Distributed Verification and Hardness of Distributed Approximation

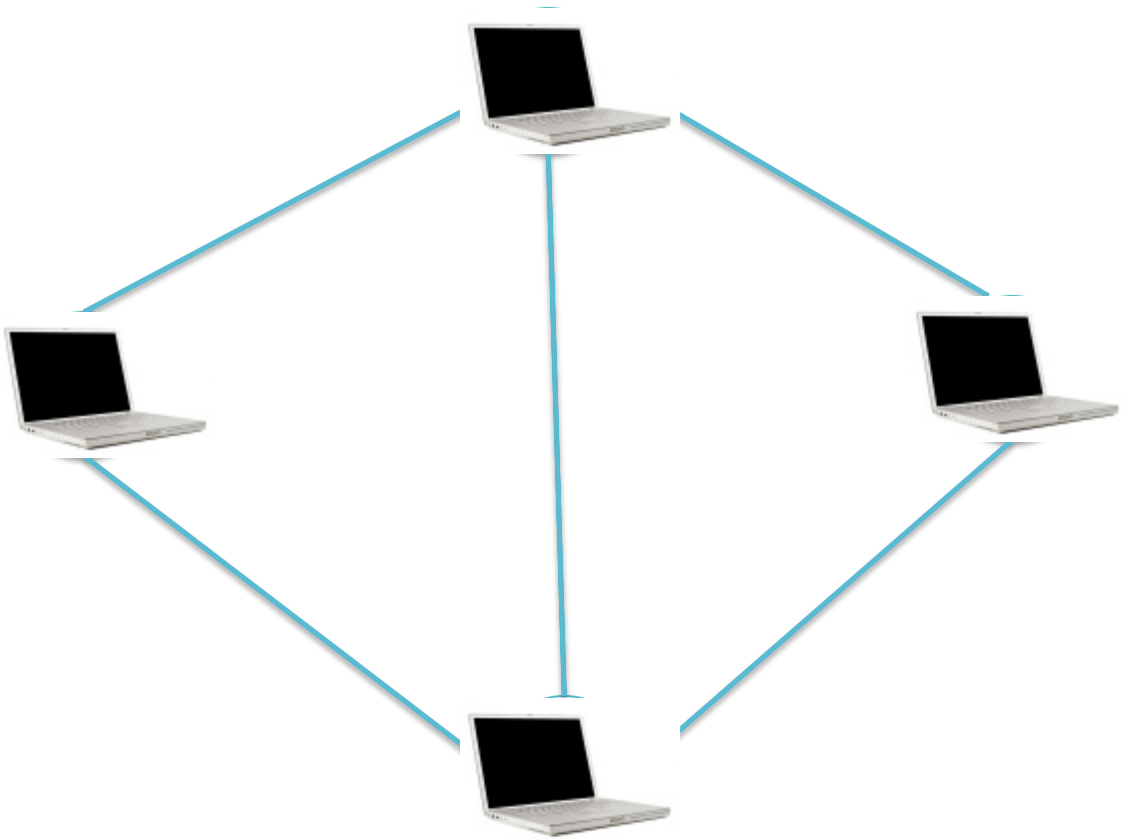
Danupon Nanongkai
KTH

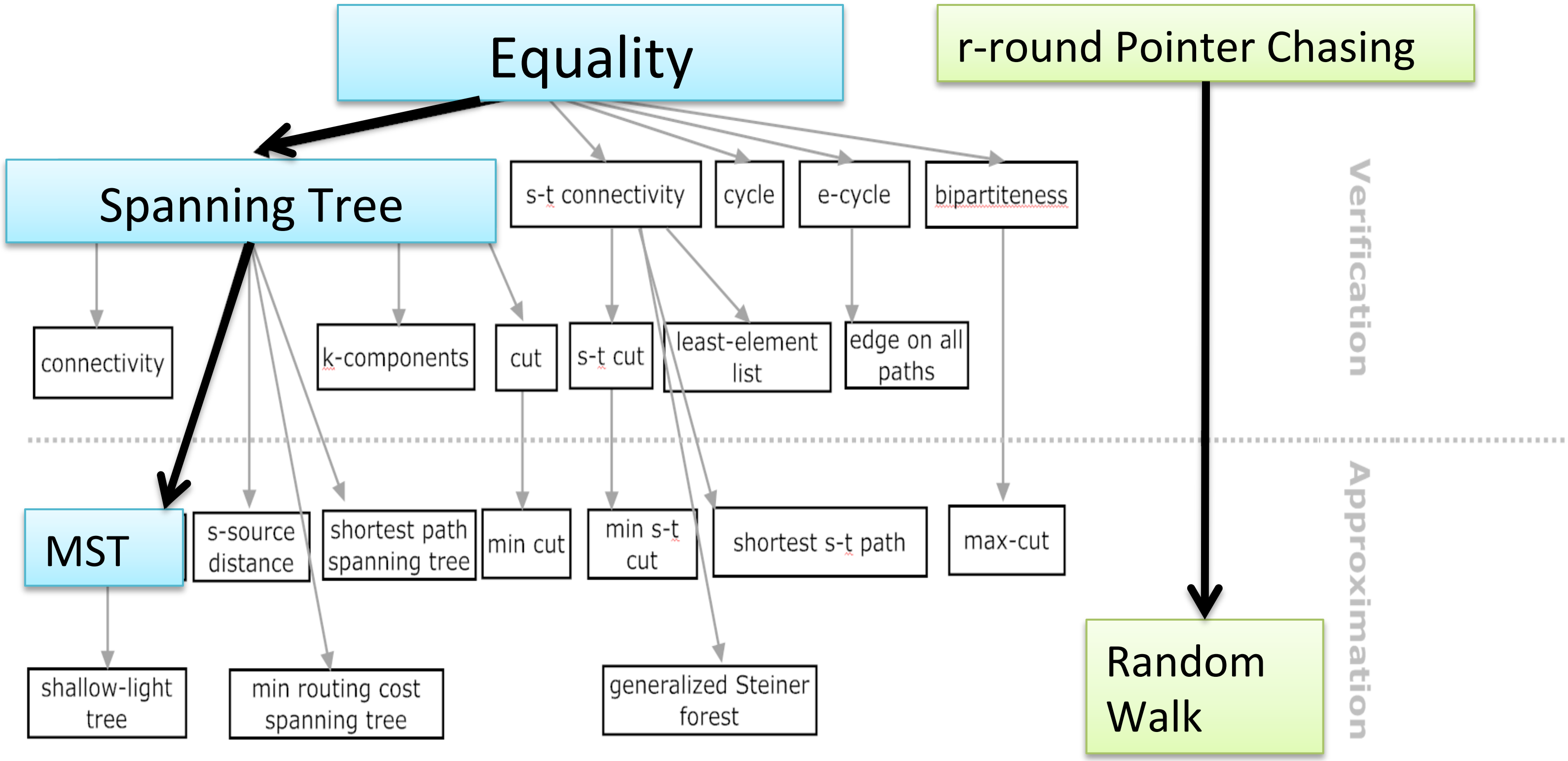
<u>Based on</u>

Distributed Verification and Hardness of Distributed Approximation, STOC 2011 & SICOMP 2012,

with Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Gopal Pandurangan, David Peleg, Roger Wattenhofer

A distributed network

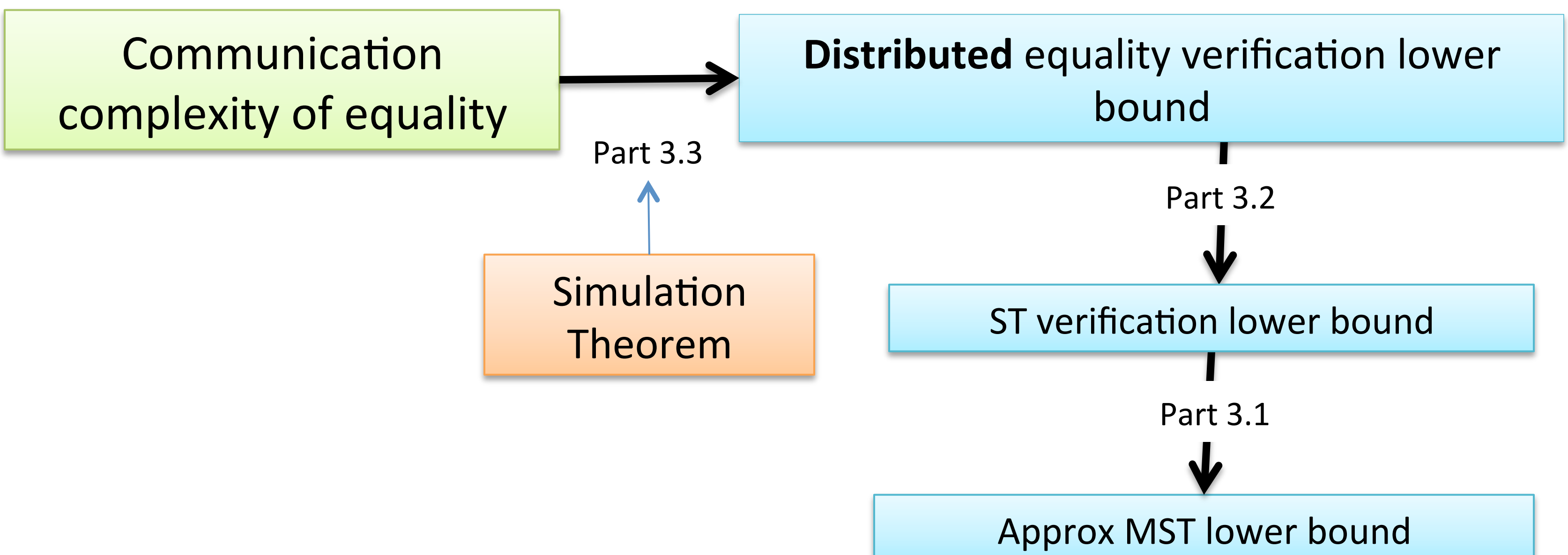




Theorem: Above problems require $\Omega(n^{1/2}+D)$ time to verify/approximate

Roadmap

- **Part 1:** The model of distributed computing
- **Part 2:** Introduction to MST and ST verification
- **Part 3:** Proof of the hardness of approx. MST

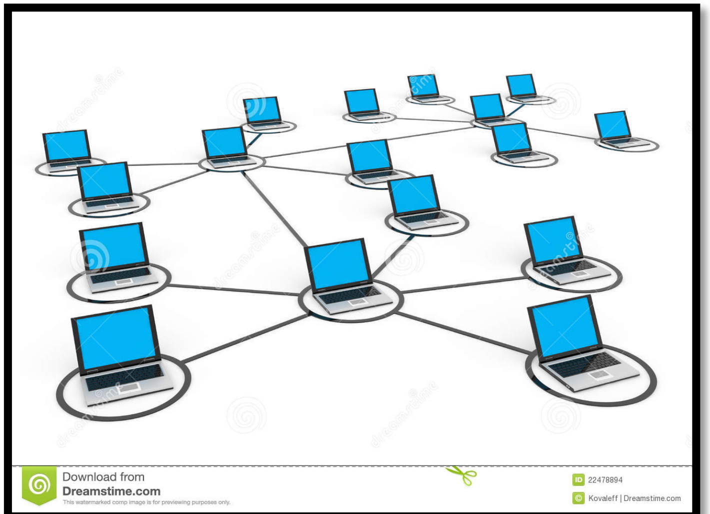


- **Part 4:** After 2011

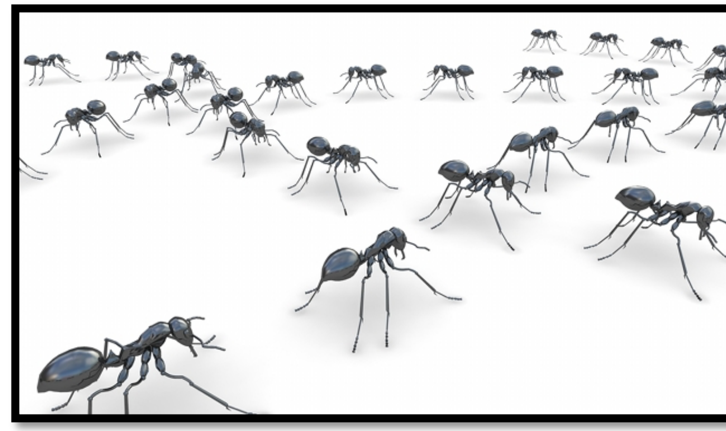
Part 1

Theory of Distributed Computing 101

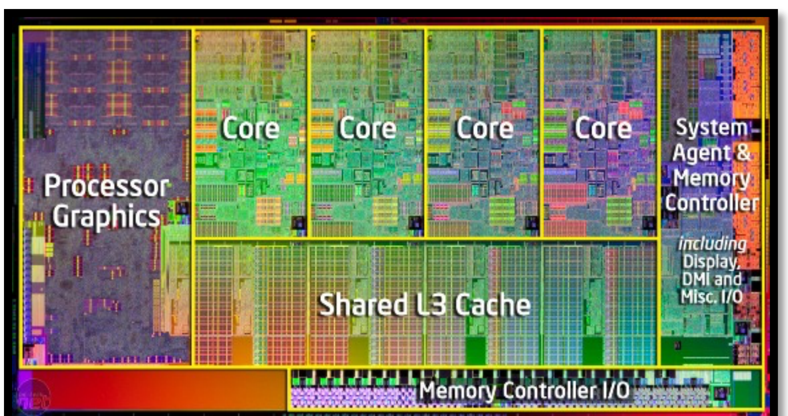
Distributed Computing



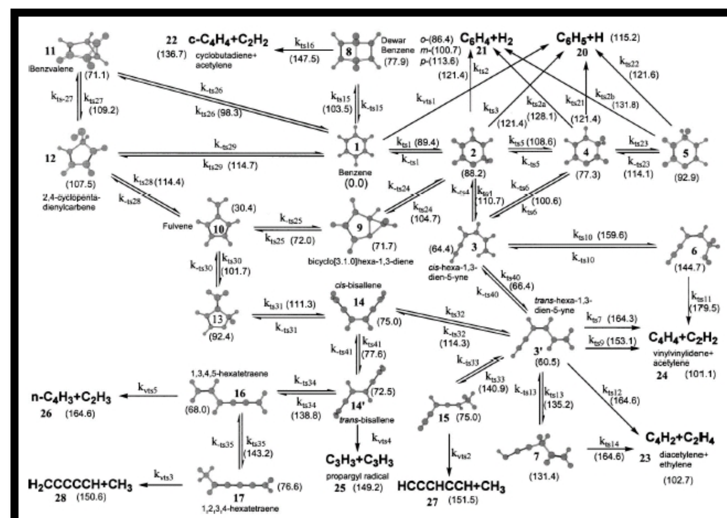
Communication Network



Ant Contact Networks

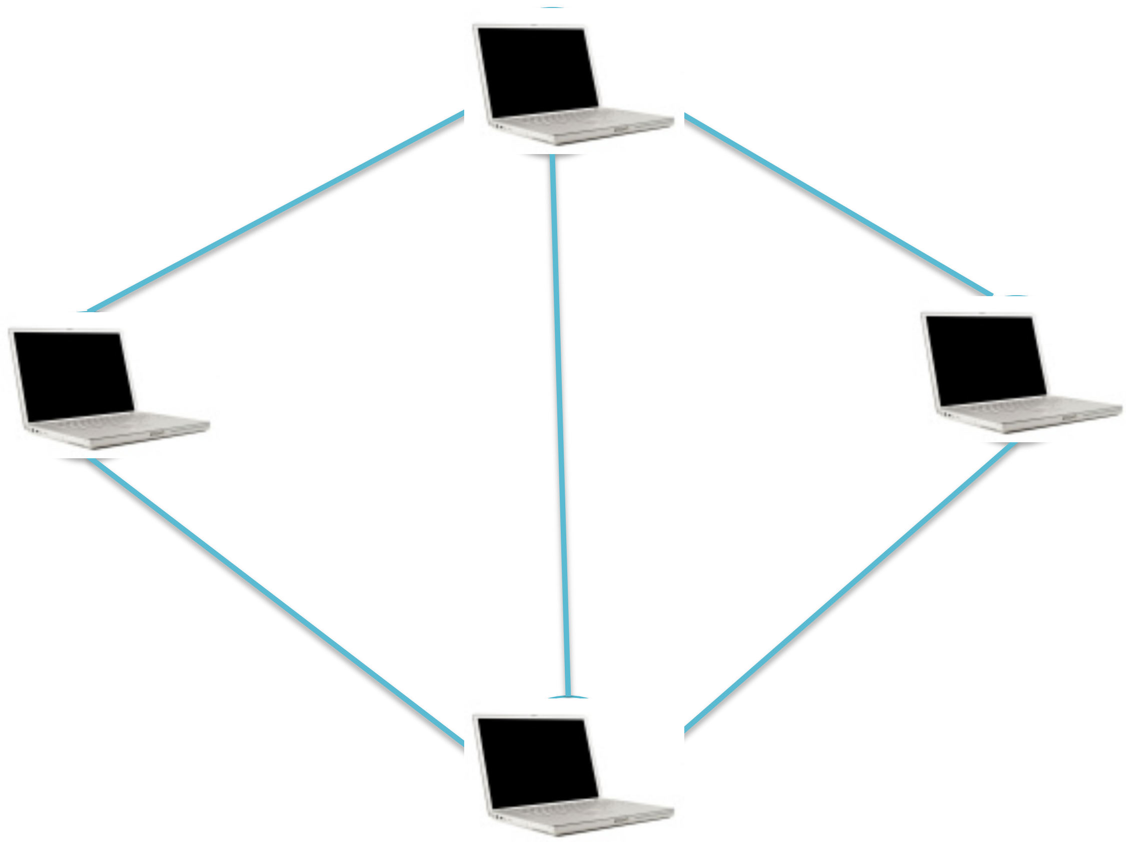


Multicore Processors

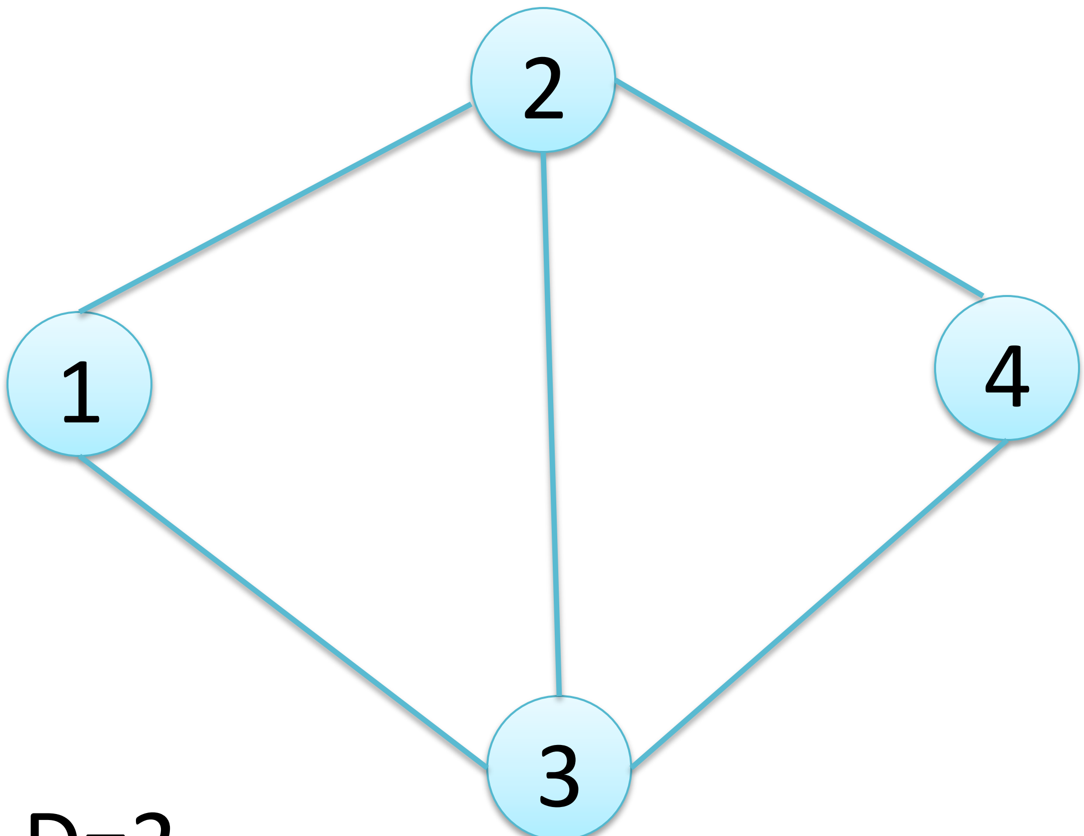


Chemical Reaction Networks

Distributed network:

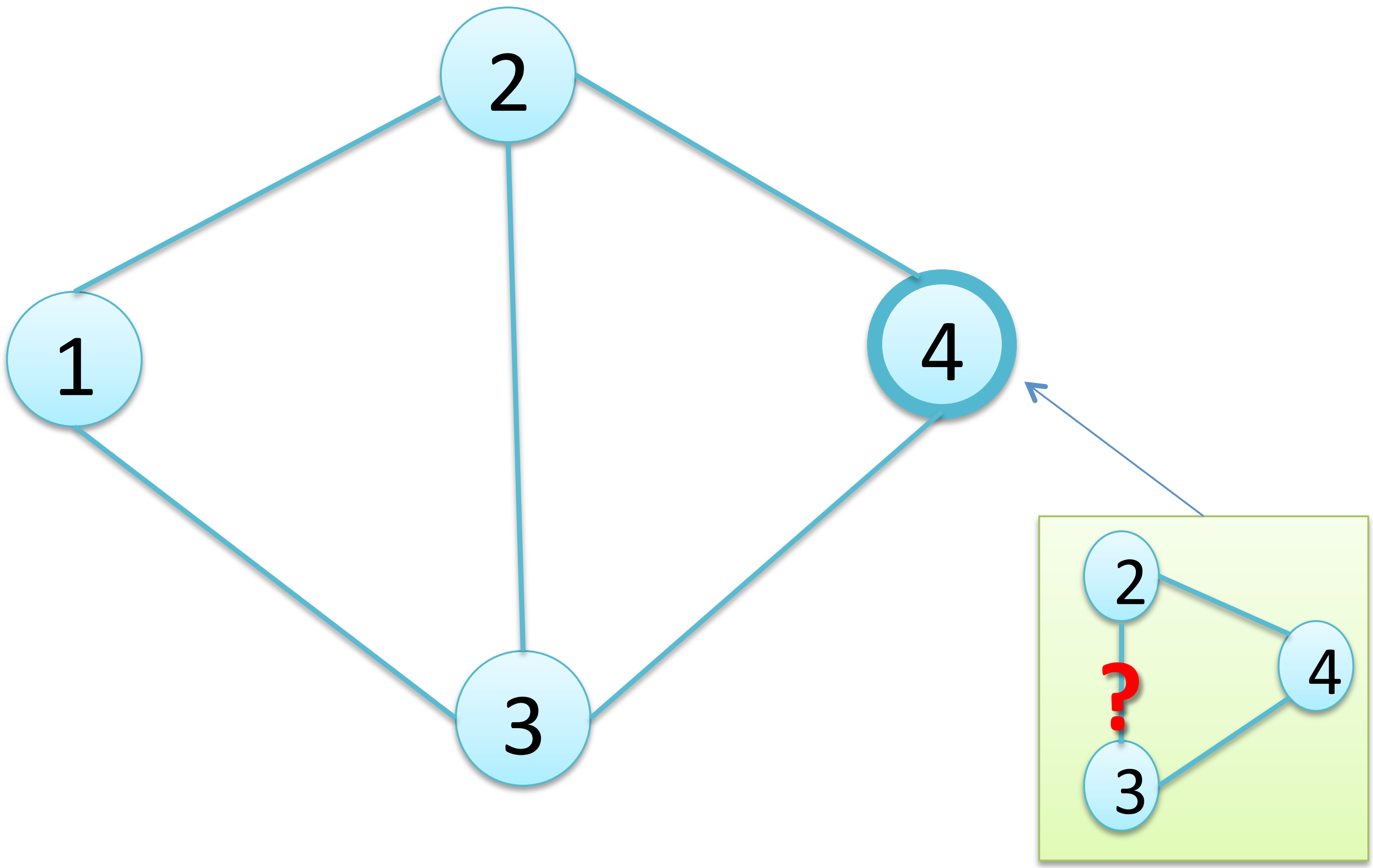


We are given a graph **G** of **n** nodes, diameter **D**



n= 4, D=2

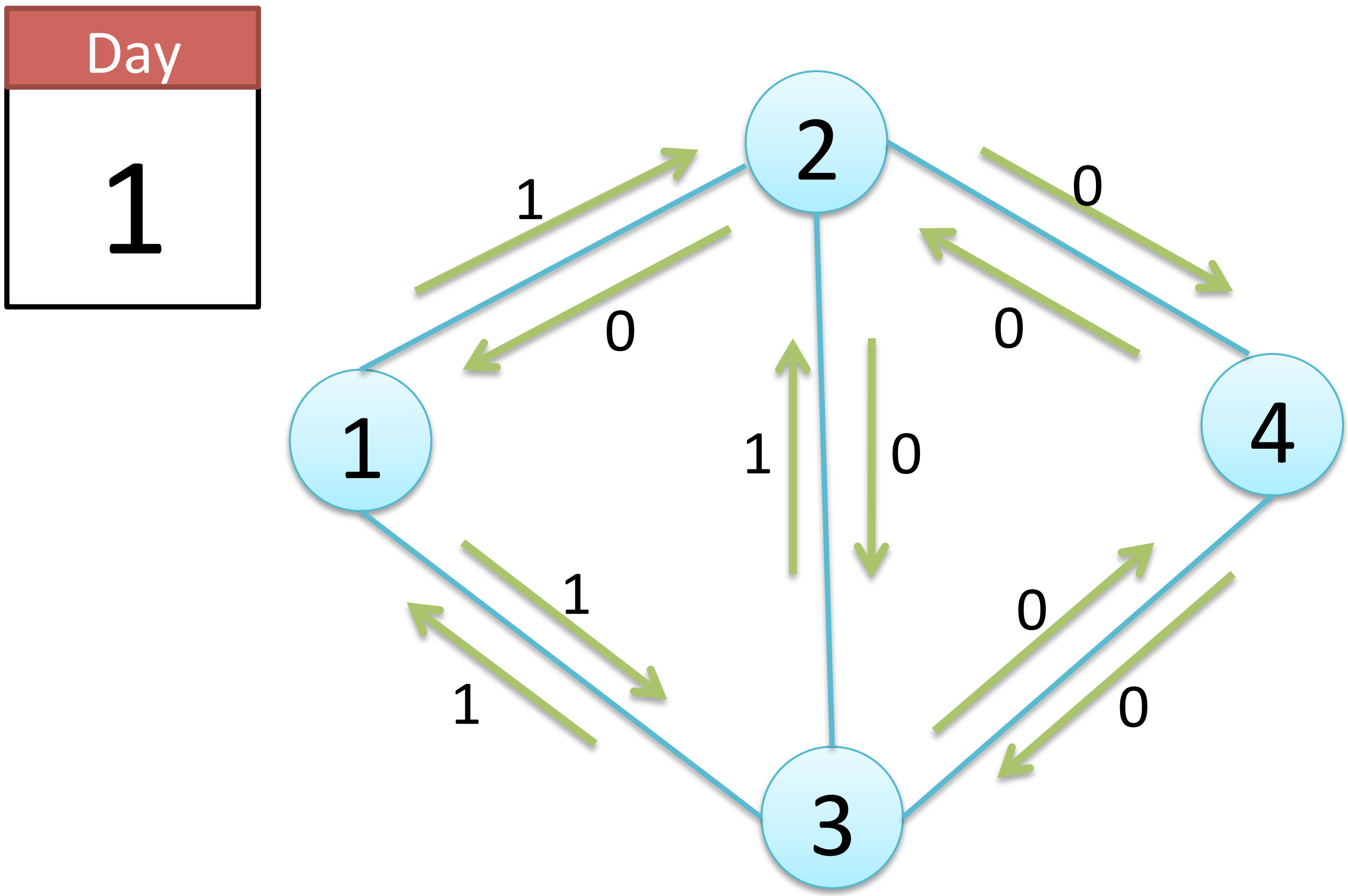
Each node knows only their neighbors



Time complexity

"number of days"

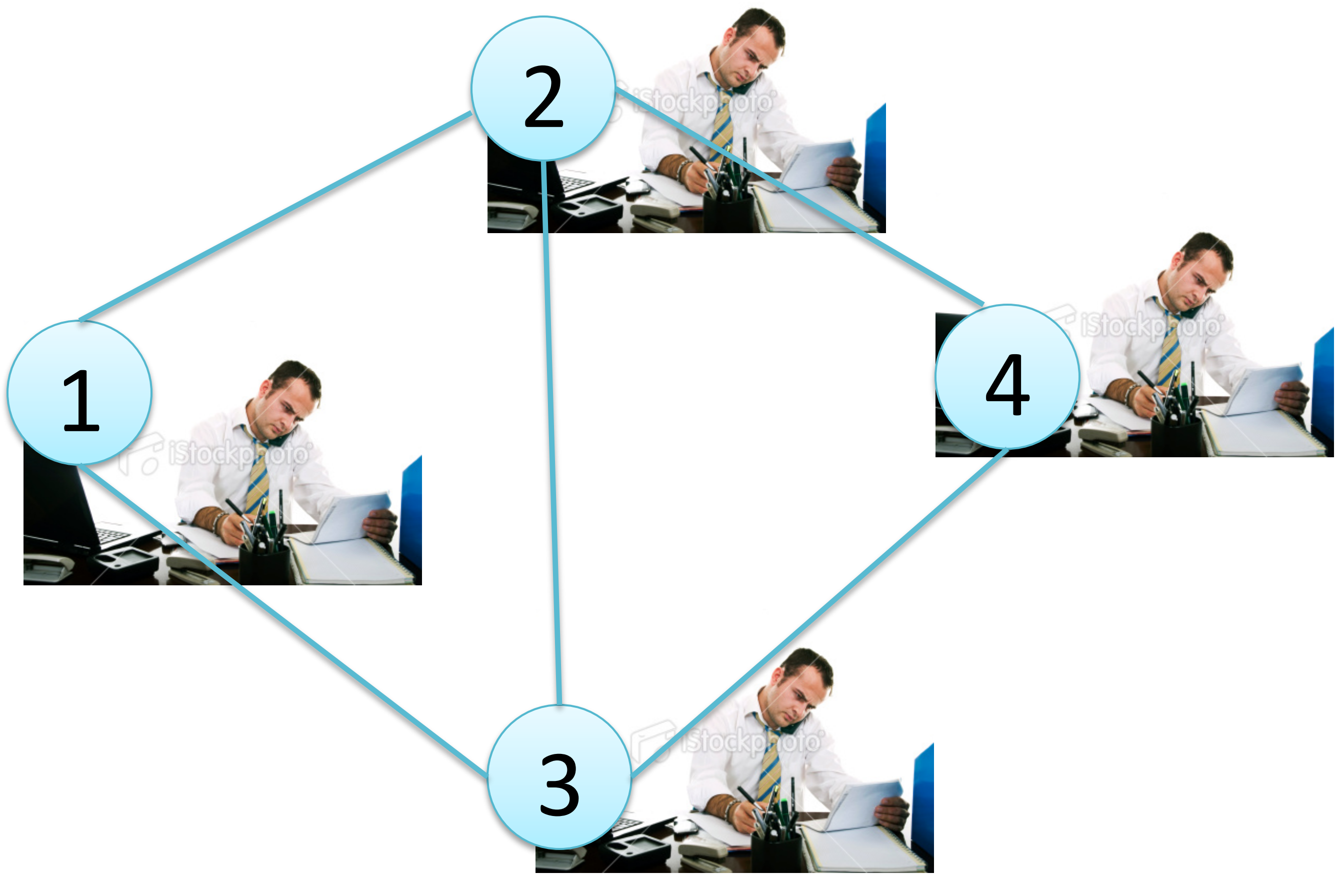
Days: Exchange one bit



Nights: Perform local computation

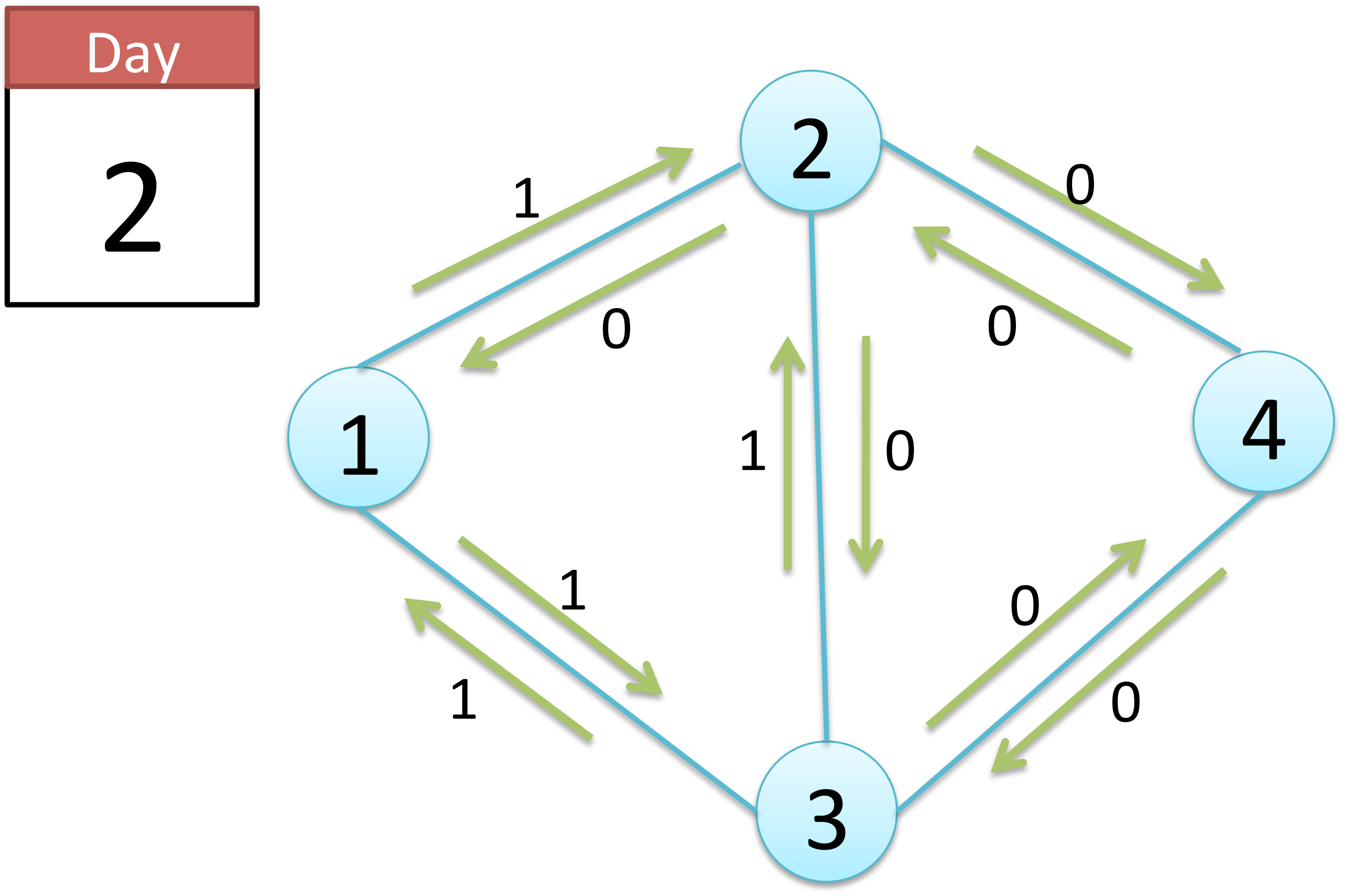
Night

1



<u>Assume:</u> Any calculation finished in one night

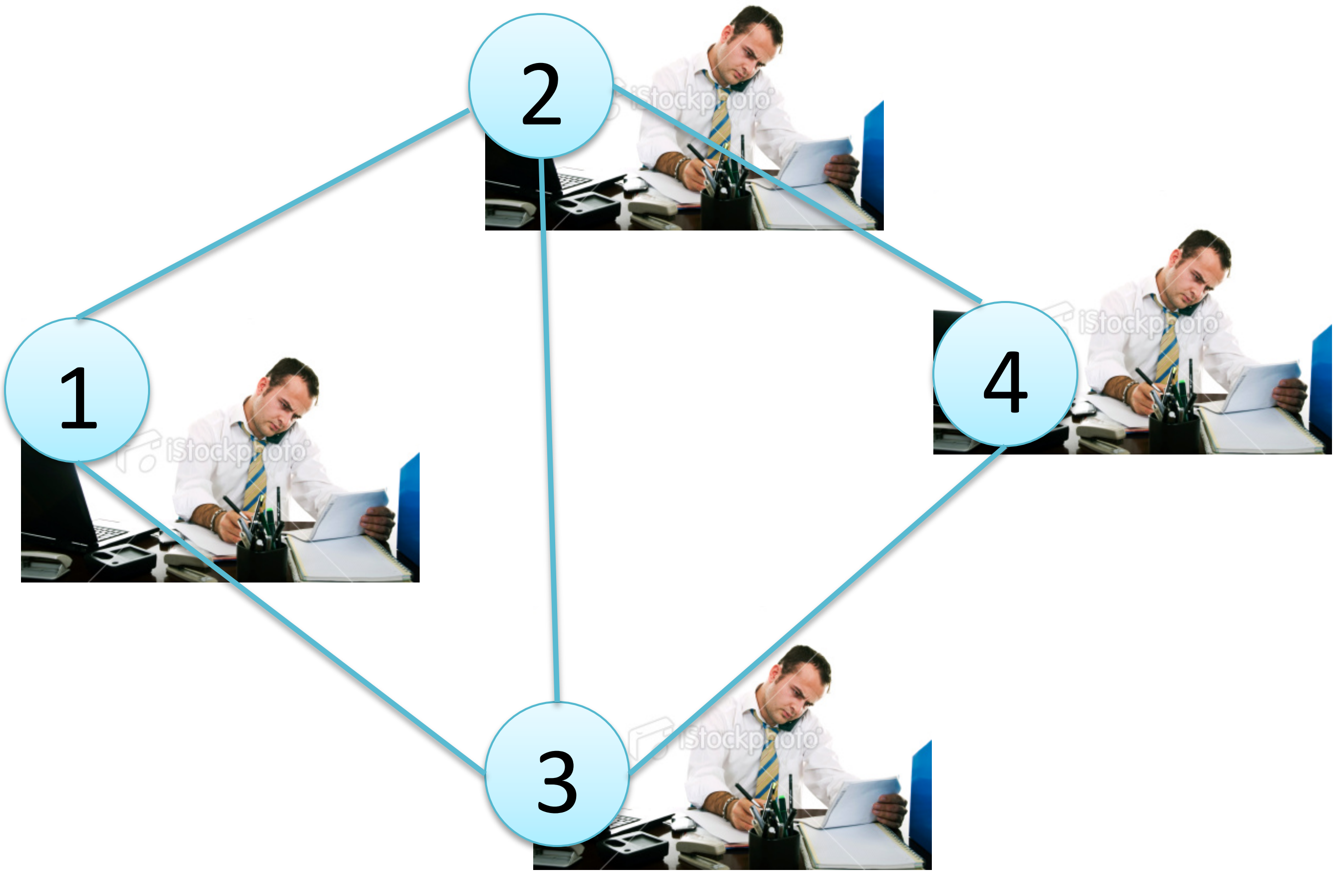
Days: Exchange one bit



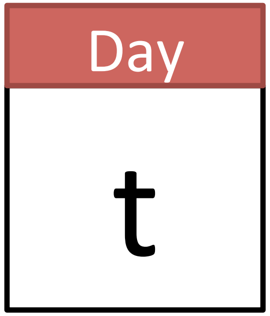
Nights: Perform local computation

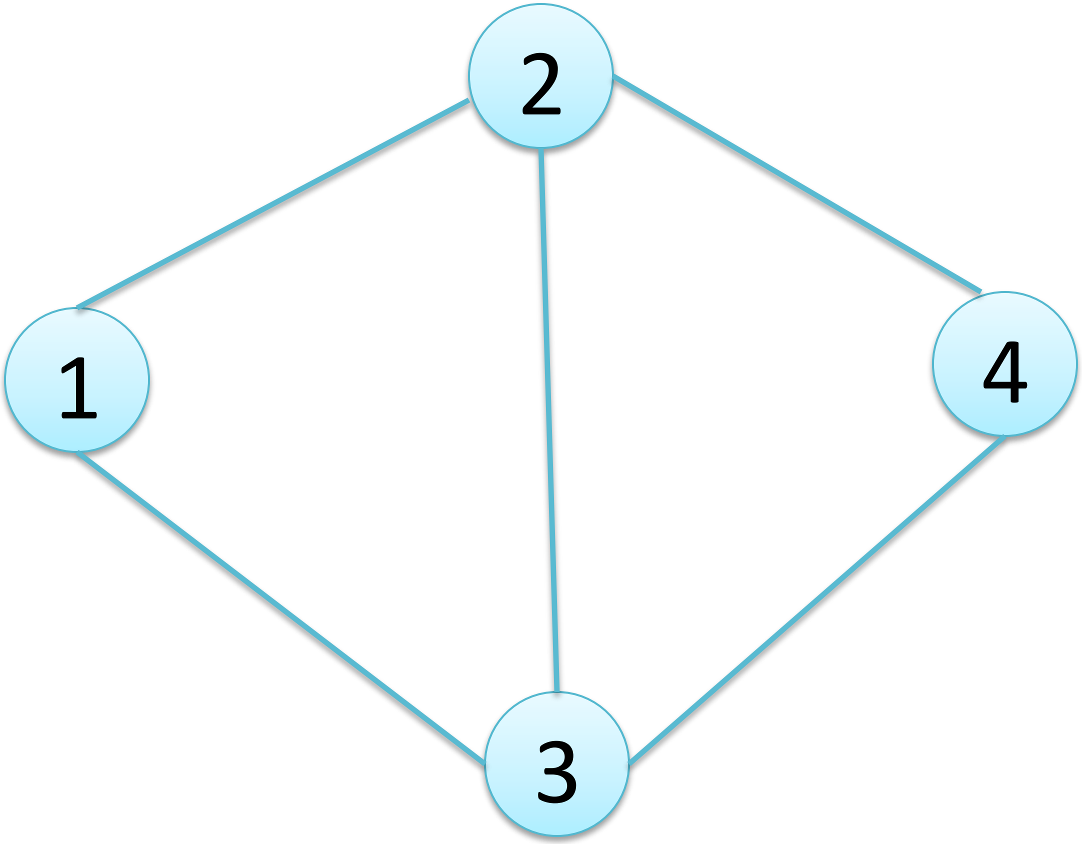
Night

2



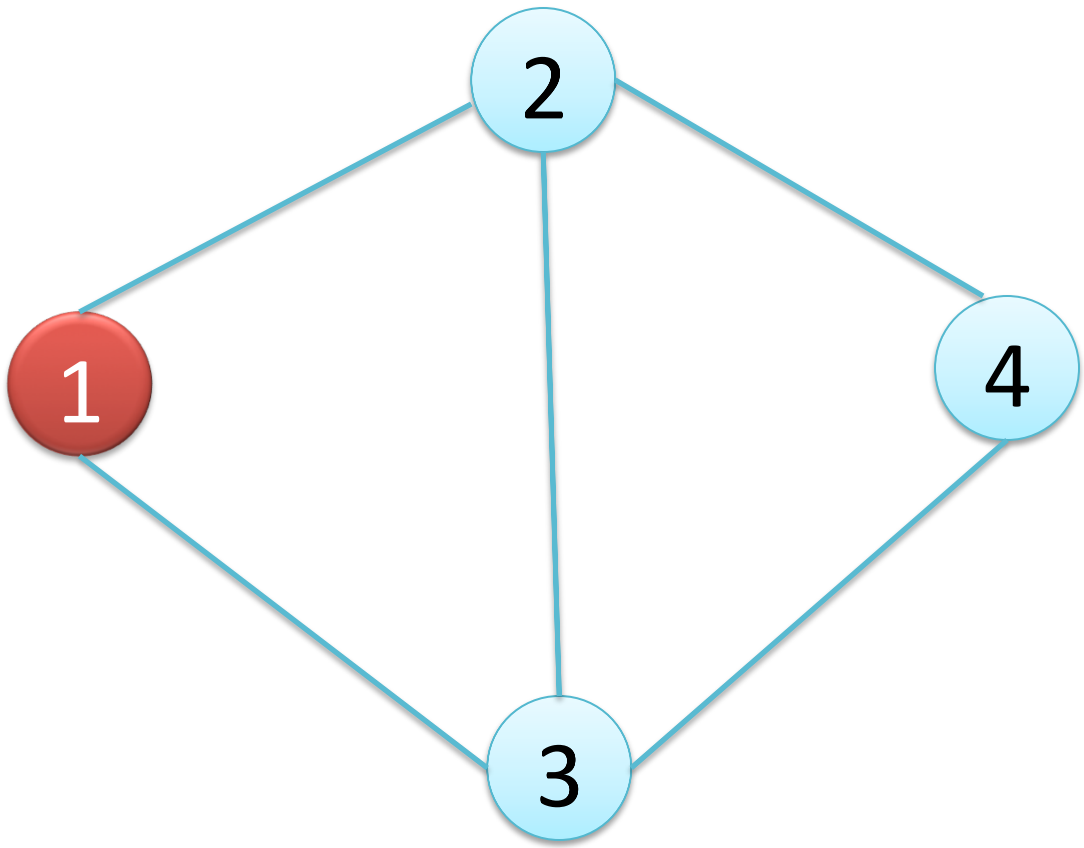
Finish on Day t → Time complexity = t



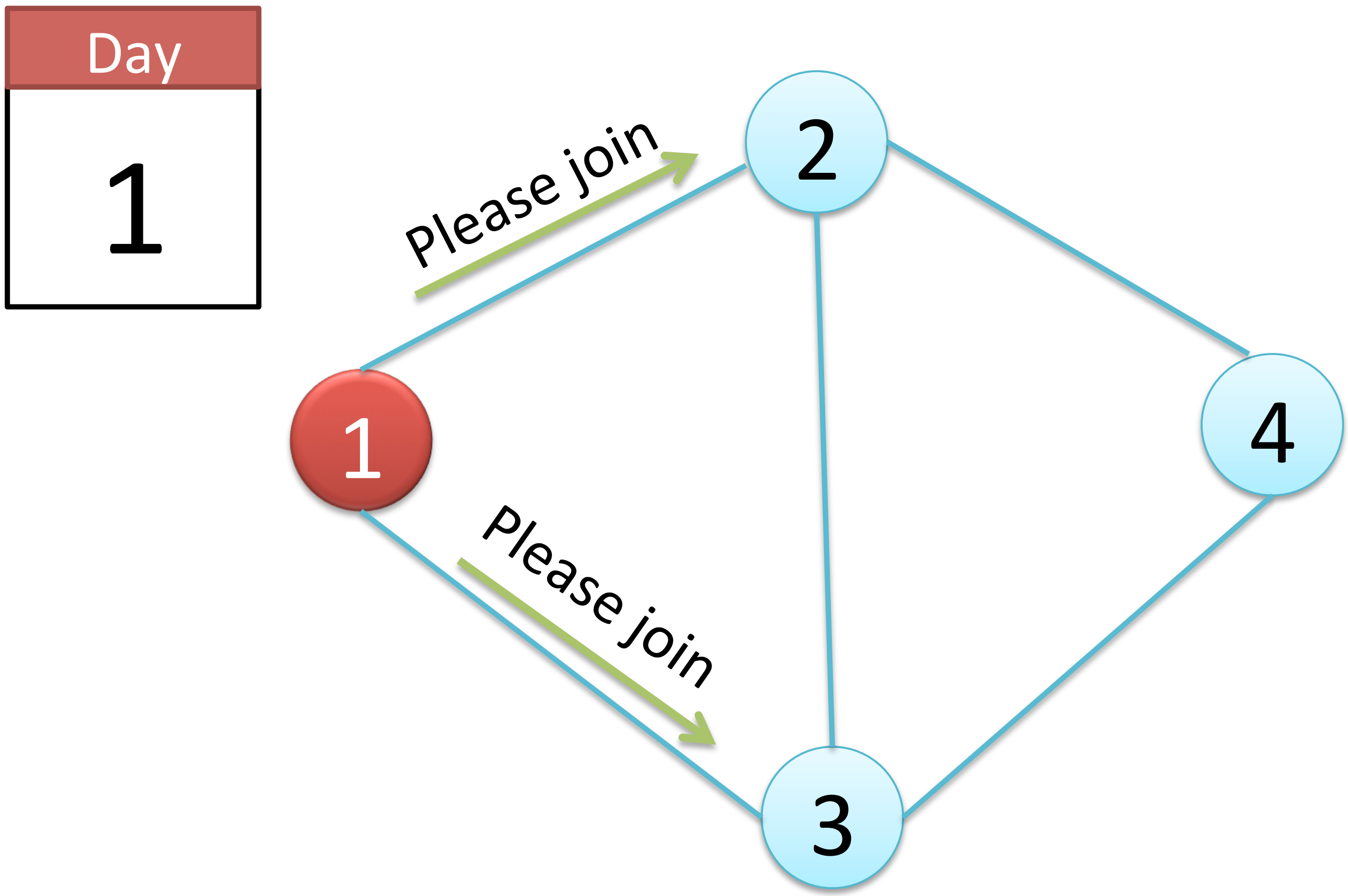


Quick Example
Finding a spanning tree

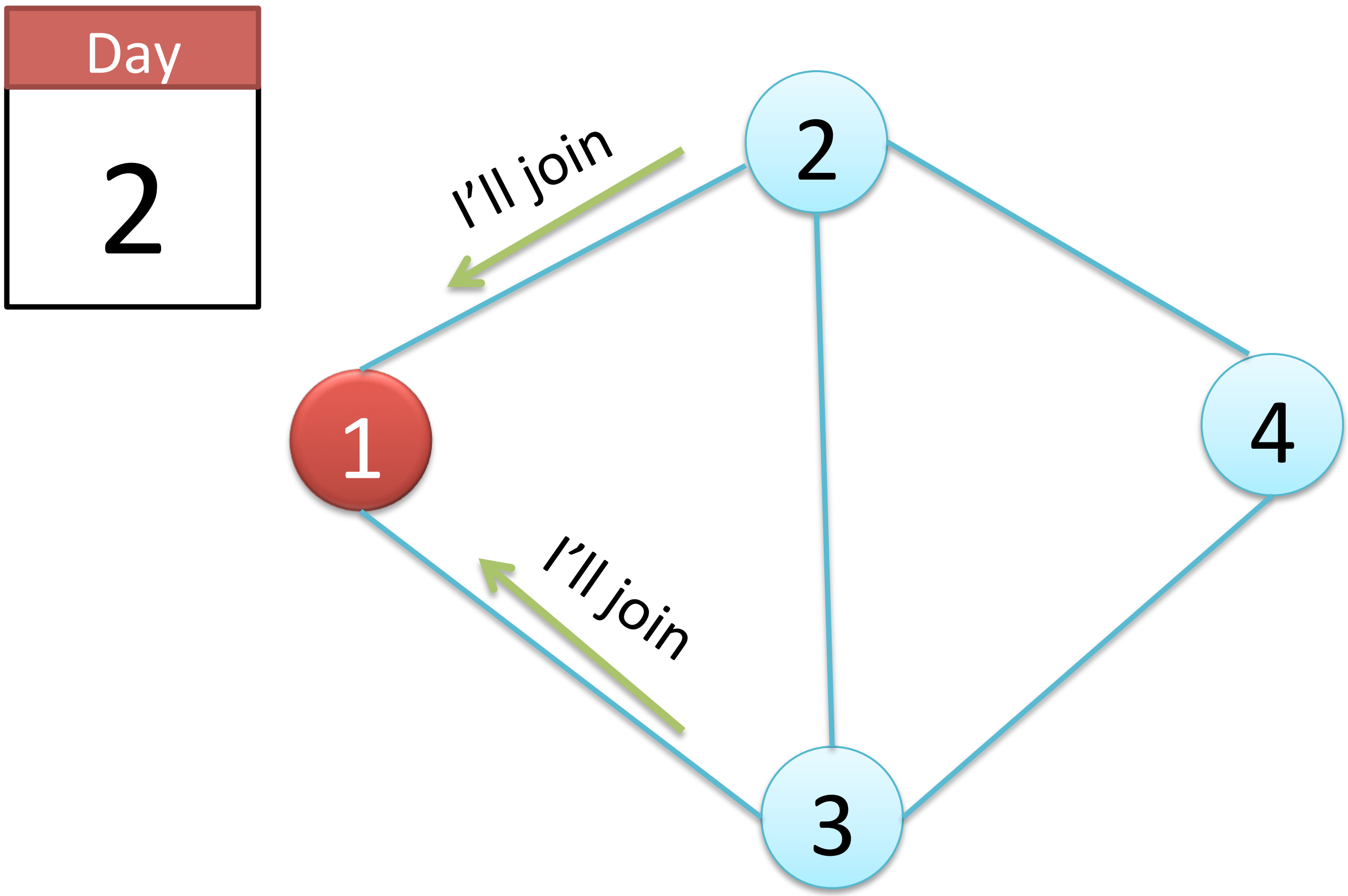
Start at an arbitrary node



New red nodes invite all neighbors



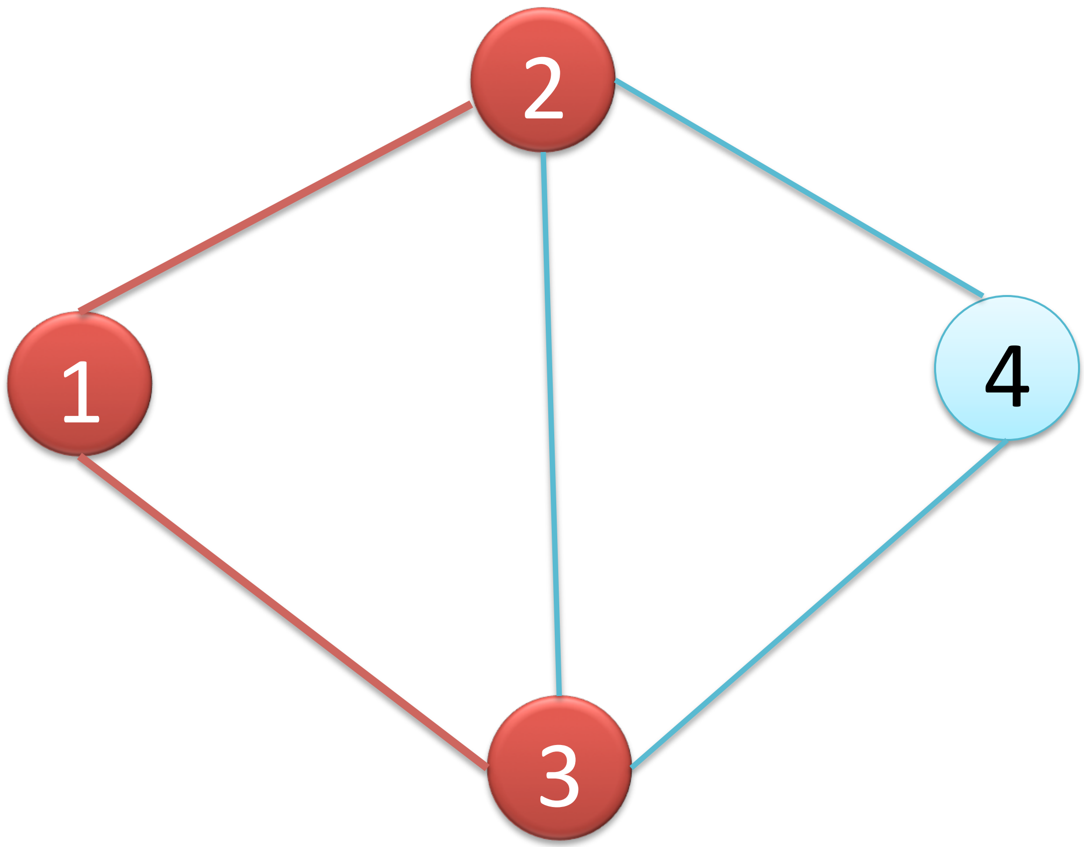
Blue nodes accept invitation of one neighbor



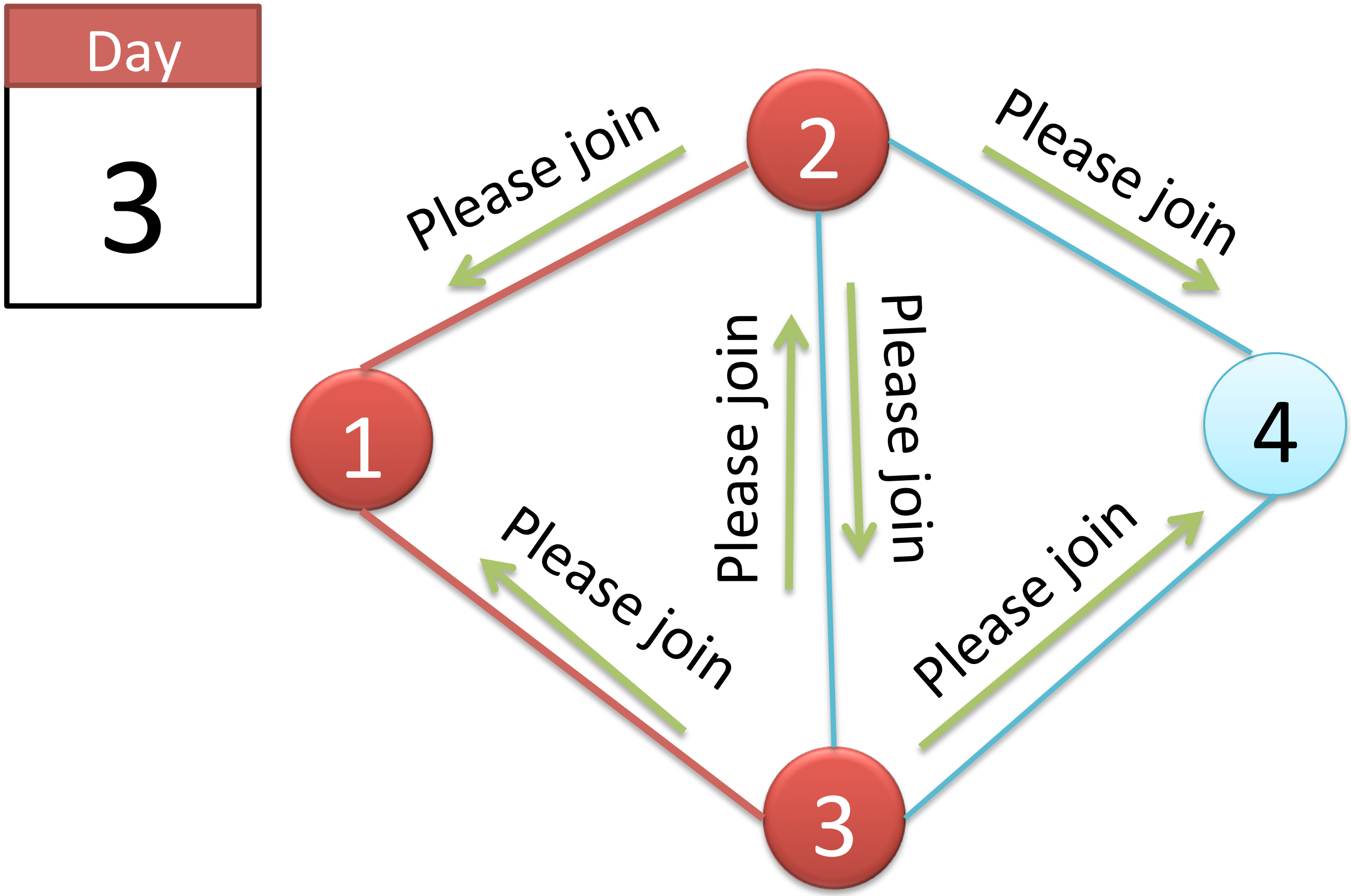
Blue nodes accept invitation of one neighbor

Day

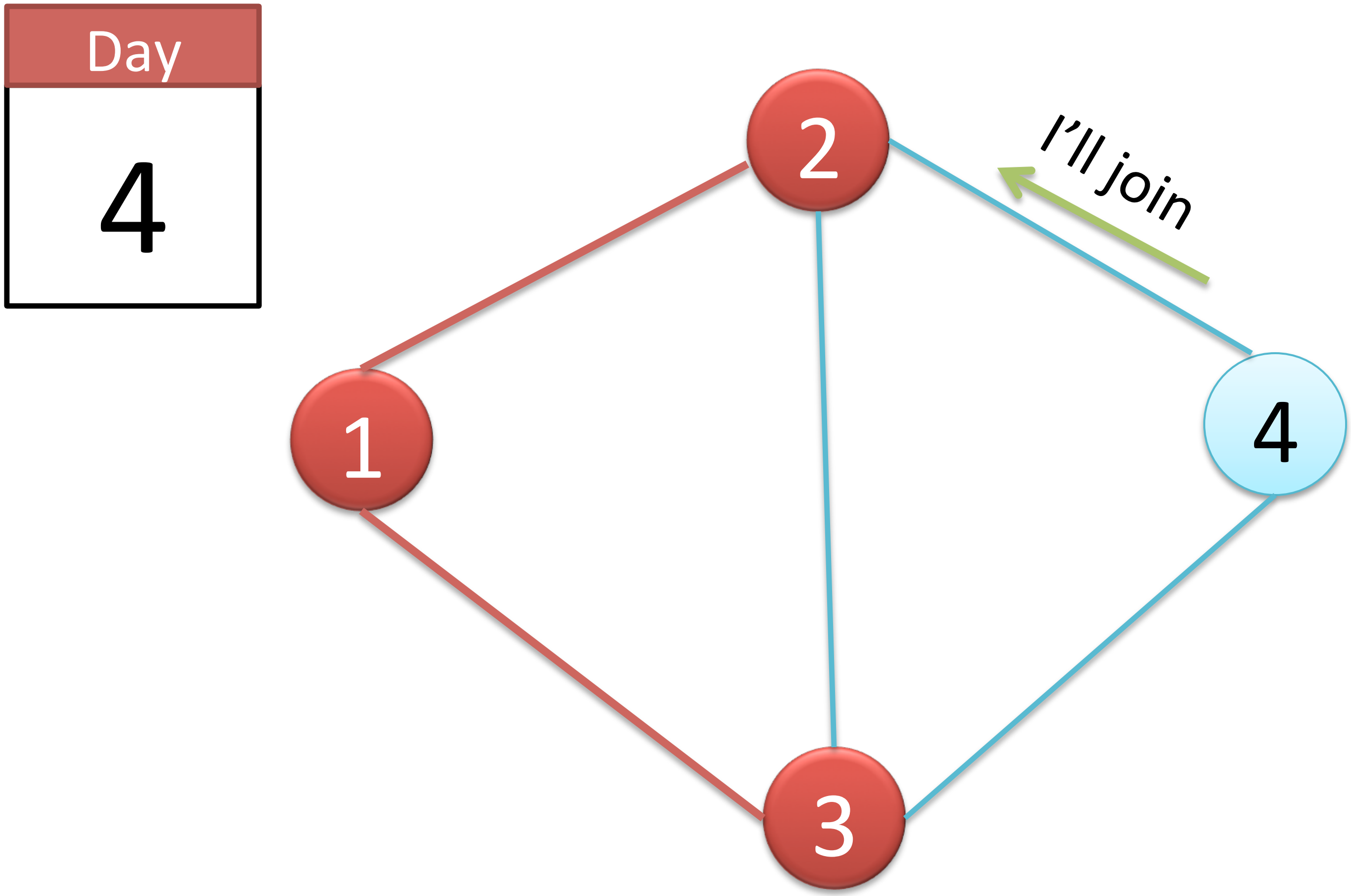
2



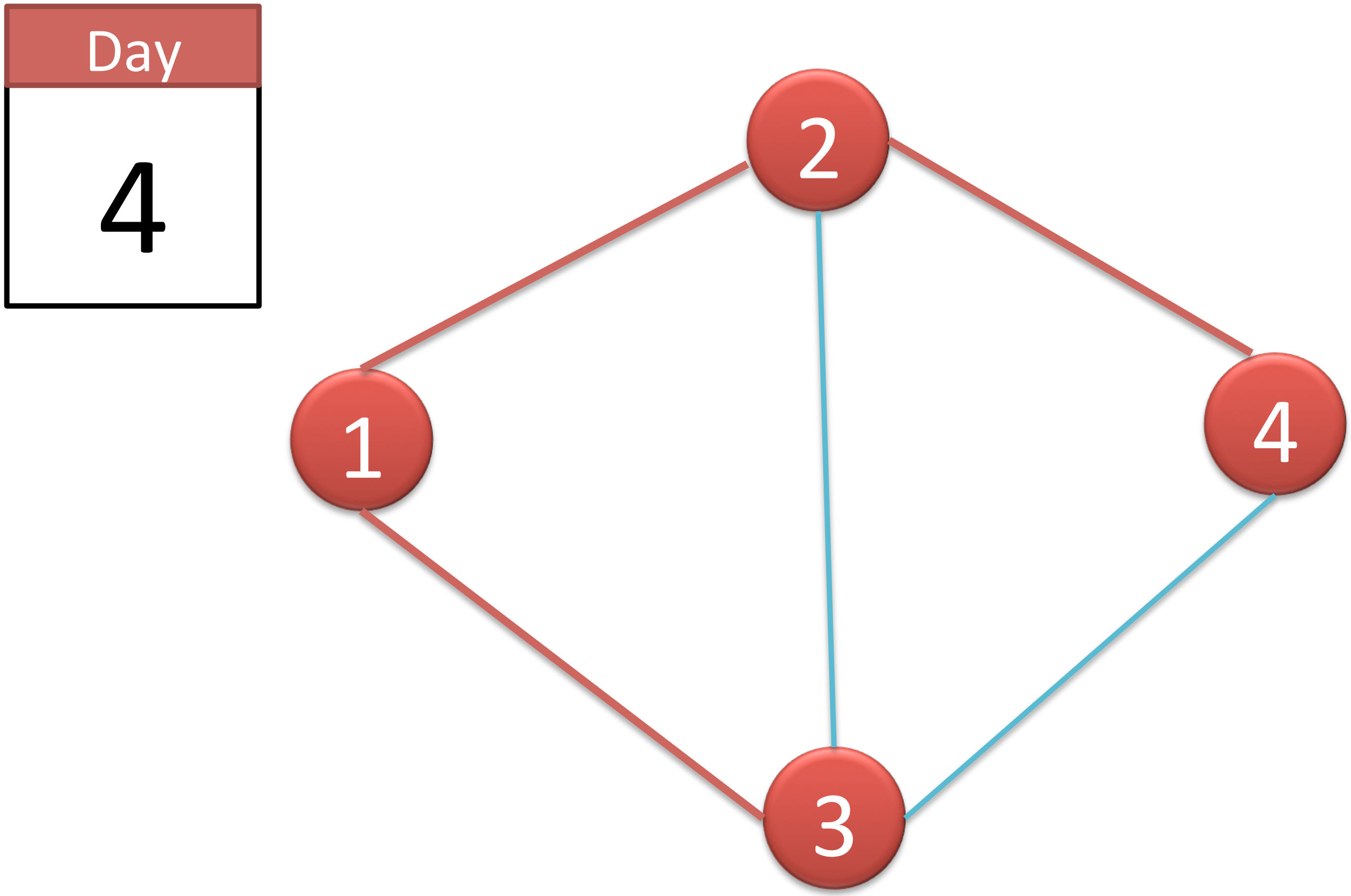
New red nodes invite all neighbors



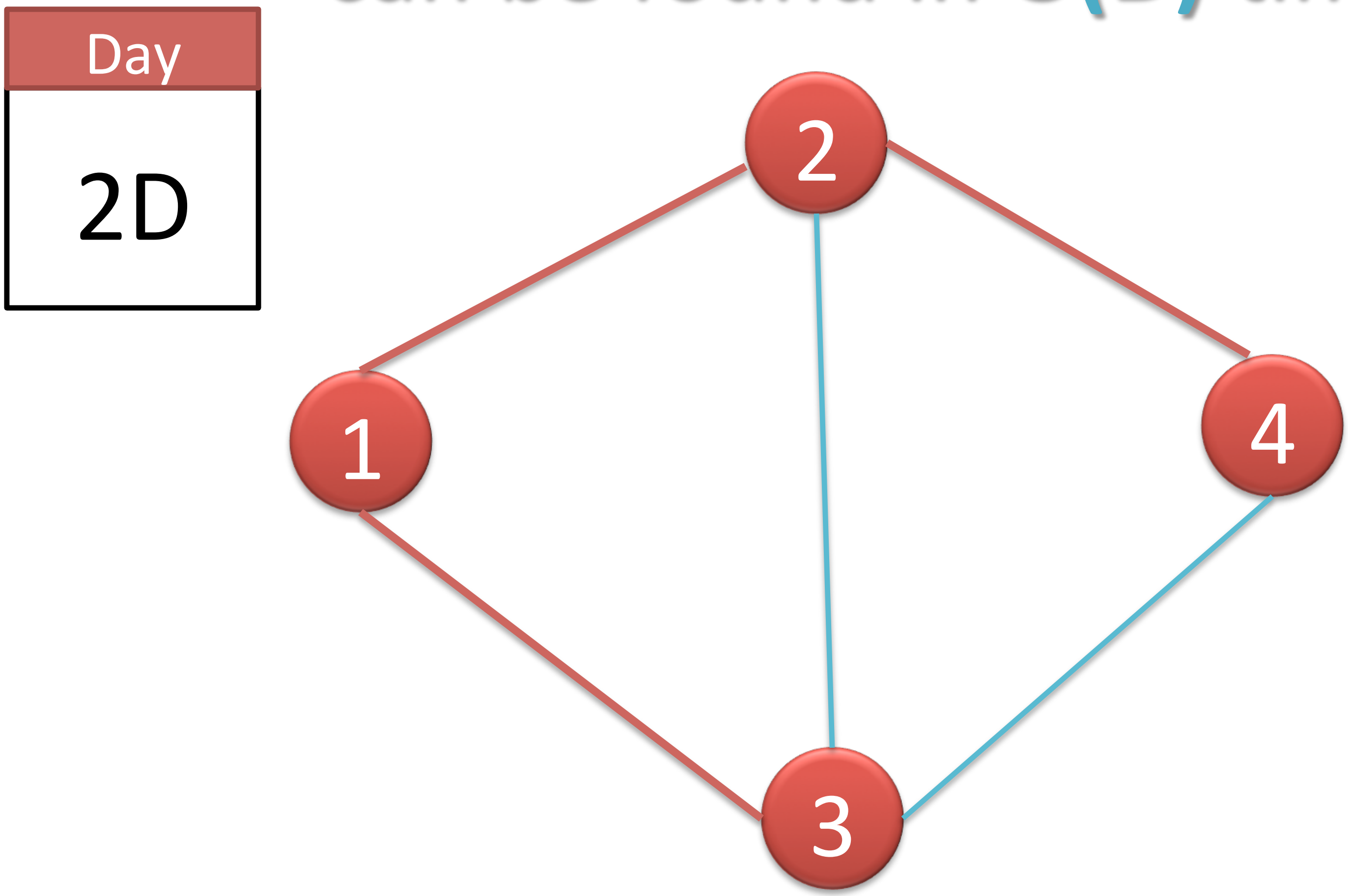
Blue nodes accept invitation of one neighbor



Blue nodes accept invitation of one neighbor



In general, a spanning tree can be found in O(D) time



State of the art (forgetting log)

Problems	Upper	Lower
Spanning tree (ST)	O(D)	

State of the art (forgetting log)

Problems	Upper	Lower
Spanning tree (ST)	O(D)	Ω(D)

Quick remarks

- This is called the **CONGEST** model
- Nodes usually exchange O(log n) or B bits a day
  - But we will ignore log n terms here anyway
- “Days” is actually called “rounds”
- Many assumptions: Global clock, no failures, no delays, unique ID, free internal computation, etc.
  - It helps us in focusing on the “locality” issue
  - And we are showing that <u>lower bounds</u> are true even with these assumptions

Part 2

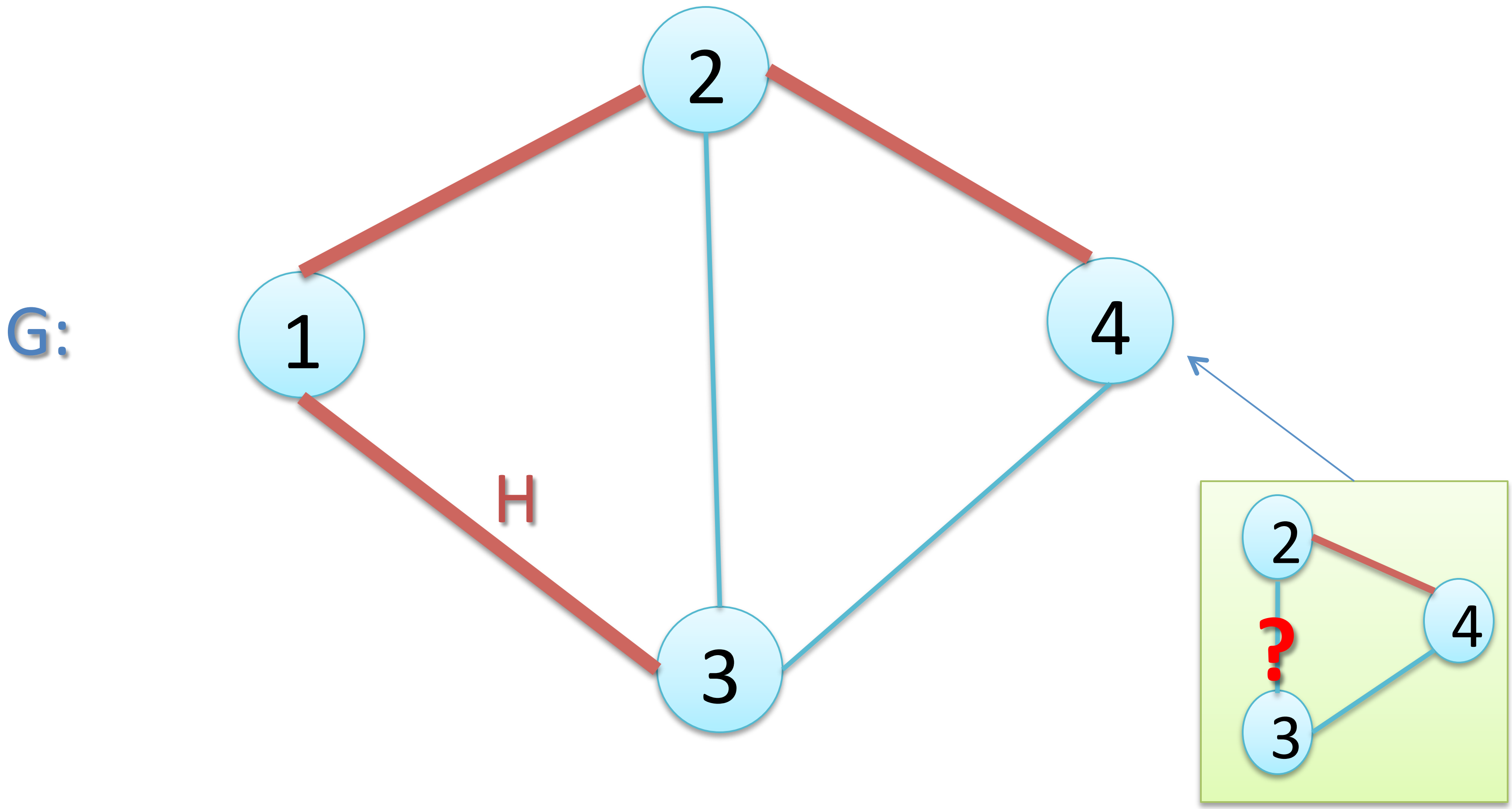
MST and ST verification

We have seen that ...

A spanning tree (ST) can be found in O(D) time

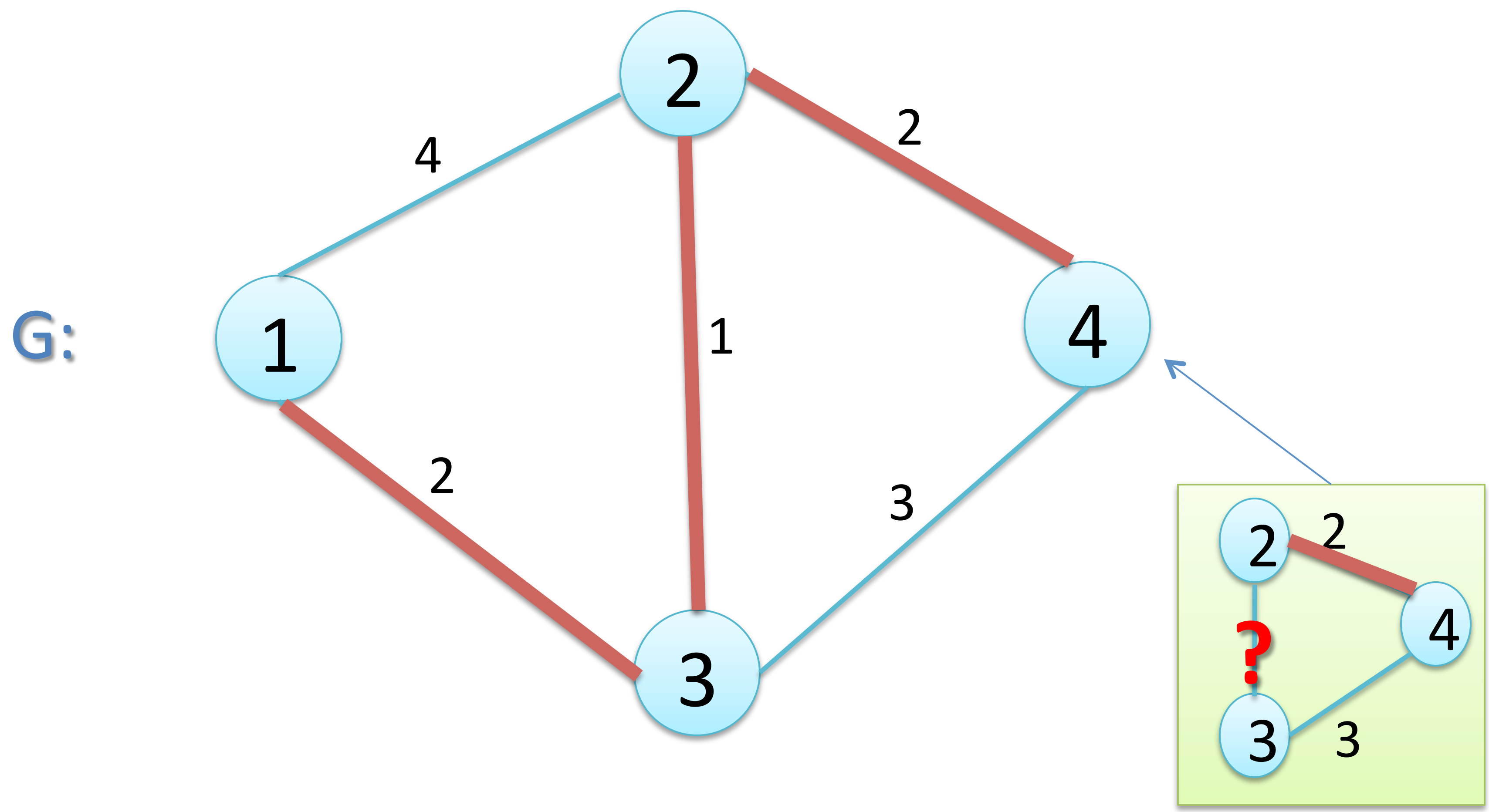
How about **verifying** that
a **subgraph** is a spanning tree?

Question 1: Given a subgraph H, can we verify that H is a spanning tree in O(D) time?



How about finding a
minimum spanning tree (MST)?

Question 2: Given edge weight **w**, can we find a minimum spanning tree in O(D) time?



Results on Minimum Spanning Tree (**MST**)

Gallager, Humblet, Spira, **TOPLAS'83**

A

cl

Chin, Teng, **FOCS'85**

Gafni, **PODC'85**

Awerbuch, **STOC'87**

Garay, Kutten, Peleg, **FOCS'93**

Kutten, Peleg, **PODC'95**

$O(D + n^{1/2} \log^* n)$-time

Lotker, Patt-Shamir, Peleg **PODC'01**

Lotker, Patt-Shamir, Peleg

Elkin **SODA'04**

Khan, Pandurangan **DISC'06**

- $O(\log n)$-approximation algorithm in $O(D+L(G, w))$-time where $L(G, w)$ is a parameter called the "local shortest path diameter"
- **Best student paper of DISC'06**

Peleg, Rubinovich **FOCS'99**

$\Omega(n^{1/2} / (\log n))$-time on an $O(\log n)$

Elkin **STOC'04**

α-approximation algorithms require $\Omega((n/\alpha)^{1/2})$ –time, even on $O(\log n)$-diameter graphs

Approximation algorithms?

Approximation algorithms?

State of the art (forgetting log)

Problems	Upper	Lower
Spanning tree (ST)	O(D)	Ω(D)
MST	O(D + $n^{1/2}$)	Ω(D + $n^{1/2}$)
α-approx. MST		Ω(D + $(n/\alpha)^{1/2}$)

State of the art (forgetting log)

Problems	Upper	Lower
Spanning tree (ST)	O(D)	Ω(D)
MST	O(D + $n^{1/2}$)	Ω(D + $n^{1/2}$)
α-approx. MST		Ω(D + $(n/\alpha)^{1/2}$)
ST verification		

State of the art (forgetting log)

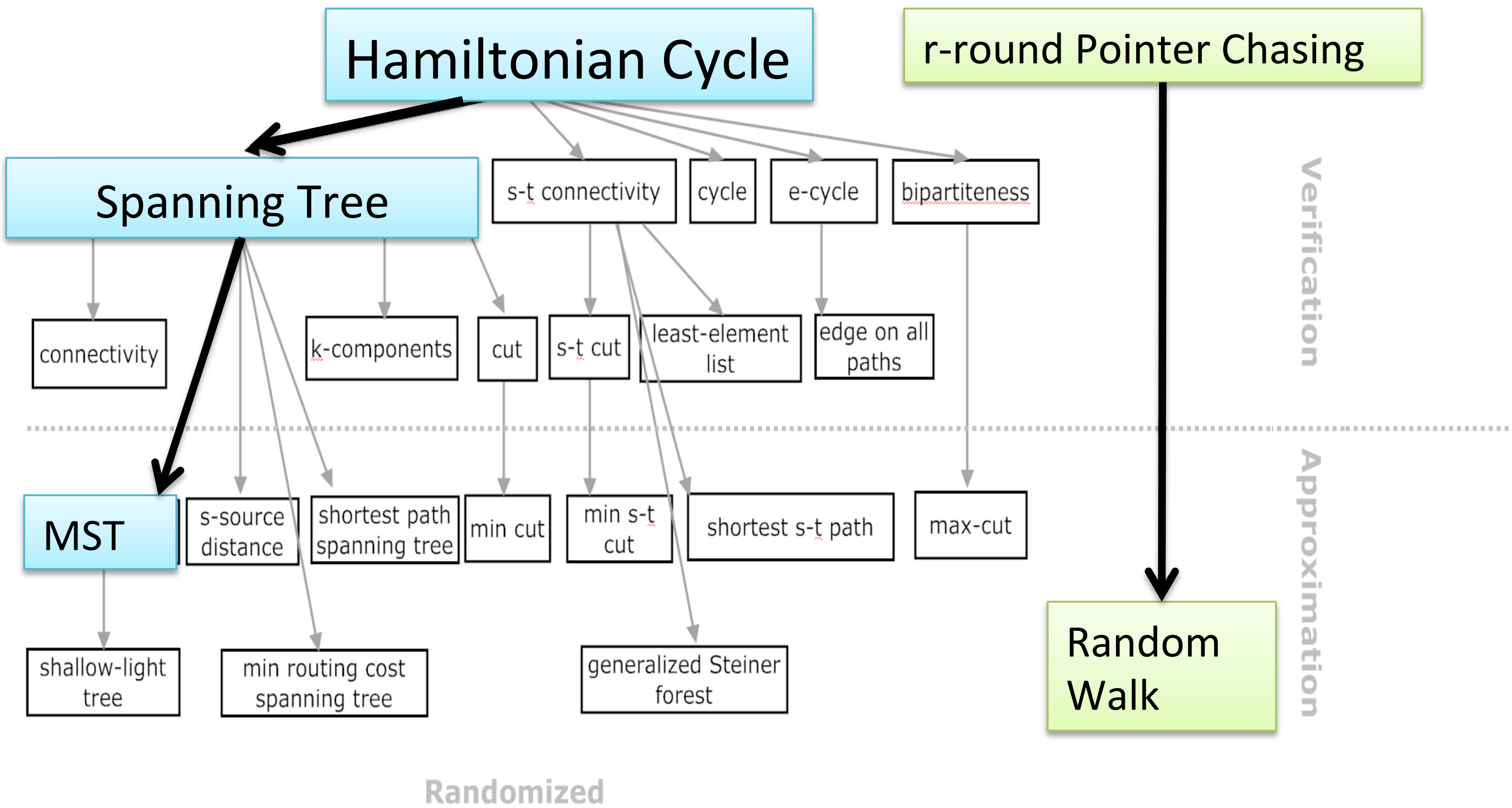
Problems	Upper	Lower
Spanning tree (ST)	O(D)	Ω(D)
MST	$O(D + n^{1/2})$	$\Omega(D + n^{1/2})$
α-approx. MST		$\Omega(D + (n/\alpha)^{1/2})$
ST verification	$O(D + n^{1/2})$	

Parameters of G, not H

Our results

State of the art (forgetting log)

Problems	Upper	Lower
Spanning tree (ST)	$O(D)$	$\Omega(D)$
MST	$O(D + n^{1/2})$	$\Omega(D + n^{1/2})$
α-approx. MST	~~ ~~	$\Omega(D + (n / \alpha)^{1/2})$
ST verification	$O(D + n^{1/2})$	$\mathbf{\Omega(D + n^{1/2})}$



Theorem: Above problems require $\mathbf{\Omega(n^{1/2})}$ time to verify/approximate

Part 3

Proofs

Direct equality verification lower bound $\Omega(b)$

Part 3.3

Distributed equality verification lower bound $\Omega(n^{1/2})$

Well-known result in communication complexity

Simulation Theorem

Part 3.2

$\Omega(b)$

ST verification lower bound $\Omega(n^{1/2})$

Part 3.1

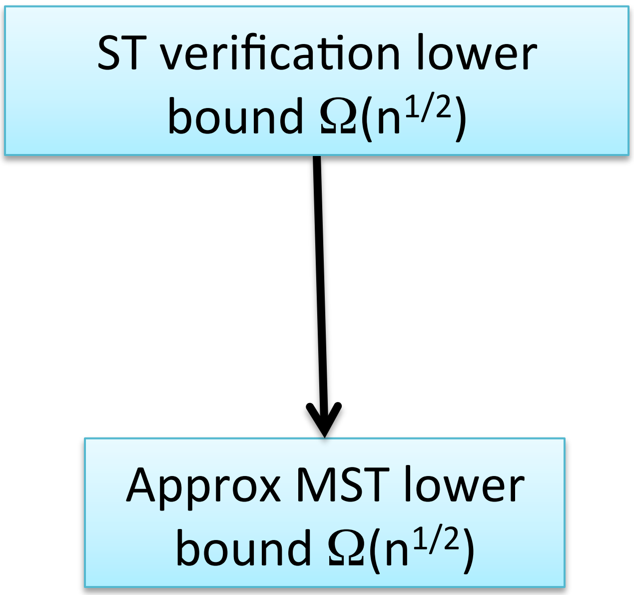
Approx MST lower bound $\Omega(n^{1/2})$

Notes

-The lower bounds hold on a graph of diameter **D=O(log n)**

- For simplicity, we will consider only **D=O($n^{1/4}$)**

Part 3.1


ST verification lower bound Ω(n^1/2)
Approx MST lower bound Ω(n^1/2)

α-approximating MST in

$O(n^{0.49}+D)$ time

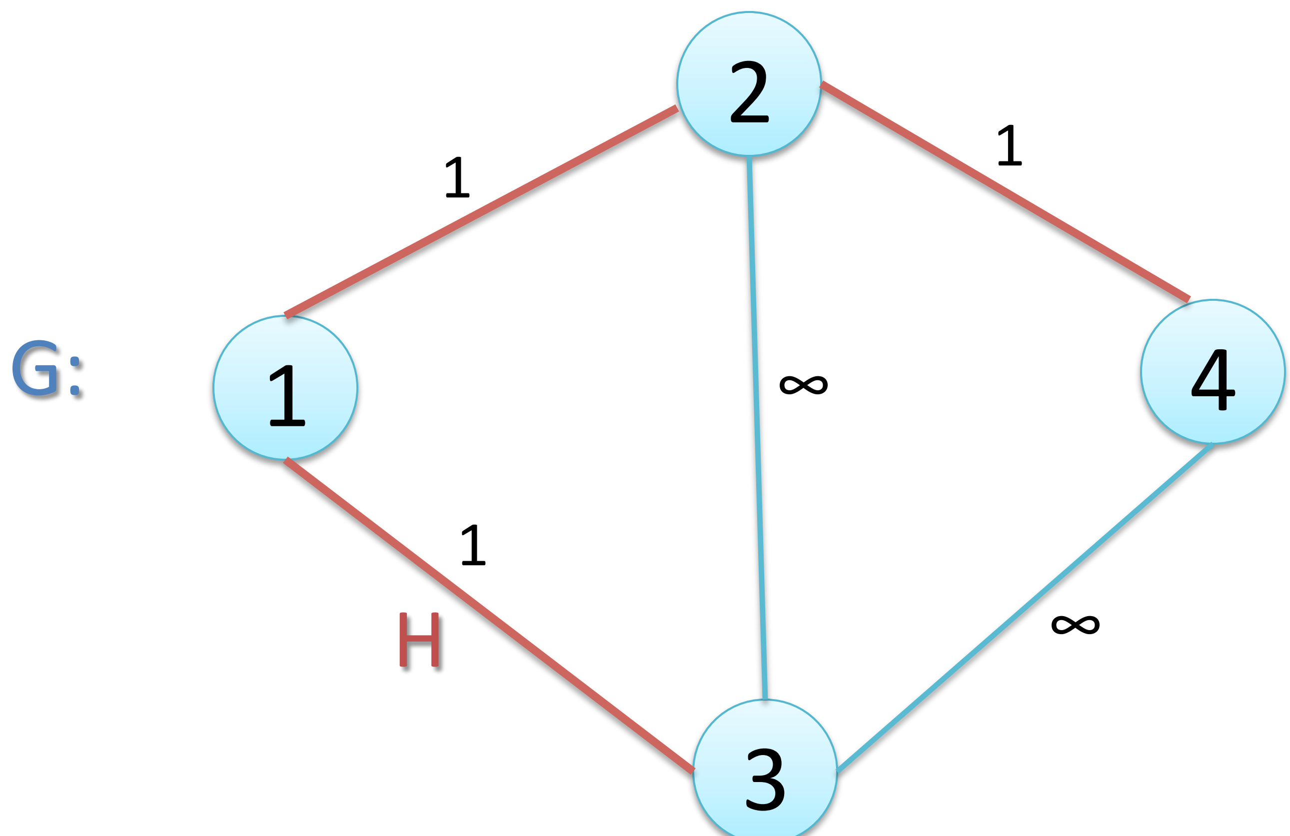
Spanning tree verification in

$O(n^{0.49}+D)$ time

Assume that algorithm **A**

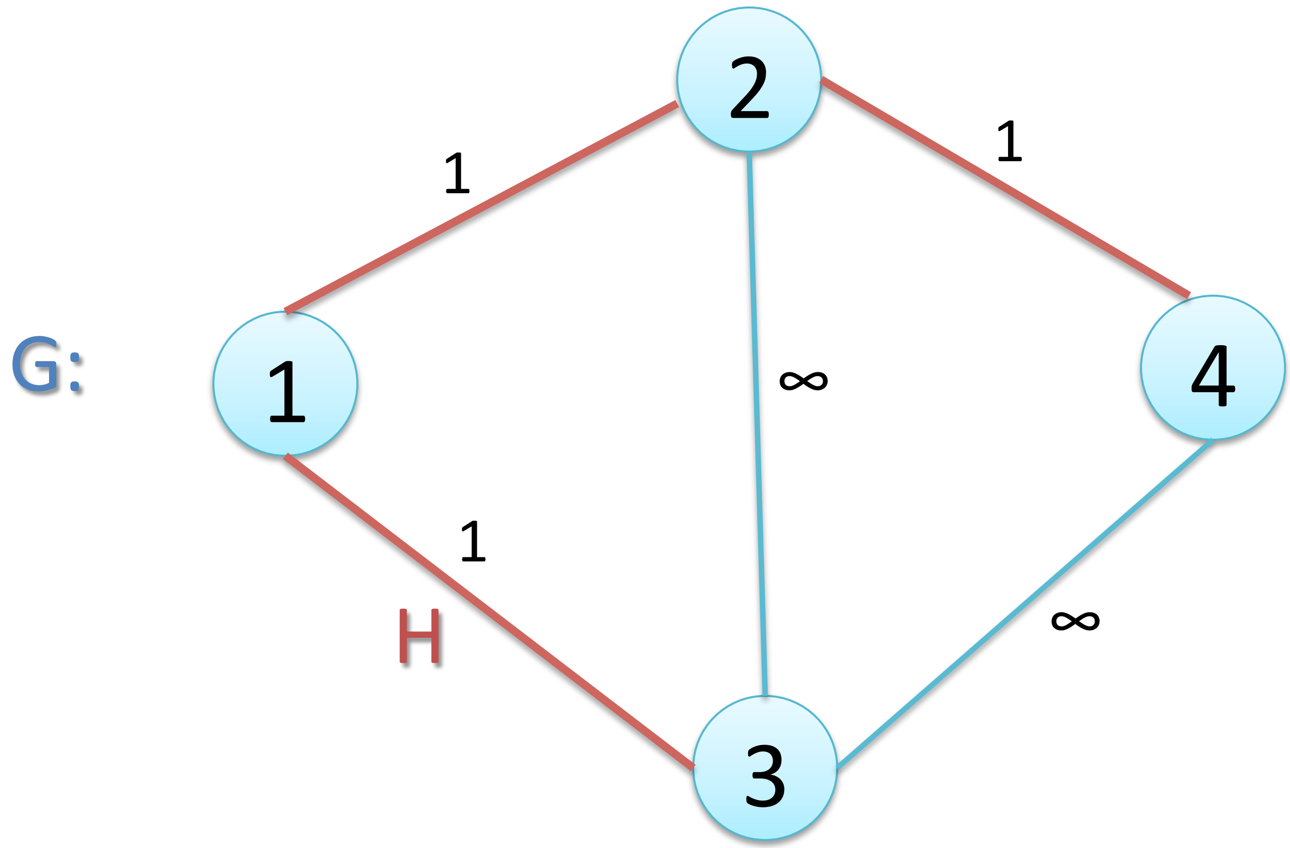
- is 10-approximation
- runs in **$O(n^{0.49}+D)$**-time

Put weight **1** to edges in H, and **∞** to others



Observe: H is a spanning tree if and only if

1) it has n-1 edges
2) MST has weight n-1 ⟺ **A** returns value $\leq 10(n-1) < 10n$



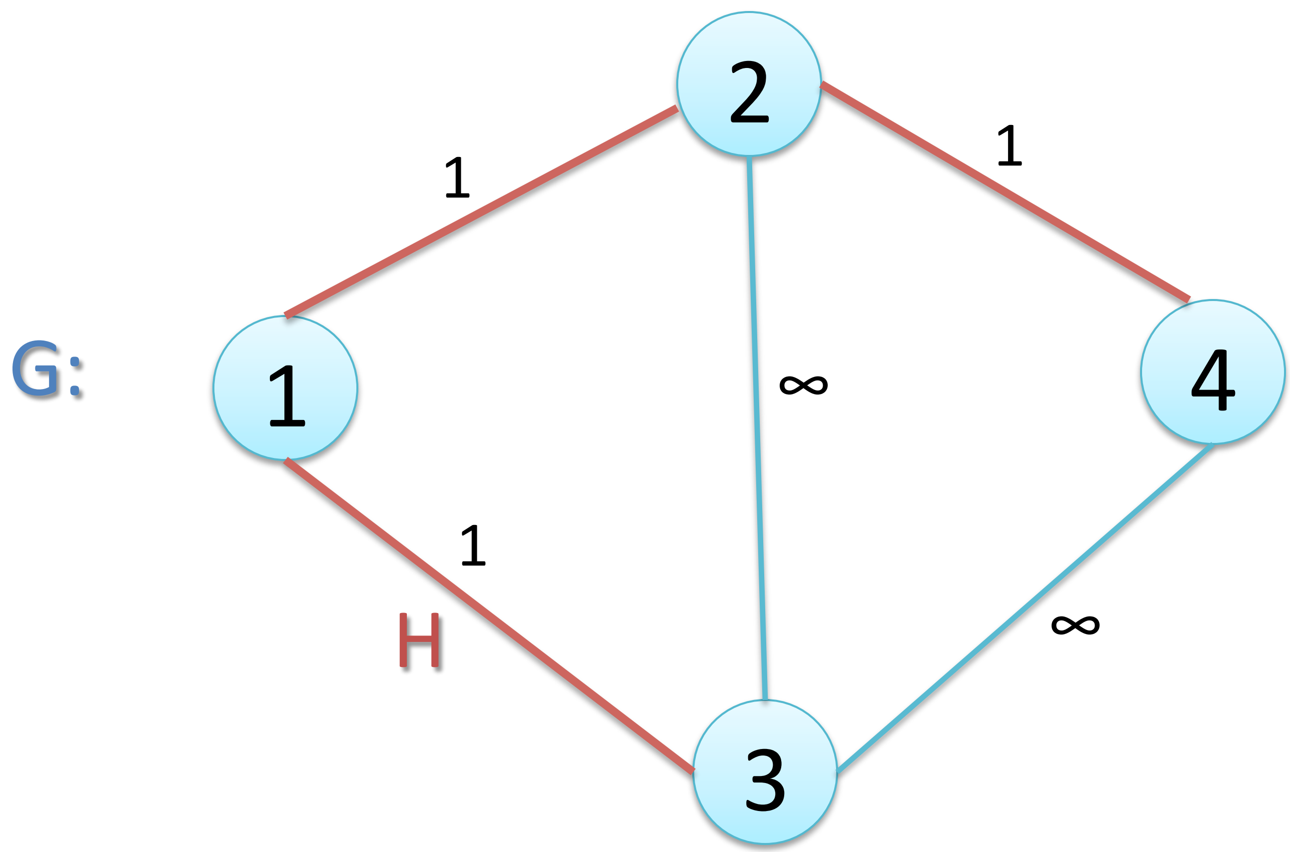
Observe: H is a spanning tree if and only if

1) it has n-1 edges ← O(D)

2) algo A returns value <10n ← $O(n^{0.49}+D)$

⇓

$O(n^{0.49}+D)$



Direct Equality Verification lower bound $\Omega(b)$

Part 3.3

Distributed Equality Verification lower bound $\Omega(n^{1/2})$

$\Omega(b)$

Well-known result in communication complexity

Part 3.2

ST verification lower bound $\Omega(n^{1/2})$

Part 3.1

Approx MST lower bound $\Omega(n^{1/2})$

Notes

-The lower bounds hold on a graph of diameter **D=O(log n)**

- For simplicity, we will consider only **D=O($n^{1/4}$)**

Distributed Equality Verification lower bound $\Omega(n^{1/2})$

Part 3.2

ST verification lower bound $\Omega(n^{1/2})$

Part 3.2

Direct Equality Verification

Distributed Equality Verification

Direct Equality Verification

Distributed Equality Verification

$x=y$?

Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

Direct Equality Verification

Distributed Equality Verification

Yes, x=y

Yes, x=y

x=y?

Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

Direct Equality Verification

Distributed Equality Verification

One solution: Alice sends everything ... time=b

x=y?

x

(b days for b bits)

Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

Direct Equality Verification

Distributed Equality Verification

One solution: Alice sends everything ... time=b
Theorem: Any algorithm needs Ω(b) time

x=y?

x

(b days for b bits)

Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

Direct Equality Verification

Distributed Equality Verification

x=y

x=y

Graph G(b)

Alice

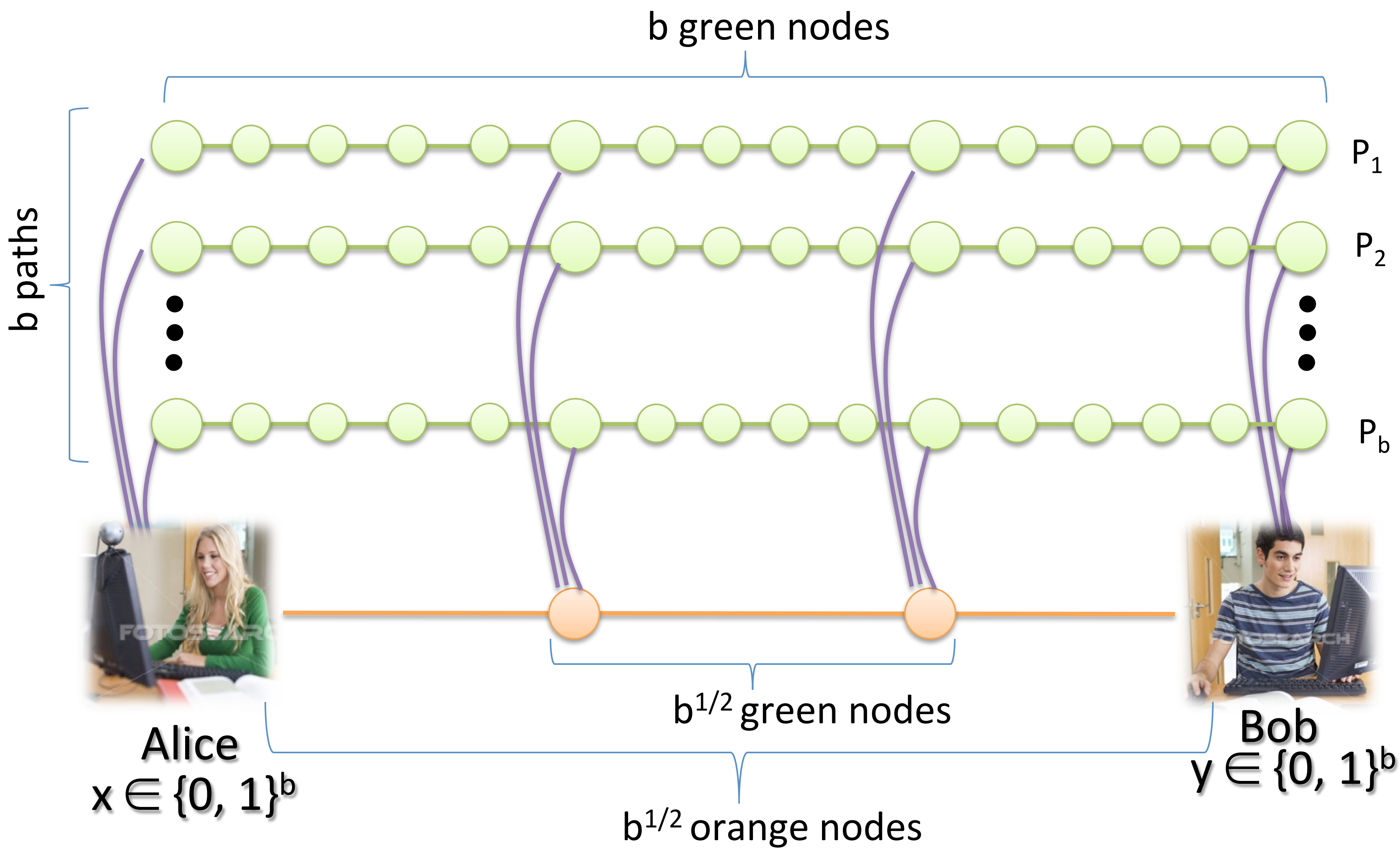
$x \in \{0, 1\}^b$

Bob

$y \in \{0, 1\}^b$

Direct Equality Verification

Distributed Equality Verification



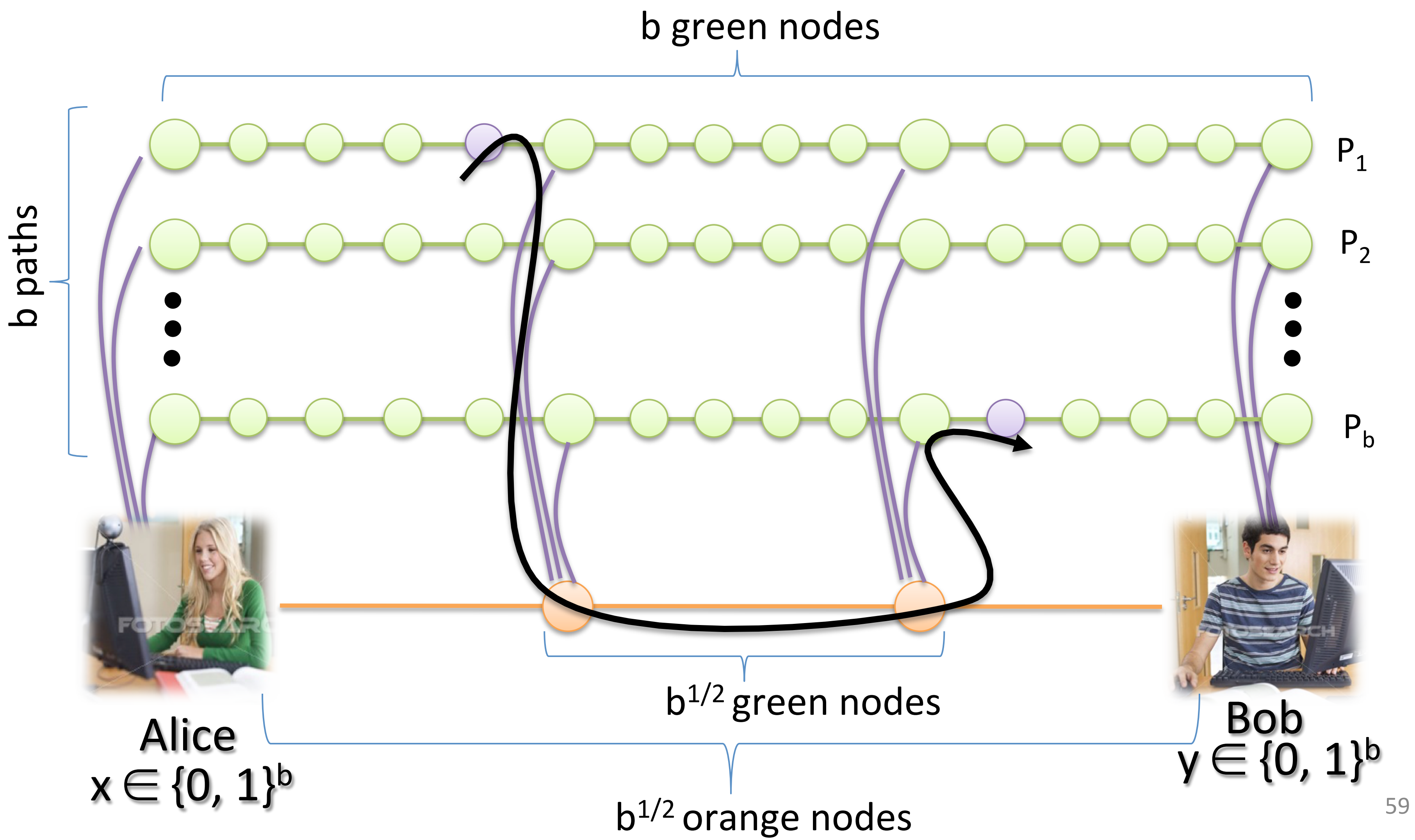
Notice:

$$n=O(b^2)$$

$$D=O(b^{1/2})=O(n^{1/4})$$

Direct Equality Verification

Distributed Equality Verification



Notice:

$$n=O(b^2)$$

$$D=O(b^{1/2})=O(n^{1/4})$$

Notice:

$D=O(n^{1/4})$

Distributed Equality Verification lower bound $\Omega(n^{1/2})$

Part 3.2

↓

ST verification lower bound $\Omega(n^{1/2})$

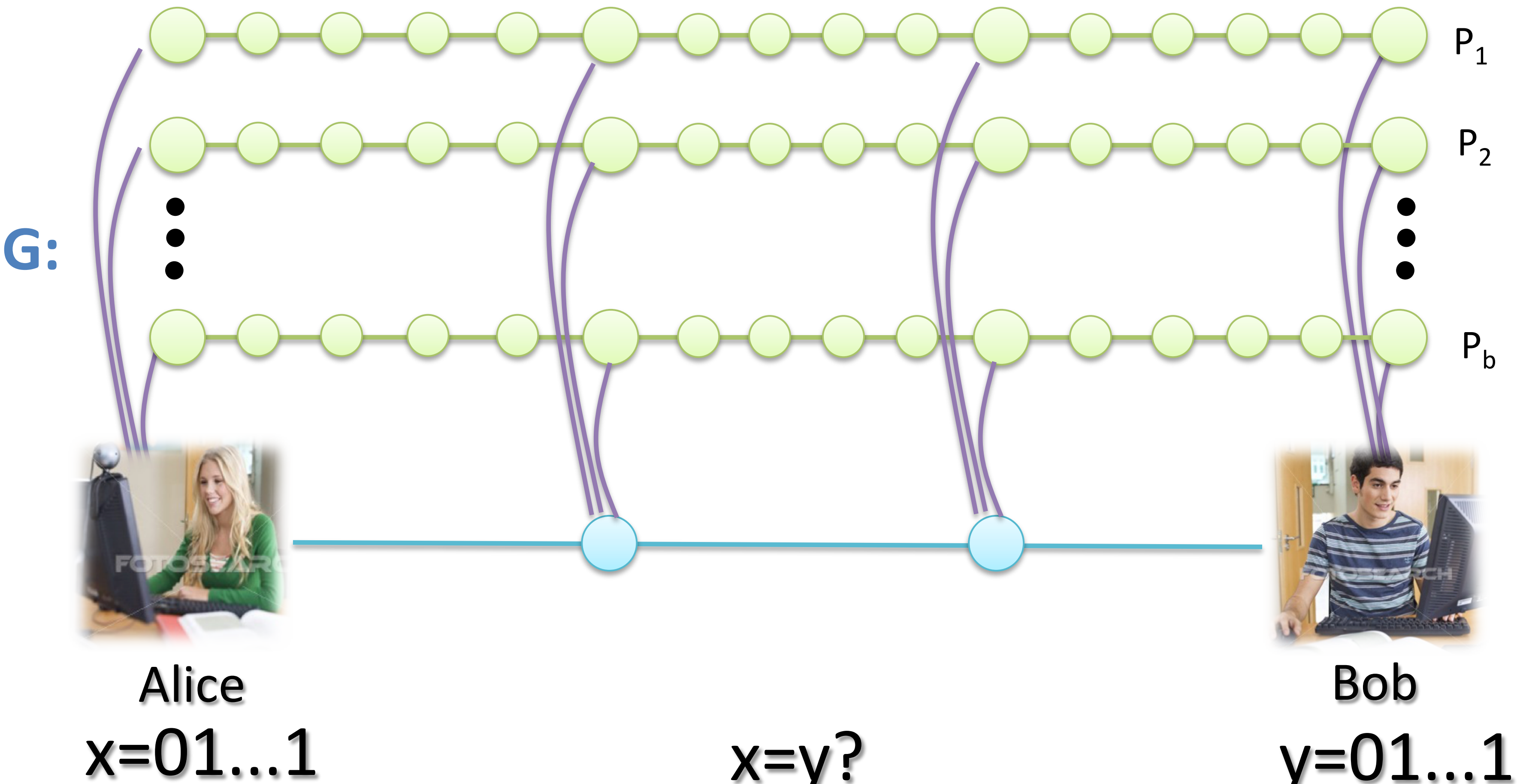
Part 3.2

ST verification
on G(b) in **$O(n^{0.49})$** time

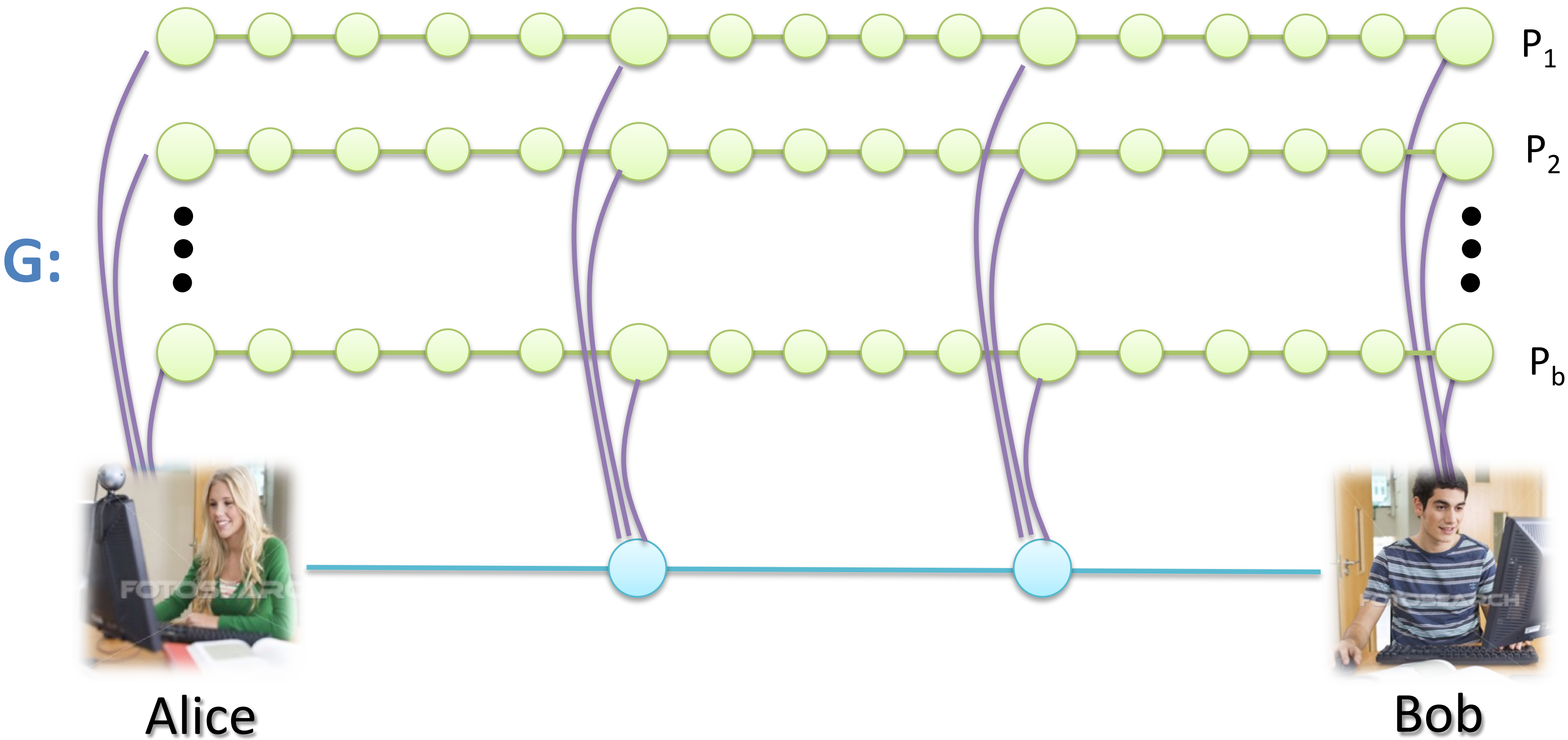
⬇

Distributed equality verification
on G(b) in **$O(n^{0.49})$** time

Let *A* be an algorithm for ST verification that runs in $O(n^{0.49})$ time



We will define subgraph H based on x and y

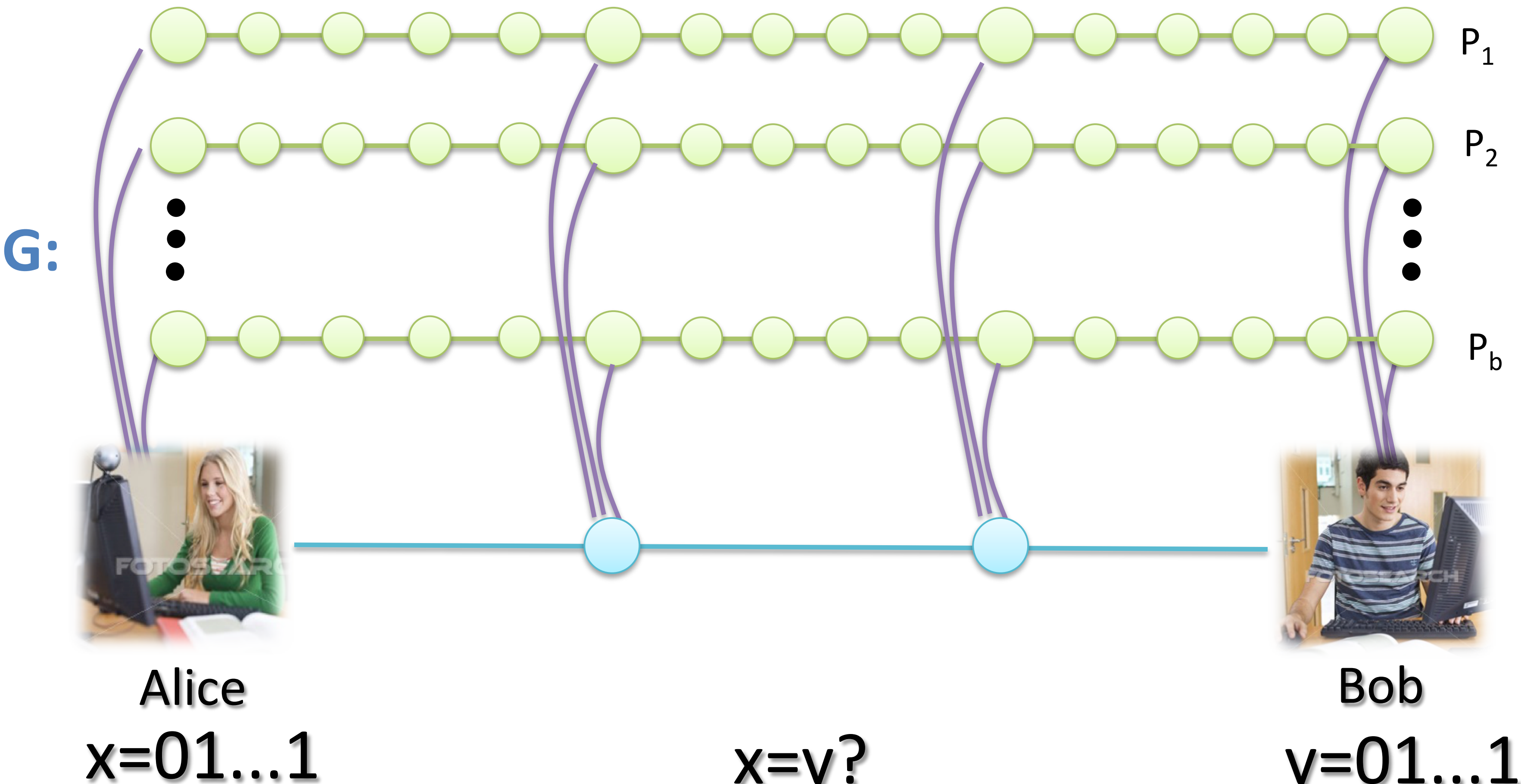


x=01...1

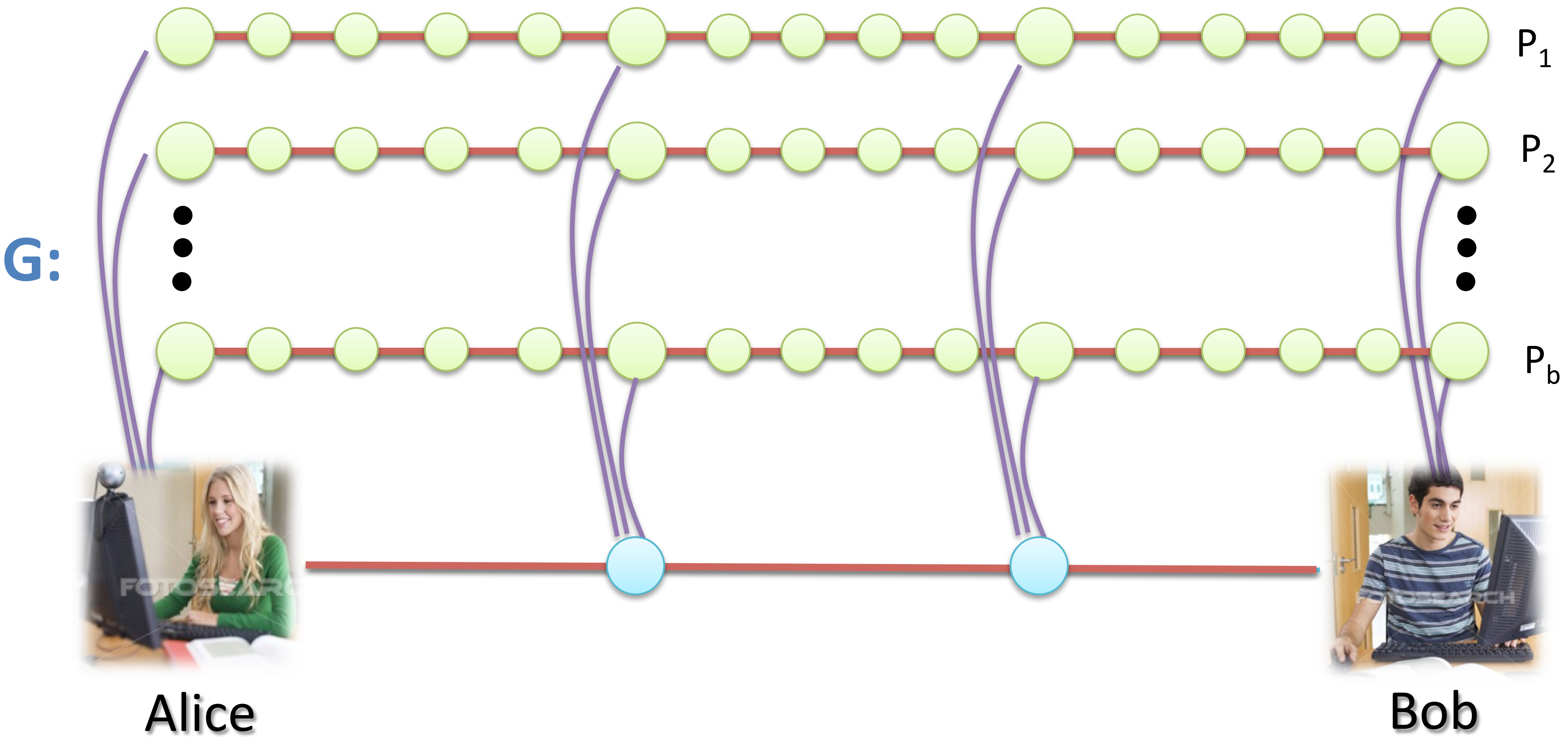
x=y?

y=01...1

1. All edges in all paths are in H

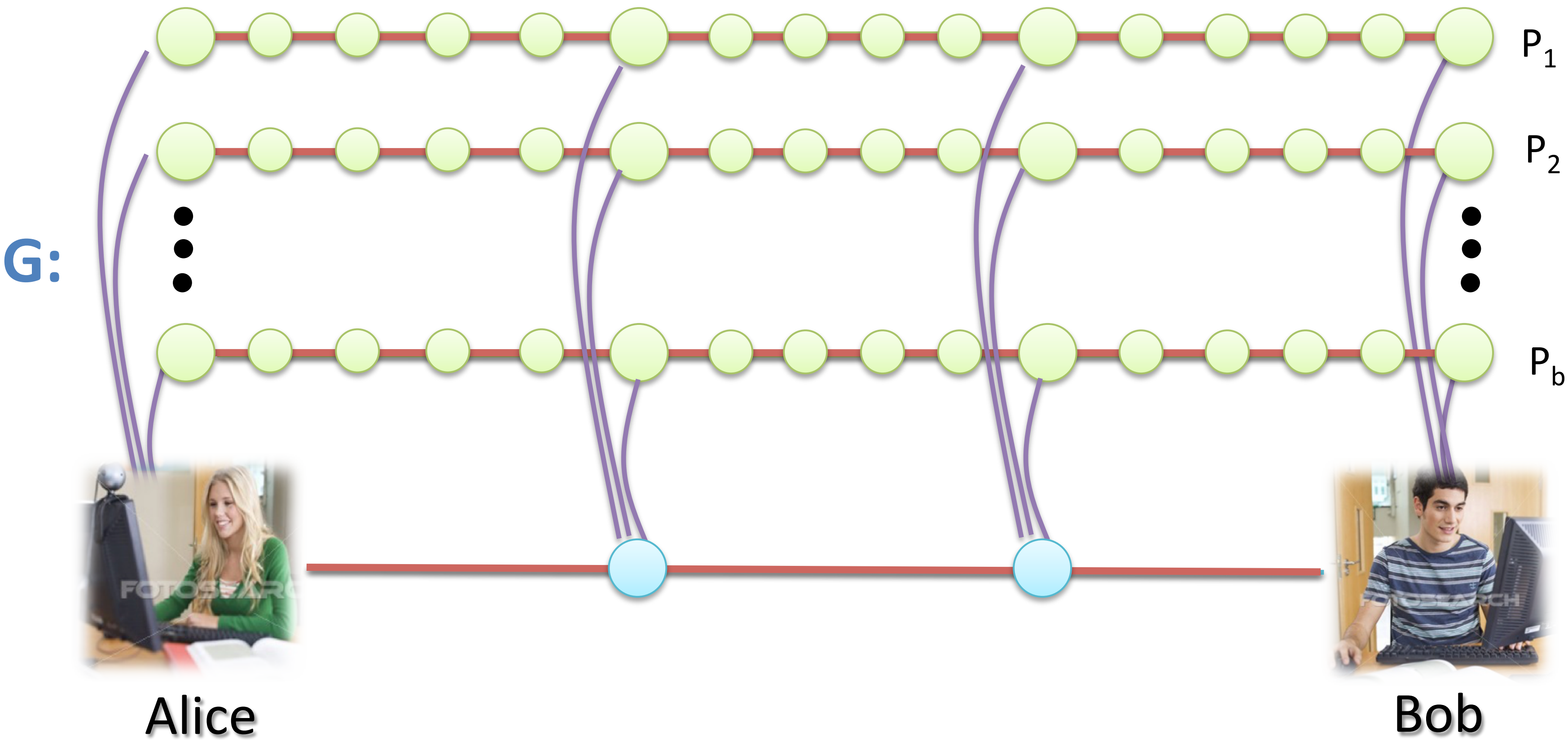


1. All edges in all paths are in H



x=01...1 x=y? y=01...1

2. Alice: all “0” edges are in H
Bob : all “1” edges are in H

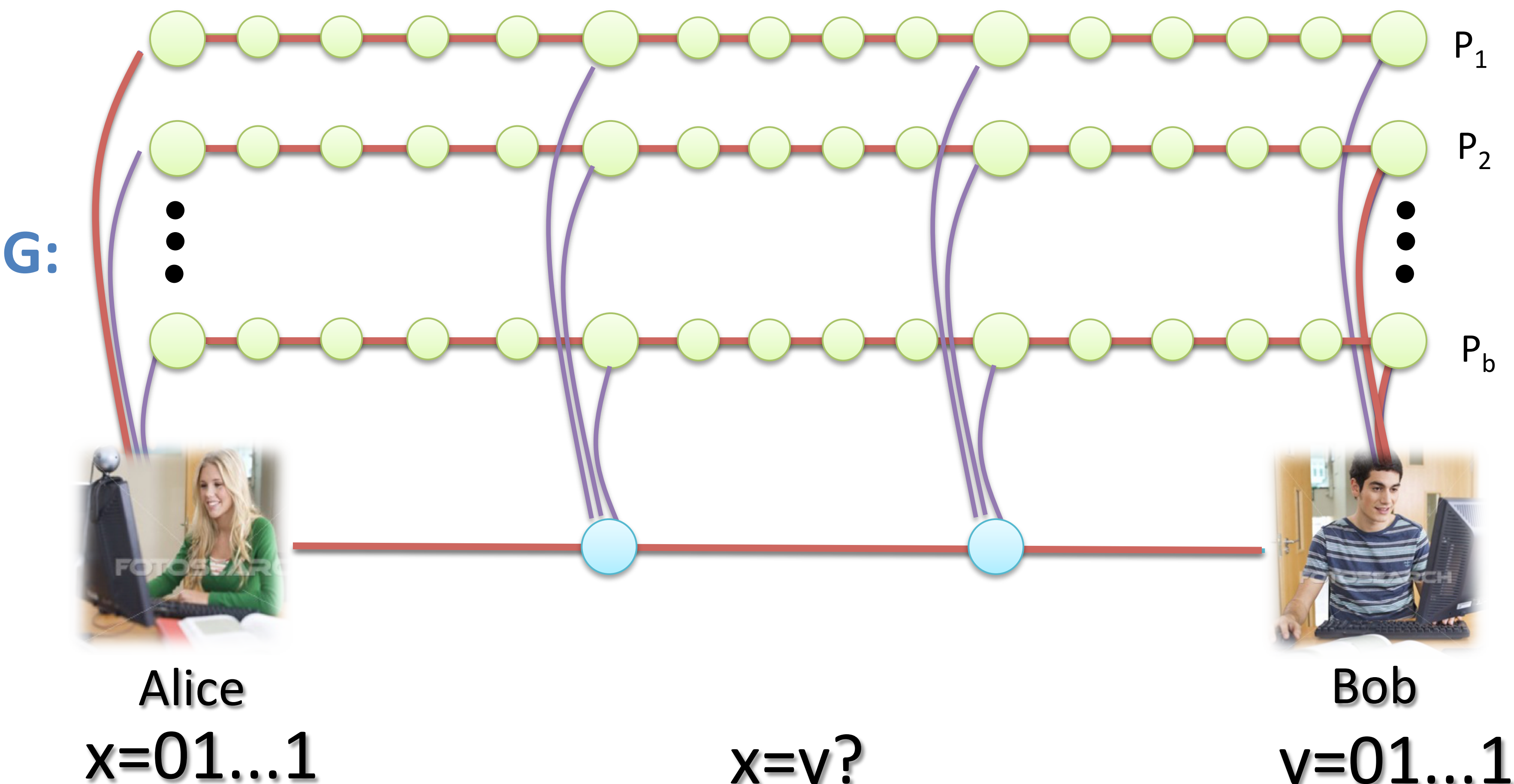


x=01...1

x=y?

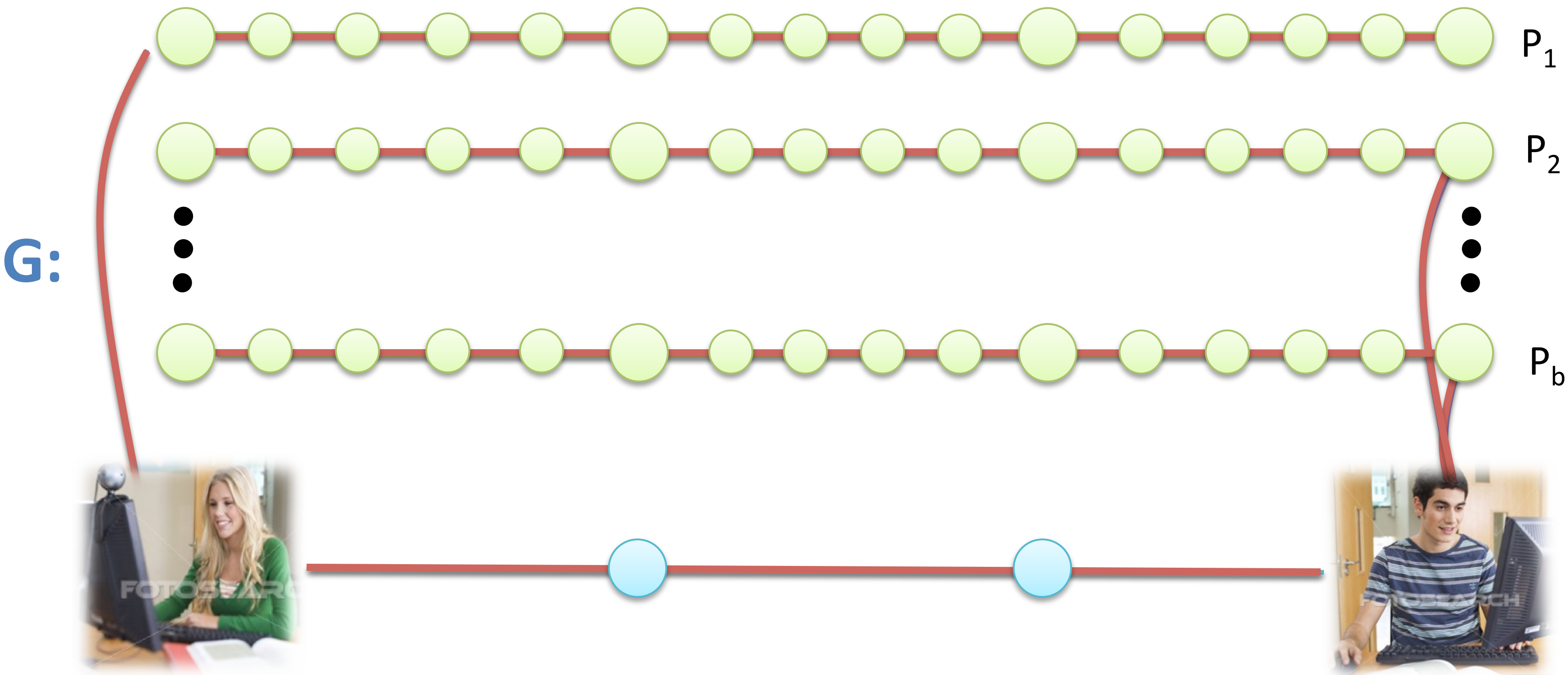
y=01...1

2. Alice: all "0" edges are in H
Bob : all "1" edges are in H



Observation 1:

If x=y then H is a spanning tree



Alice

Bob

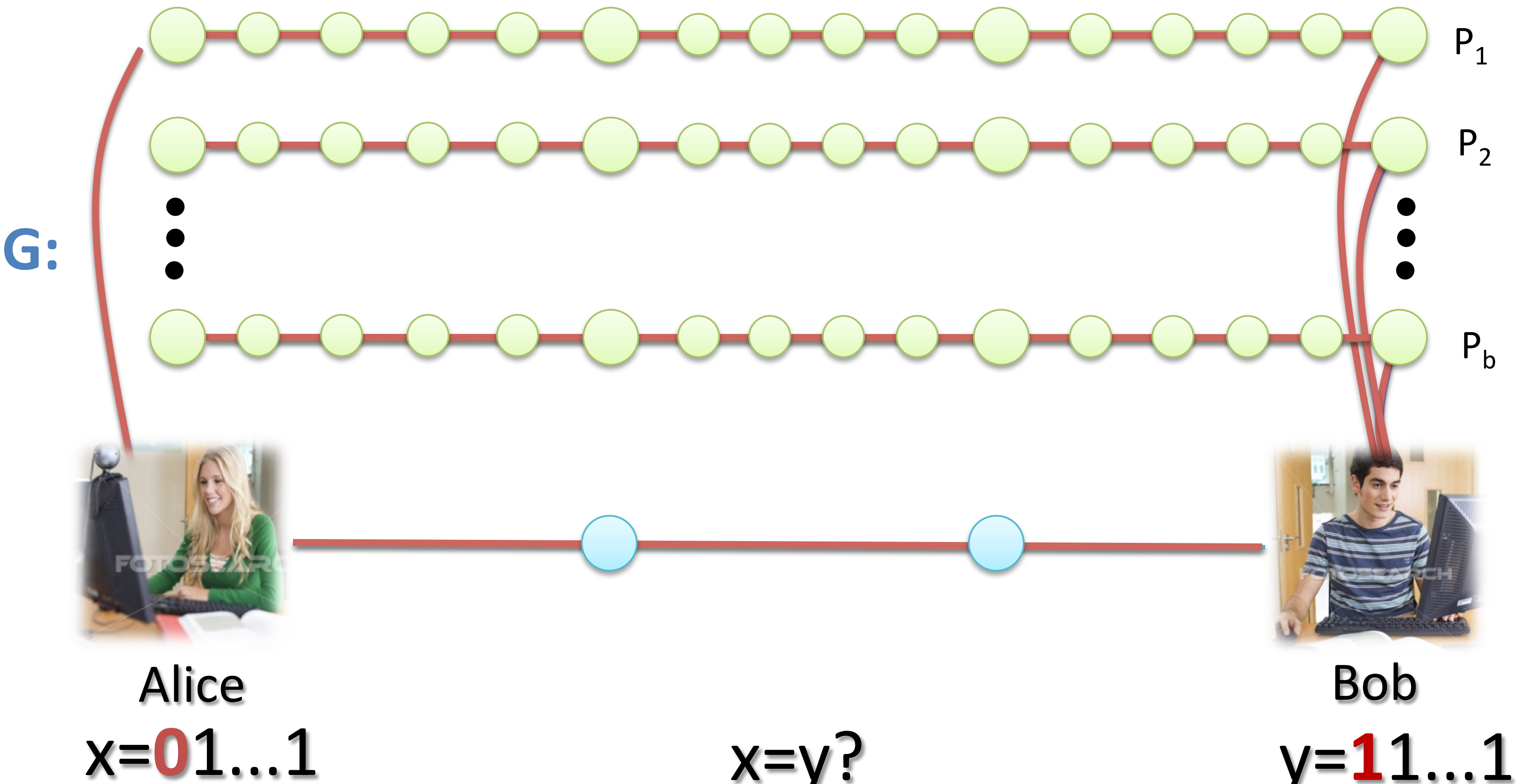
x=01...1

x=y?

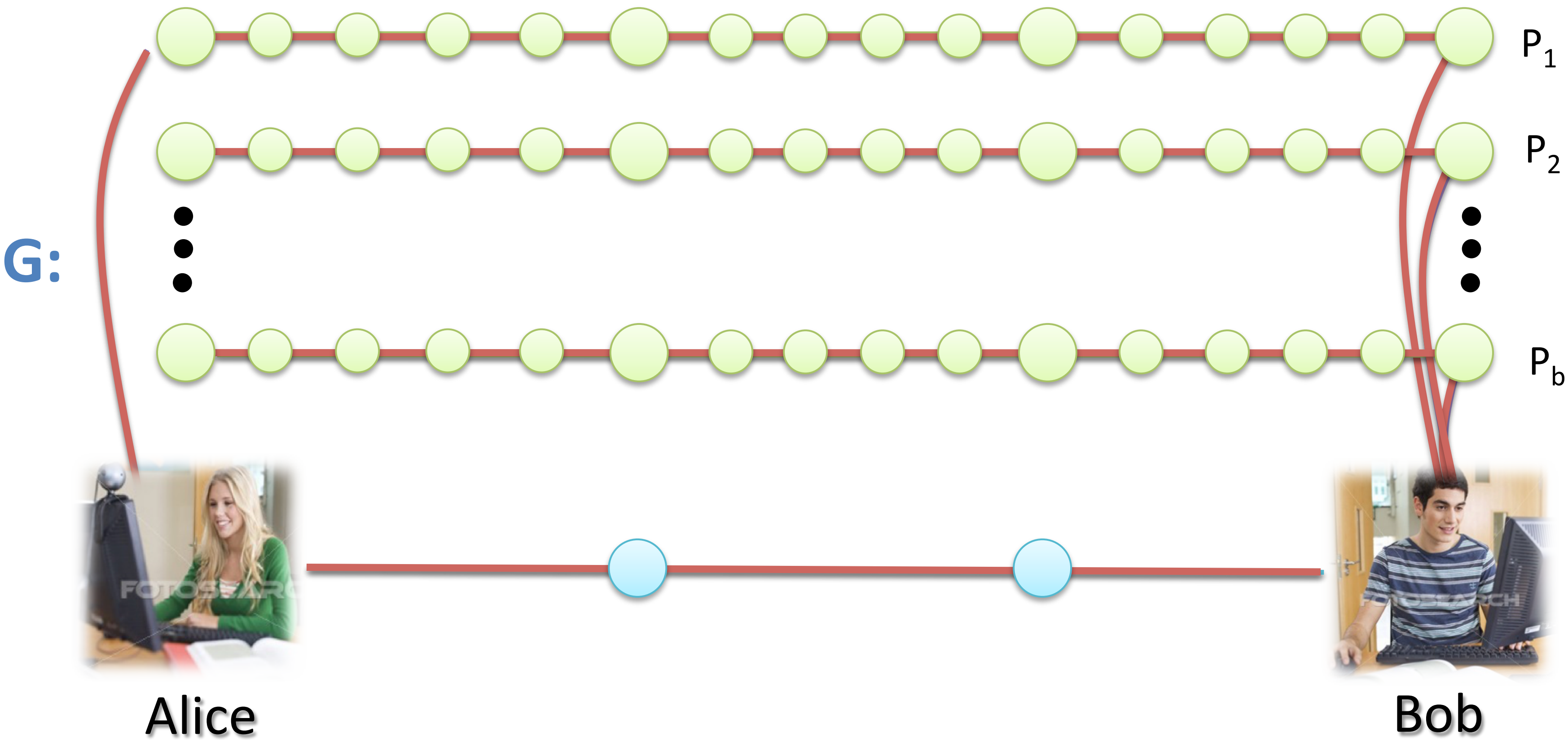
y=01...1

Observation 2:

If x≠y then H is NOT a spanning tree



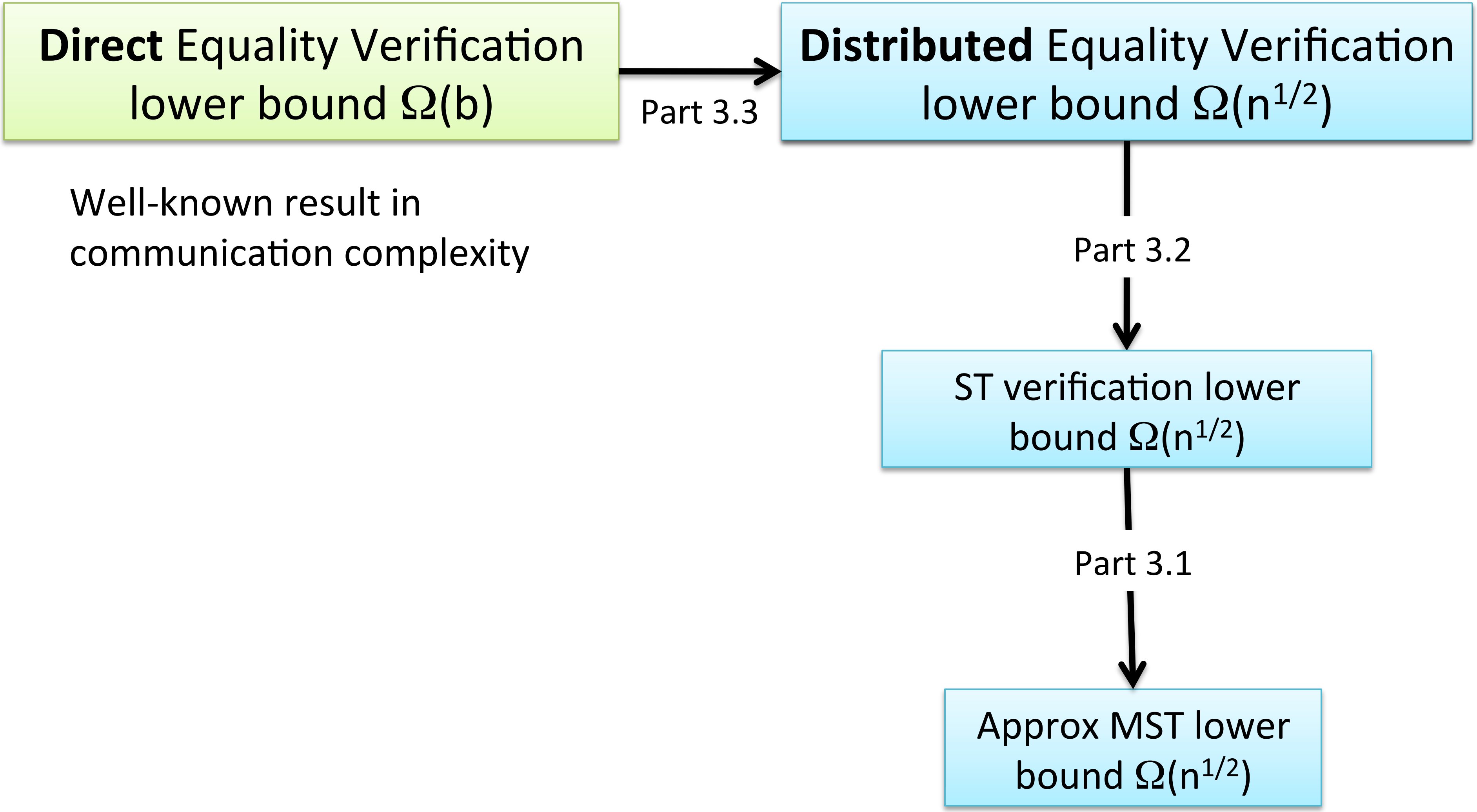
So, run **A** to verify whether H is a spanning tree



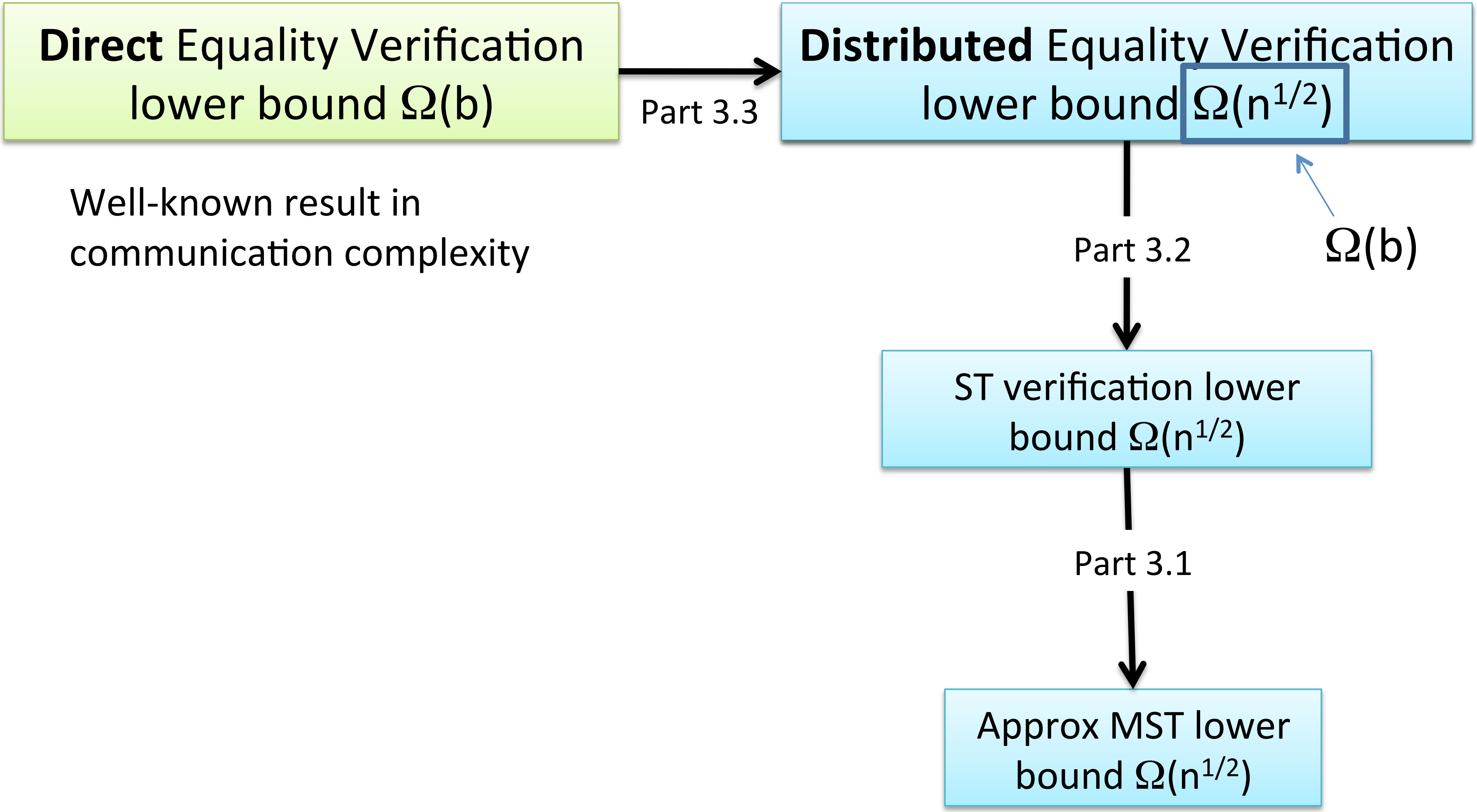
x=01...1

x=y?

y=**1**1...1



Direct Equality Verification lower bound Ω(b)
Part 3.3
Distributed Equality Verification lower bound Ω(n^1/2)
Well-known result in communication complexity
Part 3.2
ST verification lower bound Ω(n^1/2)
Part 3.1
Approx MST lower bound Ω(n^1/2)



Direct Equality Verification lower bound Ω(b)
Part 3.3
Distributed Equality Verification lower bound Ω(n^1/2)
Well-known result in communication complexity
Part 3.2
Ω(b)
ST verification lower bound Ω(n^1/2)
Part 3.1
Approx MST lower bound Ω(n^1/2)

Direct Equality Verification
lower bound $\Omega(b)$

Part 3.3

Distributed Equality Verification
lower bound $\Omega(n^{1/2})$

Well-known result in communication complexity

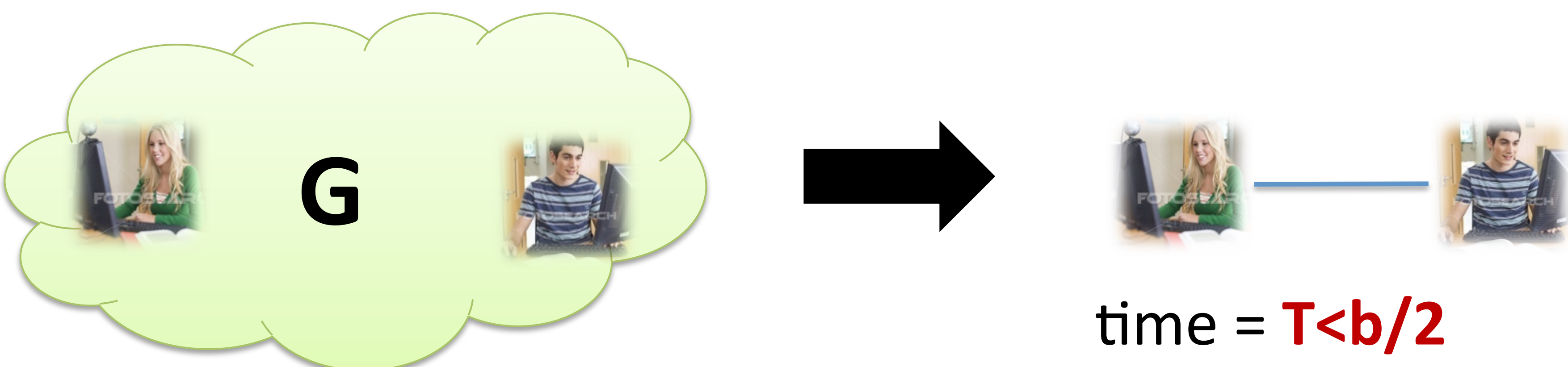
$\Omega(b)$

The Simulation Theorem

Part 3.3

Simulation Theorem

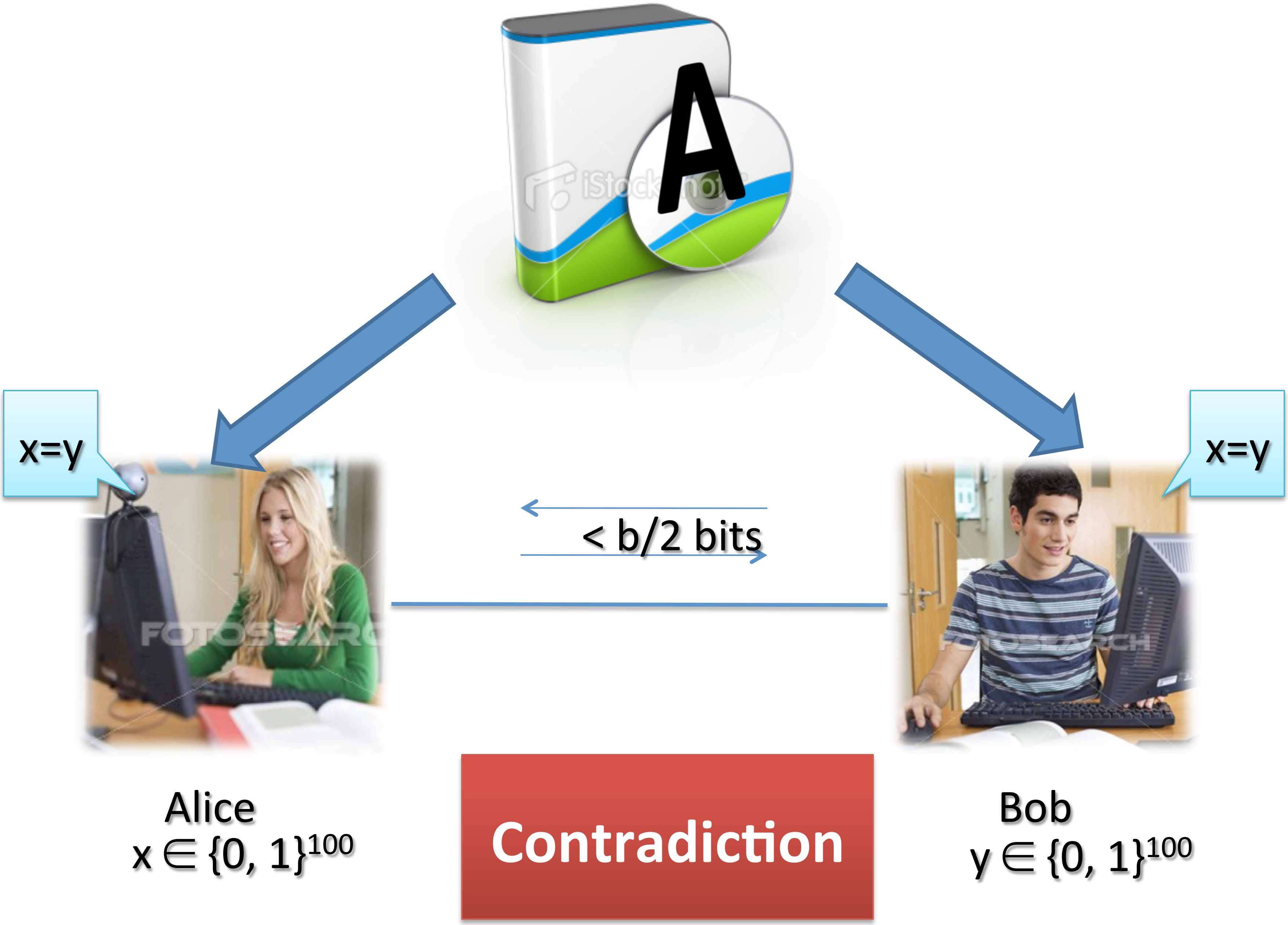
<u>If</u> the **distributed** equality verification can be solved in **T** days, for any **$T \leq b/2$**, <u>then</u> the **direct** version can be solved in **$\leq T$** days



time: **$T < b/2$**

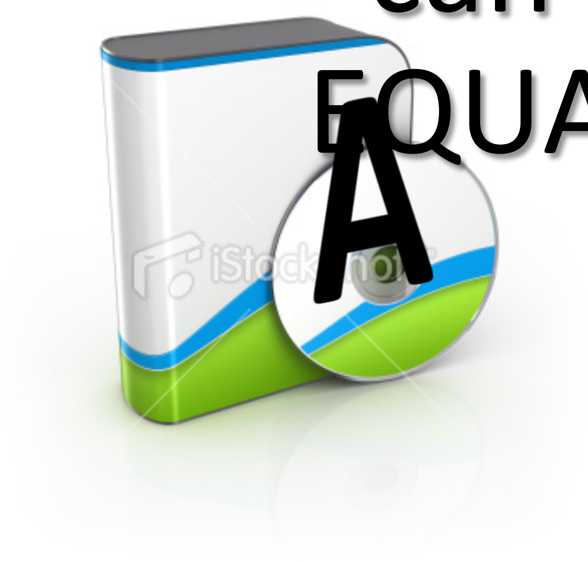
time = **$T < b/2$**
known: need **$\geq b$**
Contradiction!

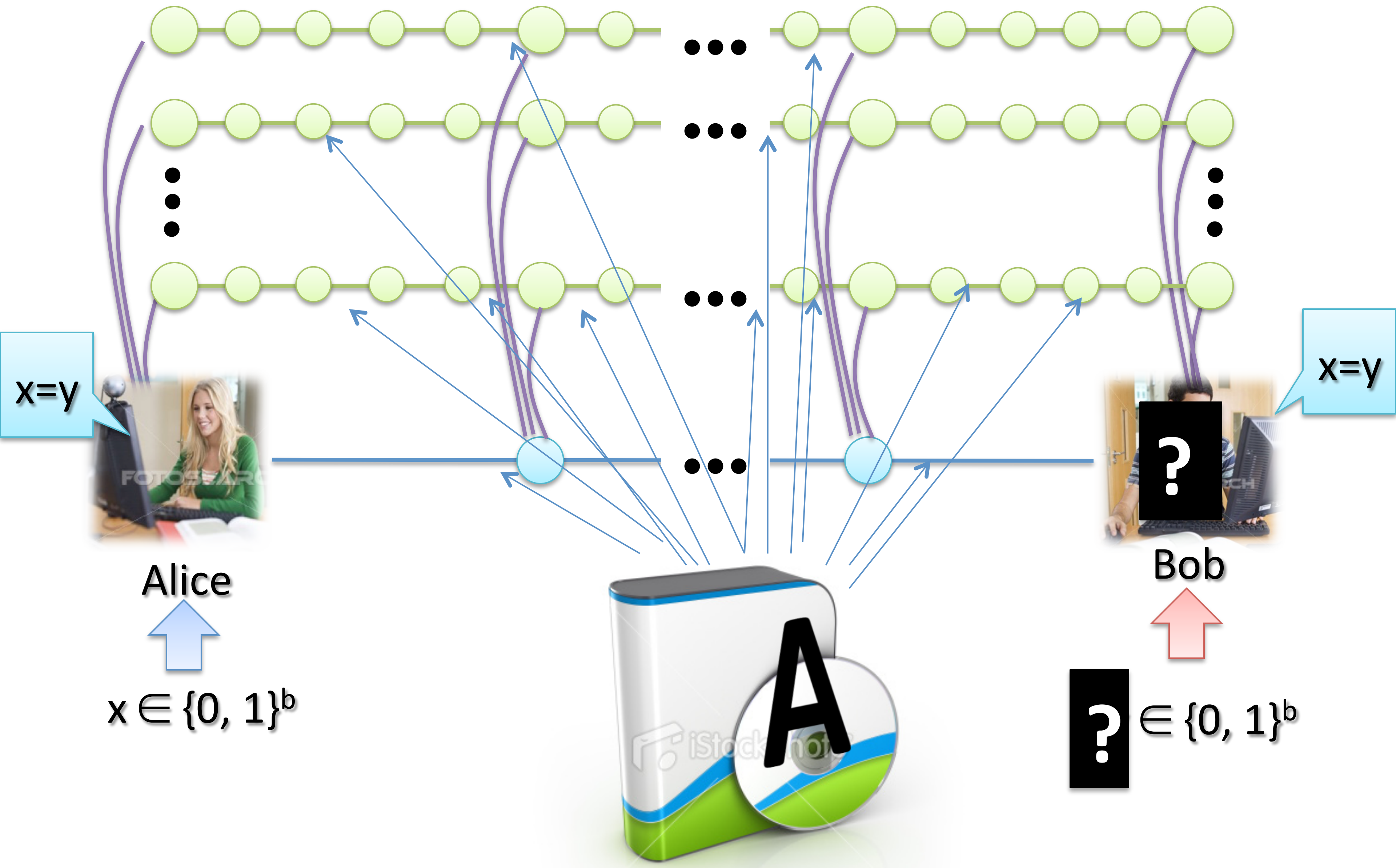
<u>Proof Idea:</u> Assume there is a distributed algorithm A that uses < b/2 steps


A
x=y
x=y
< b/2 bits
Alice
x ∈ {0, 1}^100
Contradiction
Bob
y ∈ {0, 1}^100

Proof: Assume there is a distributed algorithm A that uses <b/2 steps

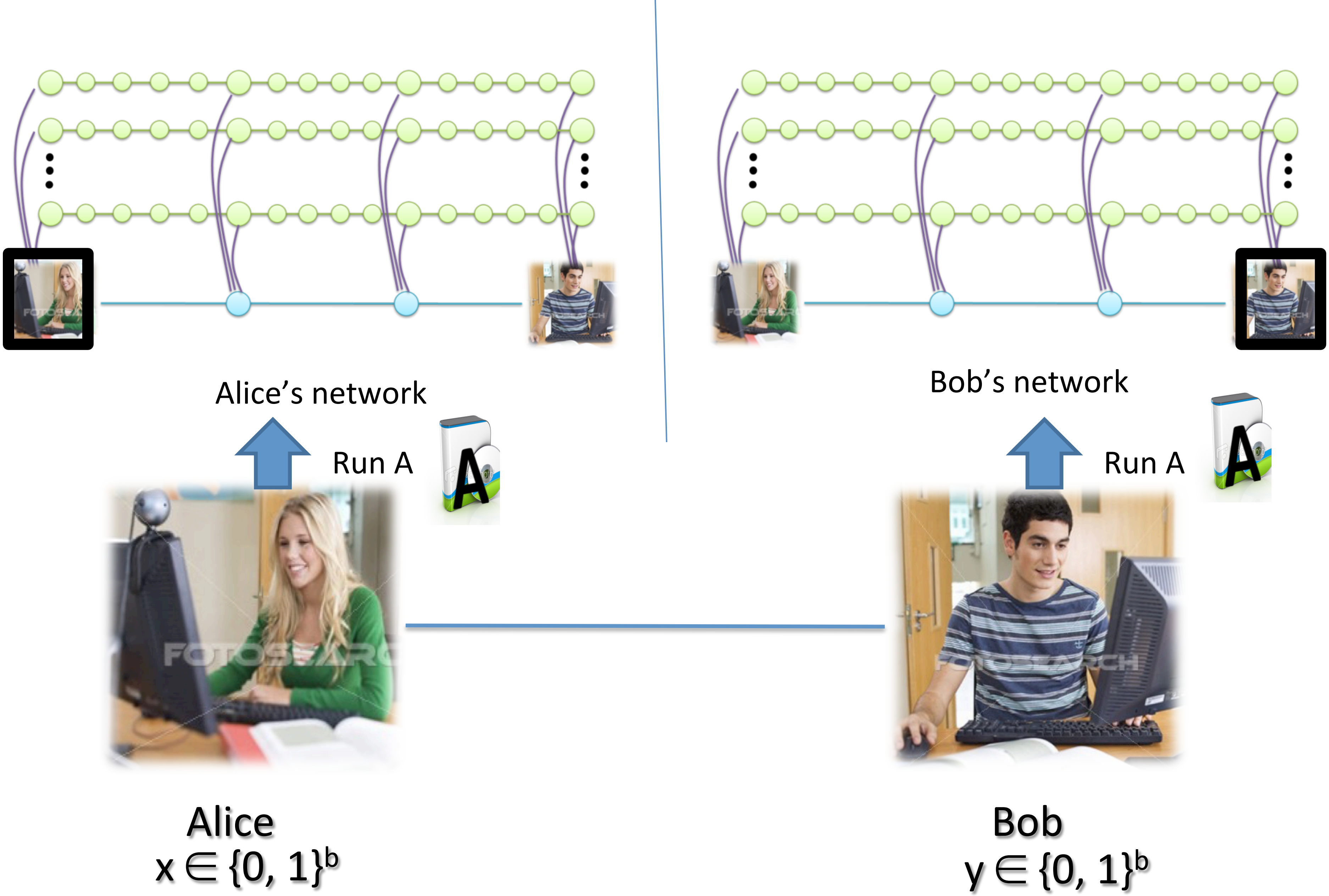
Goal: Show that Alice & Bob can use A to compute EQUALITY using b/2 bits

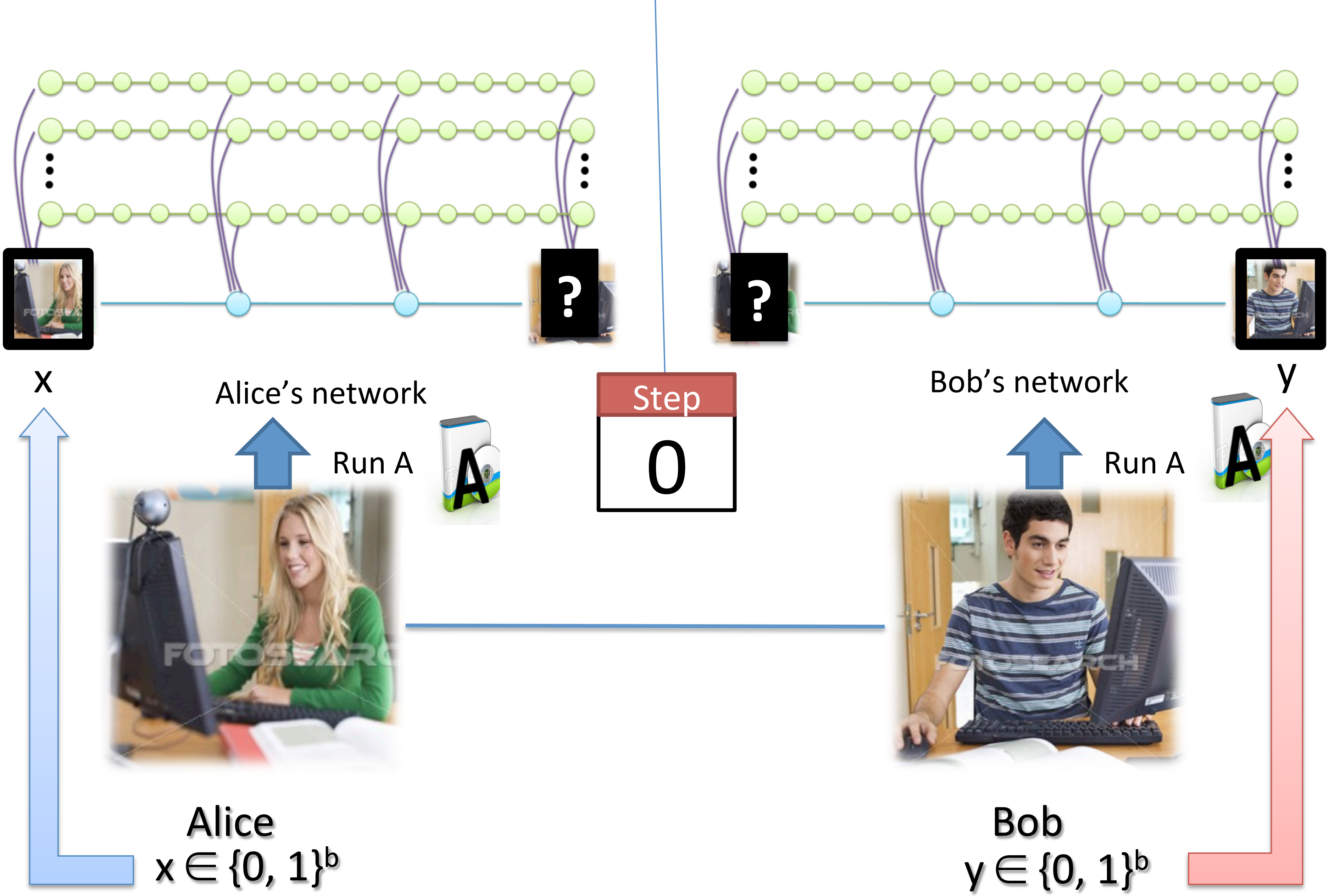



x=y
Alice
$x \in \{0, 1\}^b$
x=y
Bob
? $\in \{0, 1\}^b$

Proof of the Simulation theorem

- Let ***A*** be a distributed algorithm which runs in $T \le b/2$ time
- Alice and Bob will simulate *A* on their OWN networks
- They try to exchange minimum messages to keep their machines running as long as possible


Alice's network
Run A
Bob's network
Run A
Alice
$x \in \{0, 1\}^b$
Bob
$y \in \{0, 1\}^b$



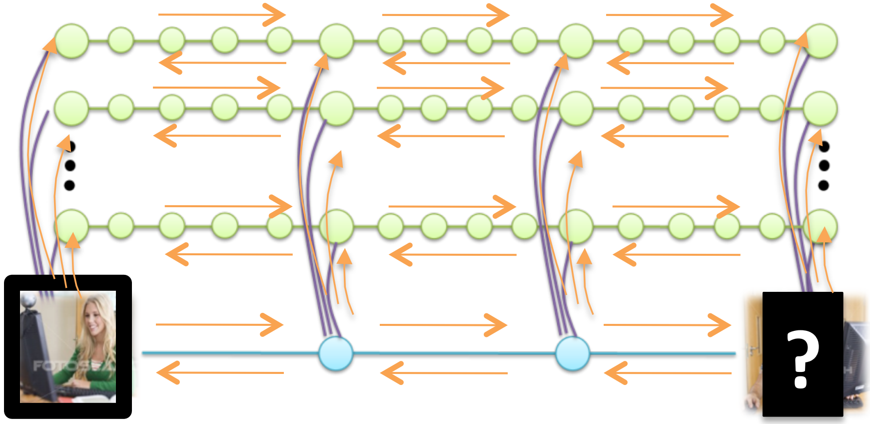
x
Alice's network
Run A
A
Step
0
Bob's network
y
Run A
A
Alice
$x \in \{0, 1\}^b$
Bob
$y \in \{0, 1\}^b$

In step 0, Alice can run A on all machines except Bob's

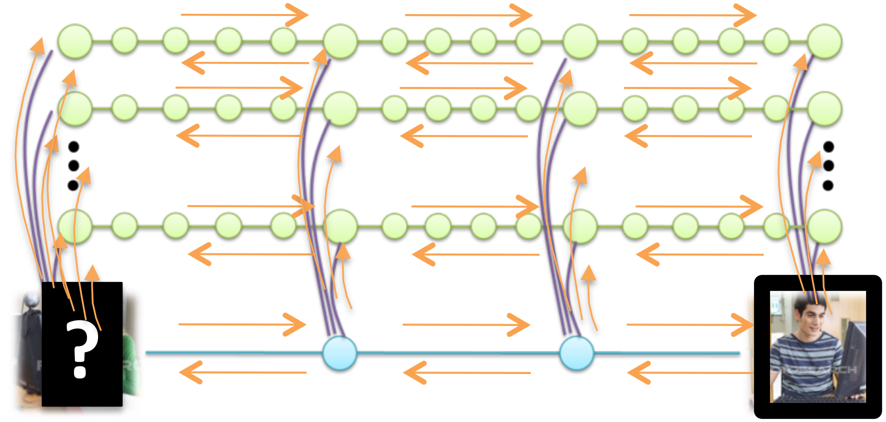
The following is an **intuition**.
It is NOT the real proof.

(intuition)

Observe: Alice and Bob can simulate *A* for $\mathbf{b}^{1/2}$ steps without exchanging messages



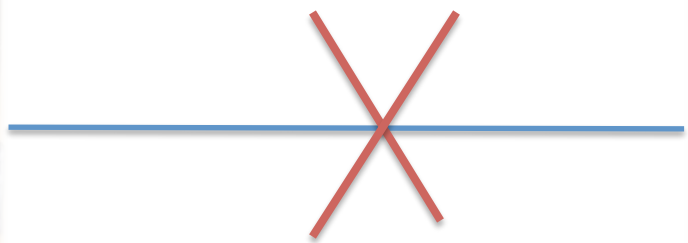
x



y

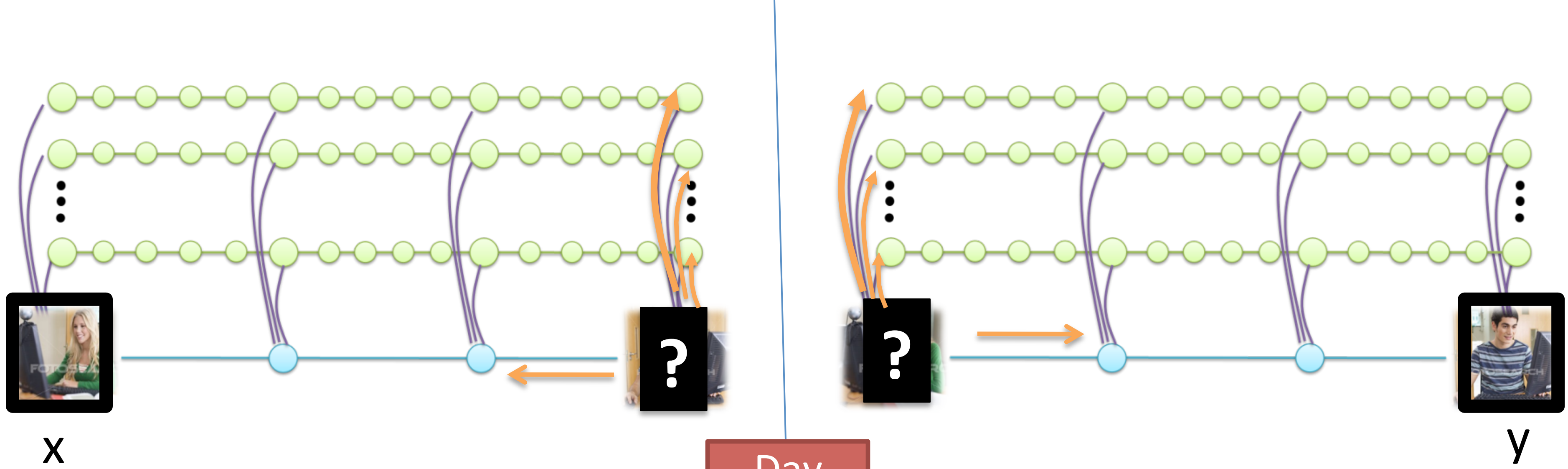
Day

1



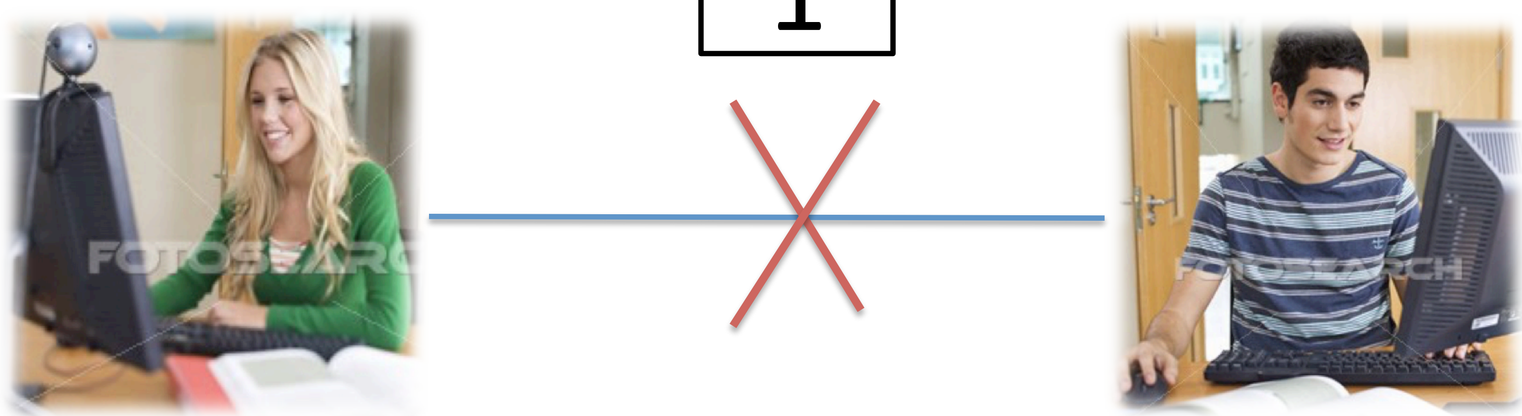
Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$



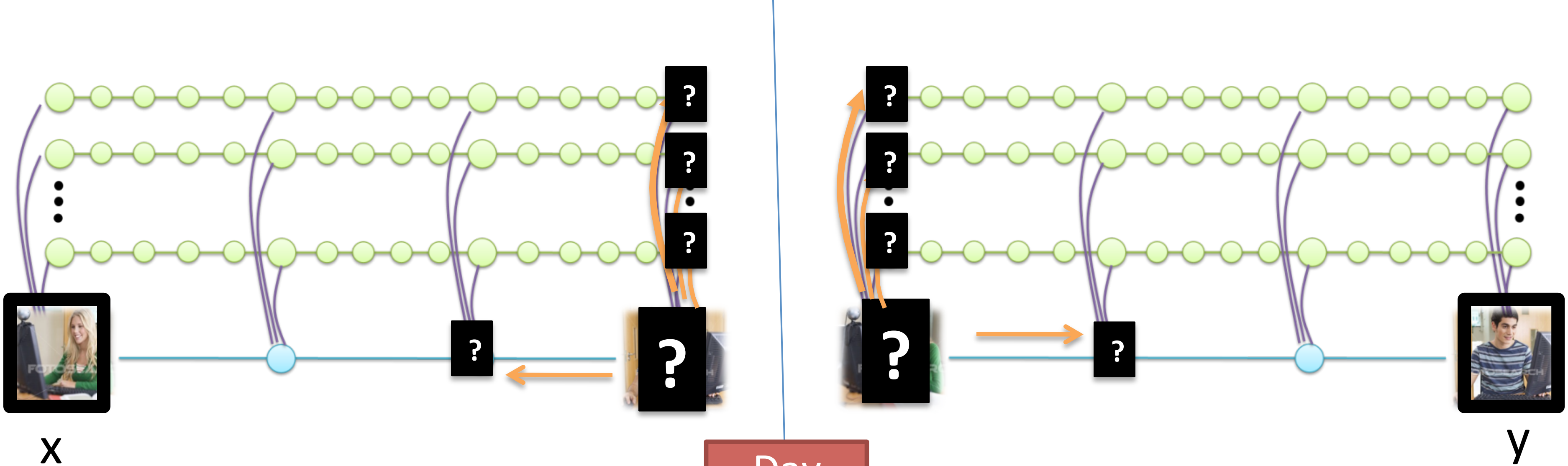
Day

1



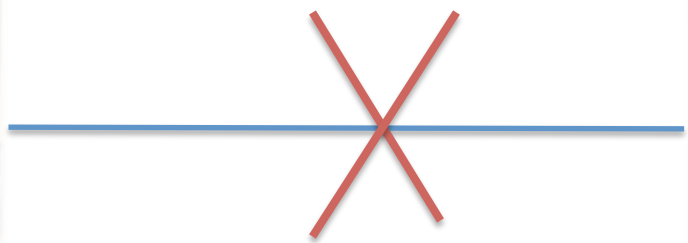
Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$



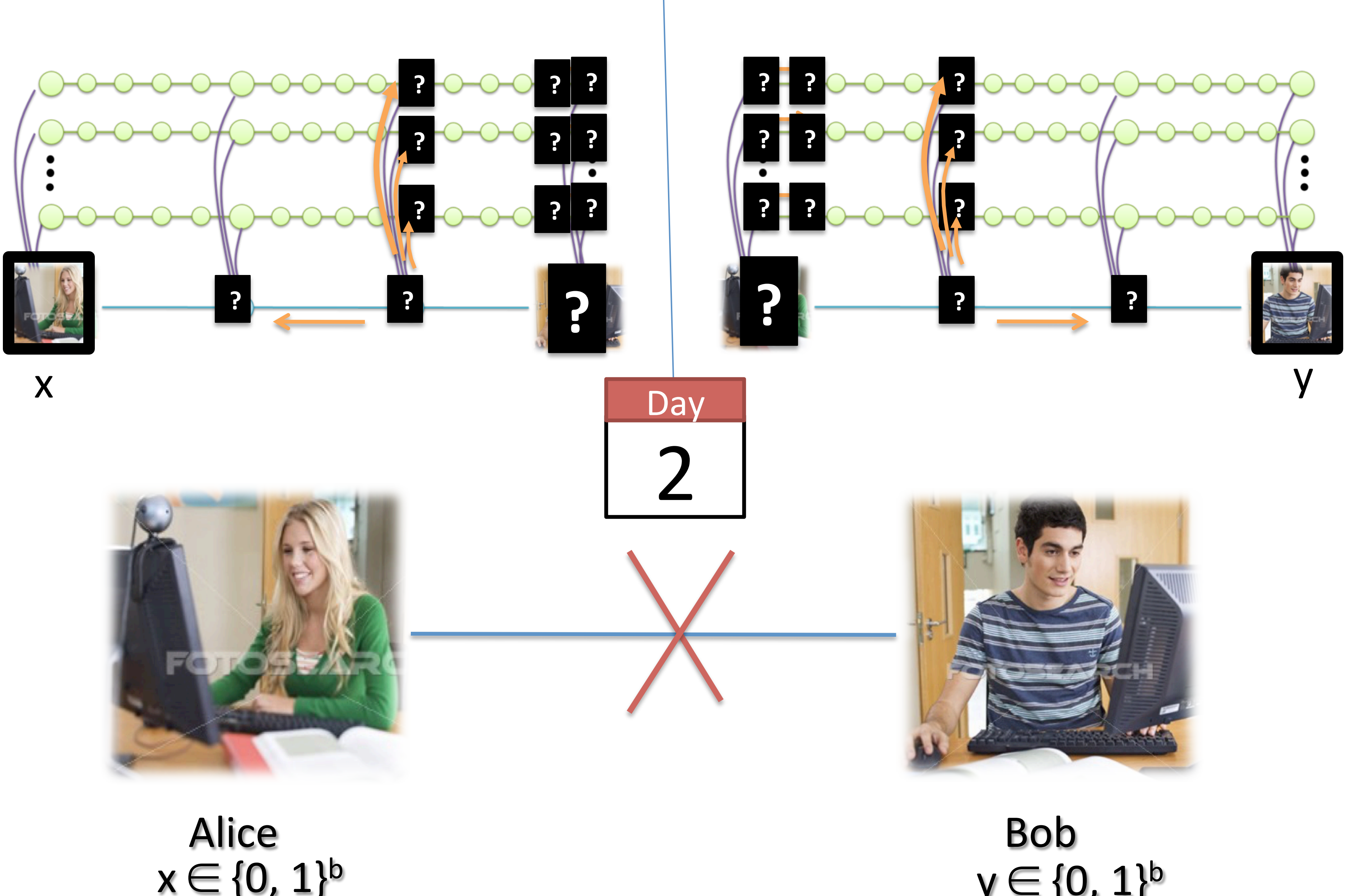
Day

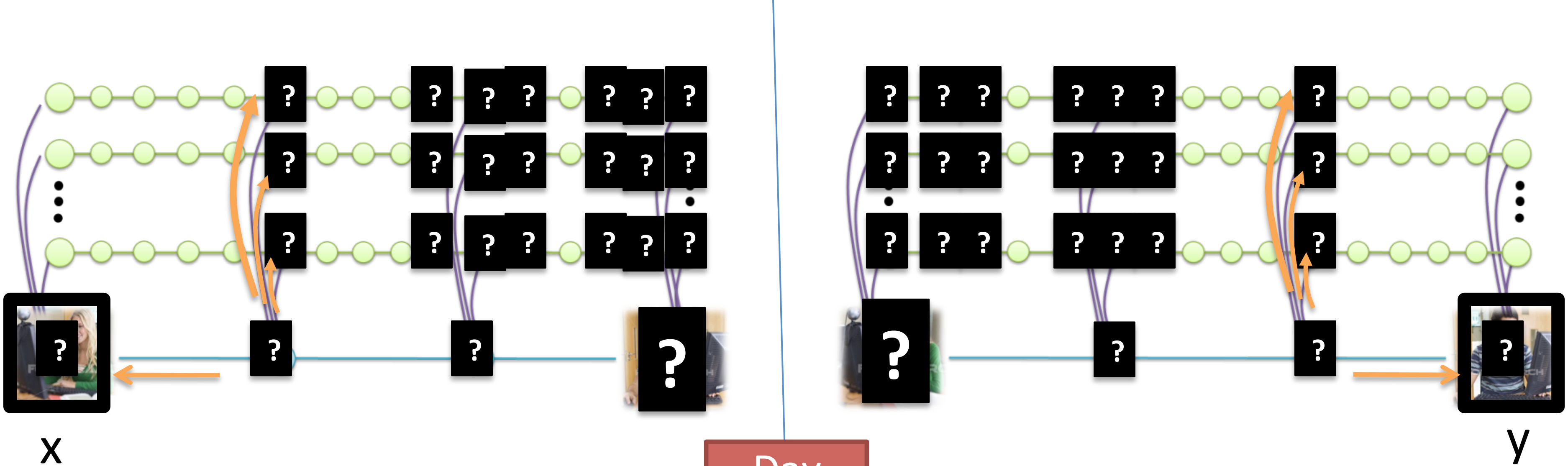
1

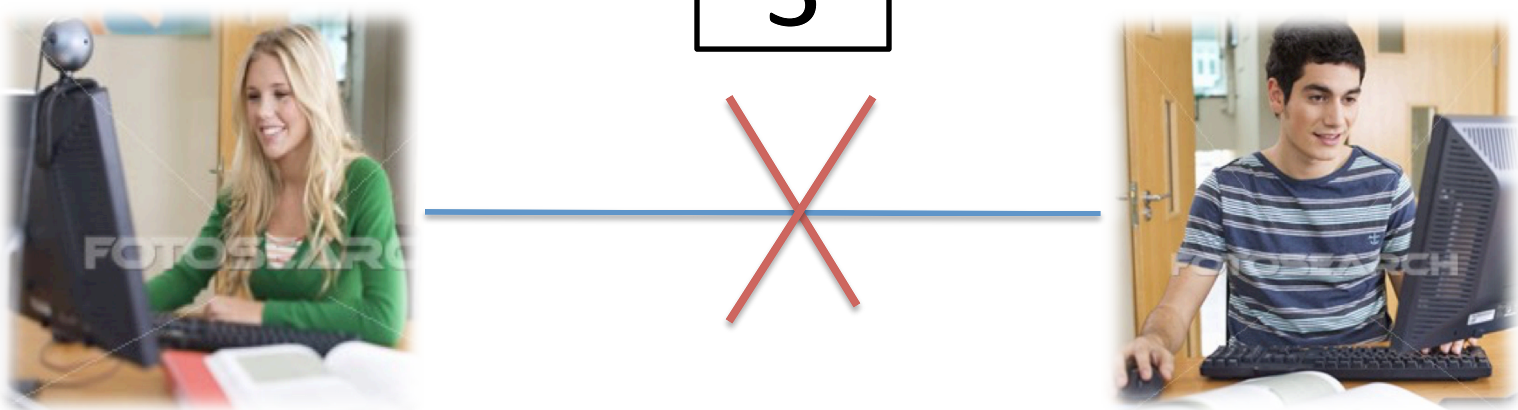


Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

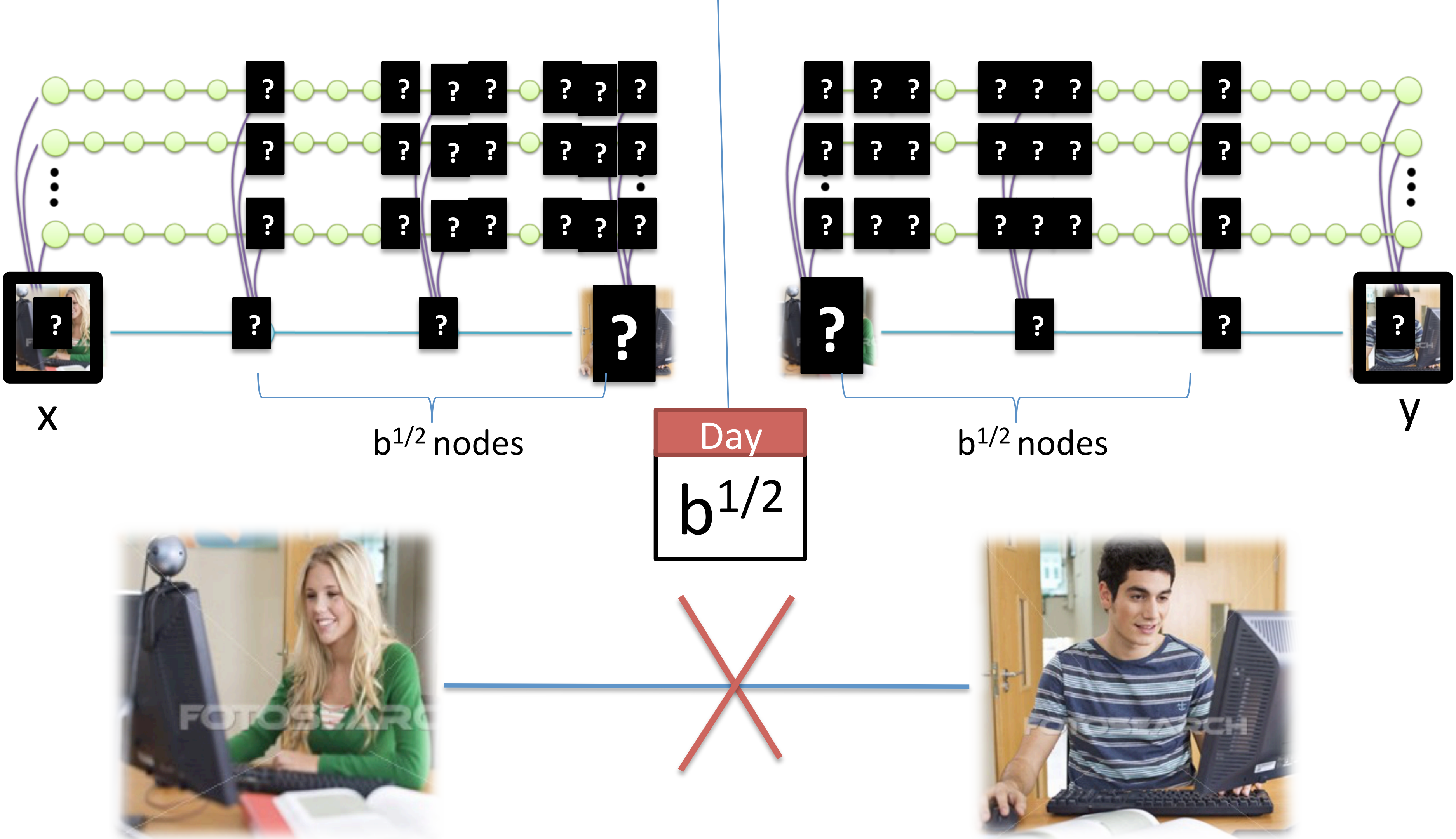

?
x
y
Day
2
Alice
$x \in \{0, 1\}^b$
Bob
$y \in \{0, 1\}^b$





Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$



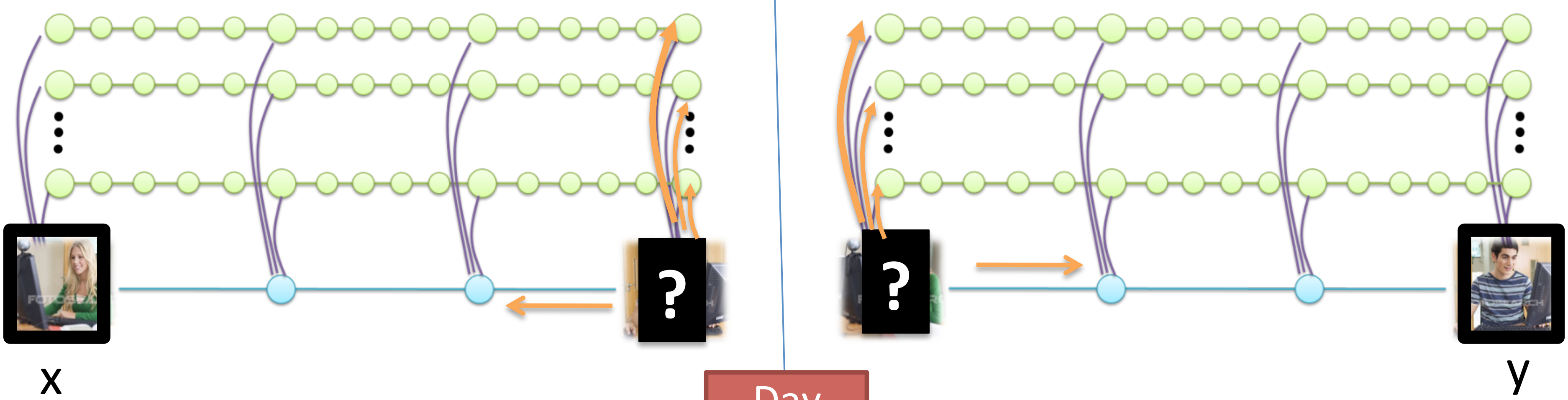
Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

(intuition)

<u>Observe</u>: Alice and Bob can simulate *A* for $\mathbf{b}^{1/2}$ steps without exchanging messages

Theorem: Alice and Bob can simulate *A* for **b/2** steps *by exchanging 1 bit per step*



Day

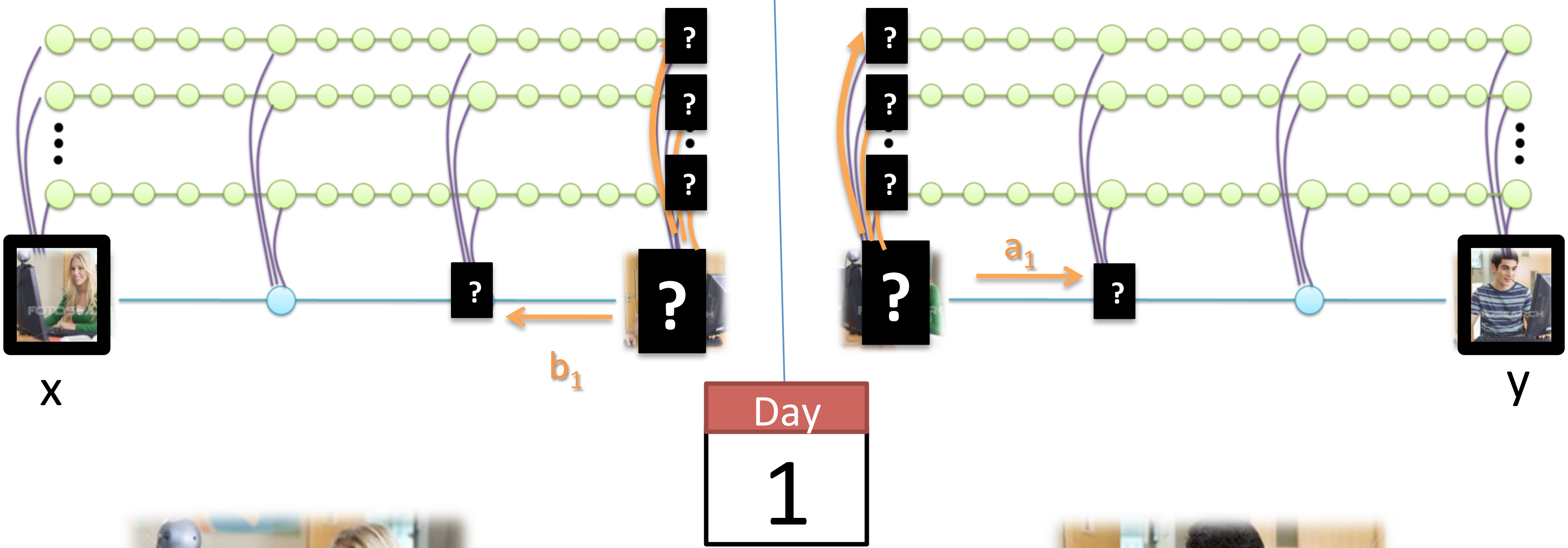
1

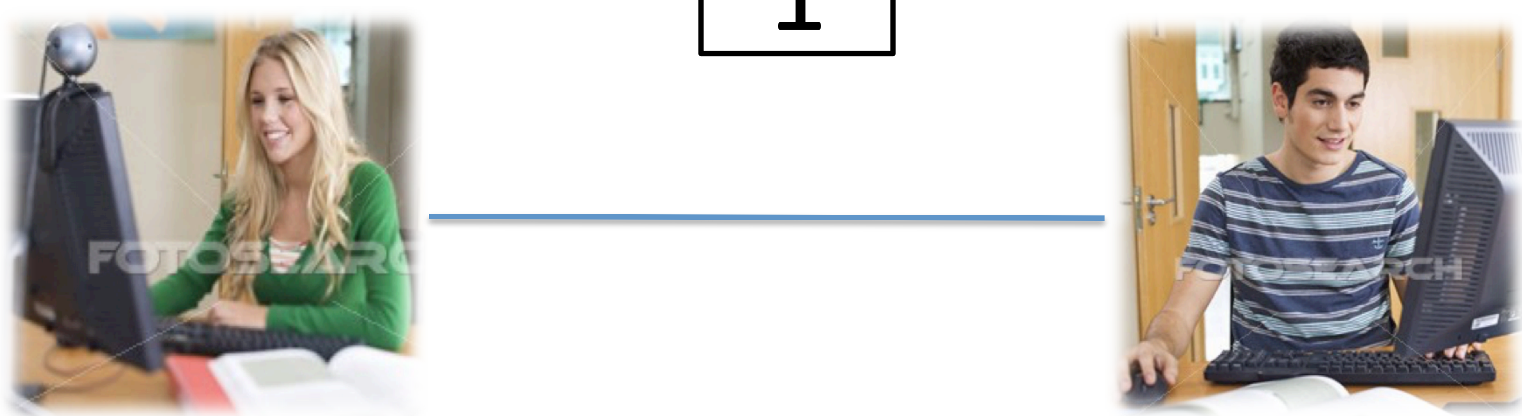
Alice

$x \in \{0, 1\}^b$

Bob

$y \in \{0, 1\}^b$

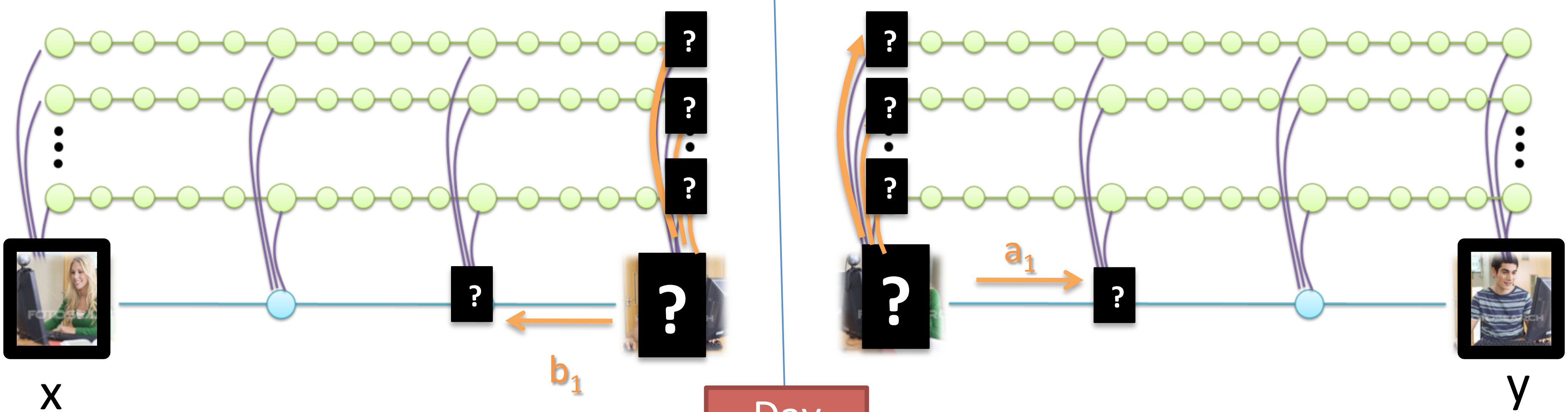




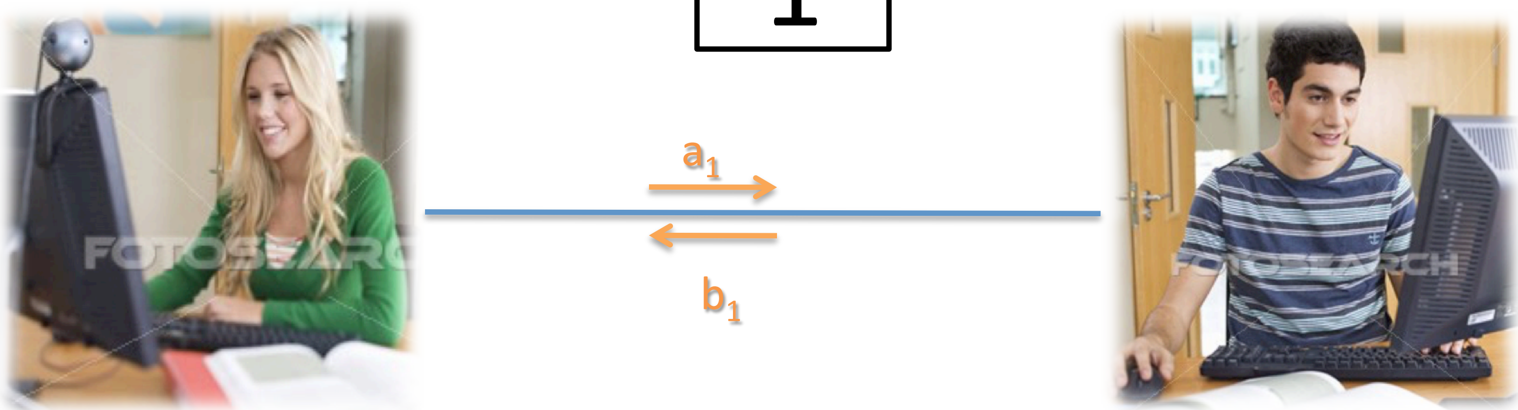
Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

b_1 = bit sent by A run on Bob's machine



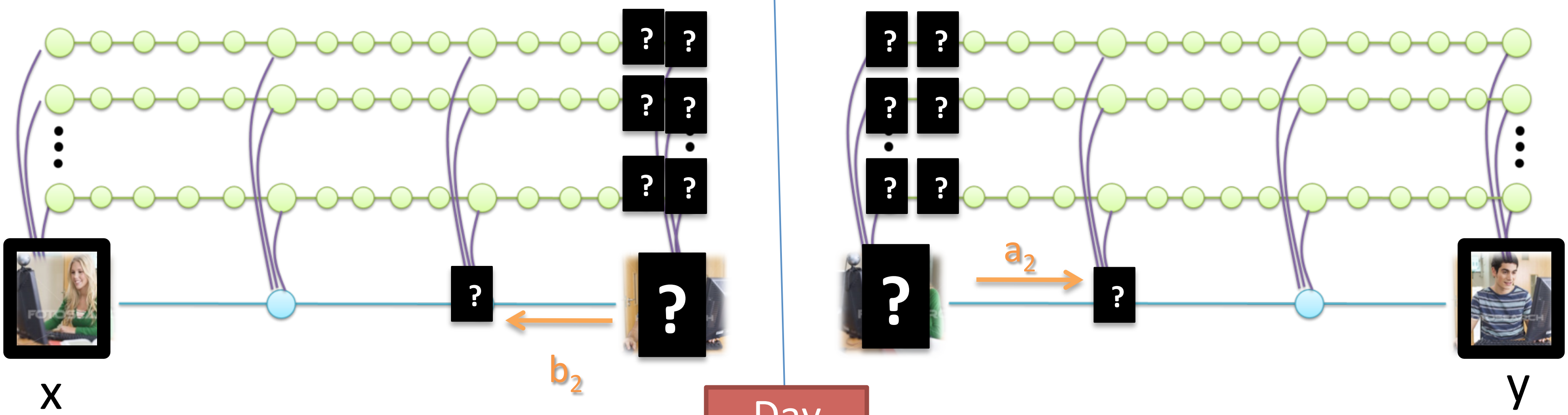
Day
1



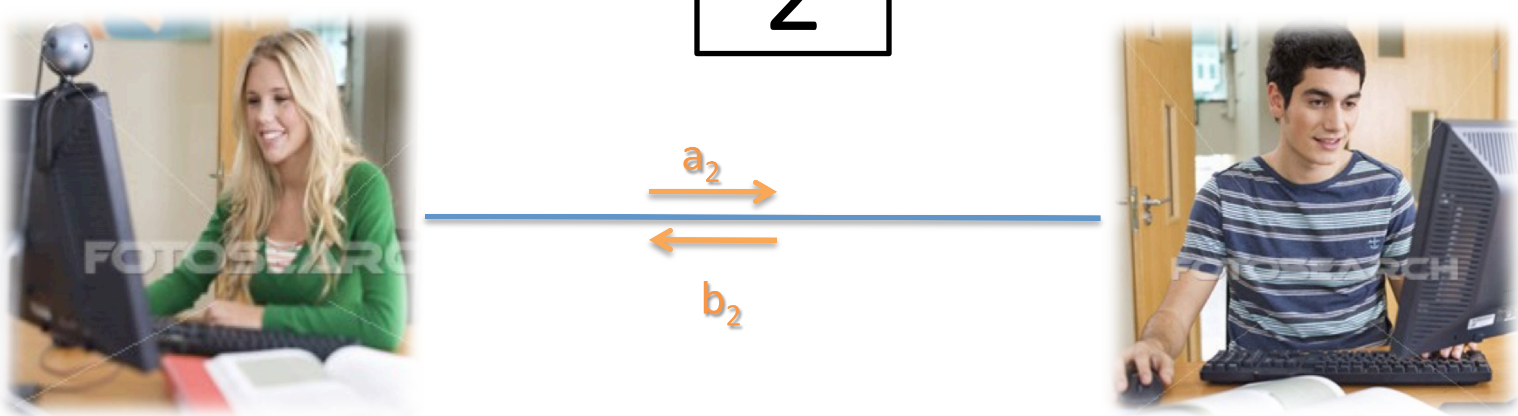
Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

b_1 = bit sent by A from Bob's machine



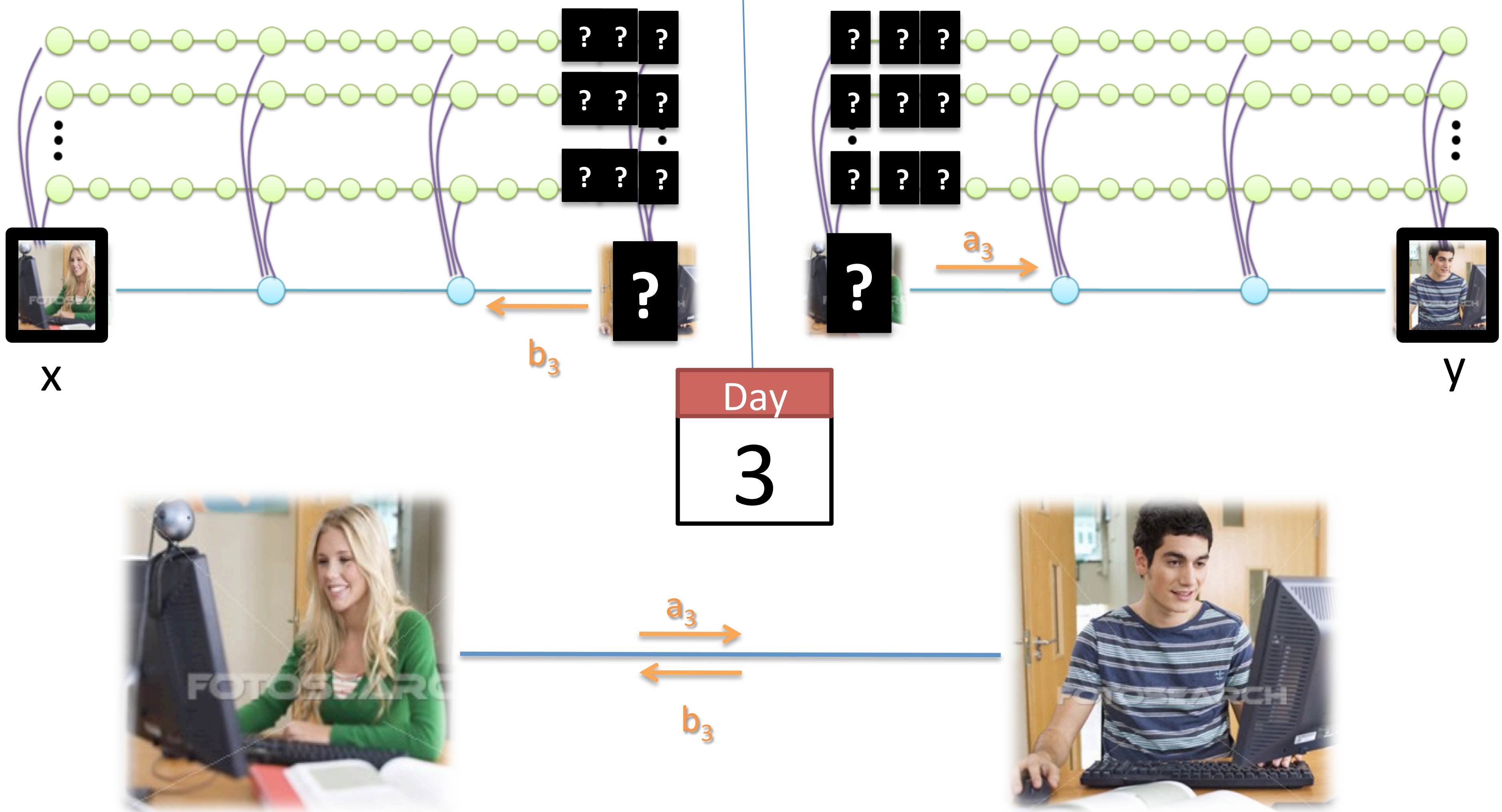
Day
2



Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

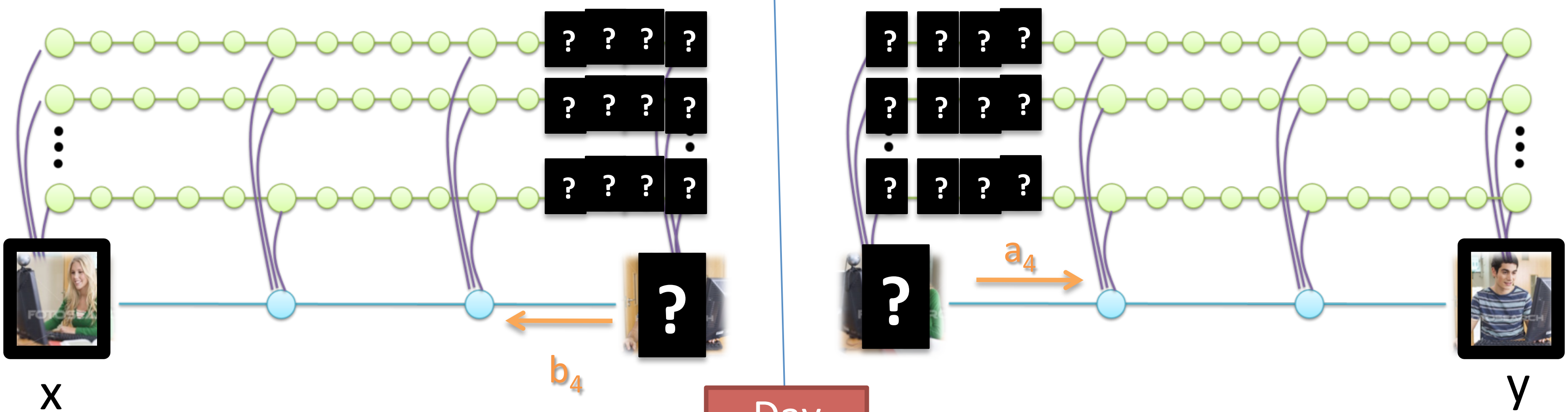
b_2= bit sent by A from Bob's machine



Alice
$x \in \{0, 1\}^b$

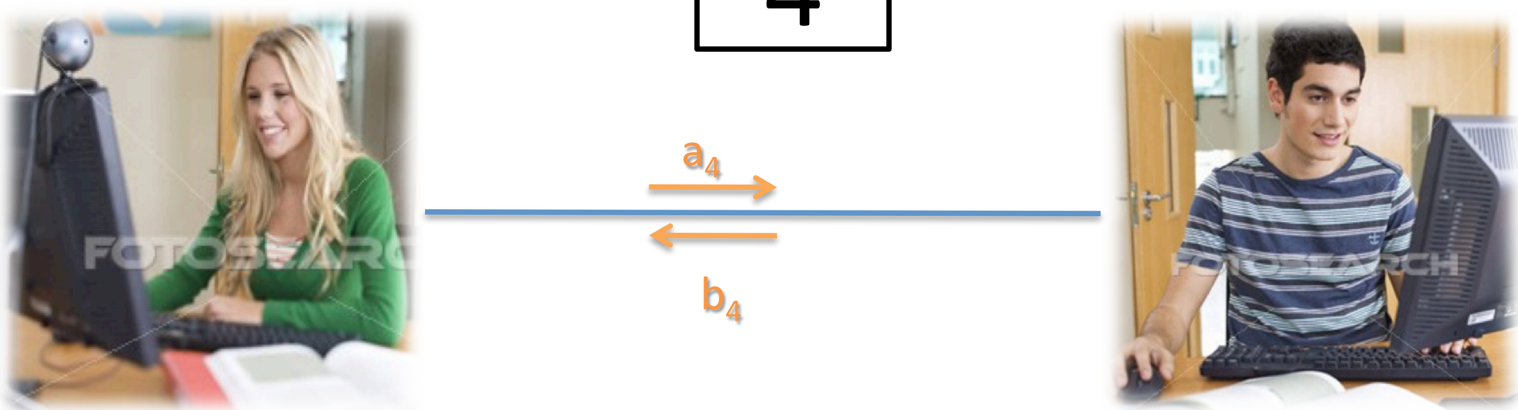
Bob
$y \in \{0, 1\}^b$

b_3 = bit sent by A from Bob's machine



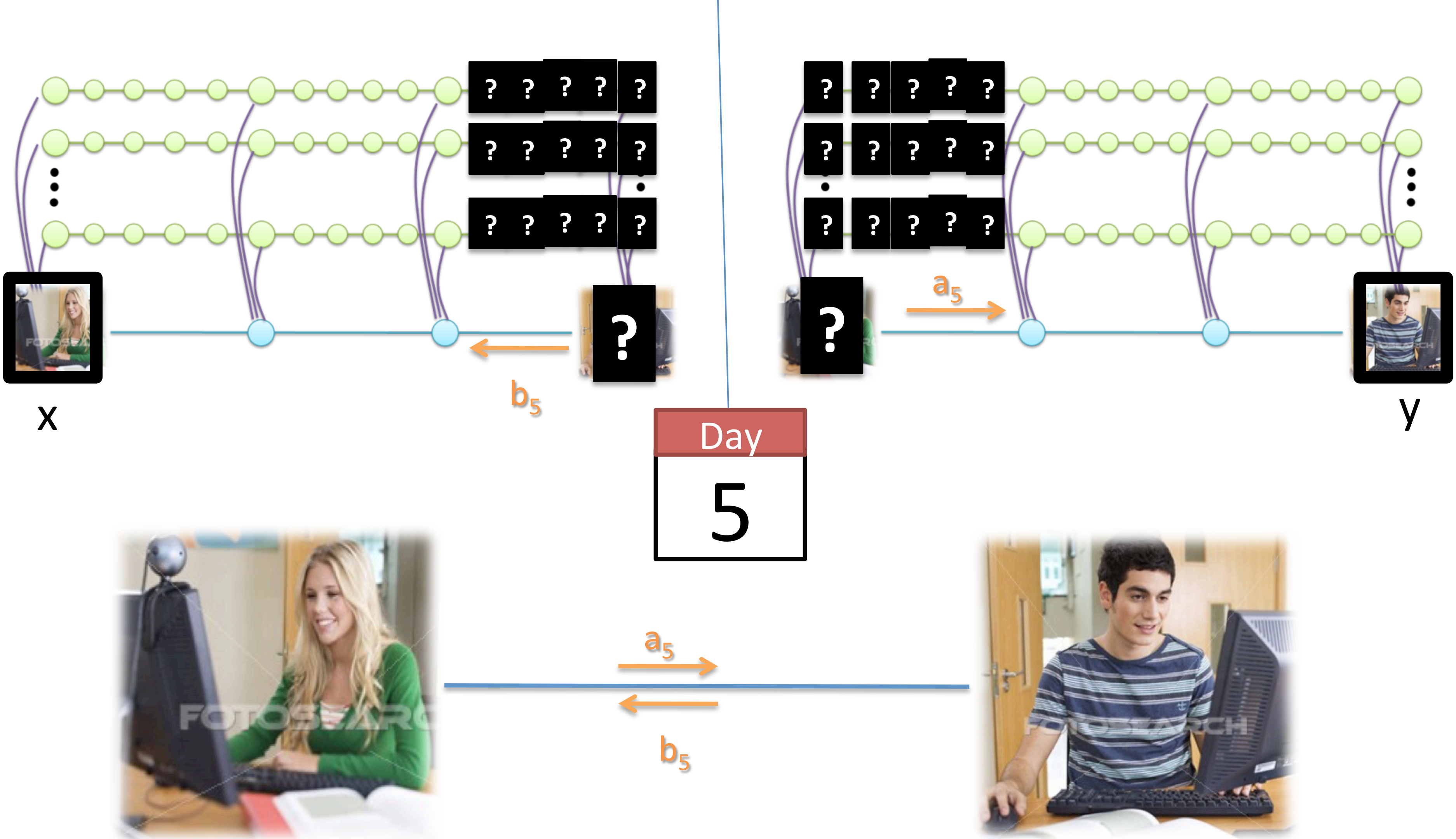
Day

4



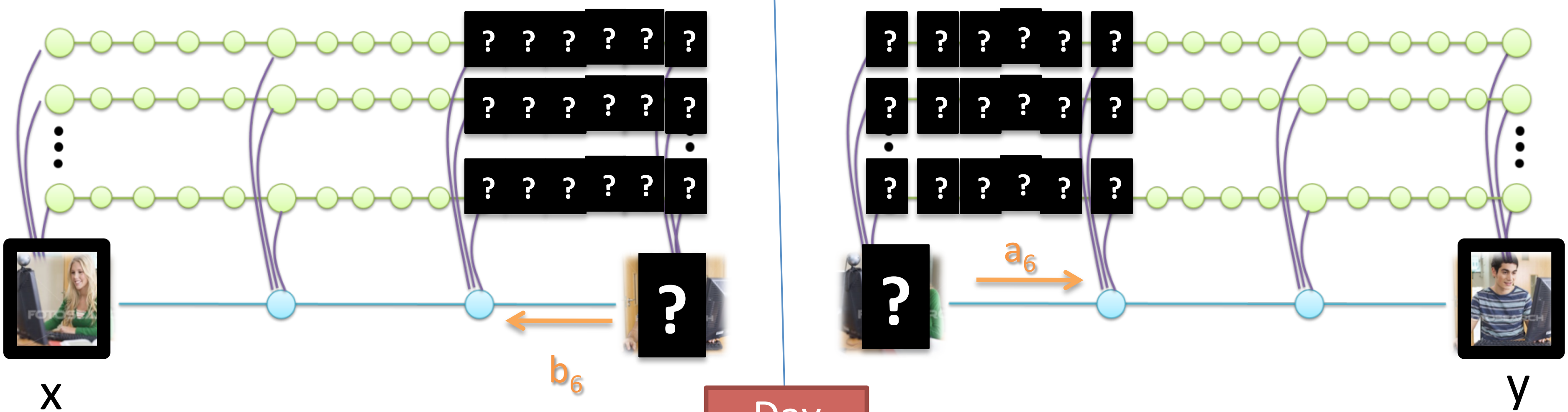
Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

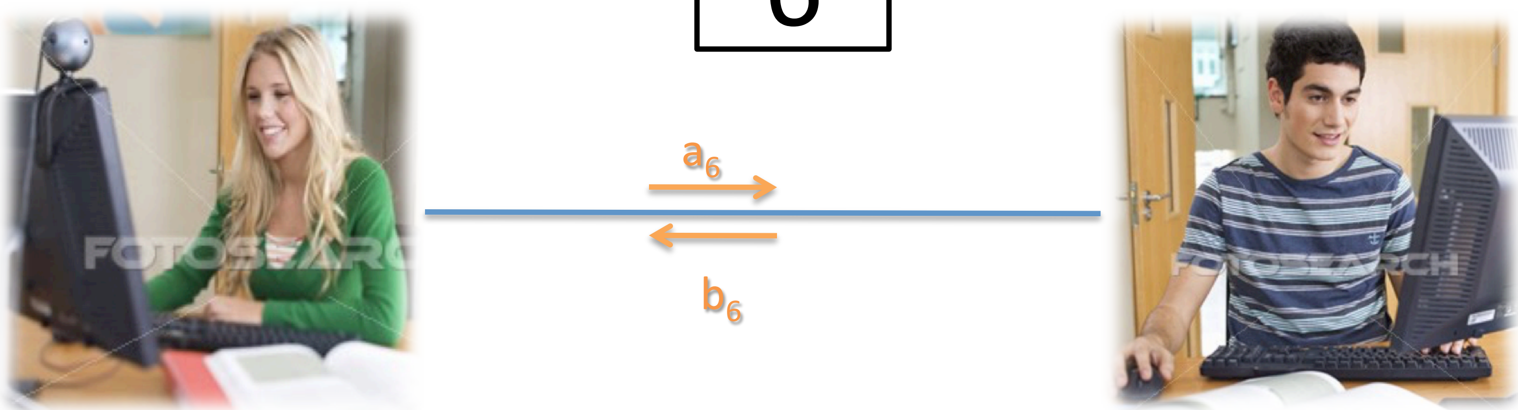


Alice $x \in \{0, 1\}^b$

Bob $y \in \{0, 1\}^b$

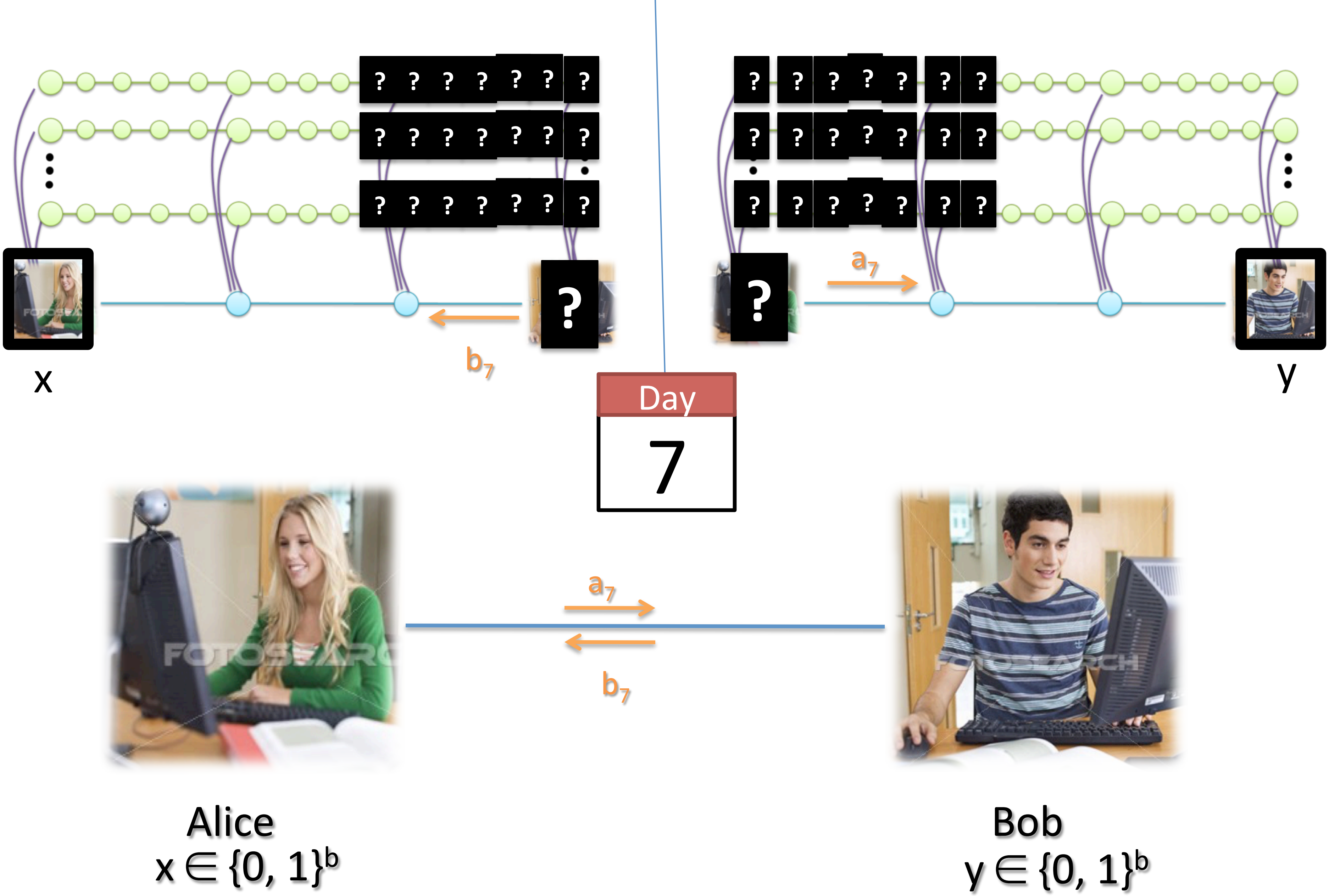


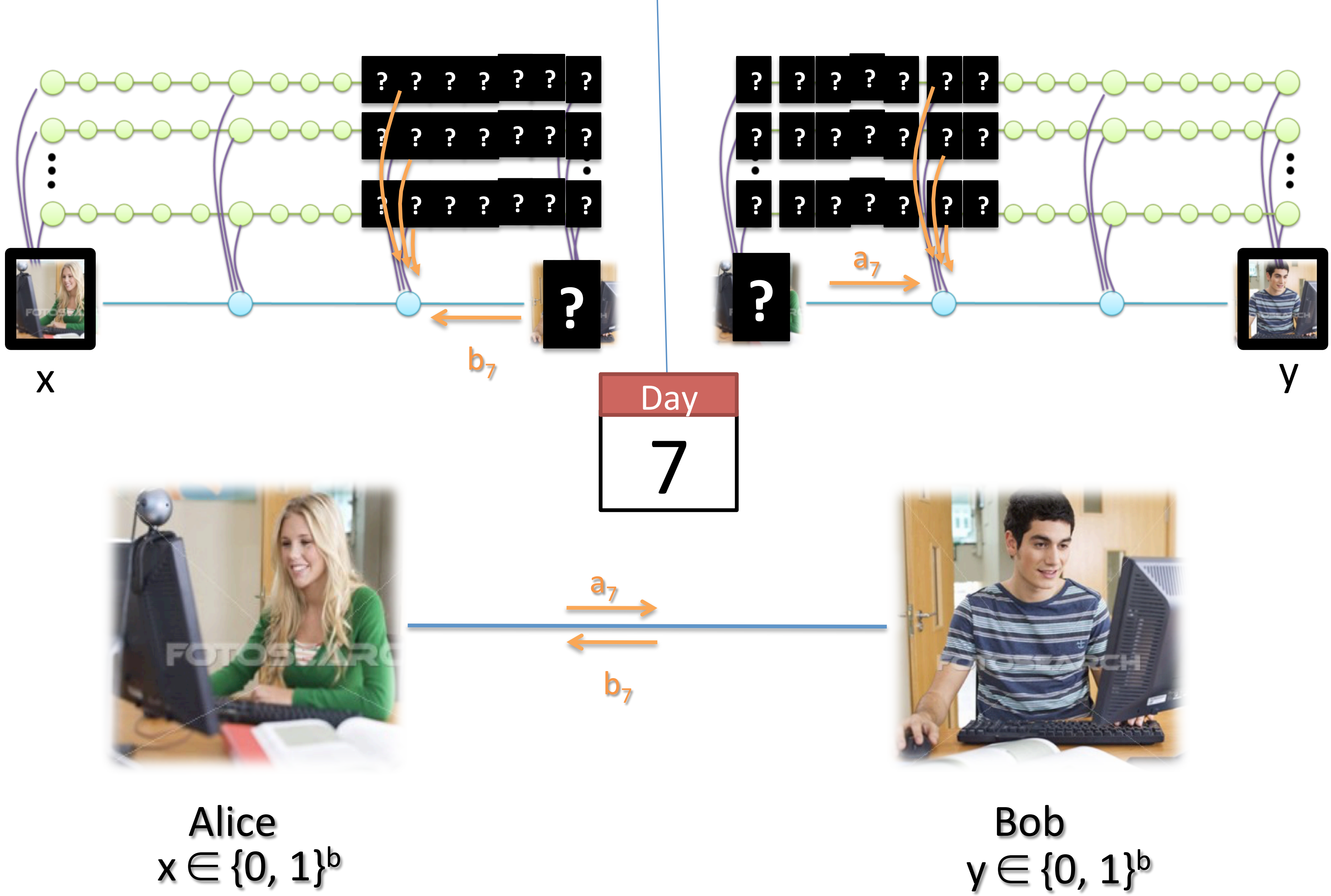
Day
6



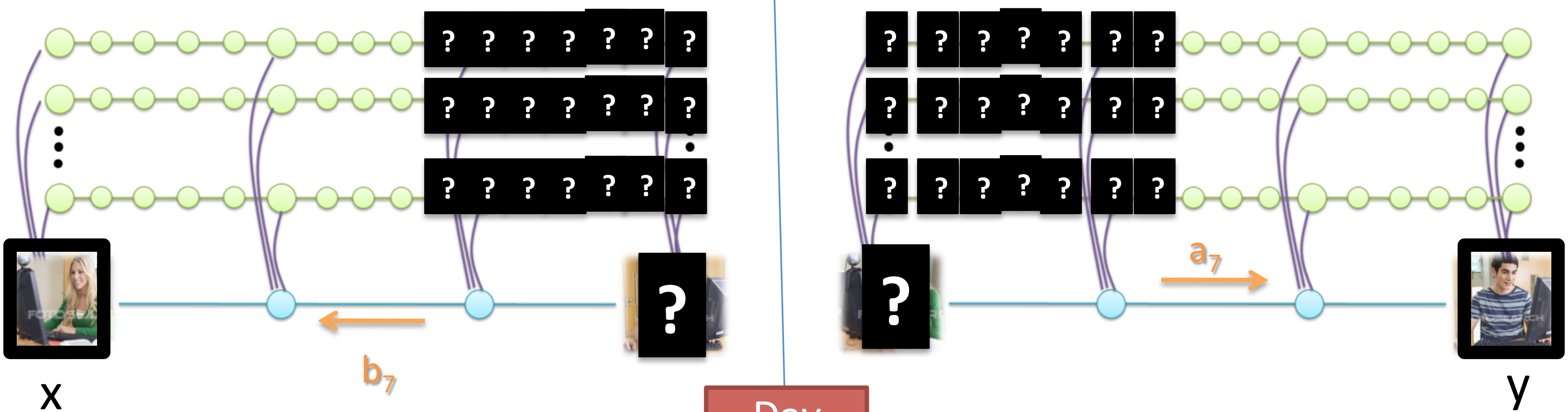
Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

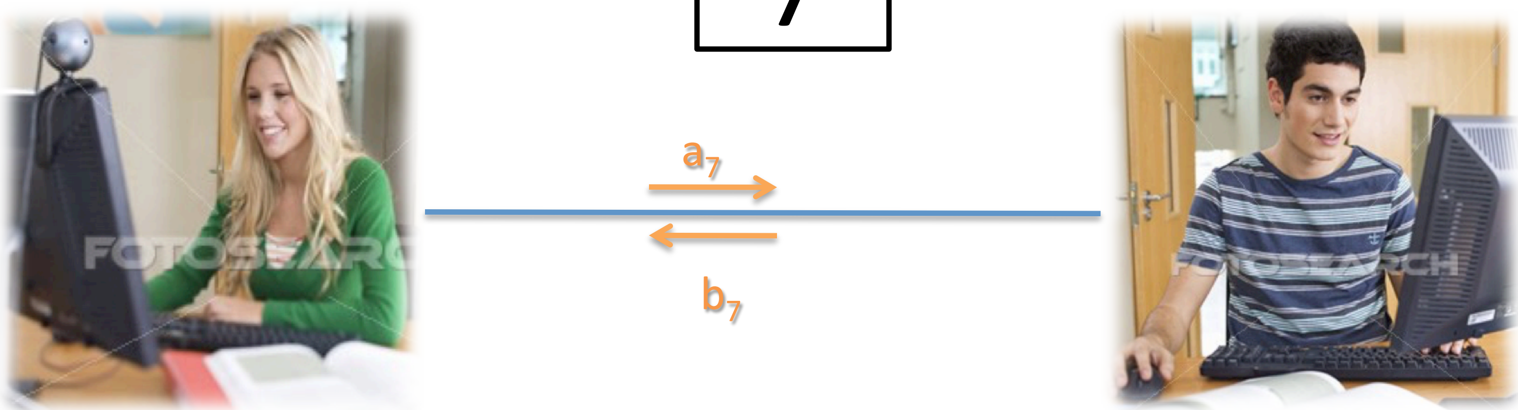

x
y
b_7
a_7
Day
7
a_7
b_7
Alice
$x \in \{0, 1\}^b$
Bob
$y \in \{0, 1\}^b$



x
y
b7
a7
Day
7
a7
b7
Alice
x ∈ {0, 1}b
Bob
y ∈ {0, 1}b

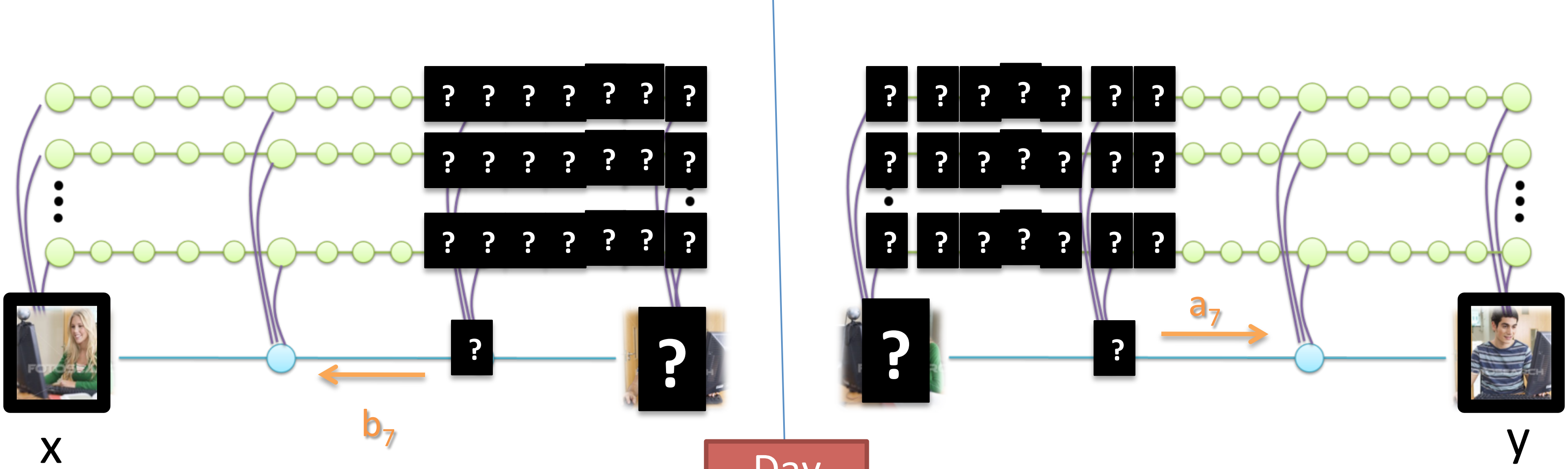


Day
7



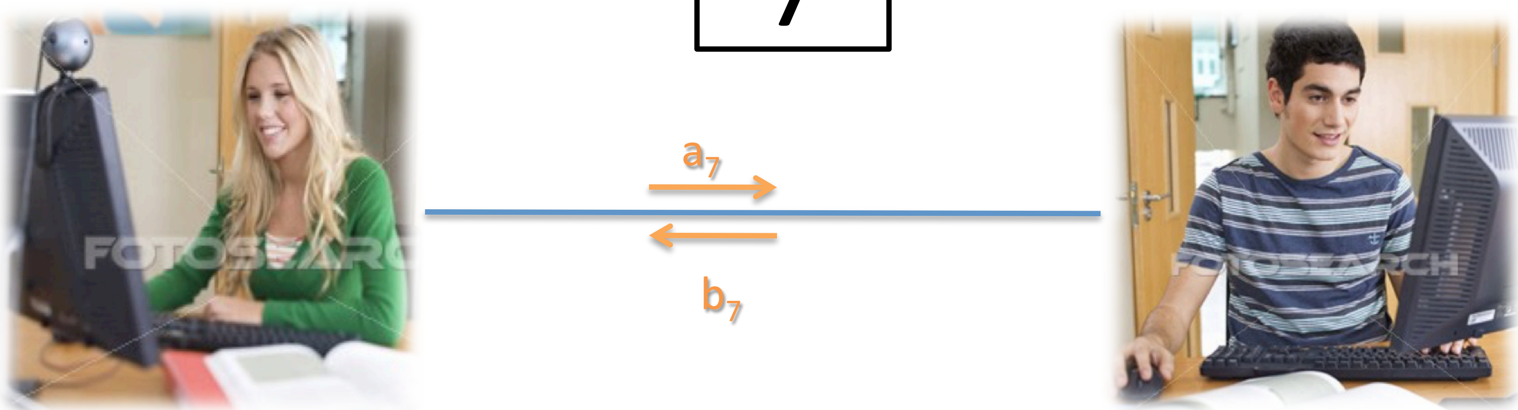
Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$



Day

7

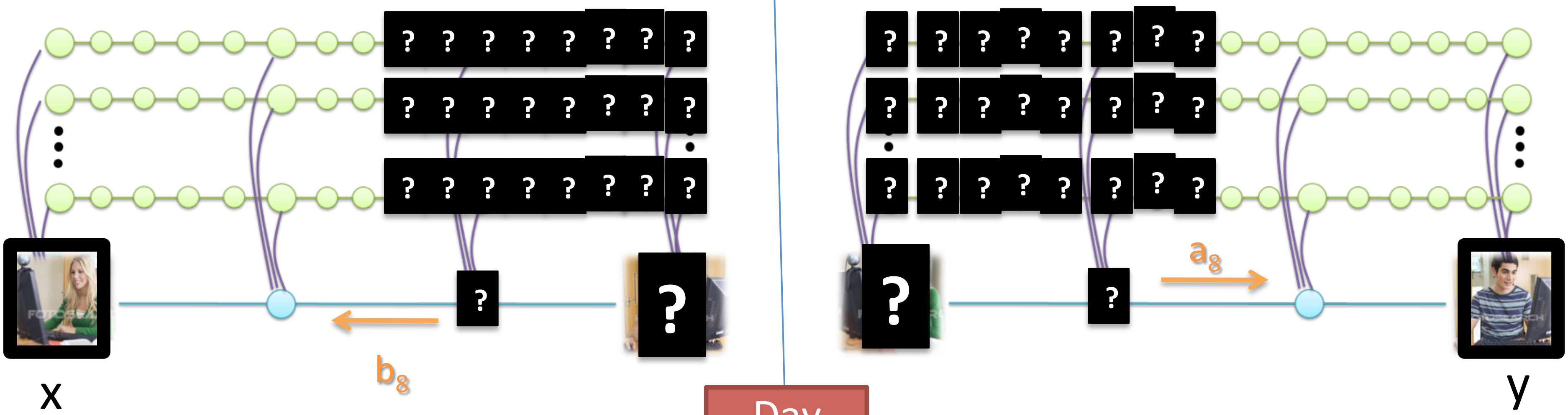


Alice

$x \in \{0, 1\}^b$

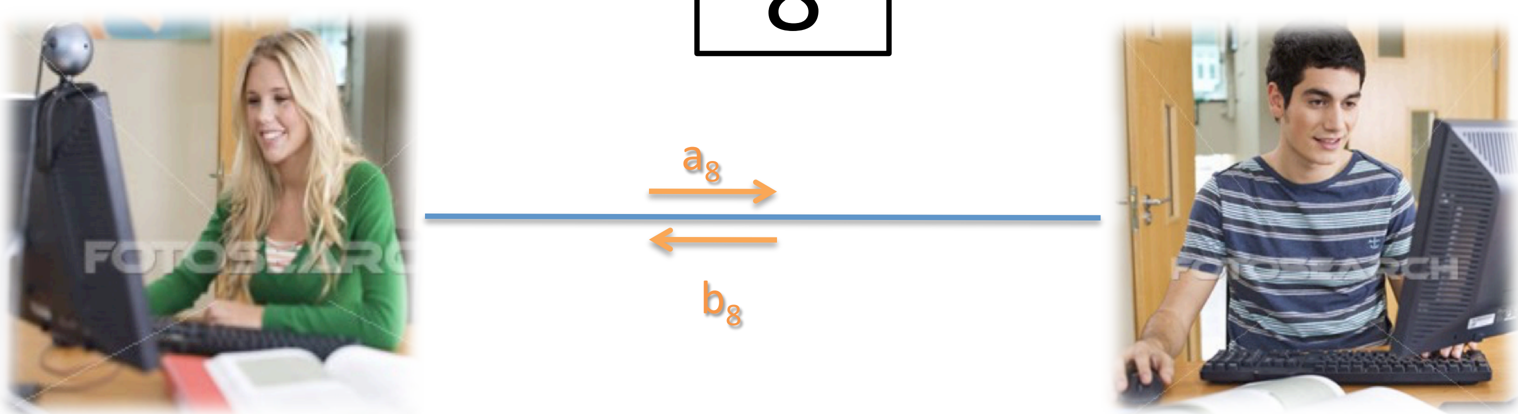
Bob

$y \in \{0, 1\}^b$



Day

8



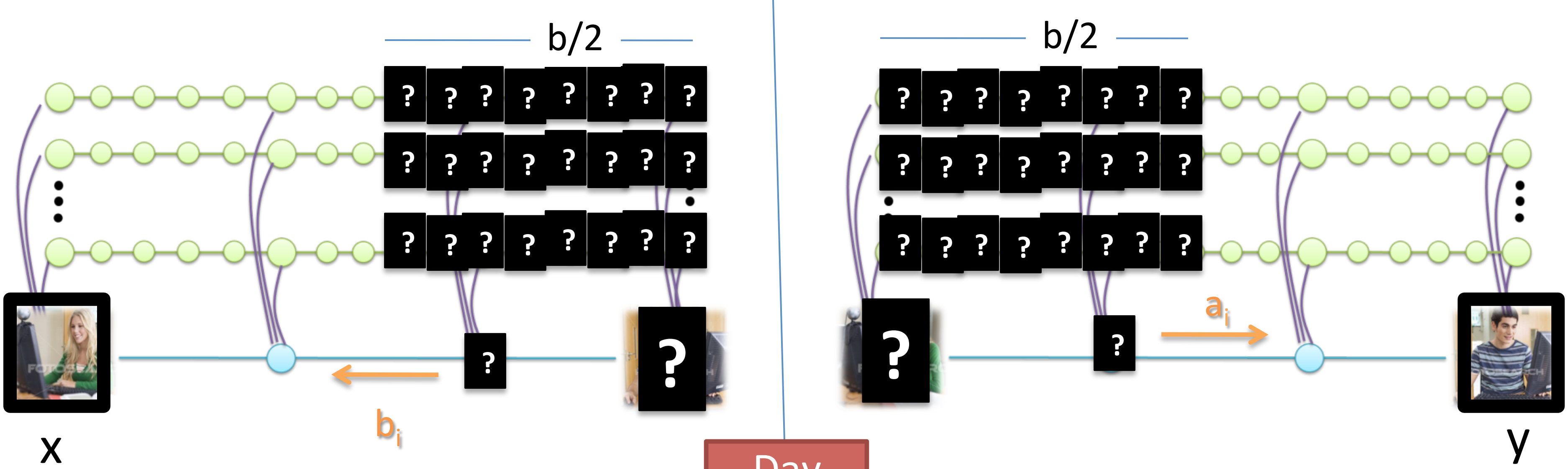
Alice

$x \in \{0, 1\}^b$

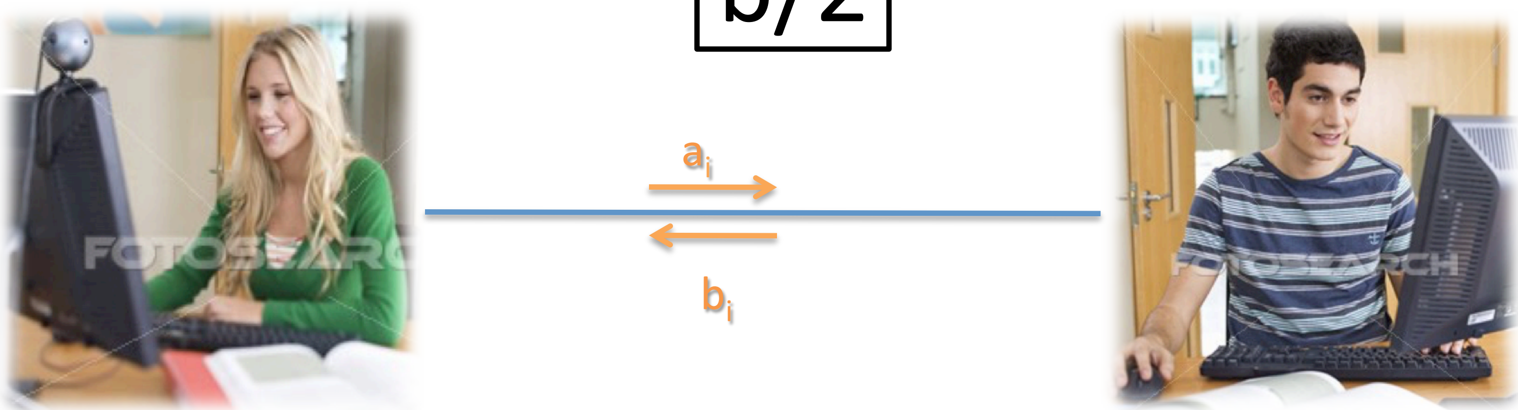
Bob

$y \in \{0, 1\}^b$

Alice and Bob can simulate the algorithm for at least **b/2** days



Day
b/2



Alice
$x \in \{0, 1\}^b$

Bob
$y \in \{0, 1\}^b$

Simulation Theorem

If the distributed equality verification can be solved in **T** days, for any **T ≤ b/2**, then the direct version can be solved in **≤T** days

Proof Alice and Bob can simulate any distributed algorithm for **b/2** days with one bit exchanged per day.

Direct Equality Verification lower bound $\Omega(b)$

Part 3.3

Distributed Equality Verification lower bound $\Omega(n^{1/2})$

Well-known result in communication complexity

By the Simulation theorem

Part 3.2

$\Omega(b)$

ST verification lower bound $\Omega(n^{1/2})$

Part 3.1

Approx MST lower bound $\Omega(n^{1/2})$

Notes

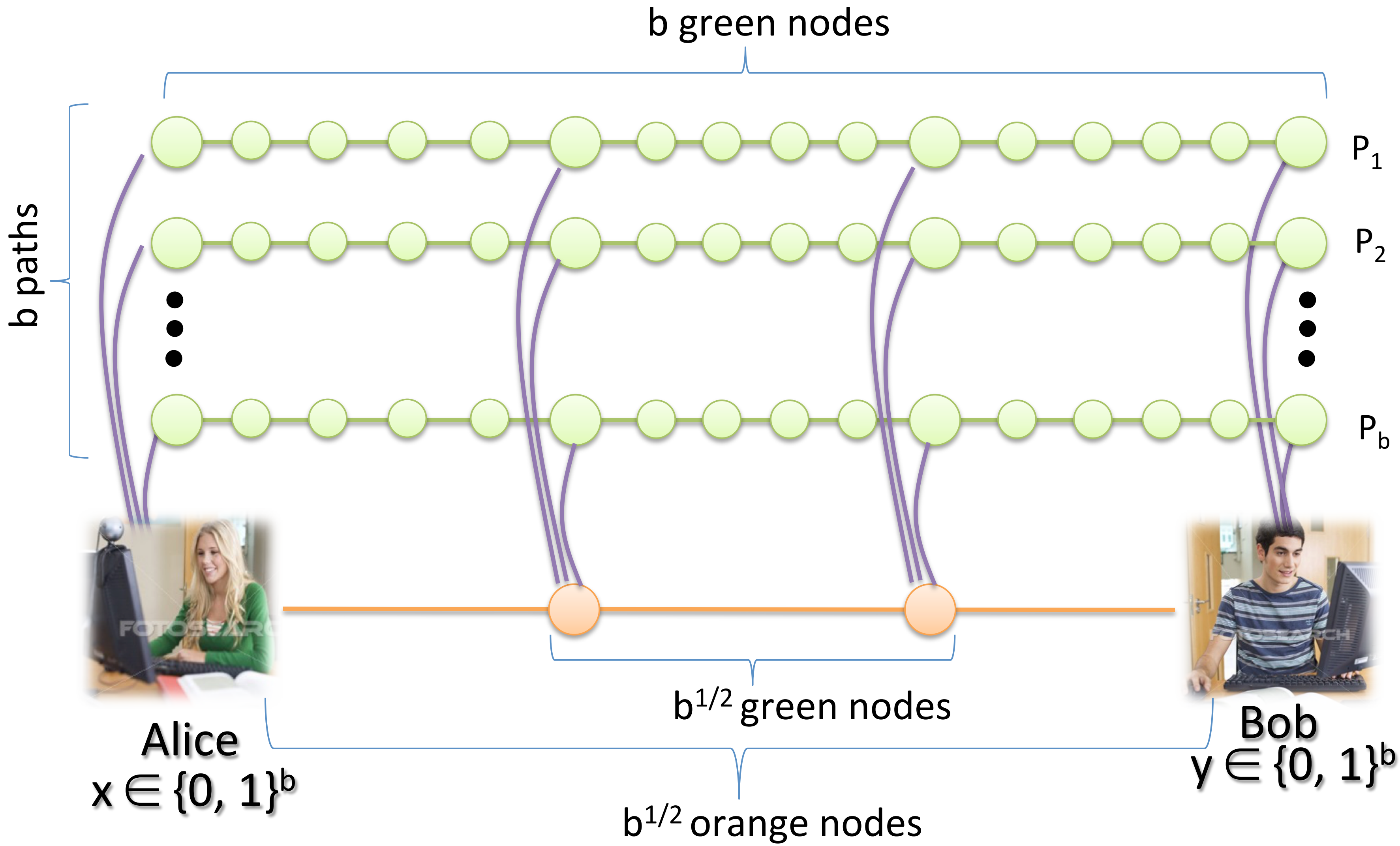
-The lower bounds hold on graphs of diameter **D=O(log n)**

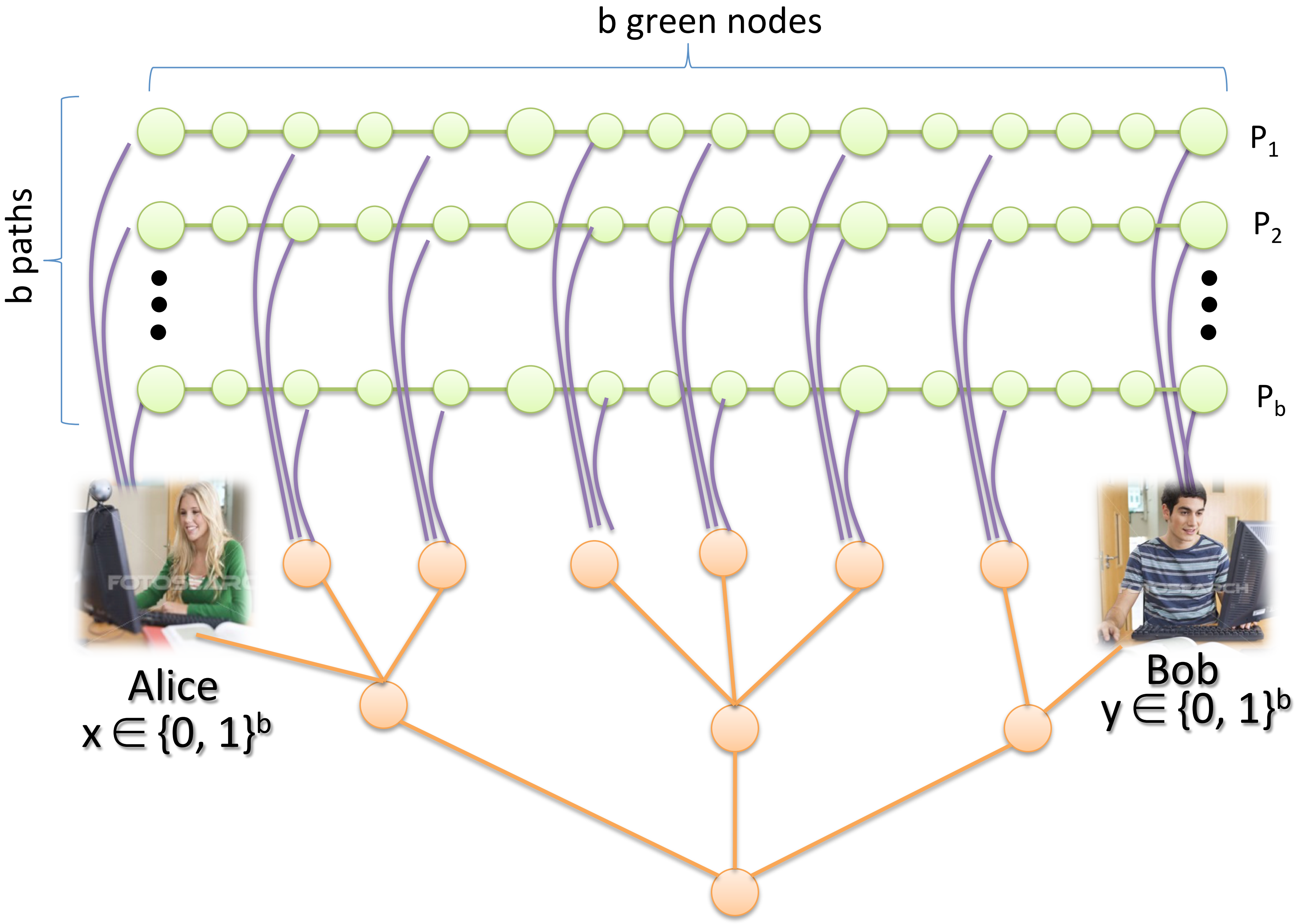
- For simplicity, we will consider only $\mathbf{D=O(n^{1/4})}$

Graph G(b) has diameter $\mathbf{n^{1/4}}$

We can use a similar analysis on some graphs of diameter

O(log n)


b green nodes
b paths
P_1
P_2
P_b
Alice
$x \in \{0, 1\}^b$
Bob
$y \in \{0, 1\}^b$
$b^{1/2}$ green nodes
$b^{1/2}$ orange nodes



b green nodes
b paths
P_1
P_2
P_b
Alice
$x \in \{0, 1\}^b$
Bob
$y \in \{0, 1\}^b$

We are done

with deterministic algorithms

How about randomized algorithms?

… to be continued