Counting in Dynamic Anonymous Networks

Giovanni Viglietta

Joint work with Giuseppe A. Di Luna

(Work in progress...)

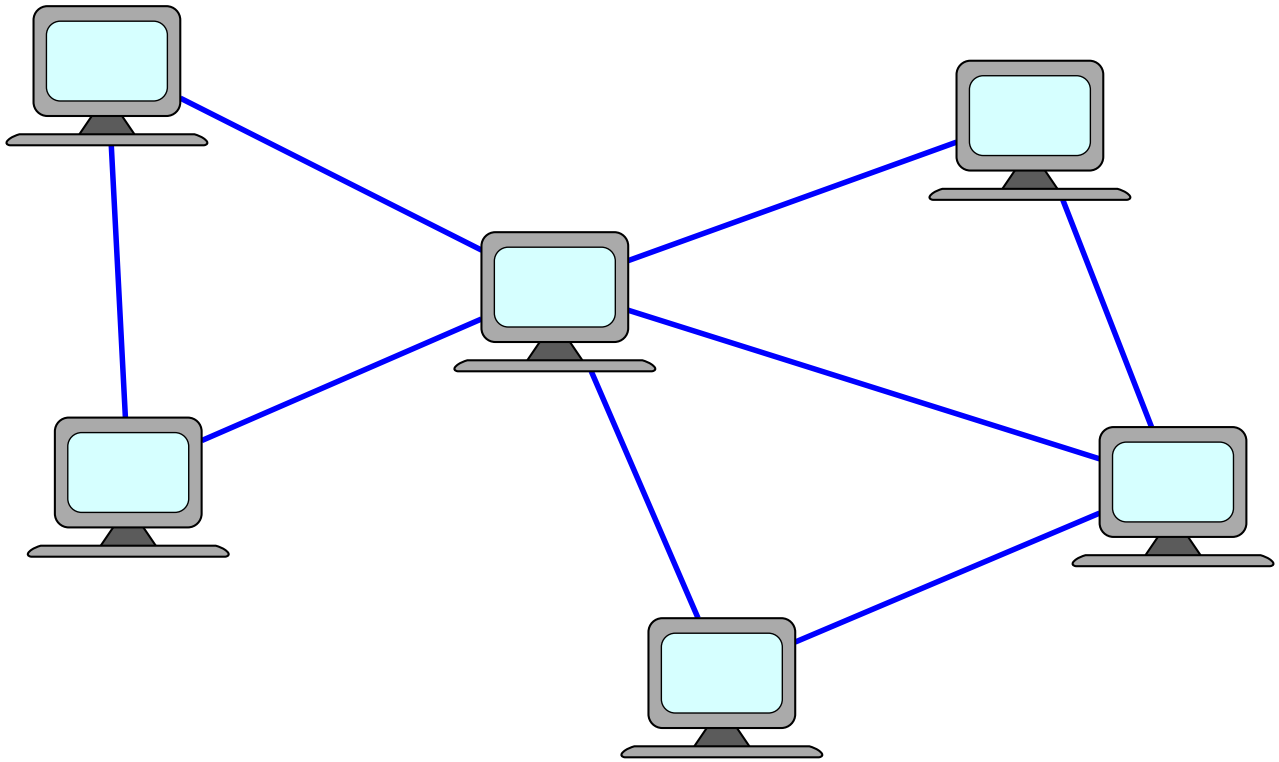
JAIST – October 4, 2021

Overview

- Problem introduction and background
  - Static vs. Dynamic networks
  - Counting anonymous agents with a unique Leader
  - Previous work
- Special case: agents communicating with the Leader
  - History tree
  - Bounding guesses
- General case
  - Generalized history tree
  - Bounding general guesses *(work in progress)*

Static networks

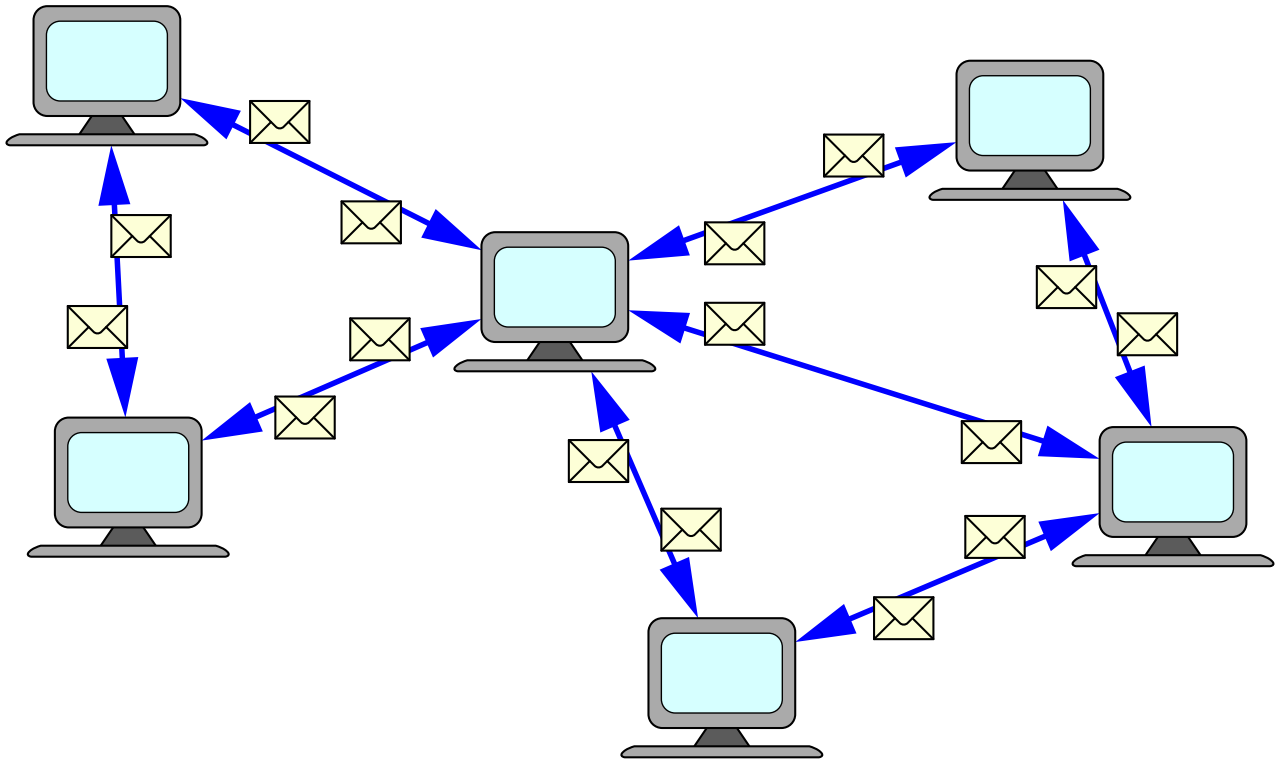
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Static networks

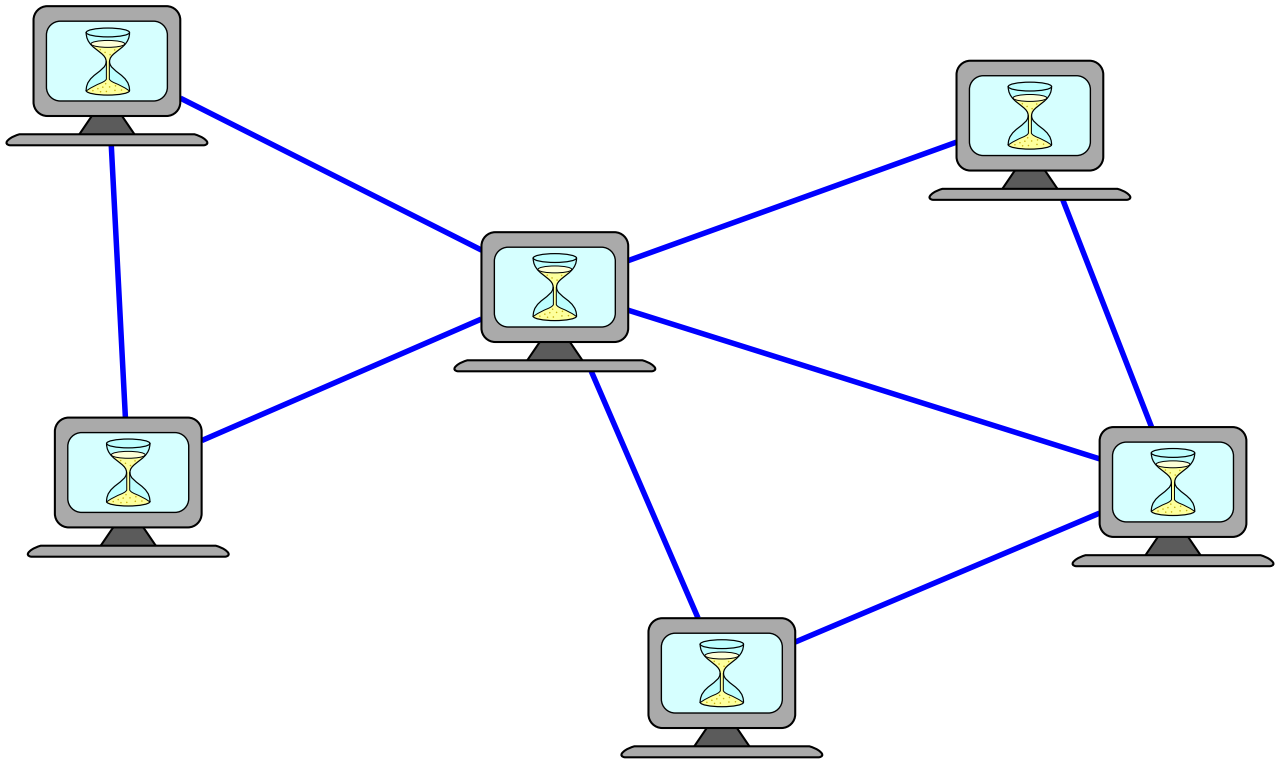
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Static networks

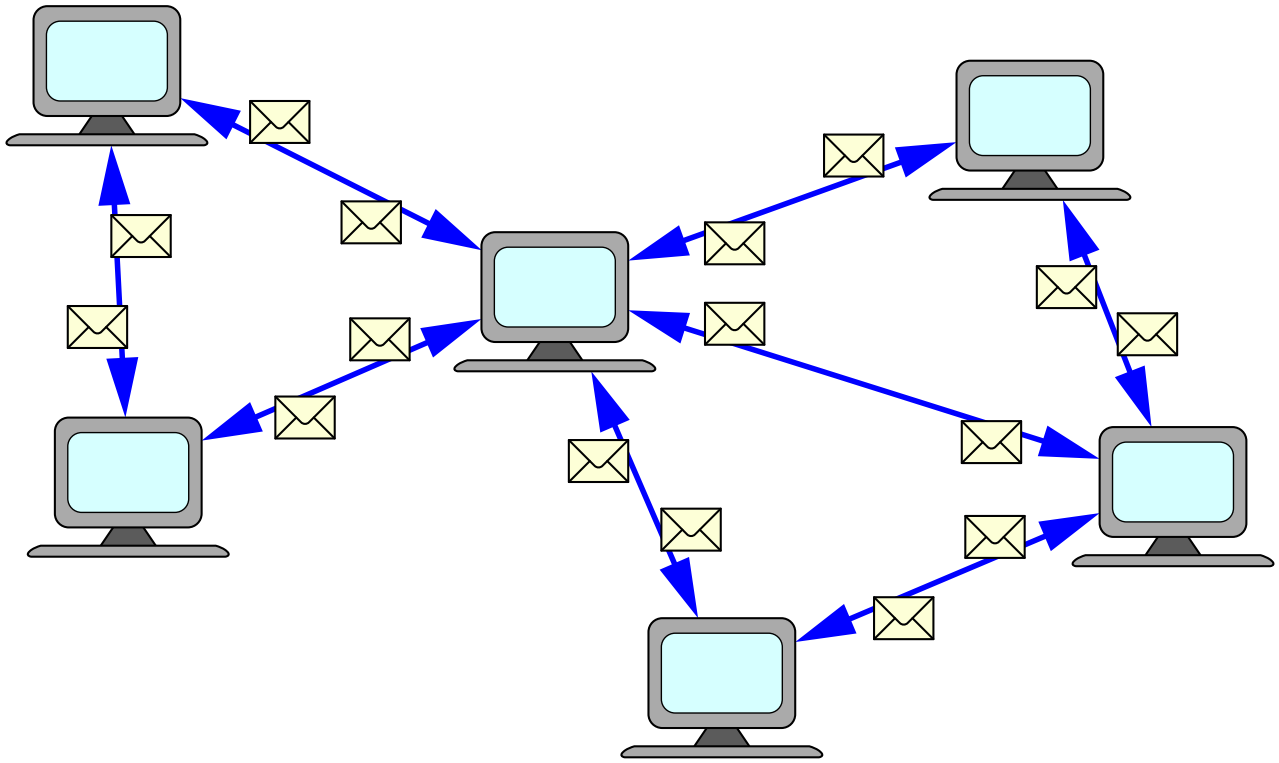
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Static networks

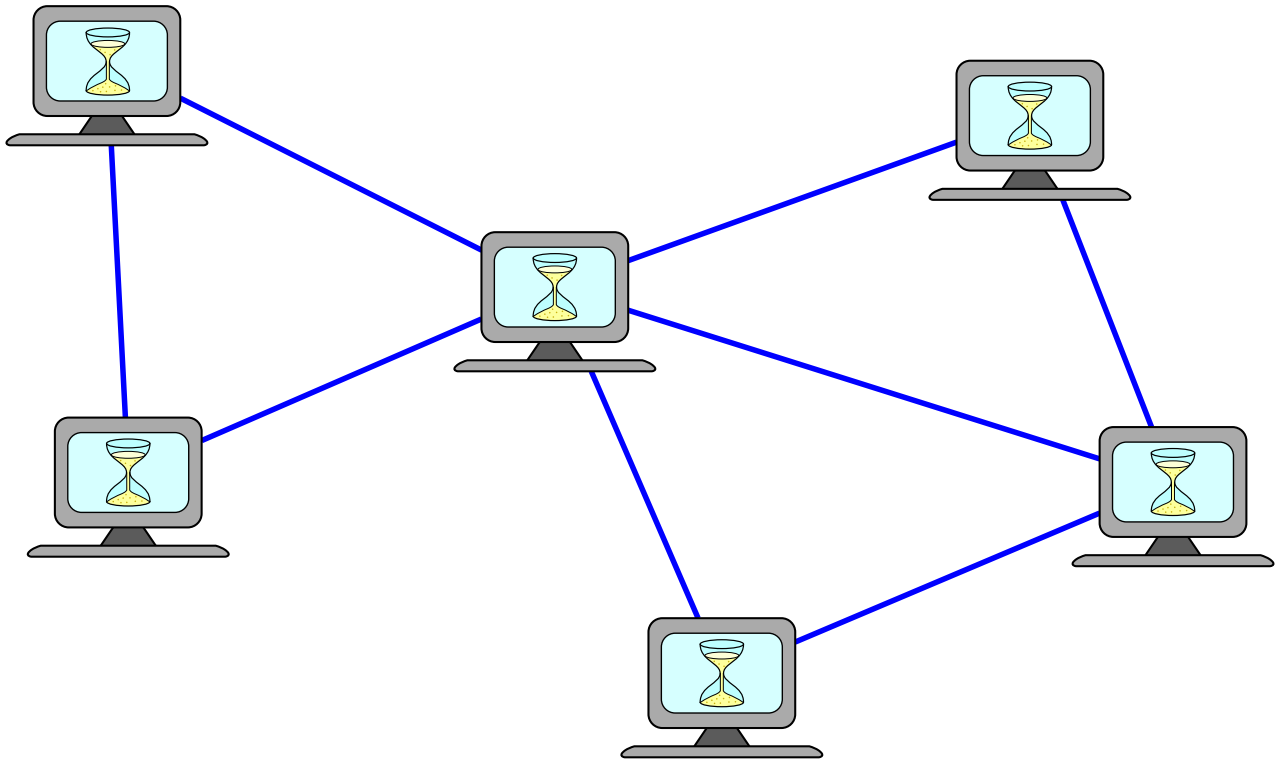
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Static networks

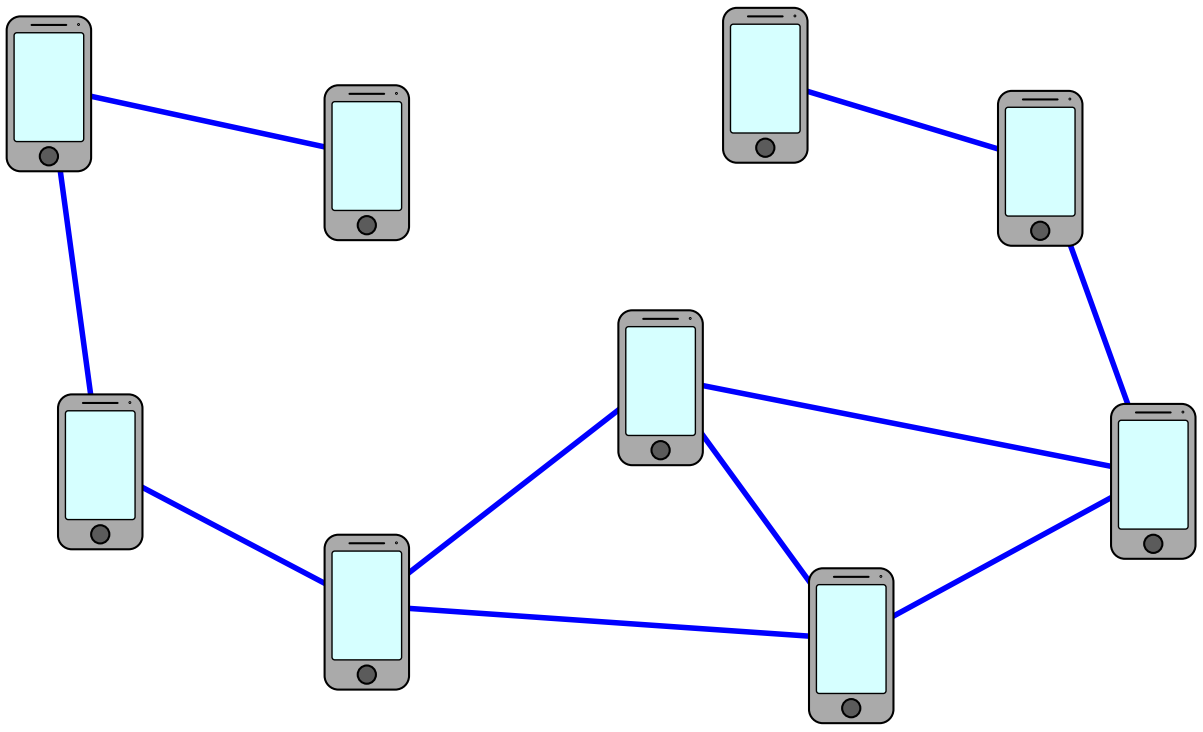
In a *static network*, some machines (or processes) are connected with each other through permanent links.



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

Dynamic networks

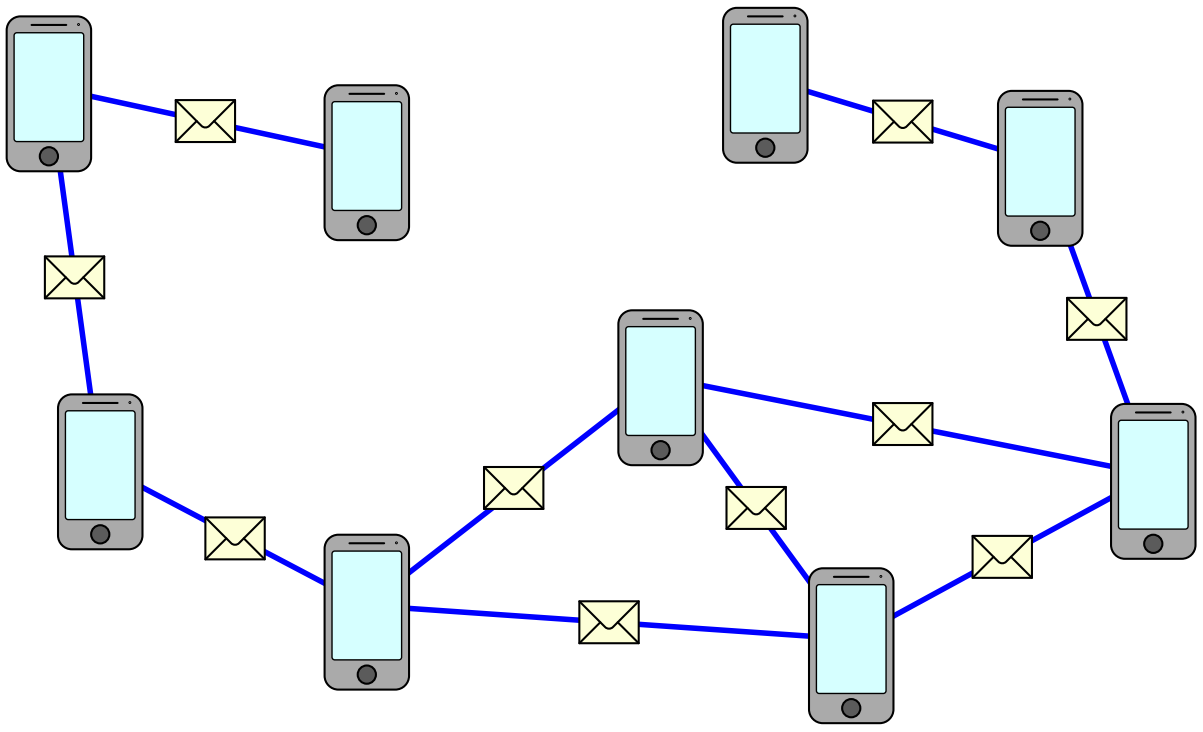
A *dynamic network* works in the same way, except that the links between machines (or agents) may change with time.



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

Dynamic networks

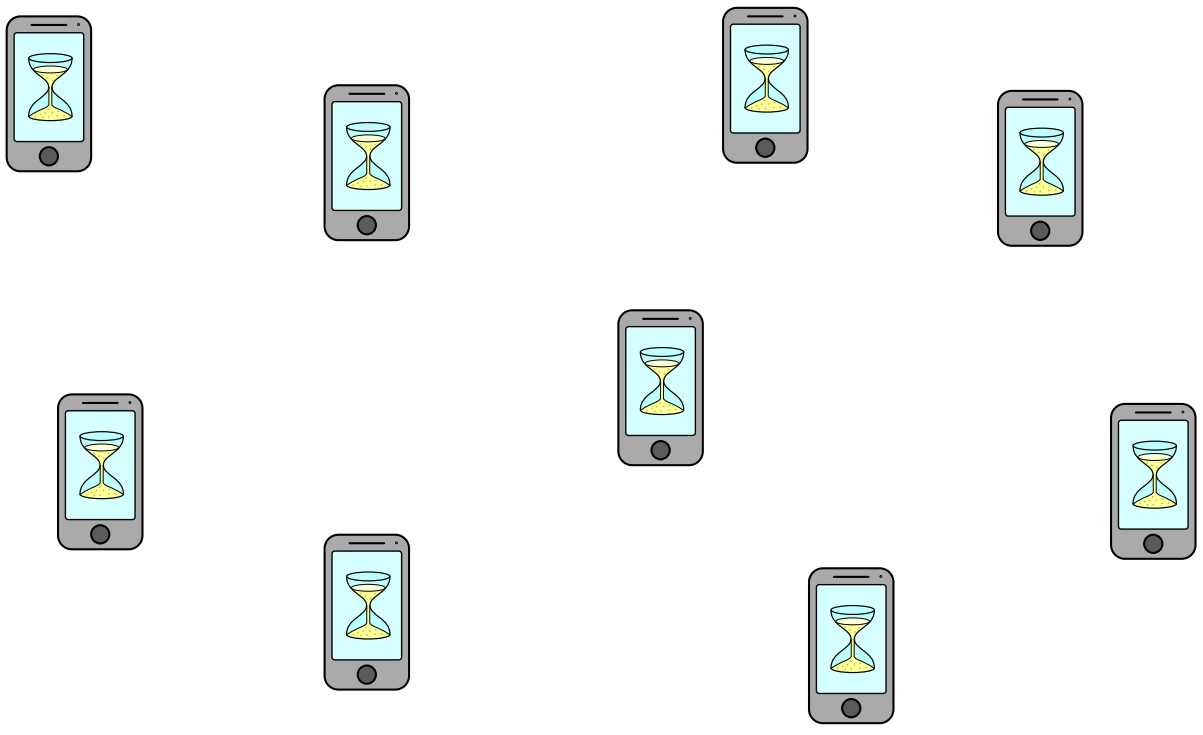
A *dynamic network* works in the same way, except that the links between machines (or agents) may change with time.



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

Dynamic networks

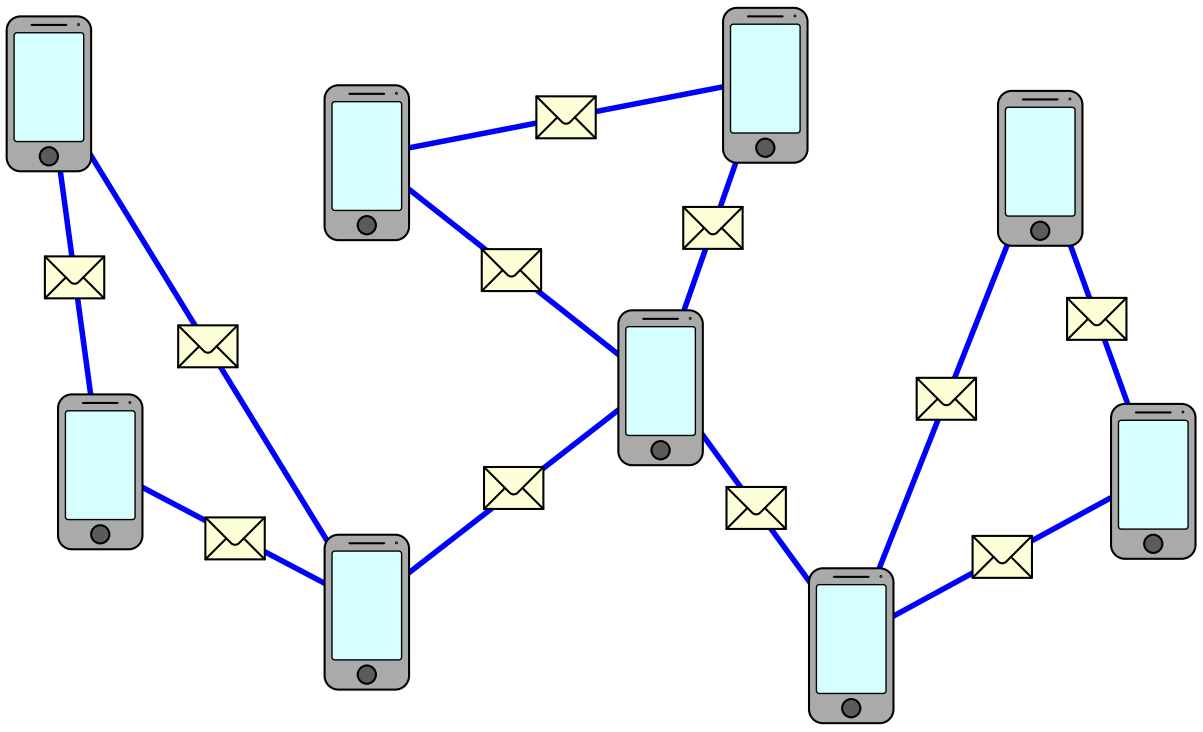
A *dynamic network* works in the same way, except that the links between machines (or agents) may change with time.



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

Dynamic networks

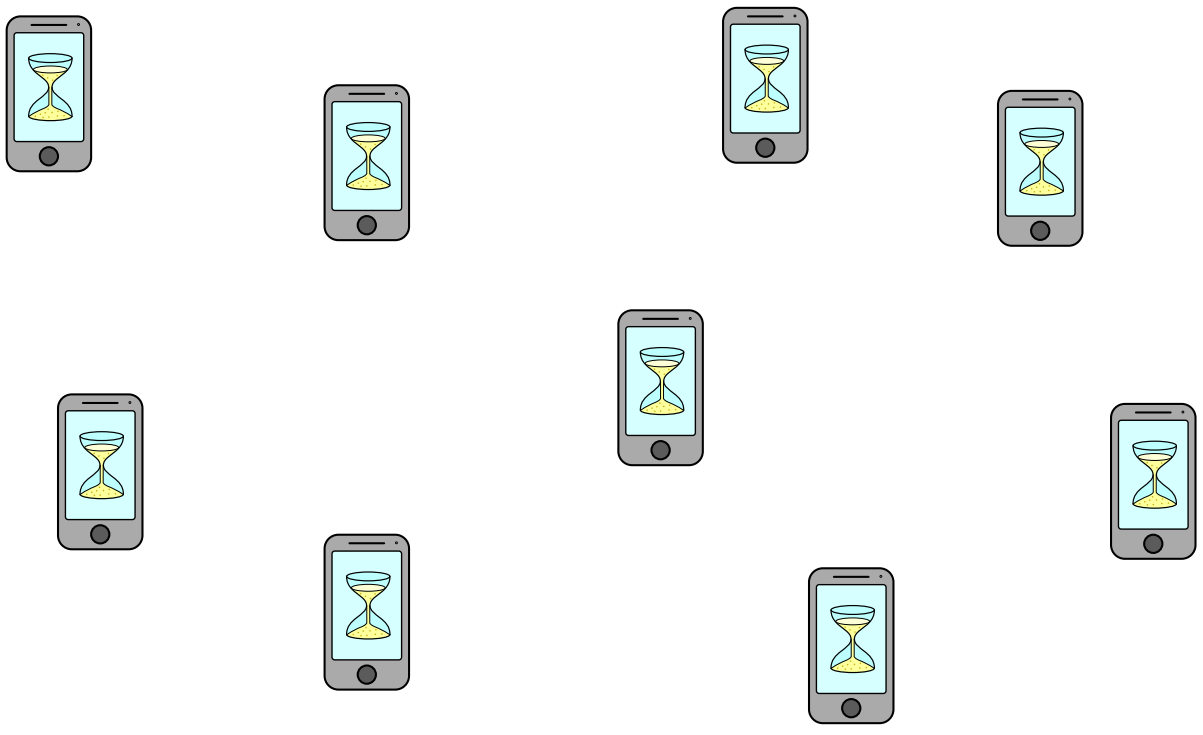
A *dynamic network* works in the same way, except that the links between machines (or agents) may change with time.



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

Dynamic networks

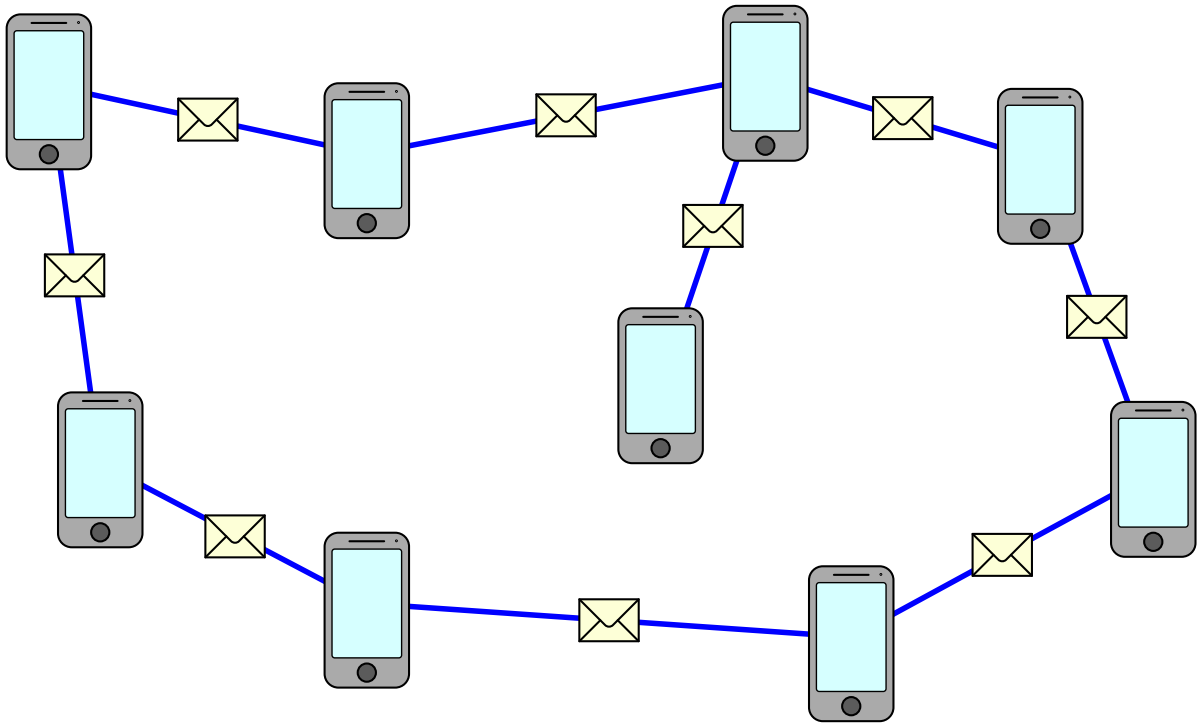
A *dynamic network* works in the same way, except that the links between machines (or agents) may change with time.



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

Dynamic networks

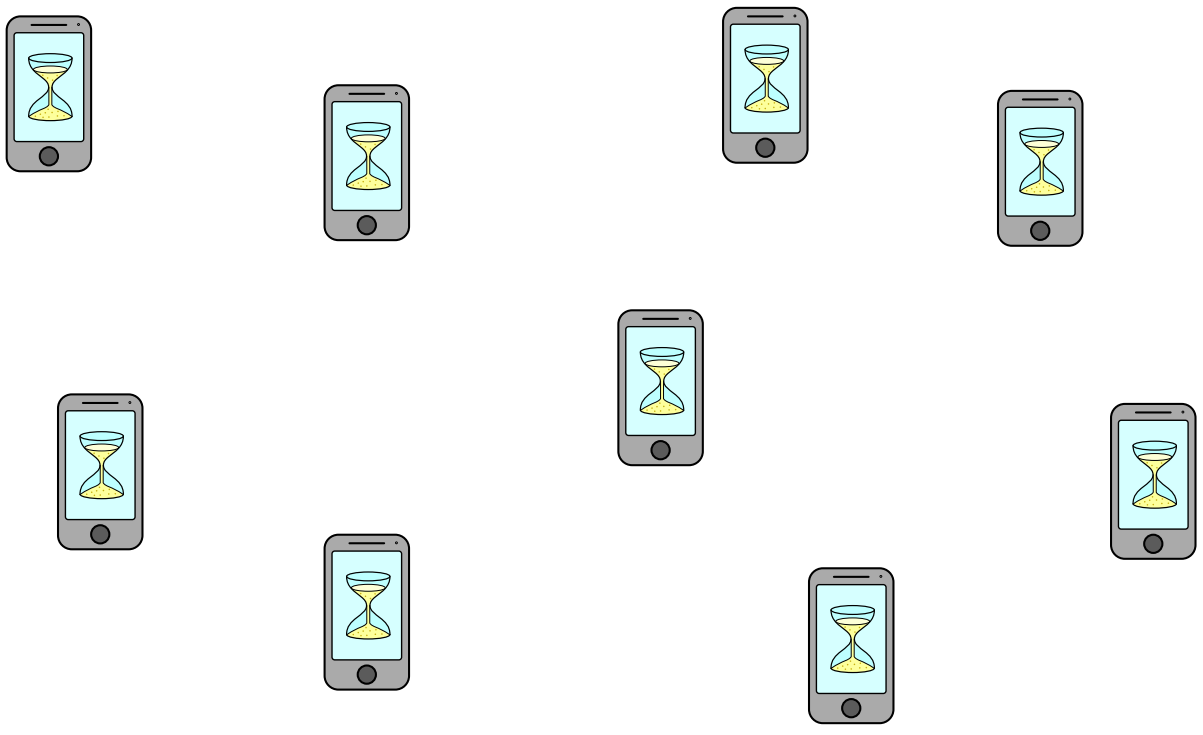
A *dynamic network* works in the same way, except that the links between machines (or agents) may change with time.



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

Dynamic networks

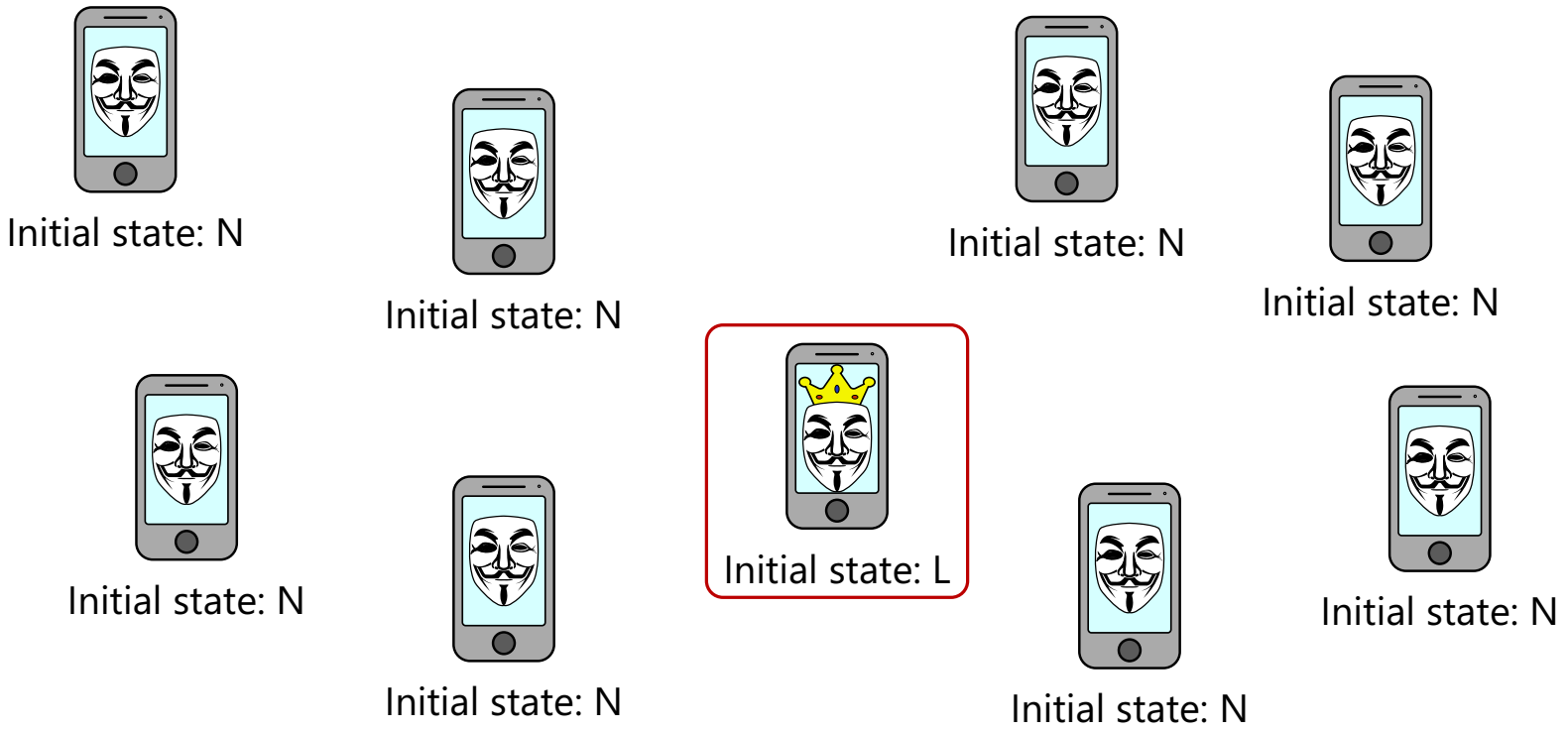
A *dynamic network* works in the same way, except that the links between machines (or agents) may change with time.



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

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*.

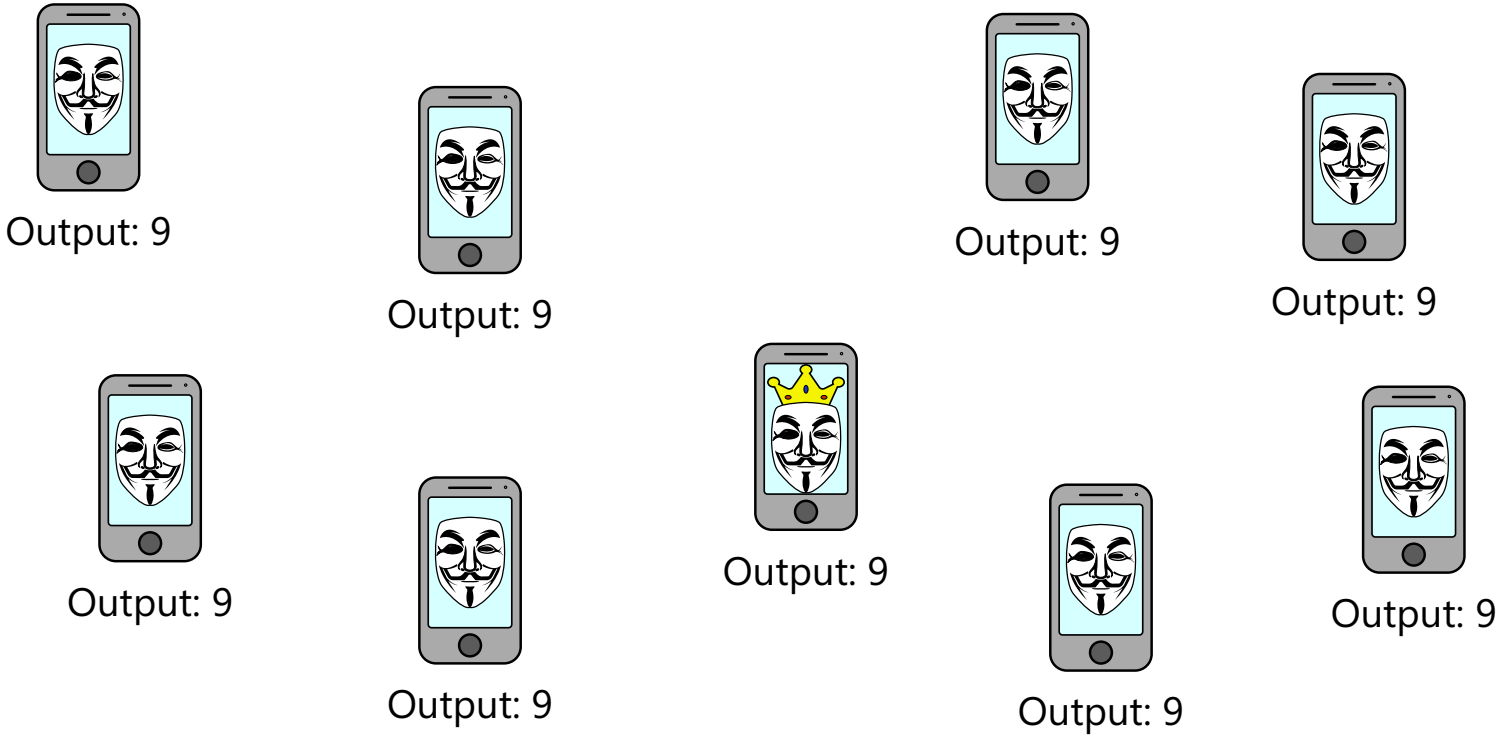


Counting Problem: Eventually, all agents must know the total number of agents, n. Is it possible? In how many steps at most?

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 steps 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)

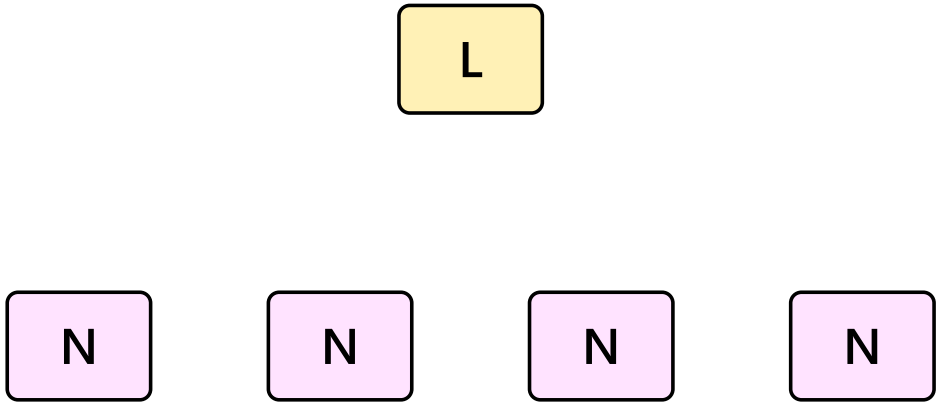
In a dynamic anonymous network with a unique Leader, counting agents can be done in an **exponential number of rounds**.

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

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

Memory and messages

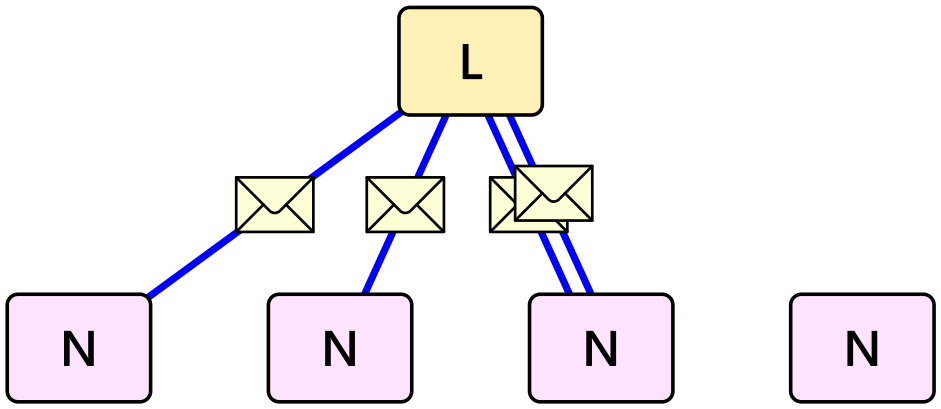
Each agent has unlimited memory and can remember everything it sees: its memory is its *state*. Also, when an agent sends a message, it sends its current state, i.e., the entire contents of its memory.



Leader's memory: empty

Memory and messages

Each agent has unlimited memory and can remember everything it sees: its memory is its *state*. Also, when an agent sends a message, it sends its current state, i.e., the entire contents of its memory.

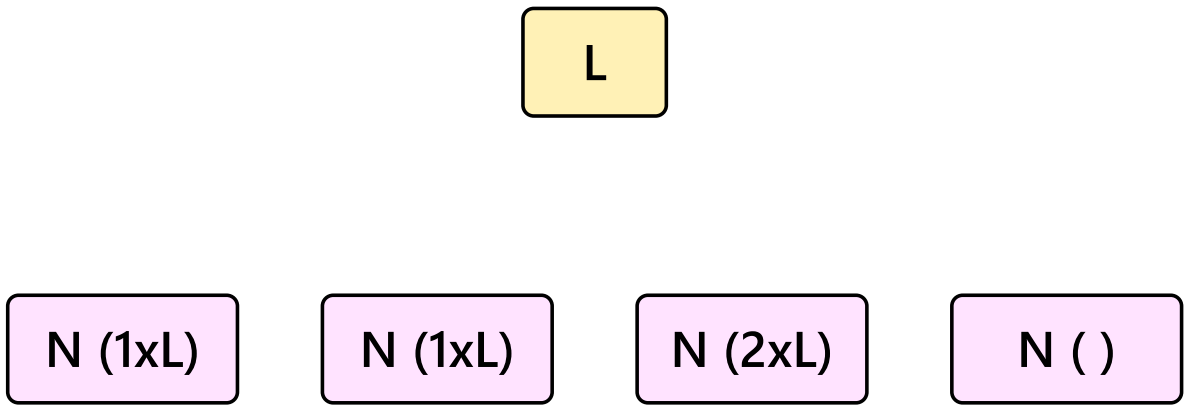


Leader's memory:

Step 1: **4xN**

Memory and messages

Each agent has unlimited memory and can remember everything it sees: its memory is its *state*. Also, when an agent sends a message, it sends its current state, i.e., the entire contents of its memory.

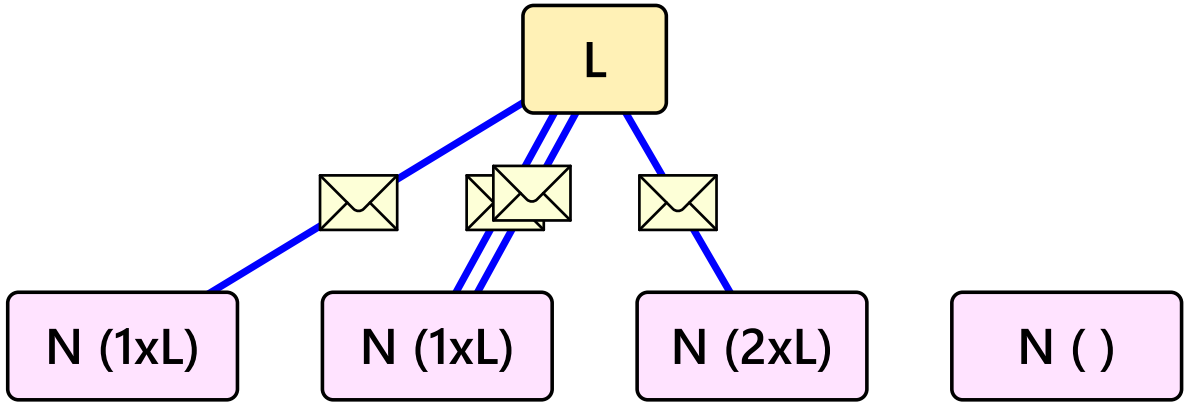


Leader's memory:

Step 1: **4xN**

Memory and messages

Each agent has unlimited memory and can remember everything it sees: its memory is its *state*. Also, when an agent sends a message, it sends its current state, i.e., the entire contents of its memory.

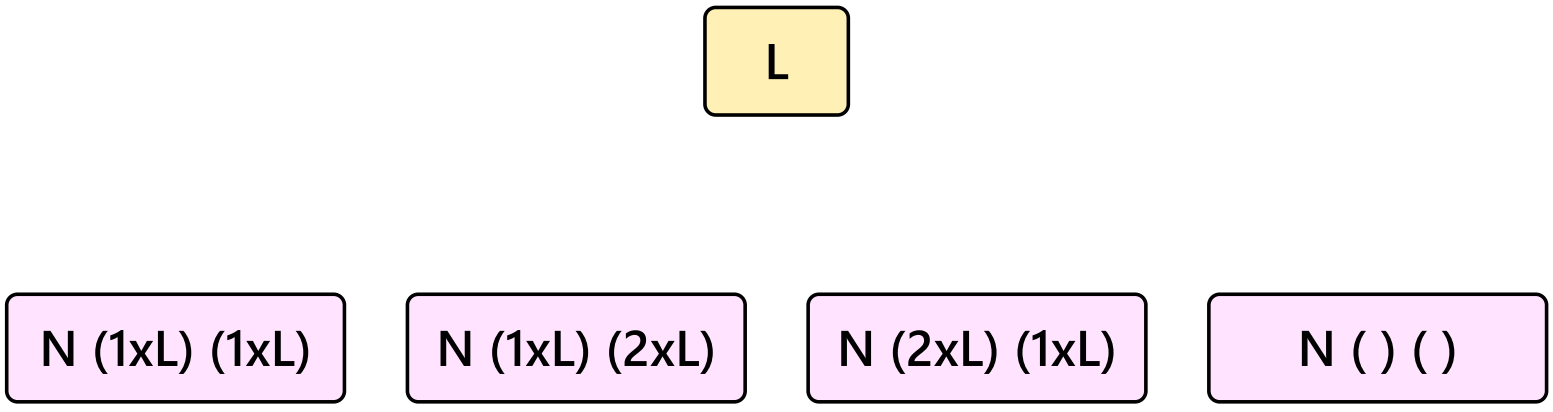


Leader's memory:

Step 1: **4xN**
Step 2: **3x[N (1xL)]**; **1x[N (2xL)]**

Memory and messages

Each agent has unlimited memory and can remember everything it sees: its memory is its *state*. Also, when an agent sends a message, it sends its current state, i.e., the entire contents of its memory.

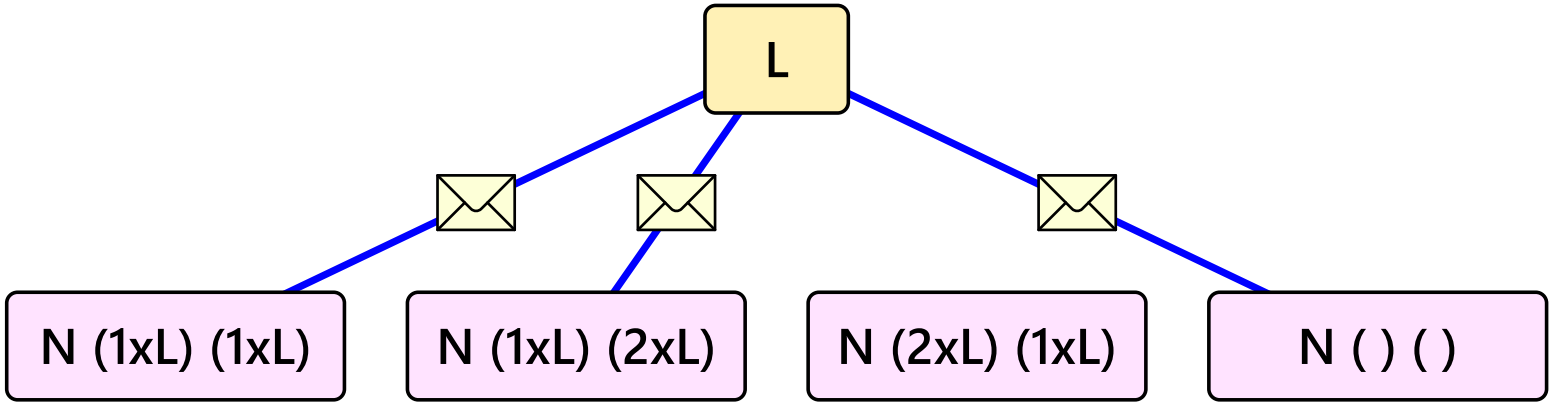


Leader's memory:

Step 1: **4xN**
Step 2: **3x[N (1xL)]**; **1x[N (2xL)]**

Memory and messages

Each agent has unlimited memory and can remember everything it sees: its memory is its *state*. Also, when an agent sends a message, it sends its current state, i.e., the entire contents of its memory.

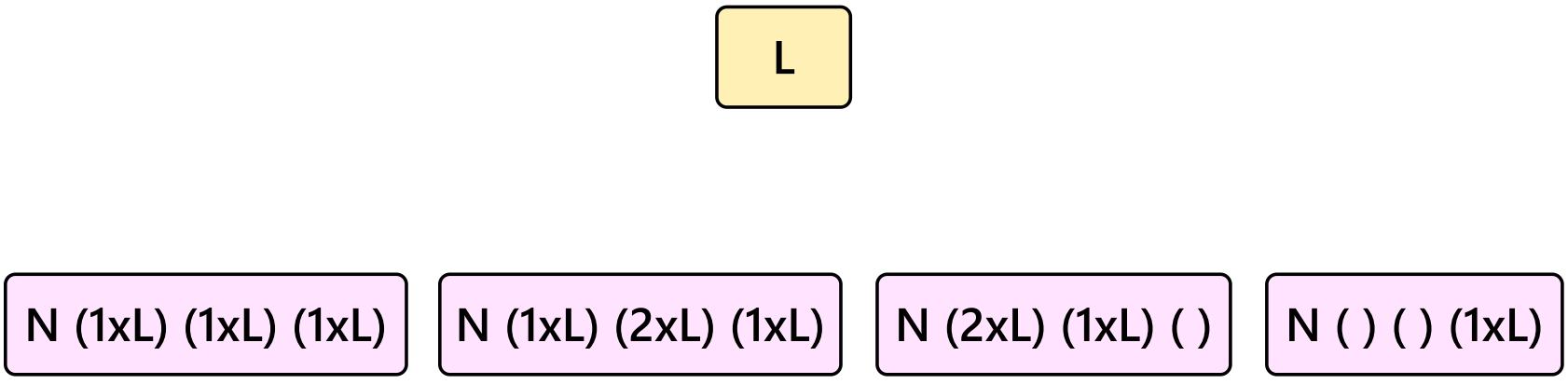


Leader's memory:

Step 1: **4xN**
Step 2: **3x[N (1xL)]**; **1x[N (2xL)]**
Step 3: **1x[N ( ) ( )]**; **1x[N (1xL) (1xL)]**; **1x[N (1xL) (2xL)]**

Memory and messages

Each agent has unlimited memory and can remember everything it sees: its memory is its *state*. Also, when an agent sends a message, it sends its current state, i.e., the entire contents of its memory.

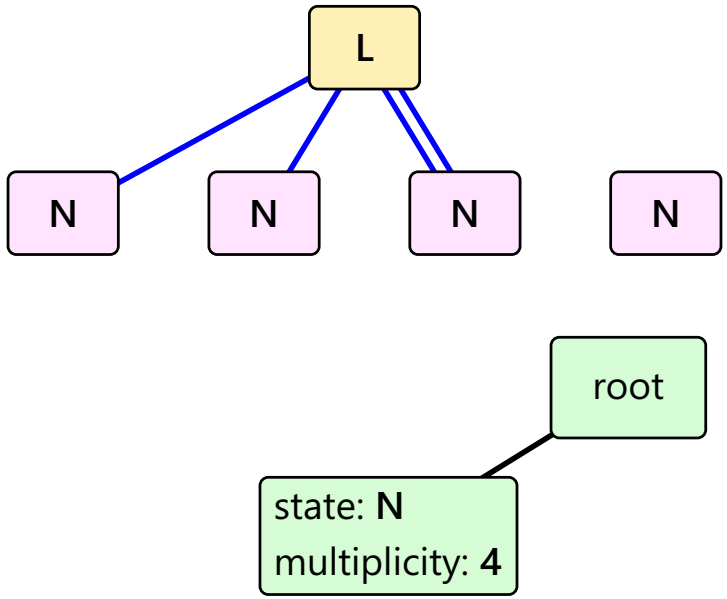


Leader's memory:

Step 1: **4xN**
Step 2: **3x[N (1xL)]**; **1x[N (2xL)]**
Step 3: **1x[N ( ) ( )]**; **1x[N (1xL) (1xL)]**; **1x[N (1xL) (2xL)]**

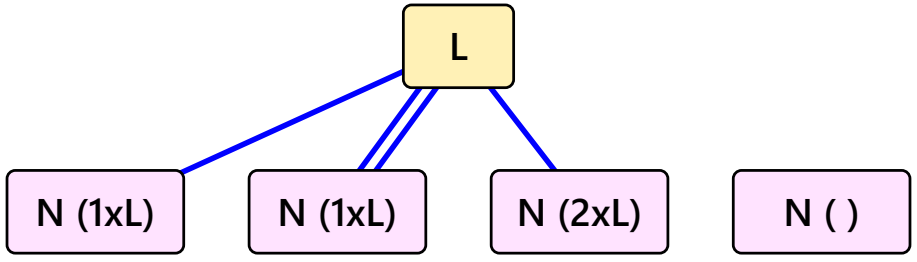
Special case: agents communicating with the Leader

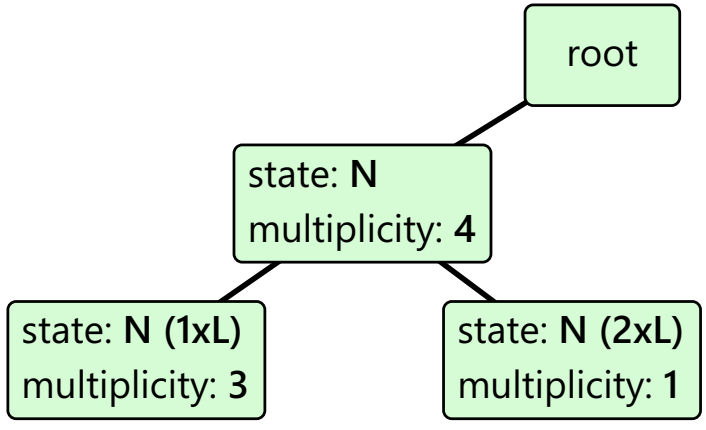
Let us focus on the agents that interact directly with the Leader.
Assume that interactions have a *multiplicity* (later we will see why).
For each new interaction, the Leader adds a node to a *History tree*.



Special case: agents communicating with the Leader

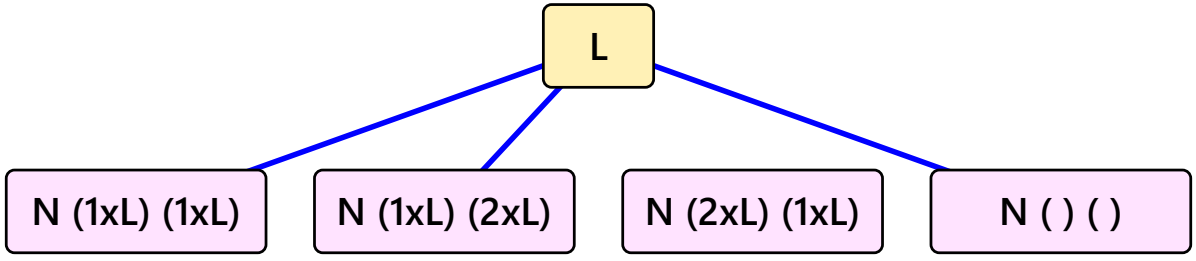
Let us focus on the agents that interact directly with the Leader.
Assume that interactions have a *multiplicity* (later we will see why).
For each new interaction, the Leader adds a node to a *History tree*.

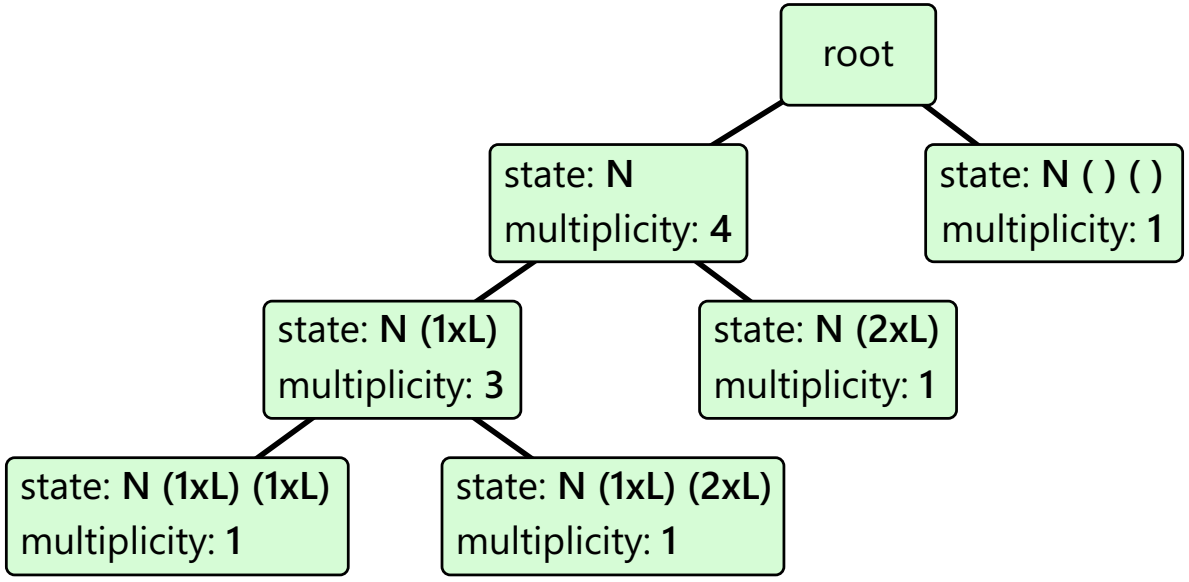




Special case: agents communicating with the Leader

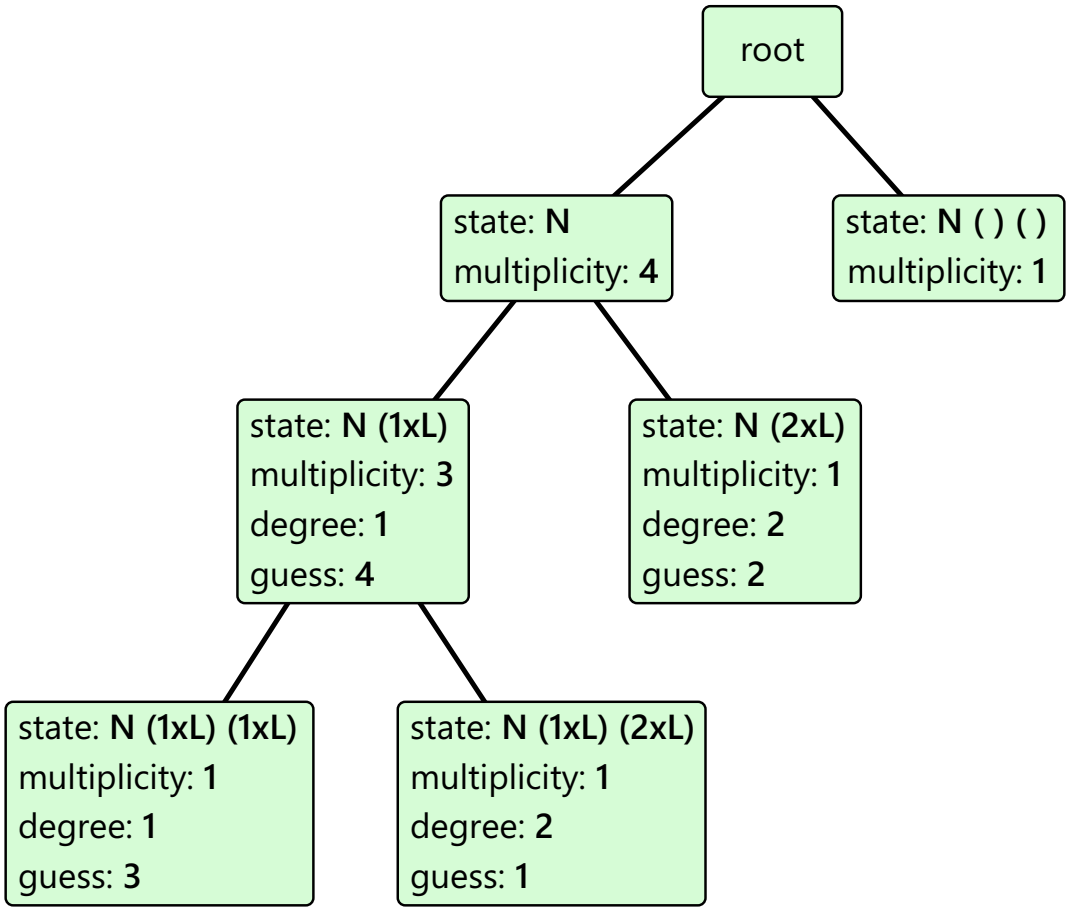
Let us focus on the agents that interact directly with the Leader.
Assume that interactions have a *multiplicity* (later we will see why).
For each new interaction, the Leader adds a node to a *History tree*.





History tree

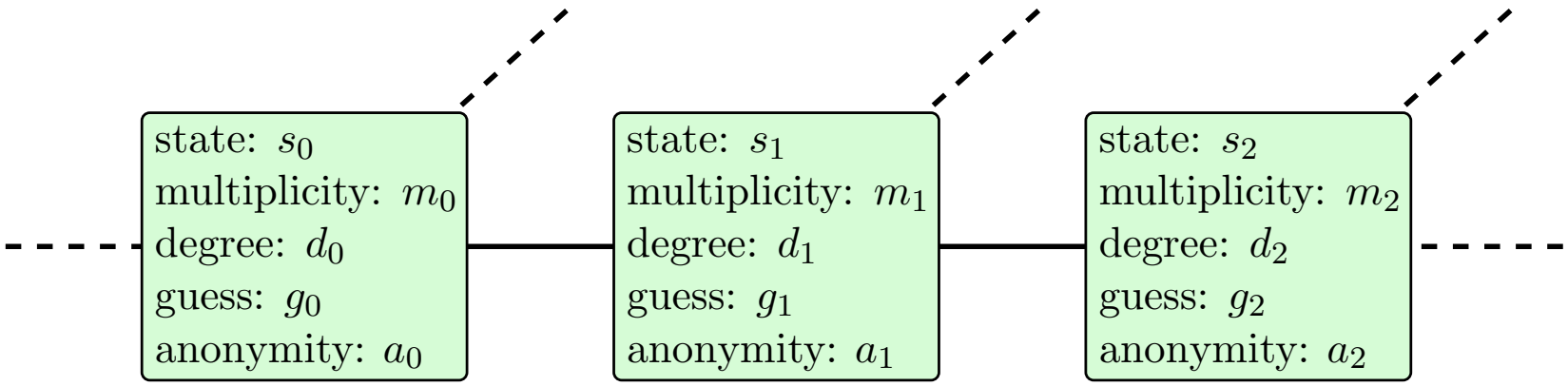
How can the Leader know how many agents correspond to a node in the History tree, i.e., its *anonymity*? When it sees an agent again, it looks at the multiplicity of its previous interaction with the Leader, i.e., its *degree*.



The Leader's *guess* for a node's anonymity is computed by dividing the multiplicity of the node's parent by the node's degree.

How guesses work

A *downward path* in the History tree looks like this:



Each guess is computed by the Leader as: $g_i = \lfloor \frac{m_{i-1}}{d_i} \rfloor$.

The *anonymity* a_i is the number of agents that were seen by the Leader when they were in state s_i (this is unknown to the leader).

Observation

$$1 \leq a_{i+1} \leq a_i \leq g_i$$

The Leader's goal is to guess the correct anonymities.
How does the Leader know when a guess is correct?

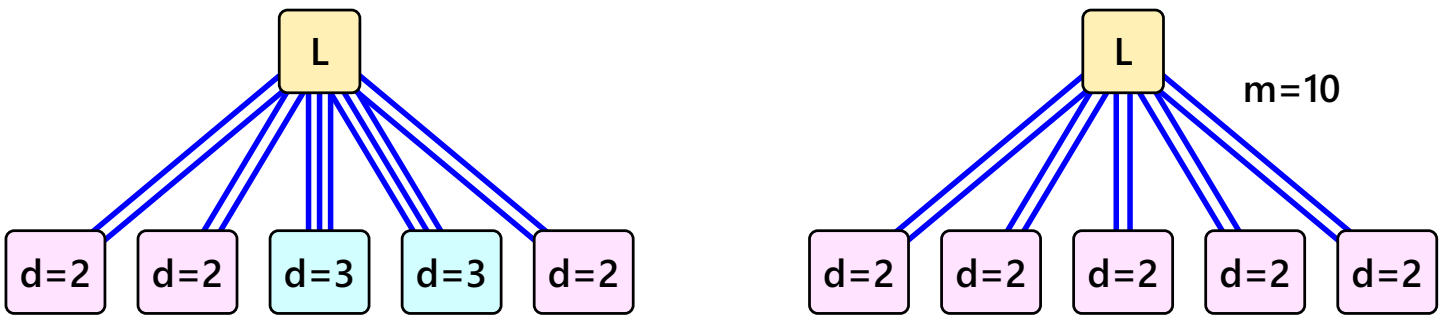
Basic lemma

When the anonymity of a node is the same as its parent's, the guess on this node is correct:

Lemma

If $a_i = a_{i+1}$, *then* $g_{i+1} = a_{i+1}$.

Proof. If the agents in state s_i seen by the Leader do not all have the same degree, then $a_{i+1} < a_i$. Hence all degrees are the same, and the next guess is correct: $g_{i+1} = \lfloor m_i/d_{i+1} \rfloor = a_i = a_{i+1}$. □



Unfortunately, the Leader cannot use this information directly, because it does not know the anonymities.

Limiting lemma

However, the Leader can use the previous lemma indirectly, thanks to the *Limiting lemma*:

Lemma

For any $d \geq 0$, if $a_0 < g_0$ and $\forall j \in [1..d], g_j > g_0 - j$, then $a_d \leq a_0 - d$.

Proof. By induction on d: the base case $d = 0$ is trivial. Now assume the claim for $d = k$, and let us prove it for $d = k + 1$. So, let $a_0 < g_0$ and $\forall j \in [1..k+1], g_j > g_0 - j$; we have to prove that $a_{k+1} \leq a_0 - k - 1$.

We know that $a_{k+1} \leq a_k$. By the inductive hypothesis, $a_k \leq a_0 - k$. Since $a_0 < g_0$, we have $a_0 - k \leq g_0 - k - 1 < g_{k+1}$, and so $a_{k+1} < g_{k+1}$.

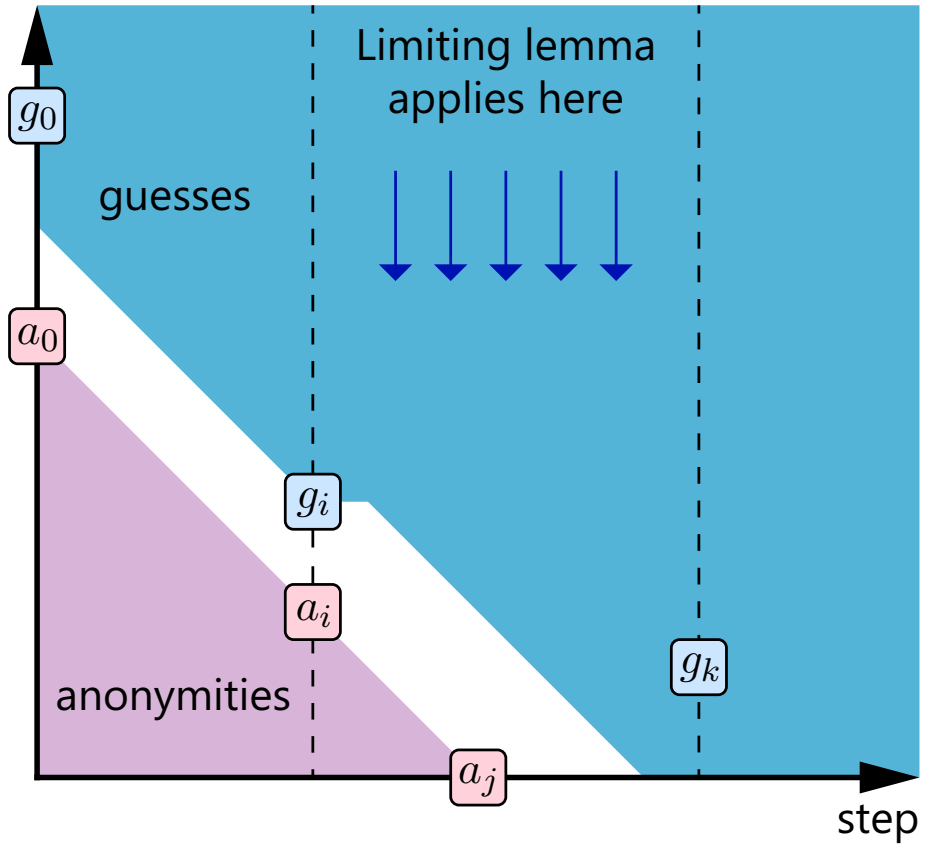
By the previous lemma, $a_{k+1} \leq a_k - 1 \leq a_0 - k - 1$, as desired. □

Detecting correct guesses

Theorem

In every downward path in the History tree with guesses $g_0 = k$, g_1, g_2, ..., g_k, the greatest index i such that $g_i + i$ is minimum corresponds to a correct guess, i.e., $g_i = a_i$.

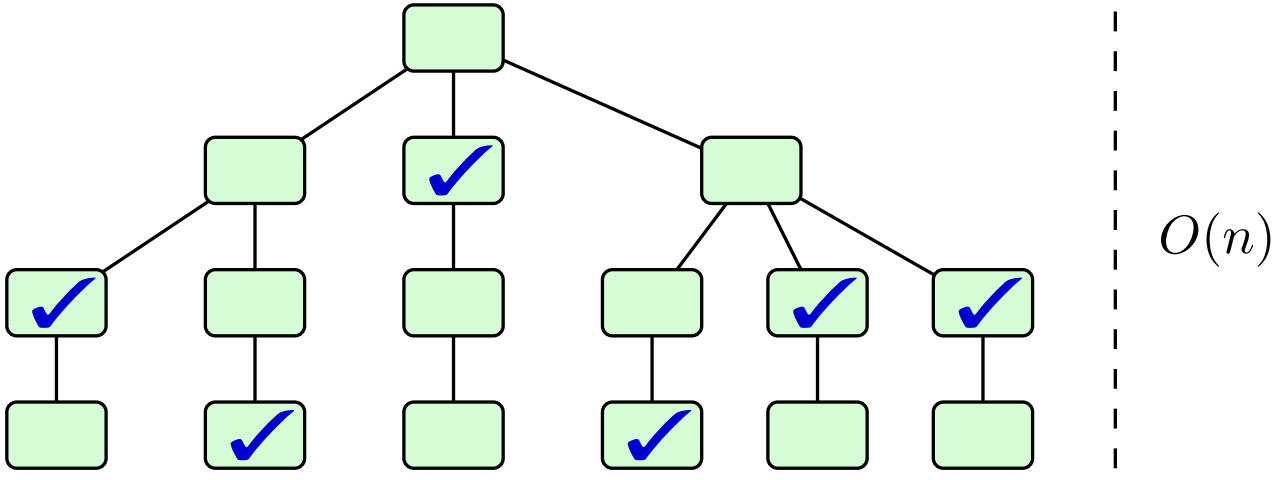
Proof. Otherwise we would have $a_j \leq 0$, with $j = a_i + i$: impossible. □



Detecting correct guesses

Assume that all multiplicities are at most n. Thus, all guesses are bounded by n, too.

By the previous theorem, in every downward path in the History tree of length at least n, the leader can determine the anonymity of at least one node.

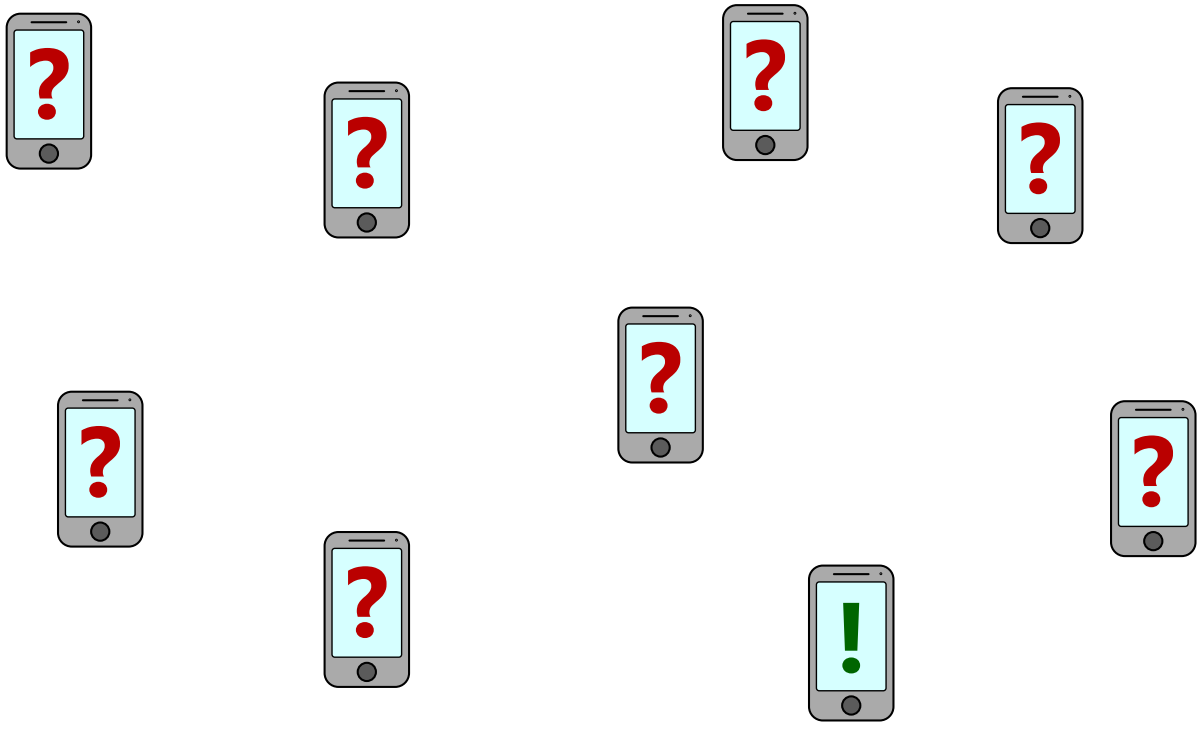


As soon as all the leaves in the History tree have depth at least n, the Leader can count all agents that have interacted with it.

Of course, there may be agents that never interact with the Leader. How can the leader count them?

Propagation of information

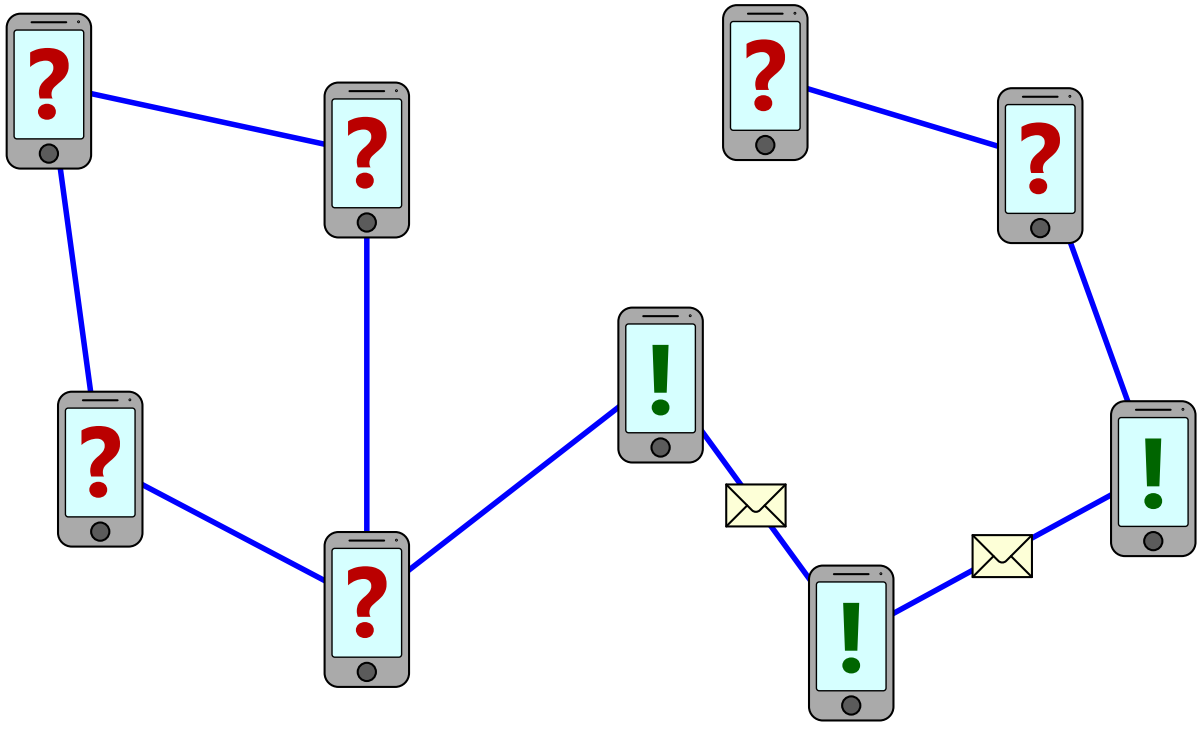
In a dynamic network, every news is communicated to everybody in at most n steps (where n is the number of agents). Hence, whenever two agents interact, the Leader will eventually know.



This gives a termination condition for a counting algorithm: When the Leader has a guess on n, it waits as many steps to confirm it.

Propagation of information

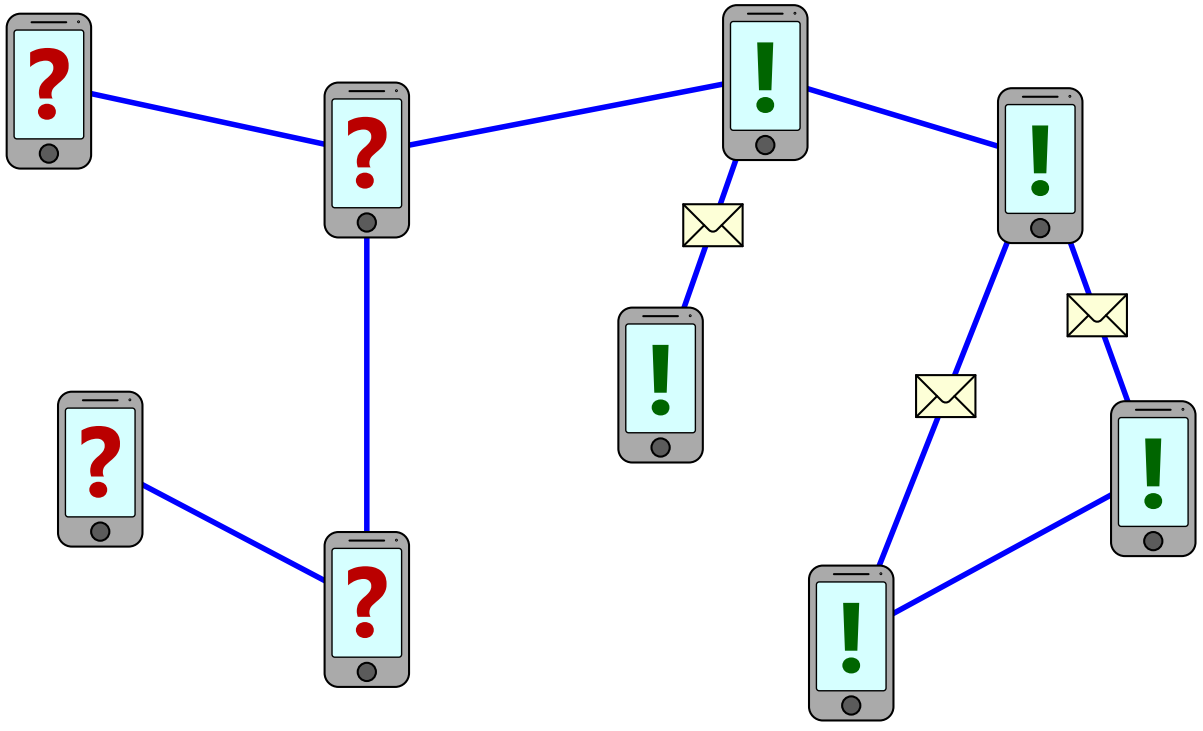
In a dynamic network, every news is communicated to everybody in at most n steps (where n is the number of agents). Hence, whenever two agents interact, the Leader will eventually know.



This gives a termination condition for a counting algorithm: When the Leader has a guess on n, it waits as many steps to confirm it.

Propagation of information

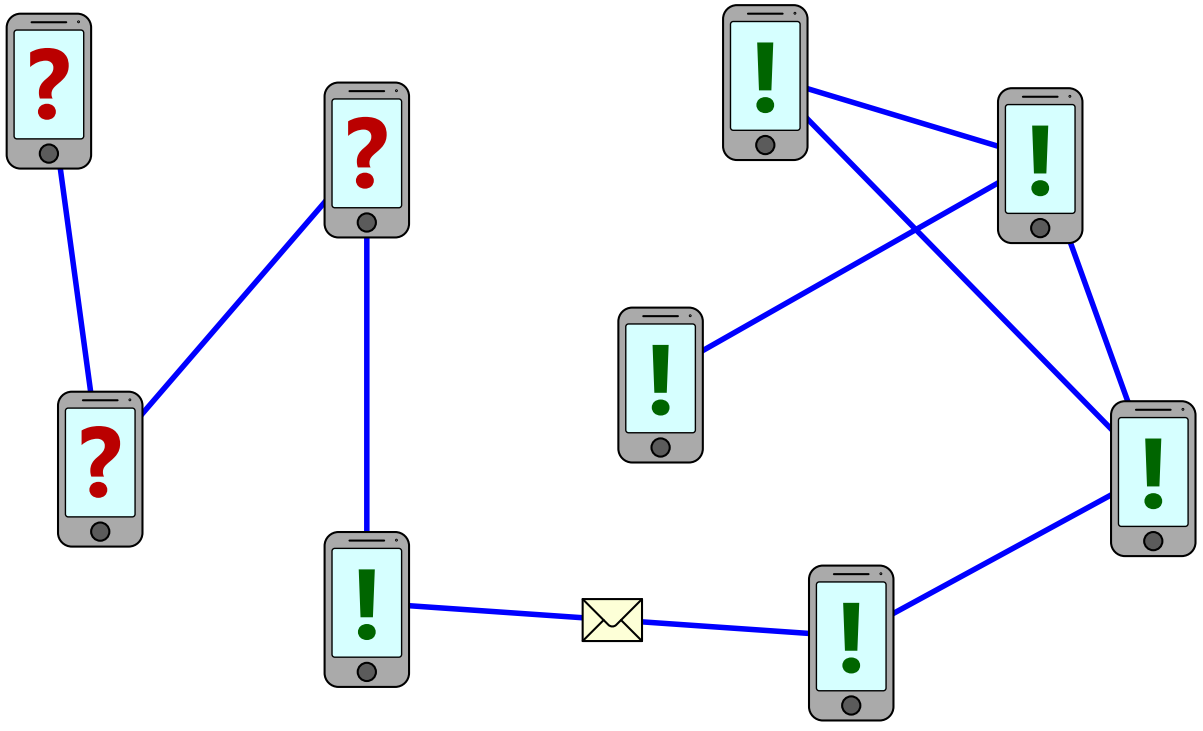
In a dynamic network, every news is communicated to everybody in at most n steps (where n is the number of agents). Hence, whenever two agents interact, the Leader will eventually know.



This gives a termination condition for a counting algorithm: When the Leader has a guess on n, it waits as many steps to confirm it.

Propagation of information

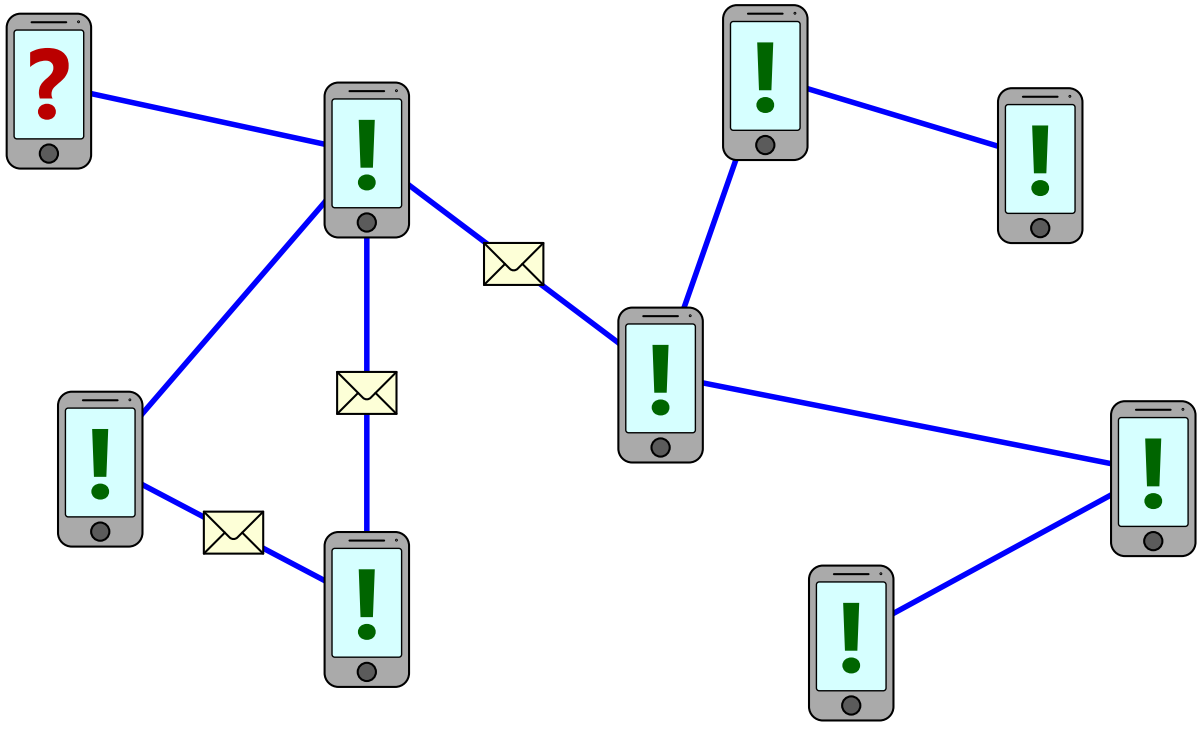
In a dynamic network, every news is communicated to everybody in at most n steps (where n is the number of agents). Hence, whenever two agents interact, the Leader will eventually know.



This gives a termination condition for a counting algorithm: When the Leader has a guess on n, it waits as many steps to confirm it.

Propagation of information

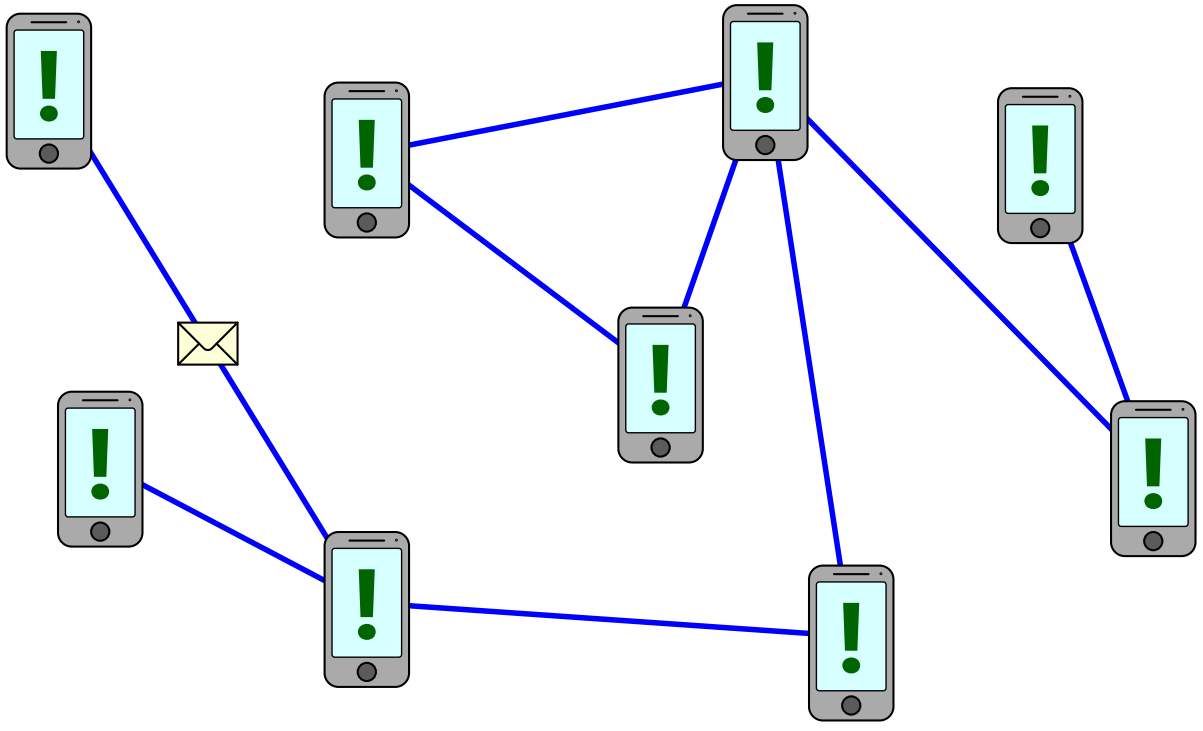
In a dynamic network, every news is communicated to everybody in at most n steps (where n is the number of agents). Hence, whenever two agents interact, the Leader will eventually know.



This gives a termination condition for a counting algorithm: When the Leader has a guess on n, it waits as many steps to confirm it.

Propagation of information

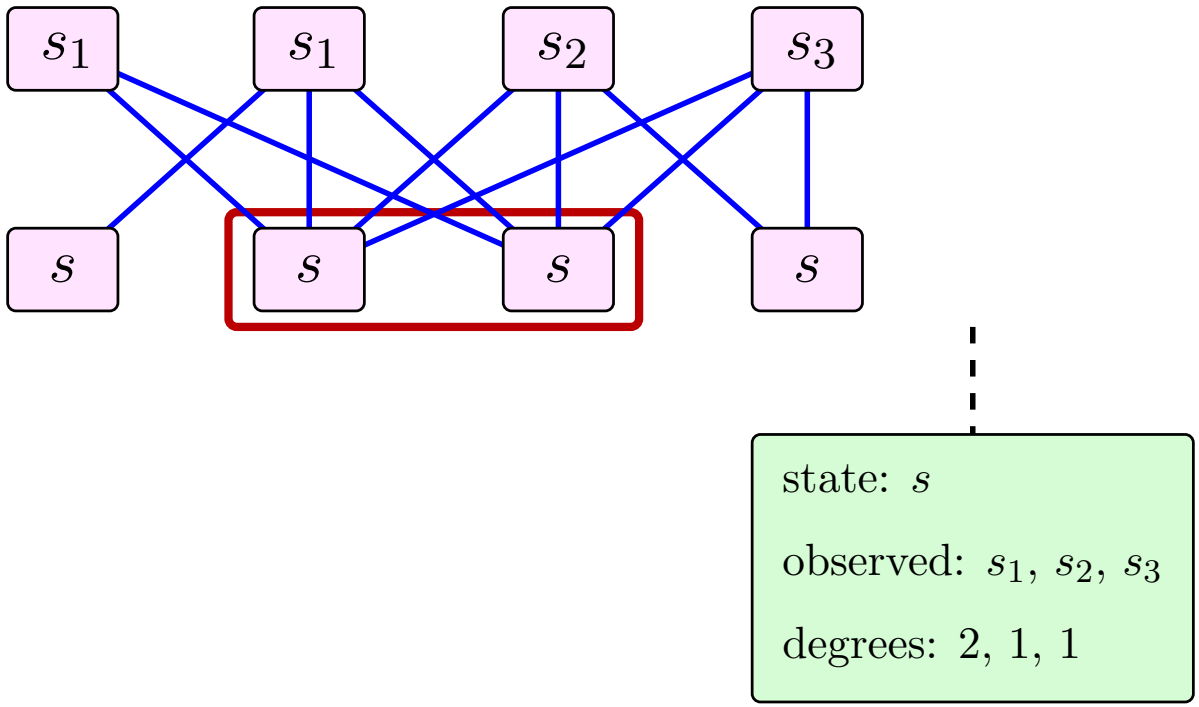
In a dynamic network, every news is communicated to everybody in at most n steps (where n is the number of agents). Hence, whenever two agents interact, the Leader will eventually know.



This gives a termination condition for a counting algorithm: When the Leader has a guess on n, it waits as many steps to confirm it.

Generalized History tree

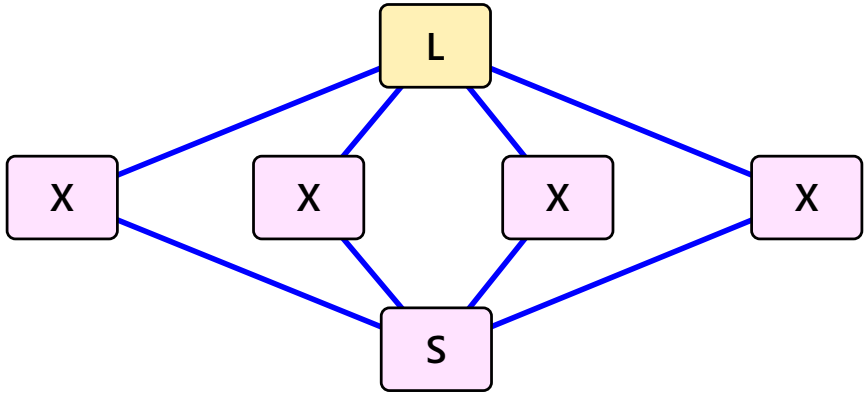
As soon as the Leader learns that some agents in state s have interacted in a single step with agents in states s_1, s_2, ..., s_k, it creates a node in the History tree, also storing the degrees.



The anonymity of this node is defined as the number of agents that, when their state was s, interacted precisely with agents in states s_1, s_2, ..., s_k, with degrees d_1, d_2, ..., d_k, respectively.

Generalized multiplicities

On its way to the Leader, some information may be "multiplied" by intermediate agents: this is why we introduced multiplicities.



In the generalized algorithm, the Leader "accepts" a piece of information only from groups of agents that it has already counted; the *anonymity* of such a group becomes the *multiplicity* of the node in the History tree corresponding to that piece of information.

Future work: Does the idea of the previous guessing algorithm work for the generalized History tree, as well? How many steps does it take for the Leader to count all agents?