Quantum Distributed Computing: Recent Results

François Le Gall

Nagoya University

Simons Institute
27 February 2020

Does quantum help for distributed computing?

Does quantum help for distributed computing?

Positive answers known in some models:

- ✓ anonymous networks: quantum leader election [Tani et al. 2007]
- ✓ faulty networks: quantum Byzantine agreement [Ben-Or, Hassidim 2005]
- ✓ quantum multiparty communication complexity

Does quantum help for distributed computing?

Positive answers known in some models:

- ✓ anonymous networks: quantum leader election [Tani et al. 2007]
- ✓ faulty networks: quantum Byzantine agreement [Ben-Or, Hassidim 2005]
- ✓ quantum multiparty communication complexity

Two other models are very popular recently in the classical distributed computing community:

CONGEST model

(limited bandwidth)

LOCAL model

(unlimited bandwidth)

Does quantum help for distributed computing?

Positive answers known in some models:

- ✓ anonymous networks: quantum leader election [Tani et al. 2007]
- ✓ faulty networks: quantum Byzantine agreement [Ben-Or, Hassidim 2005]
- ✓ quantum multiparty communication complexity

Two other models are very popular recently in the classical distributed computing community:

CONGEST model

(limited bandwidth)

LOCAL model

(unlimited bandwidth)

negative results: show impossibility of quantum distributed computing faster than classical distributed computing for many important problems (shortest paths, minimum spanning tree,…)
[Gavoille, Kosowski, Markiewicz 2009] [Elkin et al. 2014]

Does quantum help for distributed computing?

Positive answers known in some models:

- ✓ anonymous networks: quantum leader election [Tani et al. 2007]
- ✓ faulty networks: quantum Byzantine agreement [Ben-Or, Hassidim 2005]
- ✓ quantum multiparty communication complexity

Two other models are very popular recently in the classical distributed computing community:

CONGEST model

(limited bandwidth)

Quantum can be useful for some problems
[LG, Magniez 2018] [Izumi, LG 2019] [Izumi, LG, Magniez 2020]

negative results: show impossibility of quantum distributed computing faster than classical distributed computing for many important problems (shortest paths, minimum spanning tree,…)
[Gavoille, Kosowski, Markiewicz 2009] [Elkin et al. 2014]

LOCAL model

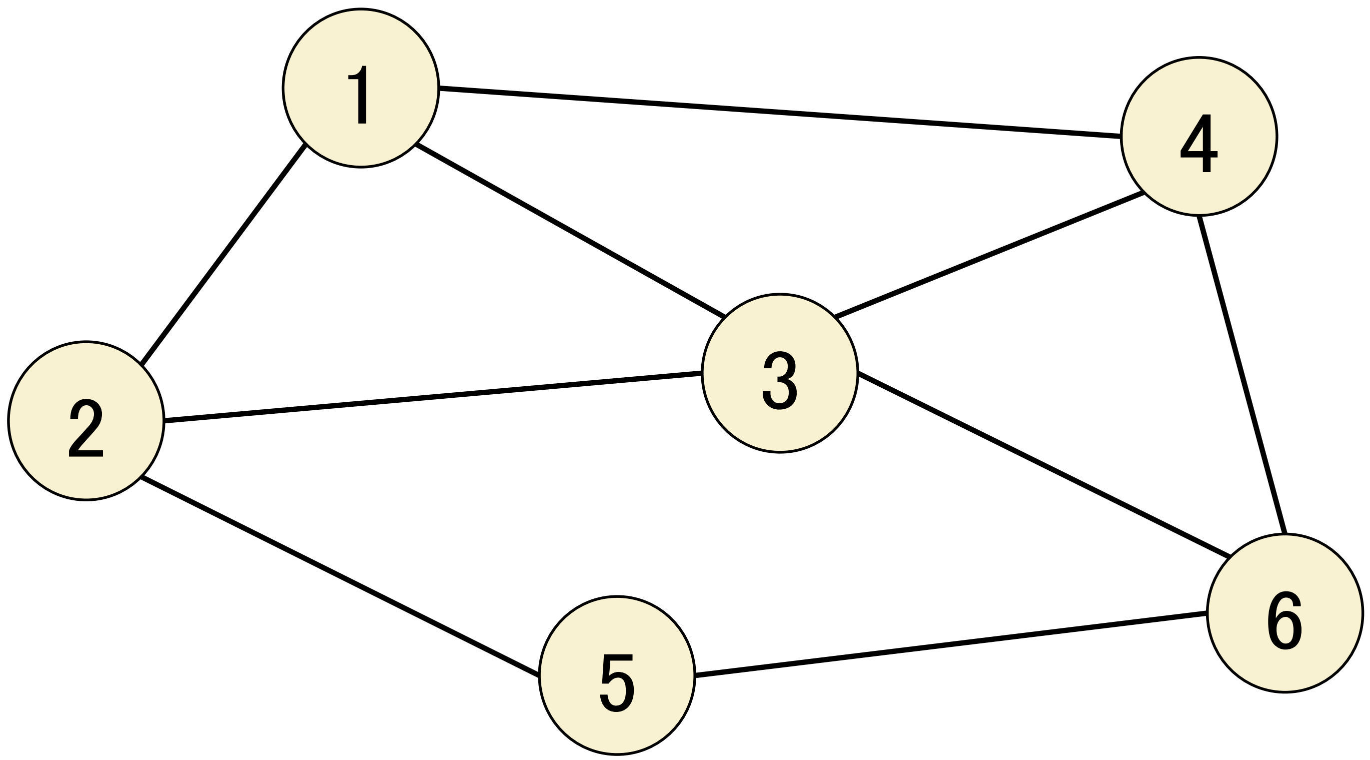
(unlimited bandwidth)

Quantum can be useful for some problems
[LG, Nishimura, Rosmanis 2019]

Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)

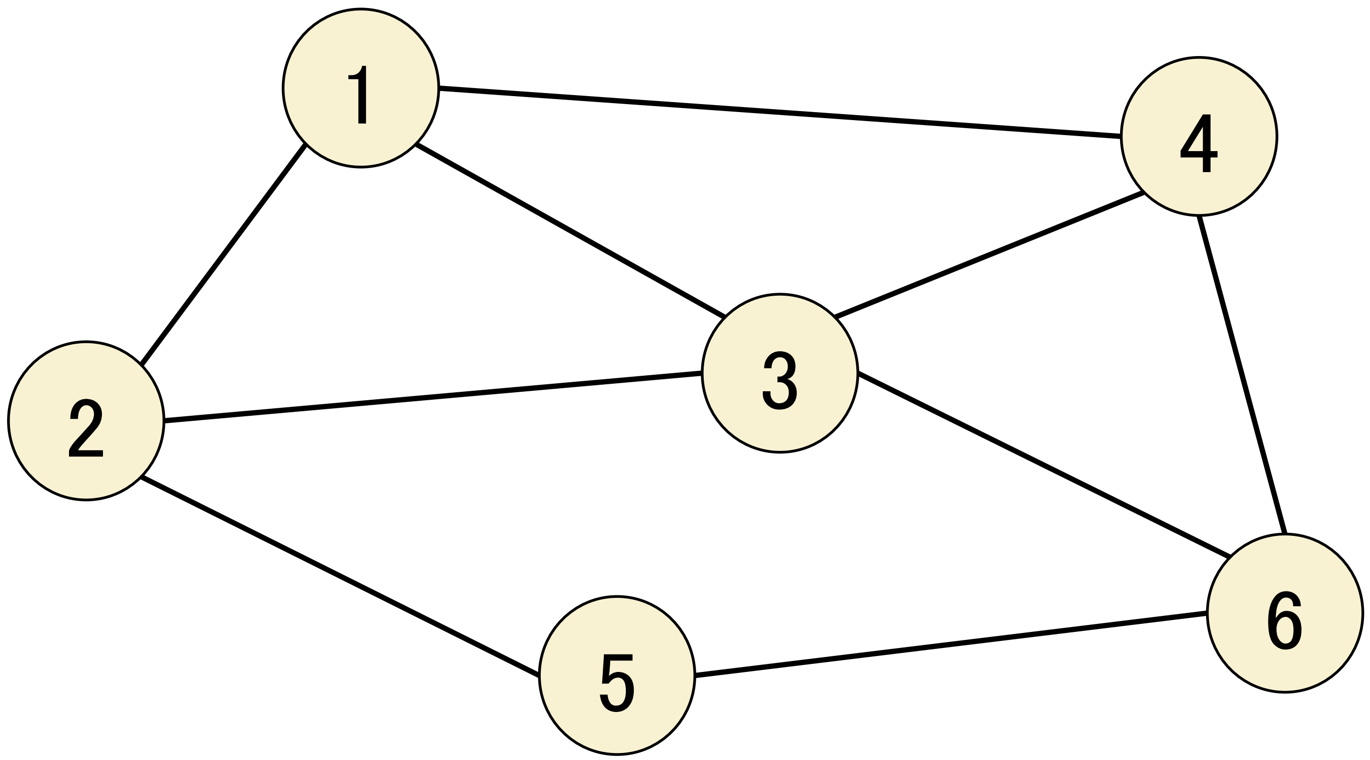


Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)

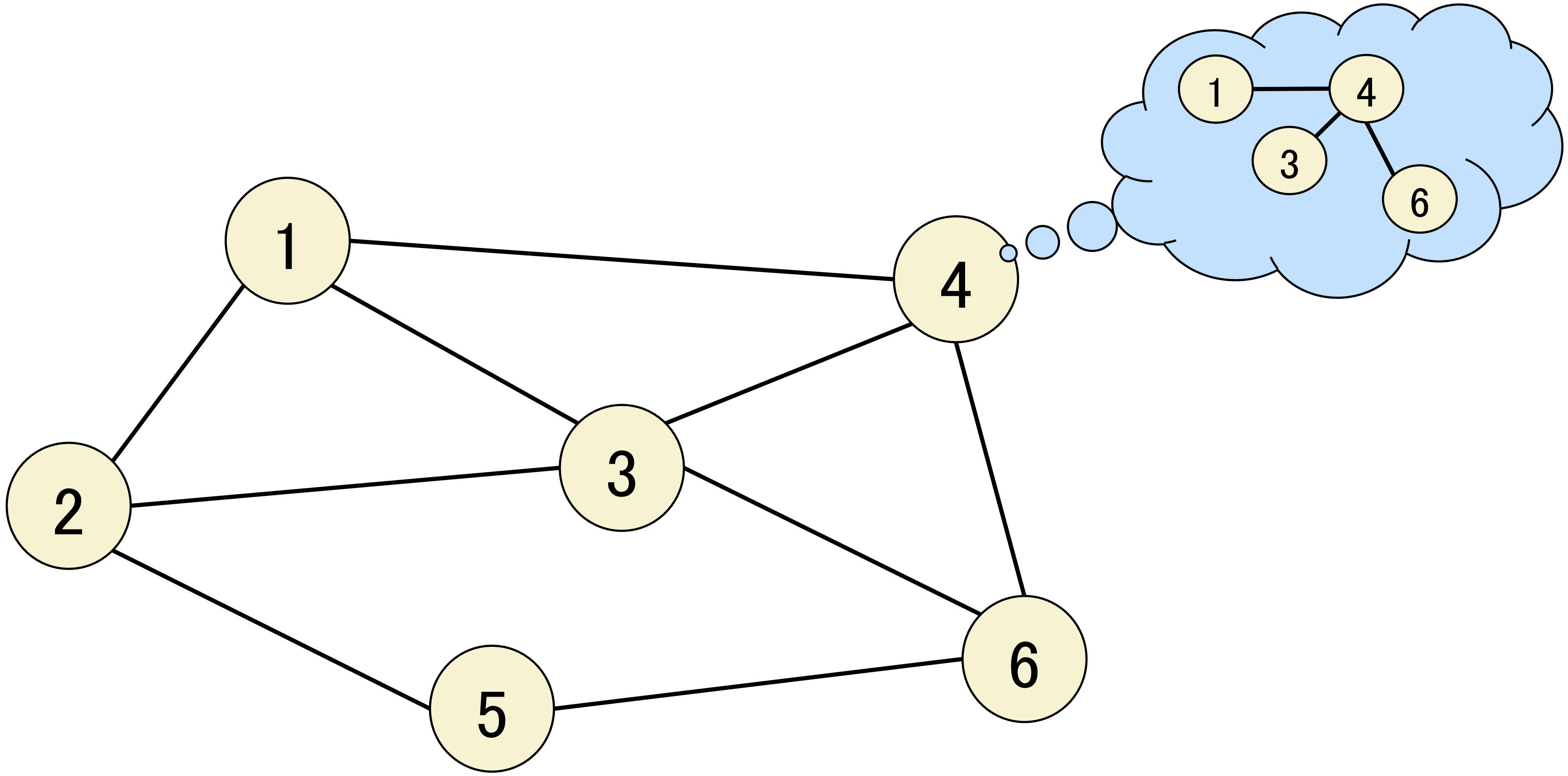
✓ each node initially knows only the identifiers of all its neighbors (and knows n)



Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

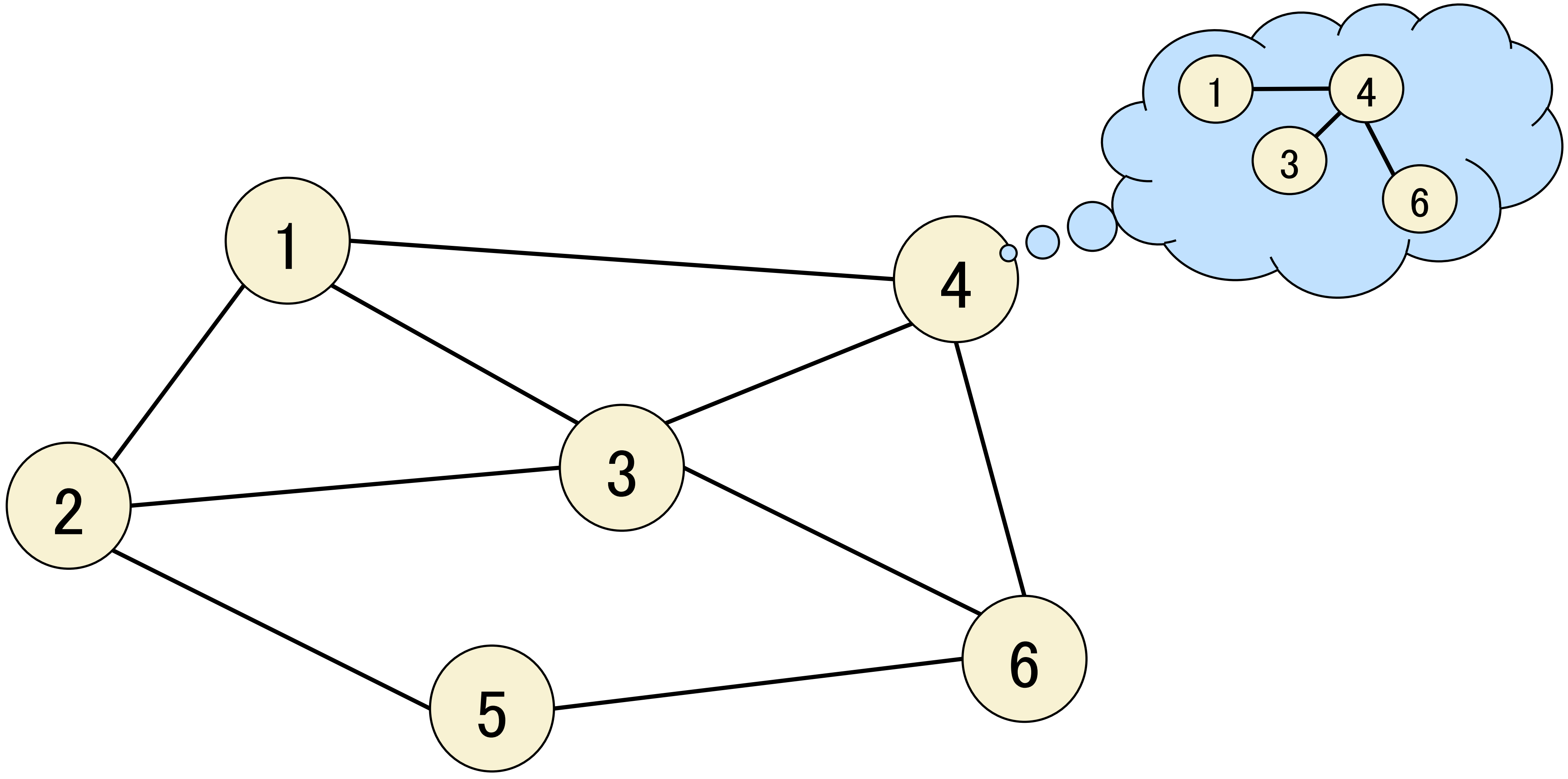
- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)



Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

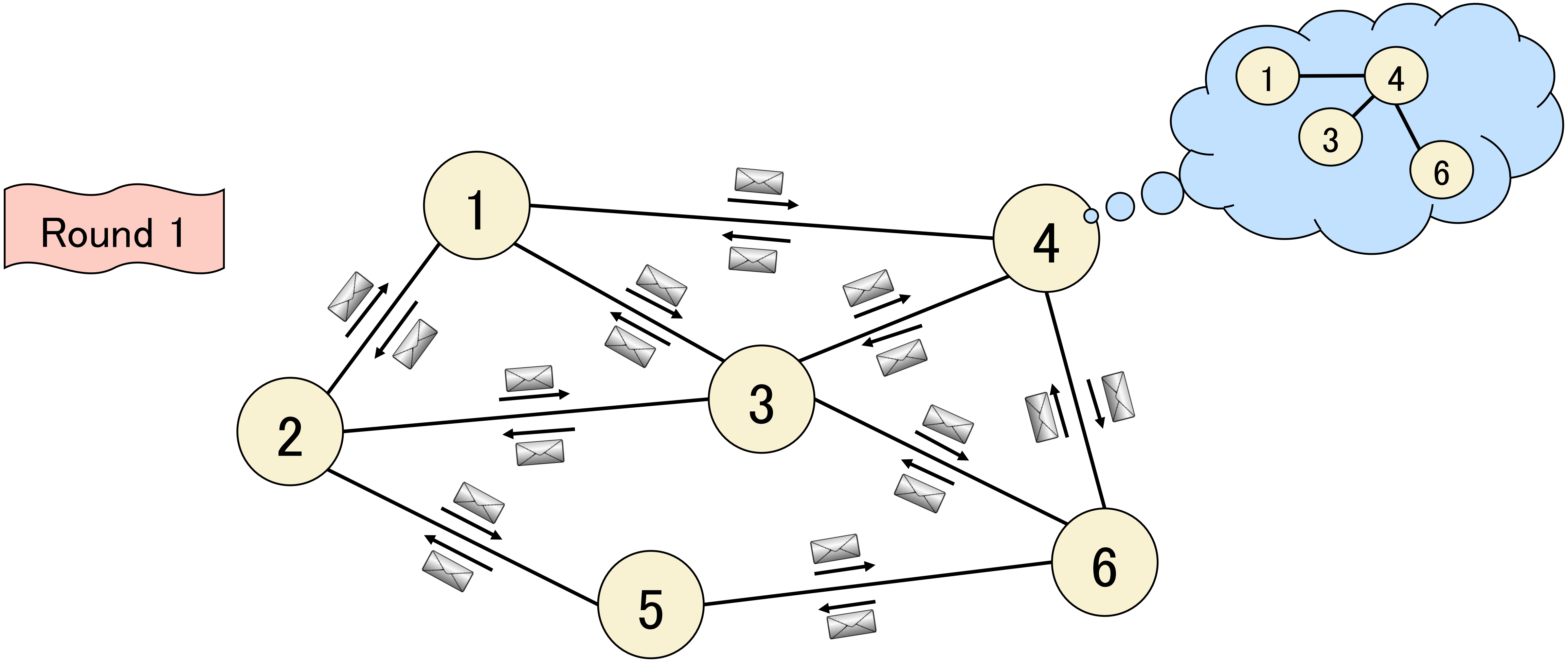
- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)



Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

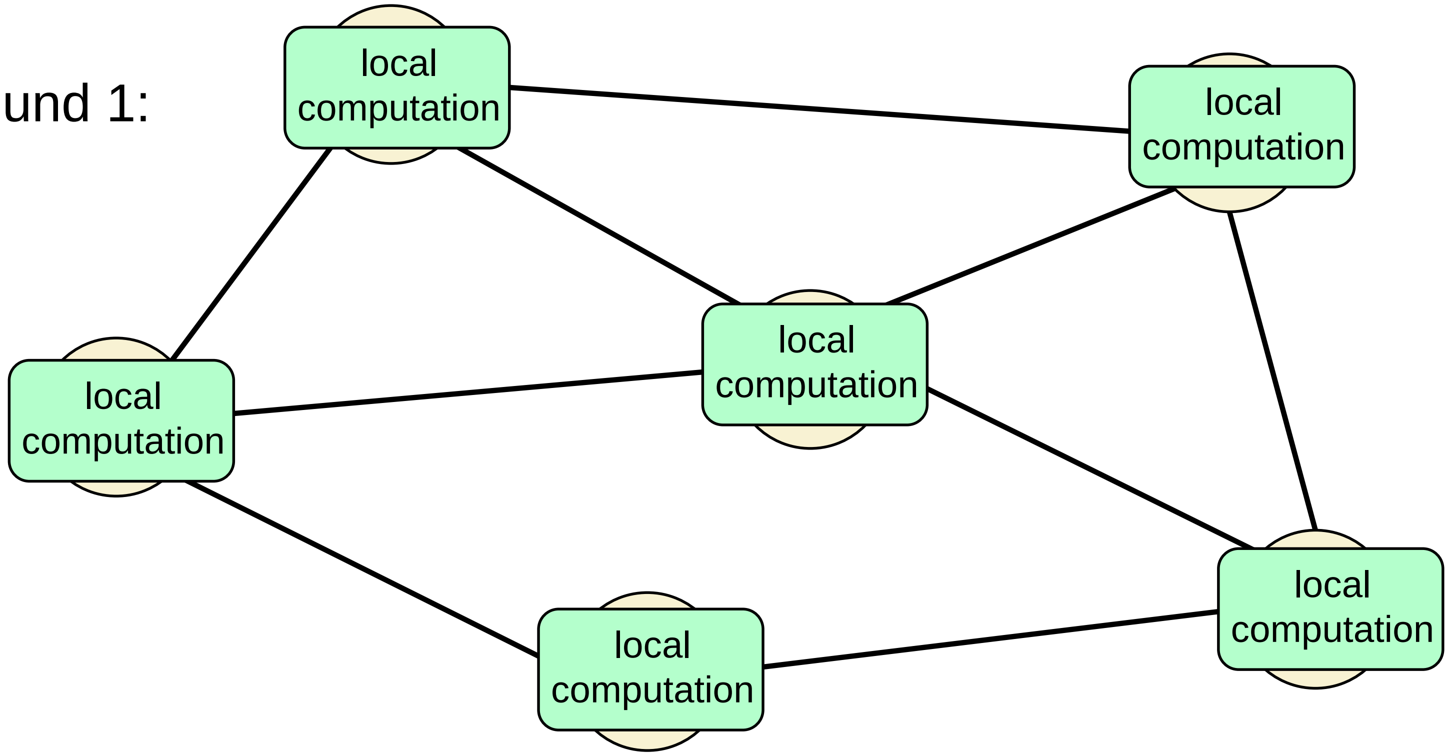


Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

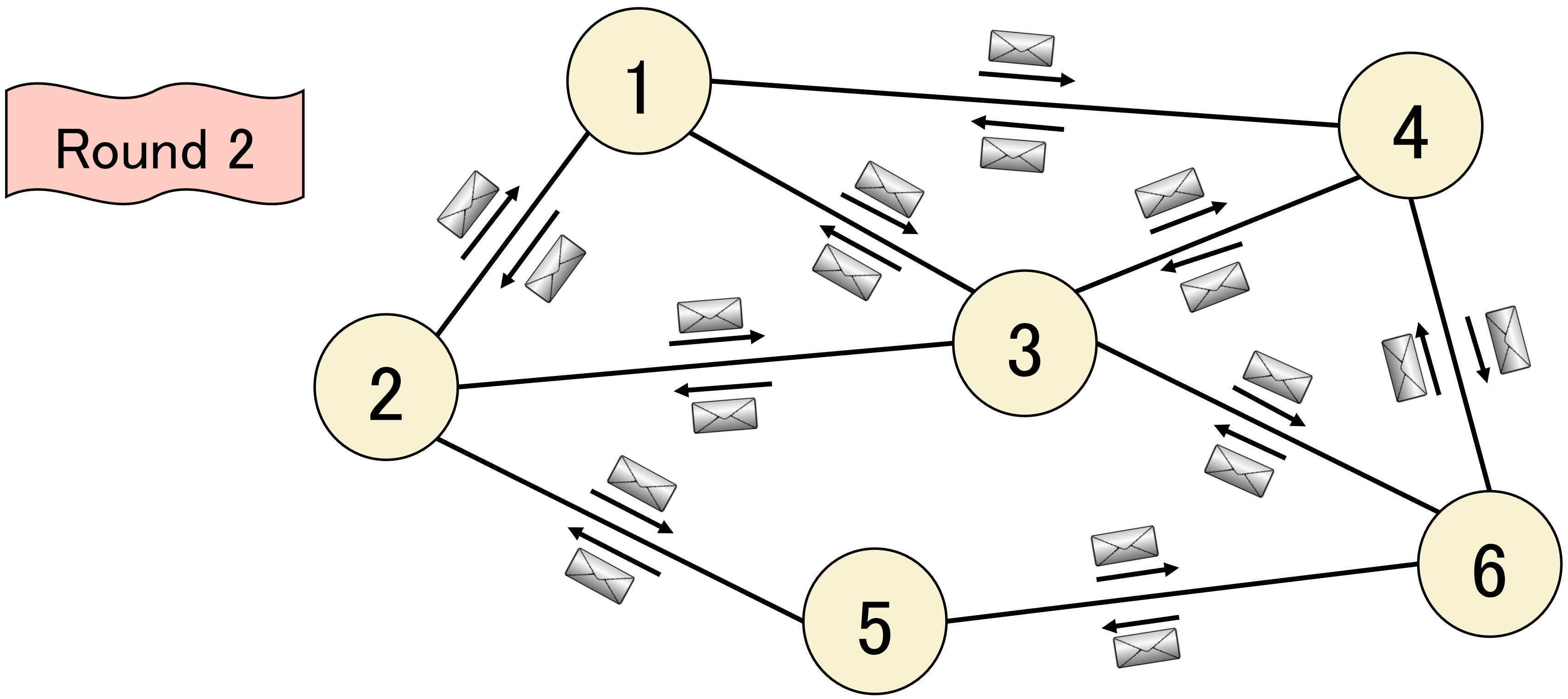
at the end of Round 1:



Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

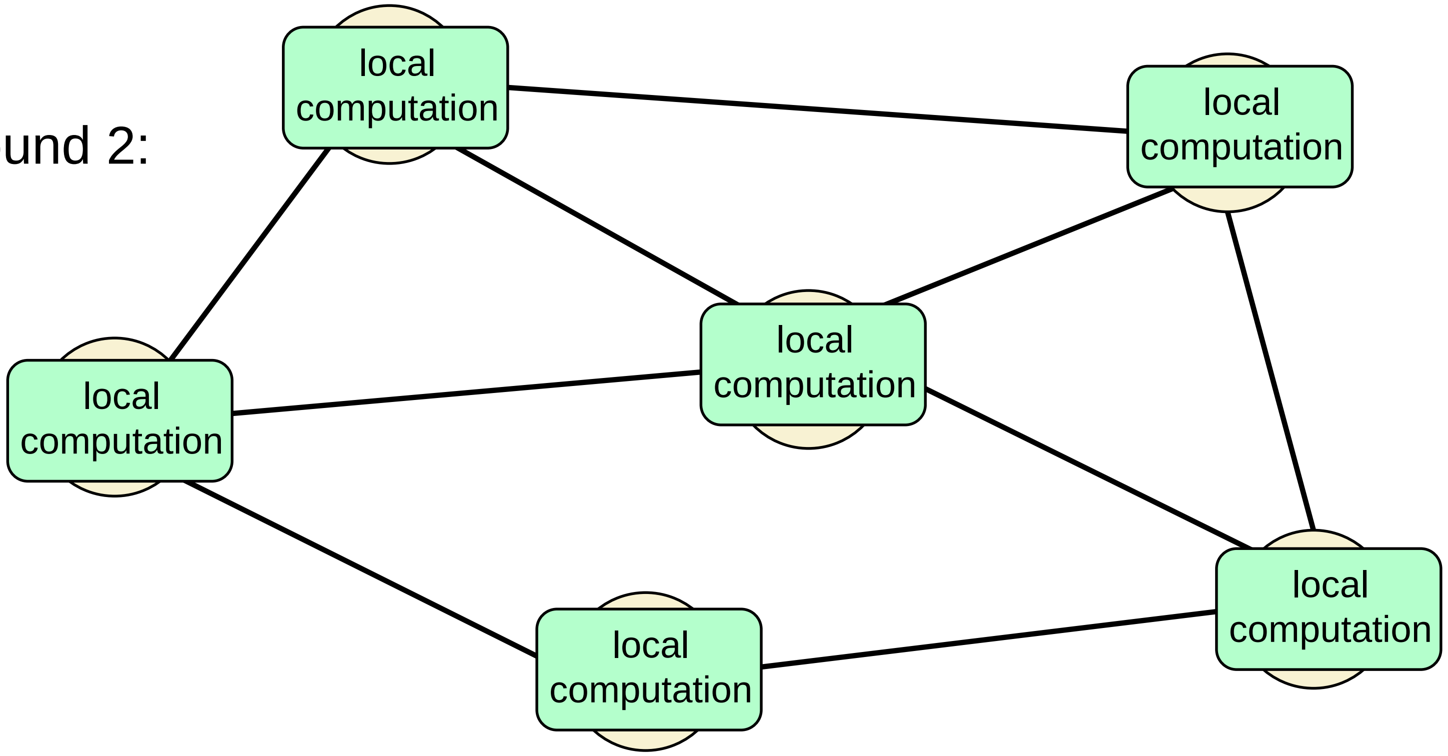


Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

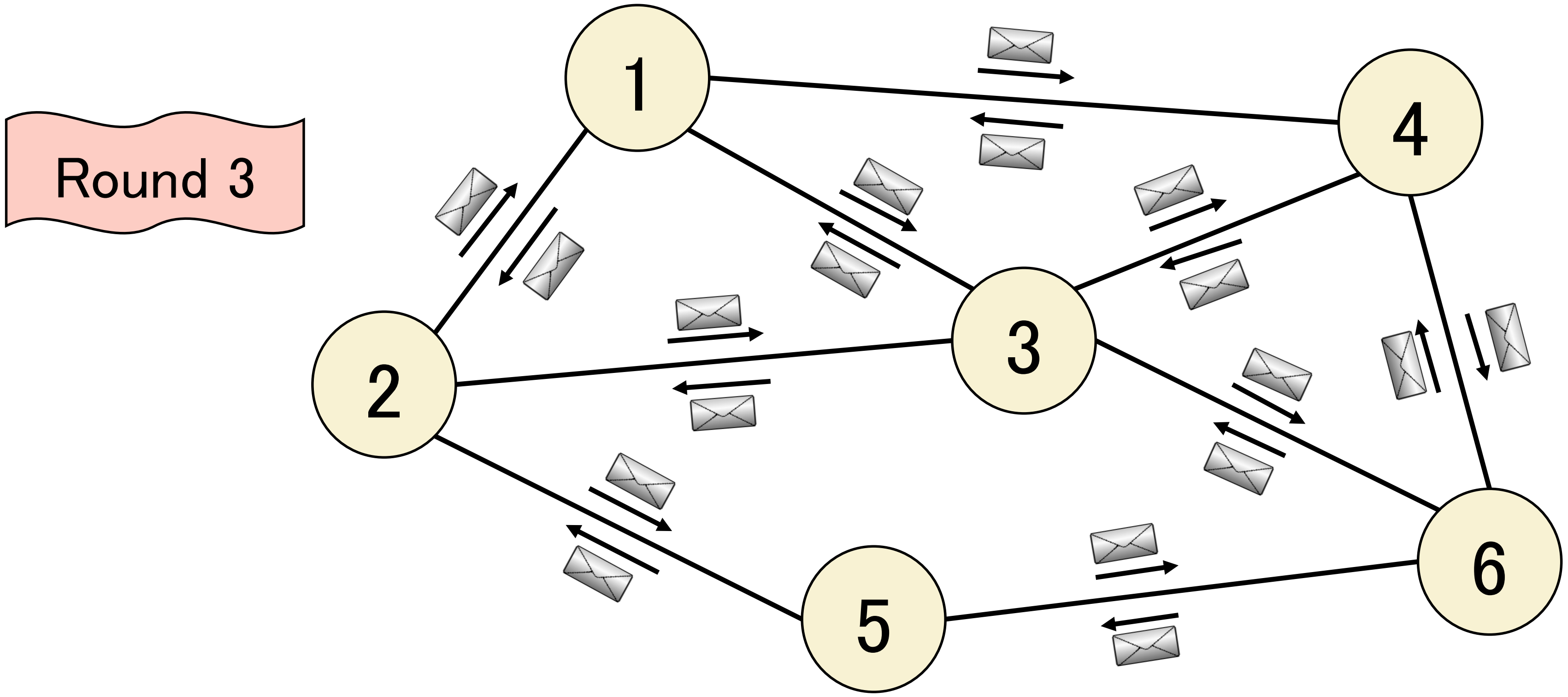
at the end of Round 2:



Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

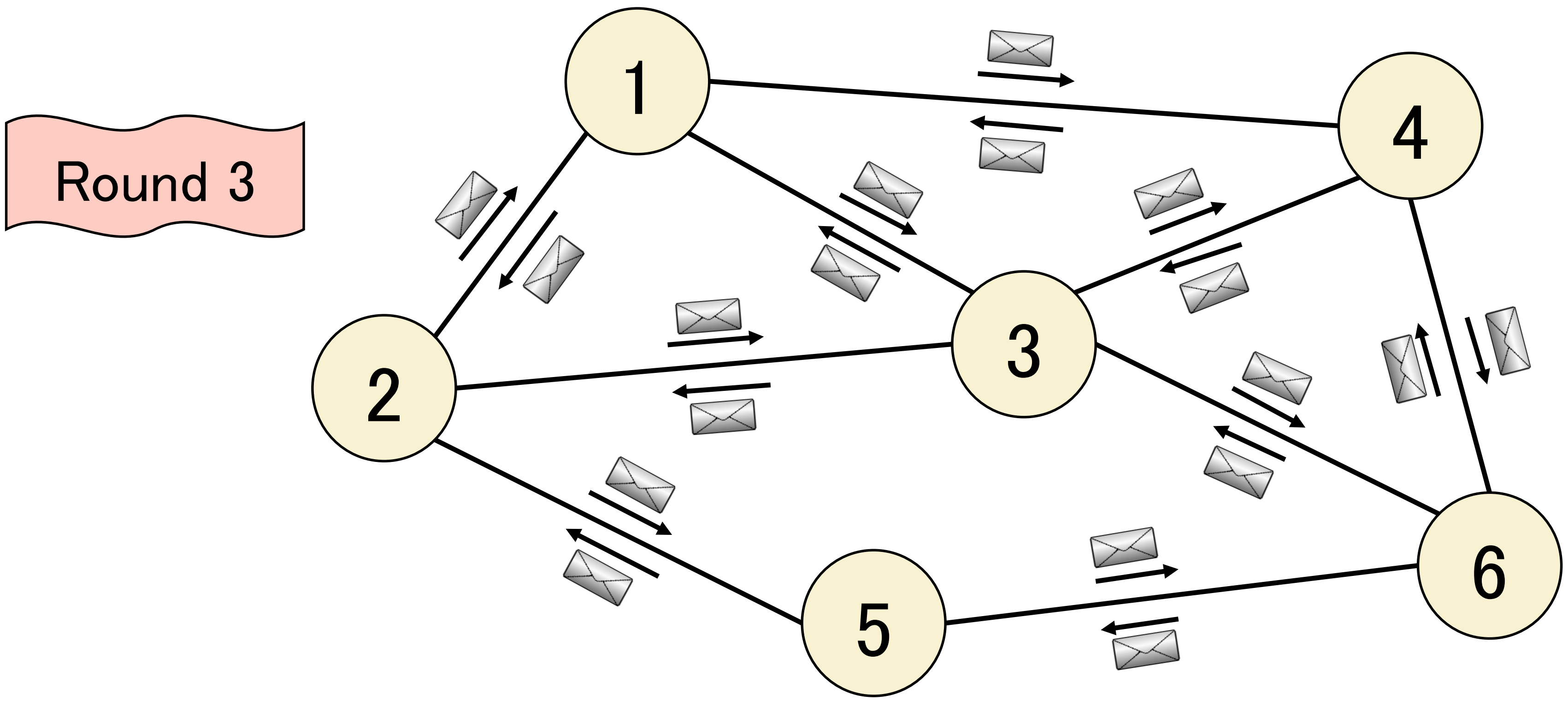


Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

Complexity: the number of rounds used



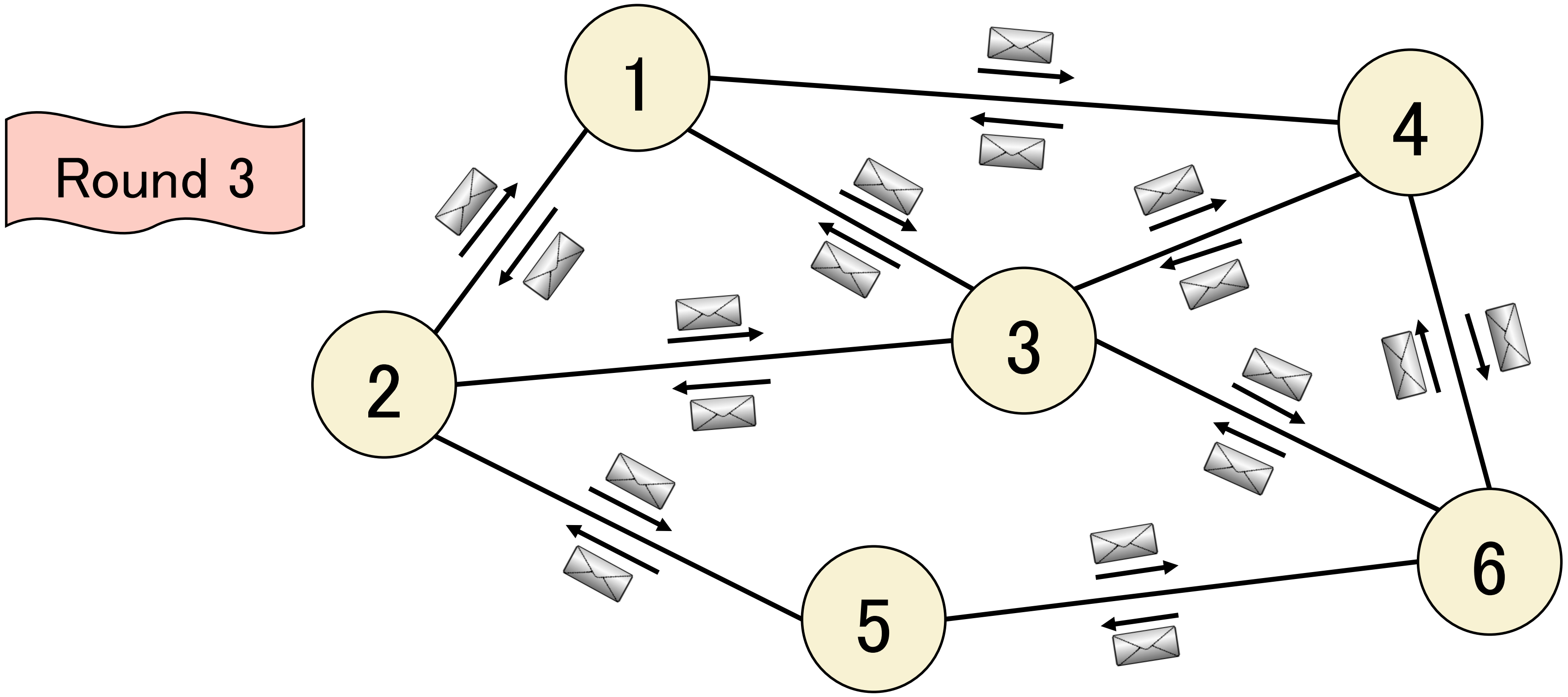
Classical Distributed Computing: CONGEST and LOCAL

Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

Complexity: the number of rounds used

what size?



Classical Distributed Computing: CONGEST and LOCAL

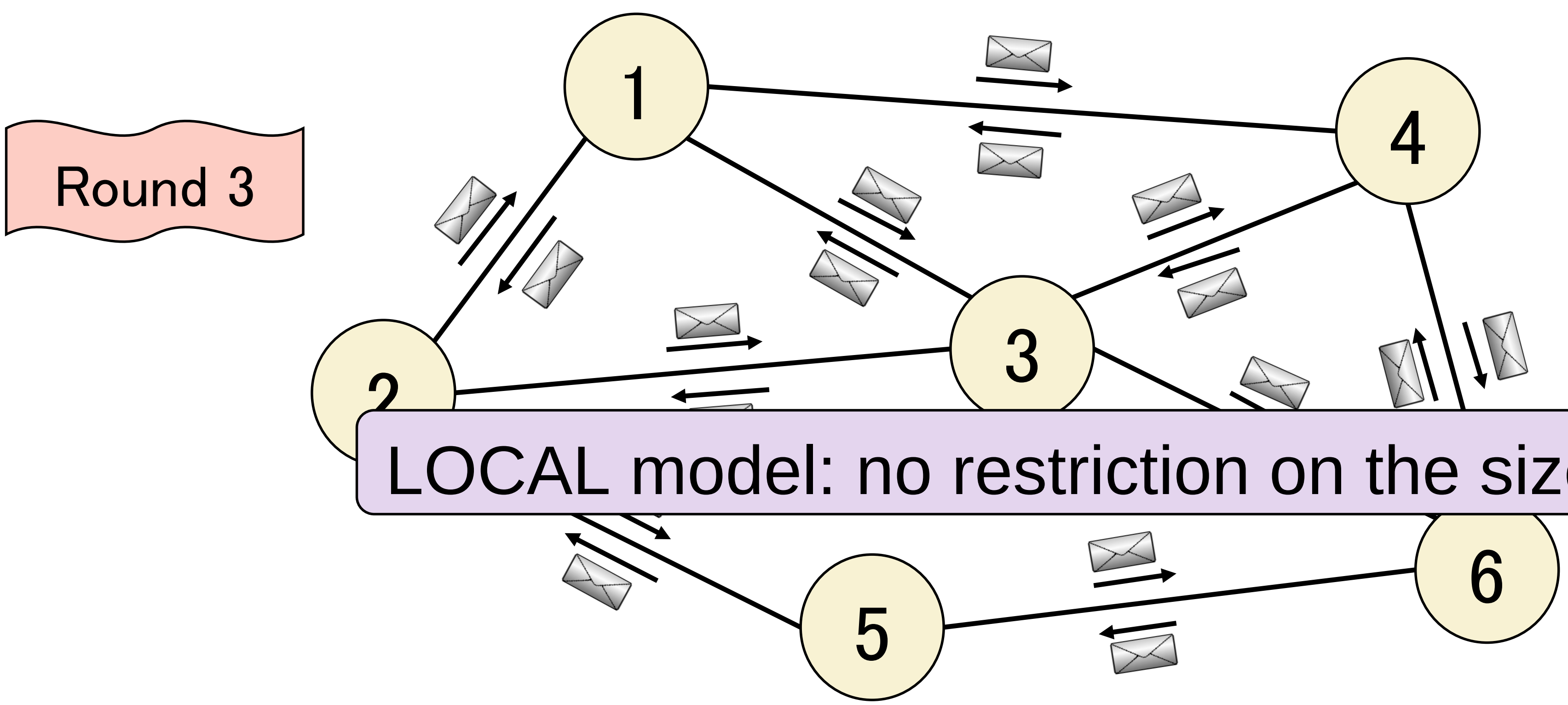
Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

Complexity: the number of rounds used

what size?

CONGEST model: only O(log n) bits per message



LOCAL model: no restriction on the size of each message

Classical Distributed Computing: CONGEST and LOCAL

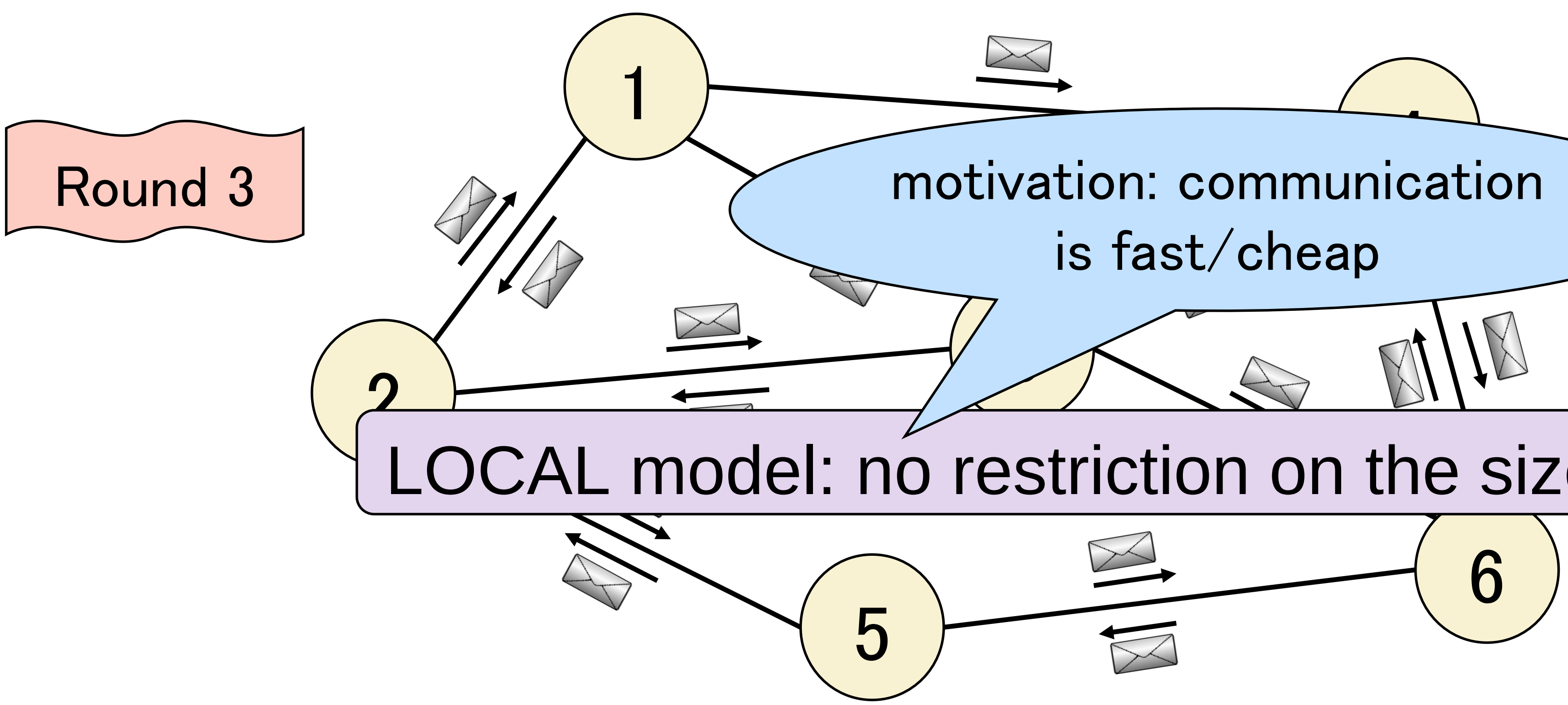
Basic setting: non-faulty, non-anonymous, synchronous

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node initially knows only the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message through each edge per round (in each direction)

Complexity: the number of rounds used

what size?

CONGEST model: only O(log n) bits per message



motivation: communication
is fast/cheap

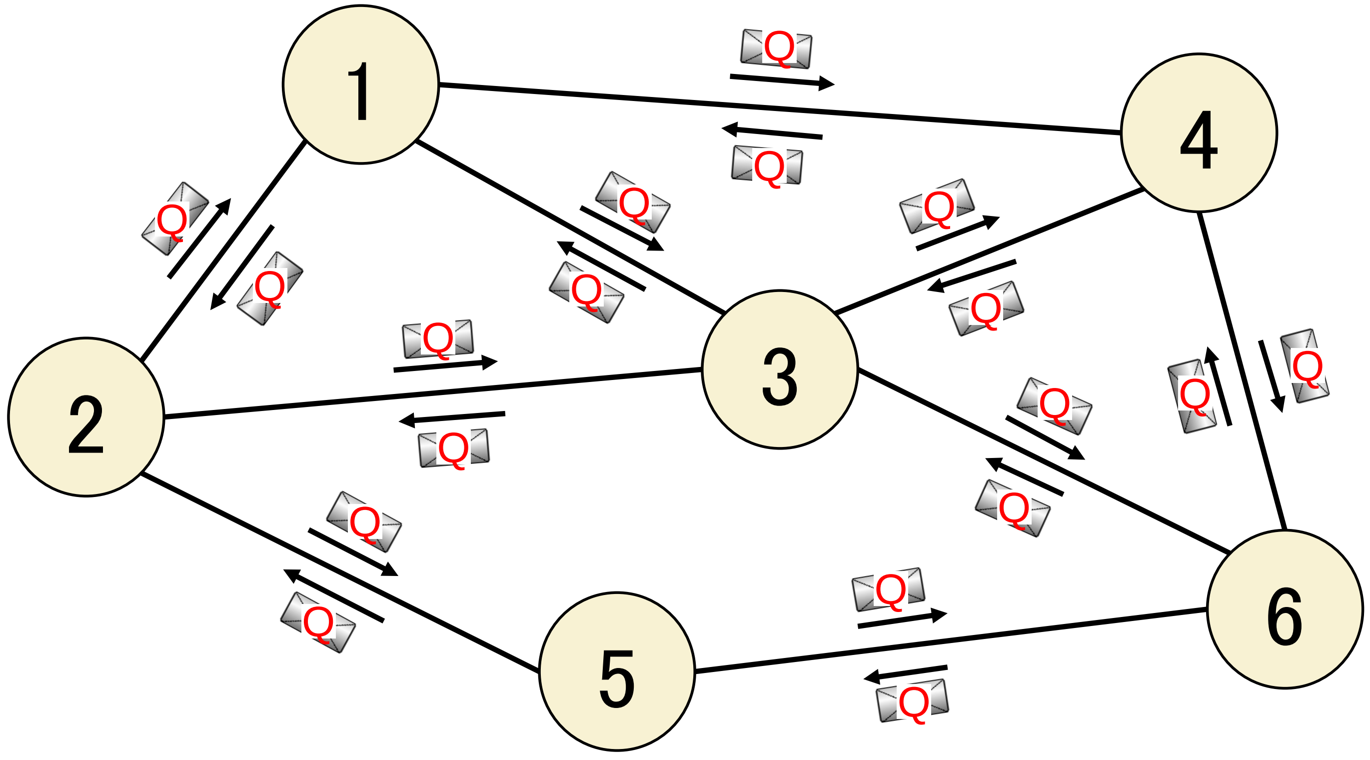
LOCAL model: no restriction on the size of each message

Quantum Distributed Computing: CONGEST and LOCAL

Quantum distributed computing

Now qubits can be sent instead of bits

(no prior entanglement between nodes)



Quantum Distributed Computing: CONGEST and LOCAL

Quantum distributed computing

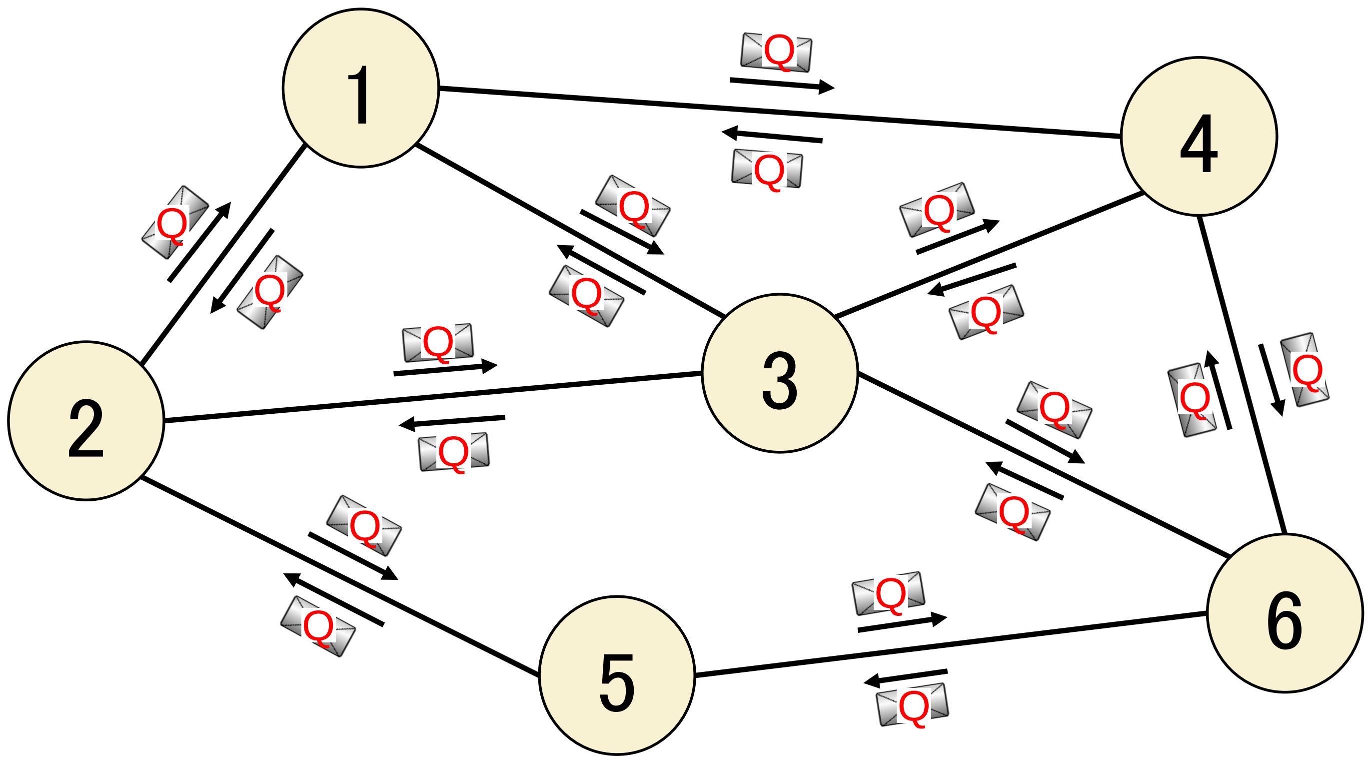
Now qubits can be sent instead of bits

(no prior entanglement between nodes)

more formally:

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node only knows the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message of qubits through each edge per round (in each direction)
- ✓ each node is a quantum processor

Complexity: the number of rounds needed for the computation



Quantum Distributed Computing: CONGEST and LOCAL

Quantum distributed computing

Now qubits can be sent instead of bits

(no prior entanglement between nodes)

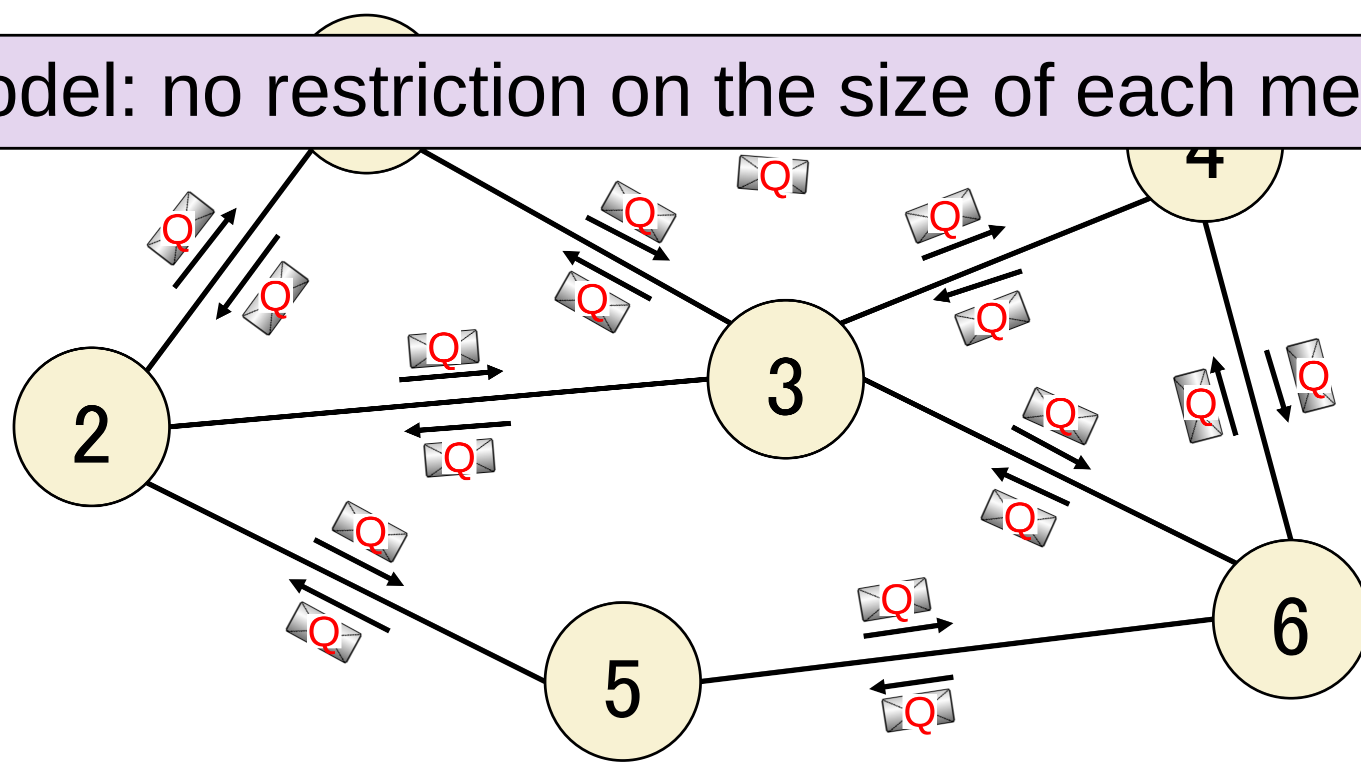
more formally:

- ✓ network G=(V,E) of n nodes (all nodes have distinct identifiers)
- ✓ each node only knows the identifiers of all its neighbors (and knows n)
- ✓ synchronous communication between adjacent nodes:
  one message of qubits through each edge per round (in each direction)
- ✓ each node is a quantum processor

CONGEST model: only O(log n) qubits per message

Complexity: the number of rounds needed for the computation

LOCAL model: no restriction on the size of each message



Quantum Advantage in the CONGEST model

Quantum distributed computing

Now qubits can be sent instead of bits

(no prior entanglement between nodes)

n: number of nodes of the network

CONGEST model: only O(log n) qubits per message

[LG, Magniez 18] ⇨ The diameter of the network can be computed in $\Theta(\sqrt{n})$ rounds in the quantum CONGEST model but requires $\Theta(n)$ rounds in the classical CONGEST model (when the diameter is constant)

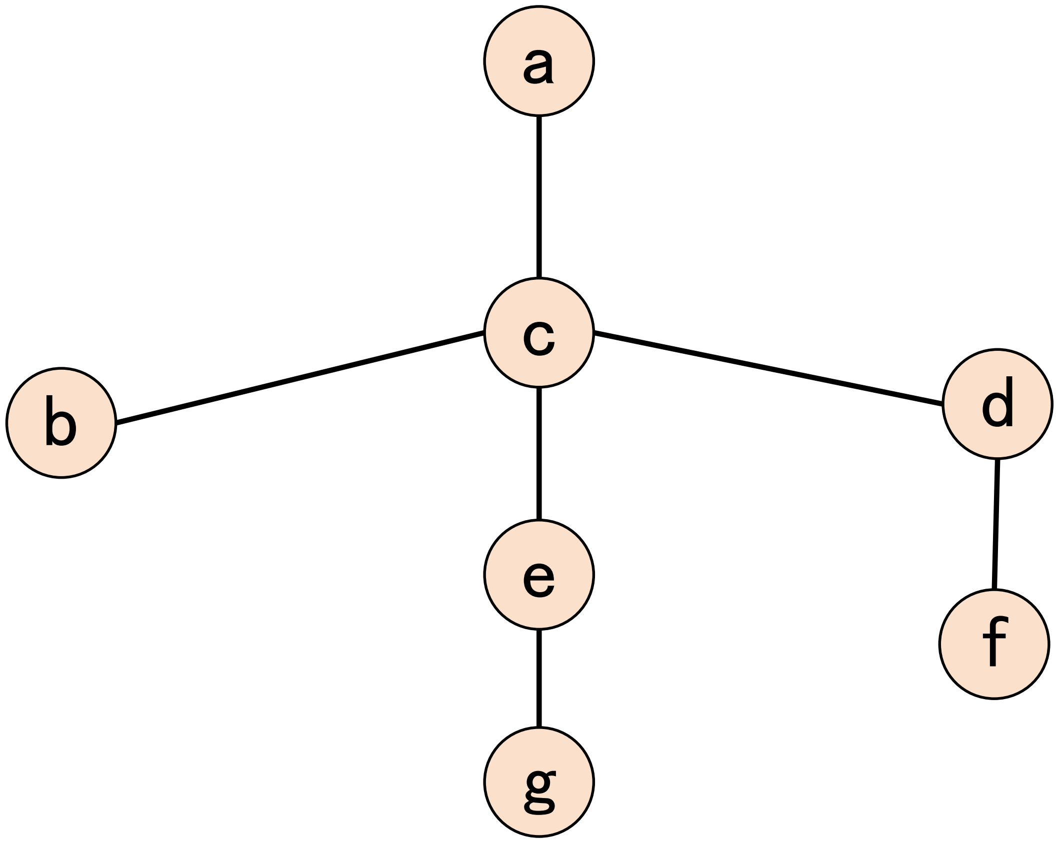
Diameter and Eccentricity

Consider an undirected and unweighted graph G = (V,E)

The diameter of the graph is the maximum distance between two nodes

$$D = \max_{u,v \in V} \{d(u,v)\}$$

d(u,v) = distance between u and v



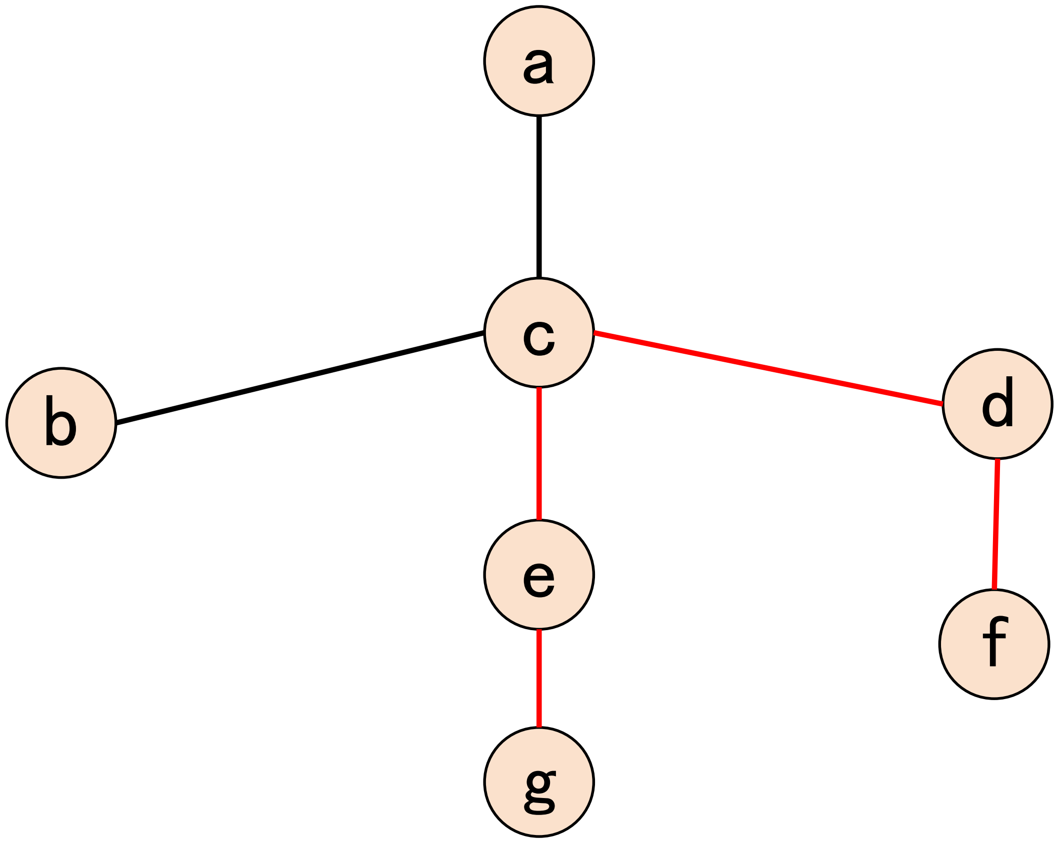
Diameter and Eccentricity

Consider an undirected and unweighted graph G = (V,E)

The diameter of the graph is the maximum distance between two nodes

$$D = \max_{u,v \in V} \{d(u,v)\}$$

d(u,v) = distance between u and v



D = 4

Diameter and Eccentricity

Consider an undirected and unweighted graph G = (V,E)

The diameter of the graph is the maximum distance between two nodes

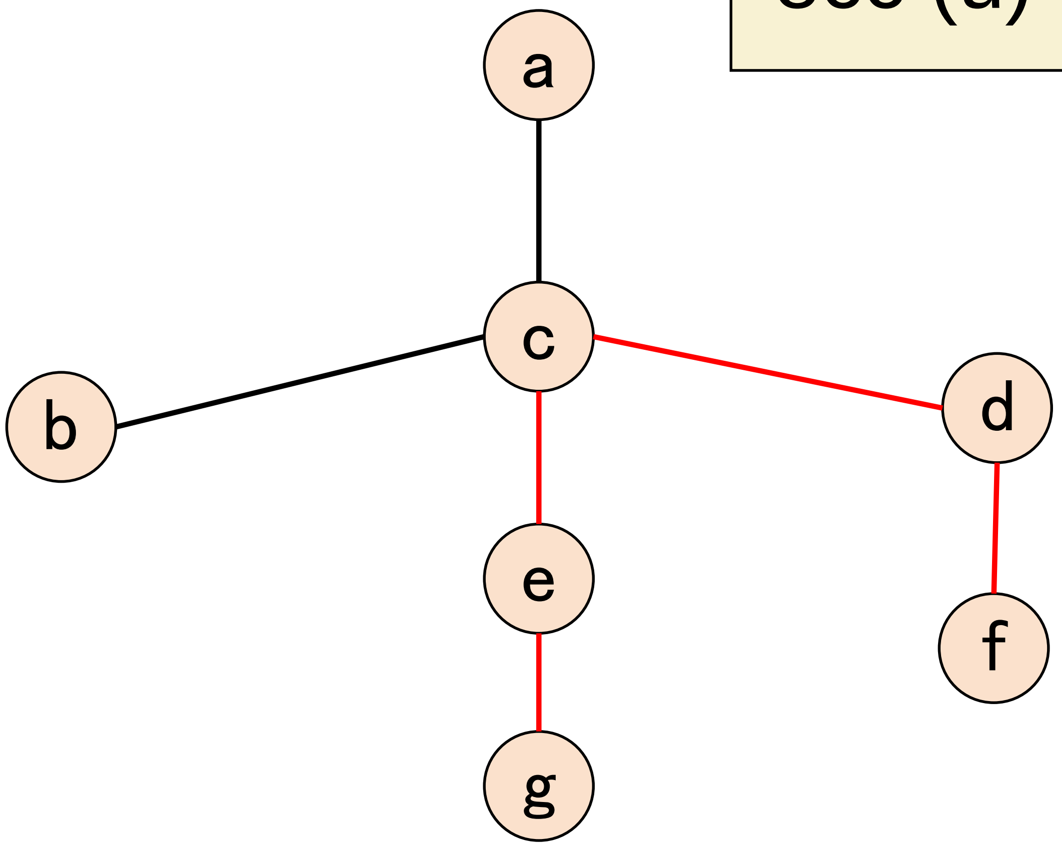
$$D = \max_{u,v \in V} \{d(u,v)\}$$

$$= \max_{u \in V} \{ecc(u)\}$$

d(u,v) = distance between u and v

The eccentricity of a node u is defined as

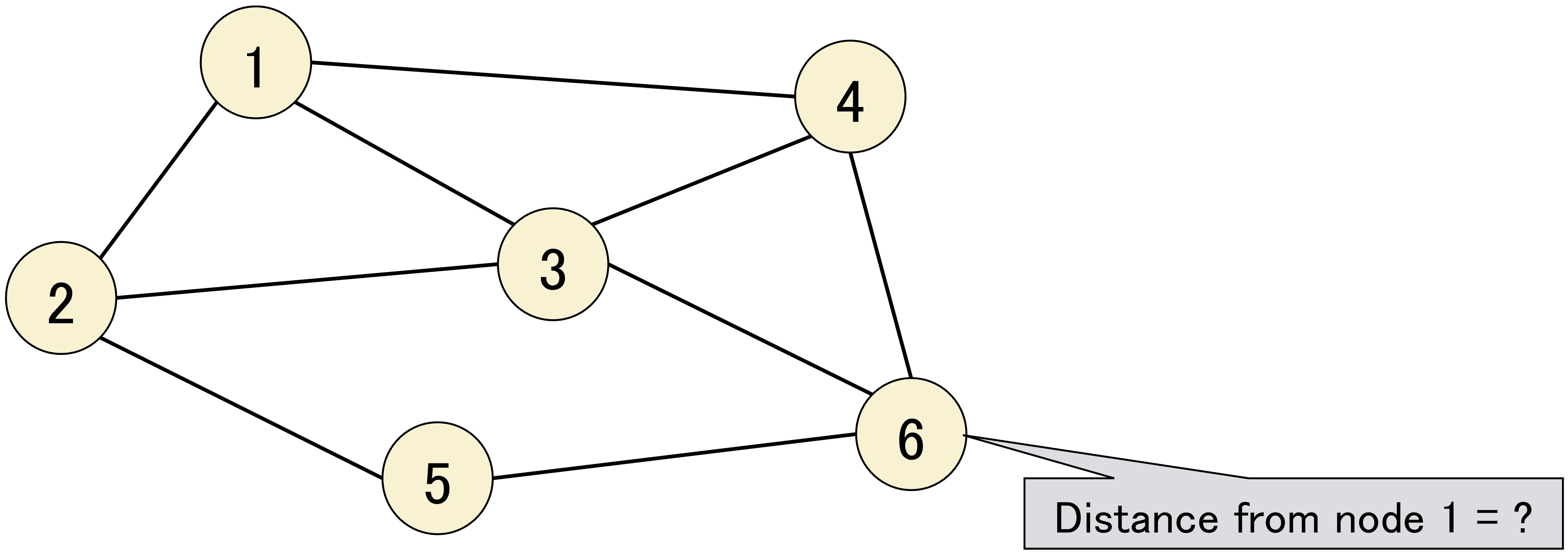
$$ecc(u) = \max_{v \in V} \{d(u,v)\}$$



ecc (a) = 3
ecc (b) = 3
ecc (c) = 2
ecc (d) = 3
ecc (e) = 3
ecc (f) = 4
ecc (g) = 4

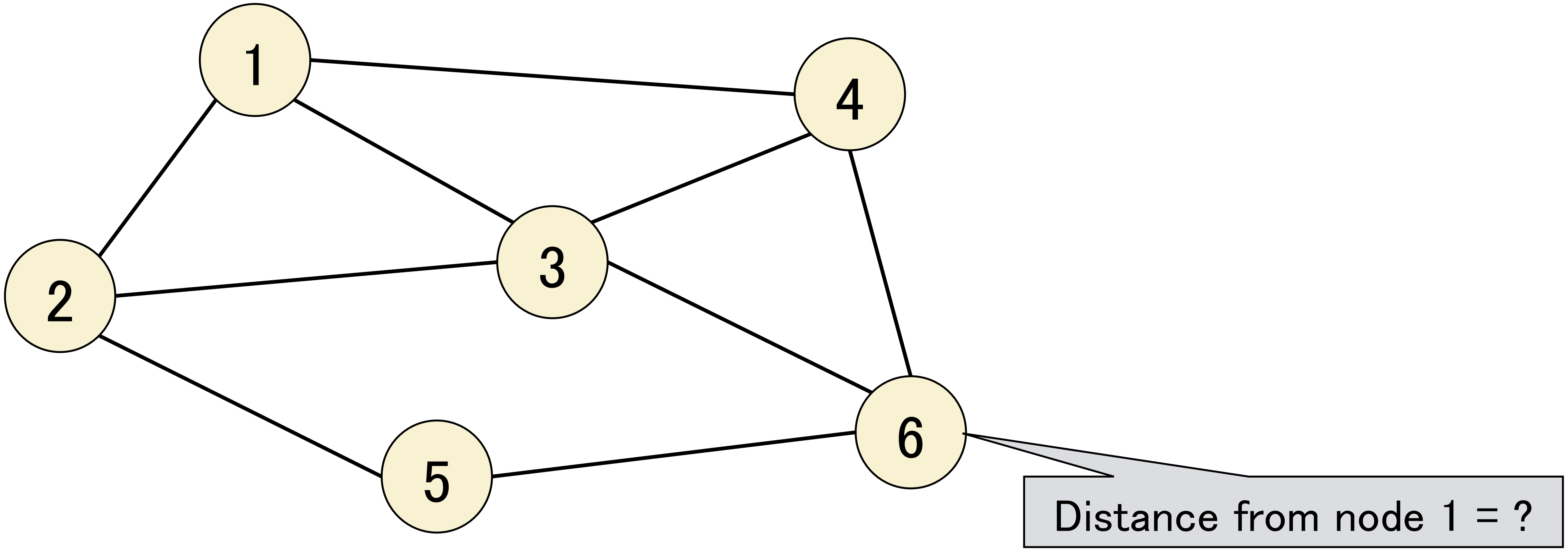
D = 4

Classical Distributed Computing: Computing Distances



Classical Distributed Computing: Computing Distances

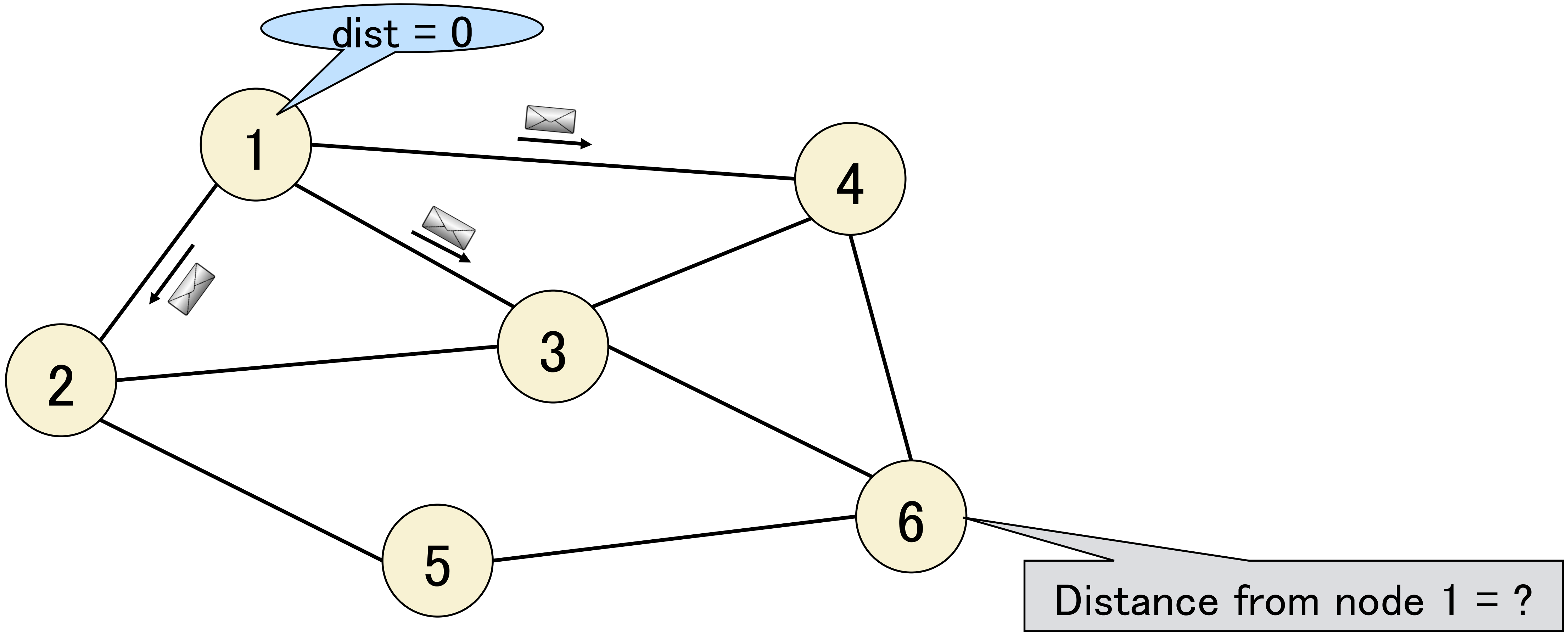
The distances from node 1 can be computed using the Breadth-First Search algorithm



Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Round 1 the source node sends a message to its neighbors

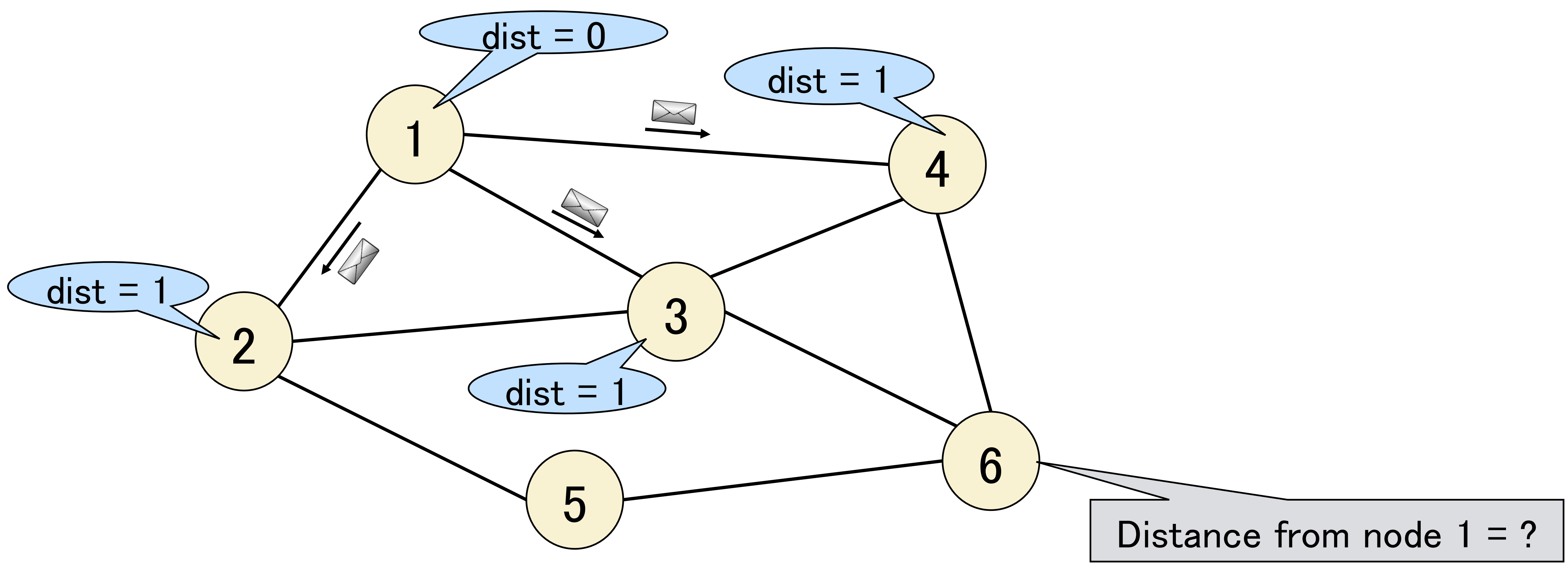


Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Round 1 the source node sends a message to its neighbors

at the end of Round 1: each node updates its distance
(nodes that received a message at Round 1 set "dist = 1")

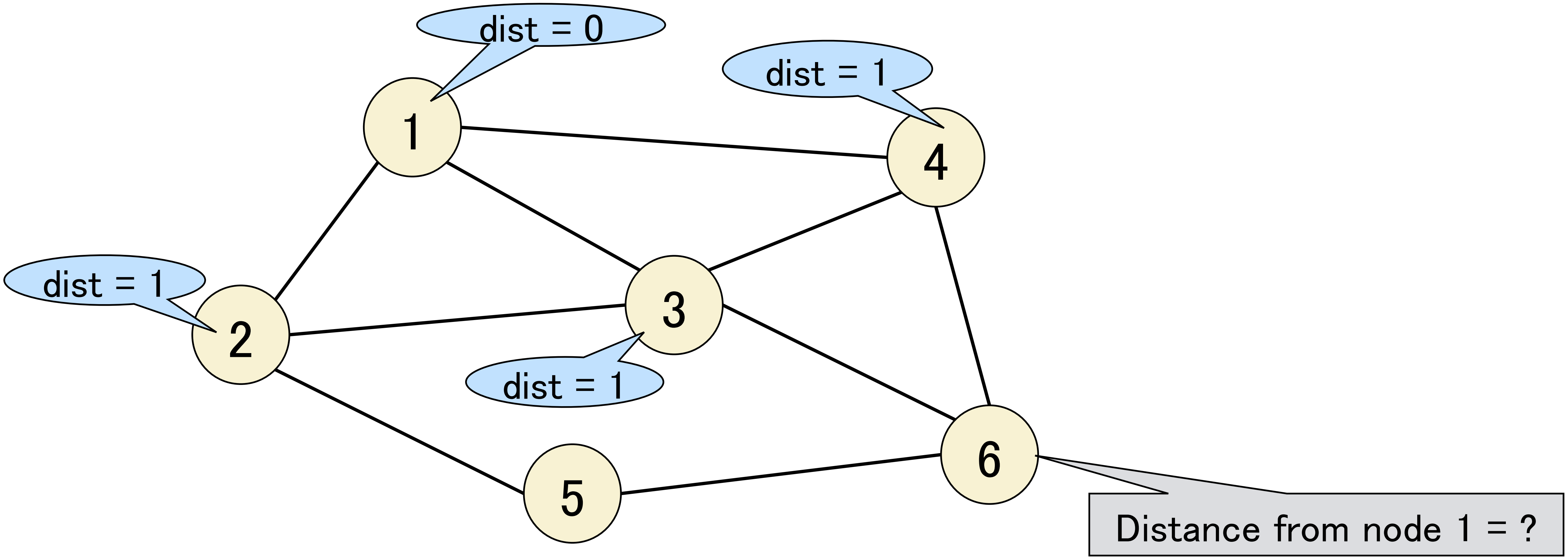


Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Round 1 the source node sends a message to its neighbors

at the end of Round 1: each node updates its distance
(nodes that received a message at Round 1 set “dist = 1”)



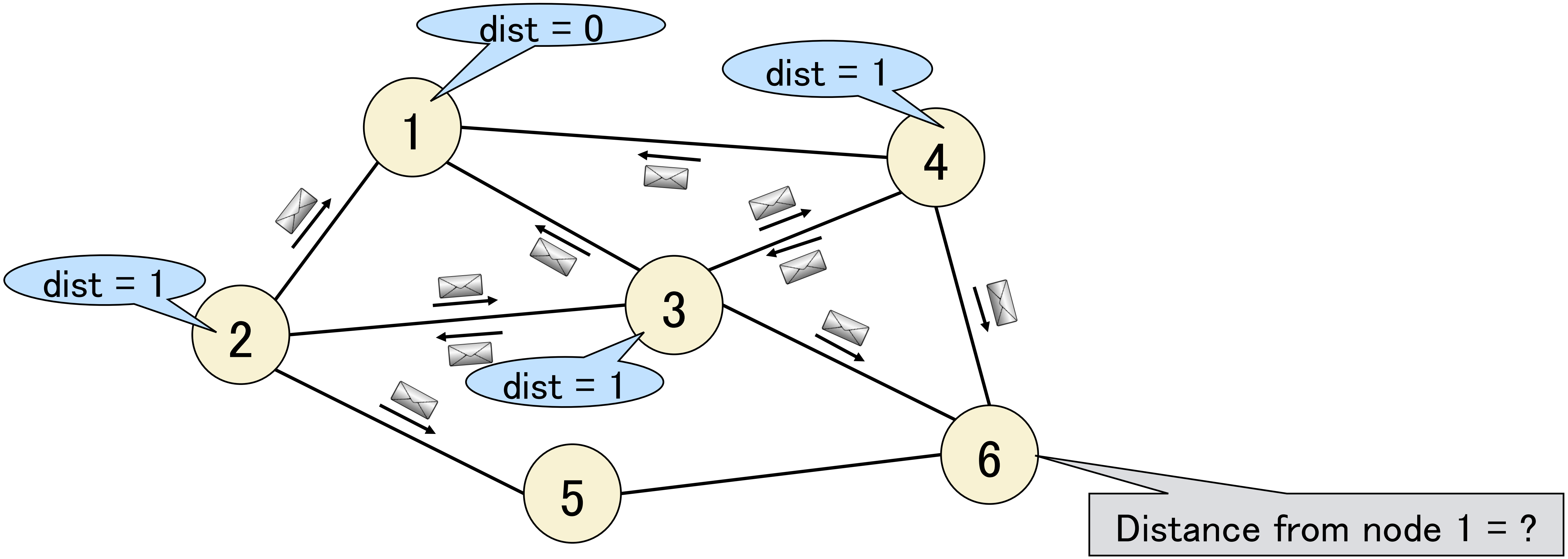
Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Round 1 — the source node sends a message to its neighbors

at the end of Round 1: each node updates its distance
(nodes that received a message at Round 1 set "dist = 1")

Round 2 — nodes tell new knowledge to neighbors



Classical Distributed Computing: Computing Distances

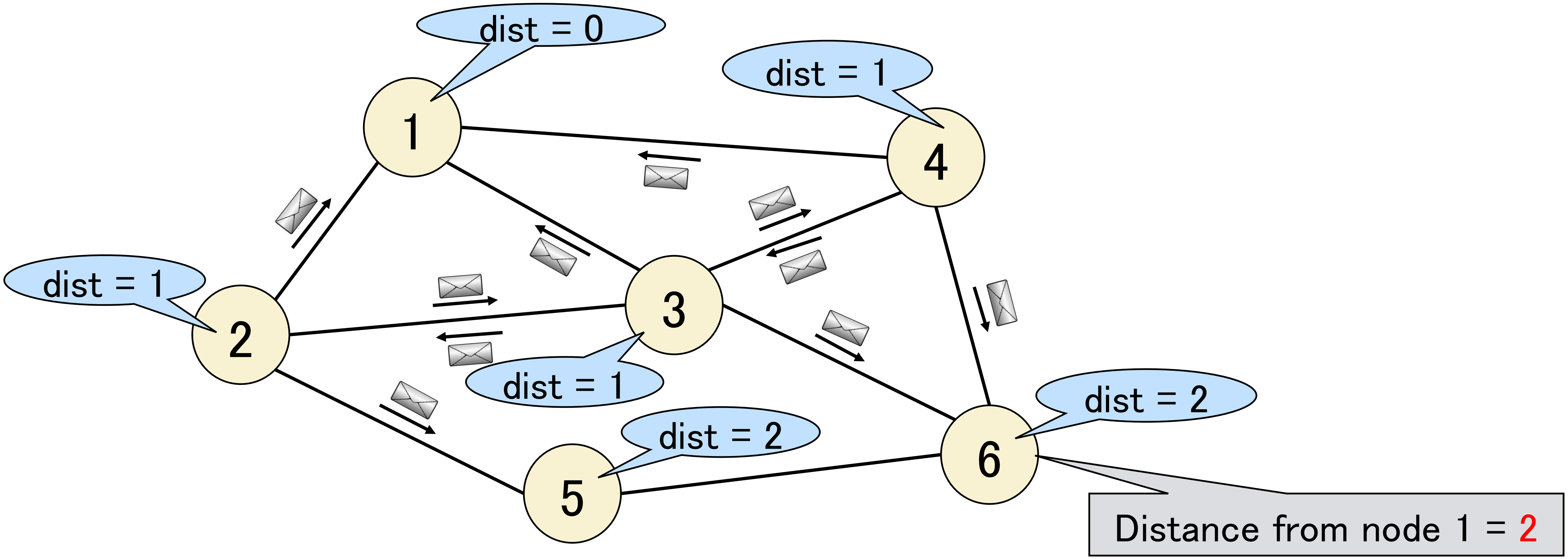
The distances from node 1 can be computed using the Breadth-First Search algorithm

Round 1 the source node sends a message to its neighbors

at the end of Round 1: each node updates its distance
(nodes that received a message at Round 1 set "dist = 1")

Round 2 nodes tell new knowledge to neighbors

at the end of Round 2: each node updates its distance



Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Complexity: ecc(1) rounds (≤ D rounds)

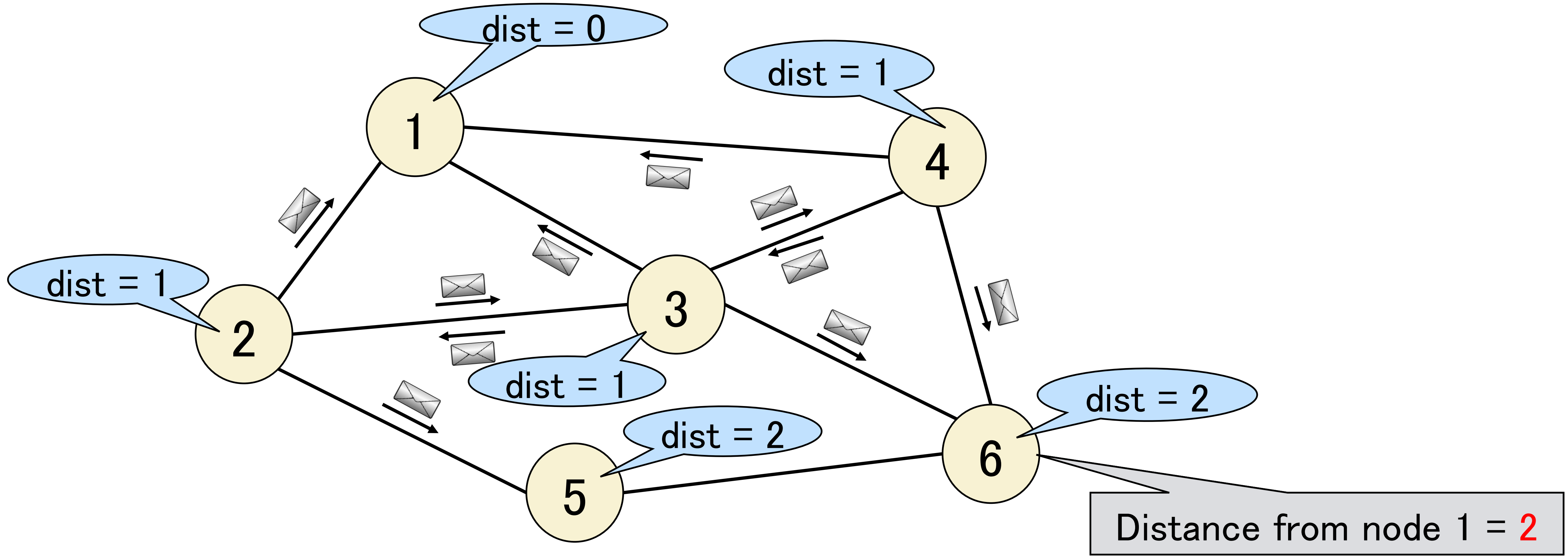
D: diameter of the network

Round 1 the source node sends a message to its neighbors

at the end of Round 1: each node updates its distance
(nodes that received a message at Round 1 set “dist = 1”)

Round 2 nodes tell new knowledge to neighbors

at the end of Round 2: each node updates its distance



Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Complexity: ecc(1) rounds (≤ D rounds)

D: diameter of the network

In classical distributed computing (CONGEST model):

- ✓ for any fixed node u, the distances from u can be computed in D rounds by the Breadth-First Search algorithm (starting at u)

Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Complexity: ecc(1) rounds (≤ D rounds)

D: diameter of the network

In classical distributed computing (CONGEST model):

- ✓ for any fixed node u, the distances from u can be computed in D rounds by the Breadth-First Search algorithm (starting at u)
- ✓ for any fixed node u, the eccentricity ecc(u) can be computed in O(D) rounds by propagating back the information to u

Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Complexity: ecc(1) rounds (≤ D rounds)

D: diameter of the network

In classical distributed computing (CONGEST model):

- ✓ for any fixed node u, the distances from u can be computed in D rounds by the Breadth-First Search algorithm (starting at u)
- ✓ for any fixed node u, the eccentricity ecc(u) can be computed in O(D) rounds by propagating back the information to u
- ✓ but computing the diameter (i.e., the maximum eccentricity) requires Θ(n) rounds even for constant D
  [Frischknecht+12, Holzer+12, Peleg+12, Abboud+16]

Classical Distributed Computing: Computing Distances

The distances from node 1 can be computed using the Breadth-First Search algorithm

Complexity: ecc(1) rounds (≤ D rounds)

D: diameter of the network

In classical distributed computing (CONGEST model):

- ✓ for any fixed node u, the distances from u can be computed in D rounds by the Breadth-First Search algorithm (starting at u)
- ✓ for any fixed node u, the eccentricity ecc(u) can be computed in O(D) rounds by propagating back the information to u
- ✓ but computing the diameter (i.e., the maximum eccentricity) requires Θ(n) rounds even for constant D
  [Frischknecht+12, Holzer+12, Peleg+12, Abboud+16]

We show that we can do better in the quantum setting

Computation of the Diameter in the CONGEST model

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

	Classical	Quantum
Exact computation (upper bounds)	$O(n)$ [Holzer+12, Peleg+12]	$O(\sqrt{nD})$
Exact computation (lower bounds)	$\widetilde{\Omega}(n)$ [Frischknecht+12]	

number of rounds needed to compute the diameter (n: number of nodes, D: diameter)

Computation of the Diameter in the CONGEST model

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

	Classical	Quantum
Exact computation (upper bounds)	$O(n)$ [Holzer+12, Peleg+12]	$O(\sqrt{nD})$
Exact computation (lower bounds)	$\widetilde{\Omega}(n)$ [Frischknecht+12]	$\widetilde{\Omega}(\sqrt{n} + D)$ [unconditional]

number of rounds needed to compute the diameter (n: number of nodes, D: diameter)

Computation of the Diameter in the CONGEST model

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

	Classical	Quantum
Exact computation (upper bounds)	$O(n)$ [Holzer+12, Peleg+12]	$O(\sqrt{nD})$
Exact computation (lower bounds)	$\widetilde{\Omega}(n)$ [Frischknecht+12]	$\widetilde{\Omega}(\sqrt{n}+D)$ [unconditional] $\widetilde{\Omega}(\sqrt{nD})$ [conditional]

number of rounds needed to compute the diameter (n: number of nodes, D: diameter)

condition: holds for quantum distributed algorithms using only polylog(n) qubits of memory per node

Computation of the Diameter in the CONGEST model

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

	Classical	Quantum
Exact computation (upper bounds)	$O(n)$ [Holzer+12, Peleg+12]	$O(\sqrt{nD})$
Exact computation (lower bounds)	$\widetilde{\Omega}(n)$ [Frischknecht+12]	$\widetilde{\Omega}(\sqrt{n}+D)$ [unconditional] $\widetilde{\Omega}(\sqrt{nD})$ [conditional]

number of rounds needed to compute the diameter (n: number of nodes, D: diameter)

condition: holds for quantum distributed algorithms using only polylog(n) qubits of memory per node

lower bounds proved using reductions from the 2-party communication complexity of the Disjointness function

Computation of the Diameter in the CONGEST model

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

	Classical	Quantum
Exact computation (upper bounds)	$O(n)$ [Holzer+12, Peleg+12]	$O(\sqrt{nD})$
Exact computation (lower bounds)	$\widetilde{\Omega}(n)$ [Frischknecht+12]	$\widetilde{\Omega}(\sqrt{n}+D)$ [unconditional] $\widetilde{\Omega}(\sqrt{nD})$ [conditional]

number of rounds needed to compute the diameter (n: number of nodes, D: diameter)

condition: holds for quantum distributed algorithms using only polylog(n) qubits of memory per node

lower bounds proved using reductions from the 2-party communication complexity of the Disjointness function

very recent result [Magniez, Nayak 2020]

$\widetilde{\Omega}(\sqrt{n}+n^{1/3}D^{2/3})$ [unconditional]

Computation of the Diameter in the CONGEST model

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

	Classical	Quantum
Exact computation (upper bounds)	$O(n)$ [Holzer+12, Peleg+12]	$O(\sqrt{nD})$
Exact computation (lower bounds)	$\widetilde{\Omega}(n)$ [Frischknecht+12]	$\widetilde{\Omega}(\sqrt{n}+D)$ [unconditional] $\widetilde{\Omega}(\sqrt{nD})$ [conditional]

number of rounds needed to compute the diameter (n: number of nodes, D: diameter)

condition: holds for quantum distributed algorithms using only polylog(n) qubits of memory per node

lower bounds proved using reductions from the 2-party communication complexity of the Disjointness function

very recent result [Magniez, Nayak 2020]

$\widetilde{\Omega}(\sqrt{n}+n^{1/3}D^{2/3})$ [unconditional]

Quantum Distributed Computation of the Diameter

Computation of the diameter (decision version)

Given an integer d, decide if diameter ≥ d

Quantum Distributed Computation of the Diameter

Computation of the diameter (decision version)

Given an integer d, decide if diameter ≥ d

there is a vertex u such that ecc (u) ≥ d

Quantum Distributed Computation of the Diameter

Computation of the diameter (decision version)

Given an integer d, decide if diameter ≥ d

there is a vertex u such that ecc (u) ≥ d

This is a search problem

Idea: try to use Grover search

Quantum Distributed Computation of the Diameter

Computation of the diameter (decision version)

> Given an integer d, decide if diameter ≥ d
>
> (diameter ≥ d: there is a vertex u such that ecc (u) ≥ d)

This is a search problem

Idea: try to use Grover search

Define the function $f: V \rightarrow \{0,1\}$ such that $f(u) = \begin{cases} 1 & \text{if } ecc(u) \geq d \\ 0 & \text{otherwise} \end{cases}$

Goal: find u such that f(u) = 1 (or report that no such vertex exists)

Quantum Distributed Computation of the Diameter

Computation of the diameter (decision version)

> Given an integer d, decide if diameter ≥ d
>
> (diameter ≥ d: there is a vertex u such that ecc (u) ≥ d)

This is a search problem
Idea: try to use Grover search

Define the function f: V → {0,1} such that f(u) = $\begin{cases} 1 \text{ if ecc (u)} \geq d \\ 0 \text{ otherwise} \end{cases}$

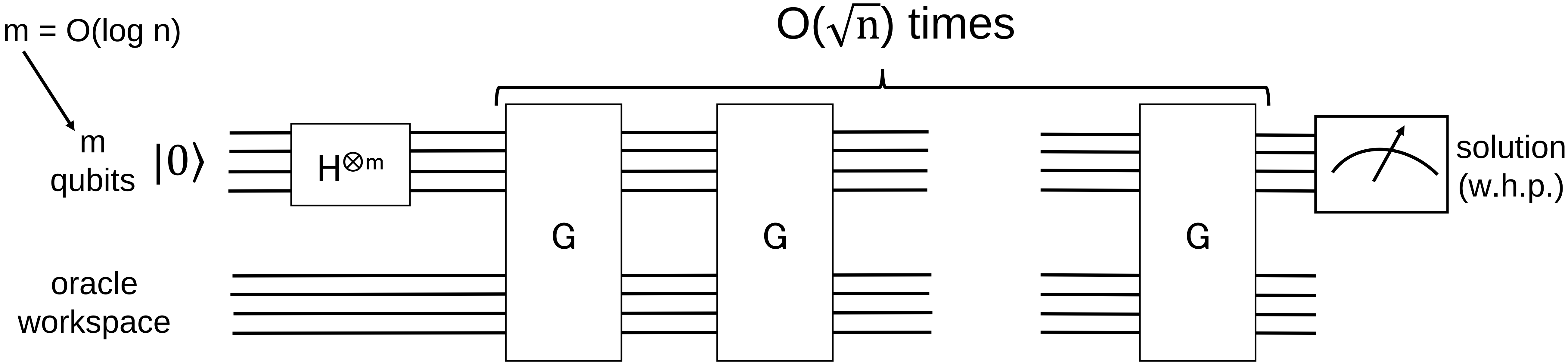
Goal: find u such that f(u) = 1 (or report that no such vertex exists)

> There is a quantum algorithm for this search problem using $O(\sqrt{n})$ calls to a black box evaluating f

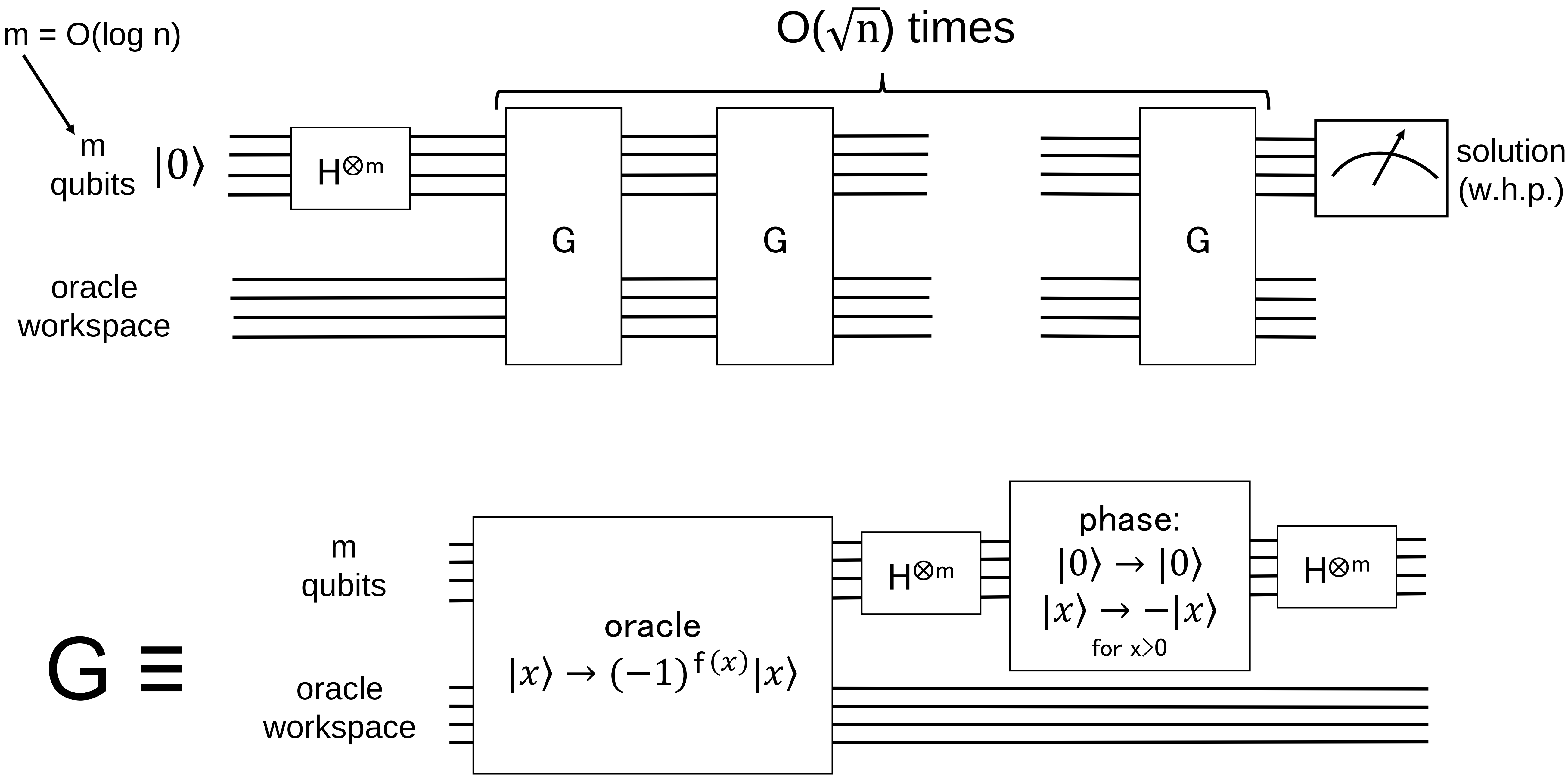
Quantum search [Grover 96]

n = |V| (number of nodes)

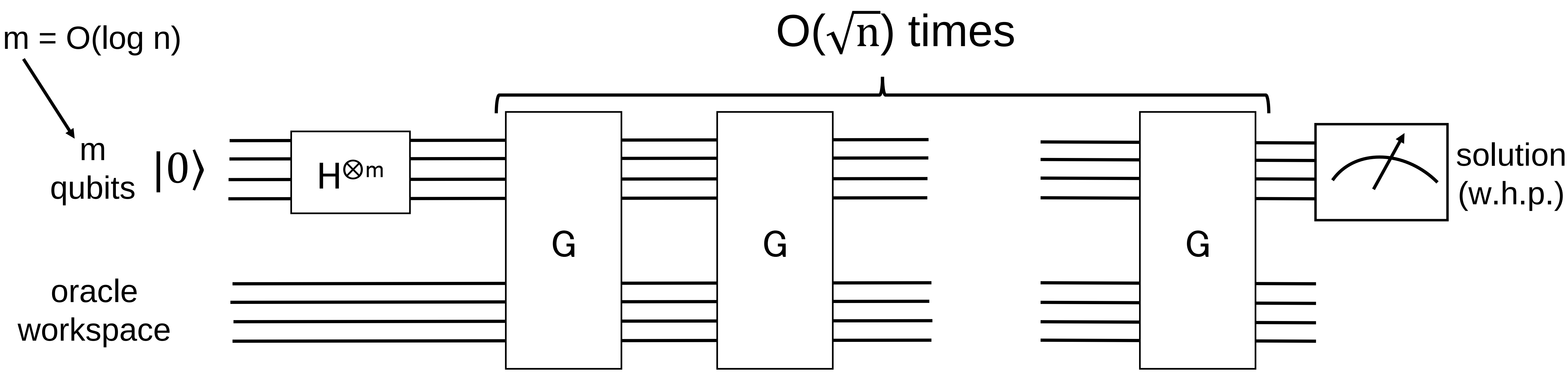
Recap: Grover Algorithm

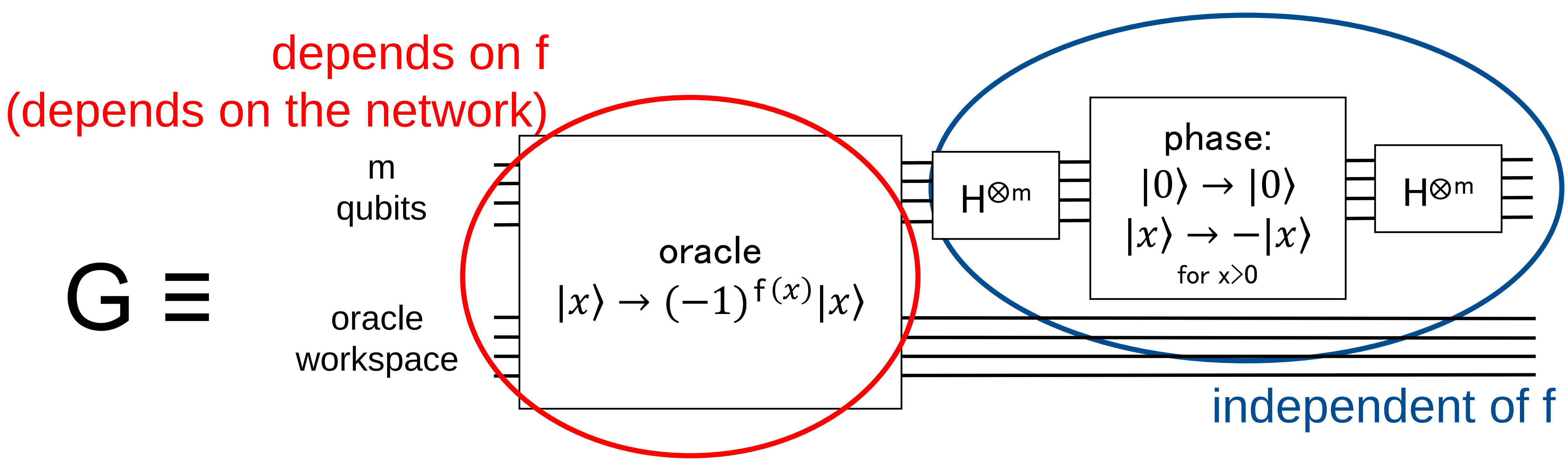


Recap: Grover Algorithm

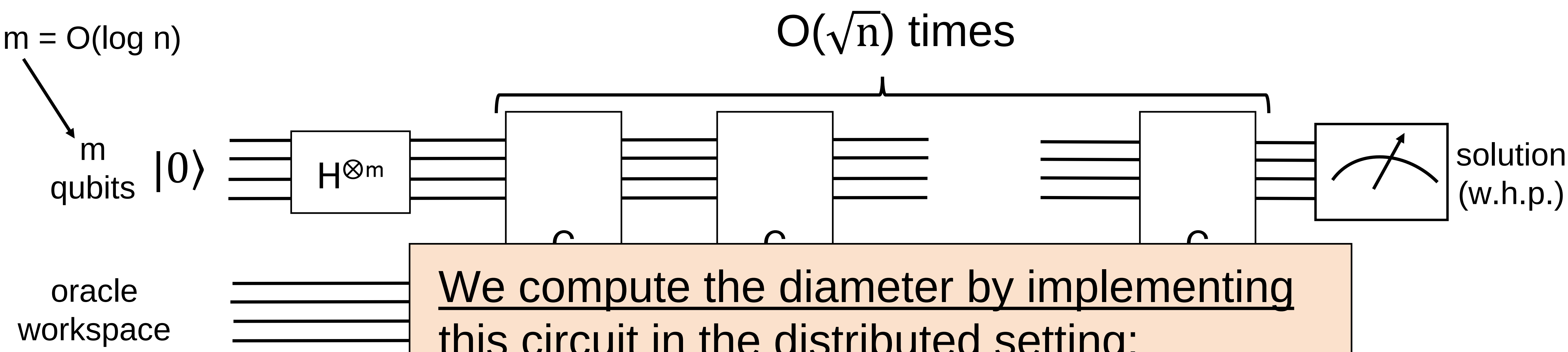


Recap: Grover Algorithm



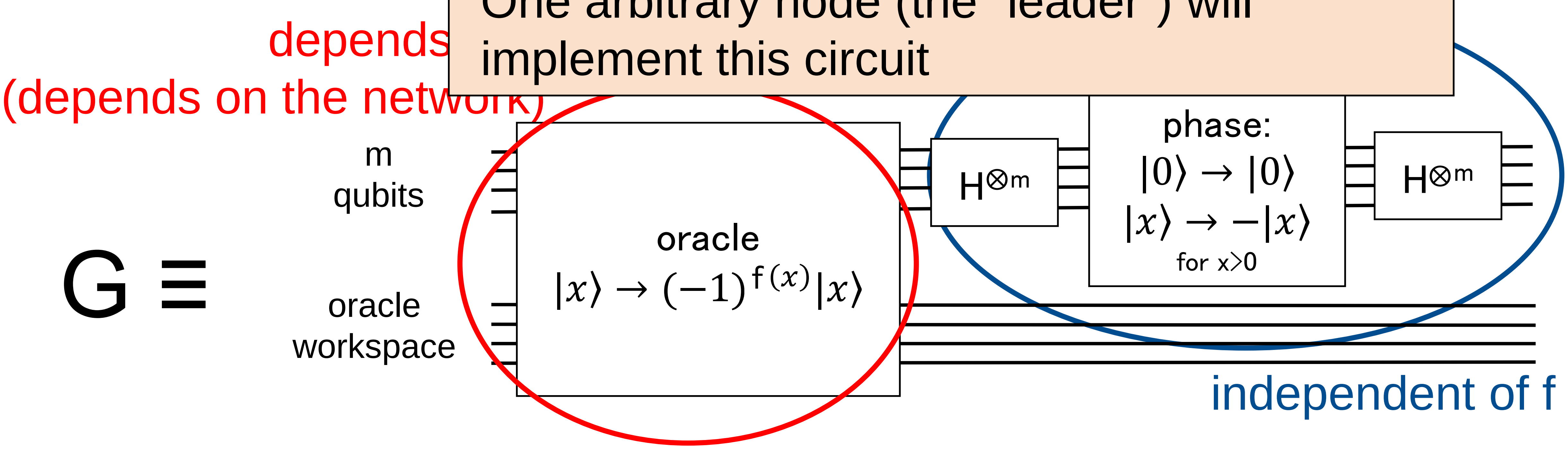


Recap: Grover Algorithm

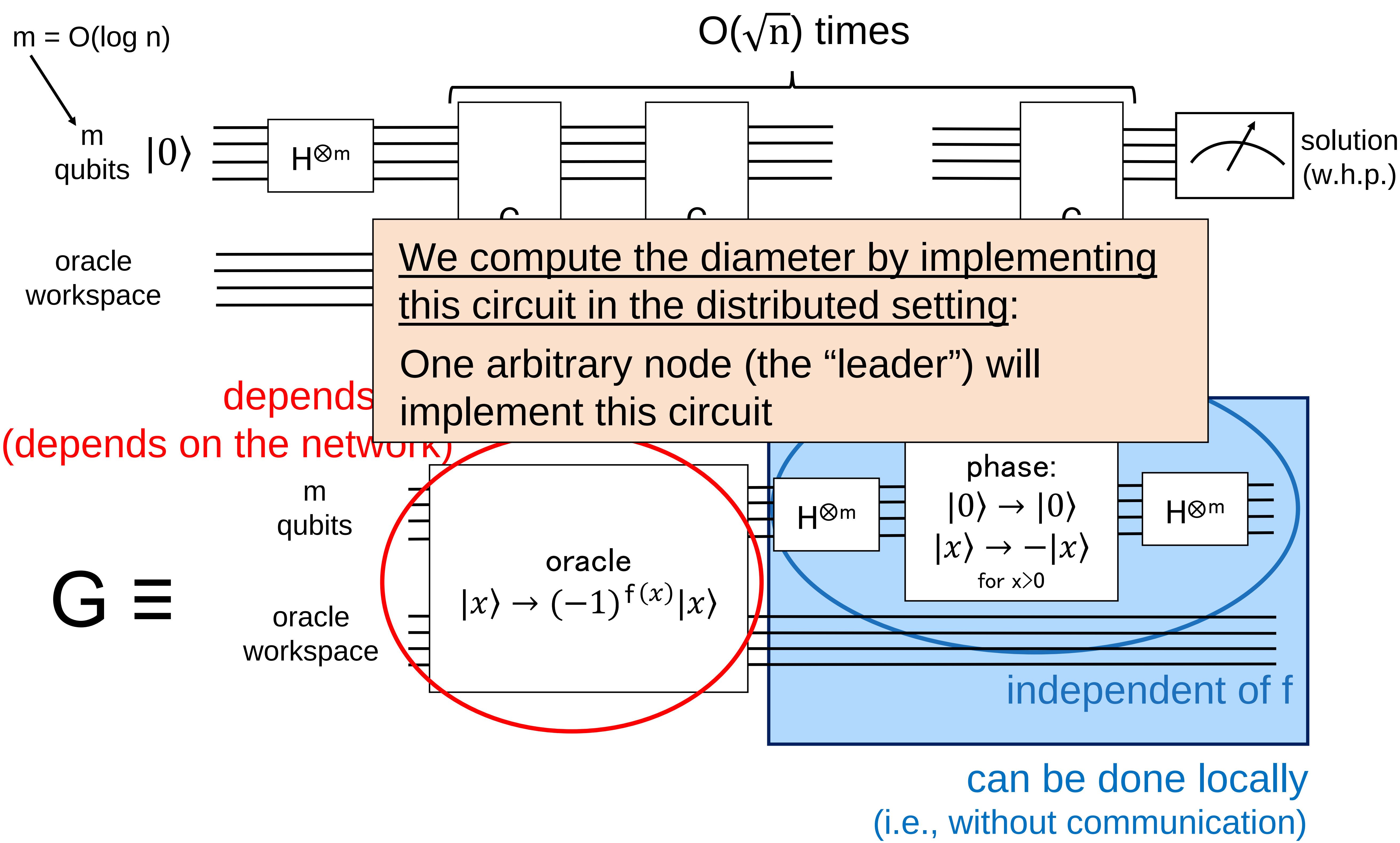


<u>We compute the diameter by implementing this circuit in the distributed setting</u>:

One arbitrary node (the "leader") will implement this circuit



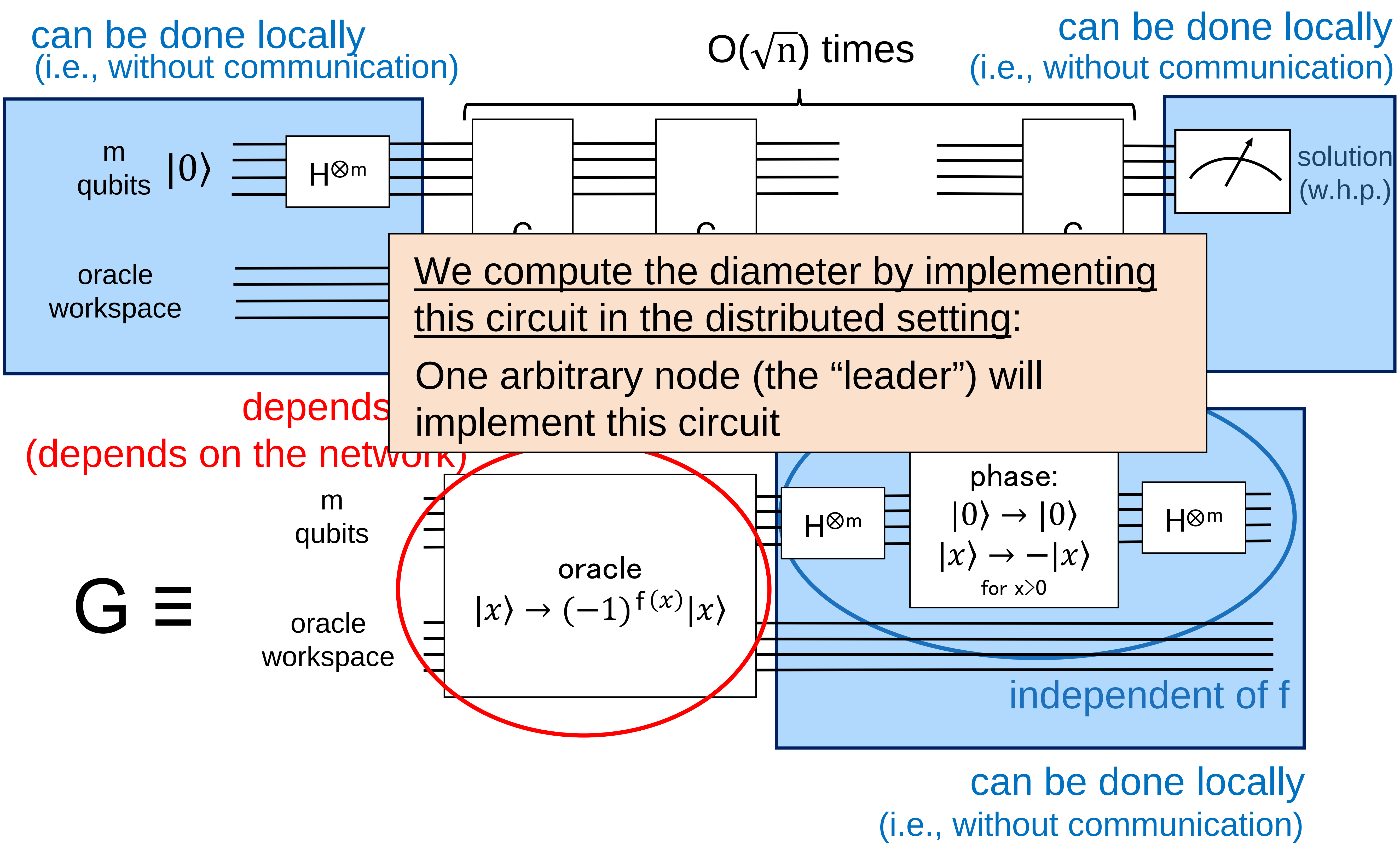
Recap: Grover Algorithm



We compute the diameter by implementing this circuit in the distributed setting:

One arbitrary node (the "leader") will implement this circuit

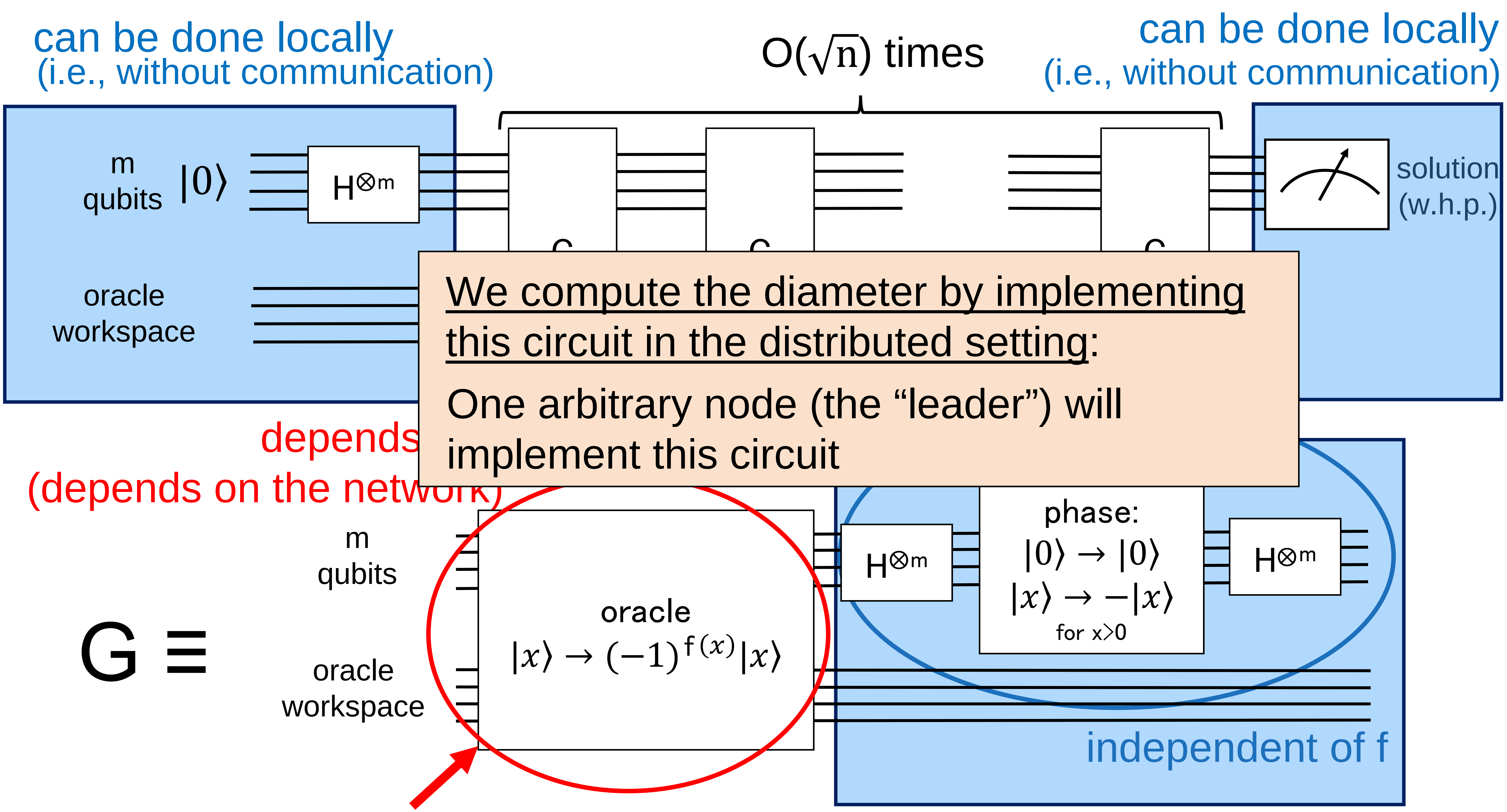
Recap: Grover Algorithm



We compute the diameter by implementing this circuit in the distributed setting:

One arbitrary node (the "leader") will implement this circuit

Recap: Grover Algorithm

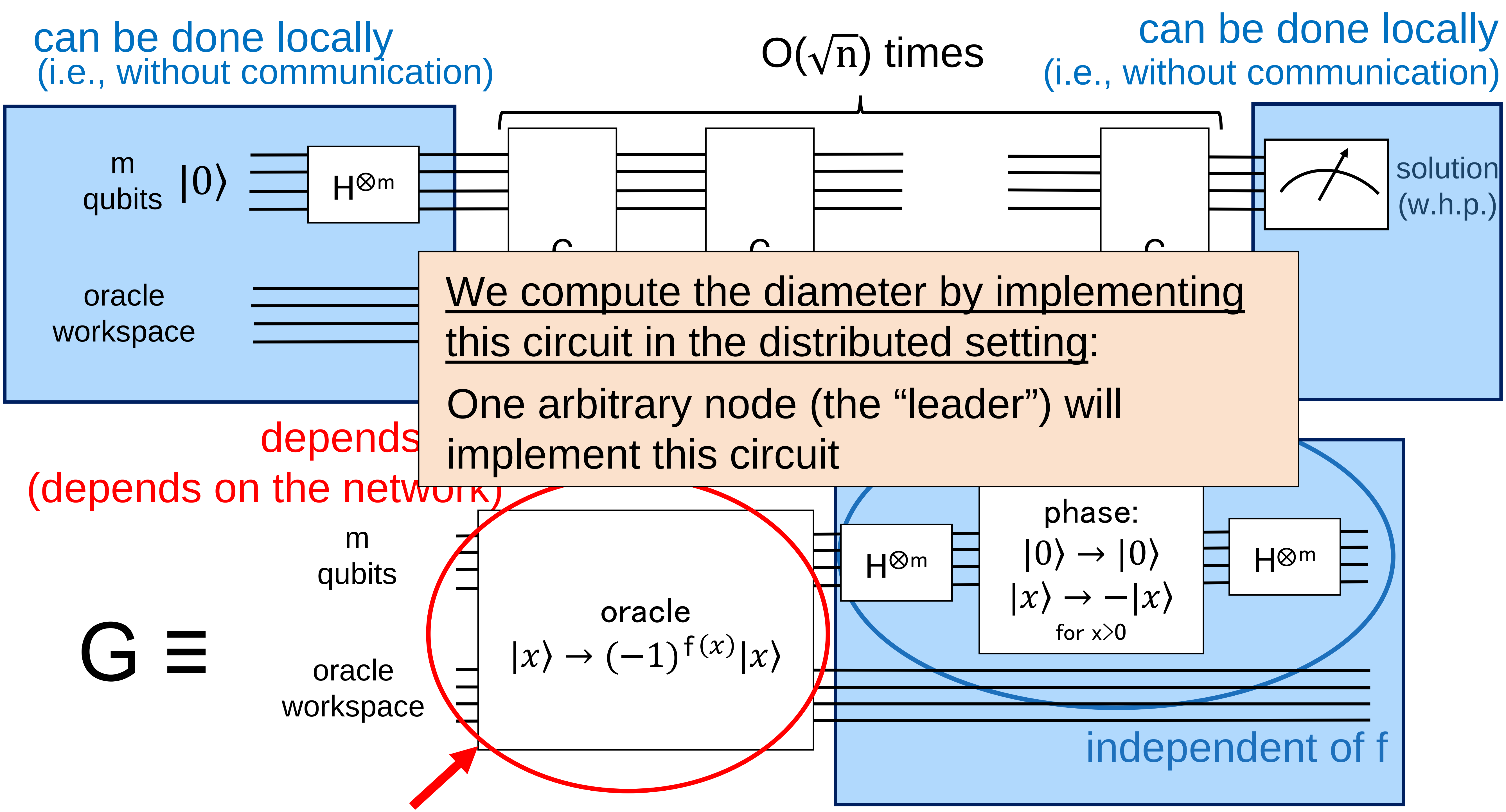


We compute the diameter by implementing this circuit in the distributed setting:

One arbitrary node (the "leader") will implement this circuit

To implement the oracle, the leader node needs to communicate with the other nodes

Recap: Grover Algorithm



We compute the diameter by implementing this circuit in the distributed setting:

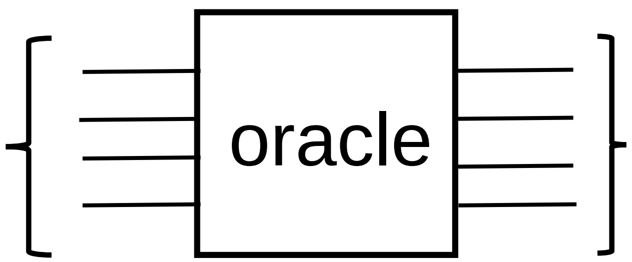
One arbitrary node (the "leader") will implement this circuit

To implement the oracle, the leader node needs to communicate with the other nodes

Total number of rounds of communication = O($\sqrt{n}$ x number of rounds to implement the oracle)

Implementation of the Oracle in O(D) rounds

$|u\rangle|0\rangle$

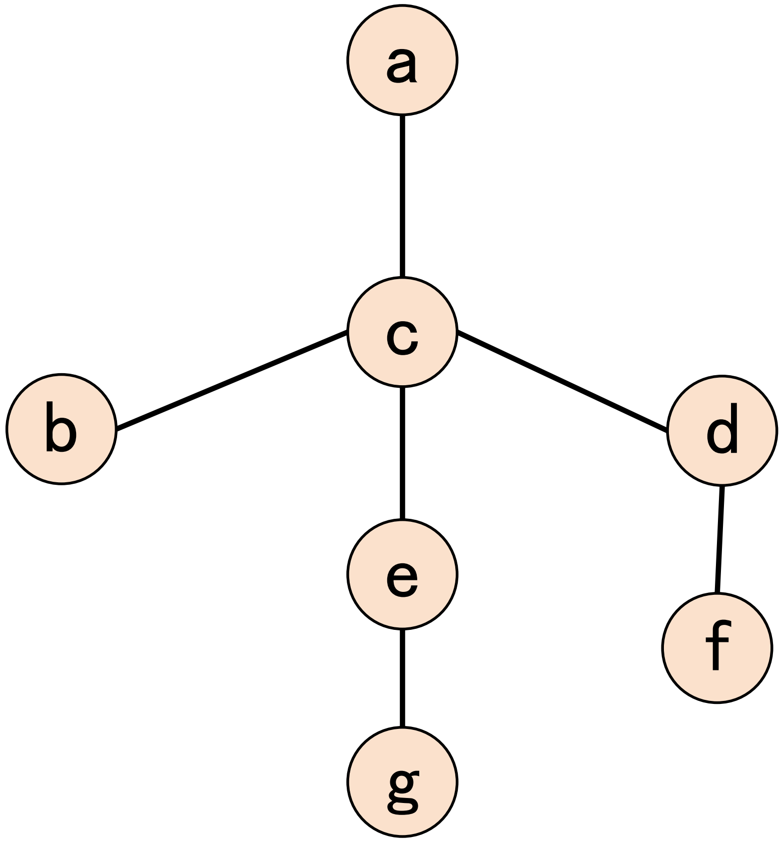


$|u\rangle|ecc(u)\rangle$

Example:

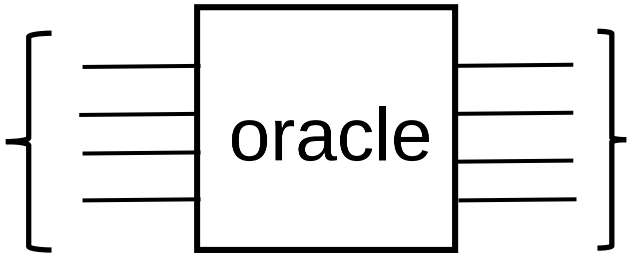
V={a,b,c,d,e,f,g}

here leader = node a



Implementation of the Oracle in O(D) rounds

$$\sum_{u \in V} \alpha_u |u\rangle|0\rangle$$

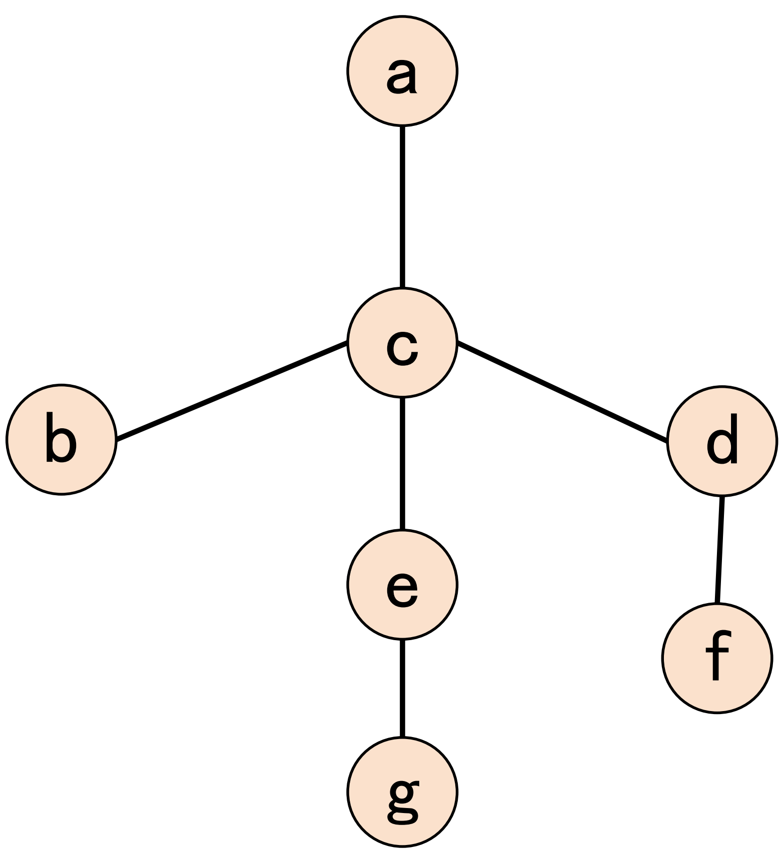


$$\sum_{u \in V} \alpha_u |u\rangle|ecc(u)\rangle$$

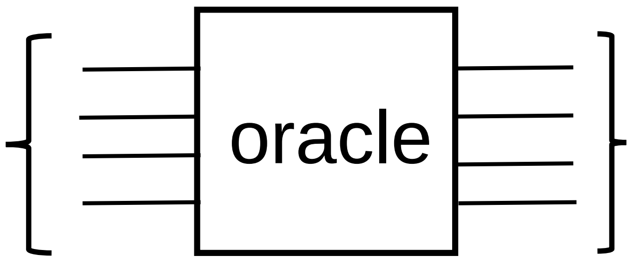
Example:

V={a,b,c,d,e,f,g}

here leader = node a



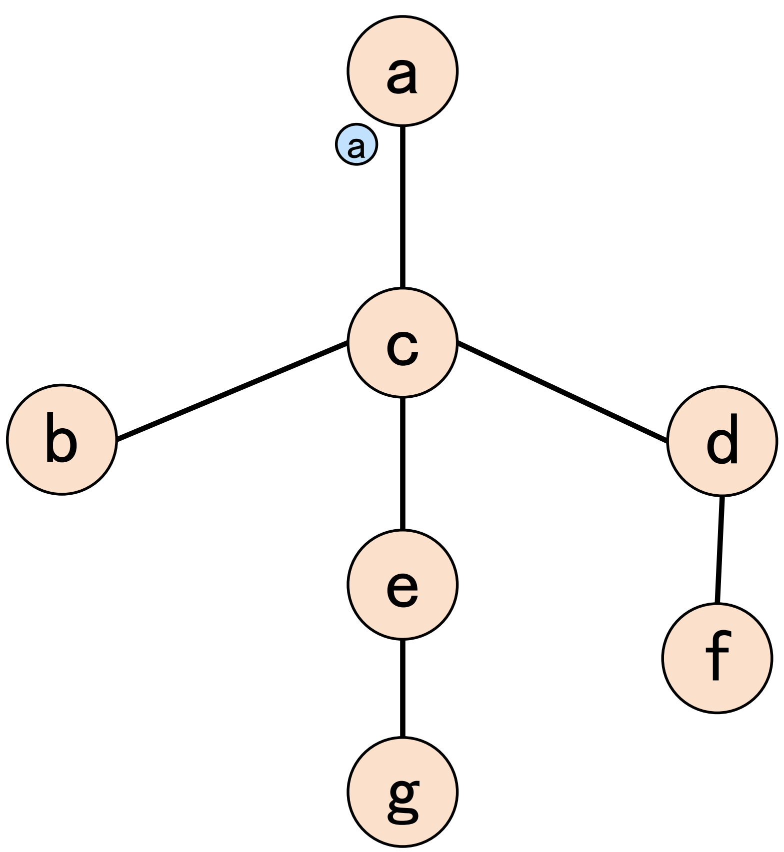
Implementation of the Oracle in O(D) rounds

$\sum_{u \in V} \alpha_u |u\rangle |0\rangle$  $\sum_{u \in V} \alpha_u |u\rangle |ecc(u)\rangle$

<u>Example</u>:

V={a,b,c,d,e,f,g}

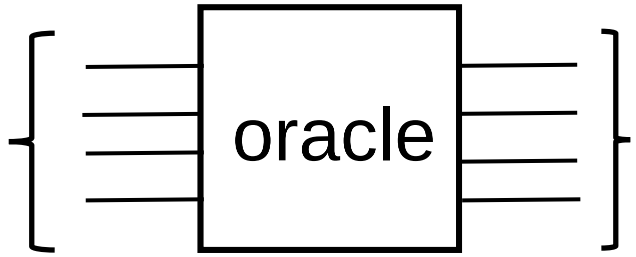
here leader = node a



Initially node a owns $\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}}$

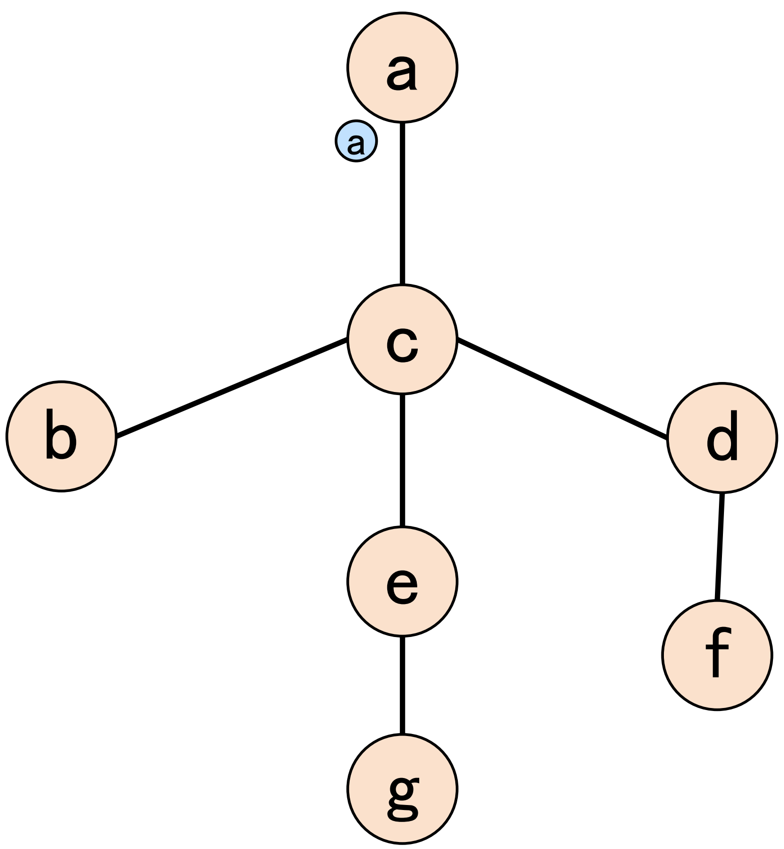
Implementation of the Oracle in O(D) rounds

$$\sum_{u \in V} \alpha_u |u\rangle|0\rangle$$



$$\sum_{u \in V} \alpha_u |u\rangle|ecc(u)\rangle$$

Example:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}}$

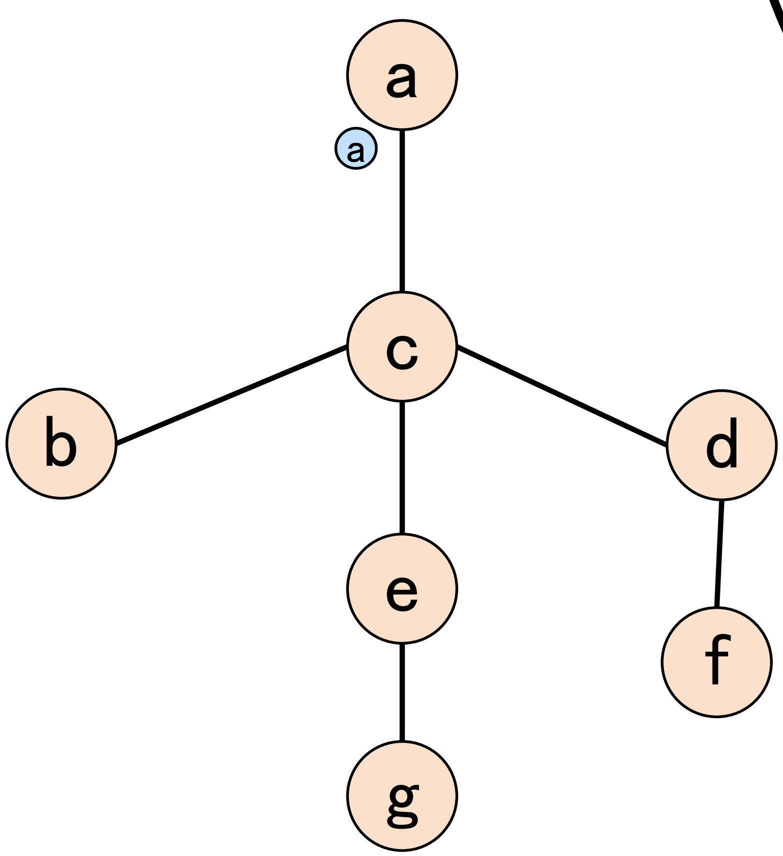
1. "Broadcast" this state, which gives

$$\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{d}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{f}}|u\rangle_{\mathrm{g}}$$

Implementati

$$\sum_{u \in V} \alpha_u |u\rangle |0\rangle$$

<u>Example</u>:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}}$

1. "<u>Broadcast</u>" this state, which gives

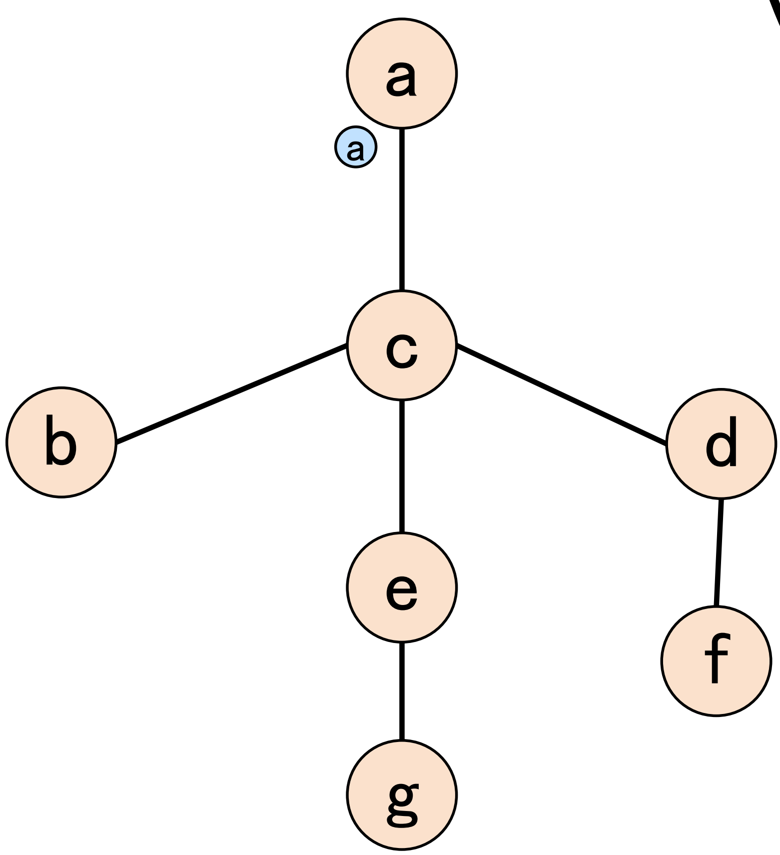
$$\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{b}} |u\rangle_{\mathrm{c}} |u\rangle_{\mathrm{d}} |u\rangle_{\mathrm{e}} |u\rangle_{\mathrm{f}} |u\rangle_{\mathrm{g}}$$

Implementati

Node a introduces 1 register $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|0\rangle$

$$\sum_{u\in V}\alpha_u|u\rangle|0\rangle$$

<u>Example</u>:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}$

1. "<u>Broadcast</u>" this state, which gives

$$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{d}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{f}}|u\rangle_{\mathrm{g}}$$

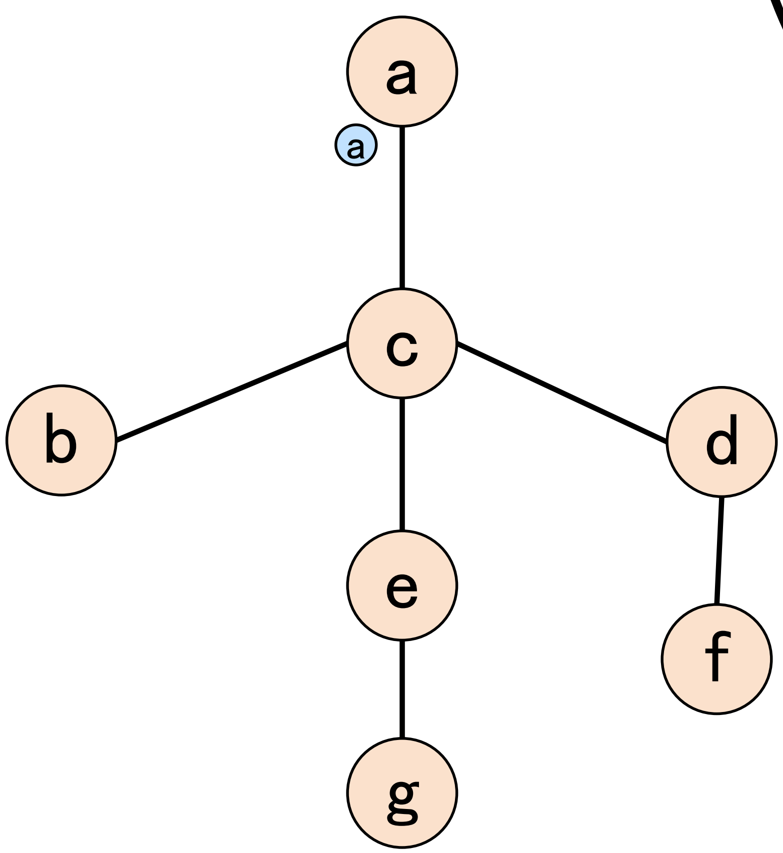
Implementati

Node a introduces 1 register $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|0\rangle$

Node a applies CNOTS $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle$

$$\sum_{u\in V}\alpha_u|u\rangle|0\rangle$$

Example:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}$

1. “Broadcast” this state, which gives

$$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{d}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{f}}|u\rangle_{\mathrm{g}}$$

Implementati

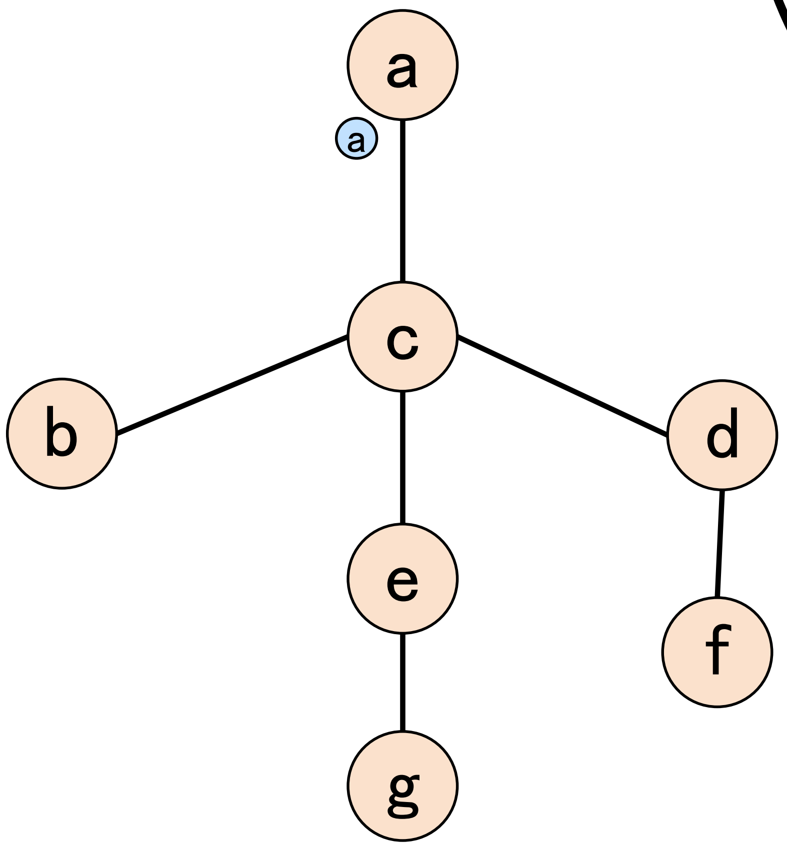
Node a introduces 1 register $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |0\rangle$

Node a applies CNOTS $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle$

Node a sends the second register to c $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{c}}$

$$\sum_{u\in V} \alpha_u |u\rangle |0\rangle$$

Example:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}}$

1. “Broadcast” this state, which gives

$$\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{b}} |u\rangle_{\mathrm{c}} |u\rangle_{\mathrm{d}} |u\rangle_{\mathrm{e}} |u\rangle_{\mathrm{f}} |u\rangle_{\mathrm{g}}$$

Implementati

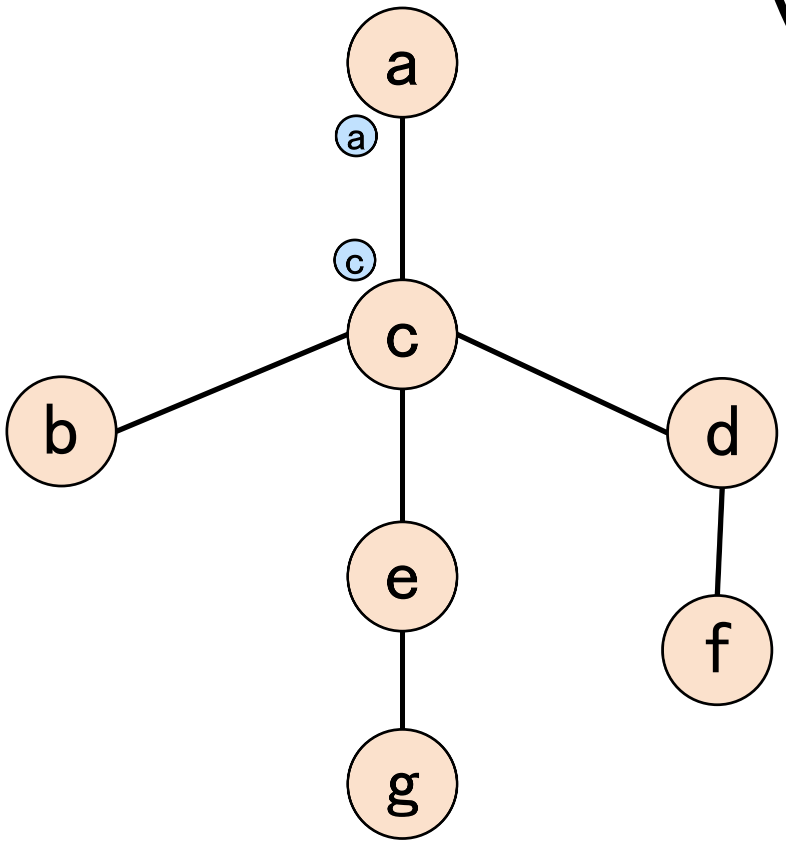
Node a introduces 1 register $\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}} |0\rangle$

Node a applies CNOTS $\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle$

Node a sends the second register to c $\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{c}}$

$$\sum_{u \in V} \alpha_u |u\rangle |0\rangle$$

Example:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}}$

1. "Broadcast" this state, which gives

$$\sum_{u \in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{b}} |u\rangle_{\mathrm{c}} |u\rangle_{\mathrm{d}} |u\rangle_{\mathrm{e}} |u\rangle_{\mathrm{f}} |u\rangle_{\mathrm{g}}$$

Implementati

Node a introduces 1 register $\sum_{u \in V} \alpha_u |u\rangle_a |0\rangle$

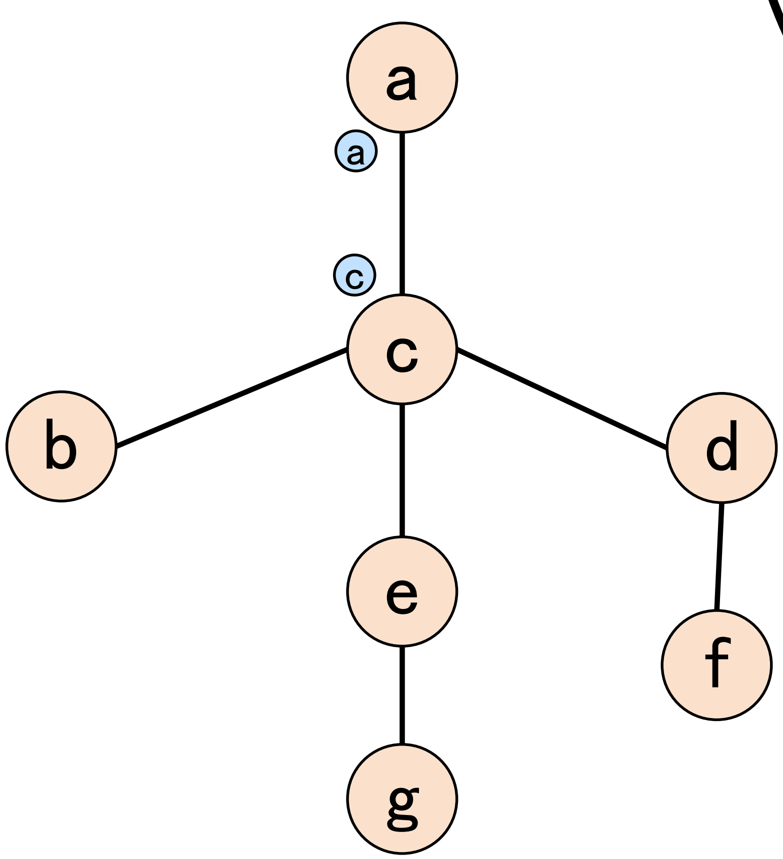
Node a applies CNOTS $\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle$

Node a sends the second register to c $\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle_c$

Node c introduces 3 registers $\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle_c |0\rangle \; |0\rangle \; |0\rangle$

$$\sum_{u \in V} \alpha_u |u\rangle |0\rangle$$

Example:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u \in V} \alpha_u |u\rangle_a$

1. "Broadcast" this state, which gives

$$\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle_b |u\rangle_c |u\rangle_d |u\rangle_e |u\rangle_f |u\rangle_g$$

Implementati

Node a introduces 1 register $\sum_{u \in V} \alpha_u |u\rangle_a |0\rangle$

Node a applies CNOTS $\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle$

Node a sends the second register to c $\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle_c$

Node c introduces 3 registers $\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle_c |0\rangle \; |0\rangle \; |0\rangle$

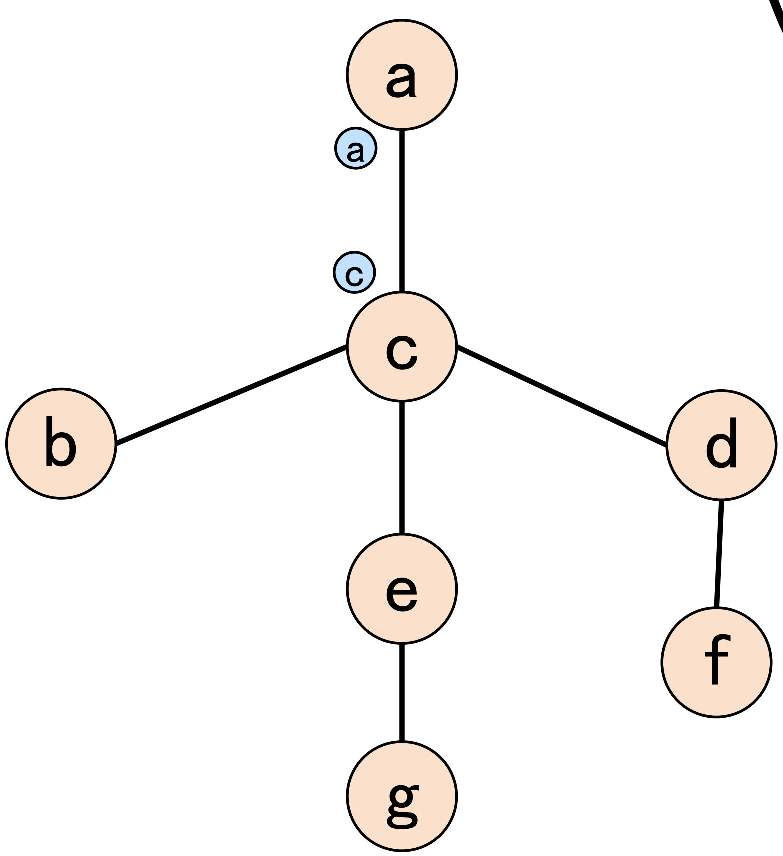
Node c applies CNOTS $\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle_c |u\rangle \; |u\rangle \; |u\rangle$

$$\sum_{u \in V} \alpha_u |u\rangle |0\rangle$$

<u>Example</u>:

V={a,b,c,d,e,f,g}

here leader = node a



Initially node a owns $\sum_{u \in V} \alpha_u |u\rangle_a$

1. "<u>Broadcast</u>" this state, which gives

$$\sum_{u \in V} \alpha_u |u\rangle_a |u\rangle_b |u\rangle_c |u\rangle_d |u\rangle_e |u\rangle_f |u\rangle_g$$

Implementati

Node a introduces 1 register $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |0\rangle$

Node a applies CNOTS $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle$

Node a sends the second register to c $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{c}}$

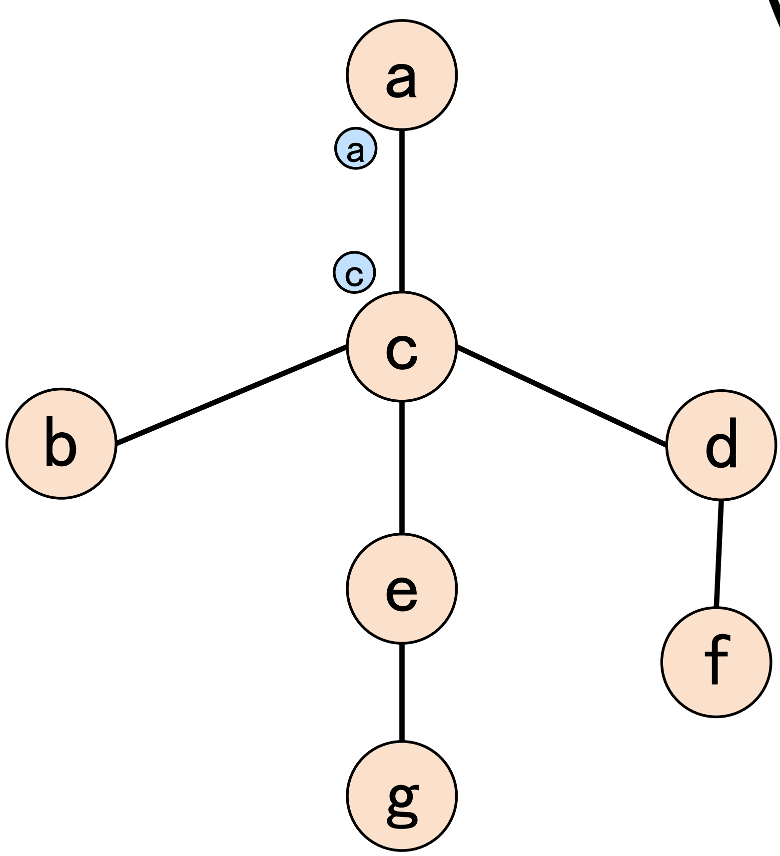
Node c introduces 3 registers $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{c}} |0\rangle \; |0\rangle \; |0\rangle$

Node c applies CNOTS $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{c}} |u\rangle \; |u\rangle \; |u\rangle$

Node c sends the registers to b,e,d $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{c}} |u\rangle_{\mathrm{b}} |u\rangle_{\mathrm{e}} |u\rangle_{\mathrm{d}}$

$$\sum_{u\in V} \alpha_u |u\rangle |0\rangle$$

Example:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}}$

1. "Broadcast" this state, which gives

$$\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{b}} |u\rangle_{\mathrm{c}} |u\rangle_{\mathrm{d}} |u\rangle_{\mathrm{e}} |u\rangle_{\mathrm{f}} |u\rangle_{\mathrm{g}}$$

Implementati

Node a introduces 1 register $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|0\rangle$

Node a applies CNOTS $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle$

Node a sends the second register to c $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}$

Node c introduces 3 registers $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|0\rangle\;|0\rangle\;|0\rangle$

Node c applies CNOTS $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|u\rangle\;|u\rangle\;|u\rangle$

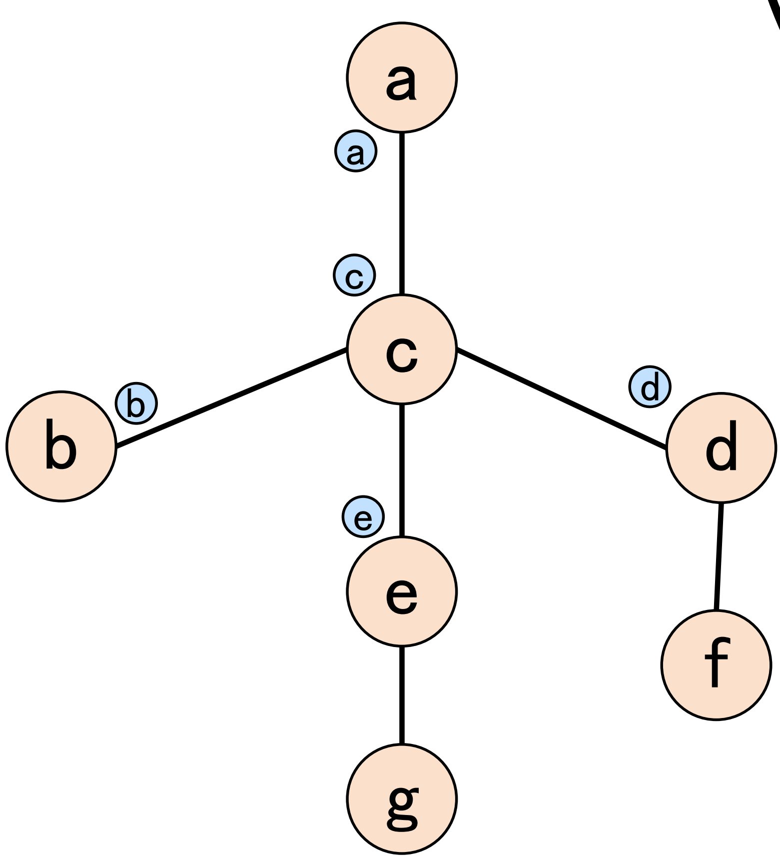
Node c sends the registers to b,e,d $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{d}}$

$$\sum_{u\in V}\alpha_u|u\rangle|0\rangle$$

<u>Example</u>:

V={a,b,c,d,e,f,g}

here leader = node a



Initially node a owns $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}$

1. "<u>Broadcast</u>" this state, which gives

$$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{d}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{f}}|u\rangle_{\mathrm{g}}$$

Implementati

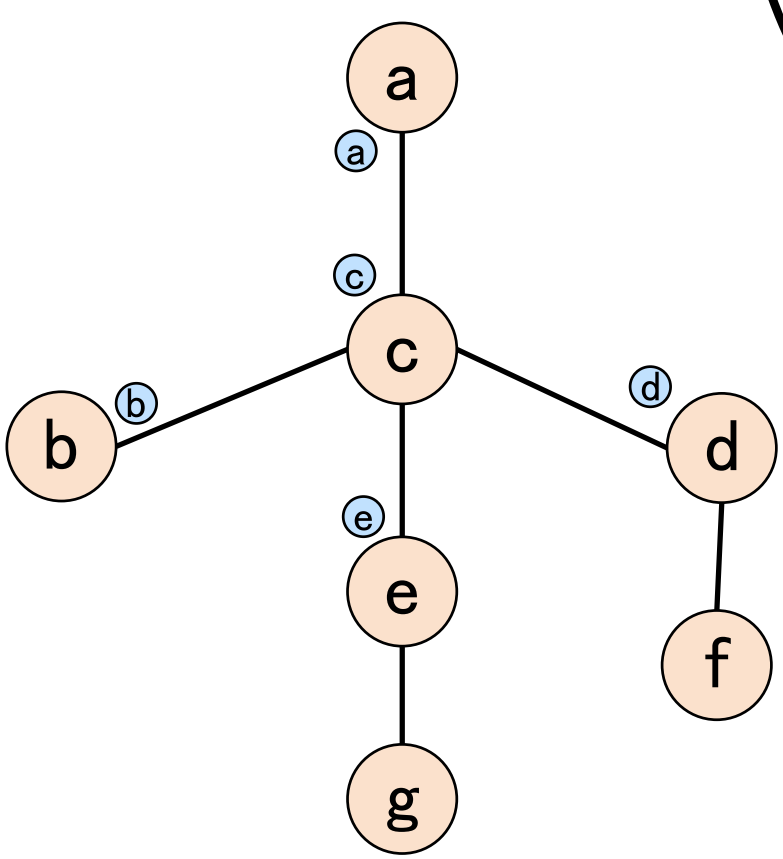
Node a introduces 1 register	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|0\rangle$
Node a applies CNOTS	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle$
Node a sends the second register to c	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}$
Node c introduces 3 registers	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|0\rangle\ \ |0\rangle\ \ |0\rangle$
Node c applies CNOTS	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|u\rangle\ \ |u\rangle\ \ |u\rangle$
Node c sends the registers to b,e,d	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{d}}$
……	

$$\sum_{u\in V}\alpha_u|u\rangle|0\rangle$$

<u>Example</u>:

V={a,b,c,d,e,f,g}

here leader = node a



Initially node a owns $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}$

1. "<u>Broadcast</u>" this state, which gives

$$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{d}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{f}}|u\rangle_{\mathrm{g}}$$

Implementati

Node a introduces 1 register $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|0\rangle$

Node a applies CNOTS $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle$

Node a sends the second register to c $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}$

Node c introduces 3 registers $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|0\rangle\ |0\rangle\ |0\rangle$

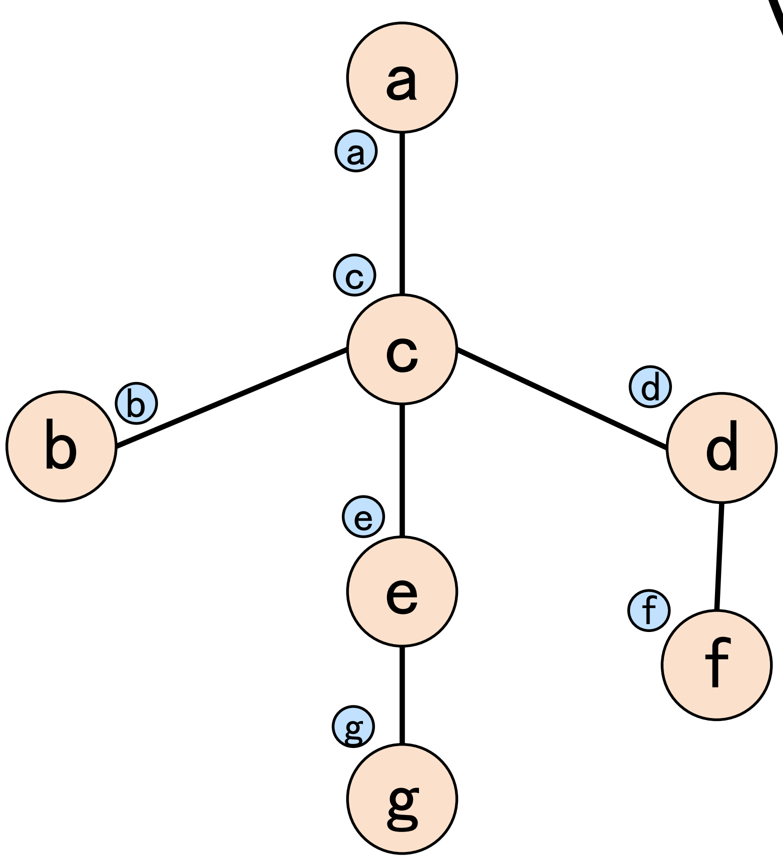
Node c applies CNOTS $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|u\rangle\ |u\rangle\ |u\rangle$

Node c sends the registers to b,e,d $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{d}}$

……

$$\sum_{u\in V}\alpha_u|u\rangle|0\rangle$$

Example:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}$

1. "Broadcast" this state, which gives

$$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{d}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{f}}|u\rangle_{\mathrm{g}}$$

Implementati

Node a introduces 1 register $\sum_{u\in V} \alpha_u |u\rangle_a |0\rangle$

Node a applies CNOTS $\sum_{u\in V} \alpha_u |u\rangle_a |u\rangle$

Node a sends the second register to c $\sum_{u\in V} \alpha_u |u\rangle_a |u\rangle_c$

Node c introduces 3 registers $\sum_{u\in V} \alpha_u |u\rangle_a |u\rangle_c |0\rangle \; |0\rangle \; |0\rangle$

Node c applies CNOTS $\sum_{u\in V} \alpha_u |u\rangle_a |u\rangle_c |u\rangle \; |u\rangle \; |u\rangle$

Node c sends the registers to b,e,d $\sum_{u\in V} \alpha_u |u\rangle_a |u\rangle_c |u\rangle_b |u\rangle_e |u\rangle_d$

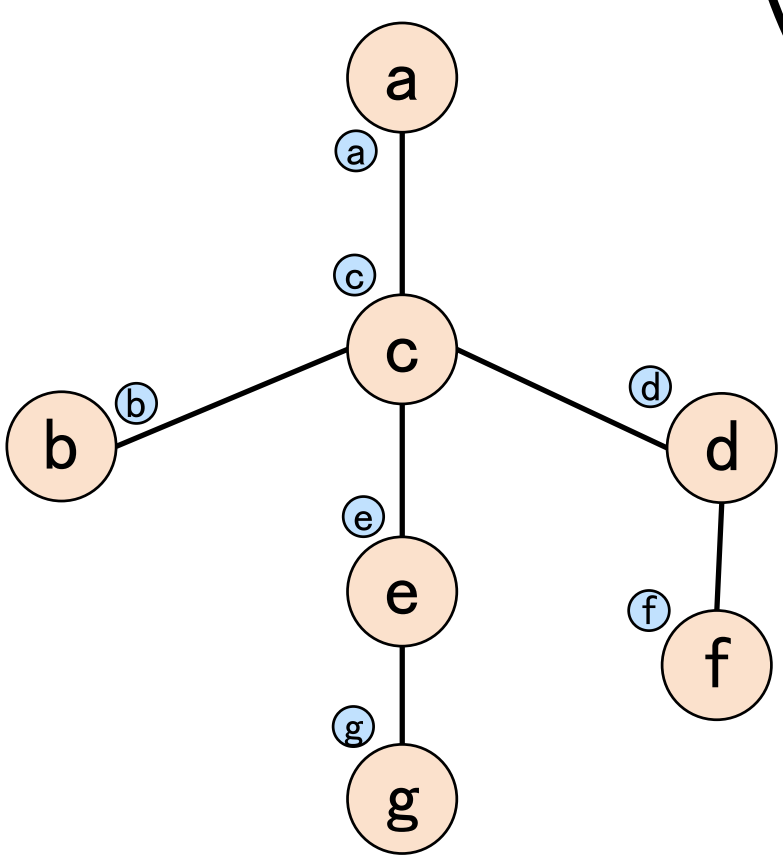
……

$$\sum_{u\in V} \alpha_u |u\rangle |0\rangle$$

<u>Example</u>:

V={a,b,c,d,e,f,g}

here leader = node a



Initially node a owns $\sum_{u\in V} \alpha_u |u\rangle_a$

1. "<u>Broadcast</u>" this state, which gives [ecc(a) ≤ D rounds]

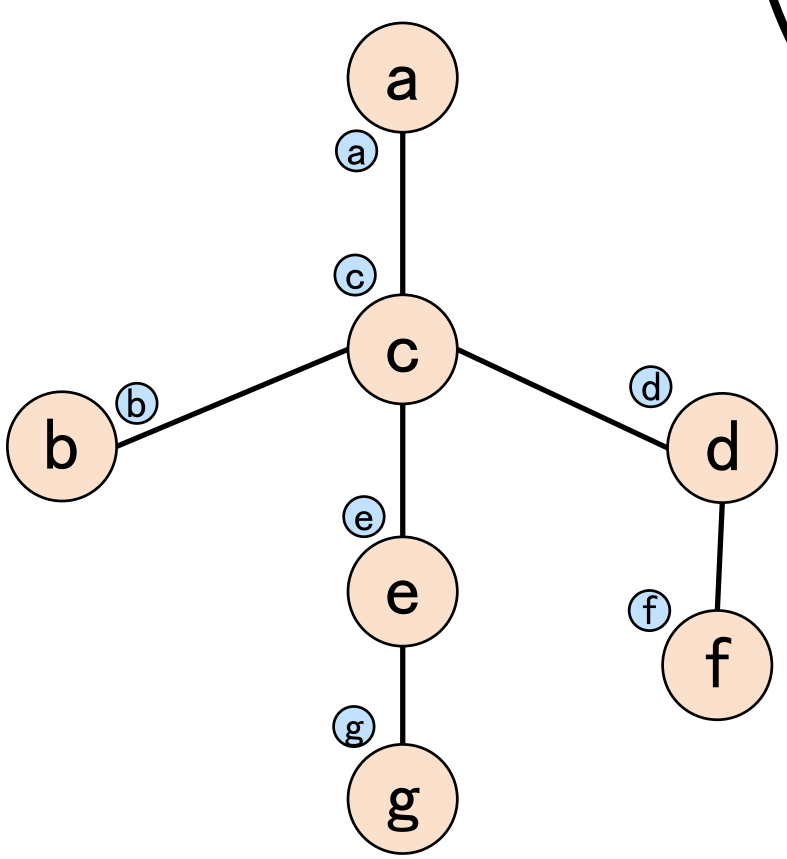
$$\sum_{u\in V} \alpha_u |u\rangle_a |u\rangle_b |u\rangle_c |u\rangle_d |u\rangle_e |u\rangle_f |u\rangle_g$$

Implementati

$$\sum_{u\in V}\alpha_u|u\rangle|0\rangle$$

Node a introduces 1 register	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|0\rangle$
Node a applies CNOTS	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle$
Node a sends the second register to c	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}$
Node c introduces 3 registers	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|0\rangle\ |0\rangle\ |0\rangle$
Node c applies CNOTS	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|u\rangle\ |u\rangle\ |u\rangle$
Node c sends the registers to b,e,d	$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{d}}$
……	

Example:
V={a,b,c,d,e,f,g}
here leader = node a



Initially node a owns $\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}$

1. "Broadcast" this state, which gives [ecc(a) ≤ D rounds]

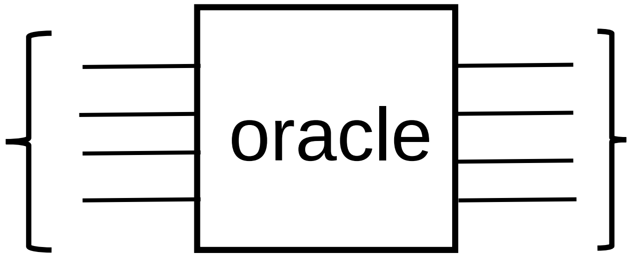
$$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{d}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{f}}|u\rangle_{\mathrm{g}}$$

2. The nodes implement the classical protocol [O(D) rounds] for computing the eccentricity of u, which gives

$$\sum_{u\in V}\alpha_u|u\rangle_{\mathrm{a}}|u\rangle_{\mathrm{b}}|u\rangle_{\mathrm{c}}|u\rangle_{\mathrm{d}}|u\rangle_{\mathrm{e}}|u\rangle_{\mathrm{f}}|u\rangle_{\mathrm{g}}|ecc(u)\rangle_{\mathrm{a}}$$

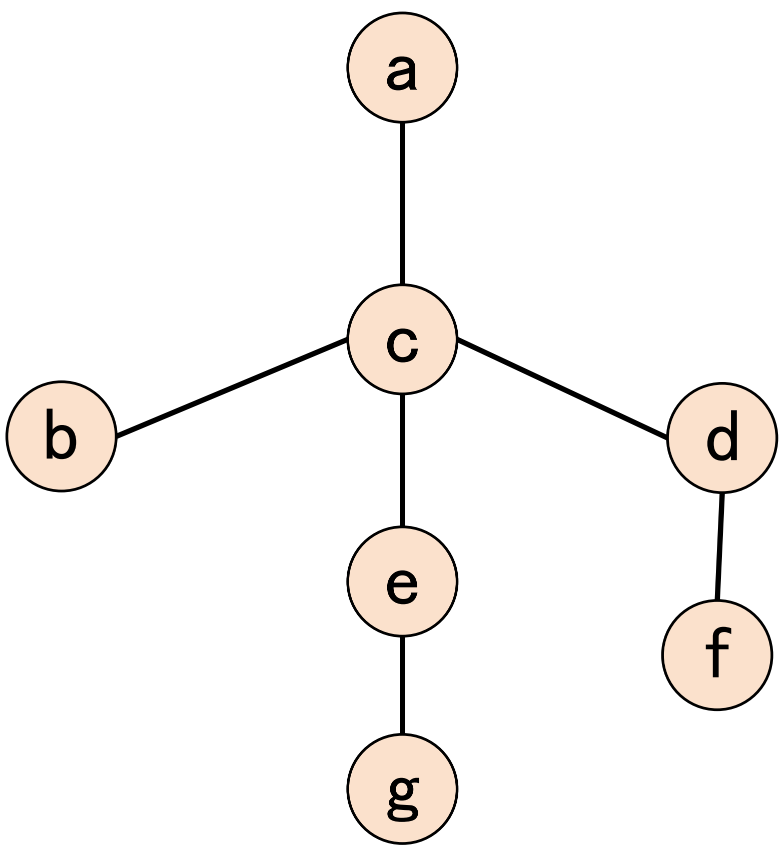
Implementation of the Oracle in O(D) rounds

$$\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |0\rangle_{\mathrm{a}}$$



$$\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |ecc(u)\rangle_{\mathrm{a}}$$

V={a,b,c,d,e,f,g}



Initially node a owns $\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}}$

1. "Broadcast" this state, which gives [ecc(a) ≤ D rounds]

$$\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{b}} |u\rangle_{\mathrm{c}} |u\rangle_{\mathrm{d}} |u\rangle_{\mathrm{e}} |u\rangle_{\mathrm{f}} |u\rangle_{\mathrm{g}}$$

2. The nodes implement the classical protocol [O(D) rounds] for computing the eccentricity of u, which gives

$$\sum_{u\in V} \alpha_u |u\rangle_{\mathrm{a}} |u\rangle_{\mathrm{b}} |u\rangle_{\mathrm{c}} |u\rangle_{\mathrm{d}} |u\rangle_{\mathrm{e}} |u\rangle_{\mathrm{f}} |u\rangle_{\mathrm{g}} |ecc(u)\rangle_{\mathrm{a}}$$

3. The nodes revert Step 1 [ecc(a) ≤ D rounds]

Quantum Distributed Computation of the Diameter: Summary

Define the function f: V → {0,1} such that $f(u) = \begin{cases} 1 & \text{if } ecc(u) \geq d \\ 0 & \text{otherwise} \end{cases}$

Goal: find u such that f(u) = 1 (or report that no such vertex exist)

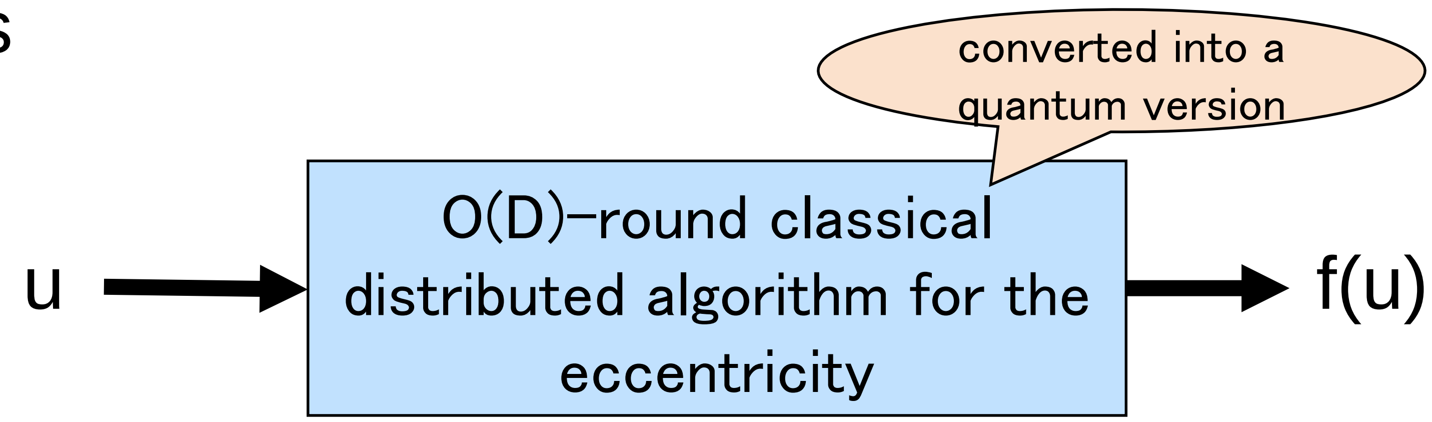
> There is a quantum algorithm for this search problem using $O(\sqrt{n})$ calls to a black box evaluating f

Quantum search [Grover 96]

<u>Quantum distributed algorithm computing the diameter</u>

- ✓ The network elects a leader
- ✓ The leader locally implements Grover algorithm. Each call to the black box is implemented by using the standard O(D)-round classical algorithm computing the eccentricity.

Complexity: $O(\sqrt{n} \times D)$ rounds



Quantum Distributed Computation of the Diameter: Summary

Define the function f: V → {0,1} such that $f(u) = \begin{cases} 1 \text{ if ecc}(u) \geq d \\ 0 \text{ otherwise} \end{cases}$

Goal: find u such that f(u) = 1 (or report that no such vertex exist)

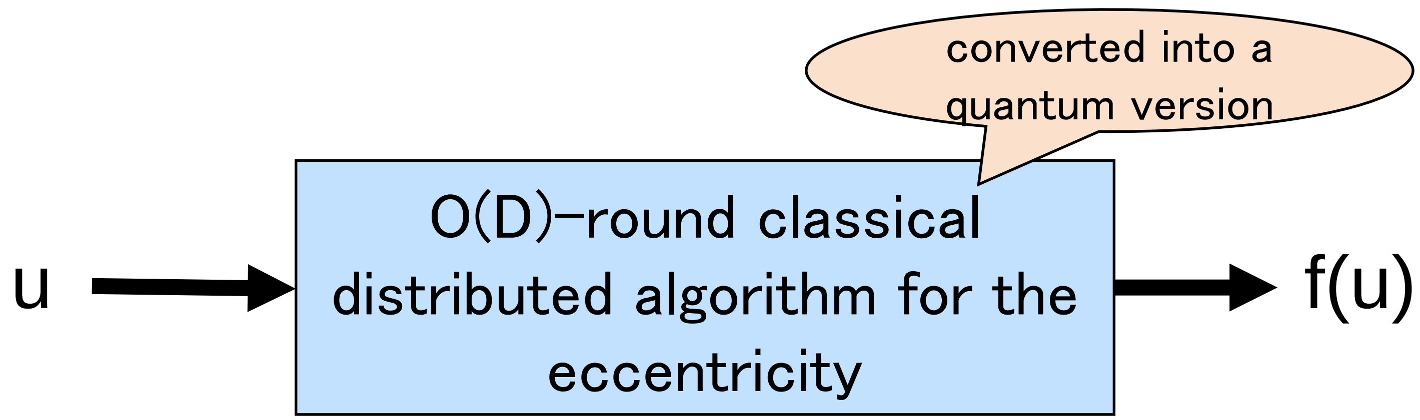
> There is a quantum algorithm for this search problem using $O(\sqrt{n})$ calls to a black box evaluating f

Quantum search [Grover 96]

Quantum distributed algorithm computing the diameter

- ✓ The network elects a leader
- ✓ The leader locally implements Grover algorithm. Each call to the black box is implemented by using the standard O(D)-round classical algorithm computing the eccentricity.

Complexity: $O(\sqrt{n} \times D)$ rounds



With further work, the complexity can be reduced to $O(\sqrt{nD})$ rounds

Quantum Distributed Computation of the Diameter: Summary

Classically in O(D) rounds it is possible to simultaneously compute the eccentricities of D vertices [Peleg+12]

Thus we can instead do a Grover search over groups of D vertices (there are n/D groups) in

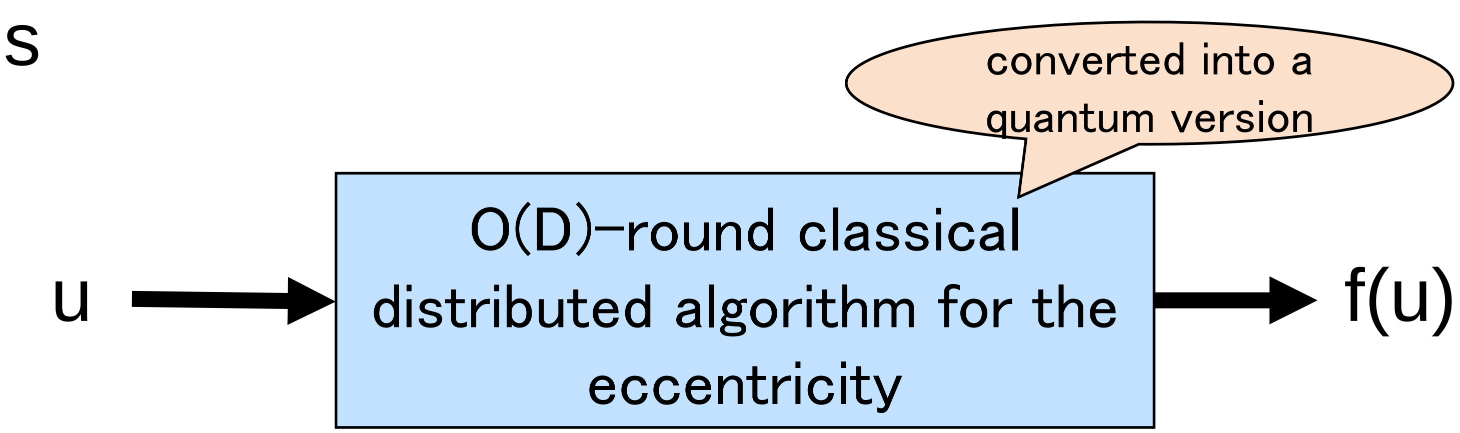
$$O(\sqrt{n/D} \times D) = O(\sqrt{nD}) \text{ rounds}$$

<u>Quantum distributed algorithm computing the diameter</u>

- ✓ The network elects a leader
- ✓ The leader locally implements Grover algorithm. Each call to the black box is implemented by using the standard O(D)-round classical algorithm computing the eccentricity.

Complexity: $O(\sqrt{n} \times D)$ rounds

With further work, the complexity can be reduced to $O(\sqrt{nD})$ rounds



Summary of the first part

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

	Classical	Quantum (our results)
Exact computation (upper bounds)	$O(n)$ [Holzer+12, Peleg+12]	$O(\sqrt{nD})$
Exact computation (lower bounds)	$\widetilde{\Omega}(n)$ [Frischknecht+12]	$\widetilde{\Omega}(\sqrt{n}+D)$ [unconditional] $\widetilde{\Omega}(\sqrt{nD})$ [conditional]

number of rounds needed to compute the diameter (n: number of nodes, D: diameter)

OPEN PROBLEM:

✓ Prove an unconditional lower bound of $\widetilde{\Omega}(\sqrt{nD})$ rounds

very recent result [Magniez, Nayak 2020]

$\widetilde{\Omega}(\sqrt{n}+n^{1/3}D^{2/3})$ [unconditional]

Summary of the first part

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

Our upper bound is obtained by showing how to implement quantum search in a distributed setting

PROMISING RESEARCH DIRECTION: find other applications of this technique

Summary of the first part

Main result [LG, Magniez 2018]

sublinear-round quantum computation of the diameter whenever D=o(n)

Our upper bound is obtained by showing how to implement quantum search in a distributed setting

PROMISING RESEARCH DIRECTION: find other applications of this technique

[Izumi, LG 2019]: quantum distributed algorithm for the All-Pairs Shortest Paths Problem faster than the best classical algorithms

- ✓ idea: implement simultaneously $\Theta(n^2)$ quantum distributed searches
- ✓ significant work needed to avoid congestions in the checking procedures

[Izumi, LG, Magniez 2020]: quantum distributed algorithm for triangle finding faster than the best classical algorithms

Quantum Distributed Computing

Quantum distributed computing

Now qubits can be sent instead of bits

(no prior entanglement between nodes)

n: number of nodes of the network

CONGEST model: only O(log n) qubits per message

Quantum can be useful for some problems
[LG, Magniez 2018] [Izumi, LG 2019] [Izumi et al. 2020]

LOCAL model: no restriction on the size of each message

Quantum Distributed Computing

Quantum distributed computing

Now qubits can be sent instead of bits

(no prior entanglement between nodes)

n: number of nodes of the network

CONGEST model: only O(log n) qubits per message

Quantum can be useful for some problems

[LG, Magniez 2018] [Izumi, LG 2019] [Izumi et al. 2020]

LOCAL model: no restriction on the size of each message

unbounded amount of quantum communication
vs.
unbounded amount of classical communication

Quantum Distributed Computing

Quantum distributed computing

Now qubits can be sent instead of bits

(no prior entanglement between nodes)

n: number of nodes of the network

CONGEST model: only O(log n) qubits per message

Quantum can be useful for some problems
[LG, Magniez 2018] [Izumi, LG 2019] [Izumi et al. 2020]

LOCAL model: no restriction on the size of each message

[Gavoille et al. 09] ⇨ There is a computational problem that can be solved in 1 round in the quantum LOCAL model but requires 2 rounds classically.

Quantum Distributed Computing

Quantum distributed computing

Now qubits can be sent instead of bits

(no prior entanglement between nodes)

n: number of nodes of the network

CONGEST model: only O(log n) qubits per message

Quantum can be useful for some problems
[LG, Magniez 2018] [Izumi, LG 2019] [Izumi et al. 2020]

LOCAL model: no restriction on the size of each message

[Gavoille et al. 09] ⇒ There is a computational problem that can be solved in 1 round in the quantum LOCAL model but requires 2 rounds classically.

[LG, Nishimura, Rosmanis 2019] ⇒ There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires $\Omega(n)$ rounds classically.

Superiority of the Quantum LOCAL model

[LG, Nishimura, Rosmanis 2019]

There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires $\Omega(n)$ rounds classically.

Superiority of the Quantum LOCAL model

We use a construction from [Barrett, Caves, Eastin, Elliot, Pironio 2007]

[LG, Nishimura, Rosmanis 2019]

There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires Ω(n) rounds classically.

Superiority of the Quantum LOCAL model

Also used in some of the recent results on quantum shallow circuits [Bravyi, Gosset, König 2018]

We use a construction from [Barrett, Caves, Eastin, Elliot, Pironio 2007]

[LG, Nishimura, Rosmanis 2019]

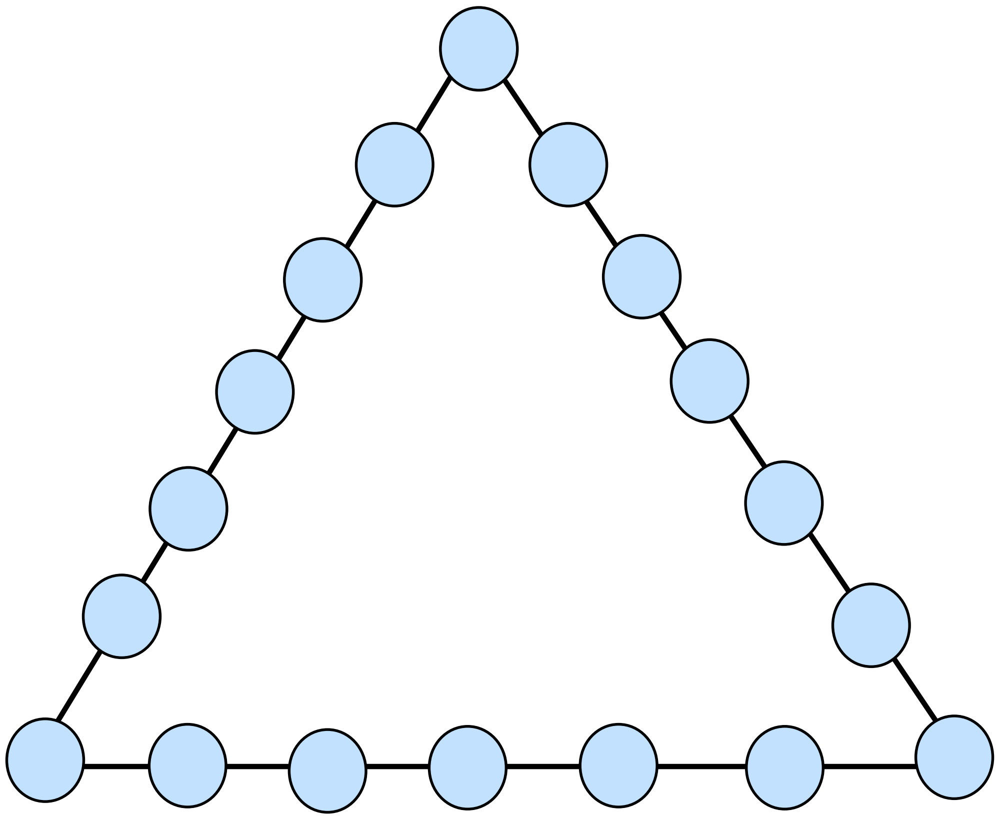
There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires Ω(n) rounds classically.

Superiority of the Quantum LOCAL model

multiple of 3

n=18

Consider a ring of size n (seen as a triangle)



We use a construction from [Barrett, Caves, Eastin, Elliot, Pironio 2007]

[LG, Nishimura, Rosmanis 2019]

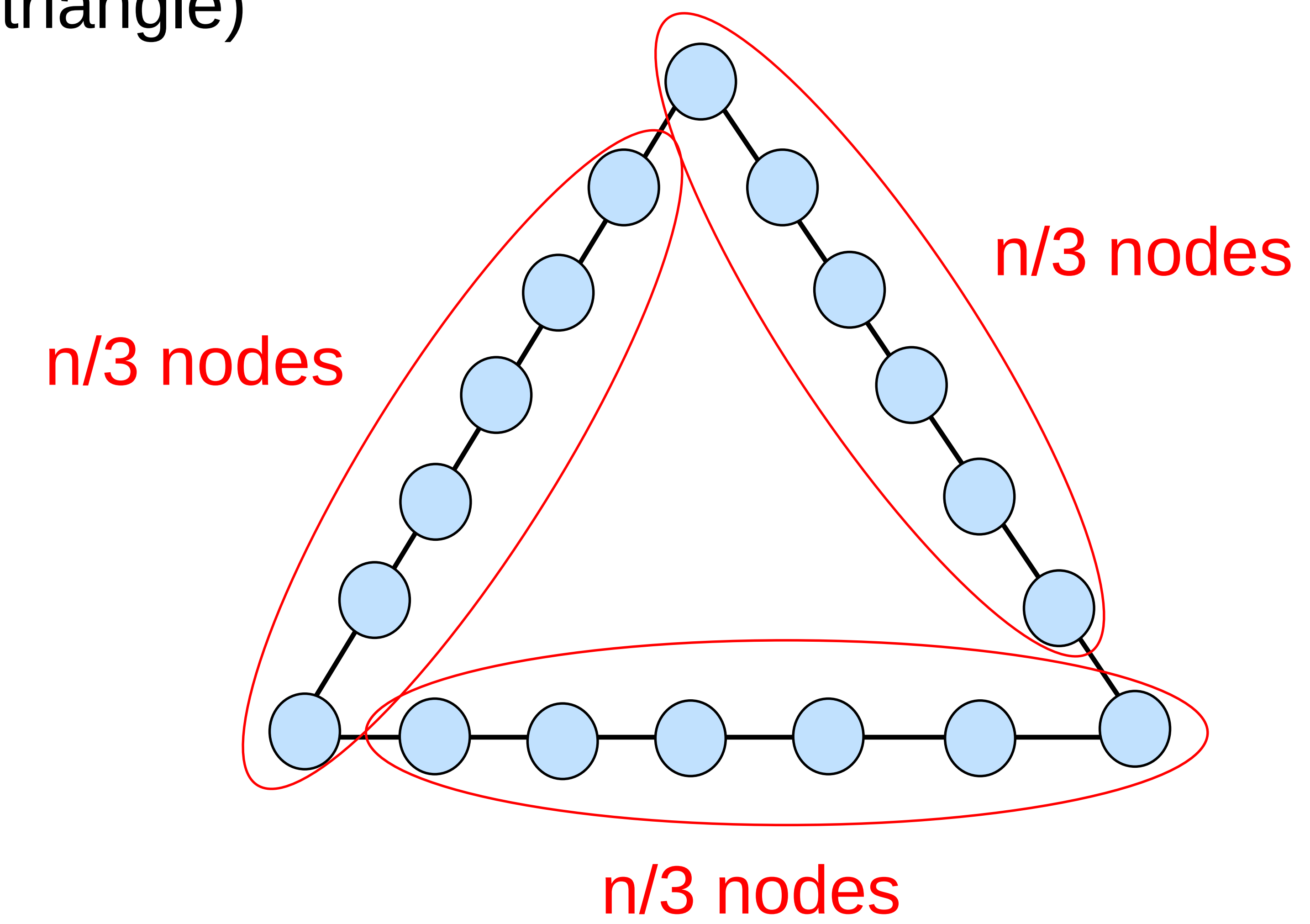
There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires $\Omega(n)$ rounds classically.

Superiority of the Quantum LOCAL model

multiple of 3

n=18

Consider a ring of size n (seen as a triangle)



We use a construction from [Barrett, Caves, Eastin, Elliot, Pironio 2007]

[LG, Nishimura, Rosmanis 2019]

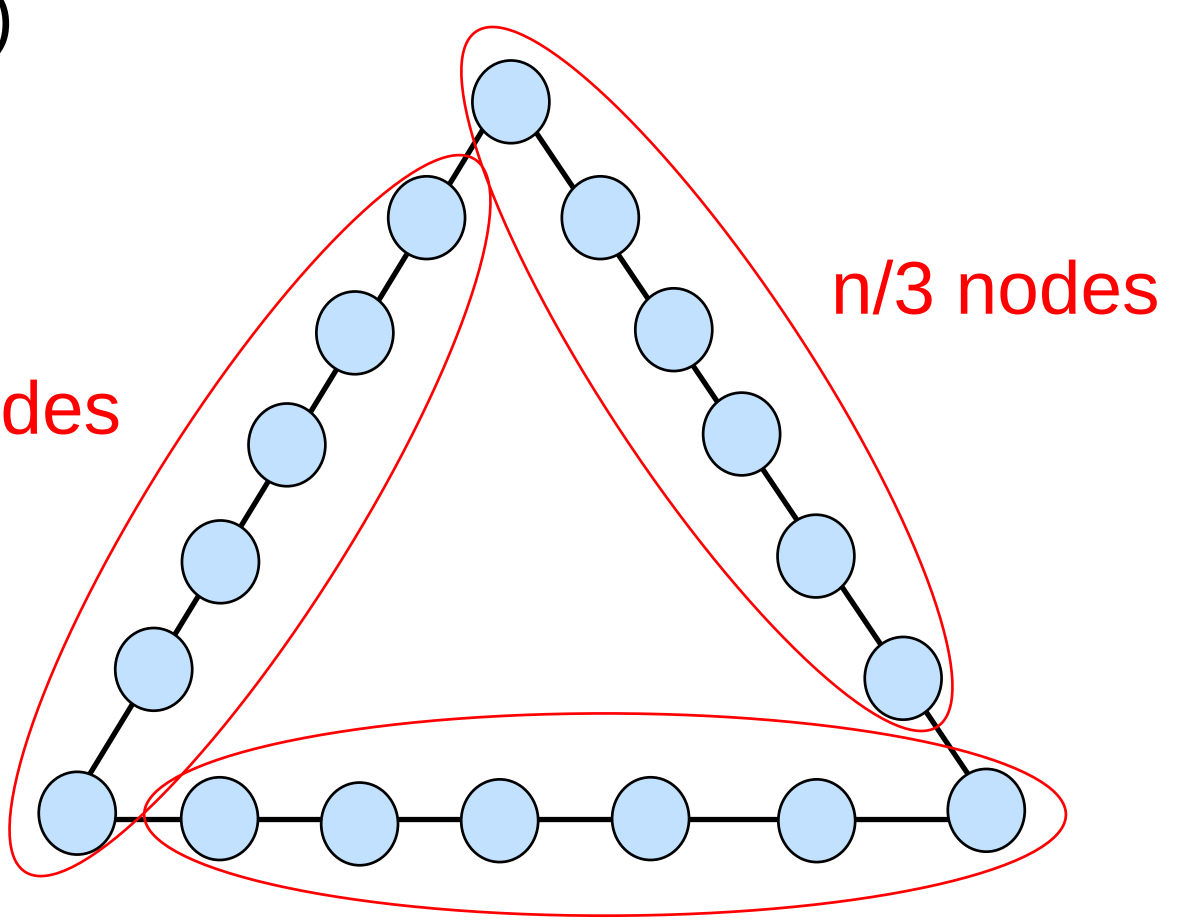
There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires $\Omega(n)$ rounds classically.

Superiority of the Quantum LOCAL model

multiple of 3

Consider a ring of size n (seen as a triangle)

Each “corner” gets a bit as input



We use a construction from [Barrett, Caves, Eastin, Elliot, Pironio 2007]

[LG, Nishimura, Rosmanis 2019]

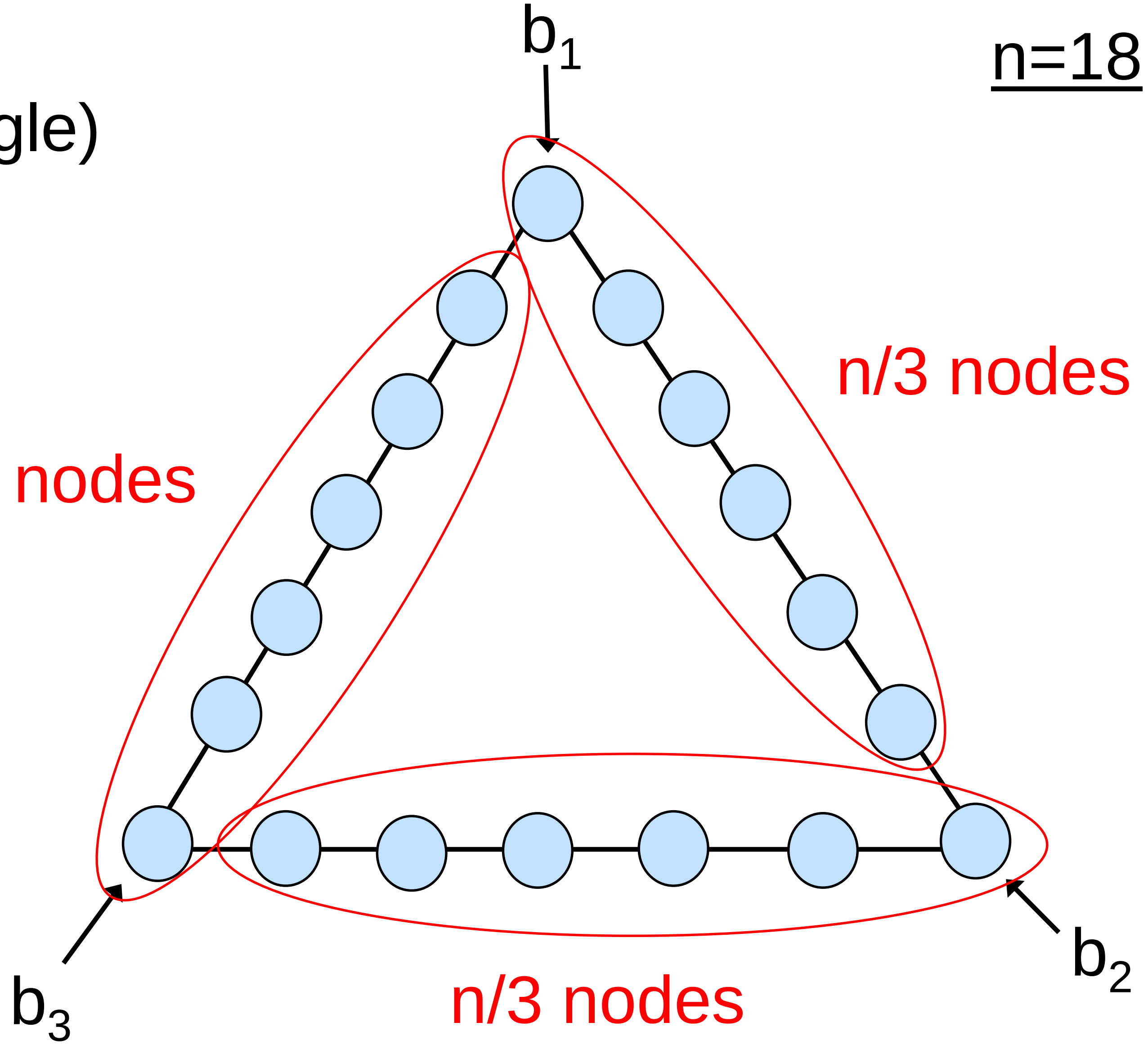
There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires $\Omega(n)$ rounds classically.

Superiority of the Quantum LOCAL model

multiple of 3

Consider a ring of size n (seen as a triangle)

Each "corner" gets a bit as input



We use a construction from [Barrett, Caves, Eastin, Elliot, Pironio 2007]

[LG, Nishimura, Rosmanis 2019]

There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires $\Omega(n)$ rounds classically.

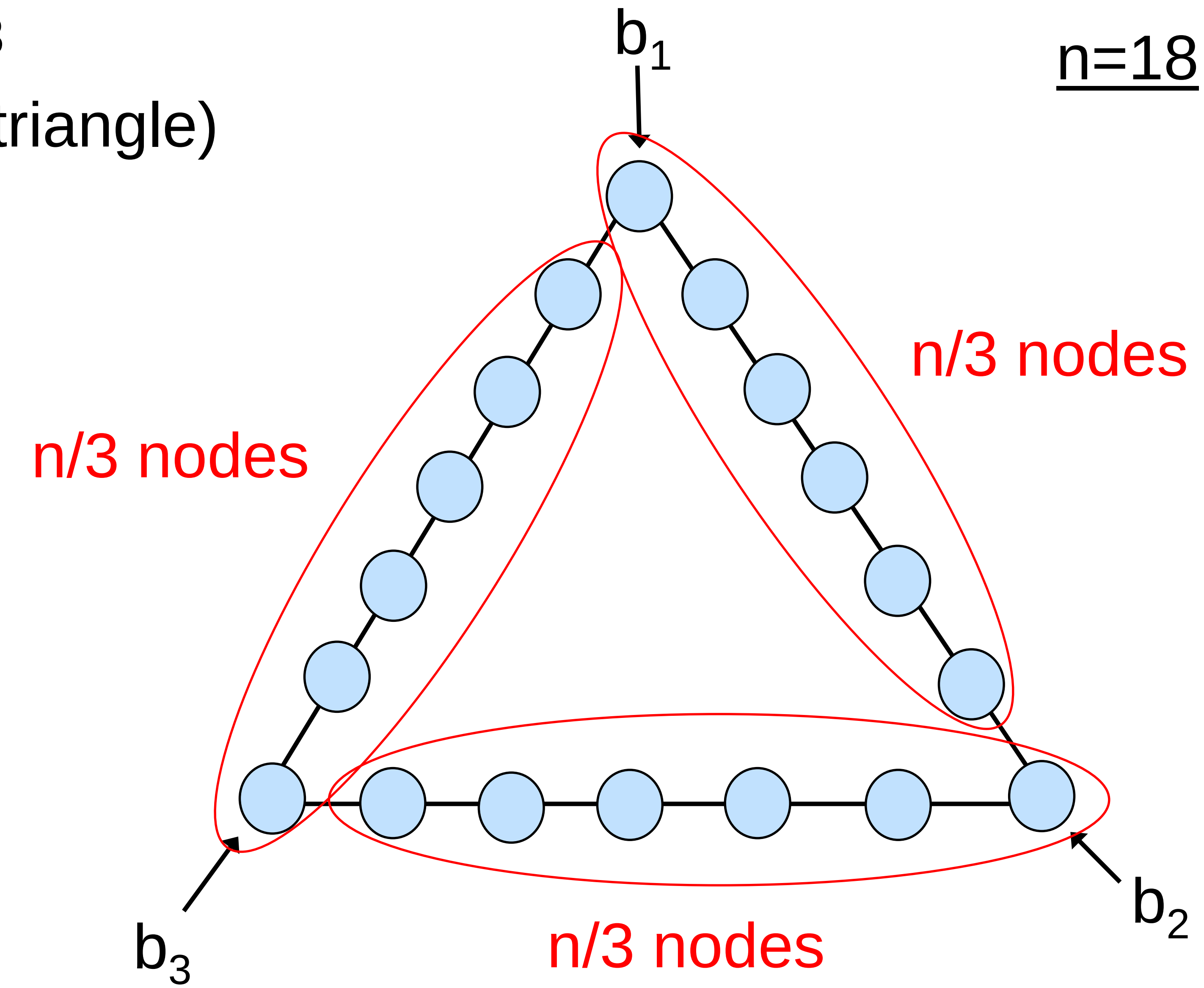
Superiority of the Quantum LOCAL model

multiple of 3

Consider a ring of size n (seen as a triangle)

Each "corner" gets a bit as input

Each node will output one bit



We use a construction from [Barrett, Caves, Eastin, Elliot, Pironio 2007]

[LG, Nishimura, Rosmanis 2019]

There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires $\Omega(n)$ rounds classically.

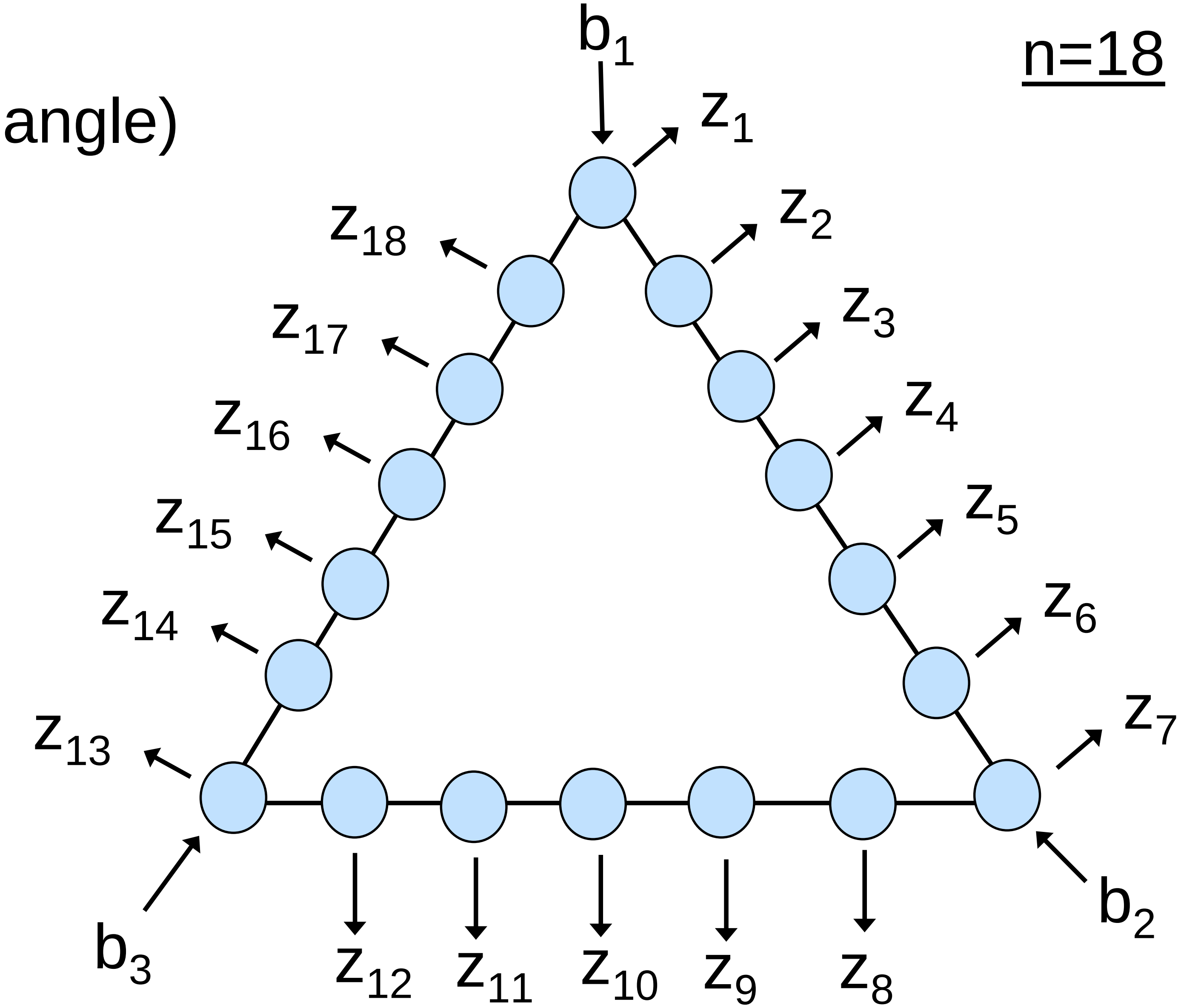
Superiority of the Quantum LOCAL model

multiple of 3

Consider a ring of size n (seen as a triangle)

Each “corner” gets a bit as input

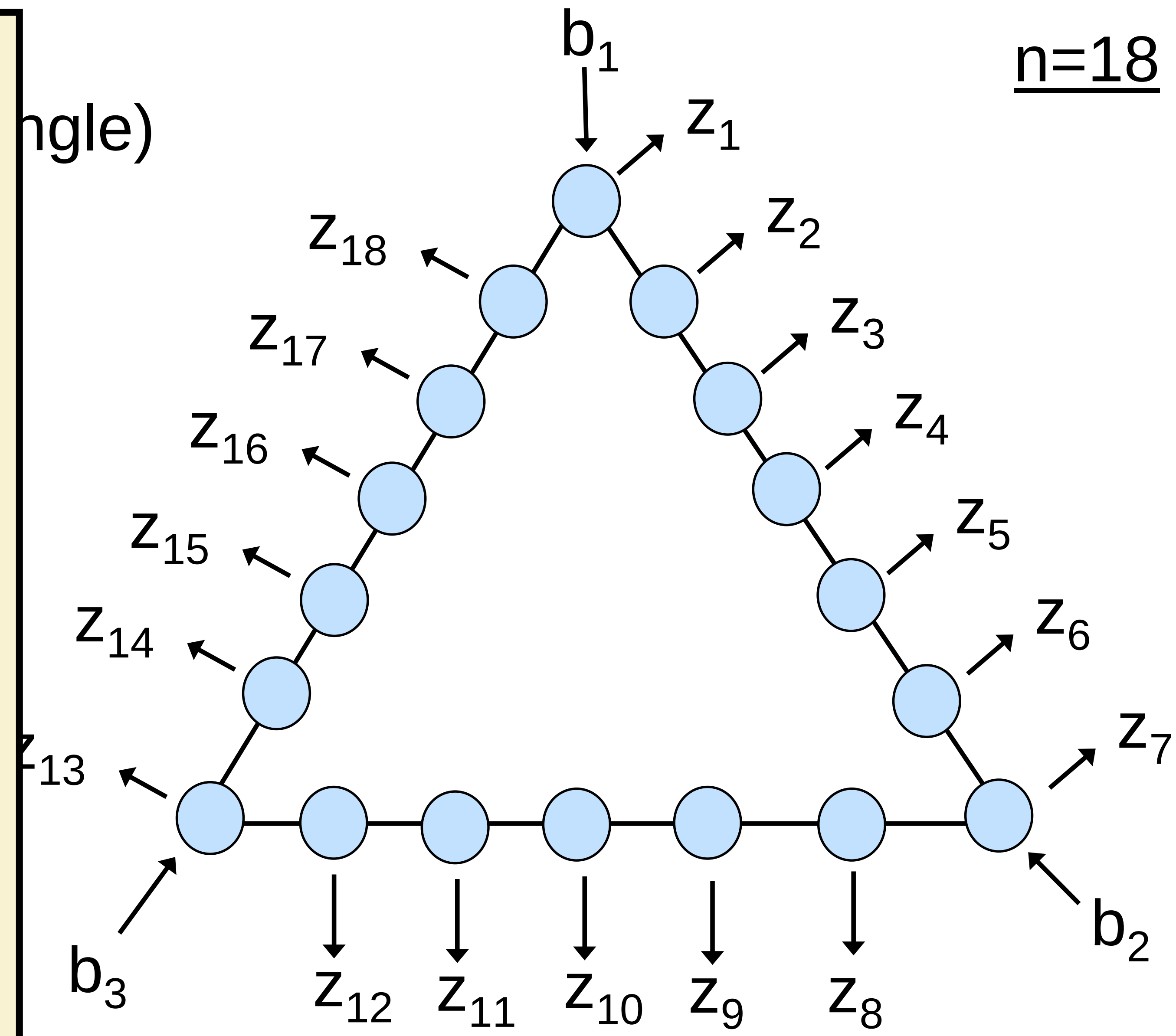
Each node will output one bit



Consider the following process:

1. The nodes prepare the graph state corresponding to the whole triangle
2. Each non-corner node measures its qubit in the X basis and then outputs the bit corresponding to the measurement outcome
3. Each corner node measures its qubit in the X basis if its input bit is 0, or measures it in the Y basis if its input bit is 1, and then outputs the bit corresponding to the measurement outcome

ngle)

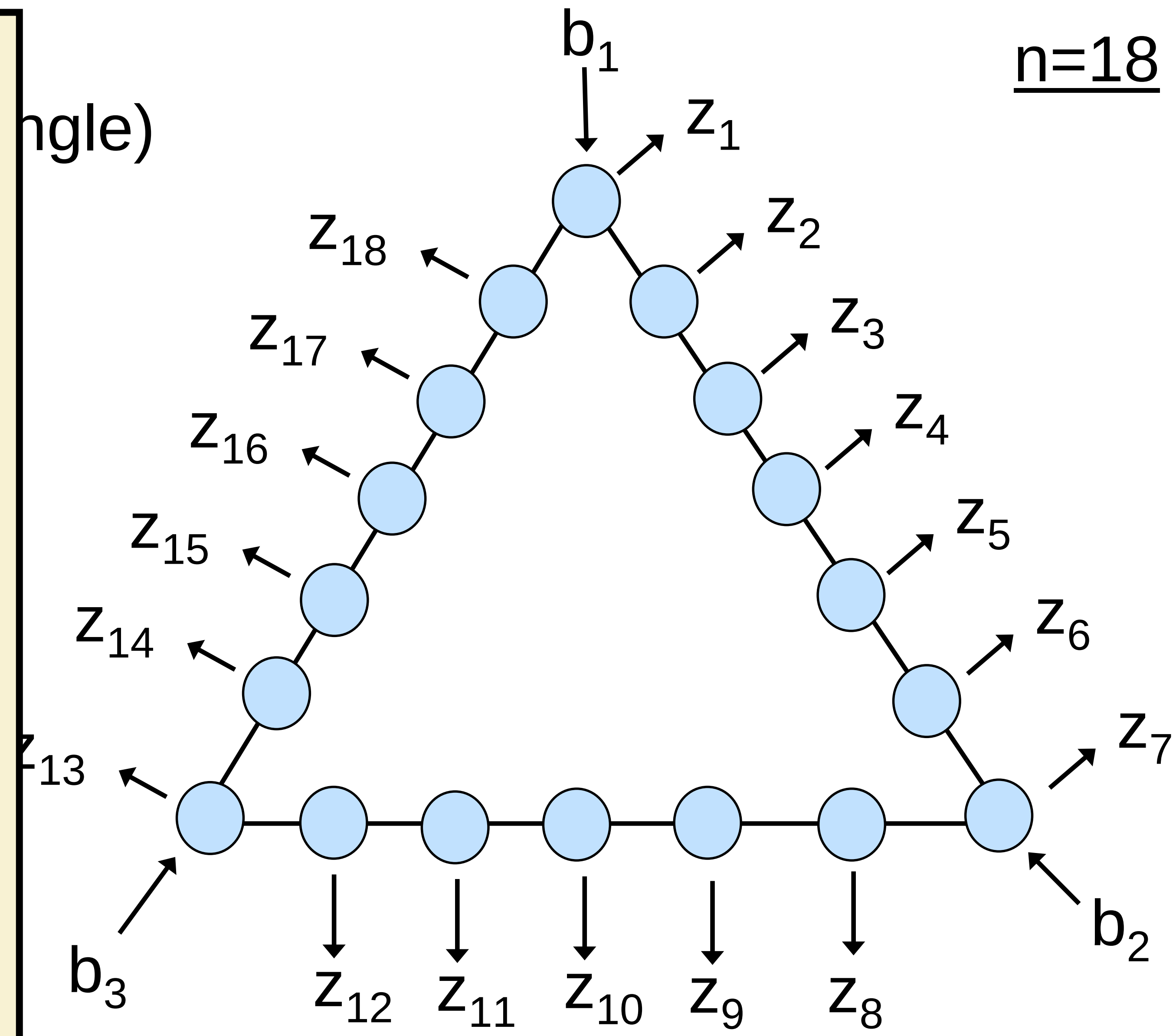


Consider the following process:

1. The nodes prepare the graph state corresponding to the whole triangle (this can be done in 2 rounds of quantum communication)

2. Each non-corner node measures its qubit in the X basis and then outputs the bit corresponding to the measurement outcome (no communication)

3. Each corner node measures its qubit in the X basis if its input bit is 0, or measures it in the Y basis if its input bit is 1, and then outputs the bit corresponding to the measurement outcome (no communication)

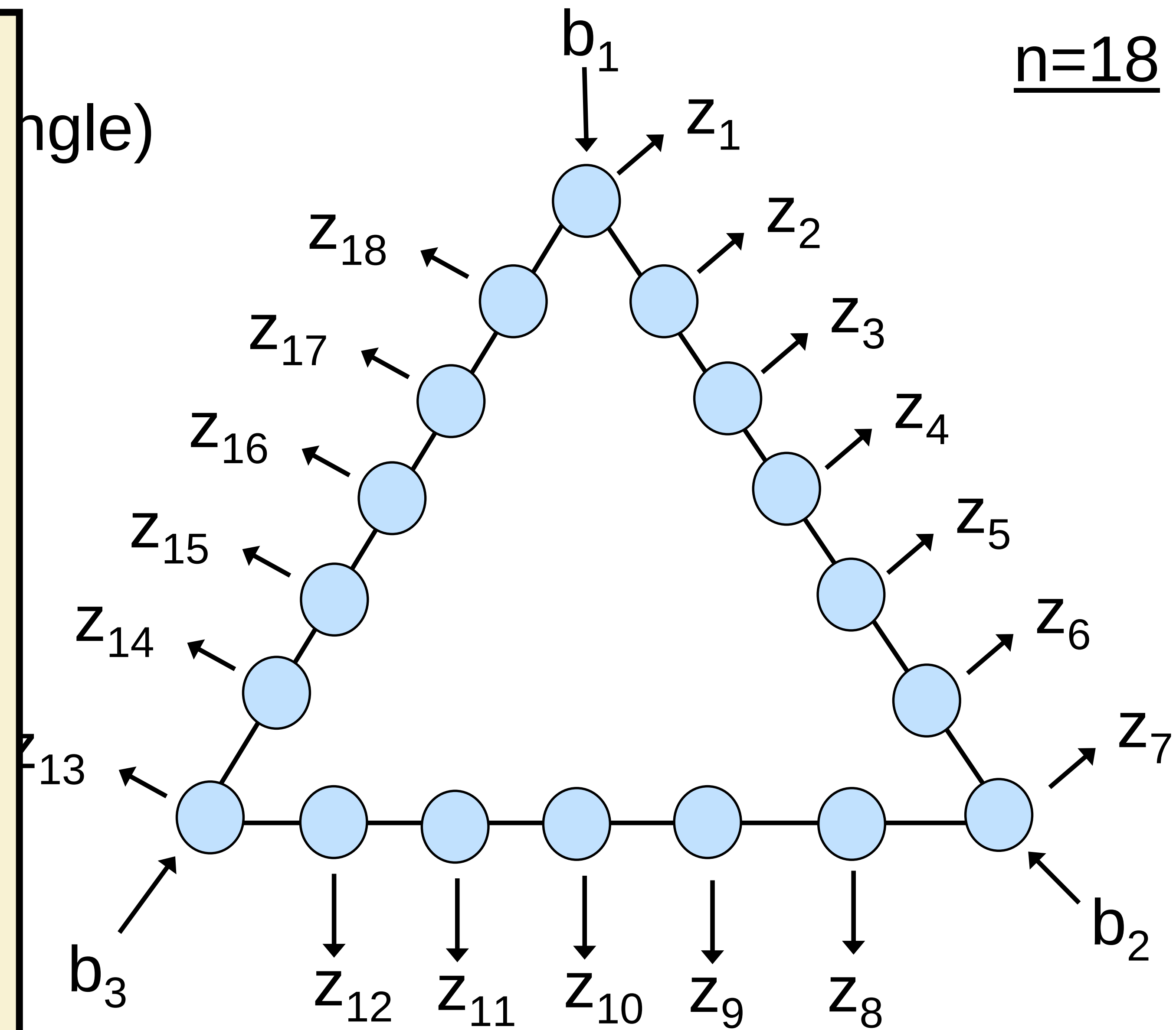


Consider the following process:

1. The nodes prepare the graph state corresponding to the whole triangle (this can be done in 2 rounds of quantum communication)

2. Each non-corner node measures its qubit in the X basis and then outputs the bit corresponding to the measurement outcome (no communication)

3. Each corner node measures its qubit in the X basis if its input bit is 0, or measures it in the Y basis if its input bit is 1, and then outputs the bit corresponding to the measurement outcome (no communication)



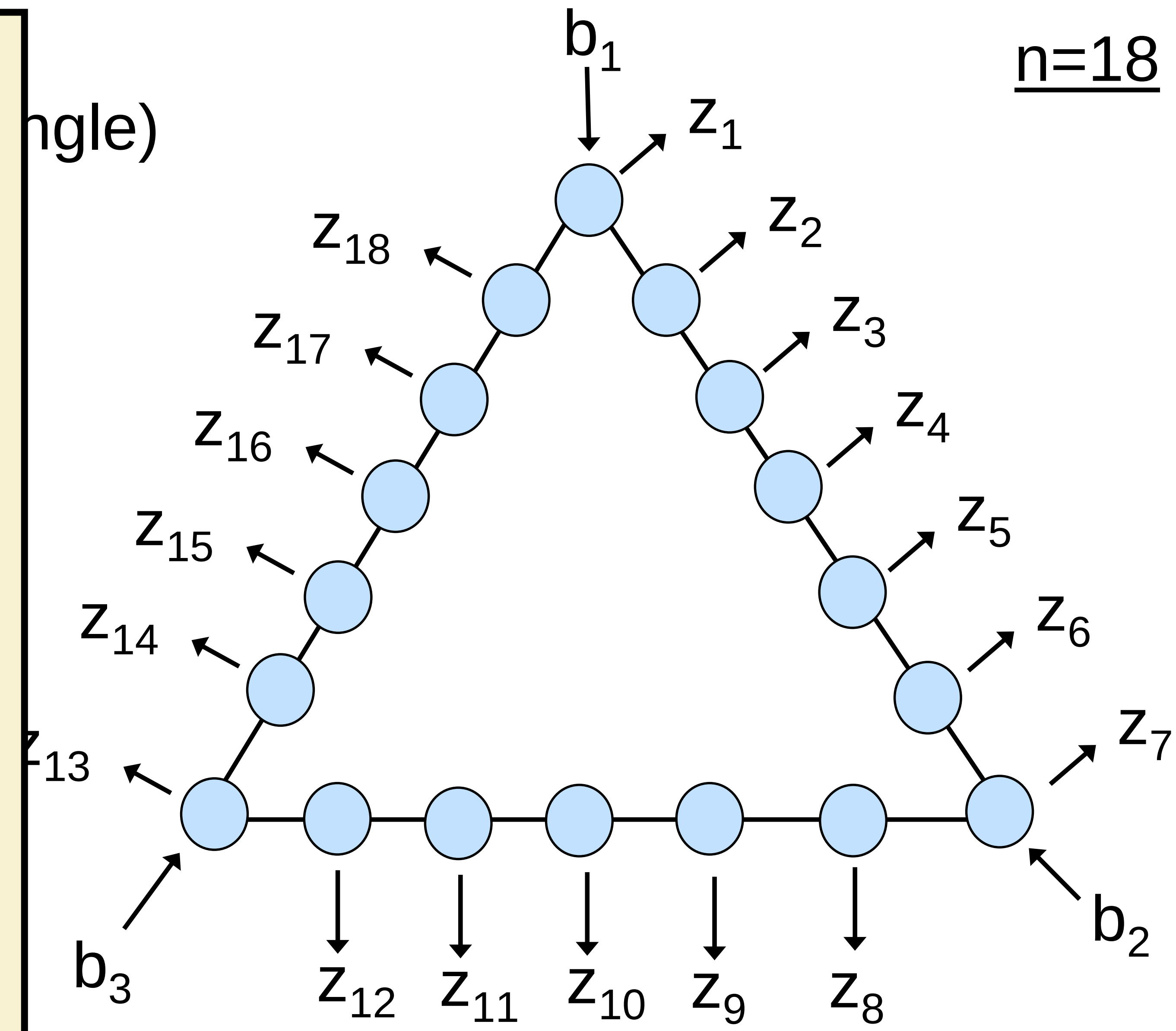
Claim:
[Barrett et al. 2007]

In the LOCAL model, any classical algorithm that samples (even approximately) from the same distribution must use at least n/6 rounds.

Consider the following process:

1. The nodes prepare the graph state corresponding to the whole triangle (this can be done in 2 rounds of quantum communication)
2. Each non-corner node measures its qubit in the X basis and then outputs the bit corresponding to the measurement outcome (no communication)
3. Each corner node measures its qubit in the X basis if its input bit is 0, or measures it in the Y basis if its input bit is 1, and then outputs the bit corresponding to the measurement outcome (no communication)

ngle)



Claim:
[Barrett et al. 2007]

In the LOCAL model, any classical algorithm that samples (even approximately) from the same distribution must use at least n/6 rounds.

Remarks

“simulate the outcome distribution of a measurement of the graph state”

[LG, Nishimura, Rosmanis 2019]

There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires $\Omega(n)$ rounds in the classical LOCAL model.

Remarks

✓ Since our quantum distributed algorithm only uses short messages (1 qubit in each message) we get the following stronger statement:

> There is a computational problem that can be solved in 2 rounds in the quantum CONGEST model but requires Ω(n) rounds in the classical LOCAL model.

"simulate the outcome distribution of a measurement of the graph state"

[LG, Nishimura, Rosmanis 2019] ⇨

> There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires Ω(n) rounds in the classical LOCAL model.

Remarks

- ✓ Since our quantum distributed algorithm only uses short messages (1 qubit in each message) we get the following stronger statement:

> There is a computational problem that can be solved in 2 rounds in the quantum CONGEST model but requires Ω(n) rounds in the classical LOCAL model.

- ✓ A similar separation can be shown for a relation ("output any outcome that appears with non-zero probability as an outcome of the measurement of the graph state")

"simulate the outcome distribution of a measurement of the graph state"

[LG, Nishimura, Rosmanis 2019] ⇨

> There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires Ω(n) rounds in the classical LOCAL model.

Remarks

- ✓ Since our quantum distributed algorithm only uses short messages (1 qubit in each message) we get the following stronger statement:

> There is a computational problem that can be solved in 2 rounds in the quantum CONGEST model but requires Ω(n) rounds in the classical LOCAL model.

- ✓ A similar separation can be shown for a relation ("output any outcome that appears with non-zero probability as an outcome of the measurement of the graph state")
- ✓ A similar separation can also be shown for a sampling problem without any input ("simulate the outcome distribution of the measurement when the bits b_1, b_2 and b_3 are taken uniformly at random")

"simulate the outcome distribution of a measurement of the graph state"

[LG, Nishimura, Rosmanis 2019] ⇨

> There is a computational problem that can be solved in 2 rounds in the quantum LOCAL model but requires Ω(n) rounds in the classical LOCAL model.

Conclusion

We now know that quantum distributed algorithms can be faster than classical distributed algorithms for several problems, in both the CONGEST model and the LOCAL model

Interesting research directions:

- ✓ Construct other quantum distributed algorithms, for important problems

  Designing quantum distributed algorithms in these models poses new algorithmic challenges since we have to focus on the round complexity (instead of time/query complexity or total communication complexity)

- ✓ Develop lower bounds techniques, especially in the quantum LOCAL model

  Can we show a nontrivial lower bound for graph coloring?

- ✓ Prove the superiority of quantum distributed algorithms in other models

Recent result: [Fraigniaud, LG, Nishimura, Paz 2020]

advantage for distributed interactive proofs