Seminar 1 – Anonymous Networks: Introduction and Basic Techniques

Distributed Computing in Anonymous Dynamic Systems

Giovanni Viglietta

Rome – March 6, 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

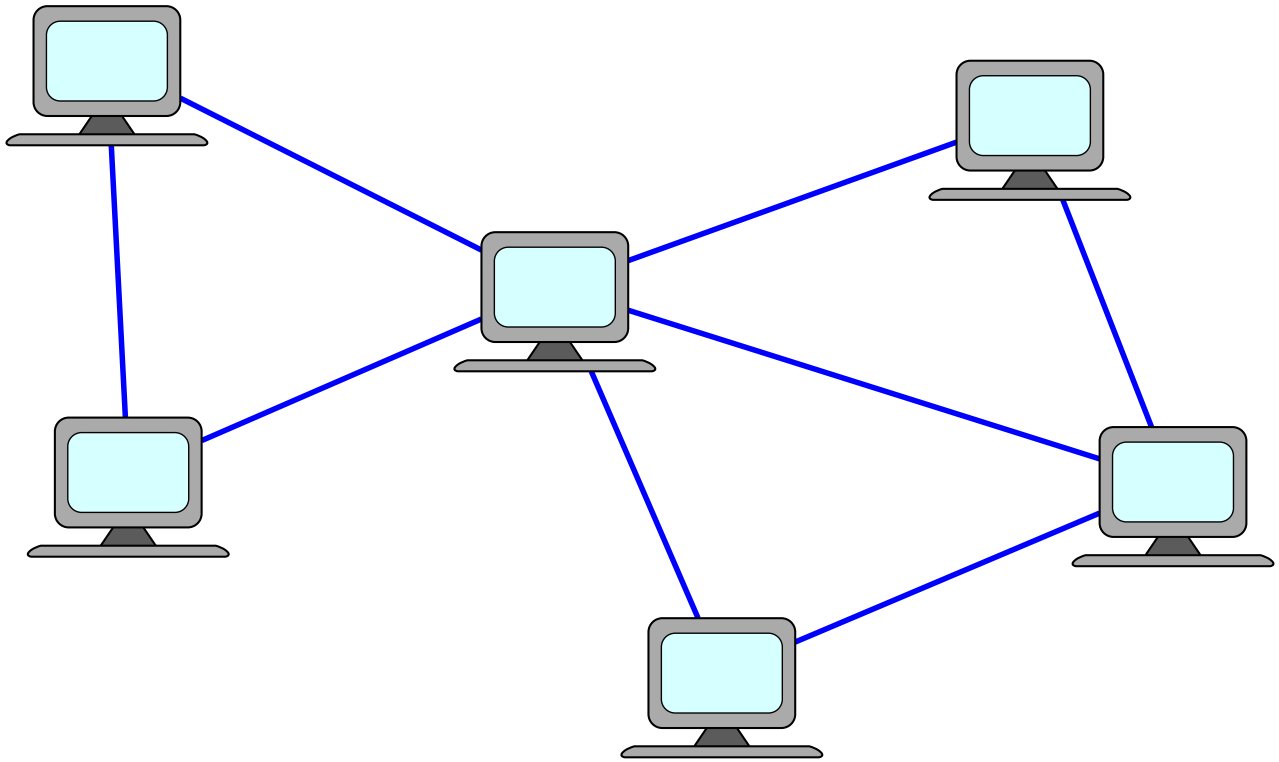
Pre-recorded 10-minute presentation video on one of the papers that will be suggested at the end of the course.

Today's seminar

- Network models
- Broadcast
- Leader election
- Spanning tree construction
- Counting
- Average consensus

Static networks

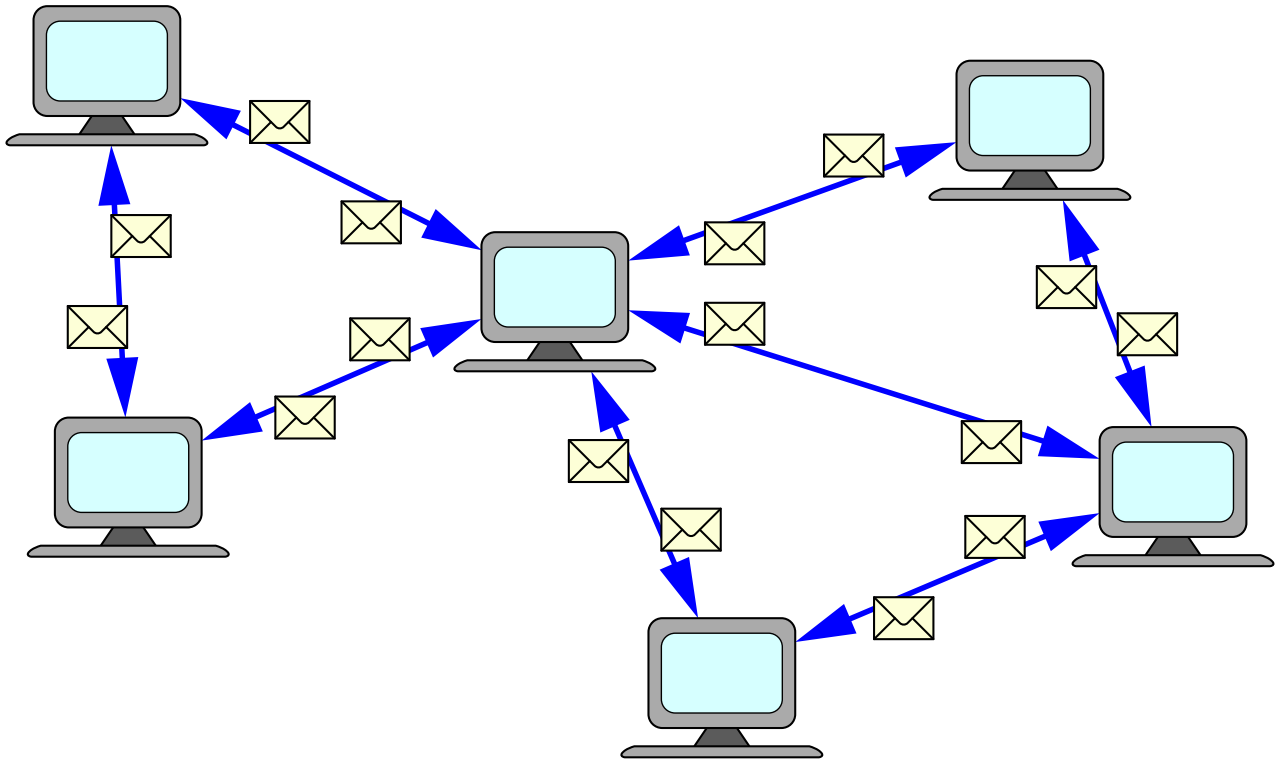
In a *static network*, some *agents* (or processes, or machines) are connected with each other through permanent links.



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

Static networks

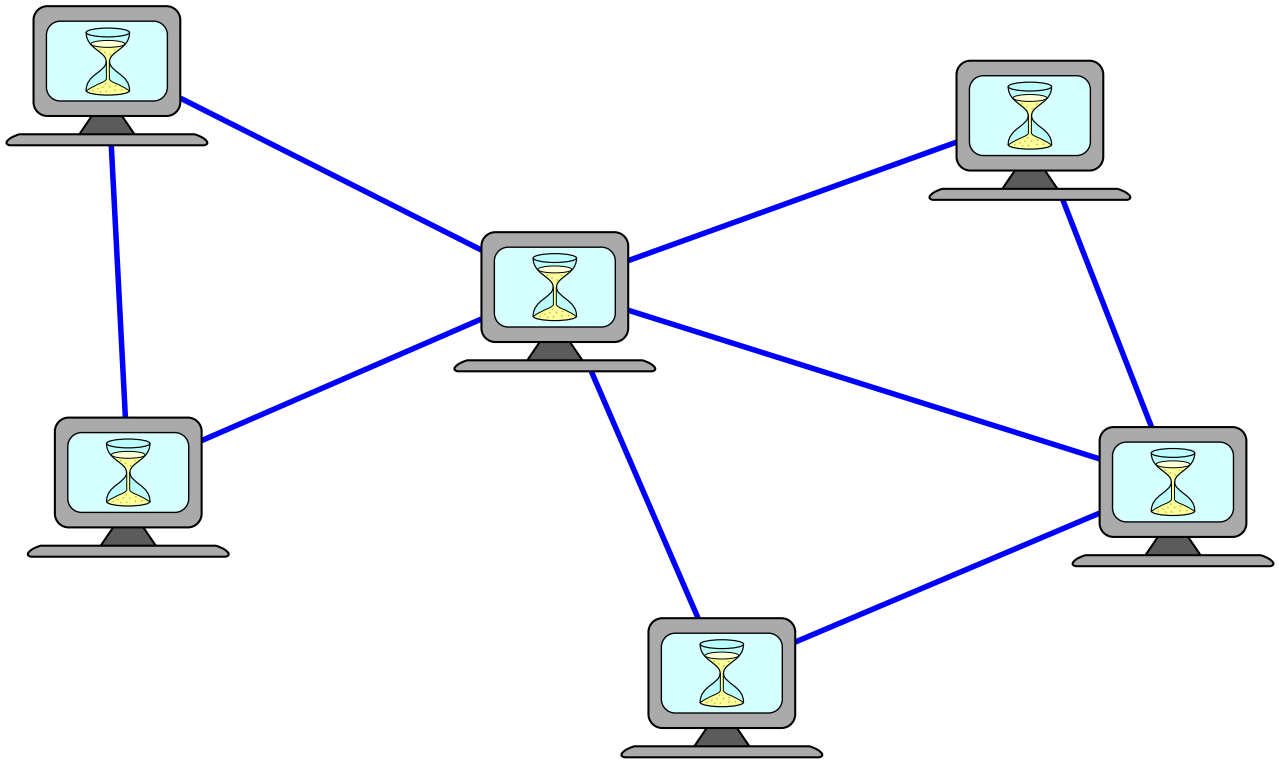
In a *static network*, some *agents* (or processes, or machines) are connected with each other through permanent links.



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

Static networks

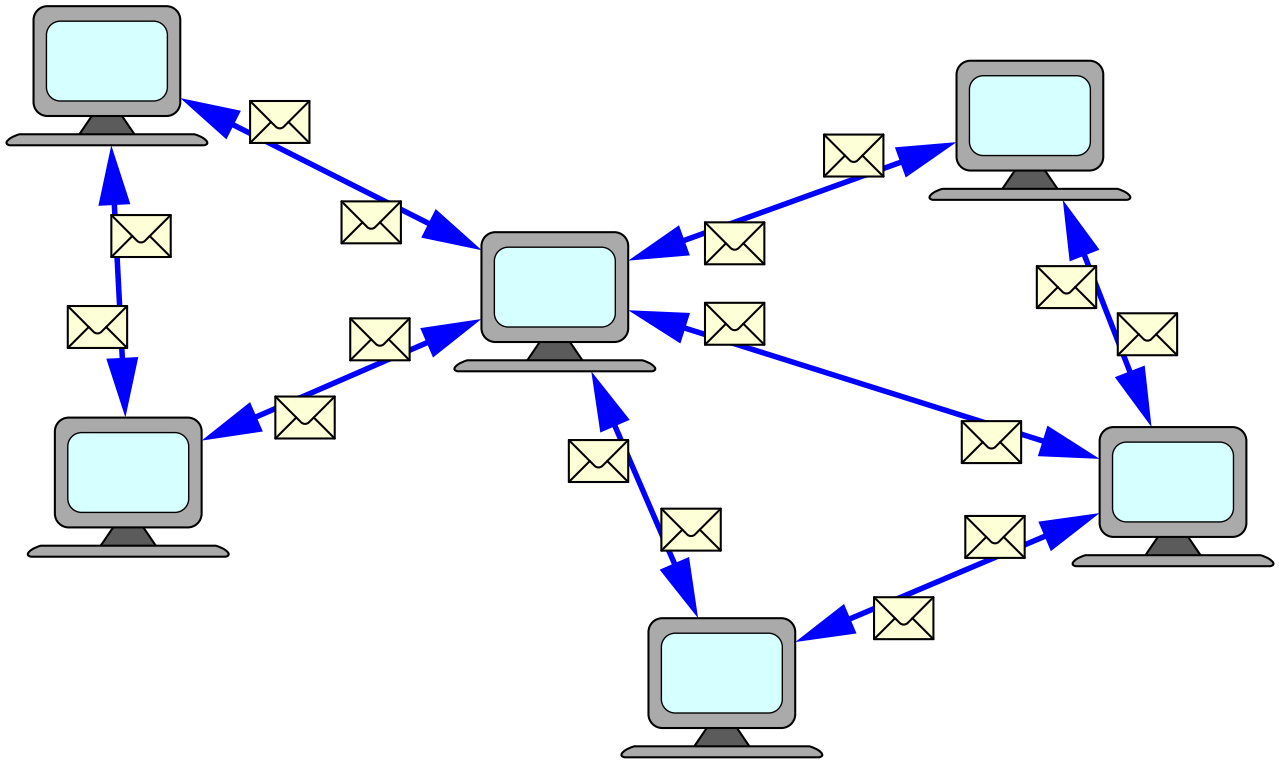
In a *static network*, some *agents* (or processes, or machines) are connected with each other through permanent links.



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

Static networks

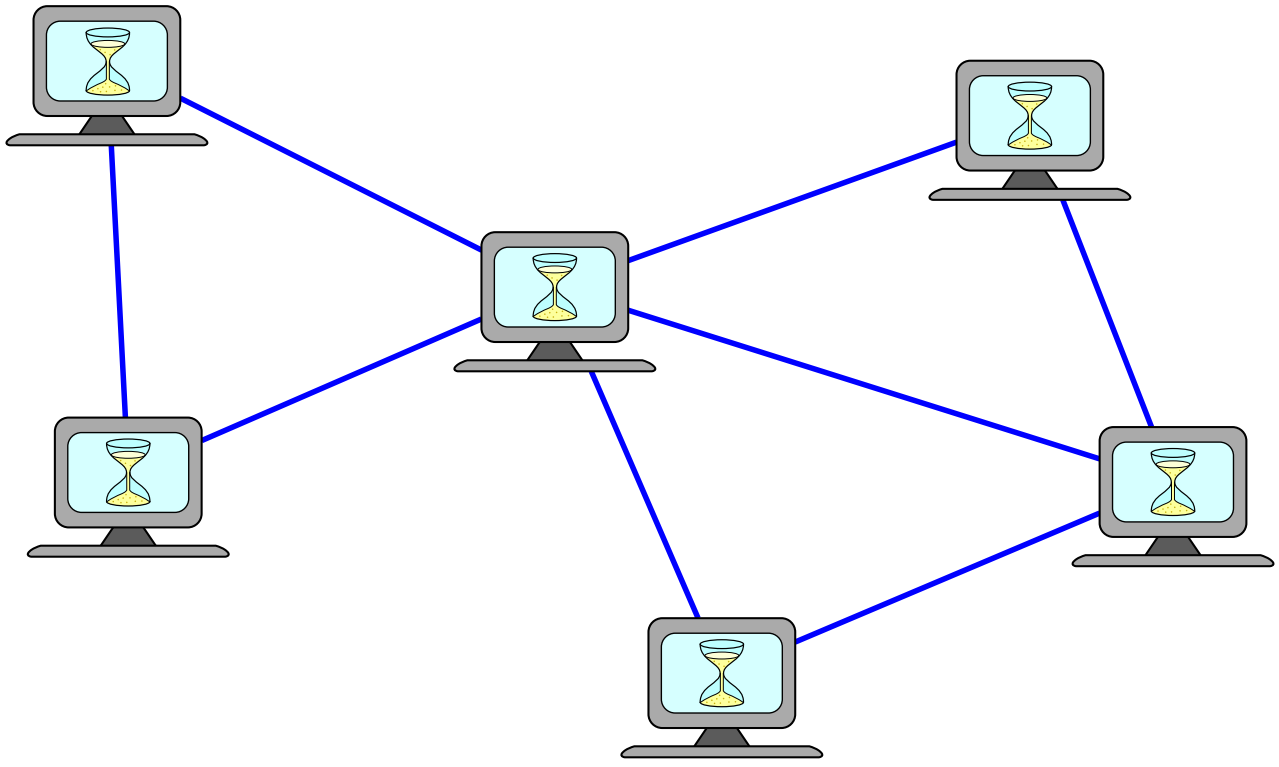
In a *static network*, some *agents* (or processes, or machines) are connected with each other through permanent links.



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

Static networks

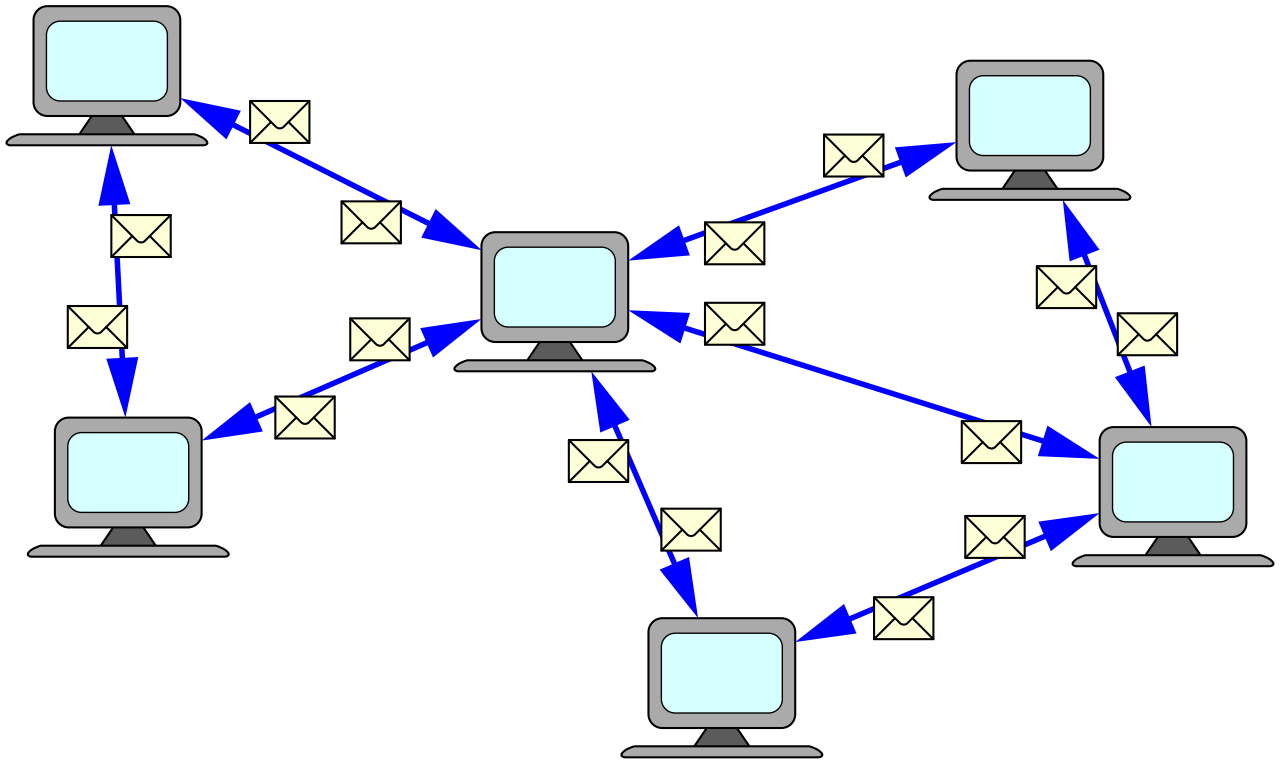
In a *static network*, some *agents* (or processes, or machines) are connected with each other through permanent links.



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

Static networks

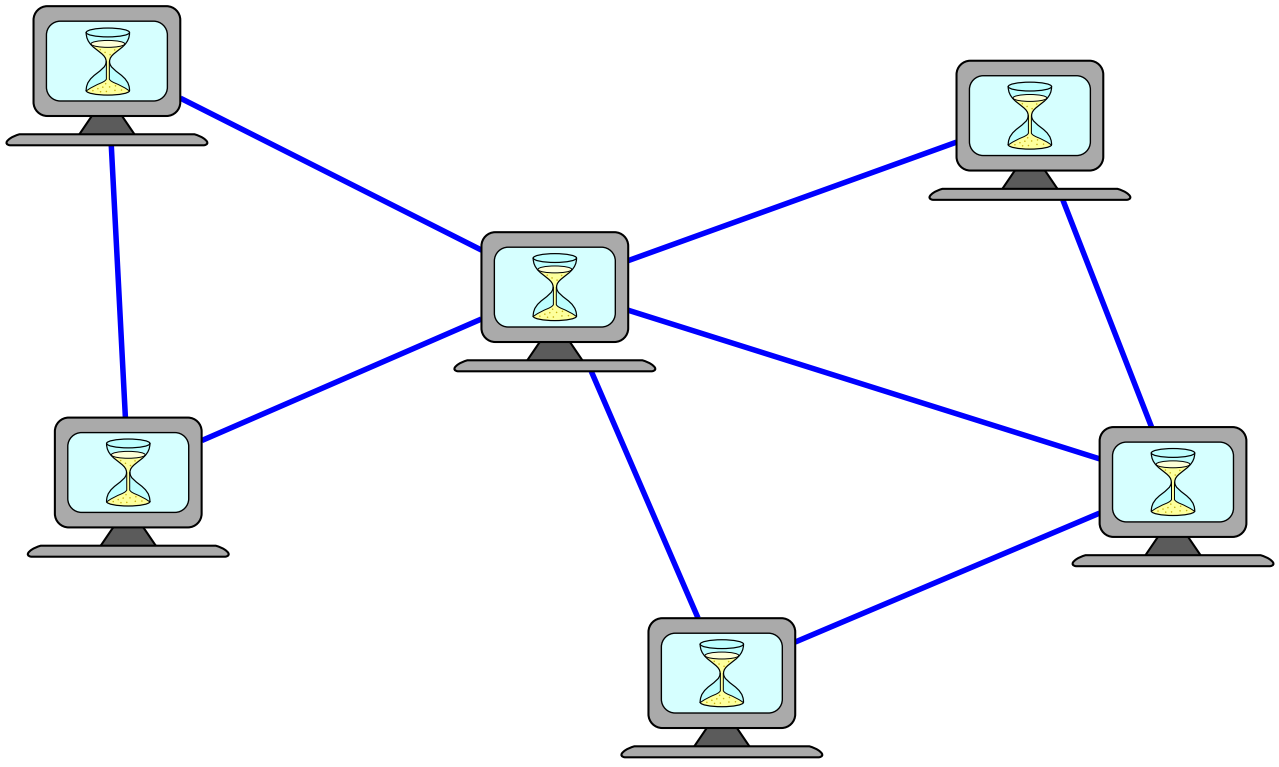
In a *static network*, some *agents* (or processes, or machines) are connected with each other through permanent links.



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

Static networks

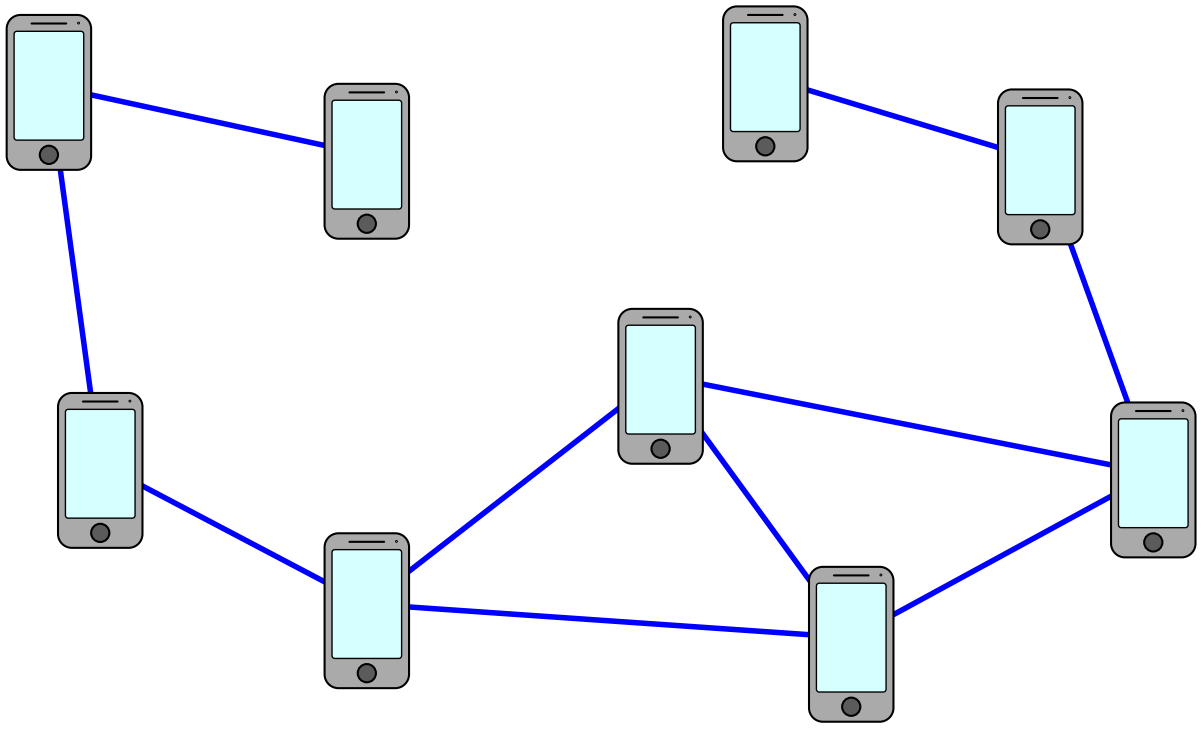
In a *static network*, some *agents* (or processes, or machines) are connected with each other through permanent links.



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

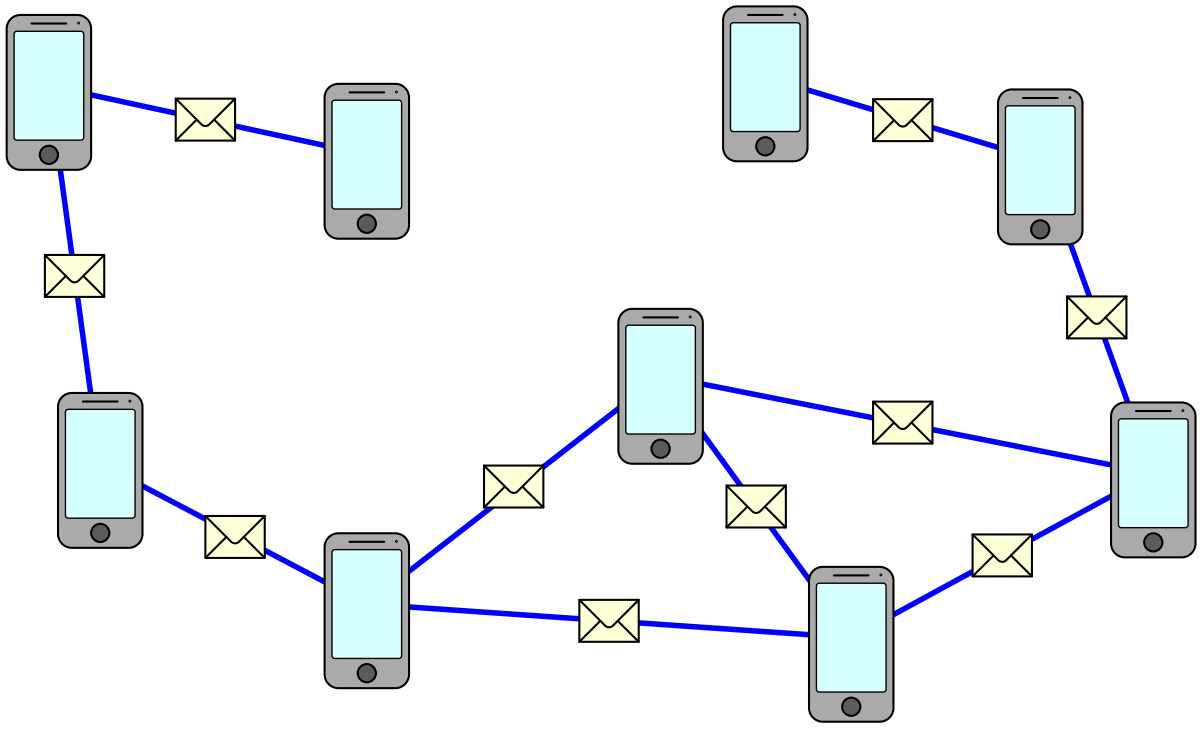
Dynamic networks

In a *dynamic network*, links may change over time.



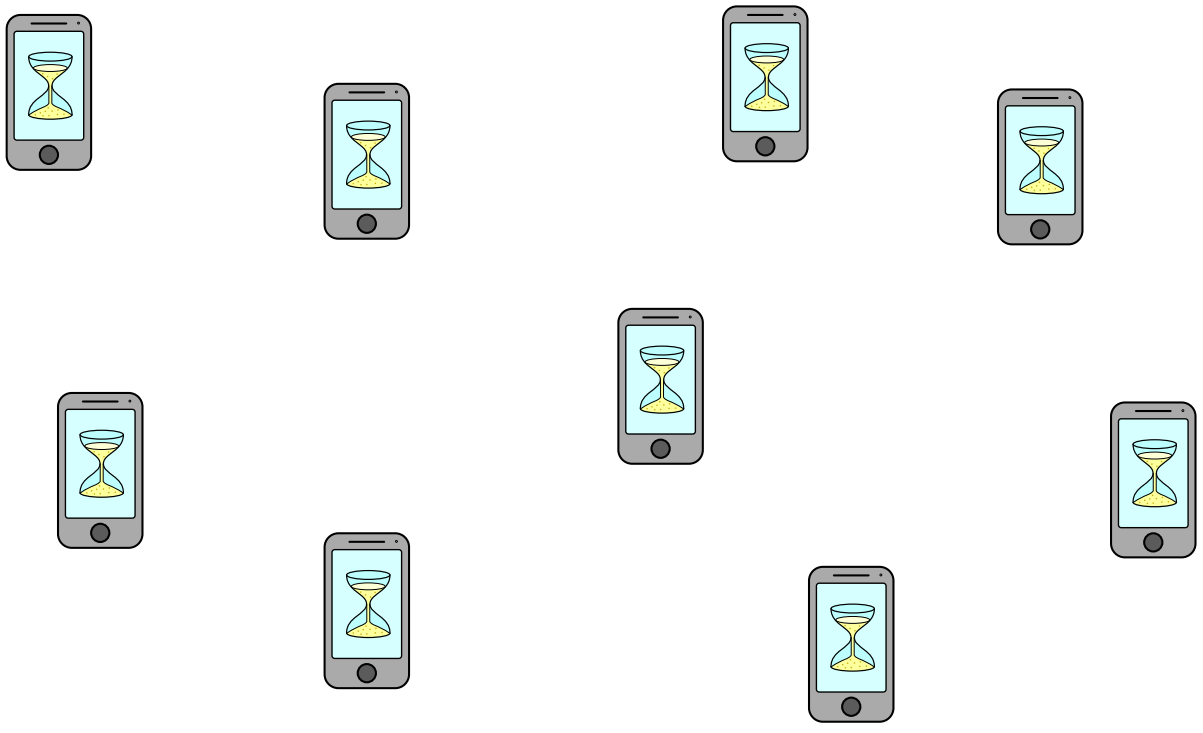
Dynamic networks

In a *dynamic network*, links may change over time.



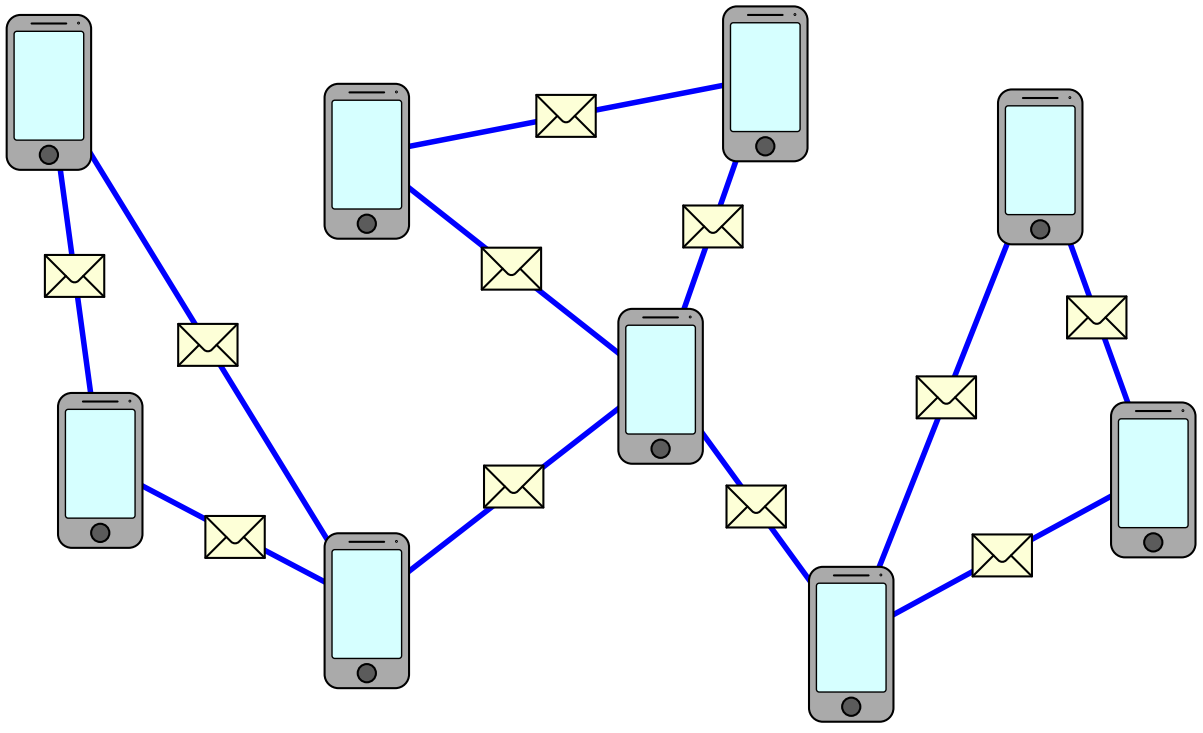
Dynamic networks

In a *dynamic network*, links may change over time.



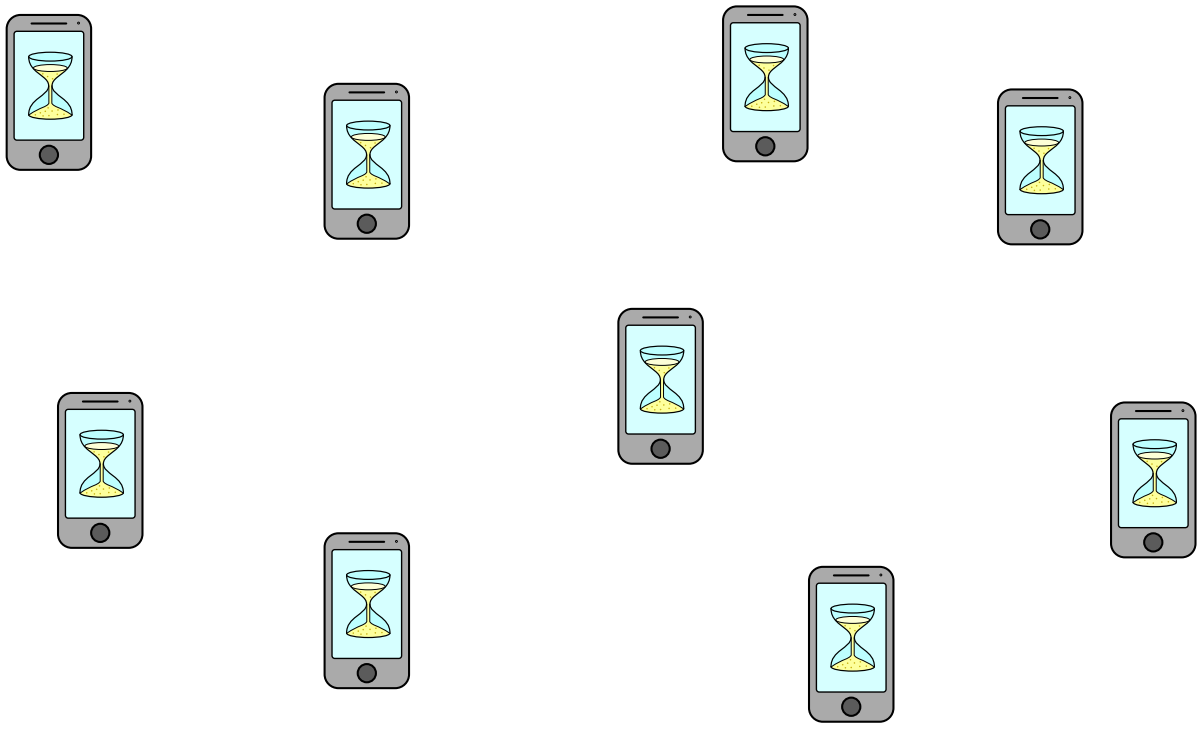
Dynamic networks

In a *dynamic network*, links may change over time.



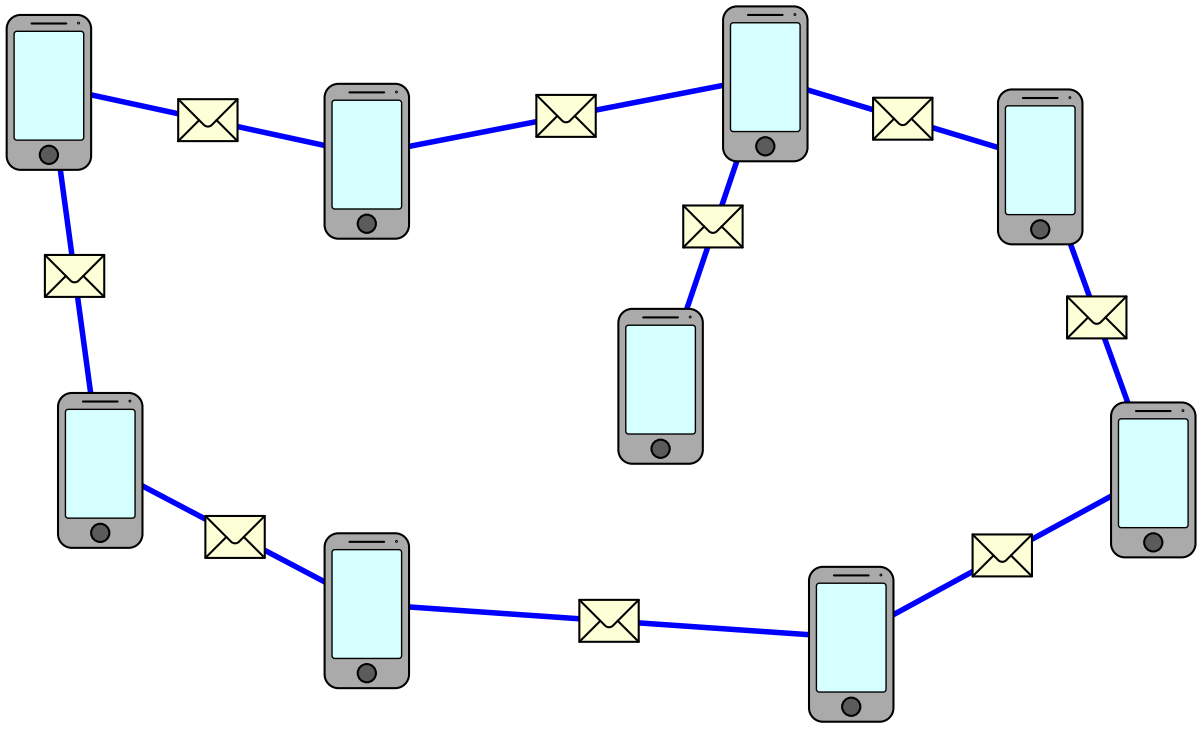
Dynamic networks

In a *dynamic network*, links may change over time.



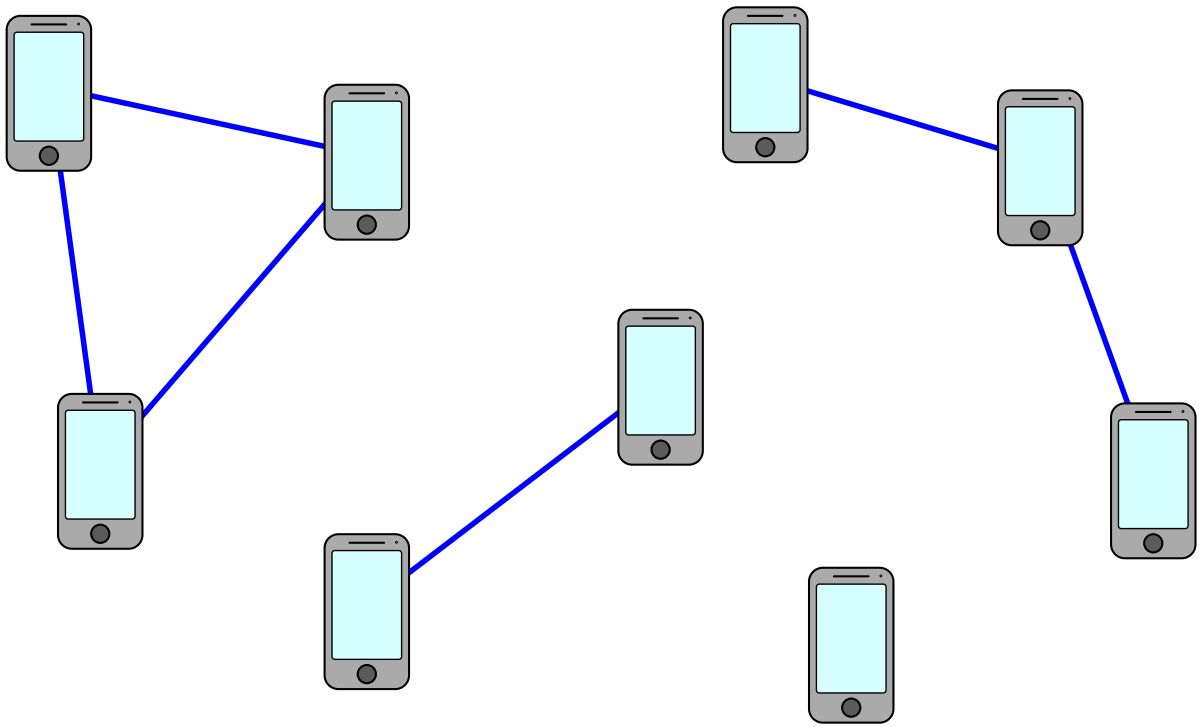
Dynamic networks

In a *dynamic network*, links may change over time.



Disconnected networks

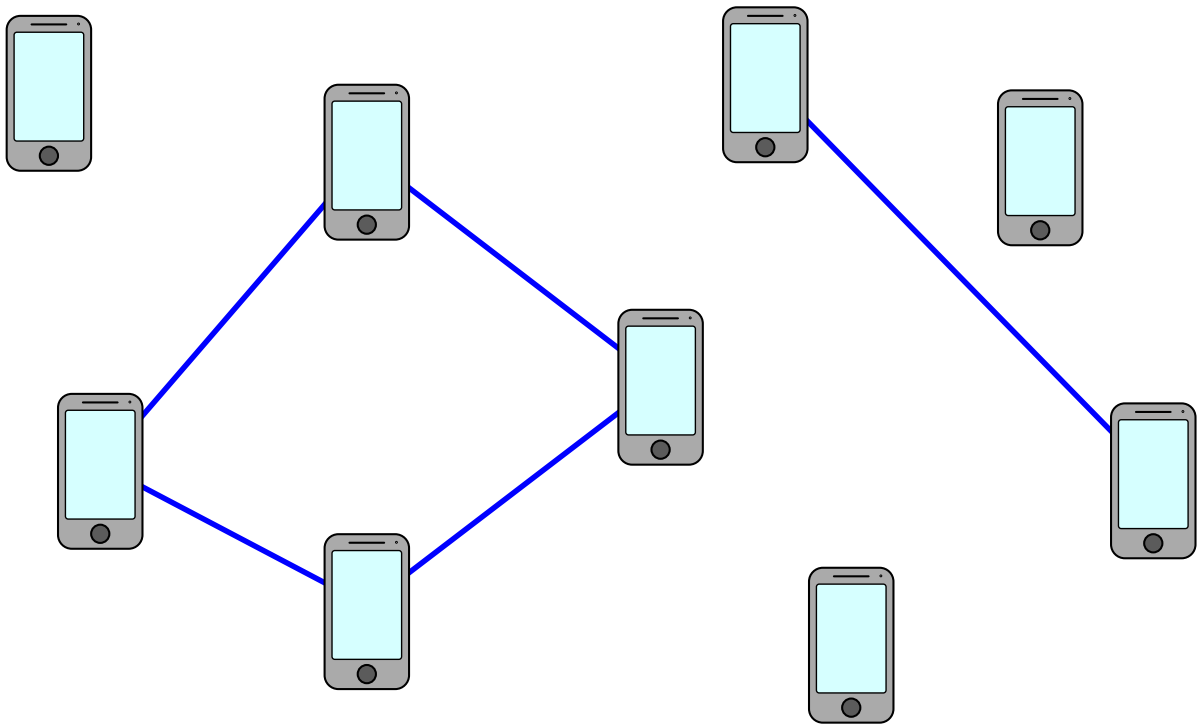
While the network is typically assumed to be (strongly) connected at every round, we may remove this assumption.



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

Disconnected networks

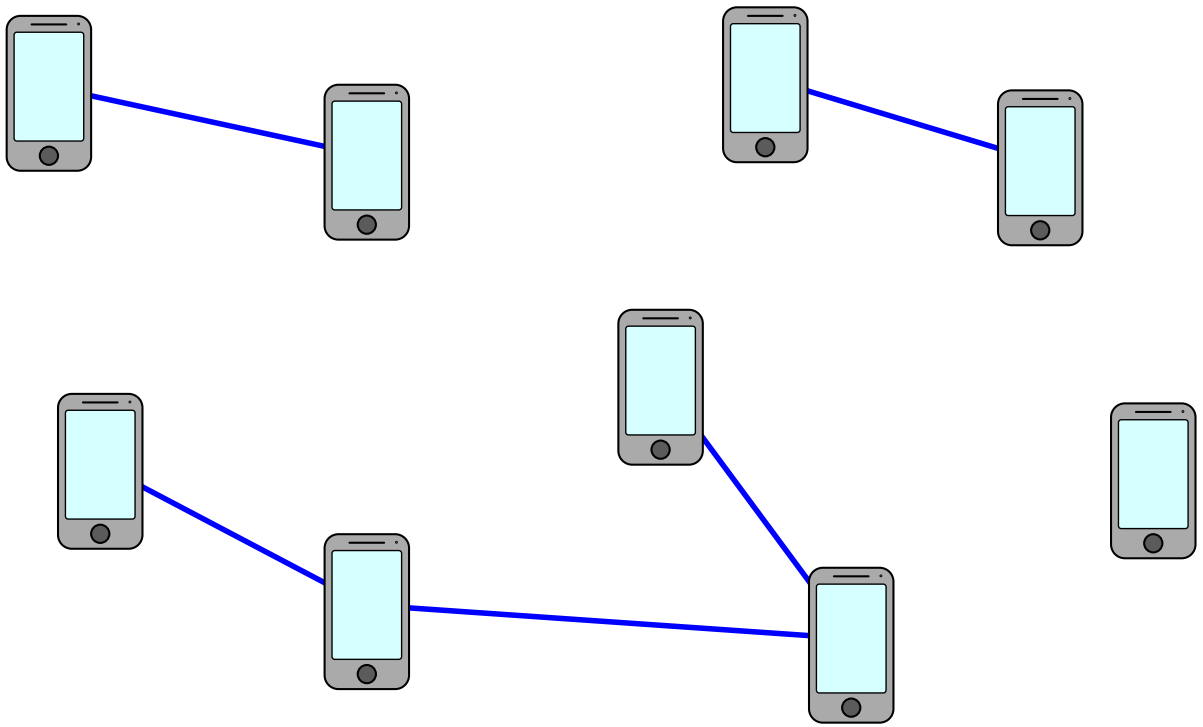
While the network is typically assumed to be (strongly) connected at every round, we may remove this assumption.



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

Disconnected networks

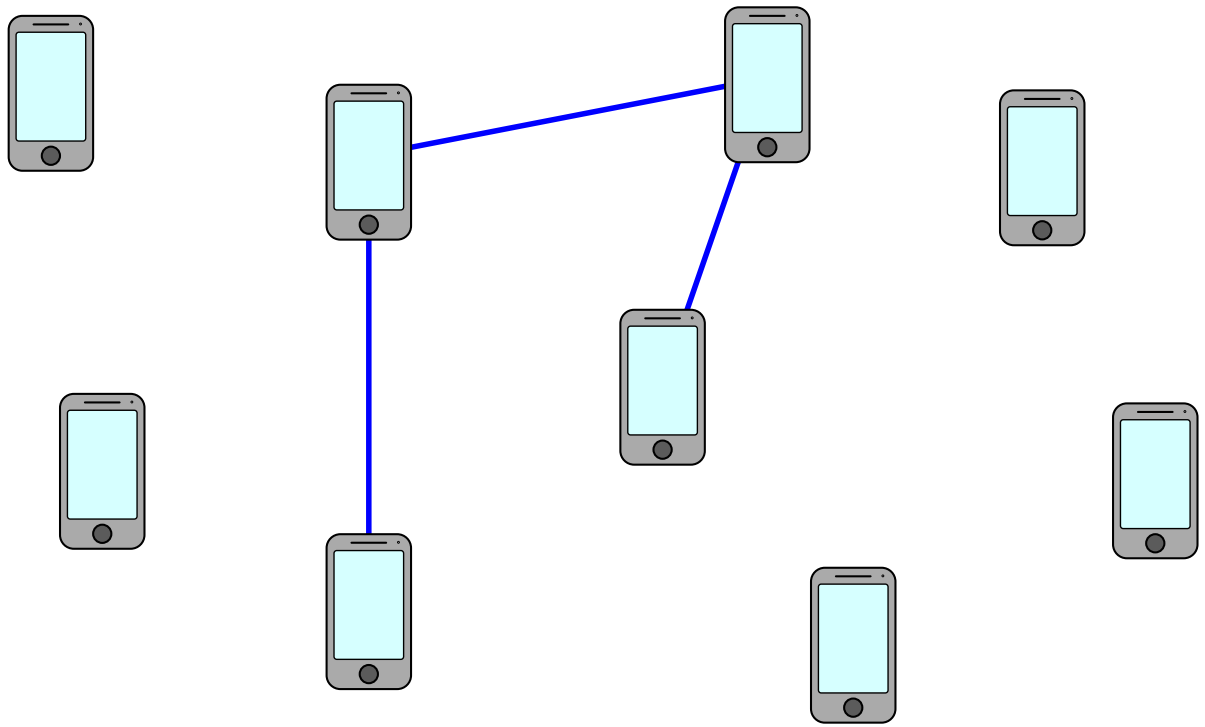
While the network is typically assumed to be (strongly) connected at every round, we may remove this assumption.



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

Disconnected networks

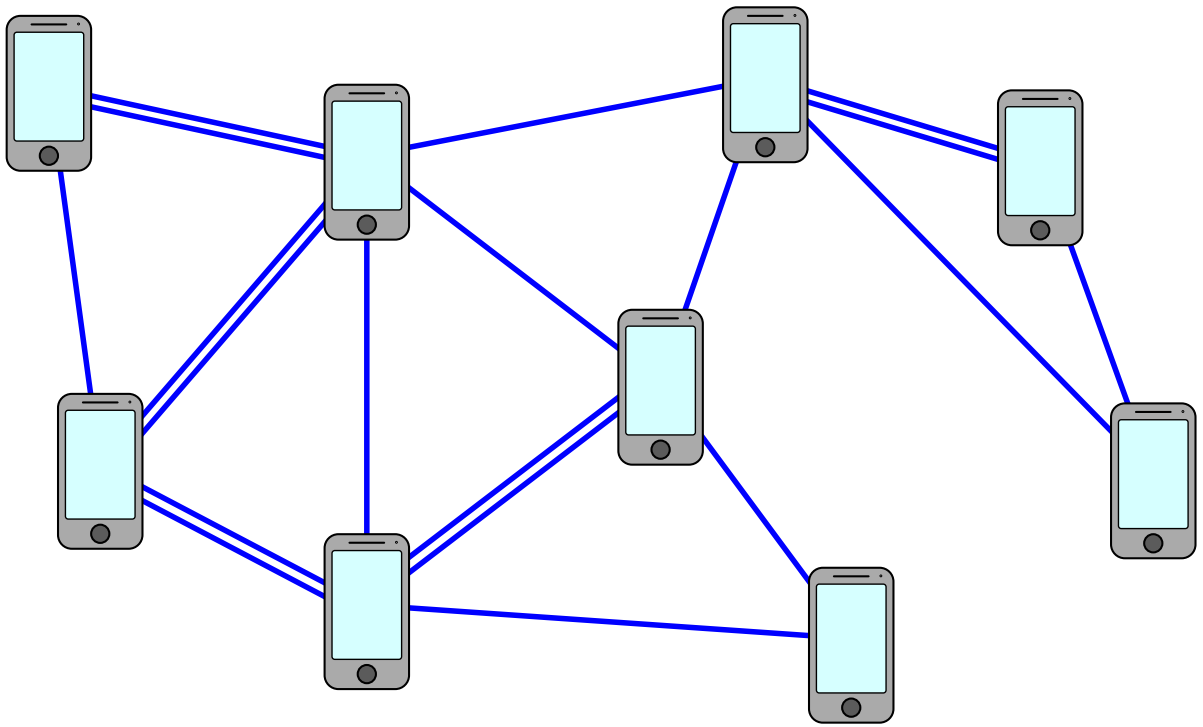
While the network is typically assumed to be (strongly) connected at every round, we may remove this assumption.



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

Disconnected networks

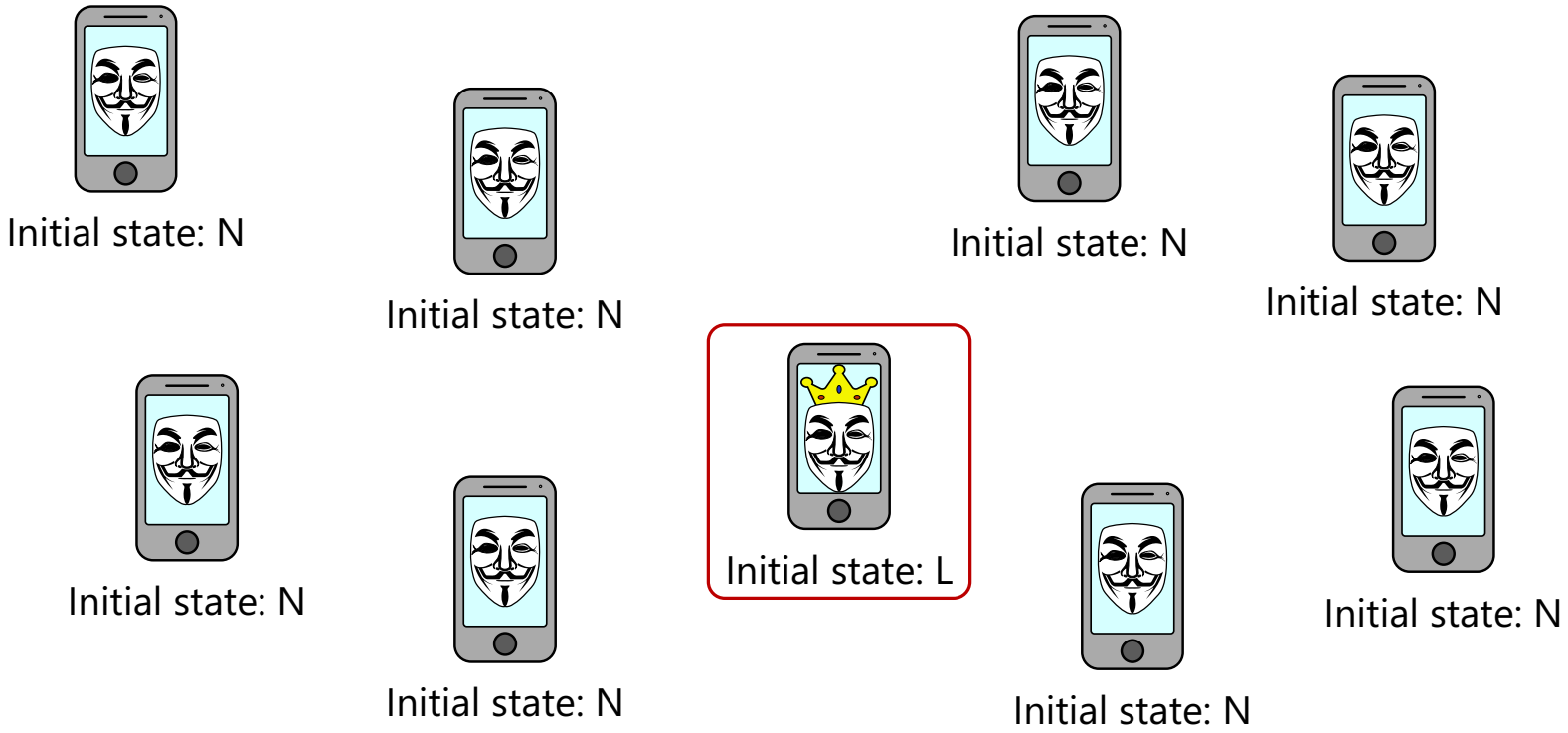
While the network is typically assumed to be (strongly) connected at every round, we may remove this assumption.



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

Anonymous networks

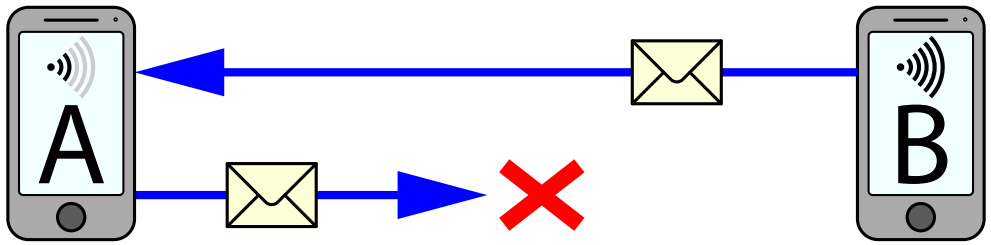
In this course, we will focus on *anonymous* networks, where agents do not have unique IDs, but are indistinguishable from each other.



Occasionally, we may assume the existence of one or more distinguished agents called *leaders*.

Directed links

The network's graph may be *undirected* or *directed*.



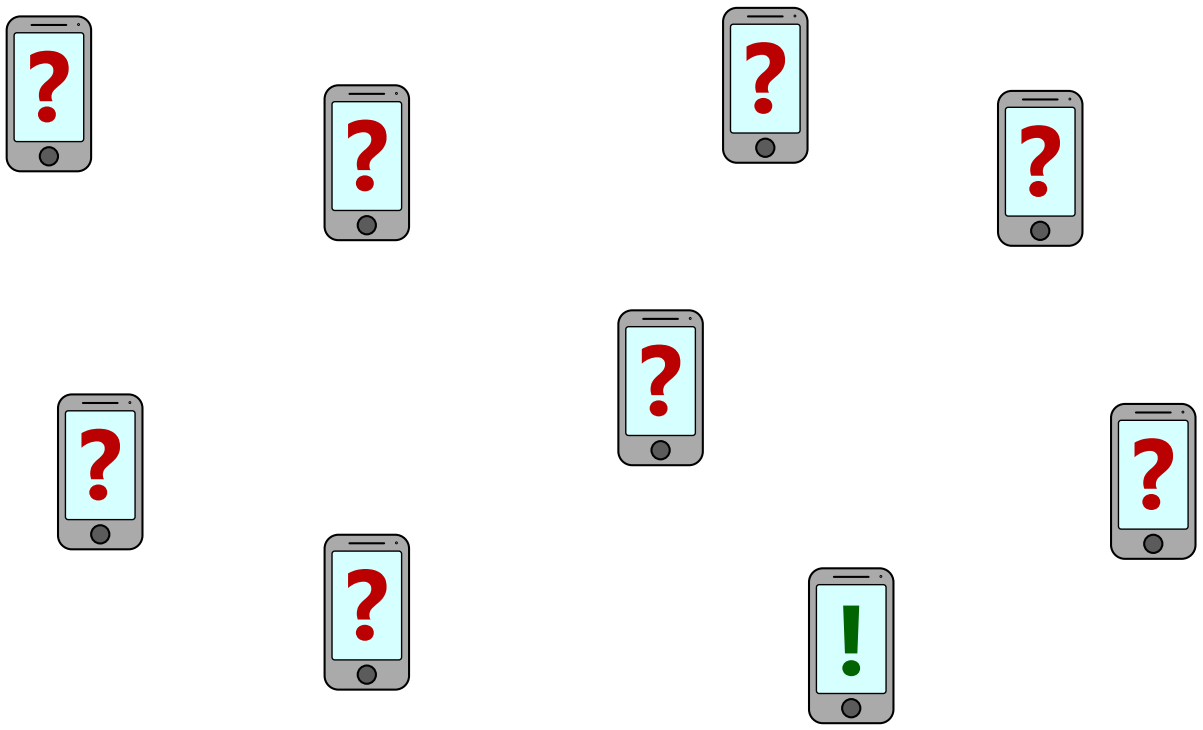
Directed networks may model a situation where agents have different communication ranges.

Other network models

- The communication network may be a *simple graph* or a *multigraph* (if there may be more than one link between the same two agents, or a link from an agent to itself).
- Communications may be *synchronous*, *asynchronous*, or *sequential*. Usually, we assume them to be synchronous.
- Interactions between processes may be *symmetric* or *asymmetric* (if there is an *initiator* and a *responder*). Usually, we assume them to be symmetric.
- Agents may have *port awareness* if their incident links have unique identifiers. A weaker assumption is *degree awareness*, where an agent (especially in a dynamic network) knows how many neighbors it has before sending them messages.
- Each agent may have an *input* assigned to it at the beginning of the first round. Agents may also be able to return an *output* as a result of their communications and computations.
- Agents may have *a-priori knowledge* about the network. E.g., number of agents, number of leaders, diameter, τ, etc.

Broadcasting

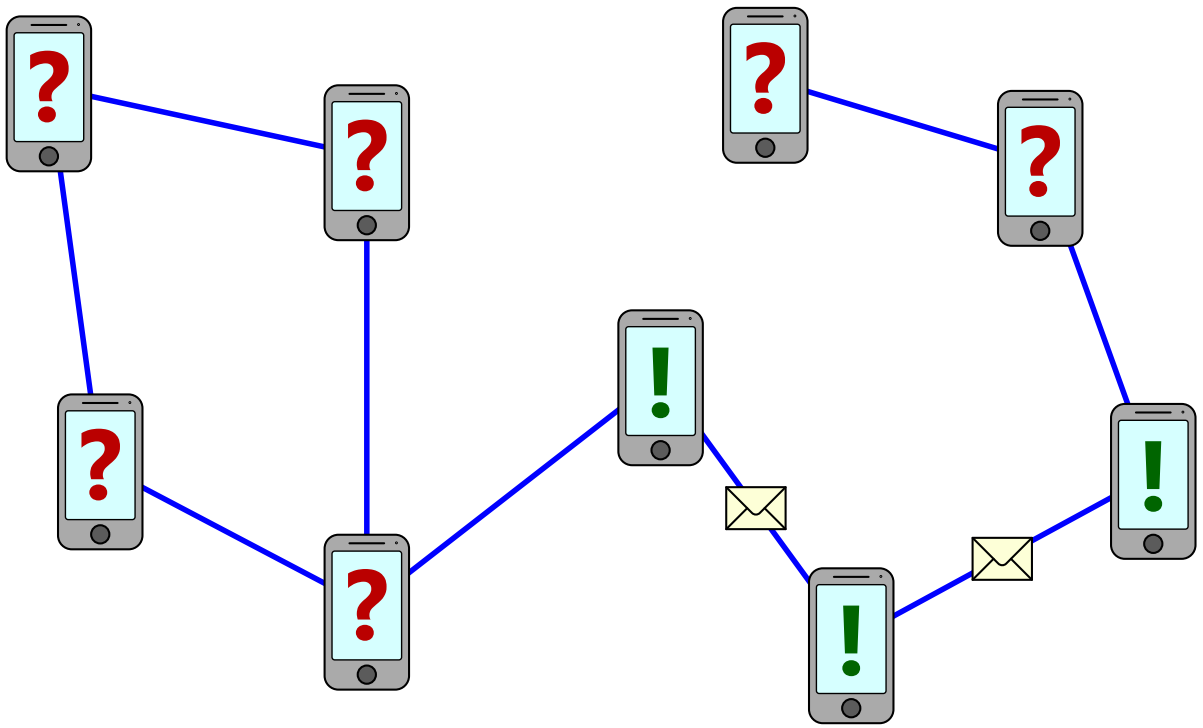
If a network of n agents is (strongly) connected at all rounds, information can be *broadcast* to all agents in $n - 1$ rounds.



Each agent simply has to forward the information to all its neighbors. Since the network is (strongly) connected, the number of agents that have the information grows at each round.

Broadcasting

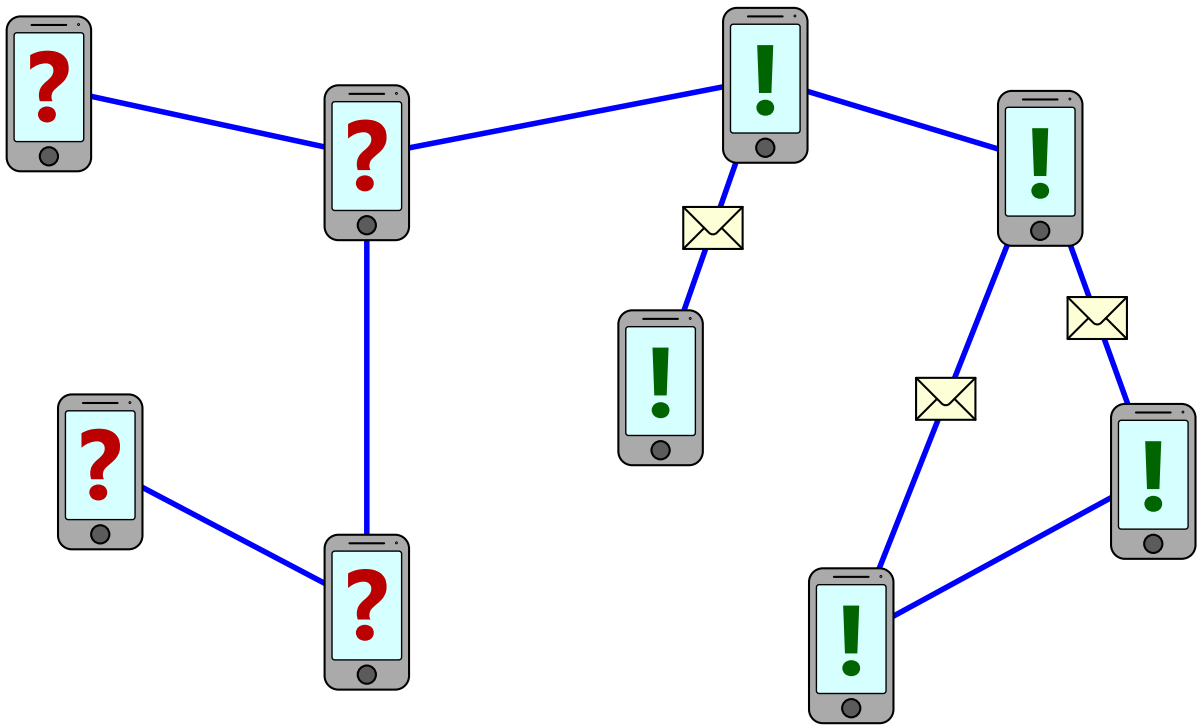
If a network of n agents is (strongly) connected at all rounds, information can be *broadcast* to all agents in $n - 1$ rounds.



Each agent simply has to forward the information to all its neighbors. Since the network is (strongly) connected, the number of agents that have the information grows at each round.

Broadcasting

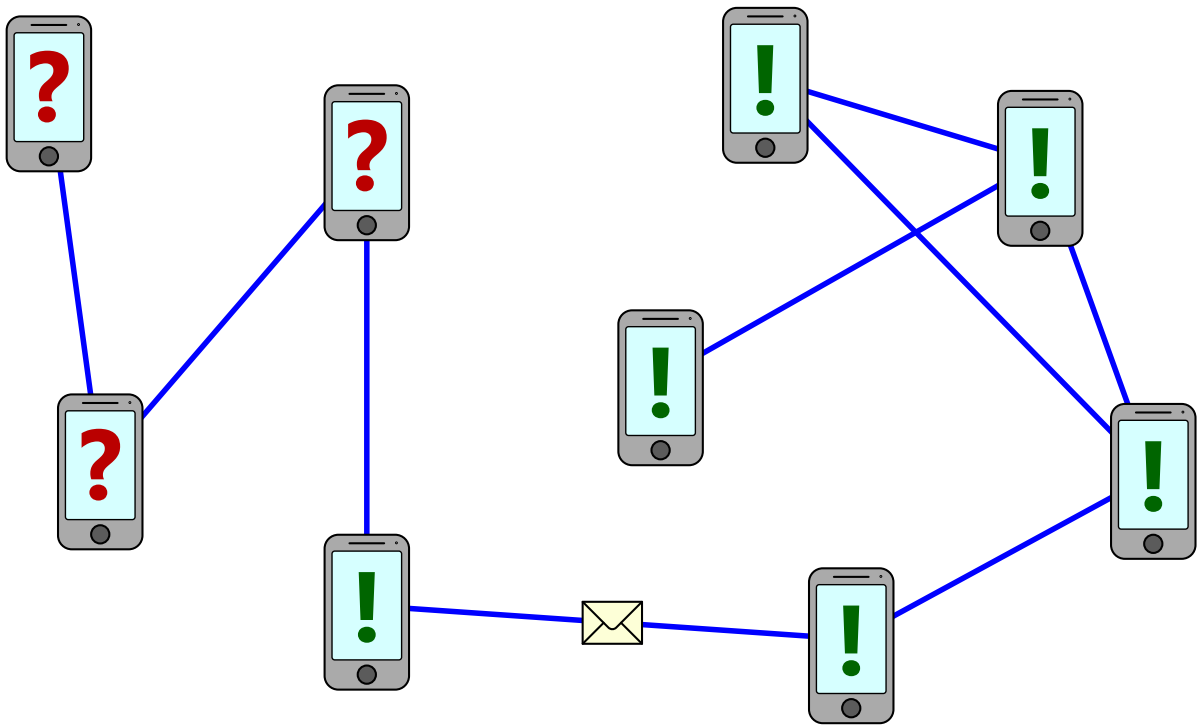
If a network of n agents is (strongly) connected at all rounds, information can be *broadcast* to all agents in $n - 1$ rounds.



Each agent simply has to forward the information to all its neighbors. Since the network is (strongly) connected, the number of agents that have the information grows at each round.

Broadcasting

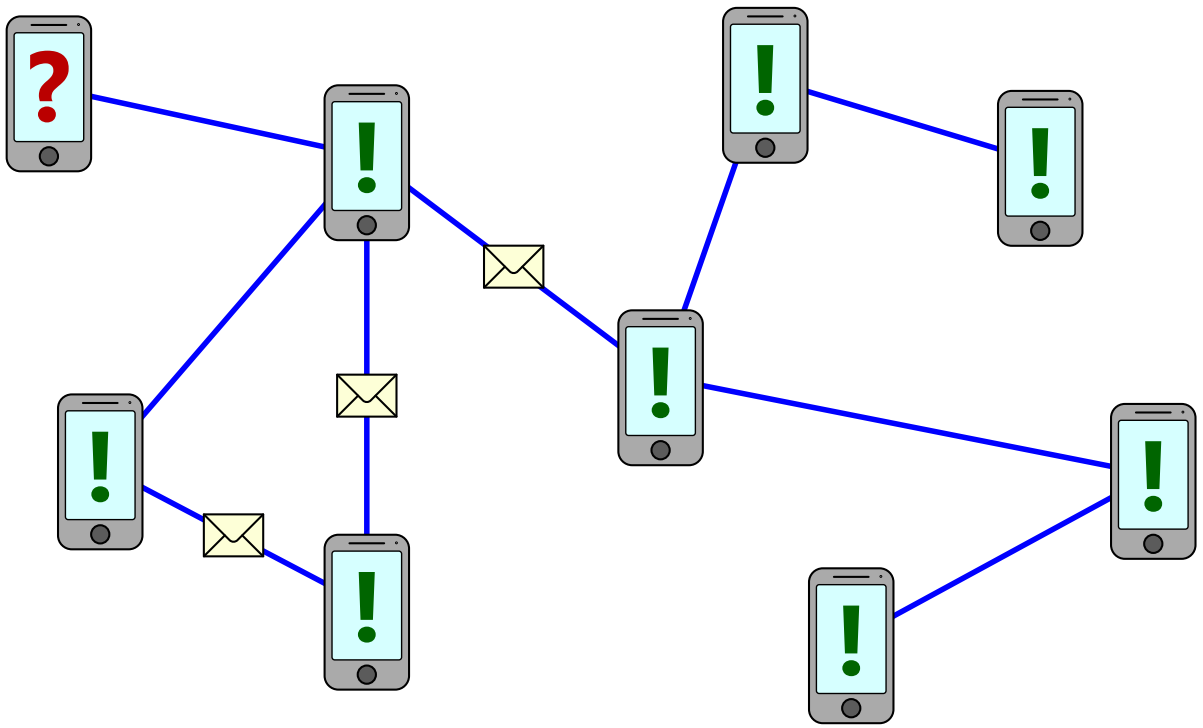
If a network of n agents is (strongly) connected at all rounds, information can be *broadcast* to all agents in $n - 1$ rounds.



Each agent simply has to forward the information to all its neighbors. Since the network is (strongly) connected, the number of agents that have the information grows at each round.

Broadcasting

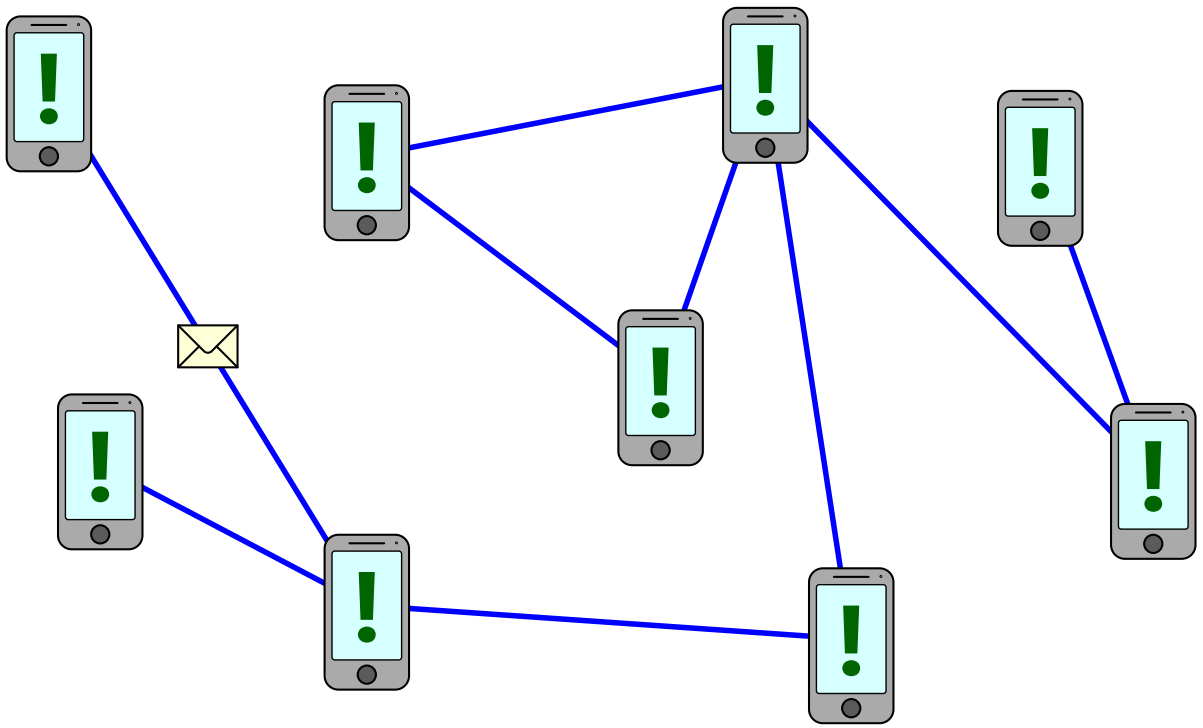
If a network of n agents is (strongly) connected at all rounds, information can be *broadcast* to all agents in $n - 1$ rounds.



Each agent simply has to forward the information to all its neighbors. Since the network is (strongly) connected, the number of agents that have the information grows at each round.

Broadcasting

If a network of n agents is (strongly) connected at all rounds, information can be *broadcast* to all agents in $n - 1$ rounds.



Each agent simply has to forward the information to all its neighbors. Since the network is (strongly) connected, the number of agents that have the information grows at each round.

Broadcasting

The broadcasting technique works even in dynamic and directed networks, assuming they are strongly connected at all rounds.

Applications of the broadcasting technique:

- **Simple broadcast:** one agent receives an input that has to be communicated to all agents.
- **Compute minimum/maximum:** all agents receive a number as input, and they have to find the minimum/maximum.
- **Compute input set:** all agents receive an input, and the goal is to find the set of all inputs (ignoring multiplicities).

These algorithms are:

- *Stabilizing* (i.e., give the correct output but may not realize that they are done) in $n-1$ rounds.
- *Terminating* in $n-1$ rounds if agents have a-priori knowledge of the (dynamic) diameter of the network (or an upper bound thereof, such as n).

All-to-all token dissemination

All-to-all token dissemination: special case of computing the input set when all agents are assumed to have *unique* inputs (i.e., there are n different tokens).

Local algorithm:

- At first, I only know my token.
- At each round, I send all the tokens I know to all my neighbors.
- If I receive any new tokens from my neighbors, I add them to the list of my known tokens.
- Increment a "round counter" at every round.
- When (round number) $=$ (number of known tokens), halt.

This algorithm terminates in n rounds with *no a-priori knowledge* about the network.

Leader election

Leader election: assuming all agents are anonymous, agree on a unique leader.

Assume that all agents have a-priori knowledge about the (static) network. A necessary condition for leader election is that the network graph has at least one vertex that cannot be moved to another vertex by an automorphism. *Is this condition sufficient?*

Leader or co-leader election

Assume that the network is a (static) tree. Then, it is possible to elect either a unique leader or a pair of adjacent *co-leaders*.

Local algorithm (*saturation*):

- At first, my status is *eligible*.
- At all rounds, I communicate my status to all neighbors and receive their statuses.
- If I am eligible and all my neighbors are *eliminated*, I become *leader* and halt.
- If I am eligible and all my neighbors except one are *eliminated*, I become a *candidate co-leader*.
- If I am a candidate co-leader and my non-eliminated neighbor is also a candidate co-leader, I become *co-leader* and halt.
- If I am a candidate co-leader and my non-eliminated neighbor is not a candidate co-leader, I become *eliminated* and halt.

It terminates in $O(n)$ rounds (even without port awareness).

Spanning tree construction

If there is *port awareness*, and a *spanning tree* of the network is given as input, then the previous algorithm can be adapted to elect a leader or a co-leader in a general (static) network, as well.

Conversely, if there is port awareness and a unique leader or two adjacent co-leaders are given, a *spanning tree* can be constructed.
Local algorithm (distributed breadth-first search):

- If there are two co-leaders, add the link connecting them to the spanning tree.
- The leader or the co-leaders become part of the spanning tree.
- If I have just become part of the spanning tree, I send a *join message* to all neighbors.
- If I am not part of the spanning tree and I receive join messages from some neighbors, select one of them and send it an *ack*. The link between us is added to the spanning tree, and I become part of the spanning tree, as well.
- If me and all my neighbors are part of the spanning tree, halt.

This algorithm terminates in $O(n)$ rounds.

Counting

Counting problem: determine the number of agents n.

Local algorithm assuming a connected static network with a unique leader (no port awareness):

- Phase 1:
  - The leader broadcasts a ping.
  - If I receive the ping at round t, then my distance from the leader is t.
  - After I know t, I communicate it to my neighbors.
  - I store the number d of my neighbors at distance $t-1$ from the leader.
  - If none of my neighbors has distance $t+1$ from the leader, I am a *farthest* agent.

Counting

- Phase 2:
  - If I am a farthest agent, I send the *mass* $1/d$ (as well as my distance t) to all my neighbors.
  - If I receive masses from neighbors at distance $t+1$ from the leader, I add them together, obtaining s, and send $(s+1)/d$ to all my neighbors (i.e., I add my own mass to the sum).
  - When the leader receives numbers from its neighbors, it adds them together, and adds 1 to the total: this is n.

- Phase 3:
  - The leader broadcasts n to all other agents.
  - After n rounds, everyone halts.

This algorithm terminates in $O(n)$ rounds and uses messages of size $O(\log n)$.

General computation

General computation: given an input to each process, determine the multiset of all inputs and compute an arbitrary function on it.

Assuming a static network with a unique leader, we can adapt the previous algorithm to this problem.

Essentially, instead of sending "a fraction of mass" toward the leader, the farthest agents send "a fraction of their input".

Then, intermediate agents add together these fractions as a *linear combination of inputs*, and add their own input to the sum.

Finally, the leader is able to reconstruct how many agents had which input.

Once the leader knows the multiset of all inputs, it can locally compute any arbitrary function on it, and broadcast the result to all agents.

Average consensus

Average consensus: compute the average of a multiset of input numbers.

Of course, the previous algorithm can solve the average consensus problem in a static undirected network with a unique leader.

What if the network is dynamic (with outdegree awareness), directed (strongly connected) and there is no leader?

Local algorithm (iterated averaging):

- Set $x := \text{my input}$.
- Repeat indefinitely:
  - If d is my outdegree for the current round, set $x := x/(d+1)$.
  - Send x to all neighbors.
  - Add all received values to x.

This algorithm may not compute the exact average in finite time, but *converges* to it, i.e., x becomes arbitrarily close to the average.