The Power of a Leader In the Stone Age

Stephan Holzer - MIT

Yuval Emek - Technion
Roger Wattenhofer - ETH Zürich

2nd Workshop on Biological Distributed Algorithms, October 11-12, 2014 in Austin, Texas USA

The Power of a Leader

In the Stone Age

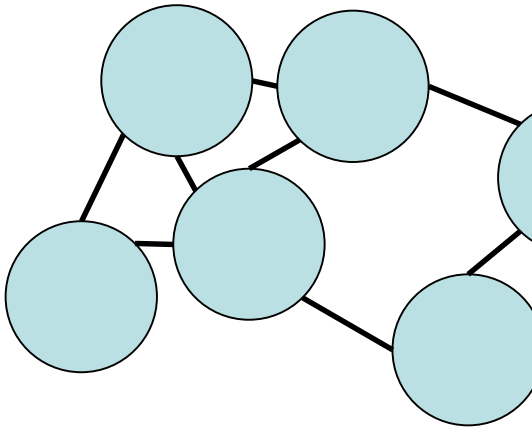
Work in progress!

Stephan Holzer - MIT

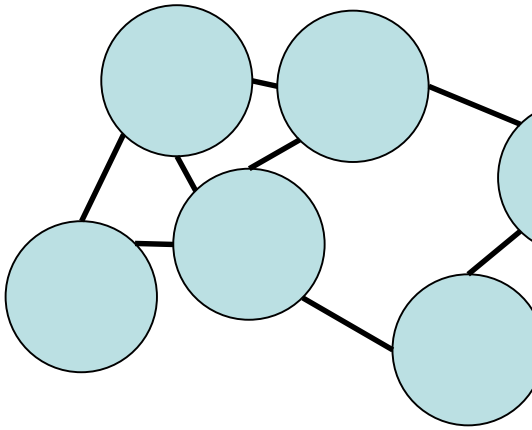
Yuval Emek - Technion
Roger Wattenhofer - ETH Zürich

2nd Workshop on Biological Distributed Algorithms, October 11-12, 2014 in Austin, Texas USA

Stone Age Model of Distributed Computing

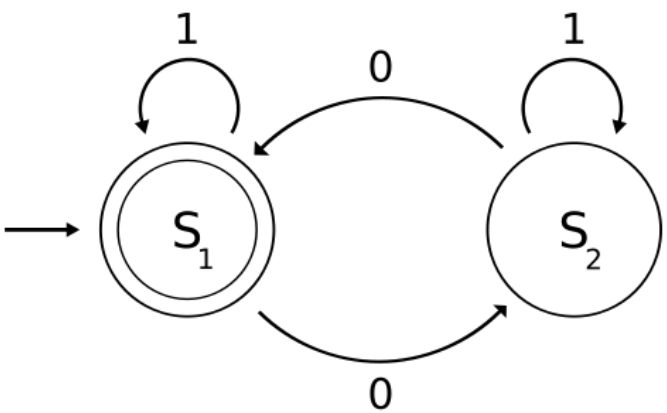


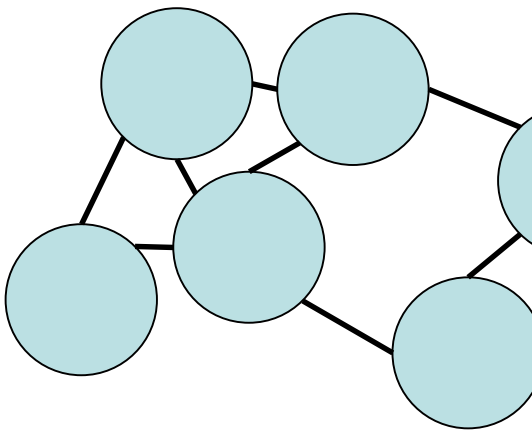
Computational Power of a Cell?



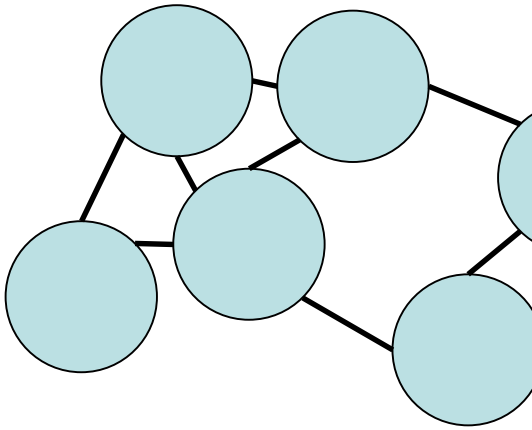
Computational Power of a Cell?

- Finite State Machine



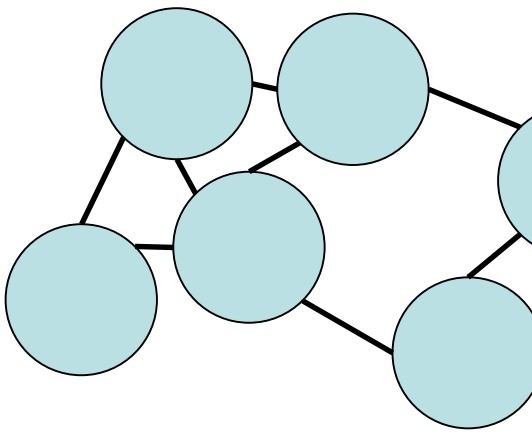


Communication?



Communication?

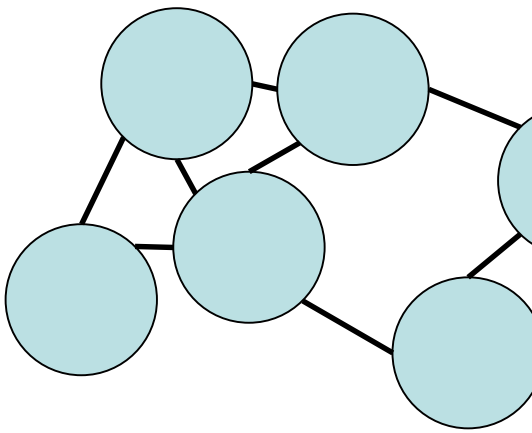
- Transmissions:
  - Same message delivered to all neighbors
- Constant size message
- Port for each neighbor
  - Stores the last message delivered



Communication?

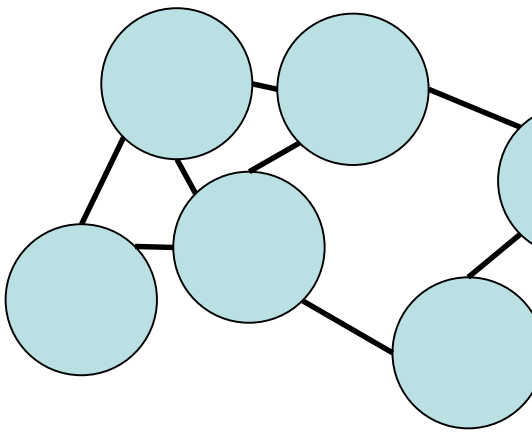
- Transmissions:
  - Same message delivered to all neighbors
- Constant size message
- Port for each neighbor
  - Stores the last message delivered
- Can detect if 0, 1 or 2+ ports store message m
- FSM changes state based on this

  sends message based on state



Stone Age Model of Distributed Computing

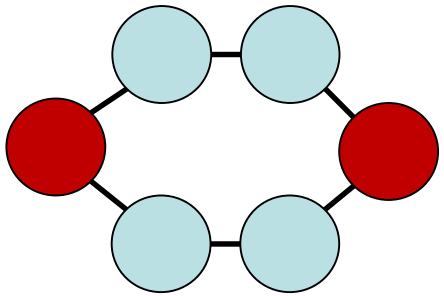
- All nodes run the same FSM
  - (random)
- Anonymous
- Weak communication
- Fully asynchronous
- Arbitrary network topology
  - (unknown)



- All features of the protocol are constant!

What is known?

- Can be synchronized
- Cannot elect a leader
- Cannot compute shortest paths
  - No Minimum Spanning Tree or Diameter



Edsger W. Dijkstra:

Actually:
I CAN COMPUTE SHORTEST PATHS!

Edsger W. Dijkstra:

Actually:
I CAN COMPUTE
SHORTEST PATHS!

Actually:
Me too!

(The slime mold)

Physarum polycephalum

- Nuclei are nodes
- Tubes / plasma are edges

?

Computes the Shortest Path

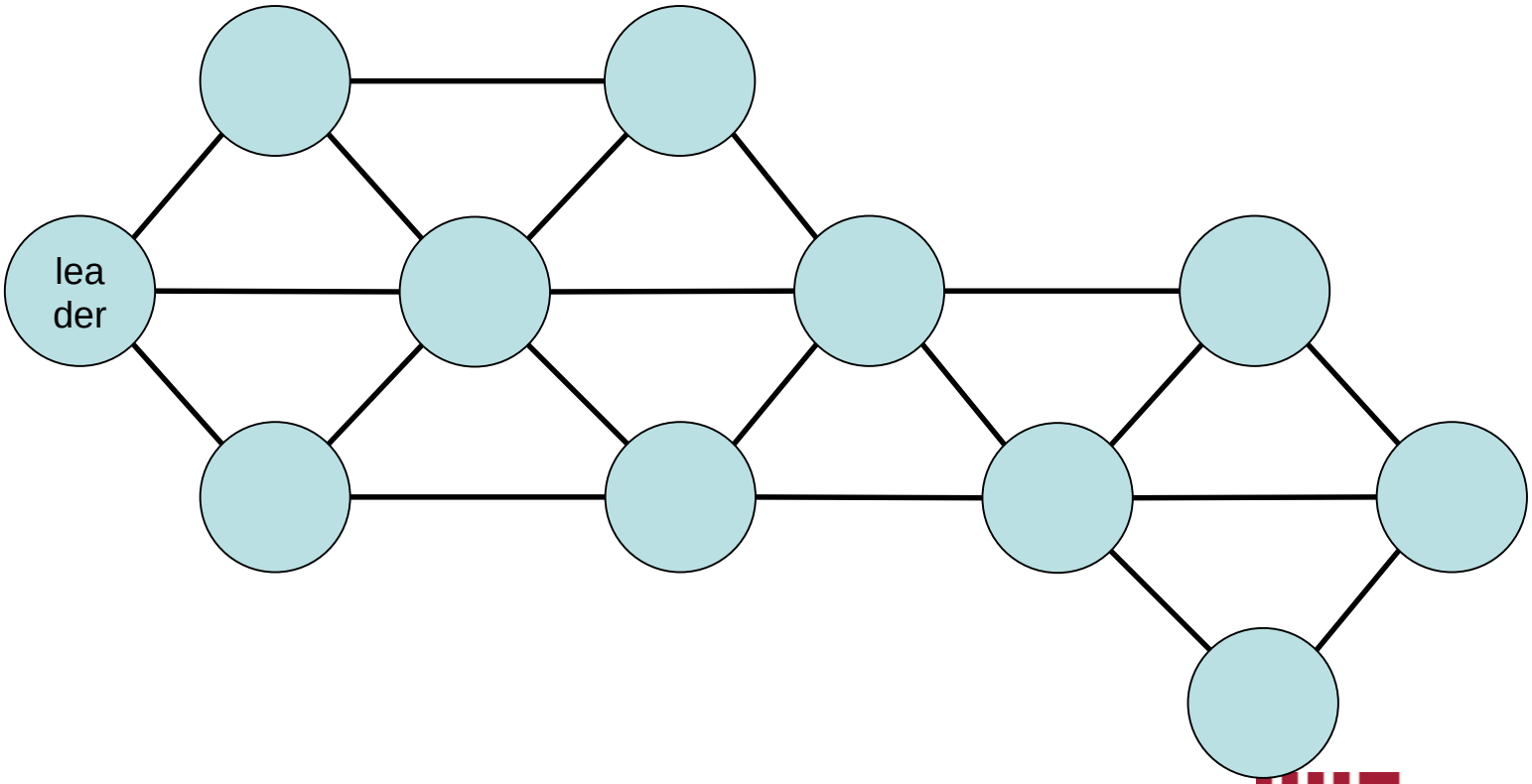
Video can be found at:
http://www.youtube.com/watch?v=czk4xgdhdY4

How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- E.g. select unique node at random

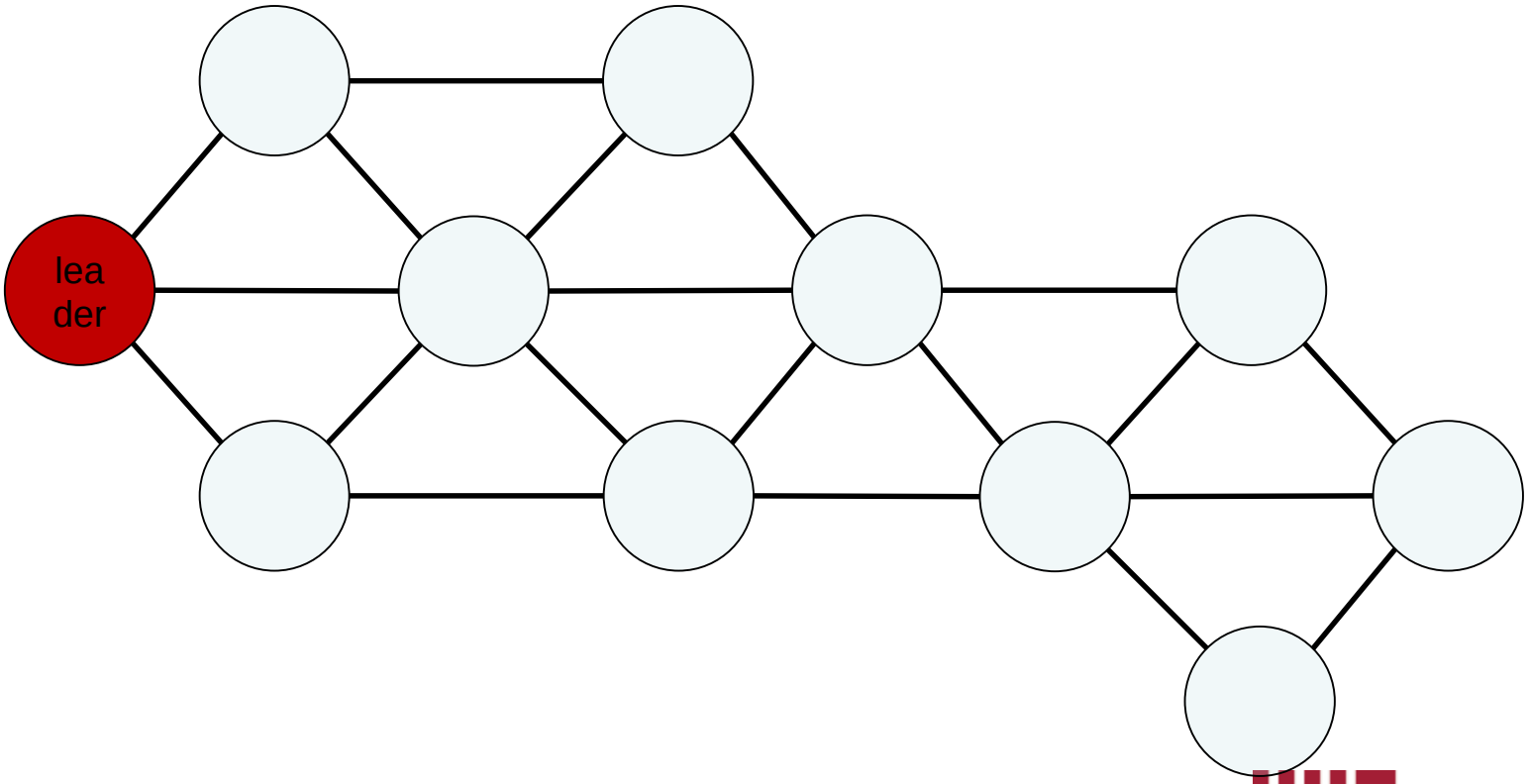
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



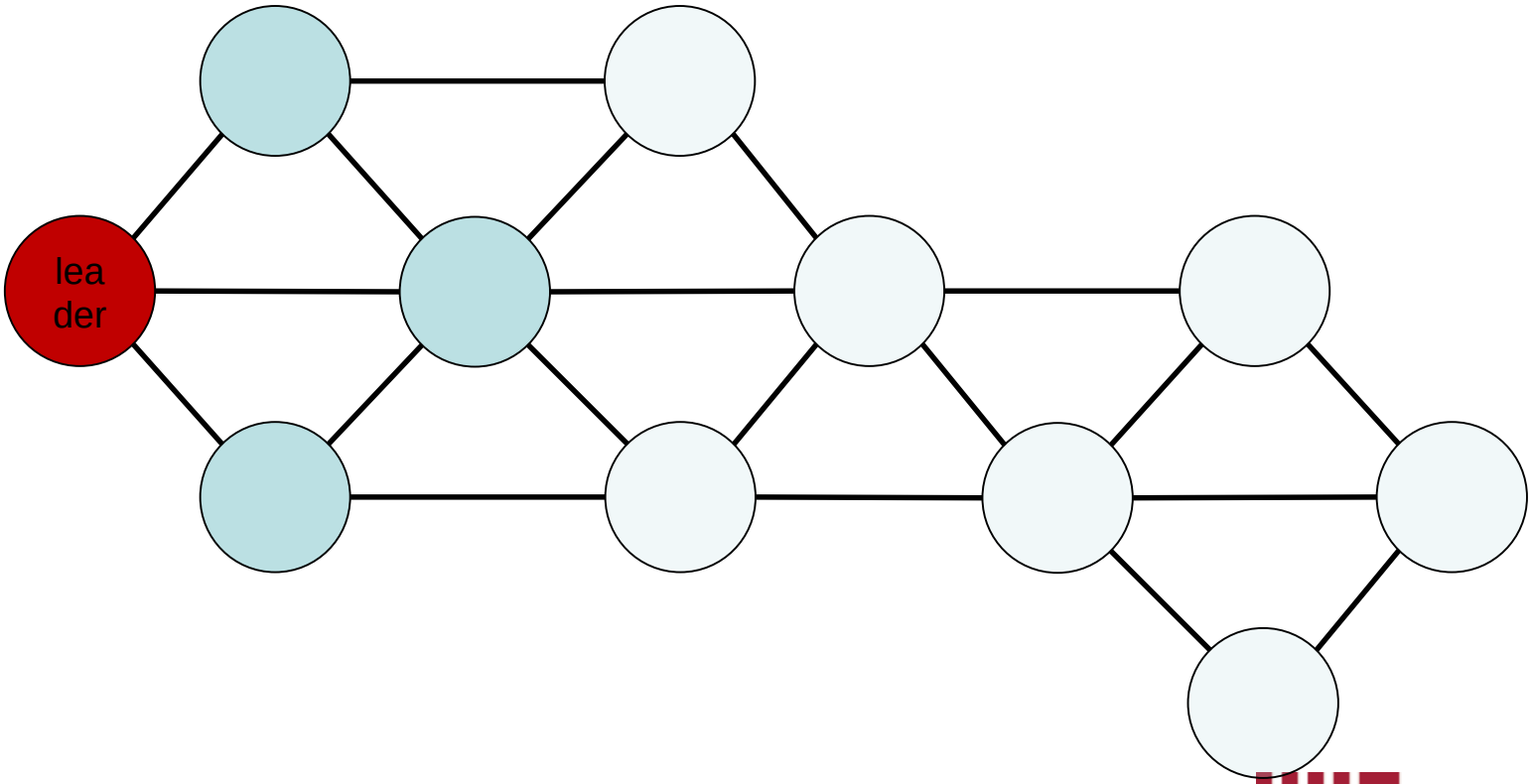
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



Sele

Wait:

Transmissions:

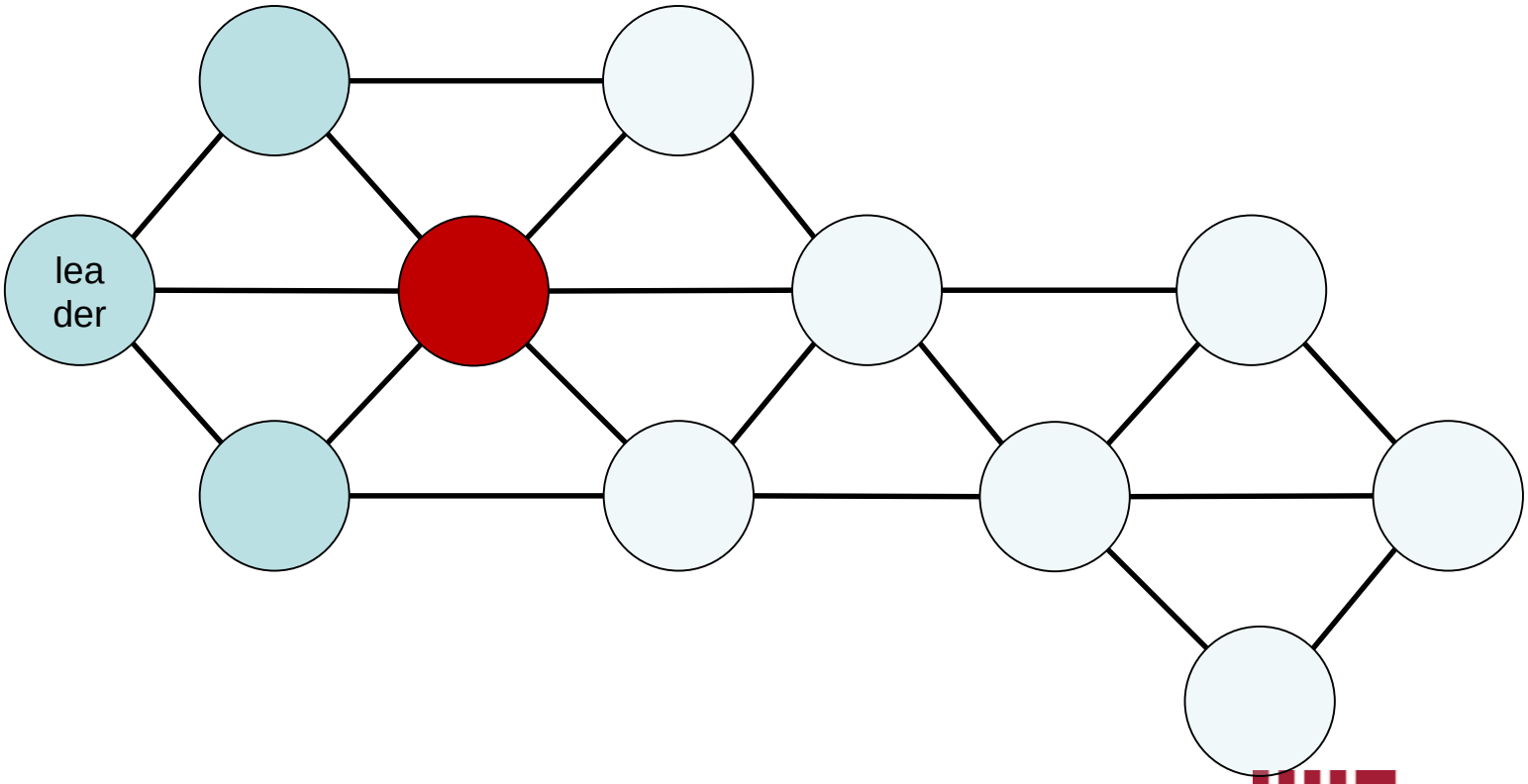
- Same message delivered to all neighbors

Solution:

- Wait until there is a neighbor in a unique state among the neighbors
- Detect this via ports
- Transmit “content of this port”

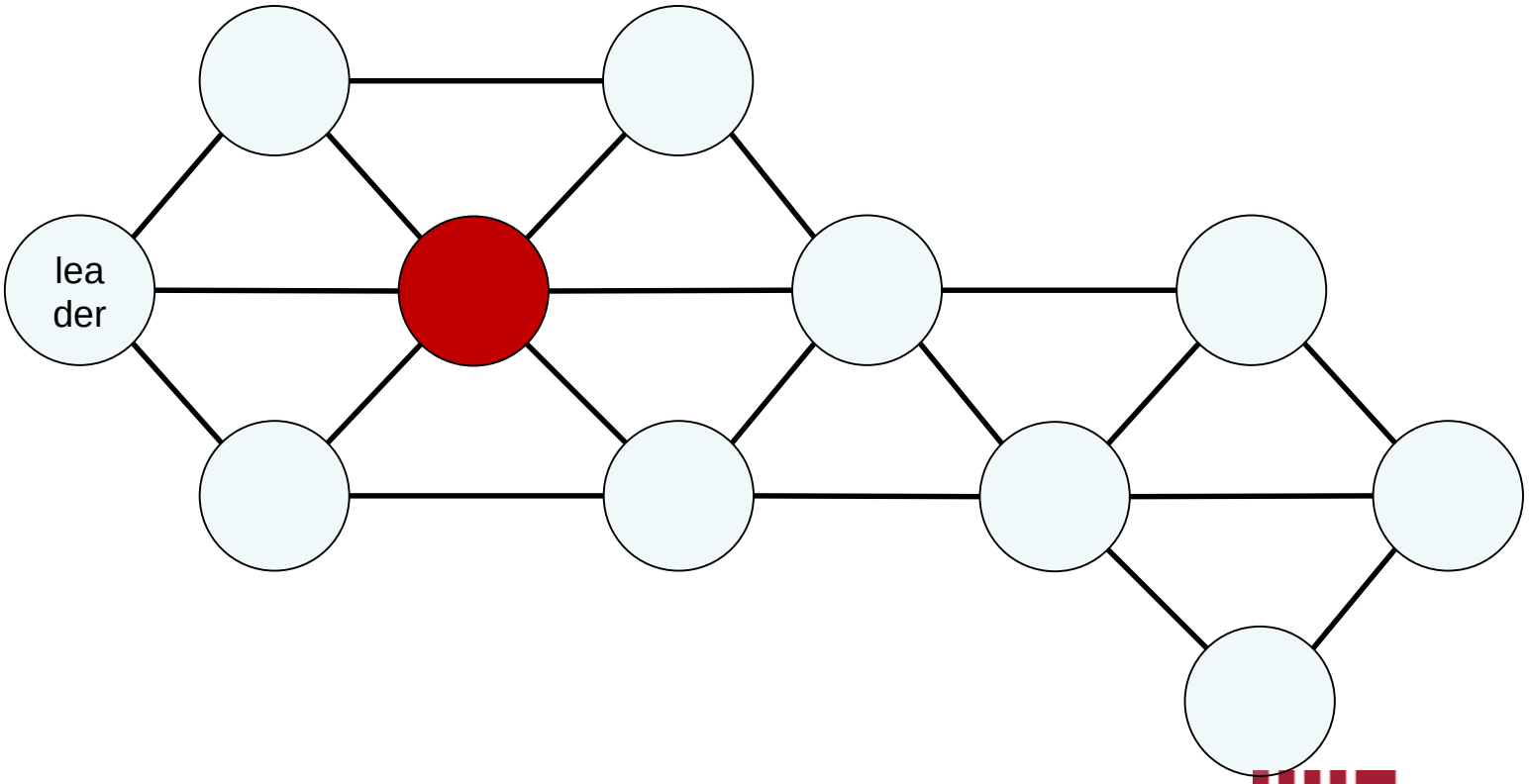
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



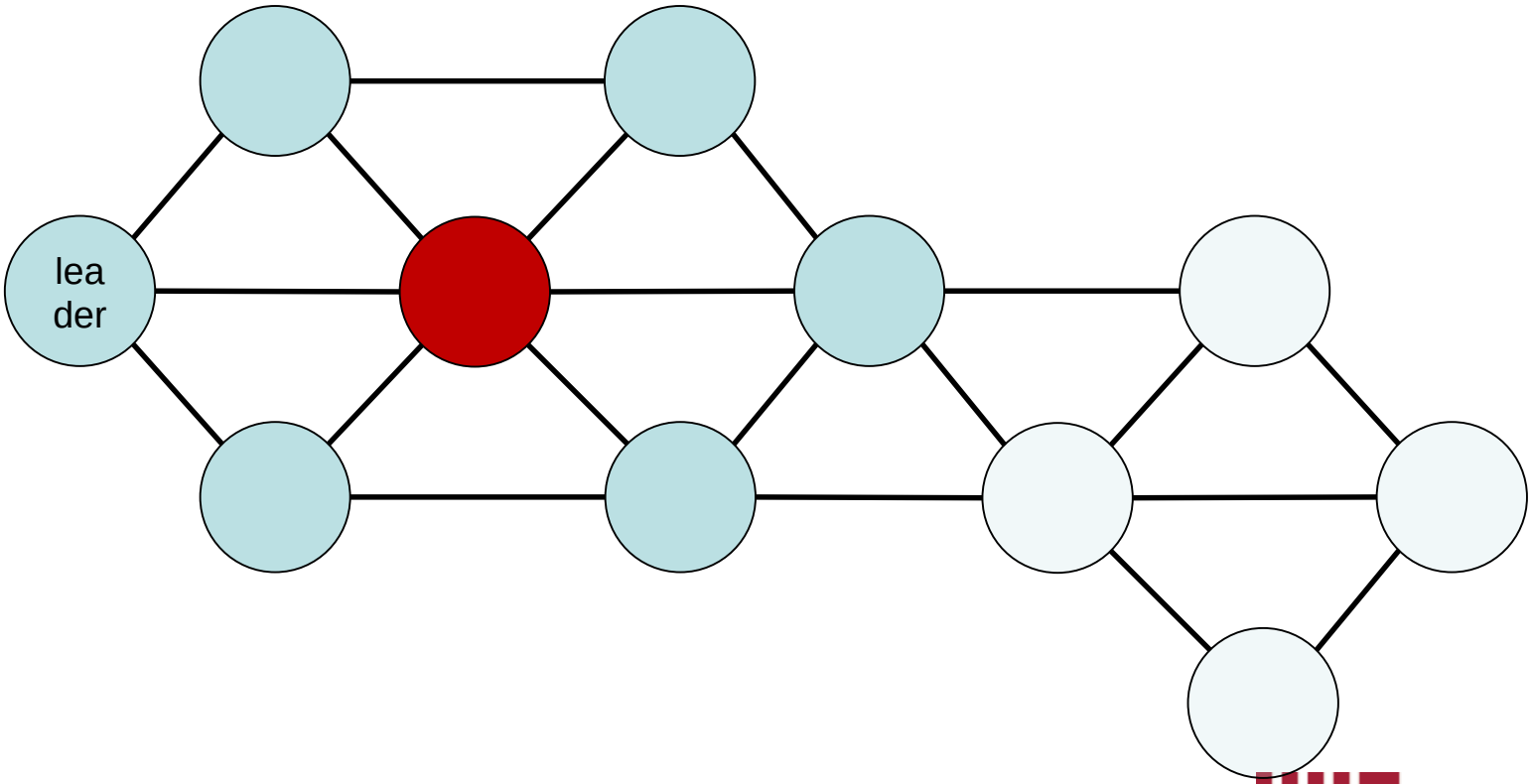
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



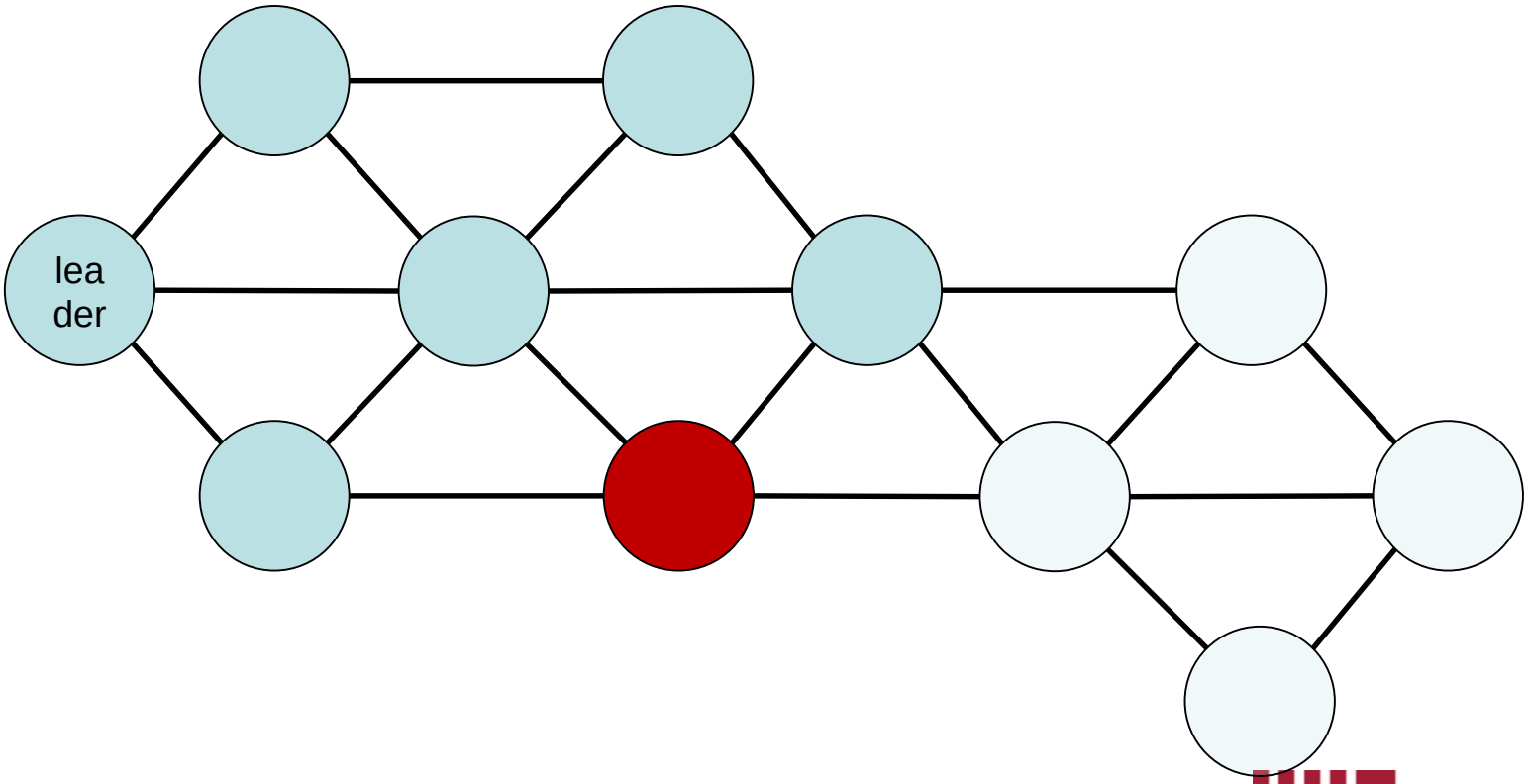
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



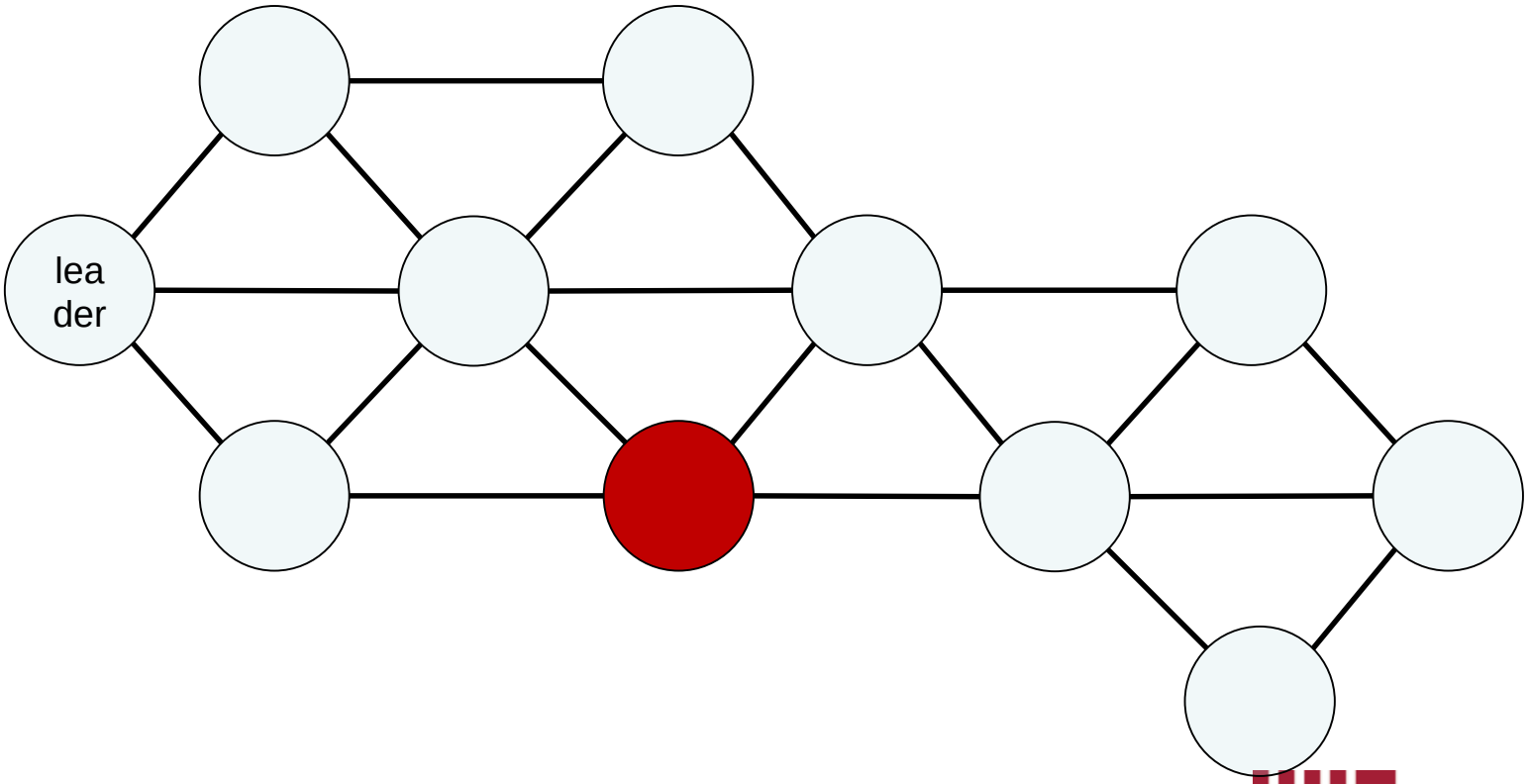
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



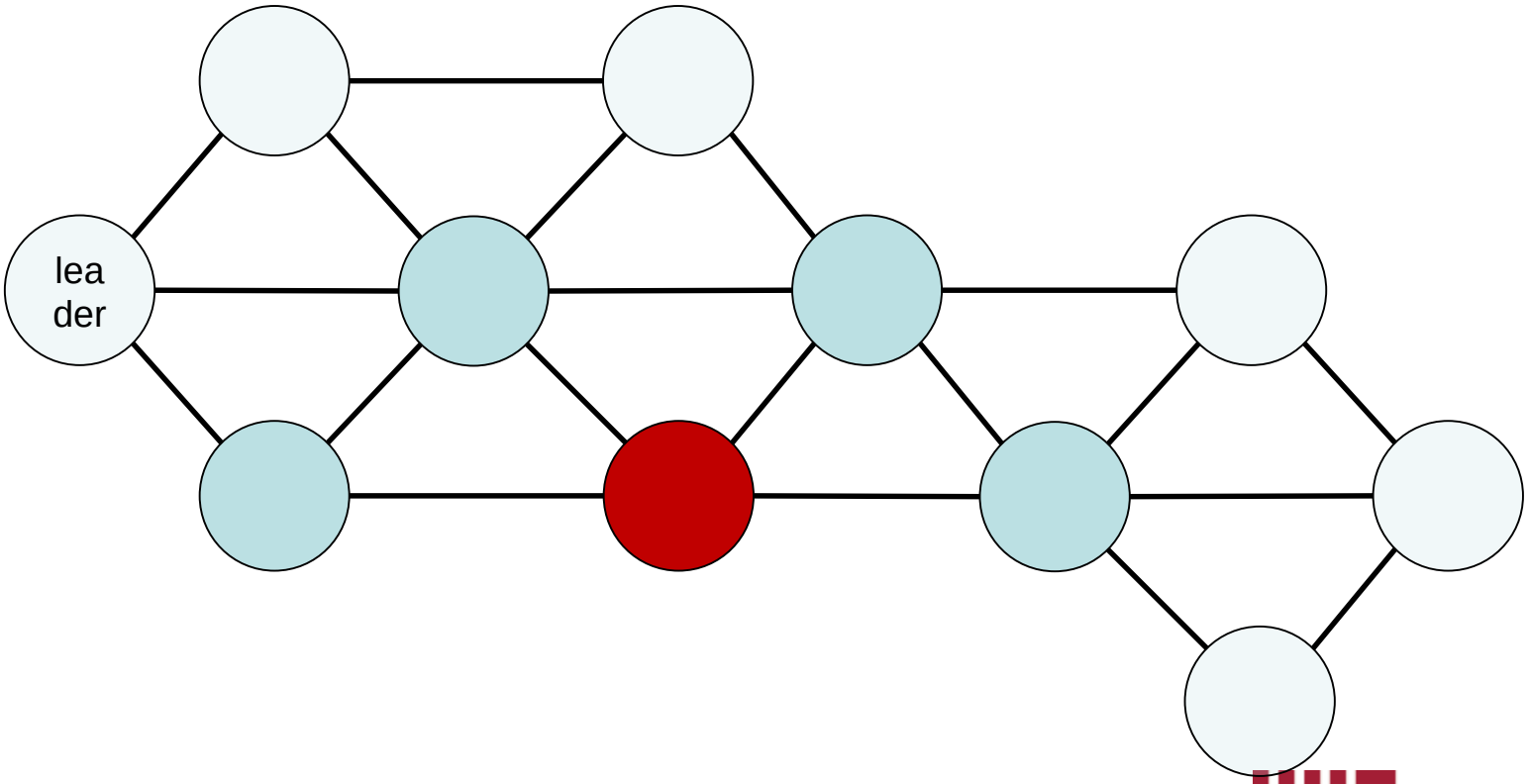
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



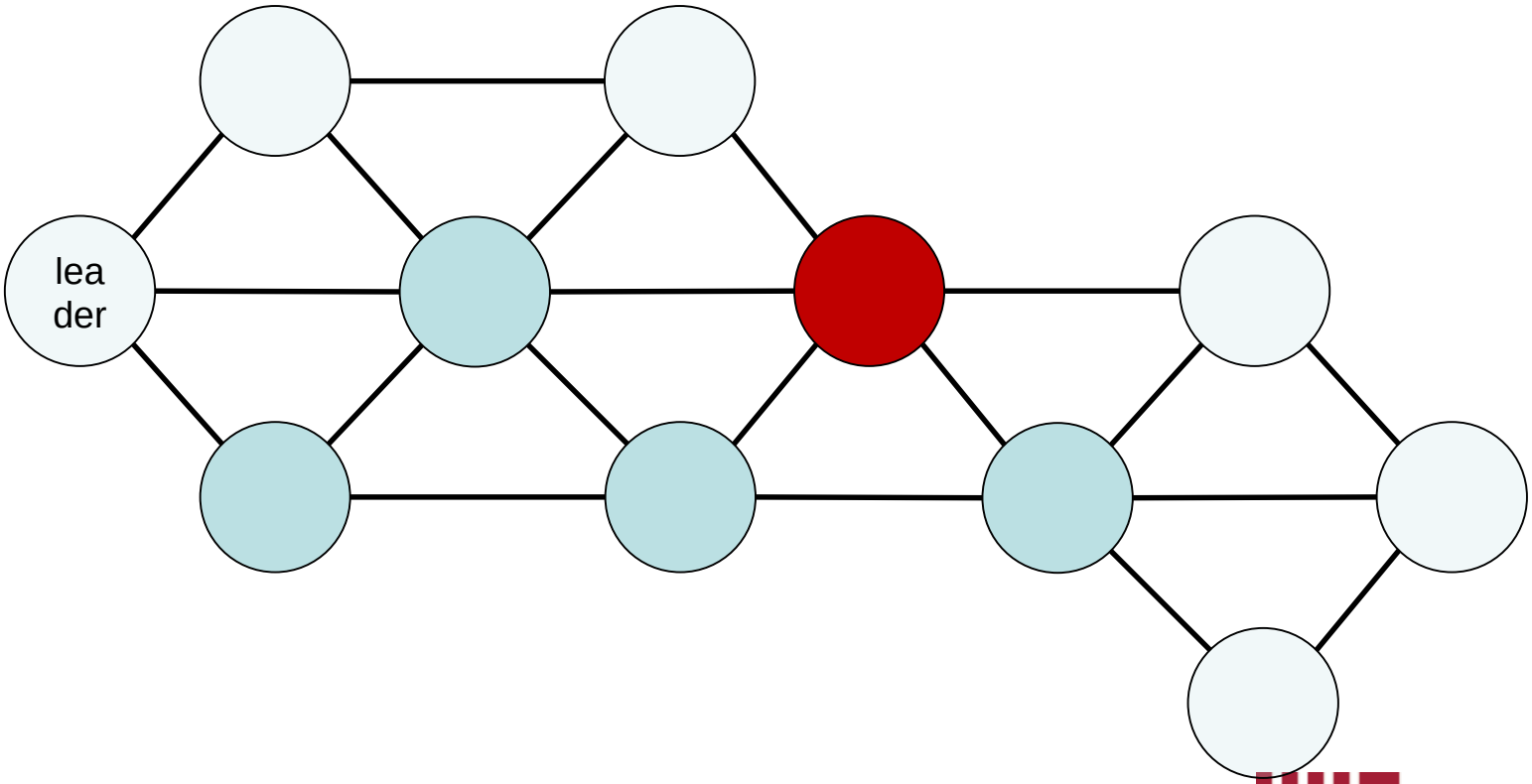
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



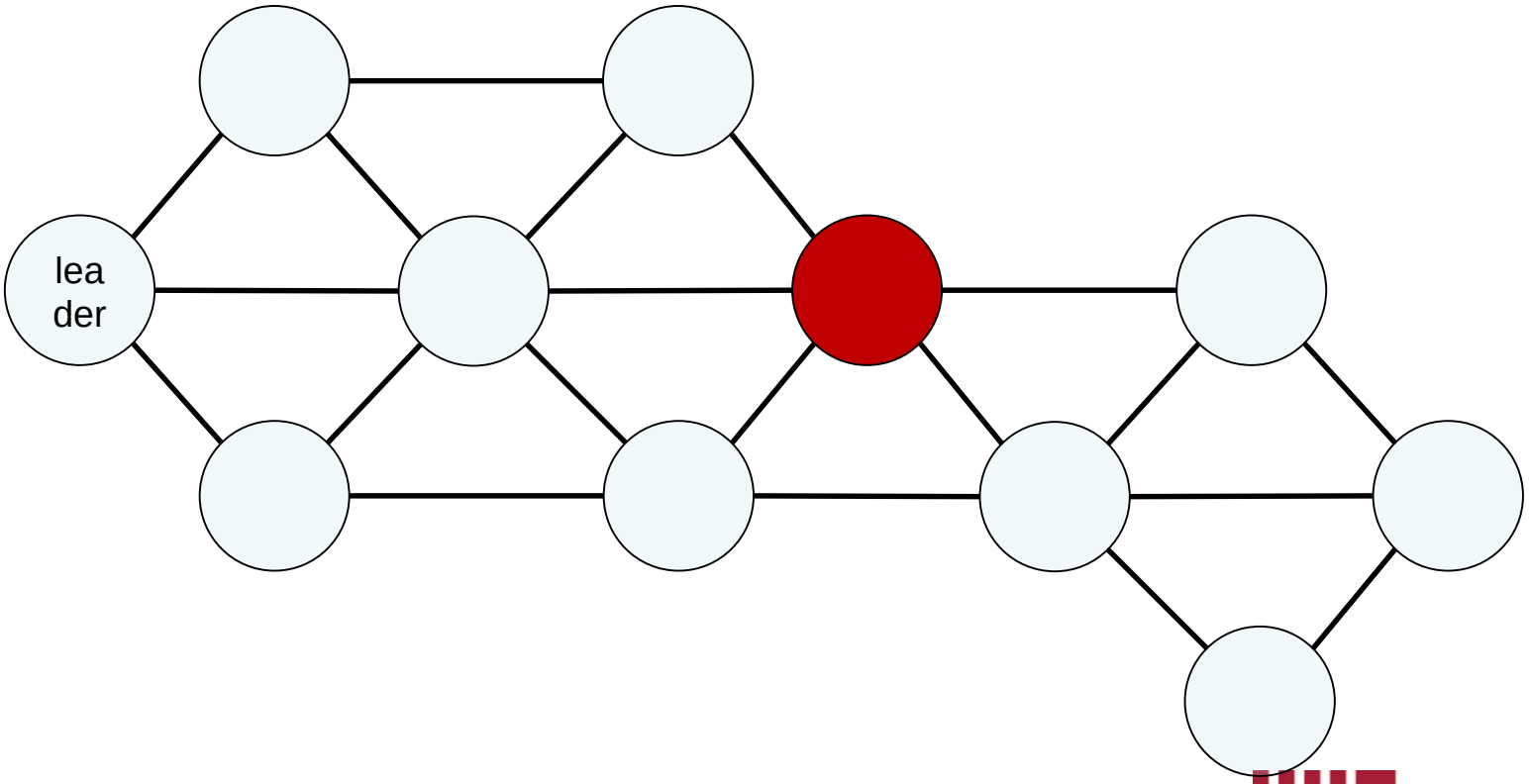
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



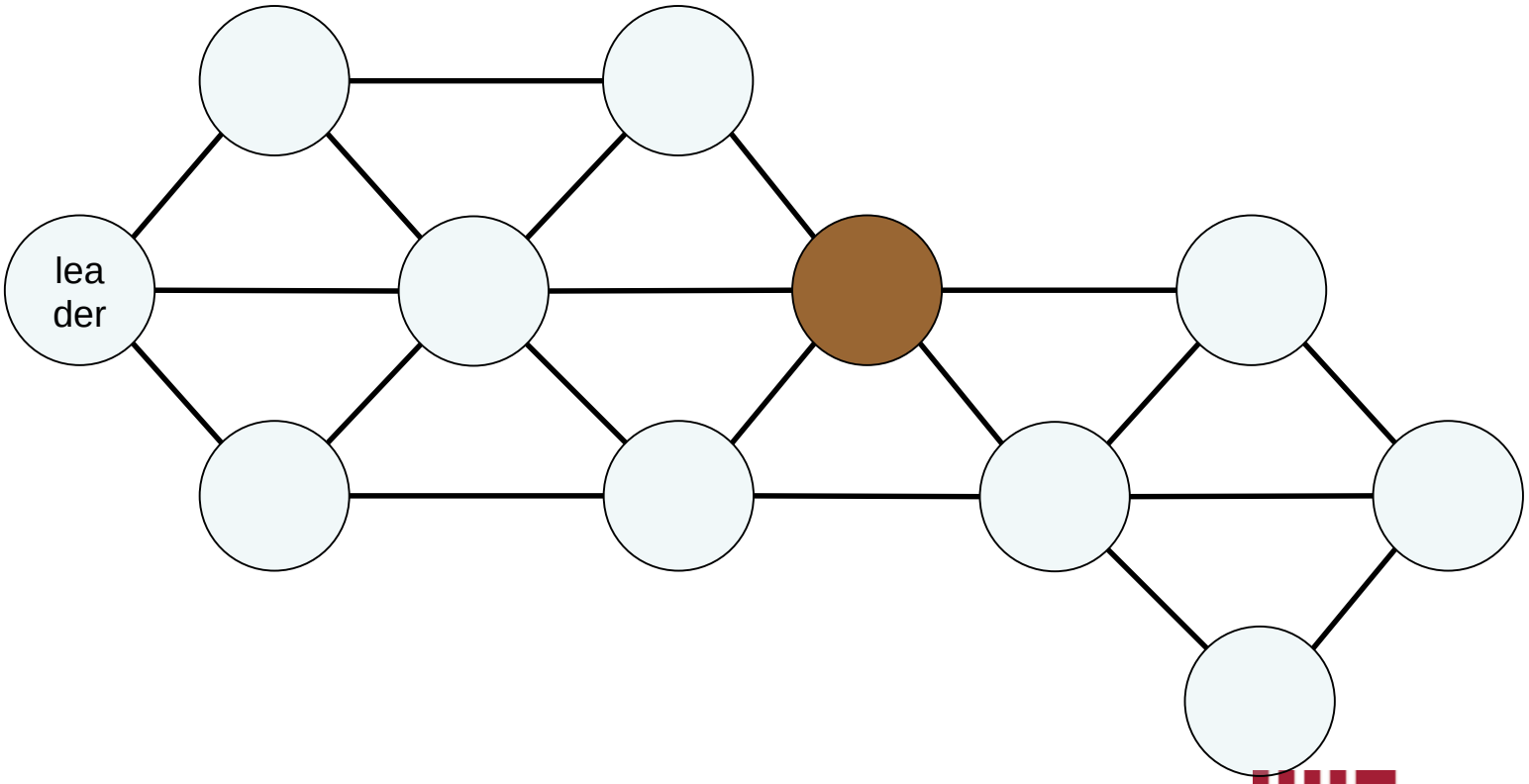
Select Unique Node at Random

- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



Select Unique Node at Random

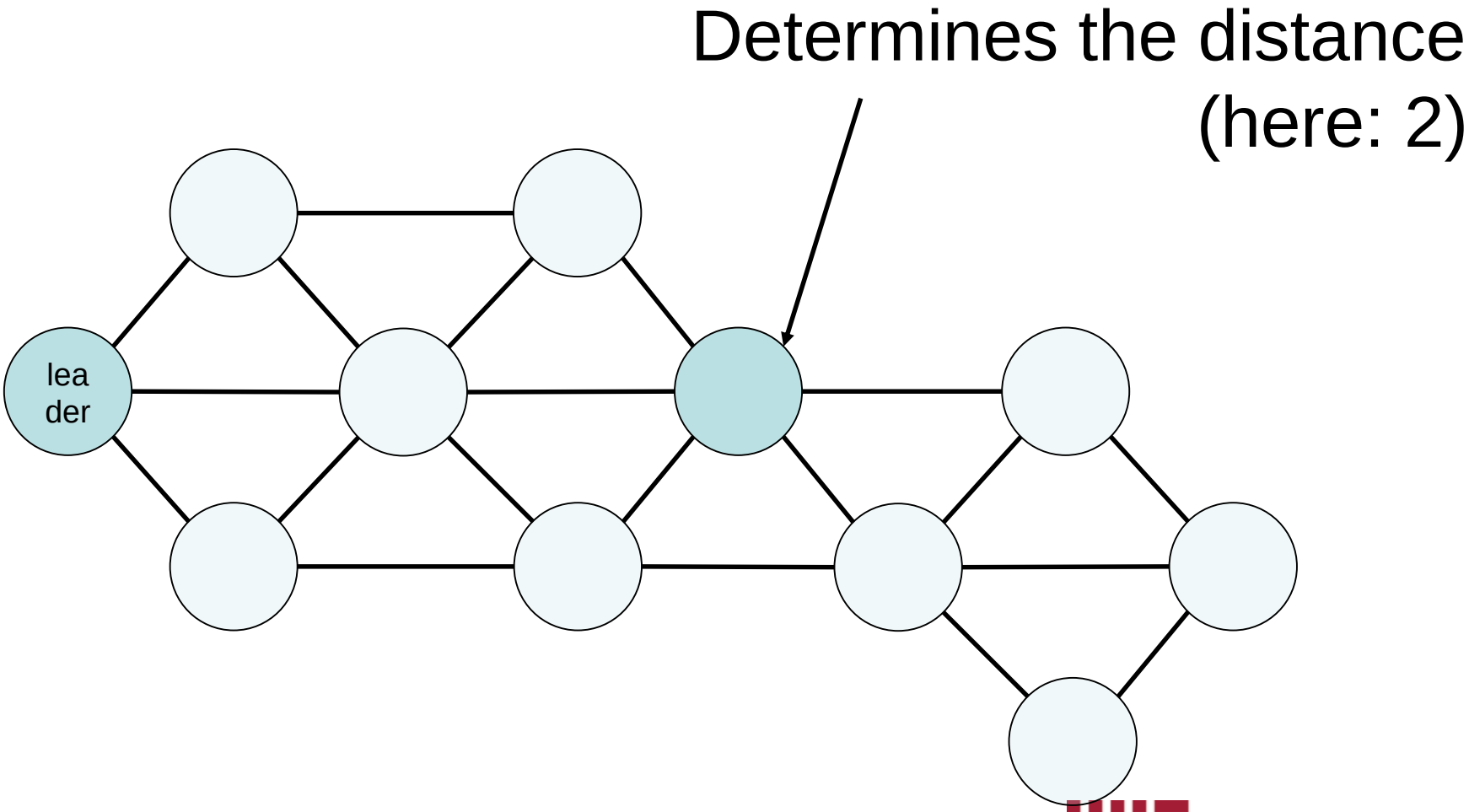
- REPEAT: 1. select random neighbor
  2. Pr[choose yourself]=1/2



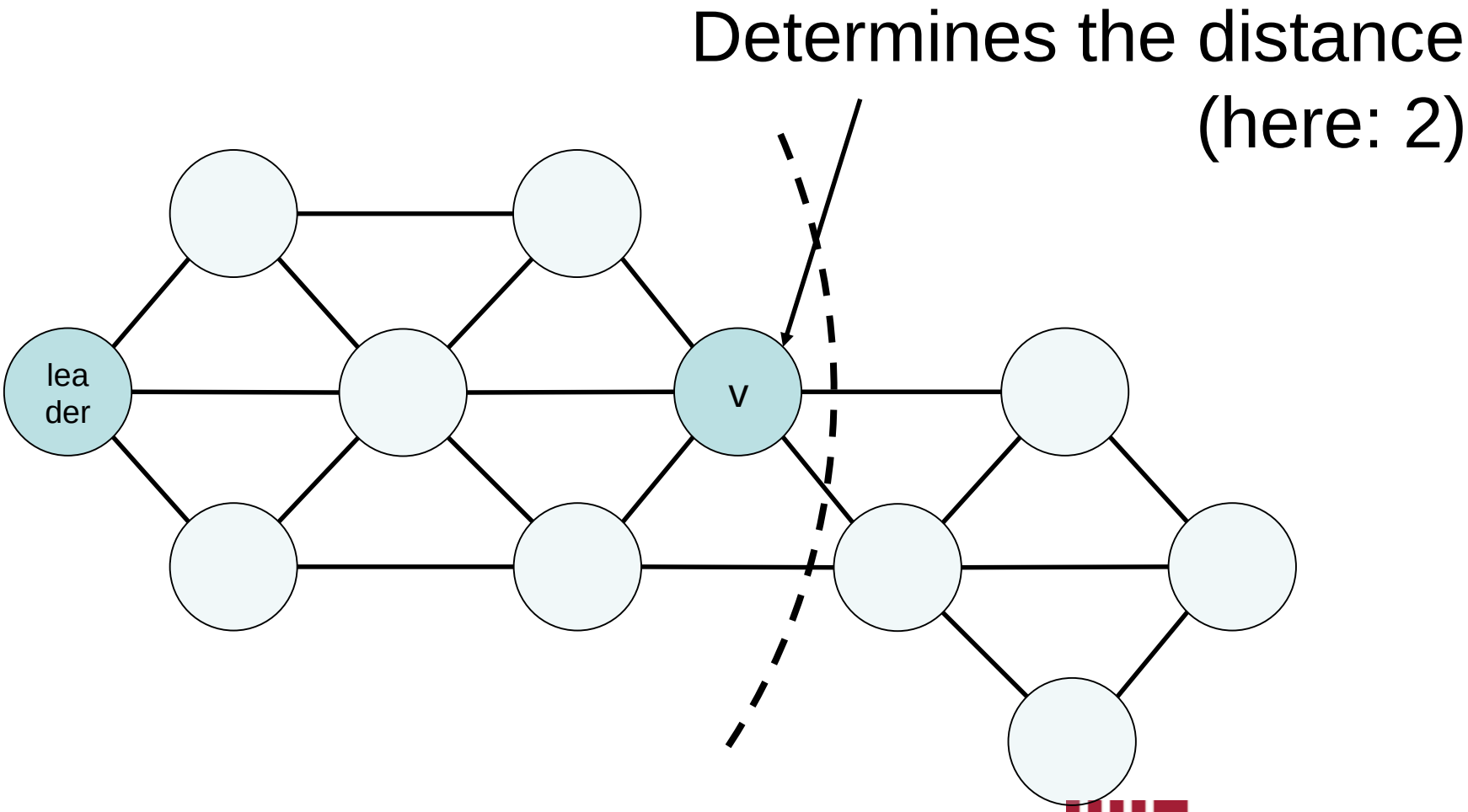
How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- Select unique node at random
- Check if all nodes up to a certain distance have a certain property

Check if all nodes up to a certain distance have a certain property



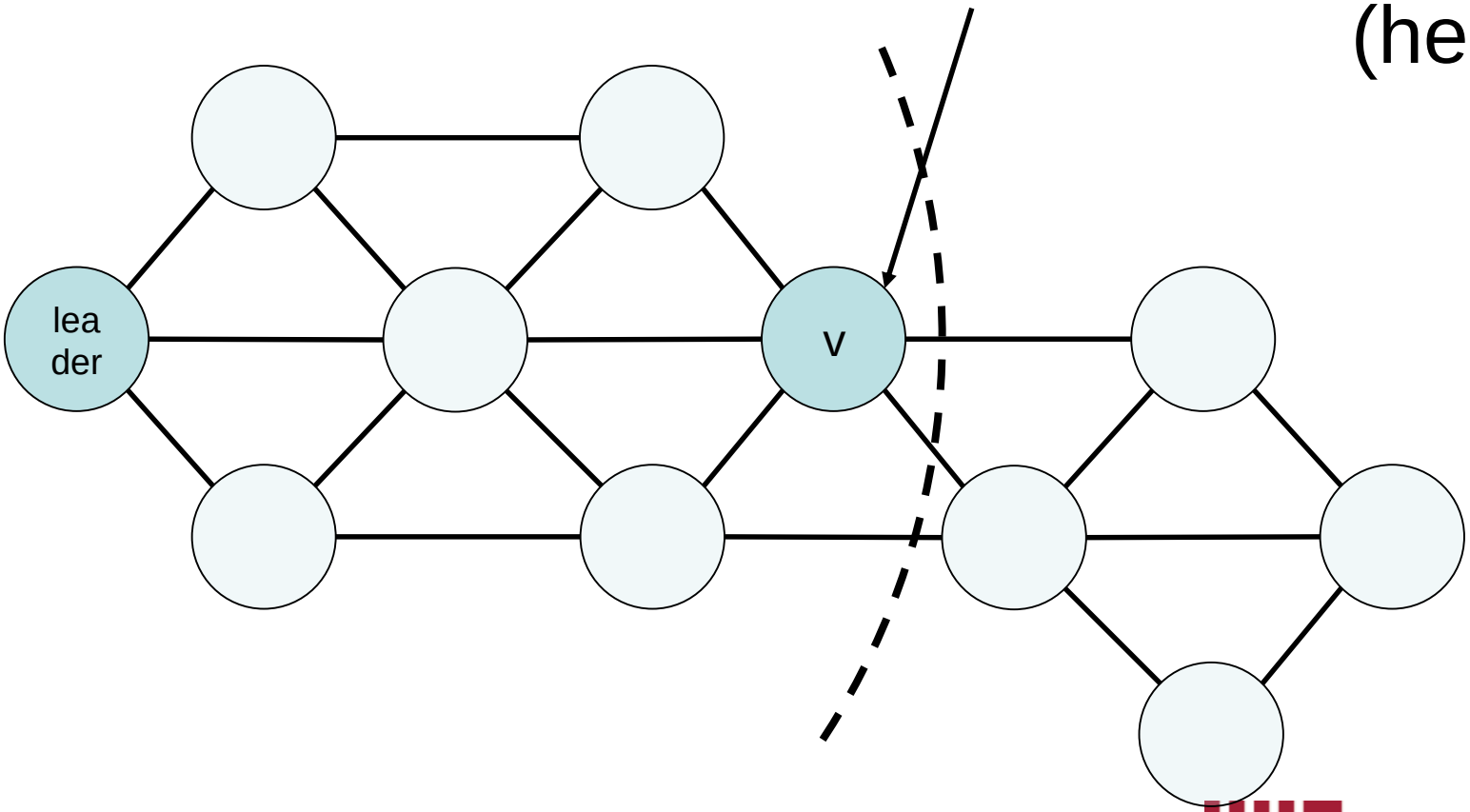
Check if all nodes up to a certain distance have a certain property



Check if all nodes up to a certain distance have a certain property

Use ping to check properties

Determines the distance (here: 2)



How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- Select unique node at random
- Check if all nodes up to a certain distance have a certain property

How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- Select unique node at random
- Check if all nodes up to a certain distance have a certain property (works for subsets as well)

How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- Select unique node at random
- Check if all nodes up to a certain distance have a certain property (works for subsets as well)
- Iterate through all nodes (**for** $v \in V$ **do** TASK)

Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} := \mathrm{u};$

u performs TASK;

Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

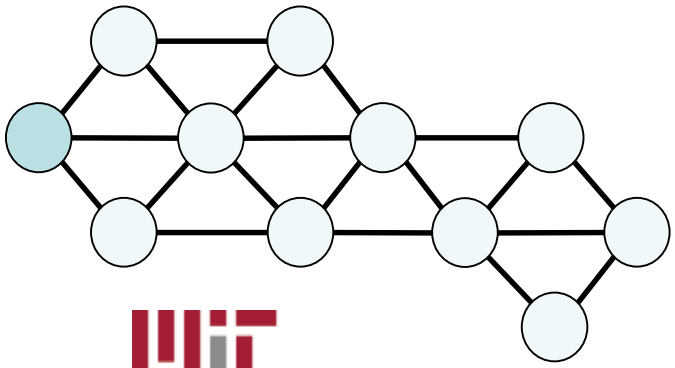
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} := \mathrm{u};$

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

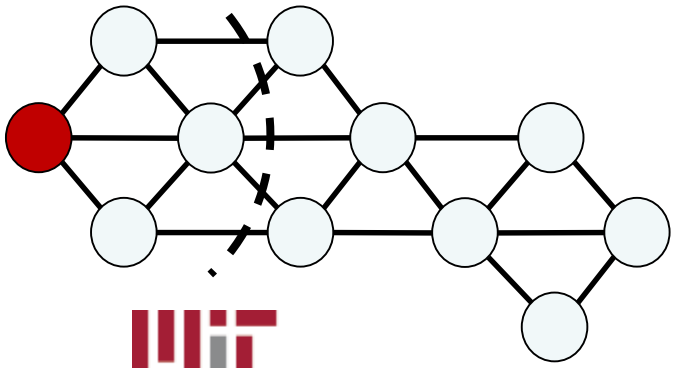
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

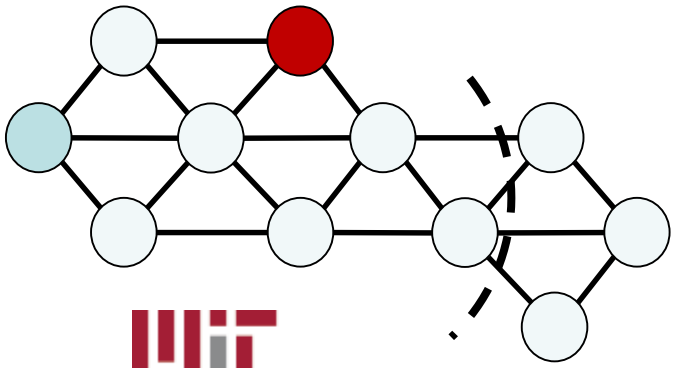
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

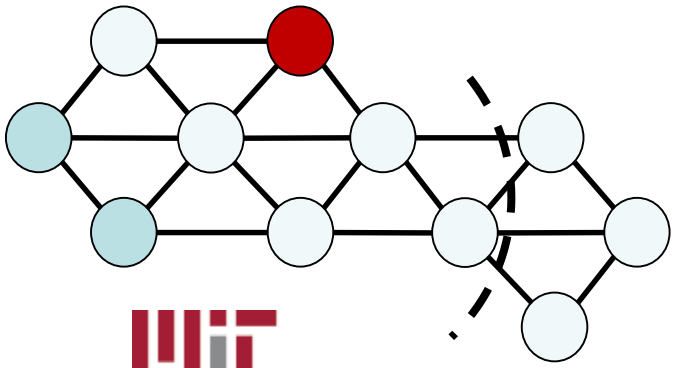
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

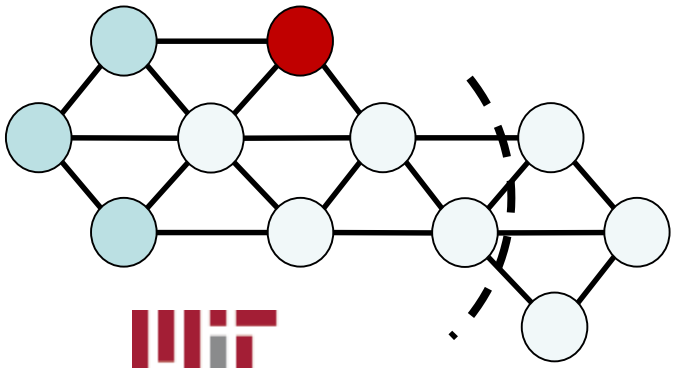
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

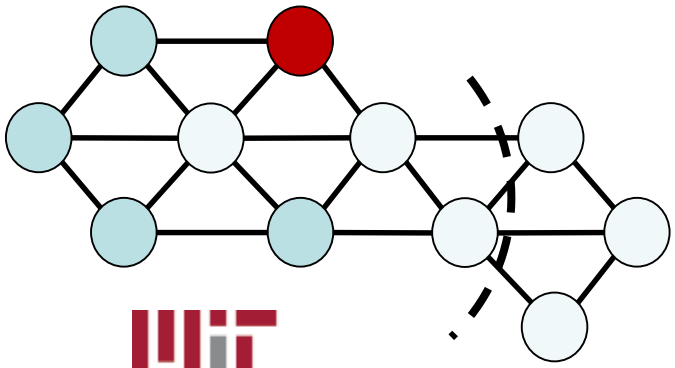
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

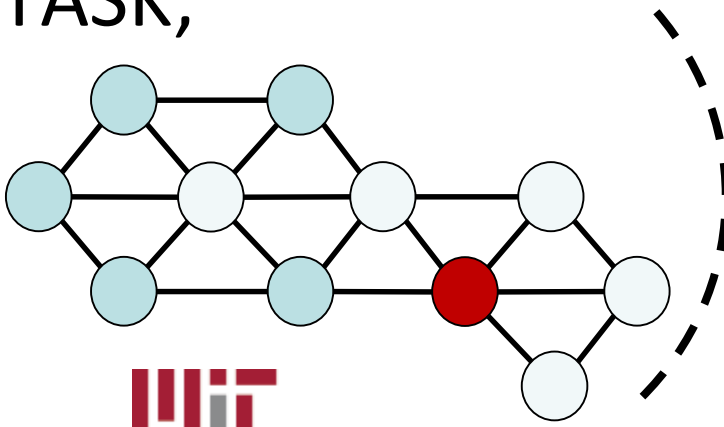
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

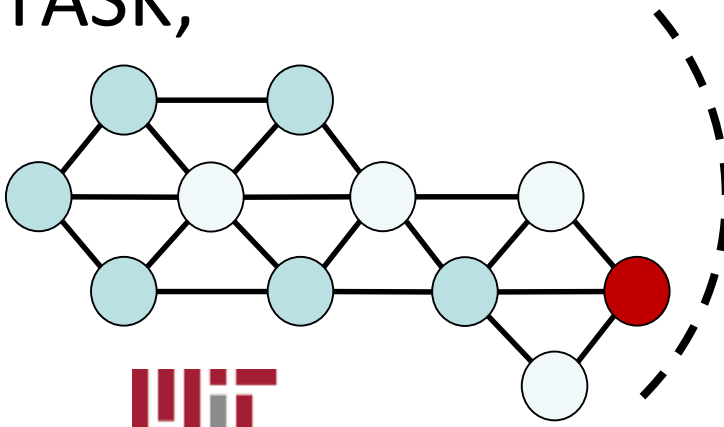
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

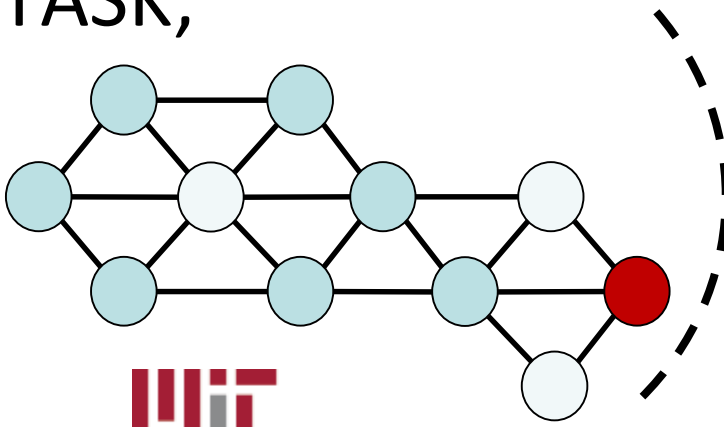
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

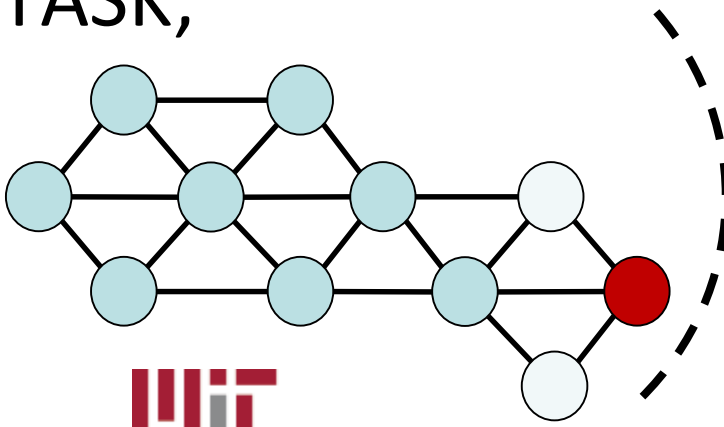
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

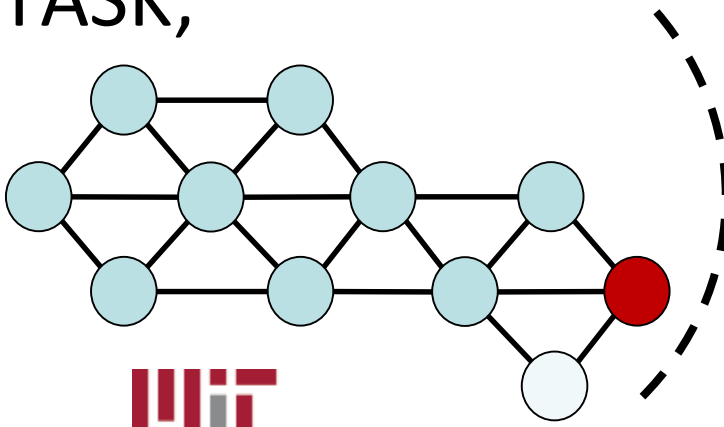
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

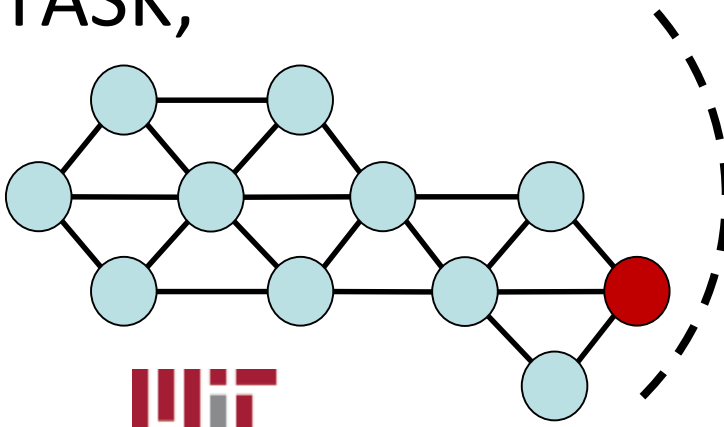
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked **do**

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



Iterate through all nodes (**for** $v \in V$ **do**)

$v_{max} := leader;$

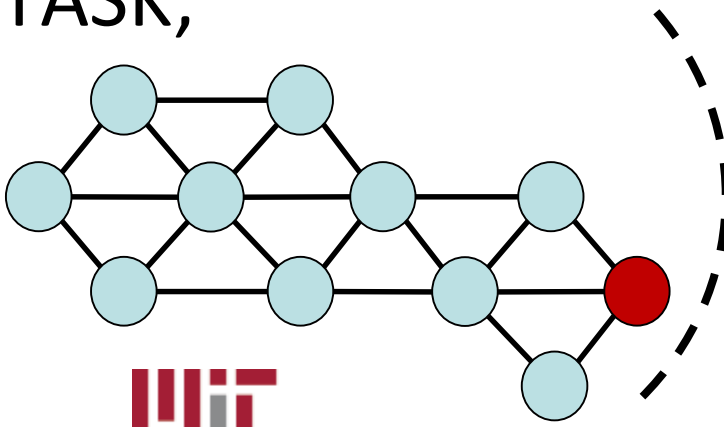
while not all nodes at distance $\leq$ dist(leader,v_{max})+1 are marked do

select random node u; // u marks itself

if dist(leader,u) > dist(leader, v_{max}) **then**

$v_{max} :=$ u;

u performs TASK;



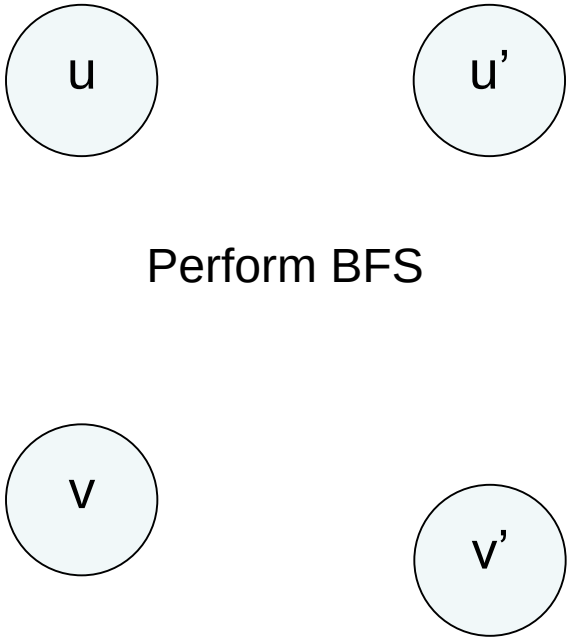
How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- Select unique node at random
- Check if all nodes up to a certain distance have a certain property (works for subsets as well)
- Iterate through all nodes (**for** $\mathrm{u} \in V$ **do** TASK)

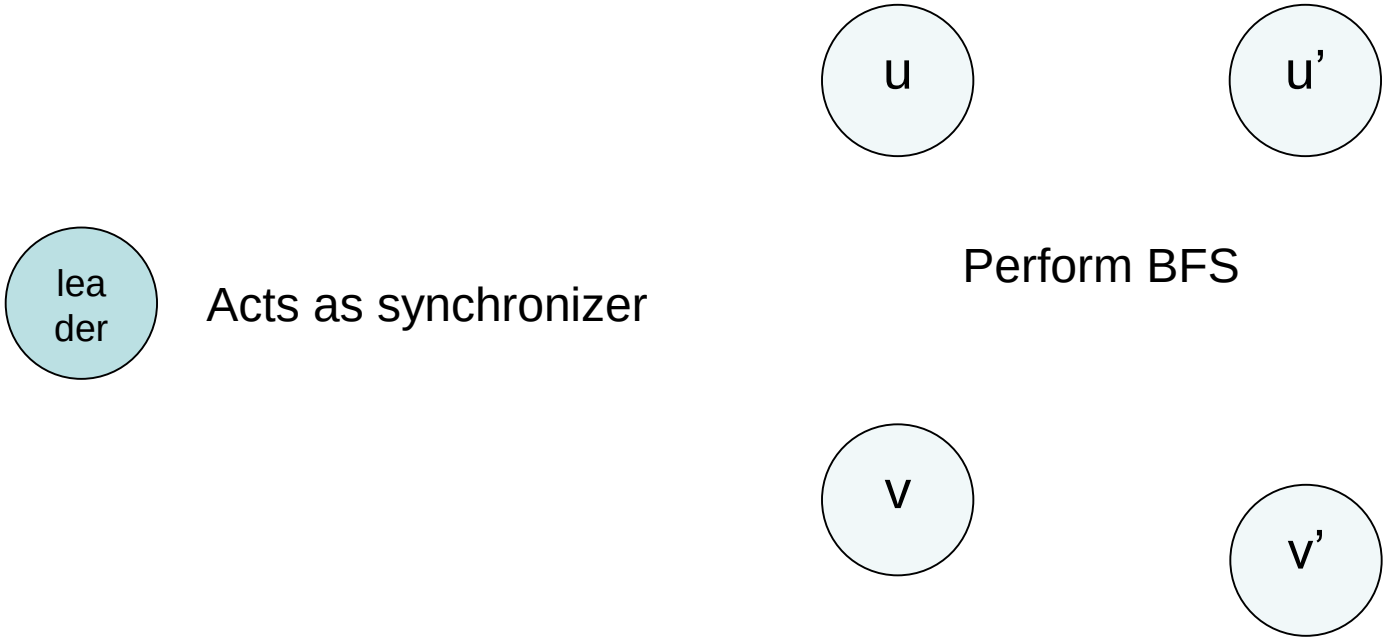
How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- Select unique node at random
- Check if all nodes up to a certain distance have a certain property (works for subsets as well)
- Iterate through all nodes (**for** $u \in V$ **do** TASK)
- Decide dist(u,v) > dist(u',v')

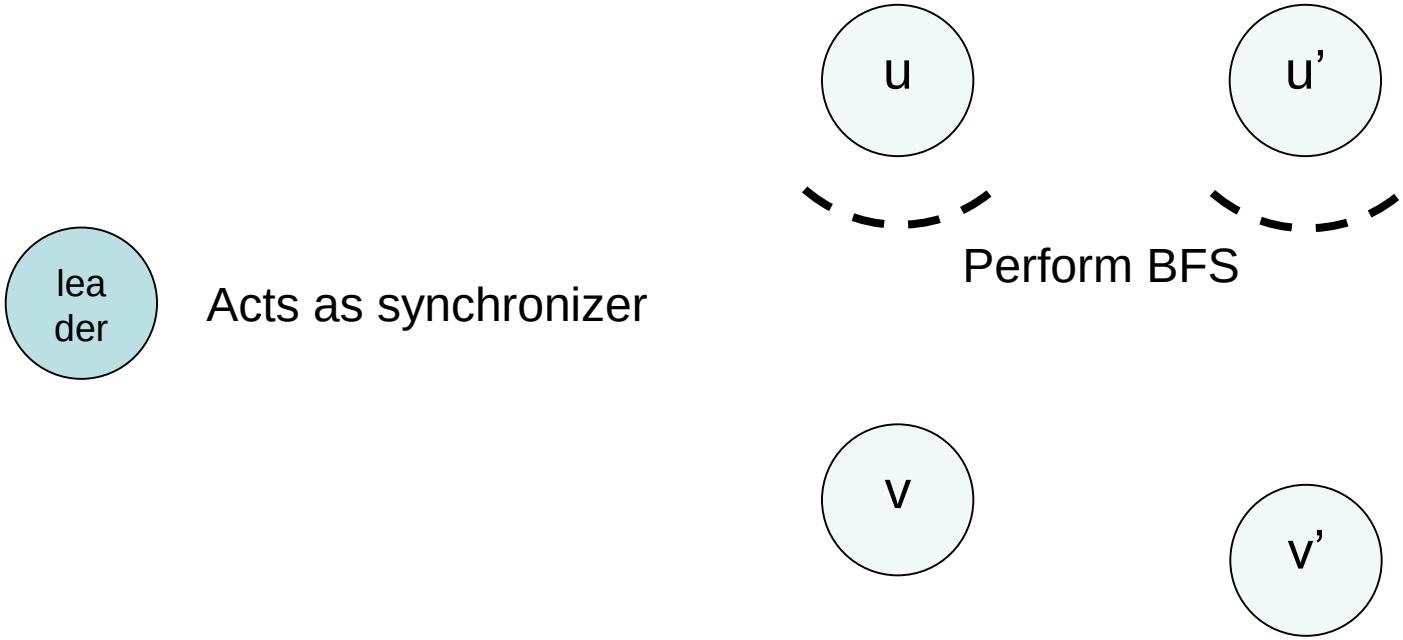
Decide dist(u,v) > dist(u’,v’)



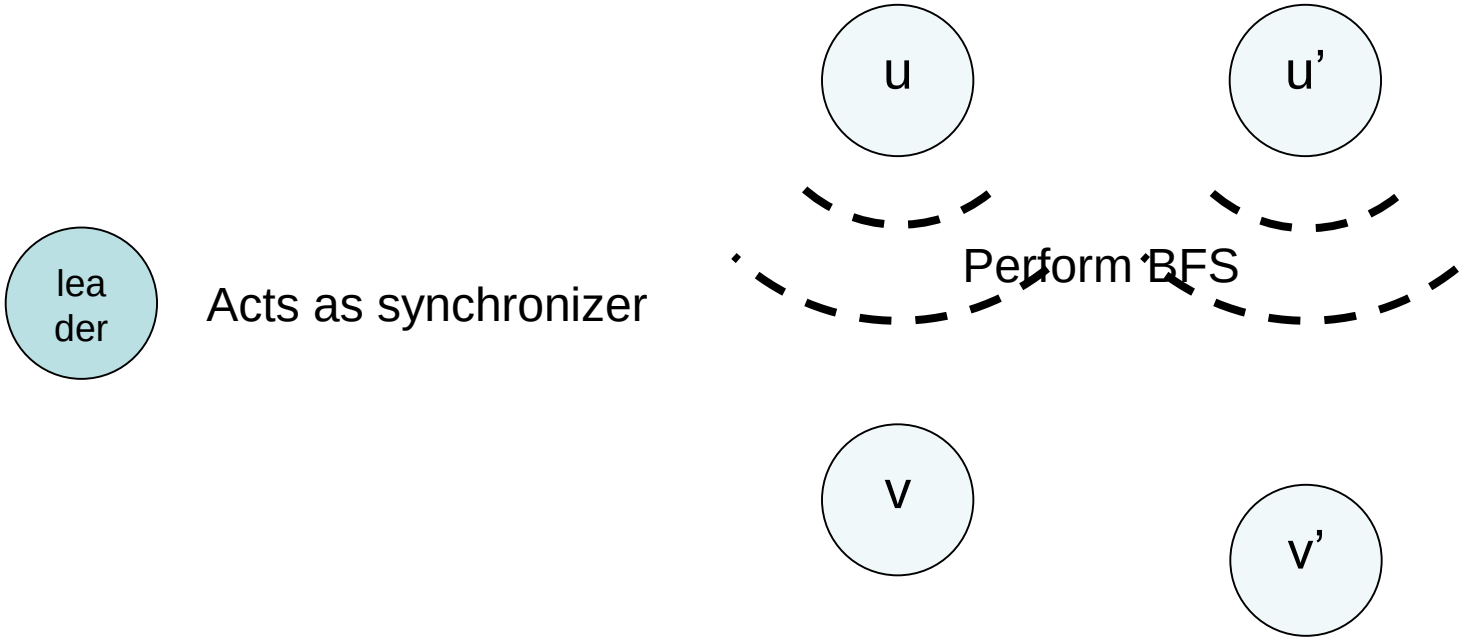
Decide dist(u,v) > dist(u’,v’)



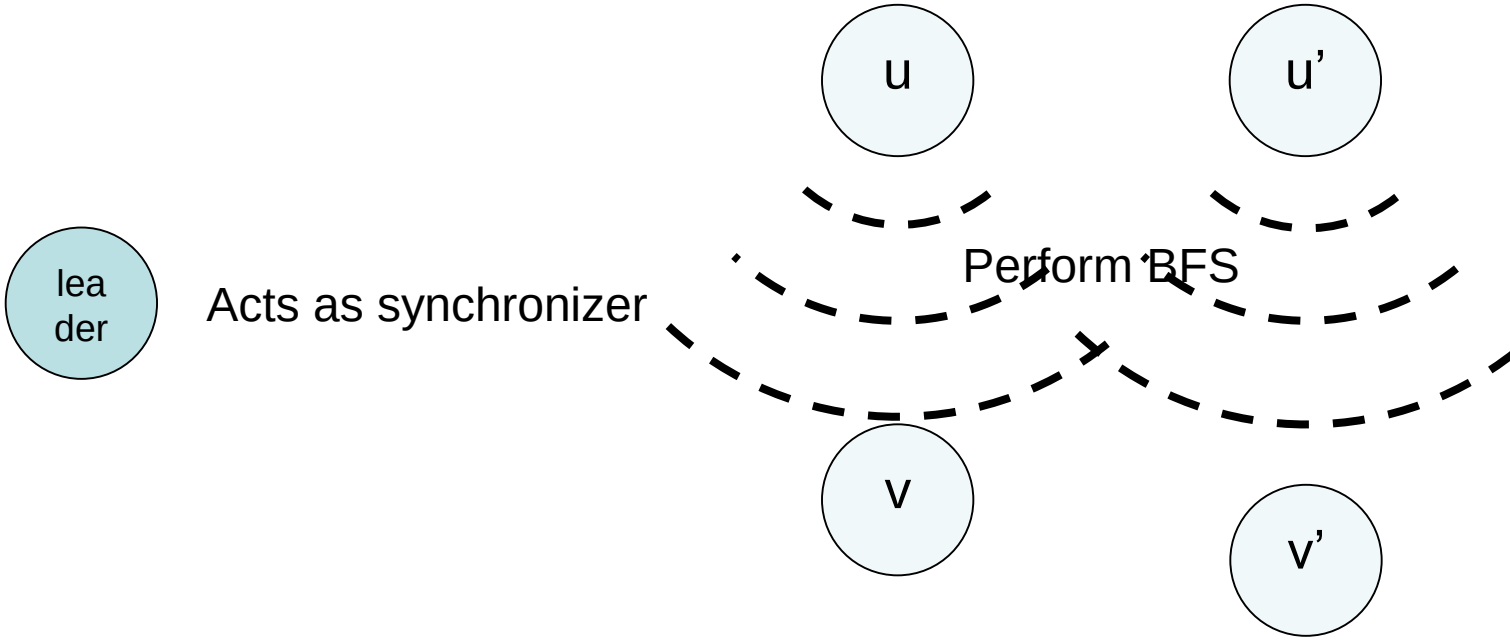
Decide dist(u,v) > dist(u’,v’)



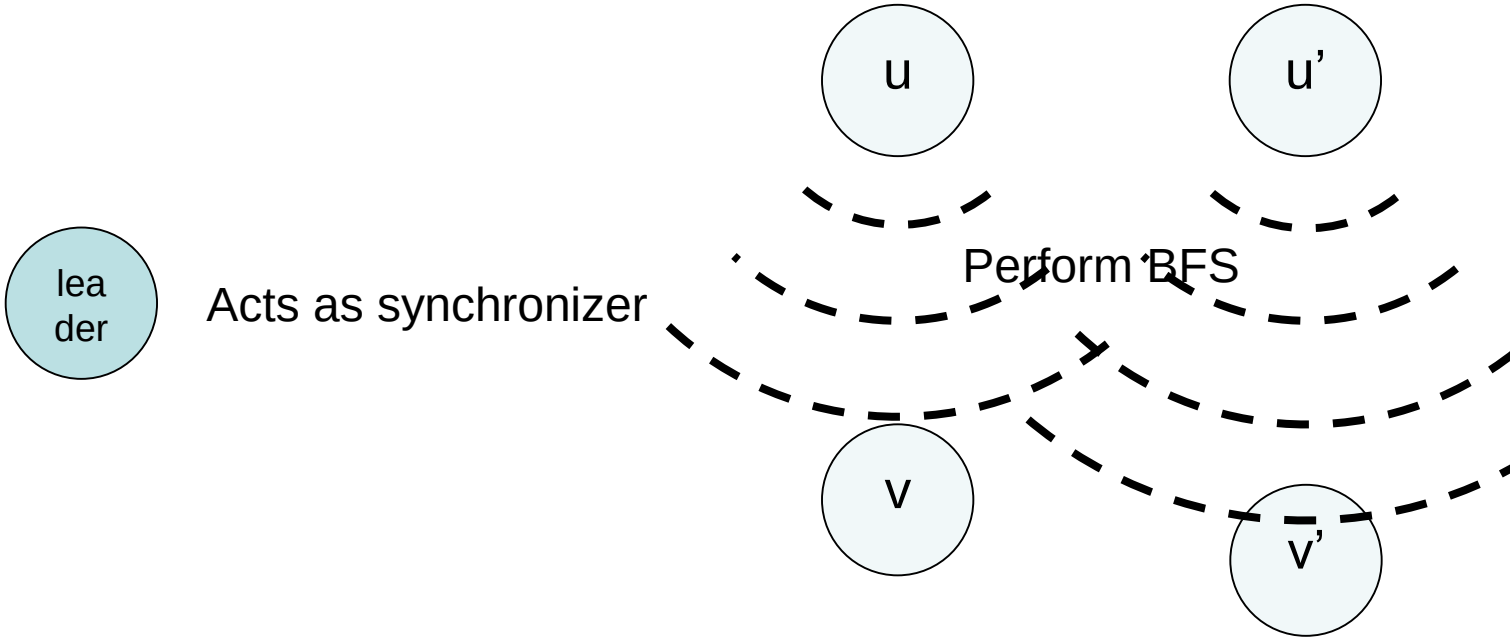
Decide dist(u,v) > dist(u’,v’)



Decide dist(u,v) > dist(u’,v’)



Decide dist(u,v) > dist(u’,v’)



How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- Select unique node at random
- Check if all nodes up to a certain distance have a certain property (works for subsets as well)
- Iterate through all nodes (**for** $\mathrm{u} \in V$ **do** TASK)
- Decide dist(u,v) > dist(u',v')

How a Leader can make a Difference!

- Symmetry is broken
- Can coordinate global computation
- Select unique node at random
- Check if all nodes up to a certain distance have a certain property (works for subsets as well)
- Iterate through all nodes (**for** $\mathrm{u} \in V$ **do** TASK)
- Decide dist(u,v) > dist(u',v')

- Constant combination of all these

Shortest Path

INPUT: nodes u,v (assume v is leader)

While $u \neq v$ **do**

u marks itself to be on the shortest path

for each neighbor w of u **do**

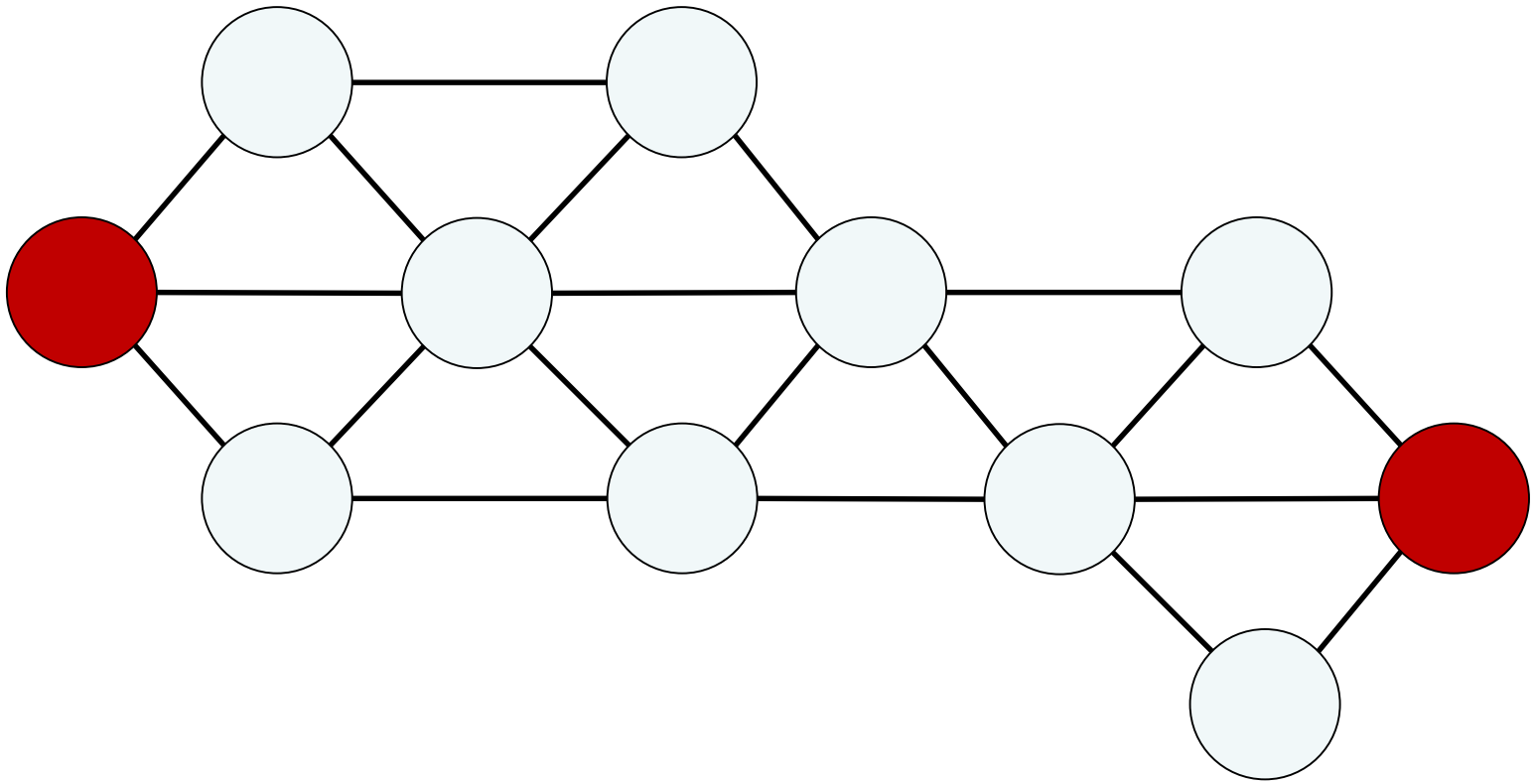
if $\mathrm{dist}(\mathrm{w}, \mathrm{v}) < \mathrm{dist}(u, \mathrm{v})$ **then**

$u := w;$

OUTPUT: marked nodes

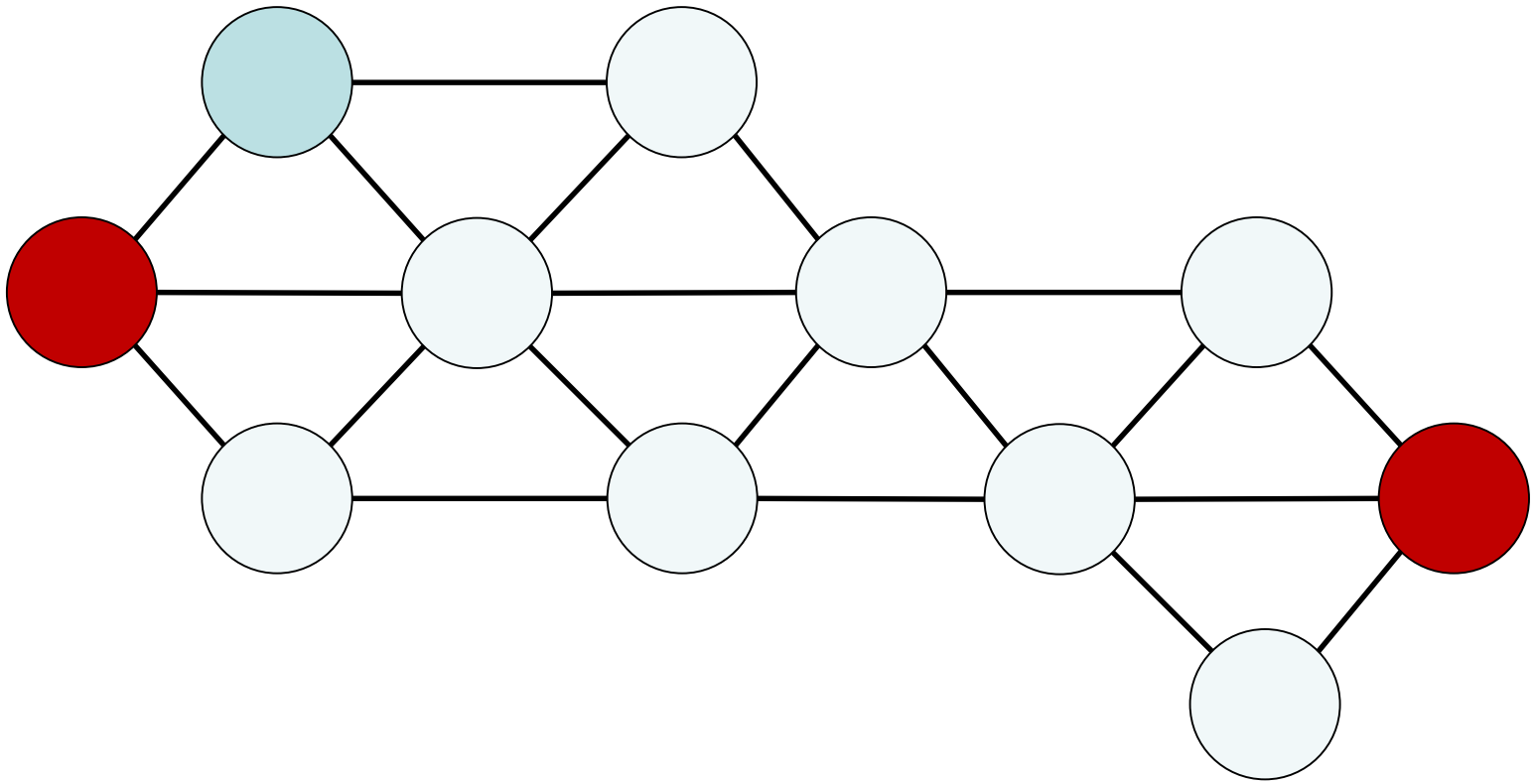
Shortest Path

- Example:



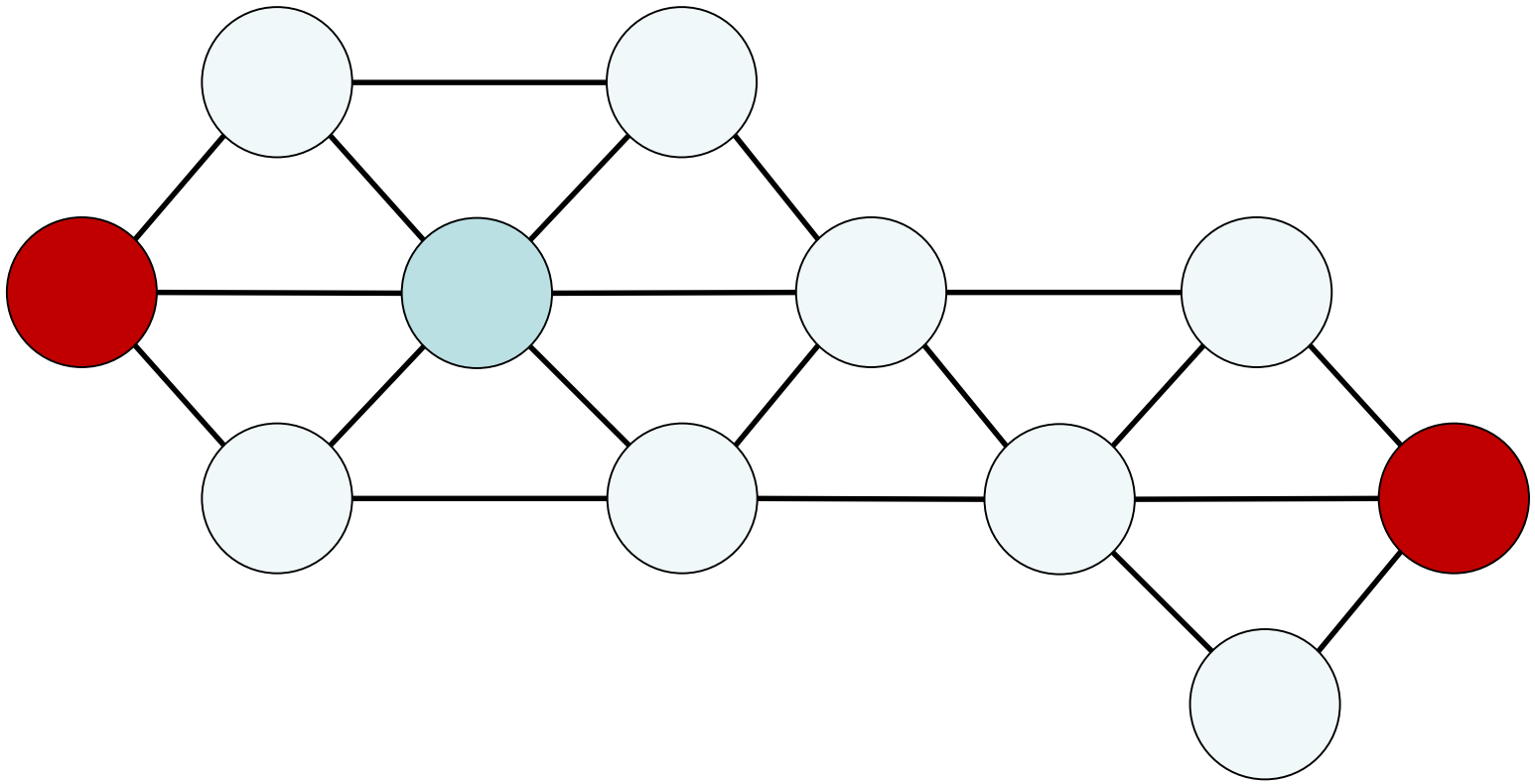
Shortest Path

- Example:



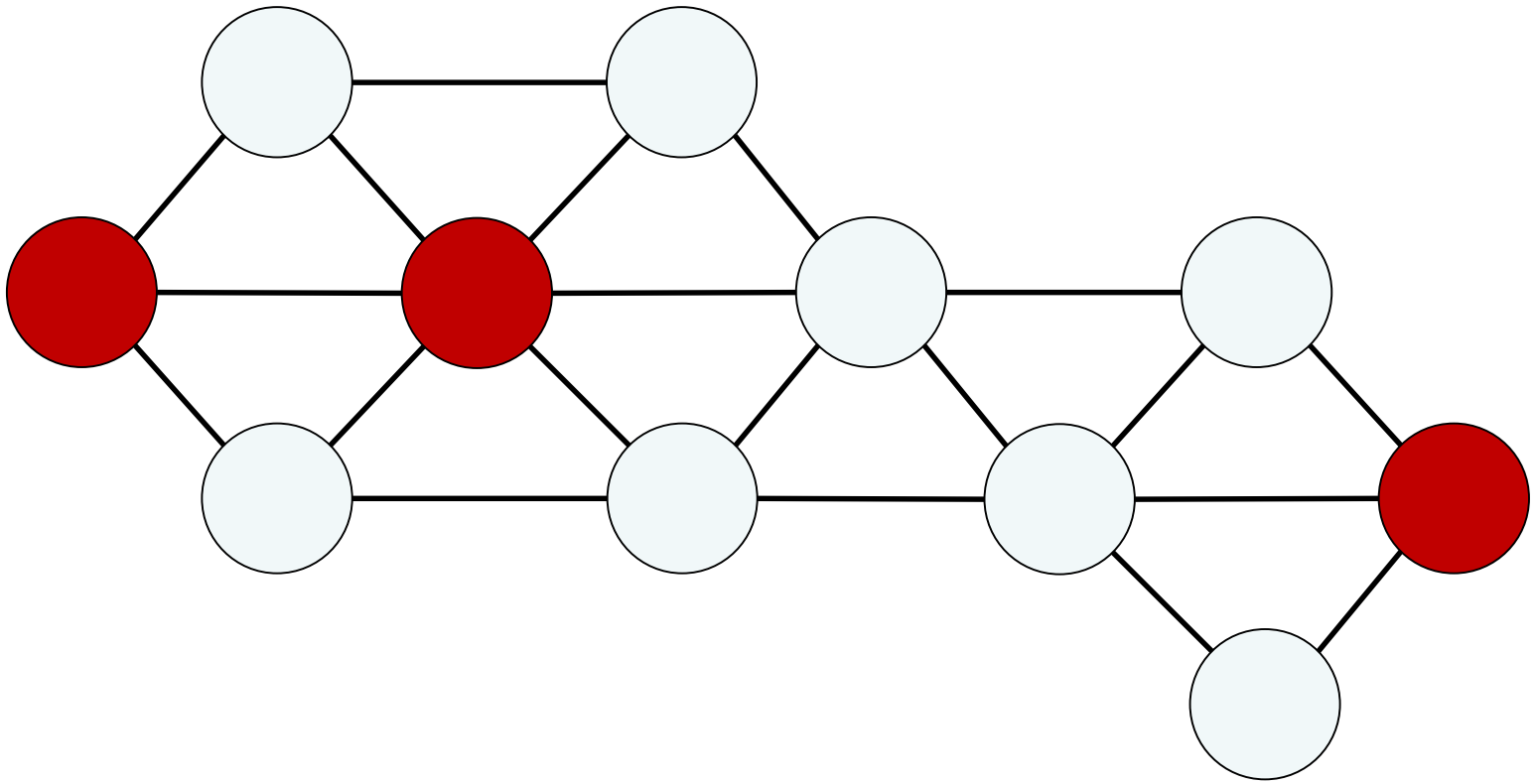
Shortest Path

- Example:



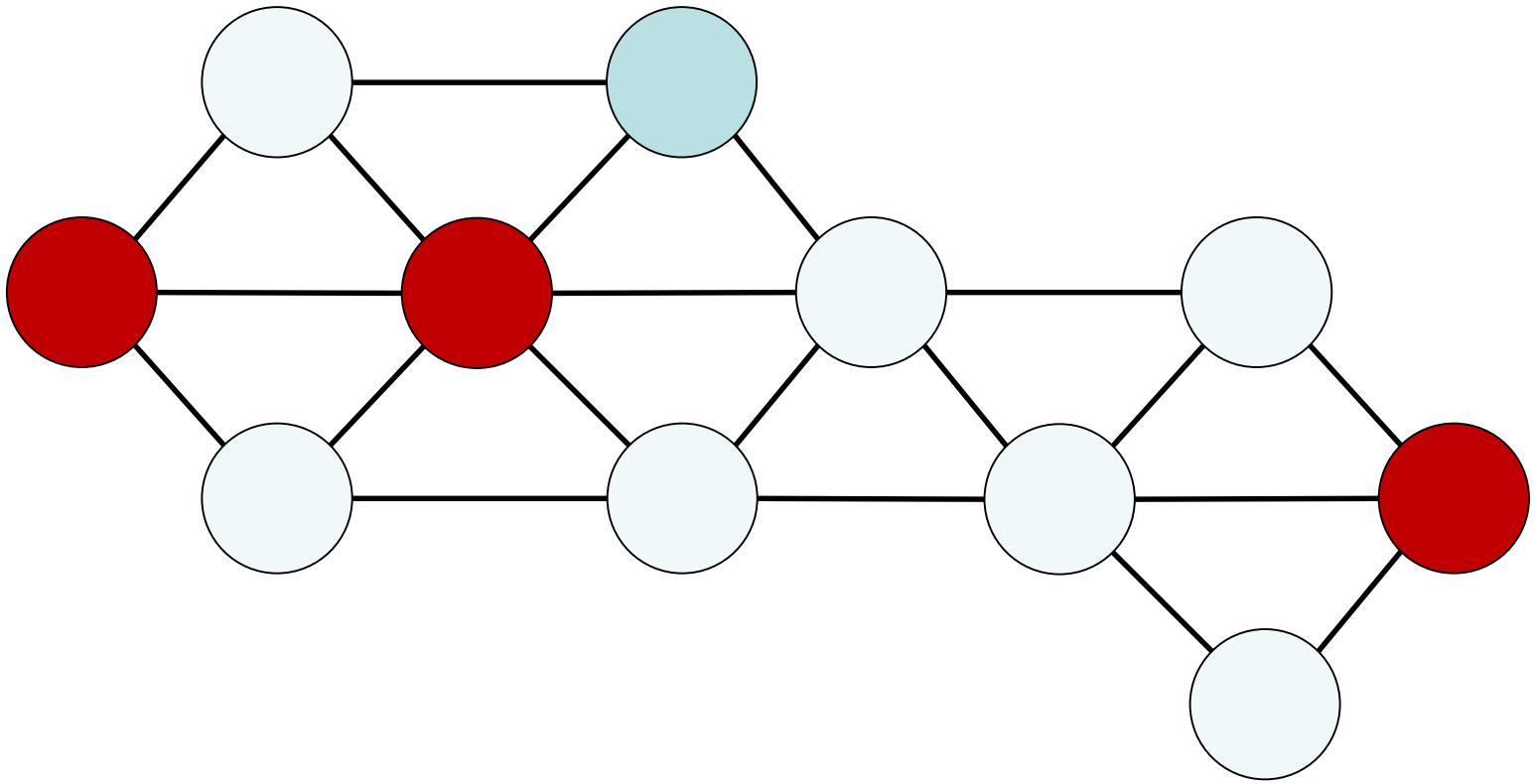
Shortest Path

- Example:



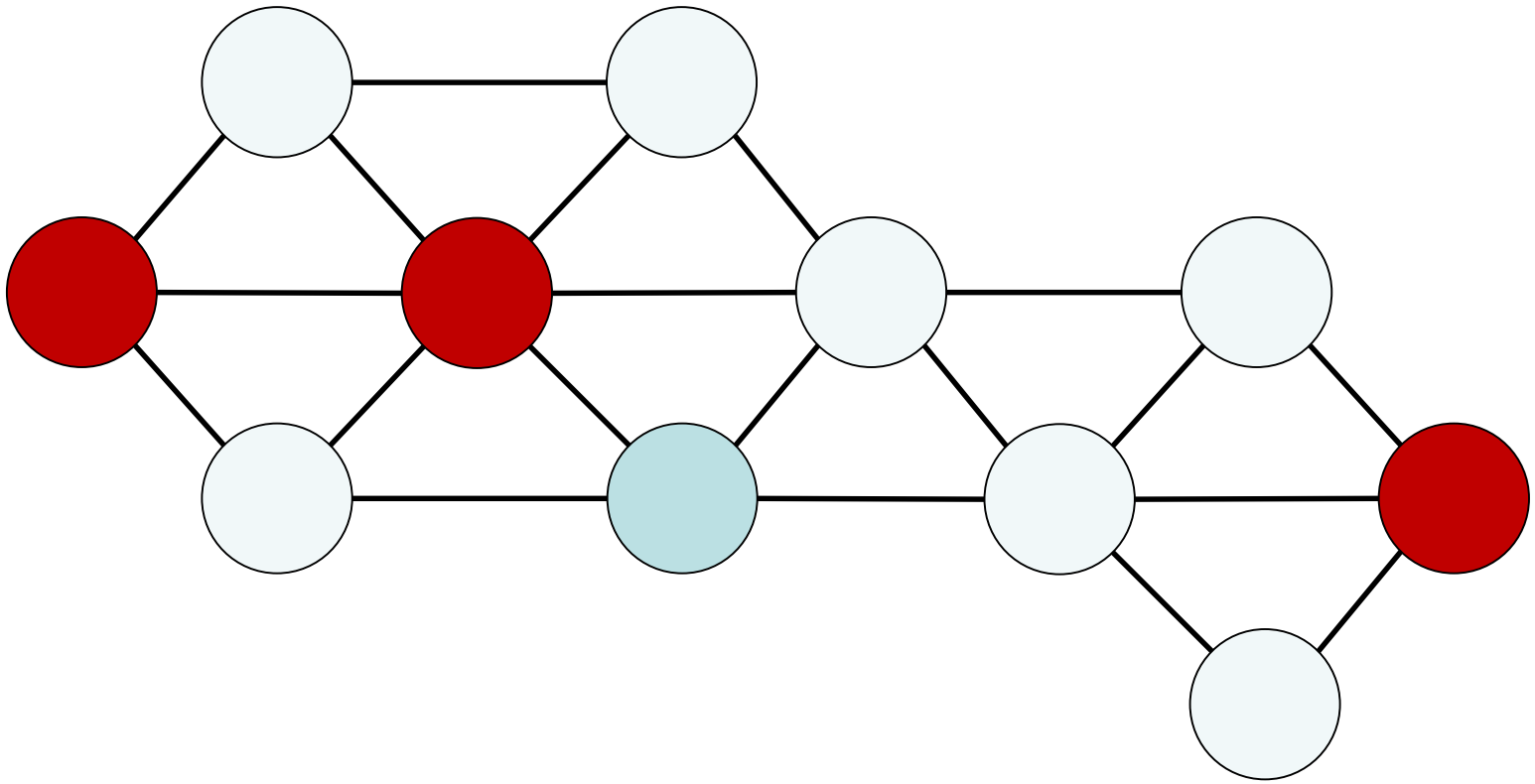
Shortest Path

- Example:



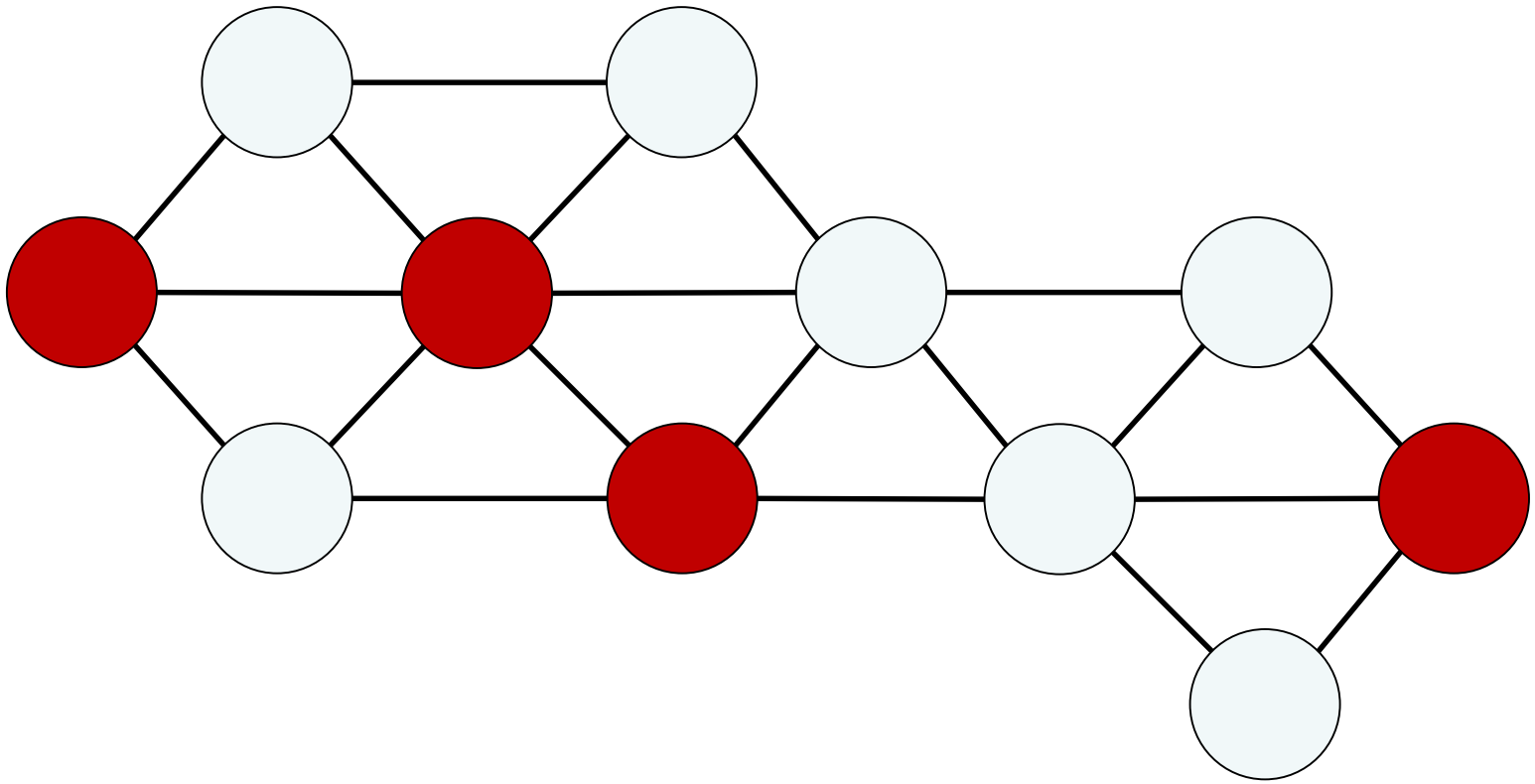
Shortest Path

- Example:



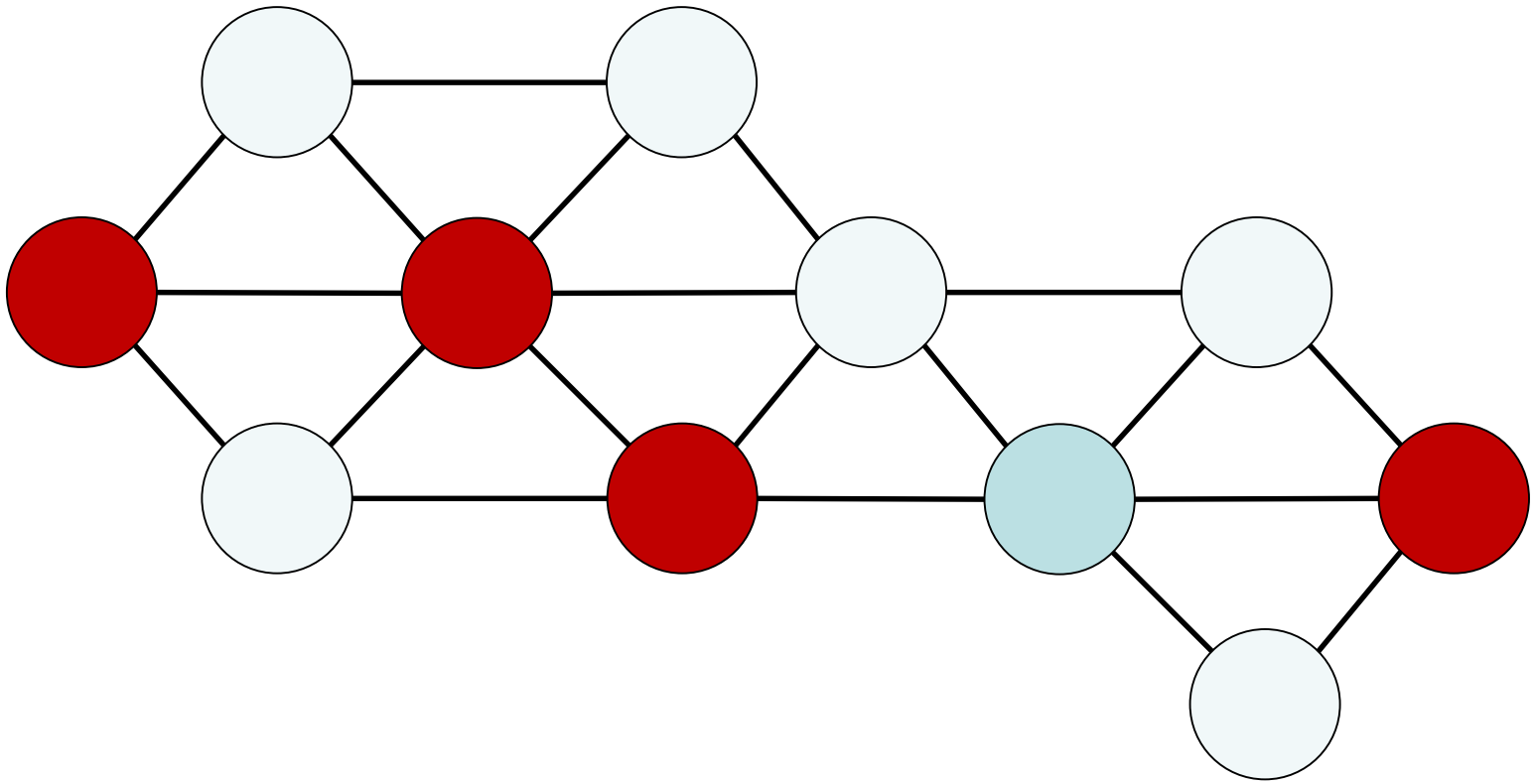
Shortest Path

- Example:



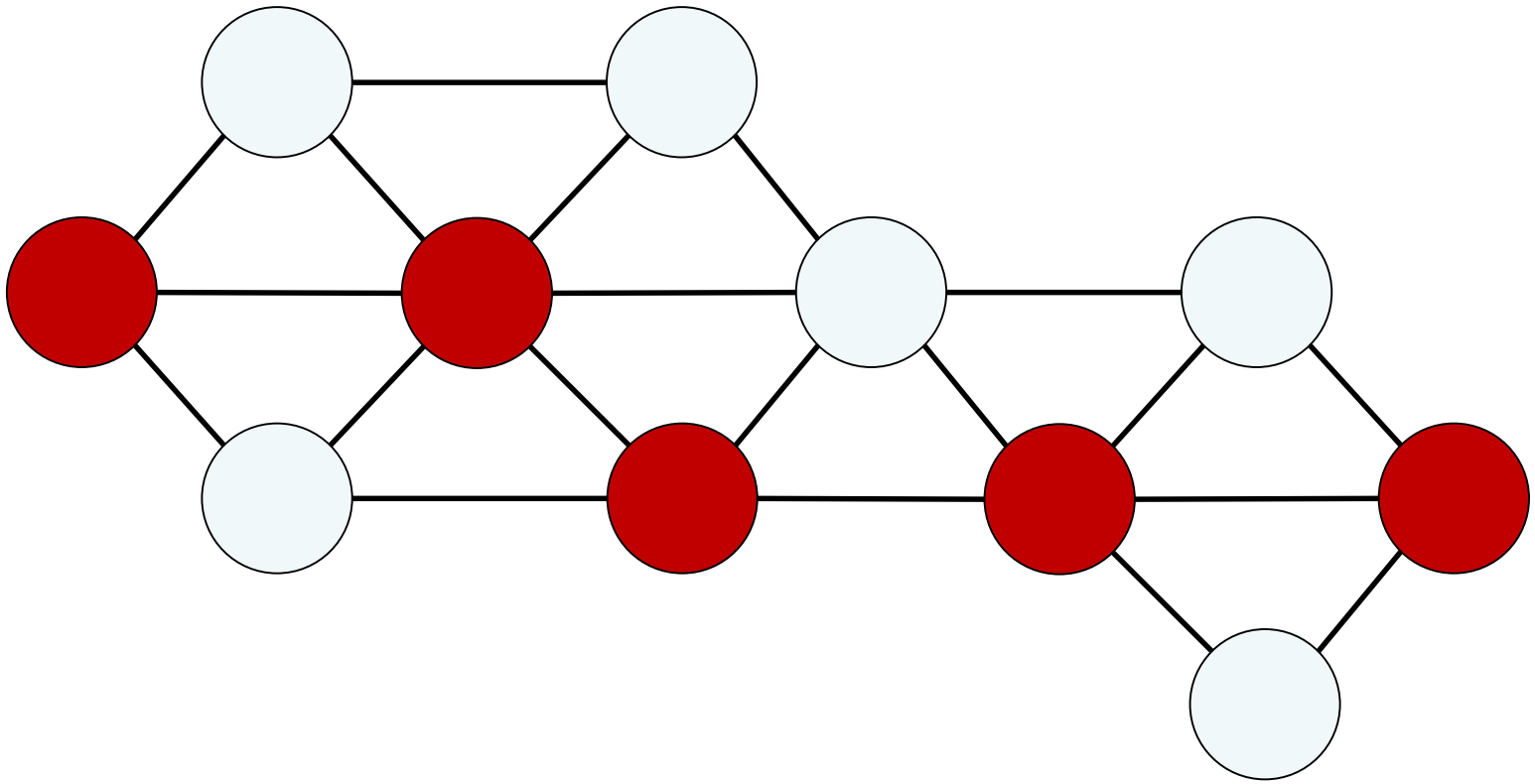
Shortest Path

- Example:



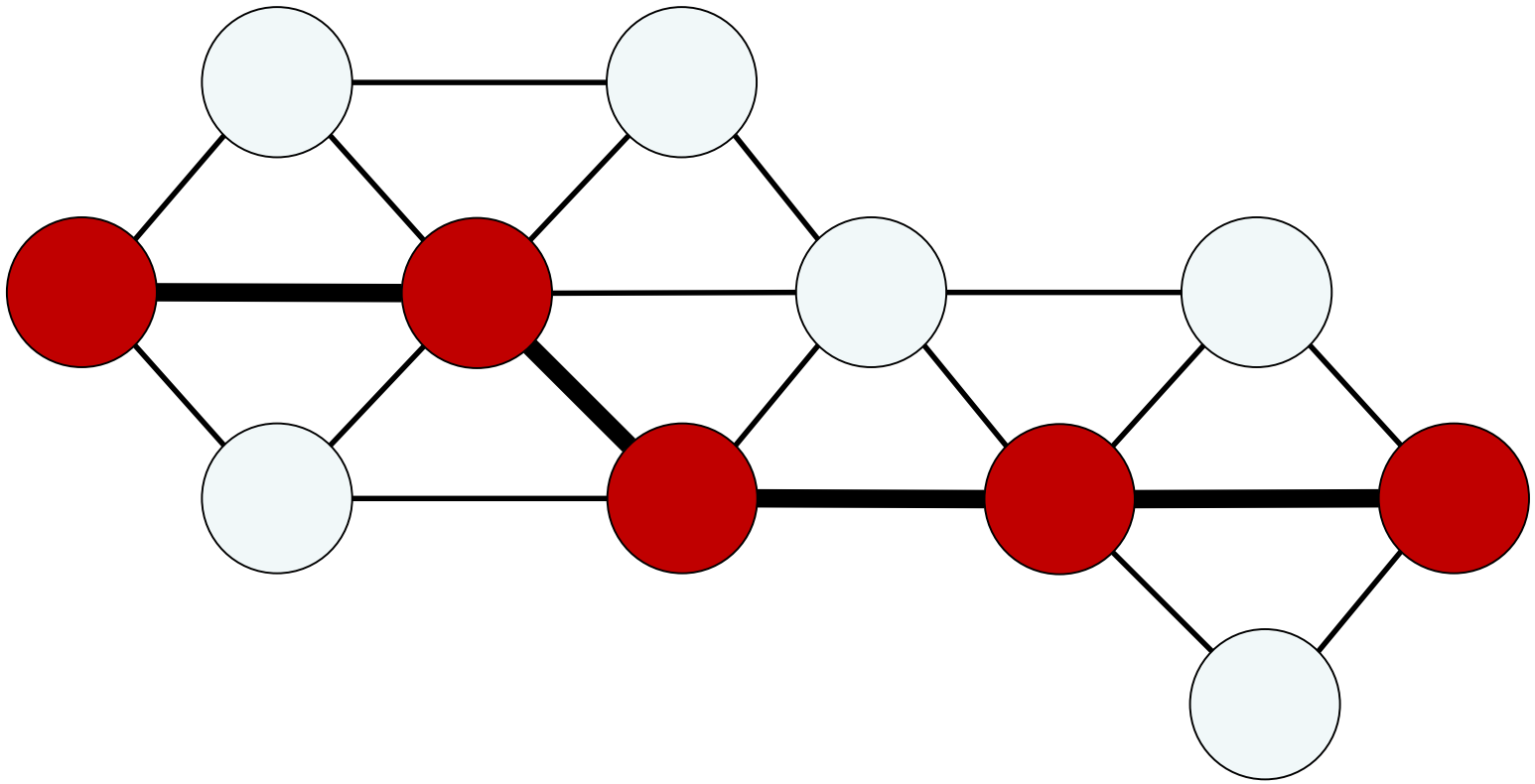
Shortest Path

- Example:



Shortest Path

- Example:



Different Model for the Slime Mold?

- Keep algorithms simple
- How does communication work?
  - Strength of flow through tubes indicates length.
  - We assume additional communication through tubes/plasma.
  - Communicate through moving nuclei?
- Nuclei:
  - How long do they live? Robustness?
  - Do they move?
  - How many are there?
  - Do they communicate / process information?

Note

These are possibility results

- which weak assumptions work
- which don’t

Thanks!

holzer@mit.edu

yemek@ie.technion.ac.il

wattenhofer@ethz.ch