Seminar 3 – Anonymous Networks: Optimal Computation with a Leader

Distributed Computing in Anonymous Dynamic Systems

Giovanni Viglietta

Rome – March 8, 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

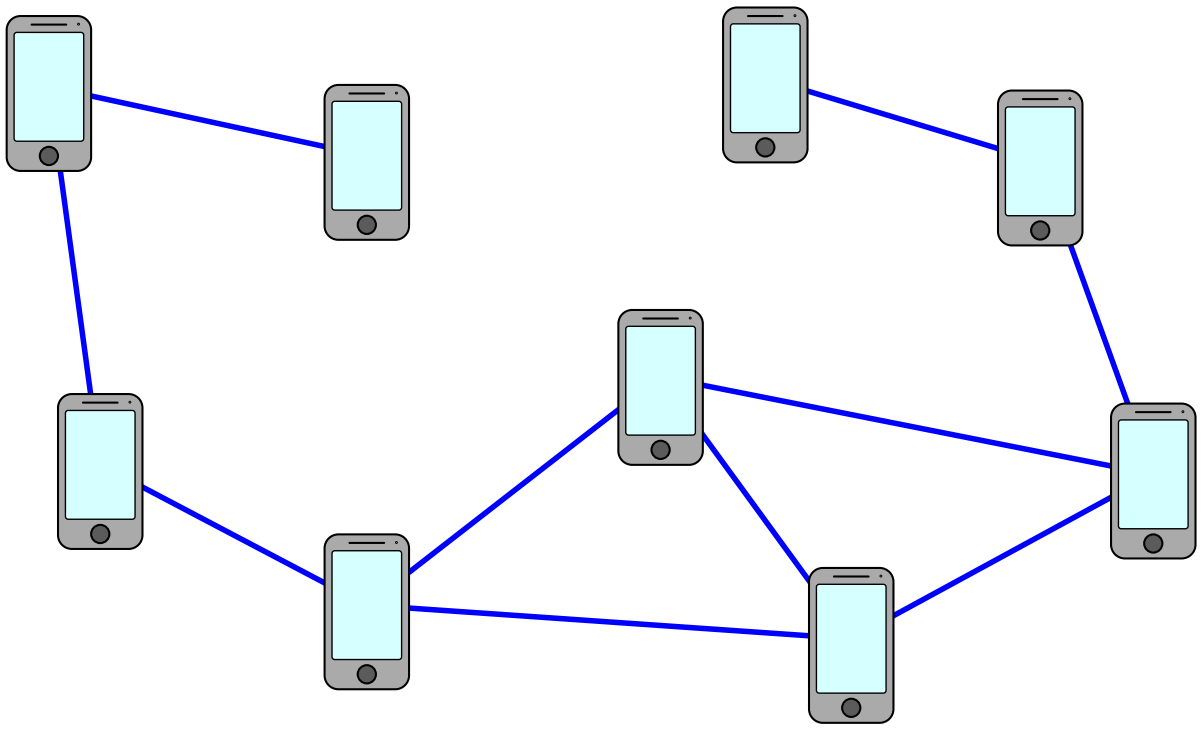
Pre-recorded 10-minute presentation video on one of the papers that will be suggested at the end of the course.

Today's seminar

- Review of history trees
- Counting in dynamic networks
- Terminating algorithm for Counting

Dynamic networks

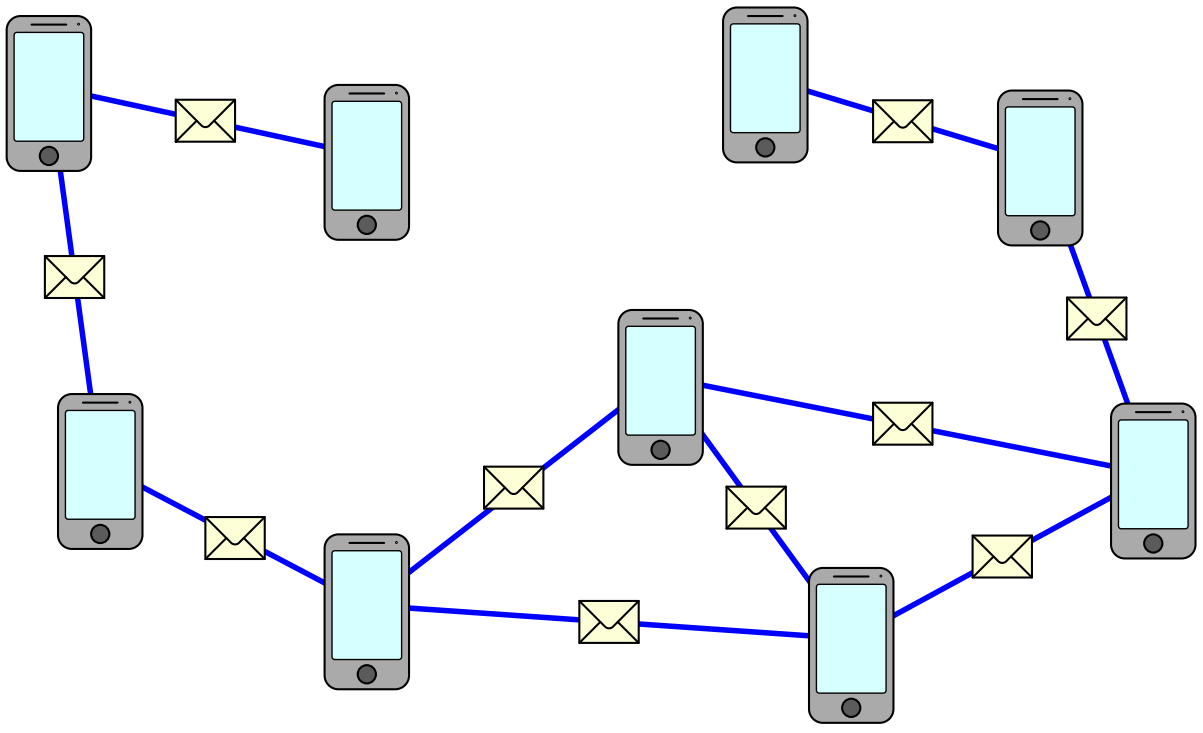
In a *dynamic network*, some machines (or agents) are connected with each other through links that may change over time.



What can be computed by this network, and in how many rounds?

Dynamic networks

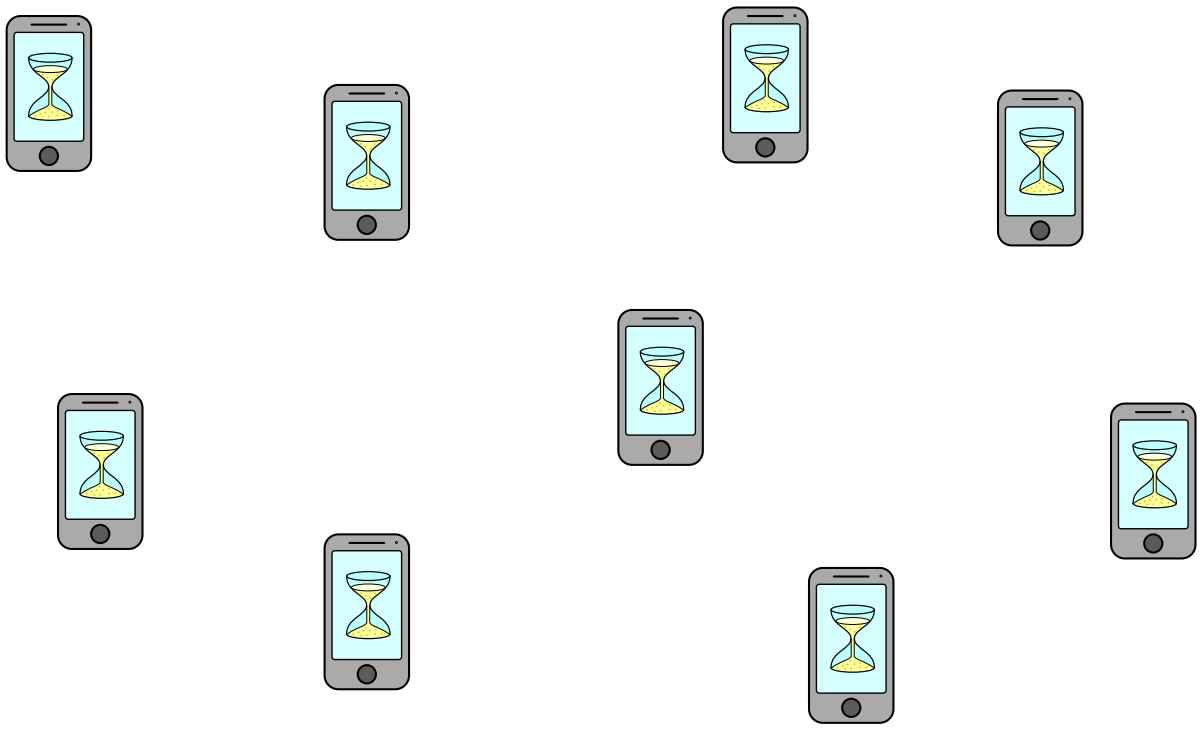
In a *dynamic network*, some machines (or agents) are connected with each other through links that may change over time.



What can be computed by this network, and in how many rounds?

Dynamic networks

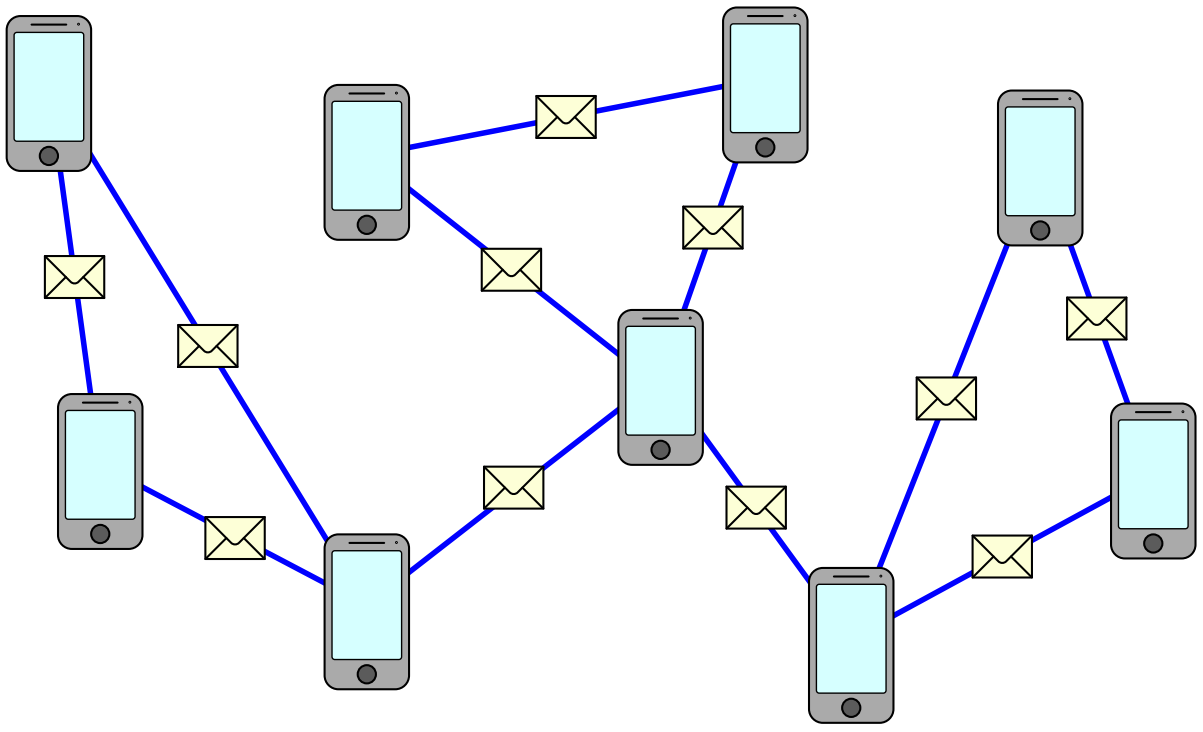
In a *dynamic network*, some machines (or agents) are connected with each other through links that may change over time.



What can be computed by this network, and in how many rounds?

Dynamic networks

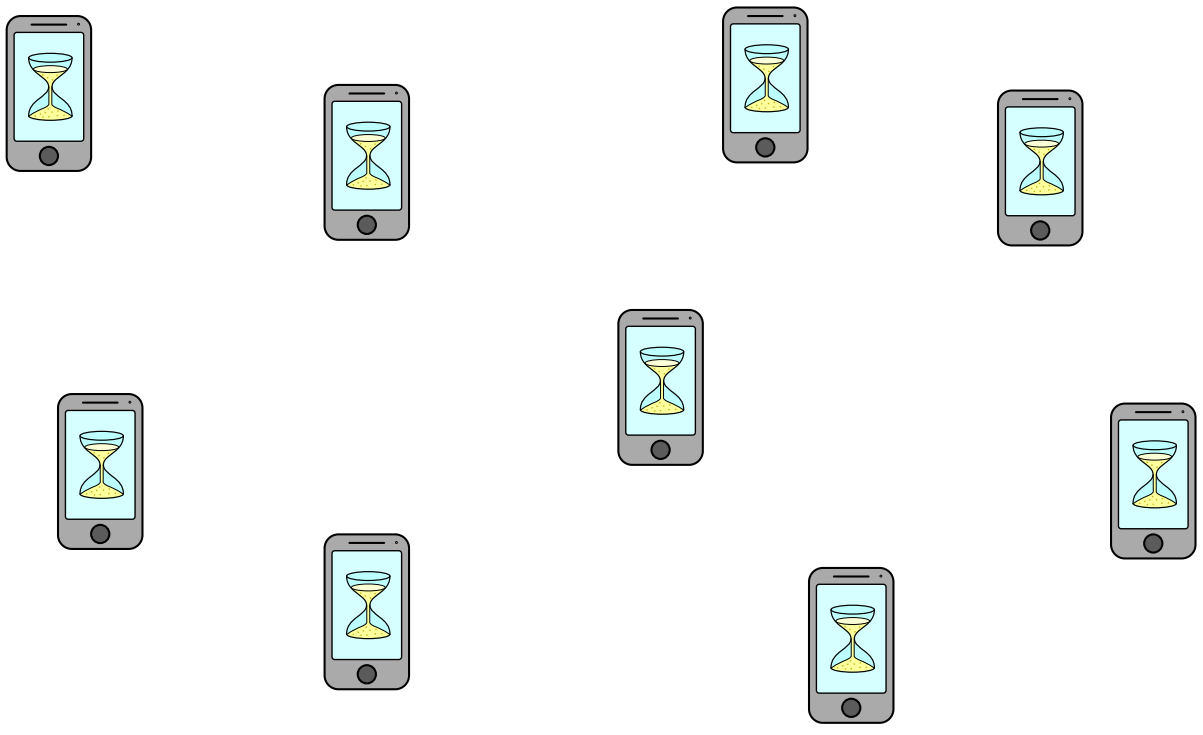
In a *dynamic network*, some machines (or agents) are connected with each other through links that may change over time.



What can be computed by this network, and in how many rounds?

Dynamic networks

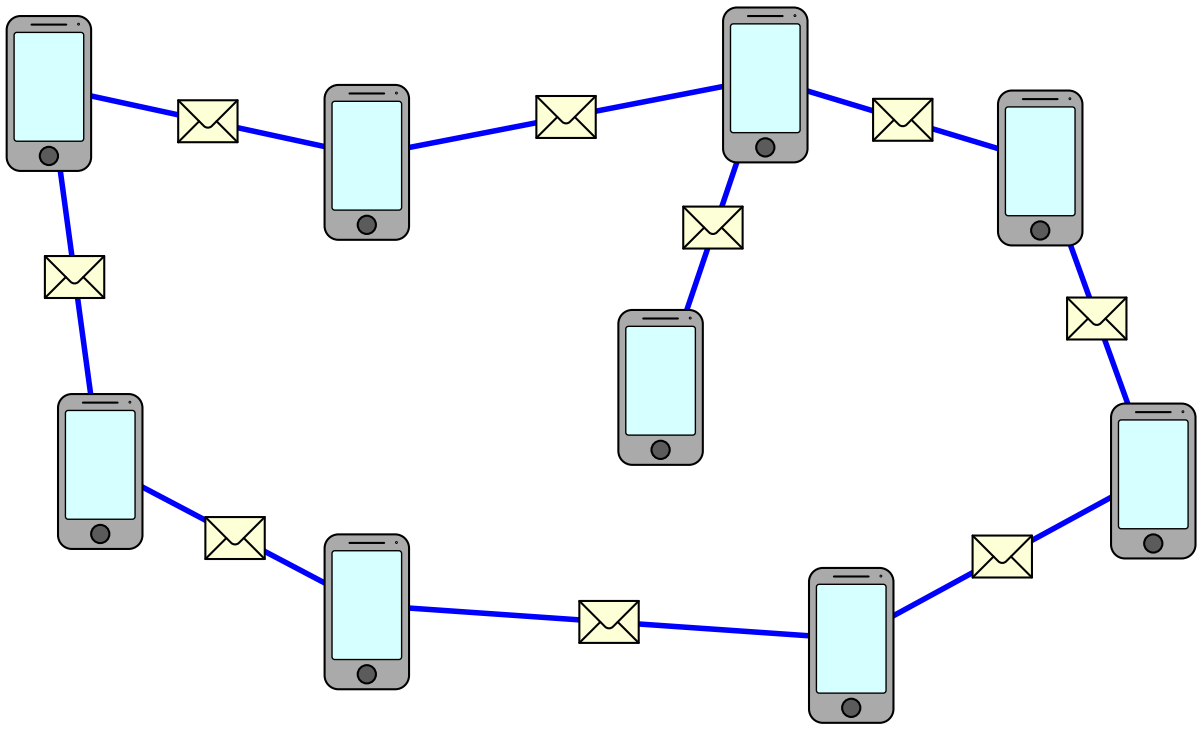
In a *dynamic network*, some machines (or agents) are connected with each other through links that may change over time.



What can be computed by this network, and in how many rounds?

Dynamic networks

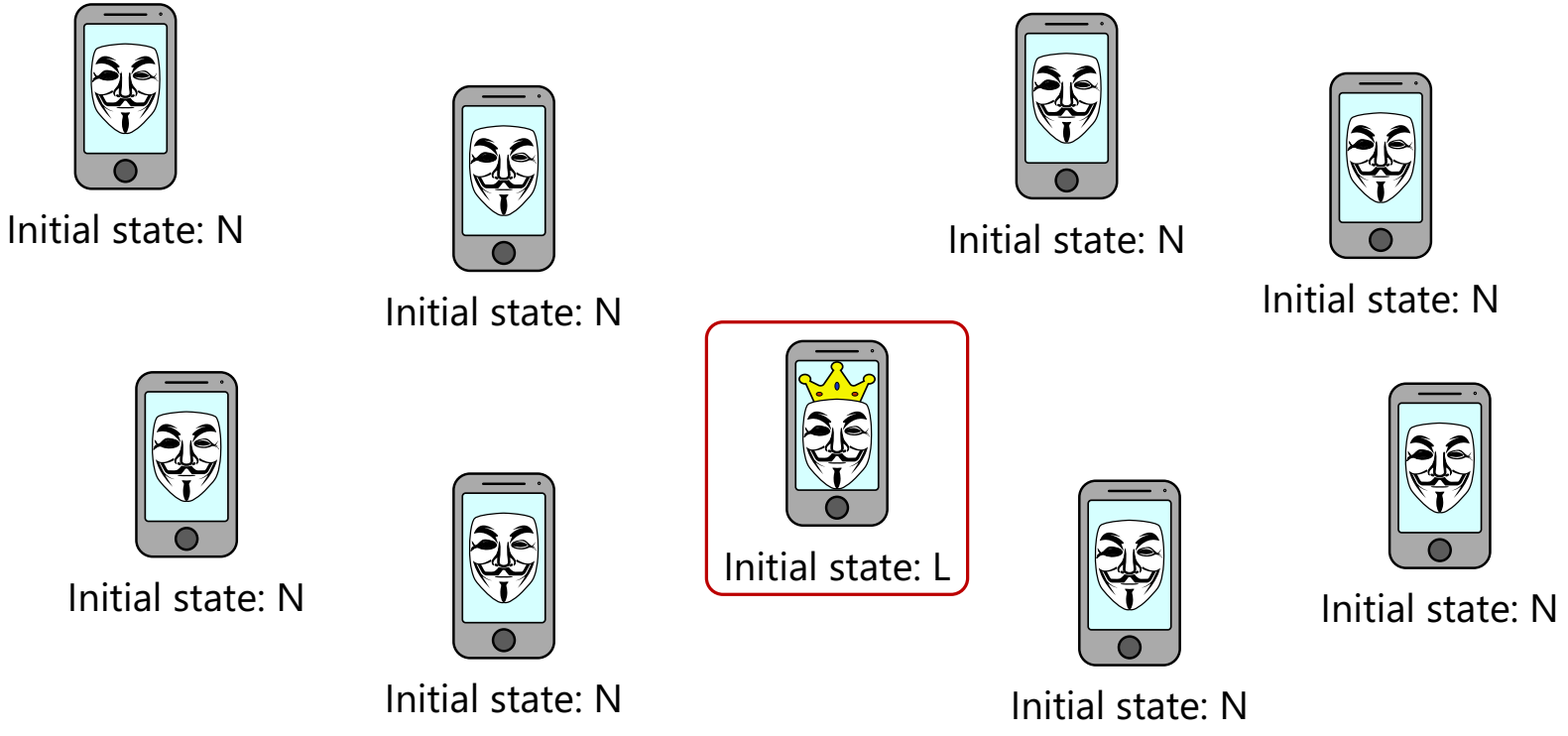
In a *dynamic network*, some machines (or agents) are connected with each other through links that may change over time.



What can be computed by this network, and in how many rounds?

Counting anonymous agents with a Leader

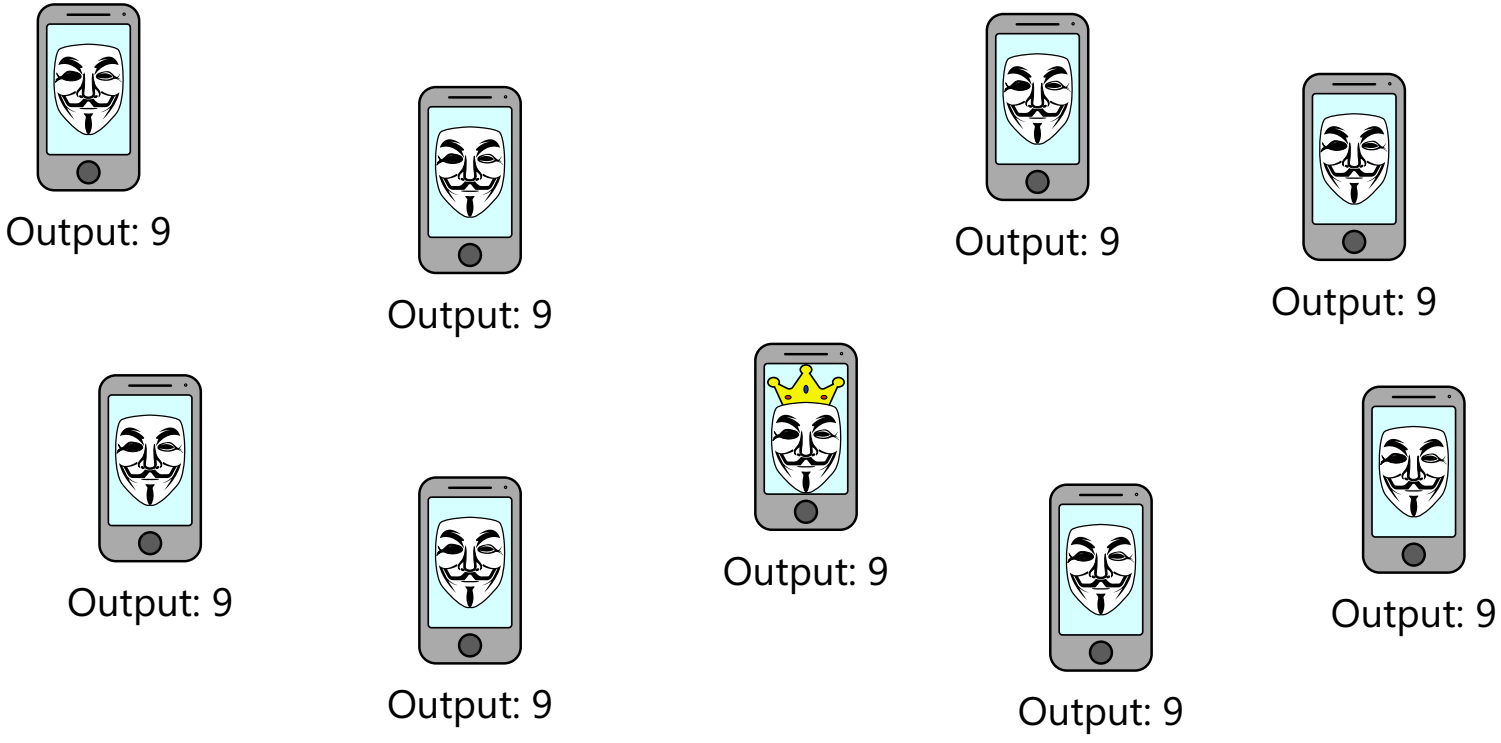
Our assumption is that the dynamic network is *anonymous*, i.e., all agents start in the same state, except one: the *Leader*.



The *complete problem* in this model is the **Counting Problem:** Eventually, all agents must know the total number of agents, n. (If agents have inputs, also compute how many agents have each input.)

Counting anonymous agents with a Leader

Our assumption is that the dynamic network is *anonymous*,
i.e., all agents start in the same state, except one: the *Leader*.



The *complete problem* in this model is the **Counting Problem:**
Eventually, all agents must know the total number of agents, n.
(If agents have inputs, also compute how many agents have each input.)

Literature on the Counting Problem

- **Michail et al.:** Looks impossible! (SSS 2013)
- **Di Luna et al.:** Solvable in $O(e^{N^2} N^3)$ rounds (ICDCN 2014)
- **Di Luna–Baldoni:** $O(n^{n+4})$ rounds (OPODIS 2015)
- **Kowalski–Mosteiro:** $O(n^5 \log^2 n)$ rounds (ICALP 2018 Best Paper)
- **Kowalski–Mosteiro:** $O(n^{4+\epsilon} \log^3 n)$ rounds (ICALP 2019)
- **Di Luna–V.:** $3n$ rounds (FOCS 2022) **(Today's seminar)**

Symbols:

- n: number of agents in the network (unknown)
- N: upper bound on n (unknown, except in ICDCN 2014)

Counting in dynamic networks with a unique leader

Theorem (Di Luna–V., FOCS 2022)

In connected anonymous dynamic networks with a unique Leader, the Counting Problem can be solved:

- *in $2n$ rounds without explicit termination,* **(Yesterday)**
- *in $3n$ rounds with termination.* **(Today)**

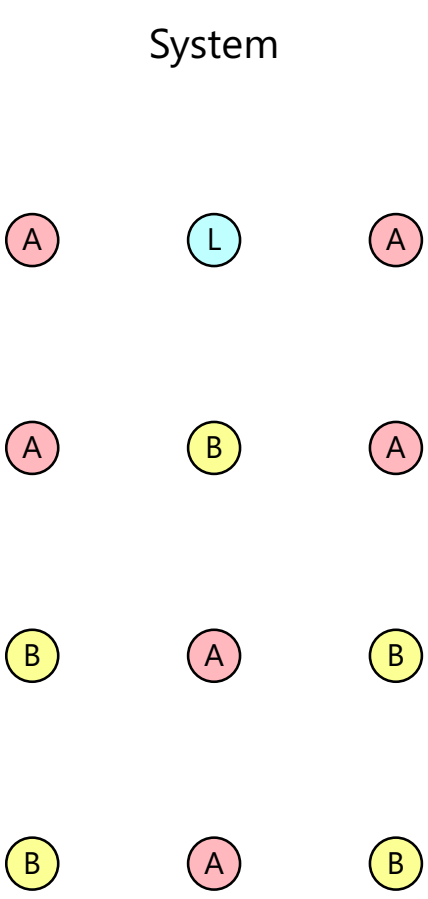
Furthermore, no (stabilizing or terminating) algorithm can solve the Counting problem in less than $2n$ rounds.

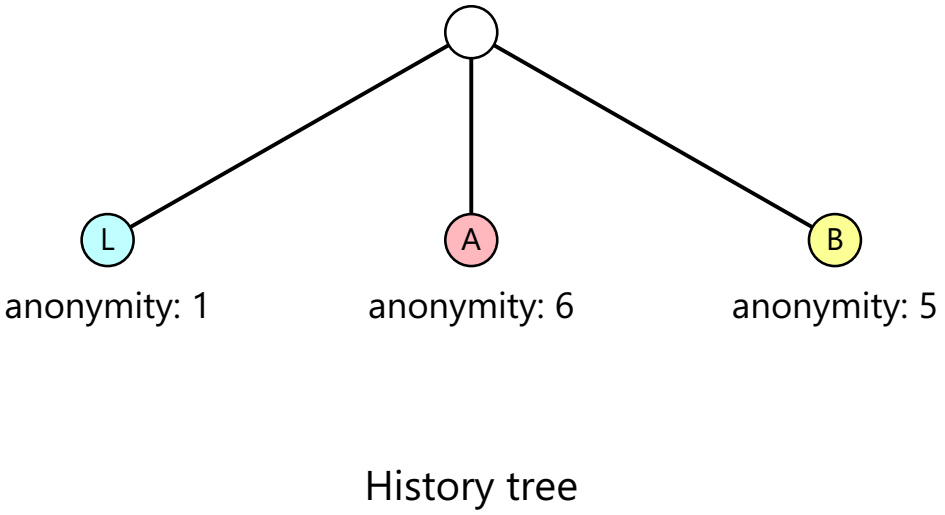
The theorem applies not only to the Counting Problem, but to *all* functions computable in anonymous (dynamic) networks.

These are precisely the *multi-aggregate* functions f:

- Agent p outputs $f(x_p, \mu)$,
- where x_p is the input of agent p,
- and μ is the multi-set of all inputs.

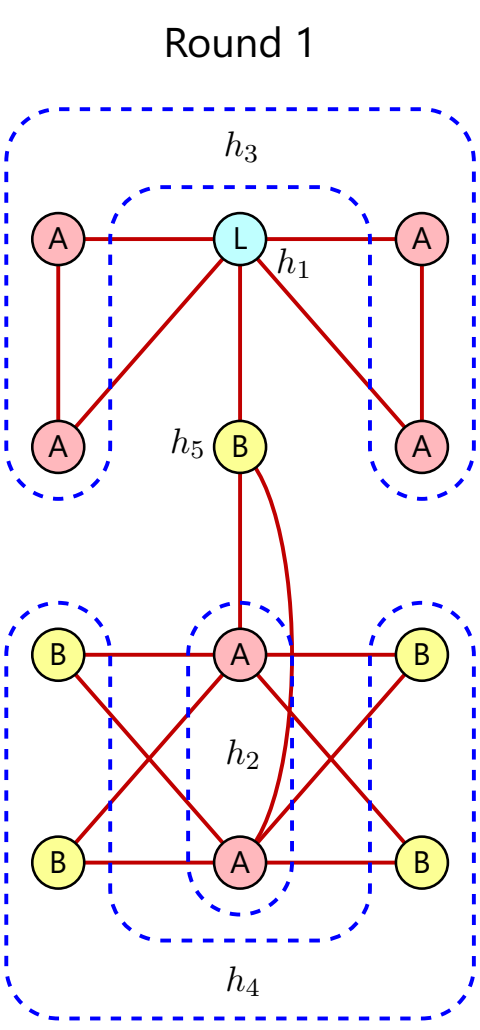
Constructing a history tree

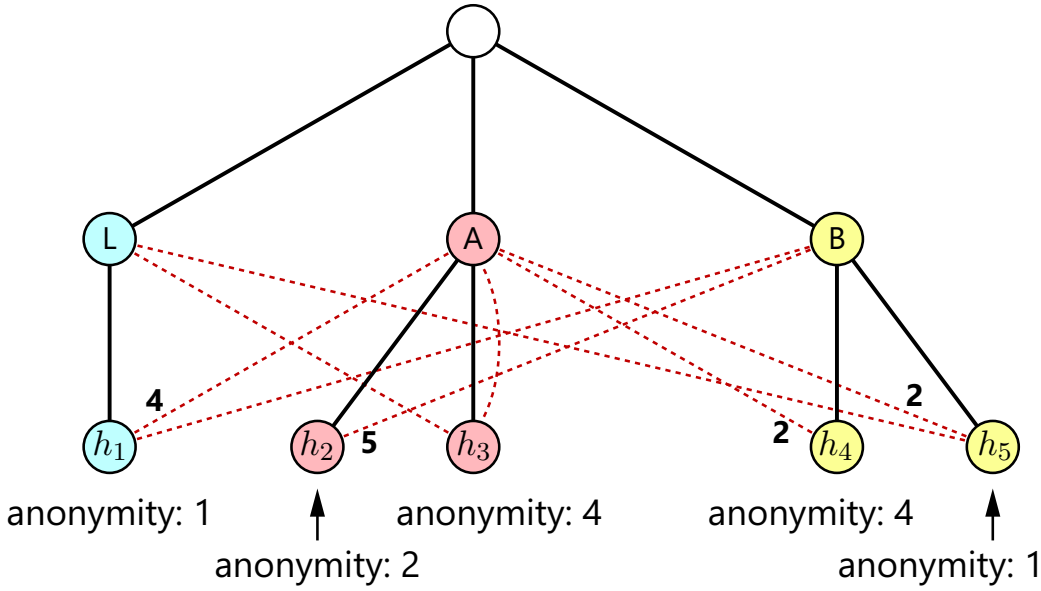




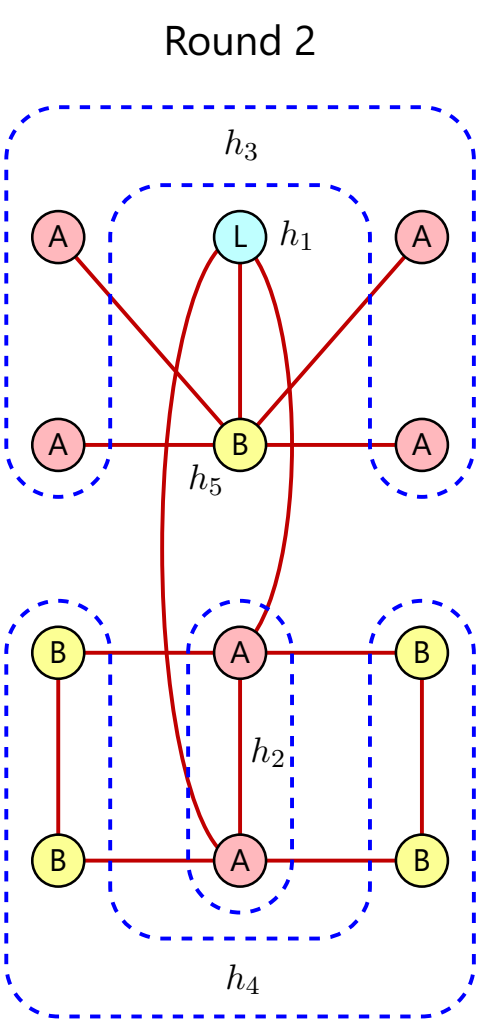
History tree

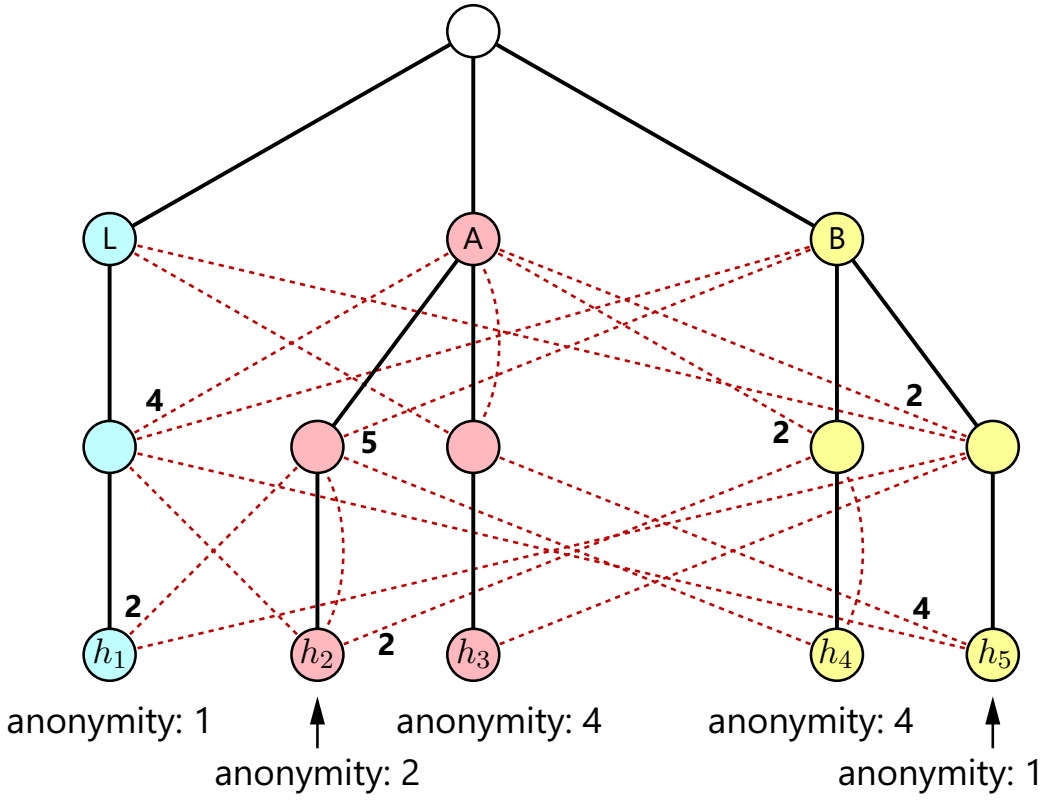
Constructing a history tree



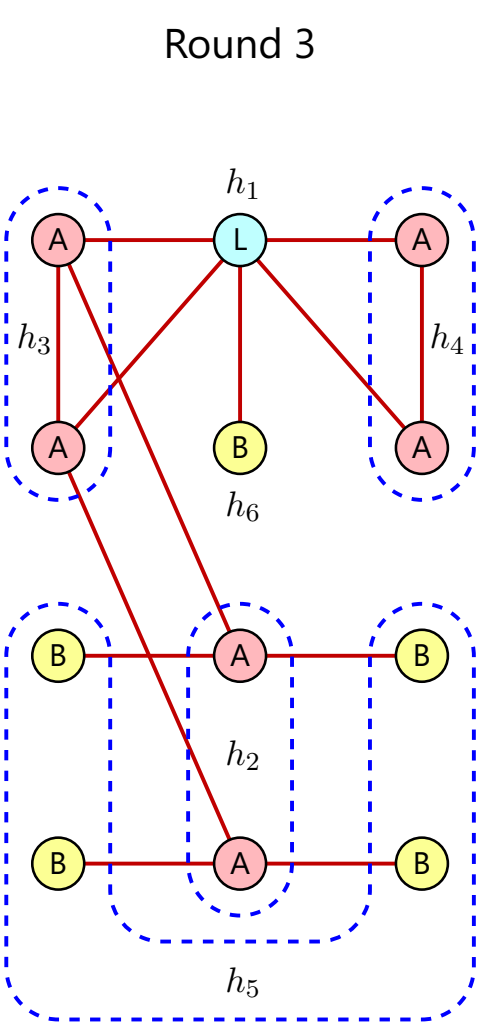


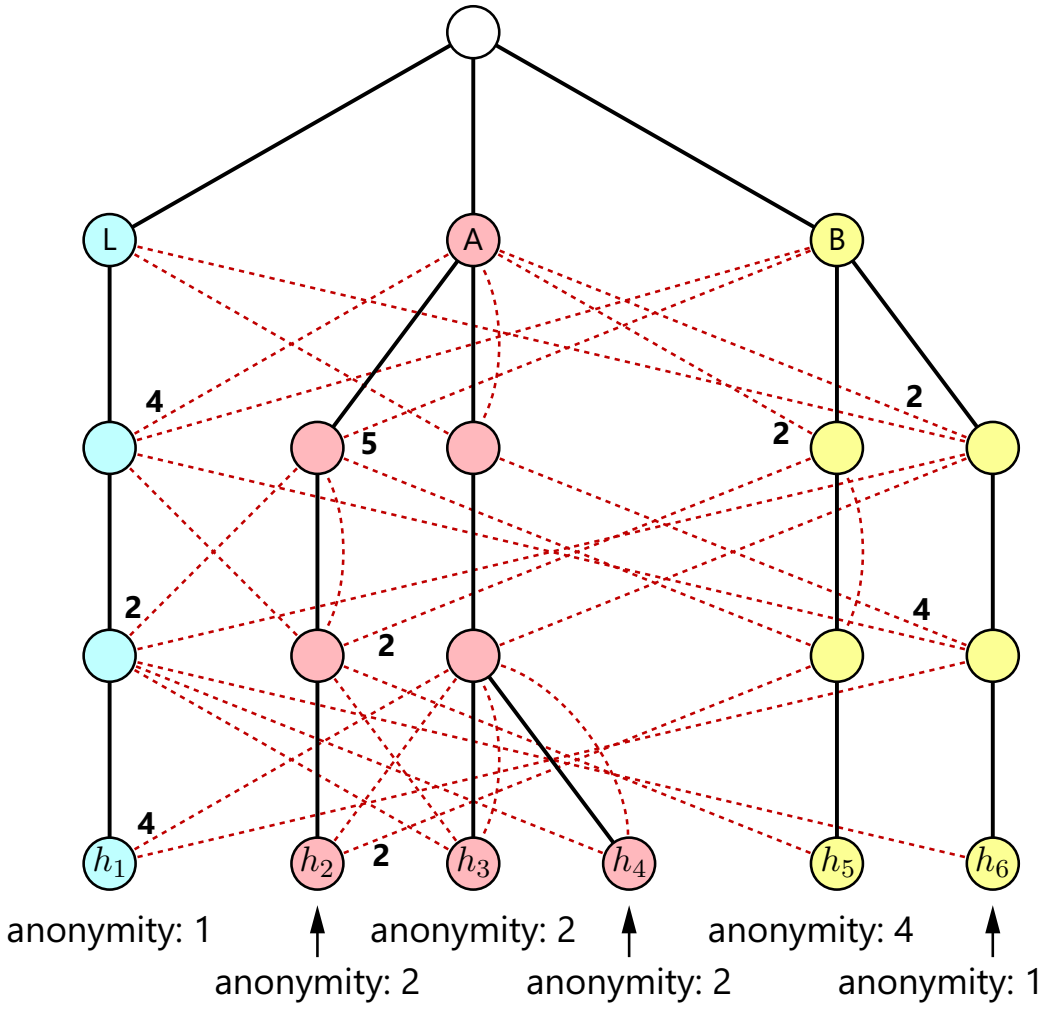
Constructing a history tree



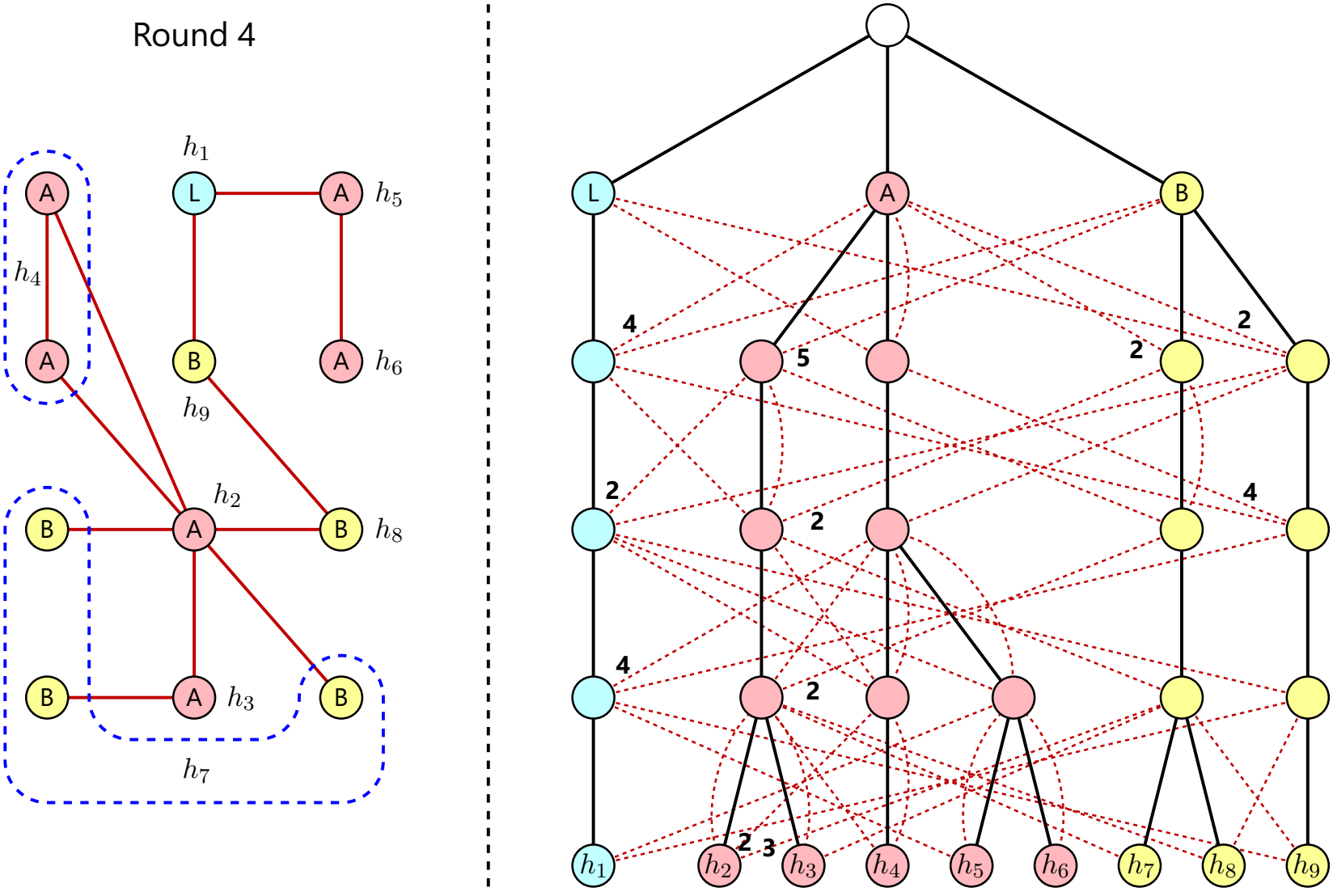


Constructing a history tree



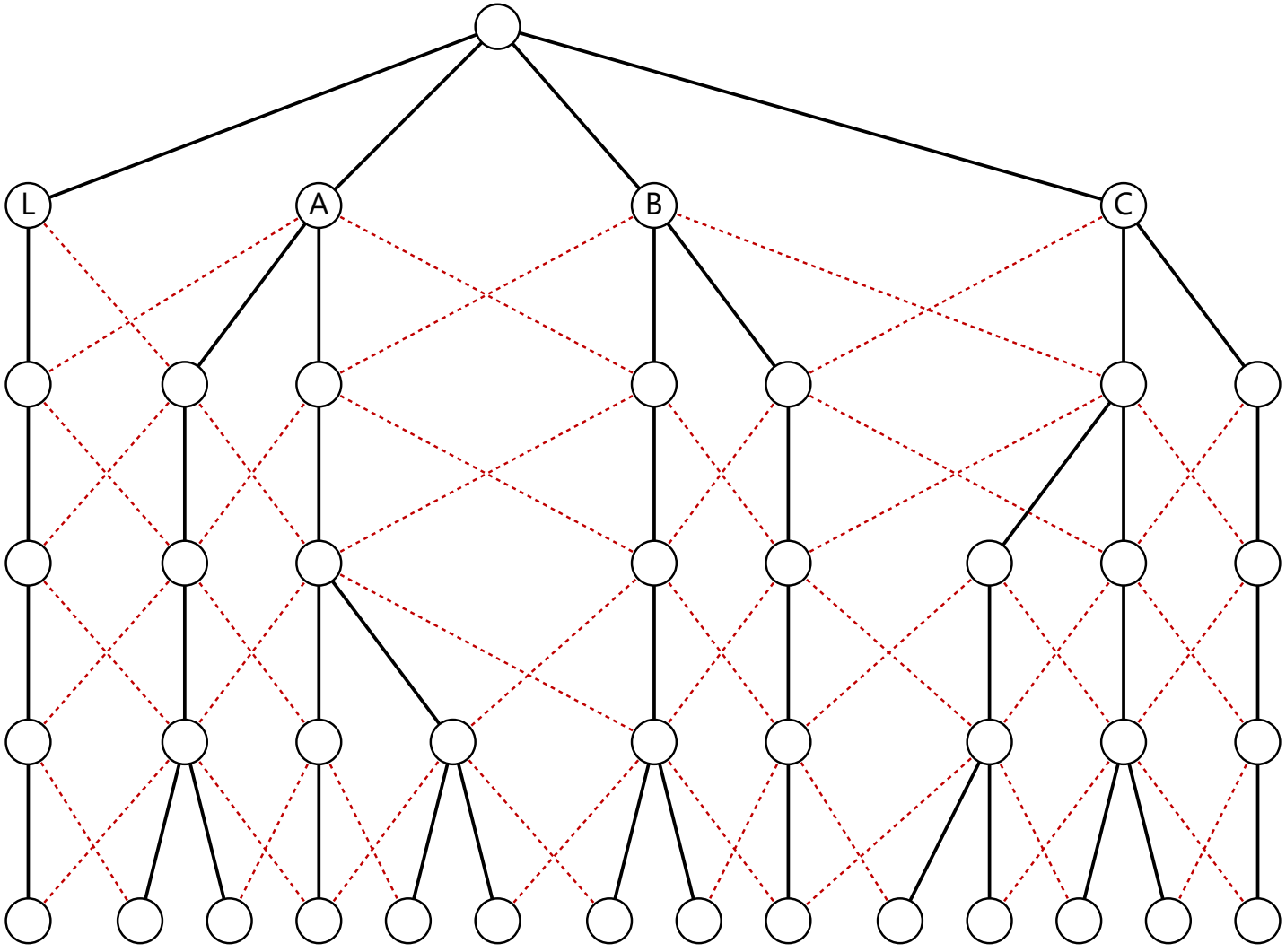


Constructing a history tree



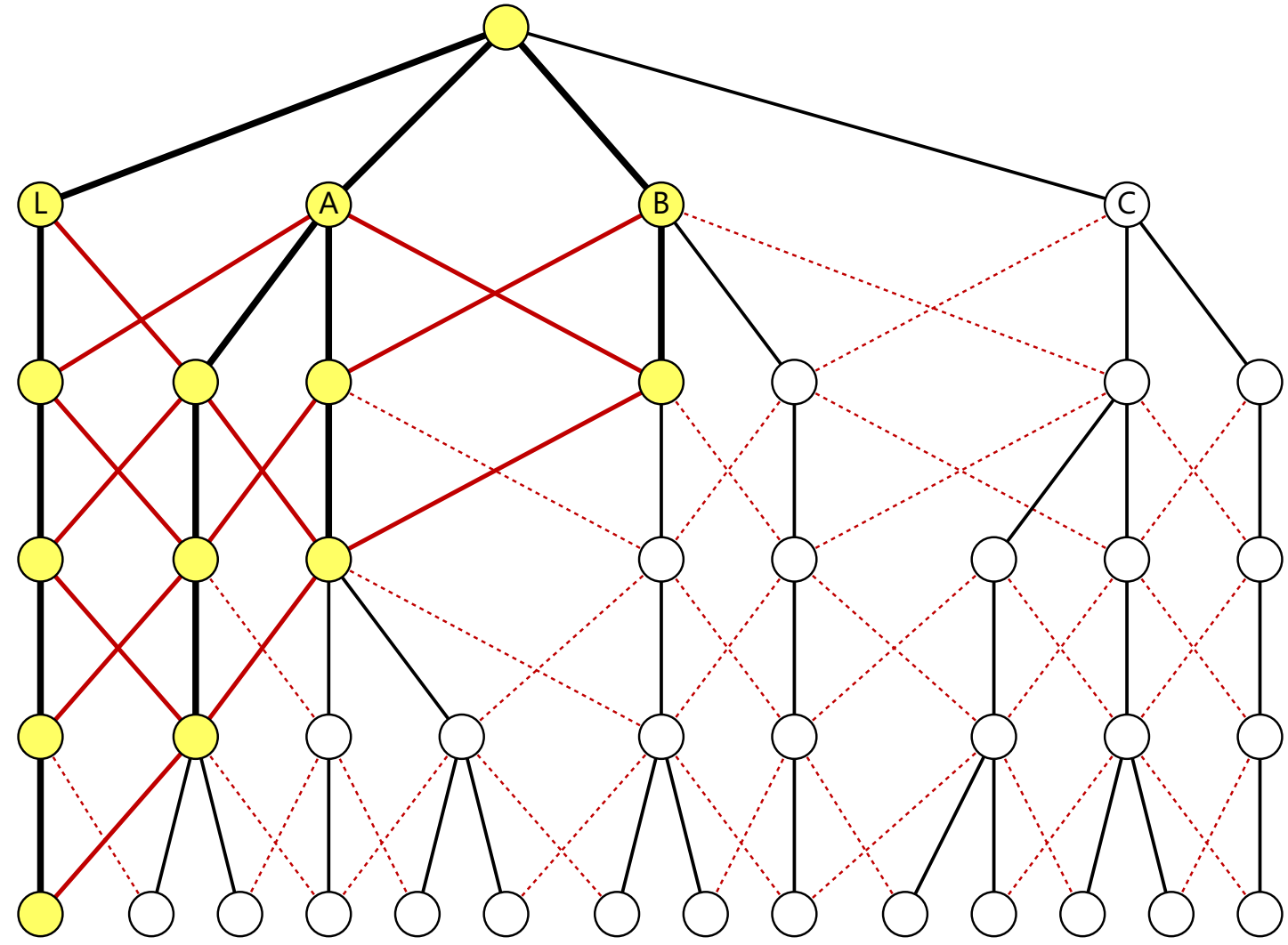
View of a history tree

At any point in time, an agent only has a *view* of the history tree.



View of a history tree

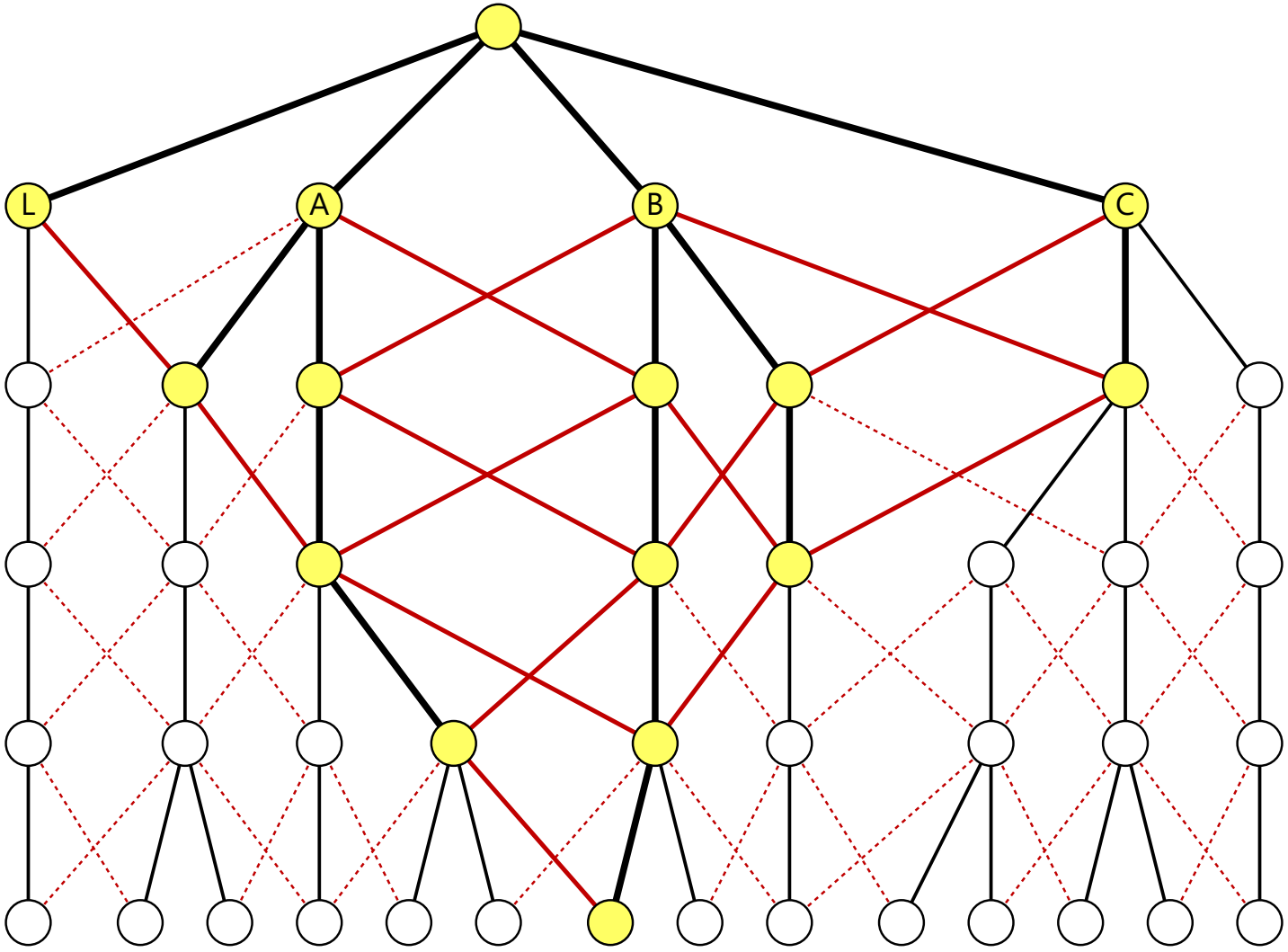
At any point in time, an agent only has a *view* of the history tree.



view

View of a history tree

At any point in time, an agent only has a *view* of the history tree.



view

Views as internal states and messages

An agent's view summarizes its whole *history* up to some round. It contains all the information the agent has learned about the network up to that point.

Observation

Without loss of generality, we may assume that an agent's internal state coincides with its view of the history tree.

Observation

Without loss of generality, we may assume that an agent broadcasts its own internal state at every round.

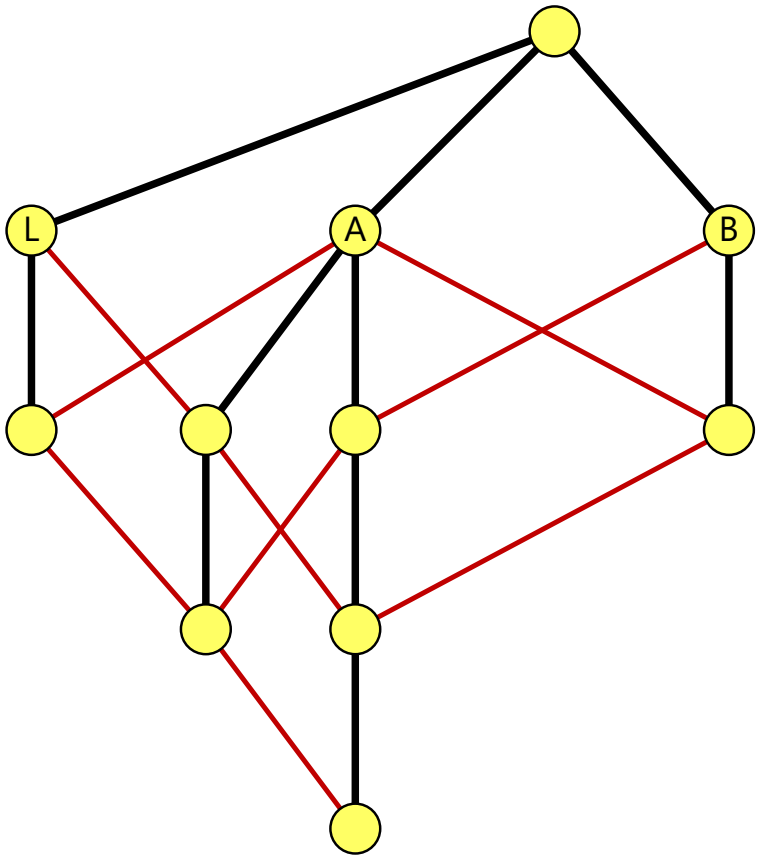
At round t, the size of a view is only $O(tn^2 \log n)$ bits.

Observation

If a problem is solvable in a polynomial number of rounds, it can be solved by using a polynomial amount of local memory and sending messages of polynomial size.

Updating the view

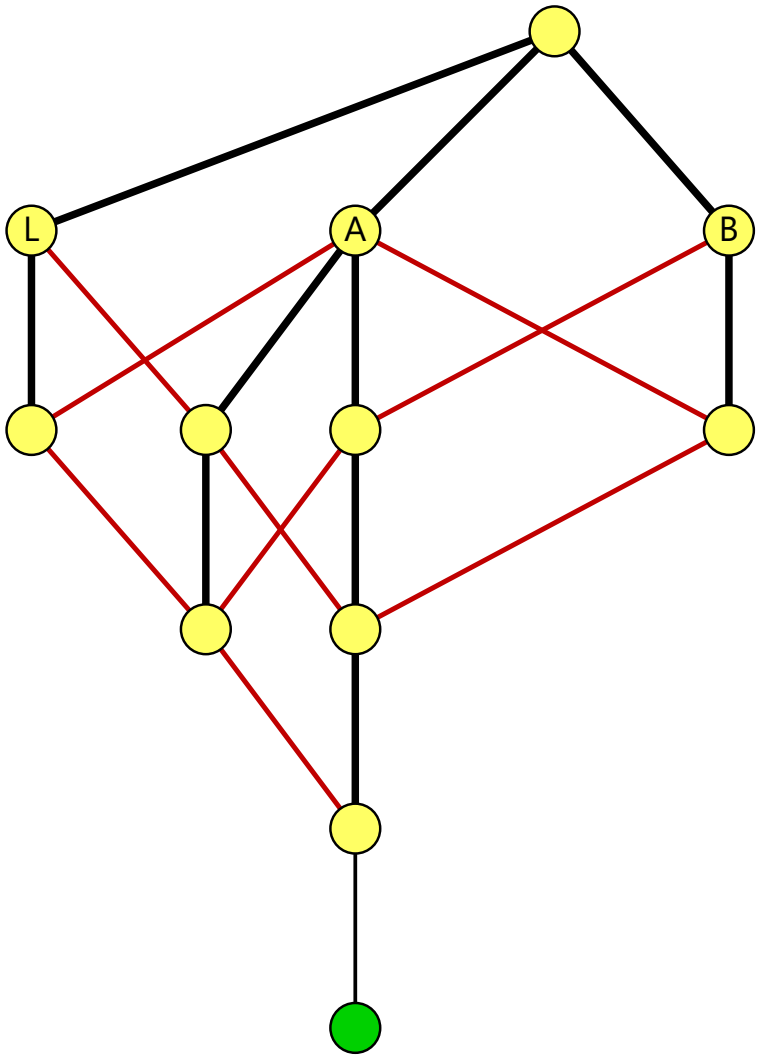
An agent updates its internal state by *merging* its view with the views it receives from its neighbors.



view

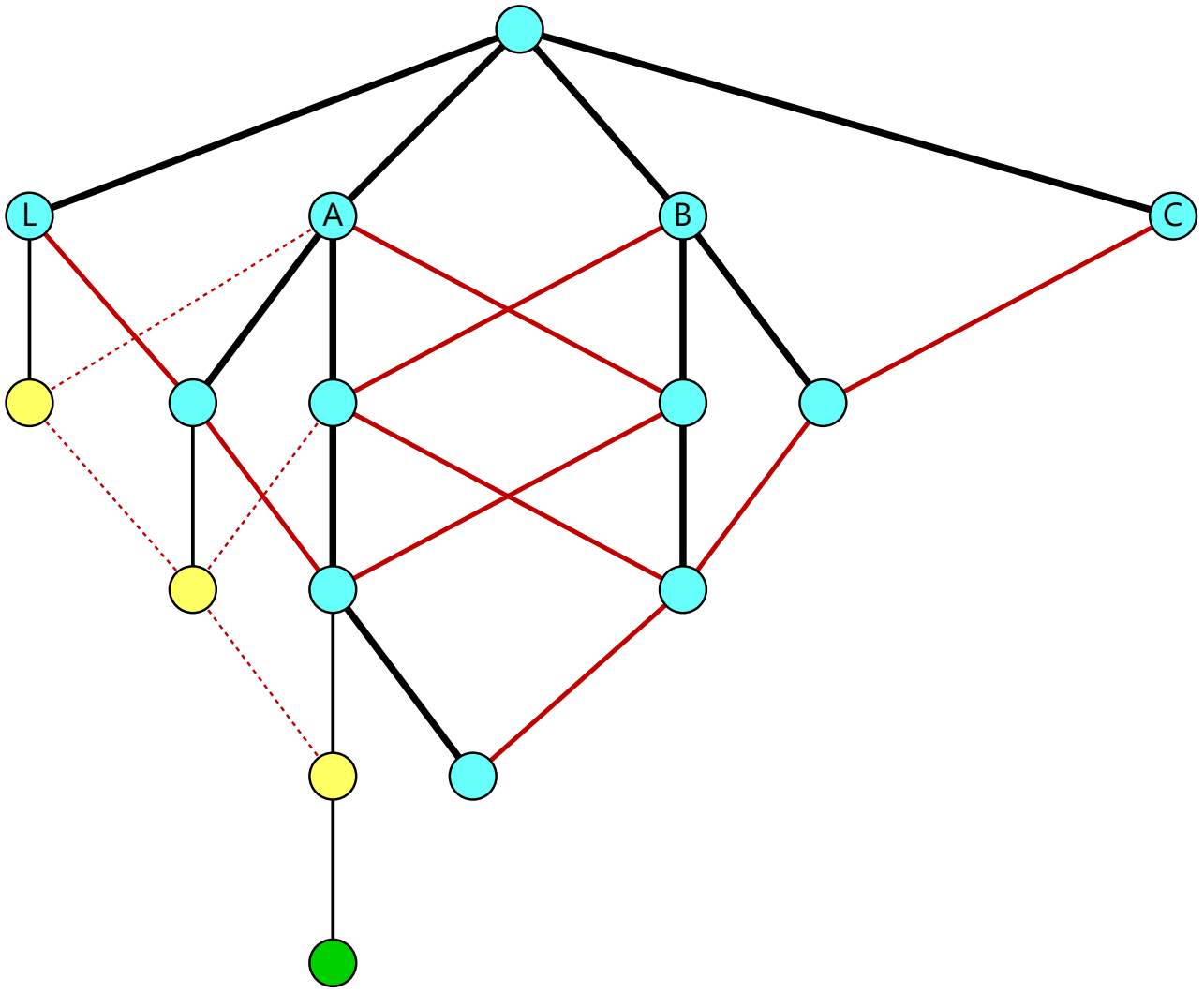
Updating the view

An agent updates its internal state by *merging* its view with the views it receives from its neighbors.



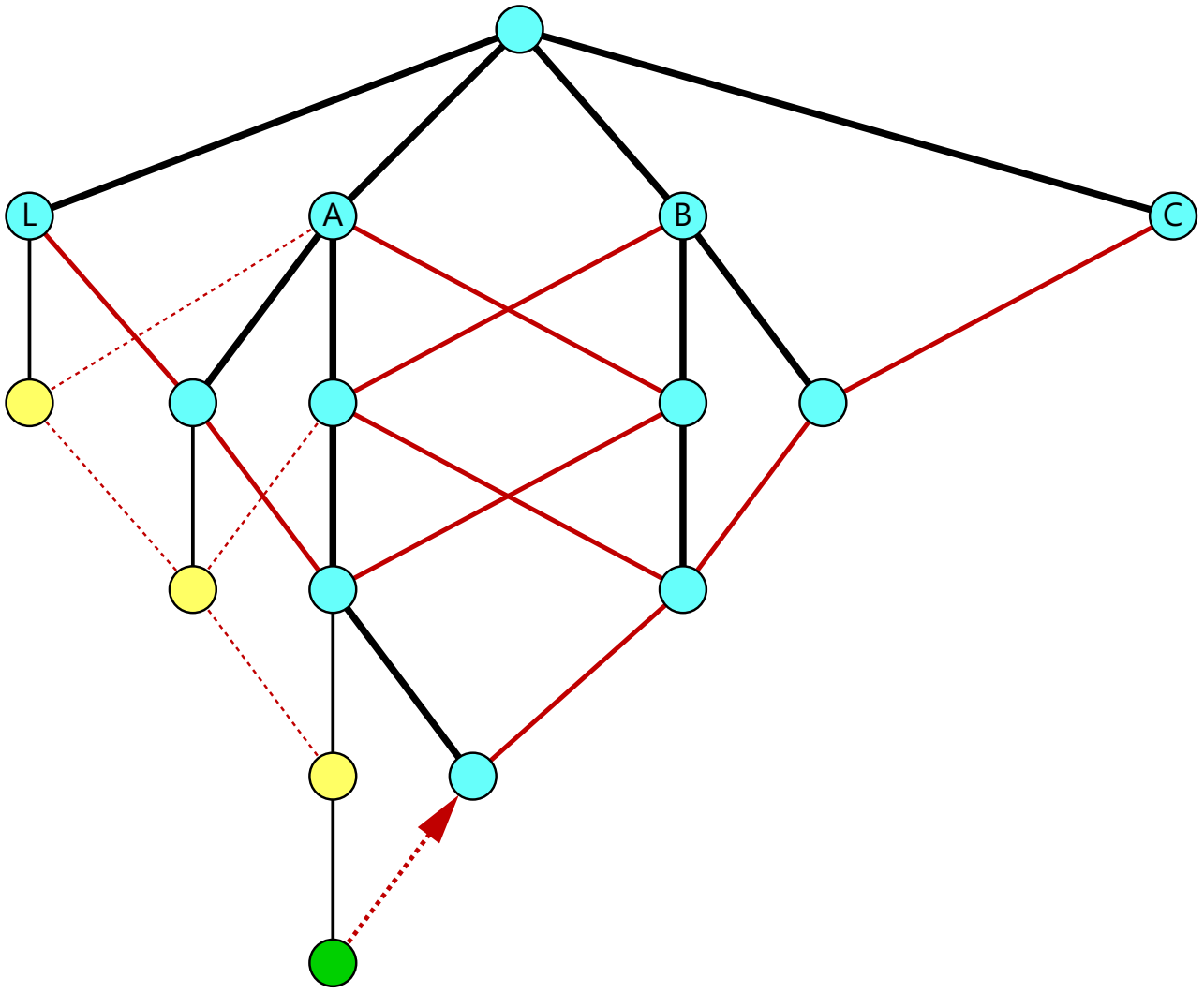
Updating the view

An agent updates its internal state by *merging* its view with the views it receives from its neighbors.



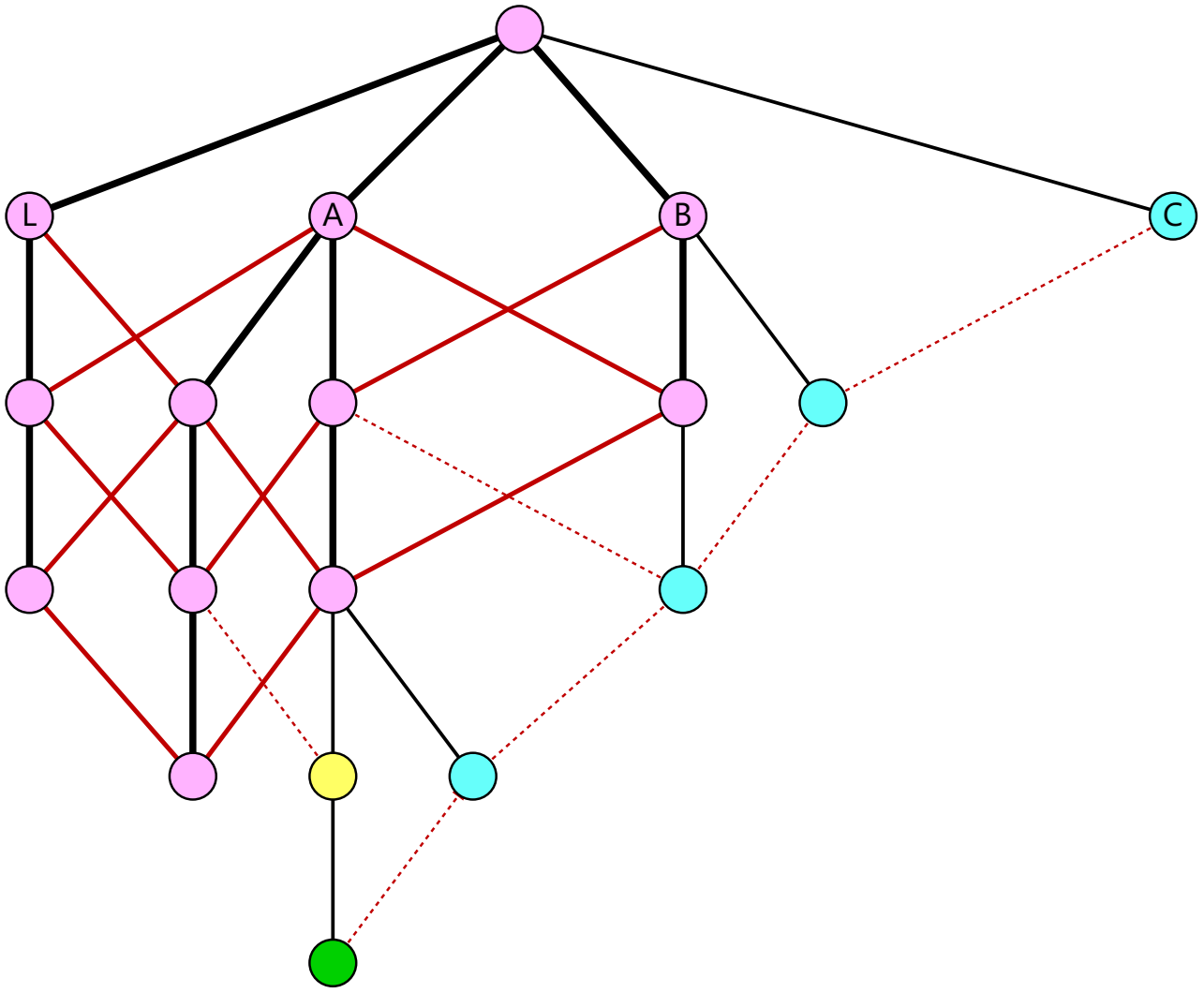
Updating the view

An agent updates its internal state by *merging* its view with the views it receives from its neighbors.



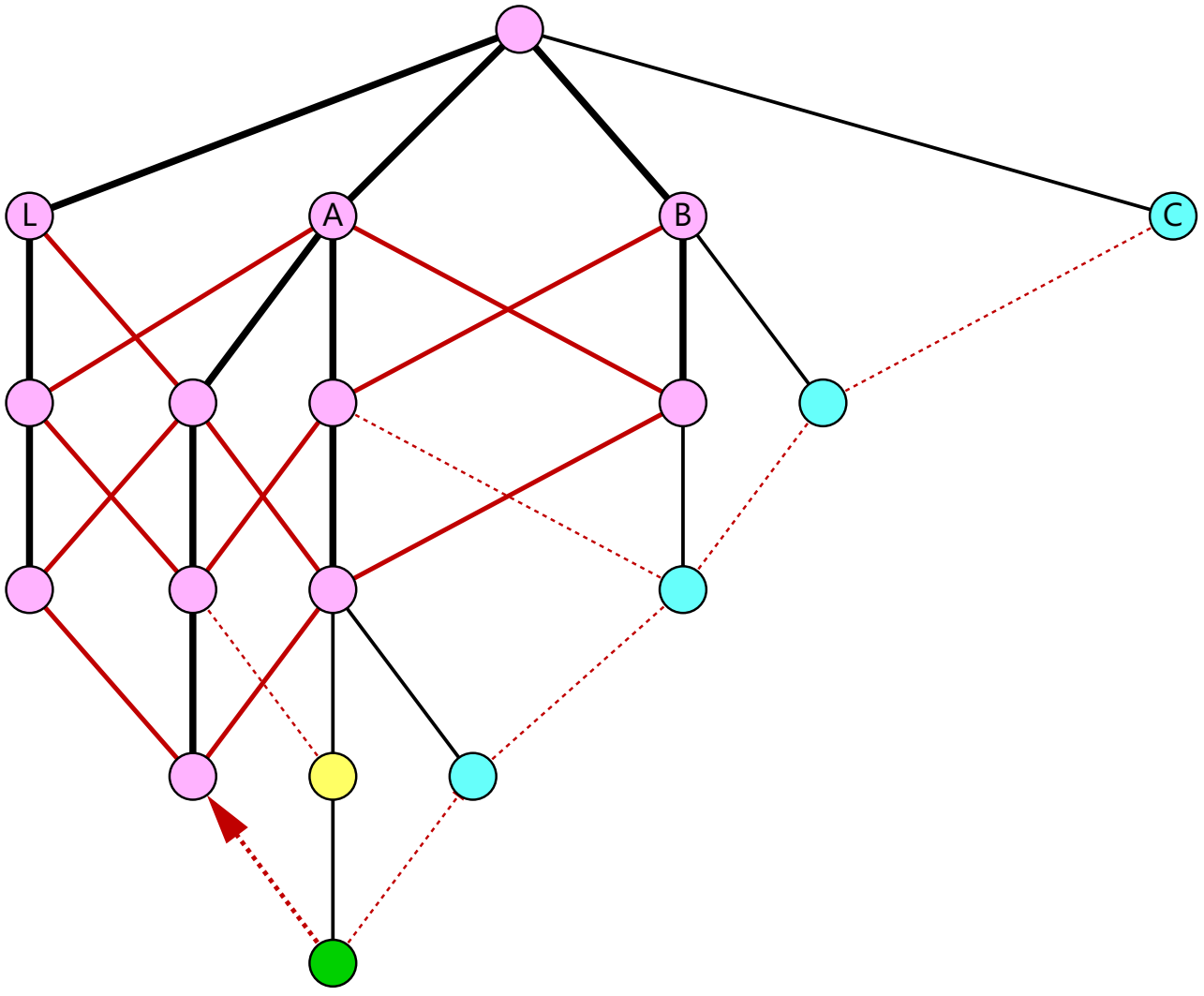
Updating the view

An agent updates its internal state by *merging* its view with the views it receives from its neighbors.



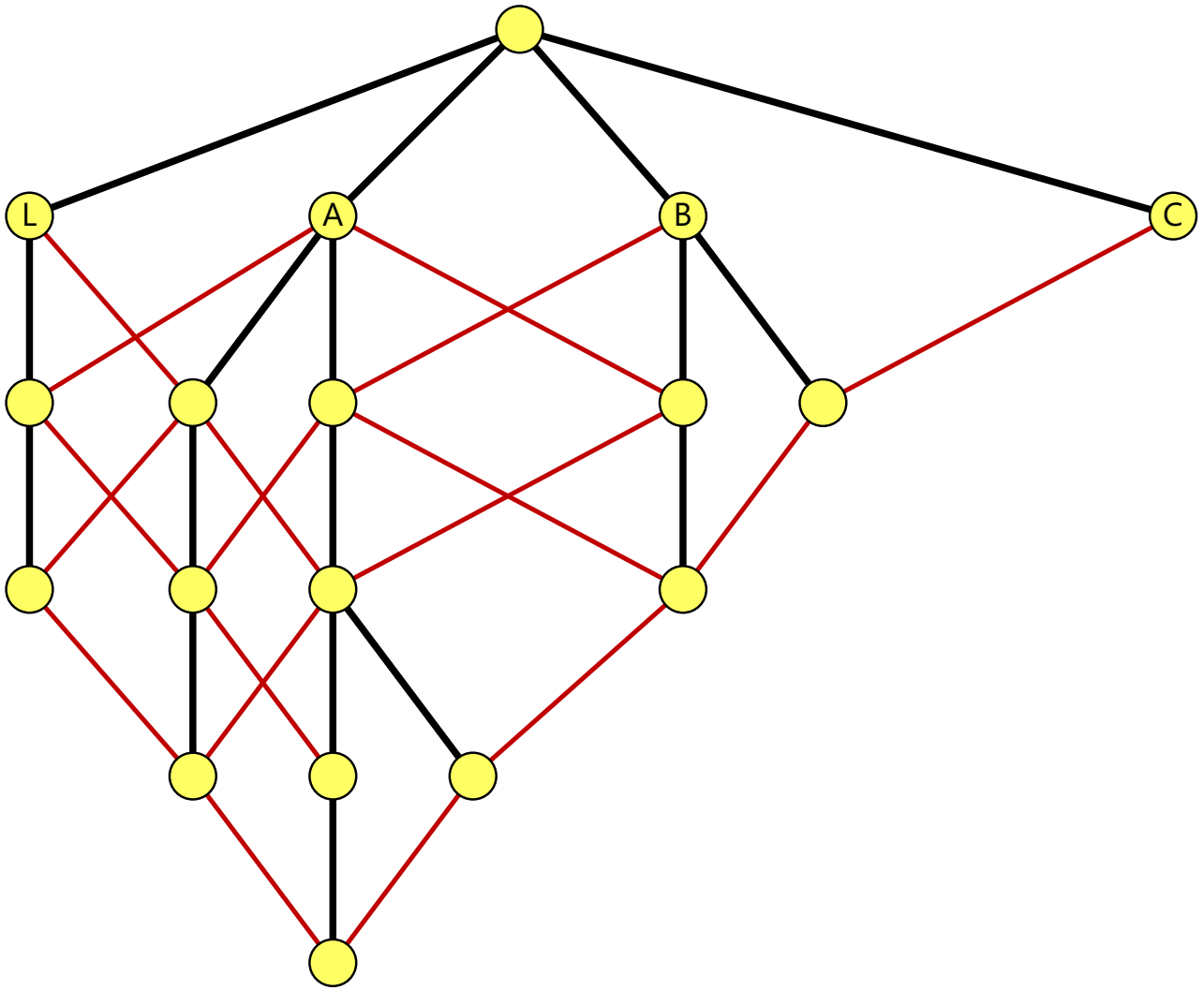
Updating the view

An agent updates its internal state by *merging* its view with the views it receives from its neighbors.



Updating the view

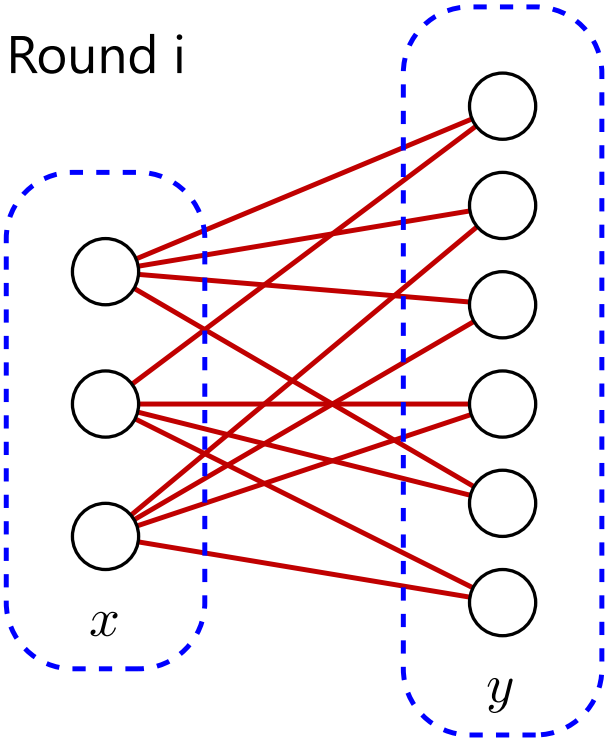
An agent updates its internal state by *merging* its view with the views it receives from its neighbors.

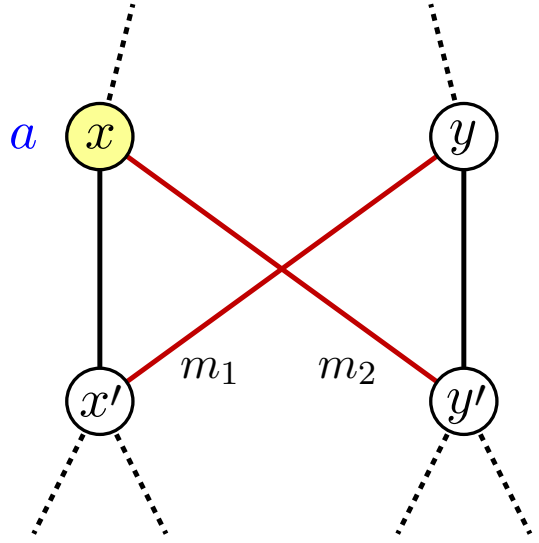


new view

Computing anonymities

Suppose we know the anonymity of a node x with a single child x'.

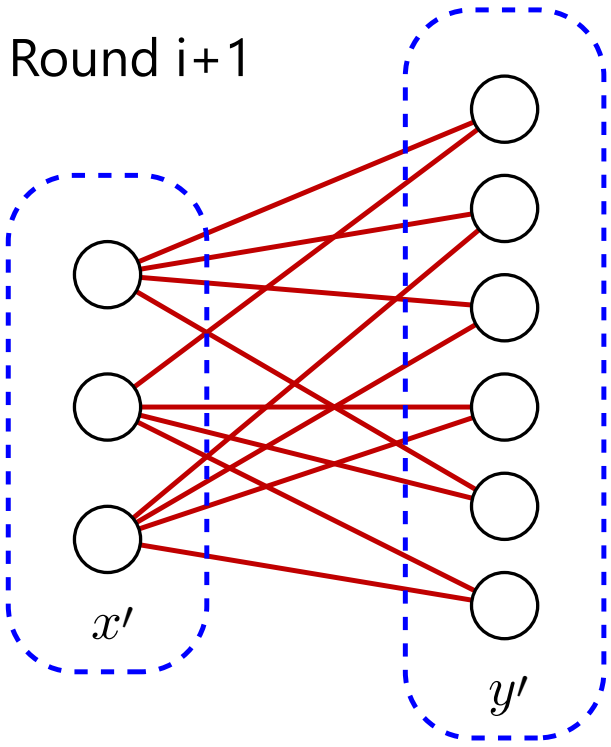


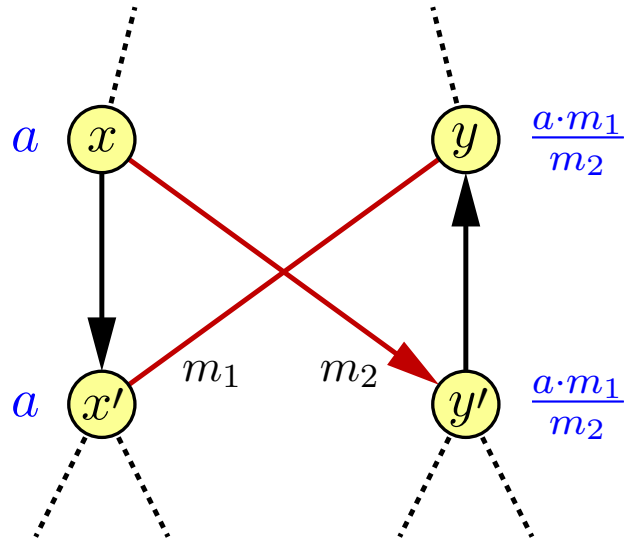


If the agents represented by x have observed agents whose corresponding node y has only one child y', then we can compute the anonymity of y and y', as well.

Computing anonymities

Suppose we know the anonymity of a node x with a single child x'.

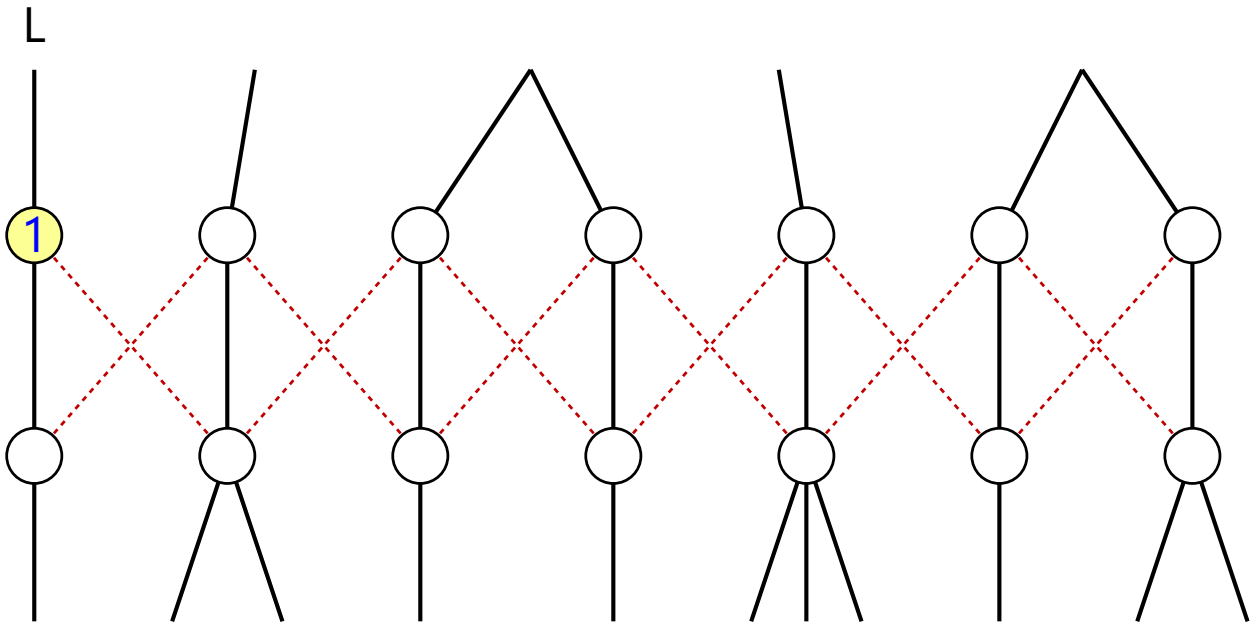




If the agents represented by x have observed agents whose corresponding node y has only one child y', then we can compute the anonymity of y and y', as well.

Stabilizing algorithm

If all nodes in a level have only one child, we can compute the anonymity of all of them (because the network is connected).

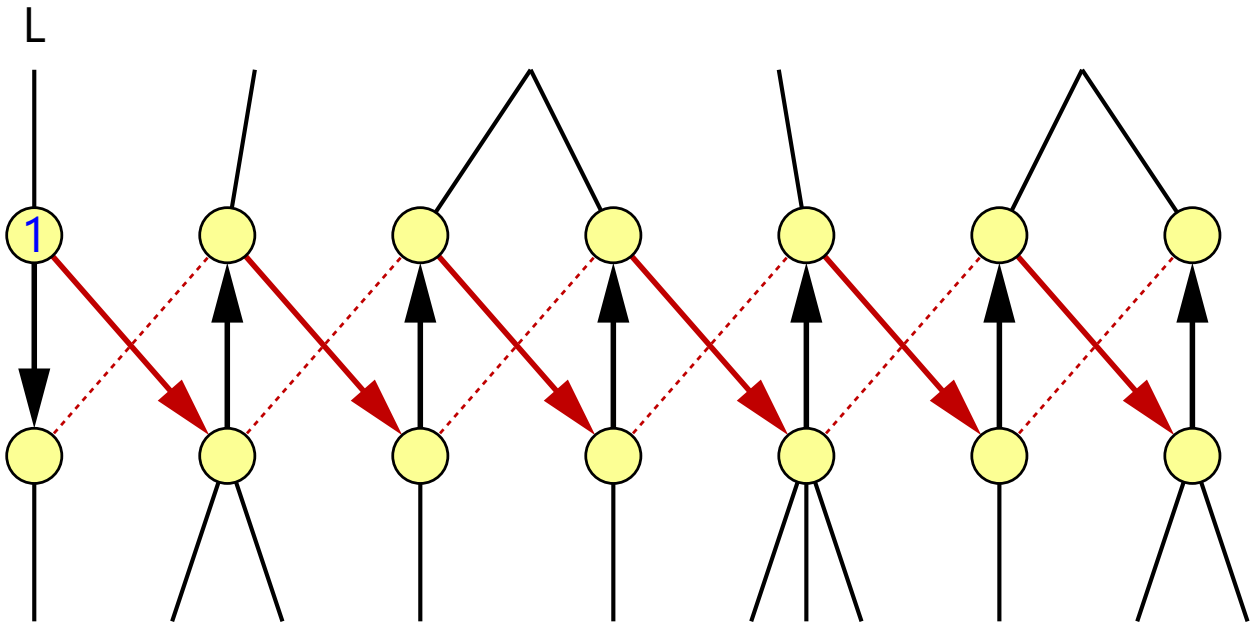


Since there are n agents, the tree can branch at most $n-1$ times.

Thus, among the first $n-1$ levels, there must be a level where no node branches. In this level, we can compute all anonymities.

Stabilizing algorithm

If all nodes in a level have only one child, we can compute the anonymity of all of them (because the network is connected).

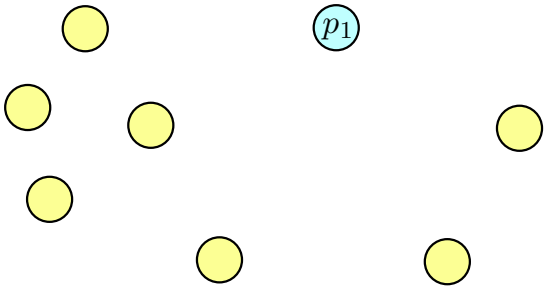


Since there are n agents, the tree can branch at most $n-1$ times.

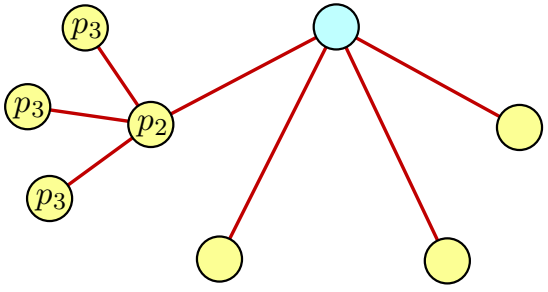
Thus, among the first $n-1$ levels, there must be a level where no node branches. In this level, we can compute all anonymities.

Counterexample for termination

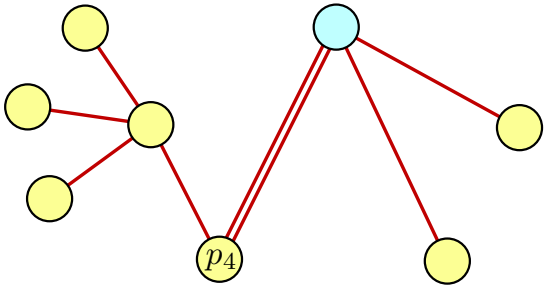
Round 0



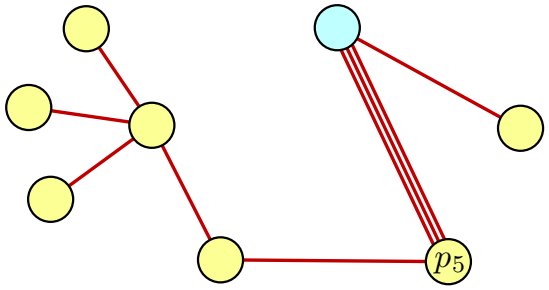
Round 1



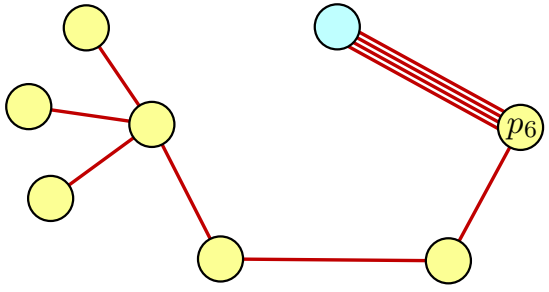
Round 2

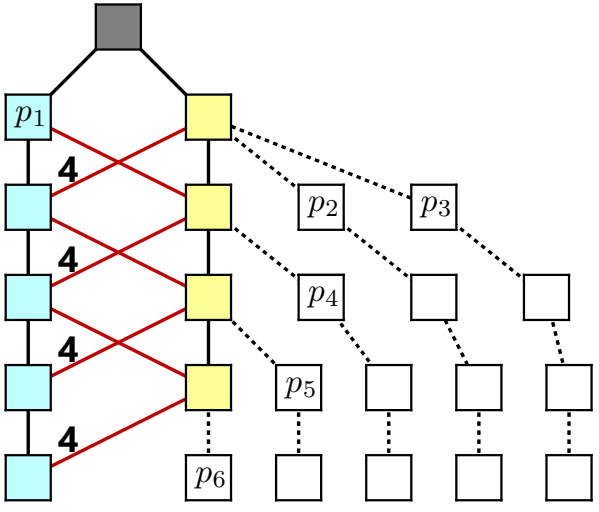


Round 3



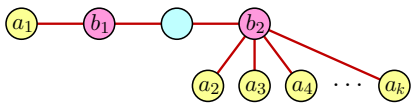
Round 4+



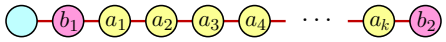


Counterexample for termination

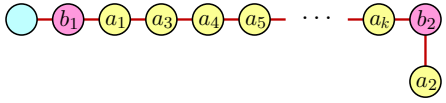
Round 1



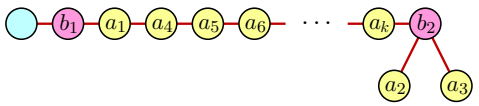
Round 2



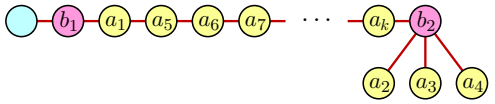
Round 3



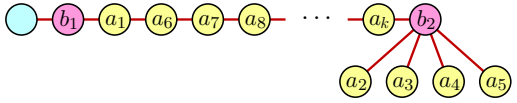
Round 4

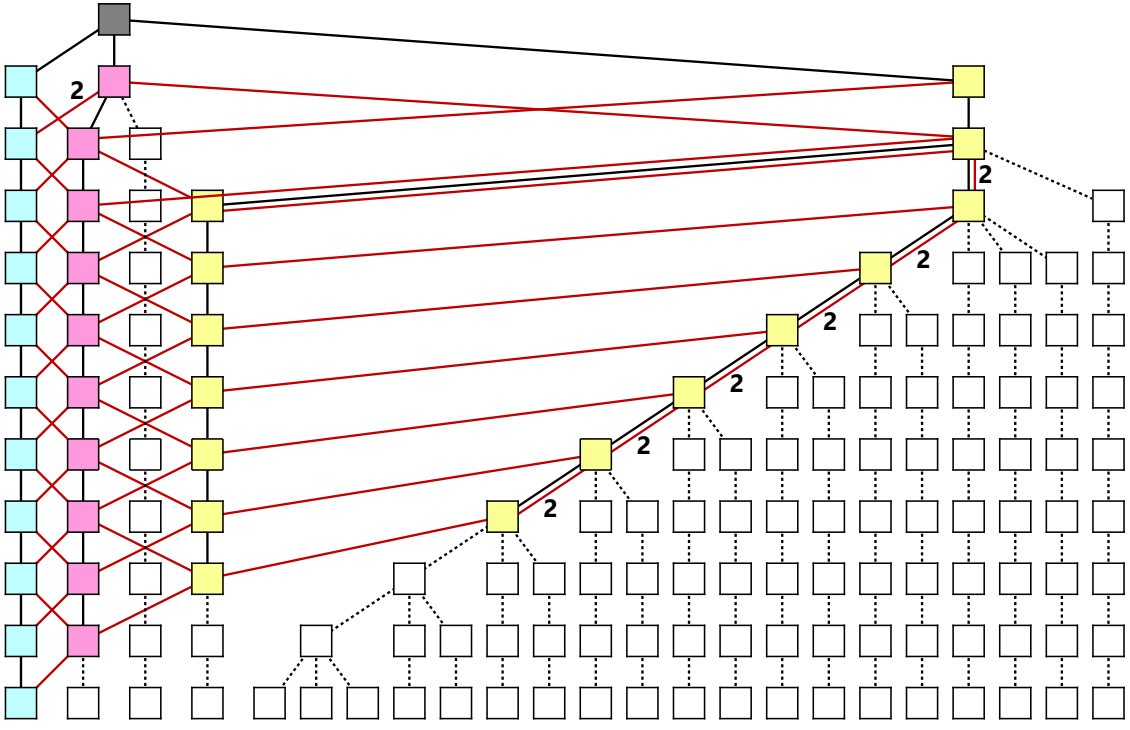


Round 5



Round 6





Terminating Computation with a Unique Leader

Terminating algorithm: Overview

We will give a *terminating* algorithm for the Counting Problem.

The algorithm is as follows:

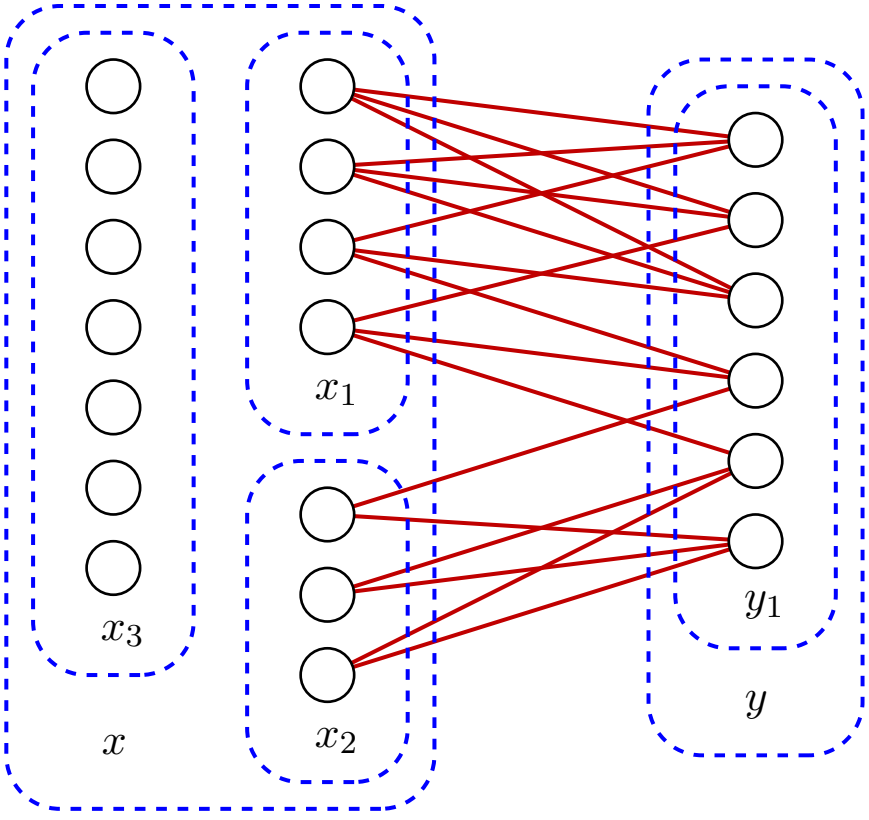
- Use the Leader's observations to make guesses on anonymities.
- In any set of n guesses, we can always identify a correct one.
- Once we have identified $n-1$ correct guesses, we can use some of them to make new guesses on anonymities.
- Repeat until we have the anonymity of all visible branches of the history tree: this gives an estimate n' on n.
- Wait n' rounds to confirm the estimate; if correct, terminate.

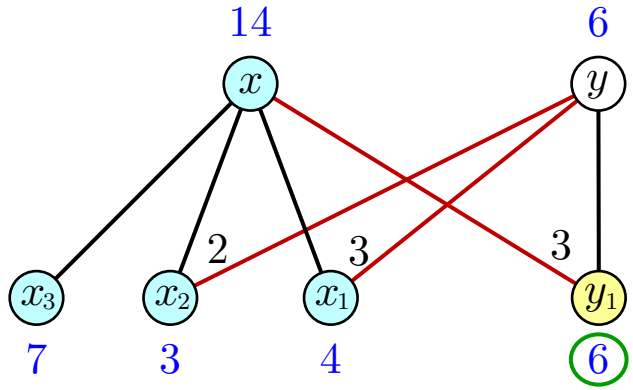
Theorem

The Generalized Counting Problem can be solved in $3n-2$ rounds with explicit termination.

Guessing anonymities

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.



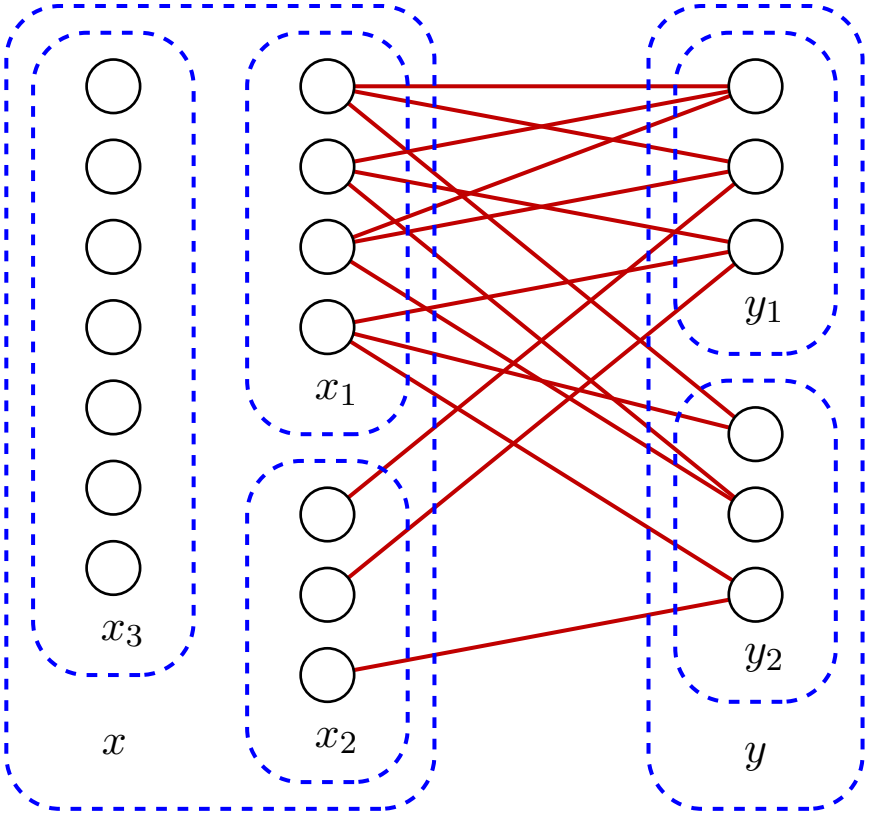


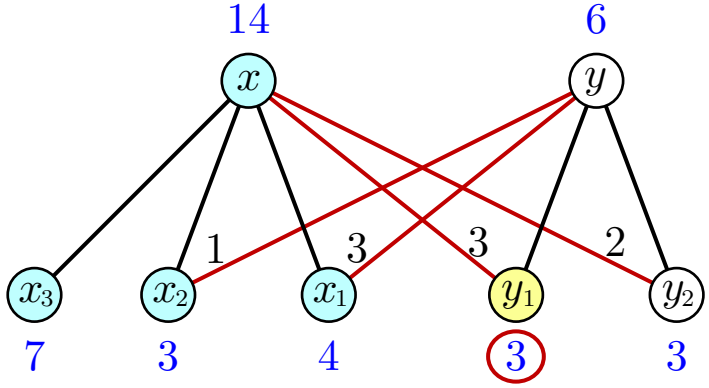
Guess on y_1: $\dfrac{4 \cdot 3 + 3 \cdot 2}{3} = 6$

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

Guessing anonymities

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.



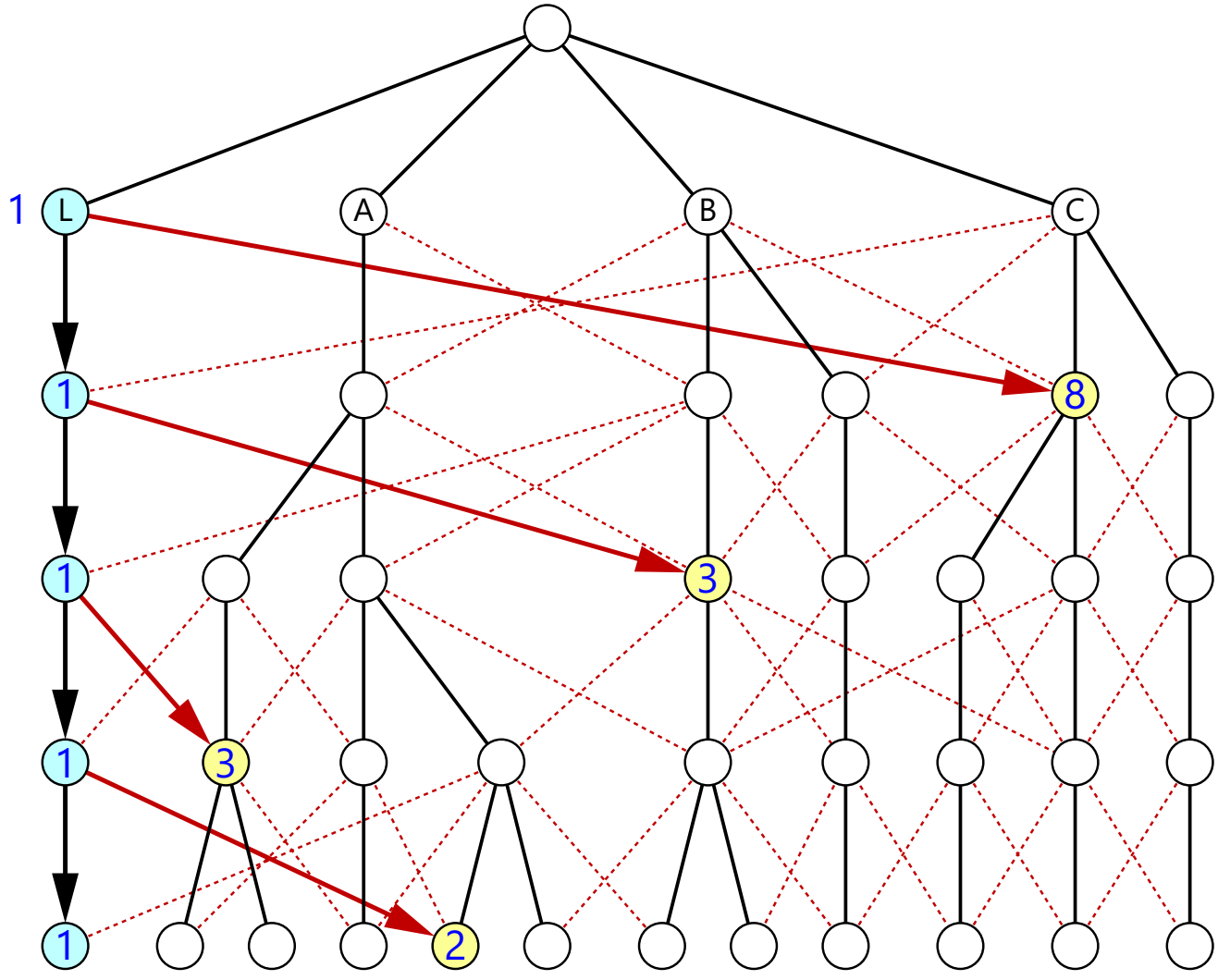


Guess on y_1: $\dfrac{4 \cdot 3 + 3 \cdot 1}{3} = 5$

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

Guessing anonymities from the Leader

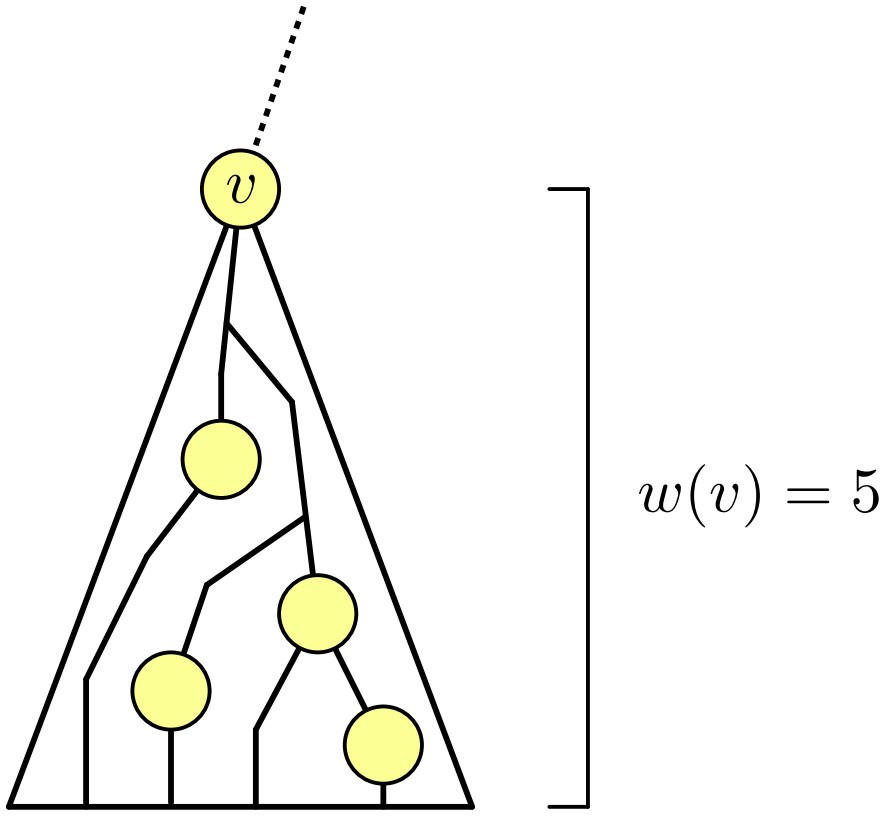
We can make one guess per round using the Leader's observations.



How do we know which guesses are correct?

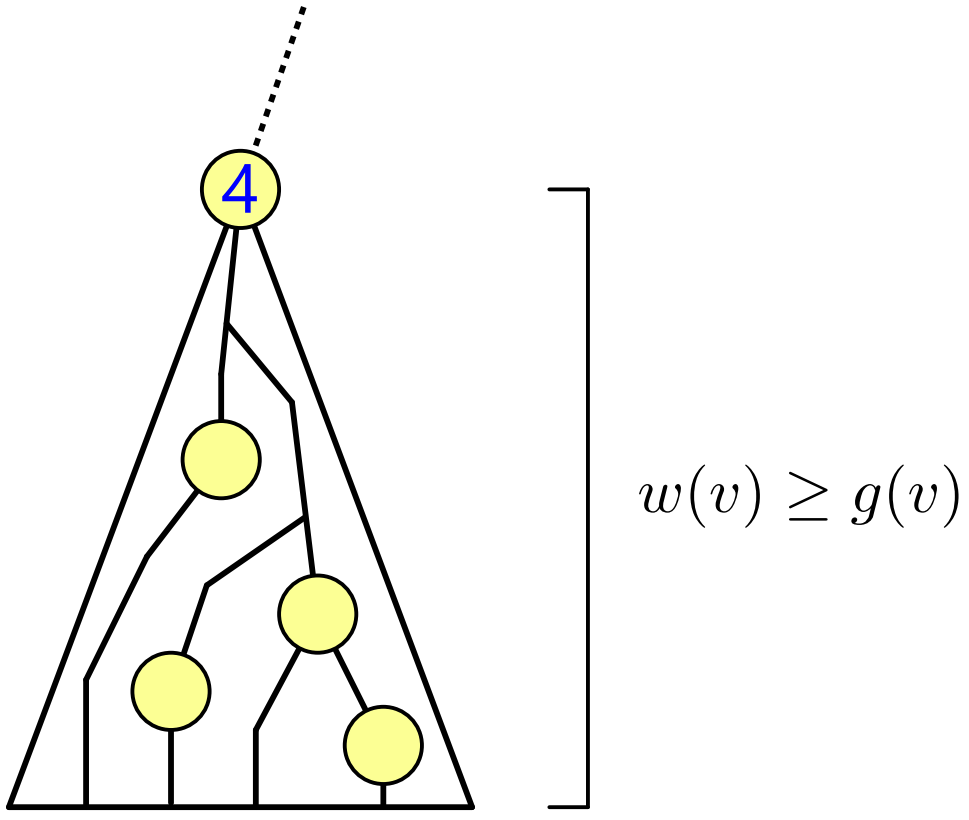
Weight of a node

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.



Weight of a node

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



Limiting theorem

We denote by $a(v)$ the anonymity of a node v, by $g(v)$ a guess on $a(v)$, and by $w(v)$ the weight of v.

Theorem

If all guesses are on different levels and $w(v) > a(v)$, then some descendants of v are heavy.

Proof. By well-founded induction on $w(v)$.

Let v_1, v_2, ... be the closest descendants of v that have guesses. Of course, $a(v) \geq \sum_i a(v_i)$.

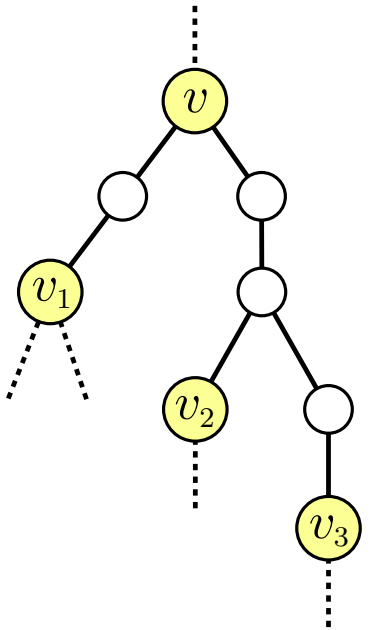
By the inductive hypothesis, $w(v_i) \leq a(v_i)$ for all i.

$w(v)-1 = \sum_i w(v_i) \leq \sum_i a(v_i) \leq a(v) \leq w(v)-1$

Thus, $w(v_i) = a(v_i)$ and $a(v) = \sum_i a(v_i)$.

The deepest node v_d has no siblings, because all guesses are on different levels.

Hence $g(v_d) = a(v_d) = w(v_d)$, and v_d is heavy. □



Criterion of correctness

Corollary

If v is heavy and no descendant of v is heavy, then $g(v) = a(v)$.

Proof. By assumption, $g(v) \leq w(v)$.

By the limiting theorem, $w(v) \leq a(v)$.

Guesses never underestimate anonymities, and so $a(v) \leq g(v)$.

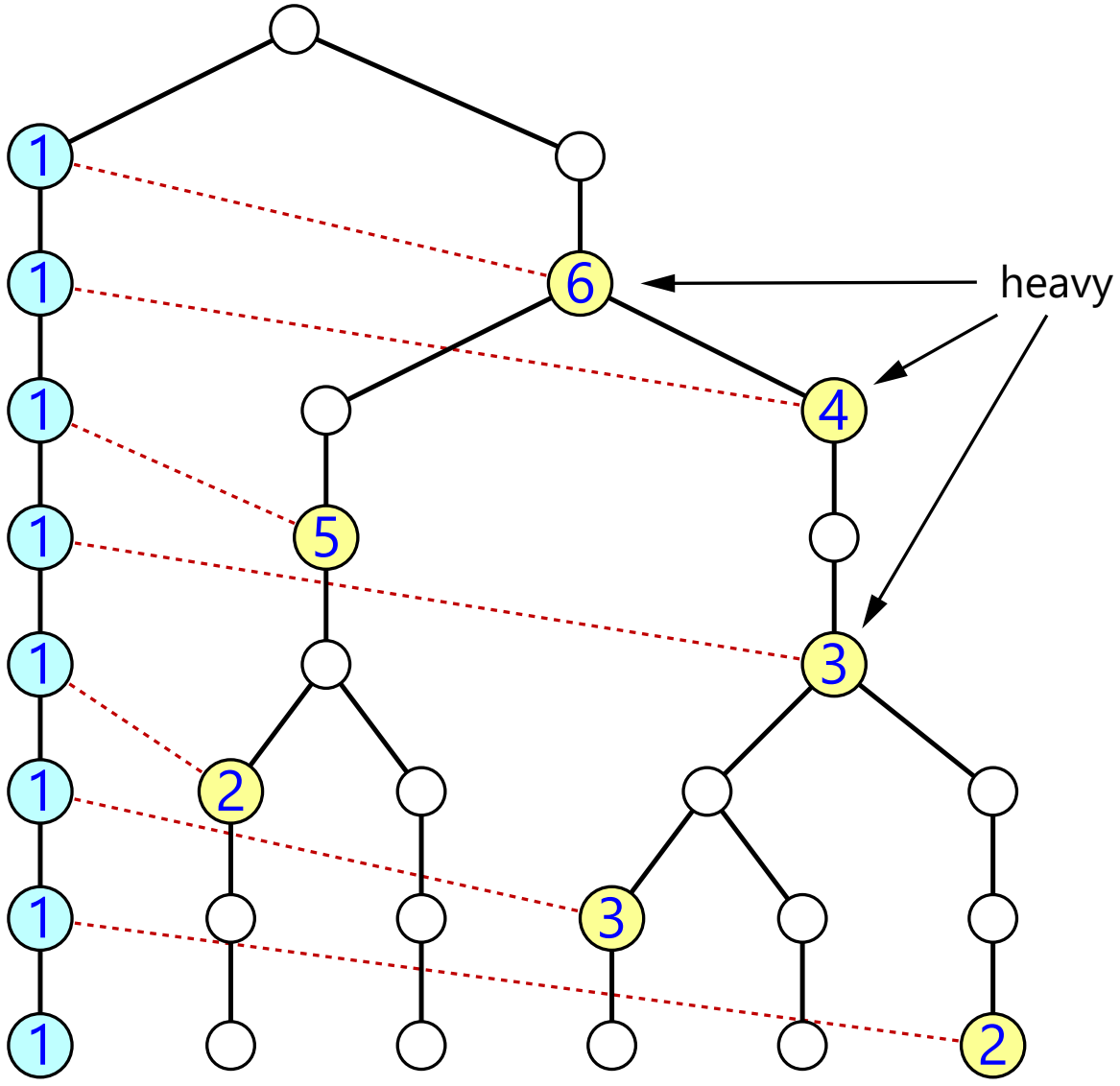
$g(v) \leq w(v) \leq a(v) \leq g(v)$, hence $g(v) = a(v)$. □

This corollary gives agents a criterion to determine when a guess is necessarily correct: **If v is heavy and no descendants of v are heavy, then the guess on v is correct.**

Moreover, by the limiting theorem, such a node v is found by the time there are n guesses in total.

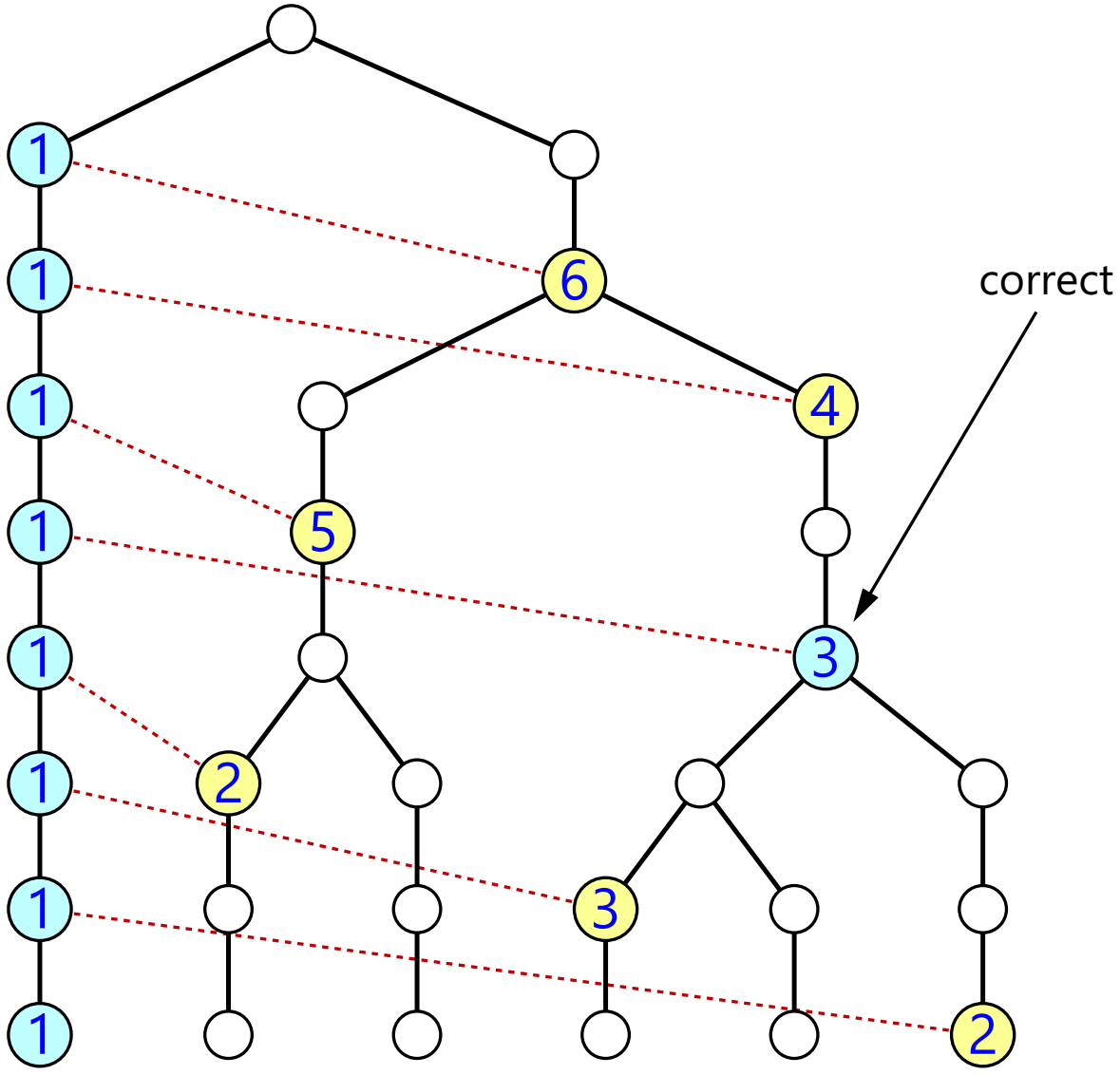
Criterion of correctness: Example

Any agent with this view is able to determine which guess is necessarily correct:



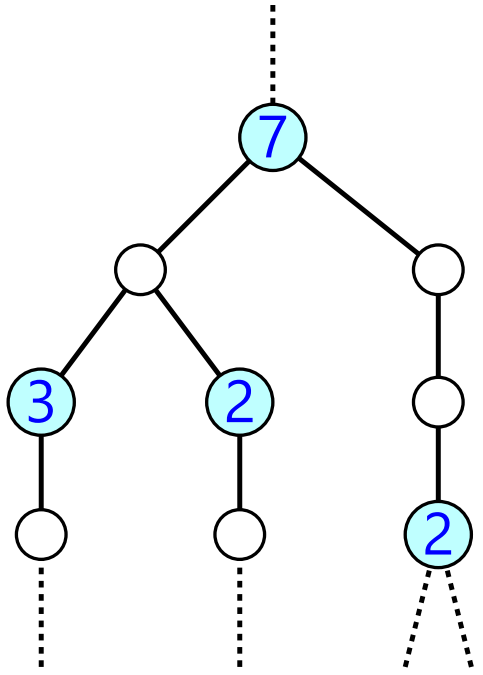
Criterion of correctness: Example

Any agent with this view is able to determine which guess is necessarily correct:



Propagation of guesses

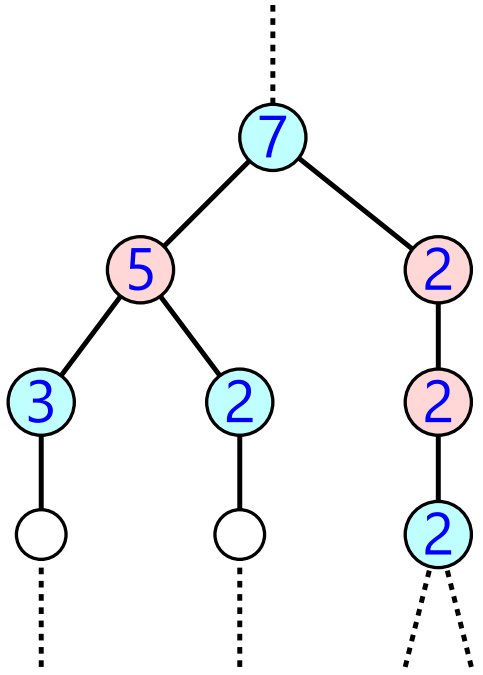
An *island* is a connected component of (a view of) the history tree that contains no leaves and does not contain the root.



Suppose that the nodes with necessarily correct guesses bound an island in the history tree.

Propagation of guesses

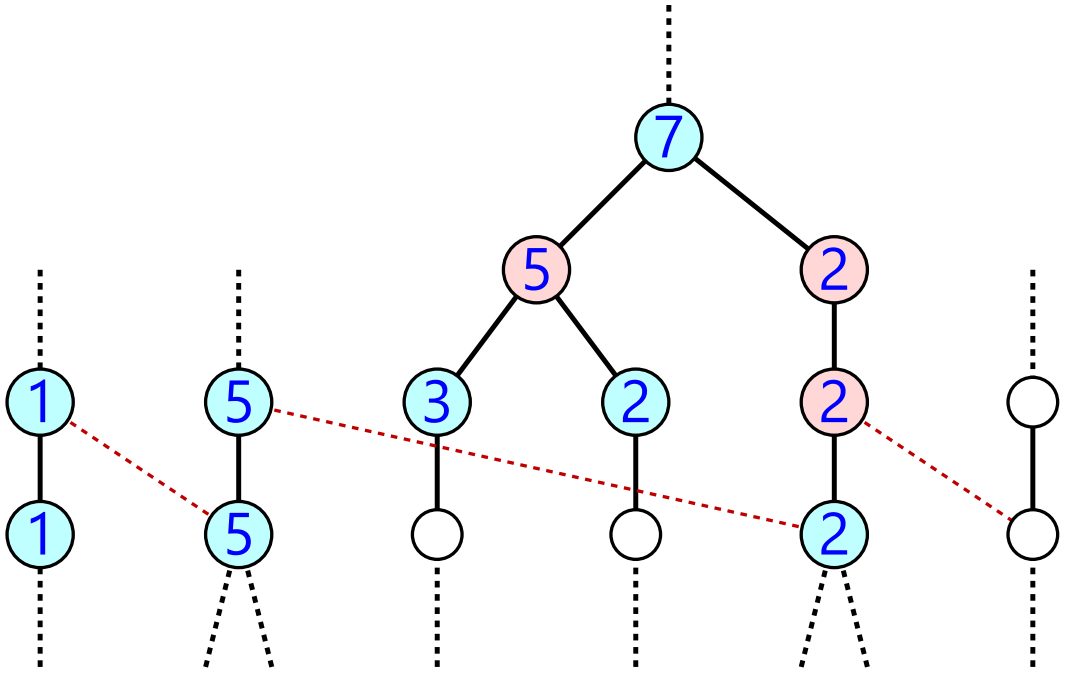
An *island* is a connected component of (a view of) the history tree that contains no leaves and does not contain the root.



If the anonymity of the top node is the sum of the bottom ones, then we can infer the anonymities of all the nodes in the island.

Propagation of guesses

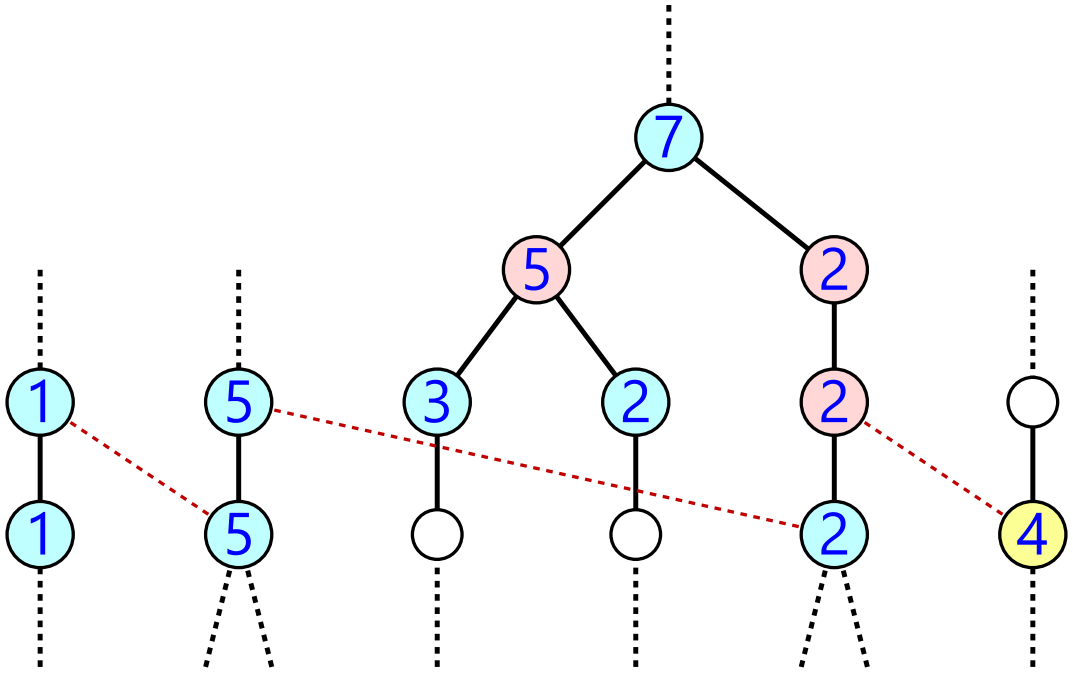
An *island* is a connected component of (a view of) the history tree that contains no leaves and does not contain the root.



Since the network is connected at every round, we can make a new guess from one of the nodes in the island.

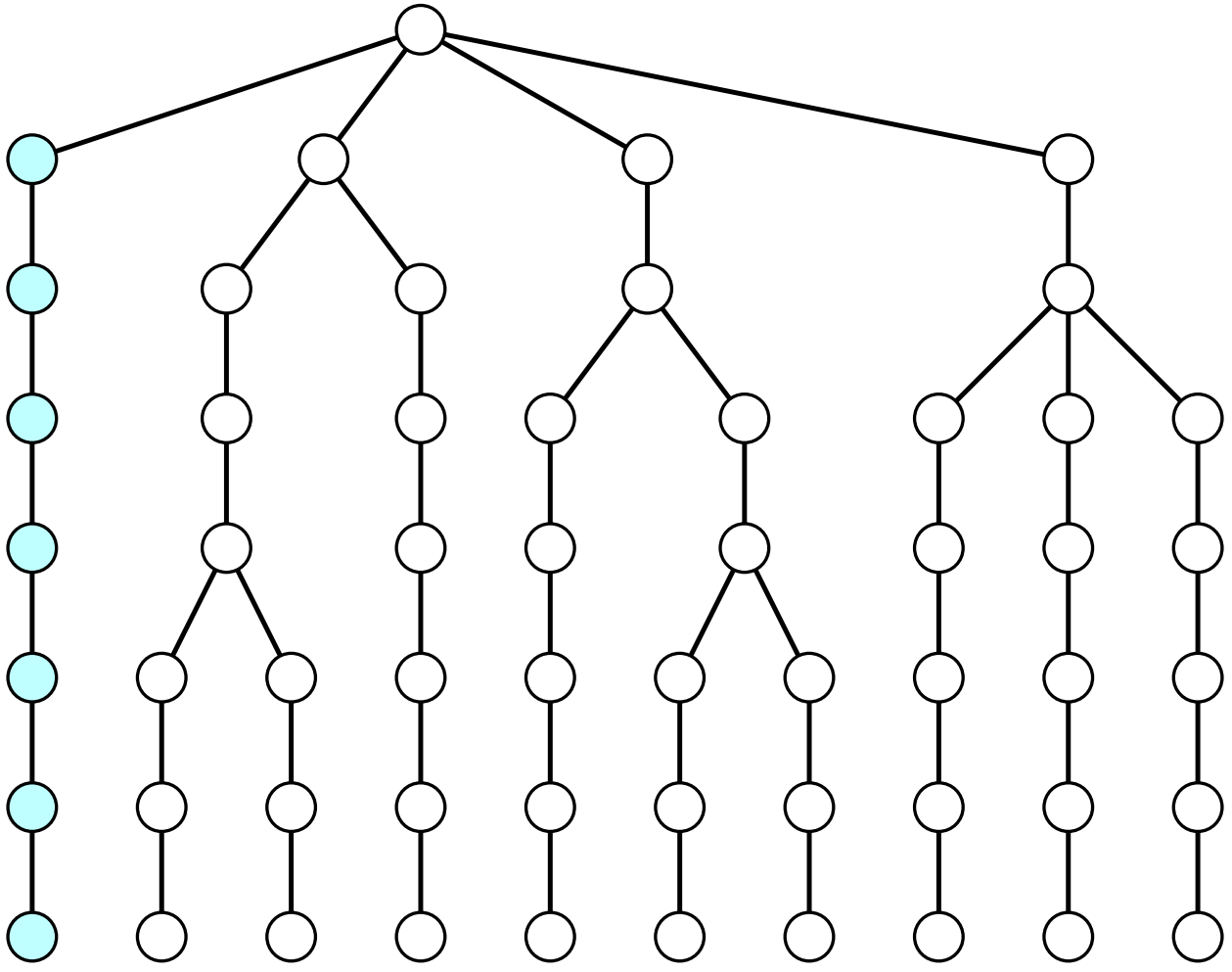
Propagation of guesses

An *island* is a connected component of (a view of) the history tree that contains no leaves and does not contain the root.



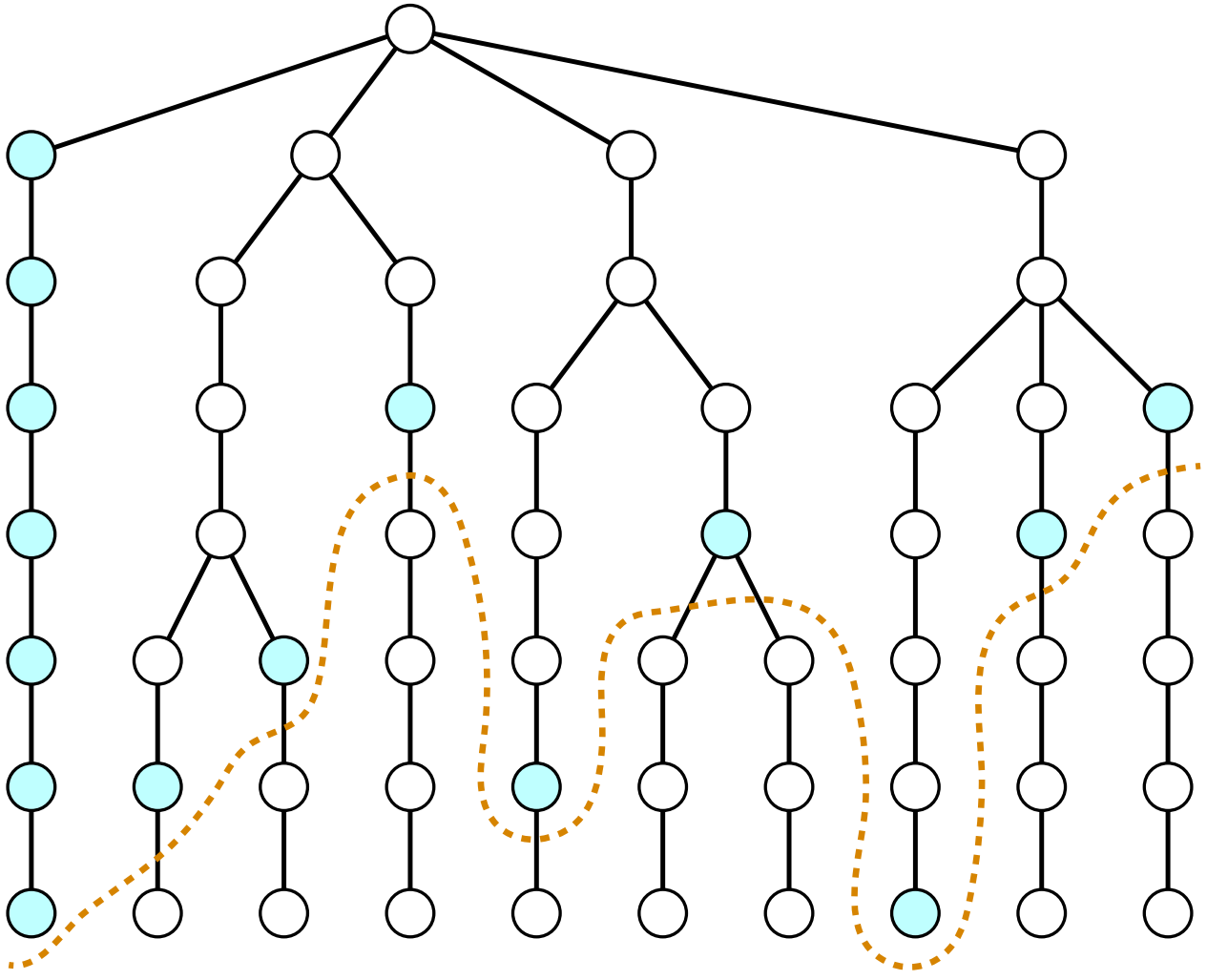
Since the network is connected at every round, we can make a new guess from one of the nodes in the island.

Propagation of guesses



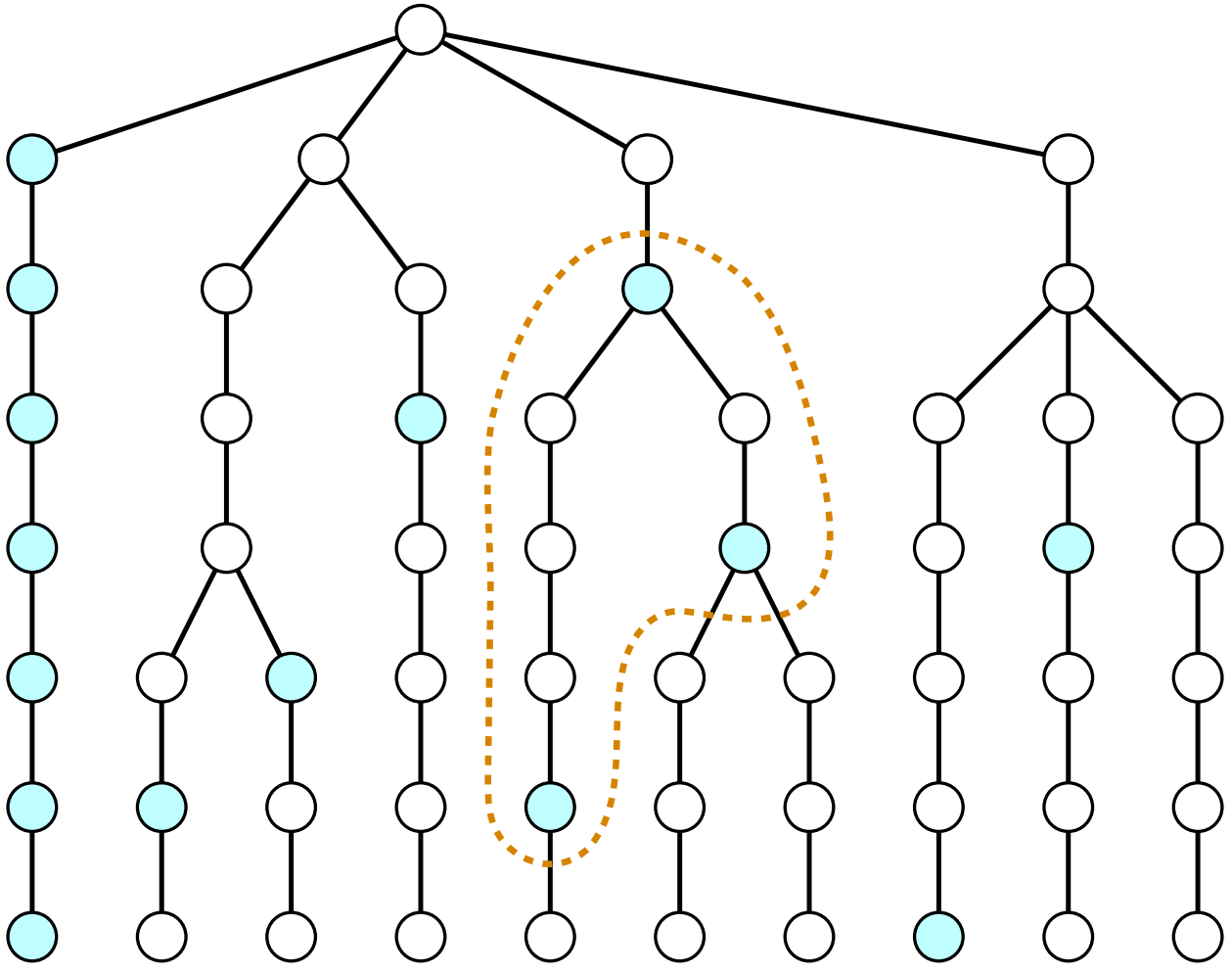
Suppose that there are $n-1$ nodes with necessarily correct guesses (other than the Leader ones). There are two cases:

Propagation of guesses



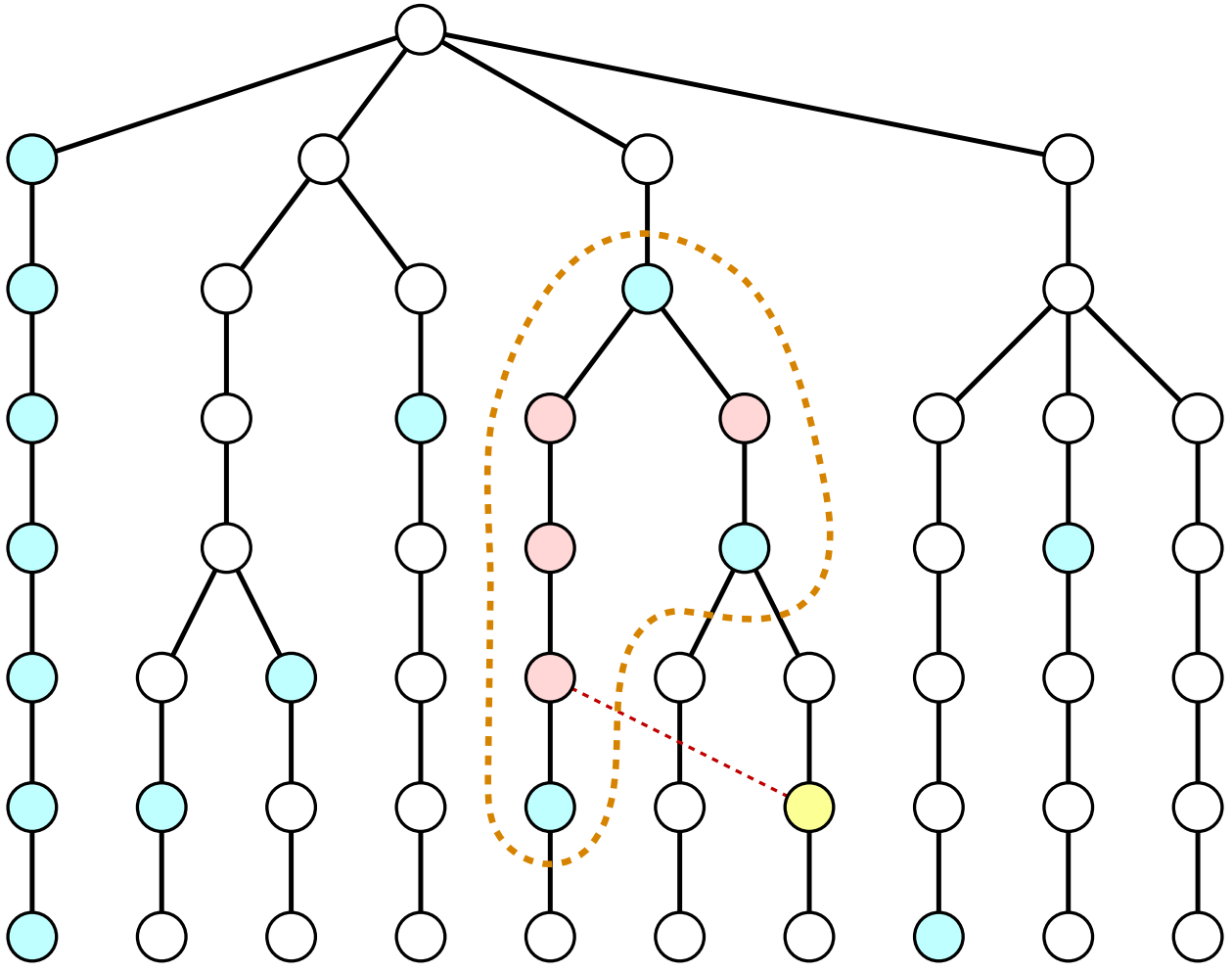
Either these nodes determine a *cut* of the history tree, in which case we have an estimate n' on n, given by their sum...

Propagation of guesses



...Or else, some of these nodes determine an island, which allows us to make a new guess, and so on.

Propagation of guesses



...Or else, some of these nodes determine an island, which allows us to make a new guess, and so on.

Dynamics of new guesses

Summarizing, we have two "buffers":

- A buffer of $n-1$ overestimating guesses (yellow nodes),
- A buffer of $n-2$ necessarily correct guesses (blue nodes).

Once the first buffer is full, every new guess (yellow node) allows us to determine a correct guess (blue node), by the limiting theorem.

Once the second buffer is full, every new correct guess (blue node) either creates a new island (hence we can make a new guess) or creates a cut.

Therefore, within $2n-2$ rounds, the chain of guesses "snowballs" and generates enough guesses to determine a *cut* of the history tree, which in turn yields an *estimate* $n' \leq n$.

Propagation of guesses

Let us make the previous argument more precise...

Lemma

If all guesses are on different levels, and n nodes have a guess, then there is a heavy node.

Proof. We will prove that there is a node v with a guess such that $w(v) > a(v)$. Hence, there is a heavy node by the limiting theorem.

All n guesses are in the non-leader subtree, whose total anonymity is $n - 1$. By induction and the basic properties of history trees, it is easy to prove that there is a node v with $w(v) > a(v)$. □

Propagation of guesses

A level is *bad* if it is entirely contained in the view, it has no nodes with a guess, and making a new guess in this level is impossible.
A node with a necessarily correct guess whose children also have necessarily correct guesses is called a *guesser*.

Lemma

If there are $n-1$ bad levels, then the nodes with a necessarily correct guess form a cut or an island.

Proof. Consider a bad level L_i. If all nodes in L_{i-1} are guessers, they constitute a cut: impossible.
So, there is a guesser in L_{i-1} that is connected to a child $v \in L_i$ of a non-guesser $v' \in L_i$.
Since L_i is a bad level, v has a necessarily correct guess.
Since v' is not a guesser, either v' does not have a necessarily correct guess or a sibling of v does not have a necessarily correct guess. We say that v is a *bad* node.
So, every bad level has a bad node, and thus there are at least $n-1$ bad nodes.

Propagation of guesses

Lemma

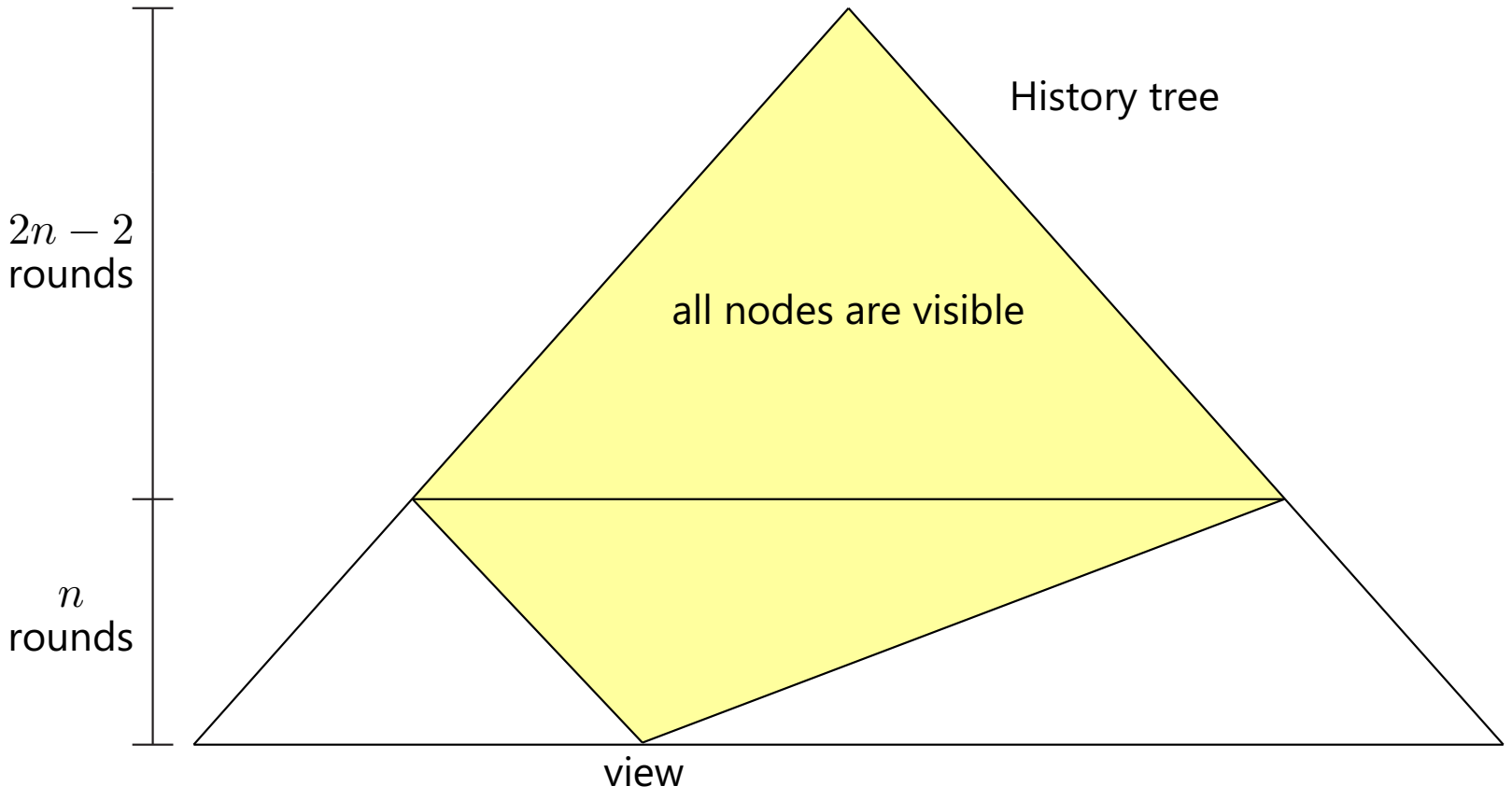
If there are $n-1$ bad levels, then the nodes with a necessarily correct guess form a cut or an island.

Proof (continued). Let v be a deepest bad node, and consider the path P connecting v to the root r.
Let u be the deepest branching node in P.
The parent of v cannot have a necessarily correct guess (unless $v = u$) or v would not be bad.
All other nodes in P between v and u cannot have necessarily correct guesses, or they would form an island with v.
Cut P just below u. The resulting tree has one less branch and one less bad node.
Proceeding by induction, we conclude that there can be at most $n-2$ bad nodes. □

Termination condition

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.

Worst case

This algorithm terminates in at most $3n - 2$ rounds.
This is an example where it terminates in $3n - 3$ rounds: