Gathering for mobile agents with a strong team in weakly Byzantine environments

○Jion Hiorse[1] Masashi Tsuchida[1] Junya Nakamura[2]

Fukuhito Ooshita[1] Michiko Inoue[1]

[1]Nara Institute of Science and Technology

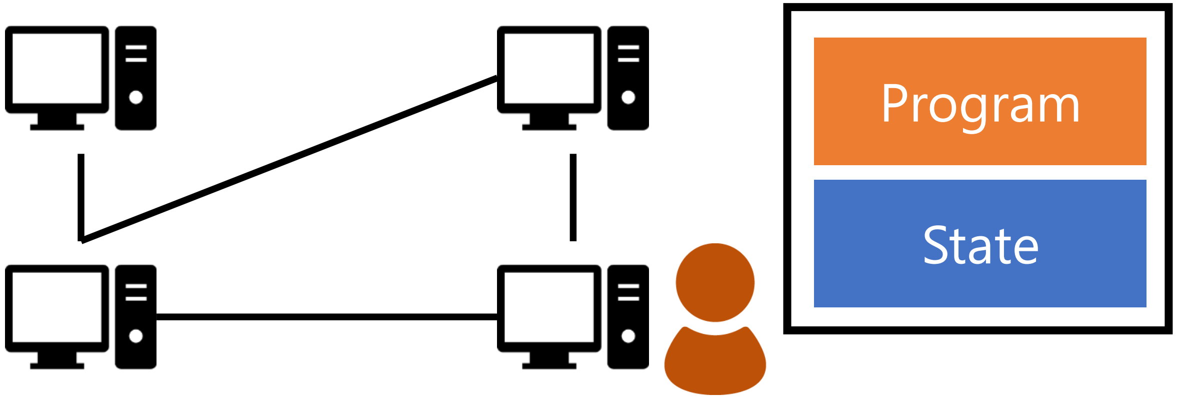
[2]Toyohashi University of Technology

Agenda

- Background
- Contribution
- Model & Goal
- Proposed Algorithm
  - Basic Idea
  - Details
- Conclusion

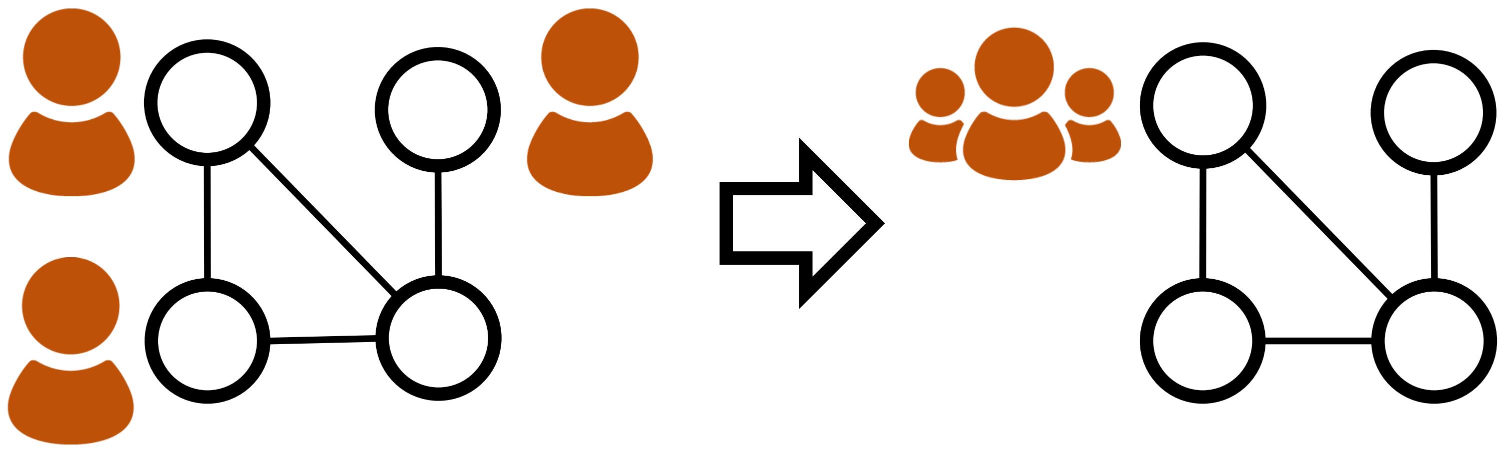
Mobile agents

- Software programs moving autonomously from node to node in a distributed system
  - An agent can keep its state and program during move
- A paradigm to design distributed systems



Gathering

- Goal: All agents meet at a single node
- Efficient information sharing scheme among all agents
- Many researches on various models:
  - Presence/Absence of whiteboards (memories on nodes)
  - Anonymous/Distinct agents (no/unique IDs)
  - Synchronous/Asynchronous agents, etc.

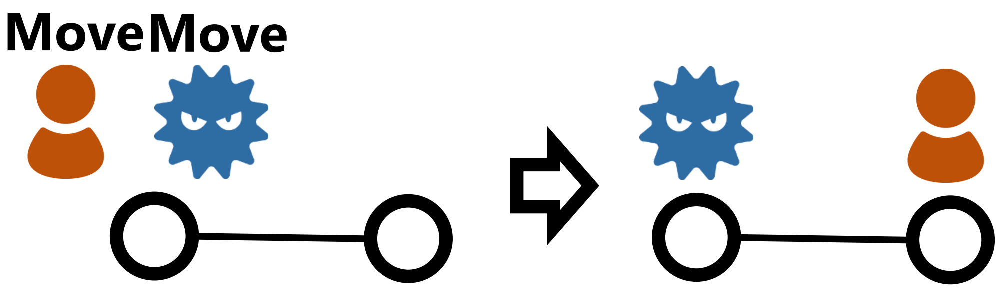


Byzantine environments

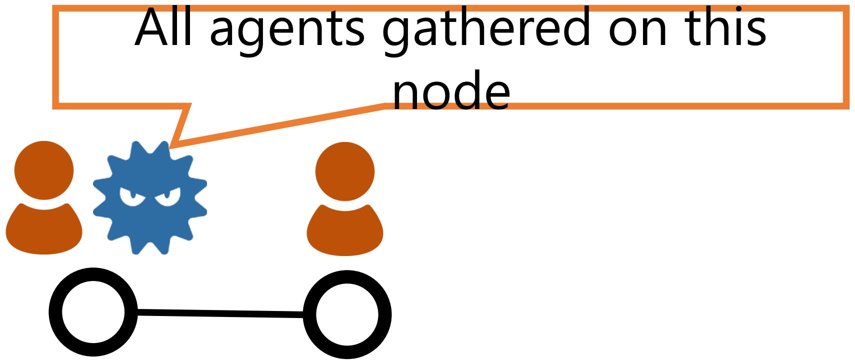
- Byzantine agents exist in the systems
  - They behave arbitrarily
  - They imitate software bugs, cracked agents, etc.

Example

Arbitrary behavior



False information



Algorithms for Byzantine environments tolerate any fault of agents

Gathering in synchronous Byzantine environments

Time complexity of the existing algorithms isn't small

	Input	Byzantine fault	Condition of #Byzantine agents	Time complexity	
[1]	n	Weak	$f+1 \leq k$ (Optimal)	$O\left(n^4 \cdot \lvert\Lambda_{good}\rvert \cdot X(n)\right)$	(1)
[1]	f	Weak	$2f+2 \leq k$ (Optimal)	Poly. of n & $\lvert\Lambda_{good}\rvert$	> (1)
[2]	n, f	Strong	$2f+1 \leq k$ (Optimal)	Exp. of n & $\lvert\Lambda_{good}\rvert$	
[2]	f	Strong	$2f+2 \leq k$ (Optimal)	Exp. of n & $\lvert\Lambda_{good}\rvert$	
[3]	$\lceil \log\log n \rceil$	Strong	$5f^2+7f+2 \leq k$	Poly. of n & $\lvert\Lambda_{good}\rvert$	> (1)

n : #nodes, k : #agents in the network, f : #Byzantine agents, $\lvert\Lambda_{good}\rvert$: The length of the largest ID among good agents,
$X(n)$: Time required to visit all n nodes

[1]Y. Dieudonné, et al., ACM Trans on Algorithms 11, (201
[2]S. Bouchard, et al., Distributed Computing 29(6), (2016
[3]S. Bouchard, et al., In:ICALP. (2020)

Contribution

- Reduce time complexity in weakly Byzantine environments by relaxing the condition of #Byzantine agents
  - #Byzantine agents is small in real networks

	Input	Startup delay	Condition of #Byzantine agents	Simultaneous termination	Time complexity
[1]	n	Presence	$f + 1 \le k$	Possible	$O\left(n^4 \cdot \lvert\Lambda_{good}\rvert \cdot X(n)\right)$
Algo. 1	N	Absence	$4f^2 + 9f + 4 \le k$	Impossible	$O\left(\left(f + \lvert\Lambda_{good}\rvert\right) \cdot X(N)\right)$
Algo. 2	N	Absence	$4f^2 + 9f + 4 \le k$	Possible	[illegible]

$f < n$ & $|\Lambda_{all}| = O(|\Lambda_{good}|)$
$\Rightarrow$ faster than [1]

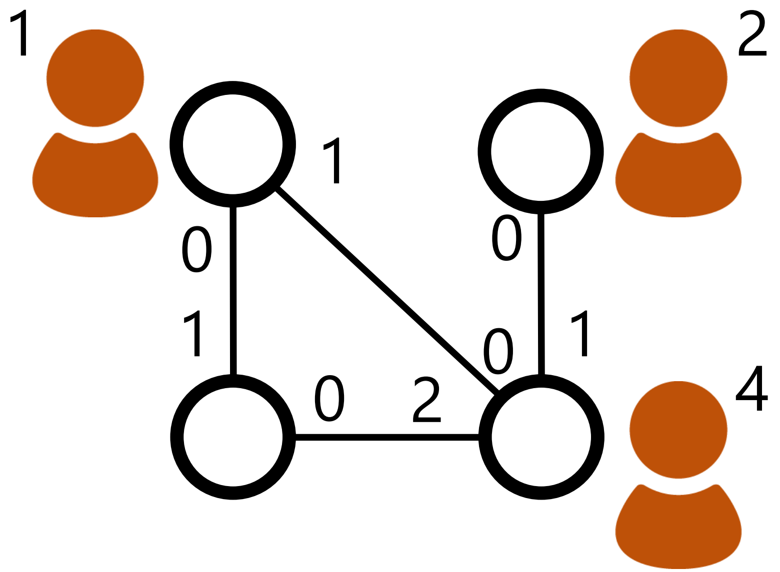
n : #nodes, N : upper bound of #nodes, k : #agents in the network,
f : #Byzantine agents, $|\Lambda_{good}|$: The length of the largest ID among good age
$|\Lambda_{all}|$: The length of the largest ID among agents, $|\Lambda_{good}| \le |\Lambda_{all}|$,
$X(n)$: Time required to visit all n nodes

Agenda

- Background
- Contribution
- Model & Goal
- Proposed Algorithm
  - Basic Idea
  - Details
- Conclusion

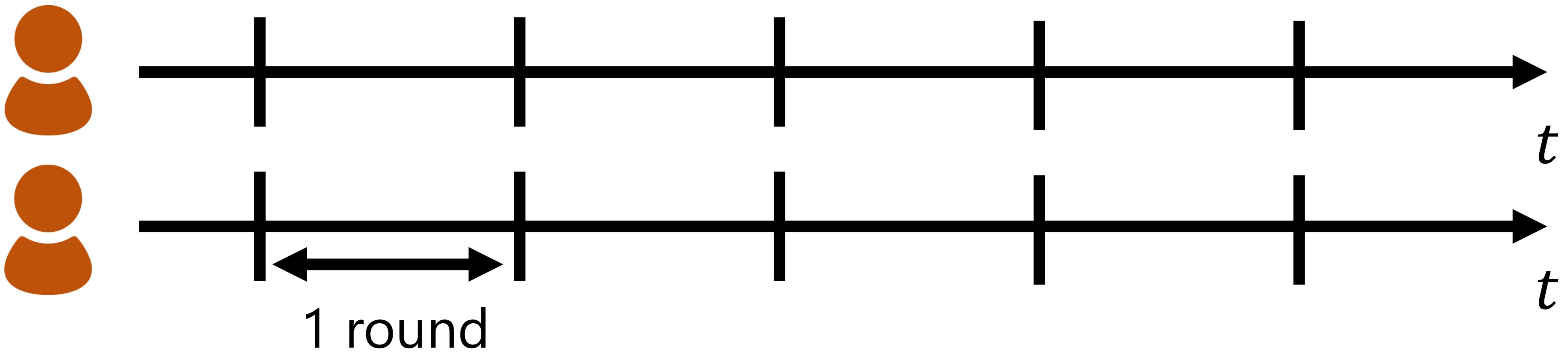
Model

- A distributed system
  - A node has neither ID nor whiteboard
  - Edges incident to a node are locally ordered with a fixed port numbering
- Agents
  - Awake at the same time (no start-up delay)
  - Have unique IDs
  - Know the upper bound N of #nodes
  - Have unlimited amount of memory
  - Can share information with other agents on the same node

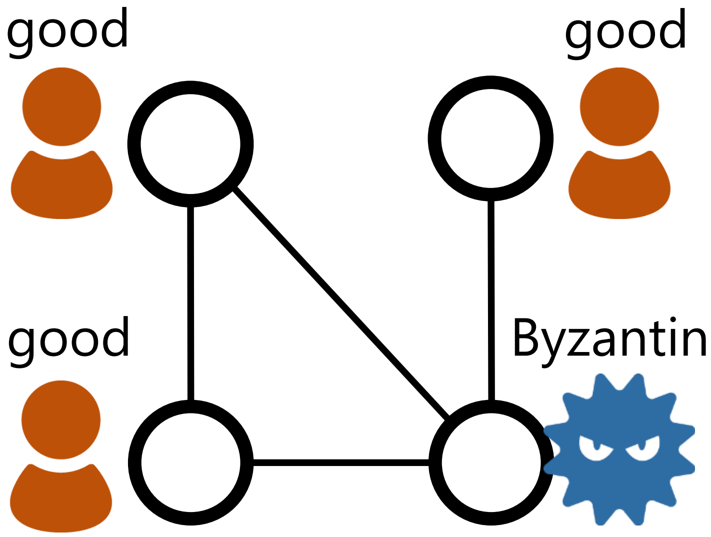


Model

- Synchronous agents
  - An agent can move to its adjacent node in 1 unit time (round)



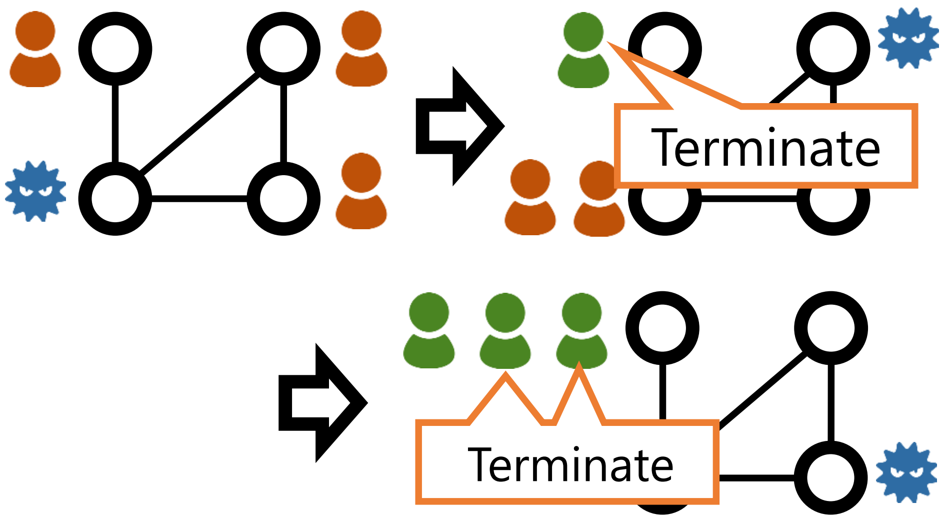
- A weakly Byzantine environment
  - f weakly Byzantine agents
    - Behave arbitrarily without following an algorithm
    - Can't change their IDs
  - At least $4f^2 + 8f + 4$ good agents



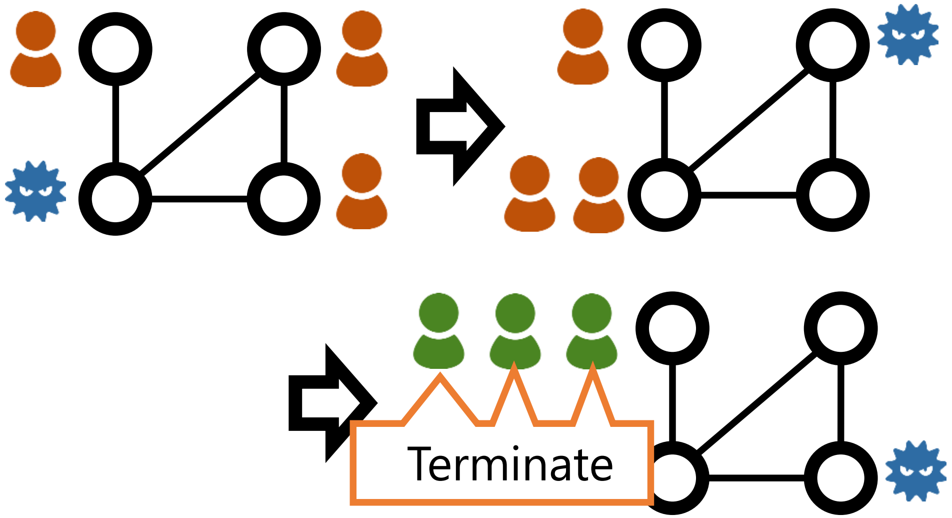
Goal

- Goal: Gathering in a weakly Byzantine environment
  - All good agents gather on a single node
    - Don't care Byzantine agents' locations
- Two types of termination
  - Non-simultaneous termination: All good agents terminate
  - Simultaneous termination: All good agents terminate at the same round

Non-simultaneous termination



Simultaneous termination



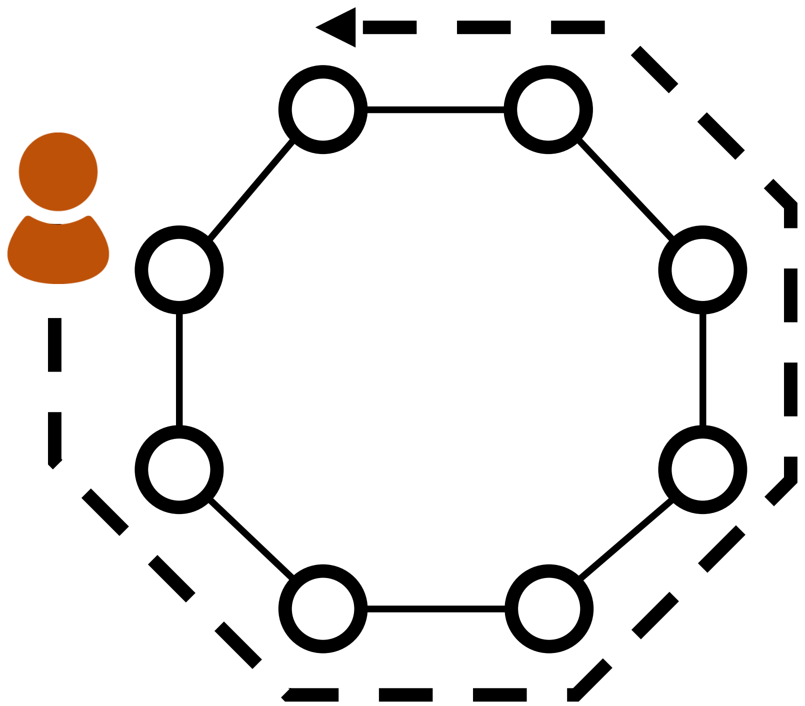
Agenda

- Background
- Contribution
- Model & Goal
- Proposed Algorithm
  - Basic Idea
  - Details
- Conclusion

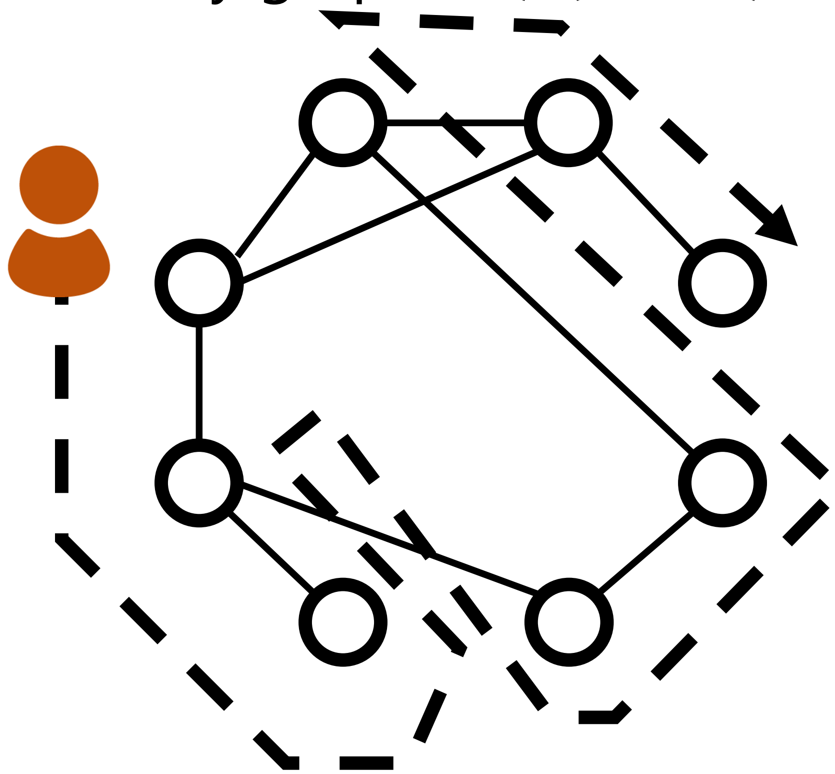
Preparation: Exploration algorithm

- Exploration: Visiting all the nodes in a network
  - We use the existing algorithm [4]
  - The time complexity is denoted by $X(N)$
    - N: Upper bound of #nodes

Tree and Ring: $X(N) = O(N)$



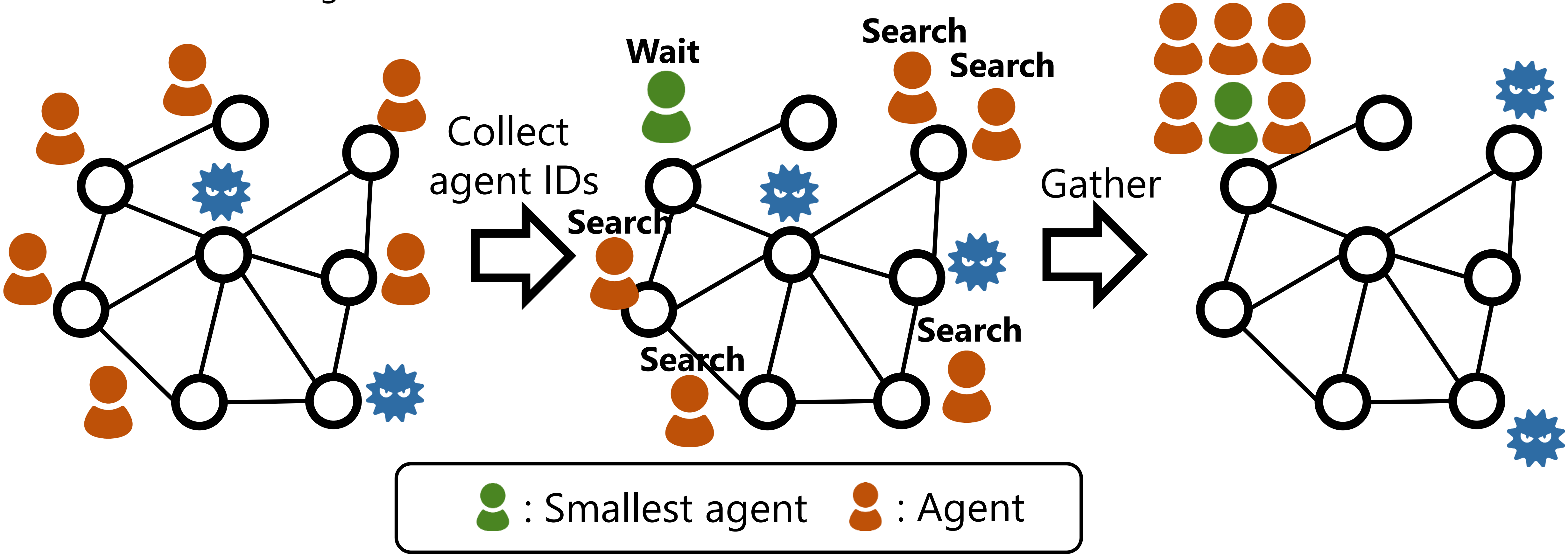
Arbitrary graph: $X(N) = O(N^5 \log N)$



[4] Ta-Shma, A., et al., In: SODA. (2007)

Basic strategy

- Gather on the node with the agent with the smallest ID
  1. Agents collect all agent IDs
  2. Agents search for the agent with the smallest ID
     - The agent with the smallest ID waits for their arrival



Gathering: Difficulty & Strategy

<u>Difficulty</u>

- If a Byzantine agent has the smallest ID, agents fail to gather
  - Agents don't meet the Byzantine agent when collecting IDs
    ⇒ The smallest ID that agents know may be different

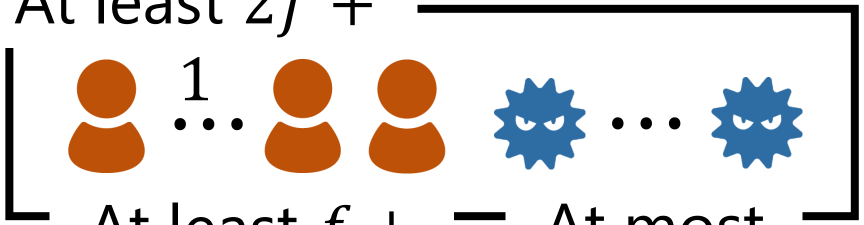
<u>Strategy</u>

- Create a reliable group consisting of enough agents & use only the information from reliable groups

Thanks to many good agents

1. Collect all the reliable group IDs
2. Gather on the node with the group of the smallest ID

At least $2f + 1$



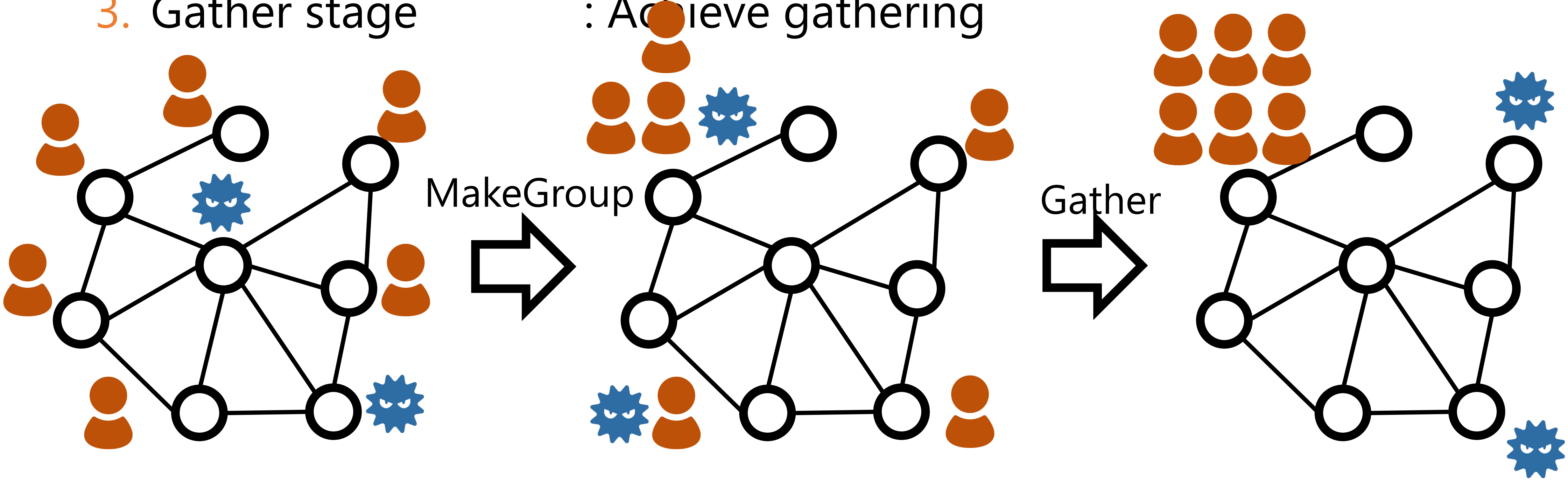
At least $f + 1$ At most f

Convey Info.

At least $f + 1$ agents convey the same Info.
⇒ This group has at least one good agent
⇒ This information can be trusted

Overview

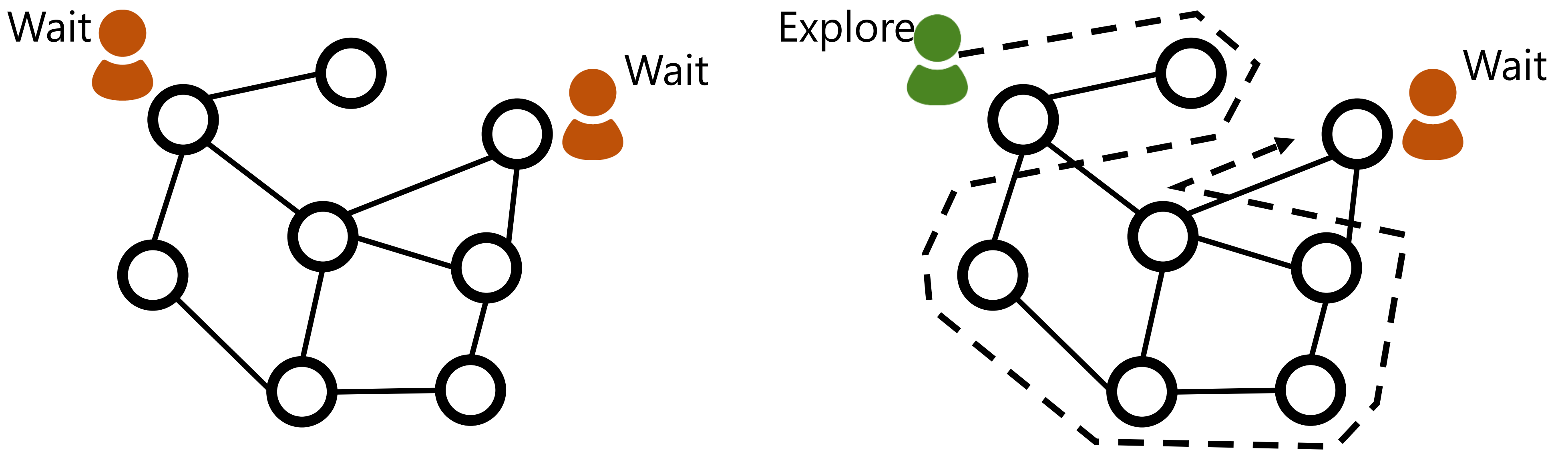
1. CollectID stage : Collect IDs including all good agents
2. MakeGroup stage : Create a reliable group
3. Gather stage : Achieve gathering



CollectID stage

- Goal: Collect IDs including all good agents
- Behavior: Execute the rendezvous algorithm [5]
  - Repeat **Exploring the network** or **Waiting for $X(N)$ rounds** based on ID
    $\Rightarrow$ A good agent can meet all the other good agents
- Time complexity: $O\left(|\Lambda_{good}| \cdot X(N)\right)$

N: upper bound of #nodes
Λ_{good}: The largest ID of good agents

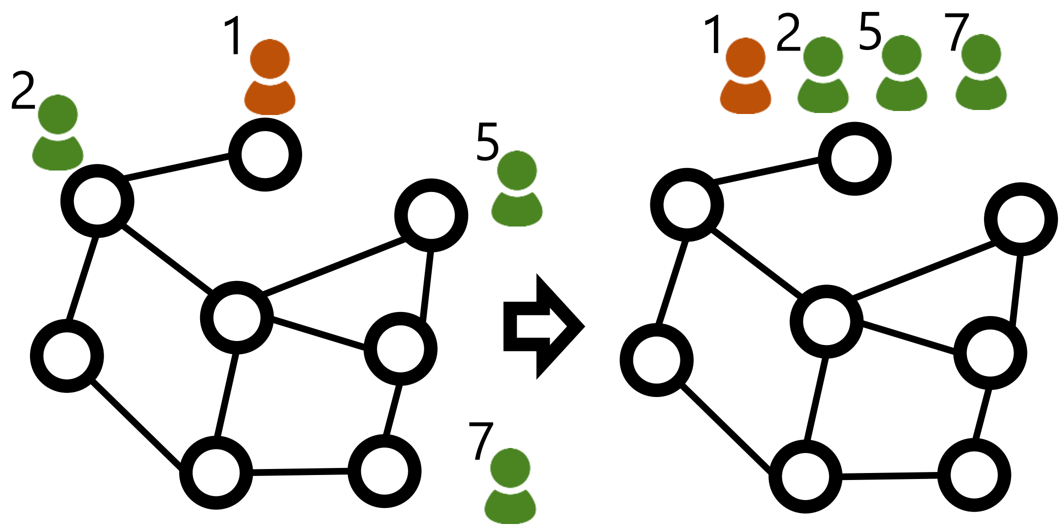


[5] Dessmark, A., et al., Algorithmica 46(1), (2006)

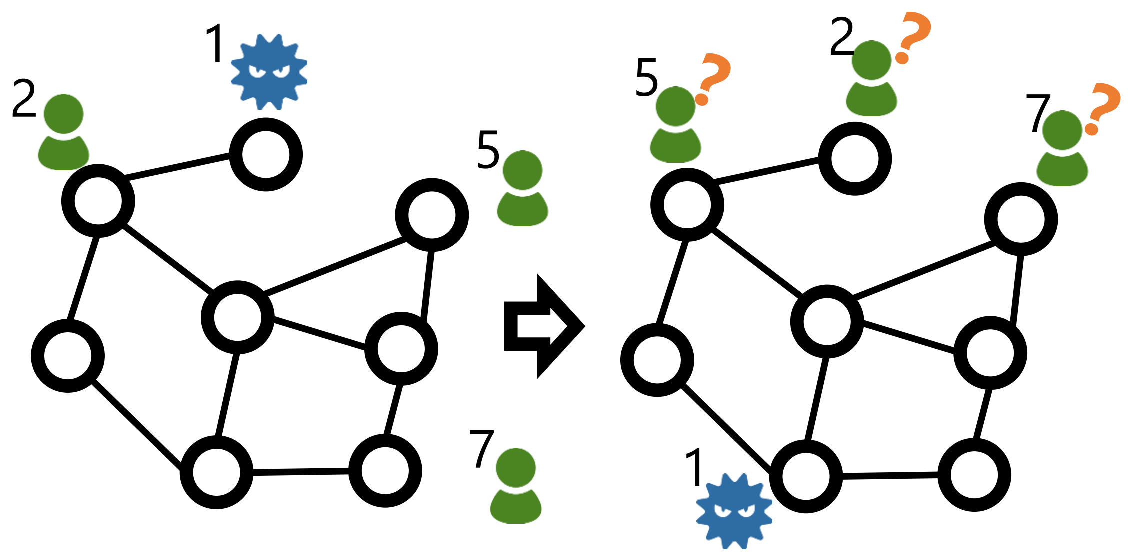
MakeGroup stage: Idea

- Goal: Create a reliable group consisting of enough agents
- Idea: Move to the node with the smallest ID agent
  - : **The smallest ID agent** waits
  - : **Other agents** search for the agent by the exploration algorithm
- If the smallest ID agent is Byzantine, agents may not create a group

Good agent with the smallest ID



Byzantine agent with the smallest ID



MakeGroup stage: Improved idea 1/2

Improved idea: Make the smallest $f+1$ agents wait

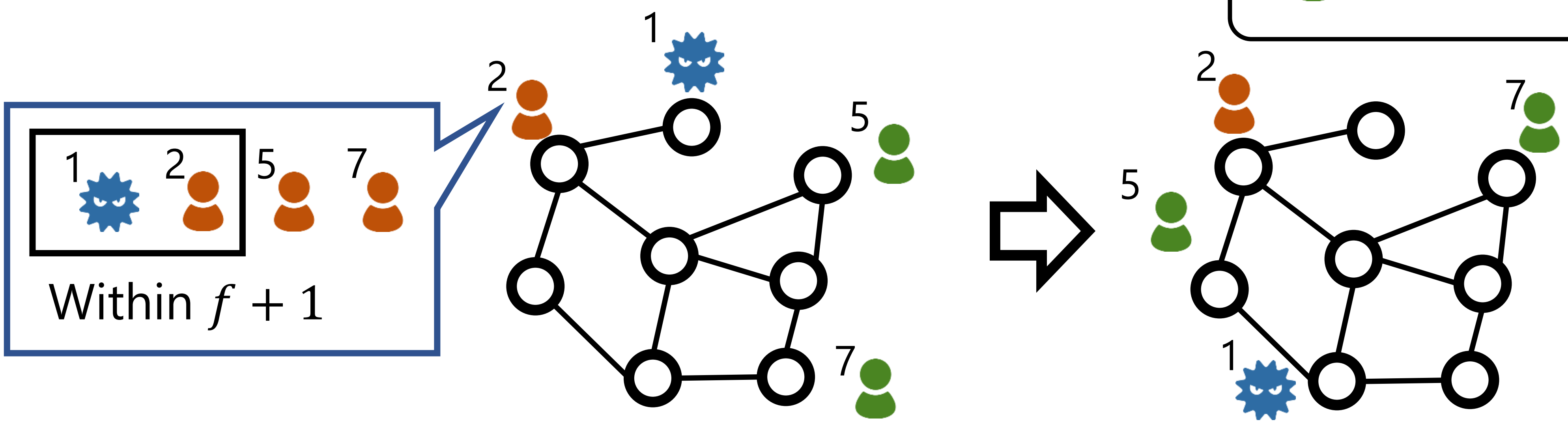
- **Agents with the smallest $f+1$ IDs w**
  ⇒ At least one good agent waits

Good agents don't know the exact f
⇒ Use an estimate value

Agents searching for waiting good agents succeed to search for them

Example ($f=1$)

Searching agents



MakeGroup stage: Improved idea 2/2

Improved idea: Make the smallest $f+1$ agents wait

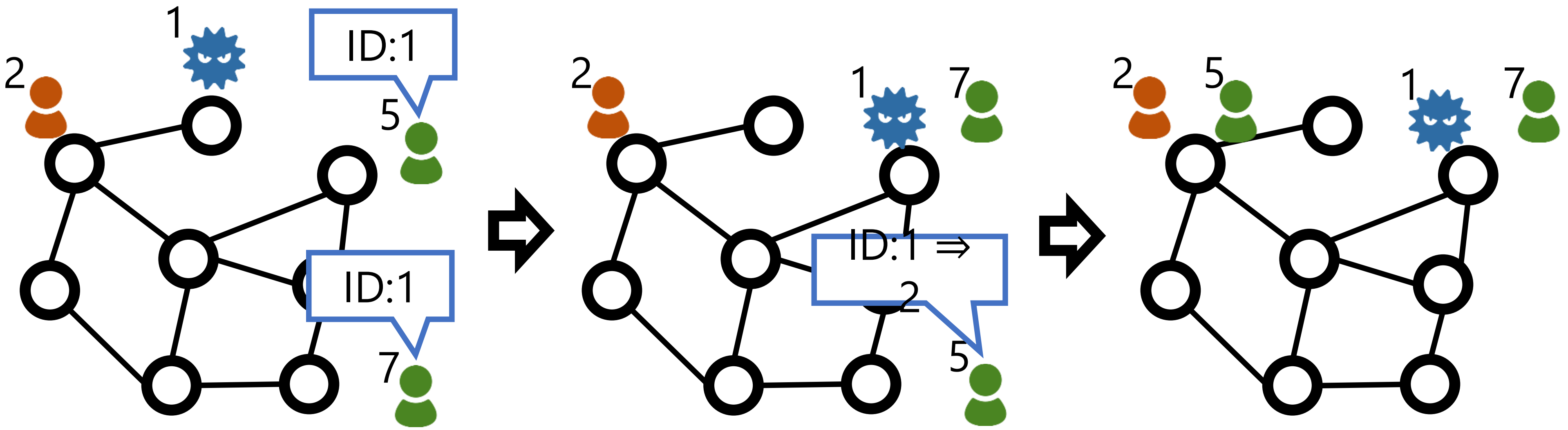
Agents with the smallest $f+1$ IDs wait

Other agents search for the smallest ID agent among collected IDs

- If they fail to find the smallest one, they search for the next one

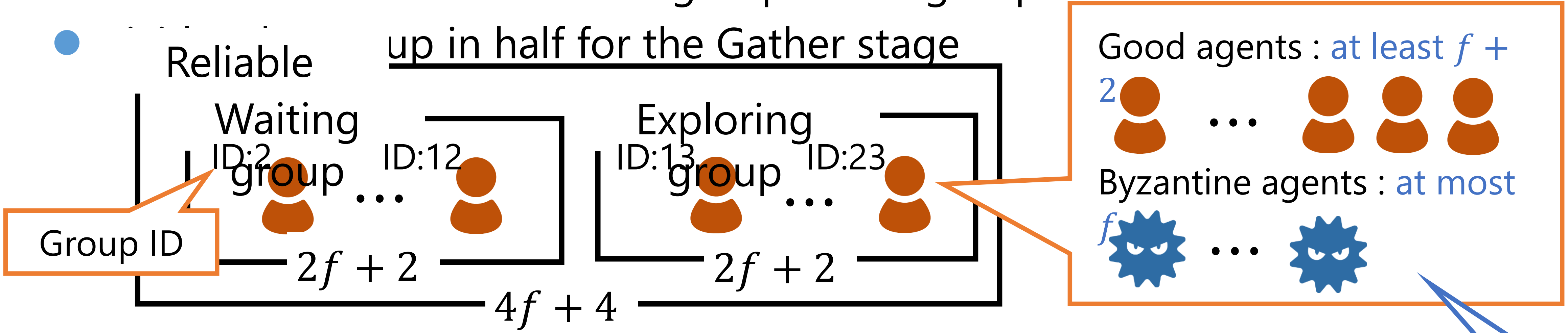
Good agents are divided into at most $f+1$ groups

Example ($f=1$)



MakeGroup stage: Reliable group

- At least one group must have at least $4f + 4$ agents (Reliable group)
  - Good agents are divided into at most $f + 1$ groups
  - Assumption of f : $4f^2 + 9f + 4 = (4f + 4)(f + 1) \leq$ #agents
- When a reliable group is created, the group
  - Decides the smallest ID in the group as the group ID
  - up in half for the Gather stage

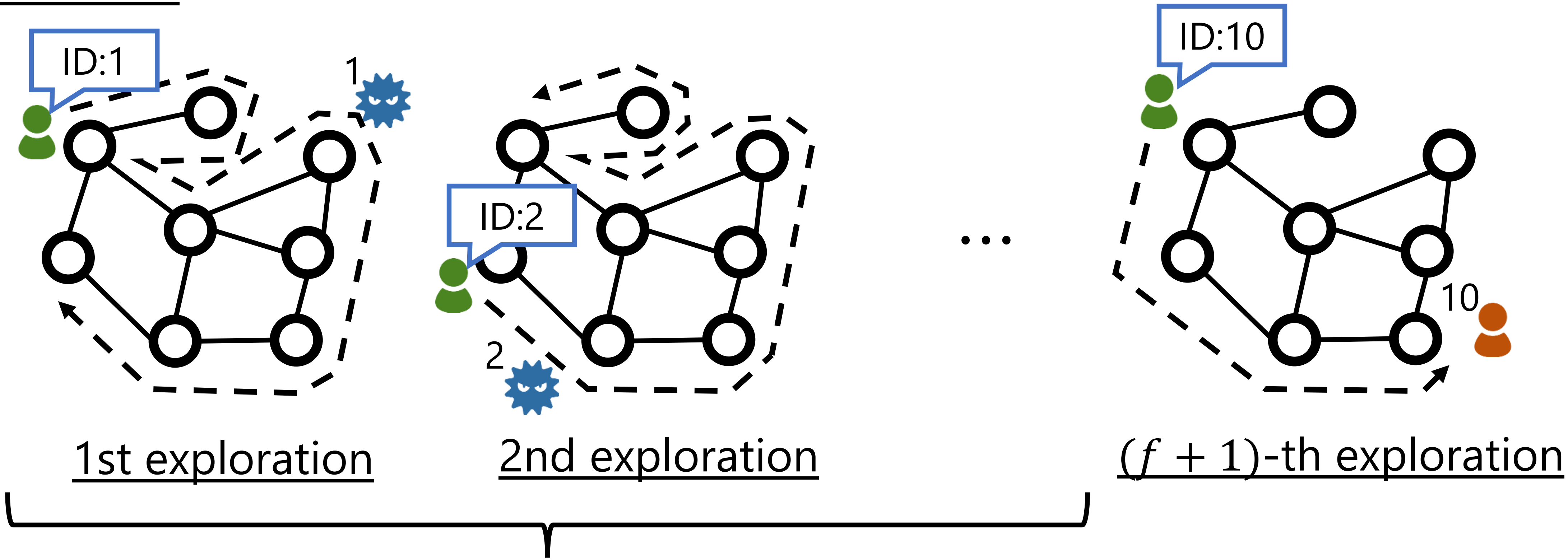


Information provided by a group of at least $2f + 2$ agents is reliable

MakeGroup stage: Time complexity

- Time complexity: $O(f \cdot X(N))$
  - #search iterations: at most $f + 1$

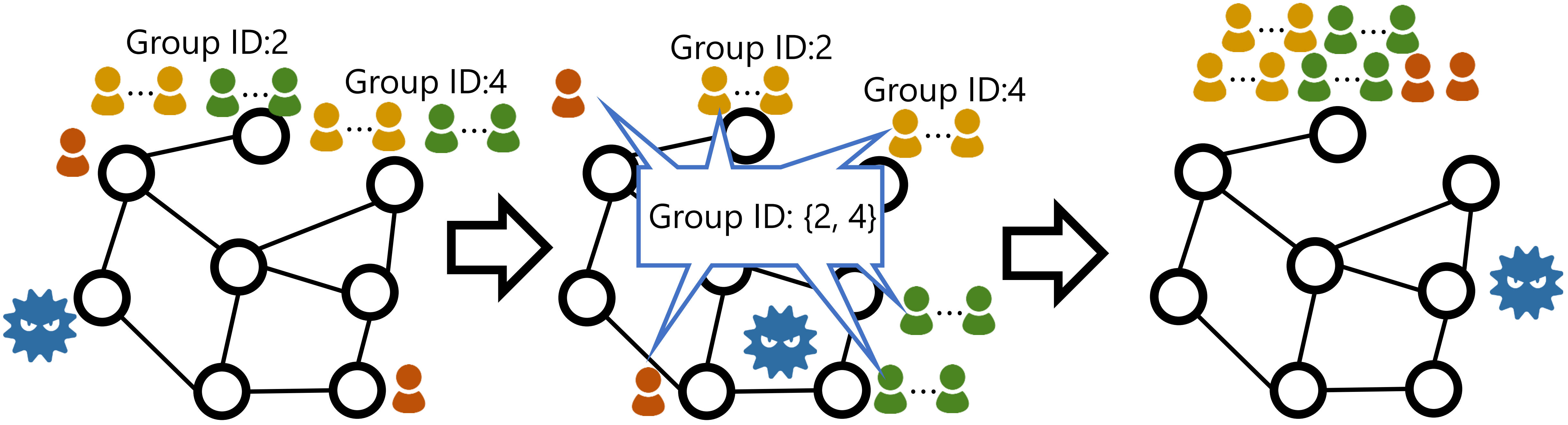
Worst case



An agent fails to search for Byzantine agents f times

Gather stage: Idea

- Goal: Achieve gathering
- Idea:
  1. Collect all reliable group IDs
  2. Gather on the node with the reliable group with the smallest group ID

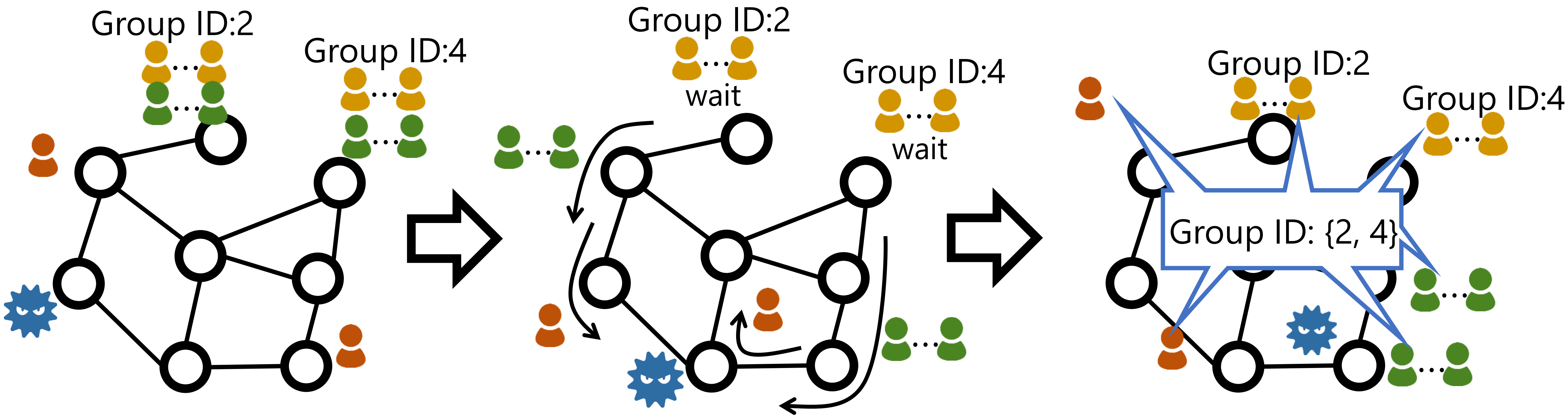


: Reliable group (Waiting) : Reliable group (Exploring) : Other agents

Gather stage: Step 1

Step 1: Collect all reliable group IDs

- Agents in a reliable group: Convey and collect group IDs
  -  (Waiting gr.): Wait for $X(N)$ rounds
  -  (Exploring gr.): Explore the network
- Other agents  : Collect group IDs, while exploring the network



Gather stage: Step 2

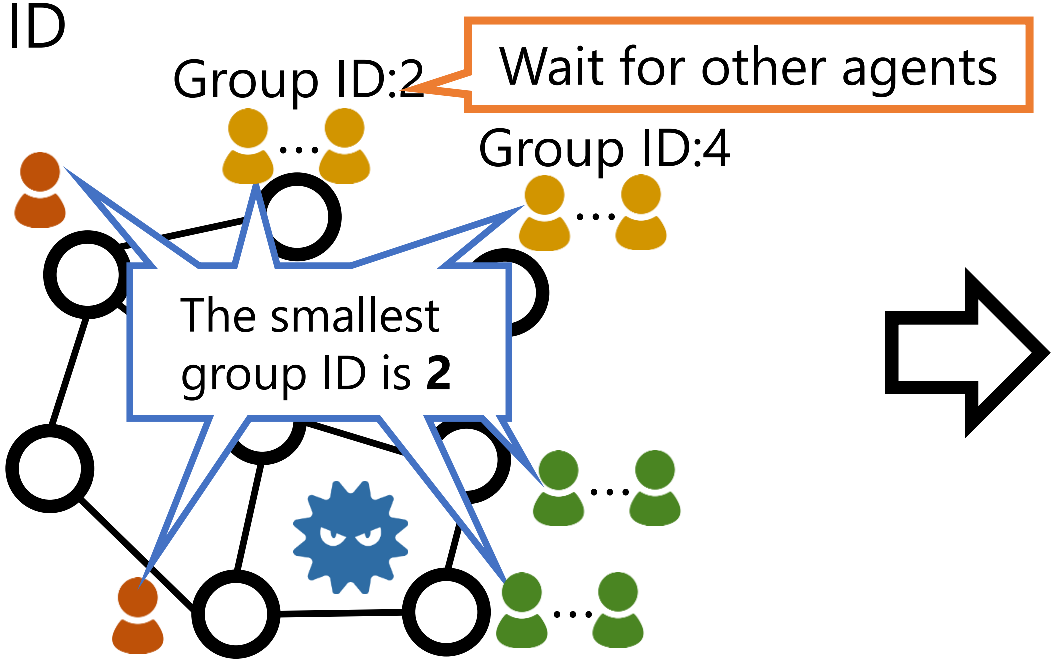
Step 2: Achieve gathering

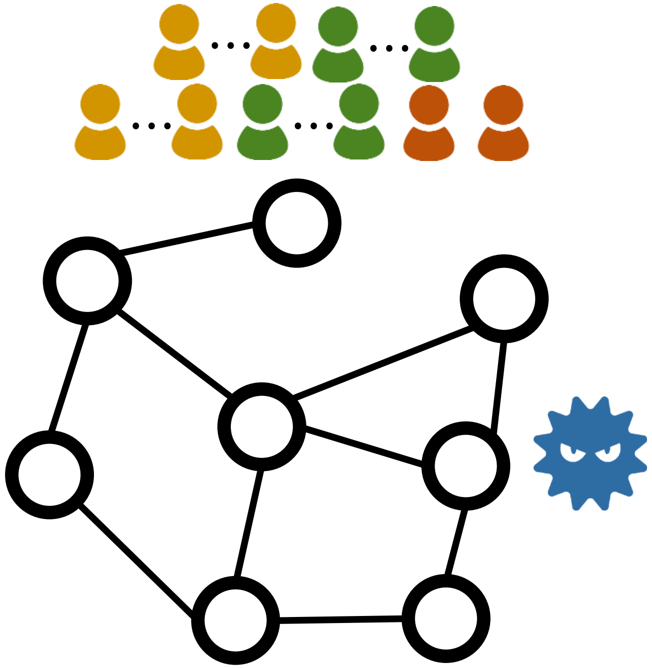
- Waiting group with the smallest group ID:

 : Waits for $X(N)$ rounds

- Other agents:

 : Search for the waiting group with the smallest group ID





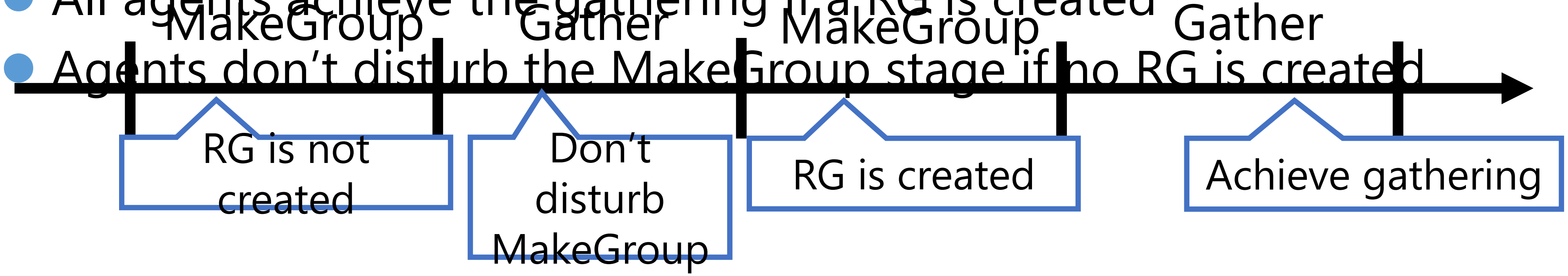
Gather stage: How to enter

<u>Requirement</u>

- All agents enter the Gather stage at the same round after they create a reliable group (RG)

<u>Solution</u>

- Enter the Gather stage every $X(N)$ rounds of the MakeGroup stage
  ⇒ All agents repeatedly enter the Gather stage at the same round
  - All agents achieve the gathering if a RG is created
  - Agents don't disturb the MakeGroup stage if no RG is created



Time complexity with non-simultaneous termination

- CollectID stage : $O\left(\left|\Lambda_{good}\right| \cdot X(N)\right)$
  - $\left|\Lambda_{good}\right|$: The length of the largest ID among good agent IDs
- MakeGroup stage : $O(f \cdot X(N))$
- Gather stage : $2X(N)$ rounds are inserted every $X(N)$ rounds
  - Step 1 : $X(N)$
  - Step 2 : $X(N)$

Total time complexity: $O\left(\left(f + \left|\Lambda_{good}\right|\right) \cdot X(N)\right)$

f: #Byzantine agents
$\left|\Lambda_{good}\right|$: The length of the largest ID among good agents
$X(n)$: Time required to visit all n nodes

Conclusion

Contribution

Reduce time complexity in weakly Byzantine environments by relaxing #Byzantine agents

Future work

Design an algorithm that works even in the presence of startup delay

	Input	Startup delay	Condition of #Byzantine agents	Simultaneous termination	Time complexity
[1]	n	Presence	$f+1 \leq k$	Possible	$O\left(n^4 \cdot \lvert\Lambda_{good}\rvert \cdot X(n)\right)$
Algo. 1	N	Absence	$4f^2+9f+4 \leq k$	Impossible	$O\left(\left(f+\lvert\Lambda_{good}\rvert\right) \cdot X(N)\right)$
Algo. 2	N	Absence	$4f^2+9f+4 \leq k$	Possible	$O\left(\left(f+\lvert\Lambda_{all}\rvert\right) \cdot X(N)\right)$