Computational Power of a Single Oblivious Mobile Agent in Two-Edge-Connected Graphs

2022/12/13

○Taichi Inoue, Naoki Kitamura, Taisuke Izumi, Toshimitsu Masuzawa
(Osaka University, Japan)

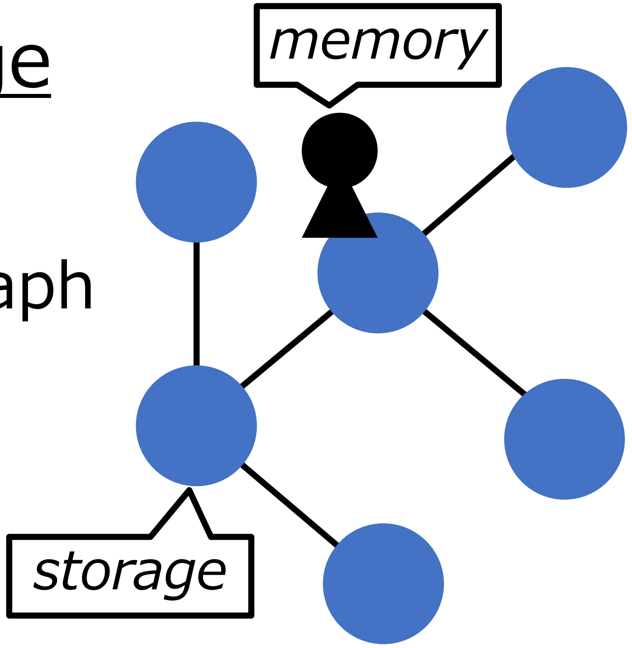
Outline

- Background
- Models
- Ideas for Our Algorithm
- Conclusion

Mobile agents

- One of the computational paradigms in distributed algorithms
  - Autonomously move around graphs, perform computation
  - Hereafter called “**agents**”

- *Our focus*:
  Single-agent systems with memory and storage
  - *Memory*: the information carried by the agent
  - *Storage*: the information held by each node of the graph



Memory size and task solvability

- When memory is large enough:
  - The agent can simulate any centralized algorithm
- When memory size is limited:
  - The agent cannot hold all information of the graph
  - Crucial how to utilize limited information

How much does the difference in memory size affect computational power?

Research question

- Y: any algorithm on graph G with a **y-bit memory agent**
- **Can we design the algorithm simulating Y by a x-bit memory agent? ($x < y$)**
  - If yes, (x-bit memory agent) = (y-bit memory agent) in computational power
- *Known [1]:* (1-bit memory agent) = ($O(n)$-bit memory agent)
  - *Graphs*: n-node, $O(1)$-bit storage per node
  - Polynomial-time overhead per round

[1] Deciding Graph Property by Single Mobile Agent: One-Bit Memory Suffices.
Taisuke Izumi, Kazuki Kakizawa, Yuya Kawabata, Naoki Kitamura, Toshimitsu Masuzawa.
https://arxiv.org/abs/2209.01906

Research question

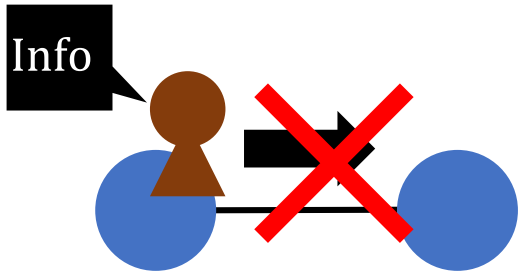
- *Known*: (1-bit memory agent) = ($O(n)$-bit memory agent)
- *Natural question*:

(0-bit memory agent) = (1-bit memory agent)?

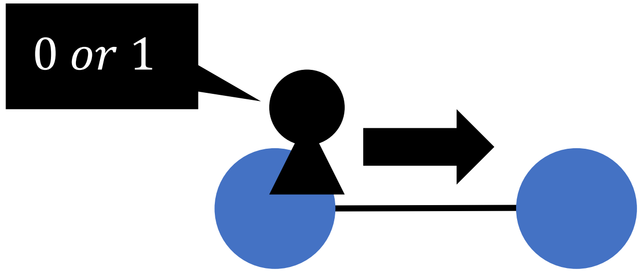
- Hereafter a 0-bit memory agent is called an oblivious agent

Difficulty in simulating 1-bit by 0-bit

- *Oblivious agent*: Cannot explicitly transfer any information



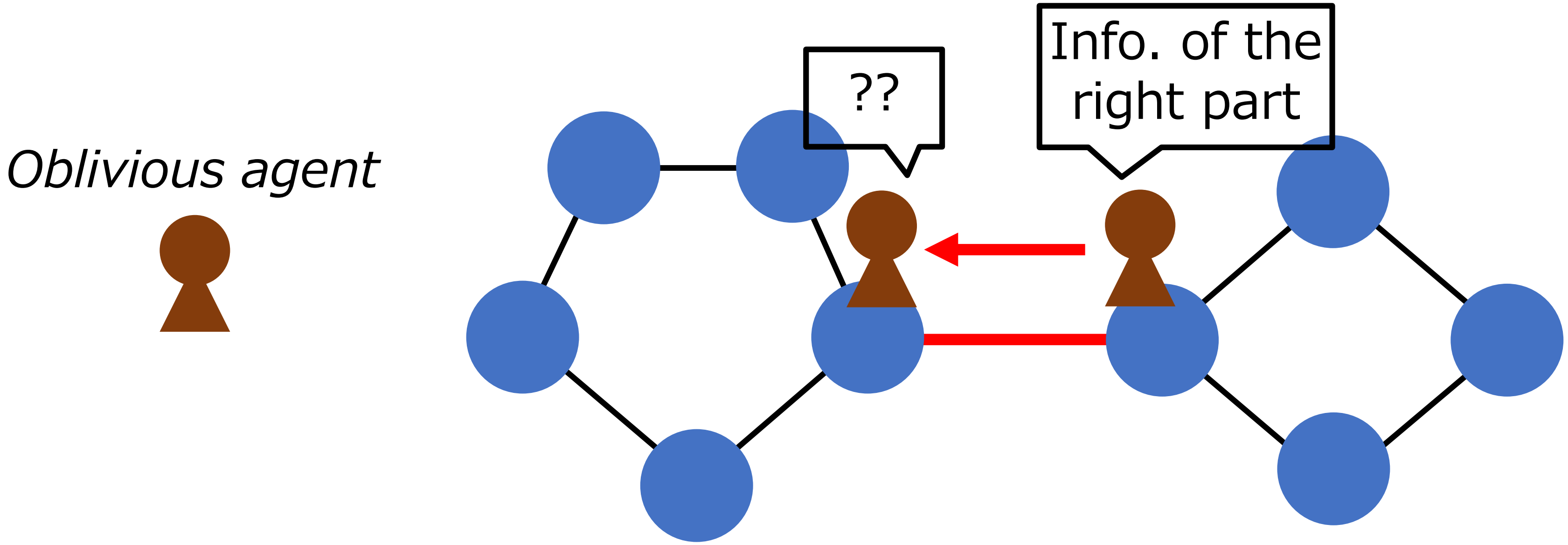
- 1-*bit memory agent*: Can transfer 0 or 1



Difficulty in simulating 1-bit by 0-bit

- **(Oblivious agent) ≠ (1-bit memory agent)**
  - in general settings

- An oblivious agent **cannot** transfer any information **through bridges** (red edge below)
  - **→ there exists a task which cannot be solved by an oblivious agent**



Simulation in 2-edge-connected graphs

- *The essential difficulty*:
  An oblivious agent cannot transfer any information through bridges

- **What happens in graphs without bridges?**
  - **→ 2-edge-connected graphs**

Our contribution

(Oblivious agent) = (1-bit memory agent)
in all graphs without bridges

- “No bridge” = “The possibility of the simulation”

Our contribution

- *Input graph* G: $\boldsymbol{n}$**-node,** $\mathbf{2}$**-edge-connected**
  - $\boldsymbol{O}(\log \boldsymbol{\Delta})$**-bit storage per node** (Δ: maximum degree)

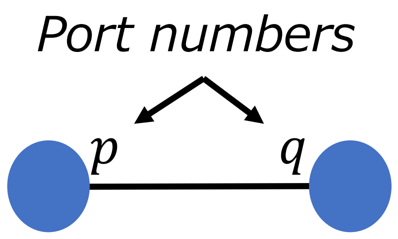
Main Theorem

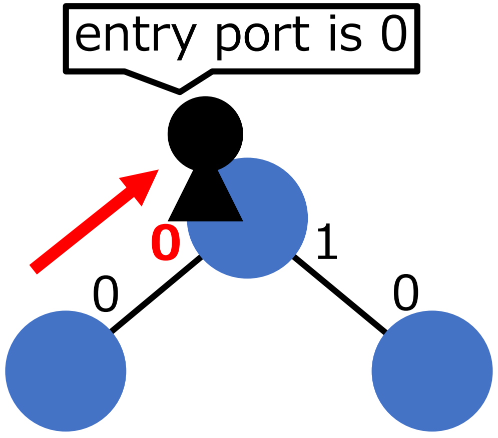
For any algorithm Y on G with a 1-bit memory agent,
we can design an algorithm simulating Y by an oblivious agent.

- (**Oblivious agent**) = ($\mathbf{1}$**-bit memory agent**)
- $O(n^2)$-time overhead per round

Carrying 1-bit data by oblivious agent

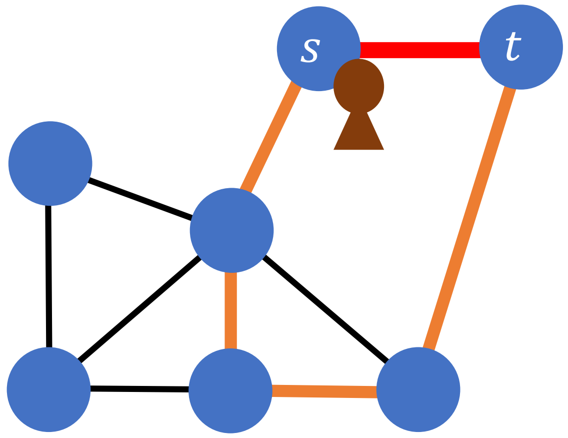
- The simulator agent needs to carry 1-bit information
- *Idea*: Agents can recognize the **port number** when it enters a node (**entry port number**)
  - **Port number** ⋯ a local identifier of each edge incident to nodes



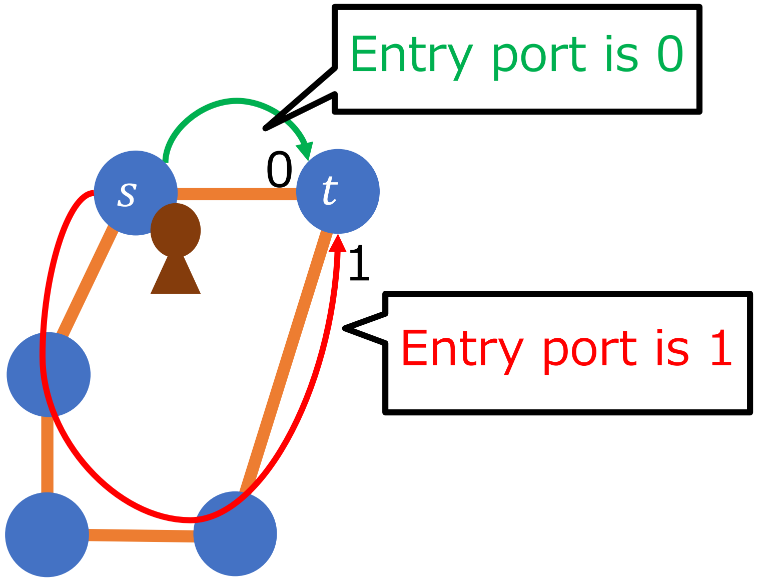


Property of 2-edge-connected graphs

- For any two neighboring nodes s, t,
  there exists one s–t path without edge (s, t)
  - ⇔ **a cycle** including s, t always exists

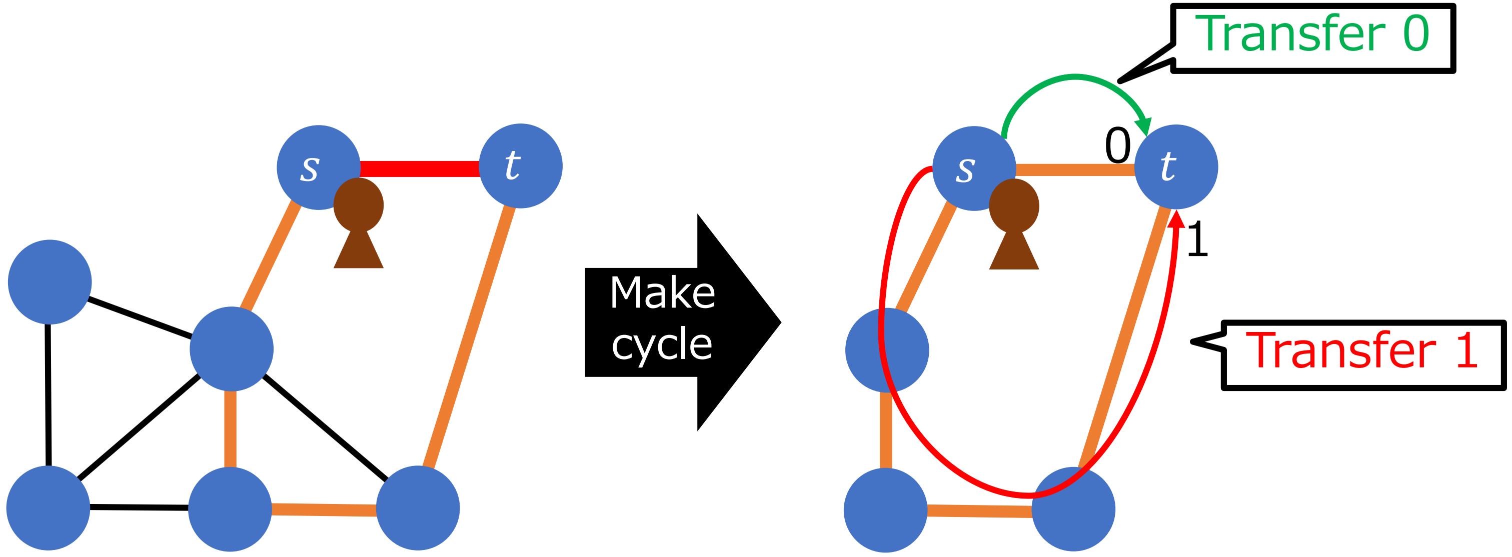


- Different moving direction on the cycle
  = Different entry port numbers



Overview of our algorithm

- Simulation of movement from s to t: Scenario
- 1) Compute the information that will be carried
- 2) Construct a cycle including s, t
- 3) Transfer 1-bit data by the difference of entry port numbers

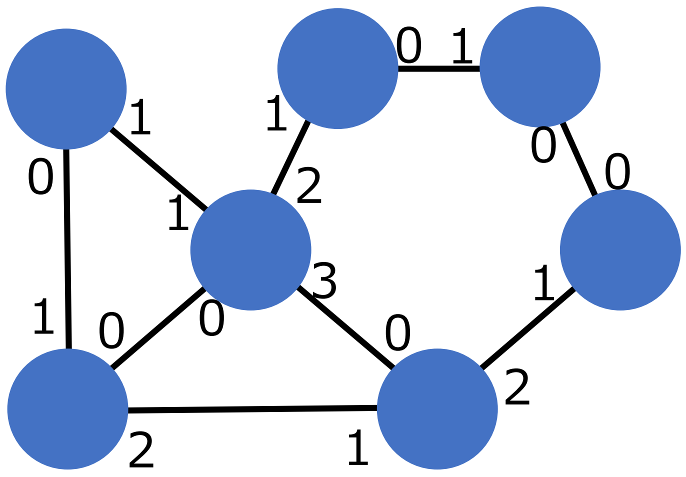


Outline

- Background
- Models
- Ideas for Our Algorithm
- Conclusion

Graphs

- Graph G: n-node, 2-edge-connected
- Each node i has Δ_i edges
  - Labeled by **port numbers** $(0\sim\Delta_i - 1)$
- Δ: the maximum degree of G
- $O(\log \Delta)$-bit storage per node



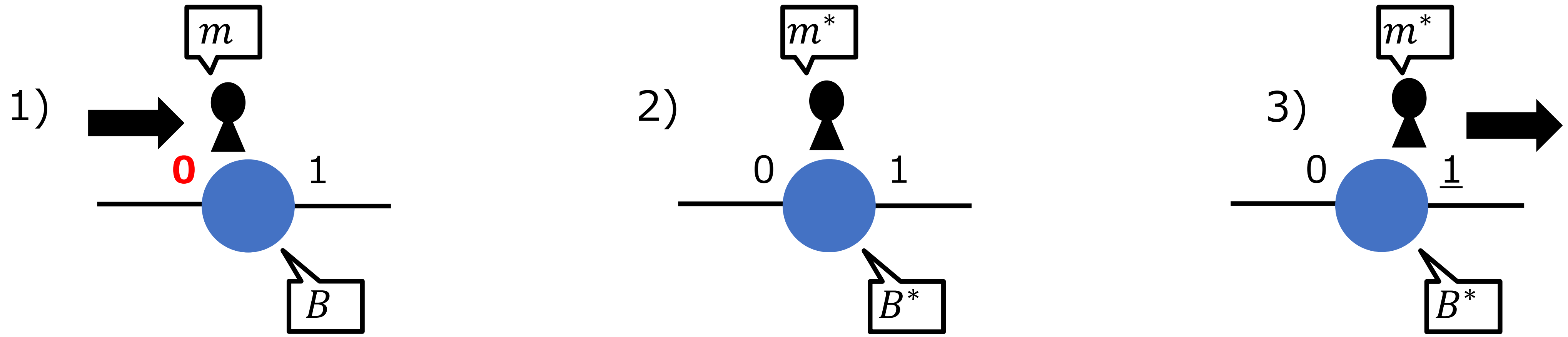
Agents and algorithms

- *Agent and algorithm using <u>1-bit memory</u>*: ***simulated***
- *Ones using <u>0-bit memory</u>*: ***simulator***

	Agents	Algorithms
1-bit memory	***Simulated*** *agent*	***Simulated*** *algorithm*
oblivious	***Simulator*** *agent*	***Simulator*** *algorithm*

Rounds

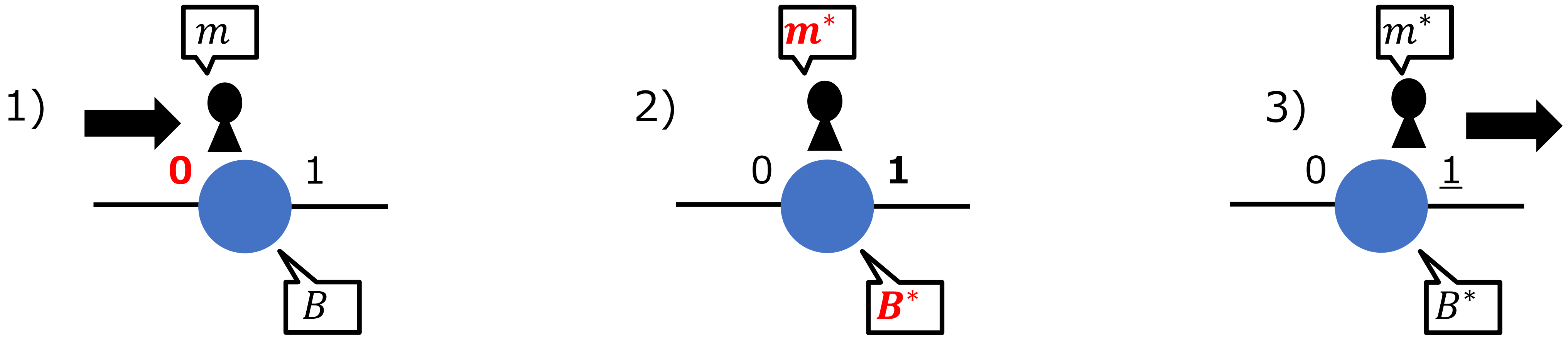
- Agents repeat atomic operations called **rounds**



1)
-enter a node
-recognize the entry port number (0)

Rounds

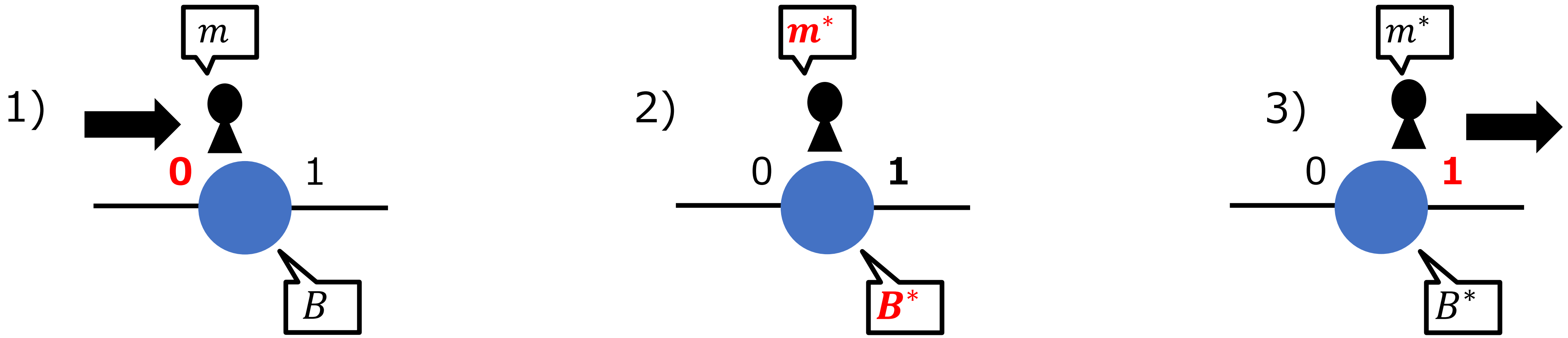
- Agents repeat atomic operations called **rounds**



2)
-compute the next destination (**outgoing port number (1)**)
-update storage/memory

Rounds

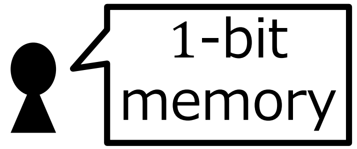
- Agents repeat atomic operations called **rounds**

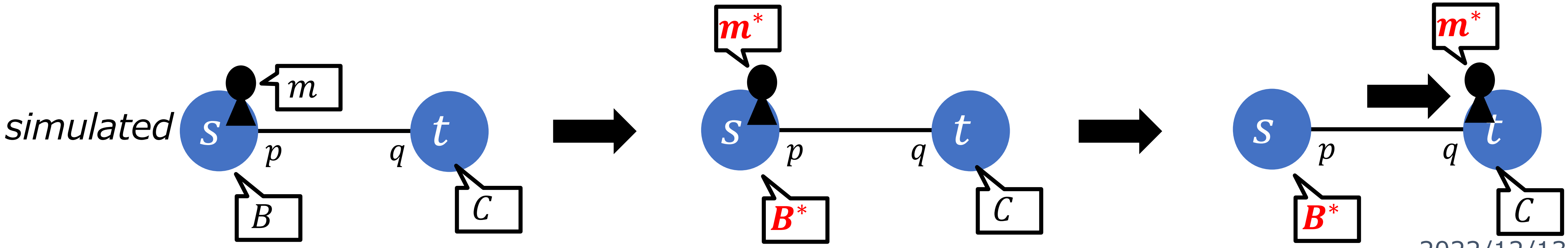


3)
-leave the node following **the outgoing port number**

Simulation

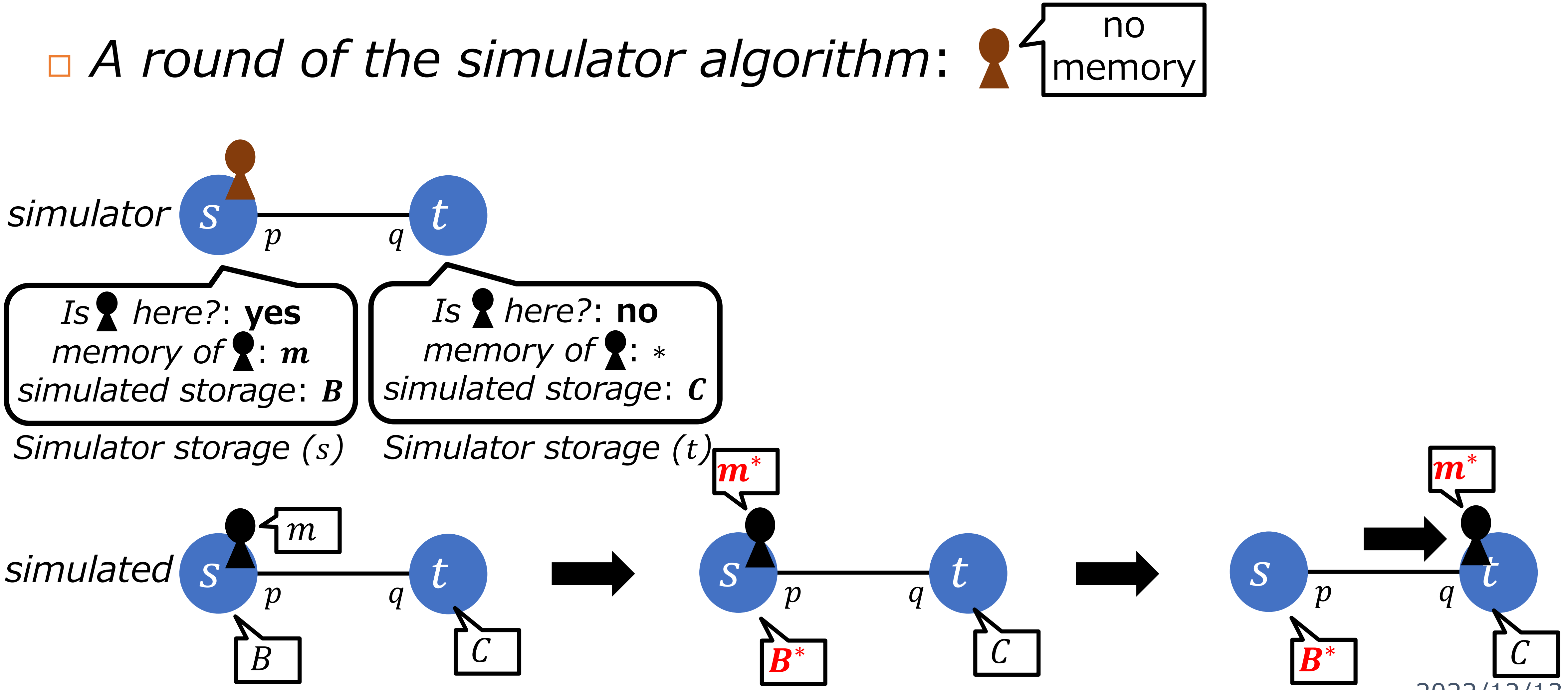
- Simulation of a 1-bit memory agent by an oblivious agent
- *A round of the simulated algorithm*:





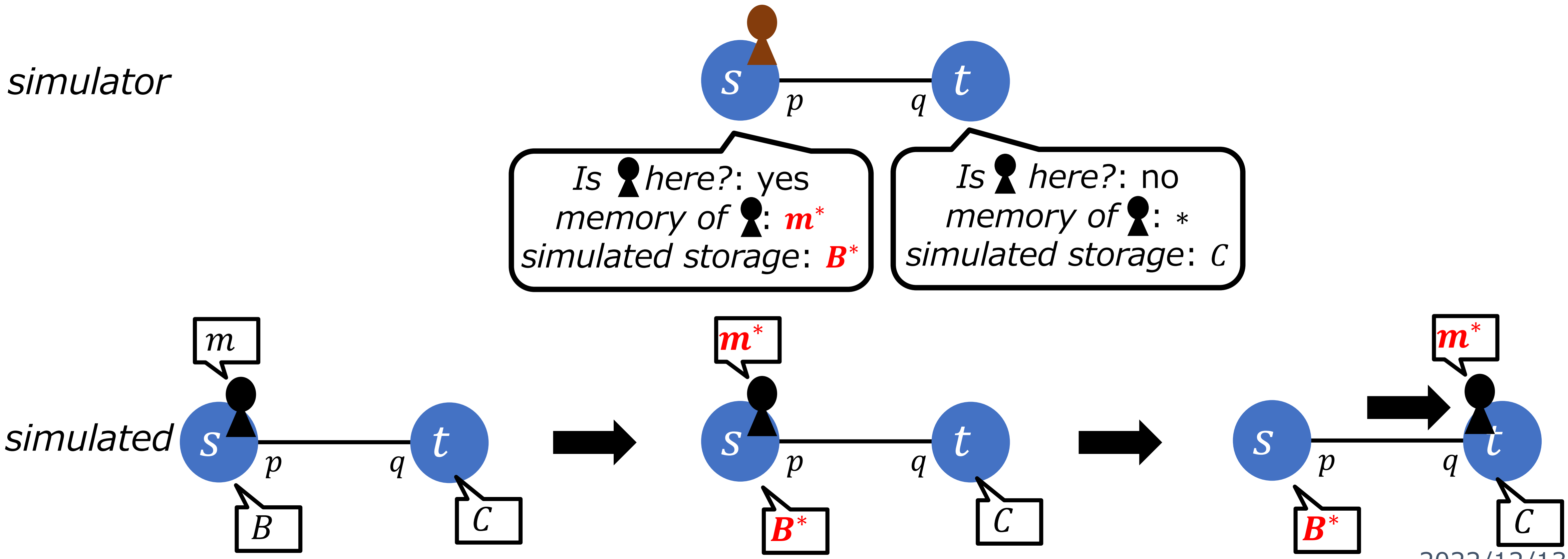
Simulation

- Simulation of a 1-bit memory agent by an oblivious agent
- *A round of the simulator algorithm*:



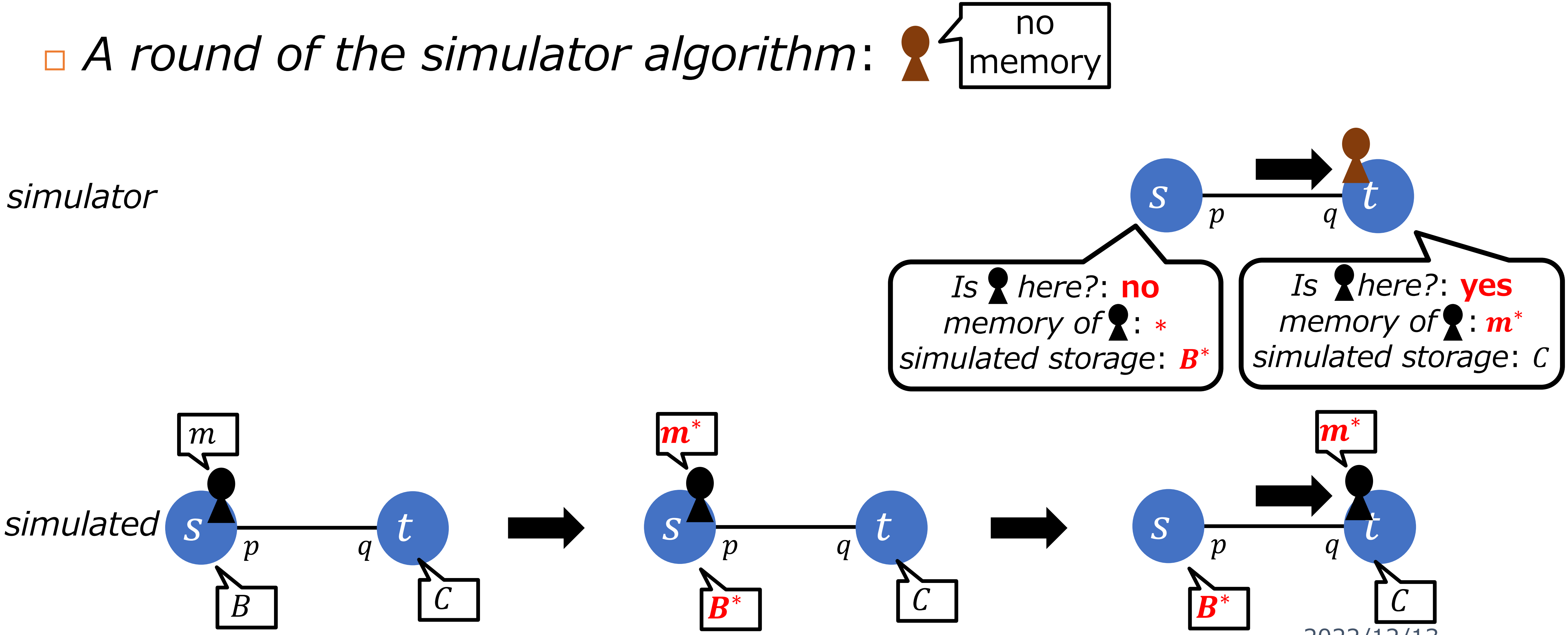
Simulation

- Simulation of a 1-bit memory agent by an oblivious agent
- *A round of the simulator algorithm*: no memory



Simulation

- Simulation of a 1-bit memory agent by an oblivious agent
- *A round of the simulator algorithm*:



Simulation of one round

- Local computation, storage update
  - Trivial
- Transfer information to the neighbor node
  - The location/memory of the simulated agent must be transferred
  - The simulator agent (oblivious) cannot do explicitly
  - **Difficult**
- **The essential problem:**
  How to transfer 1-bit information by the oblivious agent

Outline

- Background
- Models
- Ideas for Our Algorithm
- Conclusion

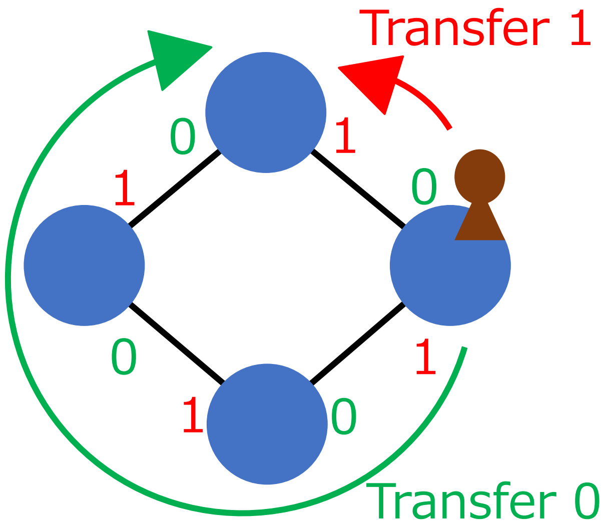
1-bit data transmission in simulator

- **How to transfer 1-bit information by the oblivious agent**
- → use **difference of entry port numbers**
  - Present the idea for **an oriented cycle**
    - the simplest case
  - Extend it for general 2-edge-connected graphs

Transmission in an oriented cycle

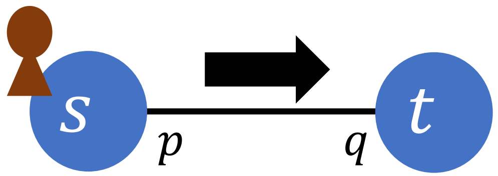
- *Remark:* Each node has 2 ports; 0 and 1

- When the simulator agent transfers value m to a node,
  it enters the node so that the entry port number will be m
  - moves in clockwise when $m = 0$,
    otherwise in counterclockwise
  - one direction = one port number

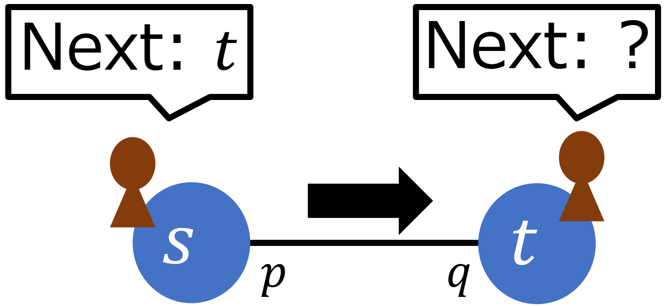


Problem of that idea

- To simulate movement from node s to t, it is enough for the simulator agent to carry 1-bit data to only t



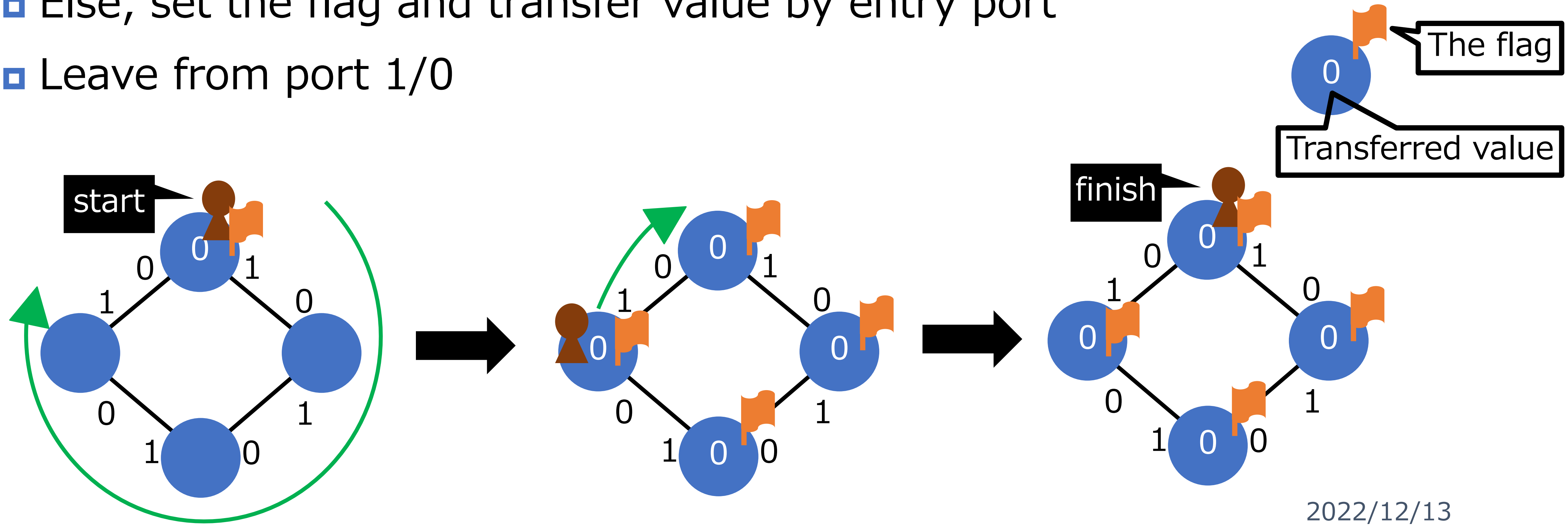
- *Problem*: forget the destination when leave s



  - Can hardly perform an operation <u>only at a specific node</u>
  - But can perform the same operation <u>at all nodes</u>

Transmission to all nodes

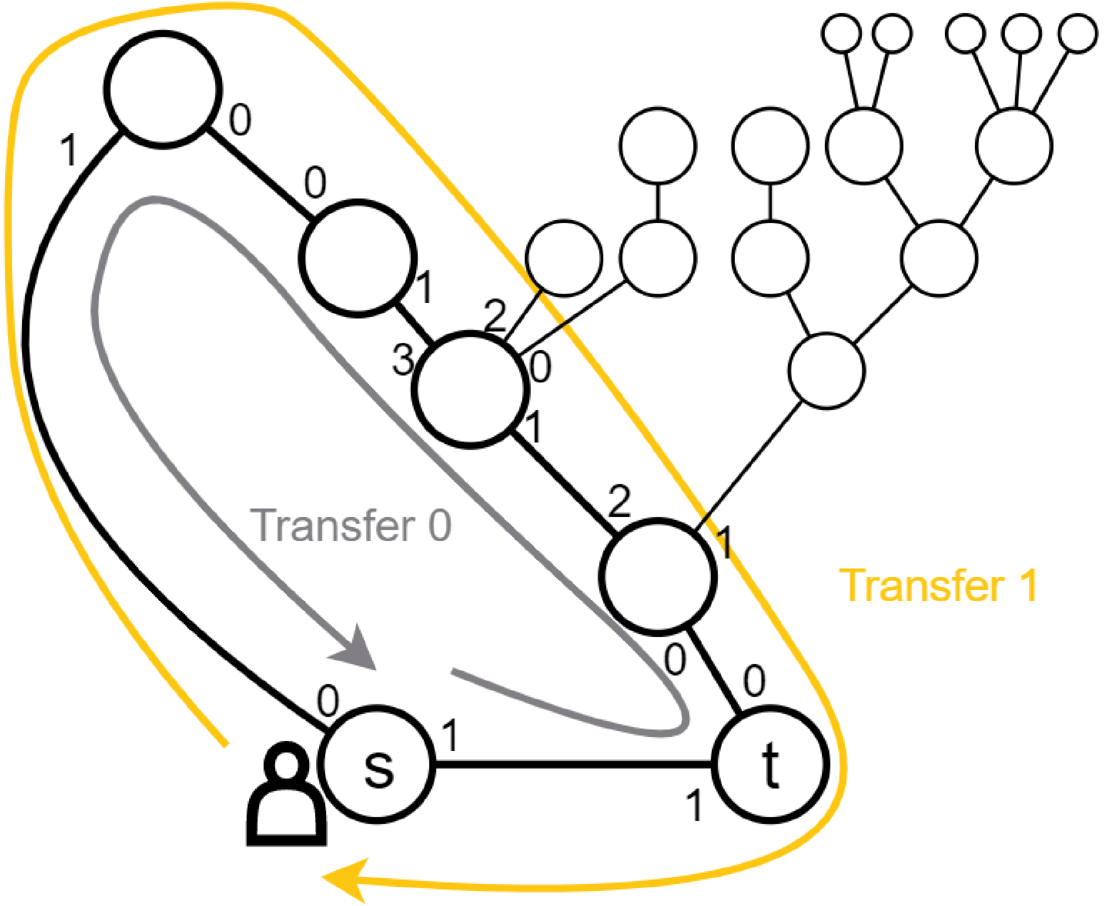
- Repeat the following processes
  - Enter from port 0/1
  - <u>Stop if flag is already set</u>
  - Else, set the flag and transfer value by entry port
  - Leave from port 1/0



Extension to 2-edge-connected graphs

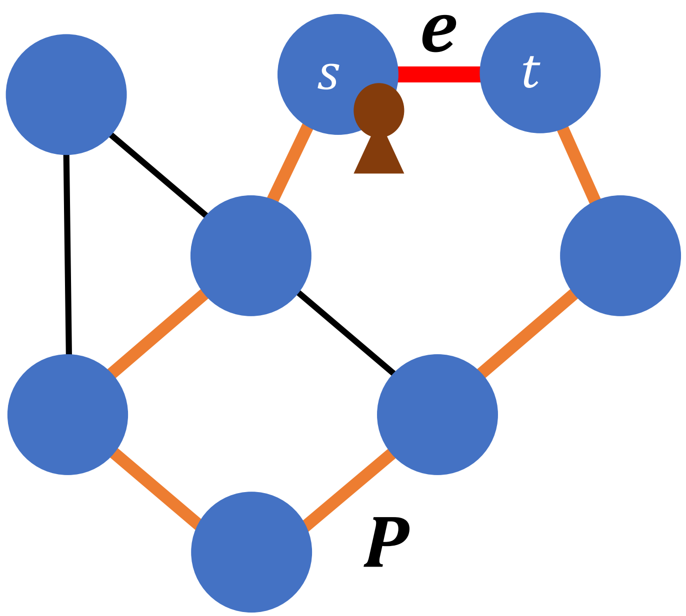
- Simulate movement from node s to t

- Construct a cycle including s, t
  - Necessarily exists
  - Nodes on the cycle manage the incident edges in the cycle
  - The cycle must be oriented
    - Ports in the cycle are not always 0 or 1



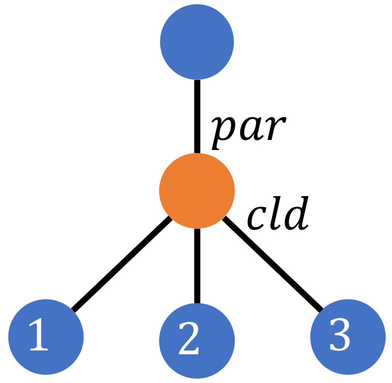
Cycle construction

- Two nodes s, t, the edge $e = (s, t)$
- At least one s–t path P without e exists
- The agent finds the path P through **DFS**
  - Starts DFS from t and continues until reaches s
  - Part of the trajectory = P
- Why DFS?
  - Runnable by an oblivious agent
  - Orientation is determined by the parent-child relation of the DFS tree



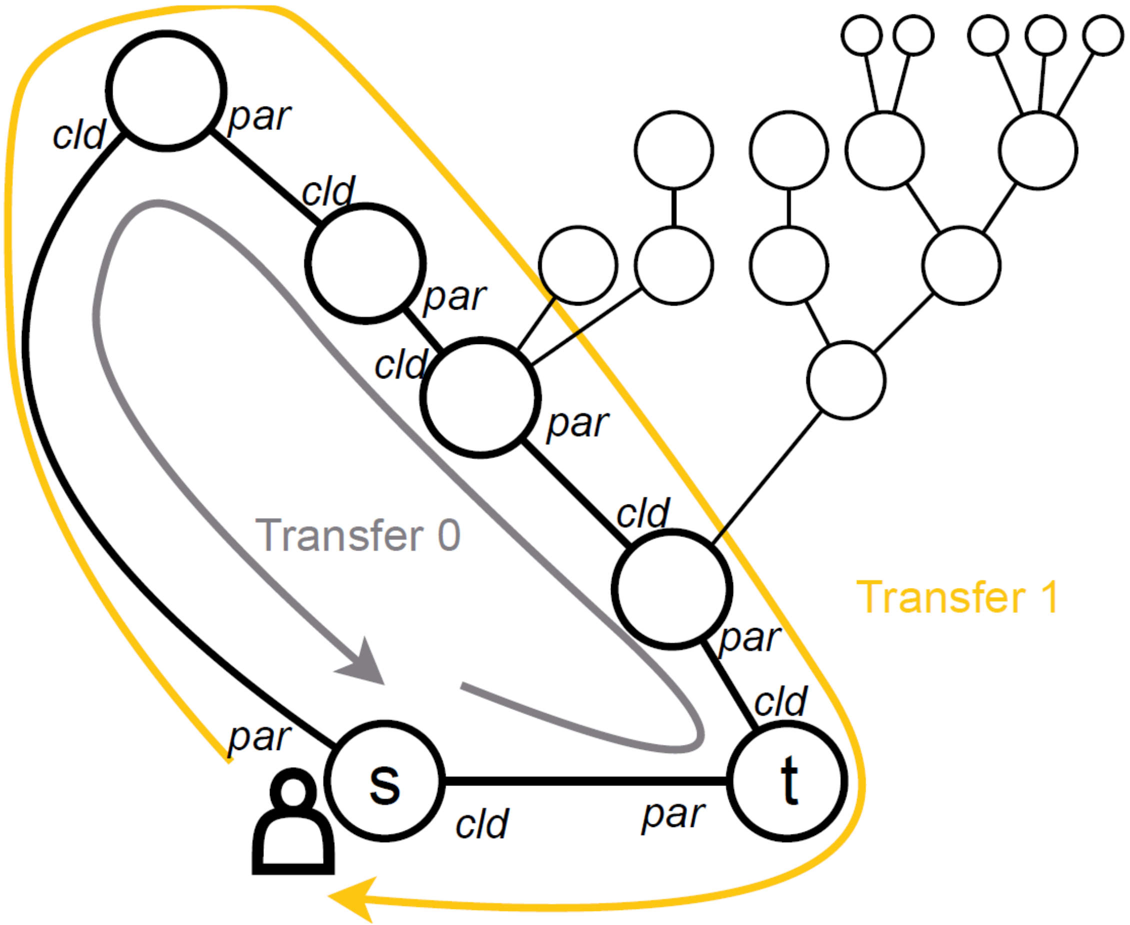
Construction of cycles by DFS

- *Searching order*: increasing order of port numbers
- During DFS, the agent stores two port numbers; parent and child nodes of the DFS tree
  - called par and cld
  - cld holds the child visited latest
  - $O(\log \Delta)$-bit storage required



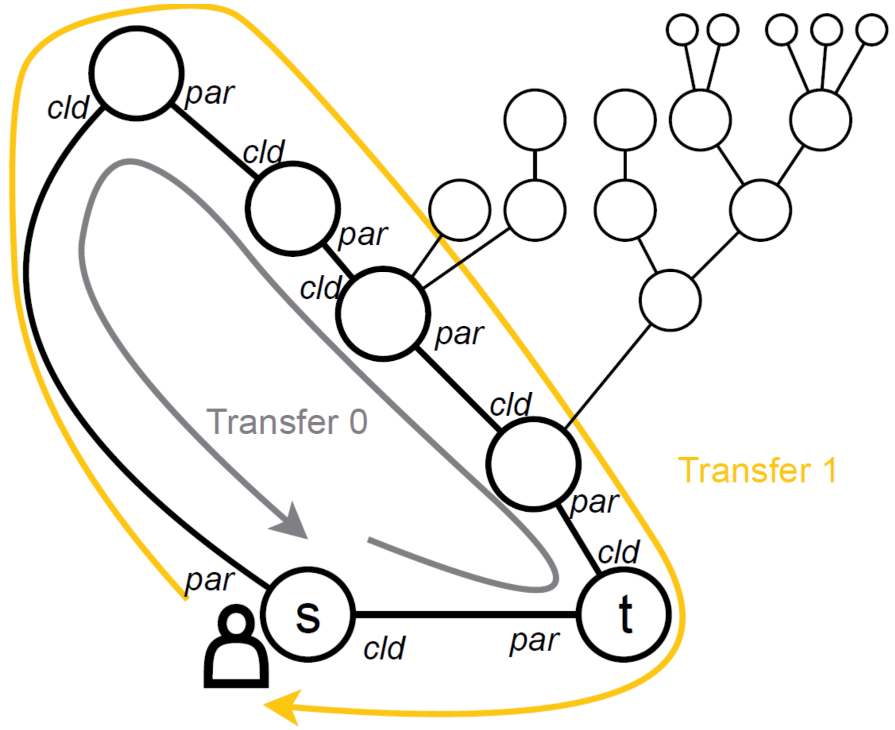
Graphs after DFS

- The cycle has the orientation by par and cld
  - Repeat to choose par = circulate in one direction
- **Methods for oriented cycle can be used**



Cleaning up garbage

- *Problem*: "garbage" information
  - Ex) parent-child ports
- *Requirement*: reset by the beginning of the simulation of the next round
- *Observation*: garbage is left only at the nodes visited by DFS procedure
- → - Re-execute DFS from t
  - Clean garbage except for the info. about the cycle



Outline

- Background
- Models
- Ideas for Our Algorithm
- Conclusion

Conclusion

Study	Graphs	Storage Size	Memory size of simulated algorithm	Memory size of simulator algorithm	Overhead per Round
[1]	arbitrary n-node	$O(1)$	$O(n)$	1	Polynomial time
Ours	n-node, 2-edge-connected	$O(\log \Delta)$ (Δ:maximum degree)	1	0	$O(n^2)$
Open Problem	n-node, 2-edge-connected	**$O(1)$?**	1	0	Polynomial time

[1] Deciding Graph Property by Single Mobile Agent: One-Bit Memory Suffices.
Taisuke Izumi, Kazuki Kakizawa, Yuya Kawabata, Naoki Kitamura, Toshimitsu Masuzawa.
https://arxiv.org/abs/2209.01906

Thank you for your kind attention!