Linear-Time Computation in Anonymous Dynamic Networks with a Leader

Giovanni Viglietta

Joint work with Giuseppe A. Di Luna

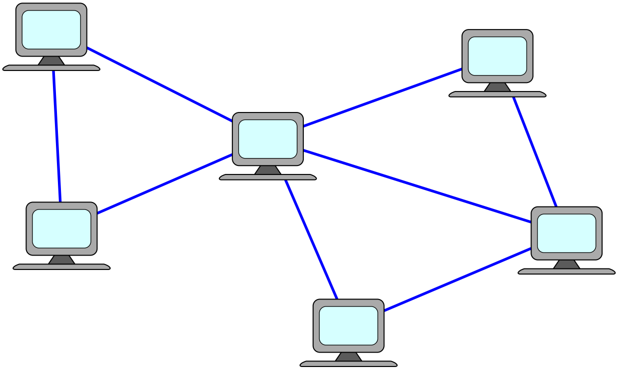
JAIST – November 10, 2021

Overview

- Introduction and background
  - Anonymous dynamic networks with a Leader
  - Previous work
- Preliminaries
  - Multi-aggregate functions
  - The Counting Problem is “complete”
  - Lower bound on computation time
- Computing in linear time
  - History trees
  - Estimating anonymities
  - Bounding stabilization time

Static networks

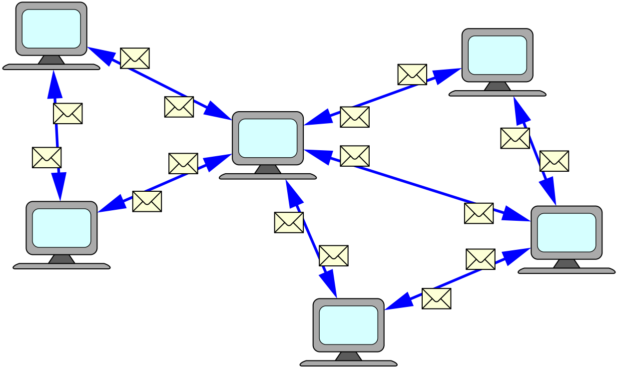
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Static networks

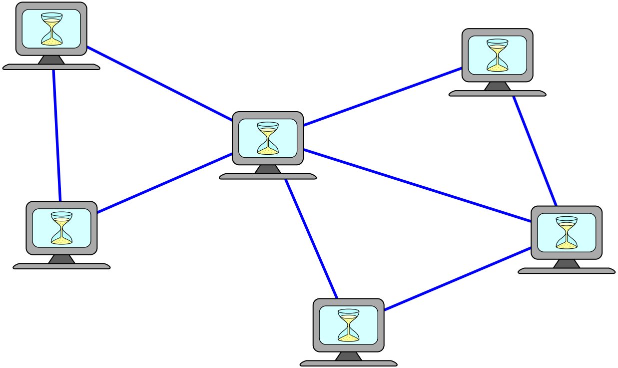
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Static networks

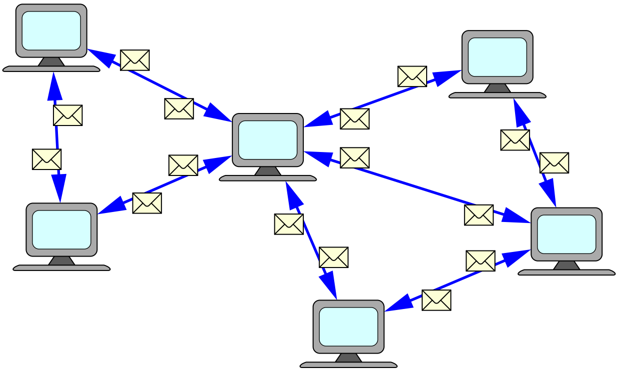
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Static networks

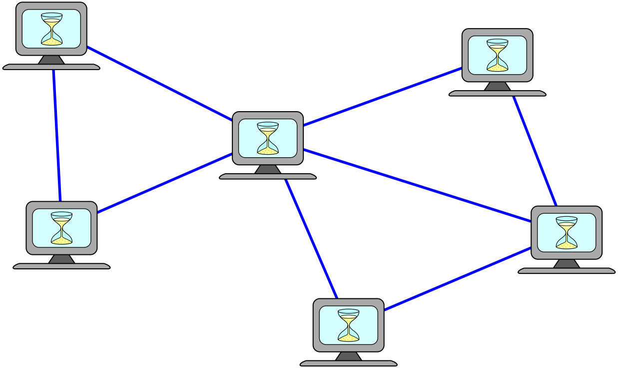
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Static networks

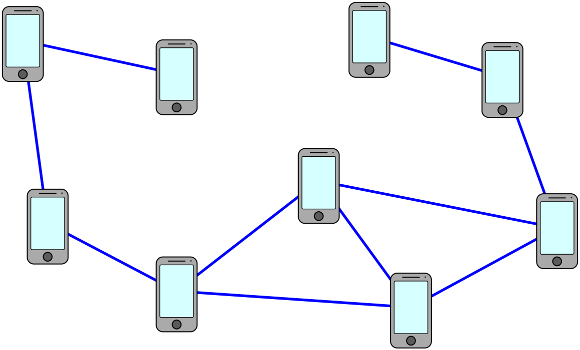
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Dynamic networks

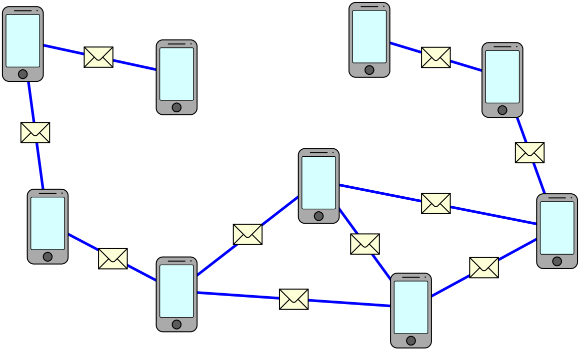
A *dynamic network* works in the same way, except that the links between machines (or agents) may change over time.



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

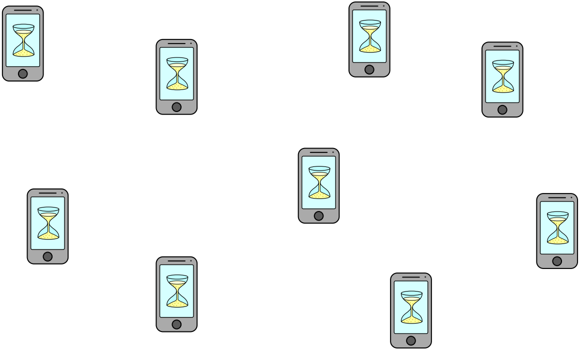
A *dynamic network* works in the same way, except that the links between machines (or agents) may change over time.



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

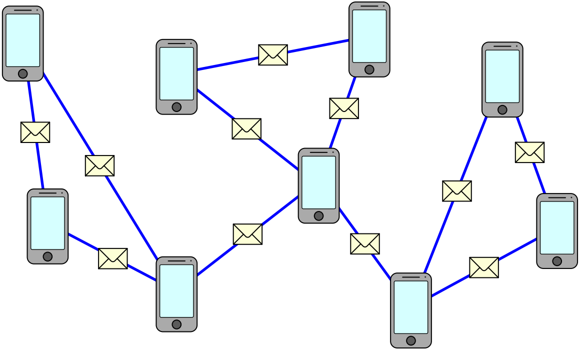
A *dynamic network* works in the same way, except that the links between machines (or agents) may change over time.



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

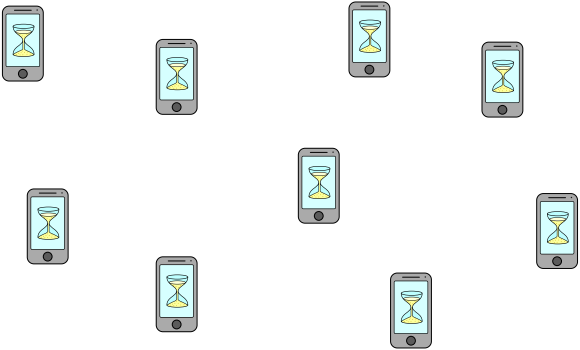
A *dynamic network* works in the same way, except that the links between machines (or agents) may change over time.



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

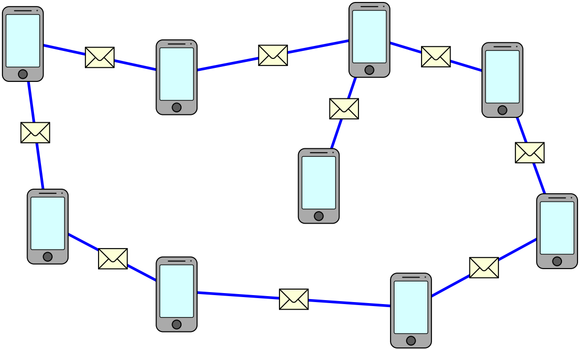
A *dynamic network* works in the same way, except that the links between machines (or agents) may change over time.



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

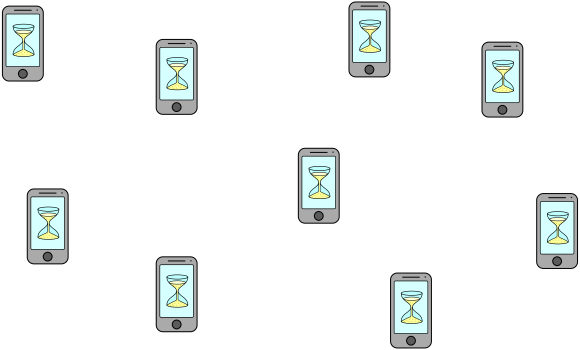
A *dynamic network* works in the same way, except that the links between machines (or agents) may change over time.



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

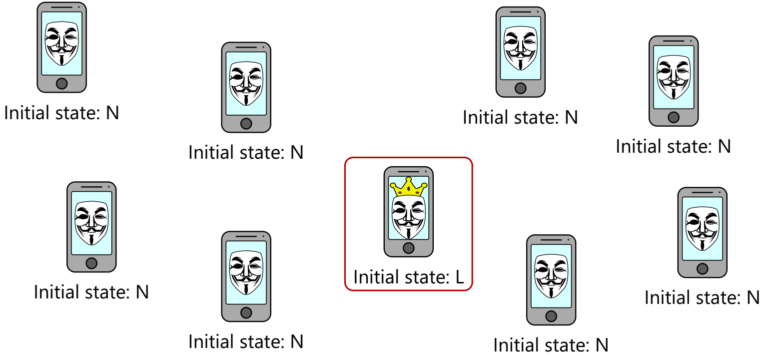
A *dynamic network* works in the same way, except that the links between machines (or agents) may change over time.



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

We assume the dynamic network to be *anonymous*, i.e., all agents start with the same internal state, except one: the *Leader*.

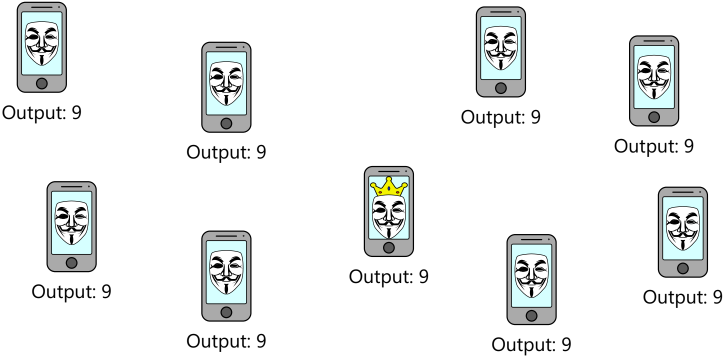


Counting Problem: Eventually, all agents must know the total number of agents, n. Is it possible? In how many rounds at most?

Note: Knowing n allows agents to solve a large class of problems.

Counting anonymous agents with a Leader

We assume the dynamic network to be *anonymous*, i.e., all agents start with the same internal state, except one: the *Leader*.



Counting Problem: Eventually, all agents must know the total number of agents, n. Is it possible? In how many rounds at most?

Note: Knowing n allows agents to solve a large class of problems.

Previous work

Theorem (Michail et al., SSS 2013)

In a static anonymous network,

1. *Without a Leader, counting processes is impossible.*

2. *With a unique Leader, counting can be done in* $\mathbf{2n}$ **rounds***.*

Conjecture. *Counting in a dynamic network is impossible even with a Leader.*

Theorem (Di Luna et al., ICDCN 2014)

In a dynamic anonymous network with a unique Leader, counting agents can be done in an **exponential number of rounds***, provided that an upper bound on* n *is known.*

Theorem (Di Luna–Baldoni, OPODIS 2015)

In a dynamic anonymous network with a unique Leader, counting agents can be done in an **exponential number of rounds***.*

Theorem (Kowalski–Mosteiro, ICALP 2018, Best Paper Award)

In a dynamic anonymous network with a unique Leader, counting agents can be done in $\mathbf{O(n^4 \log^3 n)}$ **rounds***.*

Previous work

Theorem (Michail et al., SSS 2013)

In a static anonymous network,

1. *Without a Leader, counting processes is impossible.*

2. *With a unique Leader, counting can be done in* $\mathbf{2n}$ **rounds**.

Conjecture. *Counting in a dynamic network is impossible even with a Leader.*

Theorem (Di Luna et al., ICDCN 2014)

In a dynamic anonymous network with a unique Leader, counting agents can be done in an **exponential number of rounds**, *provided that an upper bound on* n *is known.*

Theorem (Di Luna–Baldoni, OPODIS 2015)

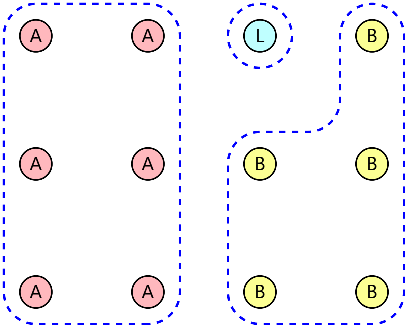
In a dynamic anonymous network with a unique Leader, counting agents can be done in an **exponential number of rounds**.

Theorem (Kowalski–Mosteiro, ICALP 2018, Best Paper Award)

In a dynamic anonymous network with a unique Leader, counting agents can be done in $\mathbf{O}(\mathbf{n}^4 \log^3 \mathbf{n})$ **rounds**. (Can we improve upon this?)

General computation

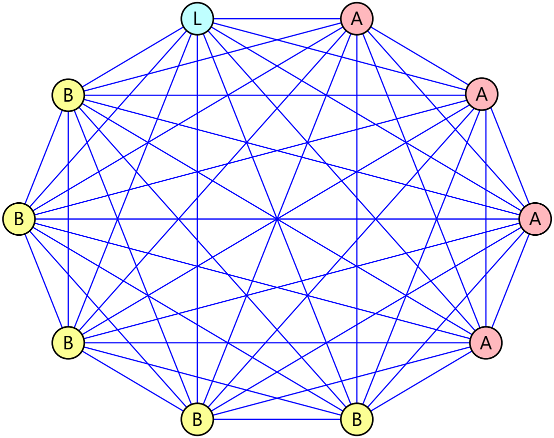
In general, we may assume that each agent has an *input* and has to compute an *output* depending on the entire network's inputs.



Agents with the same input are still indistinguishable (anonymous).

General computation

If the network is the complete graph at every round, all agents with the same input will always have the same internal state.



Thus, an agent's output can only depend on its input and the *number* of agents having each input.

Completeness of the Counting Problem

We call such functions *multi-aggregate* functions.

Observation

If a function is computable in an anonymous dynamic network (with a unique Leader), it must be a multi-aggregate function.

Examples: The average, maximum, minimum, sum, mode, variance, and most statistical functions are (multi-)aggregate.

Generalized Counting Problem: Eventually, all agents must know how many agents have each input.

Observation

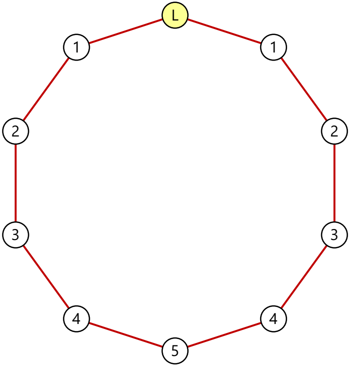
If the Generalized Counting Problem is solvable in $f(n)$ rounds, then every multi-aggregate function is computable in $f(n)$ rounds.

Lower bound

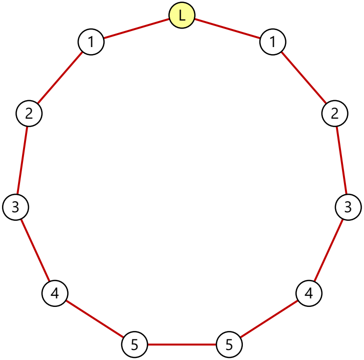
Theorem

No algorithm can solve the Counting Problem in an anonymous dynamic network of n agents in less than $1.5n - 2$ rounds.

Rounds 1 to 4



System 1



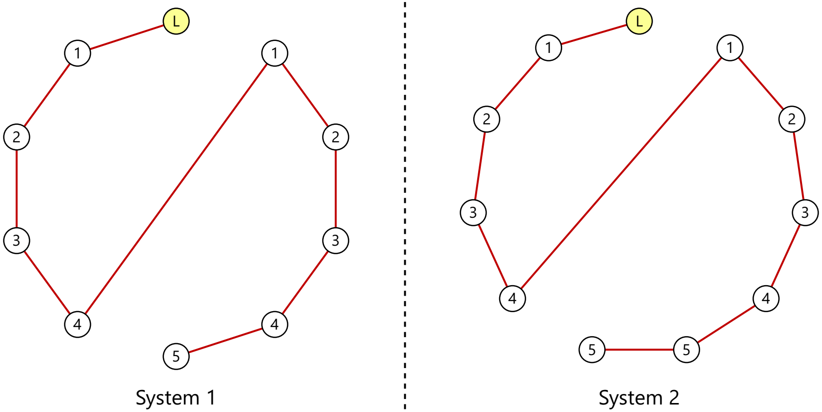
System 2

Lower bound

Theorem

No algorithm can solve the Counting Problem in an anonymous dynamic network of n agents in less than $1.5n - 2$ rounds.

Rounds 5 to 13

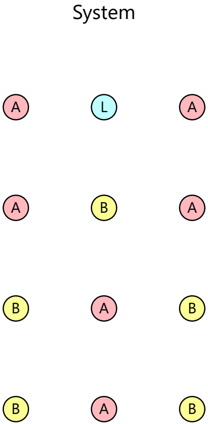


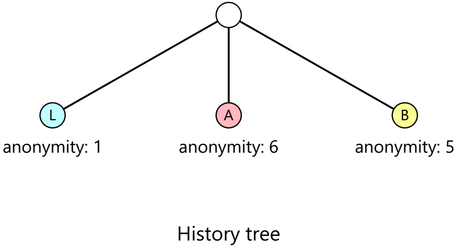
System 1

System 2

History tree

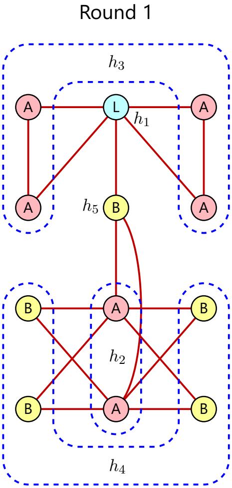
We introduce the *history tree* as a tool for studying dynamic networks.

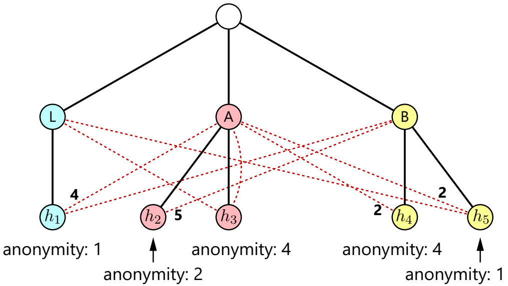




History tree

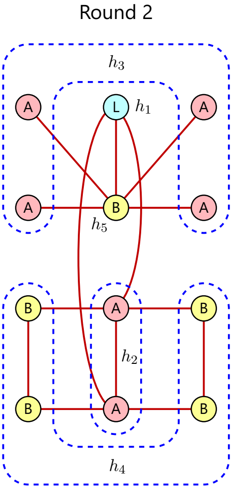
We introduce the *history tree* as a tool for studying dynamic networks.

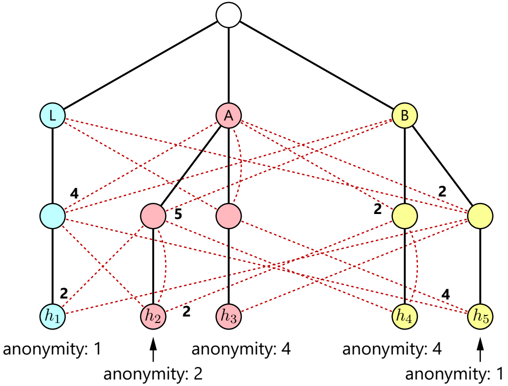




History tree

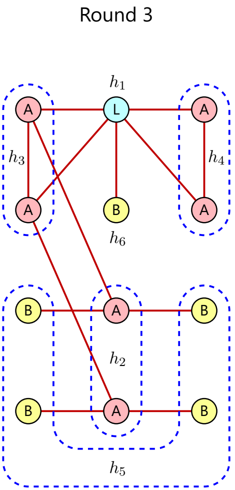
We introduce the *history tree* as a tool for studying dynamic networks.

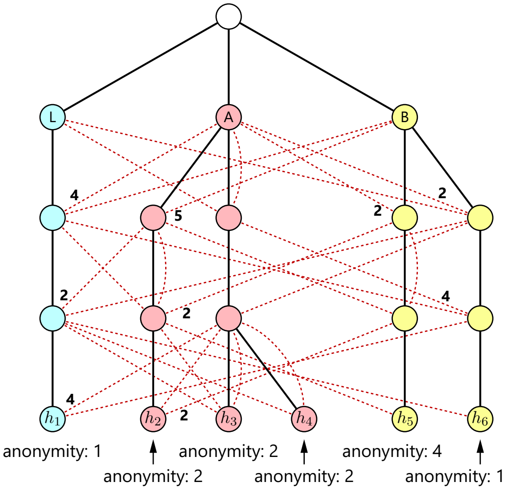




History tree

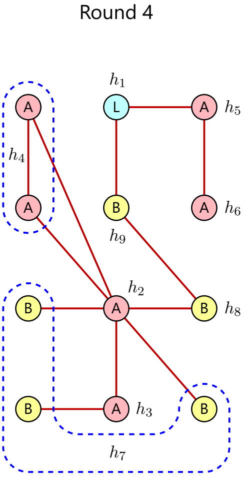
We introduce the *history tree* as a tool for studying dynamic networks.

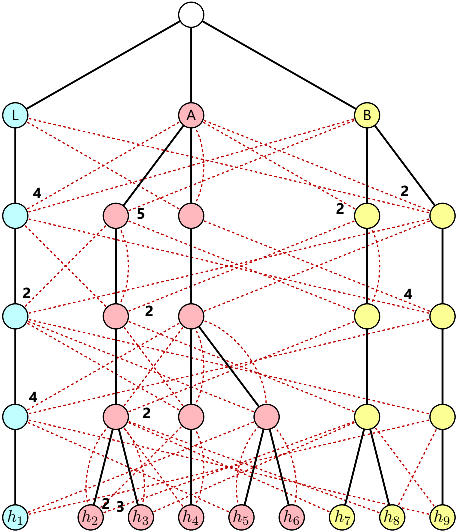




History tree

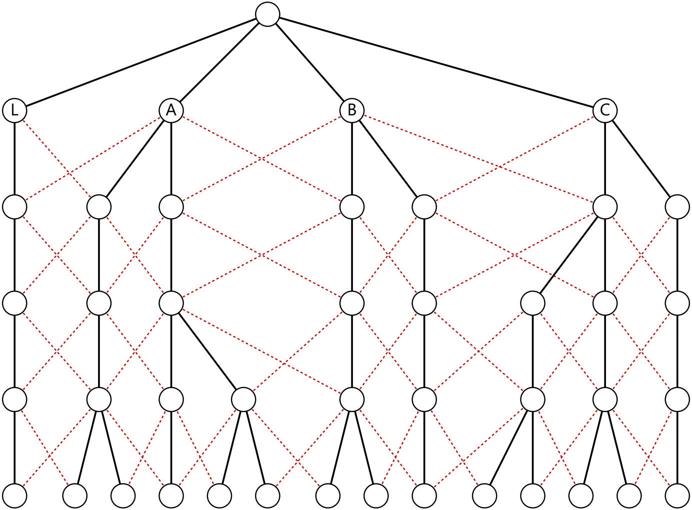
We introduce the *history tree* as a tool for studying dynamic networks.





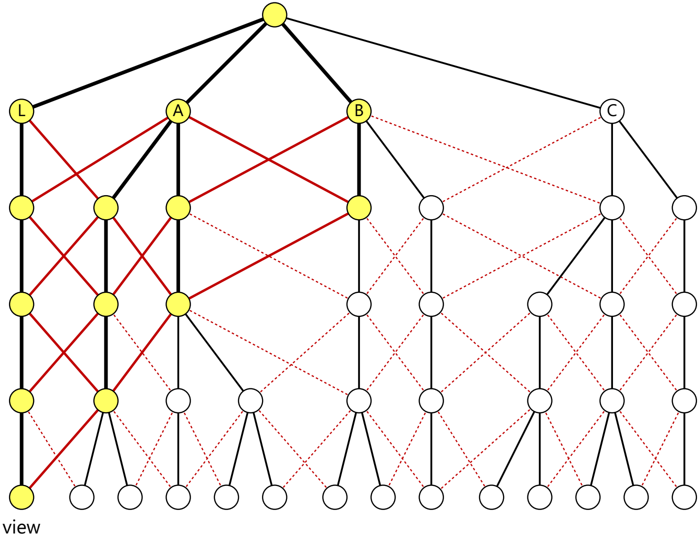
View of a history tree

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



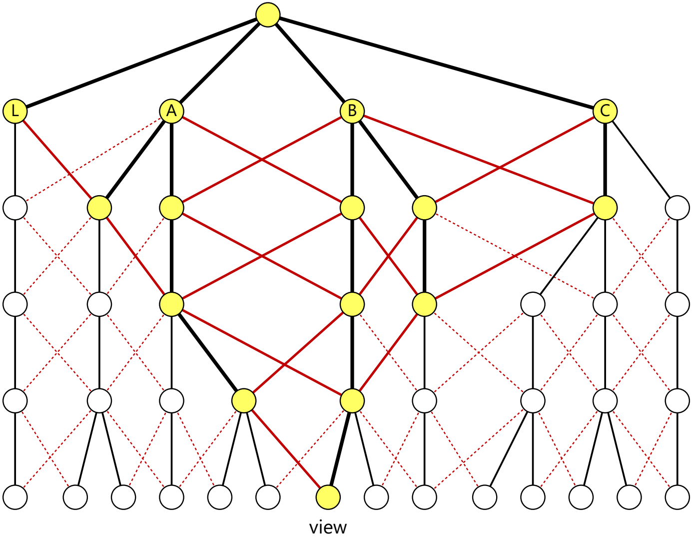
View of a history tree

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



View of a history tree

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



Views as internal states and messages

An agent's view summarizes its whole *history* up to some round.

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.

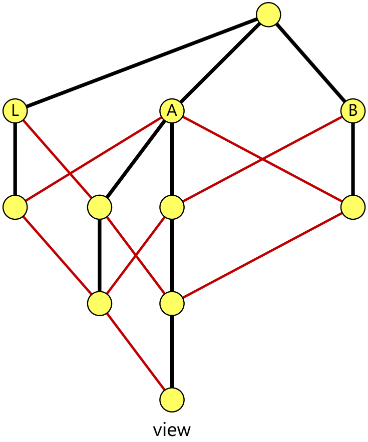
This is good because, at round i, the size of a view is only $O(i^4)$.

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.

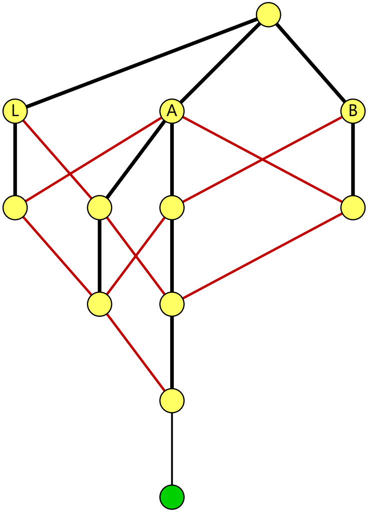
Updating the view

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



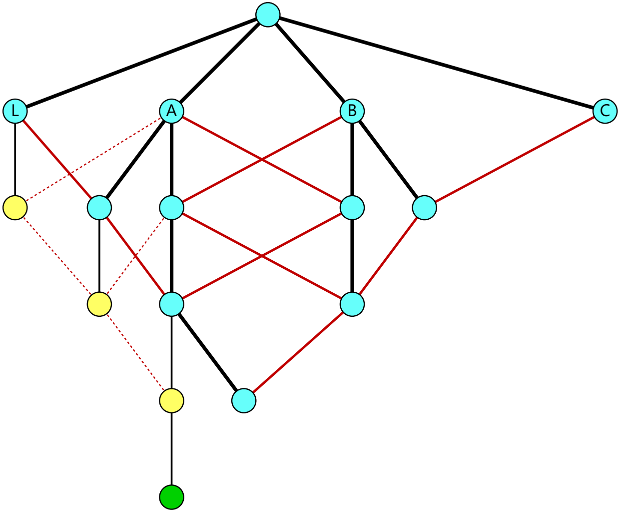
Updating the view

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



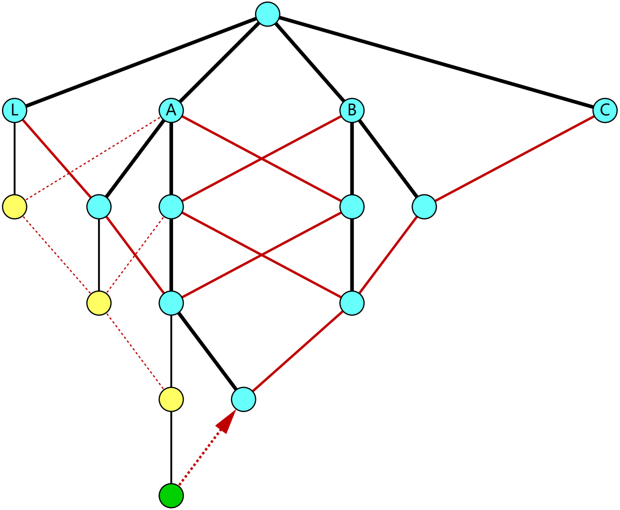
Updating the view

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



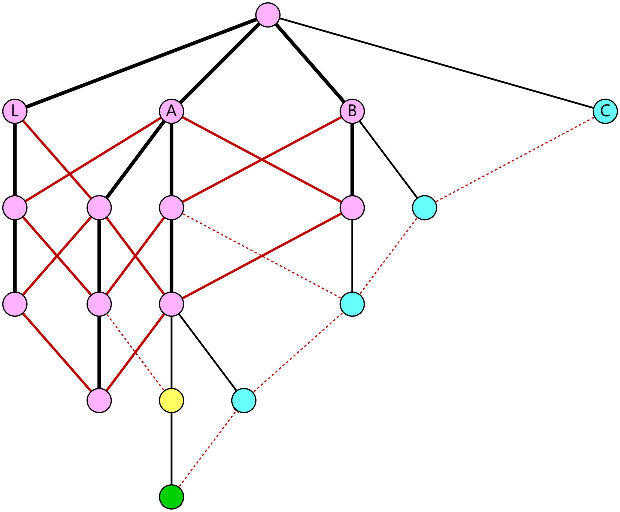
Updating the view

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



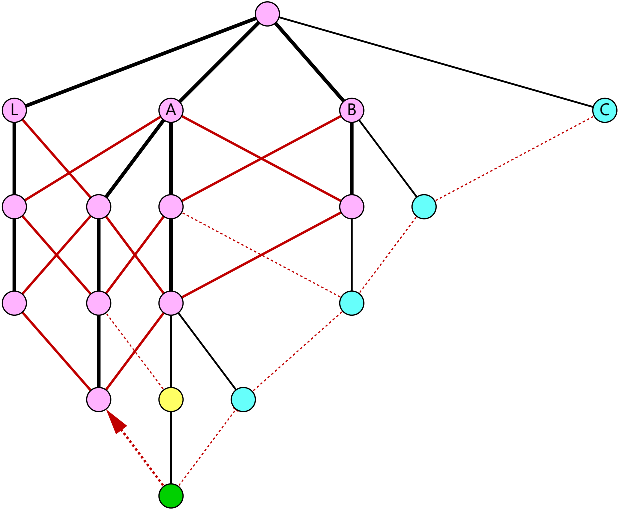
Updating the view

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



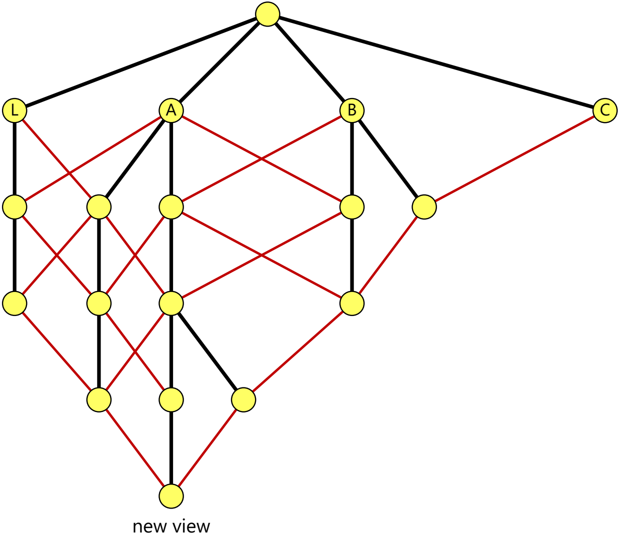
Updating the view

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



Updating the view

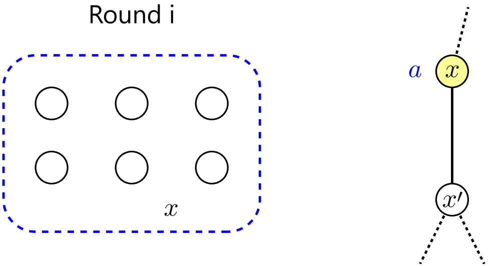
An agent updates its internal state by *merging* its view with the views it receives from its neighbors.



Computing anonymities

Suppose we know the anonymity of a node x in the history tree.

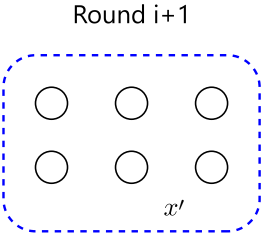
If x has only one child x', then x' must have the same anonymity.

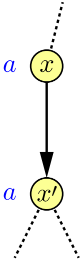


Computing anonymities

Suppose we know the anonymity of a node x in the history tree.

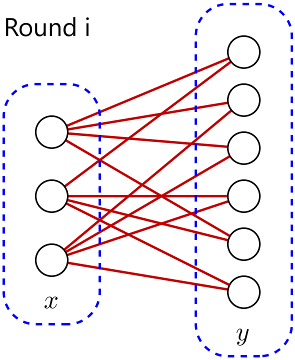
If x has only one child x', then x' must have the same anonymity.

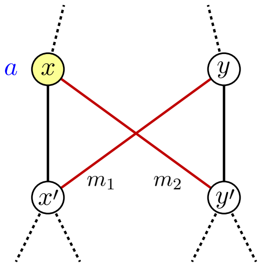




Computing anonymities

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

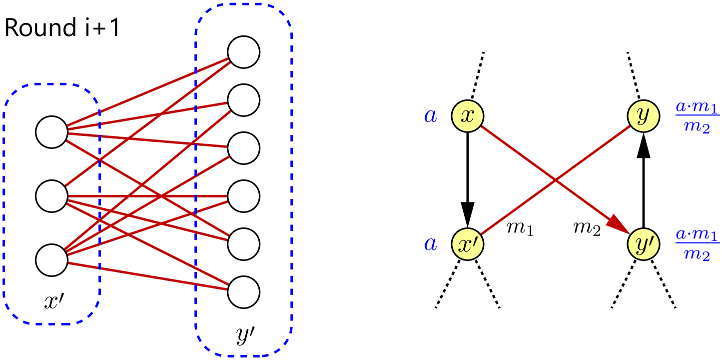




If the agents represented by x have observed agents whose corresponding node y has only one child y', then we can compute the anonymity of y and y', as well.

Computing anonymities

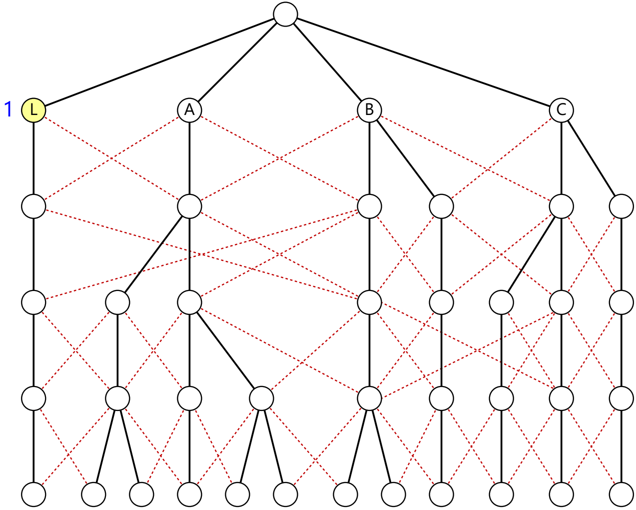
Suppose we know the anonymity of a node x with a single child x'.



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.

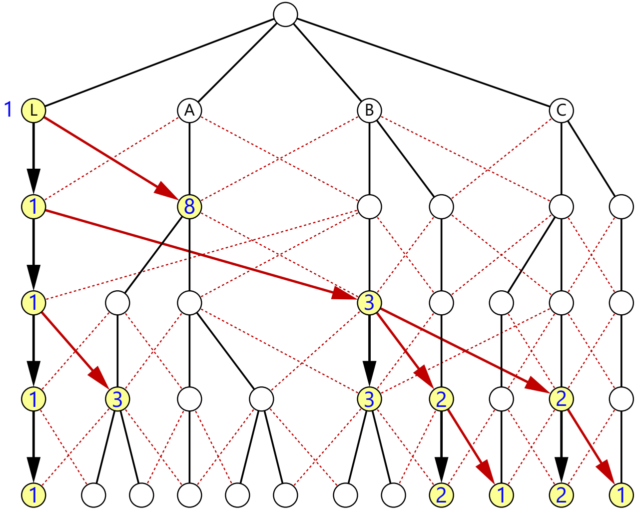
Computing anonymities

By starting from the Leader's nodes and applying these rules on the history tree, we can eventually solve the Generalized Counting Problem.



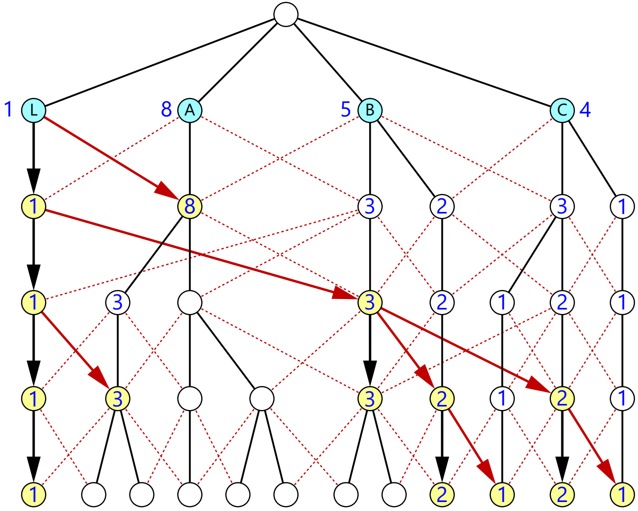
Computing anonymities

By starting from the Leader's nodes and applying these rules on the history tree, we can eventually solve the Generalized Counting Problem.



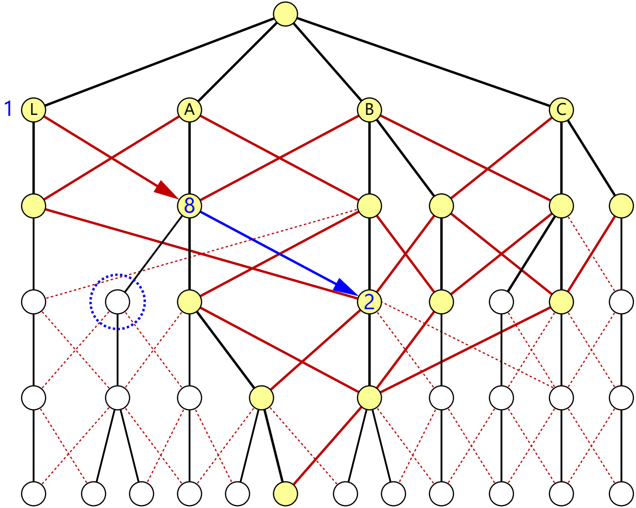
Computing anonymities

By starting from the Leader's nodes and applying these rules on the history tree, we can eventually solve the Generalized Counting Problem.



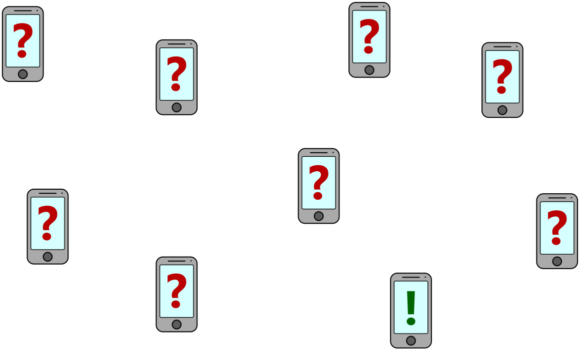
Computing anonymities

However, an agent's view is incomplete, and this may lead to incorrect computations. Are computations *eventually* correct?



Propagation of information

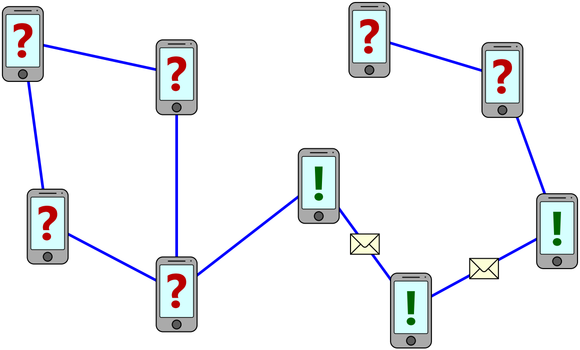
If the network is connected at all rounds, every news reaches every agent in at most n rounds (where n is the number of agents).



Hence, whenever two agents interact, all agents will know it within n rounds (and it will show in their views of the history tree).

Propagation of information

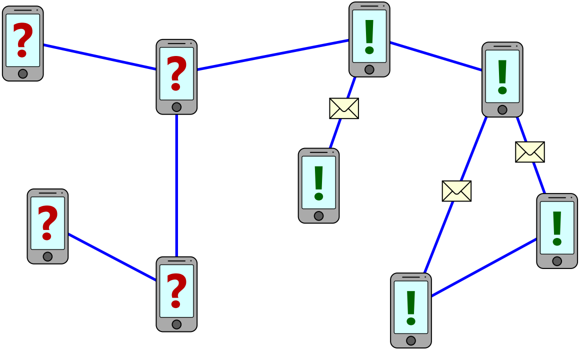
If the network is connected at all rounds, every news reaches every agent in at most n rounds (where n is the number of agents).



Hence, whenever two agents interact, all agents will know it within n rounds (and it will show in their views of the history tree).

Propagation of information

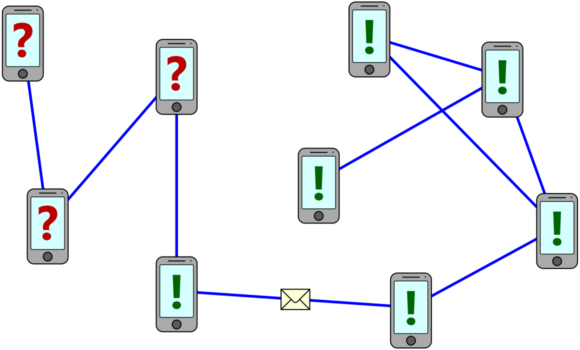
If the network is connected at all rounds, every news reaches every agent in at most n rounds (where n is the number of agents).



Hence, whenever two agents interact, all agents will know it within n rounds (and it will show in their views of the history tree).

Propagation of information

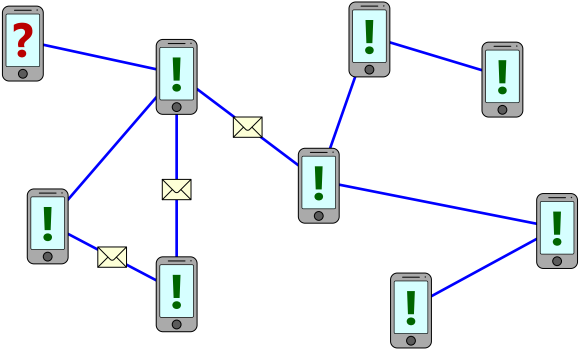
If the network is connected at all rounds, every news reaches every agent in at most n rounds (where n is the number of agents).



Hence, whenever two agents interact, all agents will know it within n rounds (and it will show in their views of the history tree).

Propagation of information

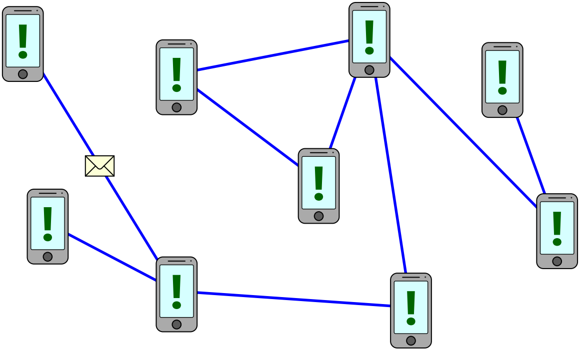
If the network is connected at all rounds, every news reaches every agent in at most n rounds (where n is the number of agents).



Hence, whenever two agents interact, all agents will know it within n rounds (and it will show in their views of the history tree).

Propagation of information

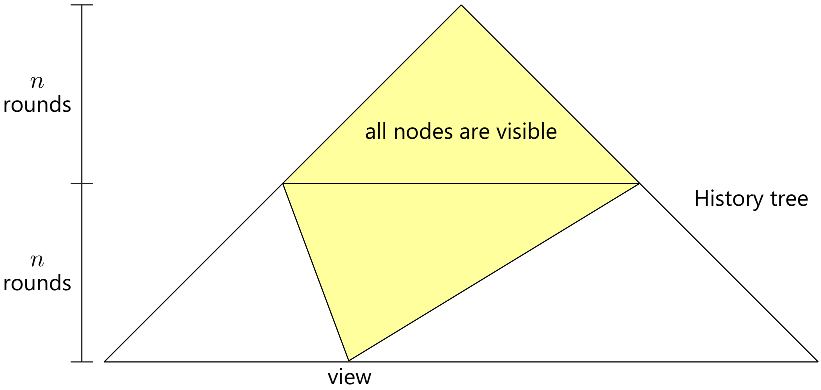
If the network is connected at all rounds, every news reaches every agent in at most n rounds (where n is the number of agents).



Hence, whenever two agents interact, all agents will know it within n rounds (and it will show in their views of the history tree).

Output stabilization

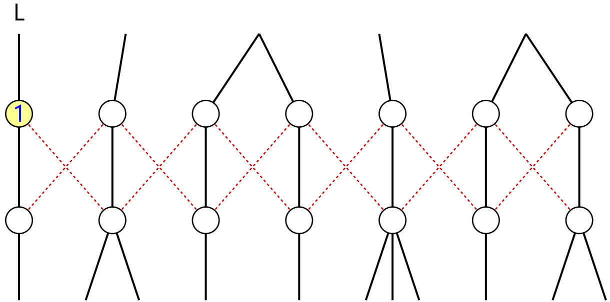
Claim: The correct output is *stably* computed after $2n$ rounds.



Note that, after $2n$ rounds, all nodes in the first n levels of the history tree are in the views of all agents.

Output stabilization

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

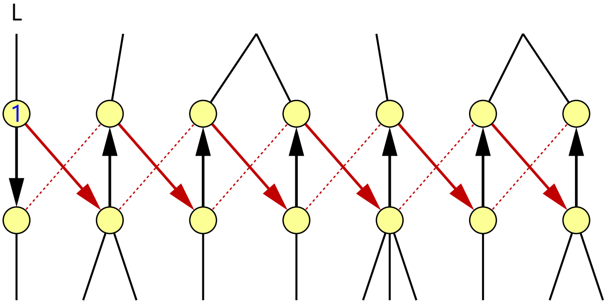


Since there are only n agents, the tree can branch at most n times.

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

Output stabilization

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 only n agents, the tree can branch at most n times.

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

Conclusions

Theorem

The Generalized Counting Problem can be solved in $2n$ rounds in a connected anonymous dynamic network with a Leader, and is not solvable in less that $1.5n - 2$ rounds.

Corollary

Any problem that is solvable in a connected anonymous dynamic network with a Leader can be solved in $2n$ rounds.

Additionally, internal states and messages have size $O(n^4)$.

Open Problem: Note that agents' outputs only *stabilize* on the correct result. Is there a way for all agents to *know* when they have solved the problem and *terminate* in $O(n)$ rounds?

Open Problem: What if the Leader is not unique, but there are ℓ indistinguishable Leaders (where ℓ is known by all agents)?