Modern Distributed Computing

Theory and Applications

Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 10
Tuesday, May 14, 2013

Part 4: Dynamic Synchronous Networks

1. Dynamic Graph Model, Causal Influence
2. Instantaneous Connectivity, T-interval connectivity, Possibly Disconnected
3. Anonymous Networks, Unknown Networks
4. Counting, Naming

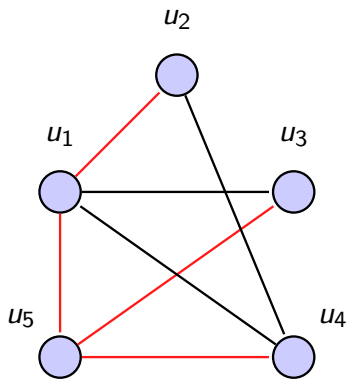
Study of Static Message Passing Model

- The assumption of static network does not reflect the real conditions of operation of distributed systems.
- However, it allows us to understand some fundamental aspects.
  - Synchronous execution, Asynchrony, Ordering of Events, Synchronization.
  - Fault-tolerance, Coordination, Symmetry Breaking.
- We investigated the correctness and performance of protocols
  - in terms of number of rounds – time complexity
  - in terms of number of message exchanges – communication complexity

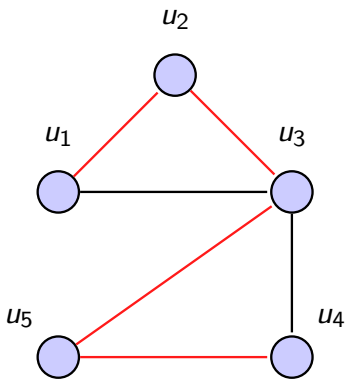
Study of Static Message Passing Model

- We define the network as a graph $G = (V, E)$:
  - comprised of a finite set V of points – the vertices – representing the processing units (i.e., processes) – $n = |V|$
  - a collection E of ordered pairs of elements of V ($E \subset [V]^2$) – the edges – representing the communication channels of the network
- Each process $u \in V$ is defined by a set of states $states_u$
  - A nonempty set of states $start_u$, known as starting states or initial states.
  - A nonempty set of states $halt_u$, known as halting states or terminating states.
- Each process
  - Is defined by a message-generator function and a state-transition function (Synchronous Model), or
  - An I/O Automaton (Asynchronous Model).

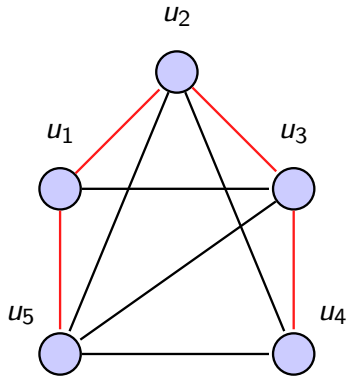
What if the communication channels are not fixed ?



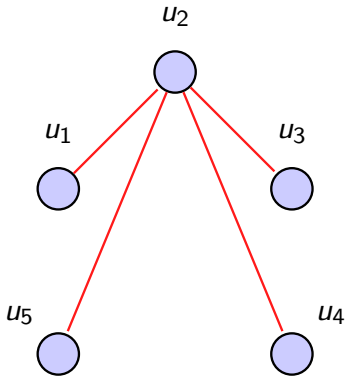
What if the communication channels are not fixed ?



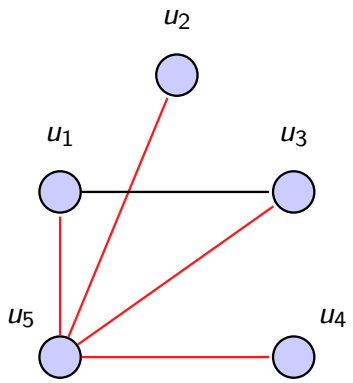
What if the communication channels are not fixed ?



What if the communication channels are not fixed ?



What if the communication channels are not fixed ?



Counting the Network Nodes

Counting Problem

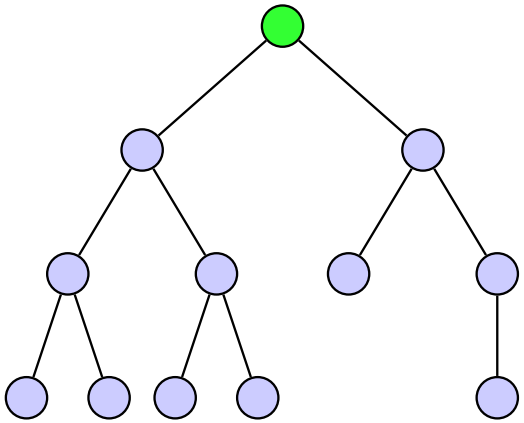
We wish to count the number of nodes participating in the network. Eventually all nodes terminate providing as output the size of the network n.

In a static synchronous network where nodes have unique ids:

- Each node u keeps a local list A_u containing all the unique IDs it has "heard",
- Each round it transmits the list to its neighbors,
- Upon receiving a list A_v from a neighboring node v it merges the contests of the lists with its own.
- After δ rounds, it terminates producing as output $|A_u|$.

Counting the Network Nodes

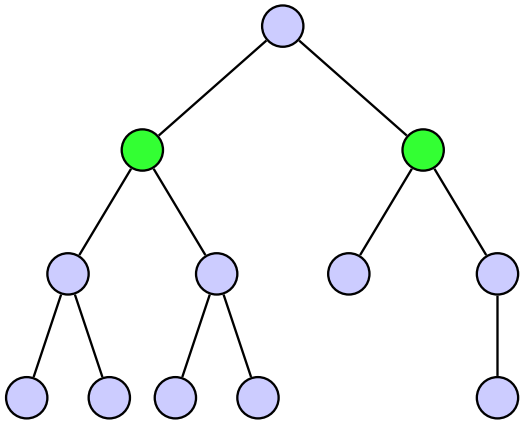
In a static synchronous network the algorithm is correct.



After $\delta = 4$ rounds, the algorithm terminates.

Counting the Network Nodes

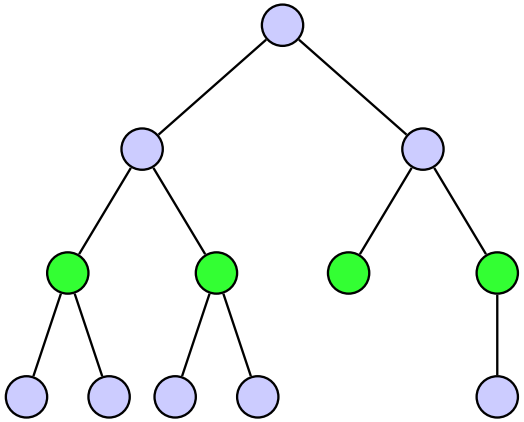
In a static synchronous network the algorithm is correct.



After $\delta = 4$ rounds, the algorithm terminates.

Counting the Network Nodes

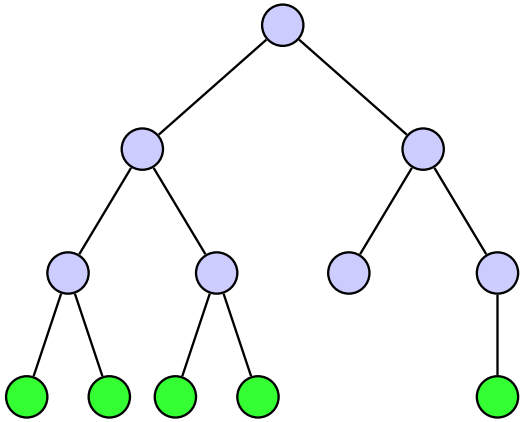
In a static synchronous network the algorithm is correct.



After $\delta = 4$ rounds, the algorithm terminates.

Counting the Network Nodes

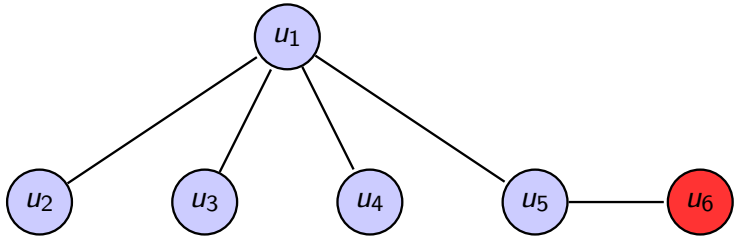
In a static synchronous network the algorithm is correct.



After $\delta = 4$ rounds, the algorithm terminates.

Counting the Network Nodes

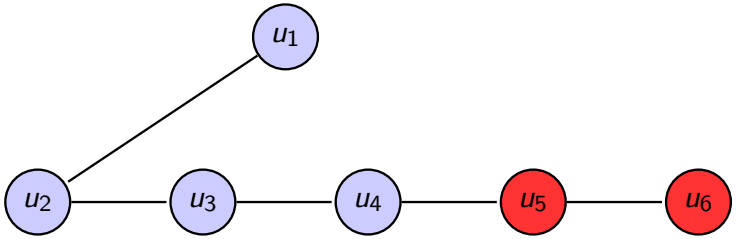
If the network is not static, the algorithm is not correct.



The diameter of the network is 3.

Counting the Network Nodes

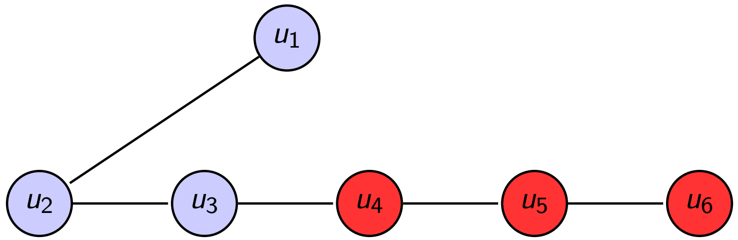
If the network is not static, the algorithm is not correct.



The diameter of the network changes to 5.

Counting the Network Nodes

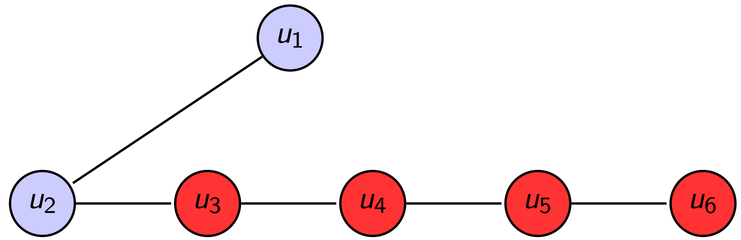
If the network is not static, the algorithm is not correct.



The diameter of the network changes to 5.

Counting the Network Nodes

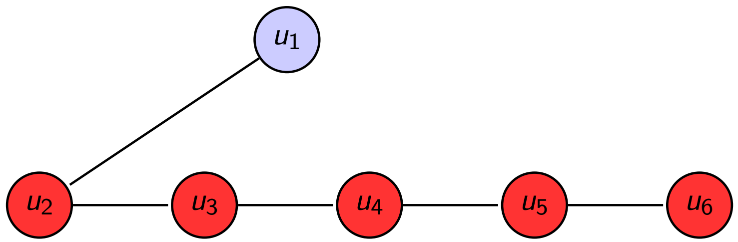
If the network is not static, the algorithm is not correct.



The diameter of the network changes to 5.

Counting the Network Nodes

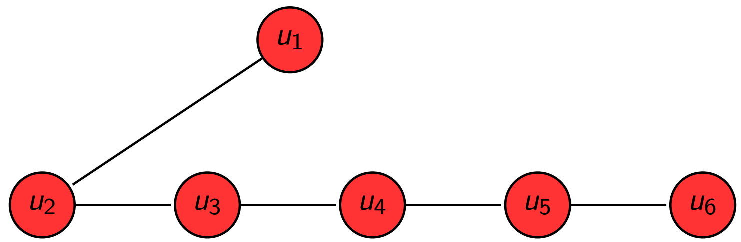
If the network is not static, the algorithm is not correct.



The diameter of the network changes to 5.

Counting the Network Nodes

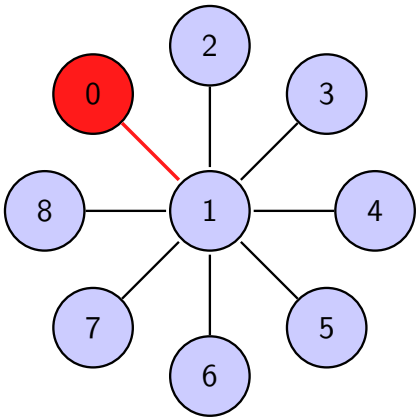
If the network is not static, the algorithm is not correct.



The diameter of the network changes to 5.

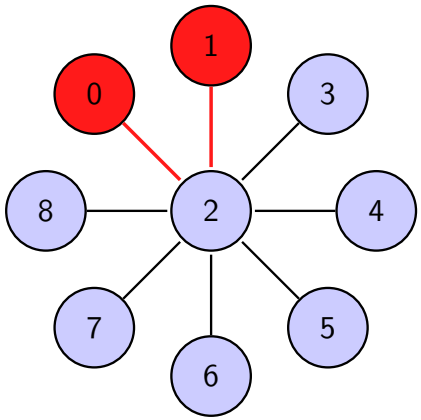
Diameter

How do we define the diameter in a dynamic network ?



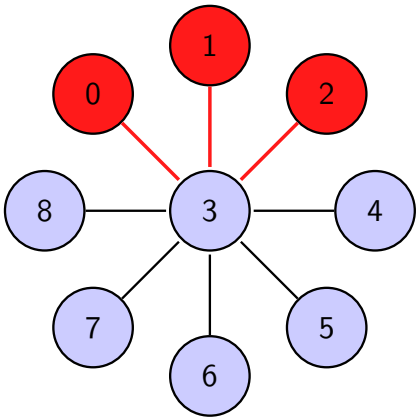
Diameter

How do we define the diameter in a dynamic network ?



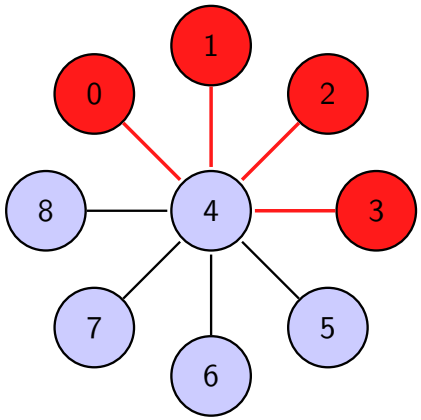
Diameter

How do we define the diameter in a dynamic network ?



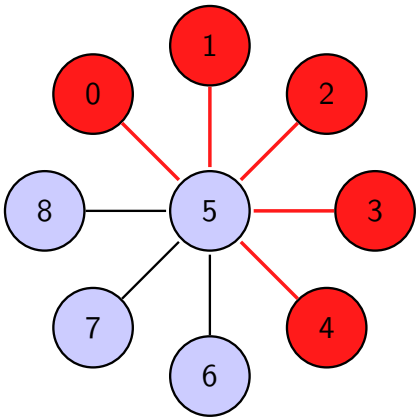
Diameter

How do we define the diameter in a dynamic network ?



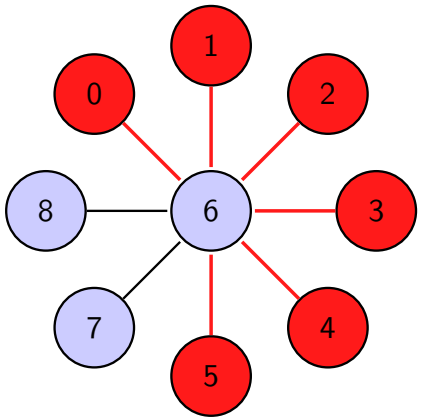
Diameter

How do we define the diameter in a dynamic network ?



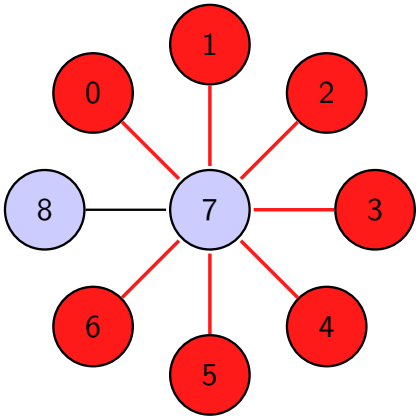
Diameter

How do we define the diameter in a dynamic network ?



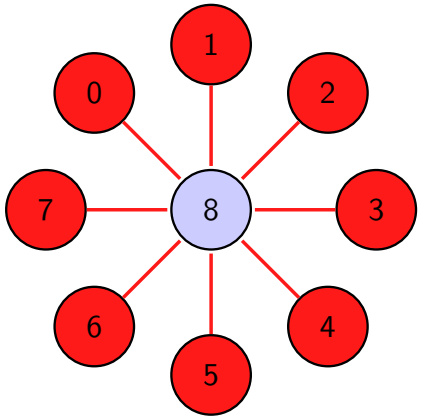
Diameter

How do we define the diameter in a dynamic network ?



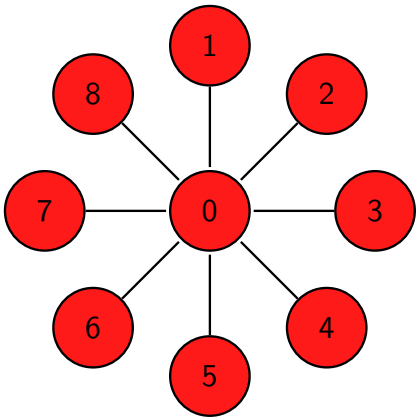
Diameter

How do we define the diameter in a dynamic network ?



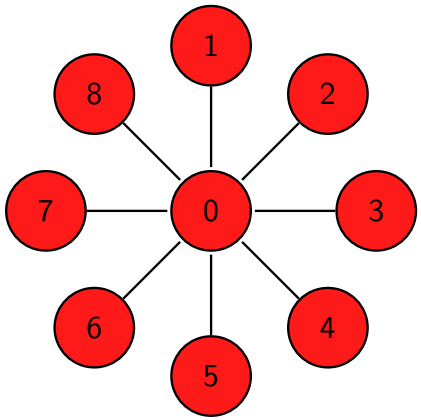
Diameter

How do we define the diameter in a dynamic network ?



Diameter

How do we define the diameter in a dynamic network ?



- In the static case, the diameter $\delta = 2$,
- while in the dynamic case, the diameter $\delta = n - 1$.

Modeling Dynamic Networks

- The nodes of the network remain static throughout the execution.
  - We do not consider additions/removals of nodes.
- Nodes are connected via communication channels that can go up/down throughout the execution.
  - The nodes cannot control when a communication channel will go up/down.
  - The nodes do not have any a-priori knowledge when a communication channel will go up/down.
- The communication channels go up/down with any, **arbitrary** order.
  - we do not assume any rate on the establishment/removal of communication channels.

Adversarial Analysis

- Real Life: A large number of undetermined factors affecting the system:
  - messages lost, channels fail arbitrarily (maybe temporarily), communication equipment "losing"' its state ...
- We wish our systems to operate in the worst case.
  - We wish to provide guarantees, "no matter what".
- We introduce the notion of an adversary that is controlling when channels go up/down.
  - We restrict the adversary so that some computation is possible.
  - The more we restrict the adversary, the easier it becomes.
  - The less restrictions we put on the adversary, our results are applicable to a broader range of cases.
- Necessary restriction: information must eventually spread
  - Advantage: results hold for all possible dynamicity/mobility patterns and not just specific cases or distributions

The Processes of the System

- The processes of the system
  - Have access to Unlimited local storage
  - Have unique identities (of $O(\log n)$ bits)
  - Defined by a set of states *states*
    - A nonempty set of states *init*, known as starting states or initial states.
    - A nonempty set of states *halt*, known as halting states or terminating states.
  - The state of process u at time/round r is denoted by (u, r).

The Dynamic Graph Model

- We define the network as a dynamic graph $G = (V, E)$ (or temporal or time-varying):
  - comprised of a finite set V of points – the vertices – representing the processing units (i.e., processes) – $n = |V|$ – that remain fixed throughout the execution.
  - a function $E : \mathbb{N}_{\geq 1} \to \mathcal{P}(\{\{u, v\} : u, v \in V\})$ mapping a round number r to a set $E(r)$ of undirected edges – representing the communication channels of the network as they become available during round r.

Synchronous Round of Execution

The system repeats in a "synchronized" manner the following steps – Round r:

1^{st} Step – Adversary's step

1. Examine processes state.
2. Determine $E(r)$.

2^{nd} Step (Send messages)

1. Generate messages for each outgoing neighbor.
2. Transmit messages over the corresponding channels.

3^{rd} Step (Receive Messages)

1. Process incoming messages and change internal state.
2. Remove all incoming messages from all channels.

T-interval Connectivity

Distributed Computation in Dynamic Networks – Fabian Kuhn, Nancy Lynch, Rotem Oshman, 2010.

- Represent dynamic networks that are connected at every instant.
- T represents the rate of connectivity changes.

T-interval Connectivity

We say a dynamic graph $G = (V, E)$ is T-interval connected, for $T \geq 1$, if, for all $r \in \mathbb{N}$, the static graph $G_{r,T} := (V, \bigcap_{i=r}^{r+T-1} E(r))$ is connected.

T-interval Connectivity

For example

- In 1-interval connected the underlying connected spanning subgraph may change arbitrarily from round to round.
- In ∞-interval connected a connected spanning subgraph is preserved forever.

Causal Influence (or Lamport Causality)

Time, clocks, and the ordering of events in a distributed system – Leslie Lamport, 1978

- Given a dynamic graph $G = (V, E)$
- We define an order $\rightarrow \subseteq (V \times \mathbb{N}_{\geq 0})^2$
- where $(u, r) \rightarrow (v, r + 1)$ iff $u = v$ or $\{u, v\} \in E(r + 1)$
- The Causal order $\rightsquigarrow \subseteq (V \times \mathbb{N}_{\geq 0})^2$ is defined as the reflexive and transitive closure of $\rightarrow$
- $(u, r) \rightsquigarrow (v, r')$: node u's state in round r influences node v's state in round r'
  - u "influences" v through a chain of messages originating at u and ending at v (possibly going through other nodes in between)

Two Useful Sets

1. Past set of a time-node (u, t') from time t
   - $\text{past}_{(u,t')}(t) := \{v \in V : (v, t) \rightsquigarrow (u, t')\}$
   - set of nodes whose t-state has causally influenced the t'-state of u
2. Future set of a time-node (u, t) at time t'
   - $\text{future}_{(u,t)}(t') := \{v \in V : (u, t) \rightsquigarrow (v, t')\}$
   - set of nodes whose t'-state has been causally influenced by the t-state of u

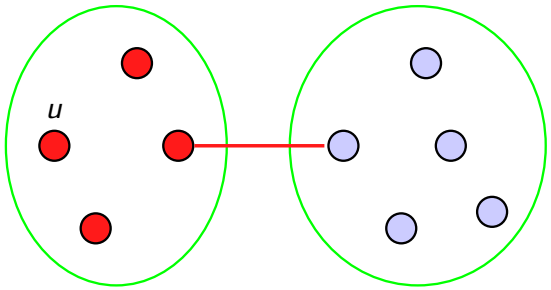
1-Token Dissemination in 1-interval Connected Graphs

During each round, all processes that know the token, broadcast it to all their neighbors. When a process receives the token, it outputs it immediately, but continues broadcasting it.

- Processes do not halt after they output the token.
- In 1-interval connected graphs, after $n - 1$ rounds, all nodes have received the token.
- Allow for constant propagation of information.

1-Token Dissemination in 1-interval Connected Graphs

- Consider a cut between the nodes that already received the token and those that have not.
- From 1-interval connectivity, there is an edge in the cut.
- The token is broadcast on that edge and some new node receives it.



1-Token Dissemination in 1-interval Connected Graphs

Lemma

For any node $u \in V$ and time $r \geq 0$ we have

1. $|\{v \in V : (u,0) \rightsquigarrow (v,r)\}| \geq \min\{r+1, n\}$,
2. $|\{v \in V : (v,0) \rightsquigarrow (u,r)\}| \geq \min\{r+1, n\}$.

- The dynamic diameter is small
  - At most linear in n
- For all times $t \geq 0$ the t-state of any node influences the $(t+n-1)$-state of every other node.

Counting in Dynamic Networks

- We use the causal influence property.
- All nodes that have identified u keep broadcasting the ID of u (id_u),
- Then every round, at least one more nodes hears the ID of u.

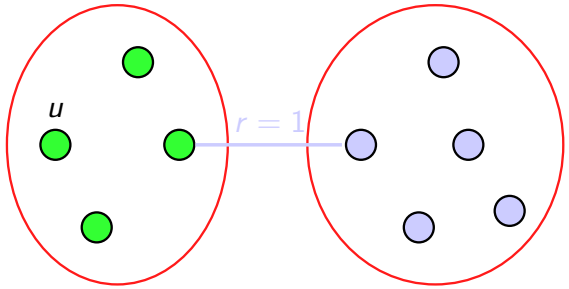
- How do we know when to stop?
- When does v know that it has heard all other nodes?

Lemma

For each round $r \geq 1$ and each node u, if all nodes re-transmit all the IDs they have heard, at the end of round r, node u must know at least $r+1$ (or all nodes if $r+1 \geq n$).

Counting in Dynamic Networks

Proof: Let $|A_u(r)| \leq r$.
We end up in a contradiction – u must have heard from another node $\notin A_u(r)$.



$|A_u(r)| \leq r$

After r rounds, at least r nodes must have heard from the right set. ■

Counting Algorithm

At each point, the processes have a guess k for the size of the network. Every process u keeps a list of IDs A_u with all the IDs it has received. At every round it transmits A_u to its neighbors. After k rounds, if it has discovered $|A_u| \geq k$, it doubles k and repeats; if $|A_u| < k$ it terminates by outputting $|A_u|$.

Time Complexity:

- Since we double k, we have $O(\log n)$ phases.
- Requires $O(n \log n)$ rounds.

Message complexity:

- Each node knows $\Theta(n)$ IDs, item Since each message has length $O(\log n)$,
- the complexity is $O(n \log n)$.

Other Fundamental Problems

Based on the previous algorithms and concepts we can also solve:

- id-dissemination
- Leader Election
- Evaluation of Predicates on Input – if all processes receive some input, then we can compute the minimum/maximum in $n - 1$ rounds.

Instantaneous Connectivity: State-of-the-art

Information dissemination in highly dynamic graphs – R. O'Dell and R. Wattenhofer, 2005.

- First appearance of the idea in an asynchronous setting.
- They studied flooding and routing.
- Flooding was solved in $O(Tn^2)$ rounds using $O(\log n)$ bit storage,
  - T is the maximum time it takes to transmit a message.
- Routing was solved in $O(Tn)$ rounds using $O(\log n)$ bit storage.

Instantaneous Connectivity: State-of-the-art

Distributed Computation in Dynamic Networks – F. Kuhn, N. Lynch, R. Oshman, 2010.

- T-interval connectivity was proposed.
- The synchronous case was studied for the first time.
- Counting and all-to-all token dissemination were solved in
  1. $O(n)$ rounds using $O(n \log n)$ bits per message
  2. $O(n^2/T)$ rounds using $O(\log n)$ bits per message
- They also gave the following lower bound:
  - Any deterministic centralized algorithm for k-token dissemination in 1-interval connected graphs requires at least, $\Omega(n \log k)$ rounds to complete in the worst case.

Anonymous Dynamic Networks

- The uniqueness of node IDs may not be guaranteed.
- In some cases distributed computation must be simple and with small resource requirements:
  - storing and exchanging IDs may be infeasible due to memory and communication constraints.
- We wish to avoid it to maintain user's privacy.

We refer to this setting as anonymous dynamic networks.

We study algorithms

- Count the size of the network,
- Provide unique labels to the processes of the network.

Existence of a Leader

- In order to solve these problems in anonymous dynamic networks we need to assume the existence of a leader process.
- The leader process
  - executes a different code,
  - may start from a different state.
- We use the leader process for symmetry breaking.
- If no such process exist, the problems are impossible to solve !
- Note that a static network is a special case of a dynamic network.

Counting and Naming in Anonymous Dynamic Networks

- No results were known for this type of worst-case dynamic networks
- All results that follow are from :
  Naming and counting in anonymous unknown dynamic networks – O. Michail, I. Chatzigiannakis, P. Spirakis, 2012.

Counting Protocol Degree_Count

- All processes keep track of the current round of execution (variable r incremented by 1 each round).
- The Leader process starts the computation by assigning labels to all neighboring processes - and does so every turn.
- All processes that receive a label from the leader, start assigning labels to all their neighboring process as well.
- Initially, one node (the leader) assigned to at most d processes label 1.
- Then the $d+1$ labeled processes assigned to at most $(d+1)d$ unlabeled processes the label 2, totaling $(d+1)+(d+1)d, \ldots$

Counting Protocol Degree_Count

- A process that has not received a label transmits a message "unassigned" signifying the round number.
- When the leader receives an unassigned message, it understands that at least $r + 1$ labels have been assigned.
  - As long as there are unlabeled processes one new label is assigned in each round to at least one node.
- The last "unassigned" message received signifies the last round during which new processes have been discovered.
- This helps get an estimation on the size of the network.

Counting Protocol Degree_Count

Theorem

Protocol Degree_Counting solves the counting upper bound problem in anonymous dynamic networks with broadcast under the assumption of a unique leader. The obtained upper bound is $O(d^n)$ (in the worst case).

- In the worst-case, each label in $\{0, 1, \ldots, n-1\}$ is assigned to precisely one node (e.g., consider a static line with the leader in the one endpoint). In this case the nodes count $O(d^n)$.
- If nodes have access to an upper bound e on the *maximum expansion*, defined as $\max_{u,r,r'}\{|\mathrm{future}_{u,r}(r'+1)| - |\mathrm{future}_{u,r}(r')|\}$ (maximum number of concurrent new influences ever occurring) the protocol provides an $O(n \cdot e)$ upper bound.

Labeling & Naming

k-Labeling

An algorithm is said to solve the k-labeling problem if whenever it is executed on a network comprising n nodes each node u eventually terminates and outputs a *label* (or *name* or *id*) id_u so that $|\bigcup_{u \in V} id_u| \geq k$.

Naming

The naming problem is a special case of the k-labeling problem in which it must additionally hold that $k = n$. This, in turn, implies that $id_u \neq id_v$ for all distinct $u, v \in V$ (so, unique labels are required for the nodes).

Naming Protocol Fair

- The unique leader assigns distinct labels to each processes of the network.
- The labels assigned are tuples (r, i), where
  - r is the round during which the label was assigned,
  - i is a unique number assigned by the leader.
- The labels can be uniquely ordered first by r, then by i (in ascending order).
- All processes keep track of the current round of execution (variable r incremented by 1 each round).
- Each round, the leader assigns a unique label to each neighboring process
  - Using i, and incrementing by 1 after each label assigned.

Naming Protocol Fair

- The protocol eventually computes a unique assignment for all the nodes, if we assume that the adversary is somehow fair:
  - The leader node at some point has become directly connected with each other node of the network.
- The protocol does not terminate.

Naming Protocol Delegate

- We extend the protocol Fair such that
  - All labeled nodes are allowed to label other nodes.
  - Without loss of generality, even if the leader does not encounter all other nodes of the network, due to the *connectivity property*, all nodes will eventually hear from the leader.
  - Therefore, all nodes will either receive a unique label from the leader or from another labeled node.
  - Thus it does not require the fairness assumption to be correct.
- The labels assigned are tuples (r, h, i), where
  - r is the round during which the label was assigned,
  - h is the label of the node that provided the label,
  - i is a unique number assigned by the leader.
- The labels can be uniquely ordered first by r, then by h and then by i (in ascending order).

Naming Protocol Delegate

- The protocol works similarly with Fair.
- All processes keep the following variables:
  - $state = \{anonymous, named, leader\}$
  - r – round counter
  - $counter$ – the number of labels generated
  - $label$ – the local label
- When a node receives a label from the leader, it stores the label received and sets $state = named$.
- Each round, the leader and all named processes will assign new labels to all their neighbors.
- The protocol eventually computes a unique assignment for all the nodes.
- The protocol does not terminate.

Naming Protocol Dynamic_Naming

- Already named nodes assign unique ids and acknowledge their id to the leader
  - Initially only the leader.
- All nodes constantly forward all assigned ids that they have heard of so that they eventually reach the leader.
- At some round r, the leader knows a set of assigned ids $K(r)$.
- The termination criterion:
  - If $|K(r)| \neq |V|$: in at most $|K(r)|$ additional rounds the leader must hear from a node outside $K(r)$
  - If $|K(r)| = |V|$: no new info will reach the leader in the future and the leader may terminate after the $|K(r)|$-round waiting period ellapses

Naming Protocol Dynamic_Naming

Theorem

Dynamic_Naming solves the naming problem in anonymous unknown dynamic networks given a unique leader. All nodes terminate in $O(n)$ rounds and use messages of size $\Theta(n^2)$.

- We can again reduce the message size to $\Theta(\log n)$ paying in $O(n^3)$ termination-time.
- The leader keeps track of the consecutive ids it has received so far.
- The leader confirms the new labels generated by other nodes after receiving the acknowledgements of the nodes.
- It re-assigns IDs so that they are consecutive.
- The named nodes do not immediately start assigning new names until they get a confirmation from the leader.

Possibly Disconnected Dynamic Networks

Not all dynamic networks have connected instances

- Most natural dynamic networks are only temporally connected
- There are dynamic networks with always disconnected instances in which information spreads as fast as in those with always connected instances
- No results were known for this type of worst-case dynamic networks
- All results that follow are from :
  Causality, Influence, and Computation in Possibly Disconnected Synchronous Dynamic Networks – O. Michail, I. Chatzigiannakis, P. Spirakis, 2012.

Metrics for Disconnectivity

1. Outgoing Influence Time (oit)
   - Maximal time until the state of a node influences the state of another node
2. Incoming Influence Time (iit)
   - Maximal time until the state of a node is influenced by the state of another node.
3. Connectivity Time (ct)
   - Maximal time until the two parts of any cut of the network become connected

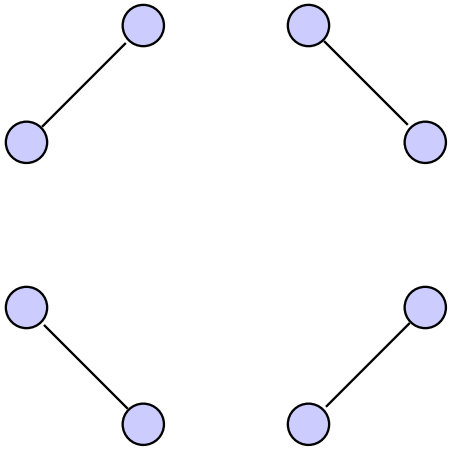
oit – Outgoing Influence Time

- Maximal time until the state of a node influences the state of another node
- Minimum $k \in \mathbb{N}$ s.t. for all $u \in V$ and all times $t, t' \geq 0$ s.t. $t' \geq t$ it holds that

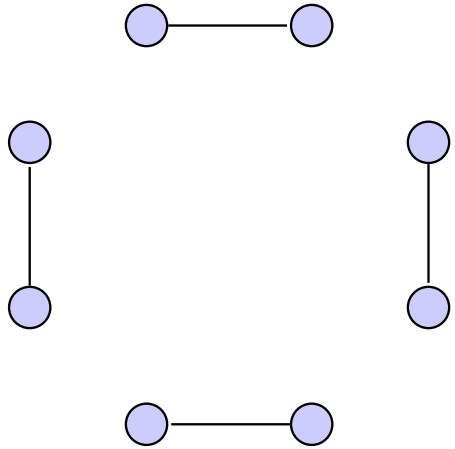
$$|\text{future}_{(u,t)}(t'+k)| \geq \min\{|\text{future}_{(u,t)}(t')|+1, n\}$$

- Example: the oit of a T-interval connected graph is 1
- If a dynamic graph $G = (V, E)$ has oit 1 then every instance has at least $\lceil n/2 \rceil$ edges.

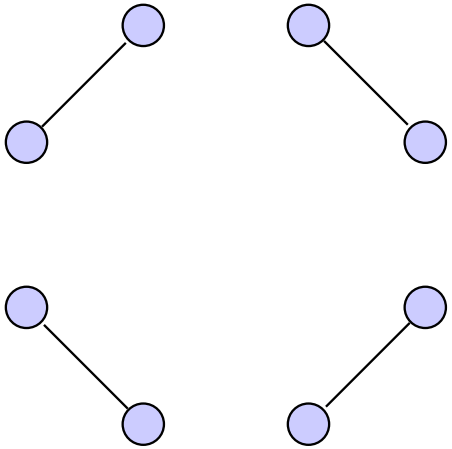
Alternating Matchings



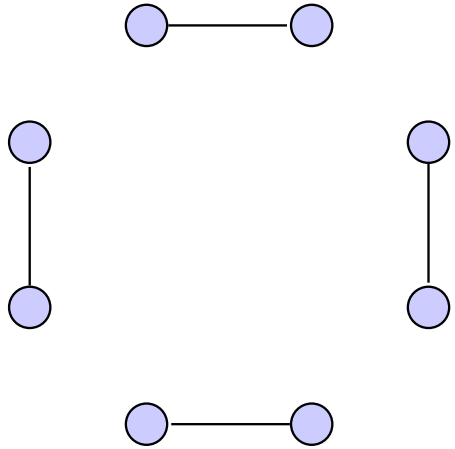
Alternating Matchings



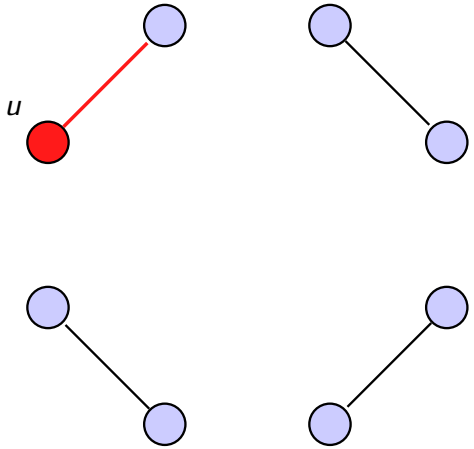
Alternating Matchings



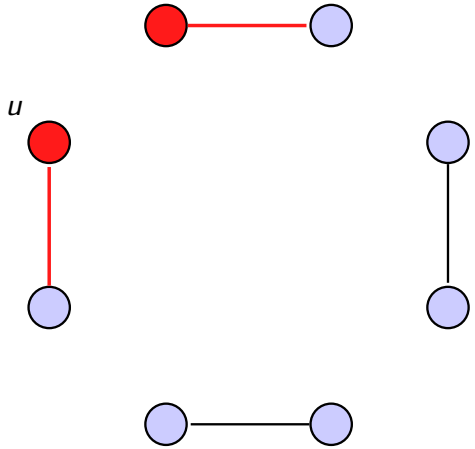
Alternating Matchings



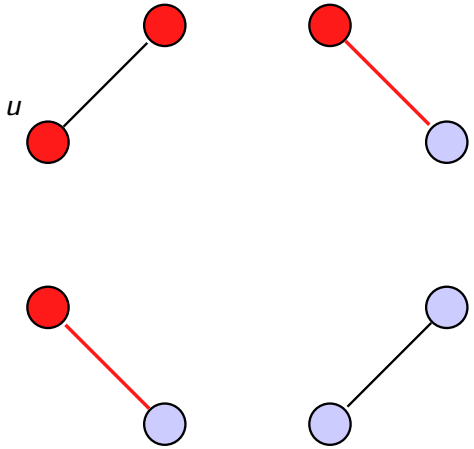
Alternating Matchings (oit=1)



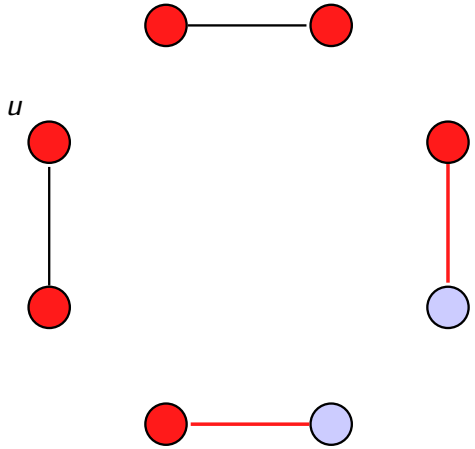
Alternating Matchings (oit=1)



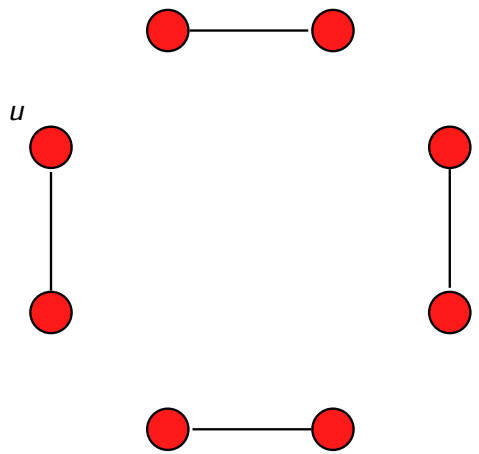
Alternating Matchings (oit=1)



Alternating Matchings (oit=1)



Alternating Matchings (oit=1)



iit – Incoming Influence Time

- Maximal time until the state of a node is influenced by the state of another node
- Minimum $k \in \mathbb{N}$ s.t. for all $u \in V$ and all times $t, t' \geq 0$ s.t. $t' \geq t$ it holds that

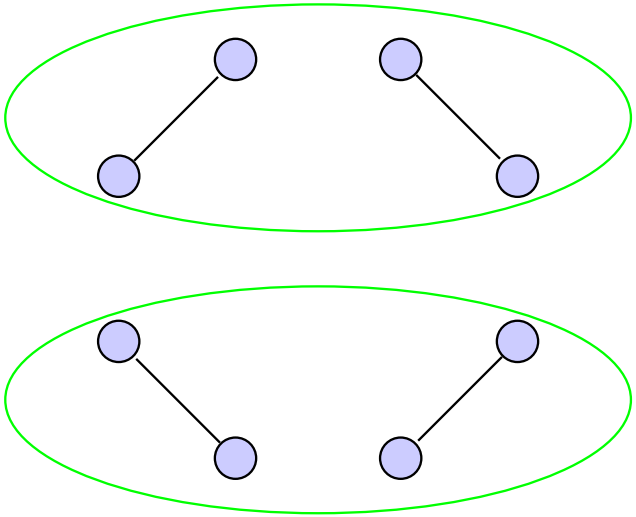
$$|\mathrm{past}_{(u,t'+k)}(t)| \geq \min\{|\mathrm{past}_{(u,t')}(t)| + 1, n\}$$

- Example: the iit of a T-interval connected graph can be up to $n-2$

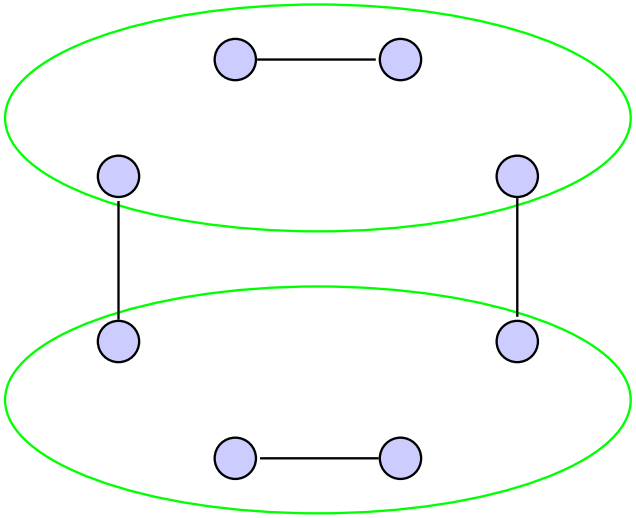
ct – Connectivity Time

- Maximal time until the two parts of any cut of the network become connected
- Minimum $k \in \mathbb{N}$ s.t. for all times $t \in \mathbb{N}$ the static graph $(V, \bigcup_{i=t}^{t+k-1} E(i))$ is connected
- If the ct is 1 then we obtain a 1-interval connected graph
- Greater ct allows for disconnected instances

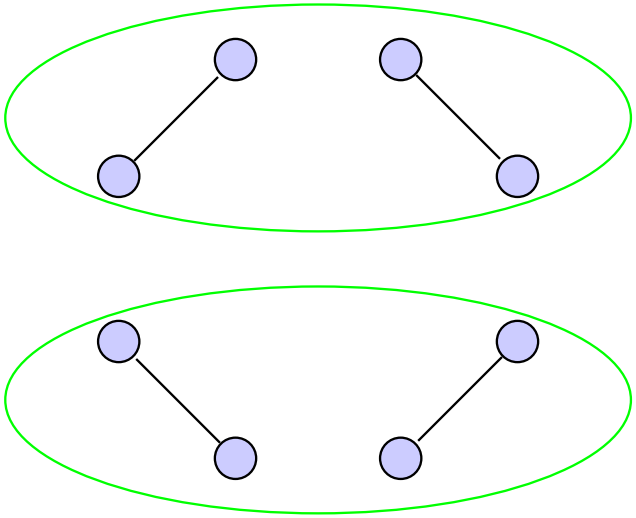
Alternating Matchings (ct=2)



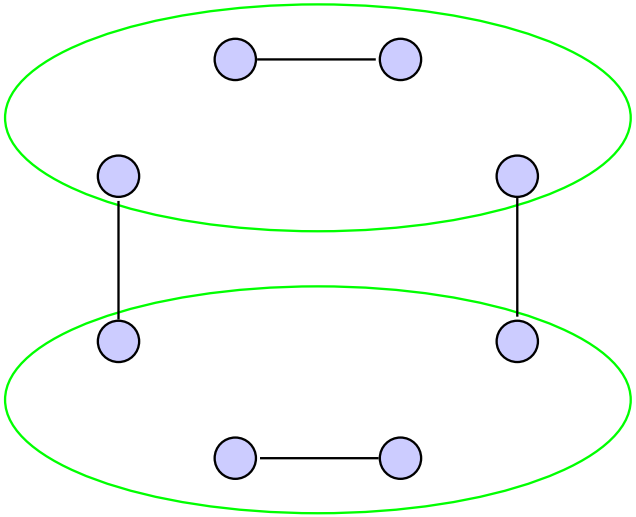
Alternating Matchings (ct=2)



Alternating Matchings (ct=2)



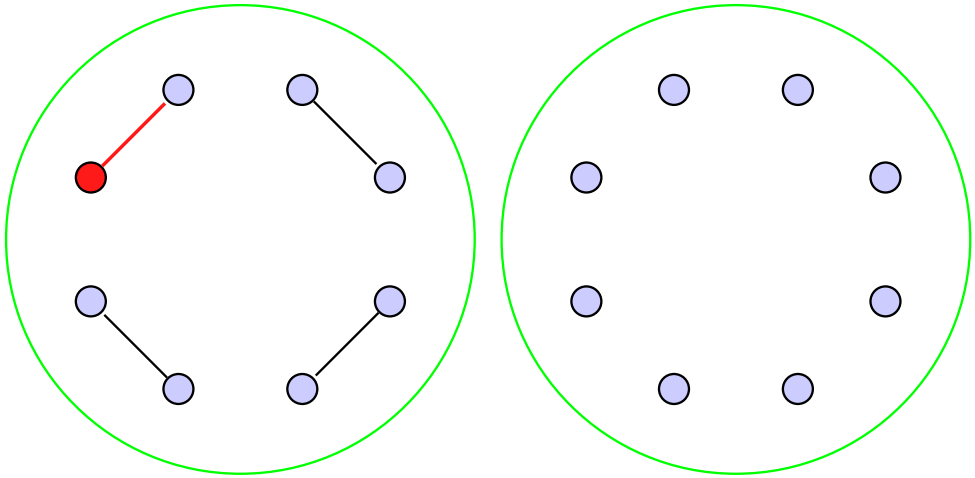
Alternating Matchings (ct=2)



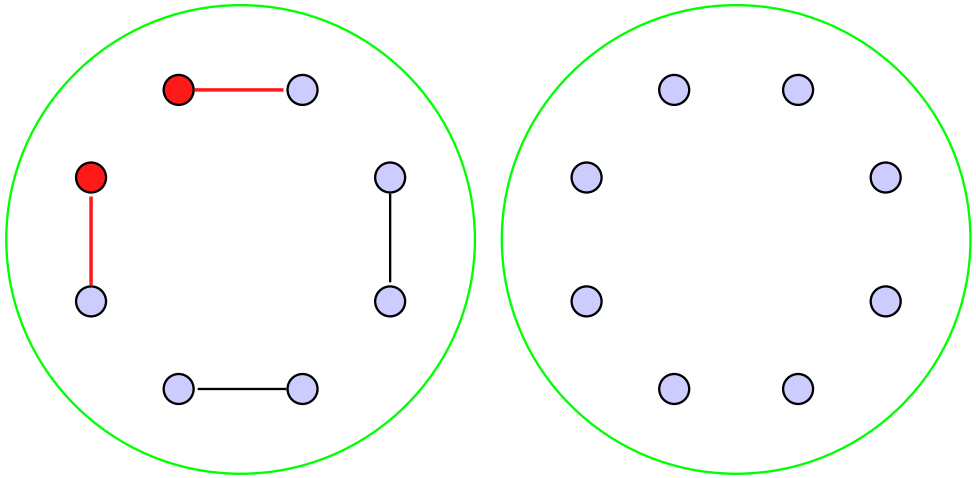
oit vs ct

1. oit $\leq$ ct but
2. there is a dynamic graph with oit 1 and ct $= \Omega(n)$.

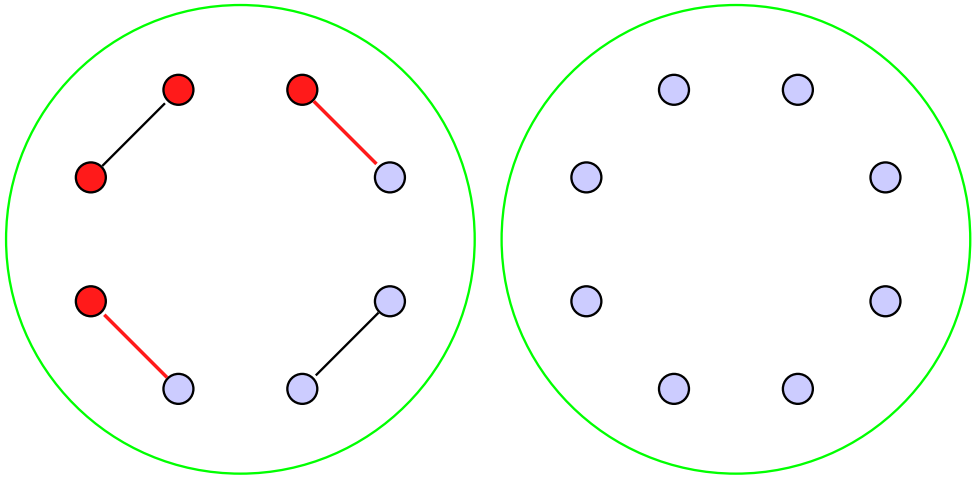
oit $= 1$ and ct $= \Omega(n)$



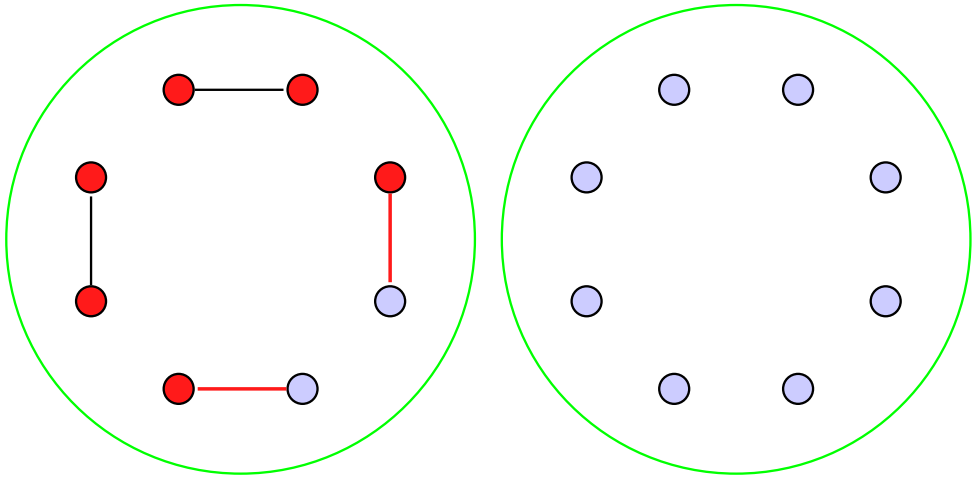
oit $= 1$ and ct $= \Omega(n)$



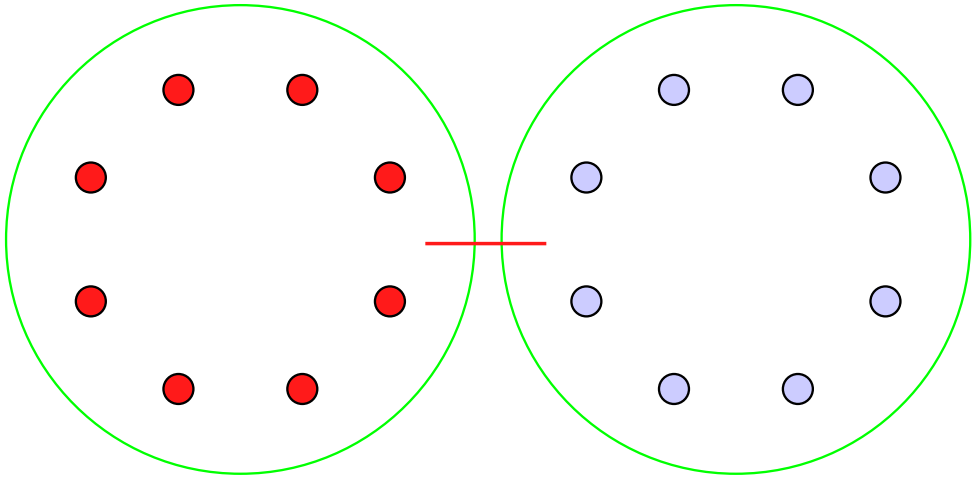
oit $= 1$ and ct $= \Omega(n)$



oit $= 1$ and ct $= \Omega(n)$



oit $= 1$ and ct $= \Omega(n)$



Counting in Disconnected Dynamic Graphs

- We assume that all processes know some upper bound T on the ct.
- We wish a process to determine whether it has heard from all nodes in the graph.
- Each process keeps track of its past sets from time 0 and from time T.
- All processes constantly forward all 0-states and T-states of processes that they have heard of so far.
  - List of ids accompanied with 0 and T timestamps
- When these two sets become equal the algorithm terminates.

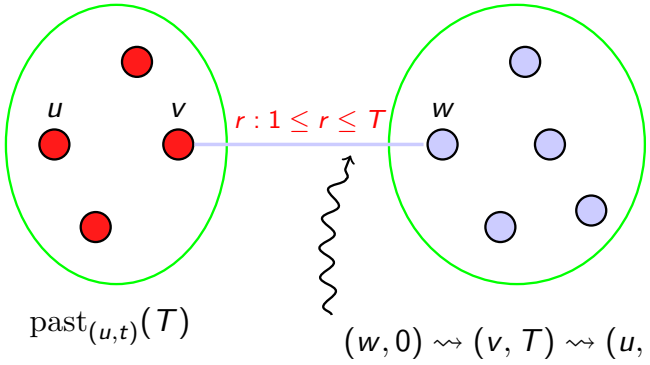
Inspired by time 0 and time 1 comparisons stated in *Coordinated consensus in dynamic networks* – F. Kuhn, R. Oshman, Y. Moses, 2011

Counting in Disconnected Dynamic Graphs

Theorem (Repeated Past)

Process u knows at time t that $\text{past}_{(u,t)}(0) = V$ *iff* $\text{past}_{(u,t)}(0) = \text{past}_{(u,t)}(T)$.

If $|\text{past}_{(u,t)}(0)| \geq \min\{|\text{past}_{(u,t)}(T)| + 1, n\}$.



Counting in Disconnected Dynamic Graphs

Only if:

- $v \in \text{past}_{(u,t)}(0) \backslash \text{past}_{(u,t)}(T)$
- u has not heard from v since time $r < T$
- Arbitrarily many nodes connected to no node until time $r - 1$ and only to v thereafter
- Thus, arbitrarily many nodes may be concealed from u
- Implies that even if $\text{past}_{(u,t)}(0) = V$ node u cannot know it

■

Suitable for counting, all-to-all token dissemination, and functions on inputs which is optimal, requiring $O(D + T)$ rounds in any dynamic network with dynamic diameter D.