Congested Anonymous Dynamic Networks

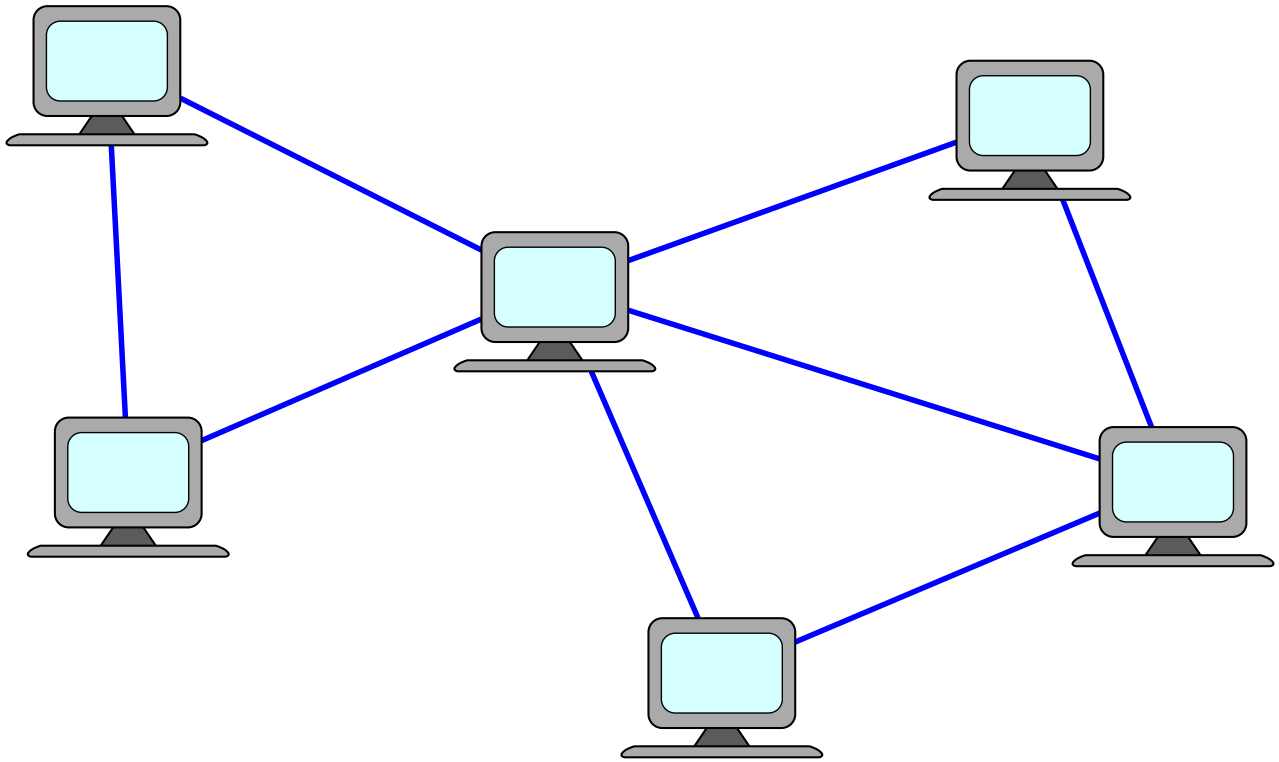
Giovanni Viglietta

Joint work with Giuseppe A. Di Luna

JAIST – November 17, 2022

Static networks

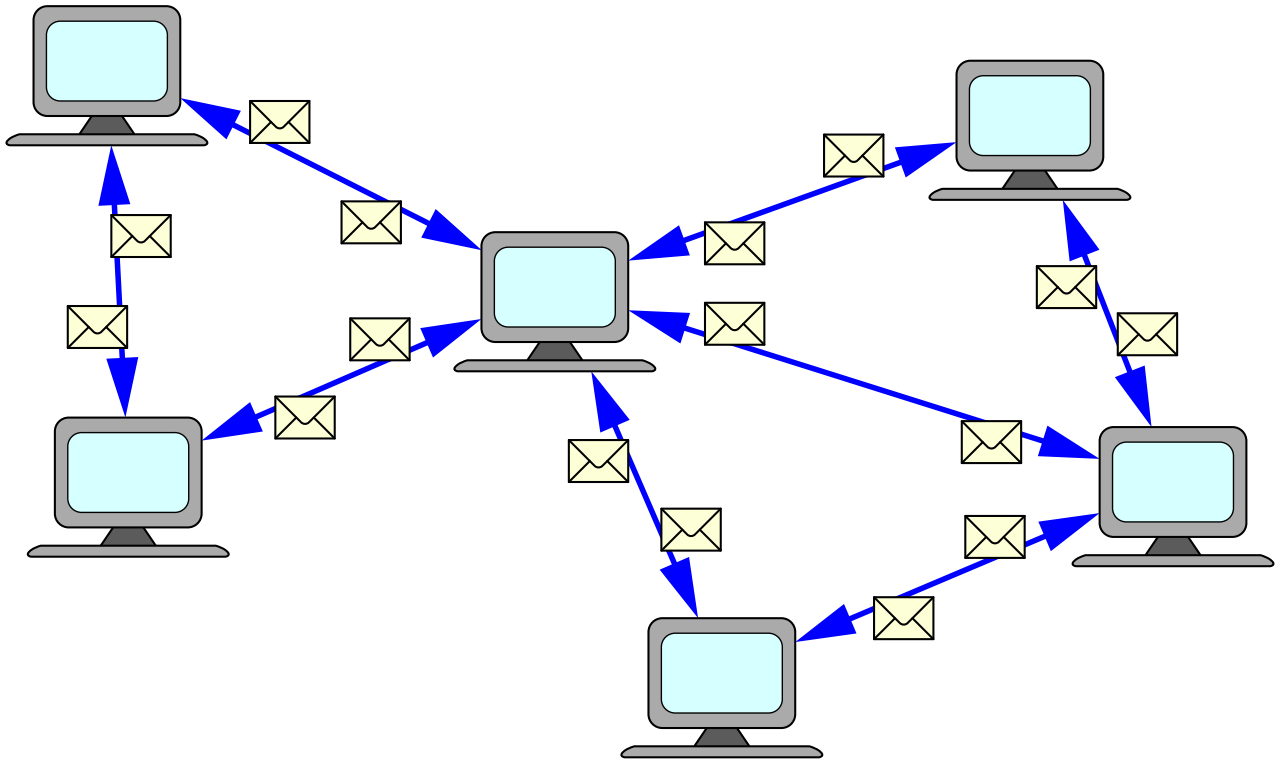
In a *static network*, some machines (or processes) are connected with each other through permanent links.



At each time unit, all machines send messages to their neighbors and do some local deterministic computation.

Static networks

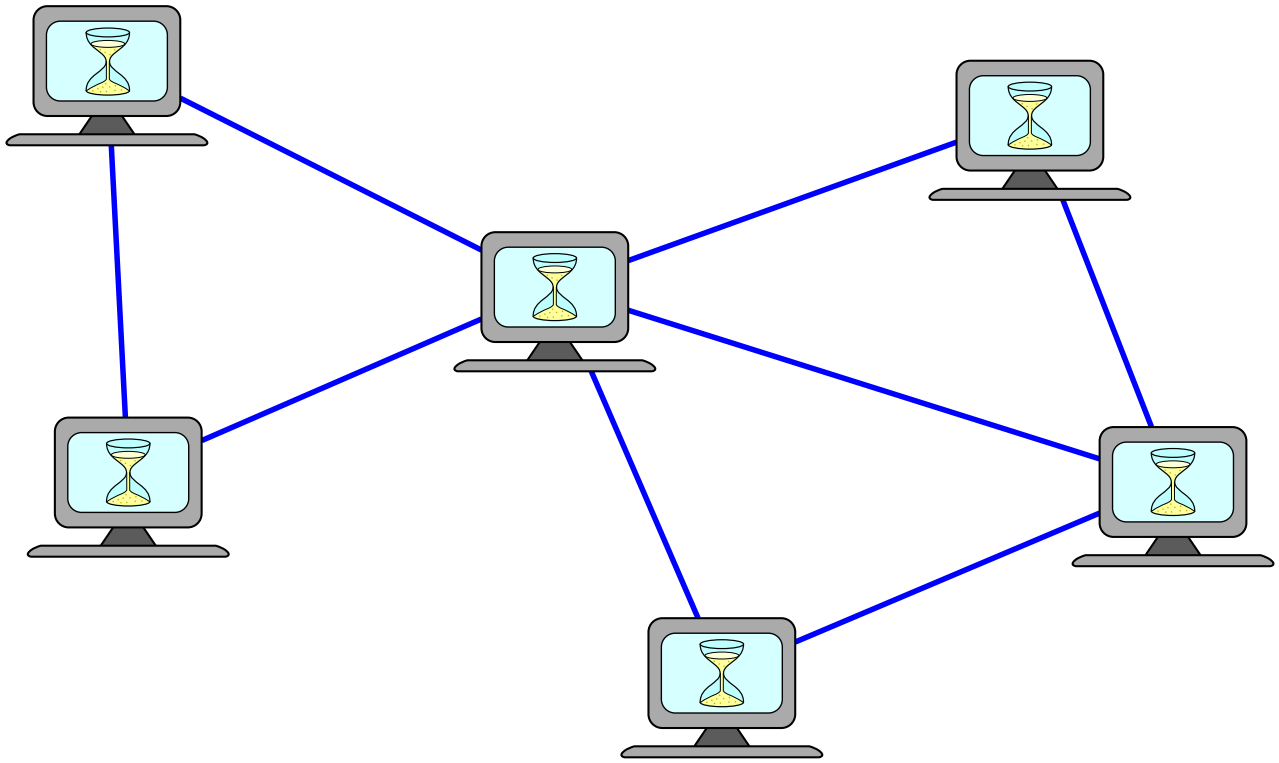
In a *static network*, some machines (or processes) are connected with each other through permanent links.



At each time unit, all machines send messages to their neighbors and do some local deterministic computation.

Static networks

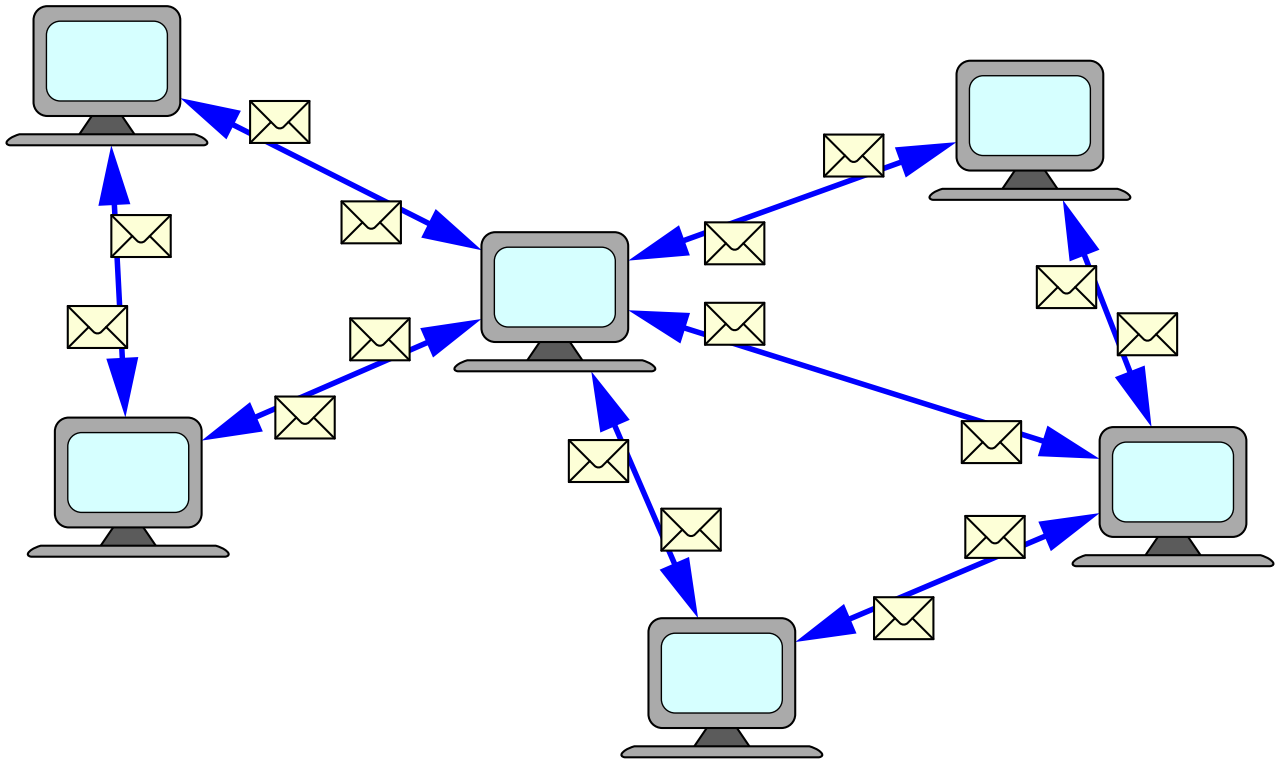
In a *static network*, some machines (or processes) are connected with each other through permanent links.



At each time unit, all machines send messages to their neighbors and do some local deterministic computation.

Static networks

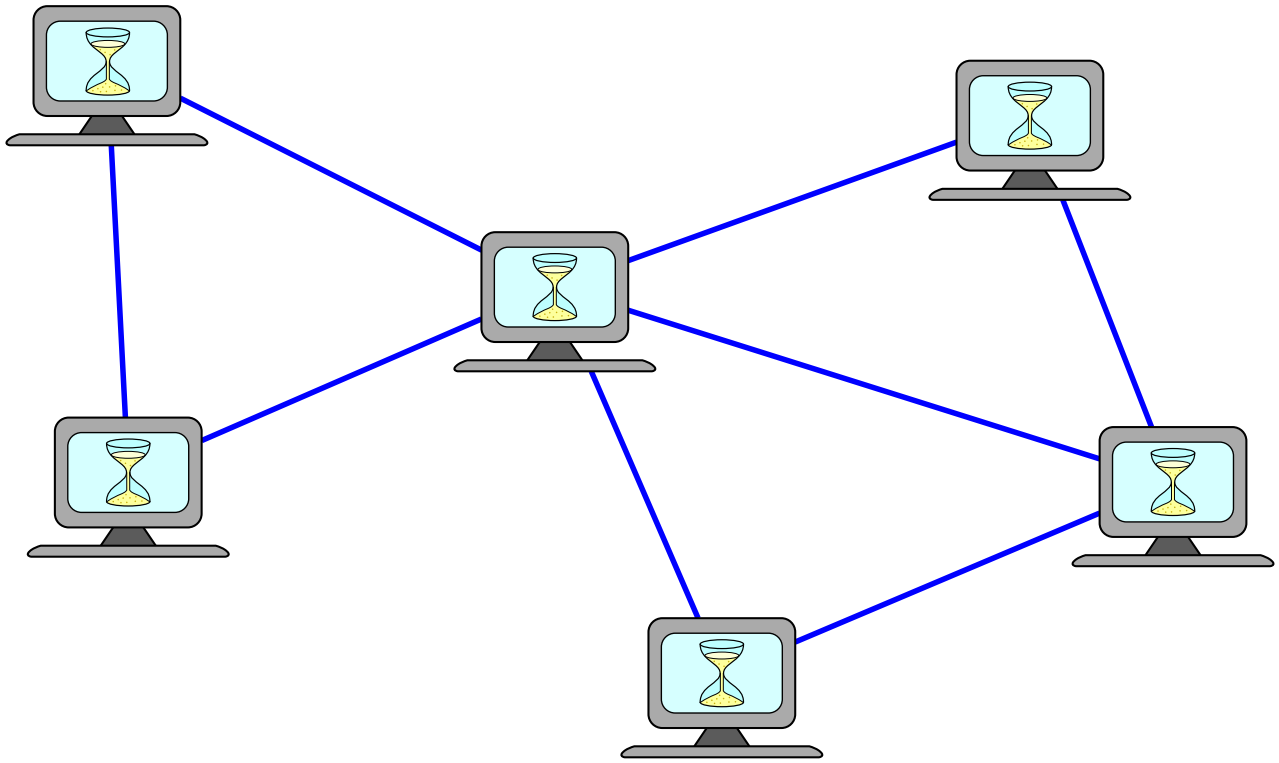
In a *static network*, some machines (or processes) are connected with each other through permanent links.



At each time unit, all machines send messages to their neighbors and do some local deterministic computation.

Static networks

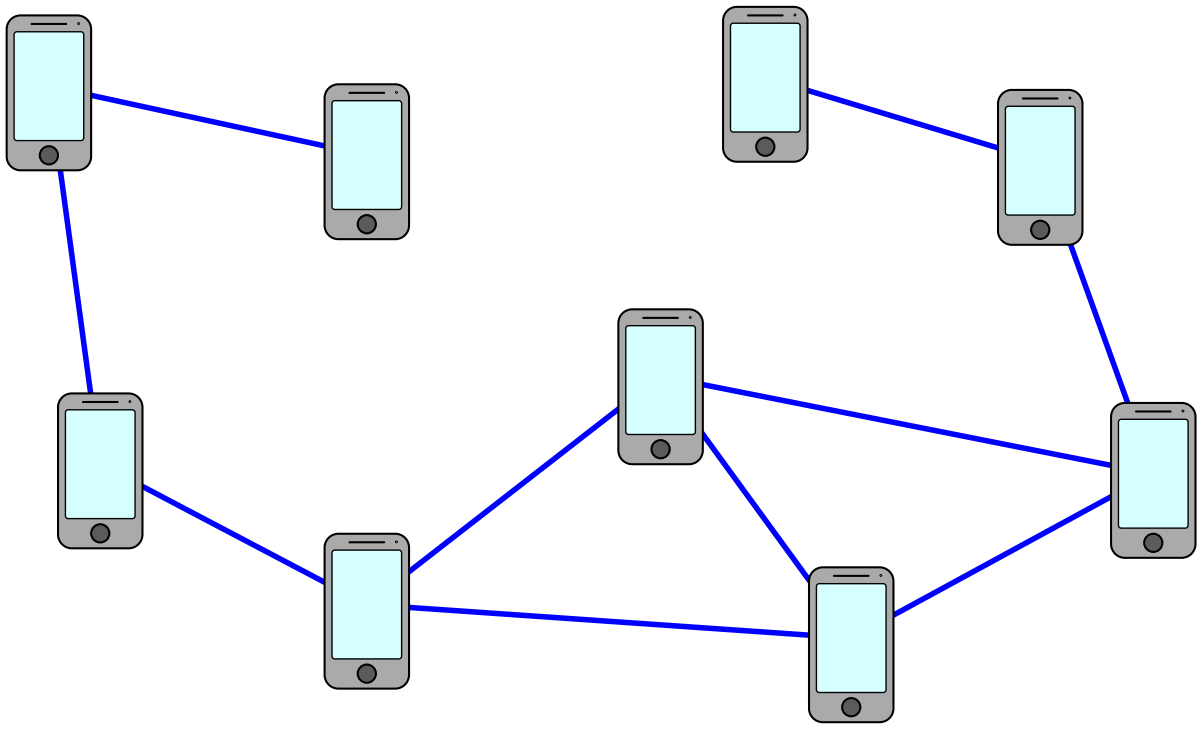
In a *static network*, some machines (or processes) are connected with each other through permanent links.



At each time unit, all machines send messages to their neighbors and do some local deterministic computation.

Dynamic networks

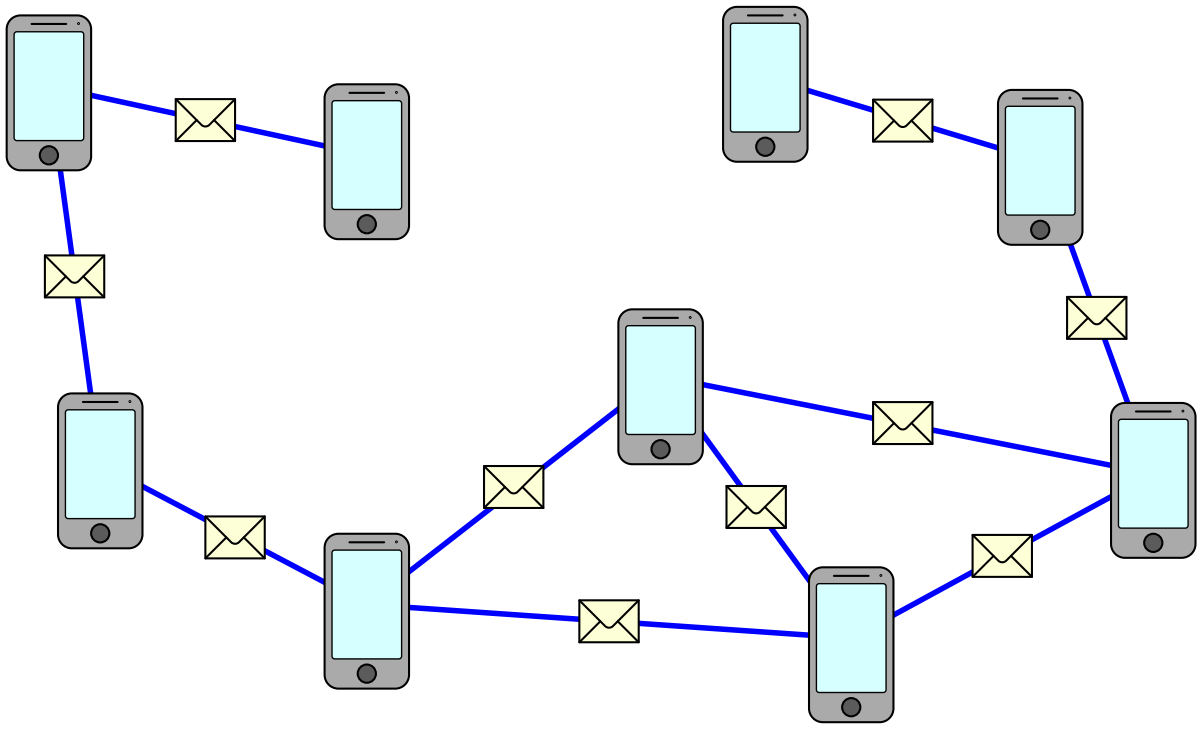
In a *dynamic network*, some machines (or agents) are connected with one another through links that may change over time.



Assume that, at every *round*, the links form a connected graph. What can be computed by this network, and in how many rounds?

Dynamic networks

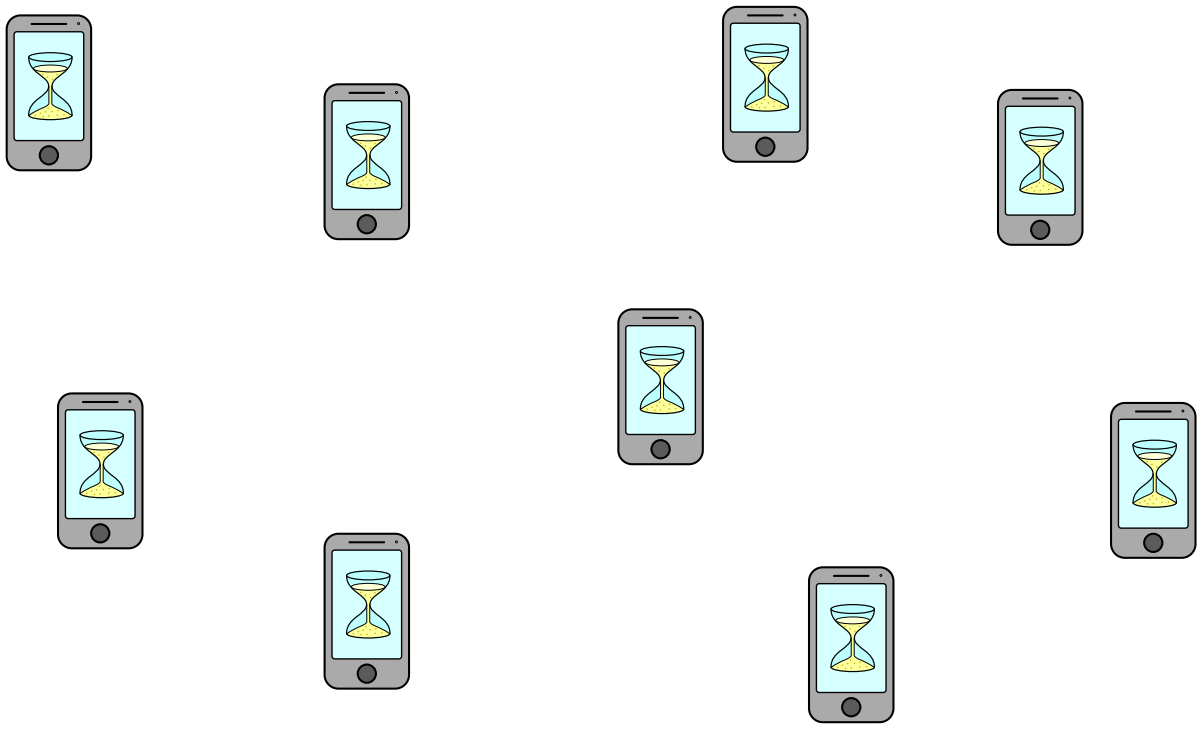
In a *dynamic network*, some machines (or agents) are connected with one another through links that may change over time.



Assume that, at every *round*, the links form a connected graph. What can be computed by this network, and in how many rounds?

Dynamic networks

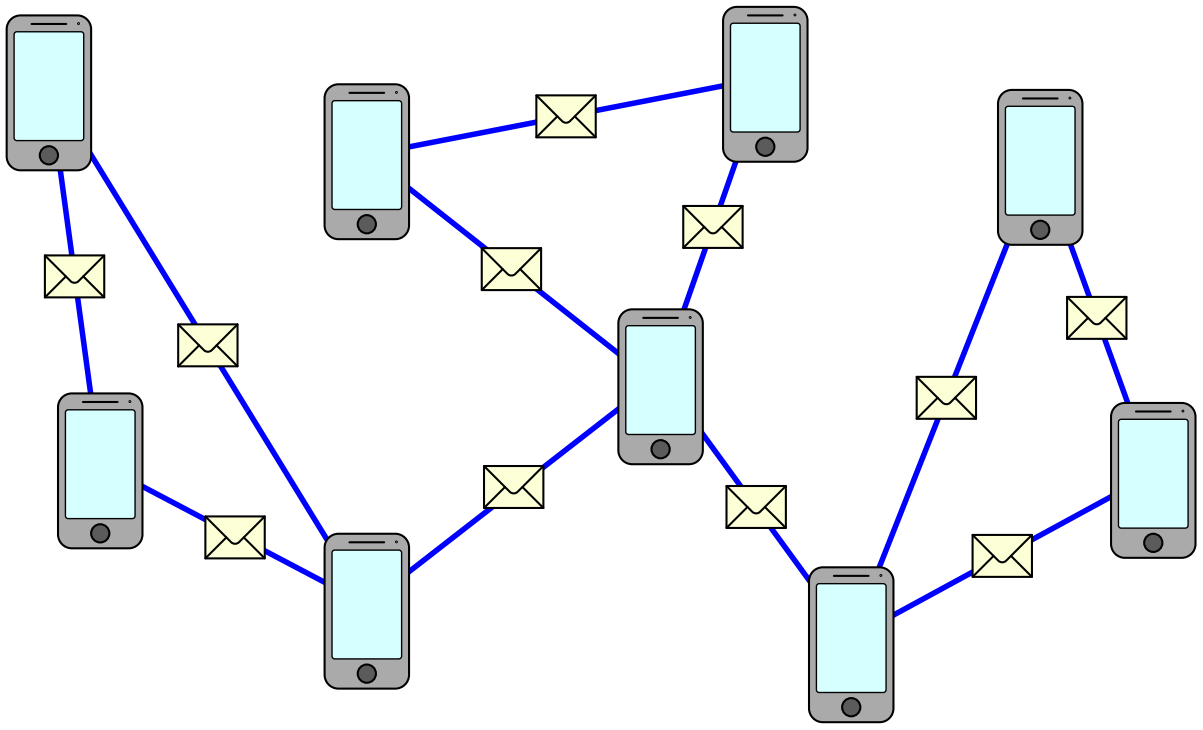
In a *dynamic network*, some machines (or agents) are connected with one another through links that may change over time.



Assume that, at every *round*, the links form a connected graph. What can be computed by this network, and in how many rounds?

Dynamic networks

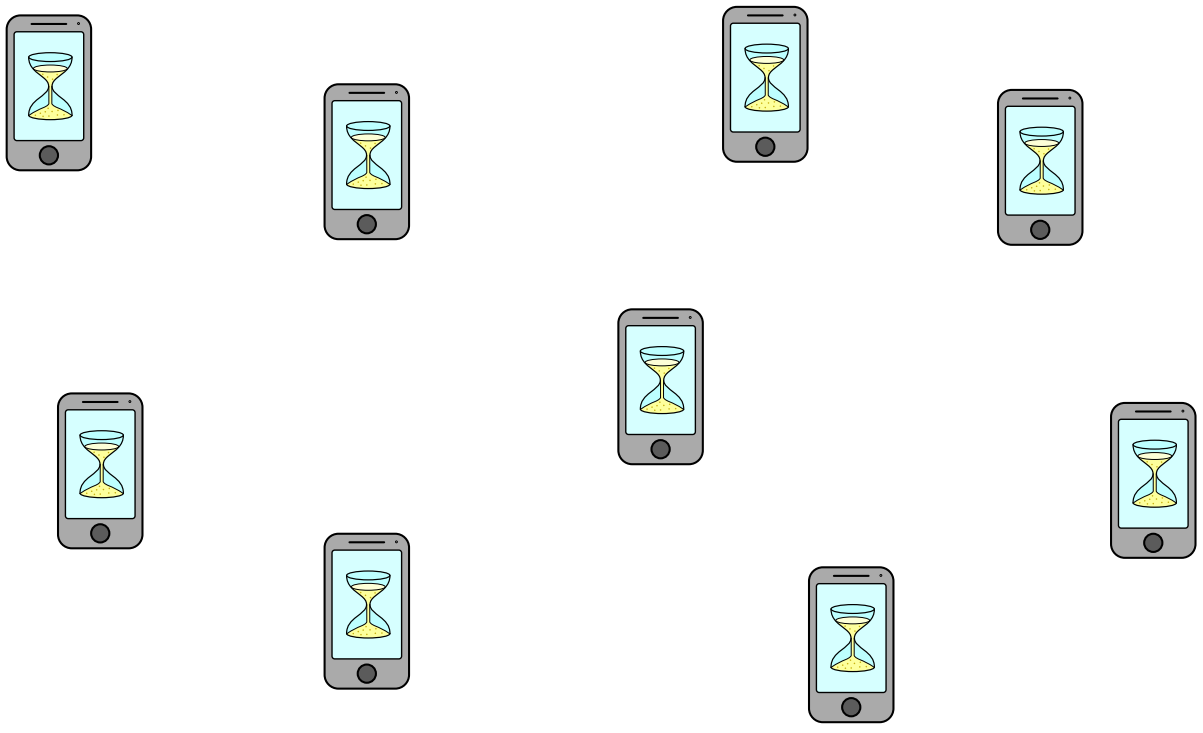
In a *dynamic network*, some machines (or agents) are connected with one another through links that may change over time.



Assume that, at every *round*, the links form a connected graph. What can be computed by this network, and in how many rounds?

Dynamic networks

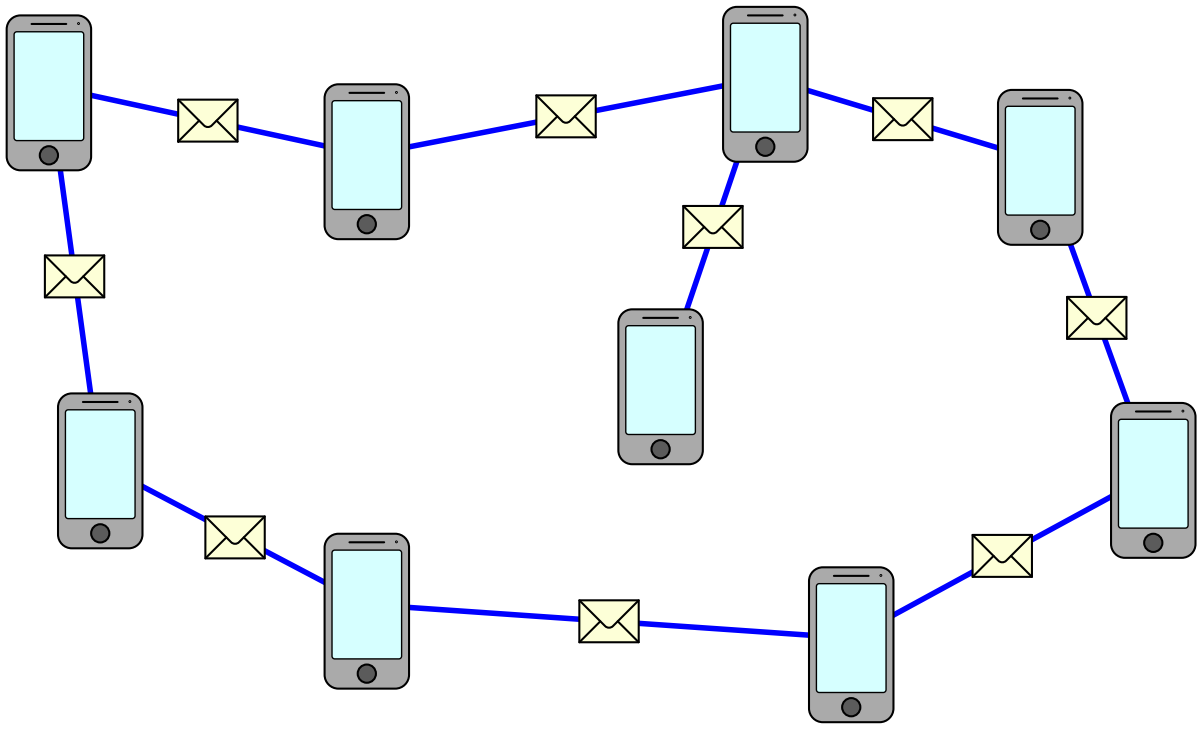
In a *dynamic network*, some machines (or agents) are connected with one another through links that may change over time.



Assume that, at every *round*, the links form a connected graph. What can be computed by this network, and in how many rounds?

Dynamic networks

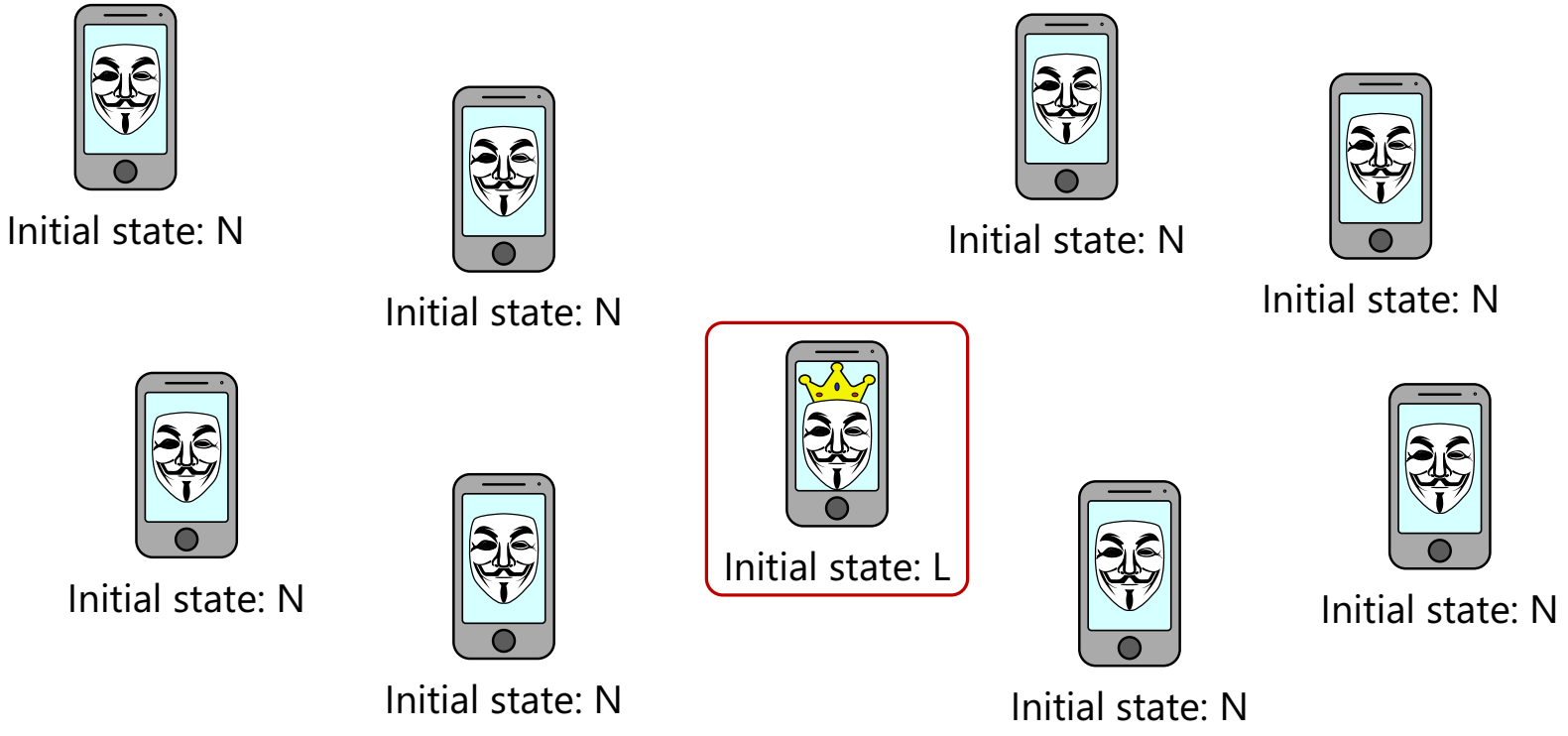
In a *dynamic network*, some machines (or agents) are connected with one another through links that may change over time.



Assume that, at every *round*, the links form a connected graph. What can be computed by this network, and in how many rounds?

Counting anonymous agents with a Leader

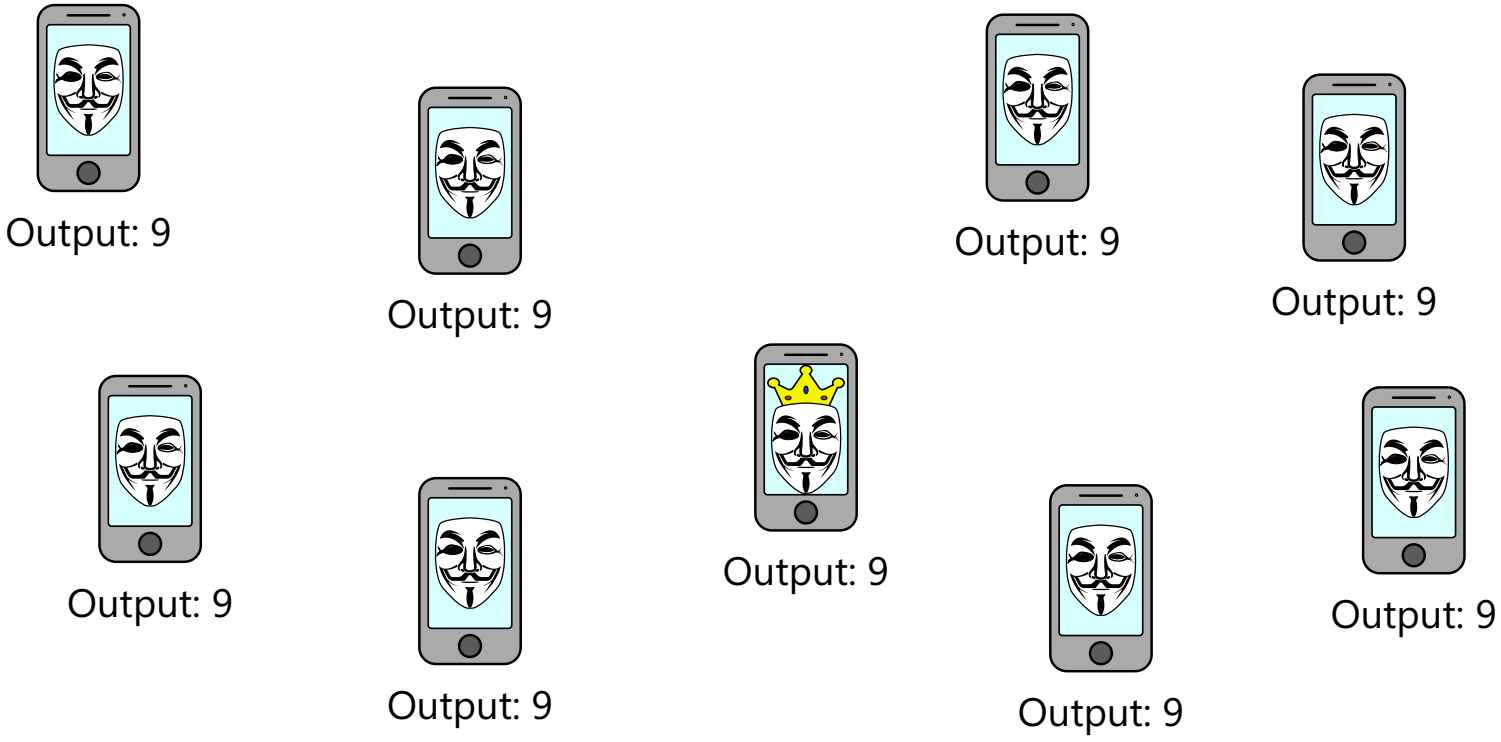
A common assumption is that the dynamic network is *anonymous*, i.e., all agents start in the same state, except one: the *Leader*.



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

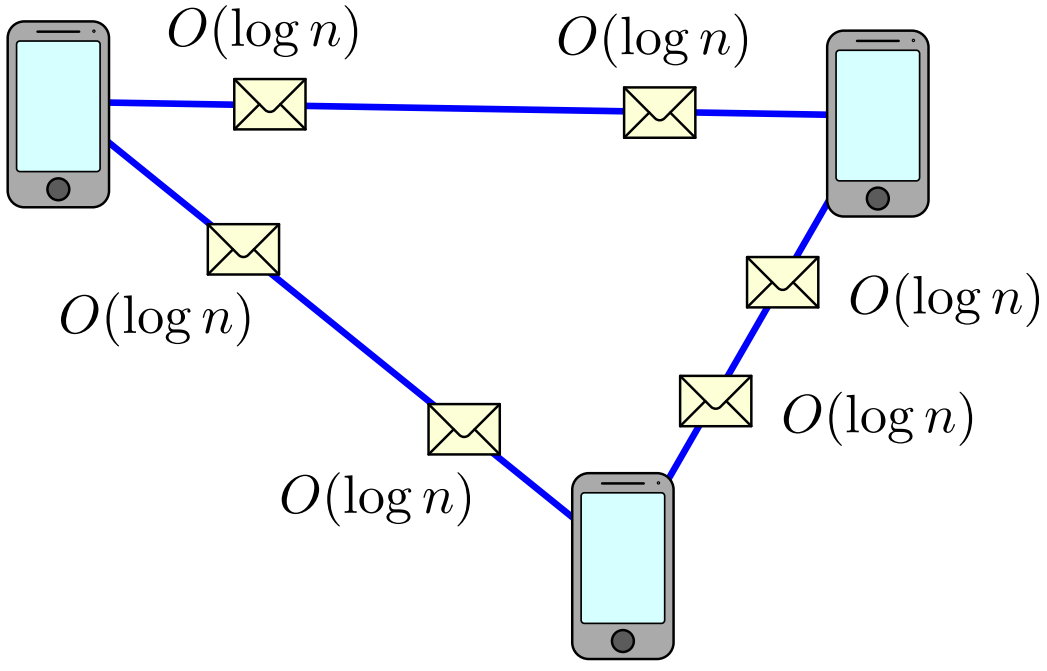
A common assumption is that the dynamic network is *anonymous*, i.e., all agents start in the same state, except one: the *Leader*.



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.)

Congested networks

Normally, there is no limit to the size of messages that can be sent through the 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.

Selected literature

Solving the Counting problem in non-congested dynamic networks:

- **Michail et al.:** Looks impossible! (SSS 2013)
- **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)

Solving the Counting problem in congested dynamic networks:

- **Dutta et al.:** Lower bound of $\Omega(n^2/\log n)$ rounds (SODA 2013)
- **Kowalski–Mosteiro:** $\widetilde{O}(n^{5+\epsilon})$ rounds (arXiv 2022)
- **Di Luna–V.:** $O(n^3)$ rounds (Today's talk)

Previous result

We will use the following result:

Theorem (Di Luna–V., FOCS 2022)

In a non-congested anonymous dynamic network with n agents and a Leader, the Counting Problem can be solved in $3n$ rounds.

Additional remarks:

Since we can solve the Counting Problem, we can compute in $3n$ rounds *all* functions that are computable in anonymous networks.

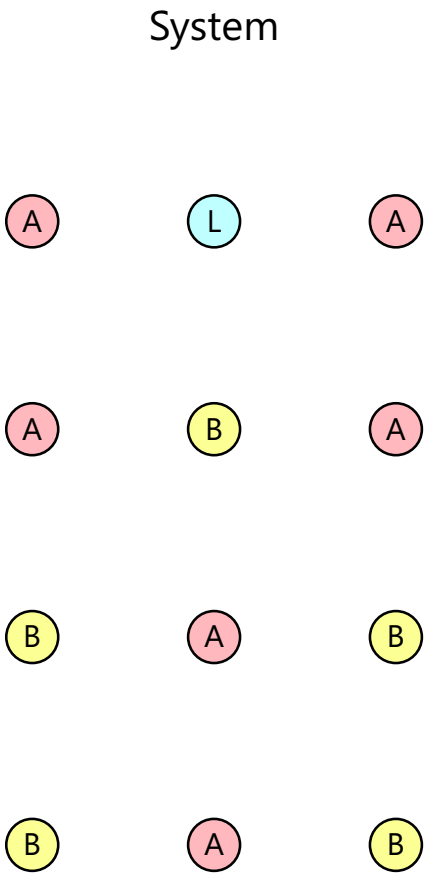
These are precisely the *multi-aggregate* functions f:

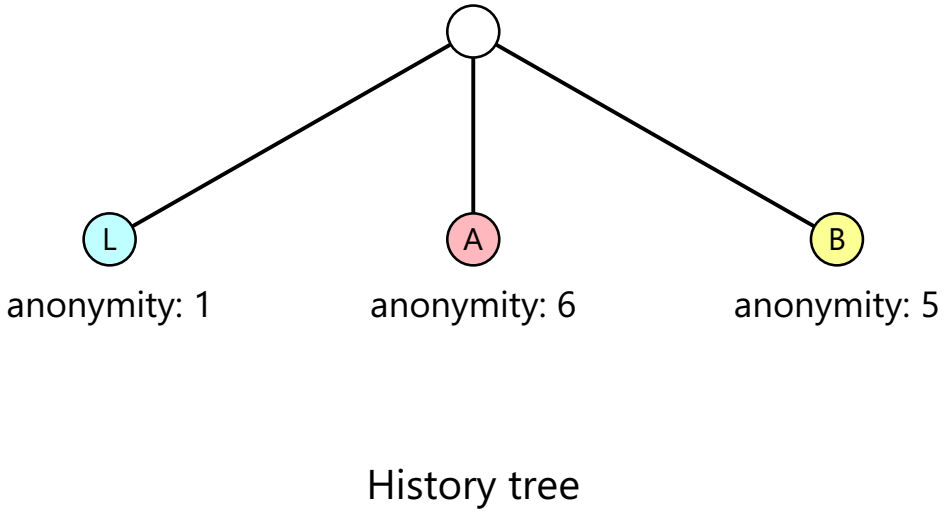
- Agent p outputs $f(x_p, \mu)$,
- where x_p is the input of agent p,
- and μ is the multi-set of all inputs.

Examples include the maximum, minimum, mean, median, mode, variance, and most statistical functions.

History trees

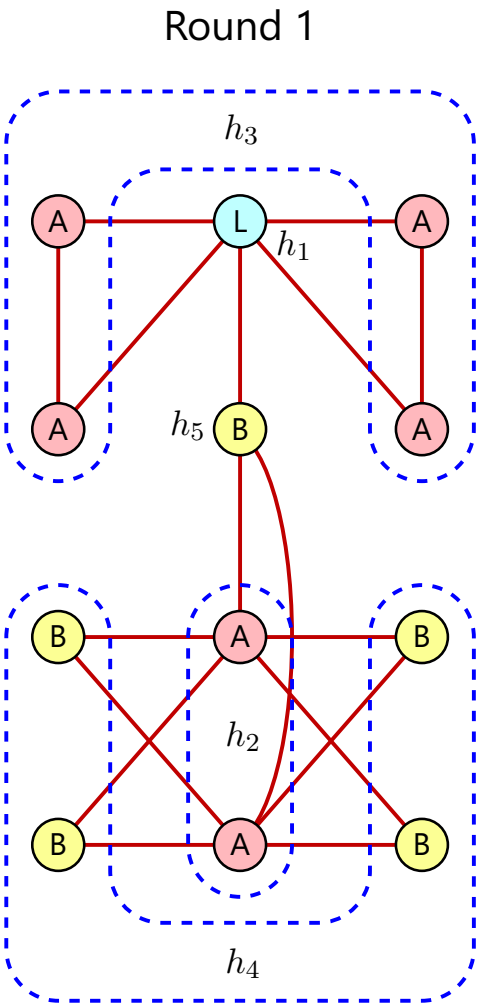
Every network has an associated *History tree* (FOCS 2022).

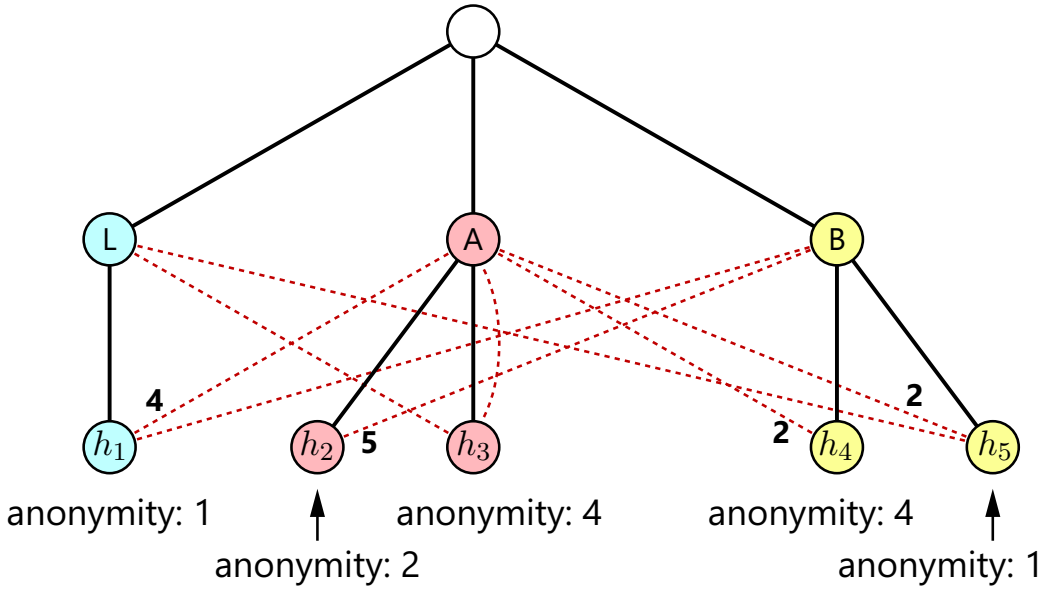




History trees

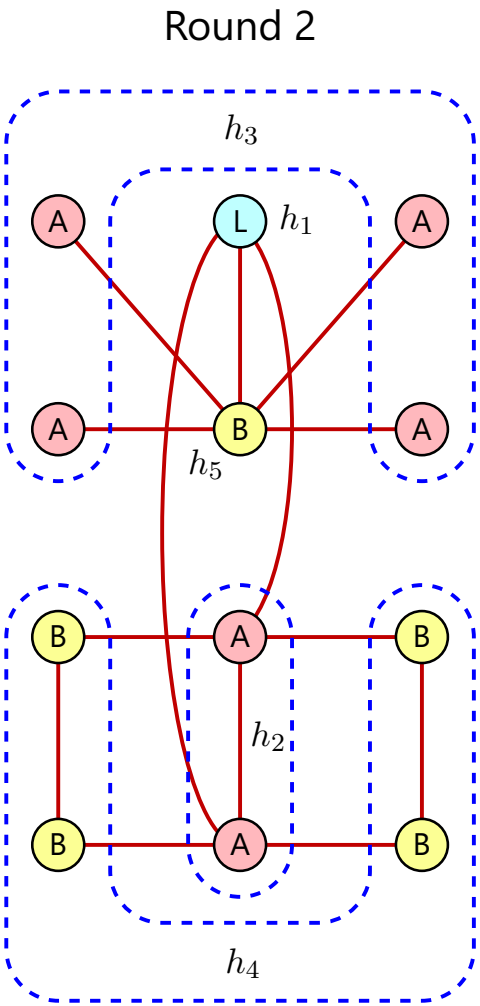
Every network has an associated *History tree* (FOCS 2022).

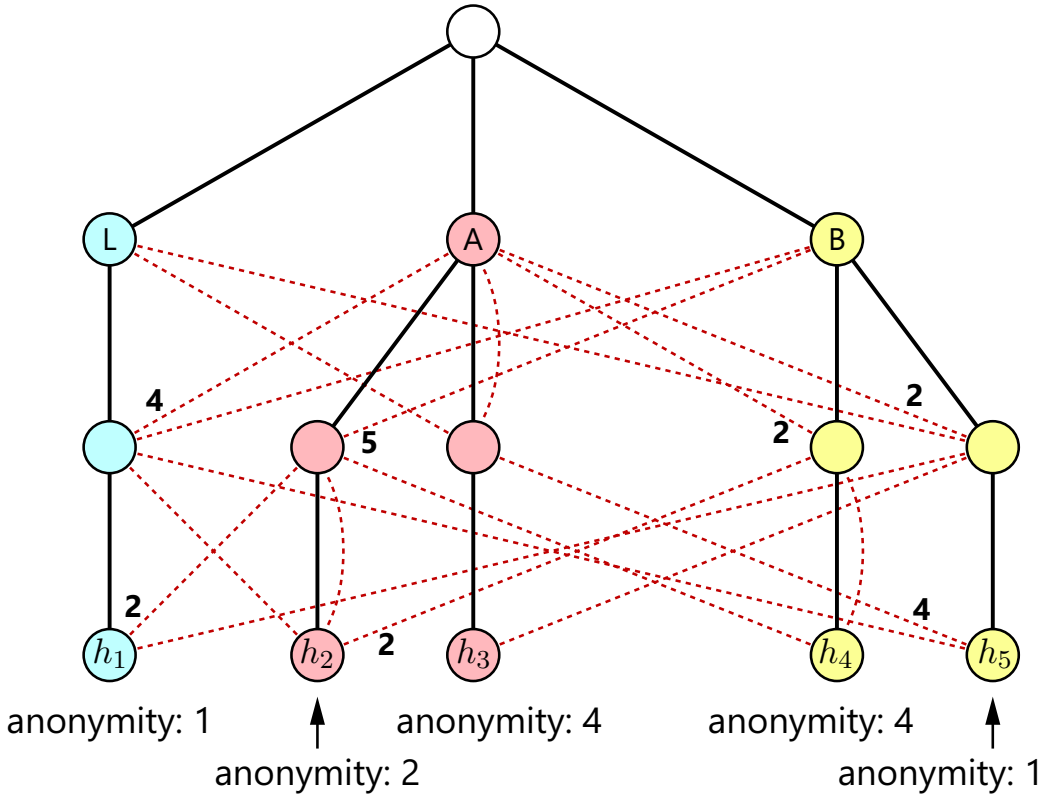




History trees

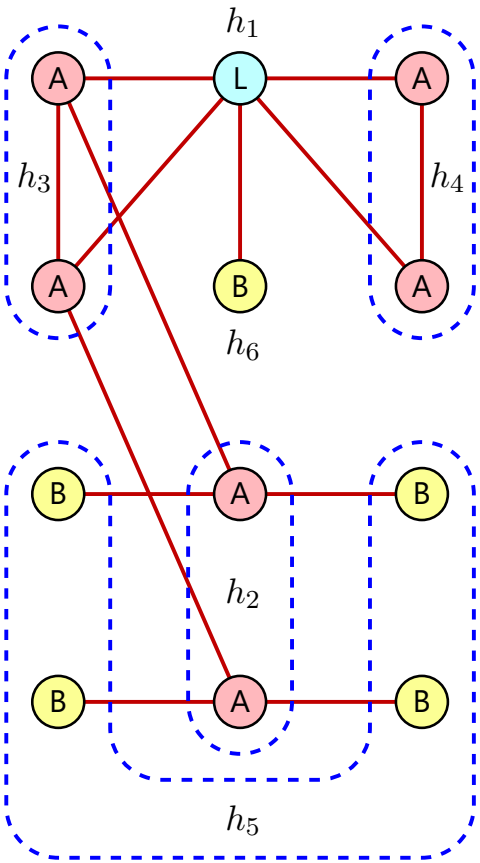
Every network has an associated *History tree* (FOCS 2022).

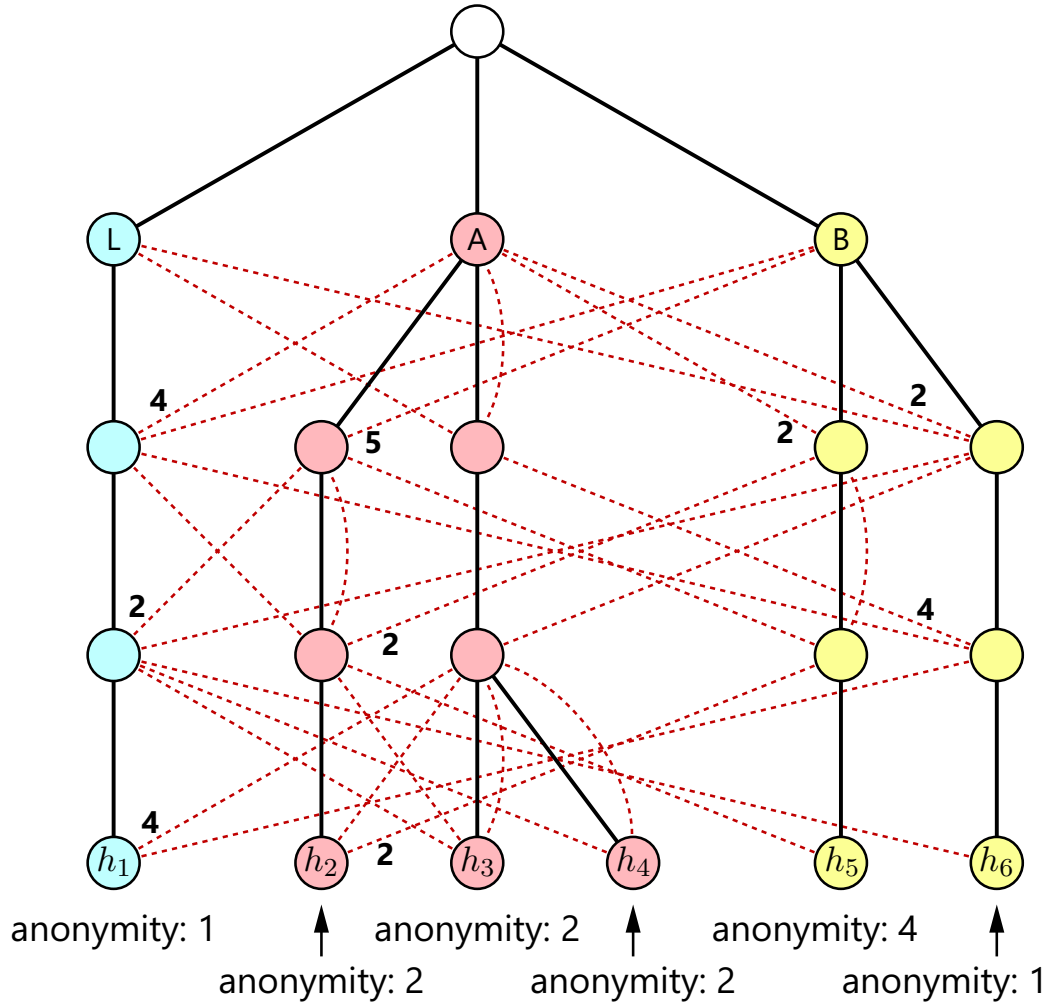




History trees

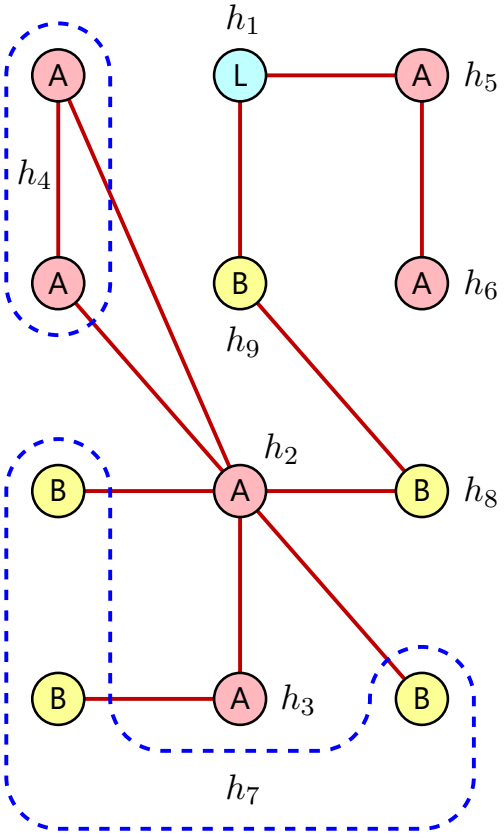
Every network has an associated *History tree* (FOCS 2022).

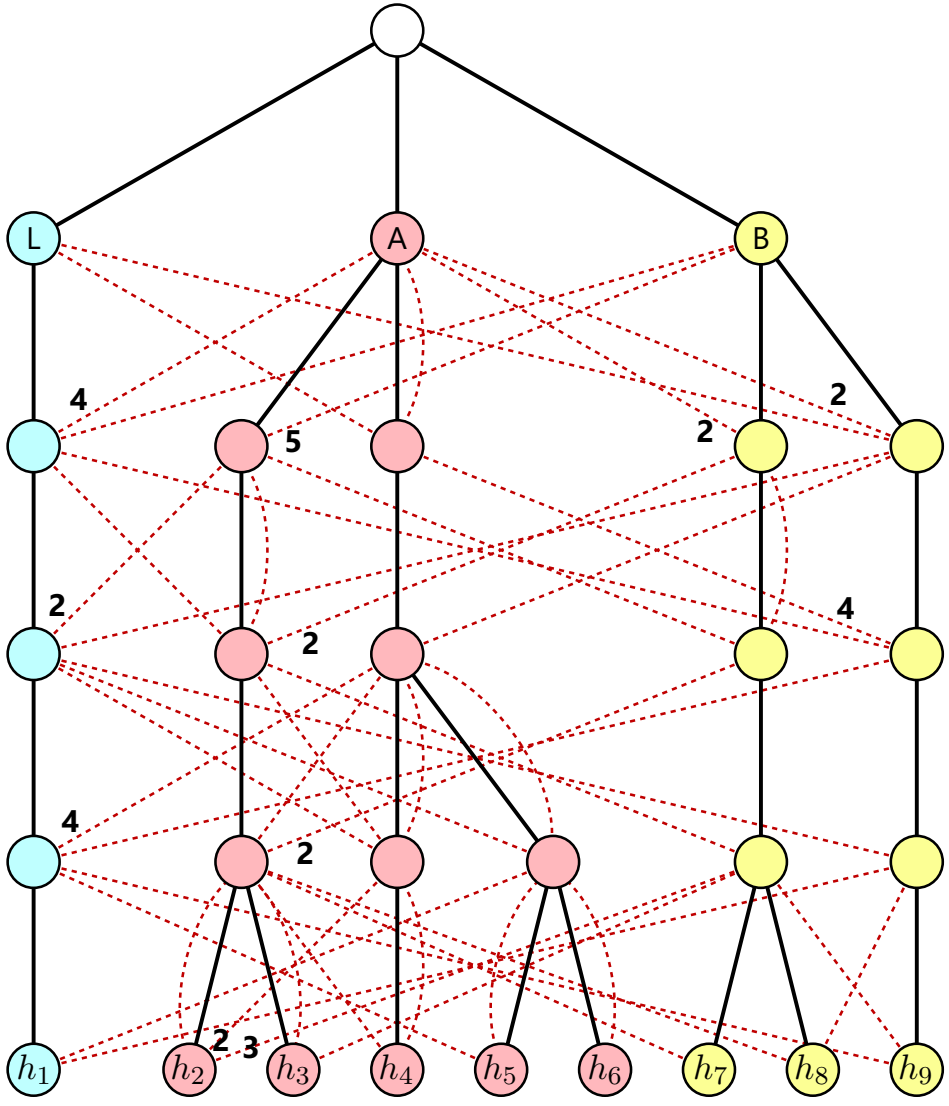




History trees

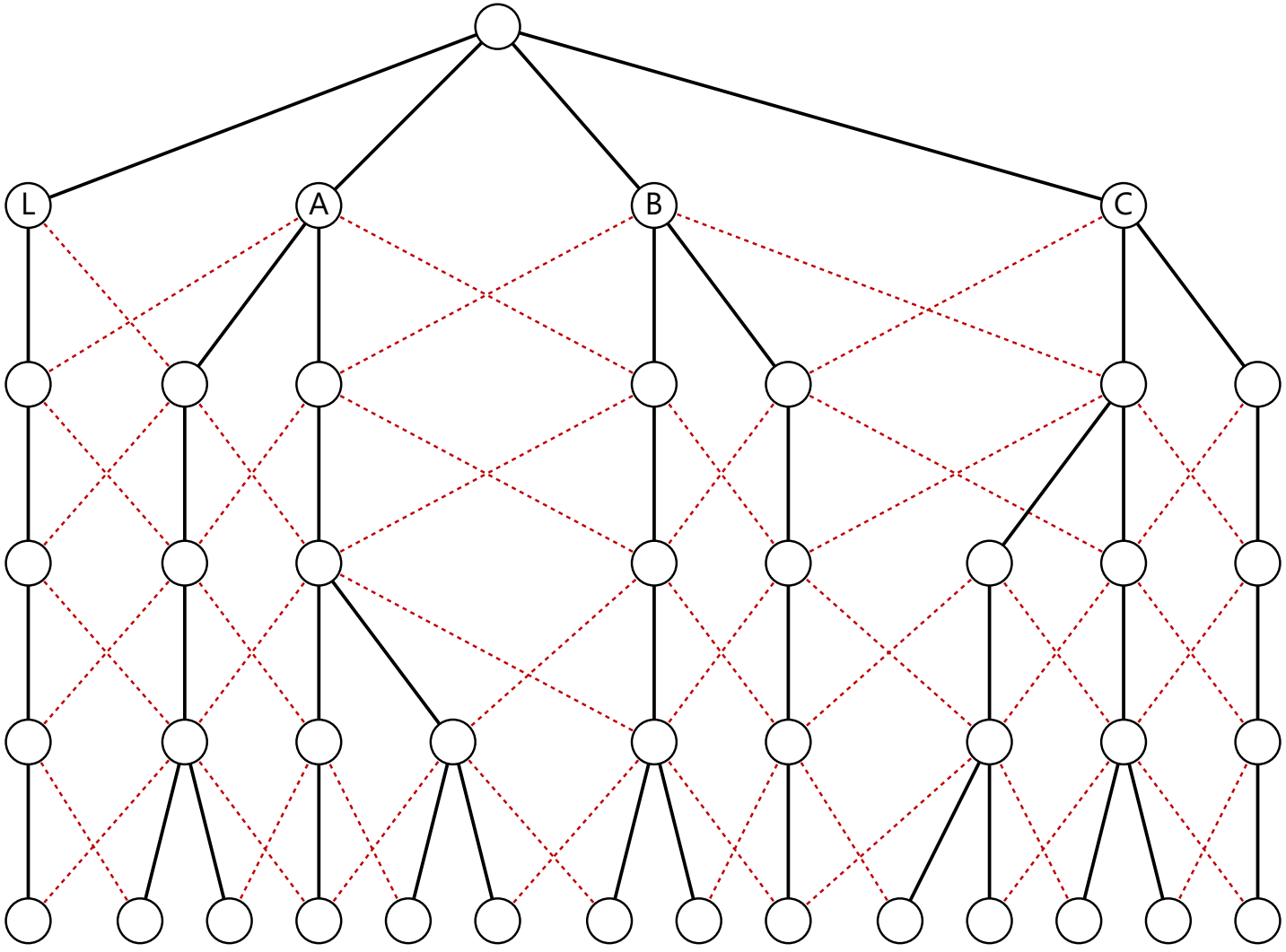
Every network has an associated *History tree* (FOCS 2022).





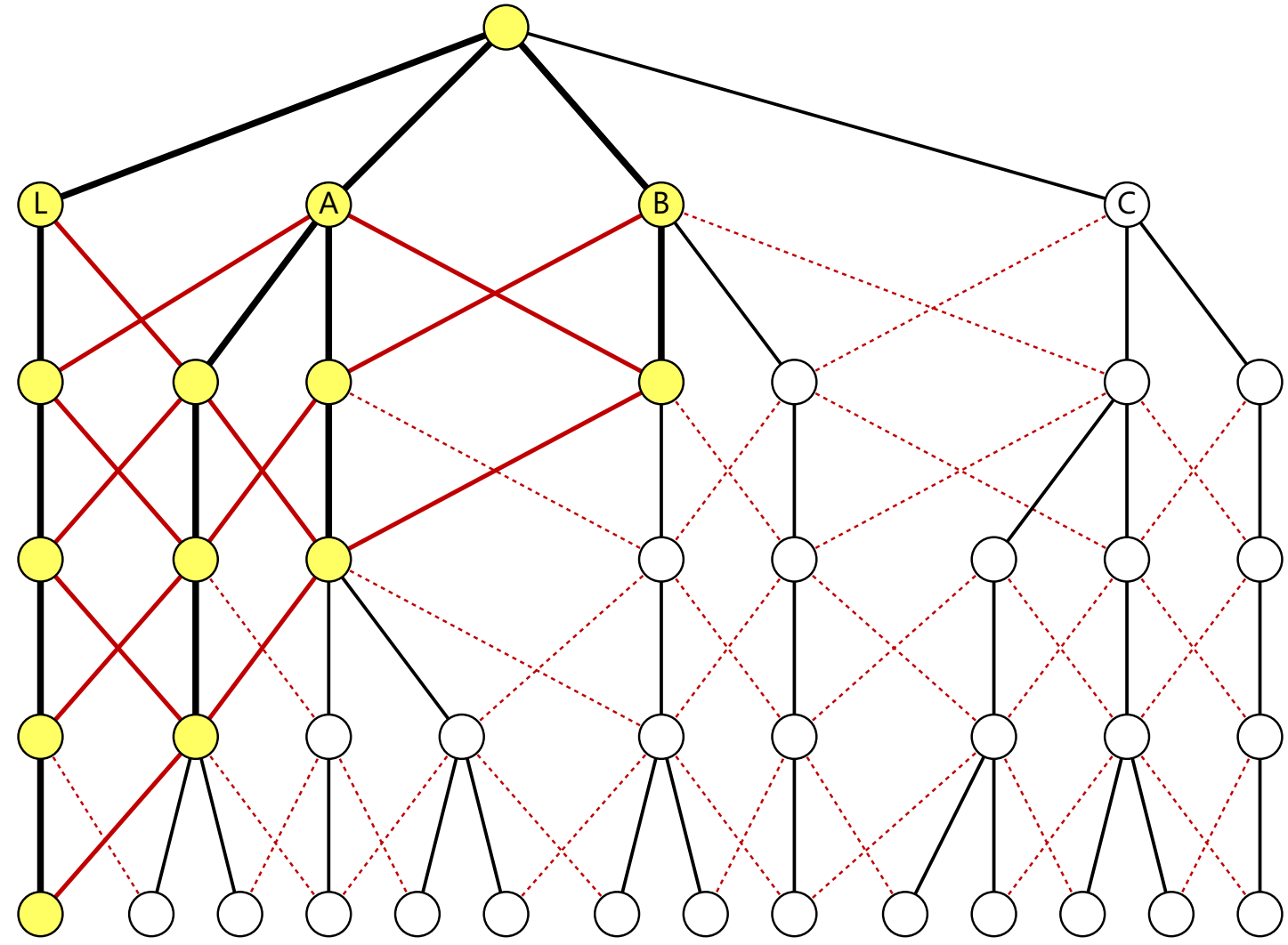
View of a history tree

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



View of a history tree

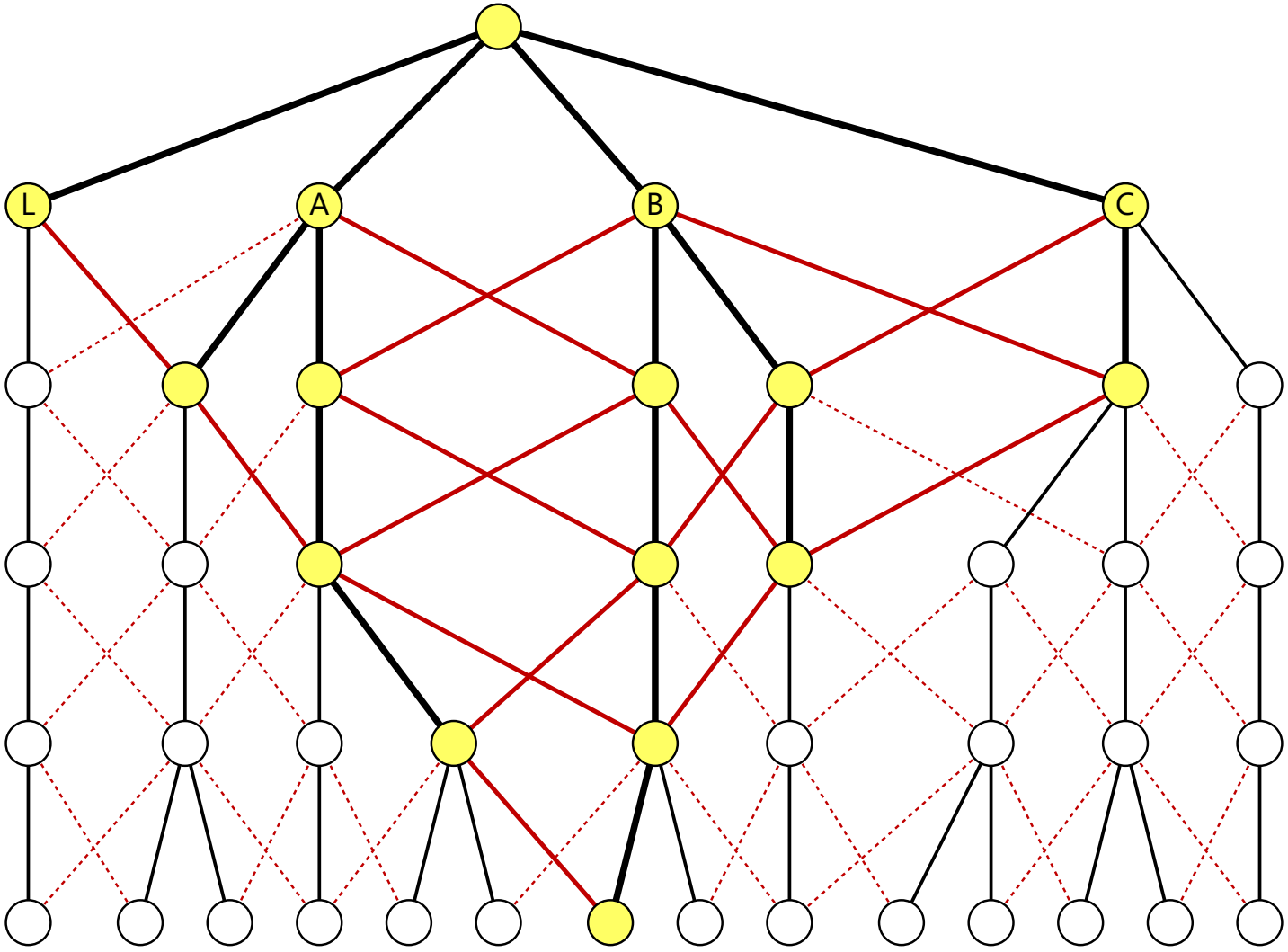
At any point in time, an agent only has a *view* of the history tree.



view

View of a history tree

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

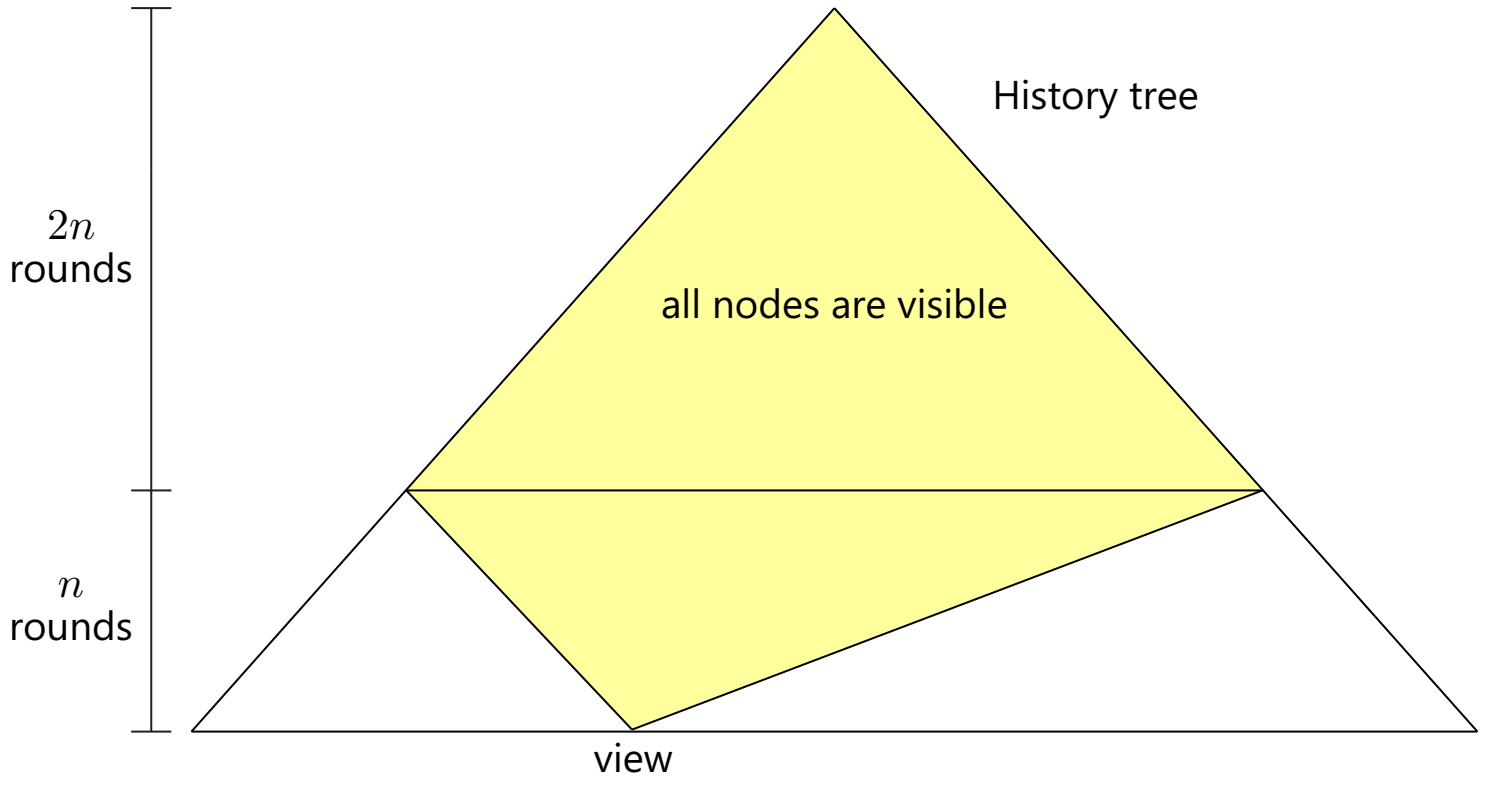


view

Outline of Counting algorithms

Non-congested networks (FOCS 2022):

- 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.

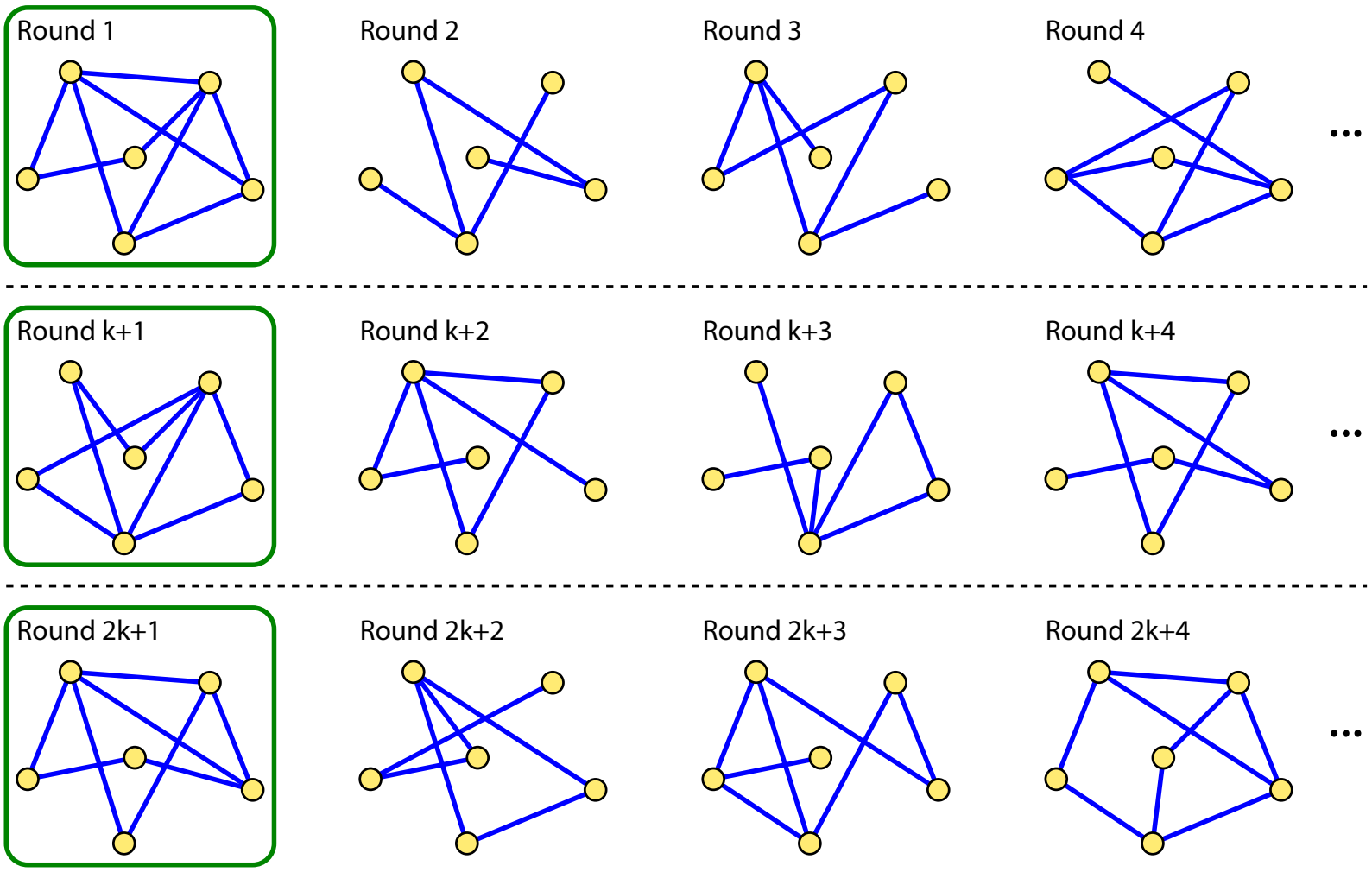


Outline of Counting algorithms

Congested networks (this talk):

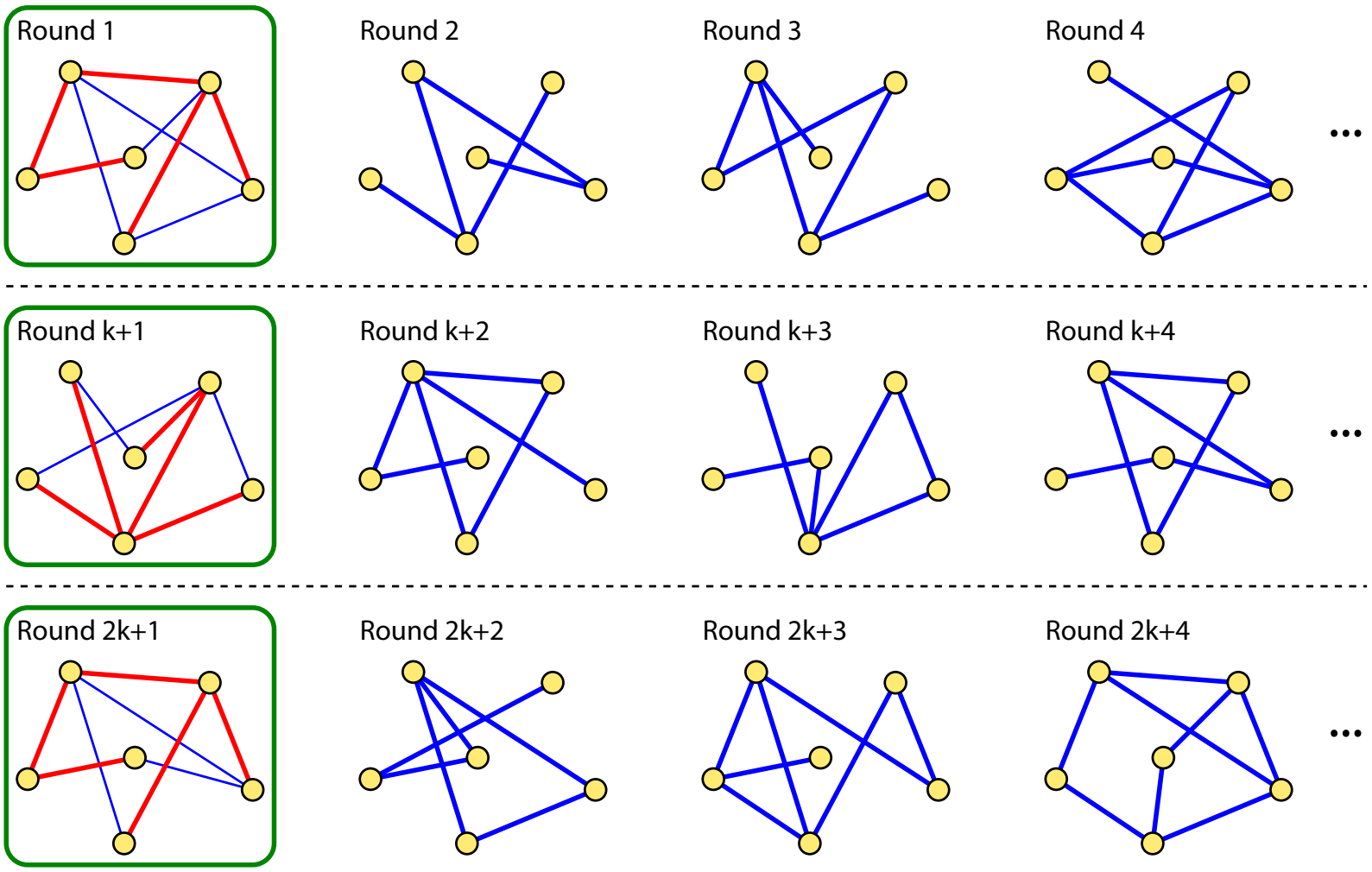
- Construct the history tree of some "artificial" network with n agents (we call it the *official history tree*, OHT).
- Each level of the OHT is constructed from a spanning tree of the real network at some round (so each level takes $O(n \log n)$ bits).
- The construction is done in a series of broadcasts, 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 FOCS 2022 algorithm on the OHT and see if it can determine n.

Construction of the OHT



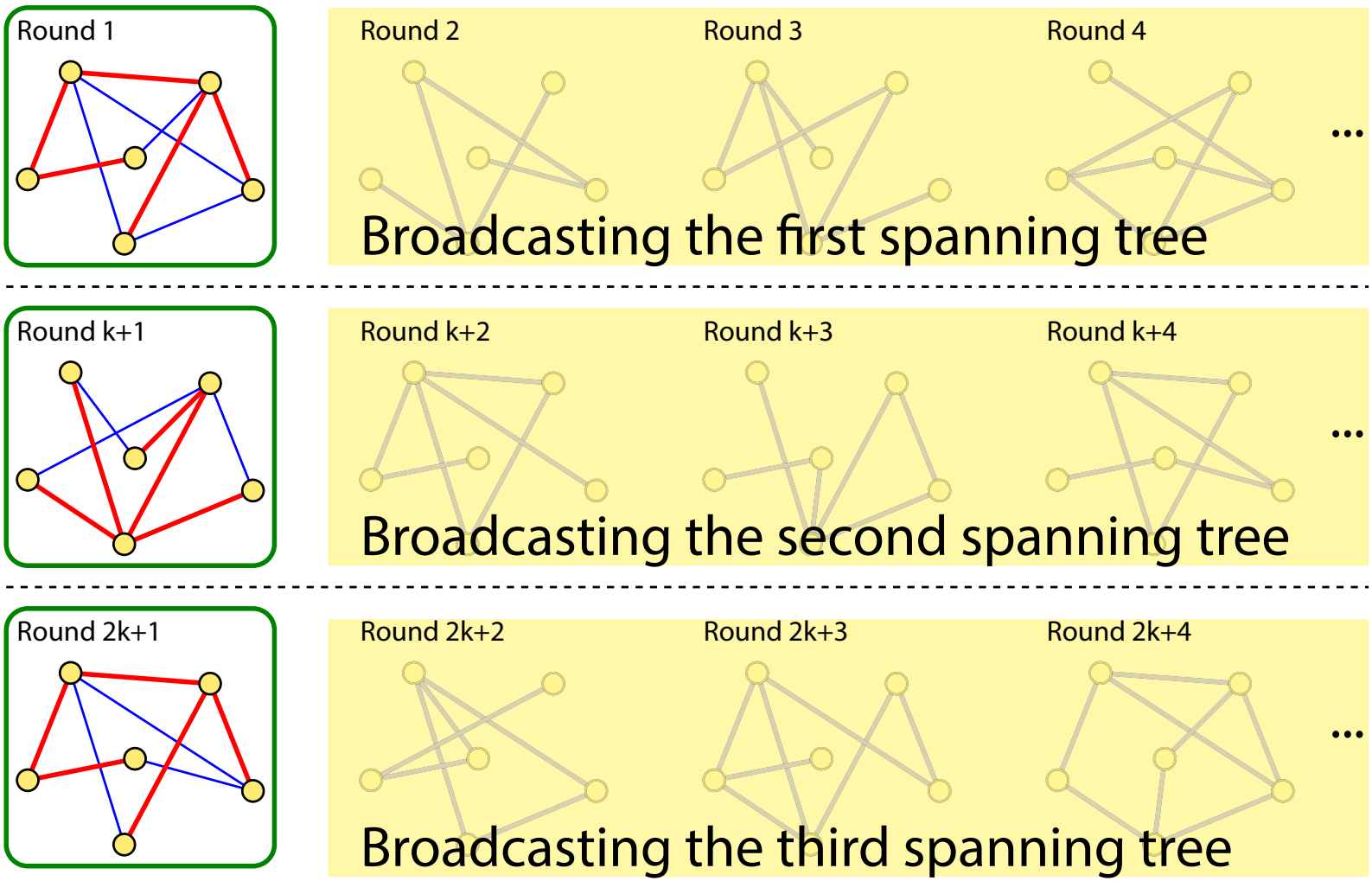
The OHT is constructed on a *subsequence* of selected network rounds.

Construction of the OHT



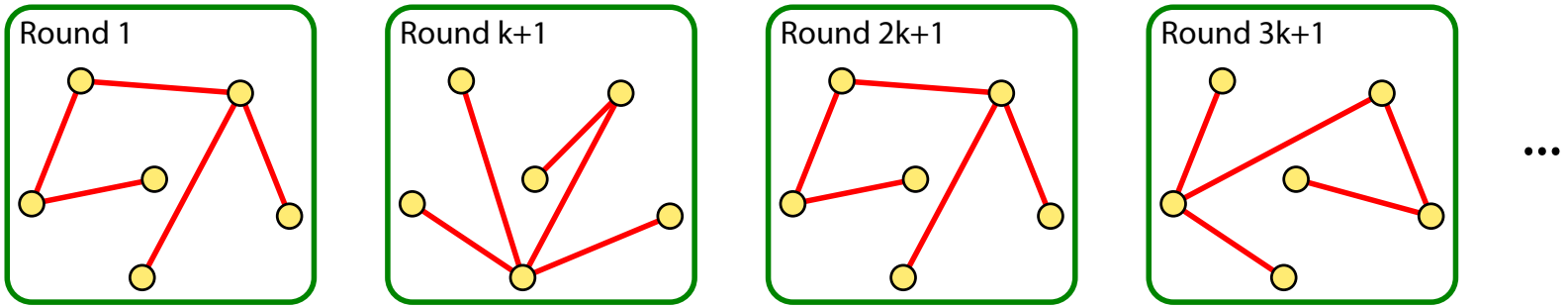
Take a spanning tree of the network at the selected rounds.

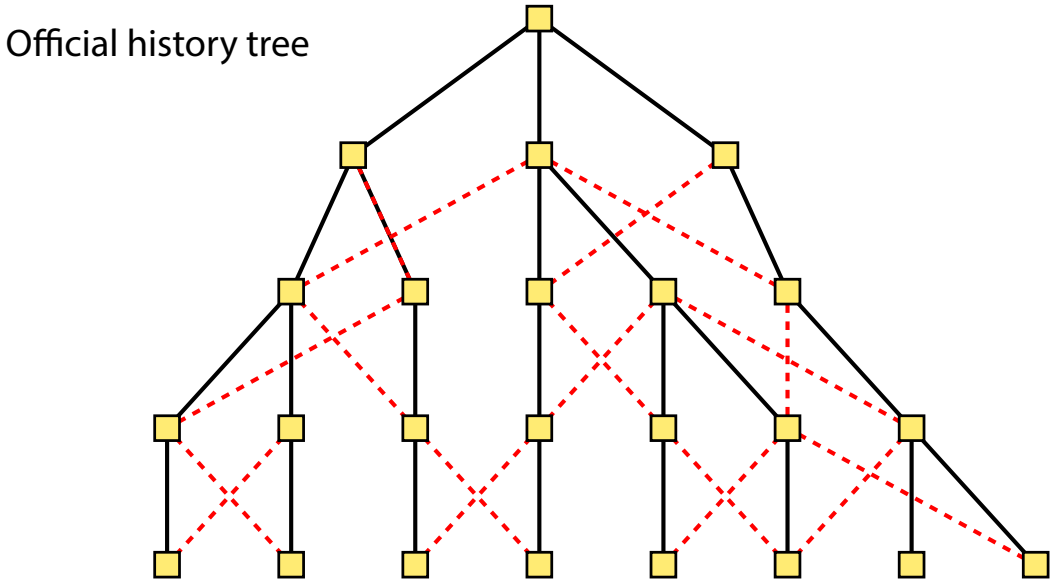
Construction of the OHT



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

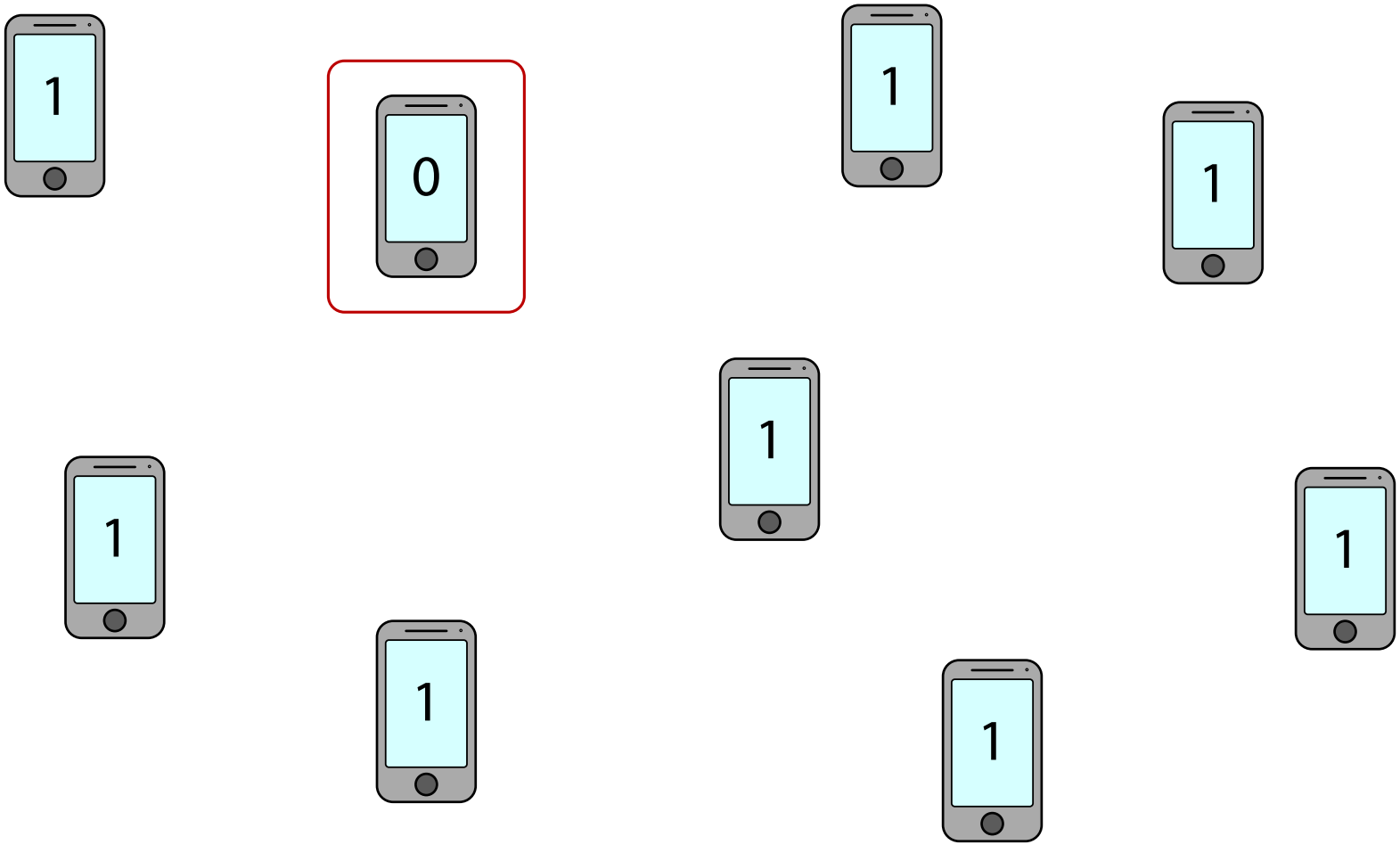
Construction of the OHT





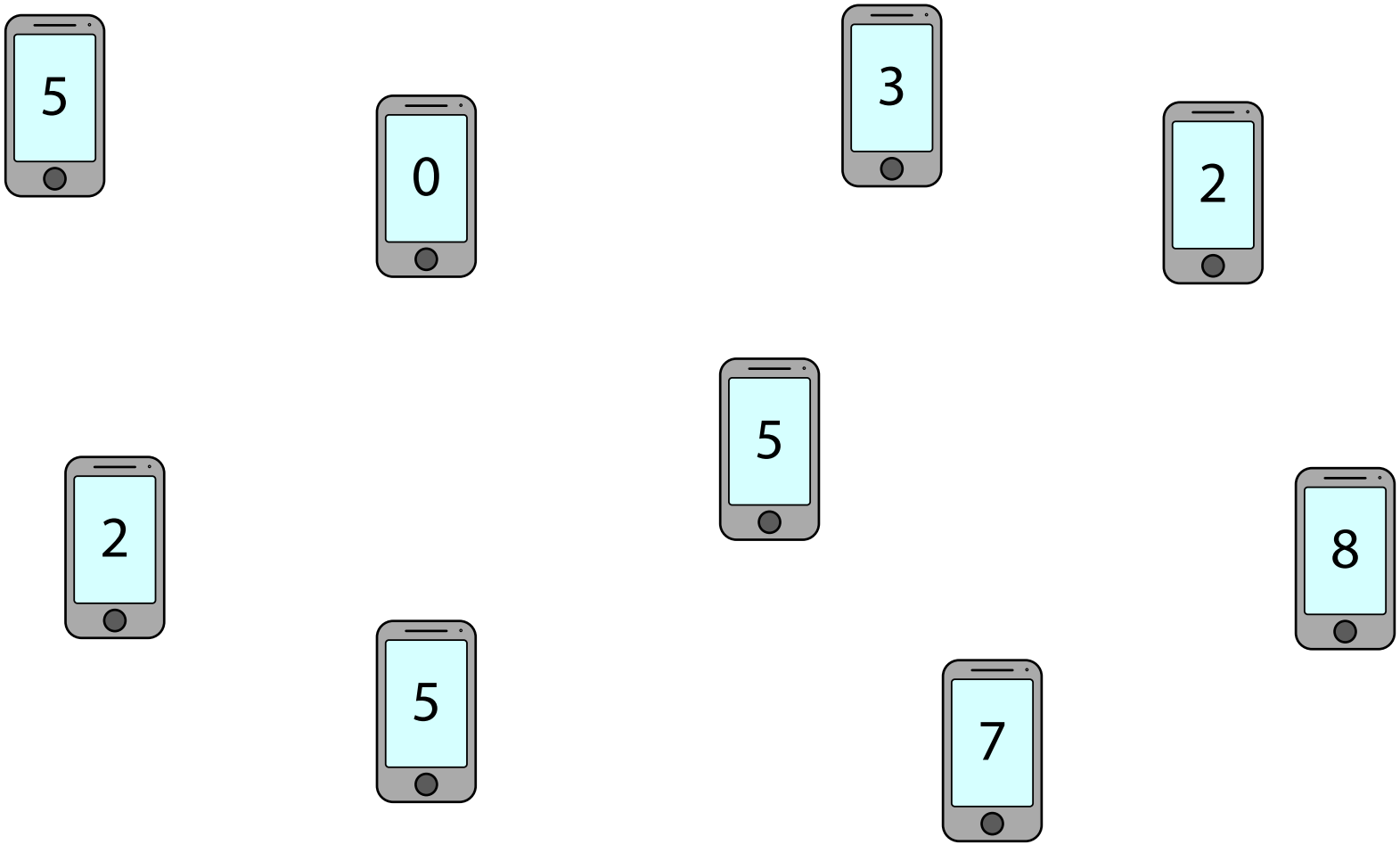
Construct the OHT level by level; continue until there are enough levels for the FOCS 2022 Counting algorithm to compute n.

Constructing a level of the OHT



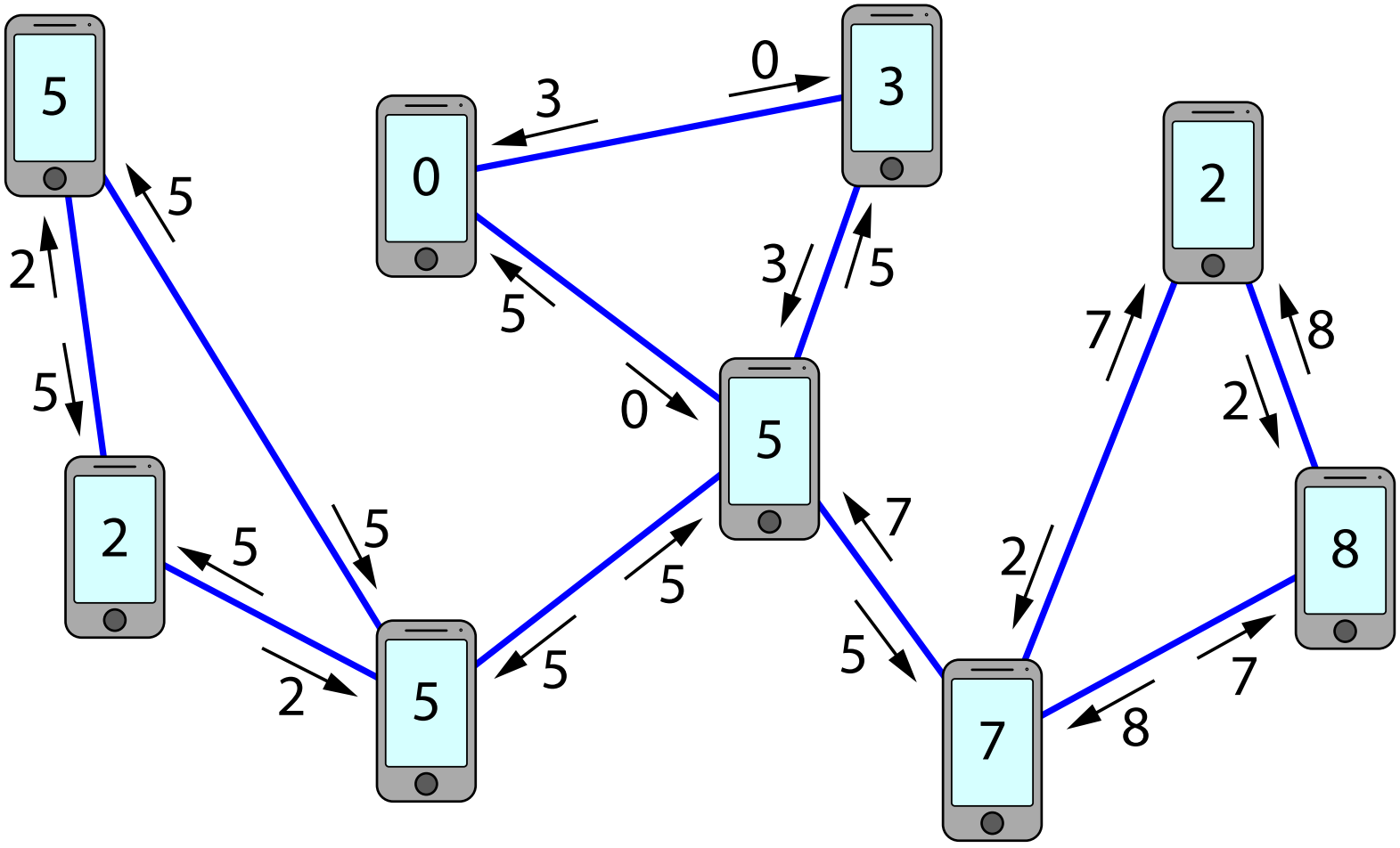
We assign IDs to agents: Initially, the Leader has ID 0, and the non-Leaders have ID 1. IDs may be modified over time...

Constructing a level of the OHT



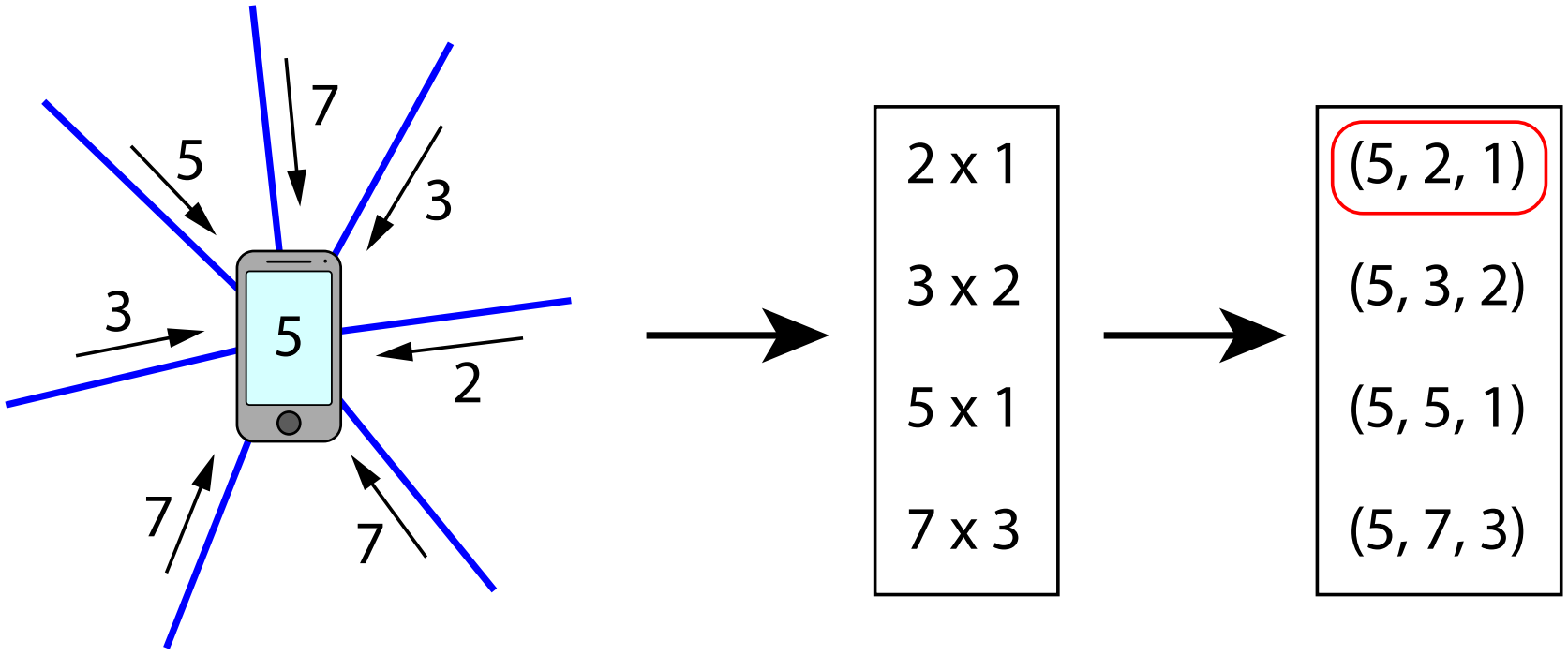
We assign IDs to agents: Initially, the Leader has ID 0, and the non-Leaders have ID 1. IDs may be modified over time...

Constructing a level of the OHT



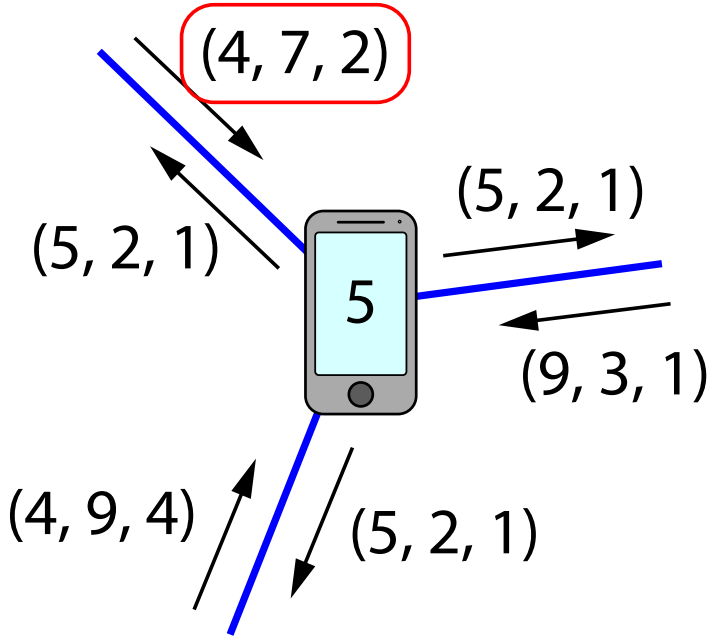
At a selected round, each agent sends its ID to all neighboring agents. Thus, each agent receives a *multiset* of IDs.

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



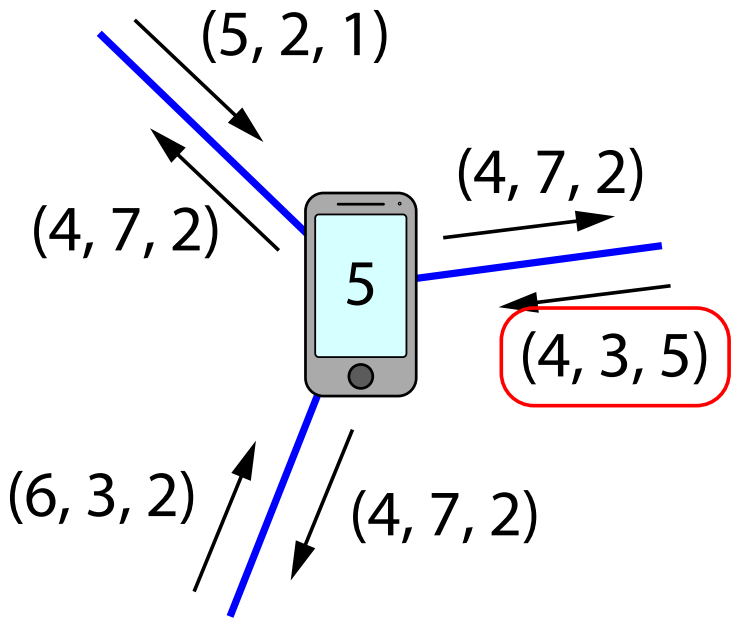
Minimum triplet:

~~(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:

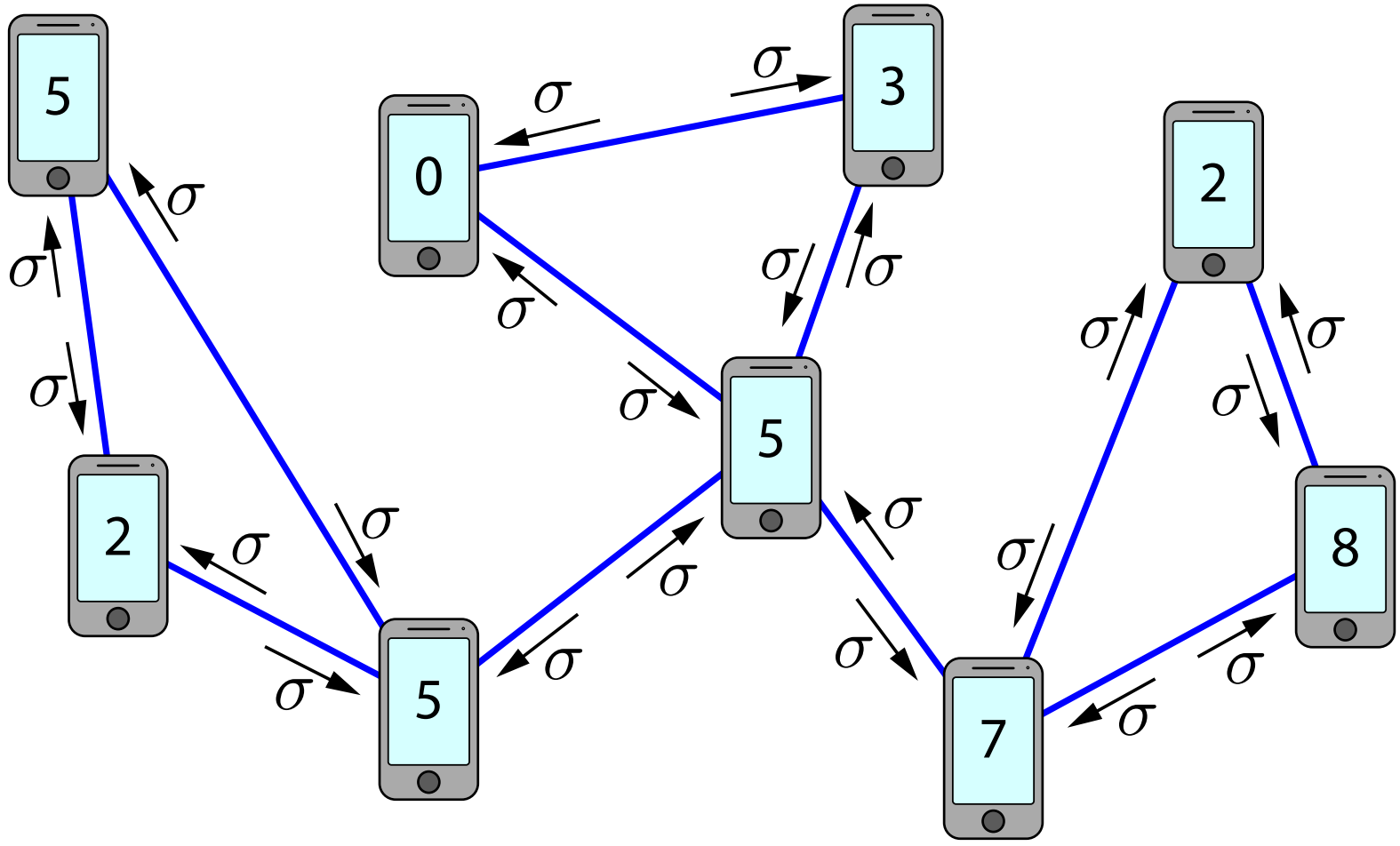
~~(5, 2, 1)~~

~~(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.

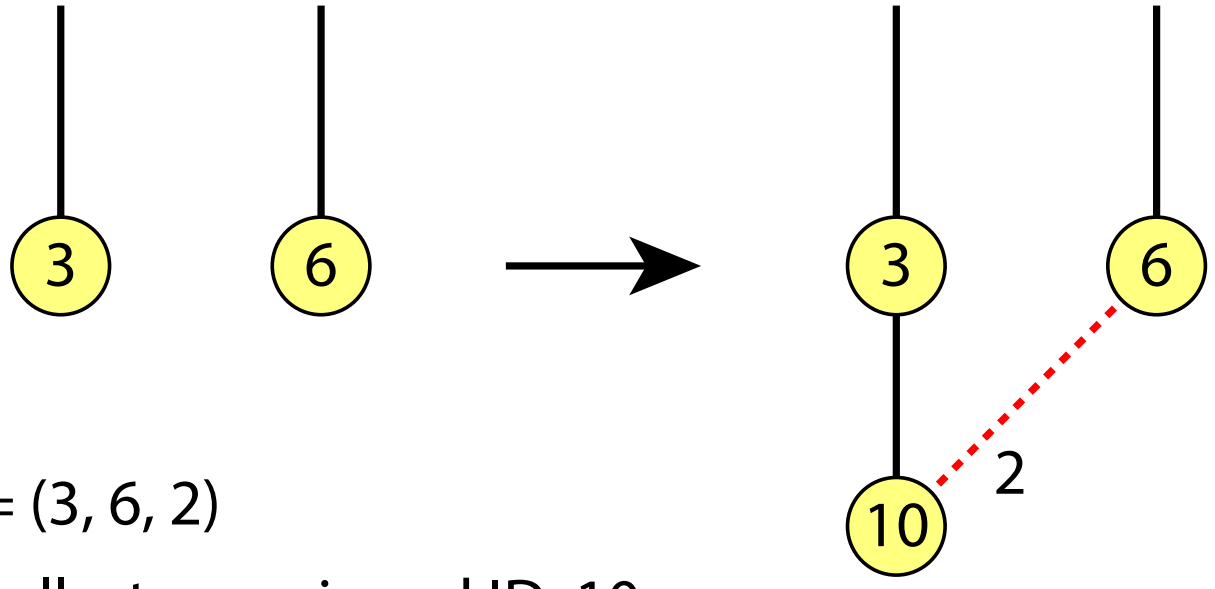
Constructing a level of the OHT



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:

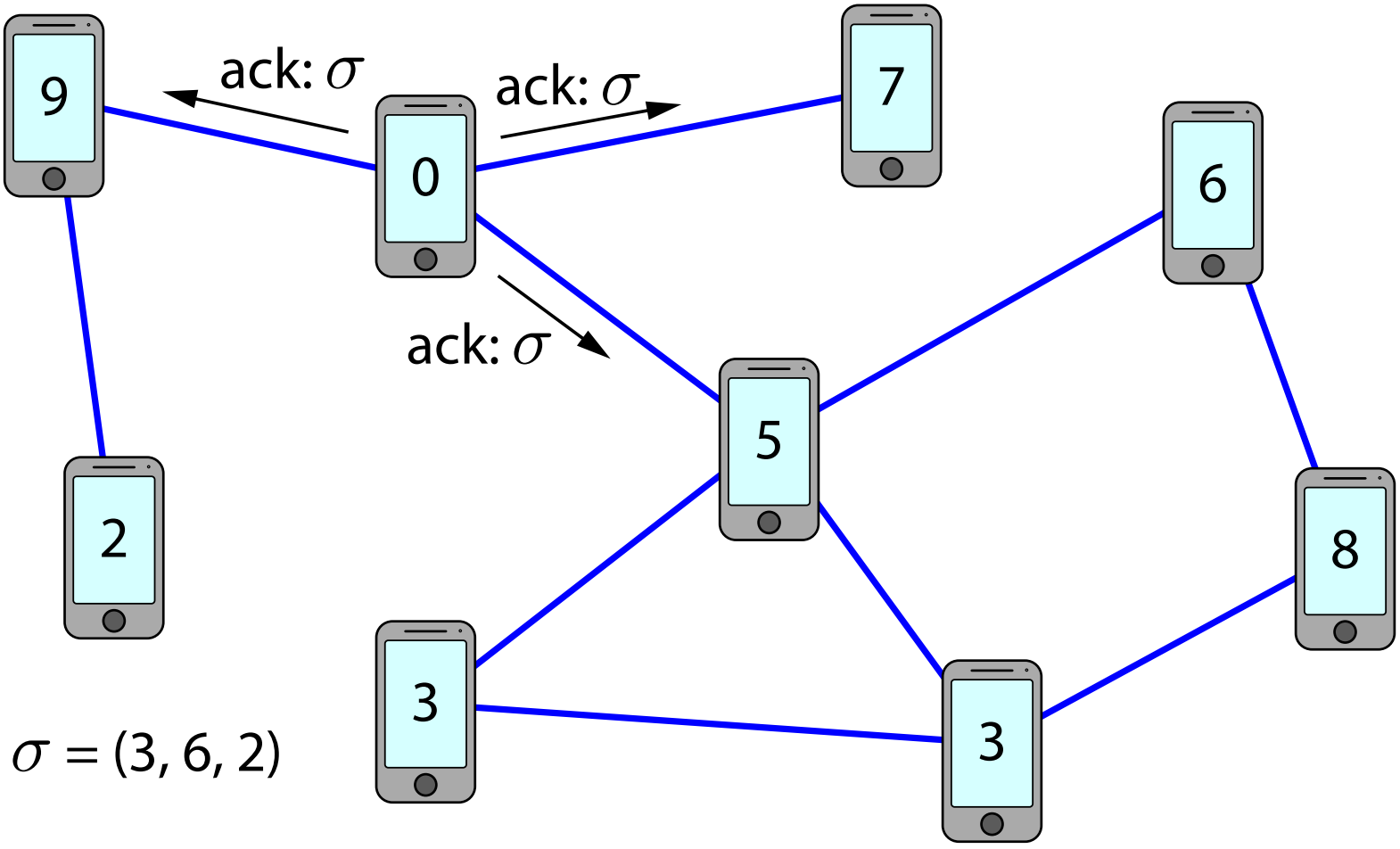


$\sigma = (3, 6, 2)$

Smallest unassigned ID: 10

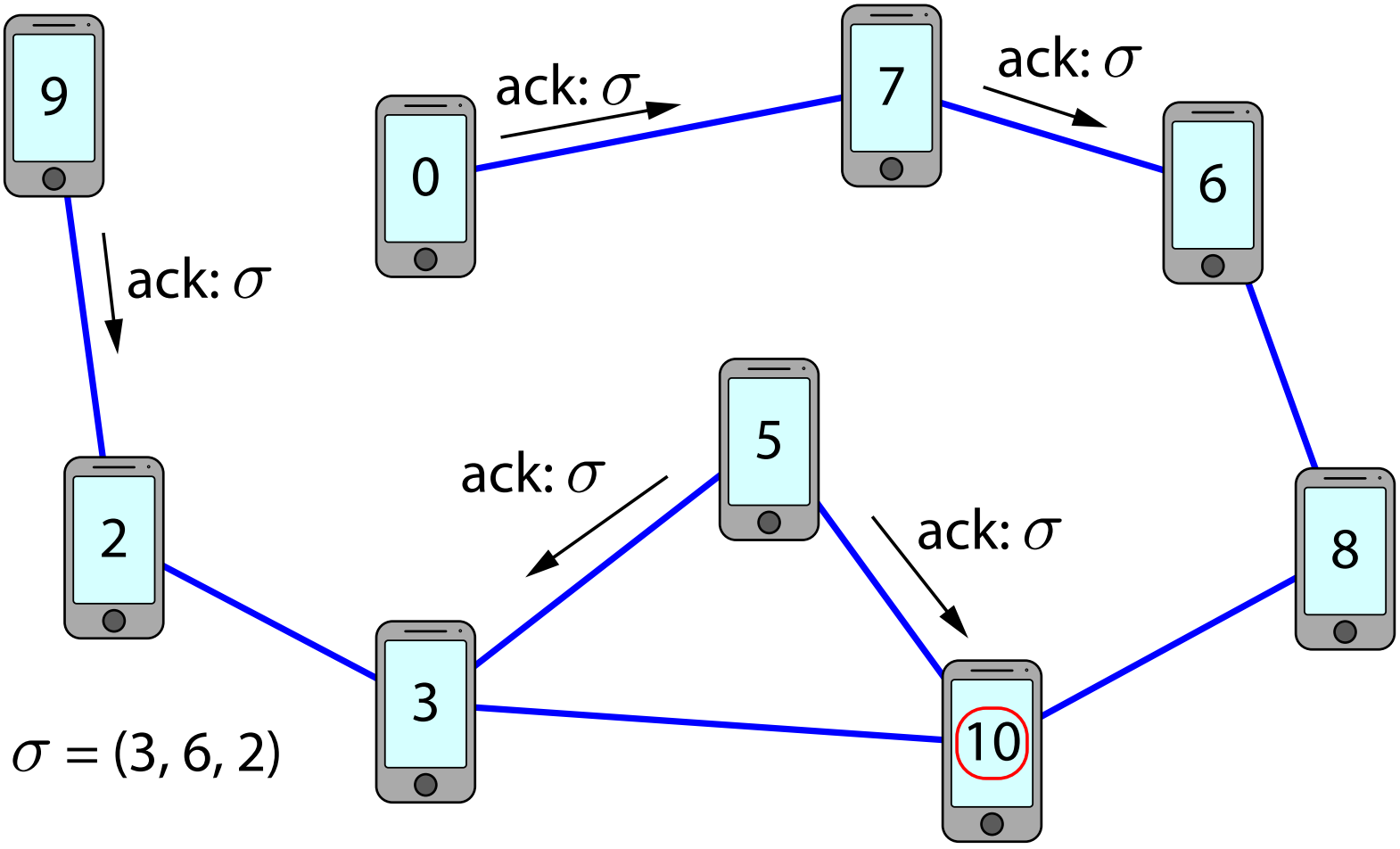
The Leader uses $\sigma = (\mathrm{ID1}, \mathrm{ID2}, m)$ to update the OHT: ID1 gets a child with a fresh name ID1' and the red edge $(\mathrm{ID1'}, \mathrm{ID2}, m)$.

Constructing a level of the OHT



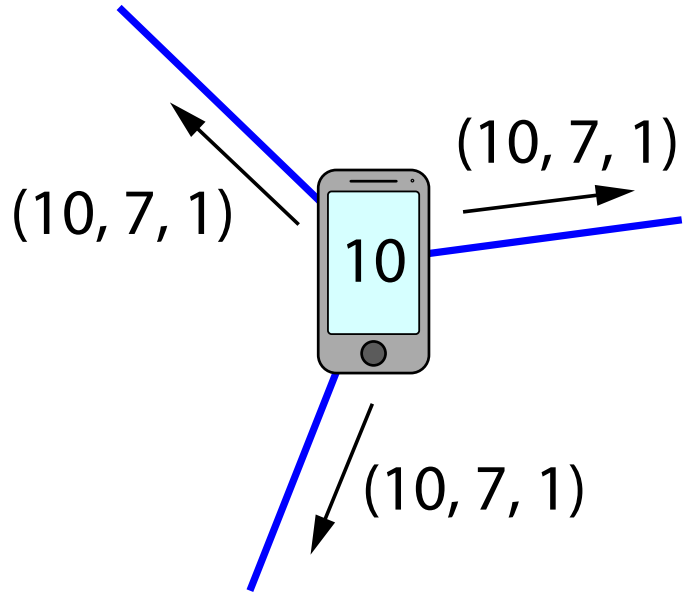
The Leader broadcasts σ as an acknowledgment to all agents, which modify their local OHT in the same way.

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)~~

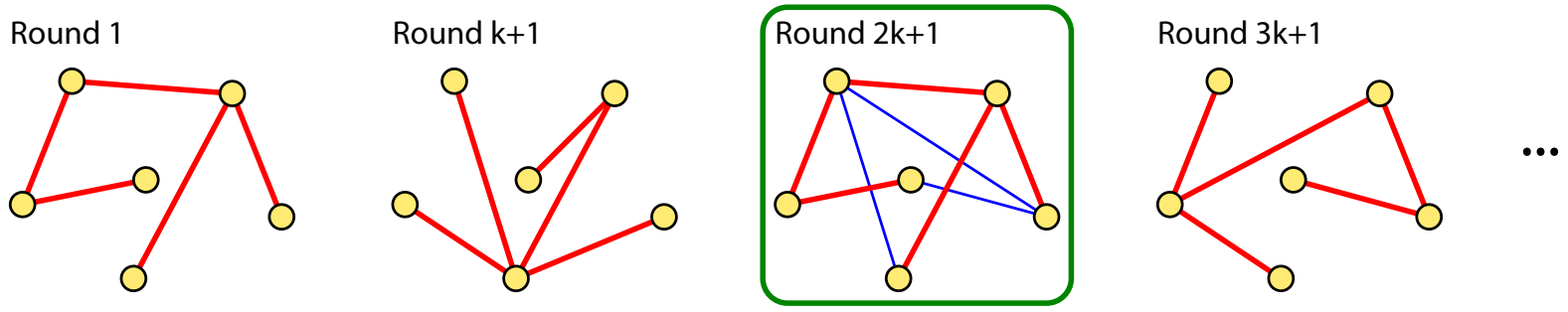
(10, 7, 1)

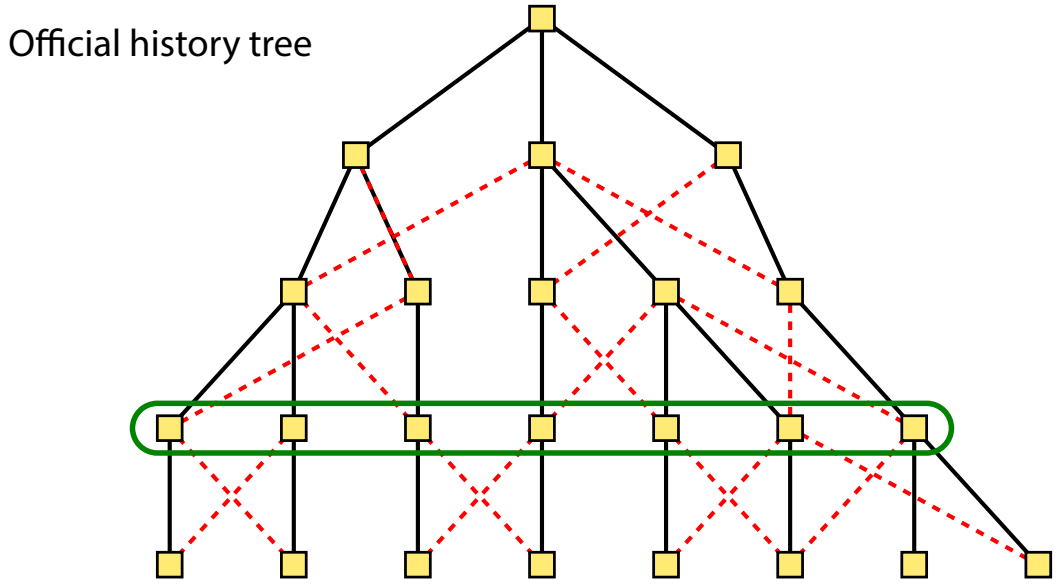
(10, 7, 5)

(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

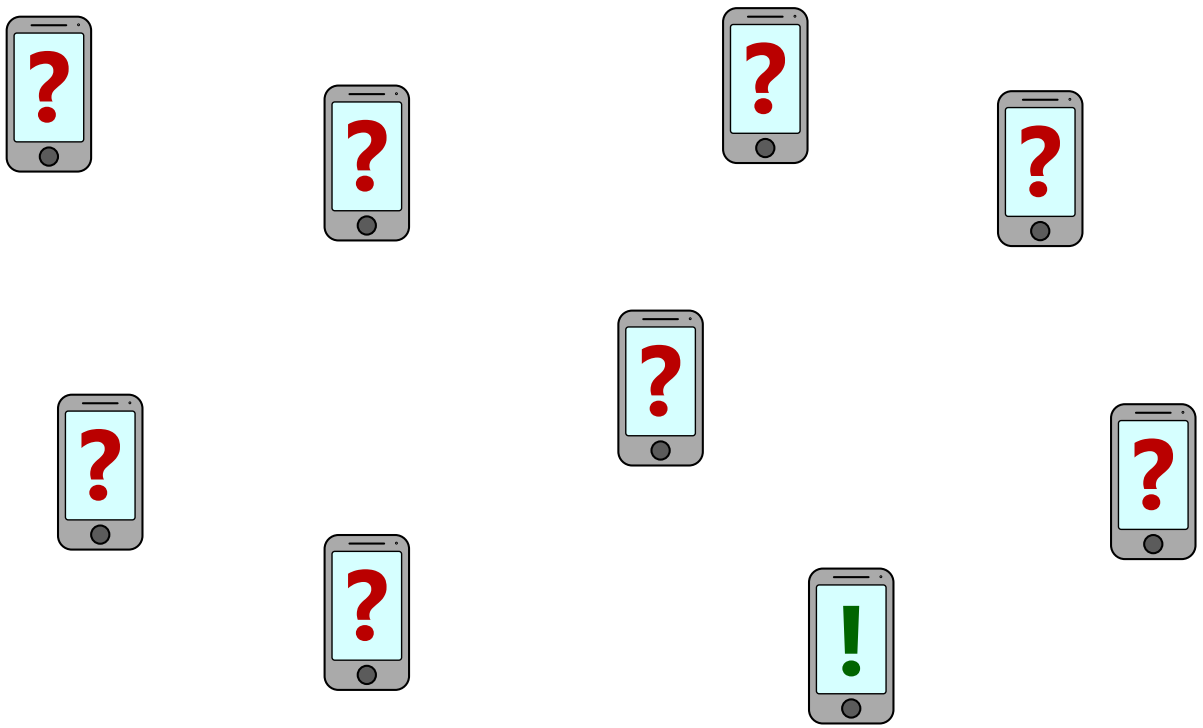




When the current level of the OHT is complete, it represents a *spanning tree* (of size $O(n)$) of the network at the selected round.

Broadcasting information

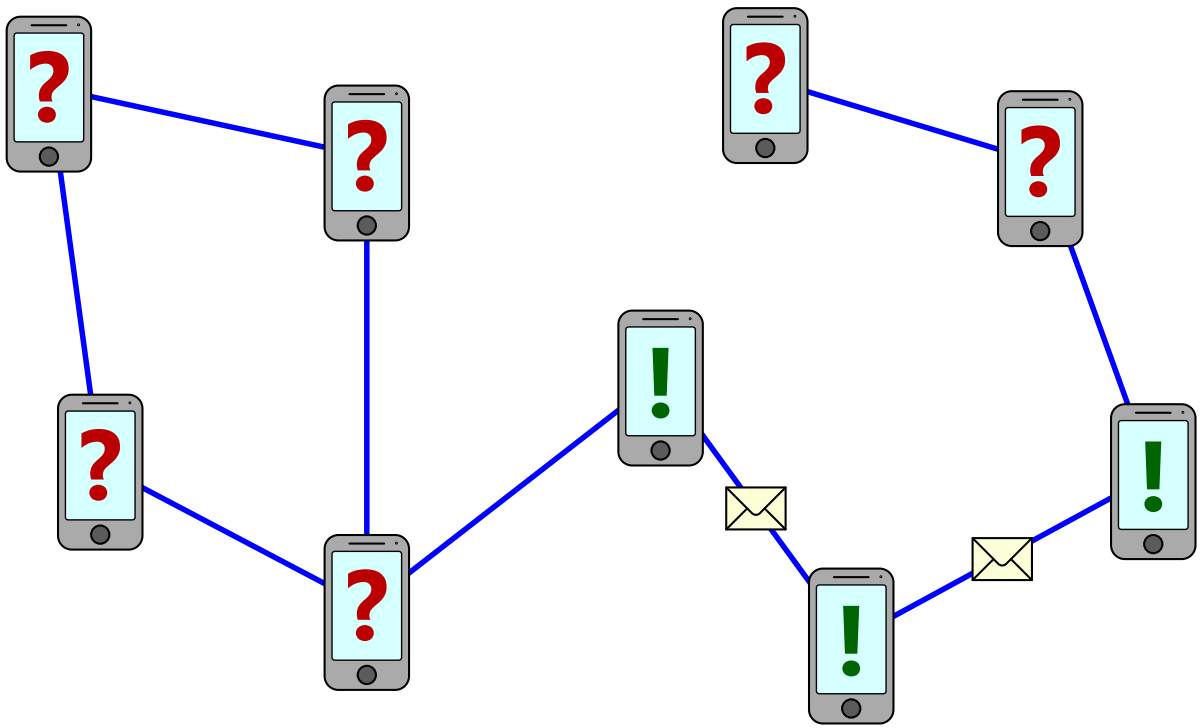
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



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

Broadcasting information

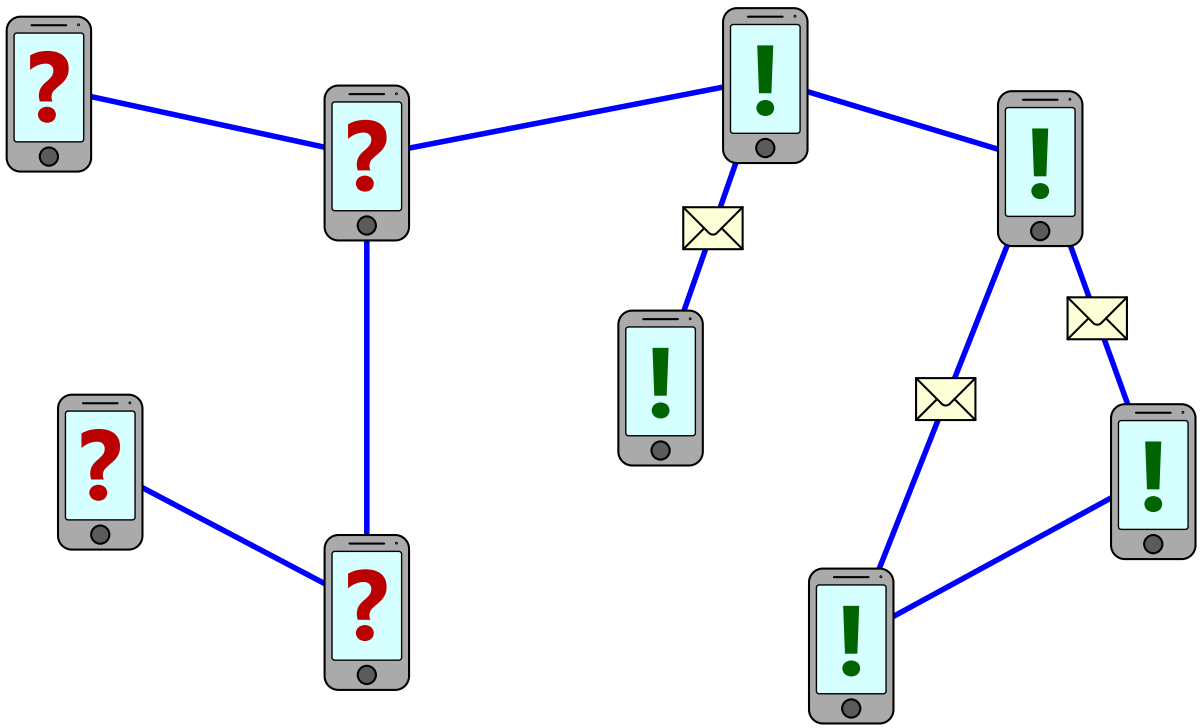
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



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

Broadcasting information

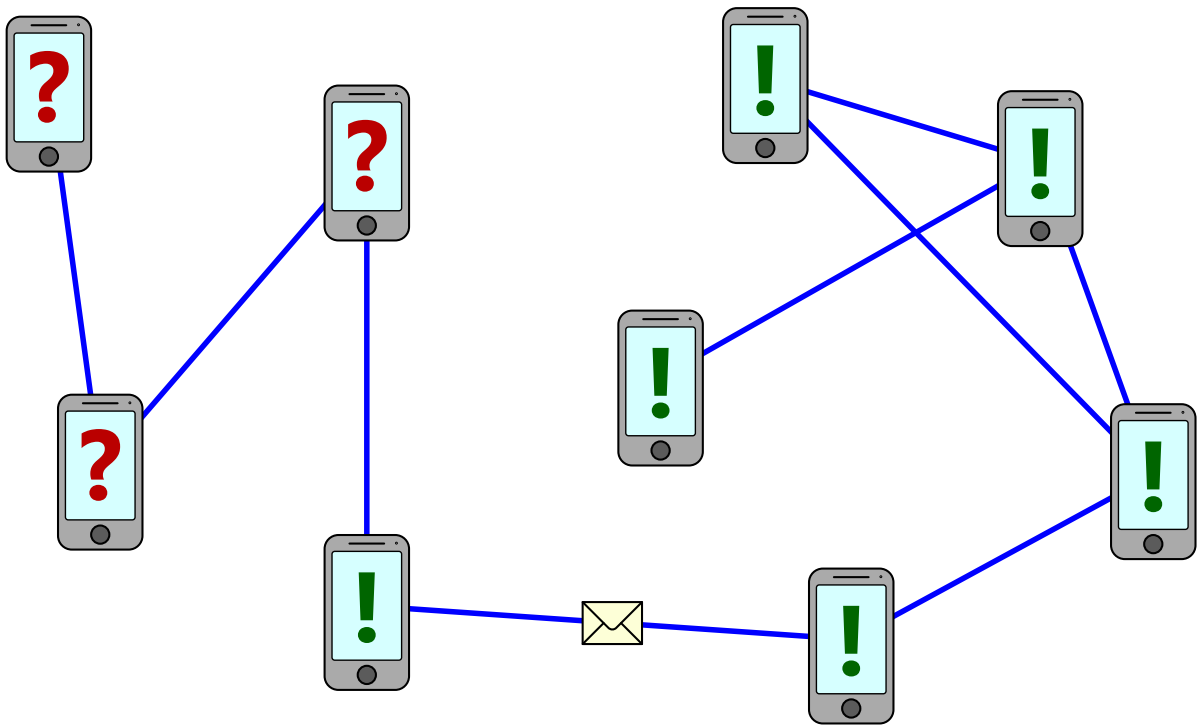
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



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

Broadcasting information

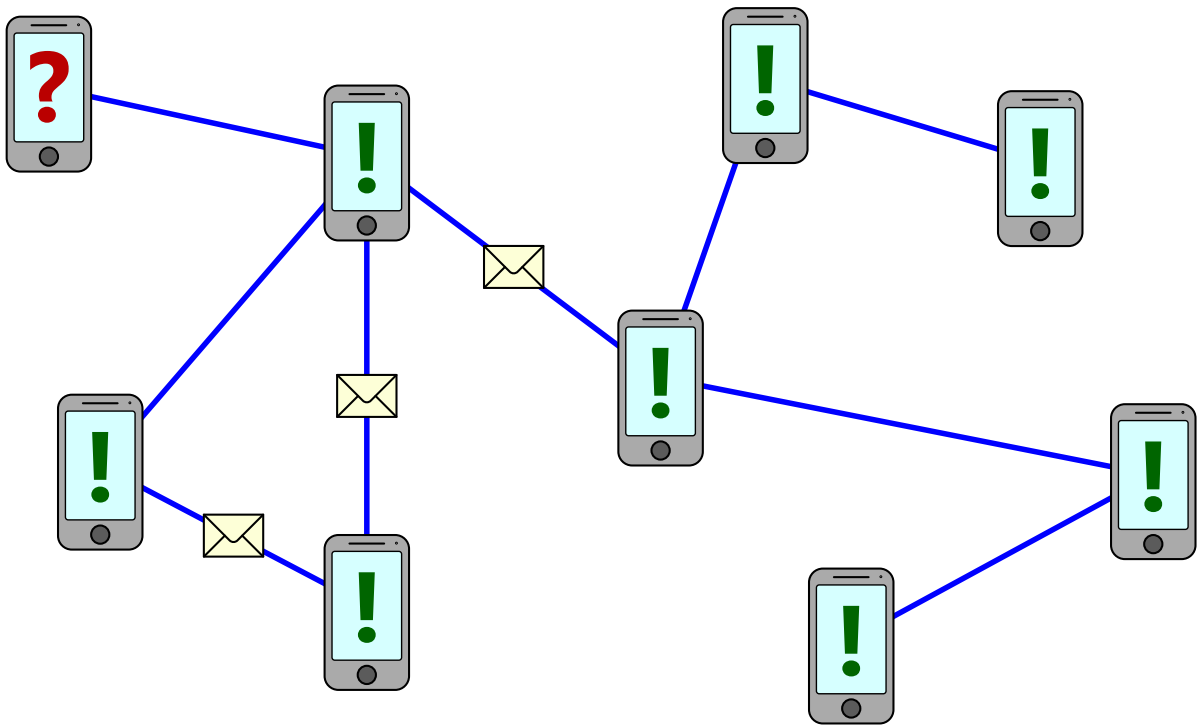
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



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

Broadcasting information

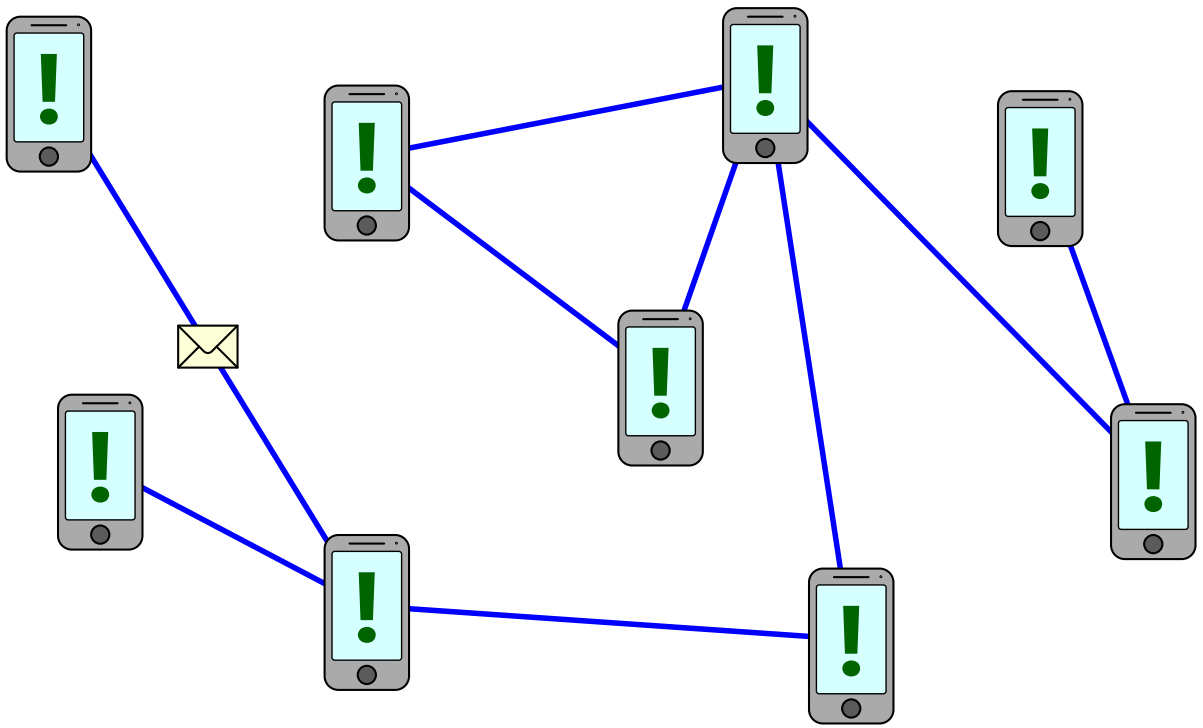
Since the network is connected at all rounds, any piece of information can reach every agent in at most $n - 1$ rounds.



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

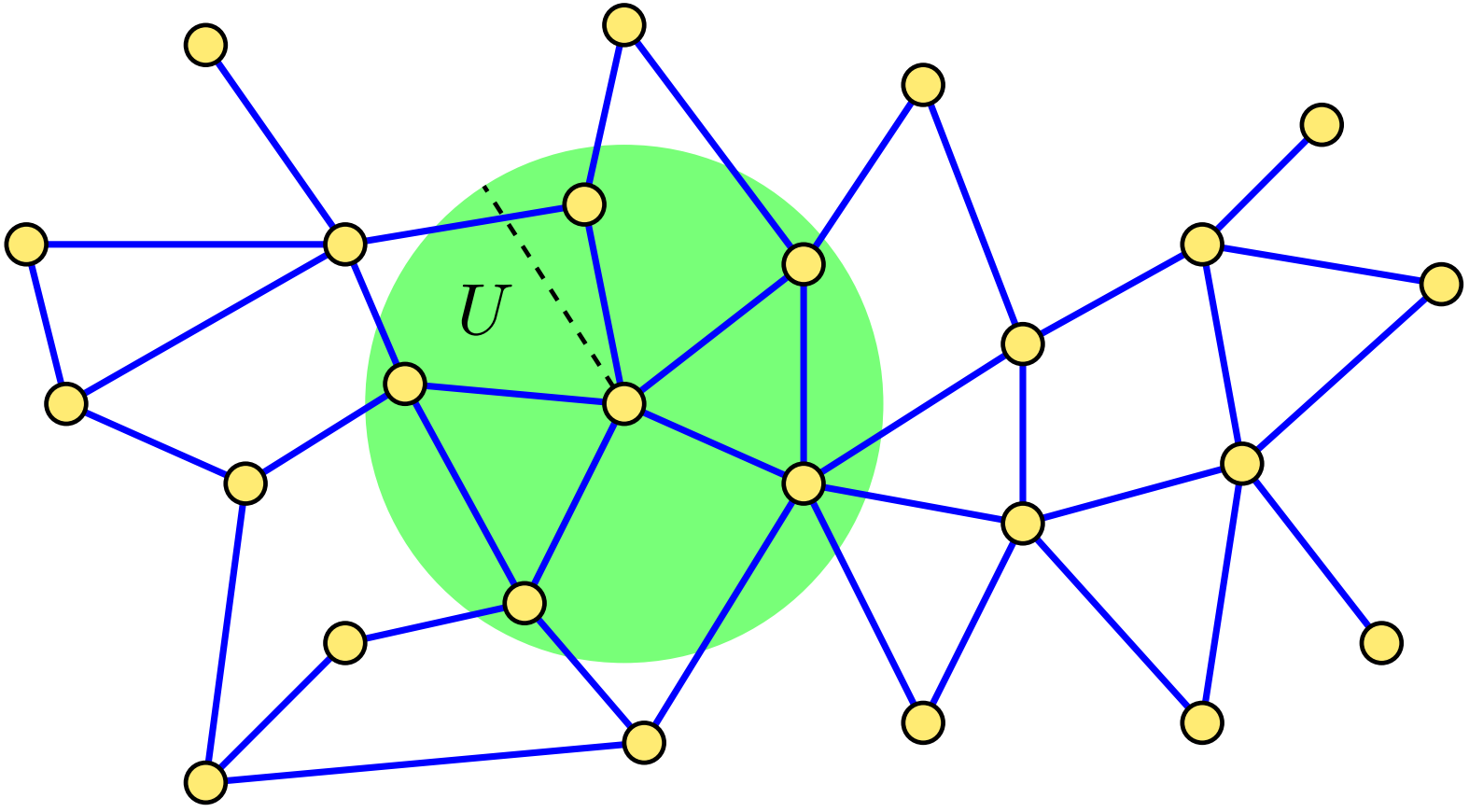
Broadcasting information

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



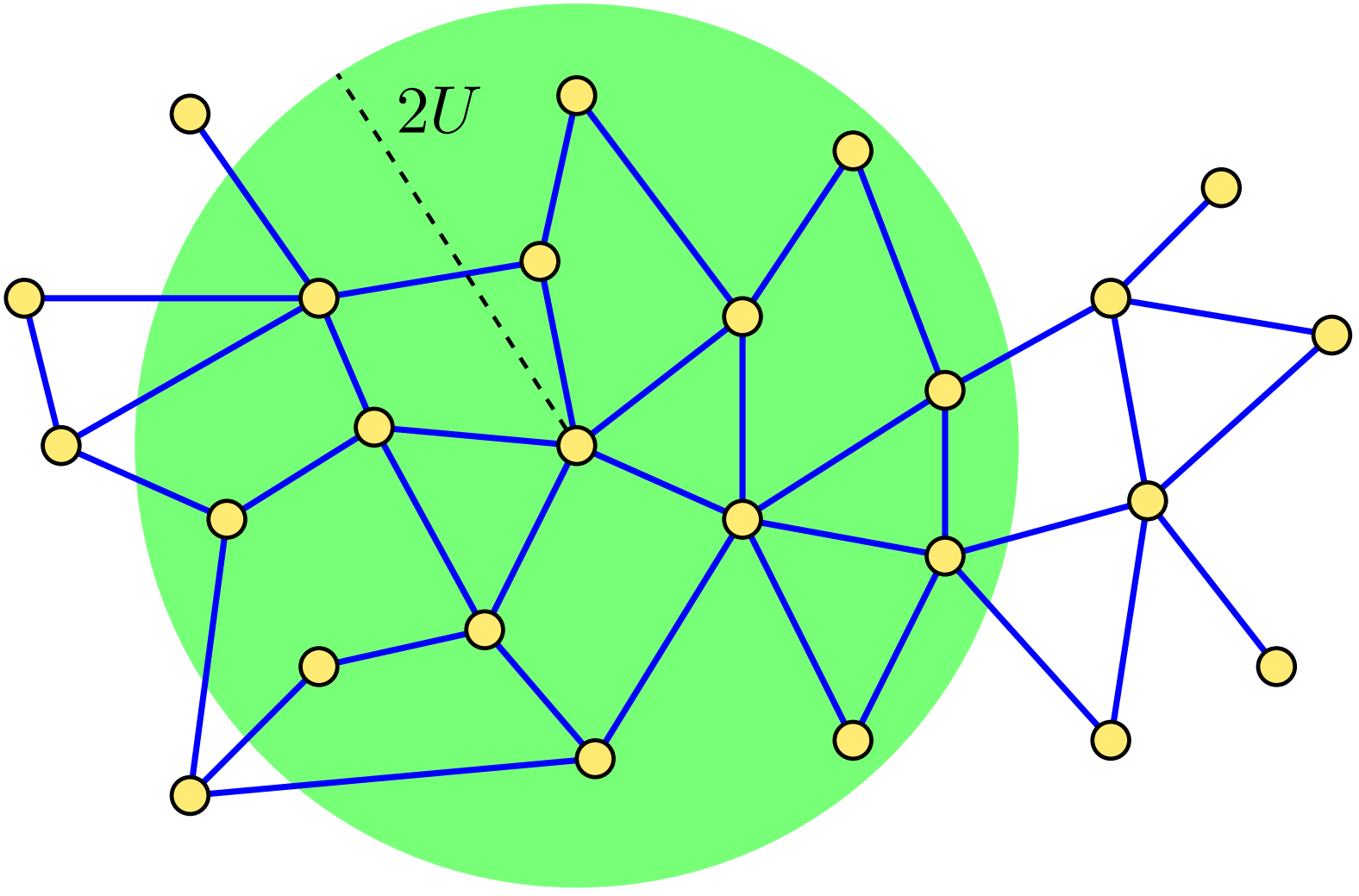
Thus, each broadcast takes at most $n - 1$ rounds to complete.

Reset module



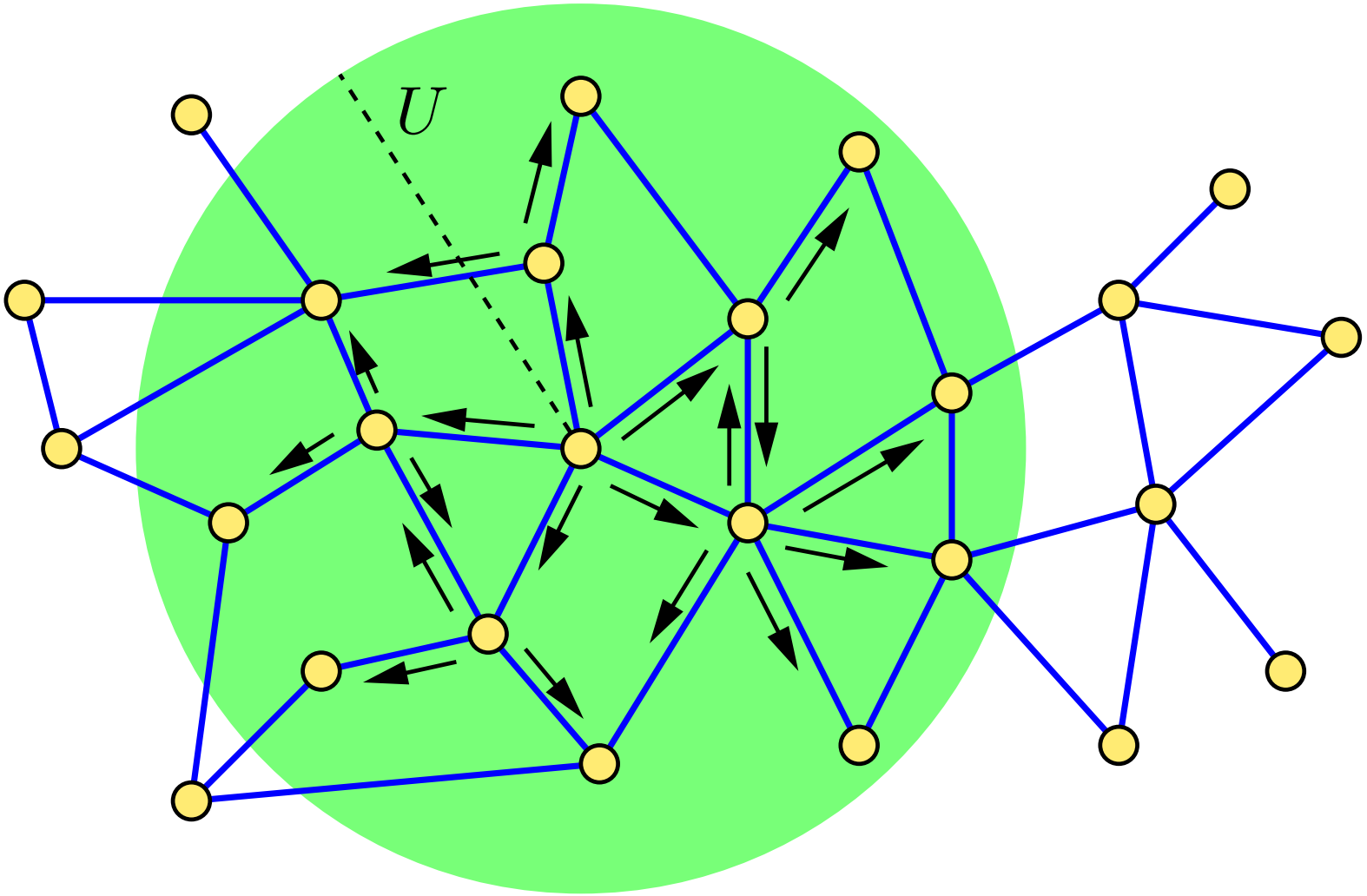
Agents do not know n, so they cannot broadcast information correctly. They only have an *estimate* U on the network size.

Reset module



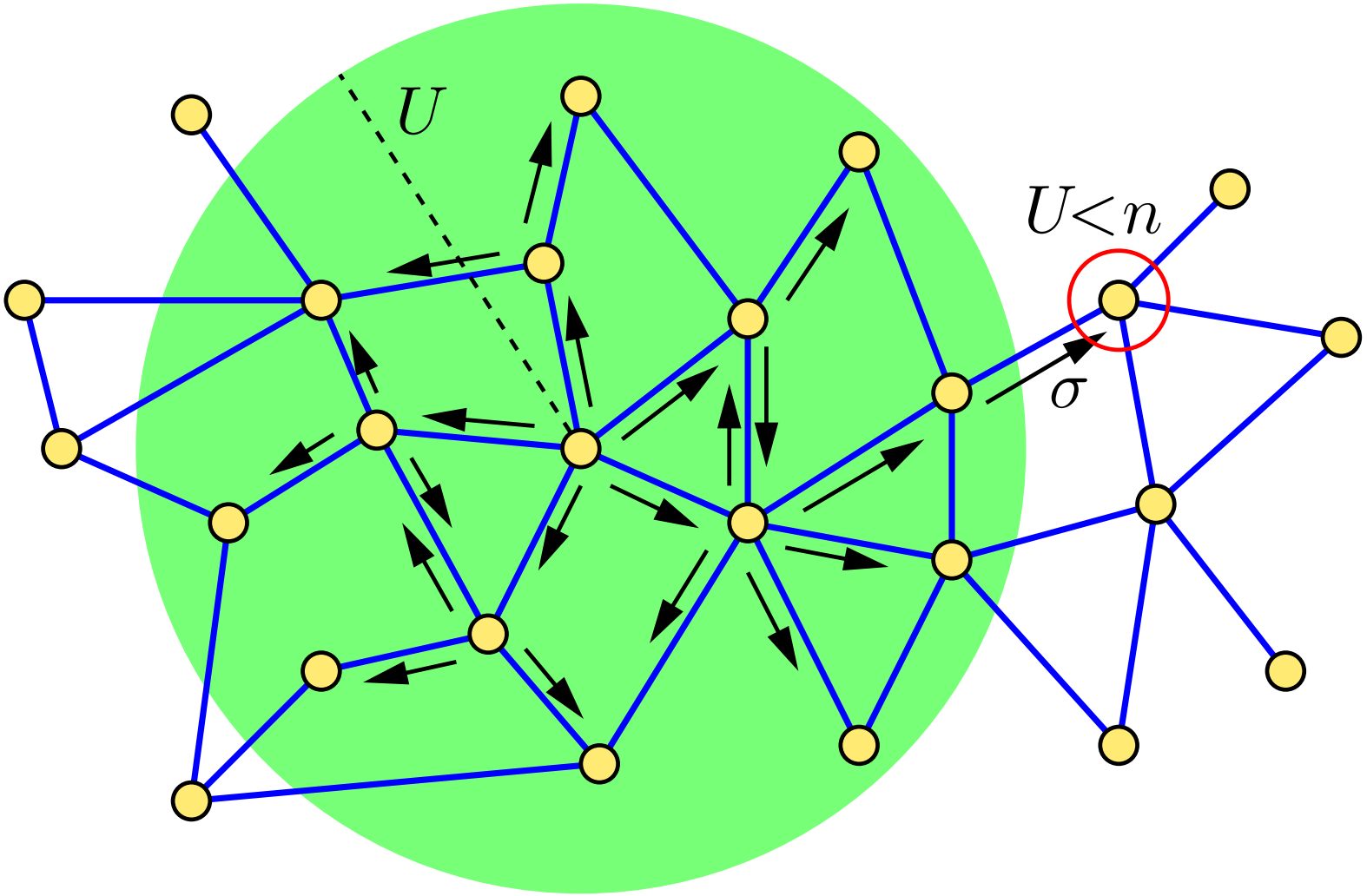
All agents begin with $U = 1$, and then implement a *reset module* that doubles U every time they detect that $U < n$.

Reset module



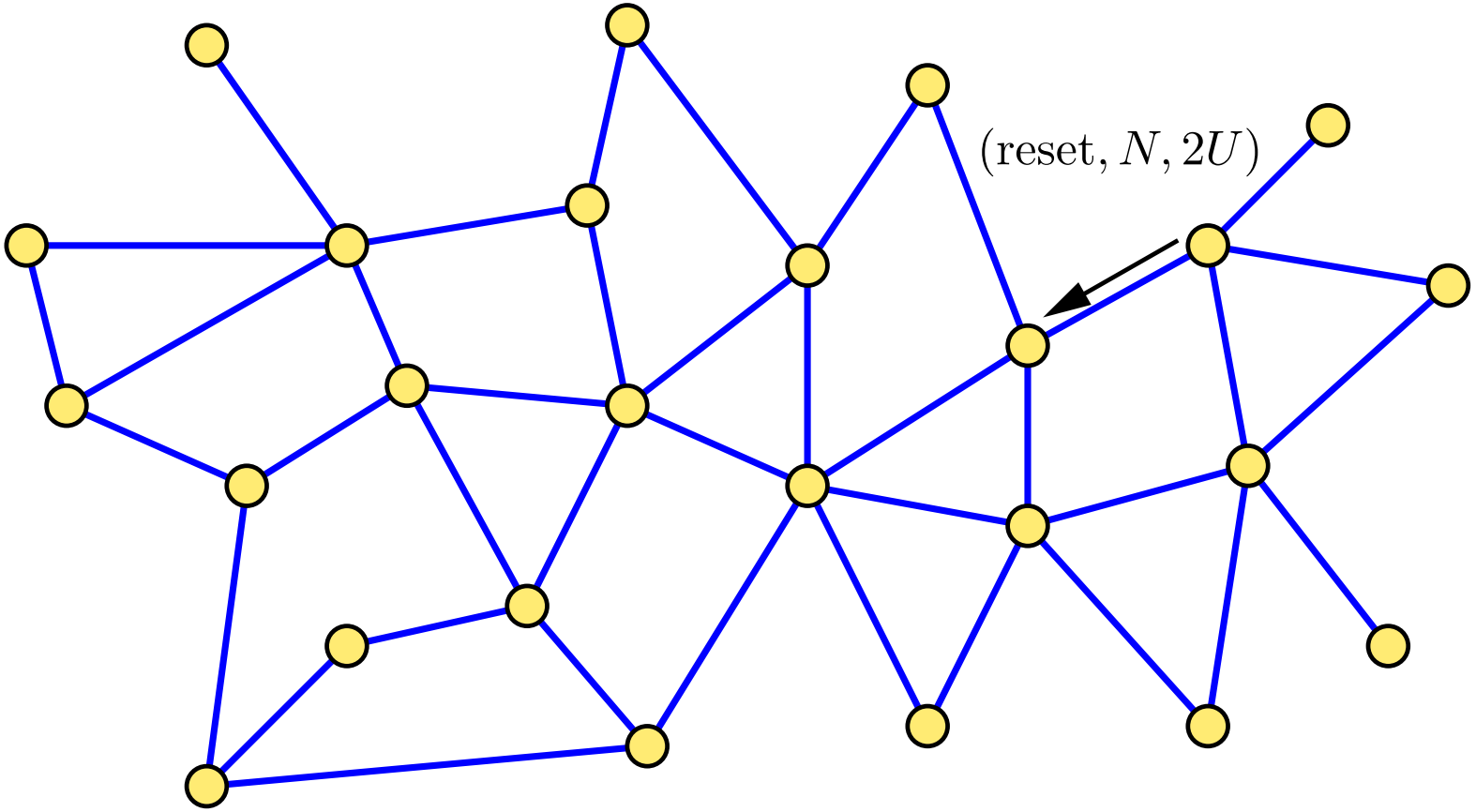
At each round, agents send triplets (N, R, U) (alongside the usual triplets $(\mathsf{ID1}, \mathsf{ID2}, m)$), indicating that this is the Nth broadcast, which started at round R and runs for U rounds.

Reset module



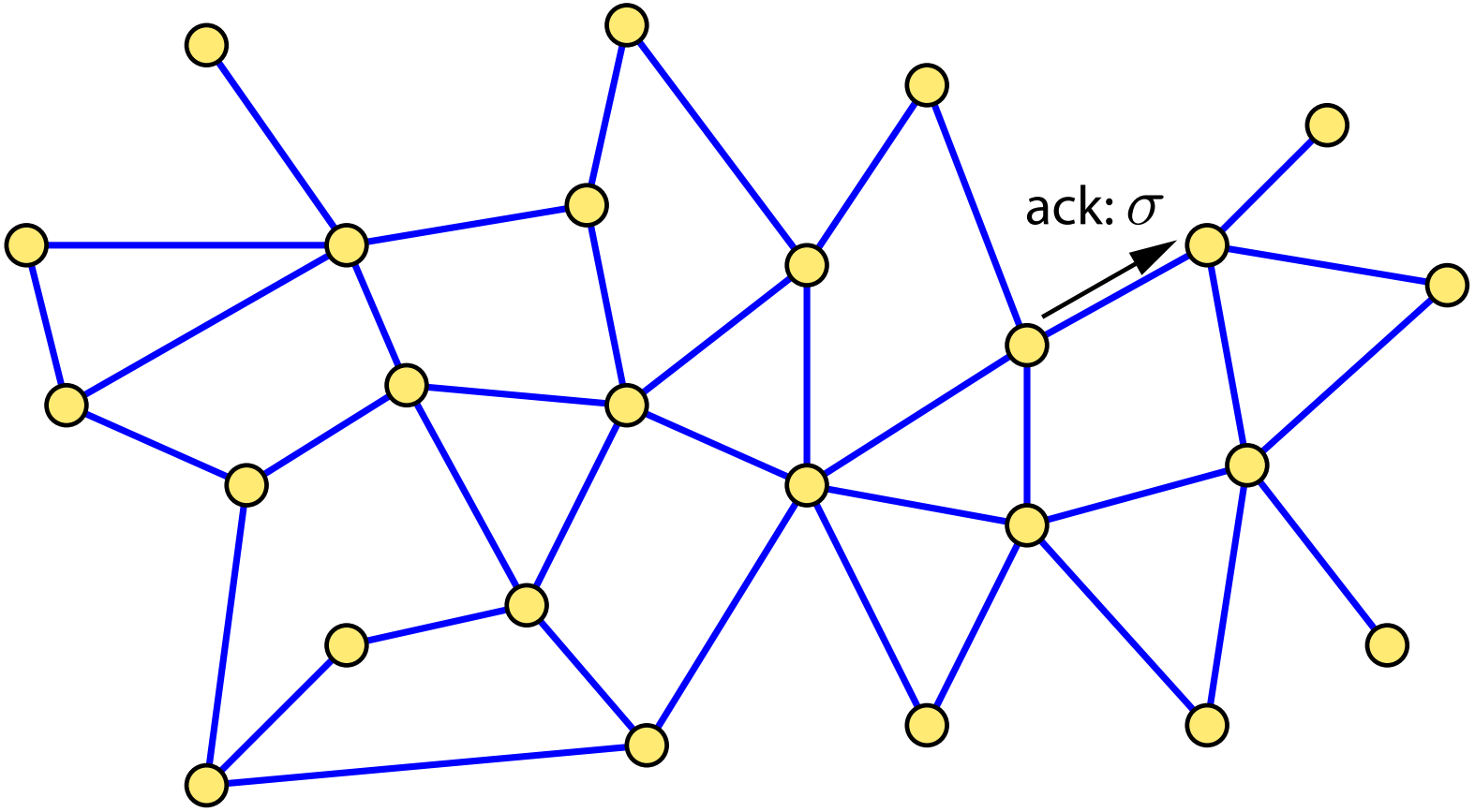
At the Uth round of a broadcast, if an agent receives a new minimum triplet σ, it knows that $U < n$, and starts broadcasting a *reset message* $(\text{reset}, N, 2U)$ that takes priority over anything else.

Reset module



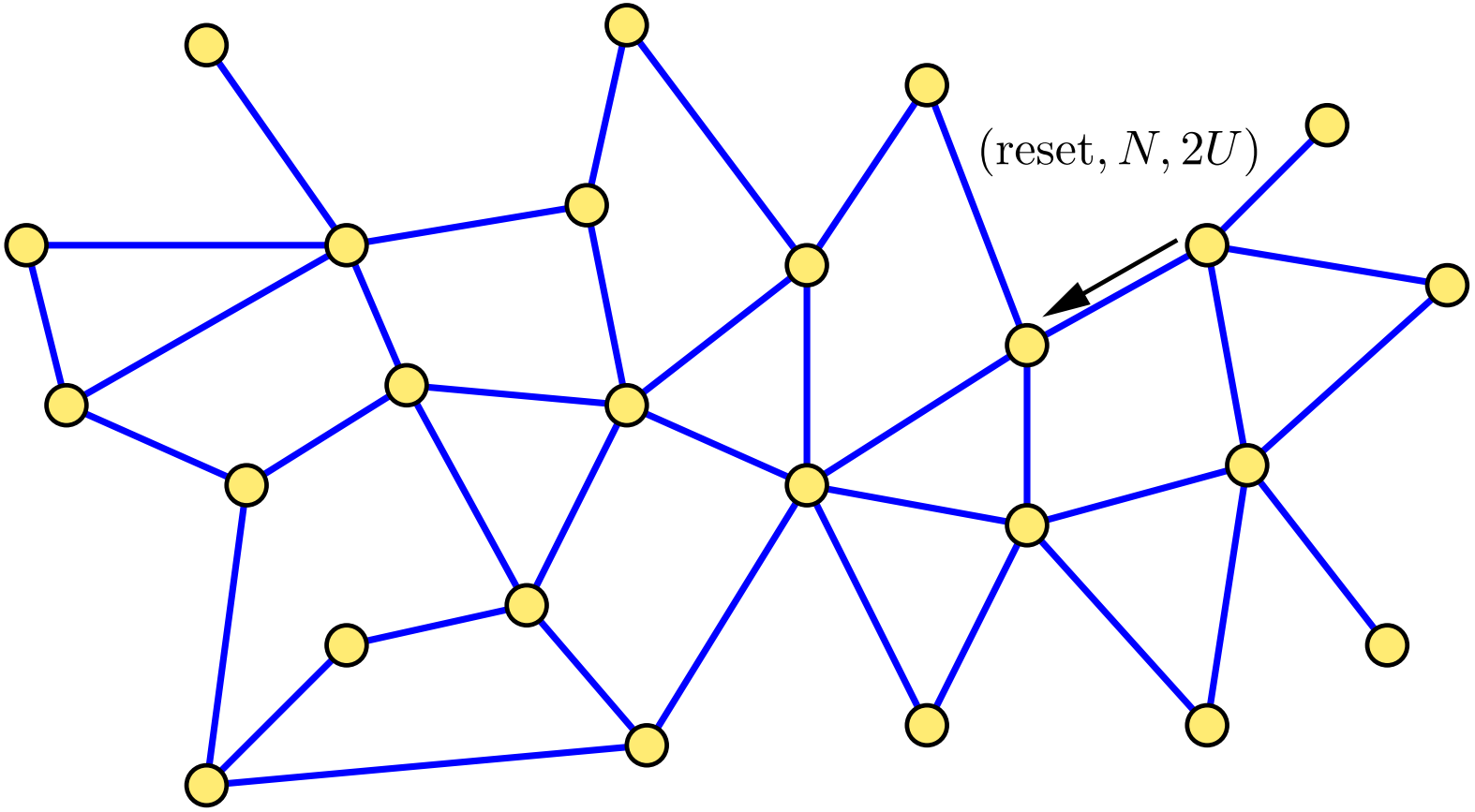
At the Uth round of a broadcast, if an agent receives a new minimum triplet σ, it knows that $U < n$, and starts broadcasting a *reset message* (reset, $N, 2U$) that takes priority over anything else.

Reset module



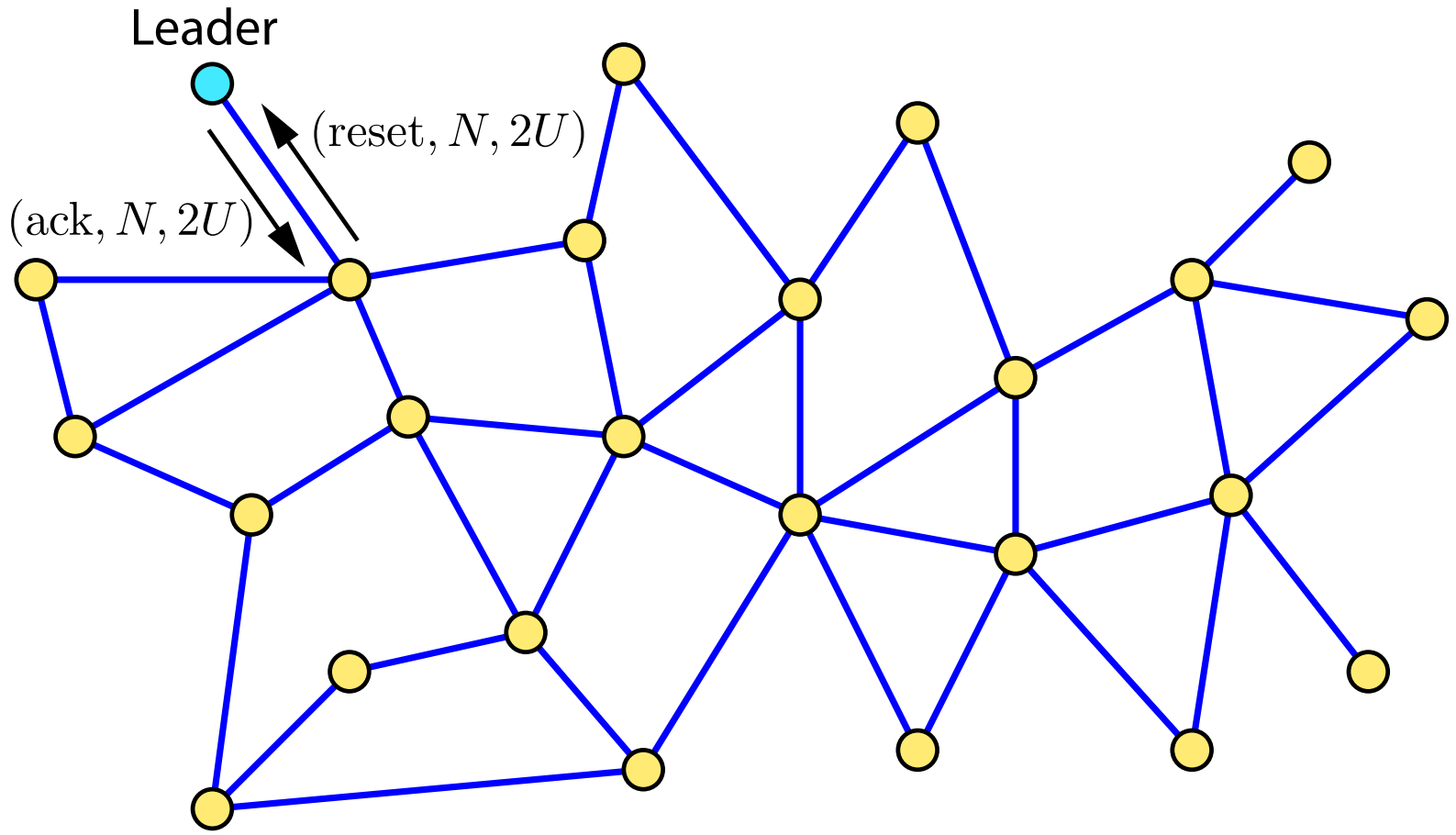
Similarly, if an agent does not receive an acknowledgment from the Leader for broadcast N or does not receive the acknowledgment it expects, it broadcasts a reset message $(\text{reset}, N, 2U)$.

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, $2U$).

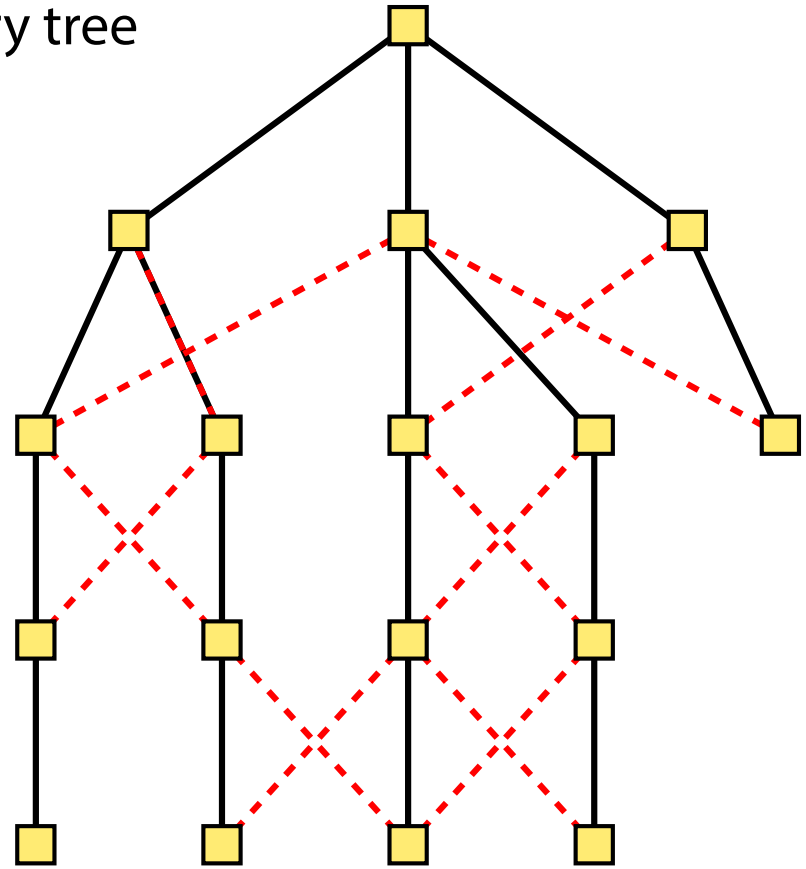
Reset module



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 $2U$.

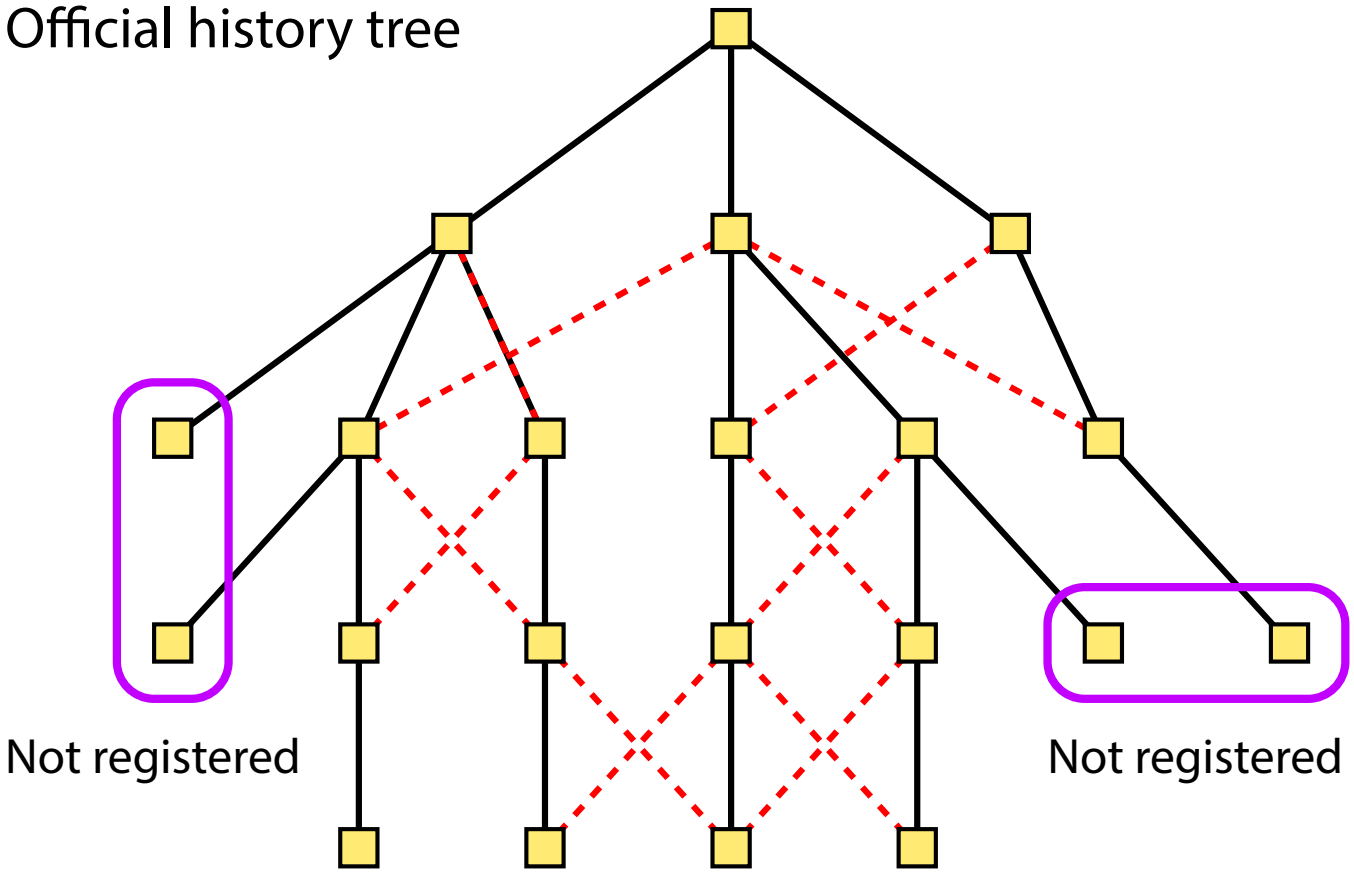
Algorithm correctness

Official history tree



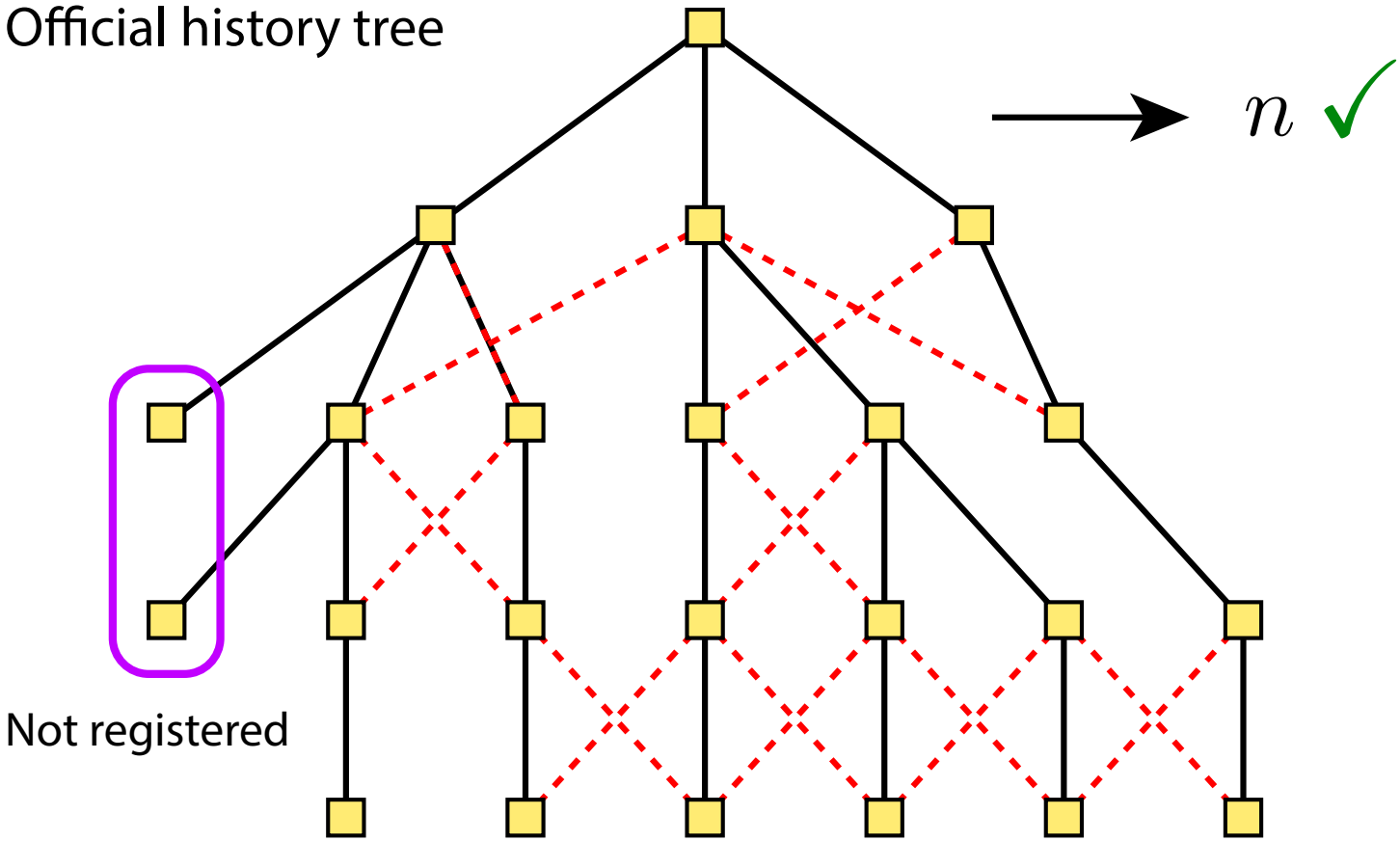
Because any piece of information becomes official only when the Leader sends an acknowledgment about it, at any time there is only *one version* of the OHT circulating in the network.

Algorithm correctness



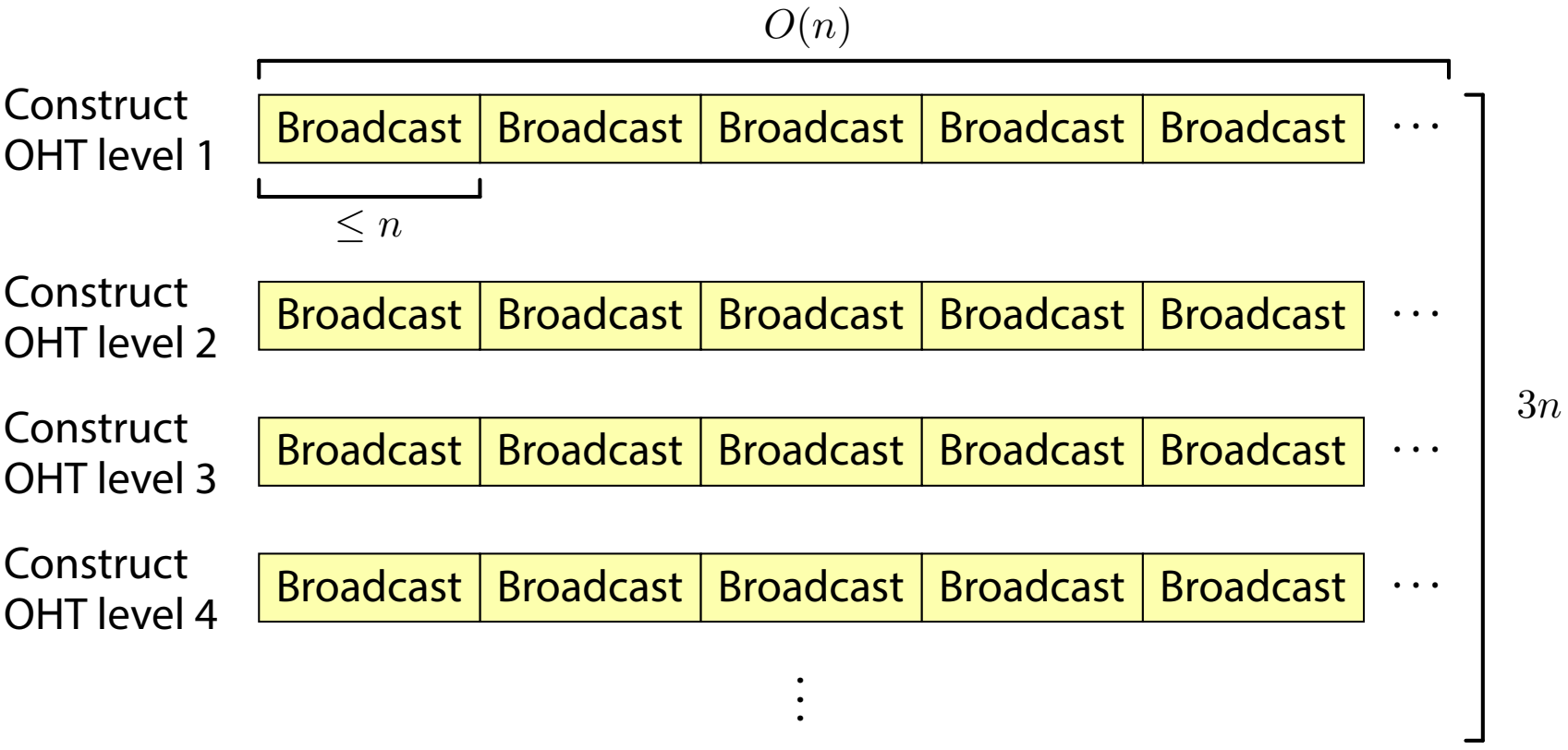
It is possible for the OHT to be missing some red edges, but the FOCS 2022 Counting algorithm can detect when an incomplete history tree does not contain enough information to compute n.

Algorithm correctness



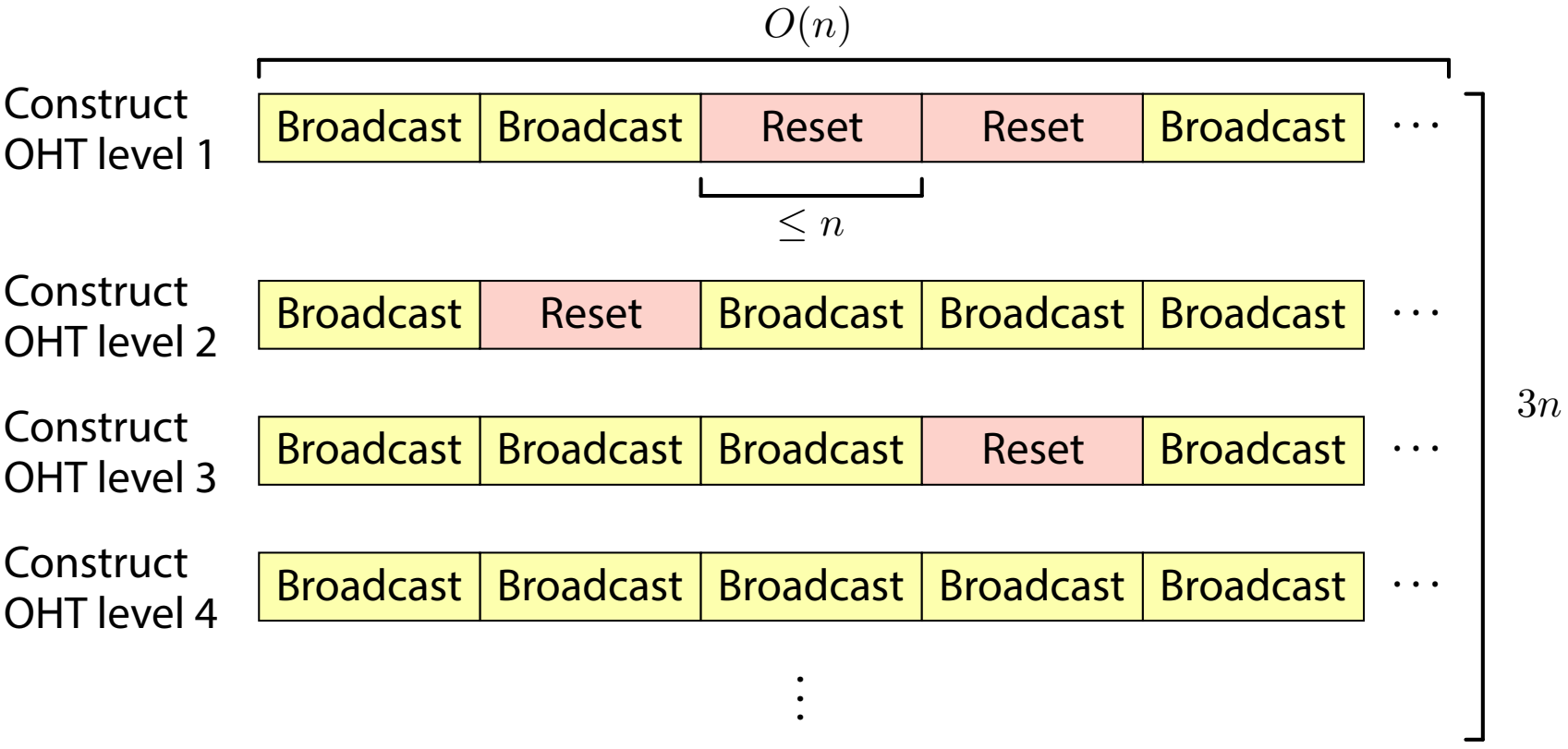
After completing each level of the OHT, we run the FOCS 2022 Counting algorithm on it. As soon as it returns a number n (as opposed to "Unknown"), we can safely output n.

Algorithm running time



We need $3n$ levels of the OHT for the FOCS 2022 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.

Conclusions and open problems

Theorem (This work)

In a congested anonymous dynamic network with n agents and a Leader, the Counting Problem can be solved in $O(n^3)$ rounds.

Open problem

Can we close the gap between $\Omega(n^2/\log n)$ and $O(n^3)$ rounds?

Theorem (Di Luna–V., arXiv 2022)

In a non-congested anonymous dynamic network with n agents and ℓ Leaders, the Counting Problem can be solved in $O(\ell^2 n)$ rounds.

Open problem

Can we solve the Counting Problem in congested networks with $\ell > 1$ Leaders, and in how many rounds?