.

Disconnected networks

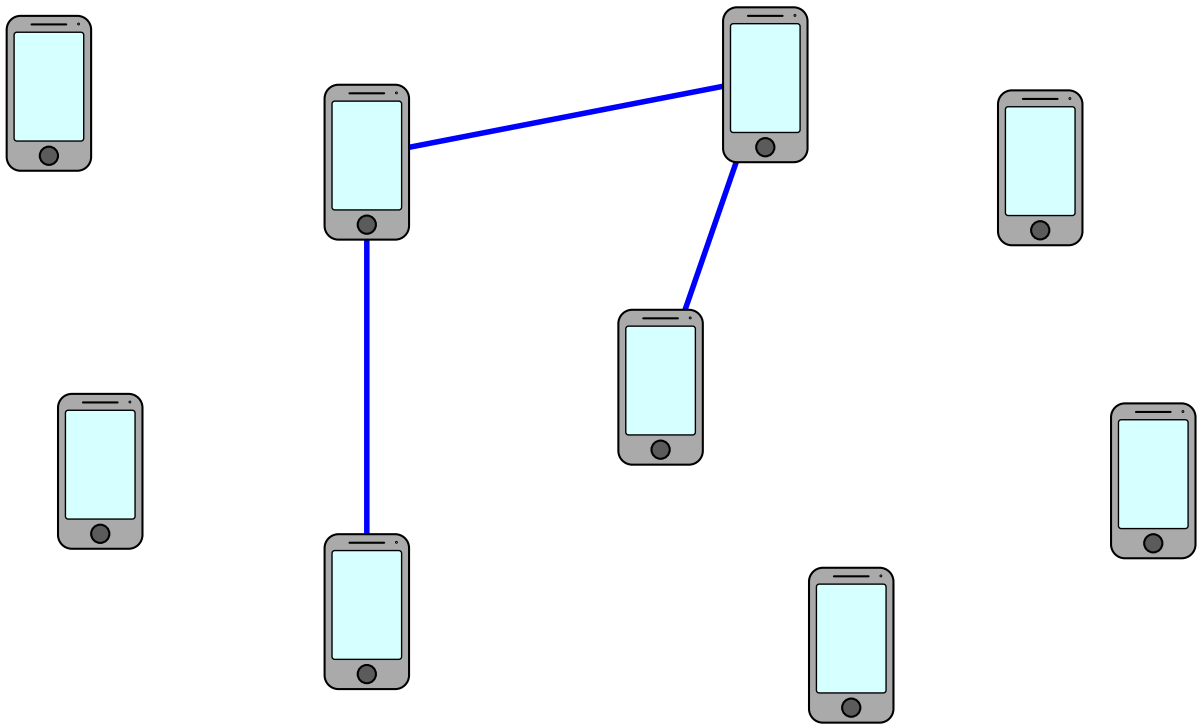
While networks are typically assumed to be (strongly) connected at every round, we may remove this assumption.



In a τ-*union-connected network*, the union of the network graphs at τ consecutive rounds is a (strongly) connected multi-graph.

Disconnected networks

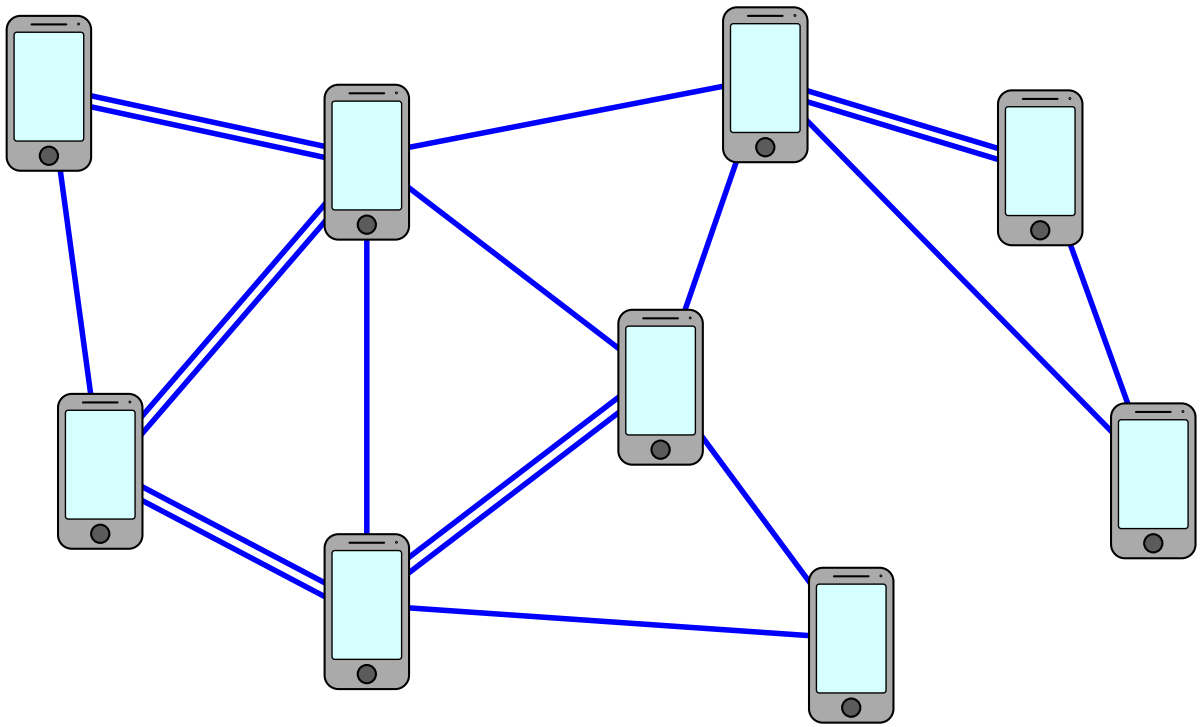
While networks are typically assumed to be (strongly) connected at every round, we may remove this assumption.



In a τ-*union-connected network*, the union of the network graphs at τ consecutive rounds is a (strongly) connected multi-graph.

Disconnected networks

While networks are typically assumed to be (strongly) connected at every round, we may remove this assumption.

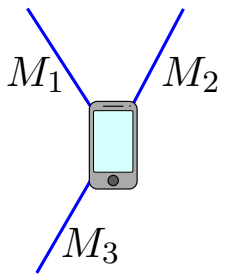


In a τ-*union-connected network*, the union of the network graphs at τ consecutive rounds is a (strongly) connected multi-graph.

τ-union-connected networks

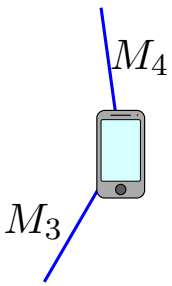
Any algorithm for $\tau = 1$ can be adapted to networks with $\tau > 1$, assuming τ is known by all agents.

Round $k\tau + 1$



Message list
$M_1 \times 1$
$M_2 \times 1$
$M_3 \times 1$

Round $k\tau + 2$

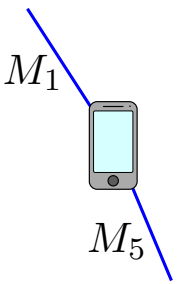


Message list
$M_1 \times 1$
$M_2 \times 1$
$M_3 \times 2$
$M_4 \times 1$

Round $k\tau + 3$

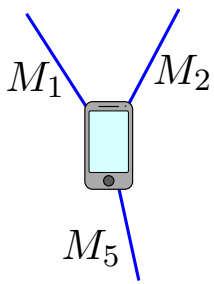
Message list
$M_1 \times 1$
$M_2 \times 1$
$M_3 \times 2$
$M_4 \times 1$

Round $k\tau + 4$



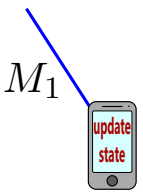
Message list
$M_1 \times 2$
$M_2 \times 1$
$M_3 \times 2$
$M_4 \times 1$
$M_5 \times 1$

Round $k\tau + 5$



Message list
$M_1 \times 3$
$M_2 \times 2$
$M_3 \times 2$
$M_4 \times 1$
$M_5 \times 2$

Round $k\tau + \tau$

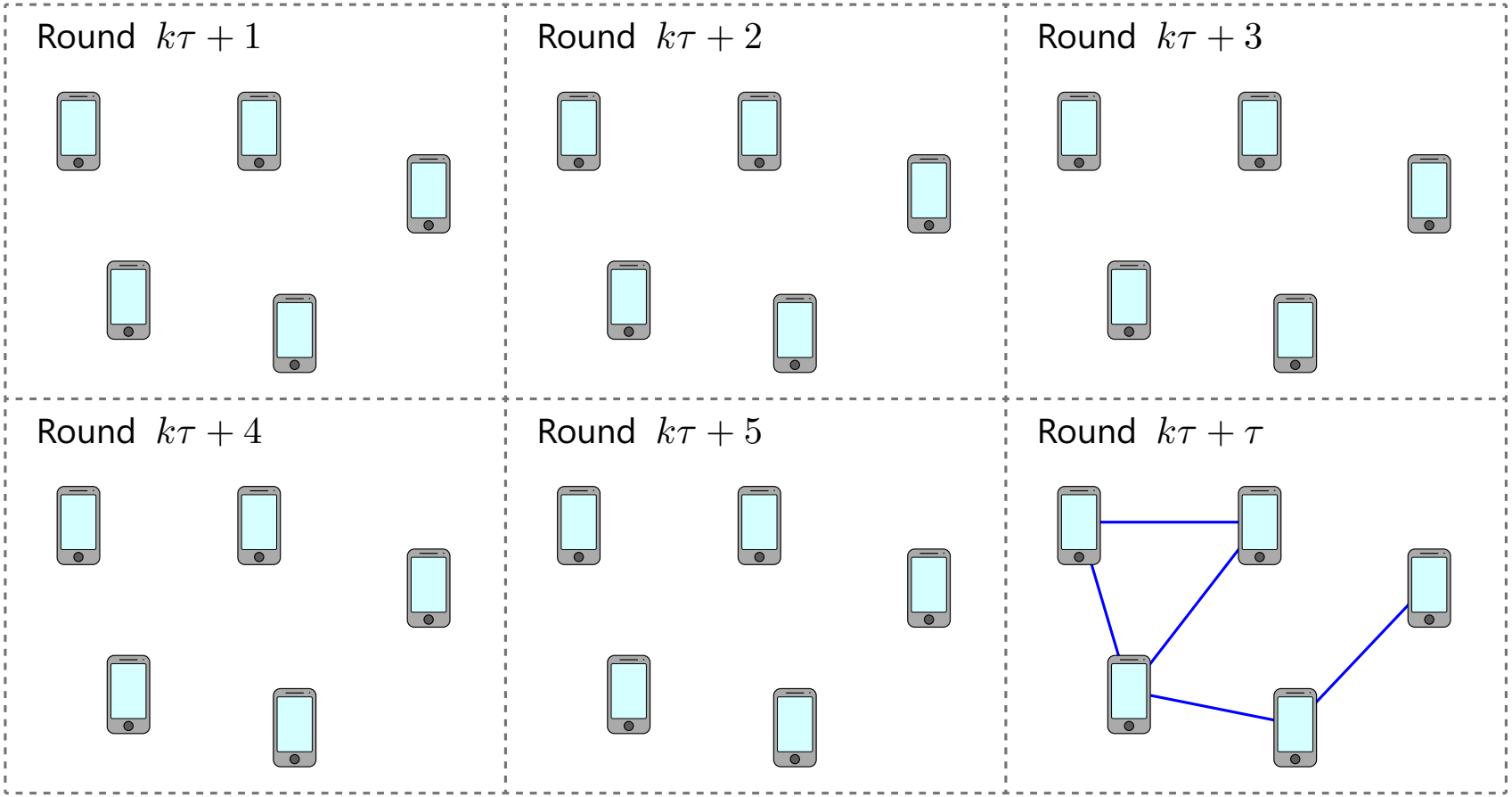


Message list
$M_1 \times 4$
$M_2 \times 2$
$M_3 \times 2$
$M_4 \times 1$
$M_5 \times 2$

Each agent accumulates messages for τ rounds, and then updates its state all at once. Hence, the running time is multiplied by τ.

τ-union-connected networks

This is the best we can do: Consider, for instance, a network that contains no links for $\tau - 1$ out of every τ rounds.



Thus, the Counting Problem has a lower bound of $2\tau n$ rounds and can be solved with termination in $3\tau n$ rounds.

Asynchronous Communications

Asynchronous communications

In the *asynchronous* model, any agent may be "asleep" at any round. The only constraint is that no agent stays asleep forever.

If an agent is asleep, it does not receive nor send any messages. The (dynamic) diameter of the network is still assumed to be finite.

We can adapt the history tree technique to this model as follows.

When two agents interact, the view of one may have fewer levels that the other. In the merge procedure, we simply append a branch of "dummy nodes" to the shortest view to match the other one.

The outcome is compatible with a network where the agent with fewer levels was asleep in the last few rounds (it may not be the correct assumption, but the result of the computation will be correct anyway).

In this model, a *round* is defined as a minimal time frame in which each agent is active at least once.

Self-Stabilization

Self-stabilizing computations

An algorithm is *self-stabilizing* if all agents are guaranteed to return the correct output regardless of their initial internal states.

For example, their initial states may encode completely arbitrary history trees, which may lead to incorrect computations...

How can we adapt our history tree technique to this scenario?

Simply update the history tree as usual, but every two rounds delete the level L_0, as well as all edges incident to it, and connect all nodes in L_1 directly to the root.

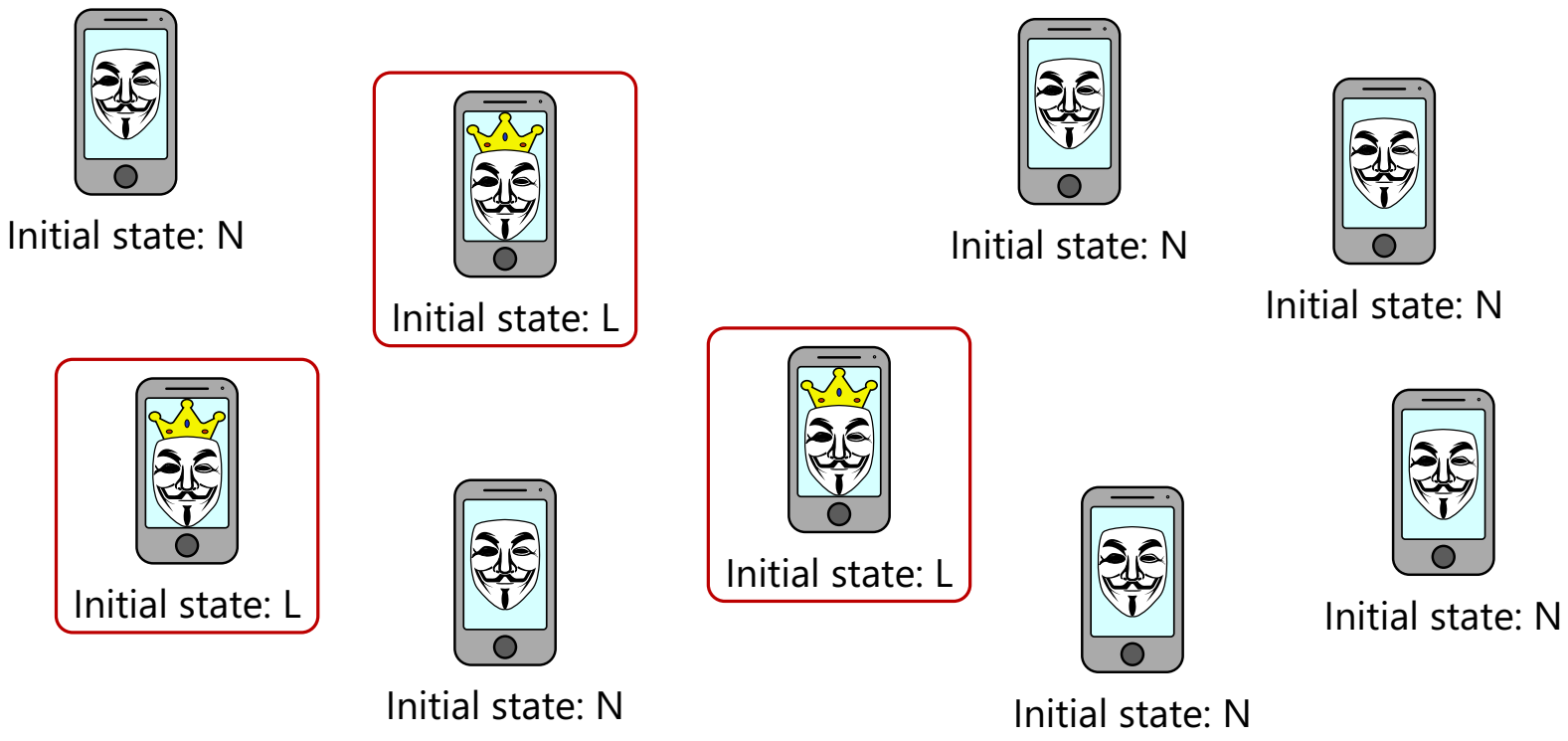
This technique converts any non-self-stabilizing algorithm with a running time of $f(n)$ rounds into a self-stabilizing algorithm that gives the correct output in $2f(n)$ rounds (plus the time it takes to erase any incorrect data initially present in the agents' states).

Note that termination is impossible: Any non-trivial computation cannot be both self-stabilizing and terminating.

Multiple Leaders

Multiple Leaders

We will now consider the scenario where *more than one Leader* is present in the network. All leaders are indistinguishable.

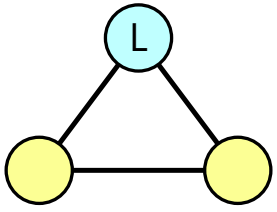


How can we solve the Counting Problem in this case?

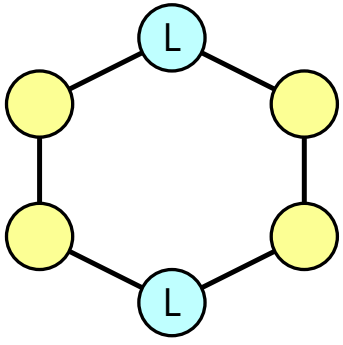
Multiple Leaders

The Counting Problem is unsolvable with no knowledge on the number of Leaders, ℓ.

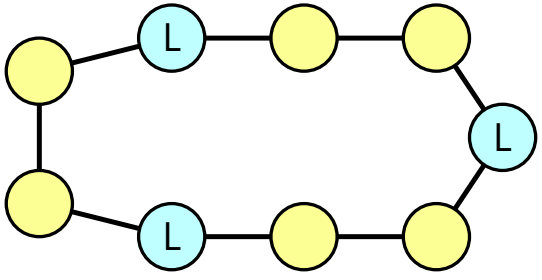
System 1



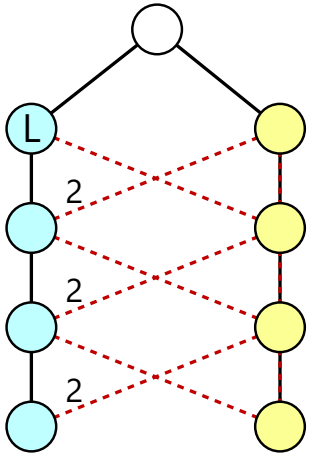
System 2



System 3



History tree

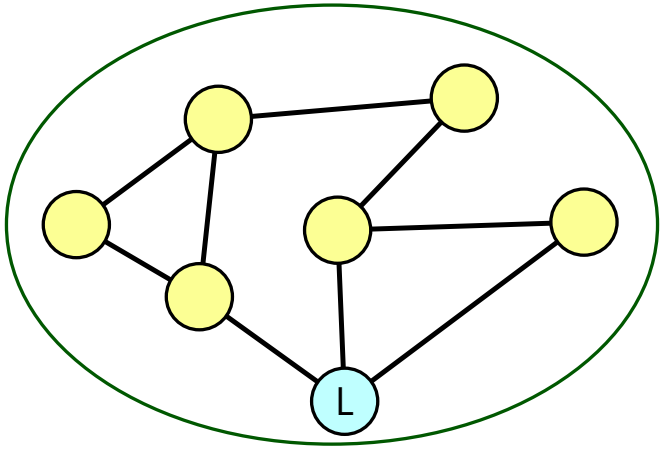


In the above (static) networks, the history tree is the same.

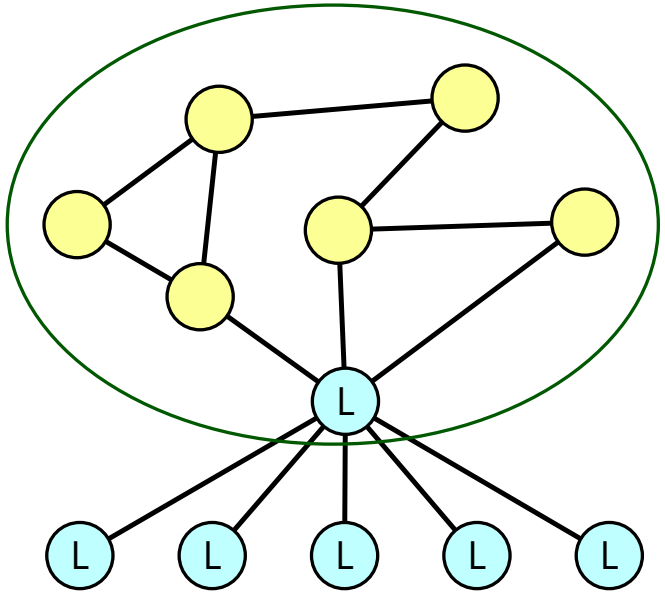
Multiple Leaders

Having more than one Leader may not be very helpful. Actually, multiple Leaders introduce more symmetry in the network.

System 1



System 2

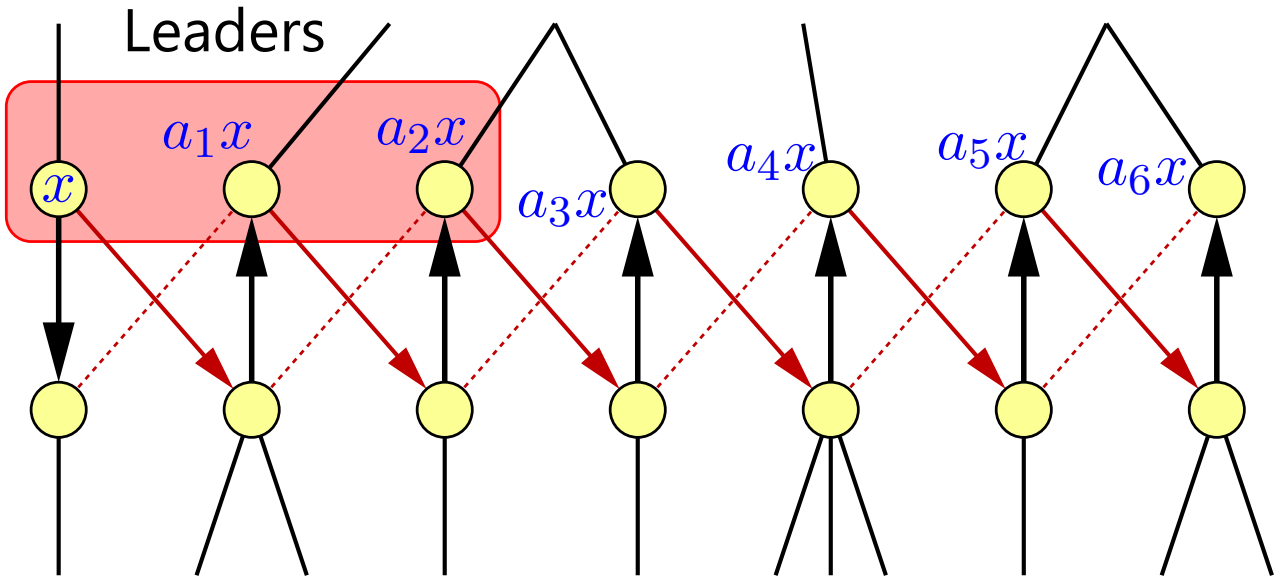


If multiple Leaders were always helpful, a single Leader could "pretend" to see several other Leaders to speed up computation.

Multiple Leaders: Stabilizing algorithm

For a stabilizing (non-terminating) Counting algorithm, we can use the single-Leader technique (assuming that all agents know ℓ).

As soon as a level has no branching nodes, we can compute all anonymities as a function of a single Leader node's anonymity, x.

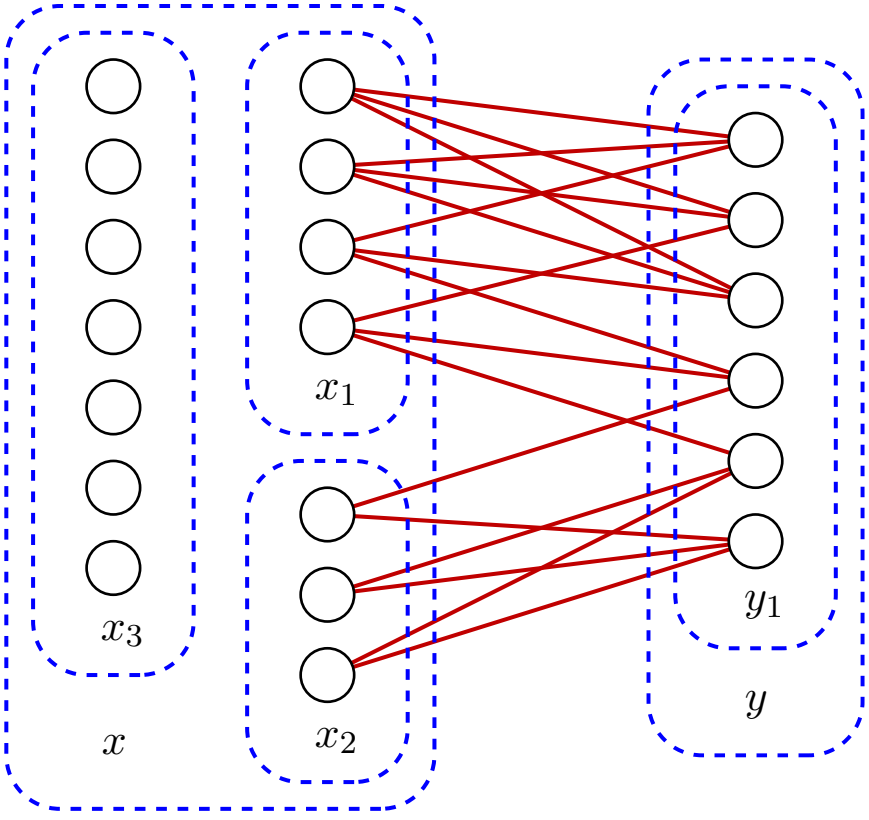


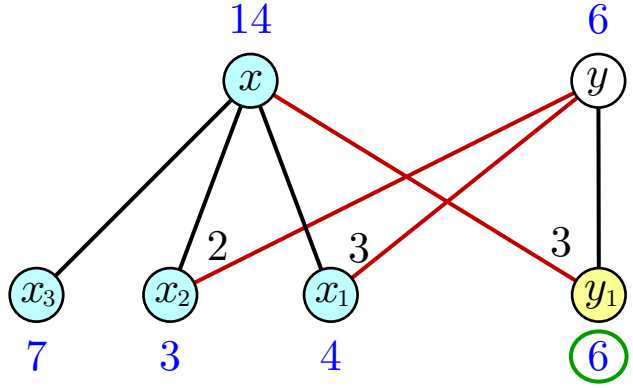
We know that $x + a_1x + a_2x = \ell$, and we can determine x.

This yields a non-terminating Counting algorithm that stabilizes in at most $2n$ rounds (optimal).

Single-Leader algorithm review

Suppose we know the anonymities of a node x and its children. If some of the agents represented by x have observed agents represented by y, we can *guess* the anonymity of a child of y.



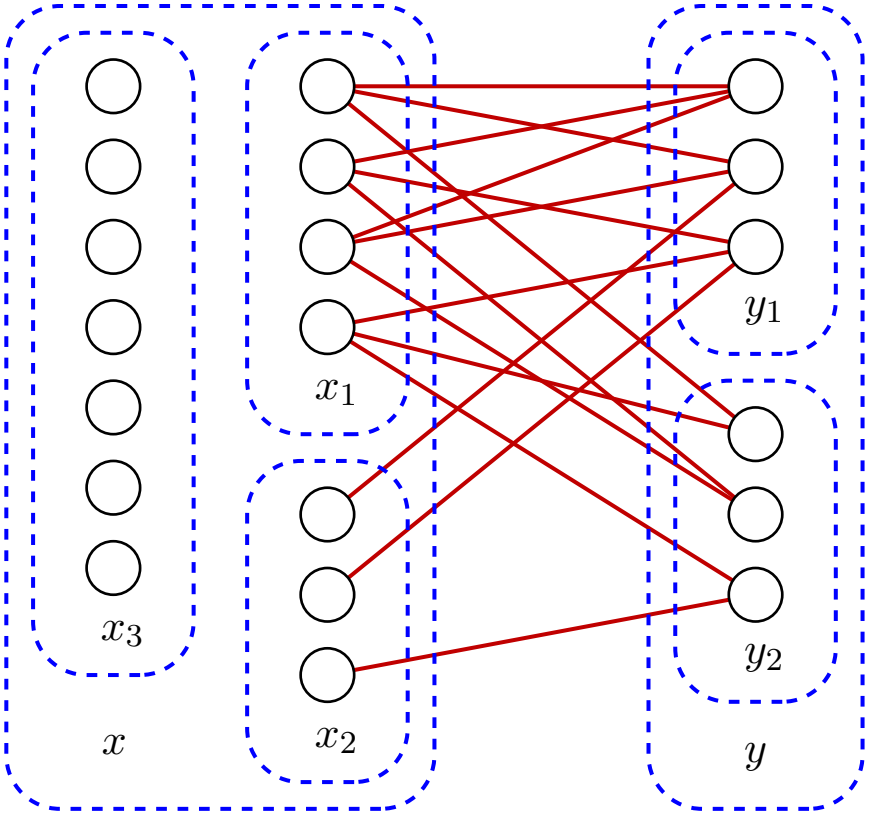


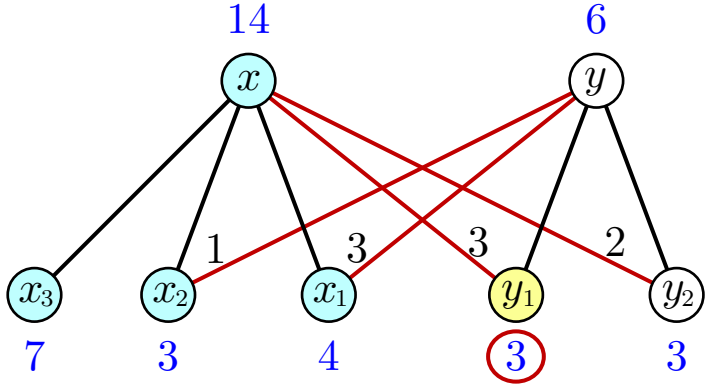
Guess on y_1: $\dfrac{4 \cdot 3 + 3 \cdot 2}{3} = 6$

If only one child of y has seen x, then the guess is *correct*.

Single-Leader algorithm review

Suppose we know the anonymities of a node x and its children. If some of the agents represented by x have observed agents represented by y, we can *guess* the anonymity of a child of y.



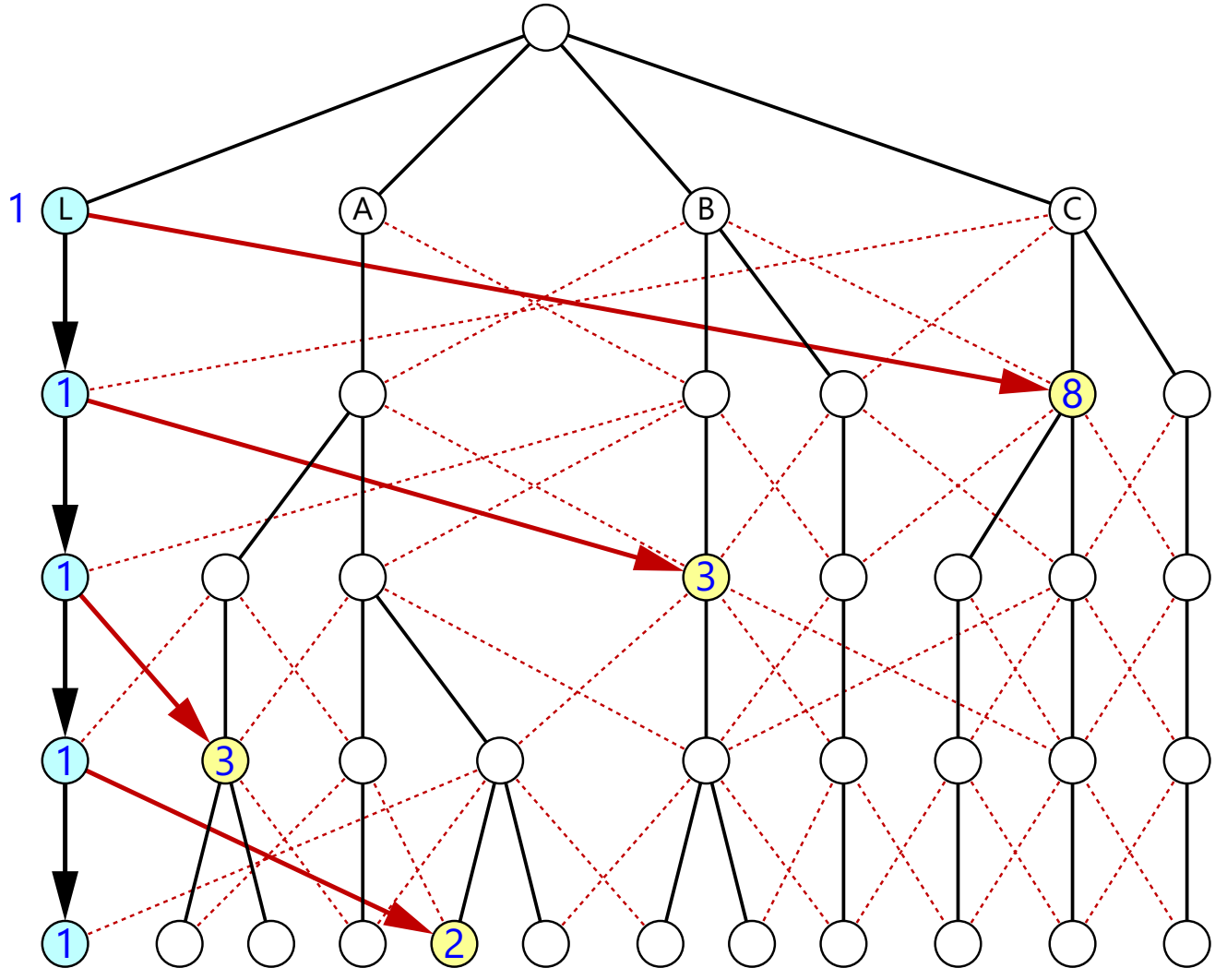


Guess on y_1: $\dfrac{4 \cdot 3 + 3 \cdot 1}{3} = 5$

Otherwise, the guess is an *overestimation* of the anonymity.

Single-Leader algorithm review

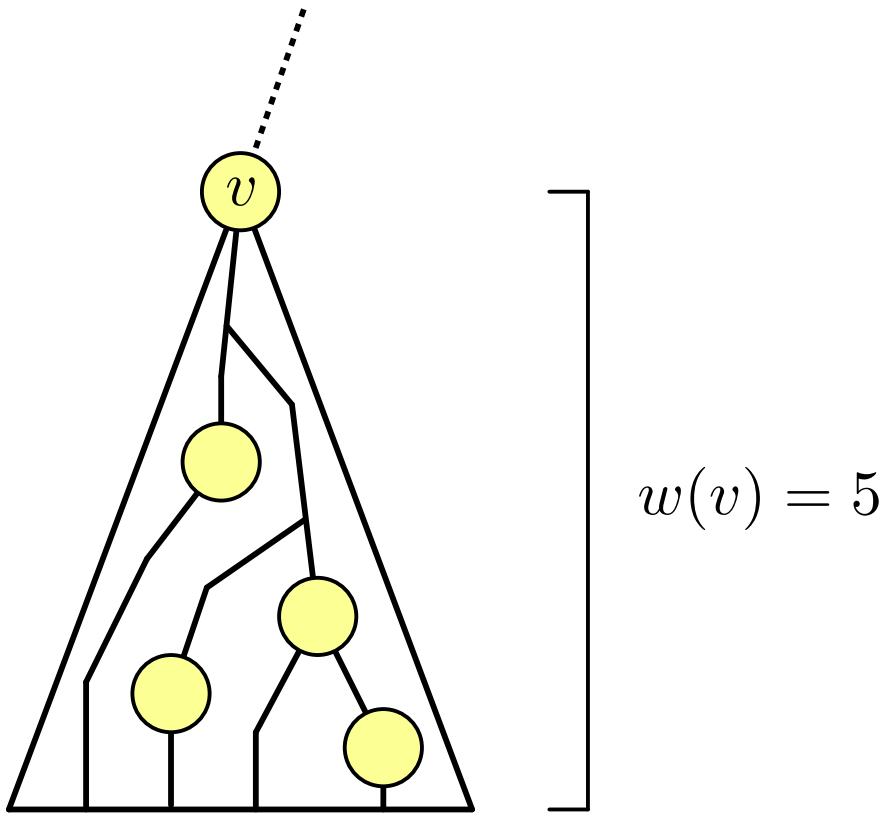
We can make one guess per round using the Leader's observations.



How do we know which guesses are correct?

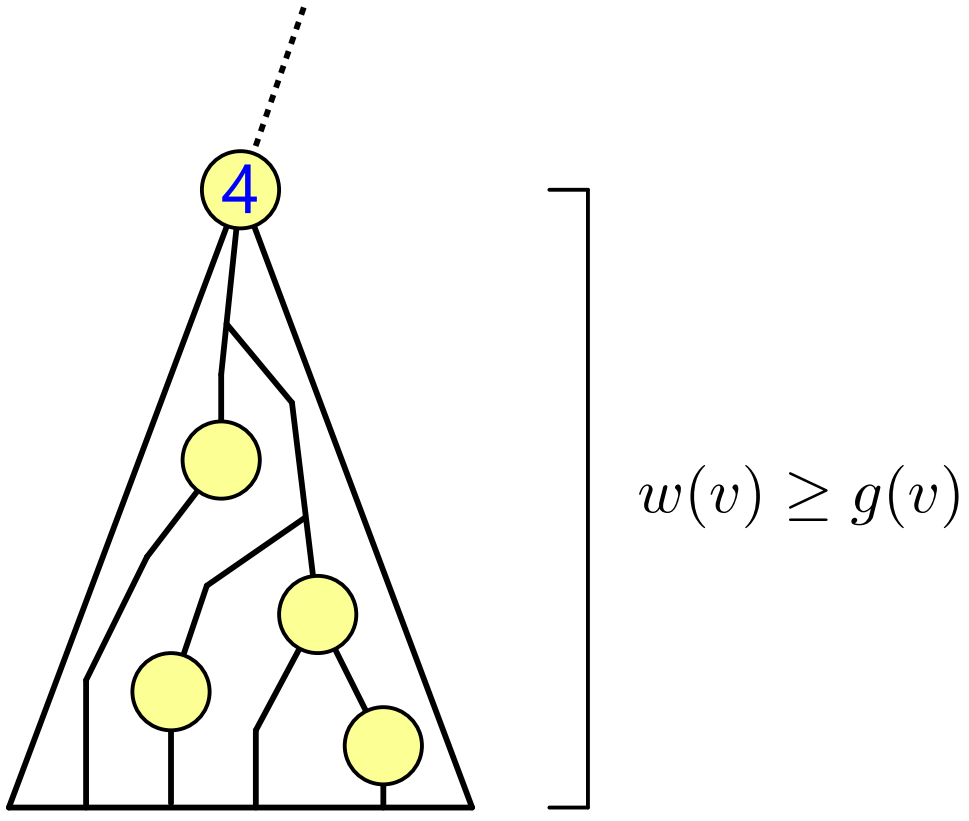
Single-Leader algorithm review

When a node v has a guess, we define its *weight* $w(v)$ as the number of nodes in the subtree hanging from v that have guesses.



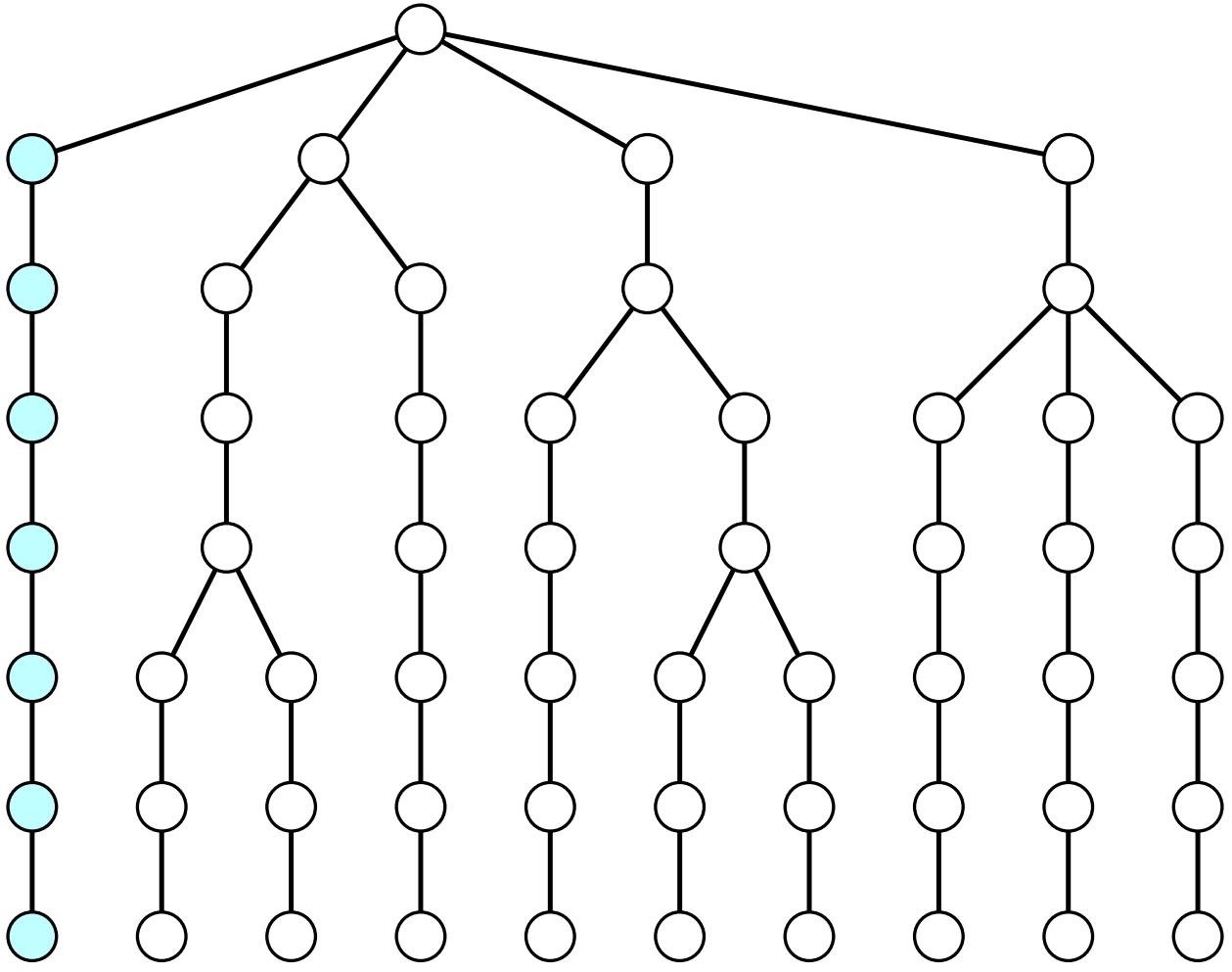
Single-Leader algorithm review

A node v is *heavy* if its weight $w(v)$ is at least as large as the value of its guess $g(v)$.



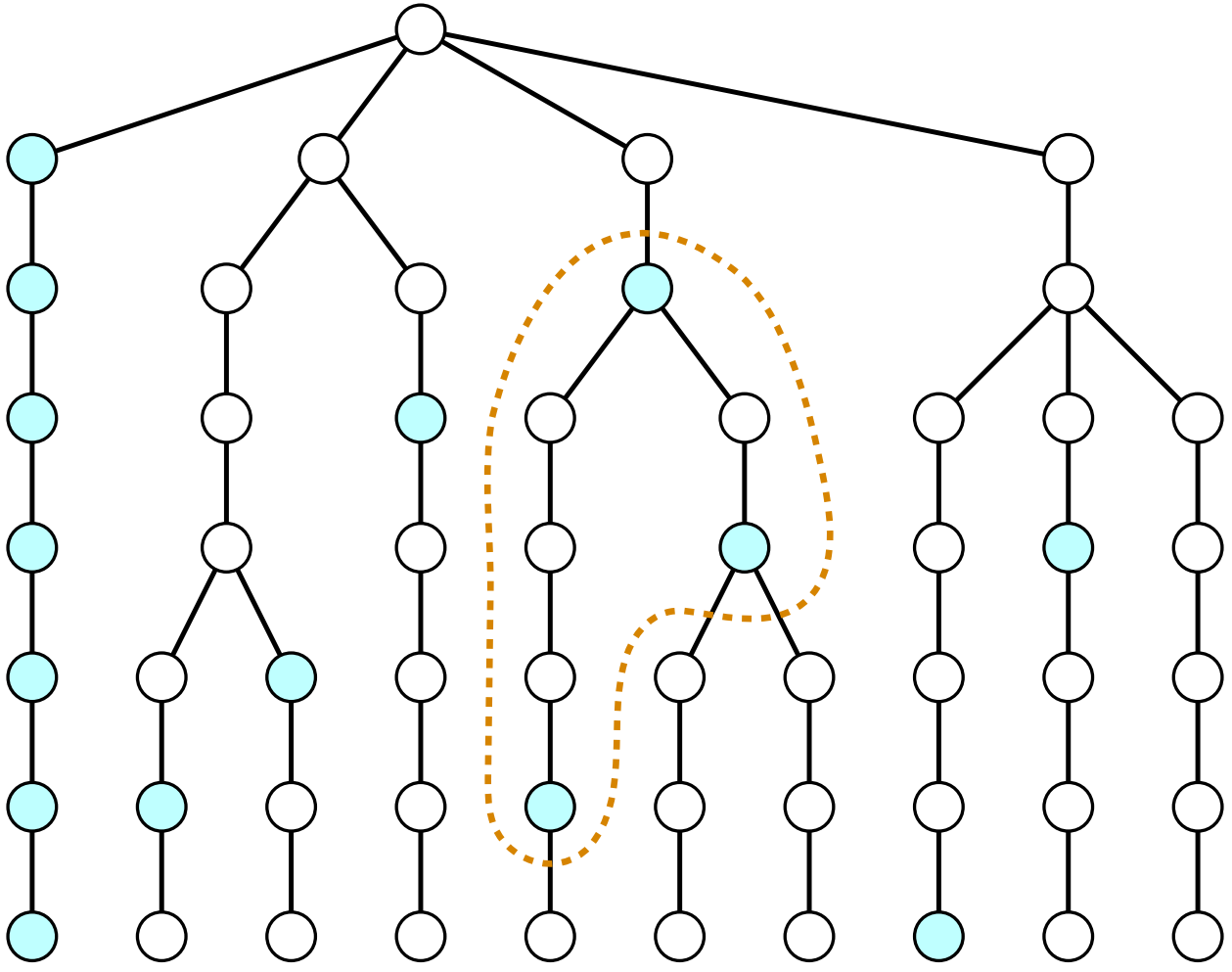
Correctness Criterion: If a node v is heavy and no descendants of v are heavy, then the guess on v is correct.

Single-Leader algorithm review



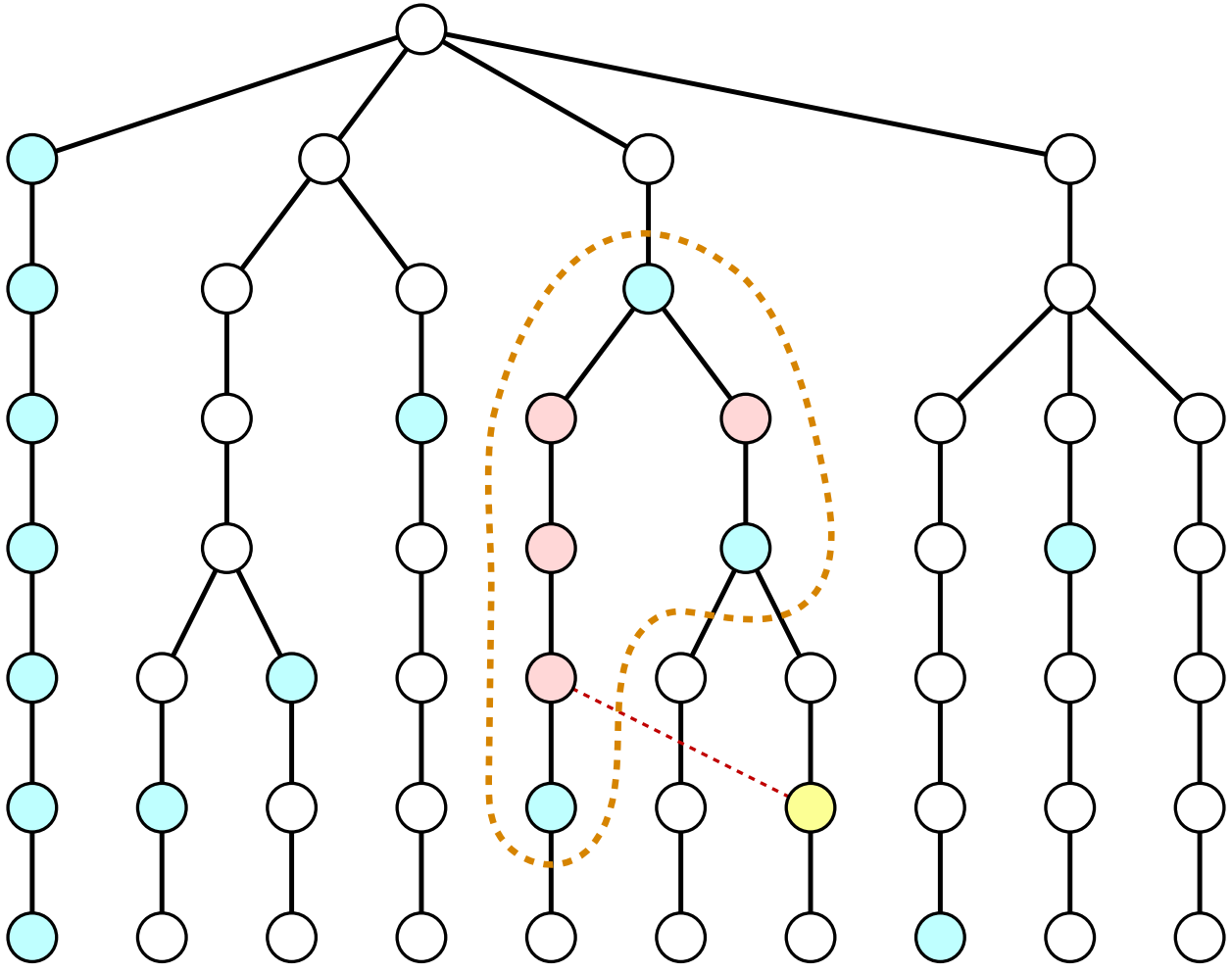
Initially, all Leader nodes are Guessers with anonymity $\ell = 1$.
Eventually, some guessed nodes become correct.

Single-Leader algorithm review



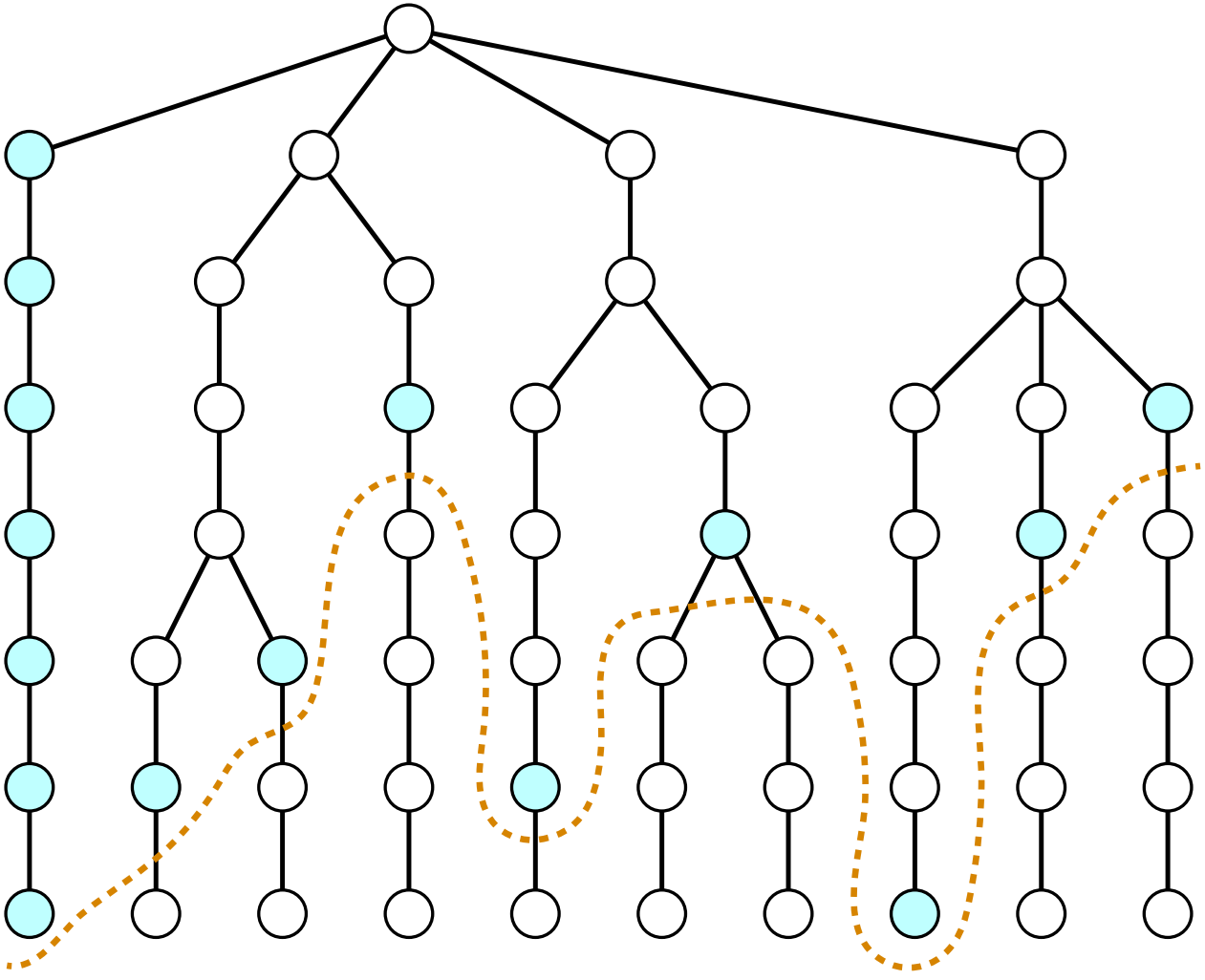
Correct nodes will form *islands*, which allow us to determine more anonymities.

Single-Leader algorithm review



In turn, all nodes whose anonymity is known and whose children's anonymities are known become new Guessers.

Single-Leader algorithm review

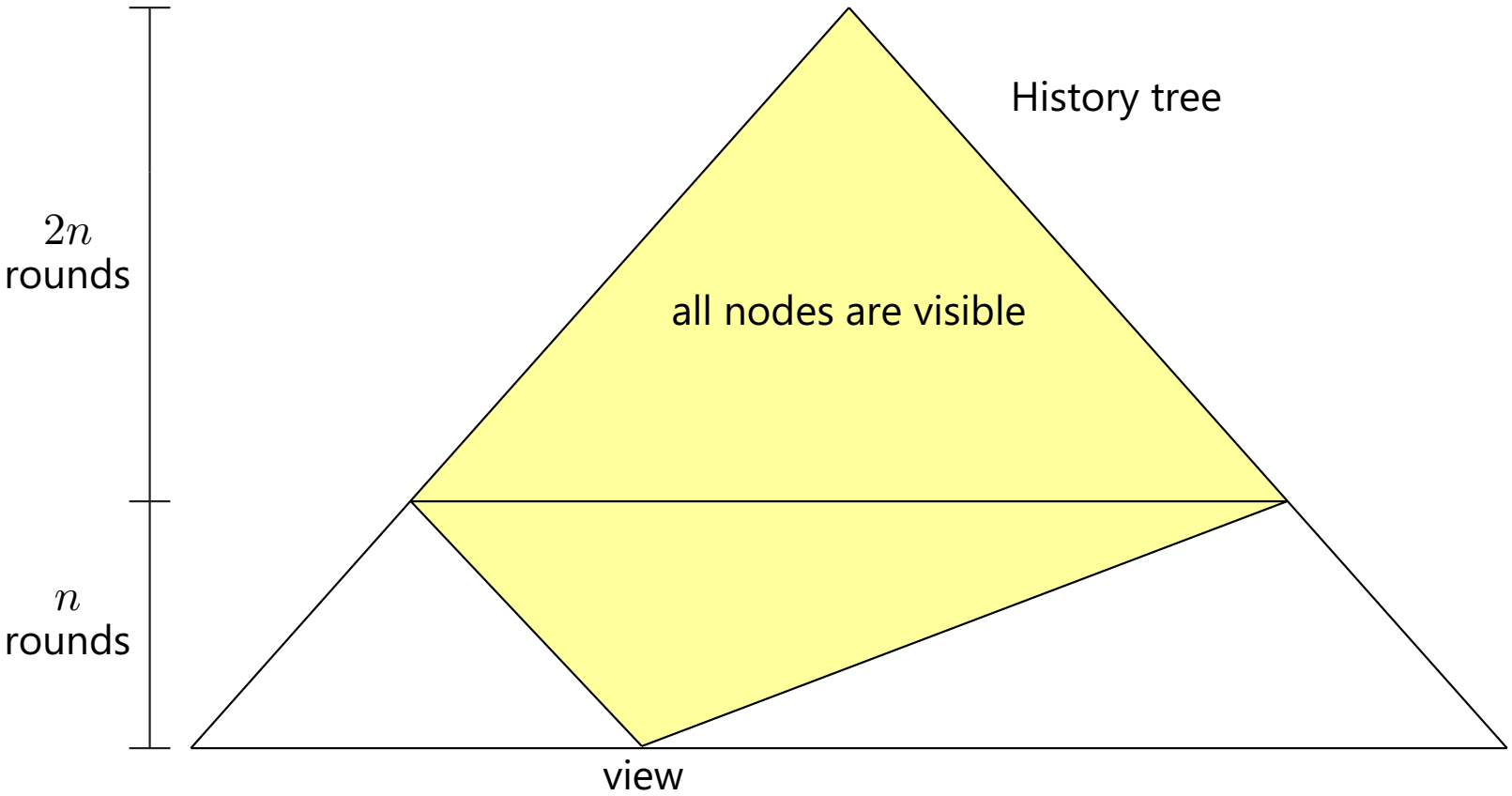


Eventually, some nodes determine a *cut* of the history tree, in which case we have an estimate n' on n, given by their sum.

Single-Leader algorithm review

Once we have a cut and an estimate $n' \leq n$, we wait n' rounds.

If $n' < n$, a new node appears in the first levels of the history tree.

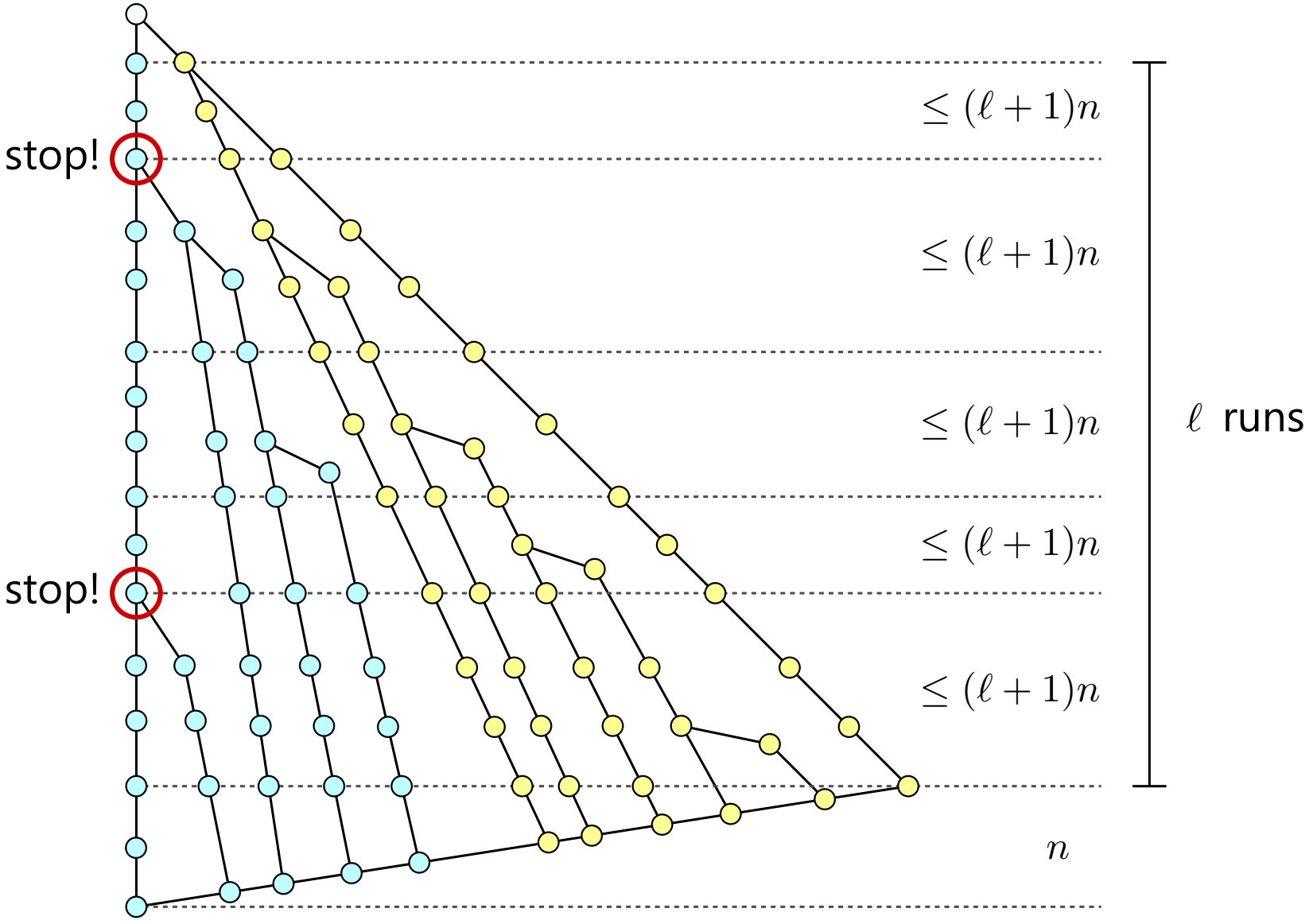


If $n' = n$, then no new nodes appear, and the algorithm terminates.

This happens within $3n$ rounds.

Multiple Leaders: Terminating algorithm

With $\ell > 1$ of Leaders, we do ℓ *runs* similar to the $\ell = 1$ algorithm.



Multiple Leaders: Terminating algorithm

Terminating algorithm for $\ell > 1$ Leaders:

- Do ℓ runs of the terminating algorithm for $\ell = 1$ as follows.
  - Choose a branch of Leader nodes and assign it anonymity x.
  - Run the terminating algorithm assuming $x = \ell$ (note that all guesses are still upper bounds of the real anonymities).
  - If we encounter a node where the chosen Leader branch splits, stop the current run and proceed with the next.
  - Else, the algorithm eventually terminates with an estimate n'_i of n, as well as an estimate of all nodes' anonymities.
  - If the sum of Leaders' anonymities is ℓ, store n'_i and proceed with the next run. Else, repeat this run with $x = \ell - 1$, etc.
- We end up with at most ℓ estimates of n: n'_1, n'_2, ... Wait another $\max\{n'_1, n'_2, \dots\}$ rounds to confirm them.
- If the n'_i's have not changed, they are all equal to n and all correct (note that at least one run must be correct, because the Leader nodes can split at most $\ell - 1$ times).

Multiple Leaders

Recall that the (Generalized) Counting Problem is *complete* for the class of multi-aggregate functions, which are all the functions that can be computed in anonymous networks with leaders.

Thus, we have the following result:

Theorem (Di Luna–V., DISC 2023)

Any problem that is solvable in τ-union-connected anonymous dynamic networks with $\ell \geq 1$ Leaders (assuming τ and ℓ are known) has a solution:

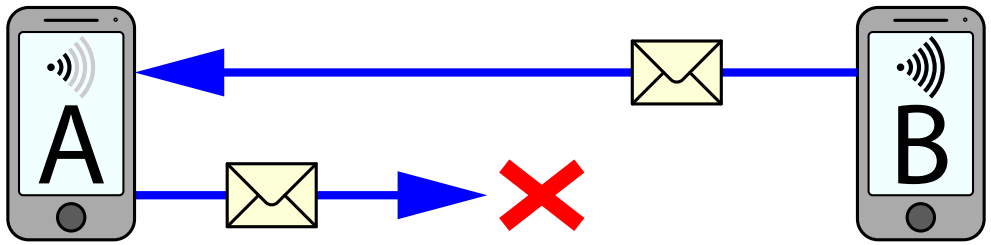
- *in $2\tau n$ rounds without explicit termination (stabilizing),*
- *in $(\ell^2 + \ell + 1)\tau n$ rounds with termination.*

$2\tau n$ rounds is a lower bound for the Counting Problem.

Directed Networks

Directed networks

We will now assume that the network is modeled by a (dynamic) *directed* graph that is (strongly) connected at all rounds.

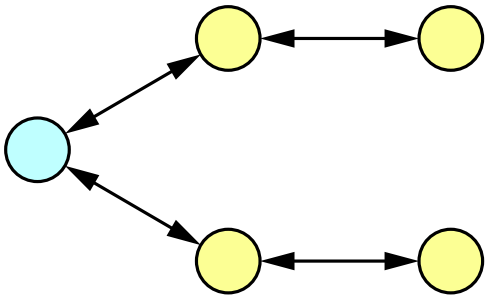


Recall that directed networks may model a situation where agents have different communication ranges.

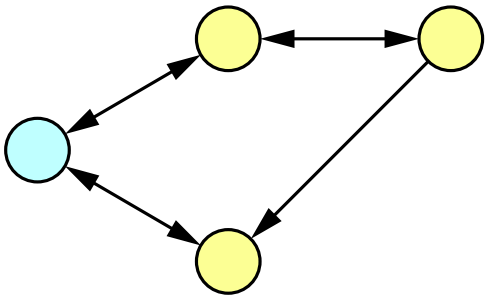
Outdegree awareness

The following example shows that some knowledge of the *outdegree* of an agent is necessary to do basic computations.

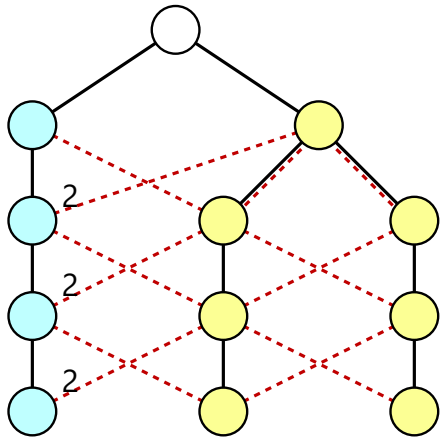
System 1



System 2



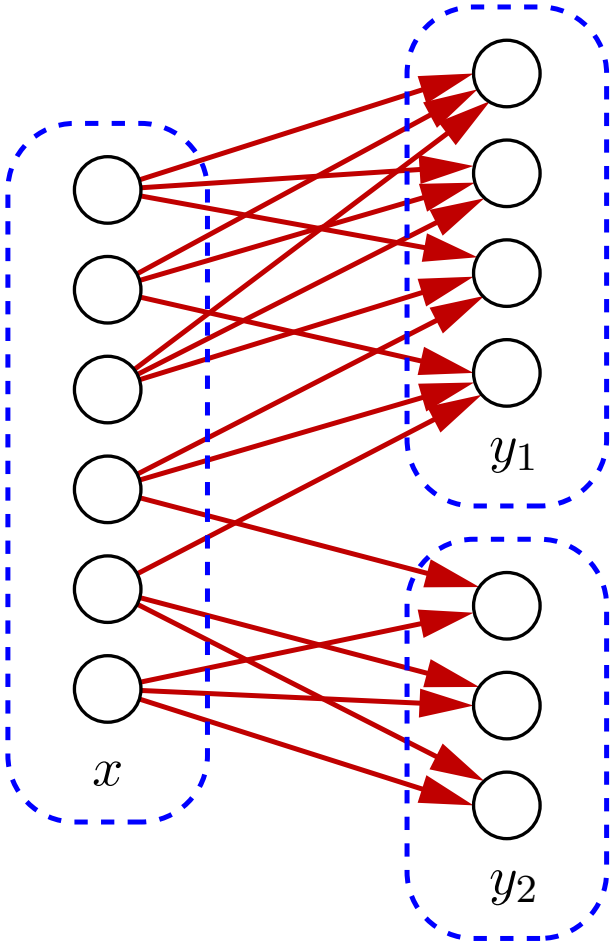
History tree

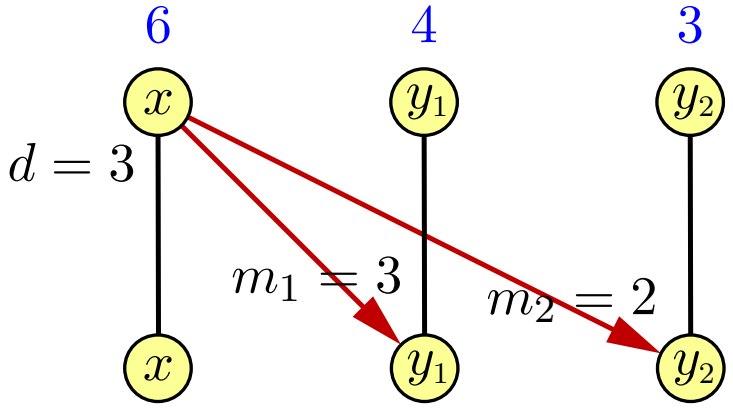


So, we will assume that an agent knows its outdegree *at the end* of each round (*weak outdegree awareness*).

Dynamic directed networks: Stabilization

In a directed network's history tree, if nodes do not branch, we have equations of the form $d \cdot a(x) = \sum_i m_i \cdot a(y_i)$, where d is the outdegree of the agents in x.





$3 \cdot 6 = 3 \cdot 4 + 2 \cdot 3$

Dynamic directed networks: Stabilization

Algorithm:

- Wait for a history tree level L_i with no branching nodes.
- Write the linear system of all equations given by L_i.
- Solve the linear system to find the anonymities of all nodes in L_i (as multiples of a free variable x).
- If there are ℓ leaders, set the sum of the corresponding nodes' anonymities to ℓ to find x and compute all anonymities.

This algorithm stabilizes in $2n$ rounds.

Proof of correctness:

The matrix A corresponding to the linear system is irreducible because the network is strongly connected. Also, $A = sI - P$, with $s > 0$ and $P \geq 0$. Note that P is also irreducible.

We have $Ax = 0$, where $x > 0$ is the vector of anonymities of the nodes in L_i. Hence, $Px = sx$. By the Perron-Frobenius theorem, s is a simple eigenvalue of P, and thus 0 is a simple eigenvalue of A. Therefore, the rank of A is $n - 1$.

Dynamic directed networks: Termination

Algorithm for networks with $\ell \geq 1$ Leaders:

- Wait for a long-enough sequence of levels with no branching nodes. Let b_1 be a Leader branch, and let $U_{b_1} = \ell$.
- Repeat (assuming that $U_{b_i} \geq a(b_i)$):
  - Take a branch b_{i+1} that gets messages from b_i for U_{b_i} rounds.
  - Each of these interactions gives us an estimate on $a(b_{i+1})$.
  - One of the estimates is actually an upper bound on $a(b_{i+1})$, because it occurs when b_i does not branch.
  - Let $U_{b_{i+1}}$ be the maximum of all estimates on $a(b_{i+1})$.

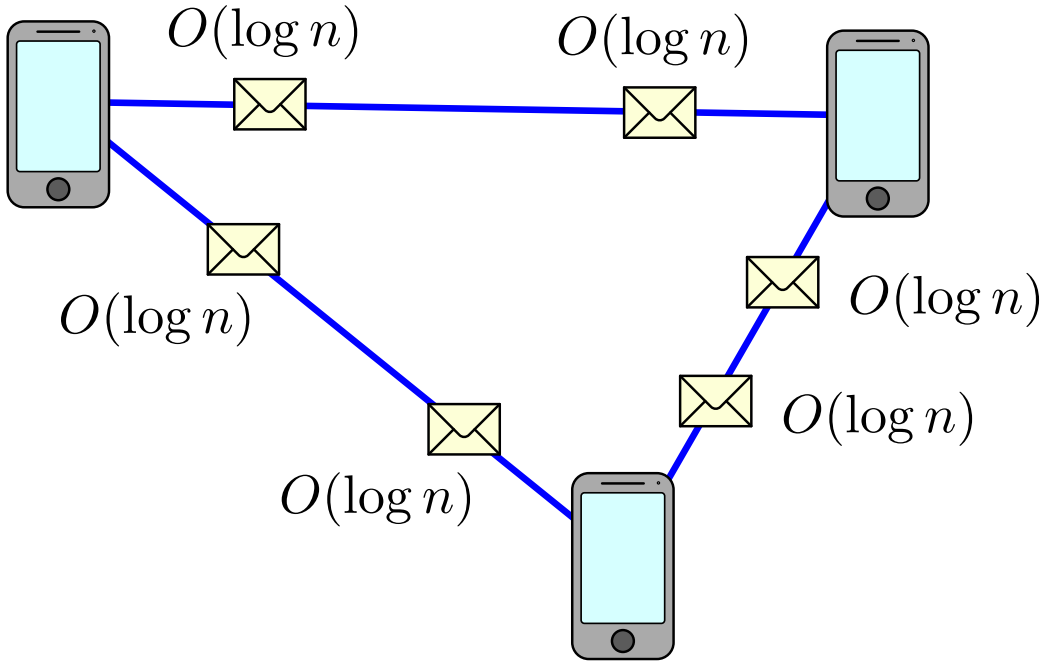
  Until we have an upper bound $M = \sum_i U_{b_i}$ on the anonymities of all visible branches.
- Wait for M rounds to confirm there are no invisible branches.
- Run the stabilizing algorithm to compute n.

This algorithm terminates in $O(n^{n+2})$ rounds.

Congested Networks

Congested networks

Normally, there is no limit to the size of messages that can be sent through a network's links. Any kind of information can be sent.



In a *congested network*, each message must have size $O(\log n)$. This is a severe limitation on how much information can be sent.

Bounds on the Counting Problem

LOCAL model (unlimited message size):

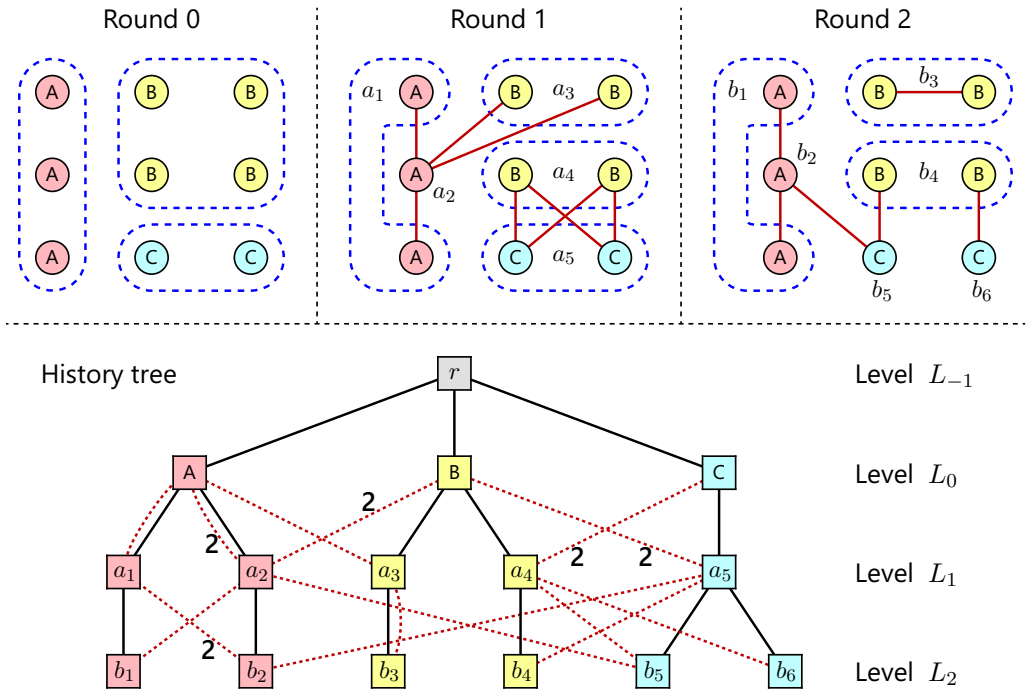
- **Looks impossible...** [Michail et al., SSS 2013]
- $\mathbf{O(n^{n+4})}$ **rounds** [Di Luna–Baldoni, OPODIS 2015]
- $\mathbf{O(n^5 \log^2 n)}$ **rounds** [Kowalski–Mosteiro, ICALP 2018 Best Paper]
- $\mathbf{O(n^{4+\epsilon} \log^3 n)}$ **rounds** [Kowalski–Mosteiro, ICALP 2019]
- $\mathbf{3n}$ **rounds (optimal)** [Di Luna–V., FOCS 2022]

CONGEST model ($O(\log n)$ message size):

- Lower bound of $\mathbf{\Omega(n^2/\log n)}$ **rounds** [Dutta et al., SODA 2013]
- $\mathbf{O(n^{5+\epsilon} \log^3 n)}$ **rounds** [Kowalski–Mosteiro, ArXiV 2022]
- $\mathbf{O(n^3)}$ **rounds** [Di Luna–V., PODC 2023 Brief Announcement]

History trees

As we know, in the LOCAL model we typically use *history trees*, which are a structure that naturally captures the idea that anonymous agents become distinguishable as soon as they have different "histories".

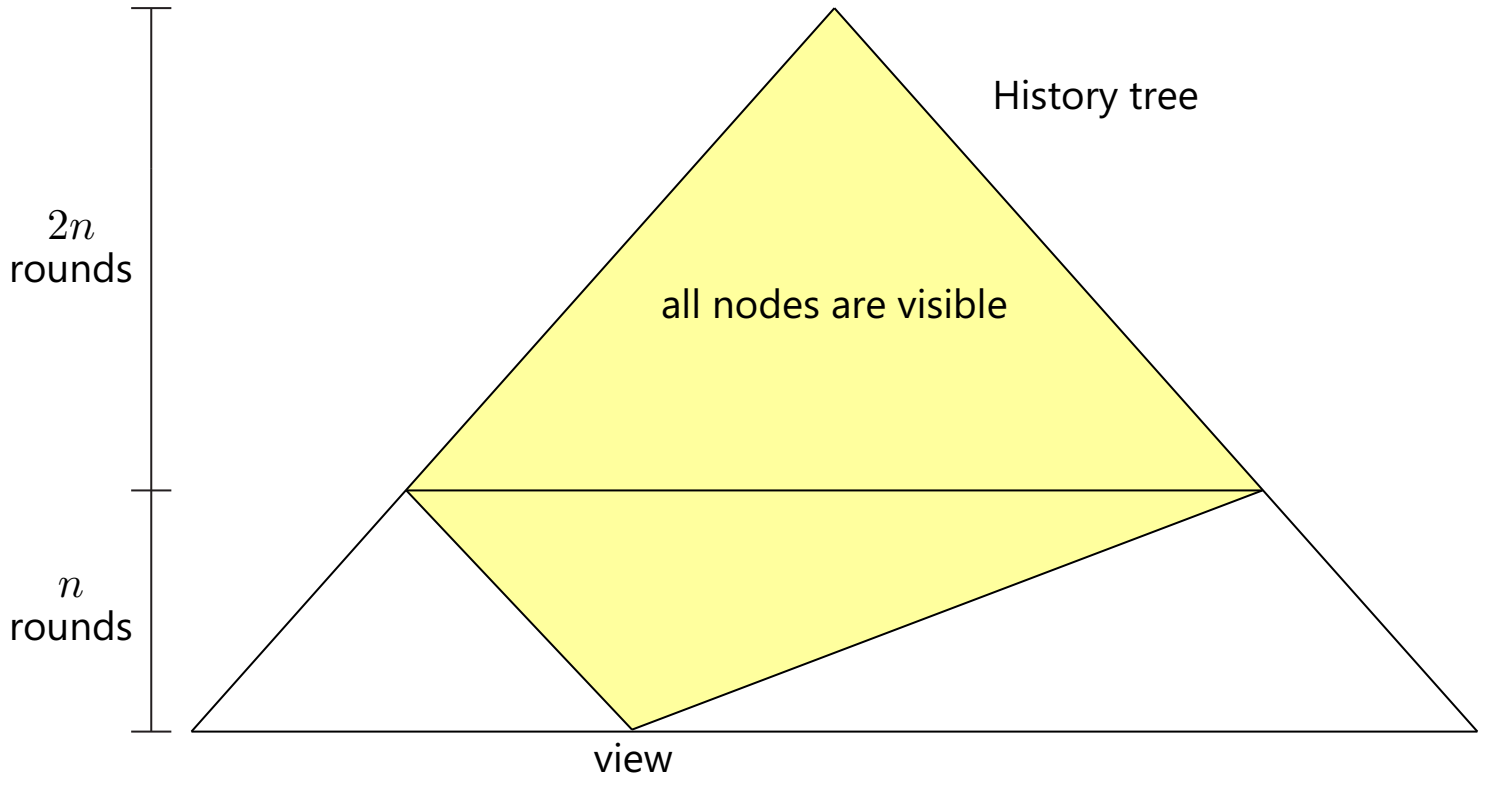


Constructing and sharing history trees is a way of doing arbitrary computations in linear time. Unfortunately, the size of a history tree is $O(n^3 \log n)$, unsuitable for the CONGEST model.

Outline of Counting algorithms

Non-congested networks (LOCAL algorithm):

- Each agent constructs its view of the history tree, updating it at every round based on the messages it receives.
- After $3n$ rounds, the view of each agent has enough information to solve the Counting problem.

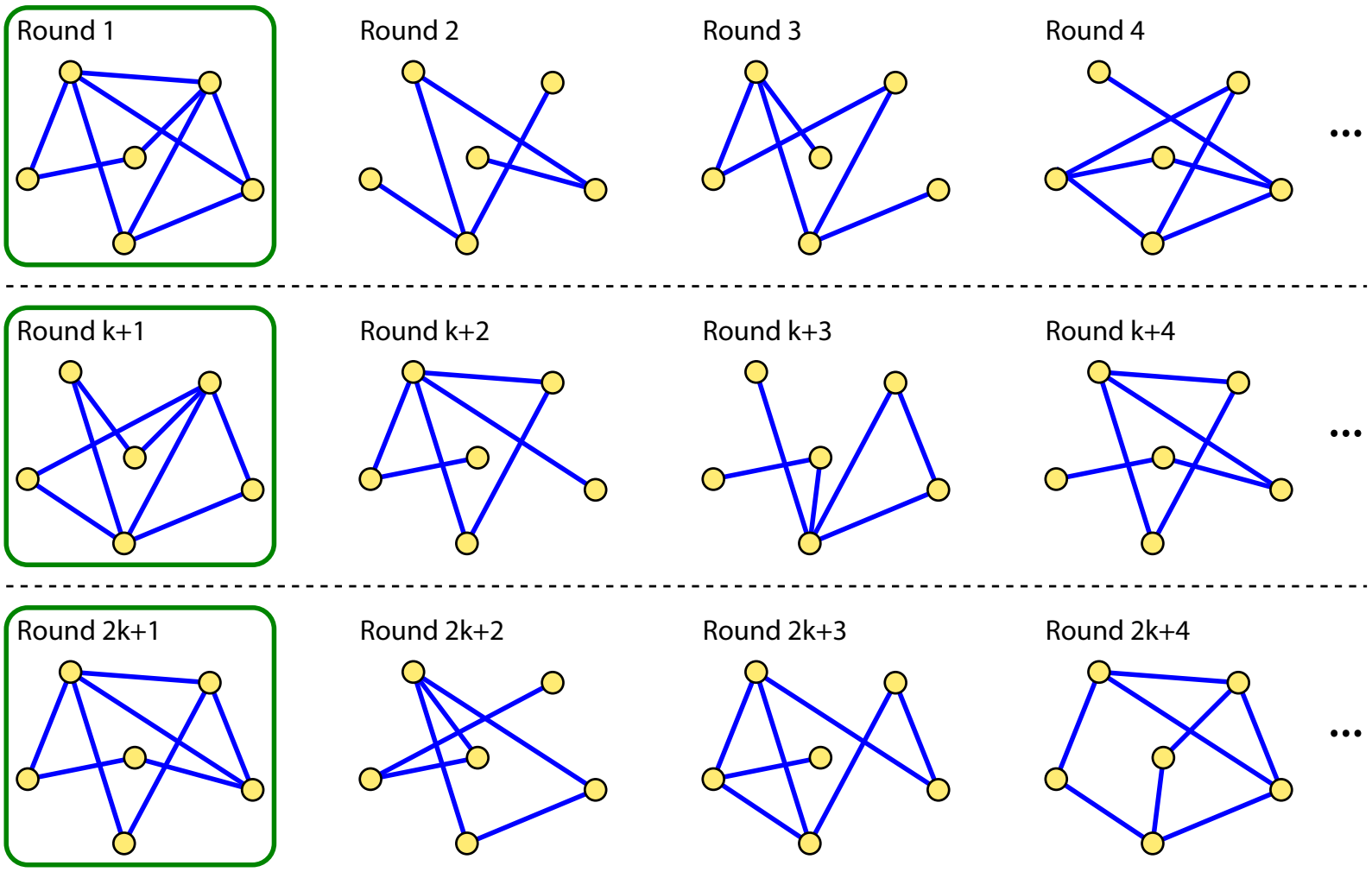


Outline of Counting algorithms

Congested networks (CONGEST algorithm):

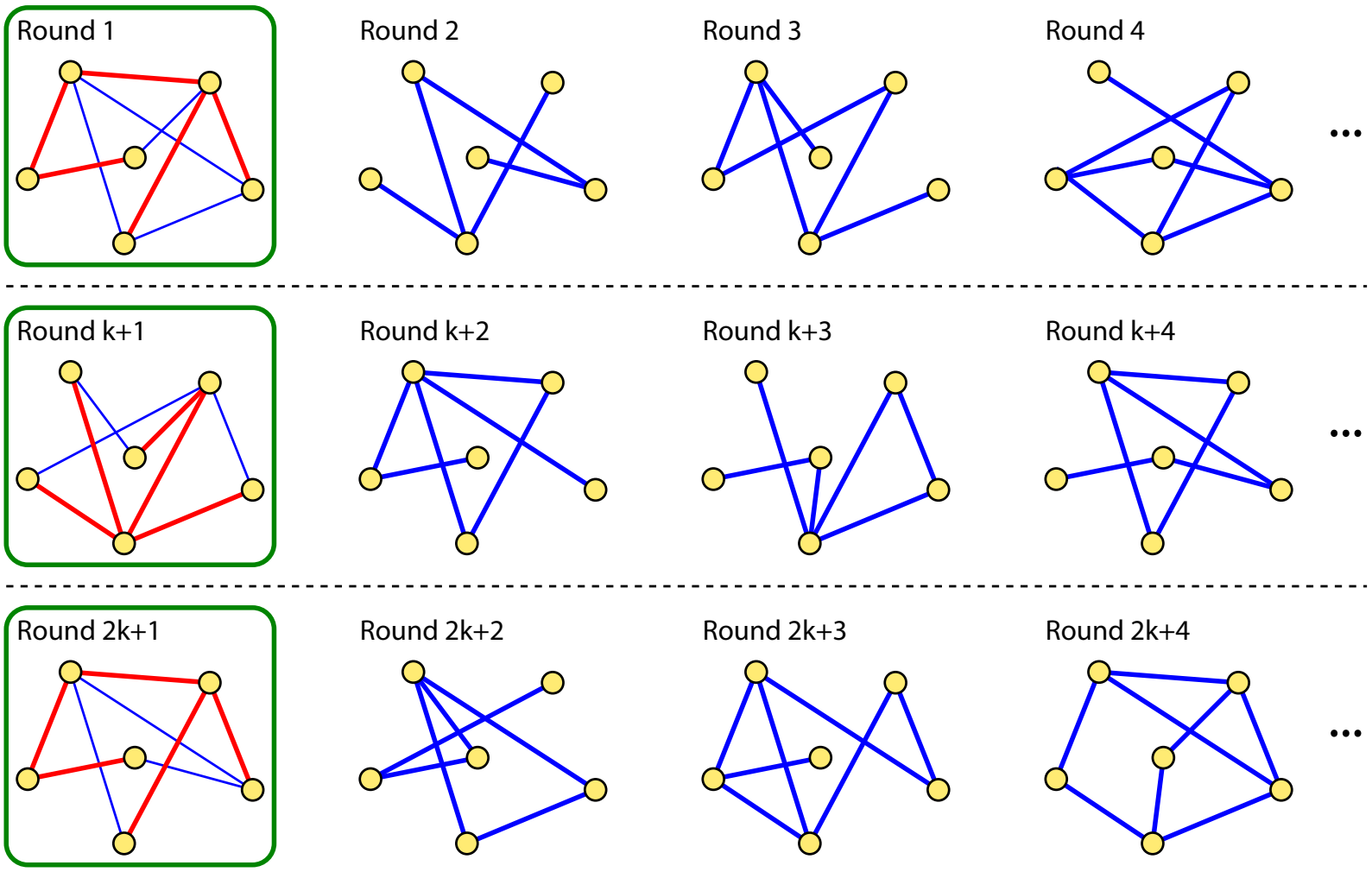
- Construct the history tree of some "artificial" network with n agents (we call it the *official history tree*, OHT).
- Each level of the OHT is constructed from a spanning tree of the real network at some round (so each level takes $O(n \log n)$ bits).
- The construction is done in a series of broadcasts, each of which results in a $O(\log n)$-size piece of information σ reaching the Leader.
- When the Leader receives σ, it records it in the OHT and broadcasts σ back to the other agents (acknowledgment).
- Assign temporary IDs to agents; when two agents get disambiguated due to some information becoming official, change their IDs.
- An estimate U on the size of the network is used to time the broadcasts; if it is determined that $U < n$, double U and reset.
- Whenever a new level of the OHT is complete, run the LOCAL algorithm on the OHT and see if it can determine n.

Construction of the OHT



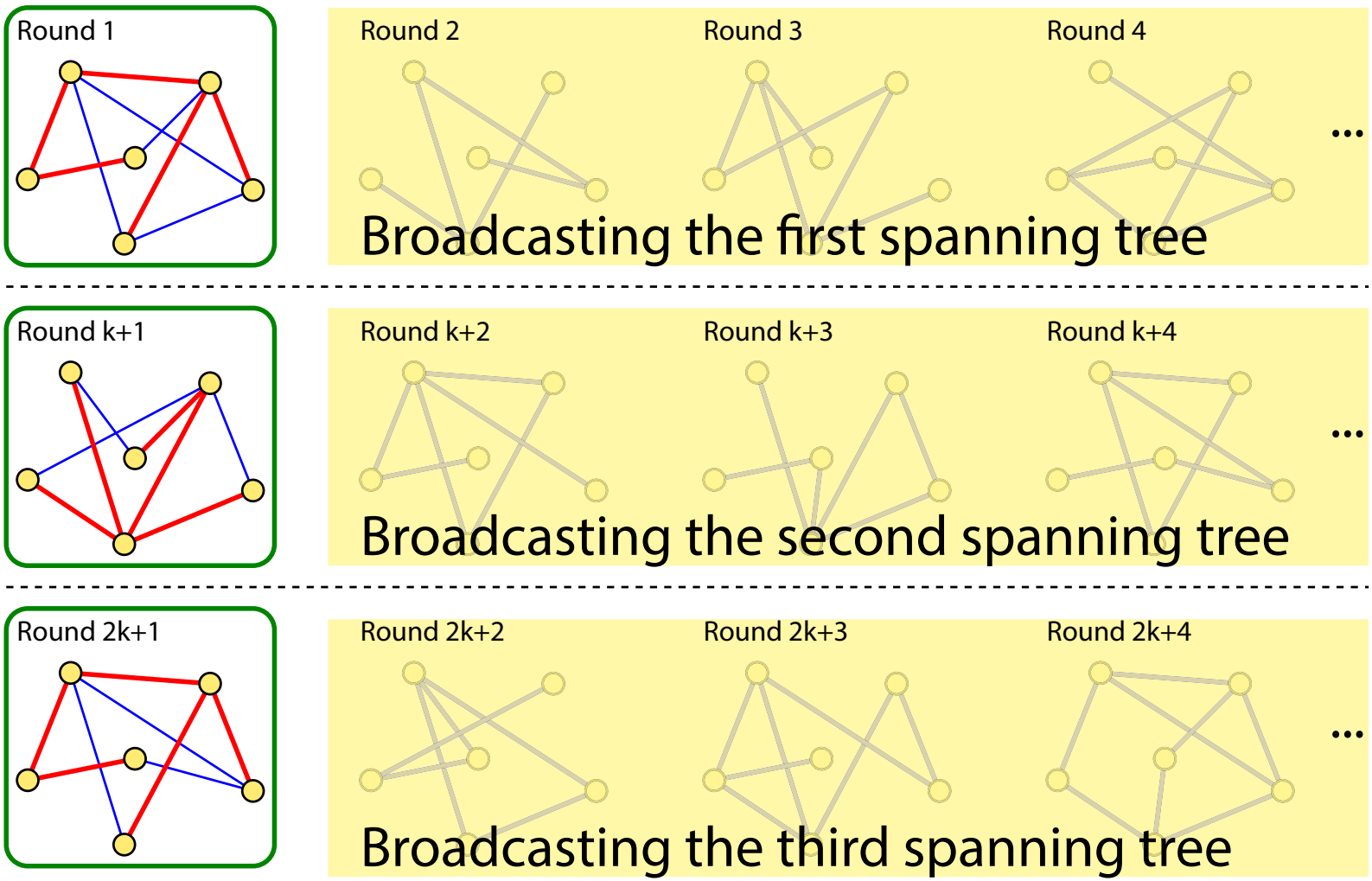
The OHT is constructed on a *subsequence* of selected network rounds.

Construction of the OHT



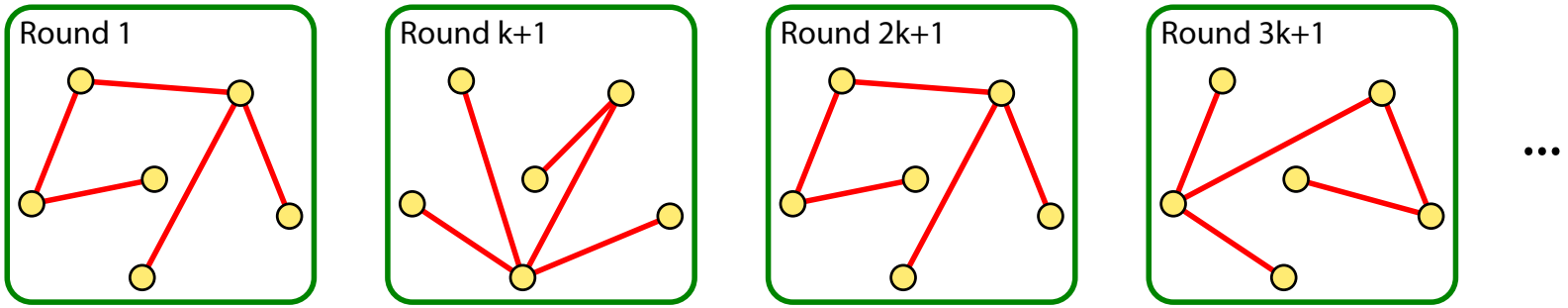
Take a spanning tree of the network at the selected rounds.

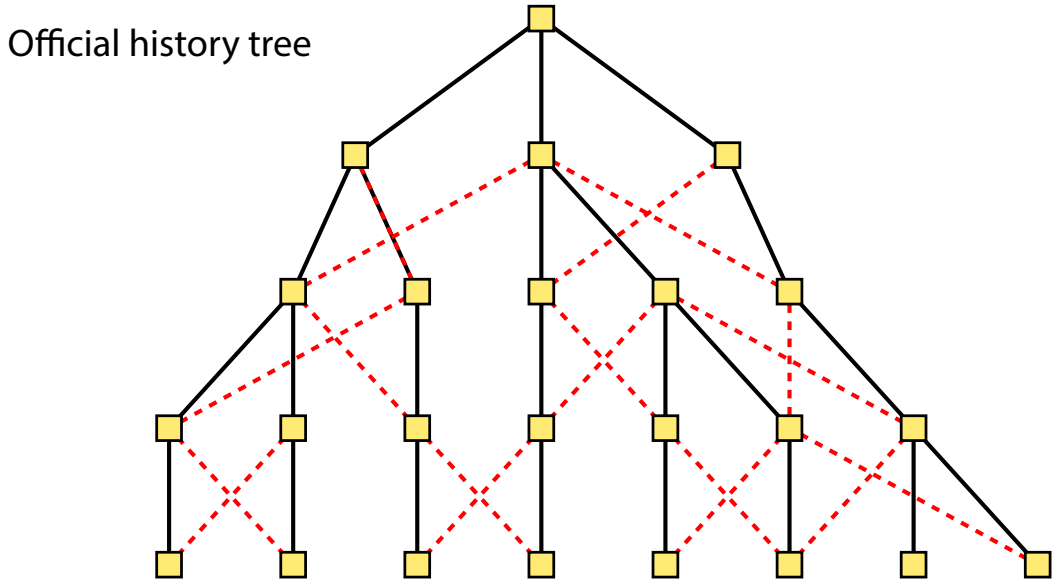
Construction of the OHT



For each spanning tree, broadcast enough information to construct a level of the OHT. It takes $O(n)$ broadcasts for each level.

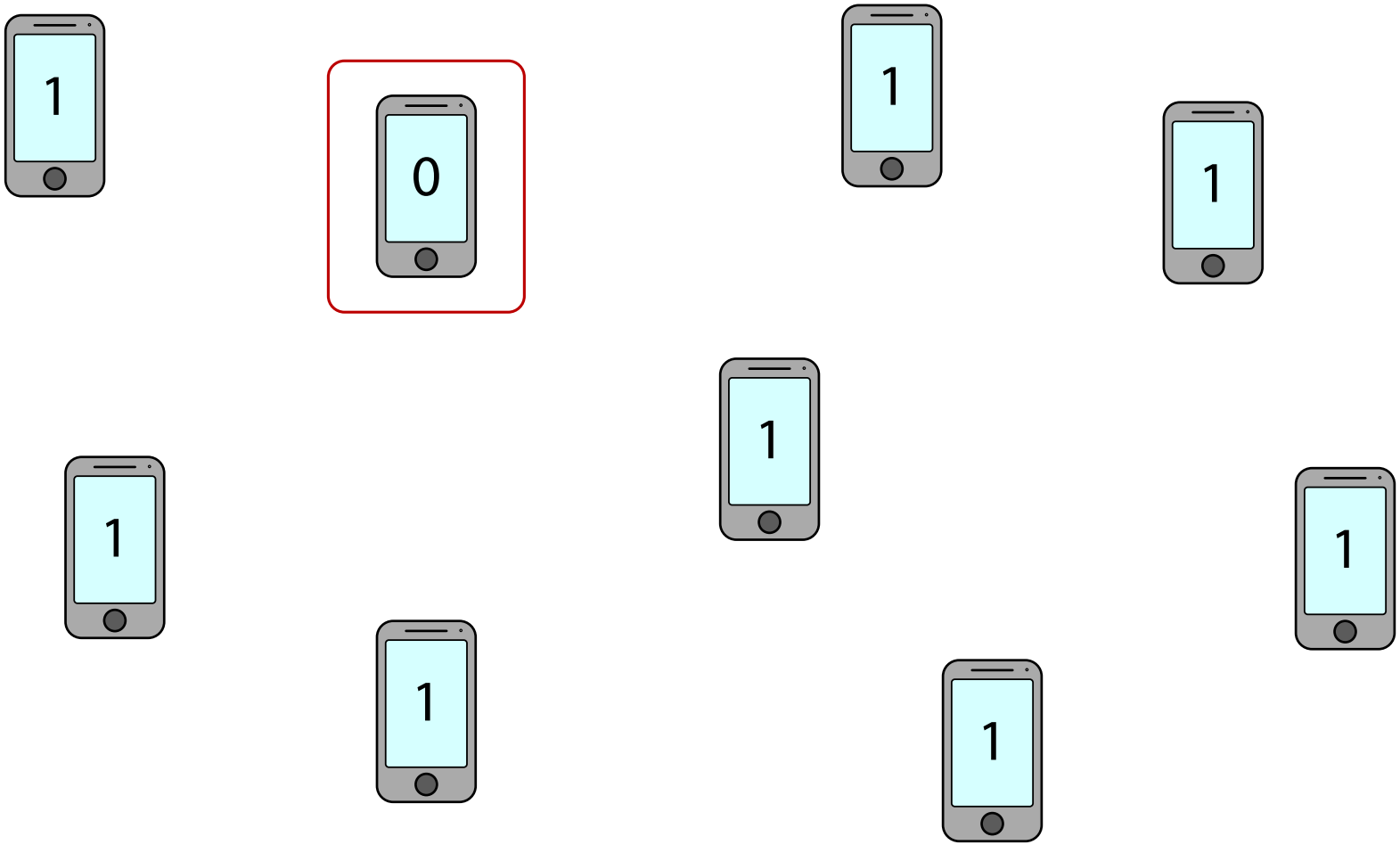
Construction of the OHT





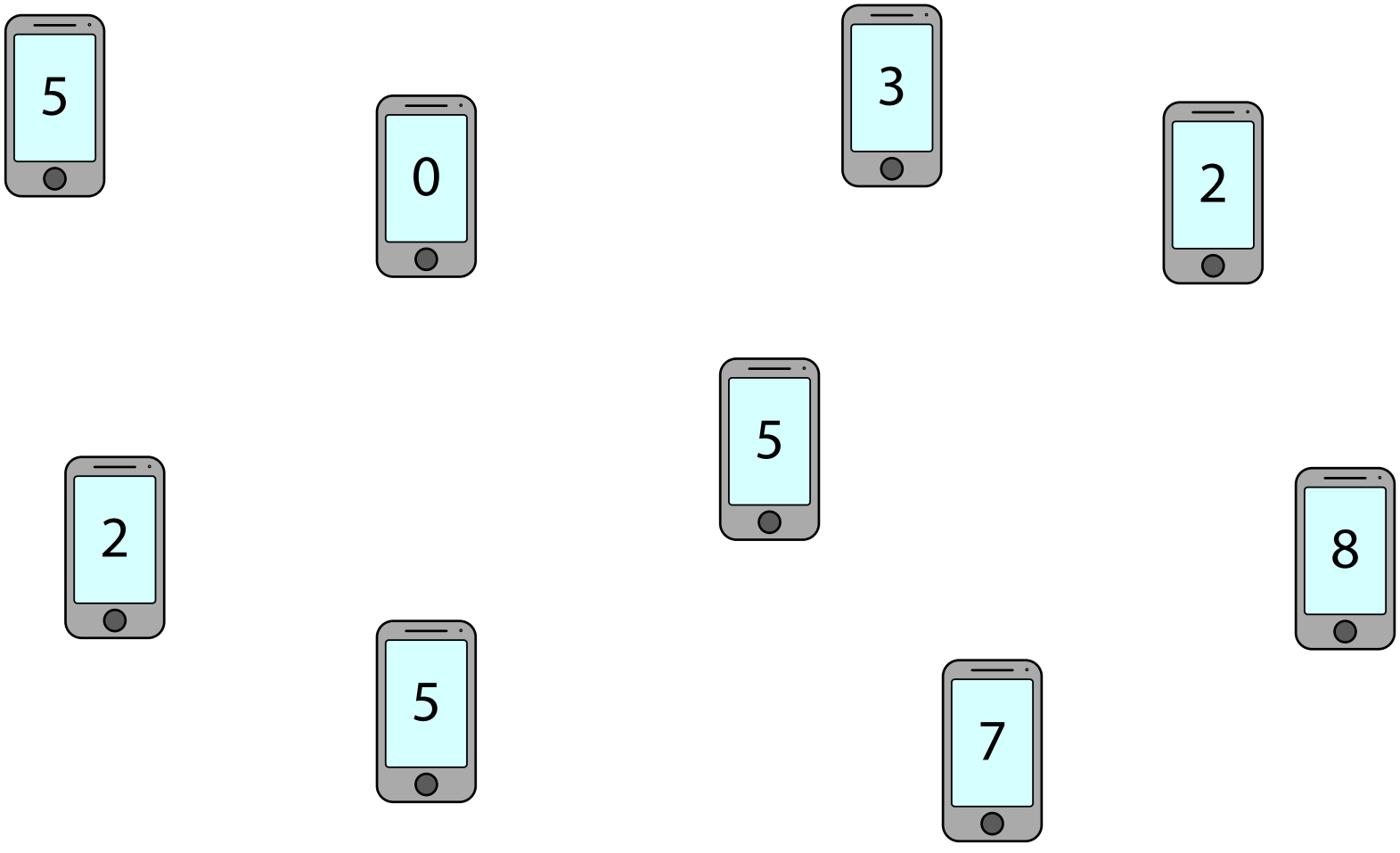
Construct the OHT level by level; continue until there are enough levels for the LOCAL Counting algorithm to compute n.

Constructing a level of the OHT



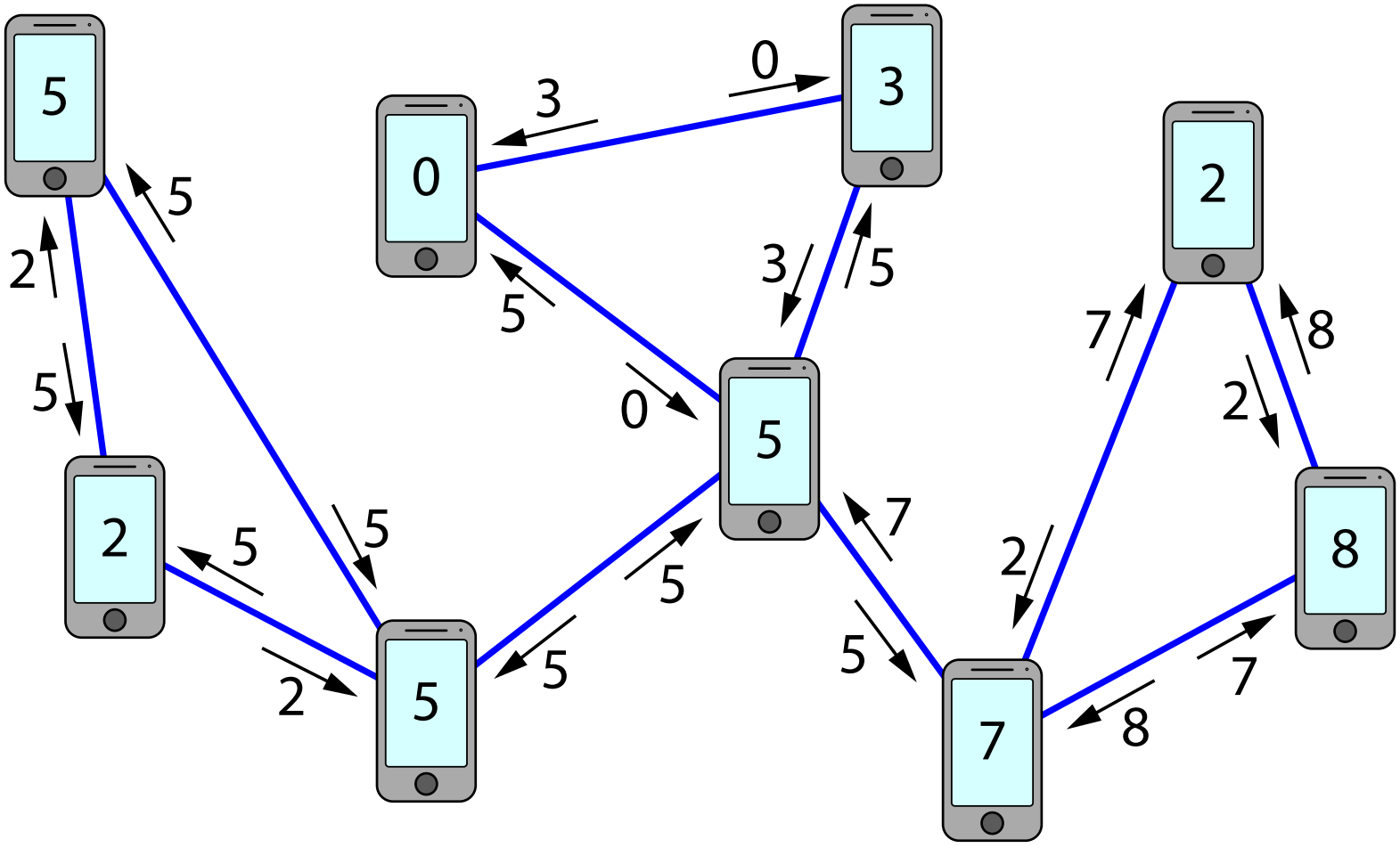
We assign IDs to agents: Initially, the Leader has ID 0, and the non-Leaders have ID 1. IDs may be modified over time...

Constructing a level of the OHT



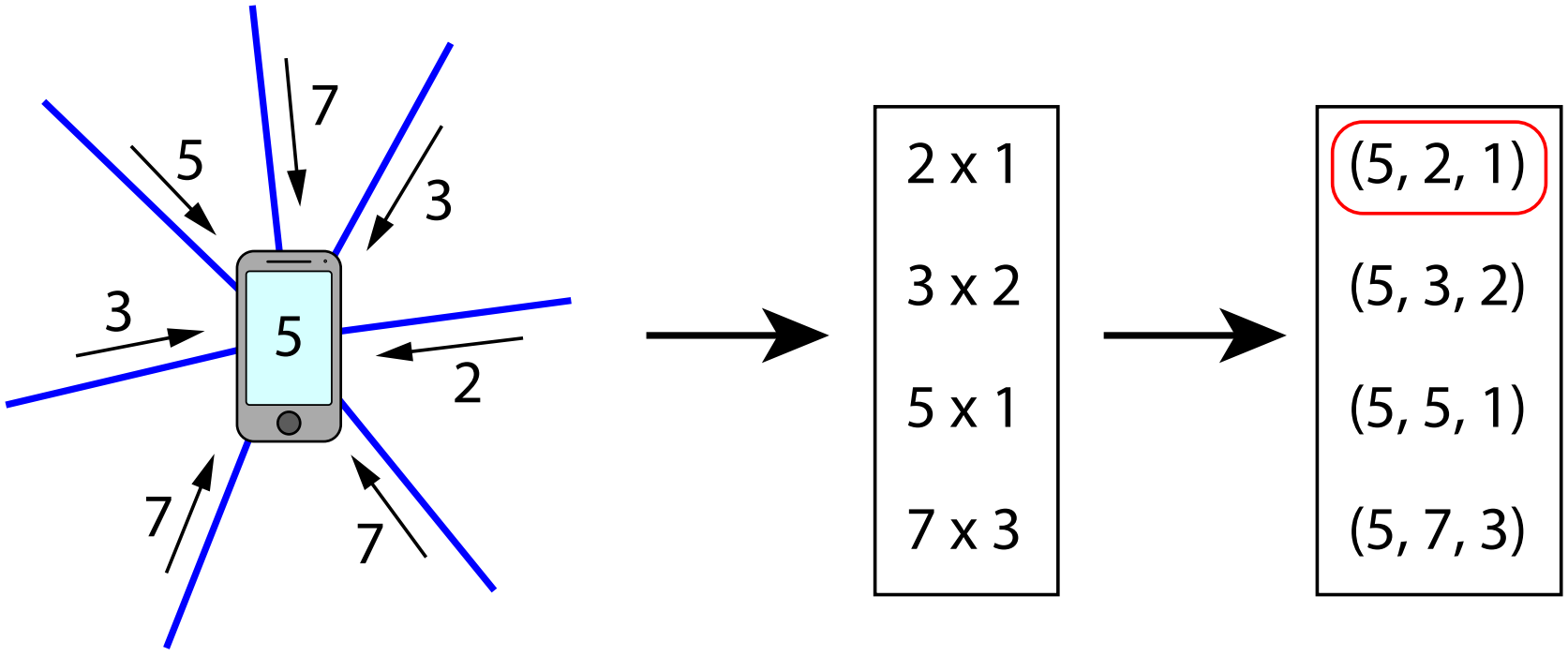
We assign IDs to agents: Initially, the Leader has ID 0, and the non-Leaders have ID 1. IDs may be modified over time...

Constructing a level of the OHT



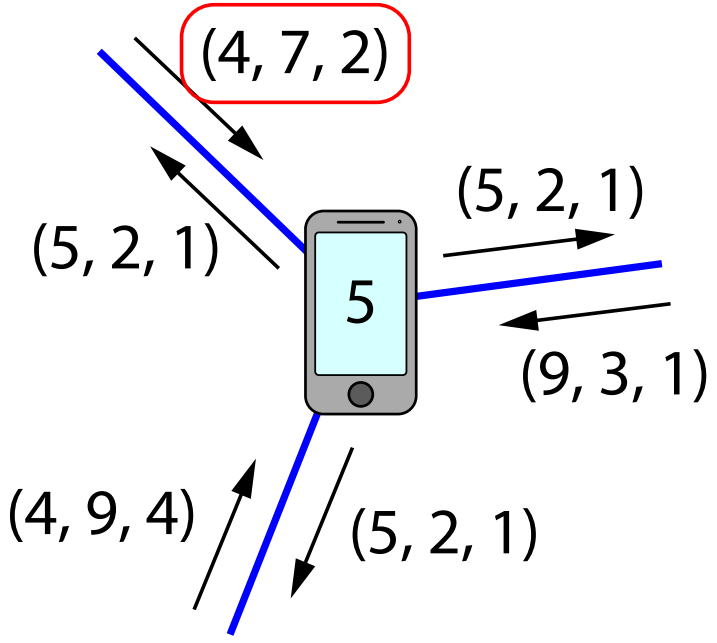
At a selected round, each agent sends its ID to all neighboring agents. Thus, each agent receives a *multiset* of IDs.

Constructing a level of the OHT



This multiset is converted into triplets $(\mathsf{ID1}, \mathsf{ID2}, m)$, meaning "An agent named ID1 received m messages from agents named ID2".

Constructing a level of the OHT



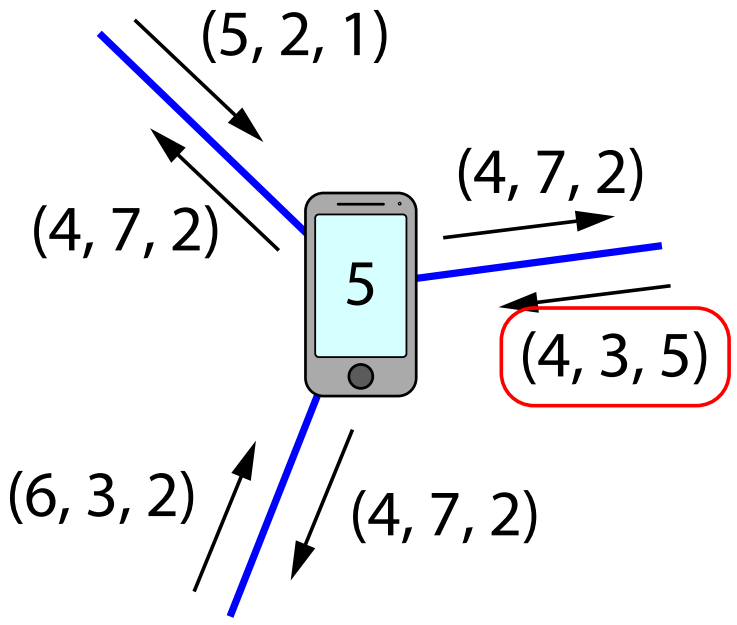
Minimum triplet:

~~(5, 2, 1)~~

(4, 7, 2)

The triplets are sorted lexicographically, and the smallest is broadcast for the next rounds.

Constructing a level of the OHT



Minimum triplet:

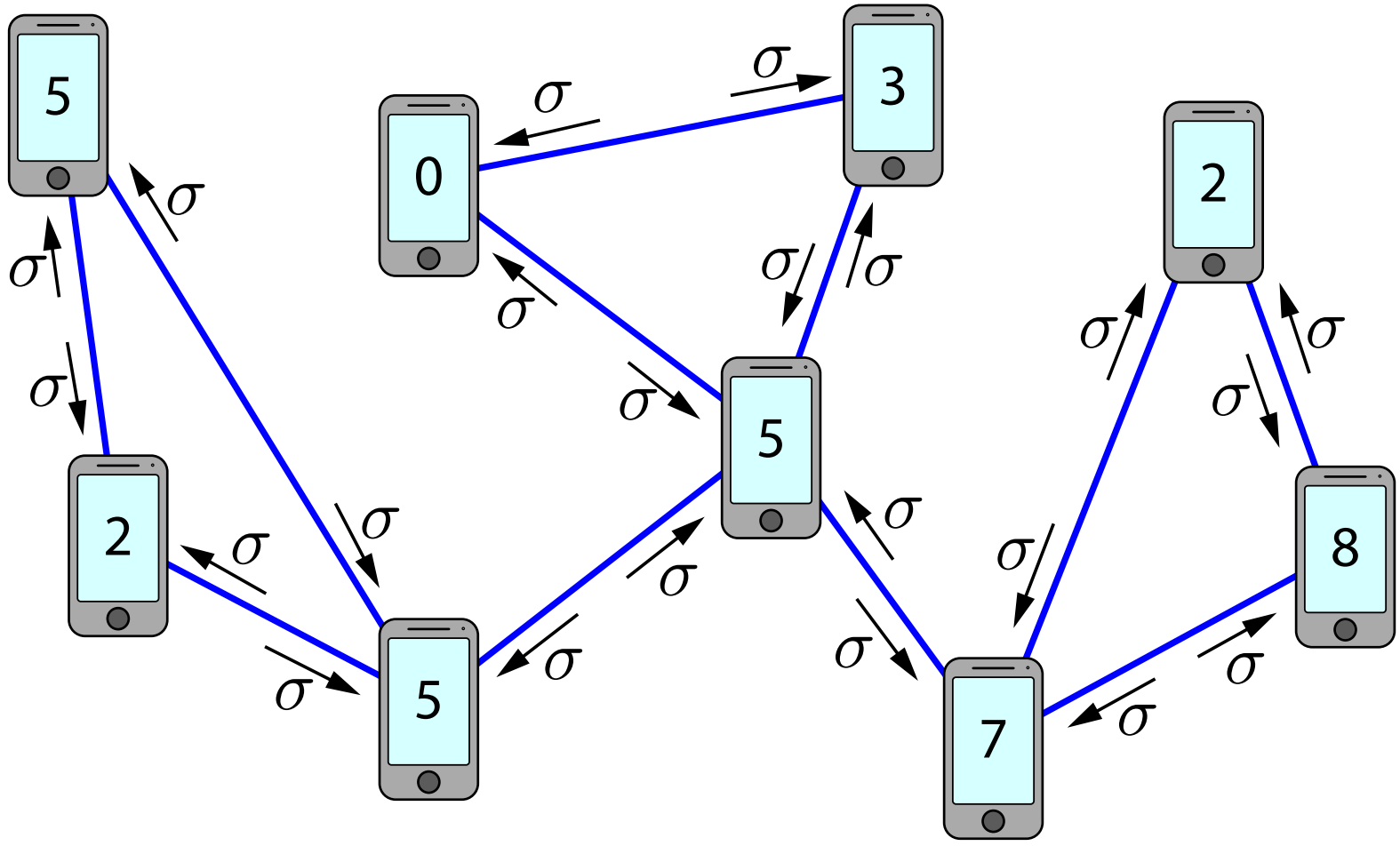
~~(5, 2, 1)~~

~~(4, 7, 2)~~

(4, 3, 5)

When an agent receives a triplet that is lexicographically smaller than the one it is currently broadcasting, it broadcasts the new one.

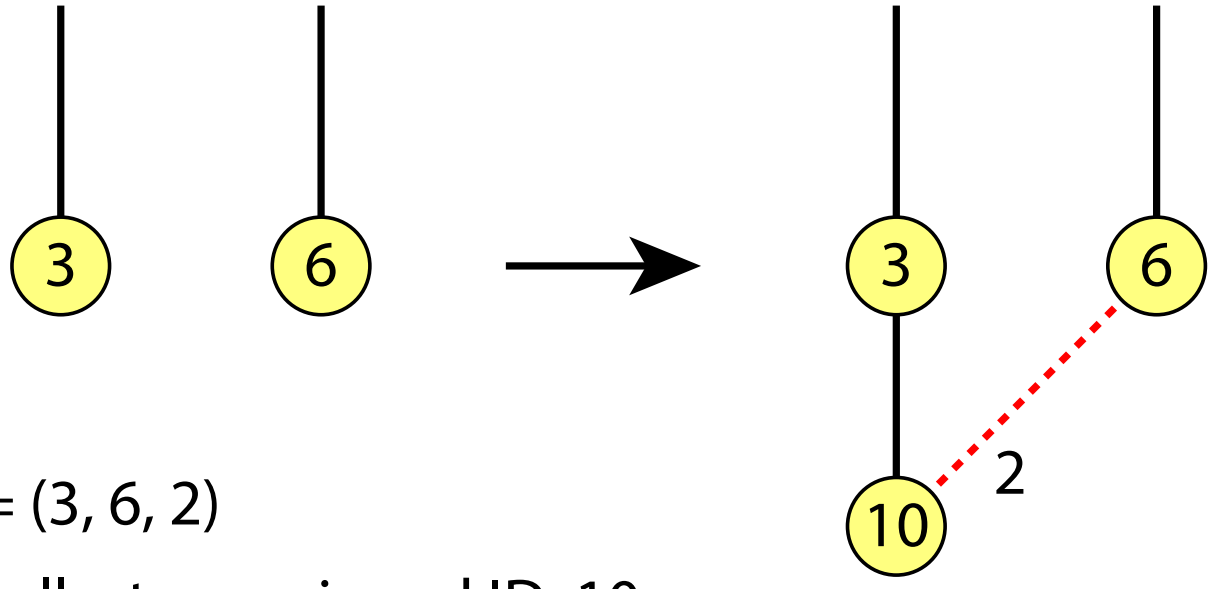
Constructing a level of the OHT



The broadcast continues for a certain number of rounds (more on this later...) until all agents get the minimum triplet σ.

Constructing a level of the OHT

Official history tree modification:

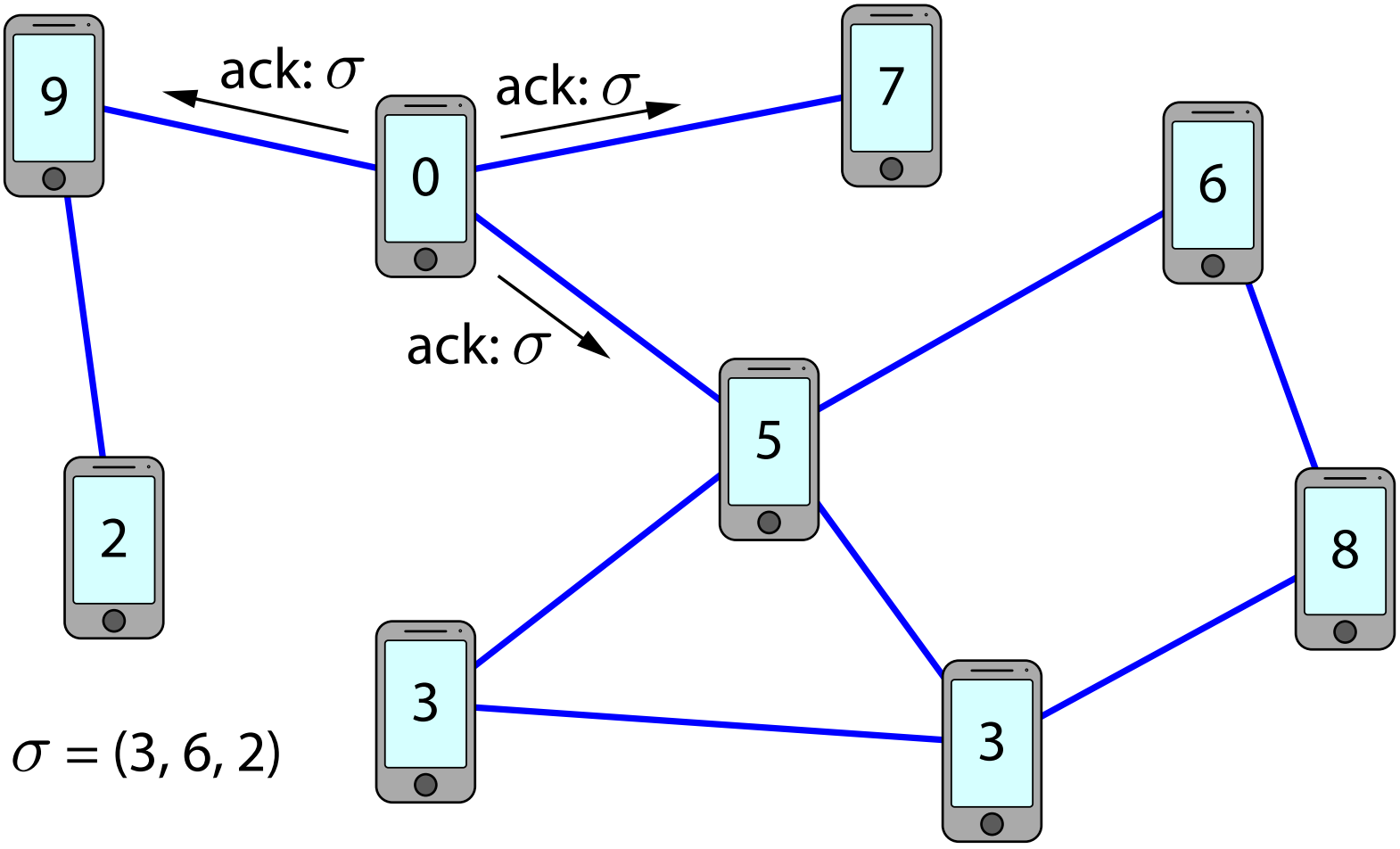


$\sigma = (3, 6, 2)$

Smallest unassigned ID: 10

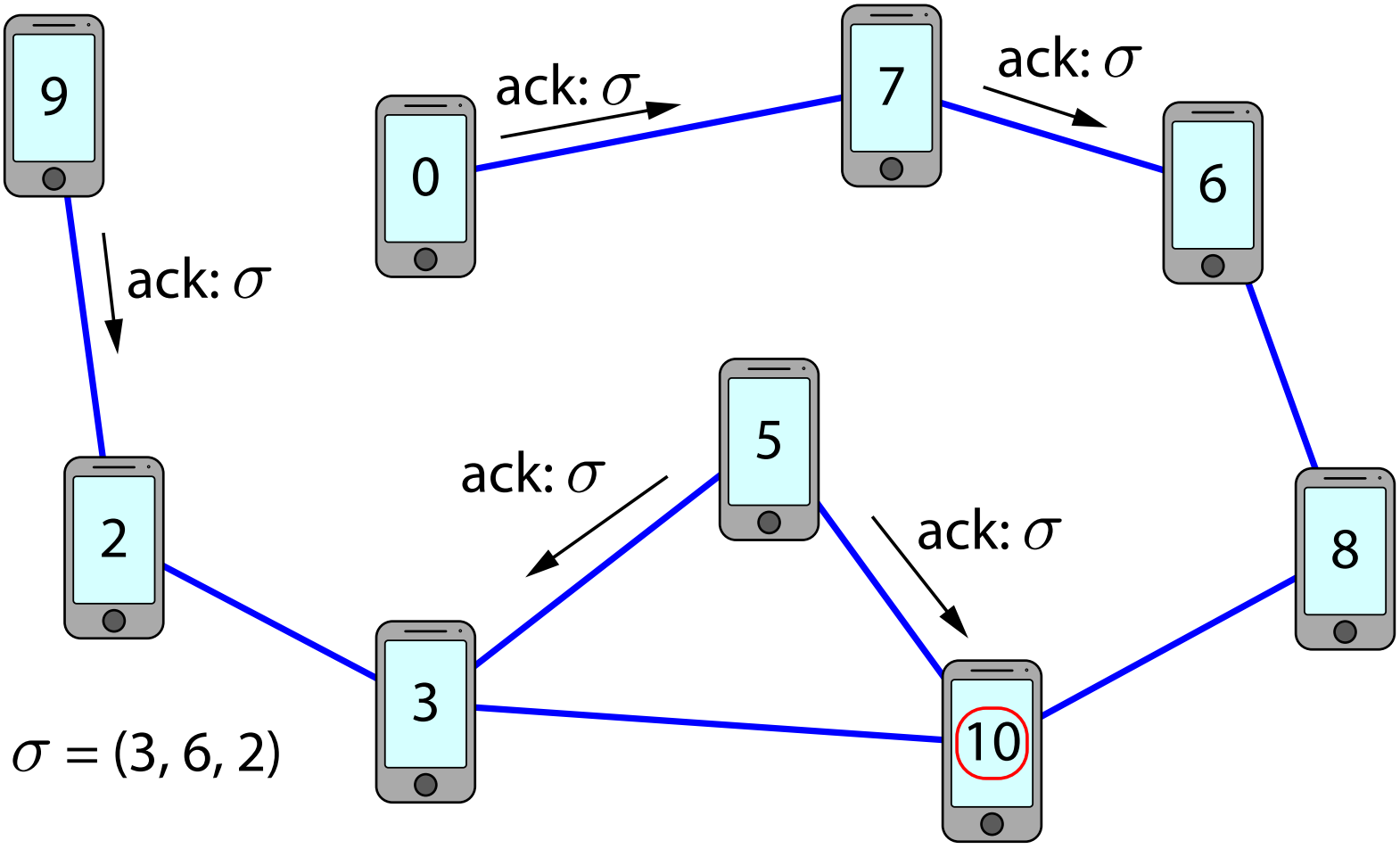
The Leader uses $\sigma = (\mathrm{ID1}, \mathrm{ID2}, m)$ to update the OHT: ID1 gets a child with a fresh name ID1' and the red edge $(\mathrm{ID1'}, \mathrm{ID2}, m)$.

Constructing a level of the OHT



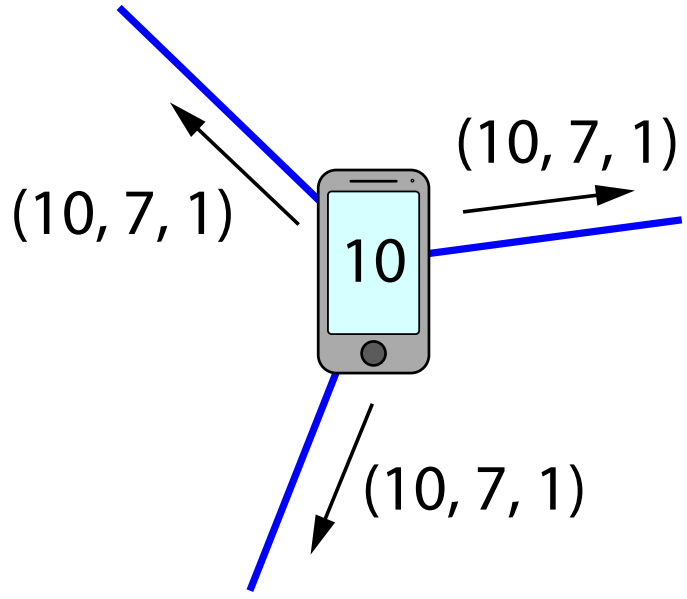
The Leader broadcasts σ as an acknowledgment to all agents, which modify their local OHT in the same way.

Constructing a level of the OHT



Also, each agent named ID1 whose list of triplets contains σ modifies its name in ID1' (all other agents keep their names).

Constructing a level of the OHT



~~(3, 6, 2)~~

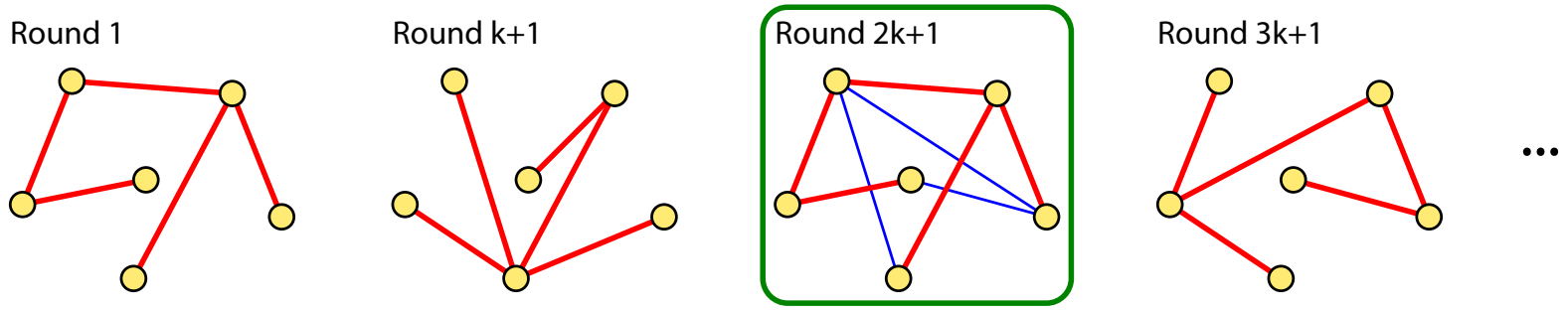
(10, 7, 1)

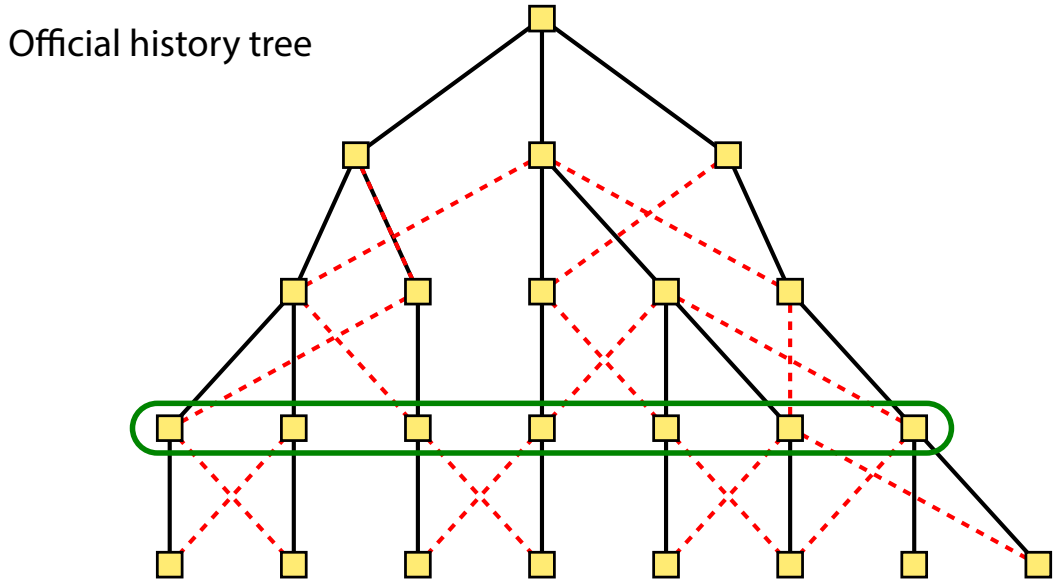
(10, 7, 5)

(10, 8, 2)

Then the next smallest triplet is broadcast, etc. Agents discard all triplets that form cycles with triplets already in the OHT.

Constructing a level of the OHT

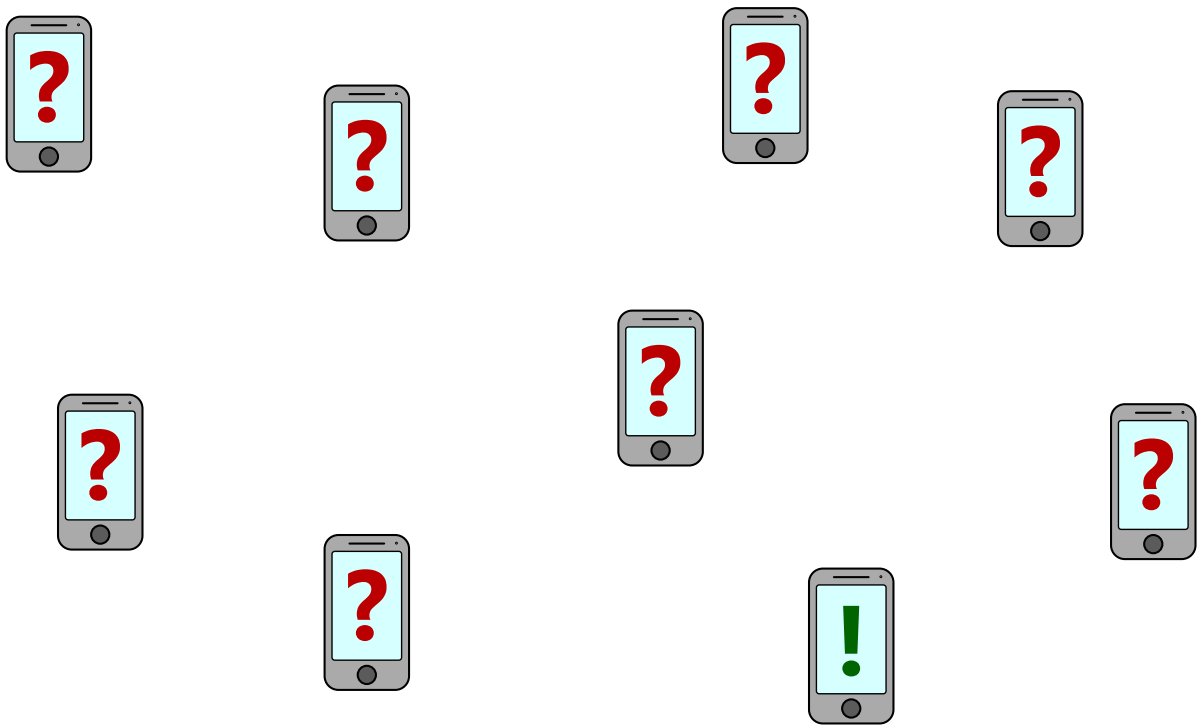




When the current level of the OHT is complete, it represents a *spanning tree* (of size $O(n)$) of the network at the selected round.

Broadcasting information

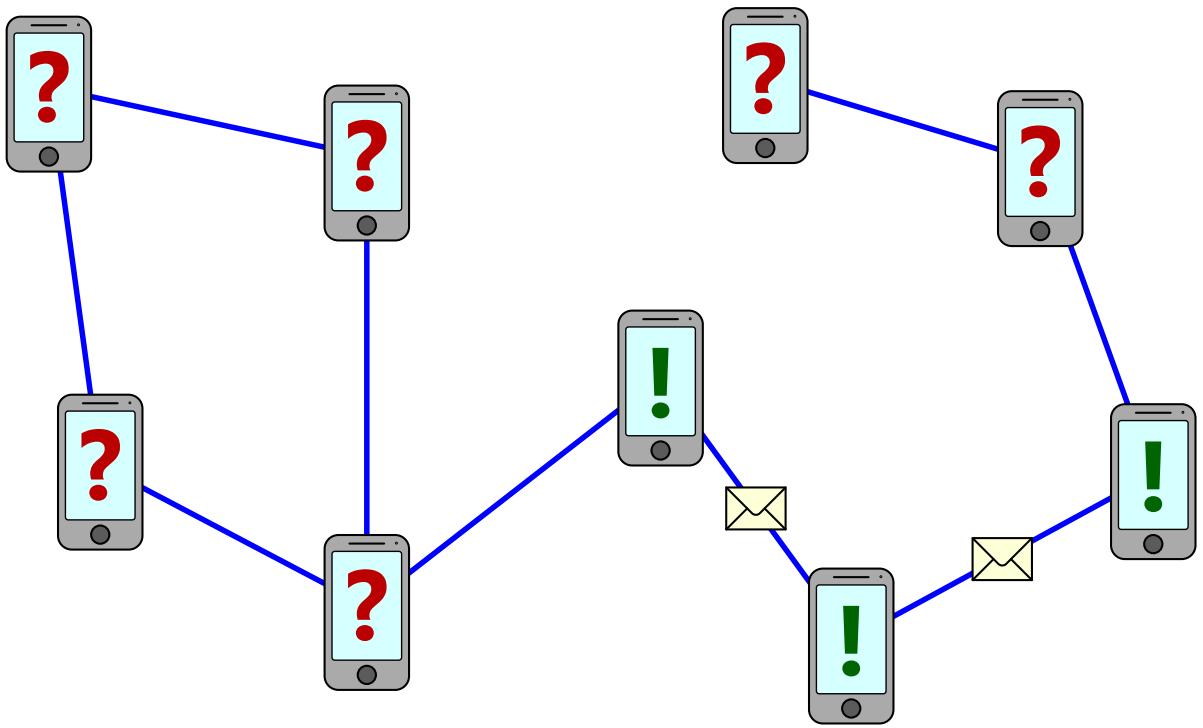
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



Thus, each broadcast takes at most $n - 1$ rounds to complete.

Broadcasting information

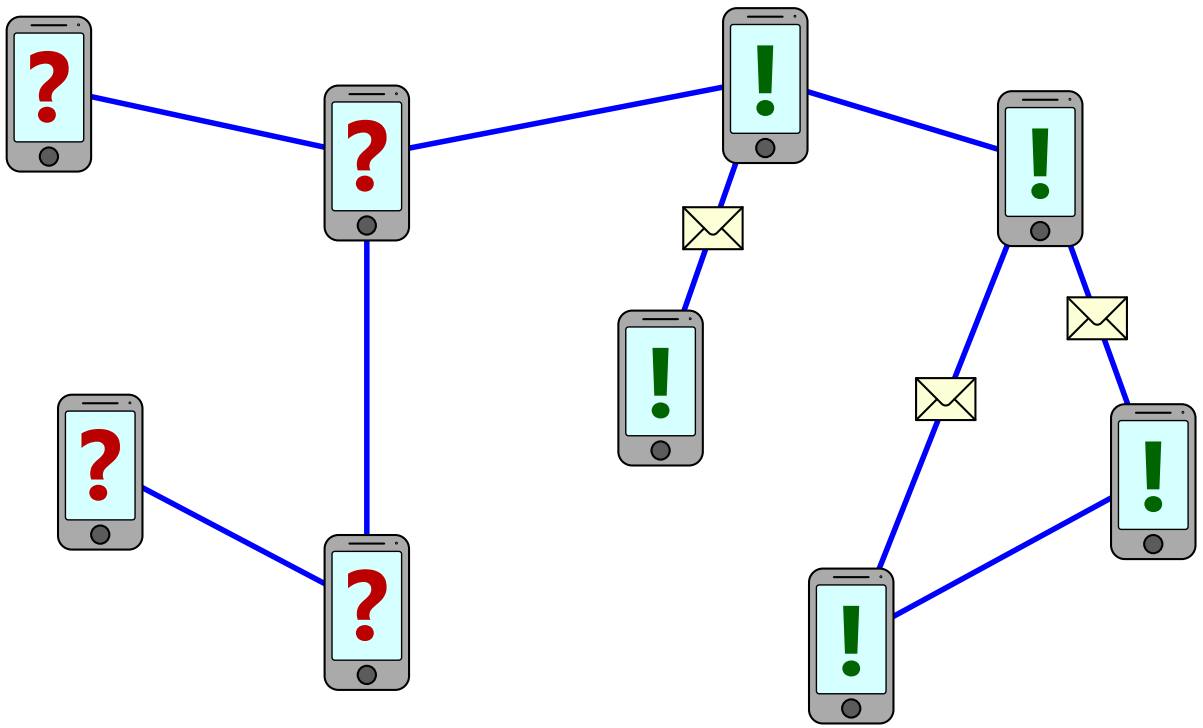
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



Thus, each broadcast takes at most $n - 1$ rounds to complete.

Broadcasting information

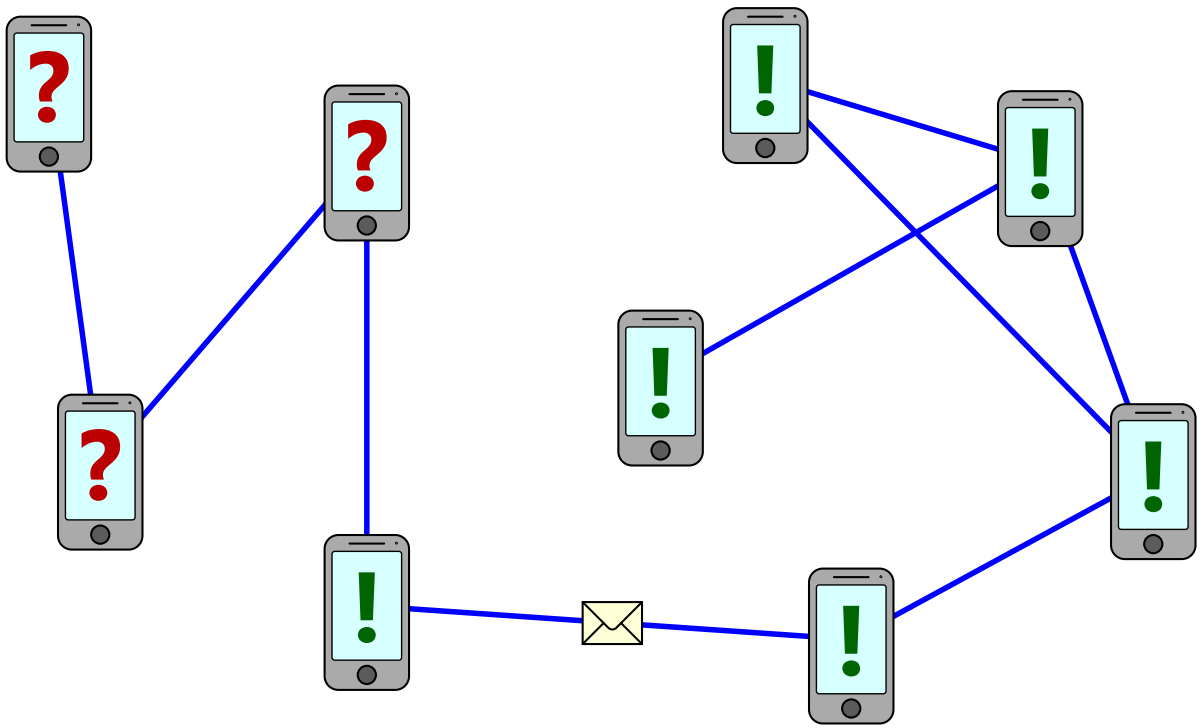
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



Thus, each broadcast takes at most $n - 1$ rounds to complete.

Broadcasting information

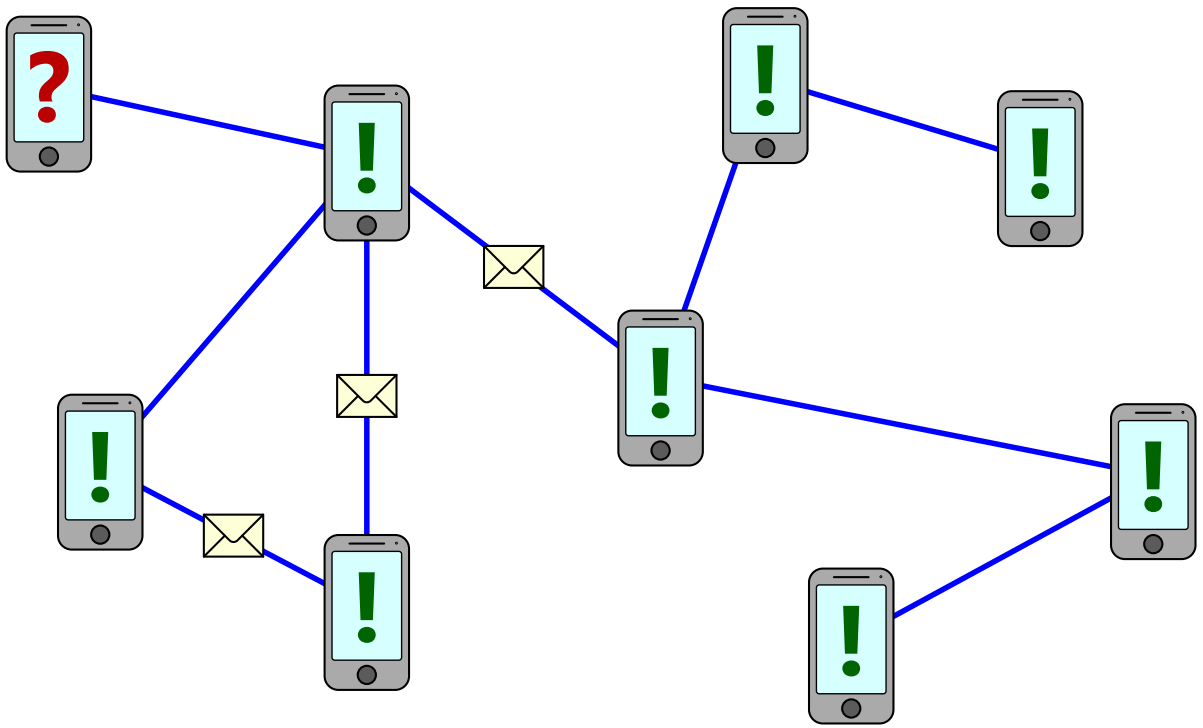
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



Thus, each broadcast takes at most $n - 1$ rounds to complete.

Broadcasting information

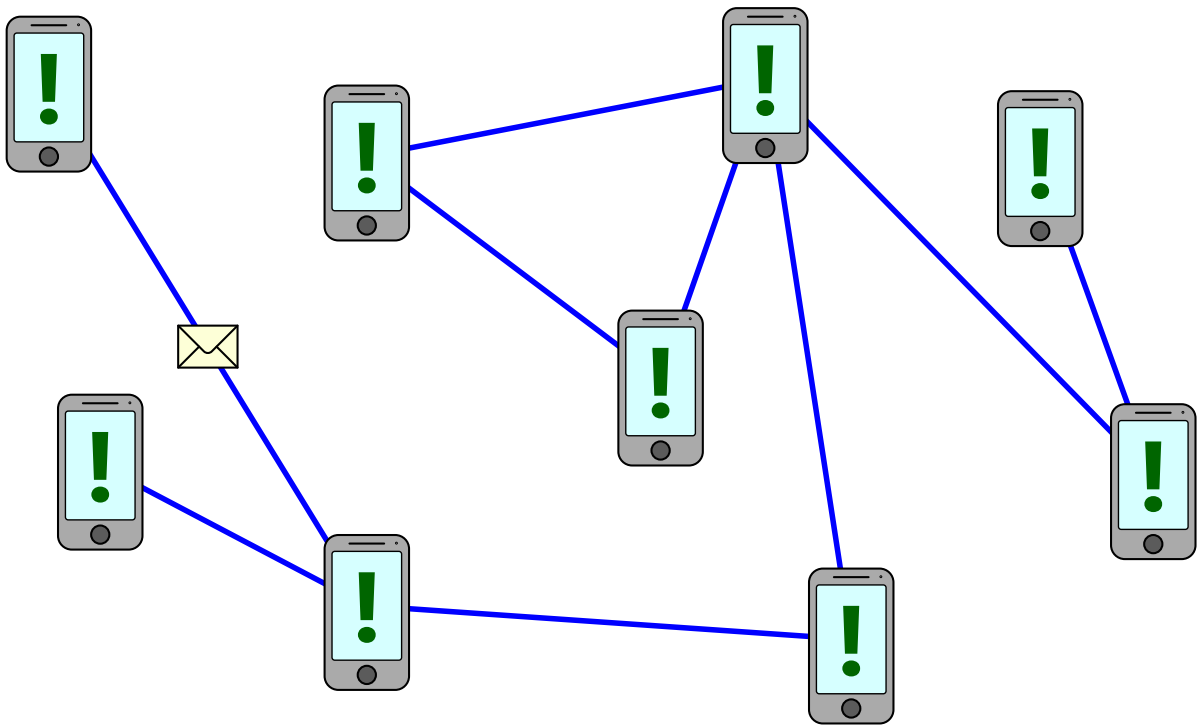
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



Thus, each broadcast takes at most $n - 1$ rounds to complete.

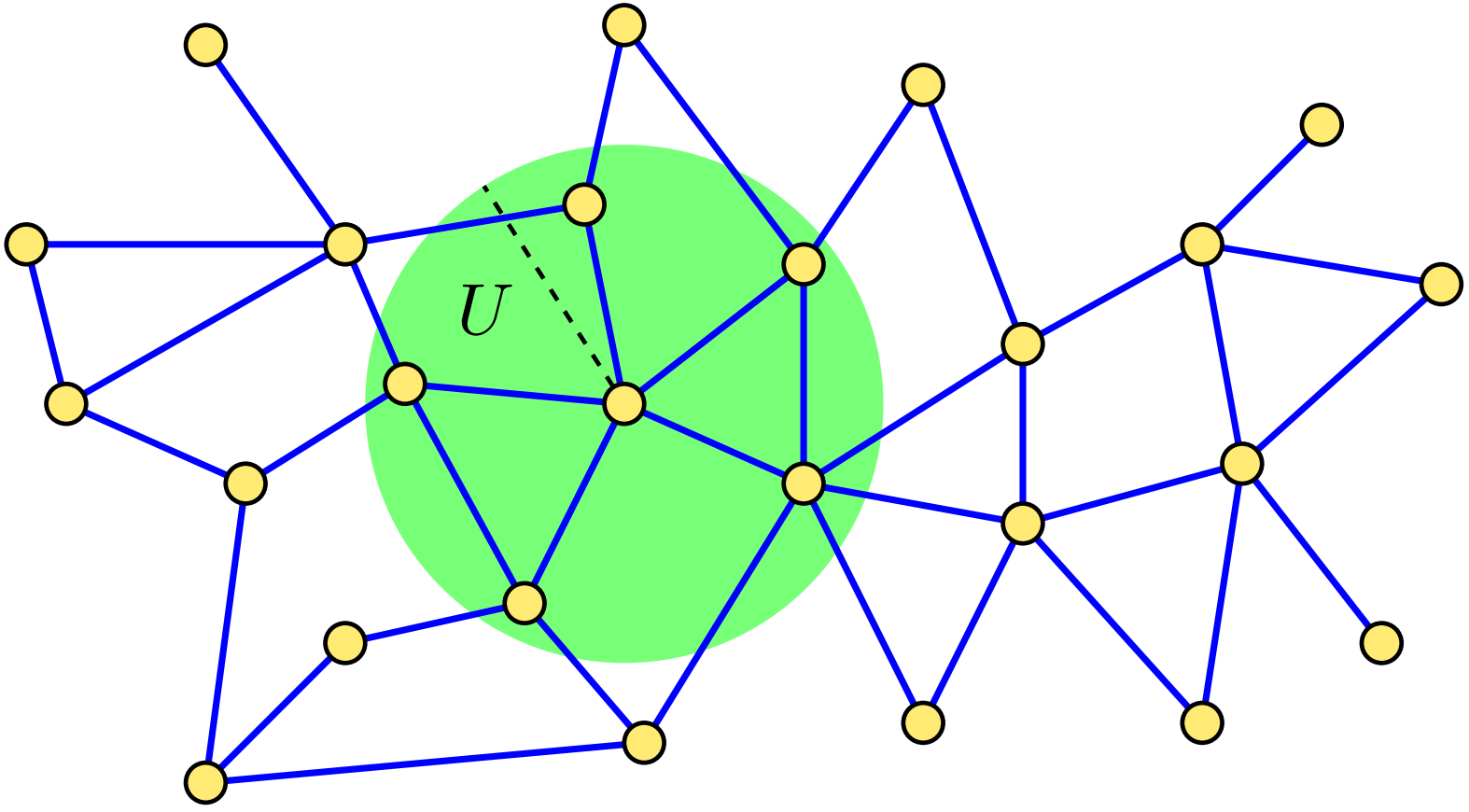
Broadcasting information

Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



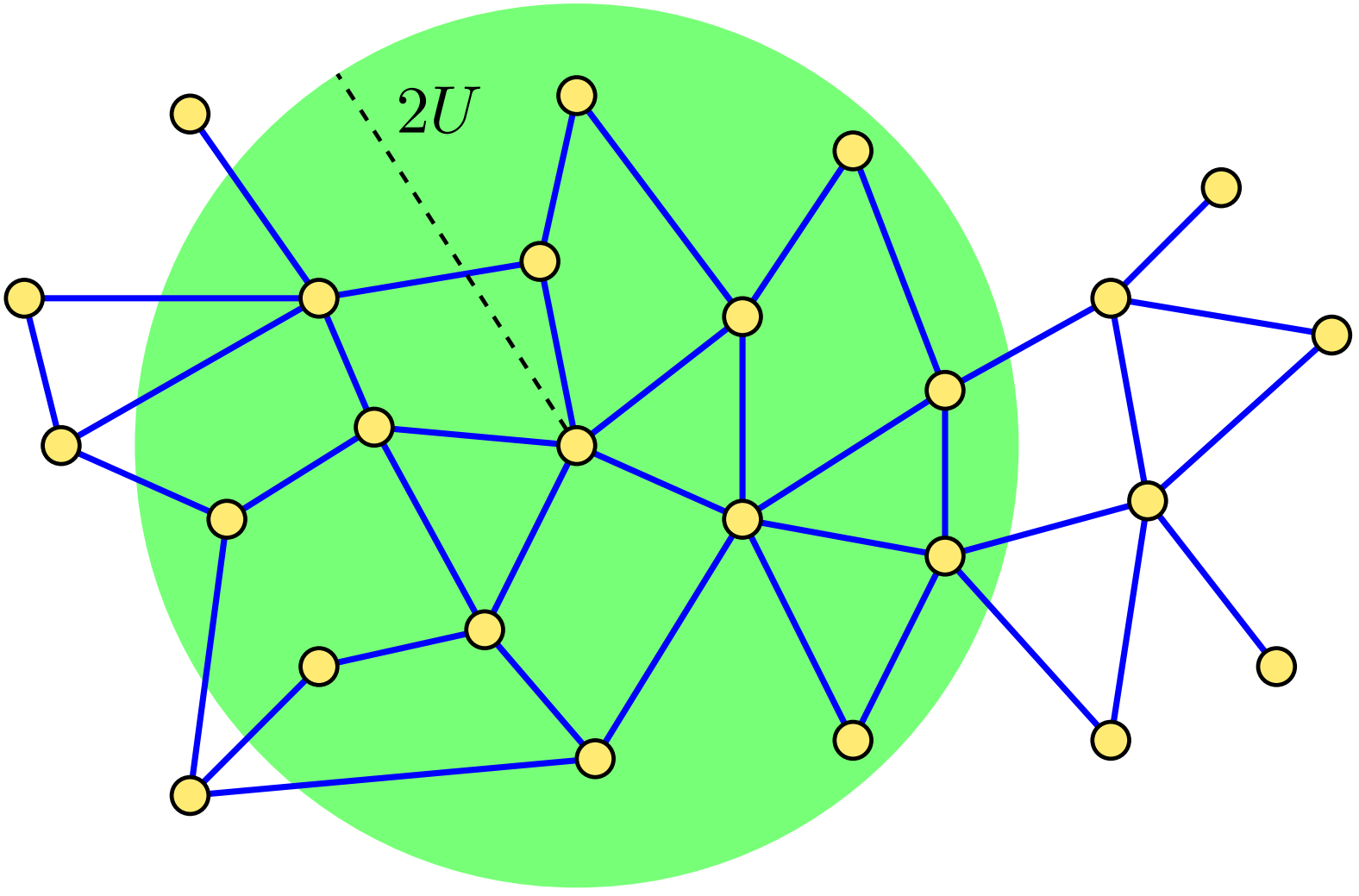
Thus, each broadcast takes at most $n - 1$ rounds to complete.

Reset module



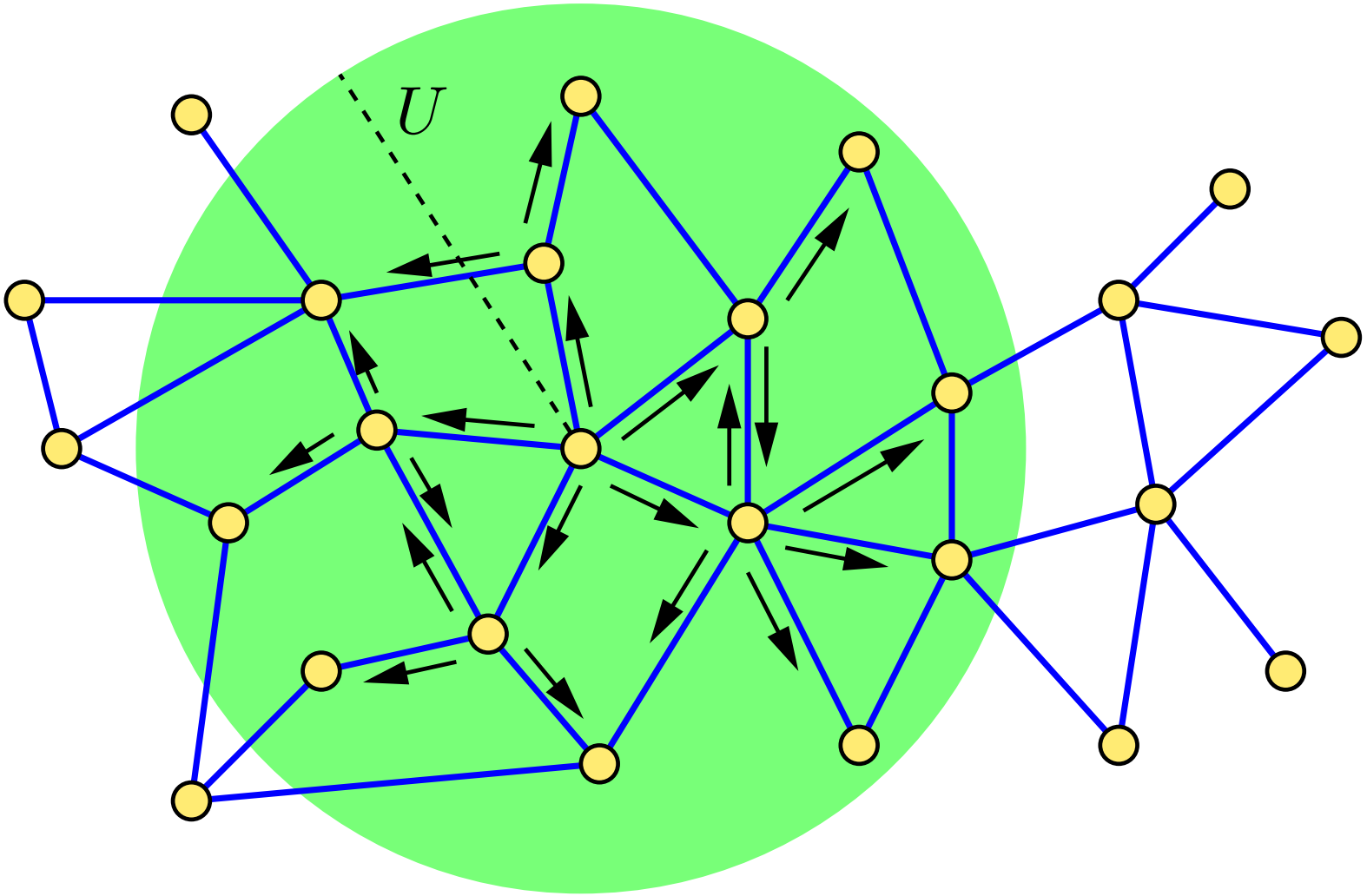
Agents do not know n, so they cannot broadcast information correctly. They only have an *estimate* U on the network size.

Reset module



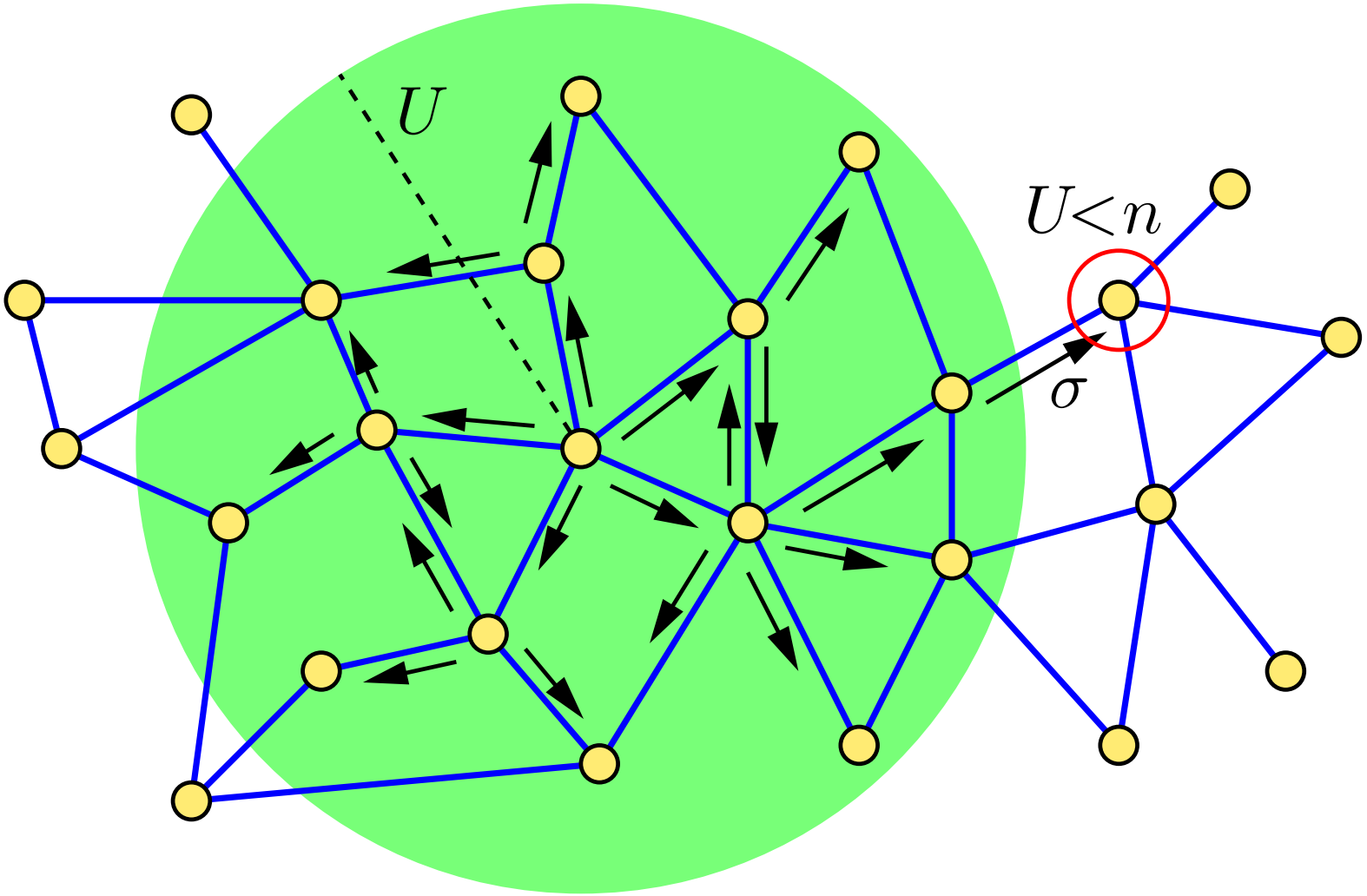
All agents begin with $U = 1$, and then implement a *reset module* that doubles U every time they detect that $U < n$.

Reset module



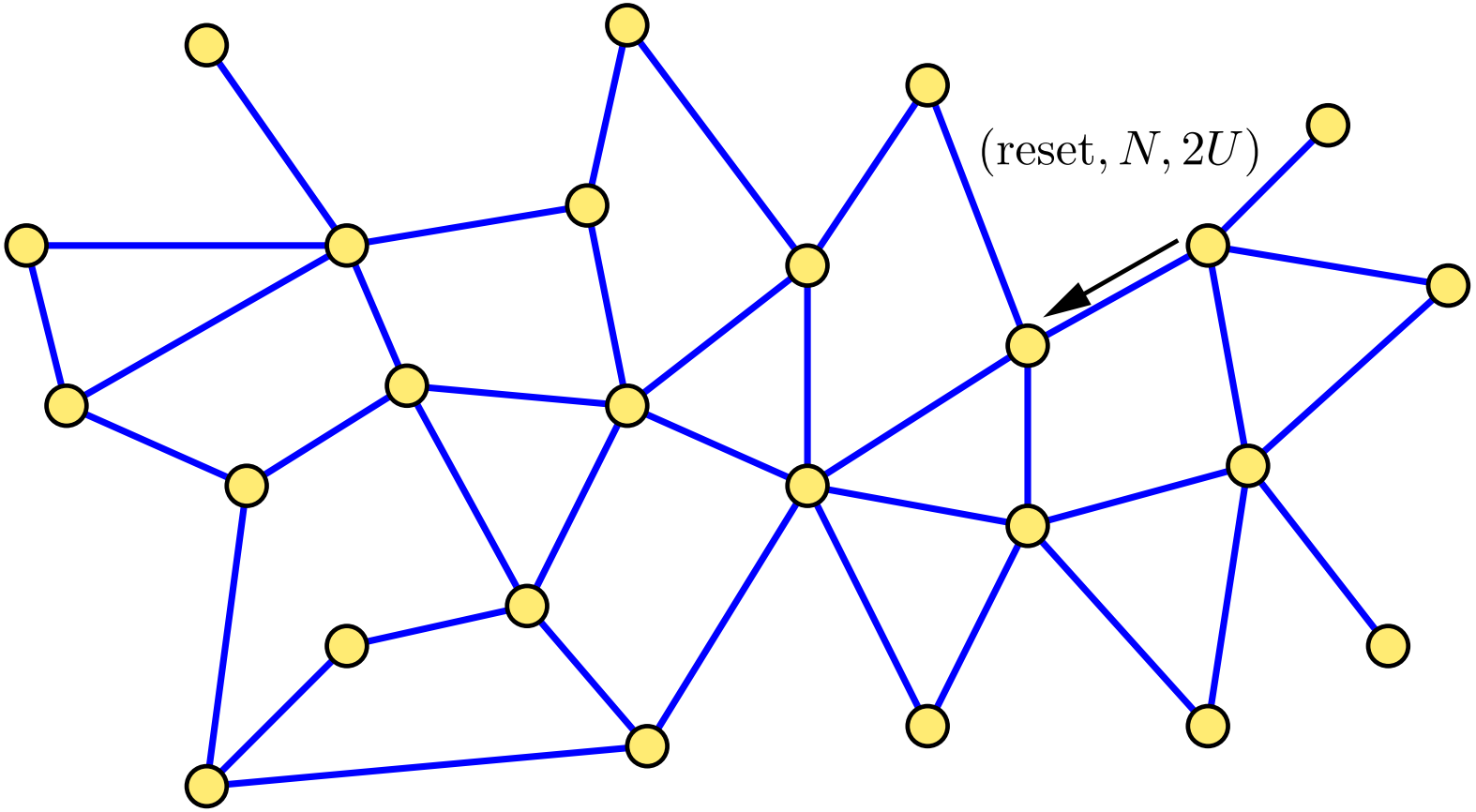
At each round, agents send triplets (N, R, U) (alongside the usual triplets $(\mathsf{ID1}, \mathsf{ID2}, m)$), indicating that this is the Nth broadcast, which started at round R and runs for U rounds.

Reset module



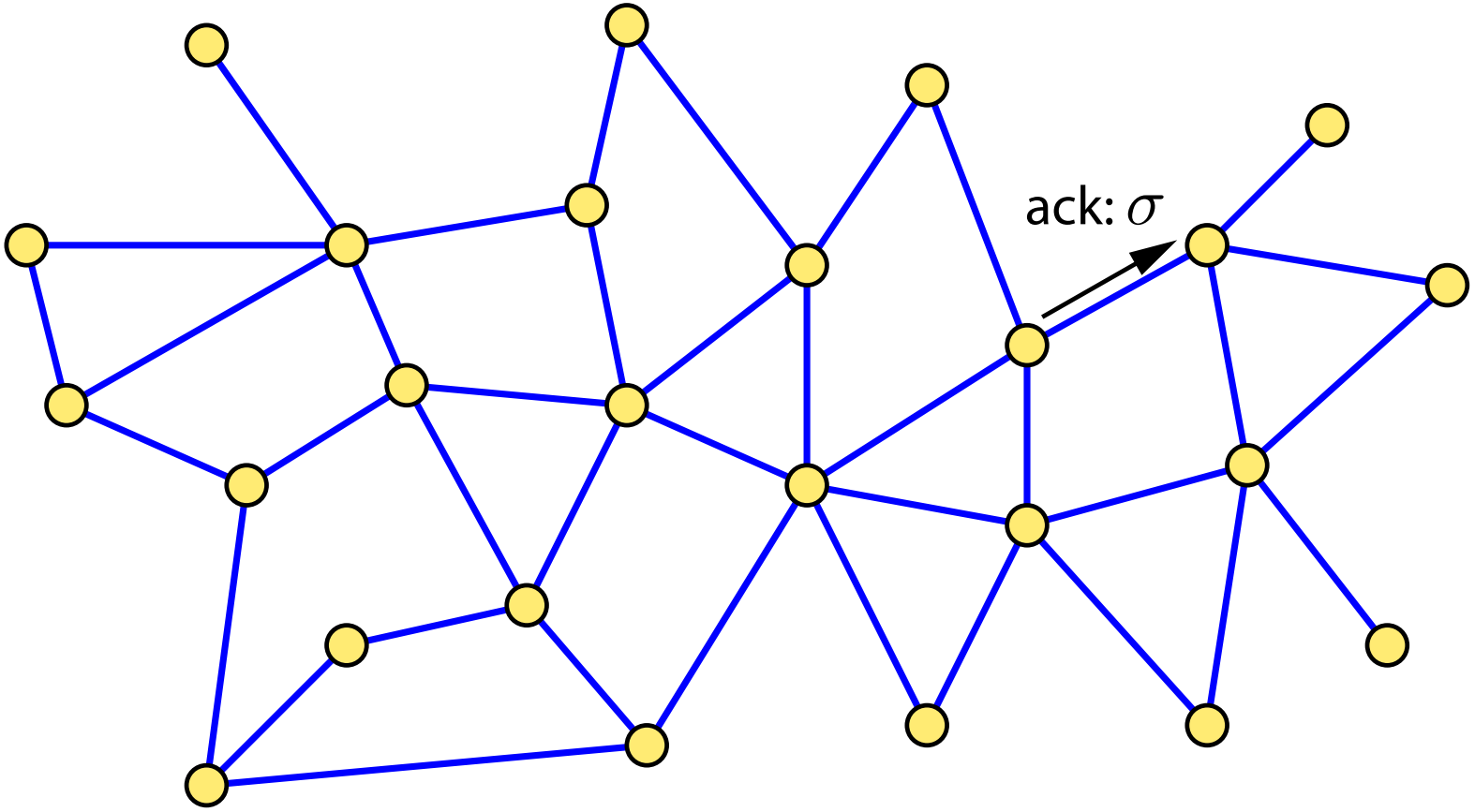
At the Uth round of a broadcast, if an agent receives a new minimum triplet σ, it knows that $U < n$, and starts broadcasting a *reset message* $(\text{reset}, N, 2U)$ that takes priority over anything else.

Reset module



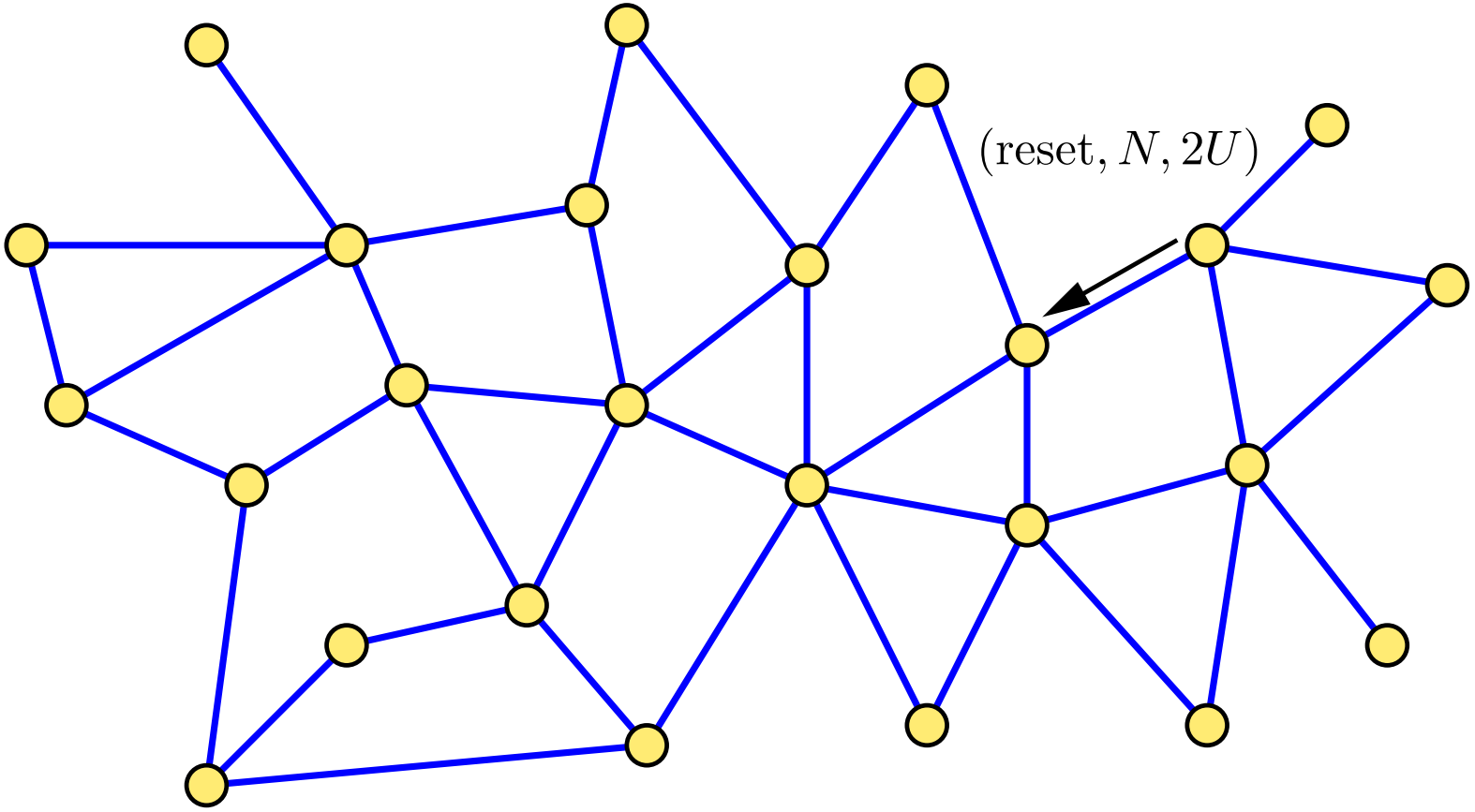
At the Uth round of a broadcast, if an agent receives a new minimum triplet σ, it knows that $U < n$, and starts broadcasting a *reset message* (reset, $N, 2U$) that takes priority over anything else.

Reset module



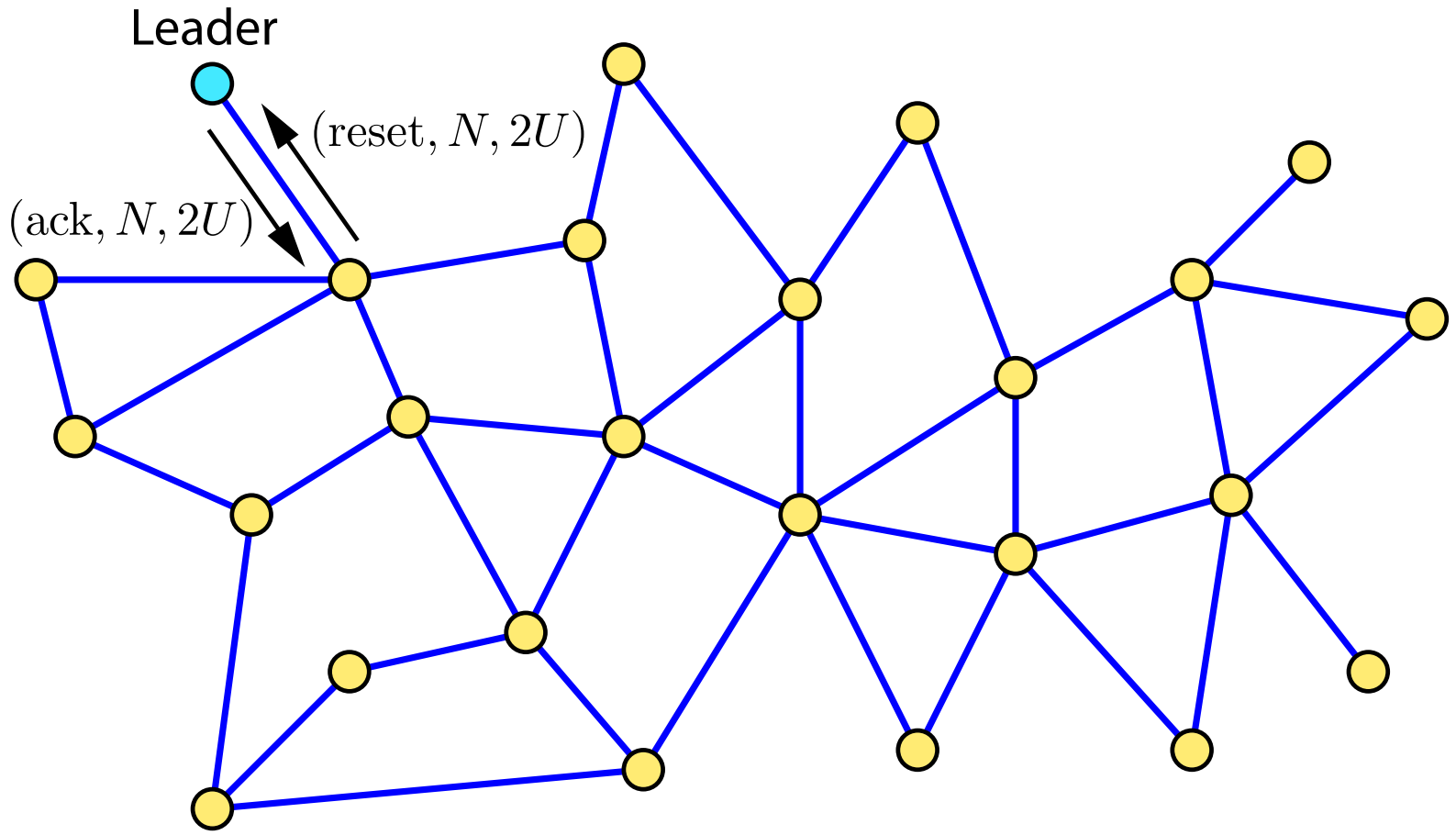
Similarly, if an agent does not receive an acknowledgment from the Leader for broadcast N or does not receive the acknowledgment it expects, it broadcasts a reset message $(\text{reset}, N, 2U)$.

Reset module



Similarly, if an agent does not receive an acknowledgment from the Leader for broadcast N or does not receive the acknowledgment it expects, it broadcasts a reset message (reset, N, $2U$).

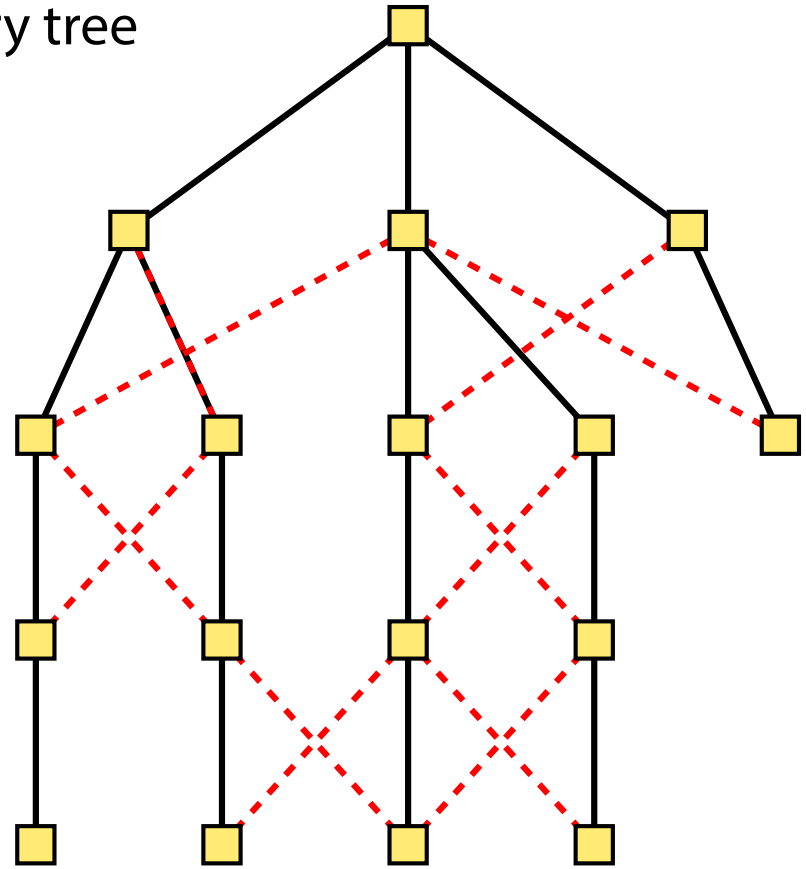
Reset module



When the Leader receives a reset message (reset, N, $2U$), it broadcasts an acknowledgment, ordering all agents to restart from broadcast N with a size estimate of $2U$.

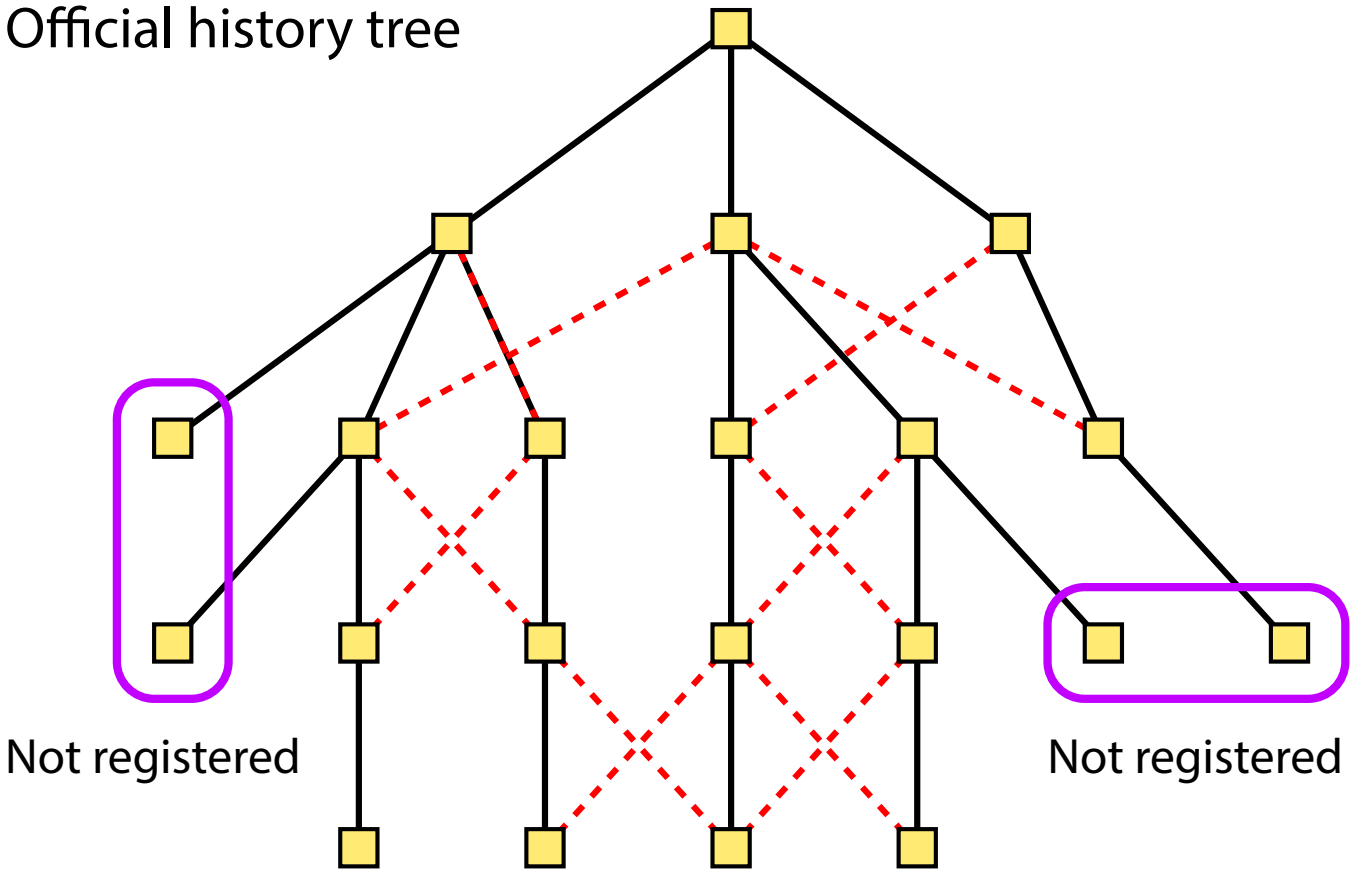
Algorithm correctness

Official history tree



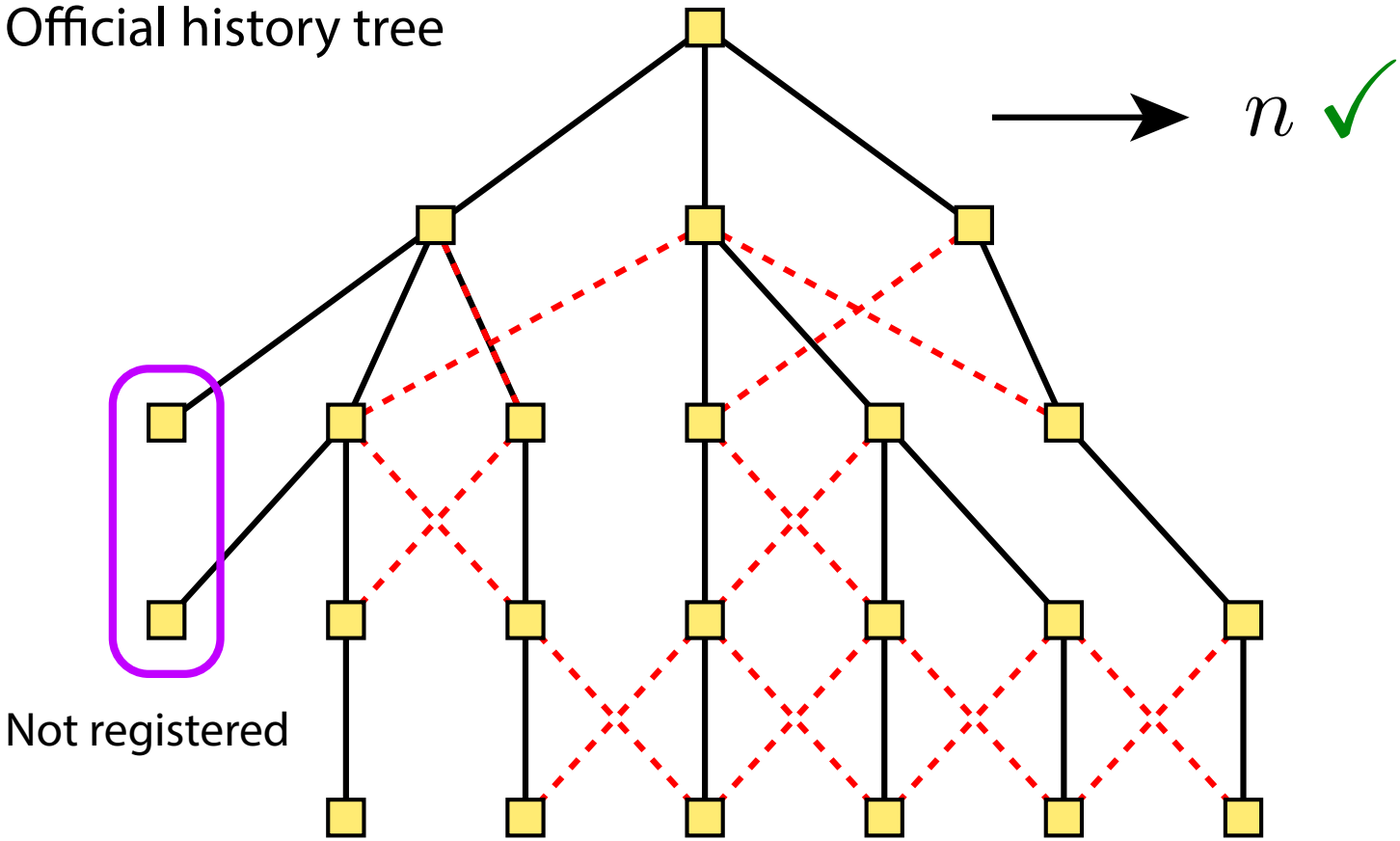
Because any piece of information becomes official only when the Leader sends an acknowledgment about it, at any time there is only *one version* of the OHT circulating in the network.

Algorithm correctness



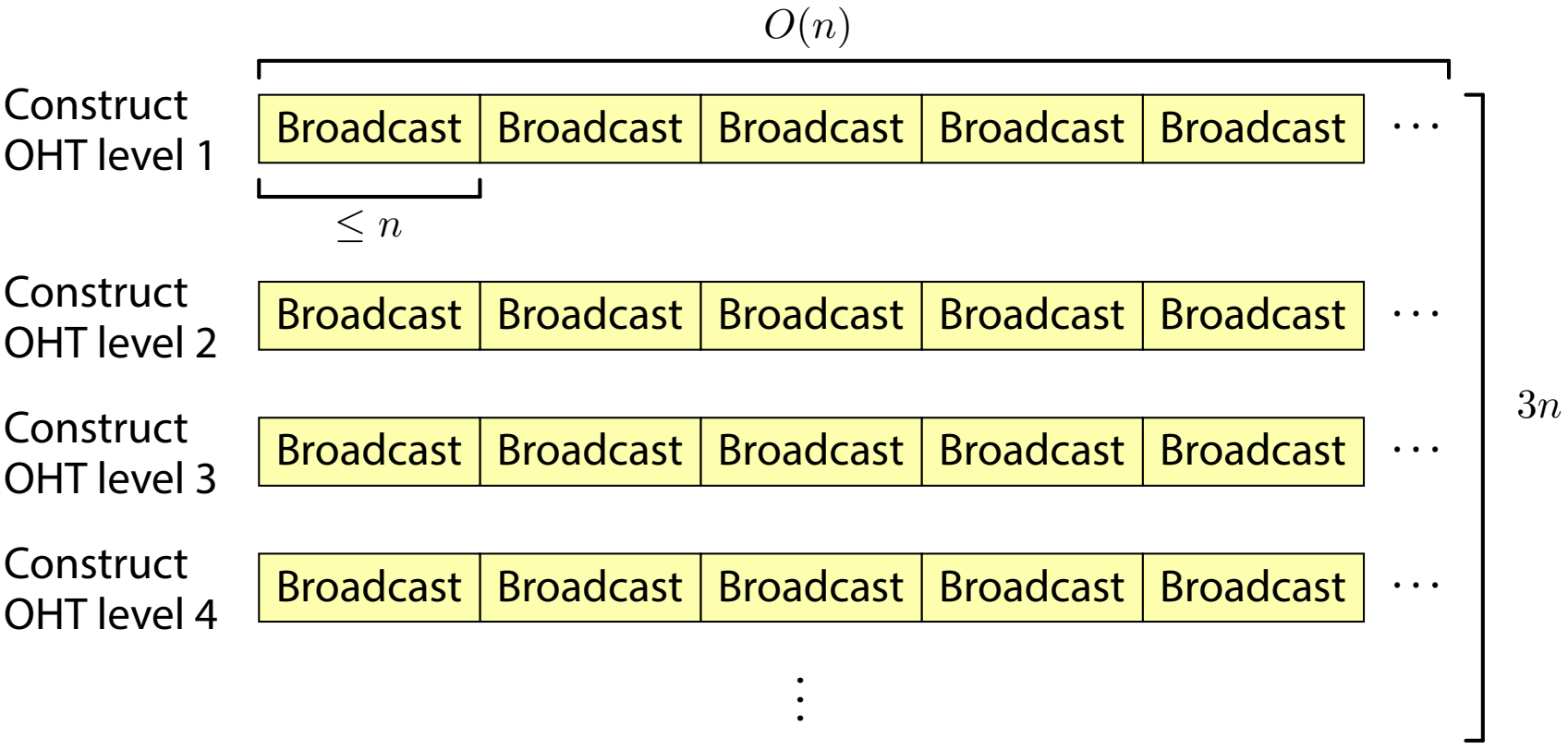
It is possible for the OHT to be missing some red edges, but the LOCAL Counting algorithm can detect when an incomplete history tree does not contain enough information to compute n.

Algorithm correctness



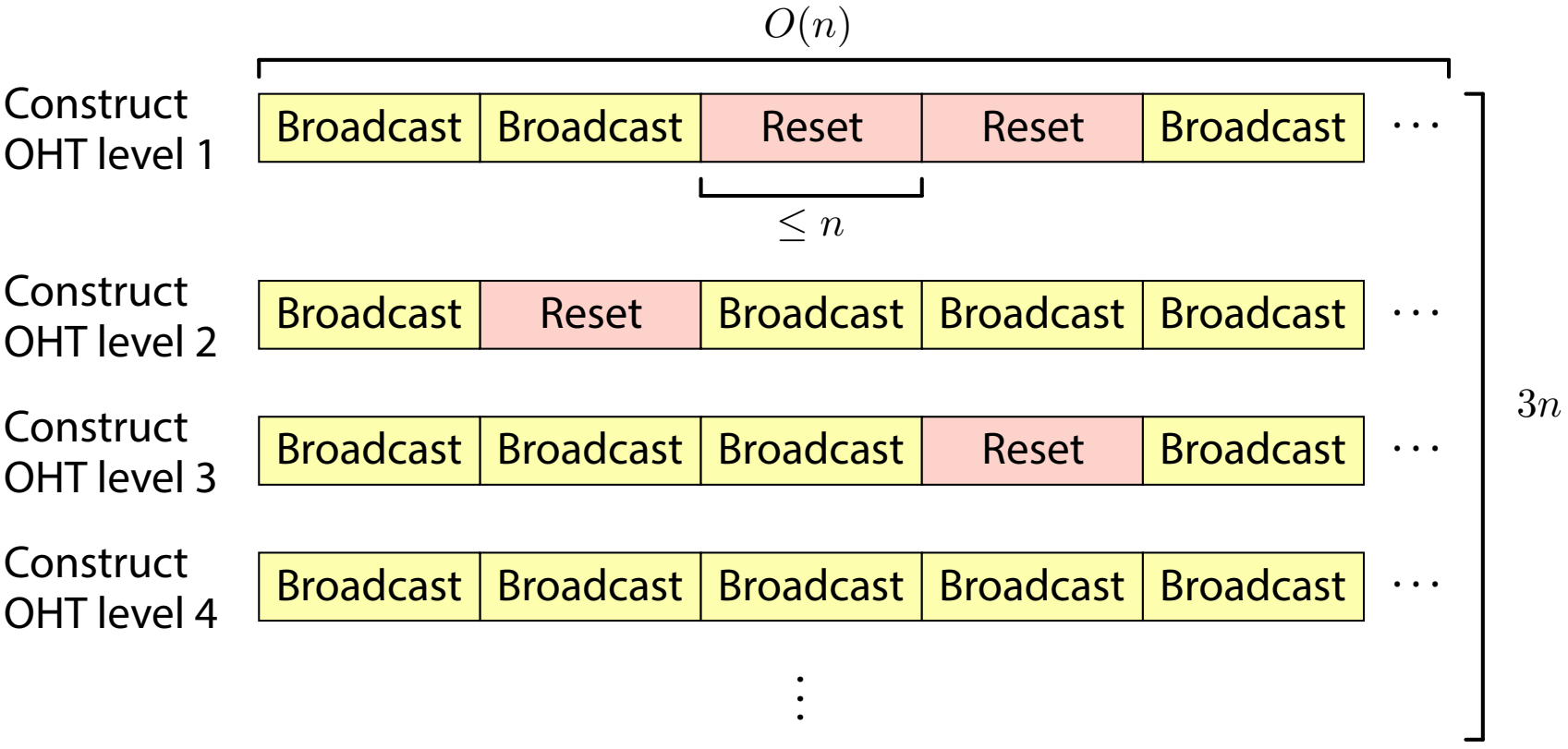
After completing each level of the OHT, we run the LOCAL Counting algorithm on it. As soon as it returns a number n (as opposed to "Unknown"), we can safely output n.

Algorithm running time



We need $3n$ levels of the OHT for the LOCAL algorithm to return n. The information on each level can be gathered in $O(n)$ broadcasts, and each broadcast takes $O(n)$ rounds.

Algorithm running time



If we add the total time lost doing resets, this is just $O(n \log n)$. Indeed, it takes at most $O(\log n)$ resets before $U \geq n$, and broadcasting a reset message cannot take more than n rounds.

In total, the algorithm takes $O(n^3)$ rounds.