This file accompanies a presentation given at the 3rd workshop on Advanced Distributed Graph Algorithms in Paris on 09/26/2016. The material is not complete and deviates from the content of the original papers for sake of simpler presentation of key ideas and concepts to this particular audience.

Recent Algorithms and Lower Bounds for Global Distributed Graph Problems

Stephan Holzer

Thanks to collaborators:
Atish Das Sarma, Benjamin Dissler, Silvio Frischknecht, Liah Kor
Amos Korman, Danupon Nanongkai, Gopal Pandurangan
David Peleg, Nathan Pinsker, Liam Roditty, Roger Wattenhofer
[STOC’11,SODA’12,PODC’12,SICOMP’12,DISC’14,OPODIS’15,Arxiv’16]

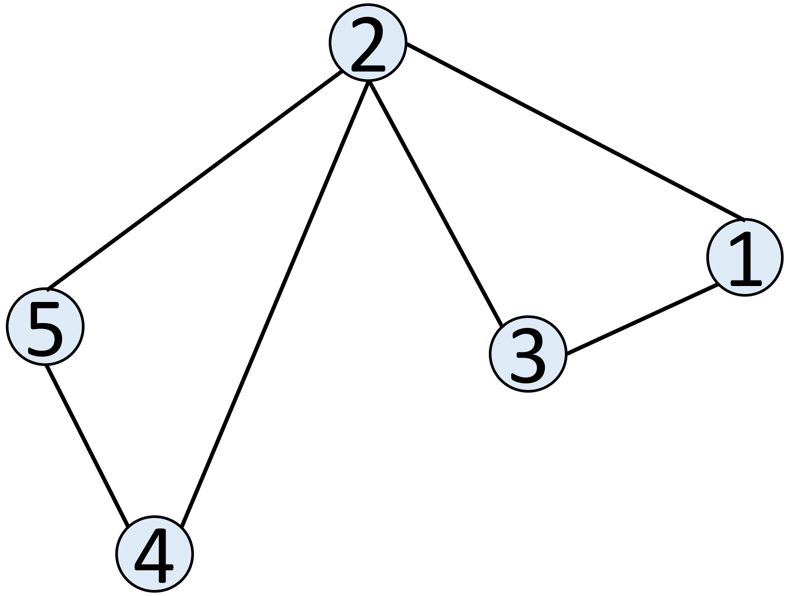
Recent Algorithms and Lower Bounds for Global Distributed Graph Problems

Stephan Holzer

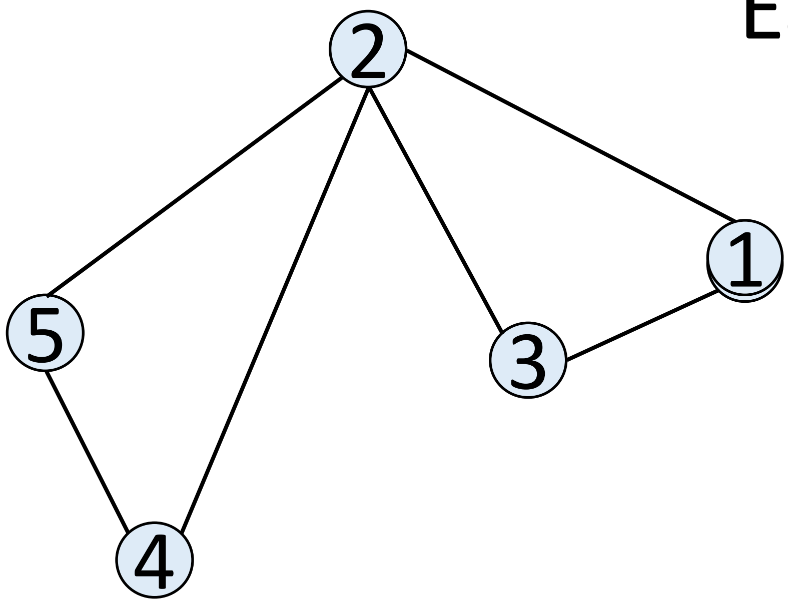
Thanks to collaborators:
Atish Das Sarma, Benjamin Dissler, Silvio Frischknecht, Liah Kor
Amos Korman, Danupon Nanongkai, Gopal Pandurangan
David Peleg, Nathan Pinsker, Liam Roditty, Roger Wattenhofer
[STOC'11,SODA'12,PODC'12,SICOMP'12,DISC'14,OPODIS'15,Arxiv'16]

Message Passing Model
Graph G of n nodes

Message Passing Model
Graph G of n nodes



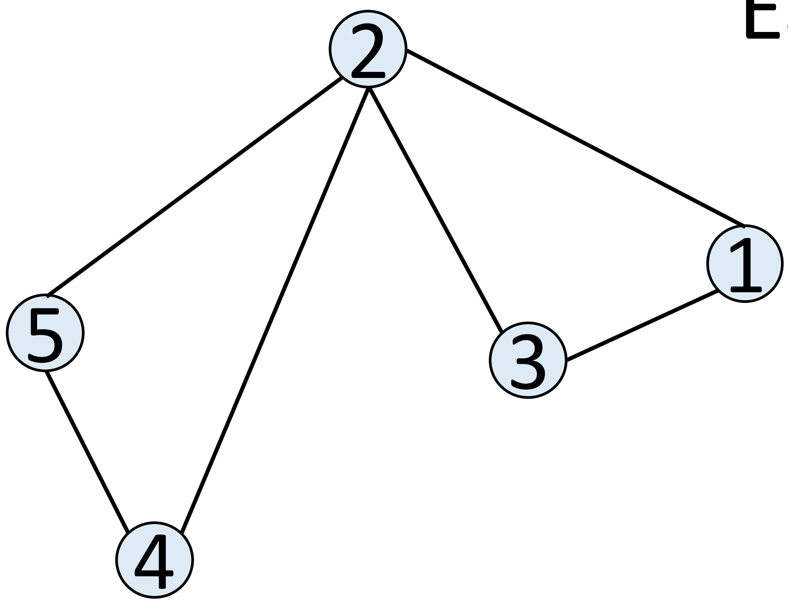
Message Passing Model
Graph G of n nodes



Each node has data

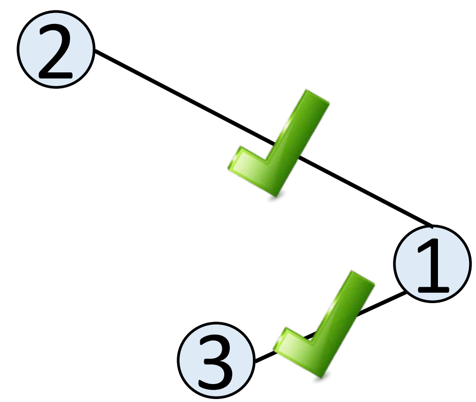
Local infor-
mation only

Message Passing Model
Graph G of n nodes



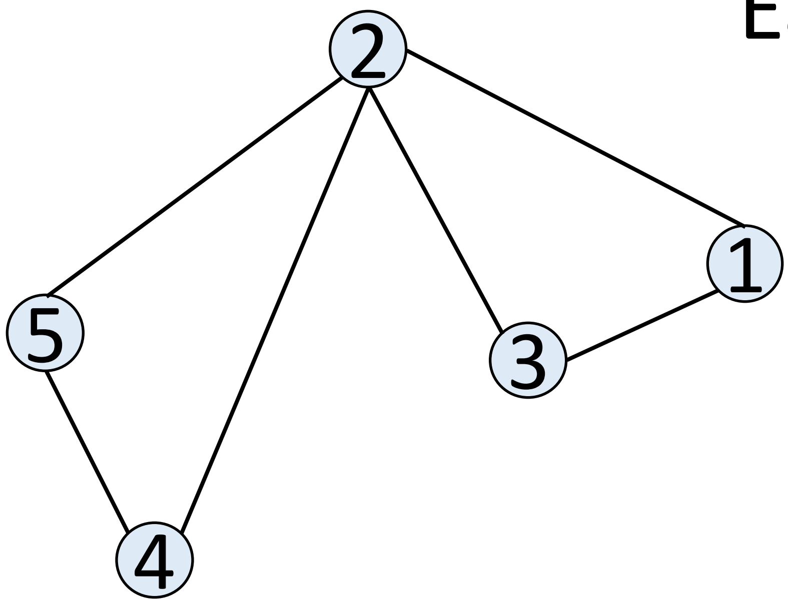
Each node has data

Local infor-
mation only

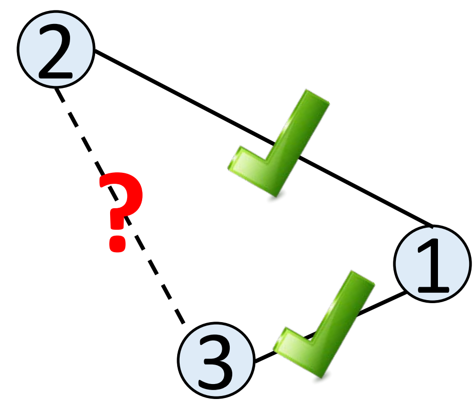


Message Passing Model
Graph **G** of **n** nodes

Each node has data



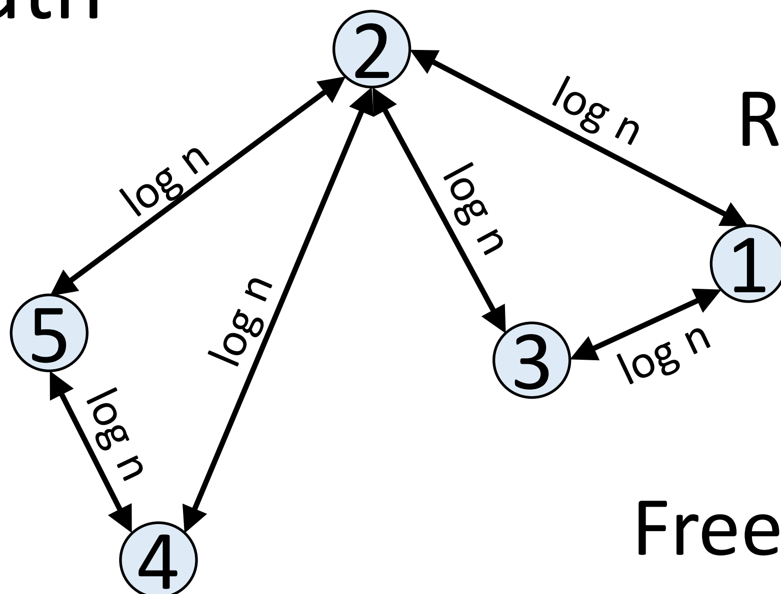
Local infor-
mation only



Message Passing Model

Graph G of n nodes

Limited bandwidth

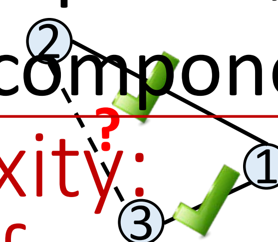


Synchronized rounds

Reliable communication

No faults/crashes

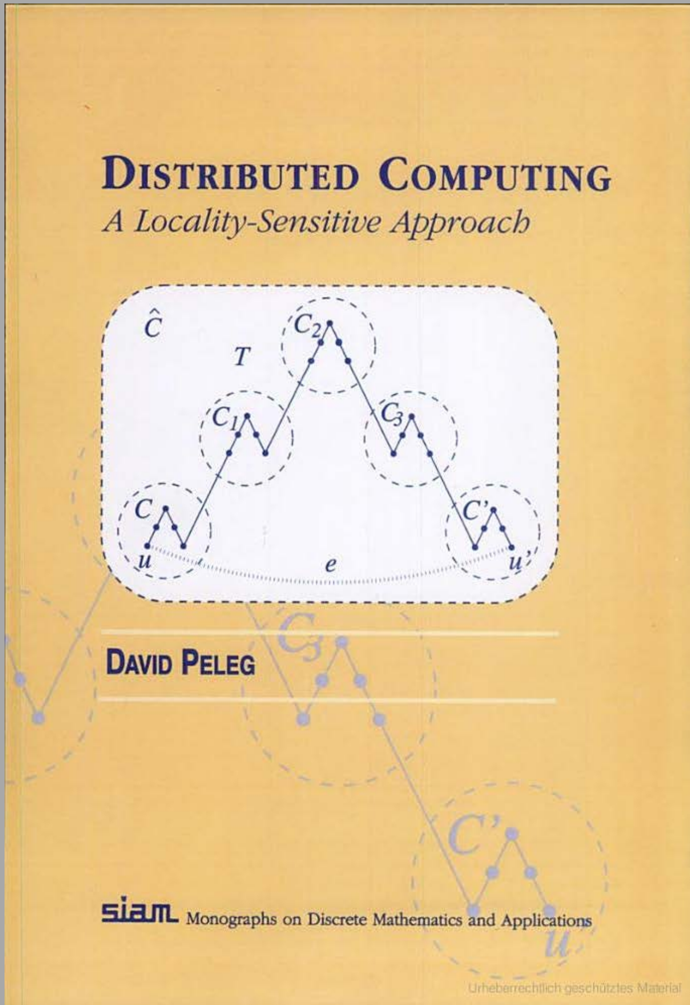
Local infor-
mation only

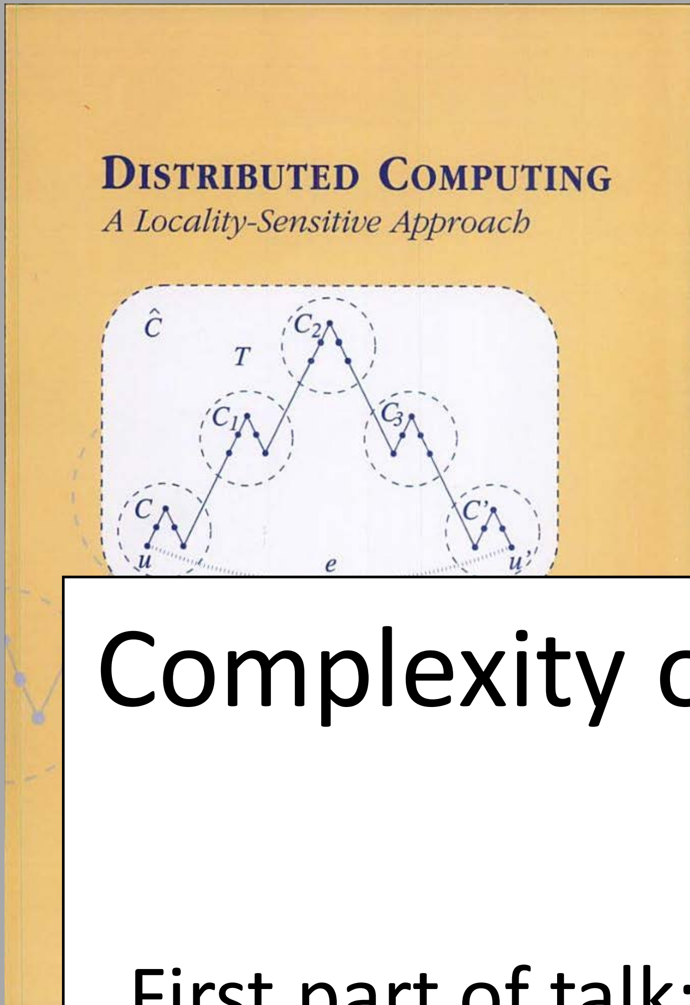


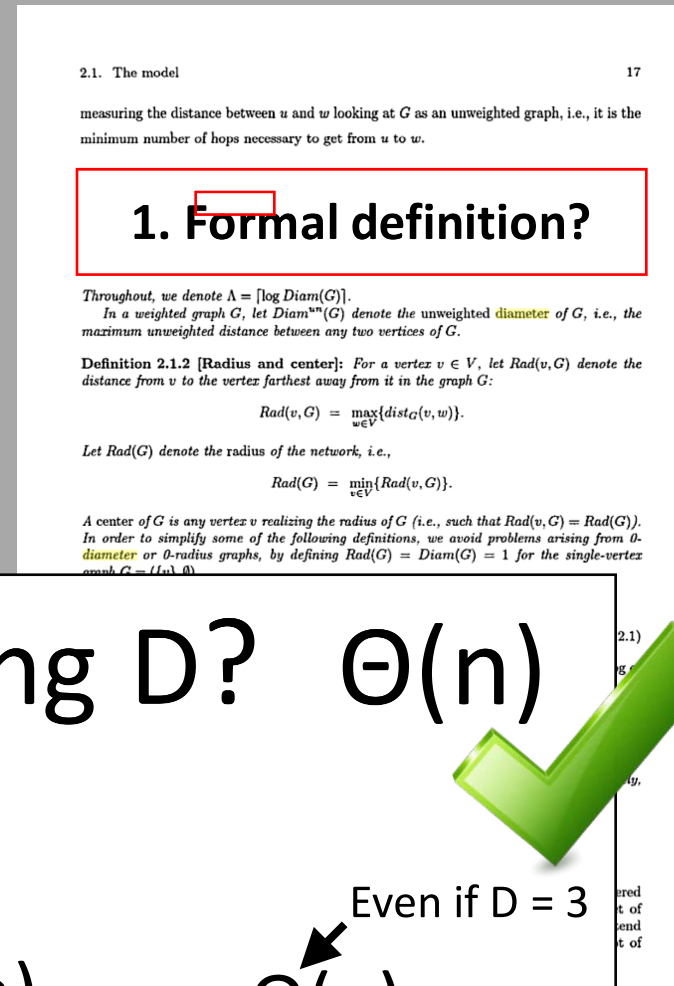
Free internal computations

Graph is one connected component

Time complexity:
number of
communication rounds


DISTRIBUTED COMPUTING
A Locality-Sensitive Approach
DAVID PELEG
siam Monographs on Discrete Mathematics and Applications
Urheberrechtlich geschütztes Material




2.1. The model 17

measuring the distance between u and w looking at G as an unweighted graph, i.e., it is the minimum number of hops necessary to get from u to w.

Throughout, we denote $\Lambda = \lceil \log Diam(G) \rceil$.

In a weighted graph G, *let* $Diam^{un}(G)$ *denote the unweighted diameter of* G, *i.e., the maximum unweighted distance between any two vertices of* G.

Definition 2.1.2 [Radius and center]: *For a vertex* $v \in V$, *let* $Rad(v, G)$ *denote the distance from* v *to the vertex farthest away from it in the graph* G:

$$Rad(v,G) = \max_{w \in V}\{dist_G(v,w)\}.$$

Let $Rad(G)$ *denote the radius of the network, i.e.,*

$$Rad(G) = \min_{v \in V}\{Rad(v,G)\}.$$

A center of G *is any vertex* v *realizing the radius of* G *(i.e., such that* $Rad(v,G) = Rad(G)$*). In order to simplify some of the following definitions, we avoid problems arising from 0-diameter or 0-radius graphs, by defining* $Rad(G) = Diam(G) = 1$ *for the single-vertex graph* ...

1. Formal definition?

Complexity of computing D? Θ(n)

Even if D = 3

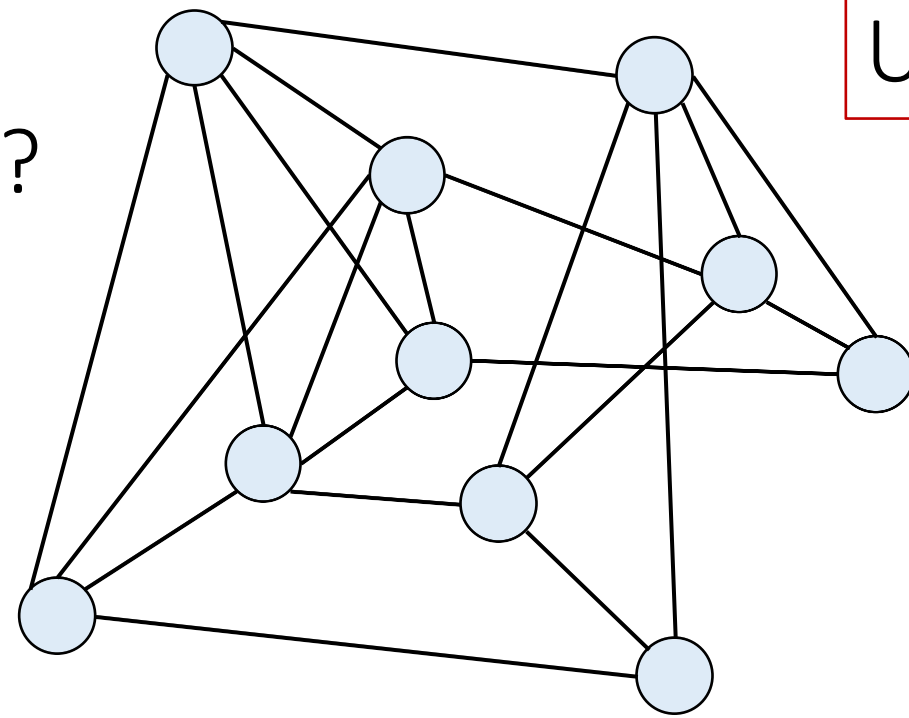
First part of talk: O(n) [PODC 2012] Ω(n) [SODA 2012]

computing

Networks cannot compute their diameter in sublinear time!

Diameter of a network

Diameter of this network?

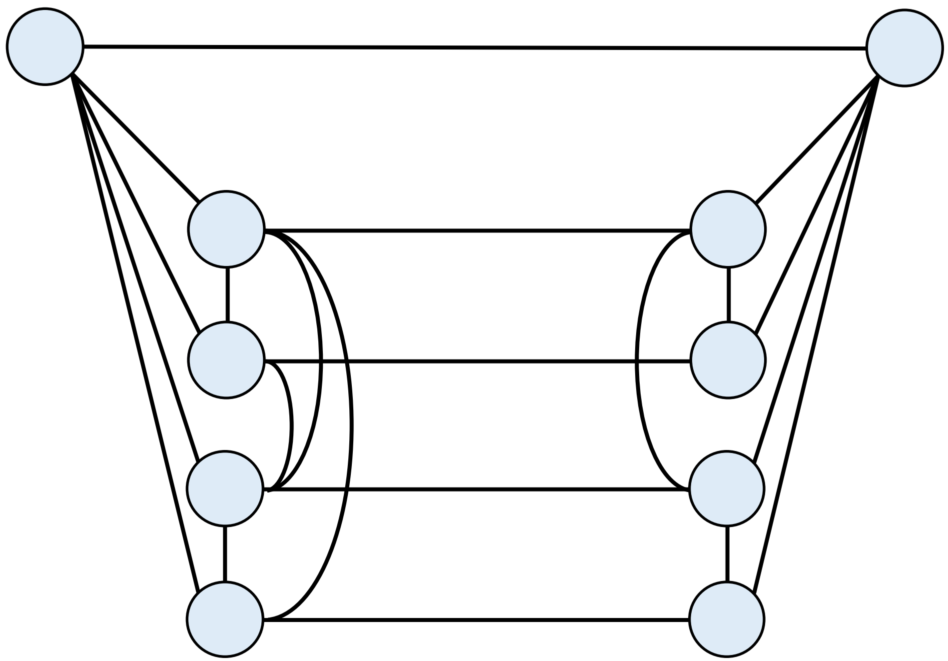


Unweighted!

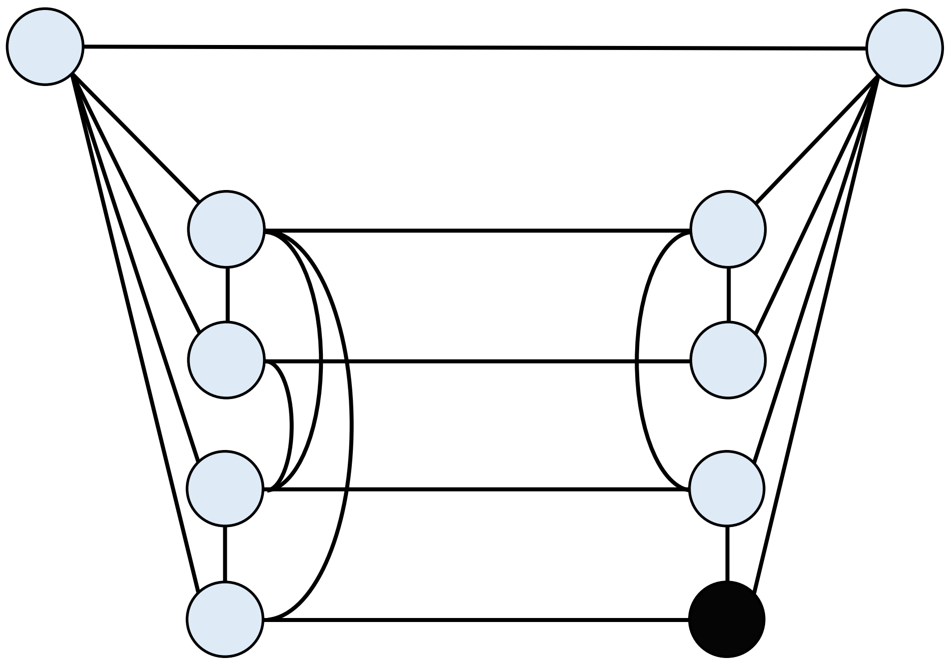
- Distance between two nodes = Number of hops of shortest path
- Diameter of network = Maximum distance, between any two nodes

Networks cannot compute their diameter in sublinear time!

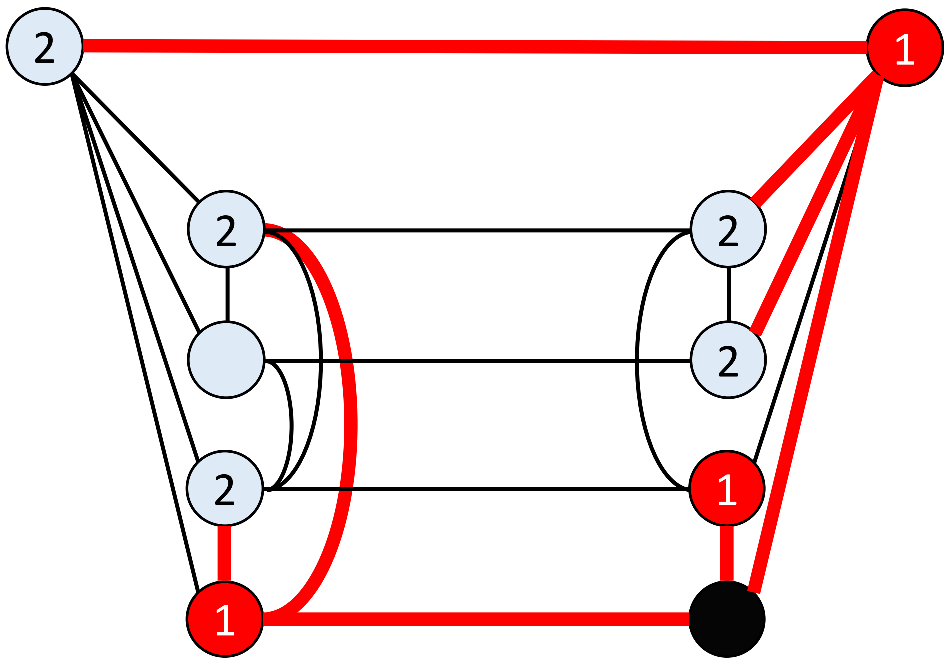
Unweighted!



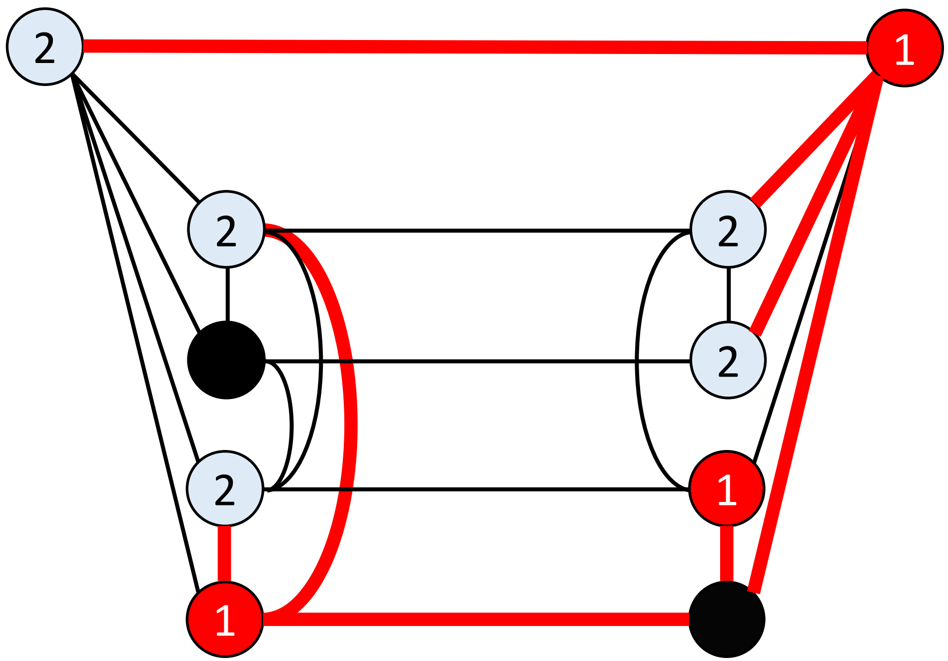
Networks cannot compute their diameter in sublinear time!



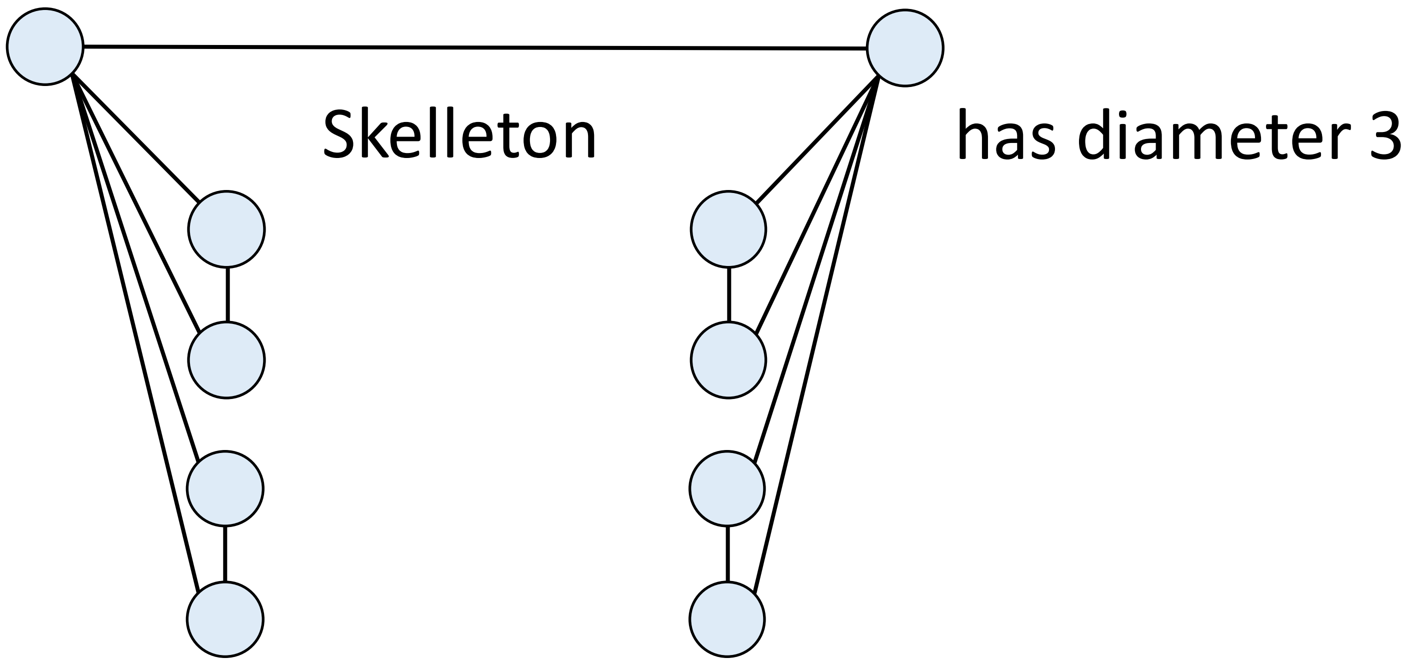
Networks cannot compute their diameter in sublinear time!



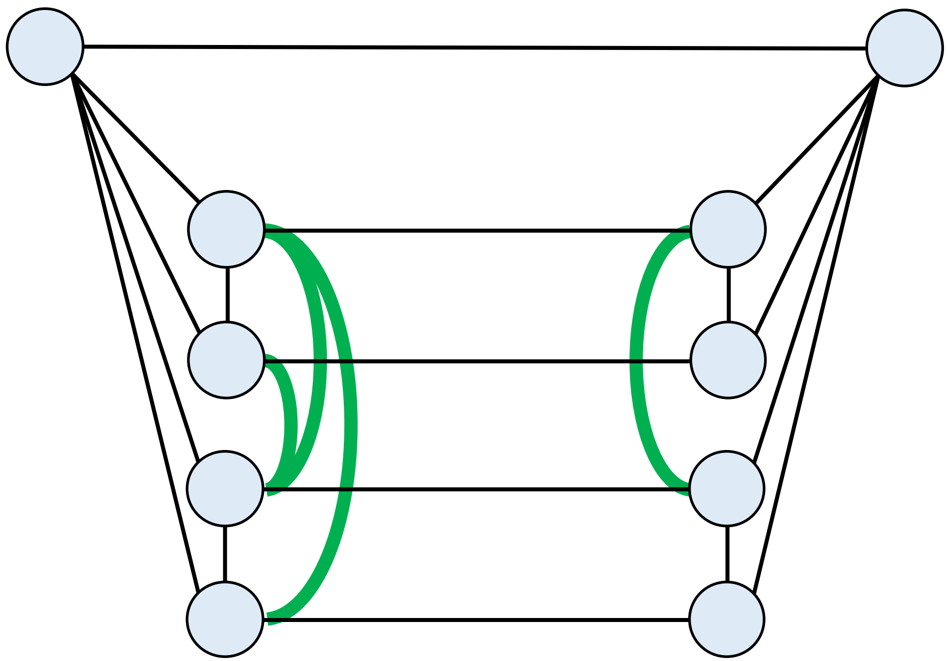
Networks cannot compute their diameter in sublinear time!



Networks cannot compute their diameter in sublinear time!

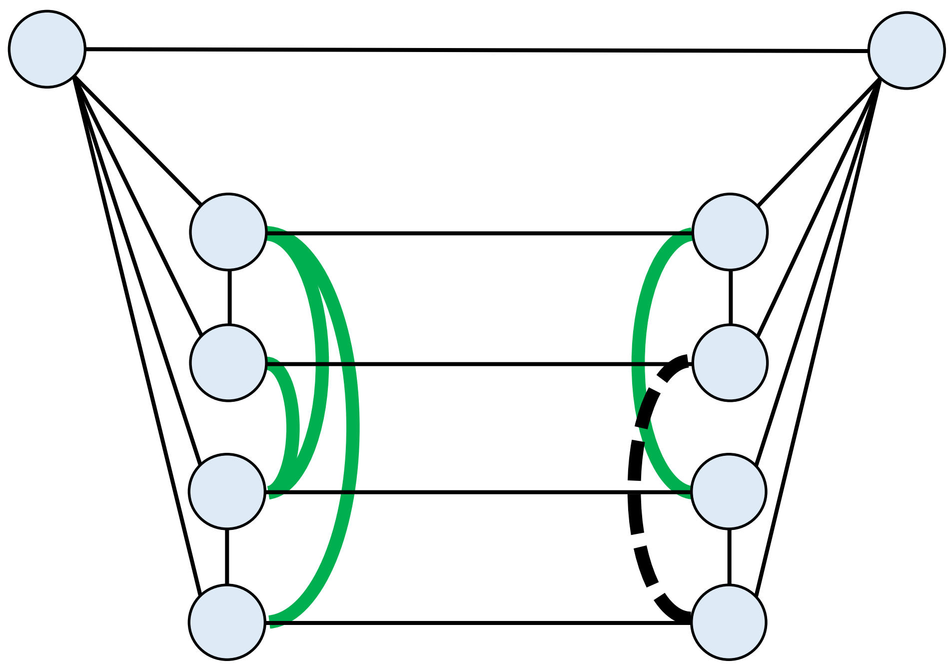


Networks cannot compute their diameter in sublinear time!



has diameter 3

Networks cannot compute their diameter in sublinear time!

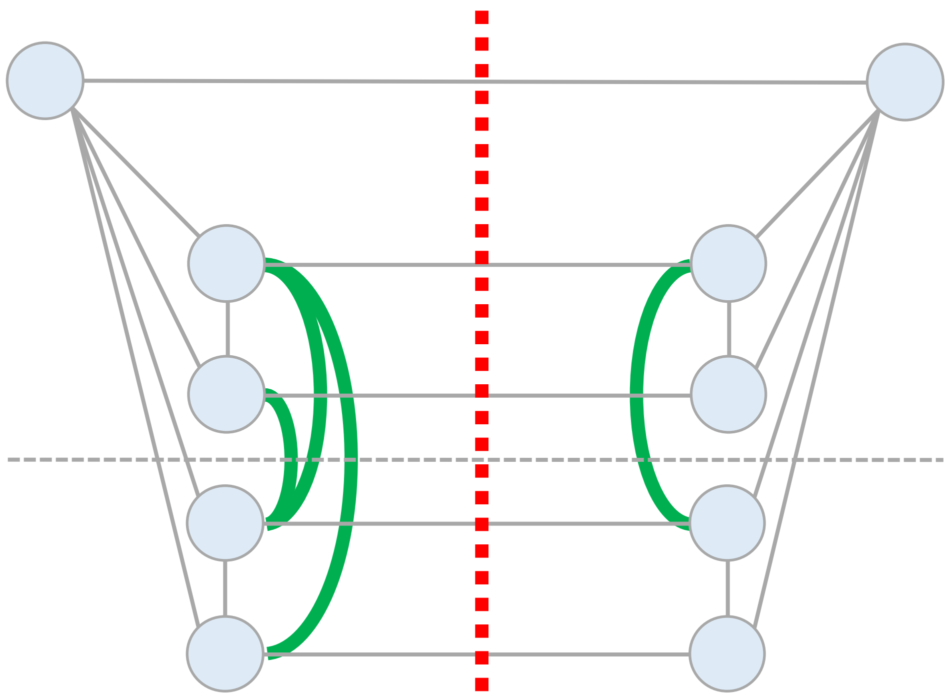


has diameter ~~3~~
2?

Networks cannot compute their diameter in sublinear time!

D = 2 or 3?

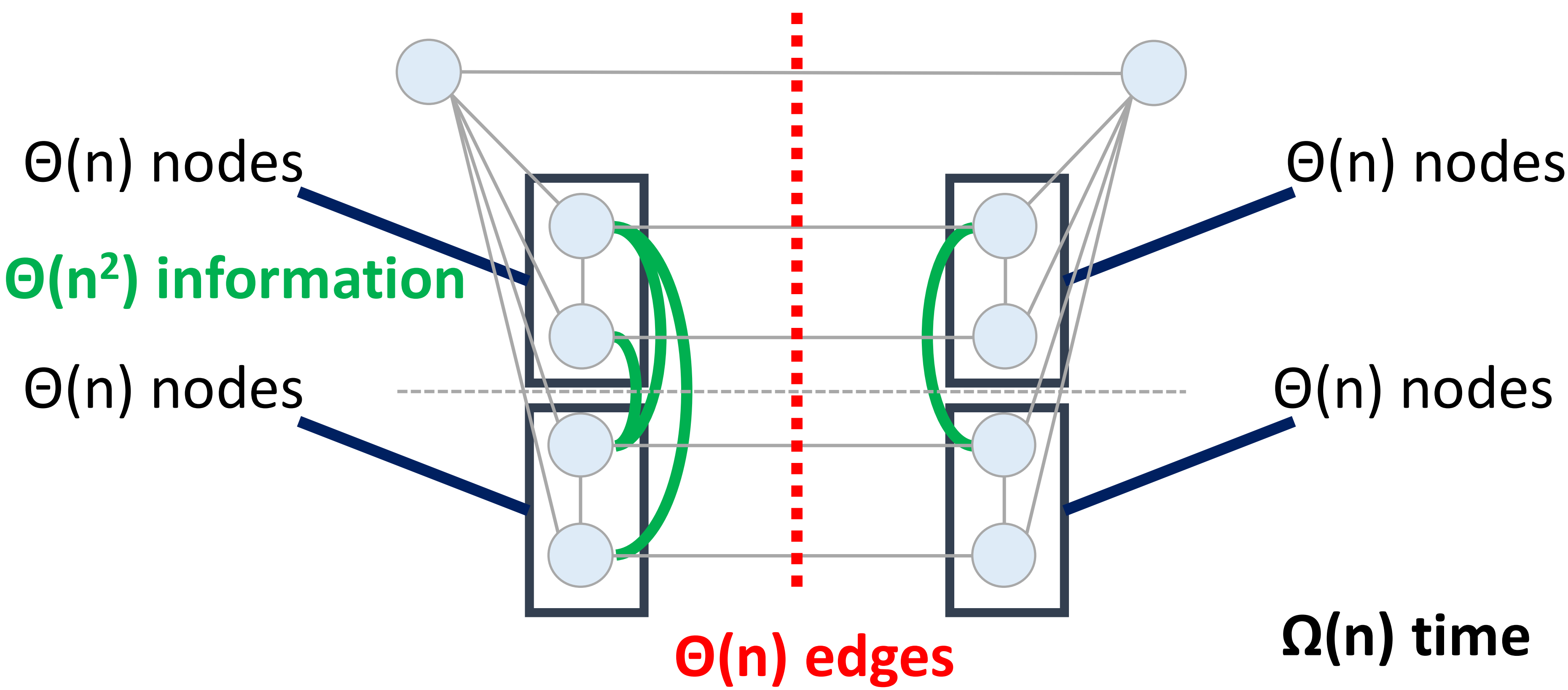
Upper and lower row not connected on any side?



Networks cannot compute their diameter in sublinear time!

D = 2 or 3?

Upper and lower row not connected on any side?



Networks cannot compute their diameter in sublinear time!

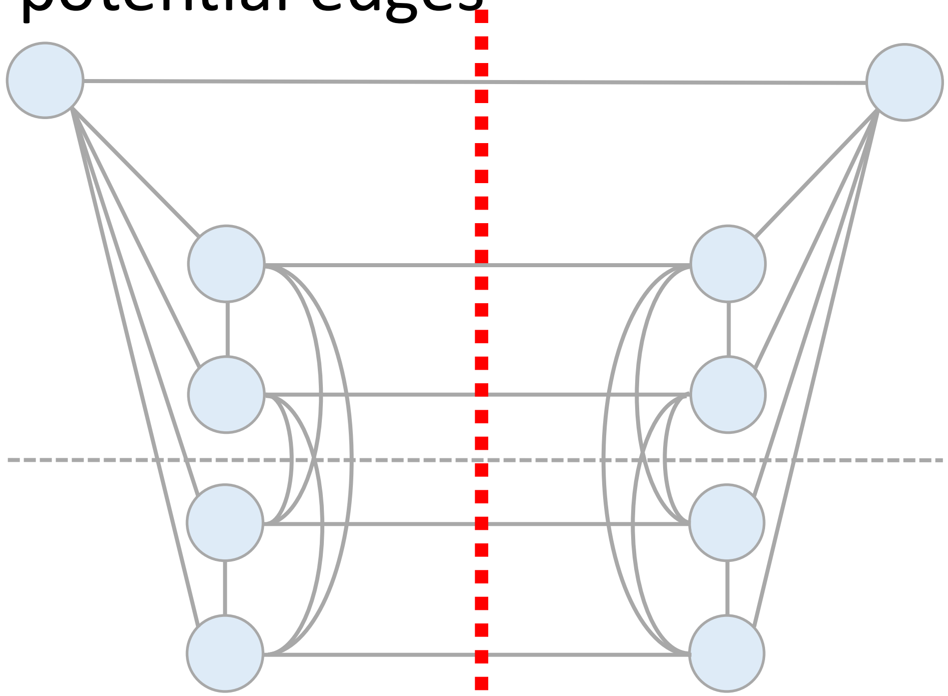
Upper and lower row not connected on any side?

Now: slightly more details

Networks cannot compute their diameter in sublinear time!

Upper and lower row not connected on any side?

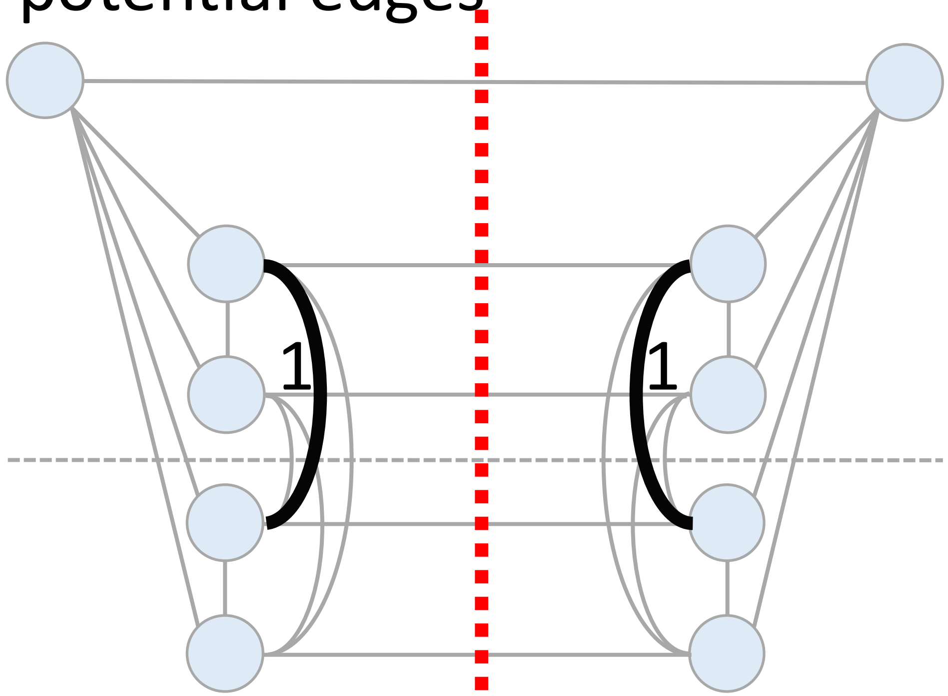
Label potential edges



Networks cannot compute their diameter in sublinear time!

Upper and lower row not connected on any side?

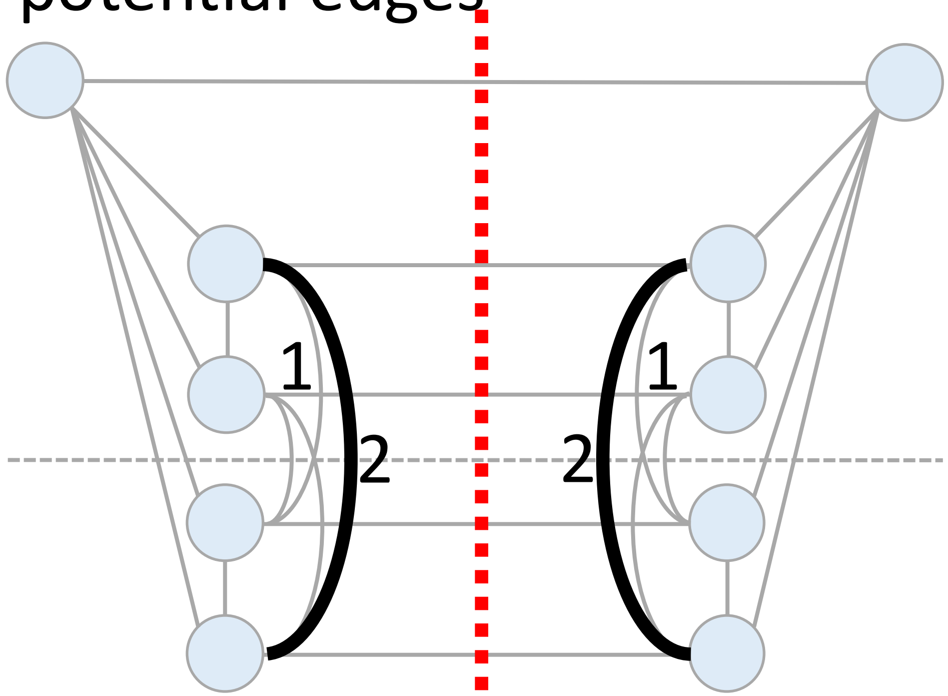
Label potential edges



Networks cannot compute their diameter in sublinear time!

Upper and lower row not connected on any side?

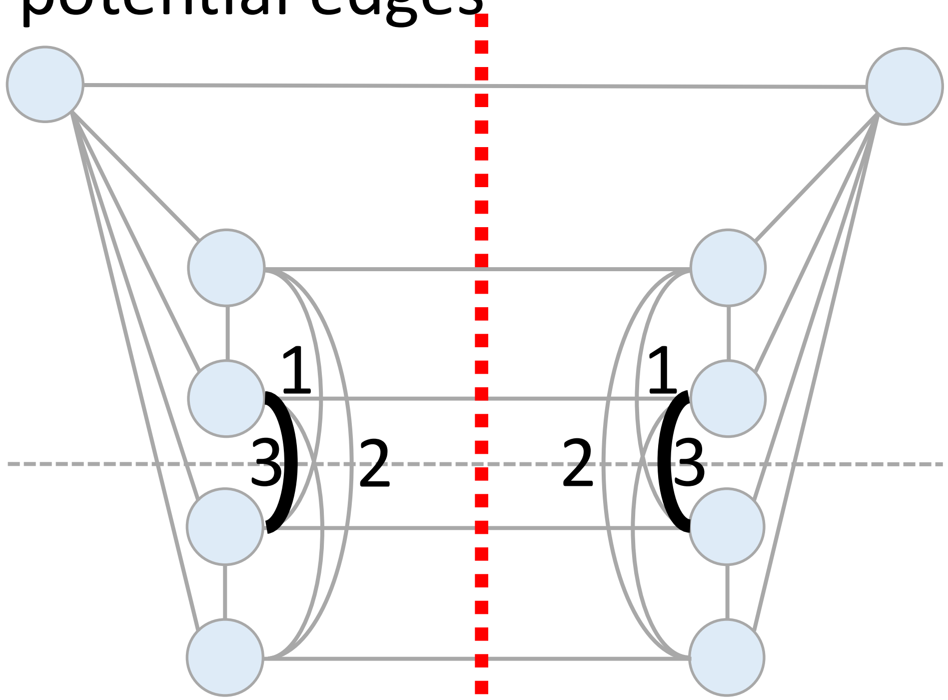
Label potential edges



Networks cannot compute their diameter in sublinear time!

Upper and lower row not connected on any side?

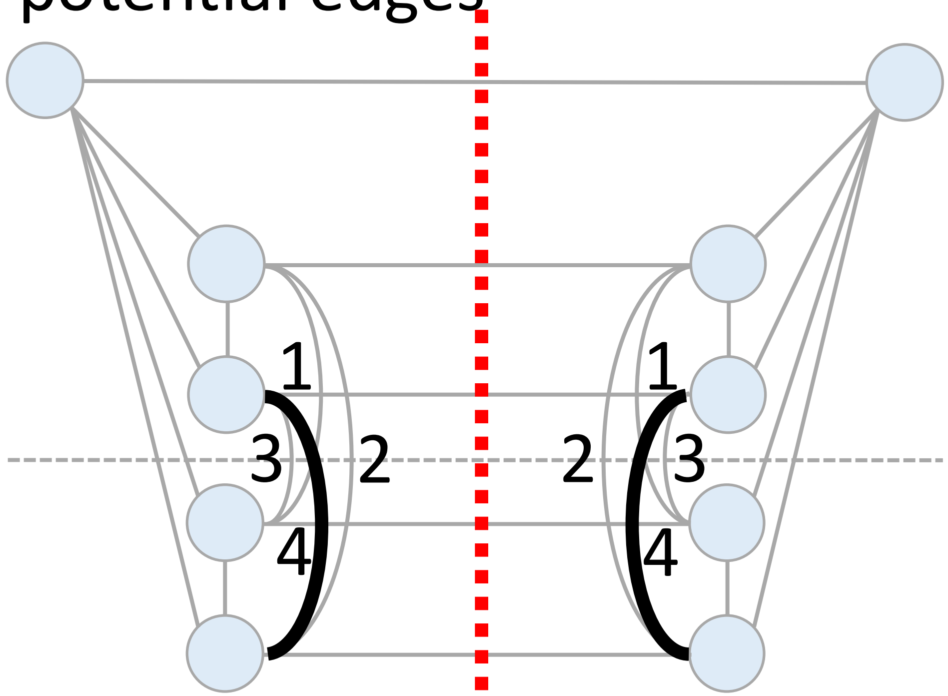
Label potential edges



Networks cannot compute their diameter in sublinear time!

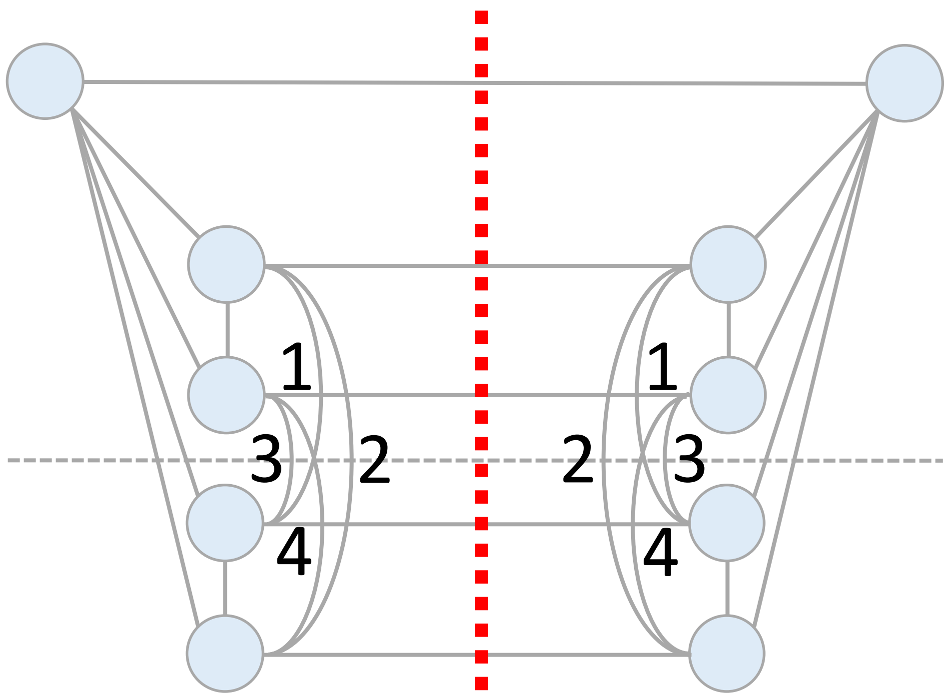
Upper and lower row not connected on any side?

Label potential edges



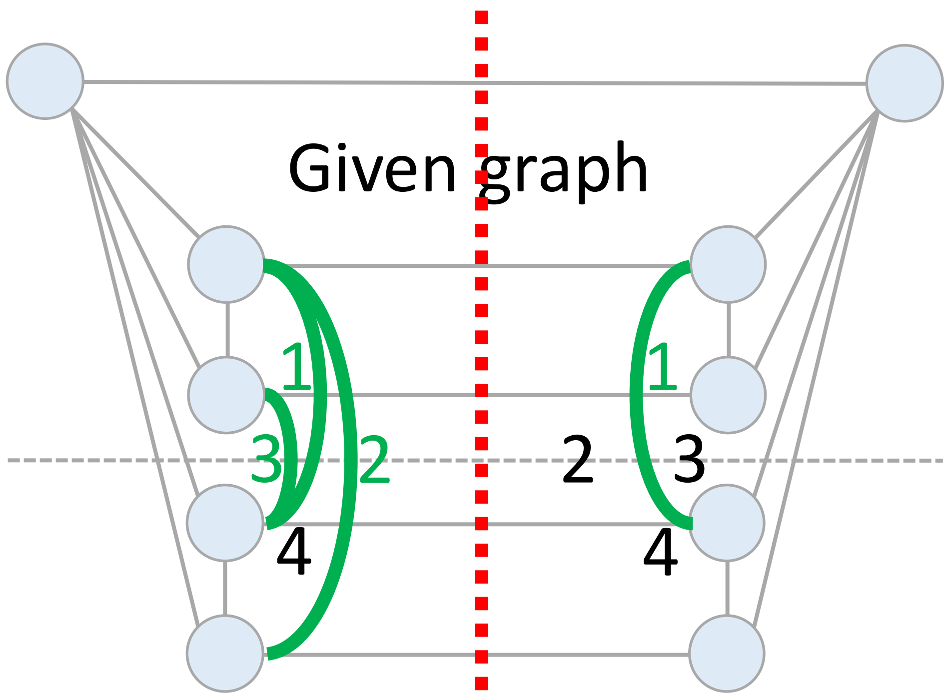
Networks cannot compute their diameter in sublinear time!

Upper and lower row not connected on any side?



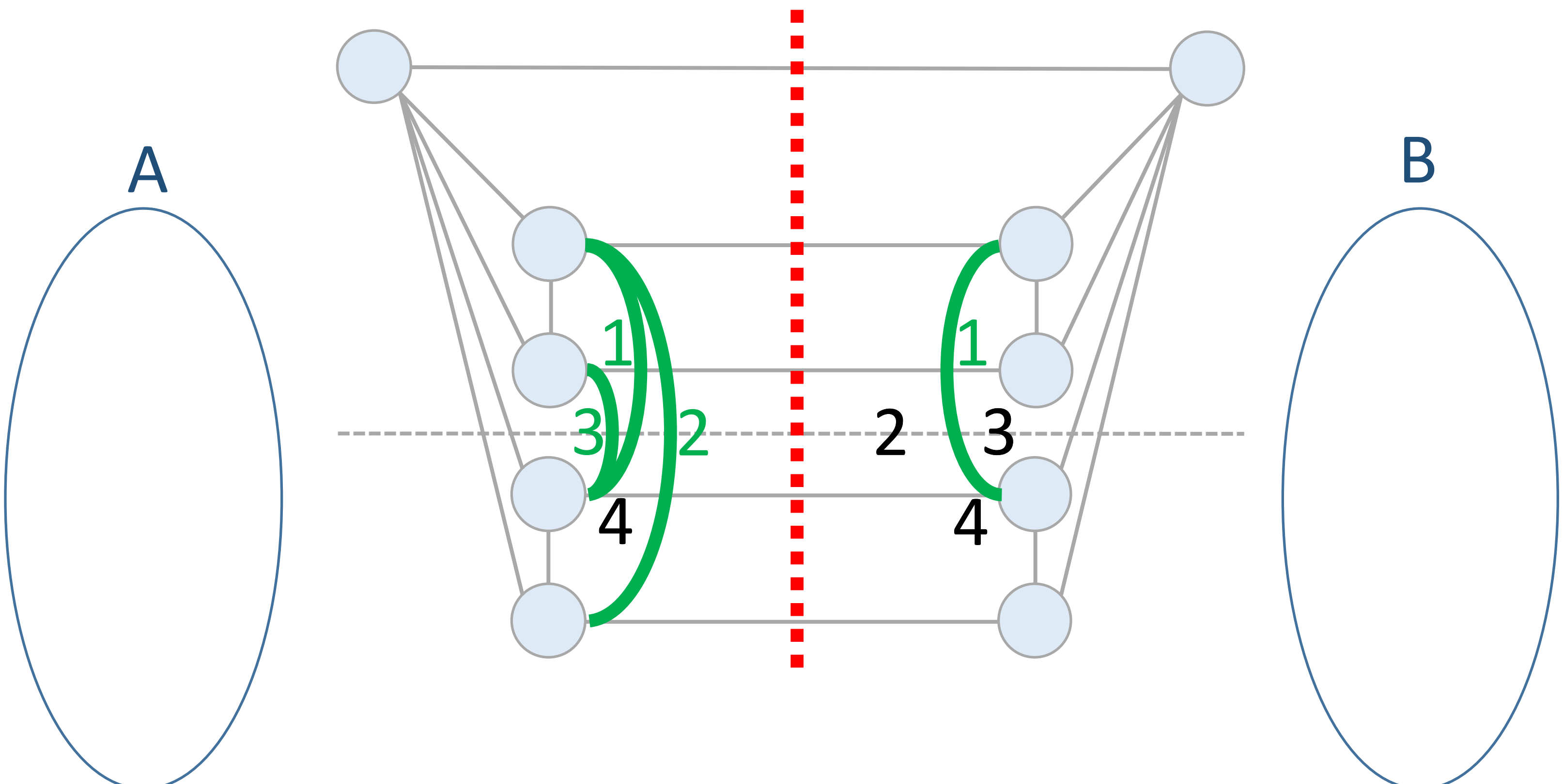
Networks cannot compute their diameter in sublinear time!

Upper and lower row not connected on any side?



Networks cannot compute their diameter in sublinear time!

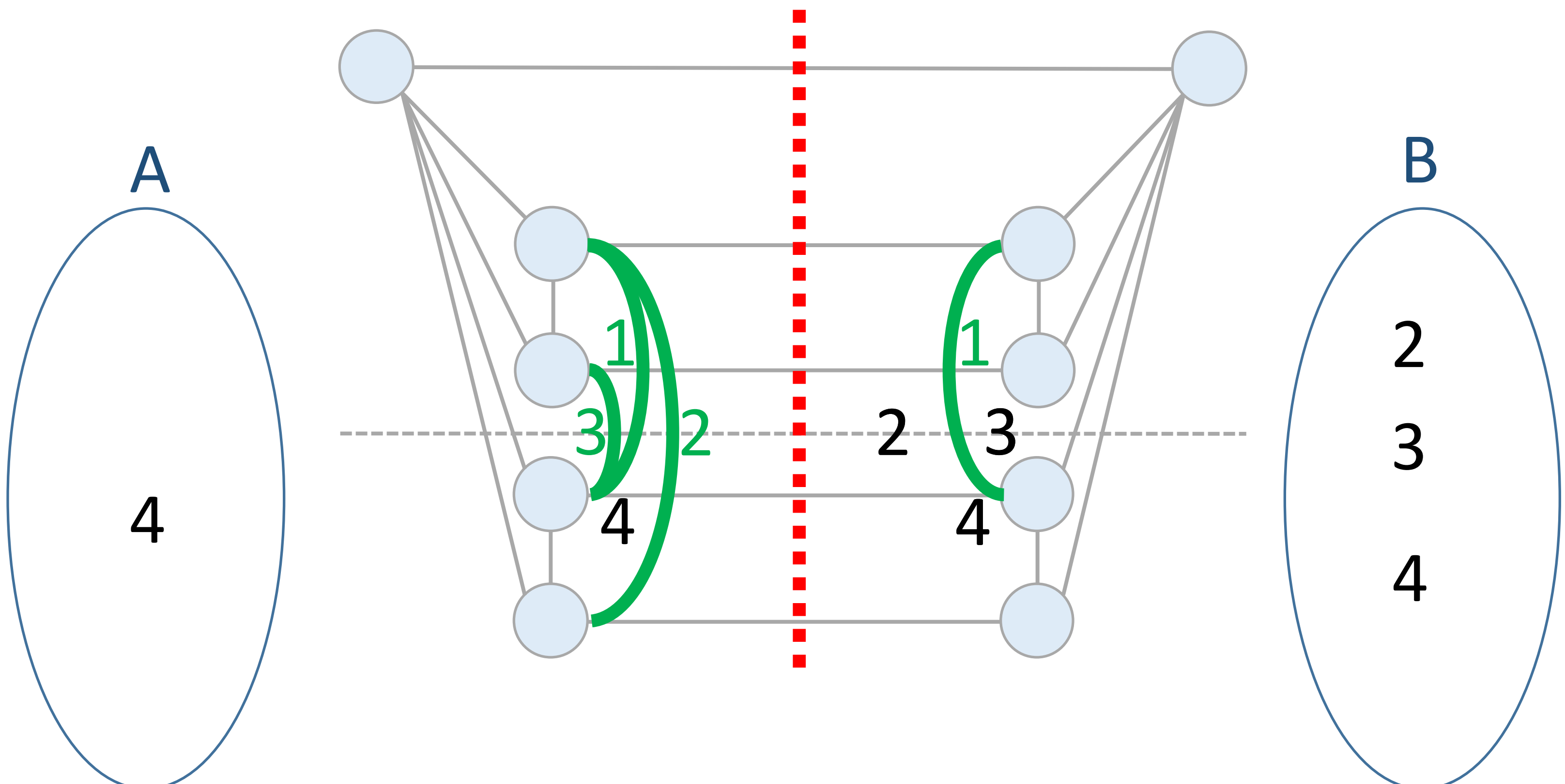
Upper and lower row not connected on any side?



Networks cannot compute their diameter in sublinear time!

Upper and lower row not connected on any side?

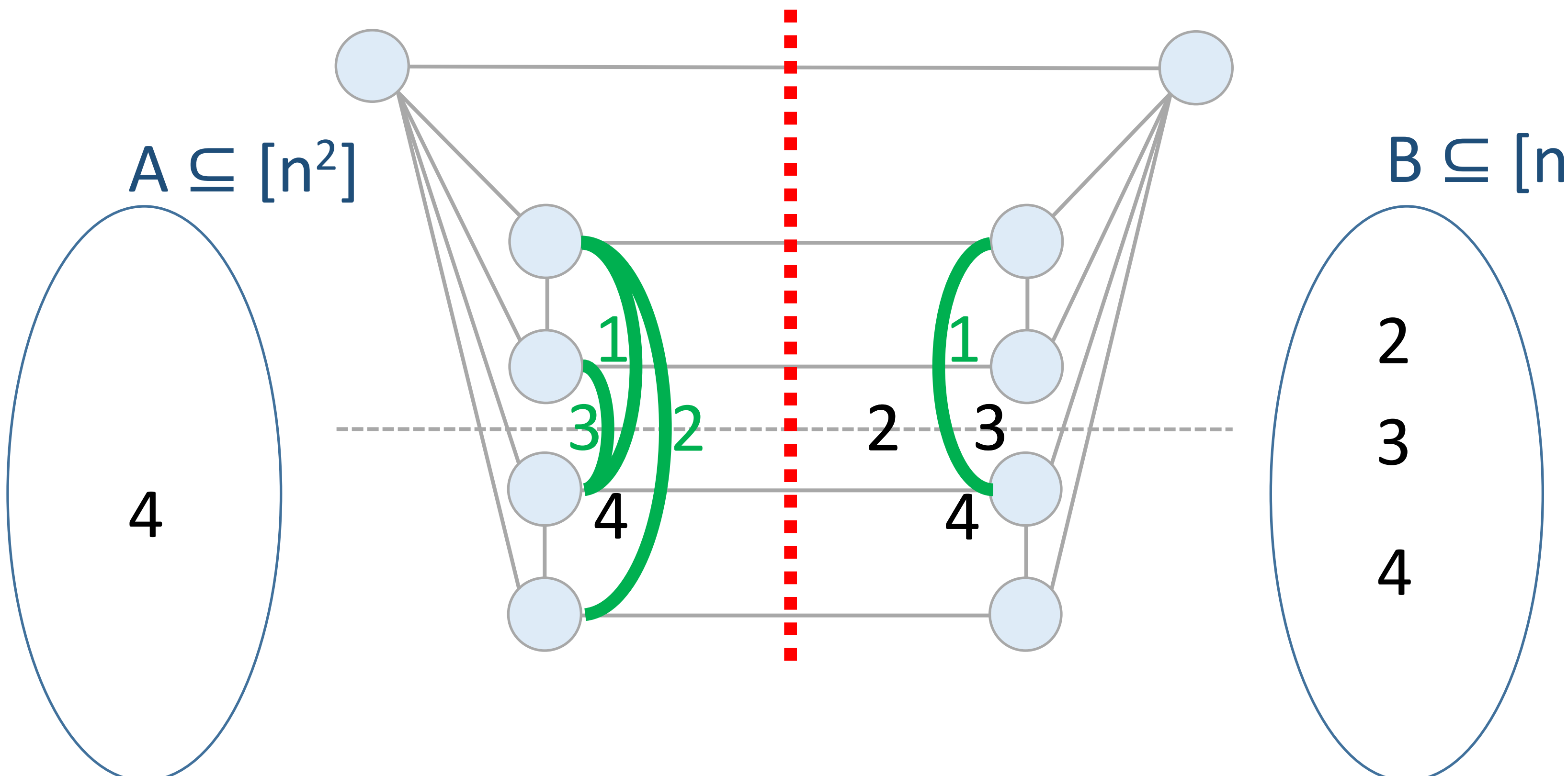
Same as "A and B not disjoint?"



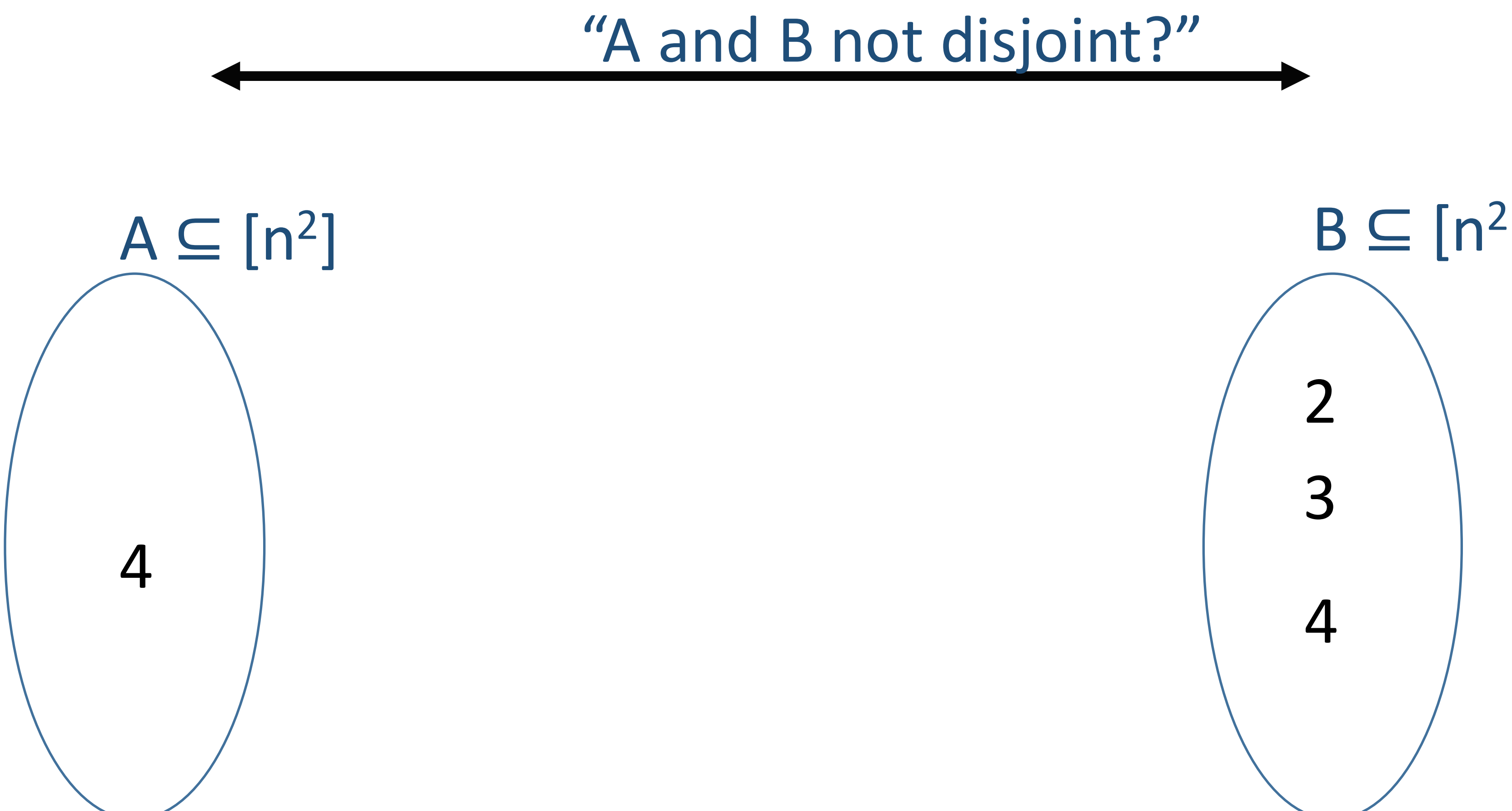
Networks cannot compute their diameter in sublinear time!

Upper and lower row not connected on any side?

Same as "A and B not disjoint?"



Networks cannot compute their diameter in sublinear time!



Networks cannot compute their diameter in sublinear time!

D = 2 or 3?

Upper and lower row not connected on any side?

Same as "A and B not disjoint?"

Communication Complexity

randomized: $\Omega(n^2)$ bits

$A \subseteq [n^2]$

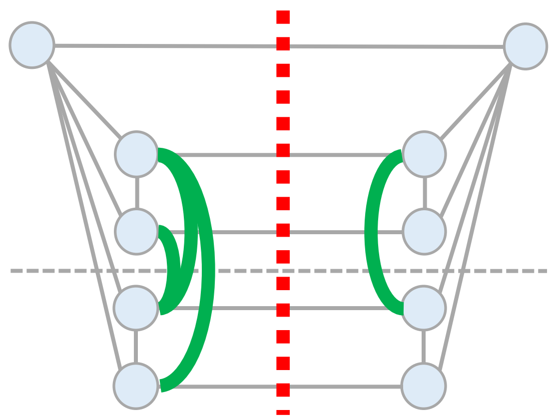
4

$B \subseteq [n^2]$

2

3

4



Ω(n) time

Θ(n) edges

Networks cannot compute their diameter in sublinear time!

Abboud, Censor-Hillel, Khoury - DISC 2016:

Even in sparse / constant degree graphs!

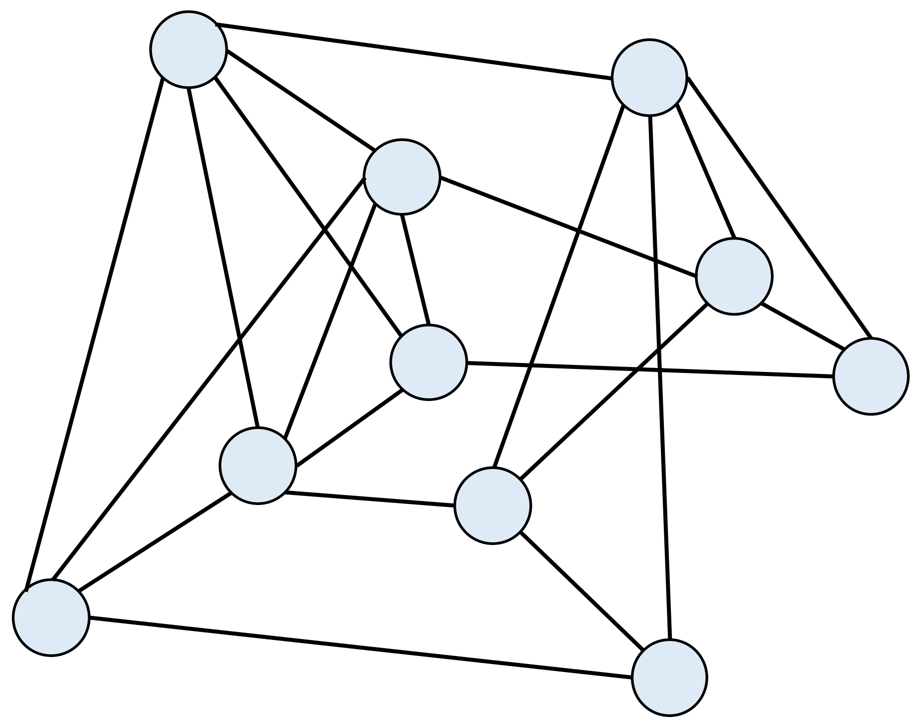
Networks cannot compute their diameter in sublinear time!

Networks can~~not~~ compute their diameter in ~~sub~~linear time!

APSP in $O(n)$

APSP in O(n)

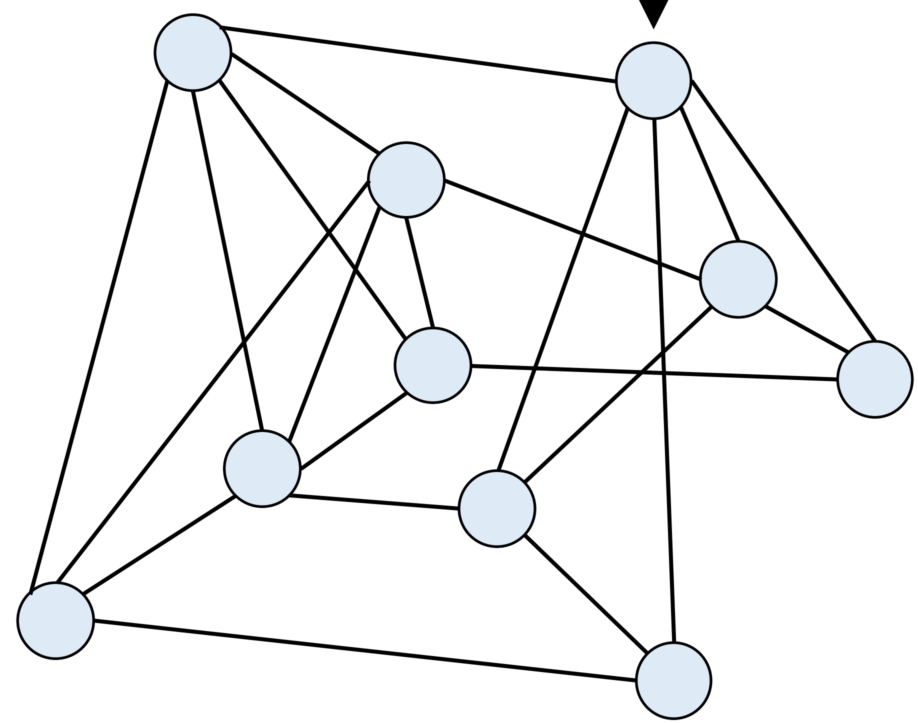
Compute All Pairs Shortest Paths



APSP in O(n)

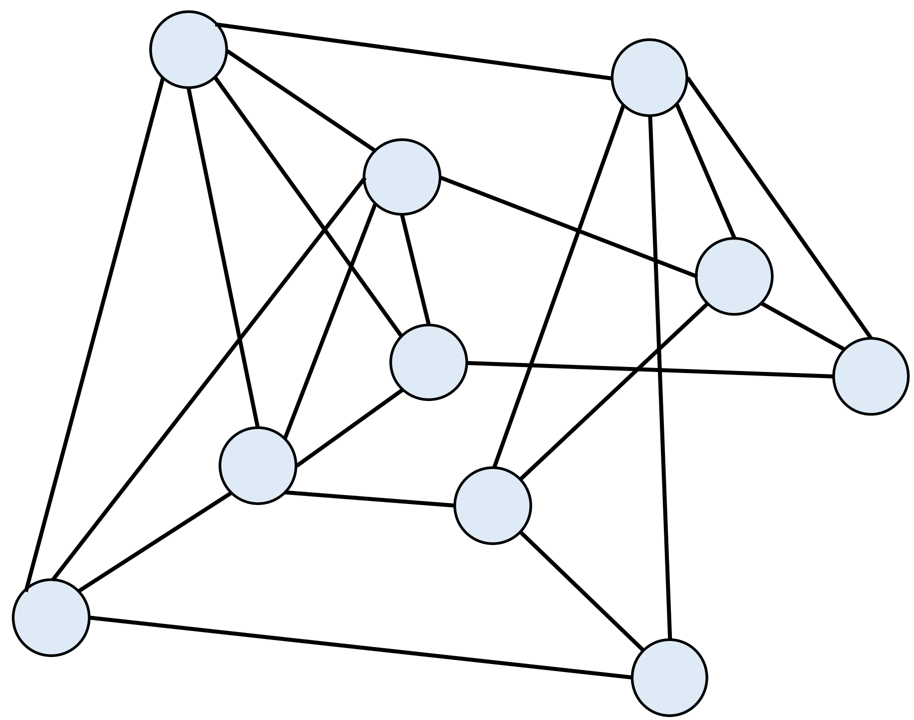
Compute All Pairs Shortest Paths

Knows its distance to all other nodes



APSP in O(n)

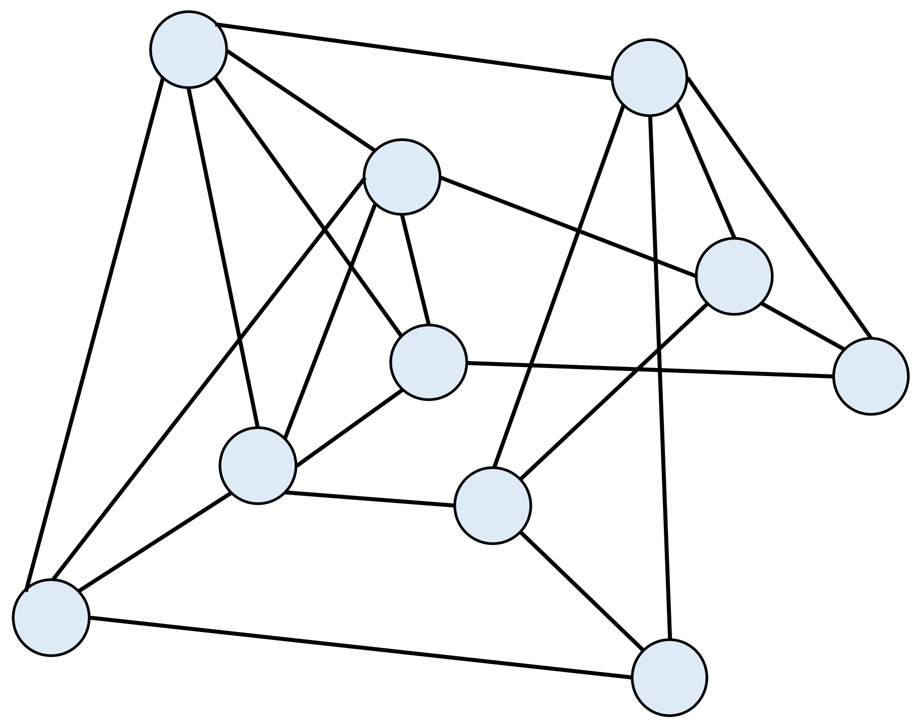
Compute All Pairs Shortest Paths



APSP in O(n)

Compute All Pairs Shortest Paths

```
For each node {
        compute distances to all other nodes;
}
```



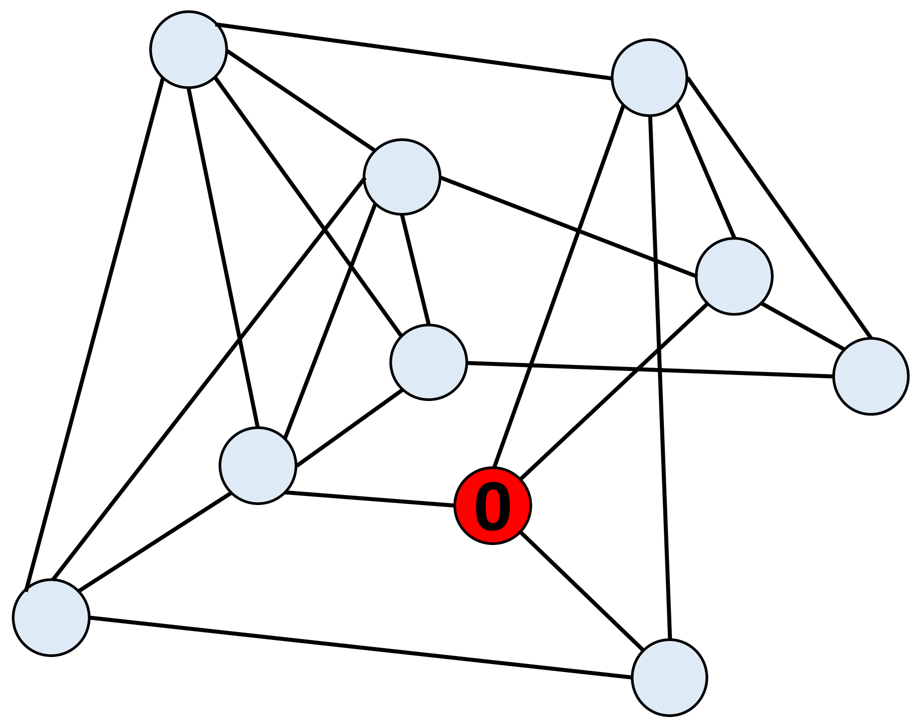
APSP in O(n)

Compute All Pairs Shortest Paths

For each node {

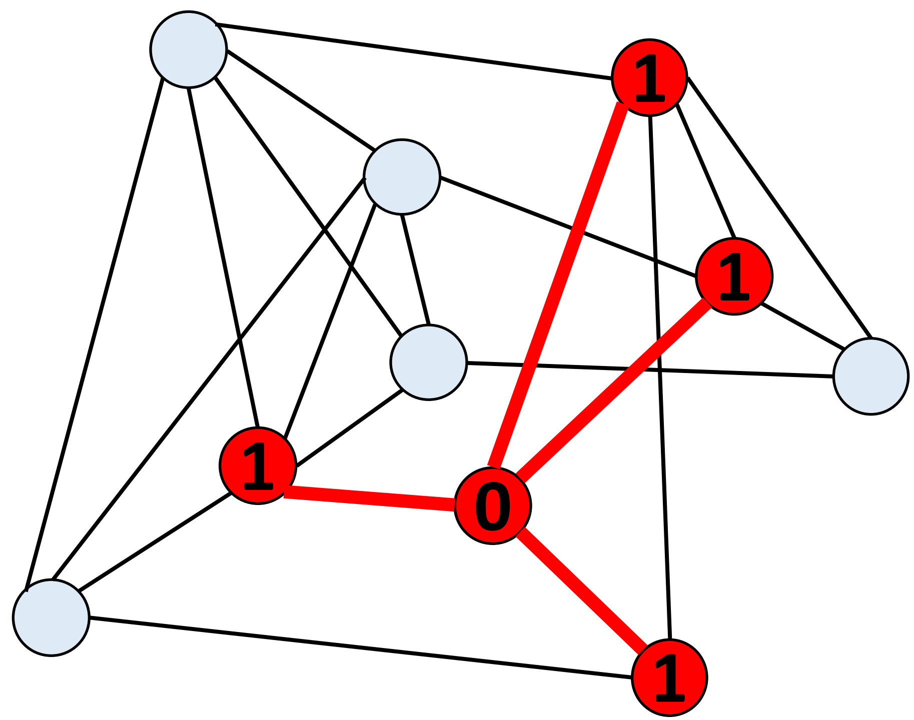
        compute distances to all other nodes;

}



APSP in O(n)

Compute All Pairs Shortest Paths
For each node {
        compute distances to all other nodes;
}



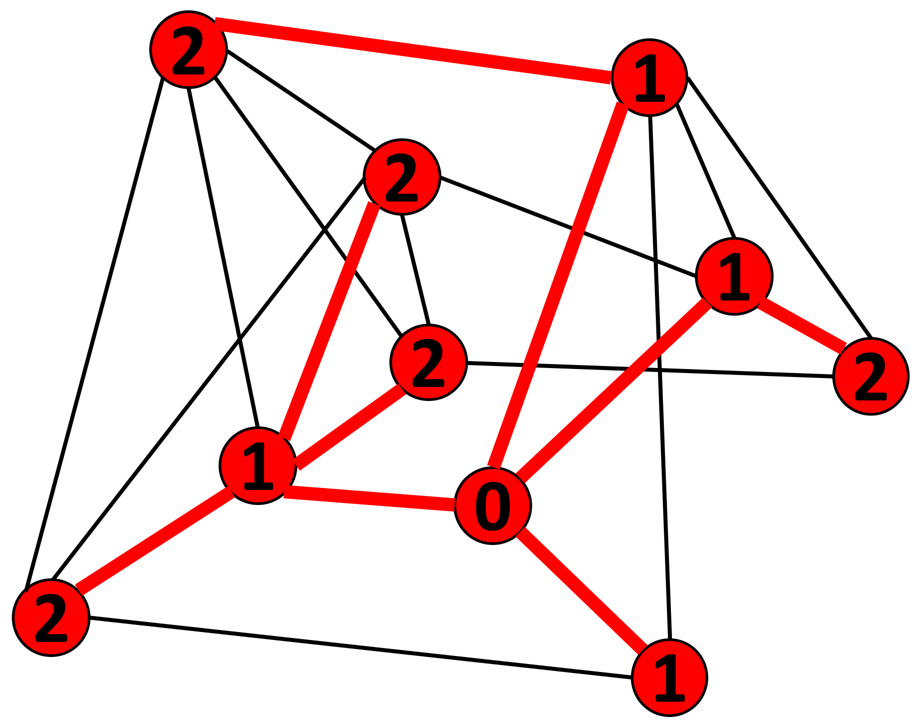
APSP in O(n)

Compute All Pairs Shortest Paths

For each node {

compute distances to all other nodes;

}



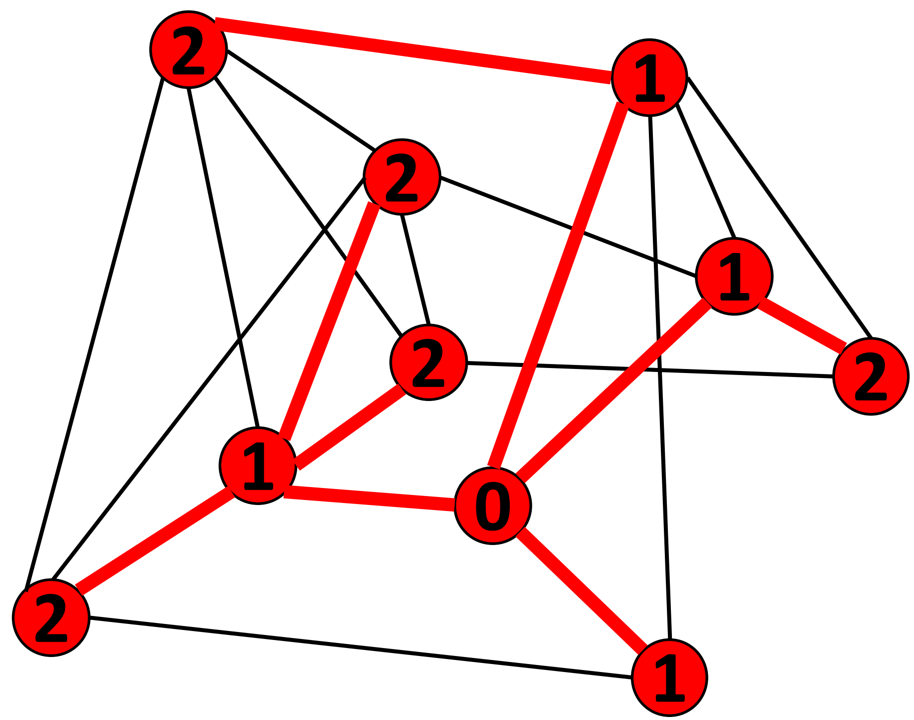
APSP in O(n)

Compute All Pairs Shortest Paths

For each node {

  compute distances to all other nodes;   O(D)

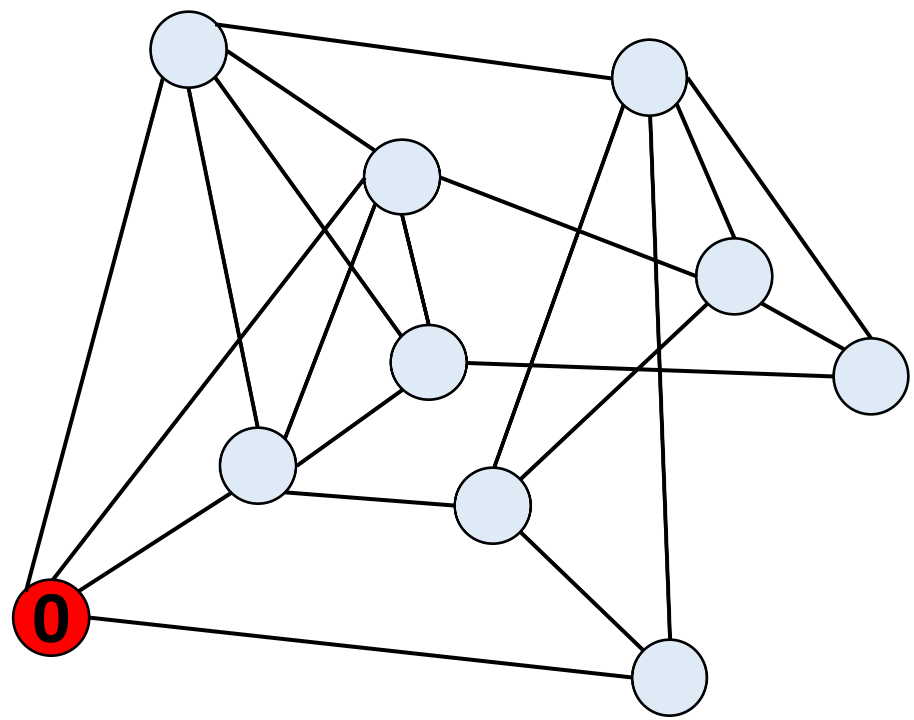
}



APSP in O(n)

Compute All Pairs Shortest Paths

```
For each node {
        compute distances to all other nodes;        O(D)
}
```



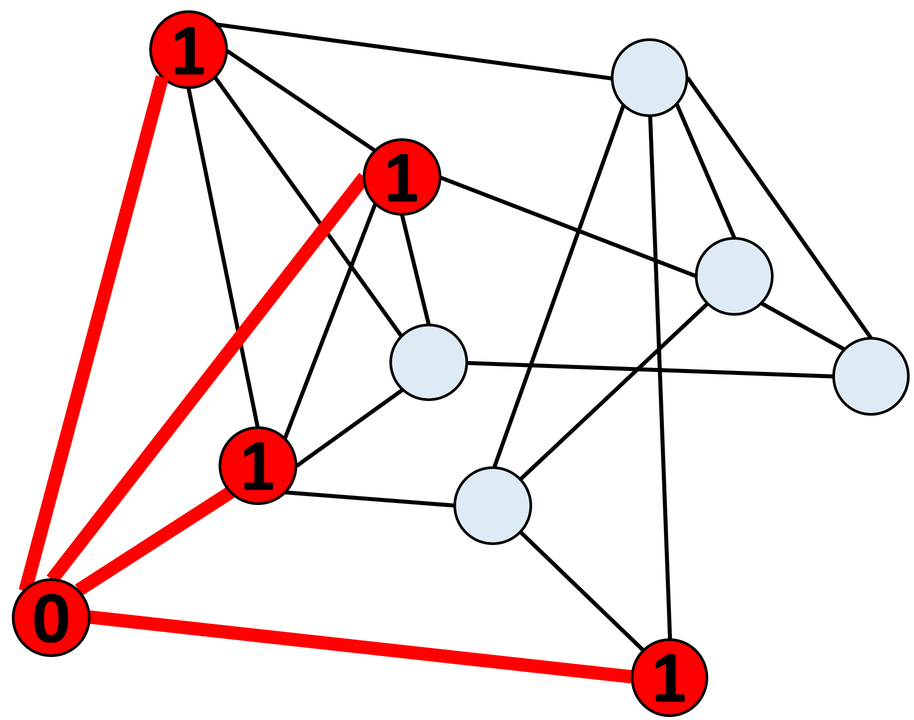
APSP in O(n)

Compute All Pairs Shortest Paths

For each node {

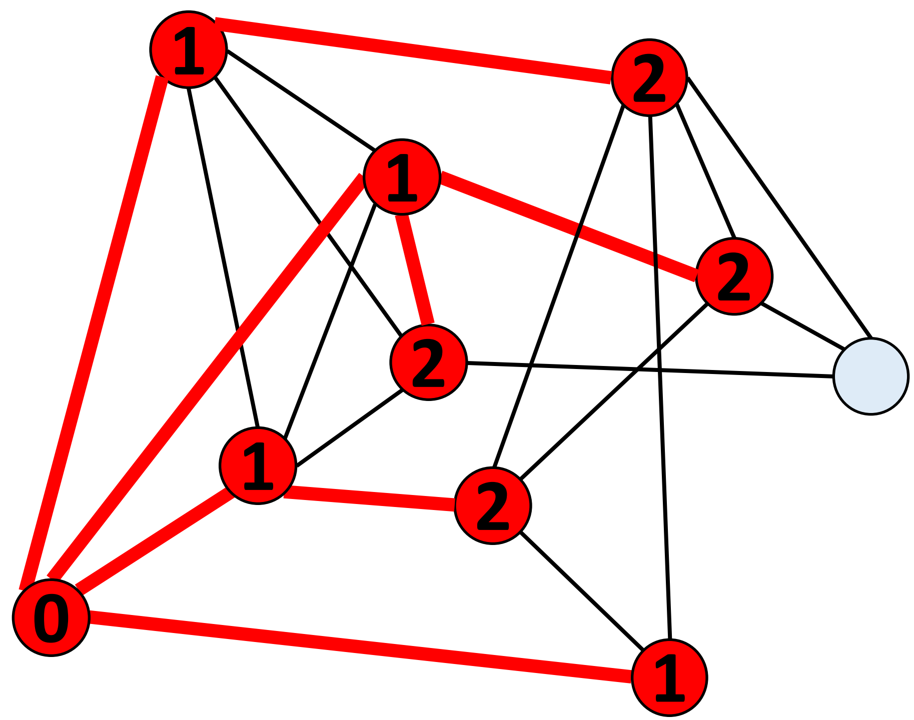
compute distances to all other nodes; O(D)

}



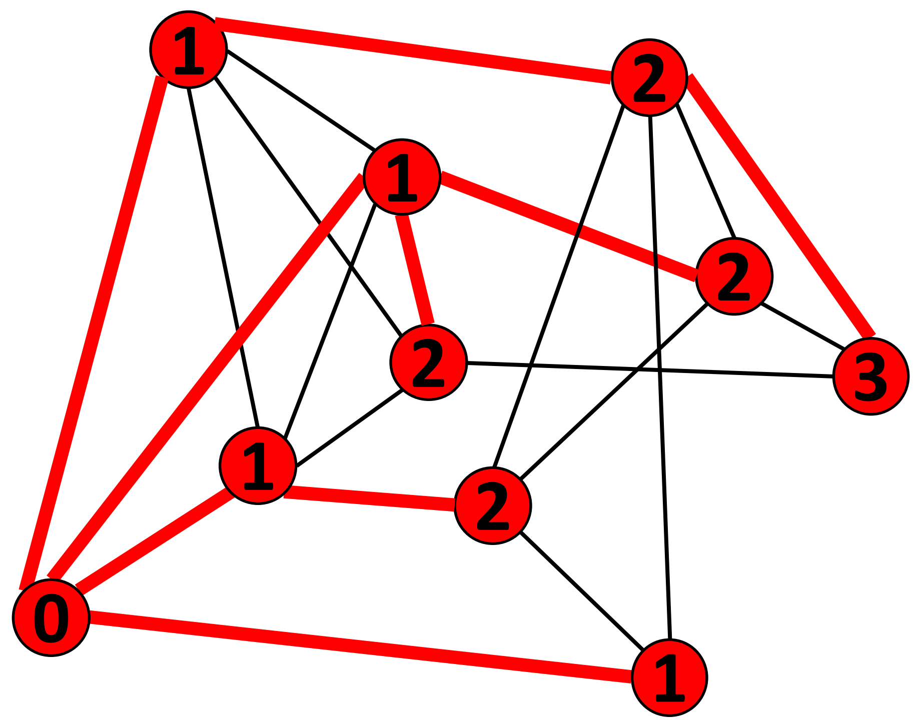
APSP in O(n)

Compute All Pairs Shortest Paths
For each node {
    compute distances to all other nodes;   O(D)
}



APSP in O(n)

Compute All Pairs Shortest Paths
For each node {
        compute distances to all other nodes;   O(D)
}



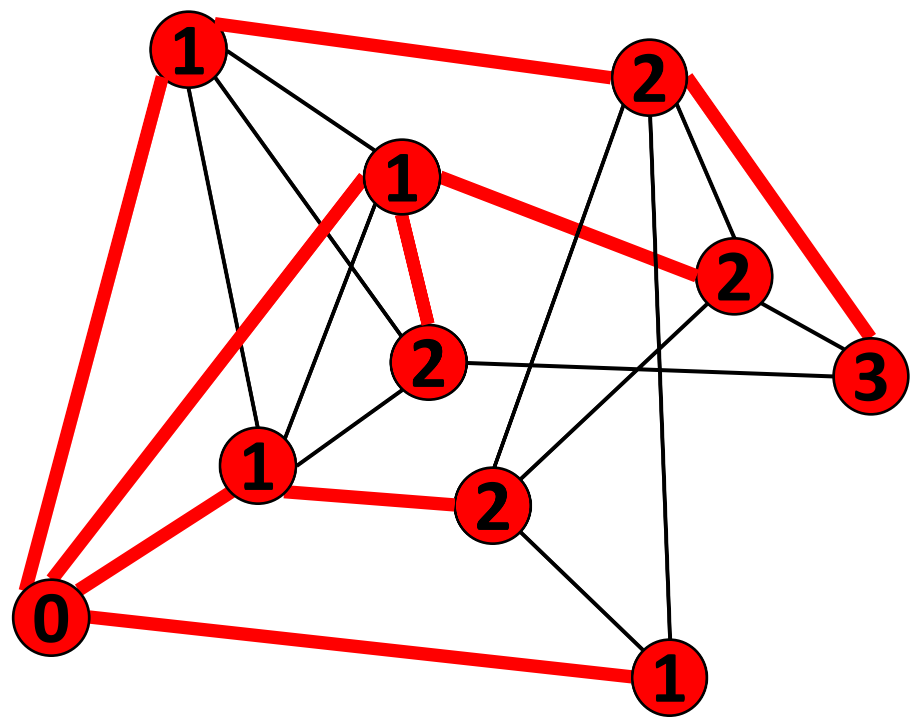
APSP in O(n)

Compute All Pairs Shortest Paths

For each node { O(n)

compute distances to all other nodes; O(D)

}



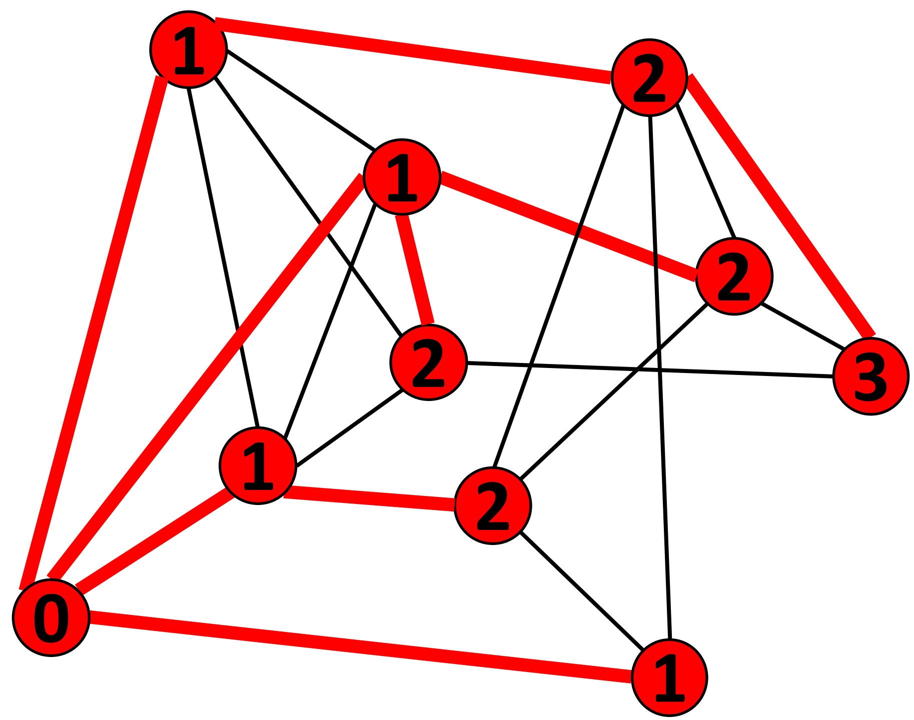
APSP in O(n)

Compute All Pairs Shortest Paths

For each node { — O(n)

compute distances to all other nodes; — O(D)

} — O(nD)



APSP in O(n)

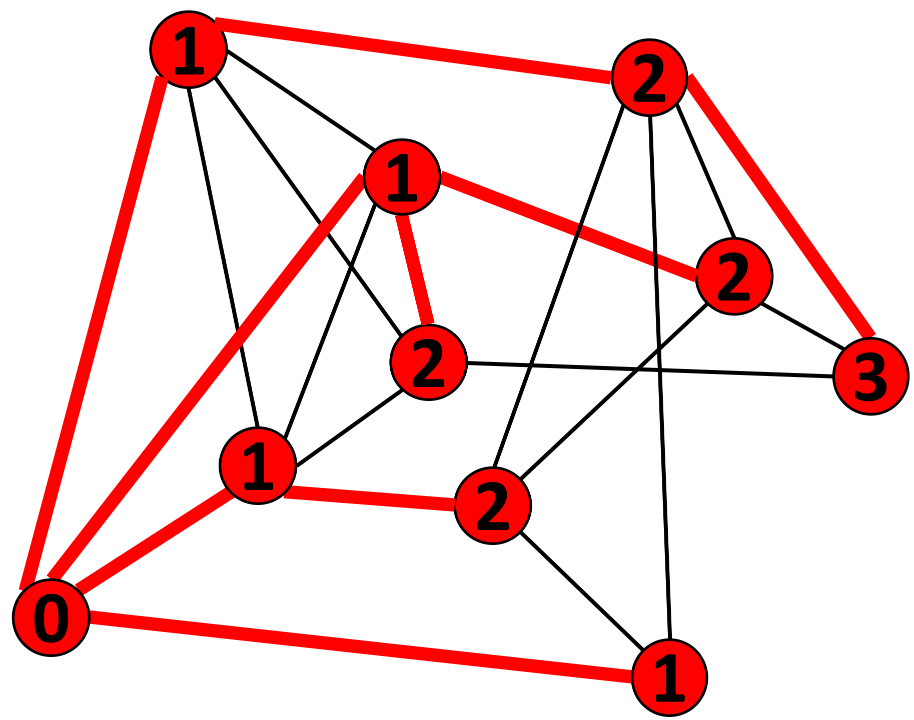
Compute All Pairs Shortest Paths

For each node { — O(n)

compute distances to all other nodes; — O(D)

} — O(nD)

Limited parallelism:
Only some nodes active.



APSP in O(n)

Compute All Pairs Shortest Paths

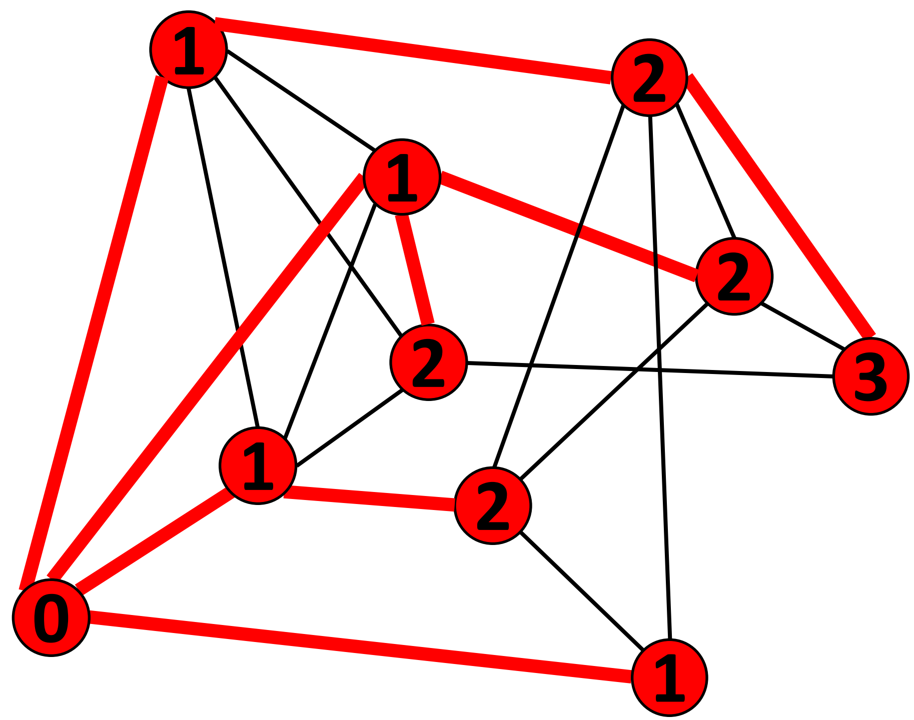
For each node { O(n)

compute distances to all other nodes; O(D)

} O(nD)

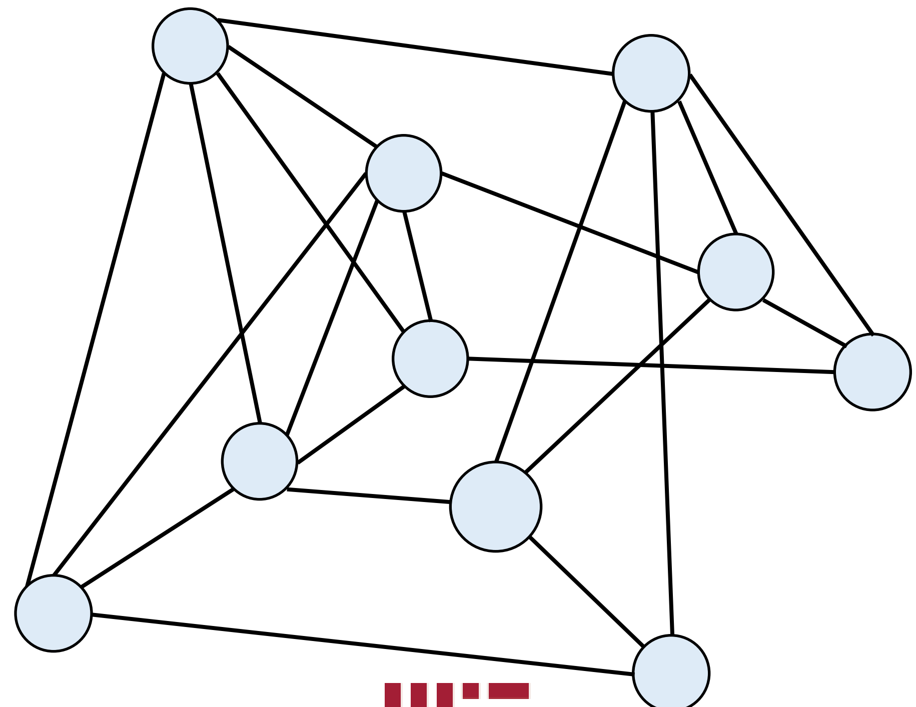
Limited parallelism:
Only some nodes active.

Wanted: All nodes active all the time!



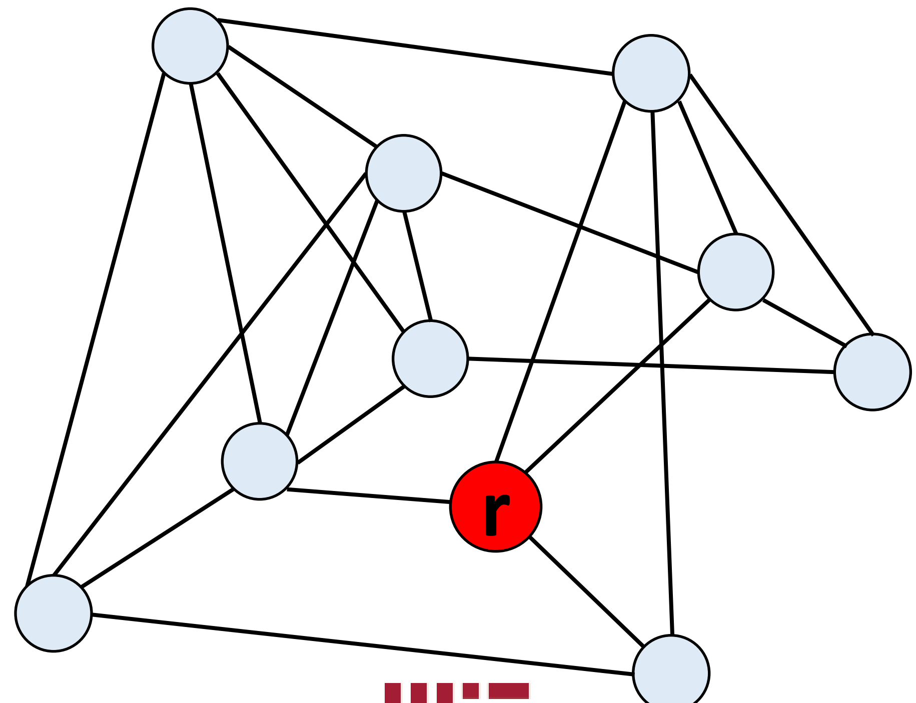
APSP in O(n)

Compute All Pairs Shortest Paths



APSP in O(n)

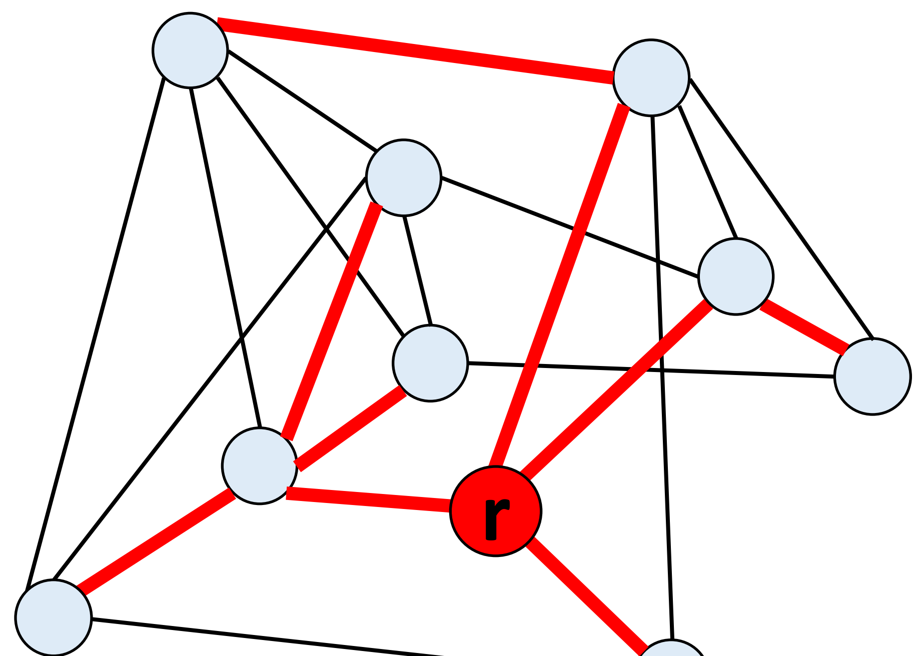
Compute All Pairs Shortest Paths



APSP in O(n)

Compute All Pairs Shortest Paths

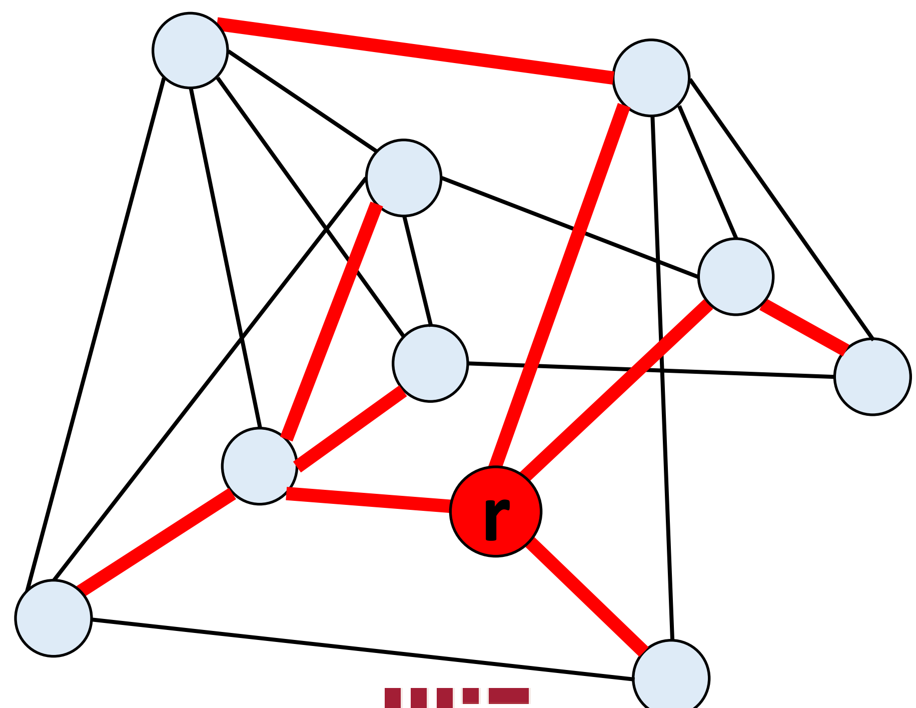
1. Pick a root-node r;



APSP in O(n)

Compute All Pairs Shortest Paths

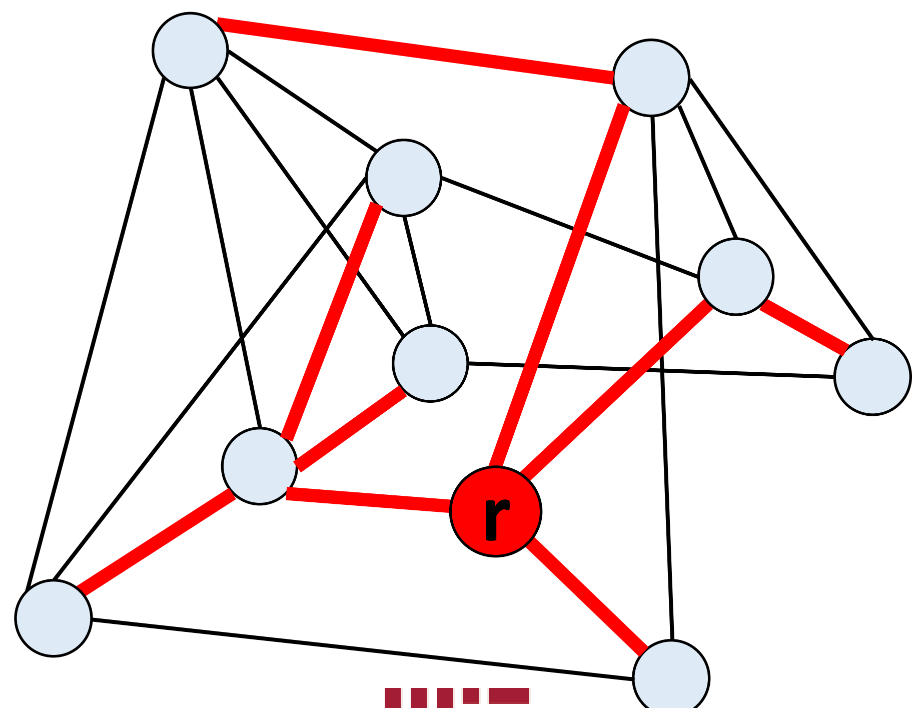
1. Pick a root-node r;
2. **T** := BFS-Tree(r);



APSP in O(n)

Compute All Pairs Shortest Paths

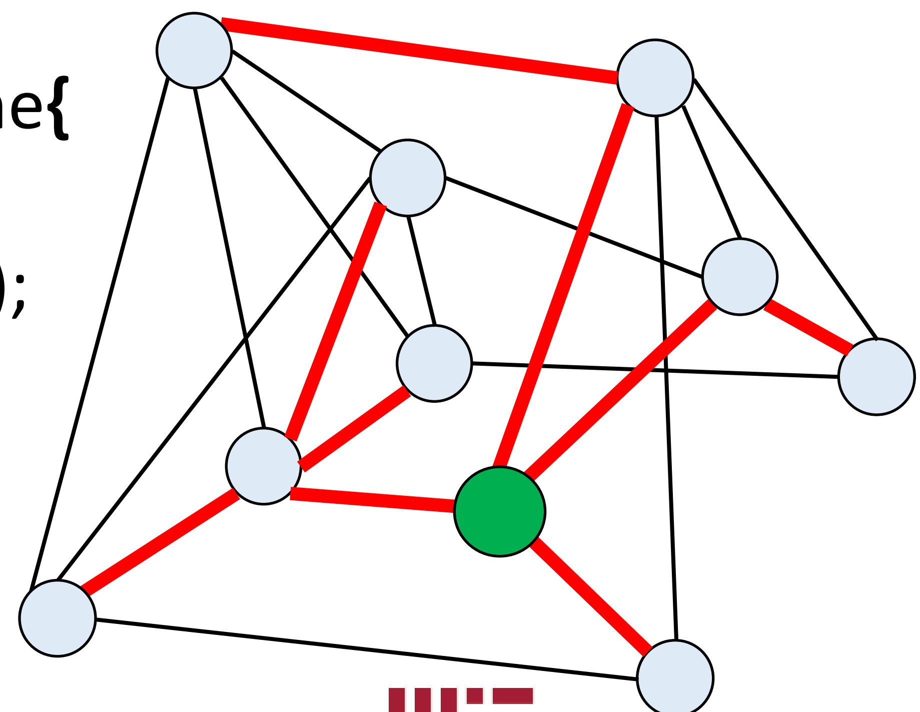
1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;



APSP in O(n)

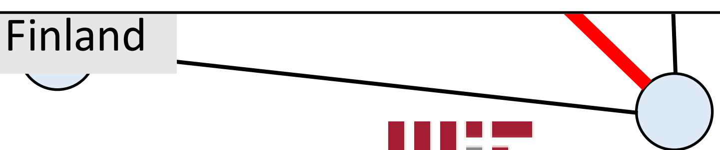
Compute All Pairs Shortest Paths

1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }



APSP in O(n)

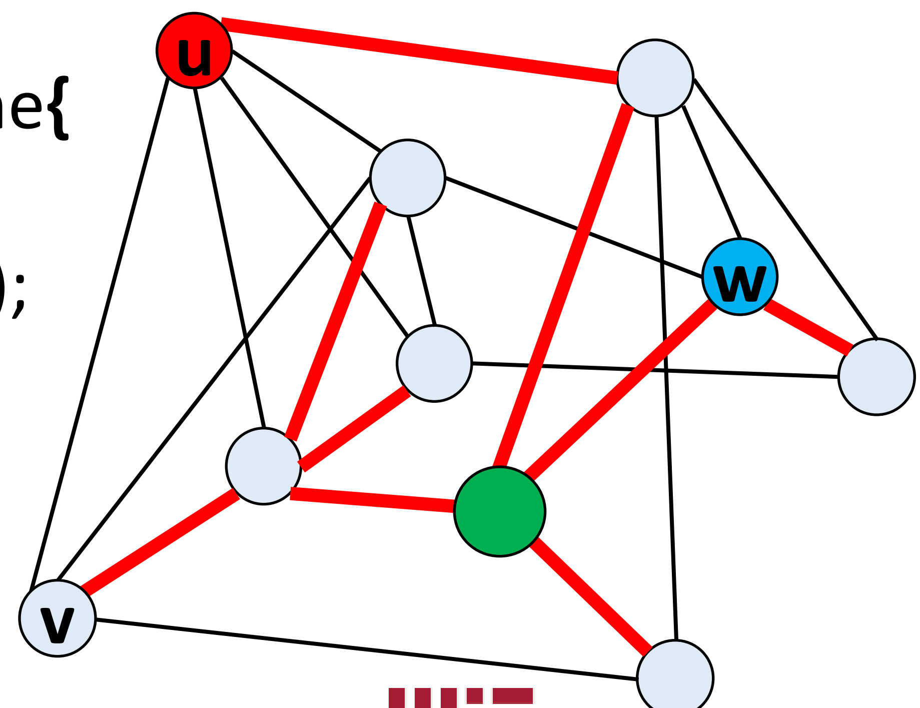
Animation by Jukka Suomela, Aalto University, Finland



APSP in O(n)

Compute All Pairs Shortest Paths

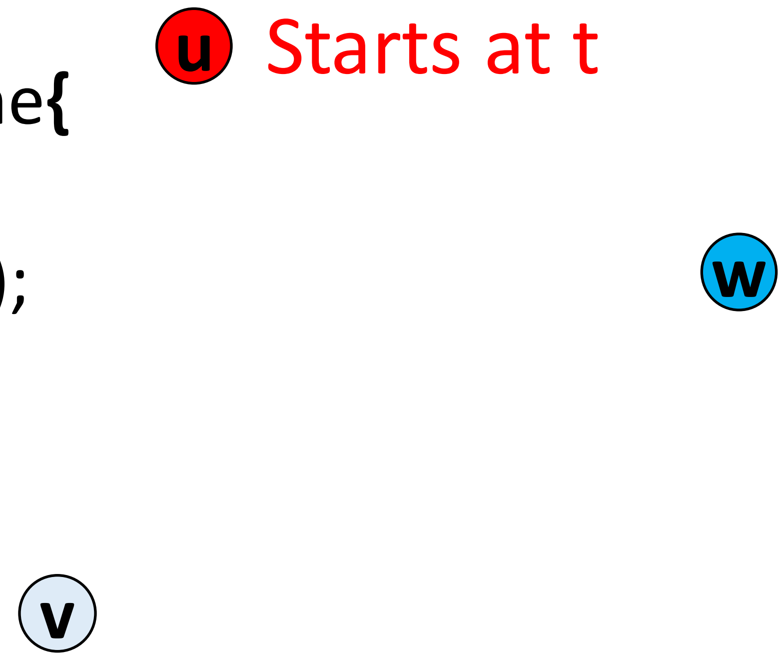
1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }



APSP in O(n)

Compute All Pairs Shortest Paths

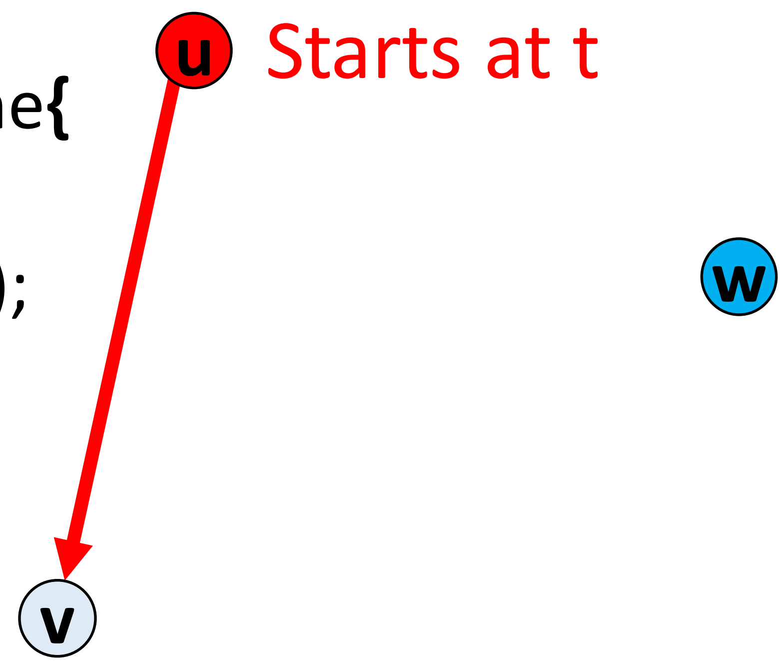
1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }



APSP in O(n)

Compute All Pairs Shortest Paths

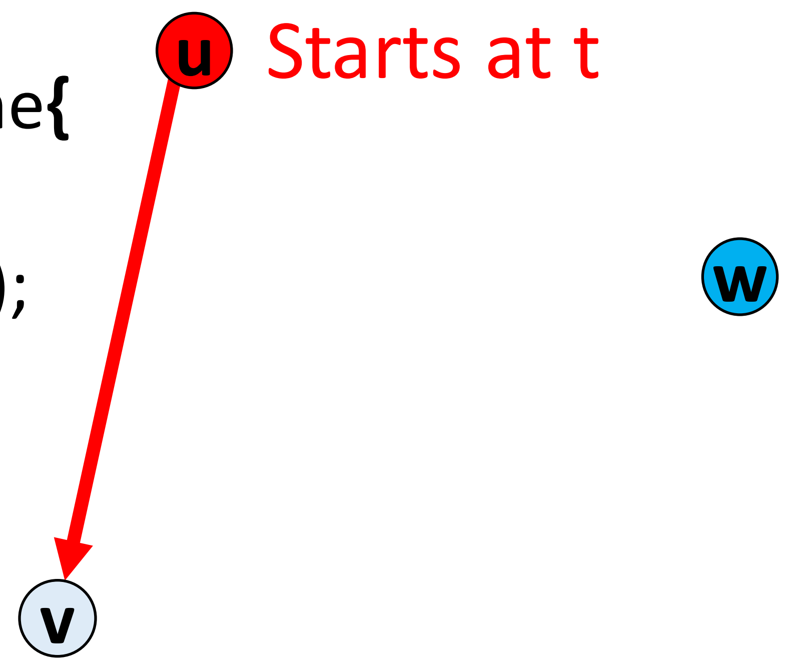
1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
    wait 1 timeslot;
    start shortest paths(v);
  }



APSP in O(n)

Compute All Pairs Shortest Paths

1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }

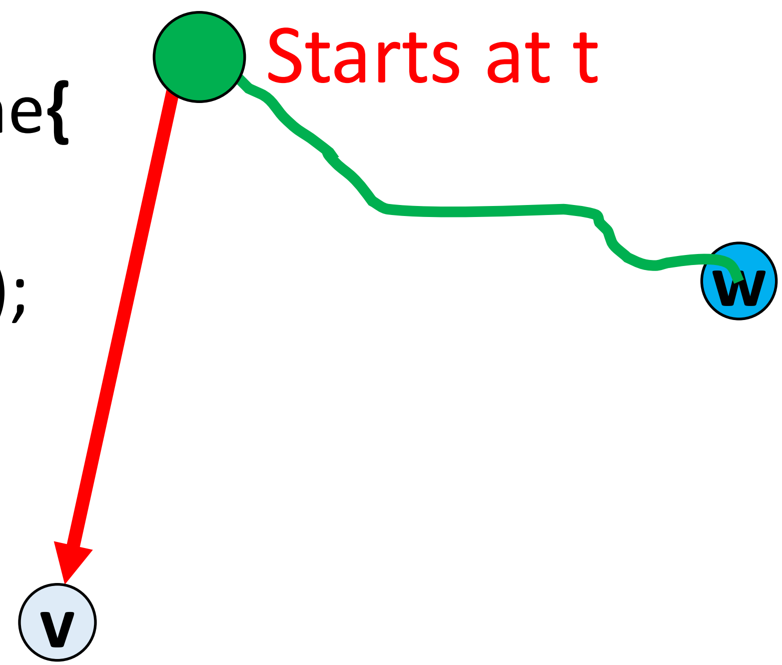


Arrives at $t + d(u, v)$

APSP in O(n)

Compute All Pairs Shortest Paths

1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }

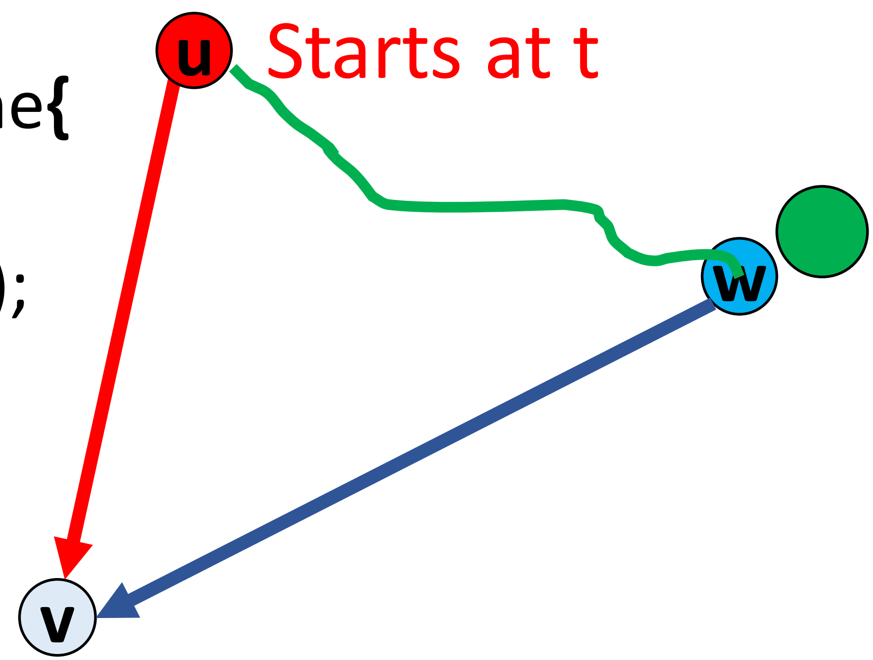


Arrives at $t + d(u, v)$

APSP in O(n)

Compute All Pairs Shortest Paths

1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }



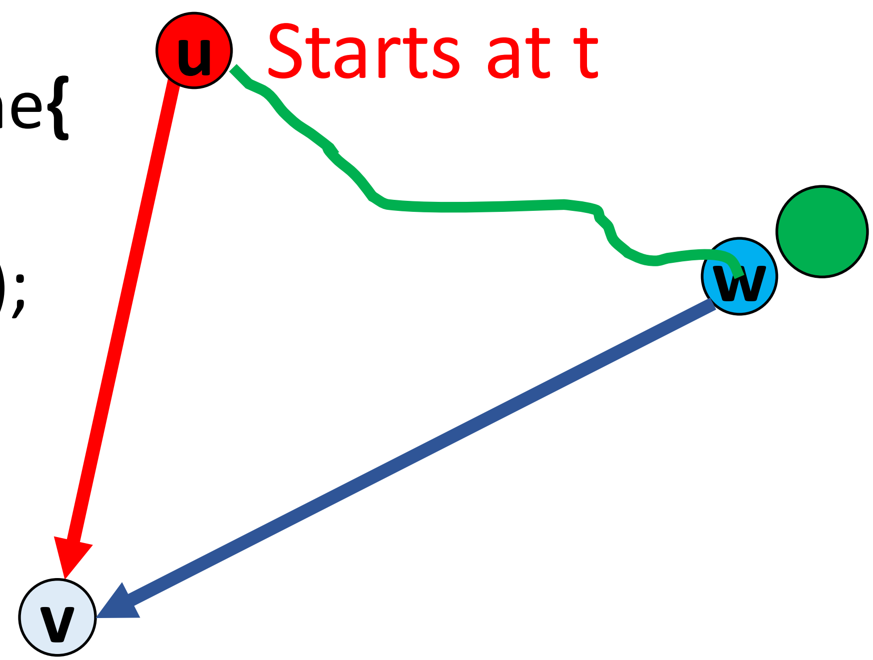
Arrives at $t + d(u, v)$

APSP in O(n)

Compute All Pairs Shortest Paths

1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);

   }



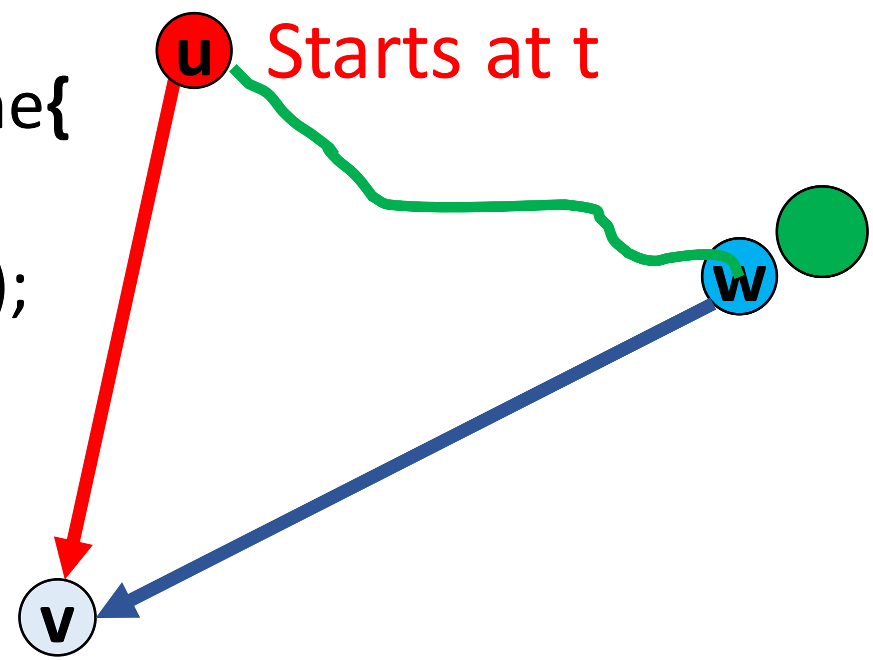
Arrives at $t + d(u, v)$

Arrives at $\geq t + d(u, v)$

APSP in O(n)

Compute All Pairs Shortest Paths

1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }



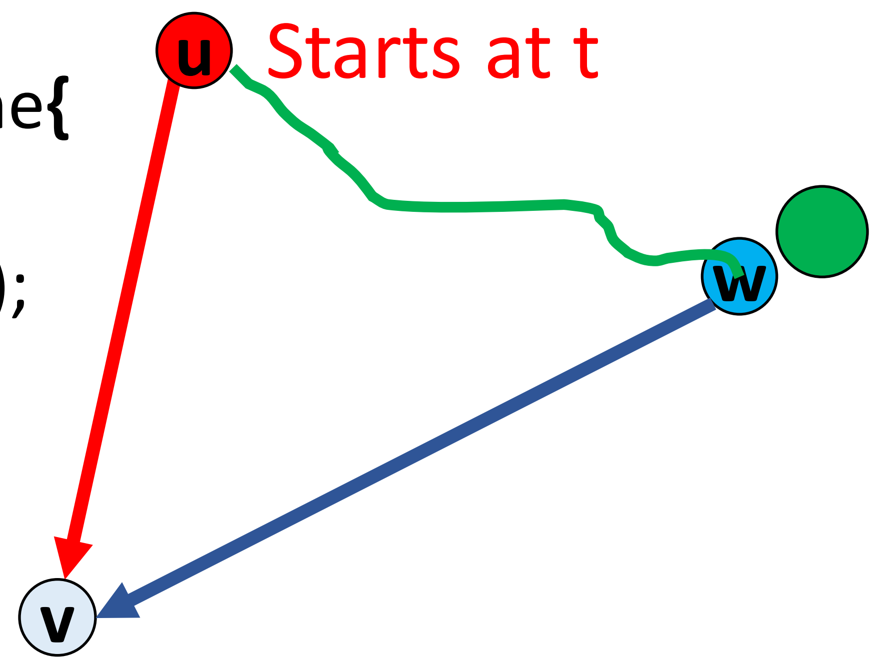
Arrives at $t + d(u, v)$

Arrives at $\geq t + d(u, v)$

APSP in O(n)

Compute All Pairs Shortest Paths

1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }



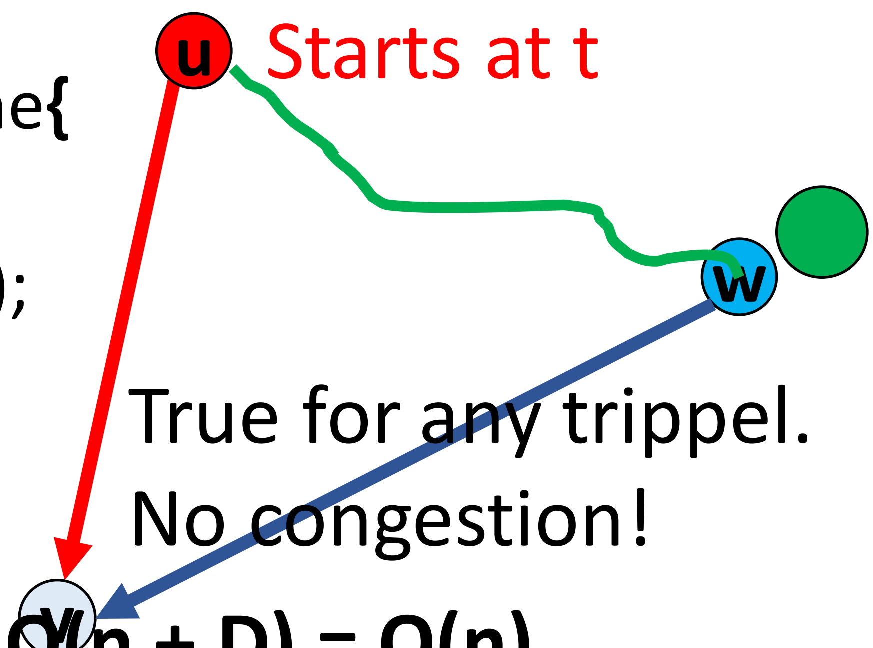
Arrives at $t + d(u,v)$

Arrives at $\geq t + d(u,v)$ **+1**

APSP in O(n)

Compute All Pairs Shortest Paths

1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }



v never active for u and w at the same time!

True for any trippel.
No congestion!

Arrives at $t + d(u, v)$

Arrives at $\geq t + d(u, v) + 1$

Runtime: O(n + D) = O(n)

Complexity of computing D? $\Theta(n)$

Sequential: open

Extentions

Extentions

Problem	Exact	(+, 1)	(x, 1 + ε)	(x, 3/2 - ε)	(x, 3/2)	(x, 3/2+ε)	(x, 2)
APSP	Θ(n)	Θ(n)	Θ(n)	Θ(n)	--	--	--
eccentricity	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	Θ(D)
diameter	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	$O(\sqrt{n}+D)$	$O\left(\sqrt{\frac{n}{D}}+D\right)$	Θ(D)
radius	O(n)	--	$O\left(\frac{n}{D}+D\right)$	--	--	--	Θ(D)
center	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
p. vertices	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
girth	O(n)	--	$O\left(\min\left(\frac{n}{g}+D\log\frac{D}{g},n\right)\right)$		--	--	--

Problem	(x, 2-ε)	(x, 2-1/g)
girth	$\Omega\left(\frac{\sqrt{n}}{D}+D\right)$	$O\left(n^{2/3}+D\log\frac{D}{g}\right)$

Extentions

Problem	Exact	(+, 1)	(x, 1 + ε)	(x, 3/2 - ε)	(x, 3/2)	(x, 3/2+ε)	(x, 2)
APSP	Θ(n)	Θ(n)	Θ(n)	Θ(n)	--	--	--
eccentricity	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	Θ(D)
diameter	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	$O(\sqrt{n}+D)$	$O\left(\sqrt{\frac{n}{D}}+D\right)$	Θ(D)
radius	O(n)	--	$O\left(\frac{n}{D}+D\right)$	--	--	--	Θ(D)
center	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
p. vertices	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
girth	O(n)	--	$O\left(\min\left(\frac{n}{g}+D\log\frac{D}{g},n\right)\right)$		--	--	--

Problem	(x, 2-ε)	(x, 2-1/g)
girth	$\Omega\left(\frac{\sqrt{n}}{D}+D\right)$	$O\left(n^{2/3}+D\log\frac{D}{g}\right)$

Extentions

Problem	Exact	(+, 1)	(x, 1 + ε)	(x, 3/2 - ε)	(x, 3/2)	(x, 3/2+ε)	(x, 2)
APSP	Θ(n)	Θ(n)	Θ(n)	Θ(n)	--	--	--
eccentricity	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	Θ(D)
diameter	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	$O(\sqrt{n}+D)$	$O\left(\sqrt{\frac{n}{D}}+D\right)$	Θ(D)
radius	O(n)	--	$O\left(\frac{n}{D}+D\right)$	--	--	--	Θ(D)
center	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
p. vertices	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
girth	O(n)	--	$O\left(\min\left(\frac{n}{g}+D\log\frac{D}{g},n\right)\right)$		--	--	--

Problem	(x, 2-ε)	(x, 2-1/g)
girth	$\Omega\left(\frac{\sqrt{n}}{D}+D\right)$	$O\left(n^{2/3}+D\log\frac{D}{g}\right)$

Extentions

Problem	Exac[illegible]	[illegible]	[illegible]ε)	(x, 3/2 - ε)	(x, 3/2)	(x, 3/2+ε)	(x, 2)
APSP	Θ(n)	Θ(n)	Θ(n)	Θ(n)	--	--	--
eccentricity	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	Θ(D)
diameter	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	$O(\sqrt{n}+D)$	$O\left(\sqrt{\frac{n}{D}}+D\right)$	Θ(D)
radius	O([illegible]	[illegible]	[illegible]	--	--	--	Θ(D)
center	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
p. vertices	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
girth	O(n)	[illegible]	[illegible]	$\log\frac{D}{g}, n))$	--	--	--

Routing tables

Social networks

Fighting spam

Problem	(x, 2-ε)	(x, 2-1/g)
girth	$\Omega\left(\frac{\sqrt{n}}{D}+D\right)$	$O\left(n^{2/3}+D\log\frac{D}{g}\right)$

Extentions

Problem	Exac		ε)	(x, 3/2 - ε)	(x, 3/2)	(x, 3/2+ε)	(x, 2)
APSP	Θ(n)	Θ(n)	Θ(n)	Θ(n)	--	--	--
eccentricity	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	Θ(D)
diameter	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	$O(\sqrt{n}+D)$	$O\left(\sqrt{\frac{n}{D}}+D\right)$	Θ(D)
radius	O(			--	--	--	Θ(D)
center	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
p. vertices	Θ(n)	$\Omega\left(\frac{n}{D}+D\right)$	$O\left(\frac{n}{D}+D\right)$	$\Omega\left(\sqrt{\frac{n}{D}}+D\right)$	--	--	0
girth	O(n)			$\log\frac{D}{g}, n))$	--	--	--

Routing tables

Social networks

Fighting spam

Problem	(x, 2-ε)	(x, 2-1/g)
girth	$\Omega\left(\frac{\sqrt{n}}{D}+D\right)$	$O\left(n^{2/3}+D\log\frac{D}{g}\right)$

Also: good approximation algorithms for weighted graphs known. [Henzinger, Nanongkai et al.]

(x,1+ε)-Approximating Diameter

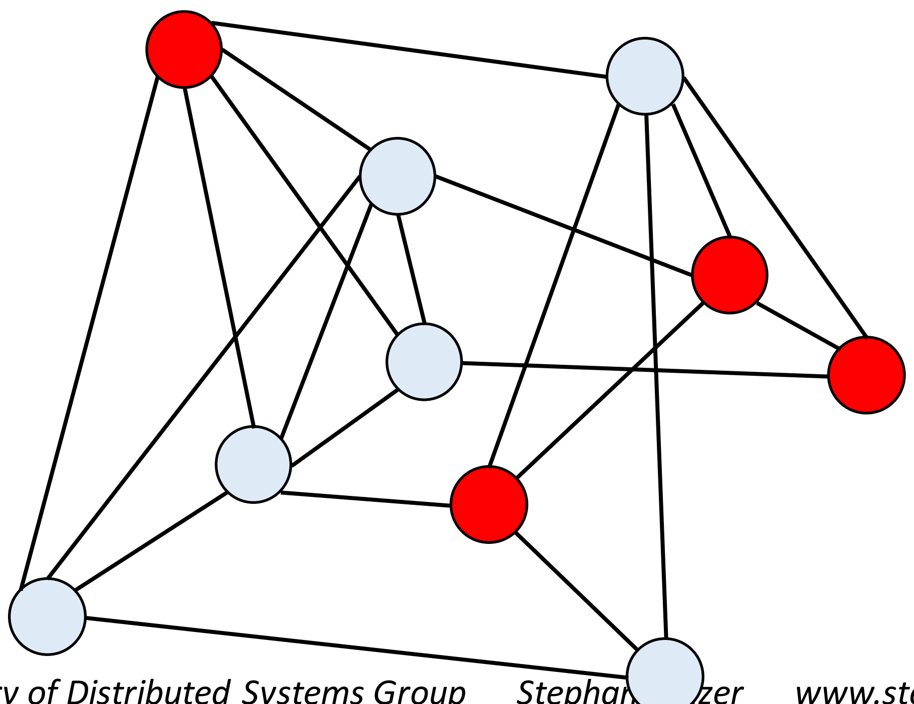
(x,1+ε)-Approximating Diameter

S-Shortest Path in O(|S| + D)

(x,1+ε)-Approximating Diameter

S-Shortest Path in O(|S| + D)

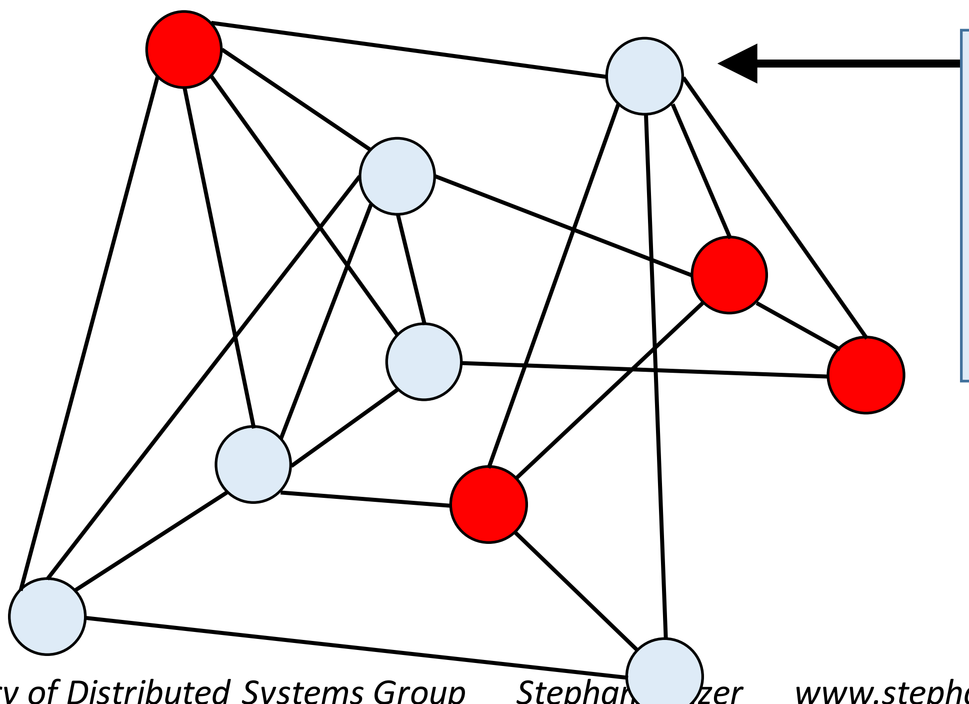
Shortest paths between S x V



(x,1+ε)-Approximating Diameter

S-Shortest Path in O(|S| + D)

Shortest paths between S x V

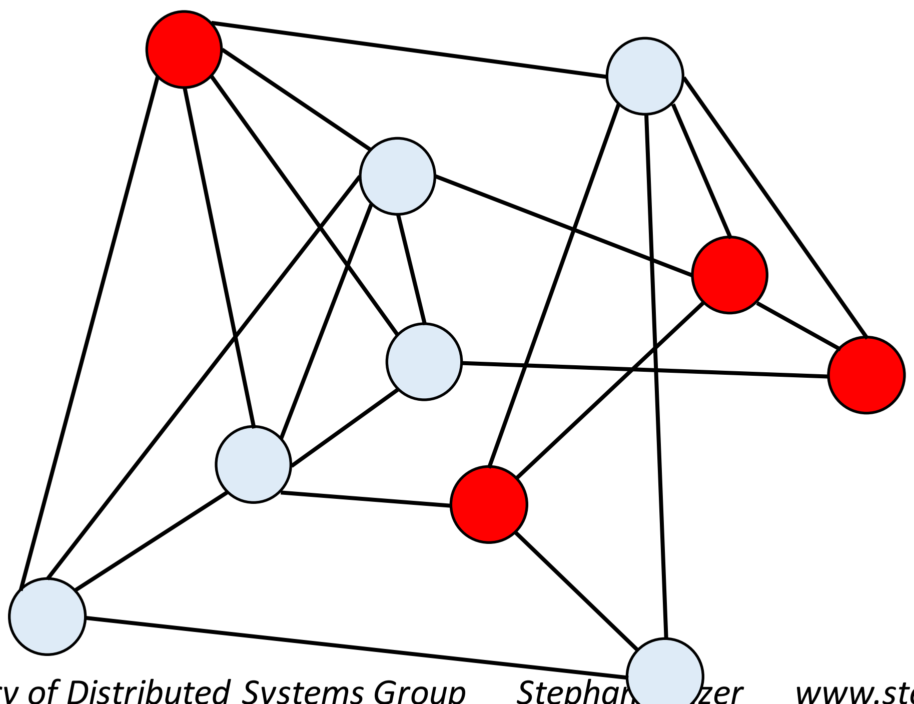


Knows its distance to nodes in S

(x,1+ε)-Approximating Diameter

S-Shortest Path in O(|S| + D)

Shortest paths between S x V



ALGO:

1. Start BFS in all S-nodes
2. Messages are forwarded depending on ID and distance traveled so far

(x,1+ε)-Approximating Diameter

S-Shortest Path in O(|S| + D)

(x,1+ε)-Approximating Diameter

S-Shortest Path in O(|S| + D)

S:= Small
O(D/ε)-Dominating Set

[Kutten, Peleg 1998]

(x,1+ε)-Approximating Diameter

S-Shortest Path in $O(|S| + D)$

S:= Small
$O(D/\varepsilon)$-Dominating Set

[Kutten, Peleg 1998]

Runtime: $O(D + \varepsilon n/D + D)$

(x,1+ε)-Approximating Diameter

S-Shortest Path in $O(|S| + D)$

S:= Small
$O(D/\varepsilon)$-Dominating Set

[Kutten, Peleg 1998]

Runtime: $O(D + \varepsilon n/D + D)$

(x,1+ε)-Approximating Diameter

S-Shortest Path in $O(|S| + D)$

S:= Small
$O(D/\varepsilon)$-Dominating Set

[Kutten, Peleg 1998]

Runtime: $O(D + \varepsilon n/D + D)$

(x,1+ε)-Approximating Diameter

S-Shortest Path in $O(|S| + D)$

S:= Small
$O(D/\varepsilon)$-Dominating Set

[Kutten, Peleg 1998]

Runtime: $O(n/D + D)$

(x,1+ε)-Approximating Diameter

S-Shortest Path in $O(|S| + D)$

S:= Small
$O(D/\varepsilon)$-Dominating Set

[Kutten, Peleg 1998]

Runtime: $O(n/D + D)$
Maximal error: D/ε

(x,1+ε)-Approximating Diameter

S-Shortest Path in $O(|S| + D)$

S:= Small

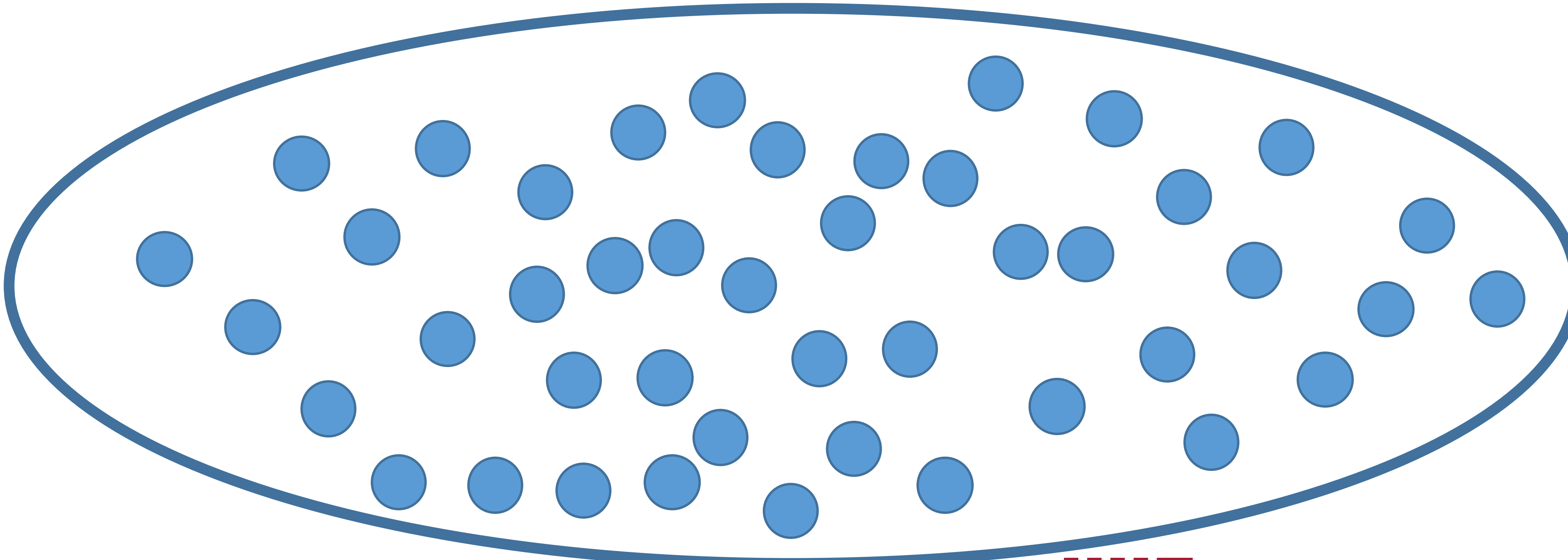
$O(D/\varepsilon)$-Dominating Set

[Kutten, Peleg 1998]

Runtime: $O(n/D + D)$

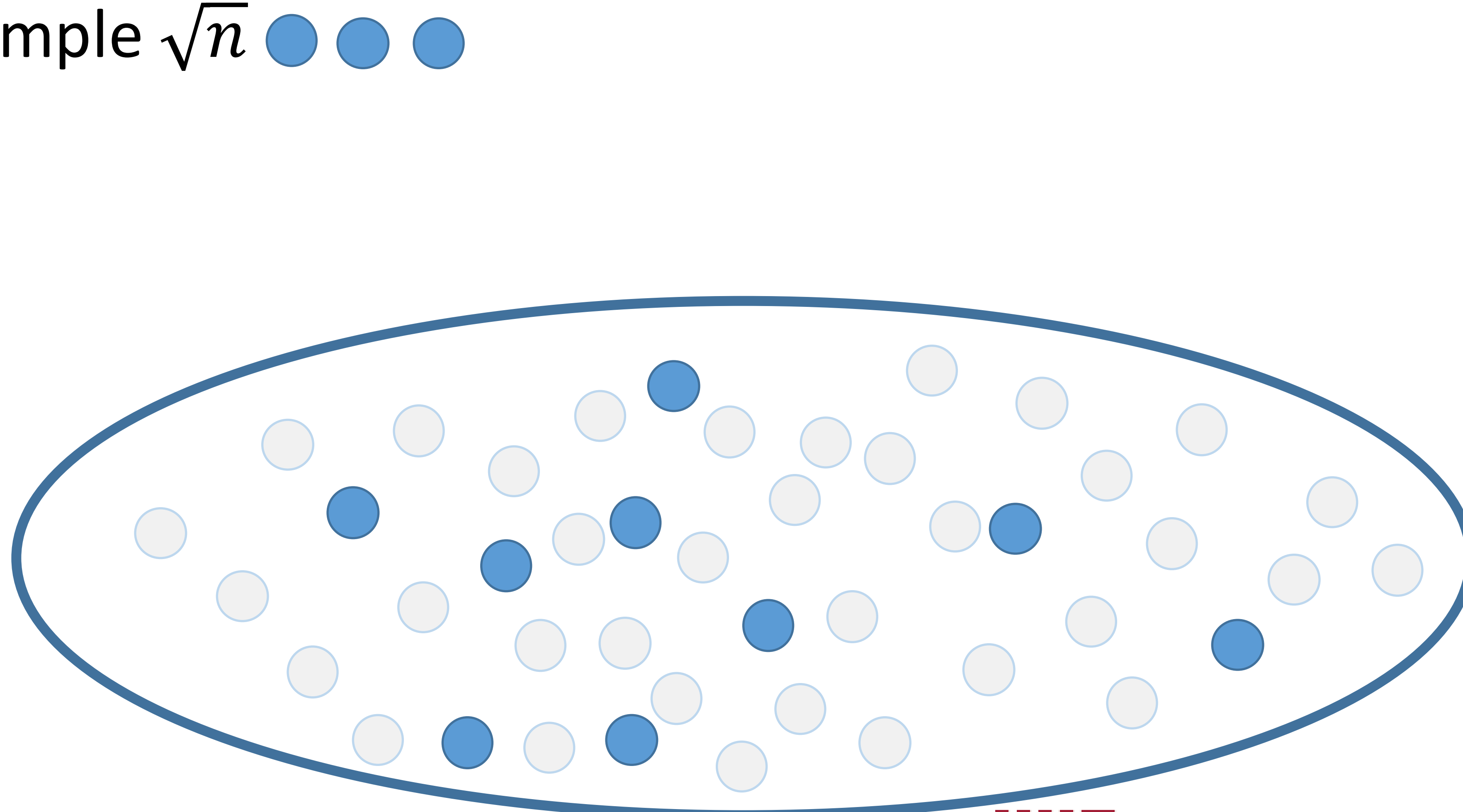
Maximal error: D/ε vs. D

3/2-approximating the Diameter in $O\left(\sqrt{n \log n} + D\right)$



3/2-approximating the Diameter in $O\left(\sqrt{n \log n} + D\right)$

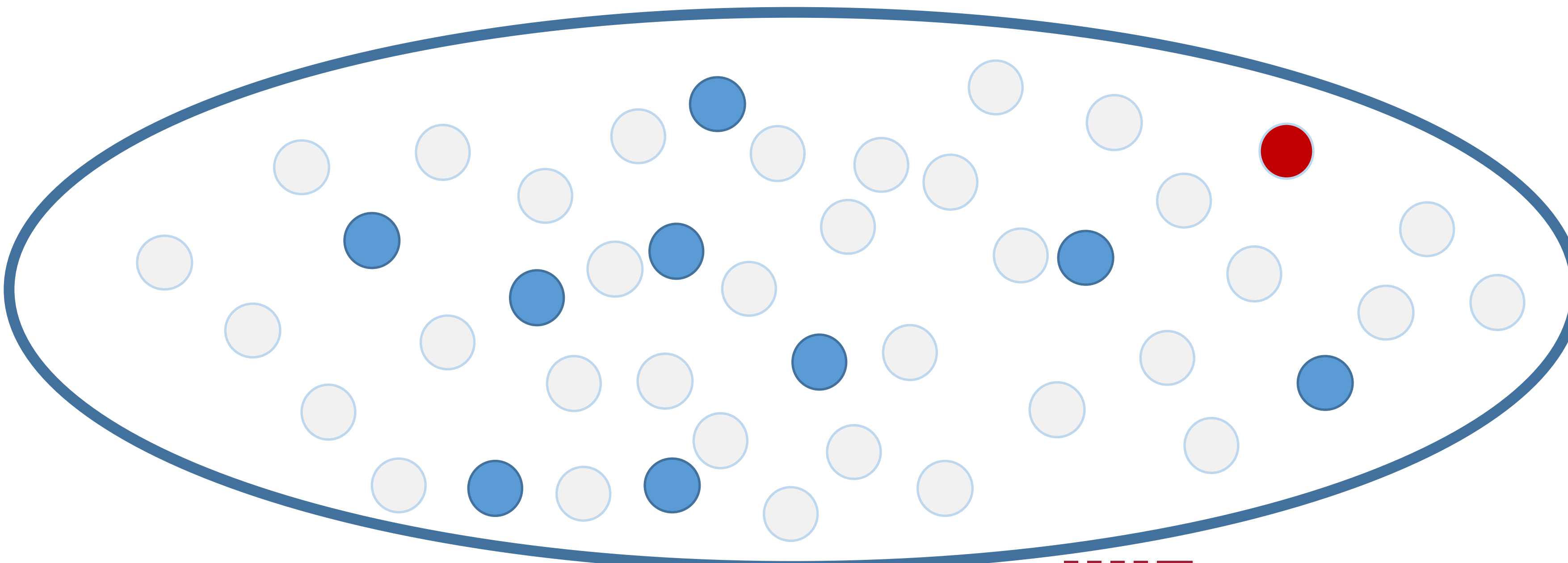
Sample $\sqrt{n}$



3/2-approximating the Diameter in $O\left(\sqrt{n \log n} + D\right)$

Sample $\sqrt{n}$ ● ● ●

● of largest distance to {● ● ●}

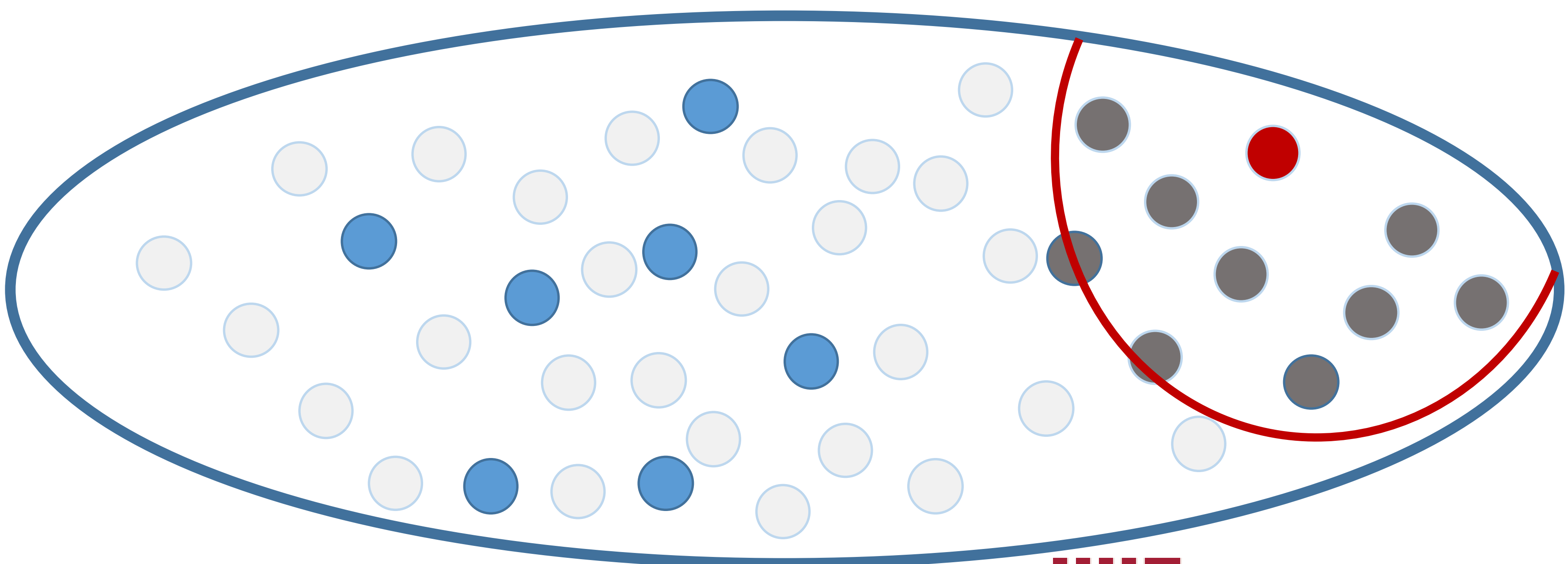


3/2-approximating the Diameter in $O(\sqrt{n \log n} + D)$

Sample $\sqrt{n}$ (blue nodes)

(red node) of largest distance to { (blue nodes) }

$\sqrt{n}$ closest (grey nodes) to (red node)



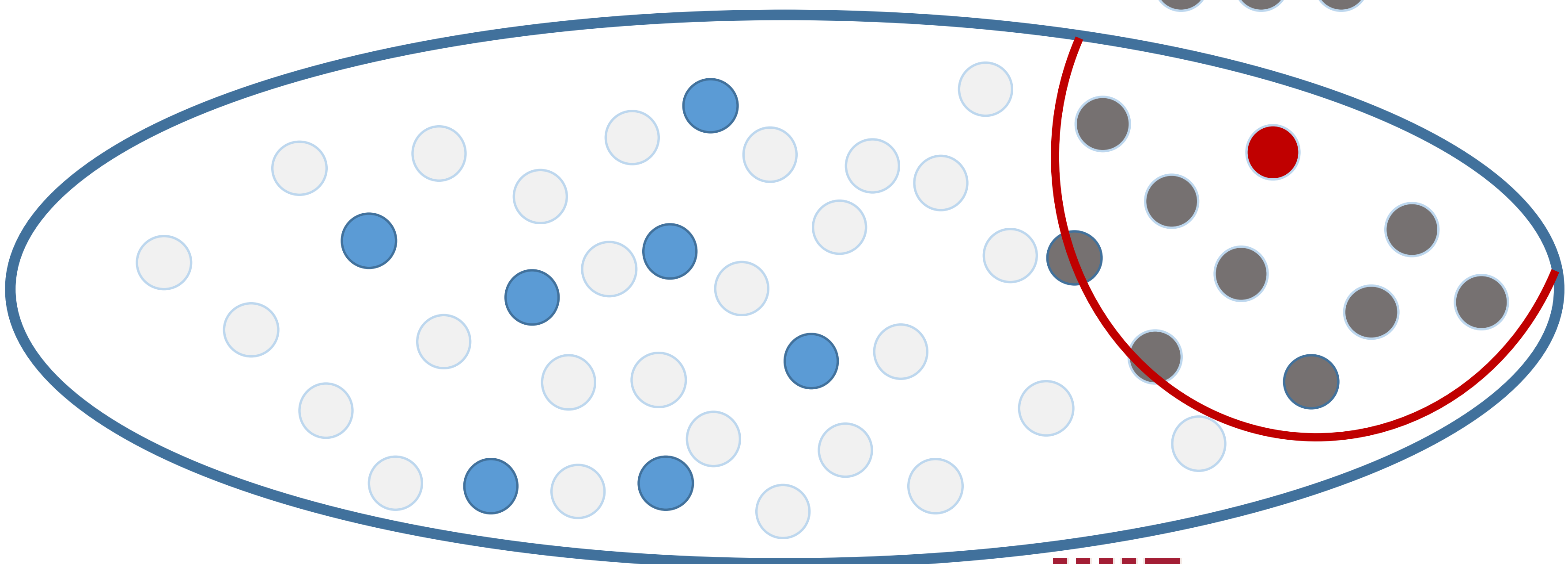
3/2-approximating the Diameter in $O\left(\sqrt{n \log n} + D\right)$

Sample $\sqrt{n}$

of largest distance to { }

$\sqrt{n}$ closest to

Compute BFS from each



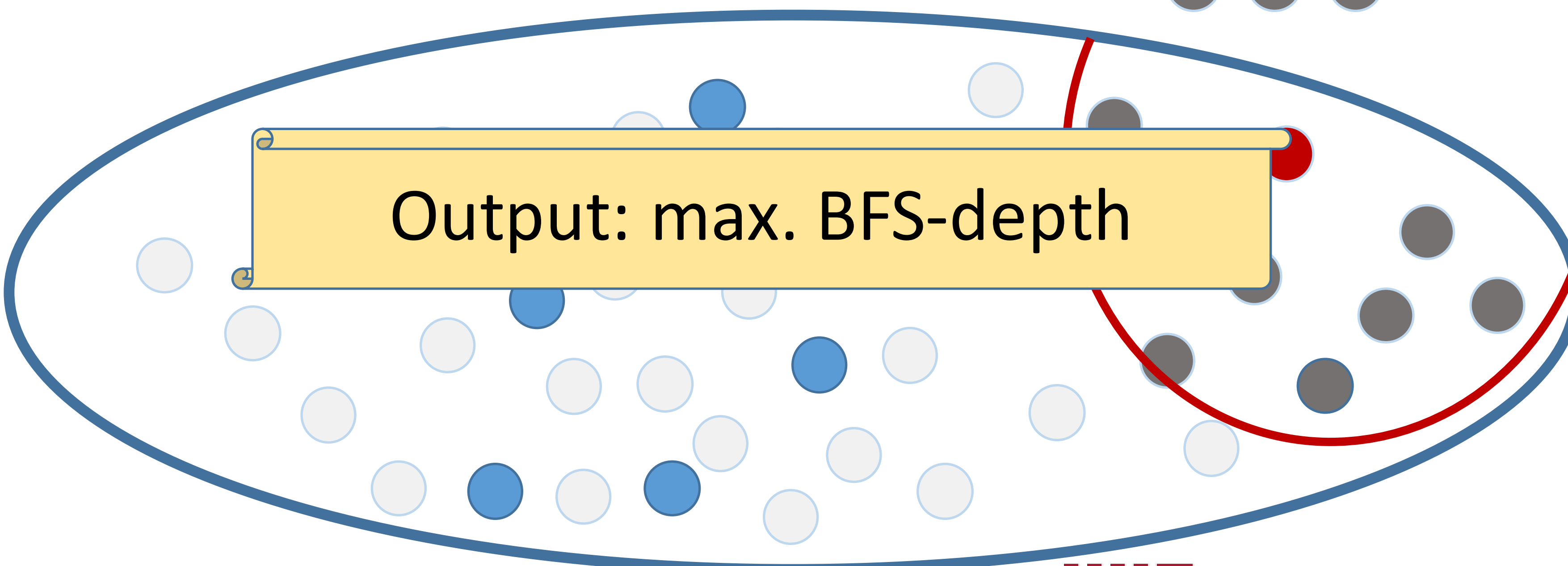
3/2-approximating the Diameter in $O(\sqrt{n \log n} + D)$

Sample $\sqrt{n}$

of largest distance to { }

$\sqrt{n}$ closest to

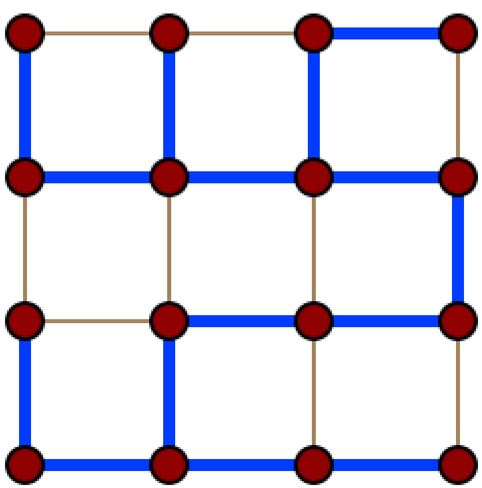
Compute BFS from each



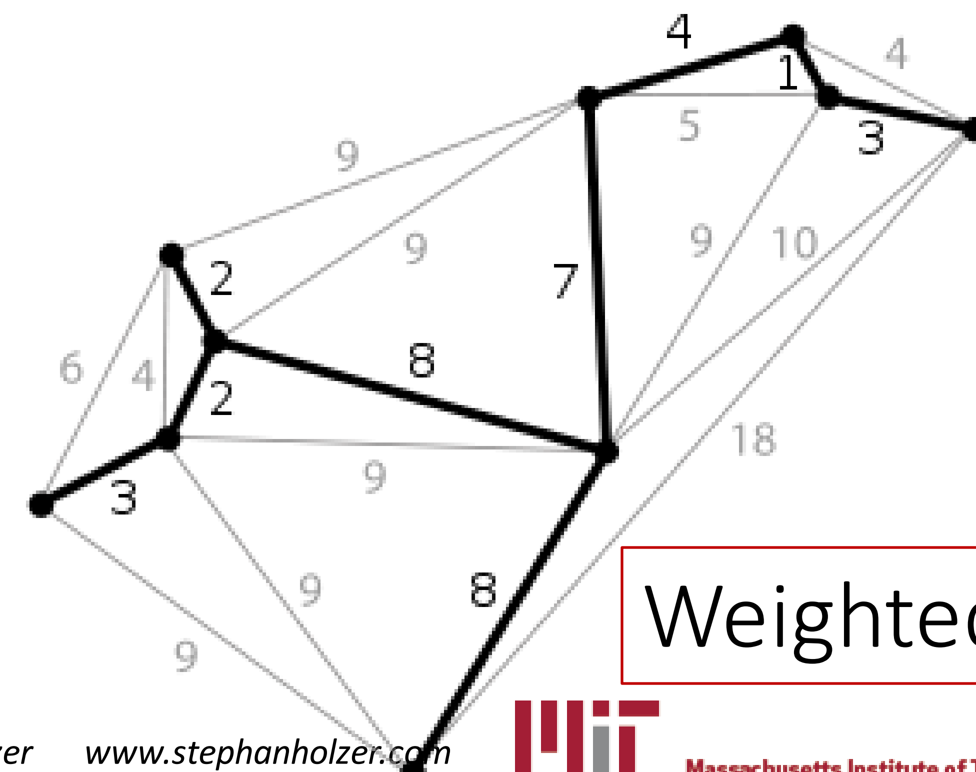
Distributed verification can be hard

(Minimum) Spanning Trees

Spanning tree:
Subgraph of a graph that includes all nodes and is a tree



Minimum spanning tree:
Spanning tree of minimal total edge weight



Weighted!

Distributed verification can be hard

Distributed Verification and Hardness of Distributed Approximation

Sequential world:

NP-complete problem SAT

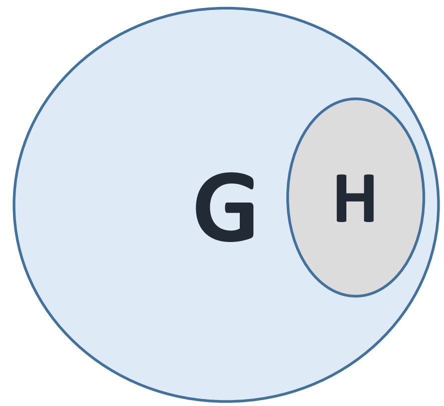
Solving: seems hard

Verifying assignment: easy

Sequential: Verification

ea

CONGEST world:



Verify: H spanning tree of G?

$\Omega(n^{1/2})$

f G

Distributed: Verification can be harder than computing

The

Time of Distributed MST-Algorithms

Problems	Upper bound	Lower bound
MST	$O(D + n^{1/2})$ [Garay, Kutten, Peleg FOCS'93]	$\Omega(D + n^{1/2})$ [Peleg, Rubinovich FOCS'99]

Slide by Danupon

Time of Distributed MST-Algorithms

Problems	Upper bound	Lower bound
MST	$O(D + n^{1/2})$ [Garay, Kutten, Peleg FOCS'93]	$\Omega(D + n^{1/2})$ [Peleg, Rubinovich FOCS'99]
α-approx. MST	OPEN	

α-approximation:

Let T be a MST of G and ω(T) its weight.

A spanning tree T` is an α-approximate MST if

$$\omega(T`) \leq \alpha\, \omega(T)$$

Time of Distributed MST-Algorithms

Problems	Upper bound	Lower bound
MST	$O(D + n^{1/2})$ [Garay, Kutten, Peleg FOCS'93]	$\Omega(D + n^{1/2})$ [Peleg, Rubinovich FOCS'99]
α-approx. MST	OPEN	$\Omega(D + (n/\alpha)^{1/2})$ [Elkin STOC'04]

α-approximation:

Let T be a MST of G and ω(T) its weight.

A spanning tree T\` is an α-approximate MST if

$$\omega(T`) \leq \alpha\, \omega(T)$$

Time of Distributed MST-Algorithms

Problems	Upper bound	Lower bound
MST	$O(D + n^{1/2})$ [Garay, Kutten, Peleg FOCS'93]	$\Omega(D + n^{1/2})$ [Peleg, Rubinovich FOCS'99]
α-approx. MST	OPEN	$\Omega(D + (n/\alpha)^{1/2})$ [Elkin STOC'04]
ST Verification		

Time of Distributed MST-Algorithms

Problems	Upper bound	Lower bound
MST	$O(D + n^{1/2})$ [Garay, Kutten, Peleg FOCS'93]	$\Omega(D + n^{1/2})$ [Peleg, Rubinovich FOCS'99]
α-approx. MST	OPEN	$\Omega(D + (n/\alpha)^{1/2})$ [Elkin STOC'04]
ST Verification	$O(D + n^{1/2})$	

Time of Distributed MST-Algorithms

Problems	Upper bound	Lower bound
MST	$O(D + n^{1/2})$ [Garay, Kutten, Peleg FOCS'93]	$\Omega(D + n^{1/2})$ [Peleg, Rubinovich FOCS'99]
α-approx. MST	~~OPEN~~	$\Omega(D + (n / \alpha)^{1/2})$ [Elkin STOC'04]
ST Verification	$O(D + n^{1/2})$	$\Omega (D + n^{1/2})$

Time of Distributed MST-Algorithms

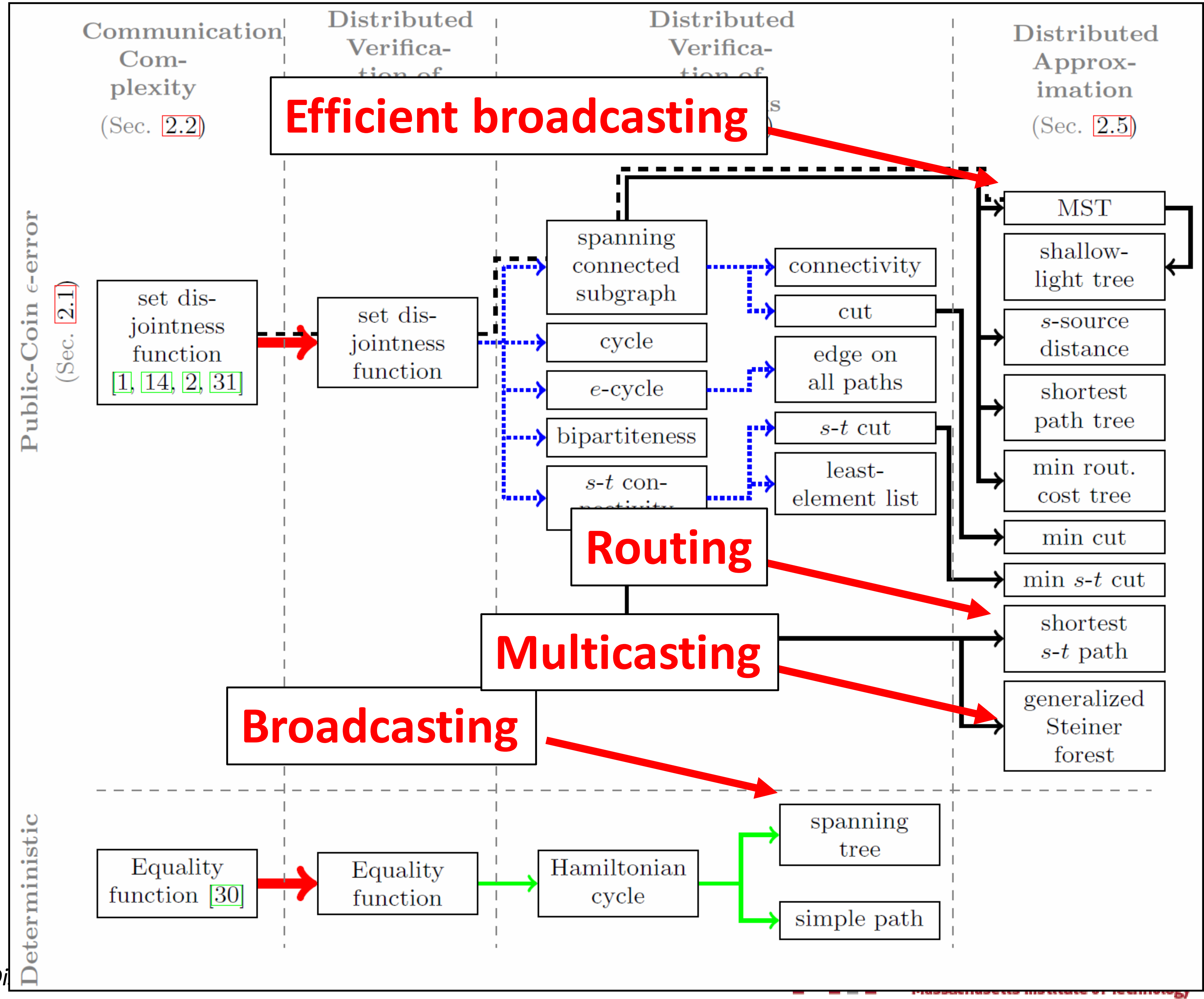
Problems	Upper bound	Lower bound
MST	$O(D + n^{1/2})$ [Garay, Kutten, Peleg FOCS'93]	$\Omega(D + n^{1/2})$ [Peleg, Rubinovich FOCS'99]
α-approx. MST	~~HOPELESS ☹~~	$\Omega(D + (n /\alpha)^{1/2})$ [Elkin STOC'04]
ST Verification	$O(D + n^{1/2})$	$\Omega(D + n^{1/2})$

Time of Distributed MST-Algorithms

Problems	Upper bound	Lower bound
MST	$O(D + n^{1/2})$ [Garay, Kutten, Peleg FOCS'93]	$\Omega(D + n^{1/2})$ [Peleg, Rubinovich FOCS'99]
α-approx. MST	~~HOPELESS ☹~~	$\Omega(D + (n/\alpha)^{1/2})$ [Elkin STOC'04]
ST Verification	$O(D + n^{1/2})$	$\Omega(D + n^{1/2})$

King, Kutten, Thorup PODC'15:

Message Complexity $o(m)$



Communication Complexity (Sec. 2.2)
Distributed Verification of
Distributed Verification of
s
Distributed Approximation (Sec. 2.5)
Efficient broadcasting
Public-Coin ε-error (Sec. 2.1)
set disjointness function [1, 14, 2, 31]
set disjointness function
spanning connected subgraph
cycle
e-cycle
bipartiteness
s-t connectivity
connectivity
cut
edge on all paths
s-t cut
least-element list
MST
shallow-light tree
s-source distance
shortest path tree
min rout. cost tree
min cut
min s-t cut
shortest s-t path
generalized Steiner forest
Routing
Multicasting
Broadcasting
Deterministic
Equality function [30]
Equality function
Hamiltonian cycle
spanning tree
simple path

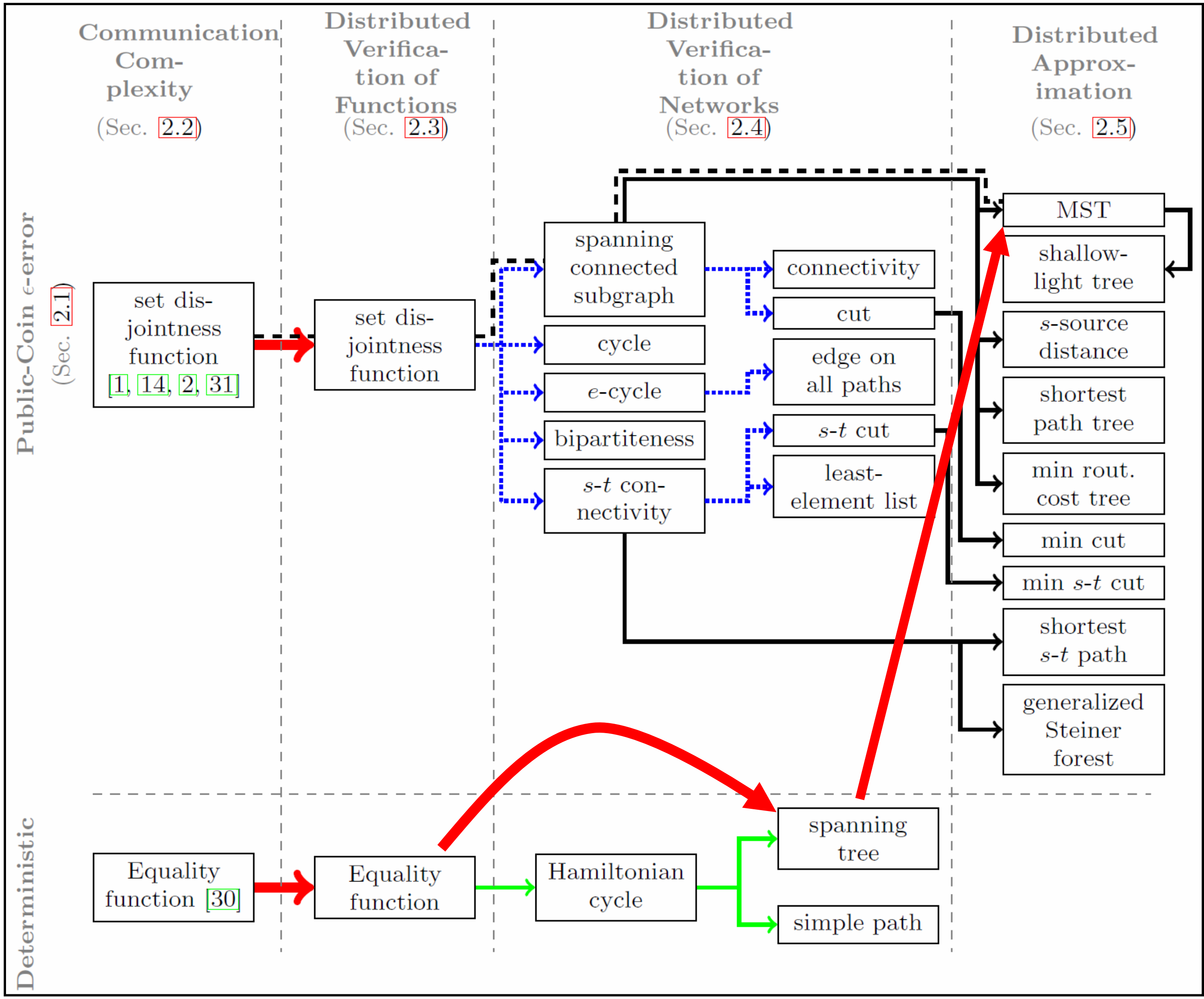
General technique for lower bounds

Connects communication complexity to distributed comp.

Connects verification to approximation

Many bounds tight

Systematic study of distributed verification



Distributed algorithms for the above problems require

$\Omega(n^{1/2}+D)$ time

Three steps of reduction

Communication Complexity

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

Well-known result in communication complexity

simulation theorem

Distributed Algorithms

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

Similar to lower bounds of graph streaming algorithms

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

Similar to hardness of TSP

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

Slide by Danupon

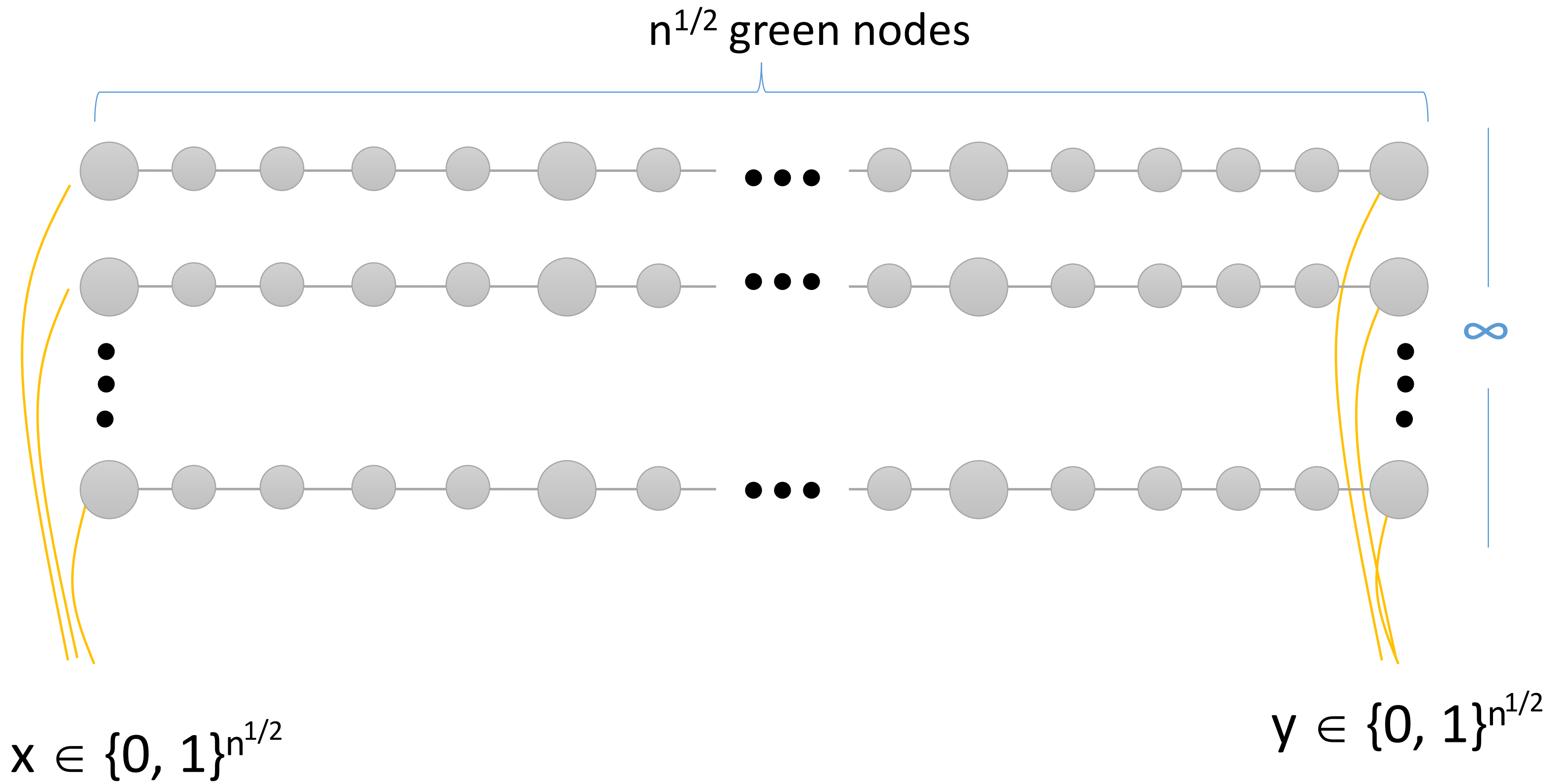
Communication complexity of EQUALITY

x=y?

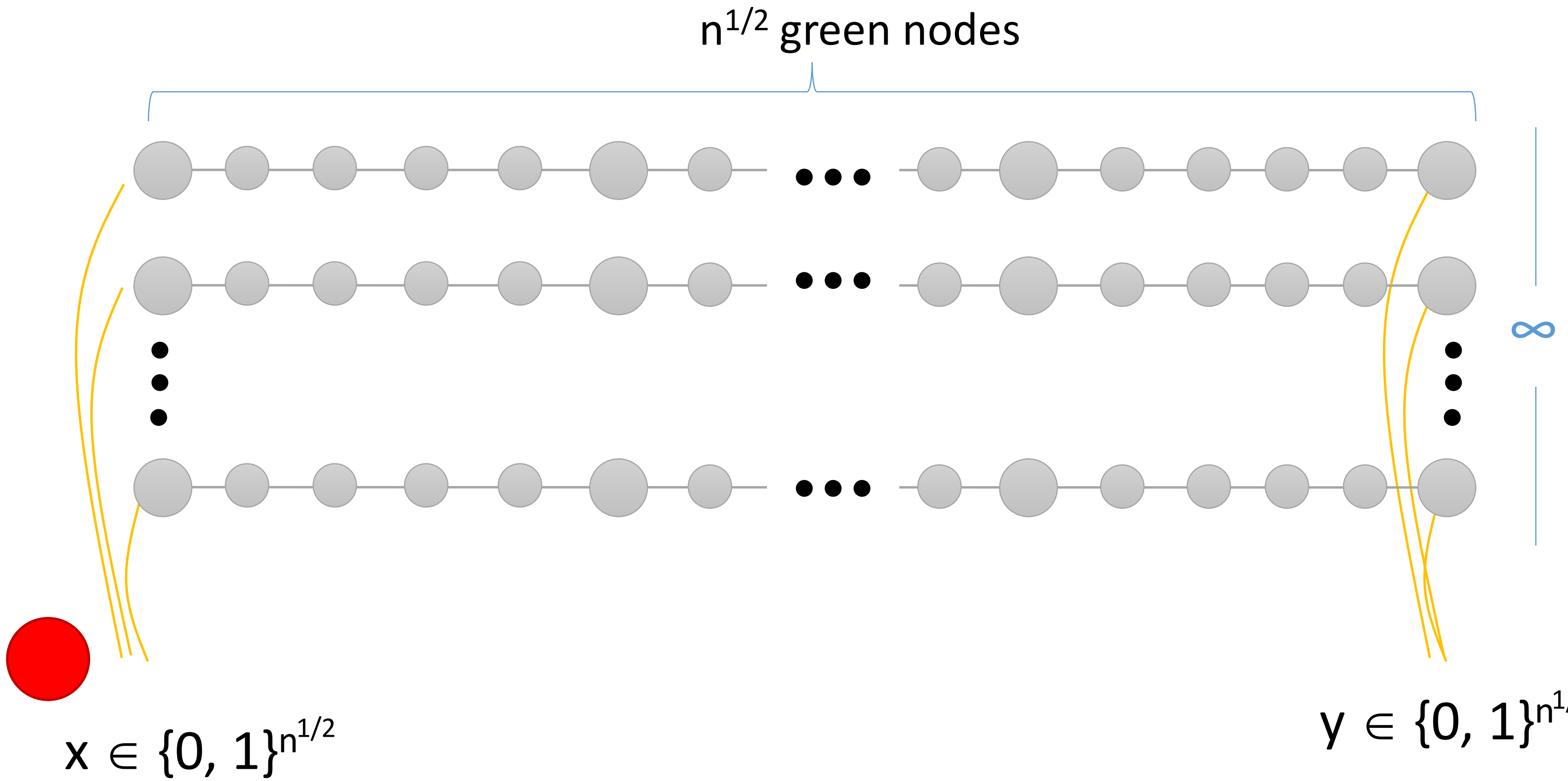
$x \in \{0, 1\}^k$ **Deterministic: $\Omega(k)$** $y \in \{0, 1\}^k$

Distributed time complexity of EQUALITY

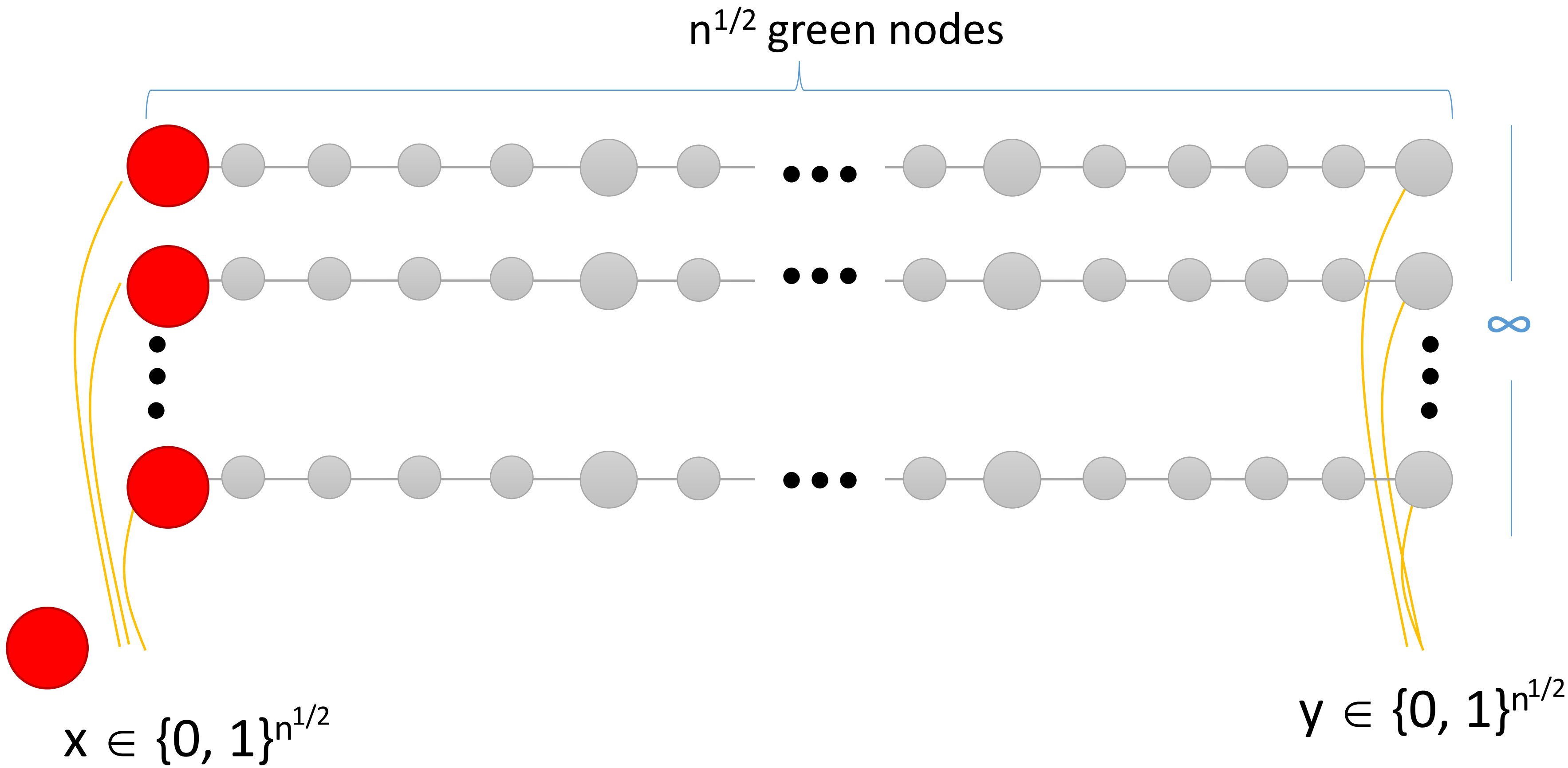
Alice and Bob are connected by many paths of length $n^{1/2}$



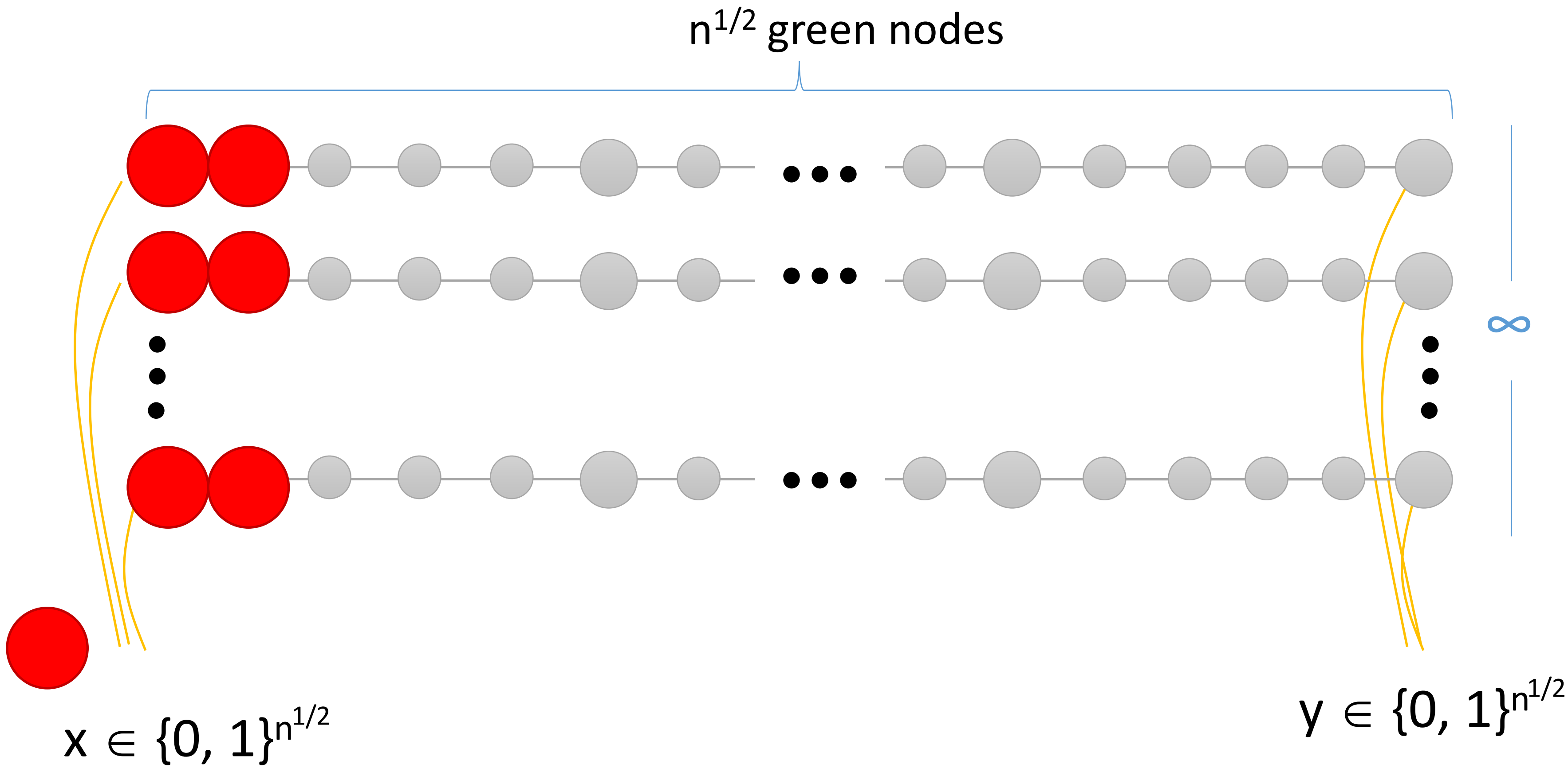
Alice and Bob are connected by many paths of length $n^{1/2}$



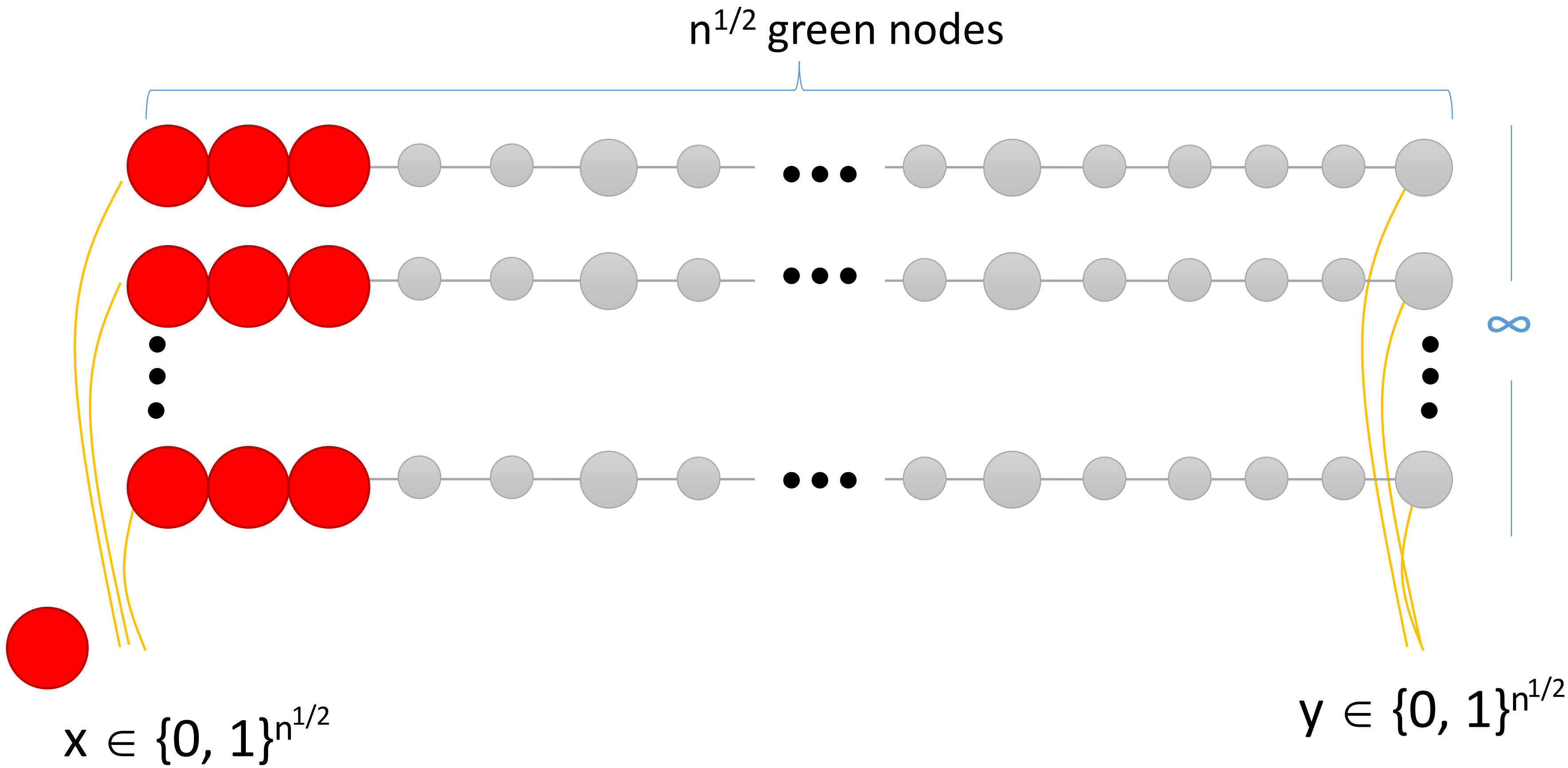
Alice and Bob are connected by many paths of length $n^{1/2}$



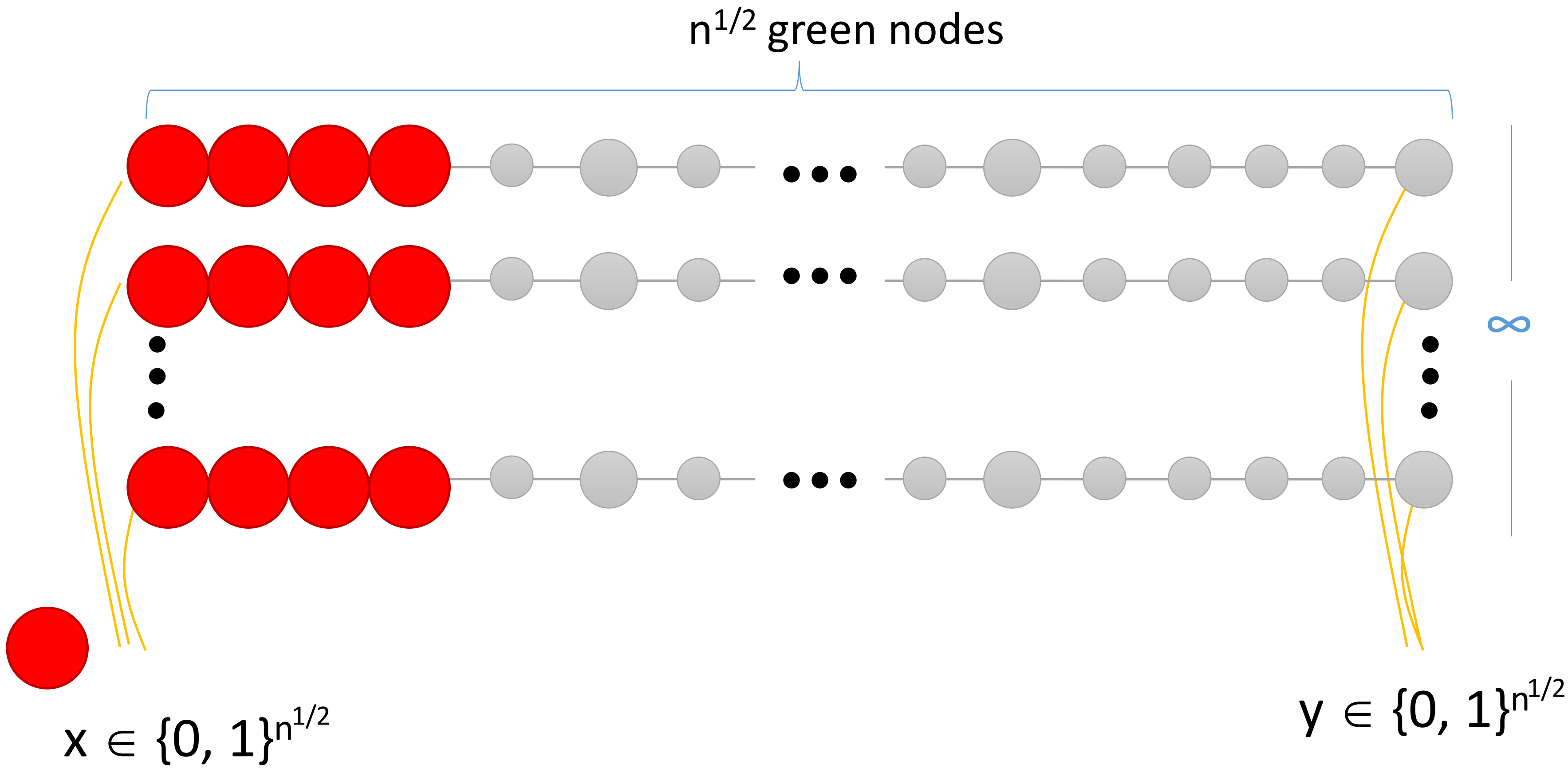
Alice and Bob are connected by many paths of length $n^{1/2}$



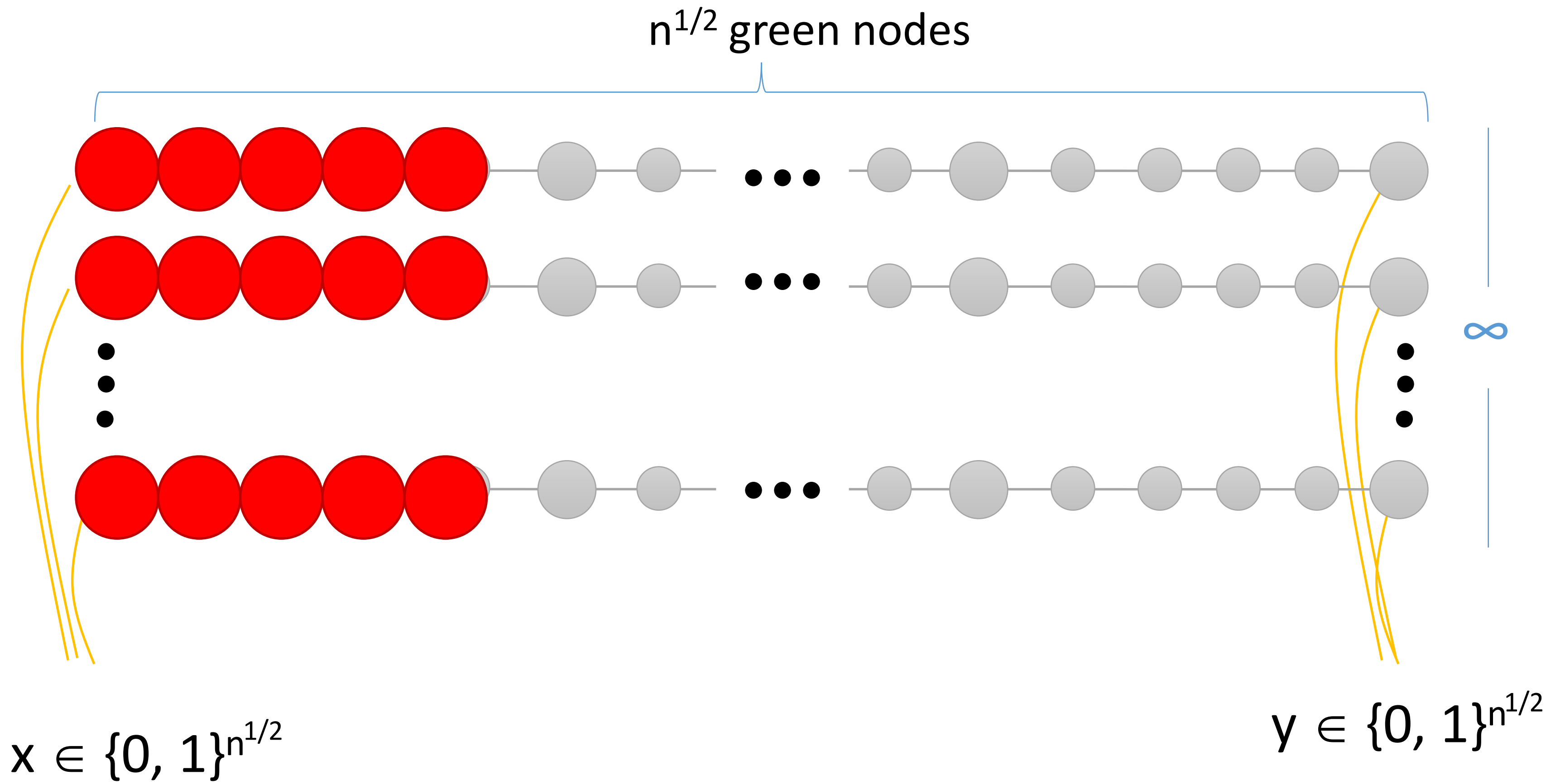
Alice and Bob are connected by many paths of length $n^{1/2}$



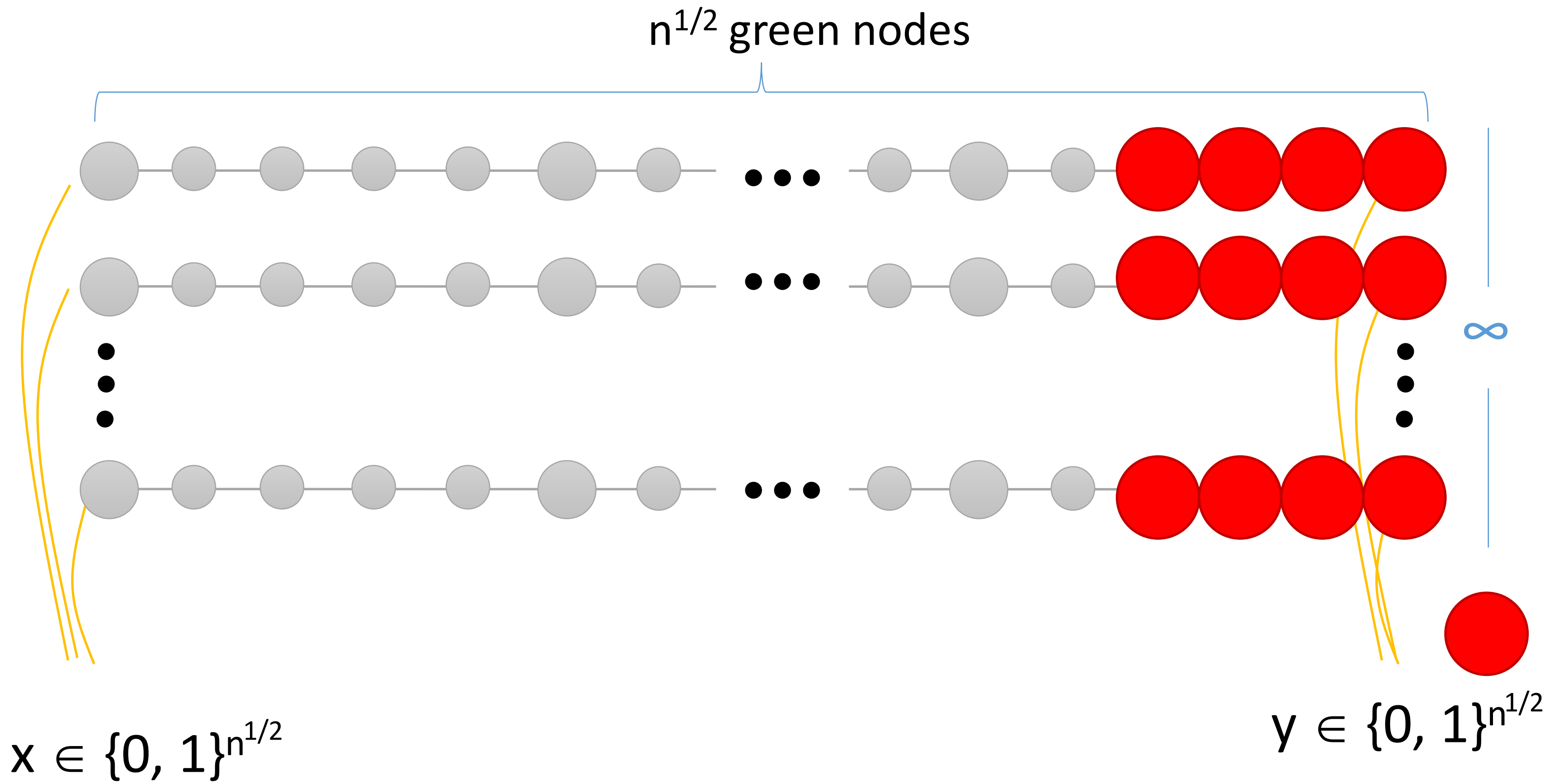
Alice and Bob are connected by many paths of length $n^{1/2}$



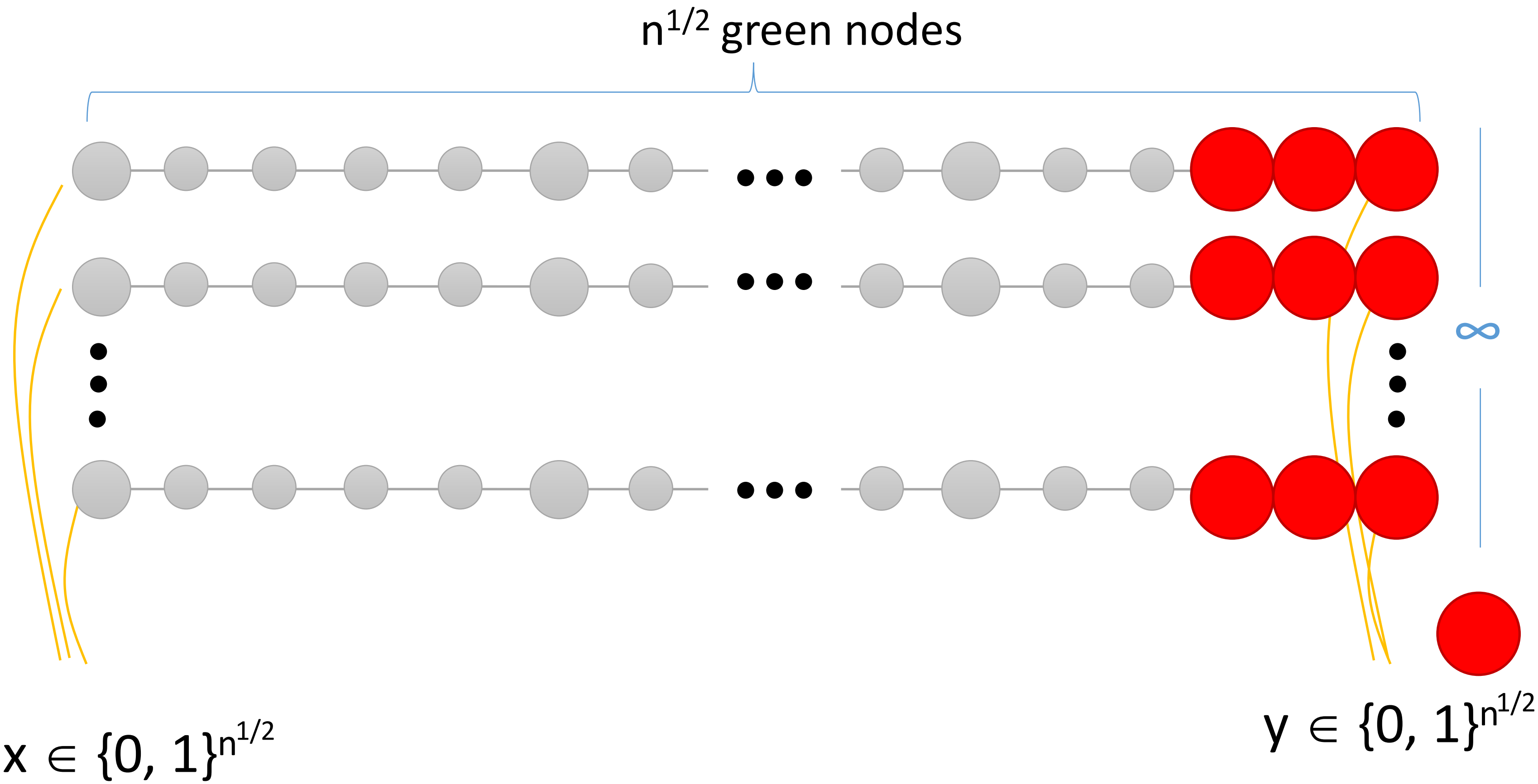
Alice and Bob are connected by many paths of length $n^{1/2}$



Alice and Bob are connected by many paths of length $n^{1/2}$

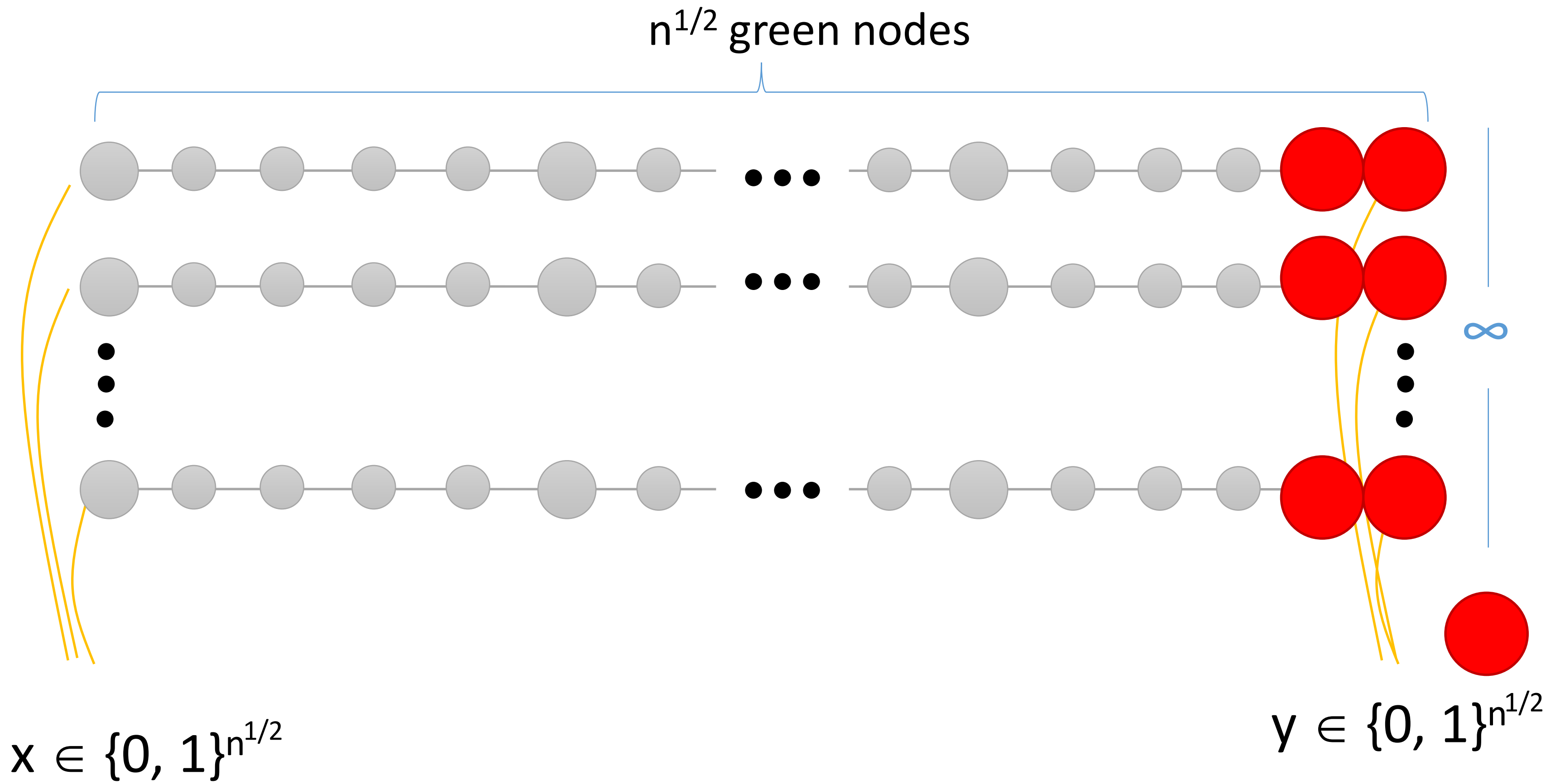


Alice and Bob are connected by many paths of length $n^{1/2}$

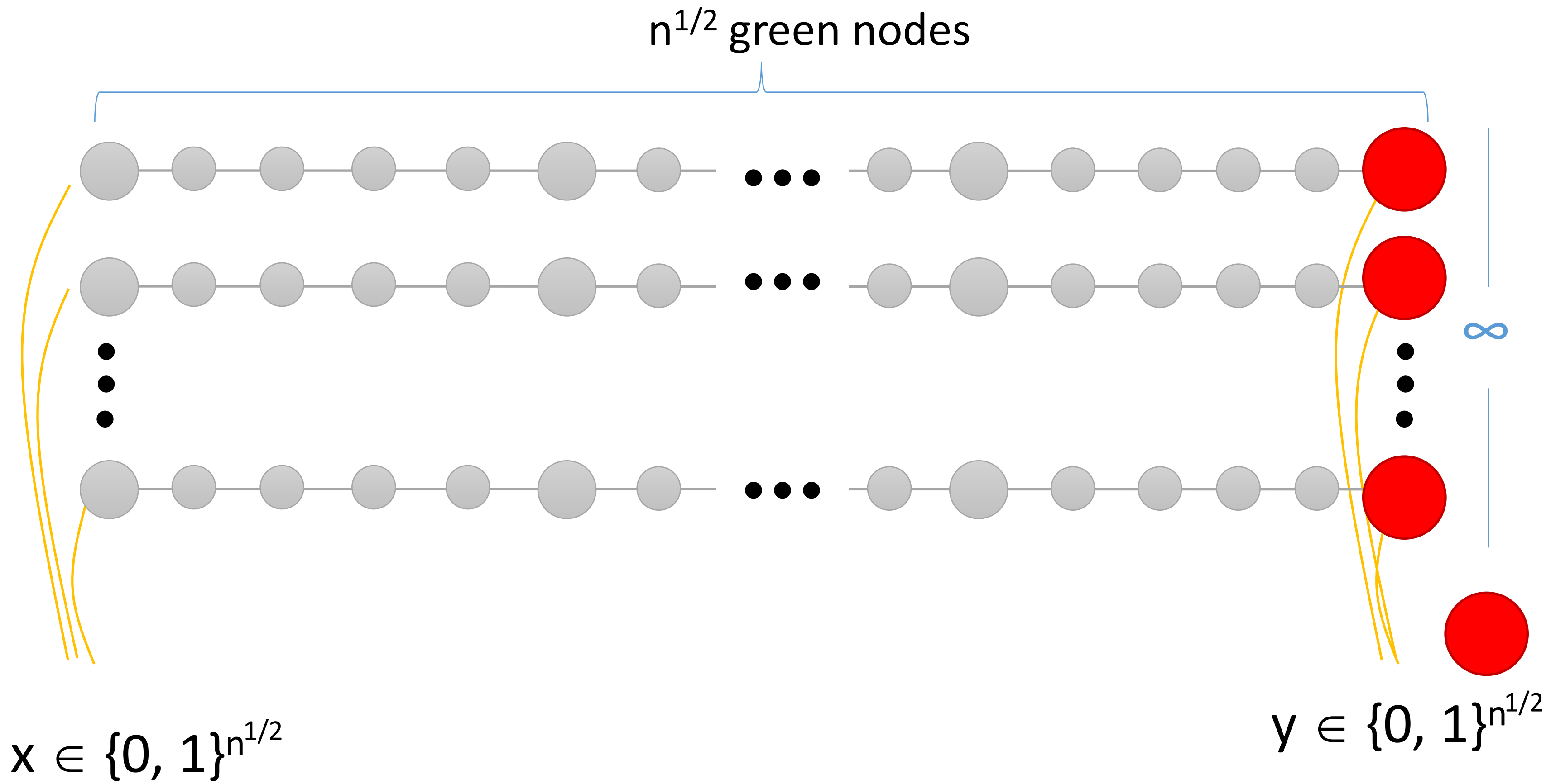


Slide by Danupon

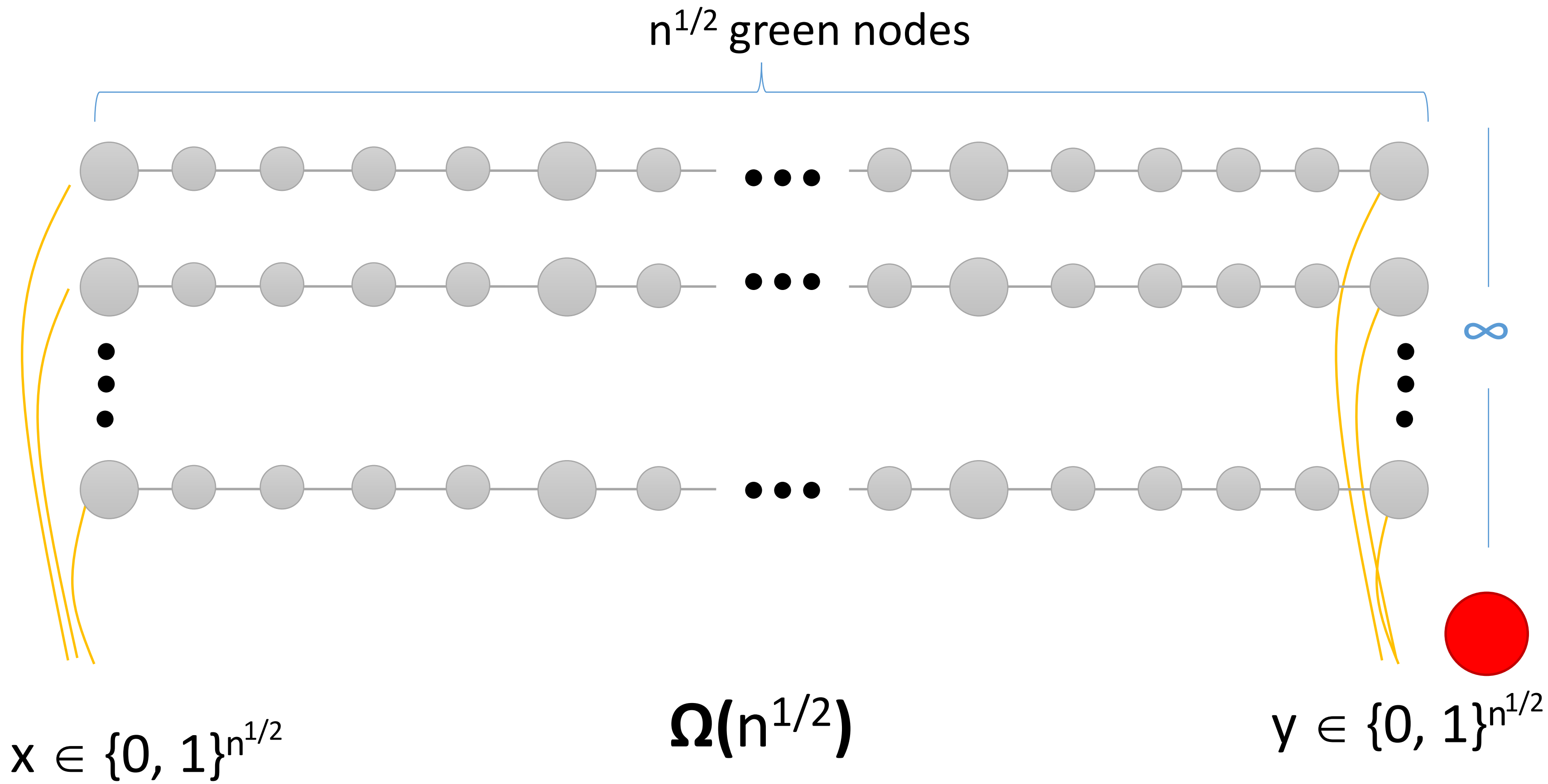
Alice and Bob are connected by many paths of length $n^{1/2}$



Alice and Bob are connected by many paths of length $n^{1/2}$

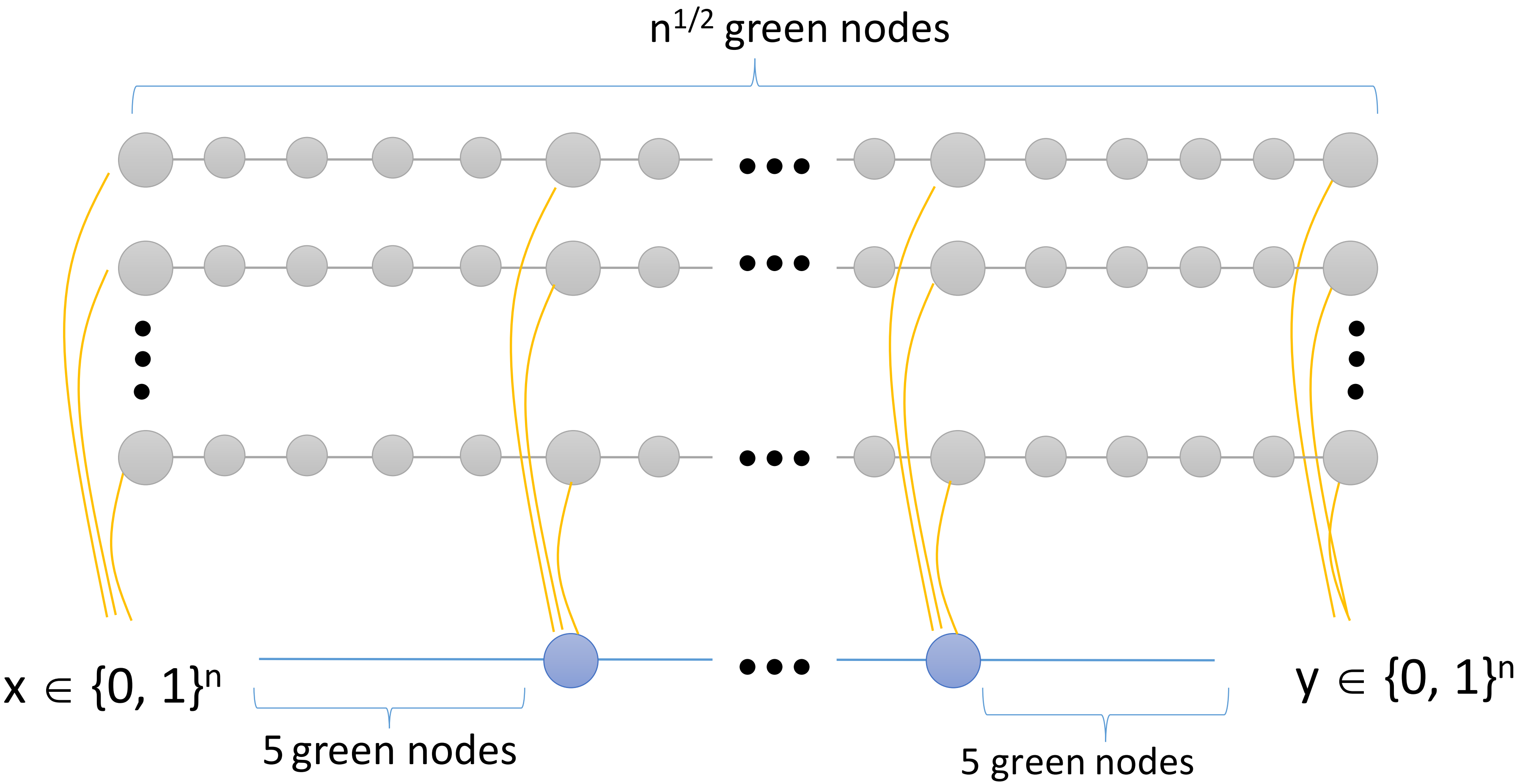


Alice and Bob are connected by many paths of length $n^{1/2}$

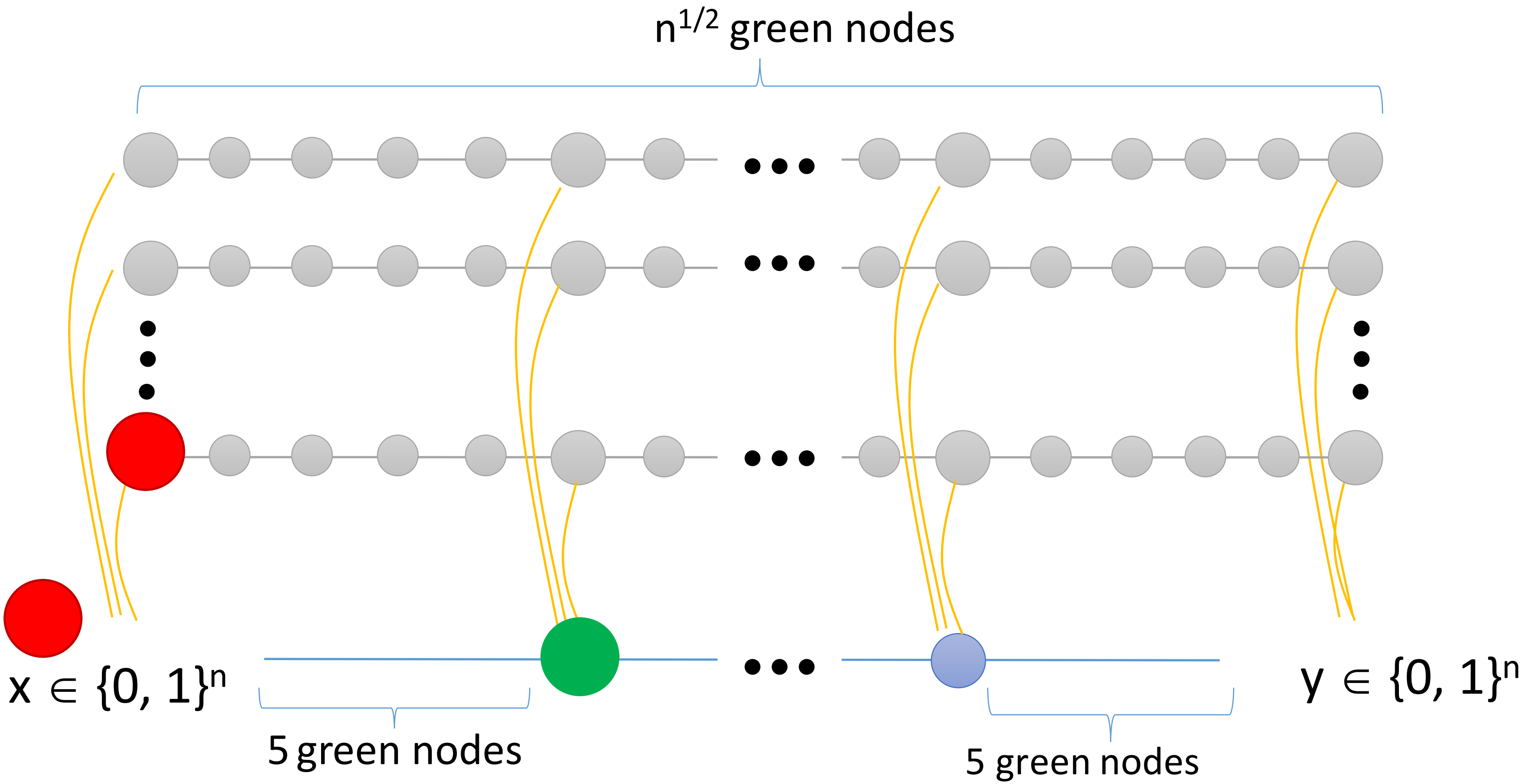


Make the diameter smaller

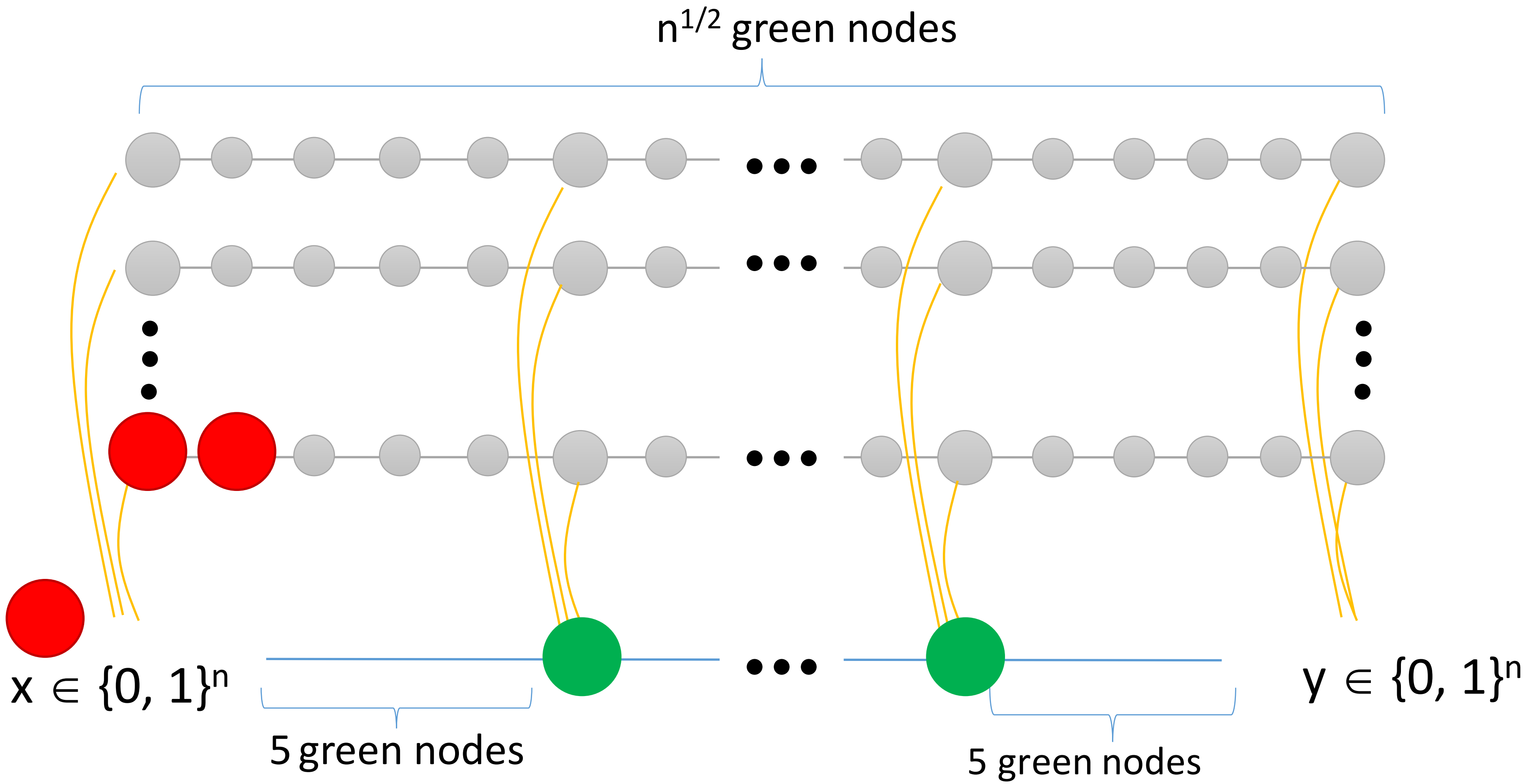
Now the diameter is $n^{1/2}/5$
How many steps do we need?



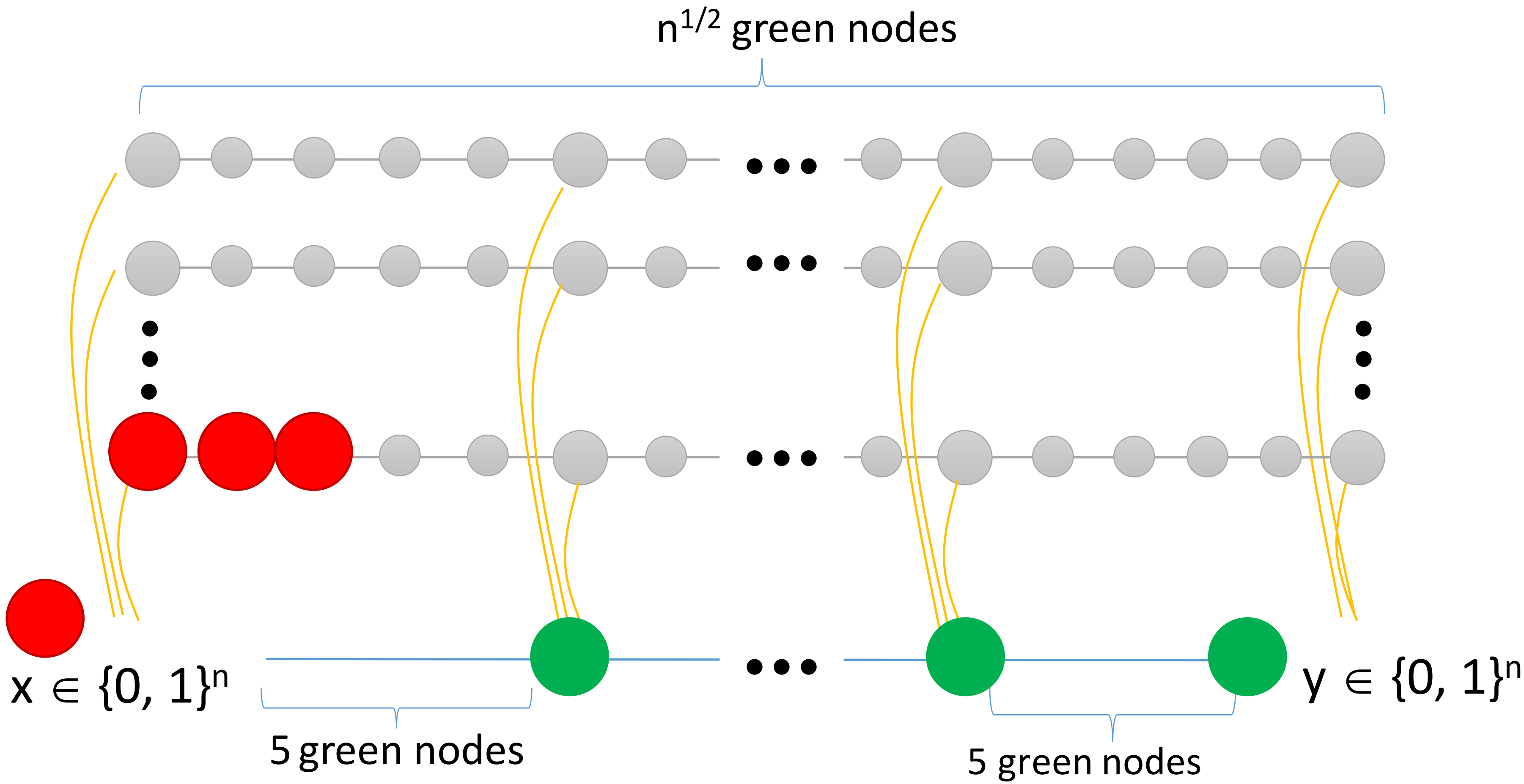
Now the diameter is $n^{1/2}/5$ How many steps do we need?



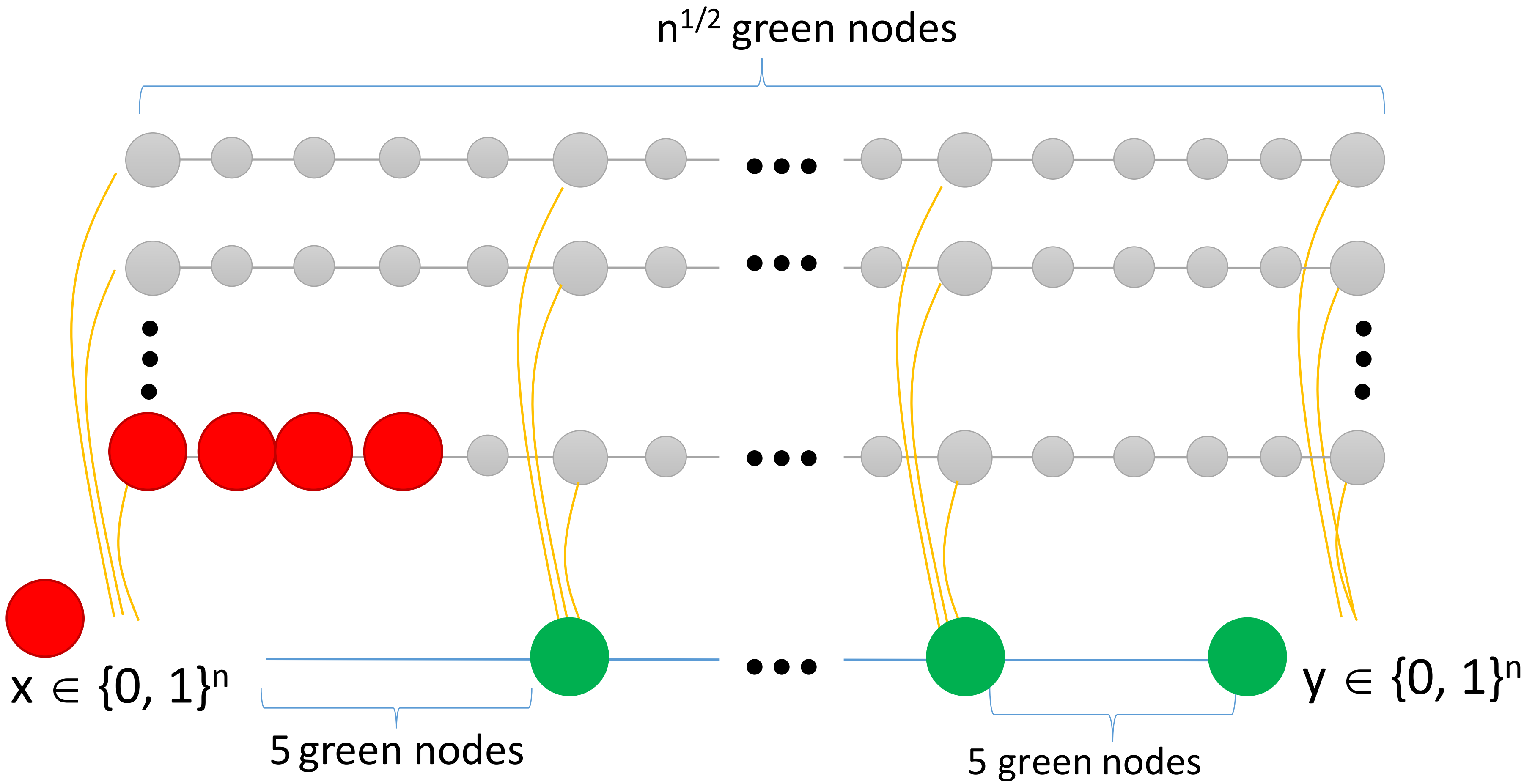
Now the diameter is $n^{1/2}/5$
How many steps do we need?



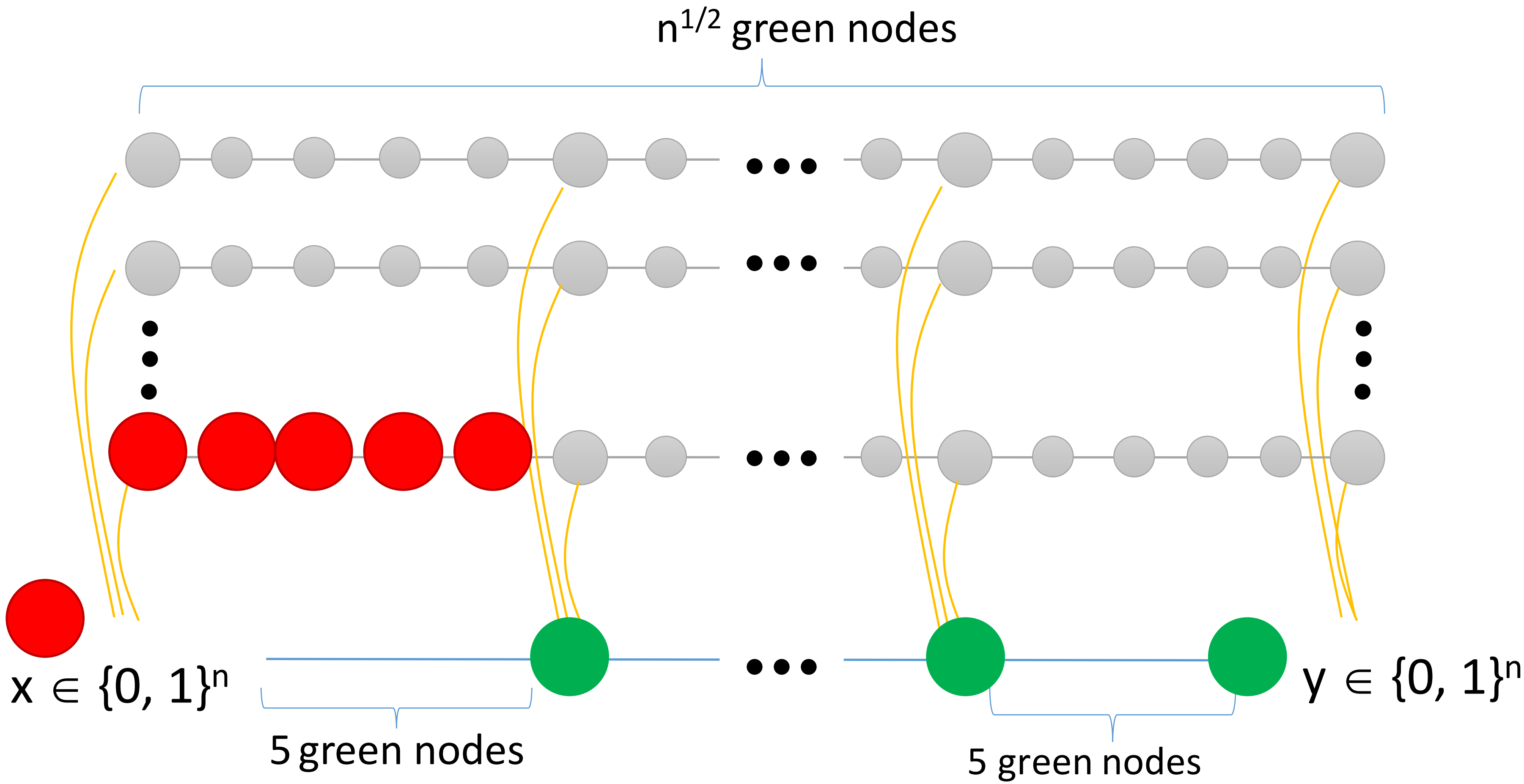
Now the diameter is $n^{1/2}/5$
How many steps do we need?



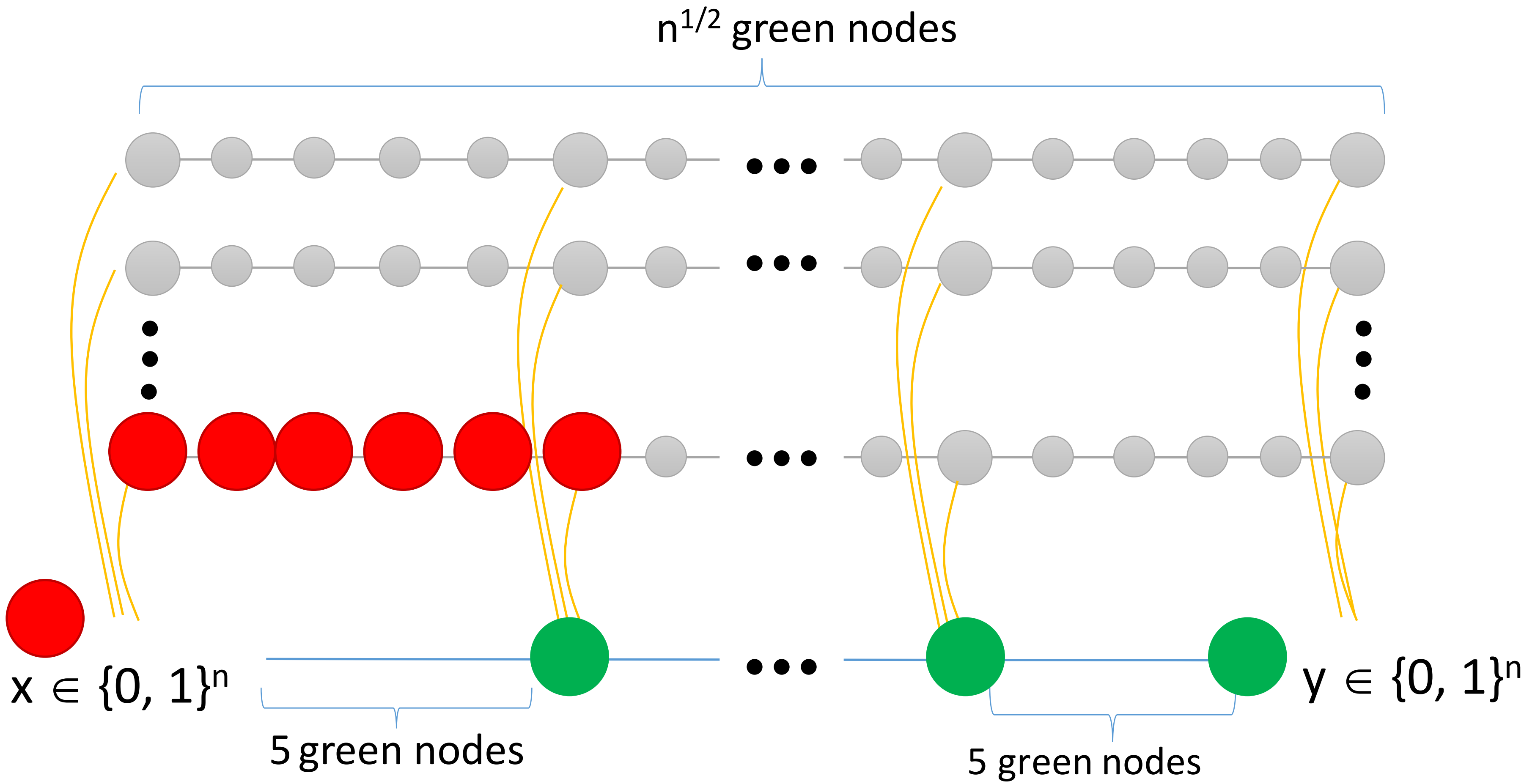
Now the diameter is $n^{1/2}/5$
How many steps do we need?



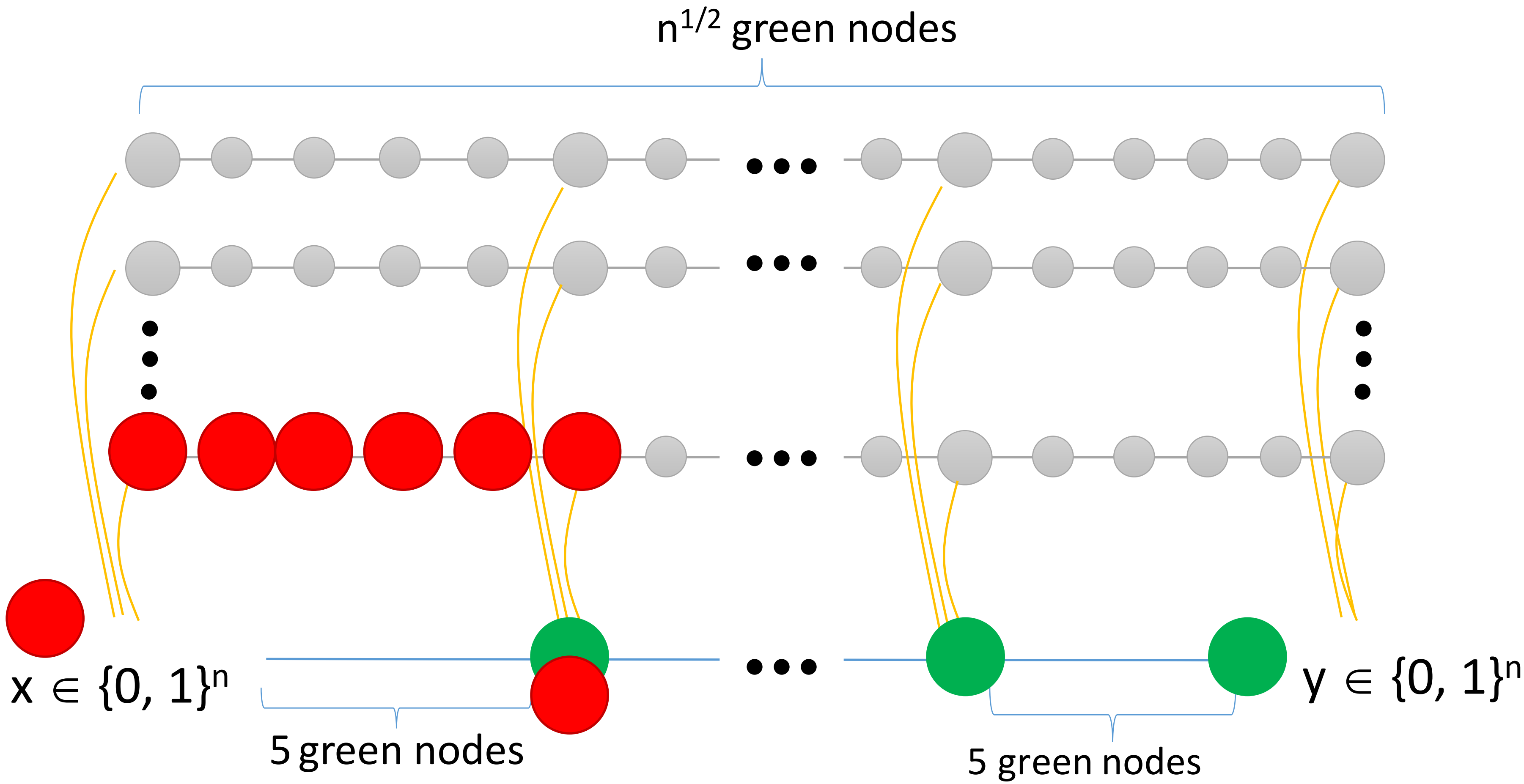
Now the diameter is $n^{1/2}/5$ How many steps do we need?



Now the diameter is $n^{1/2}/5$ How many steps do we need?



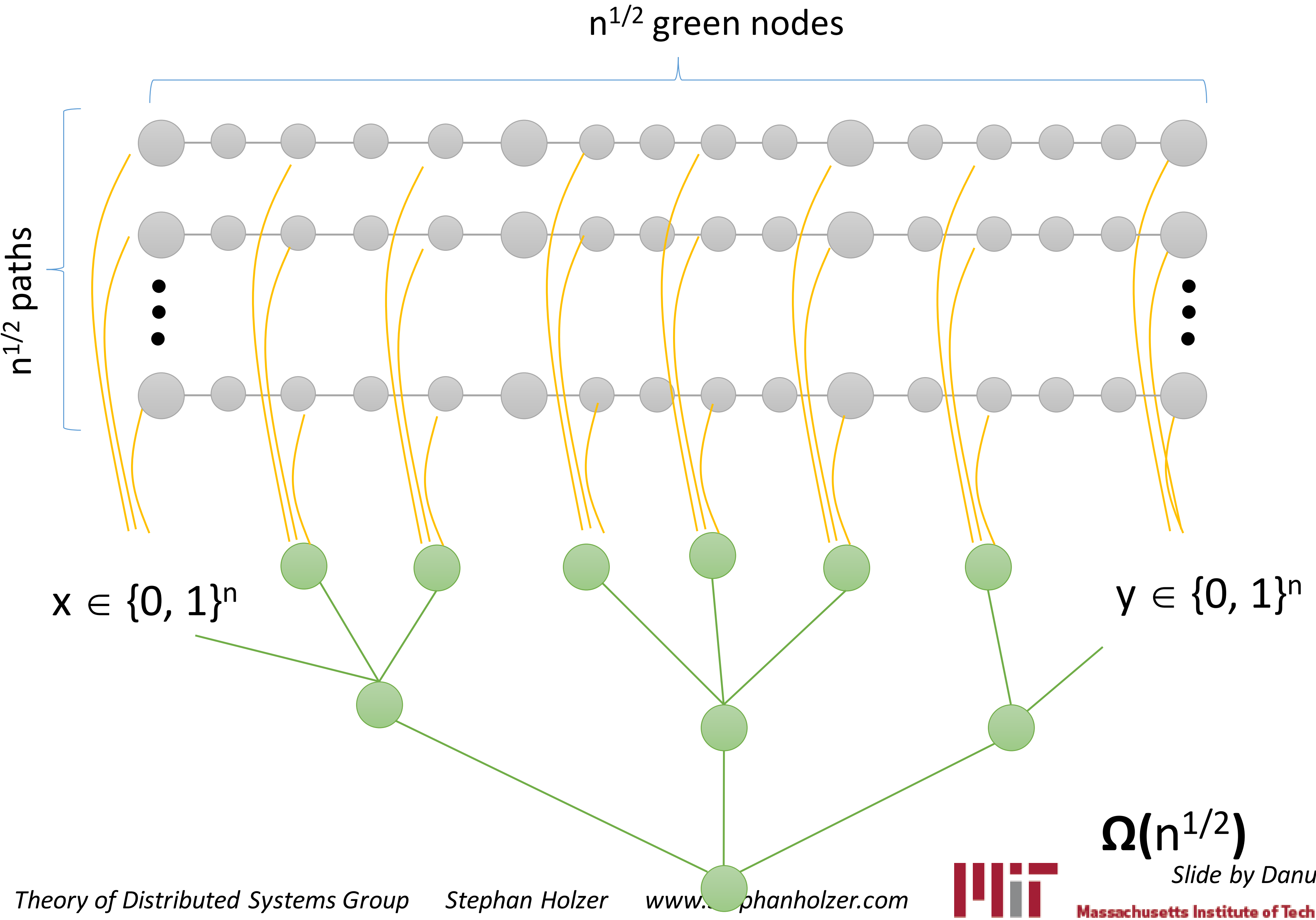
Now the diameter is $n^{1/2}/5$
How many steps do we need?



No high speedup

Reduce diameter ...

Diameter = log n



Ω(n^{1/2})

Slide by Danupon

Three steps of reduction

Communication Complexity

Distributed Algorithms

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

→ (simulation theorem) →

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

Well-known result in communication complexity

simulation theorem

Similar to lower bounds of graph streaming algorithms

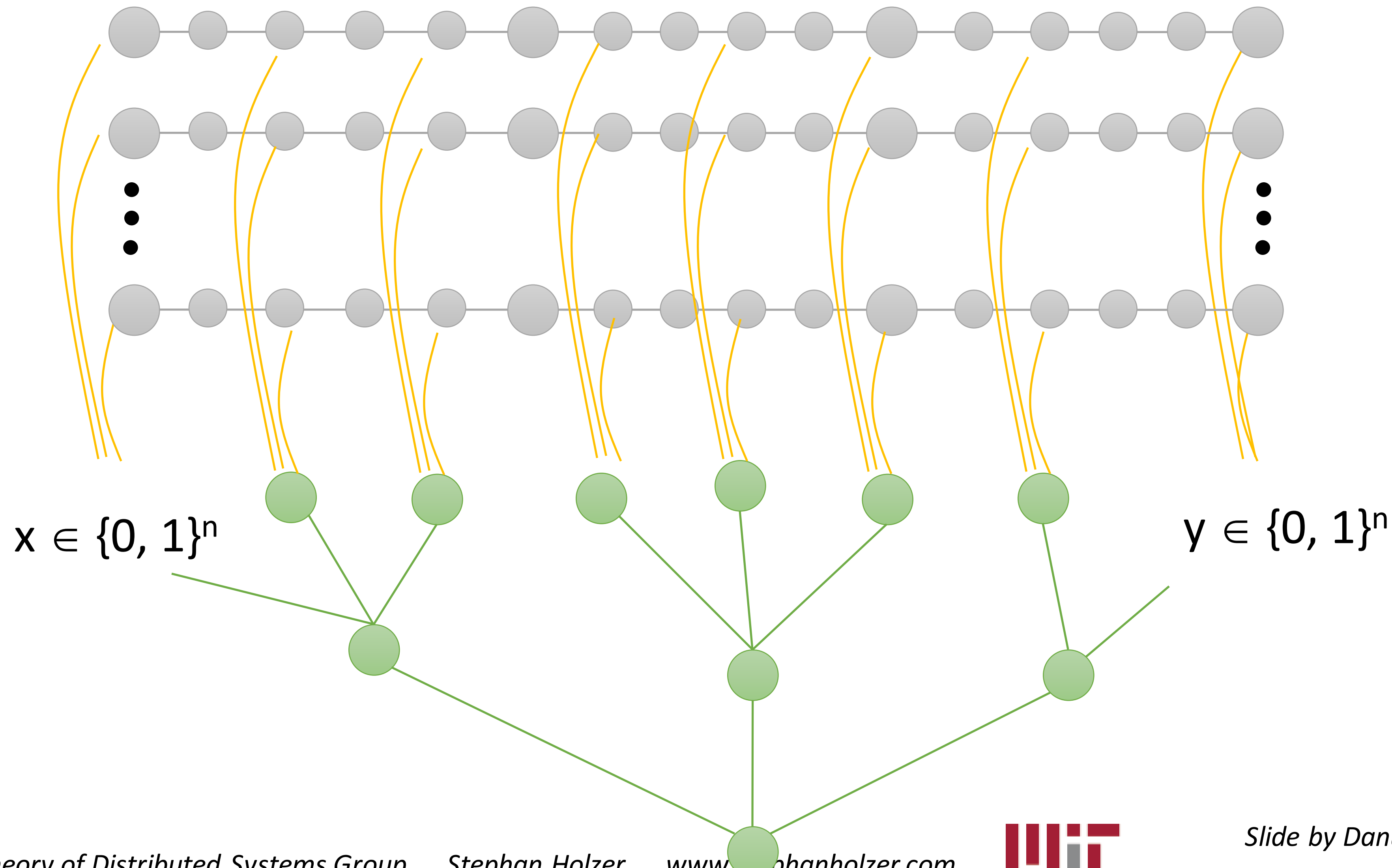
↓

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

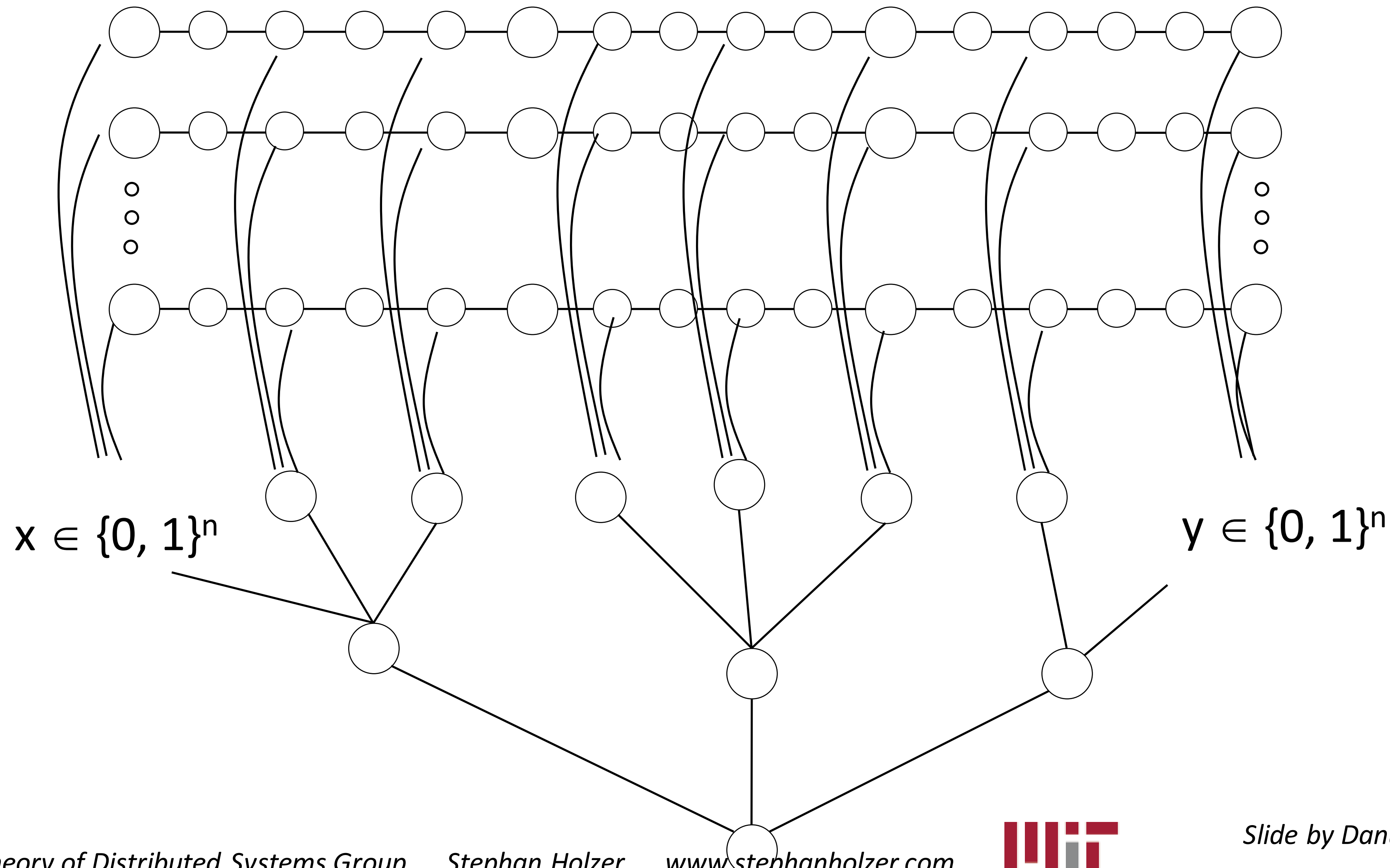
Similar to hardness of TSP

↓

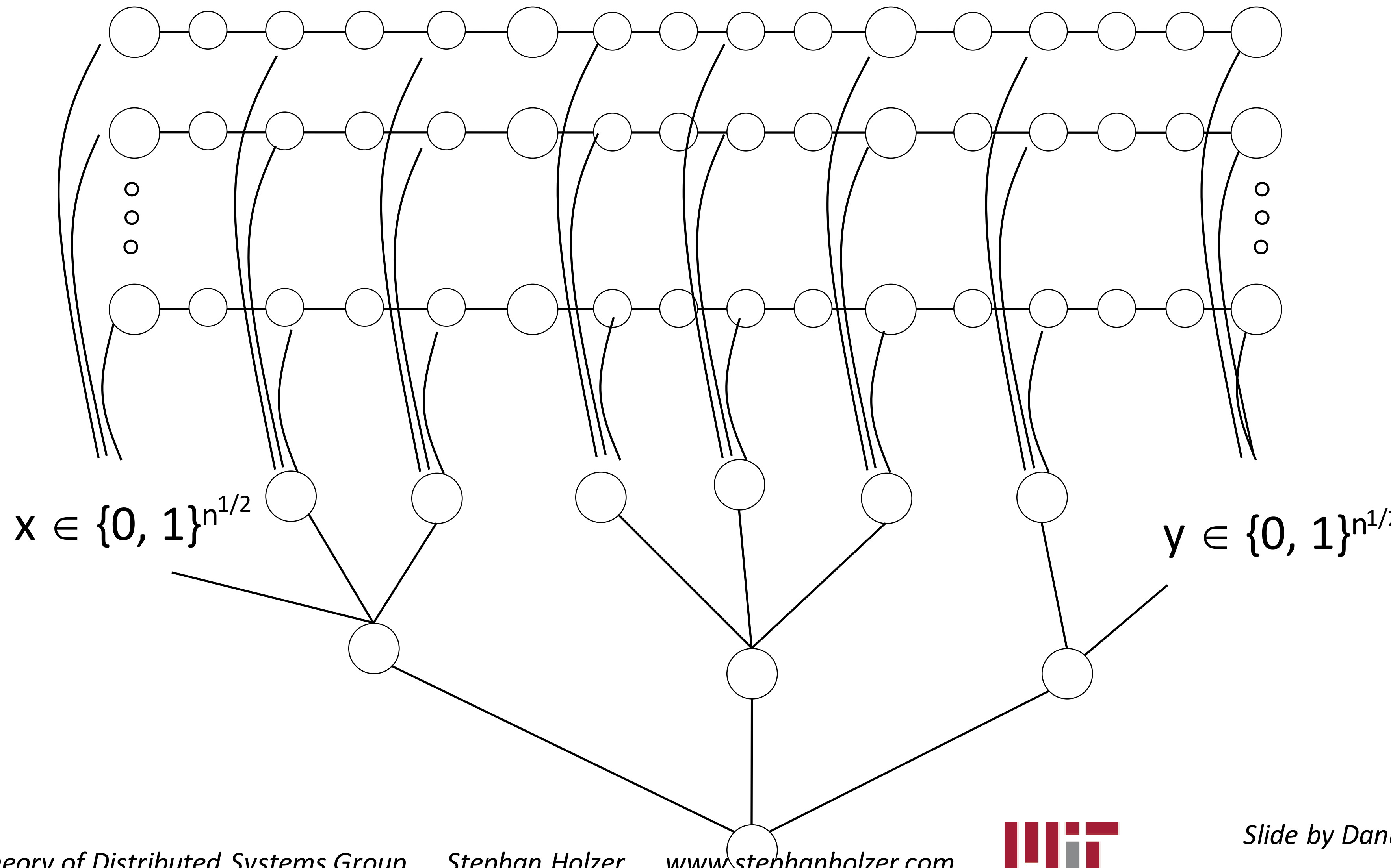
Approx MST lower bound $\Omega(n^{1/2})$



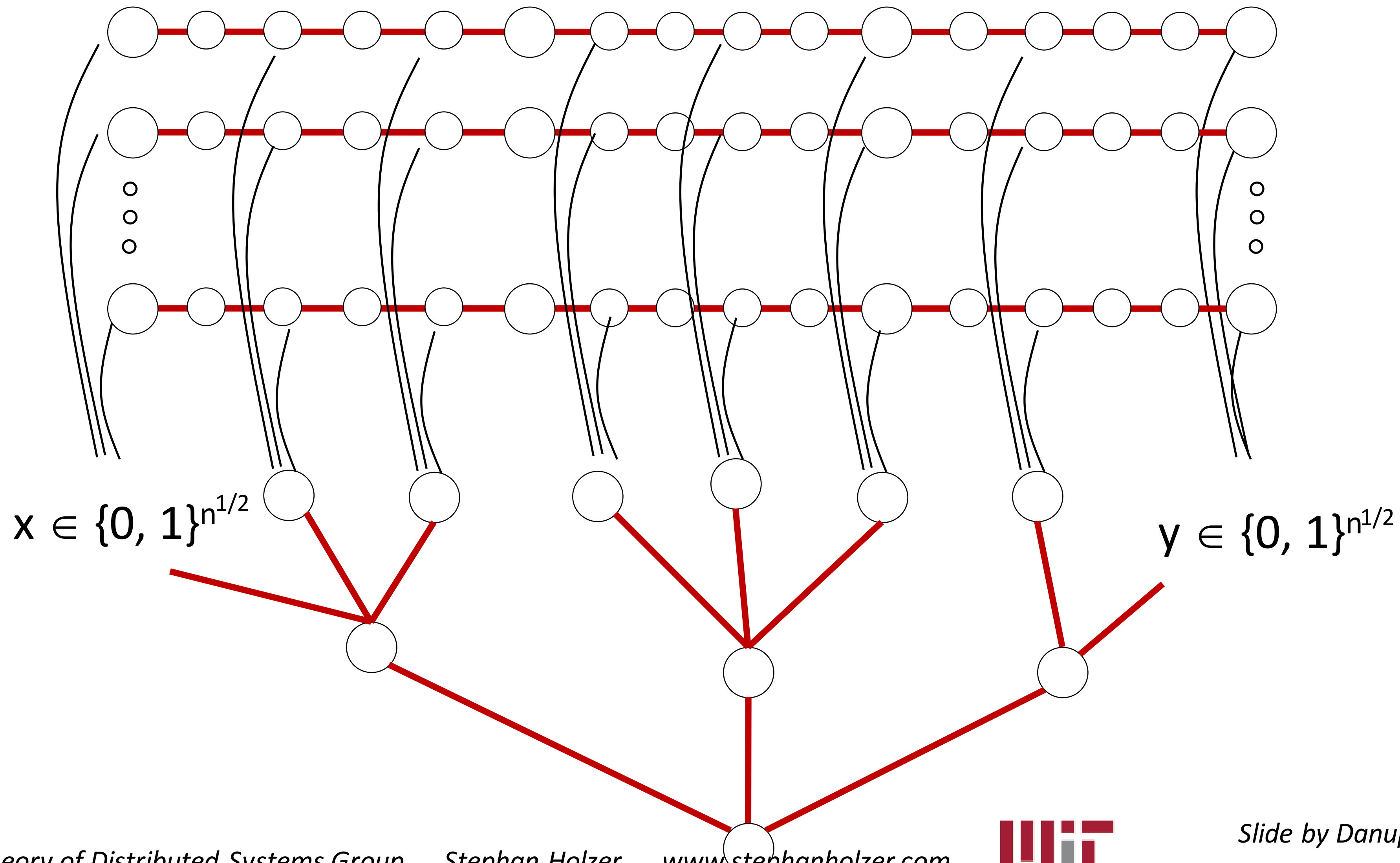
x ∈ {0, 1}ⁿ
y ∈ {0, 1}ⁿ



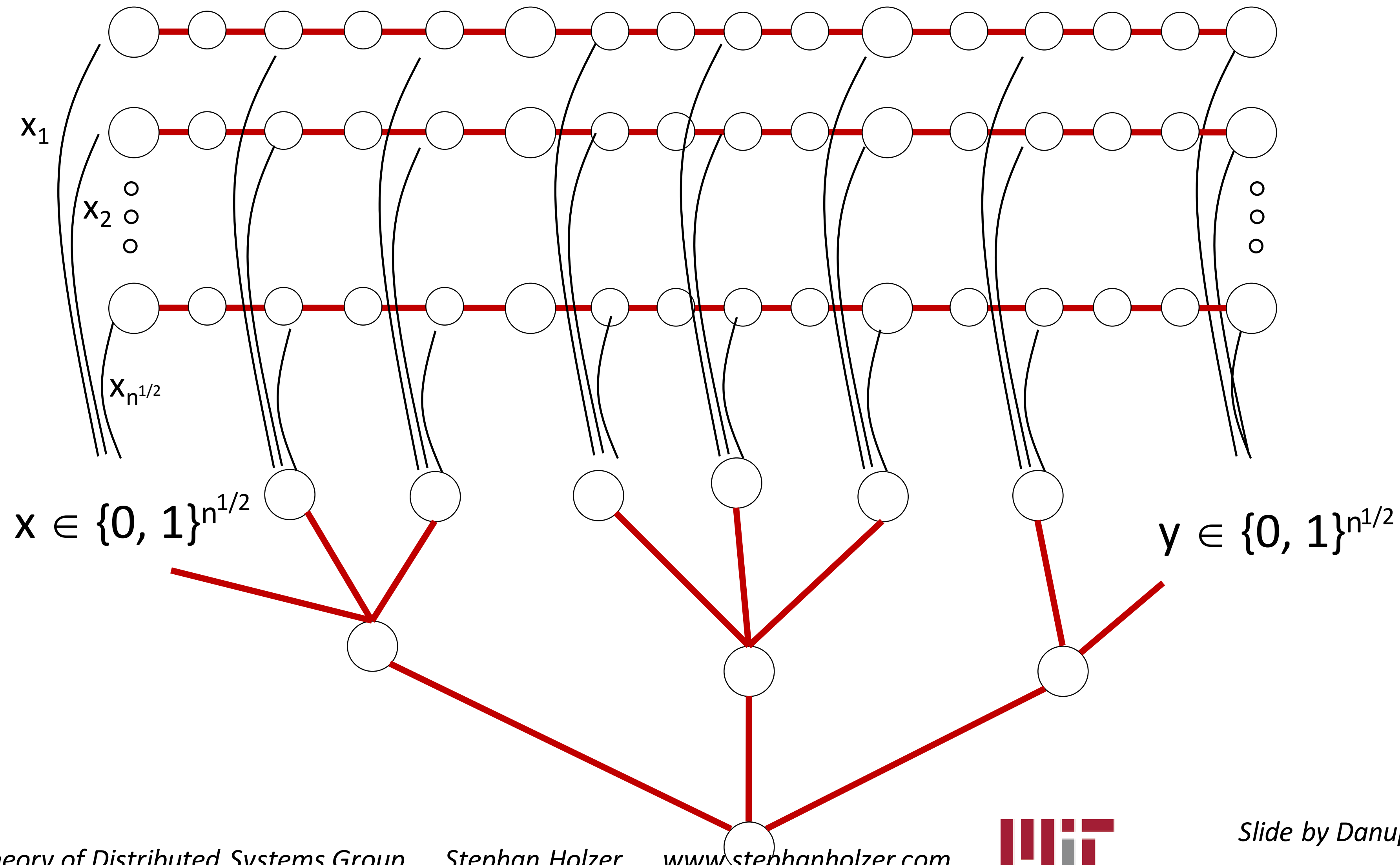
Slide by Danupon

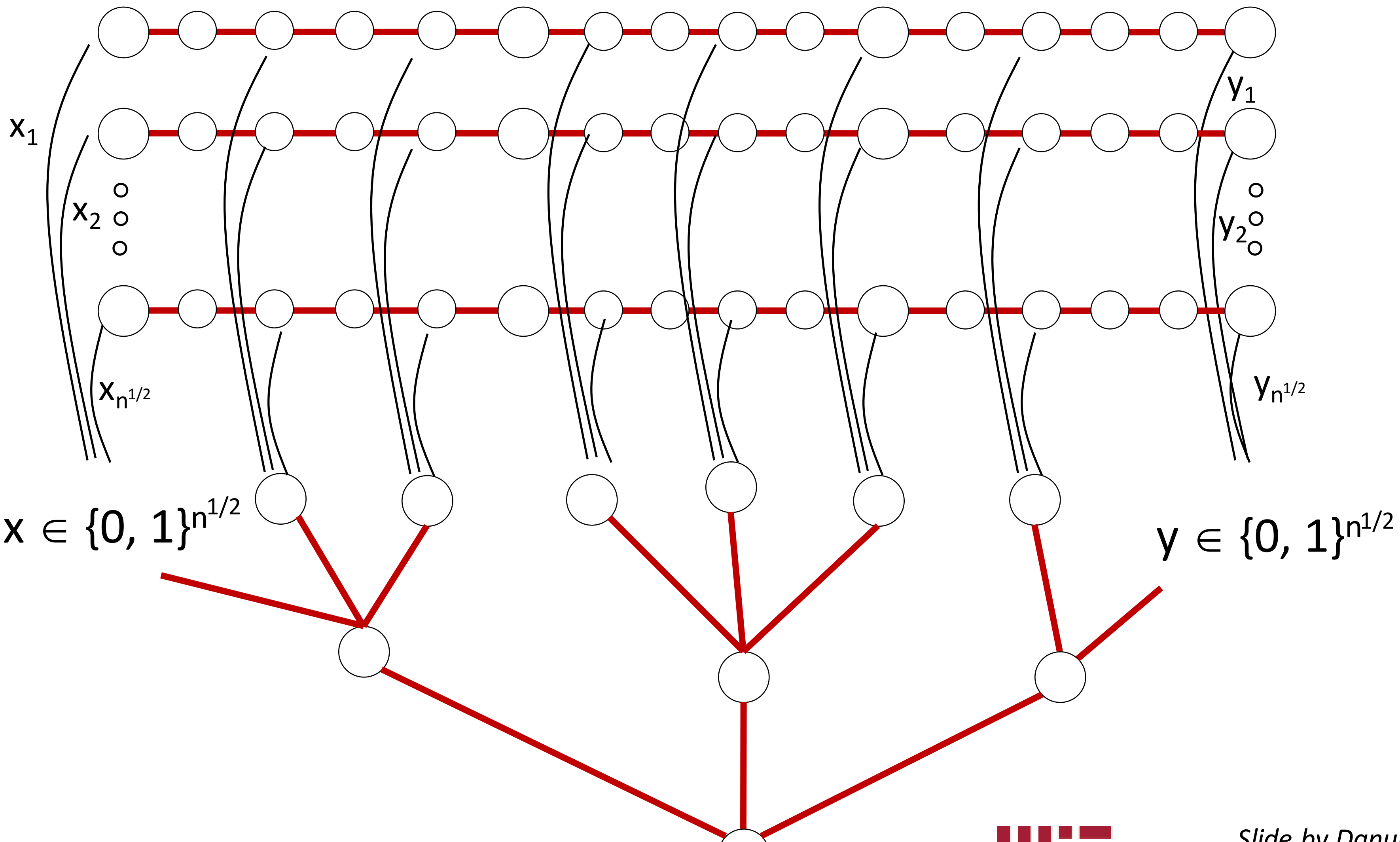


$x \in \{0, 1\}^{n^{1/2}}$
$y \in \{0, 1\}^{n^{1/2}}$

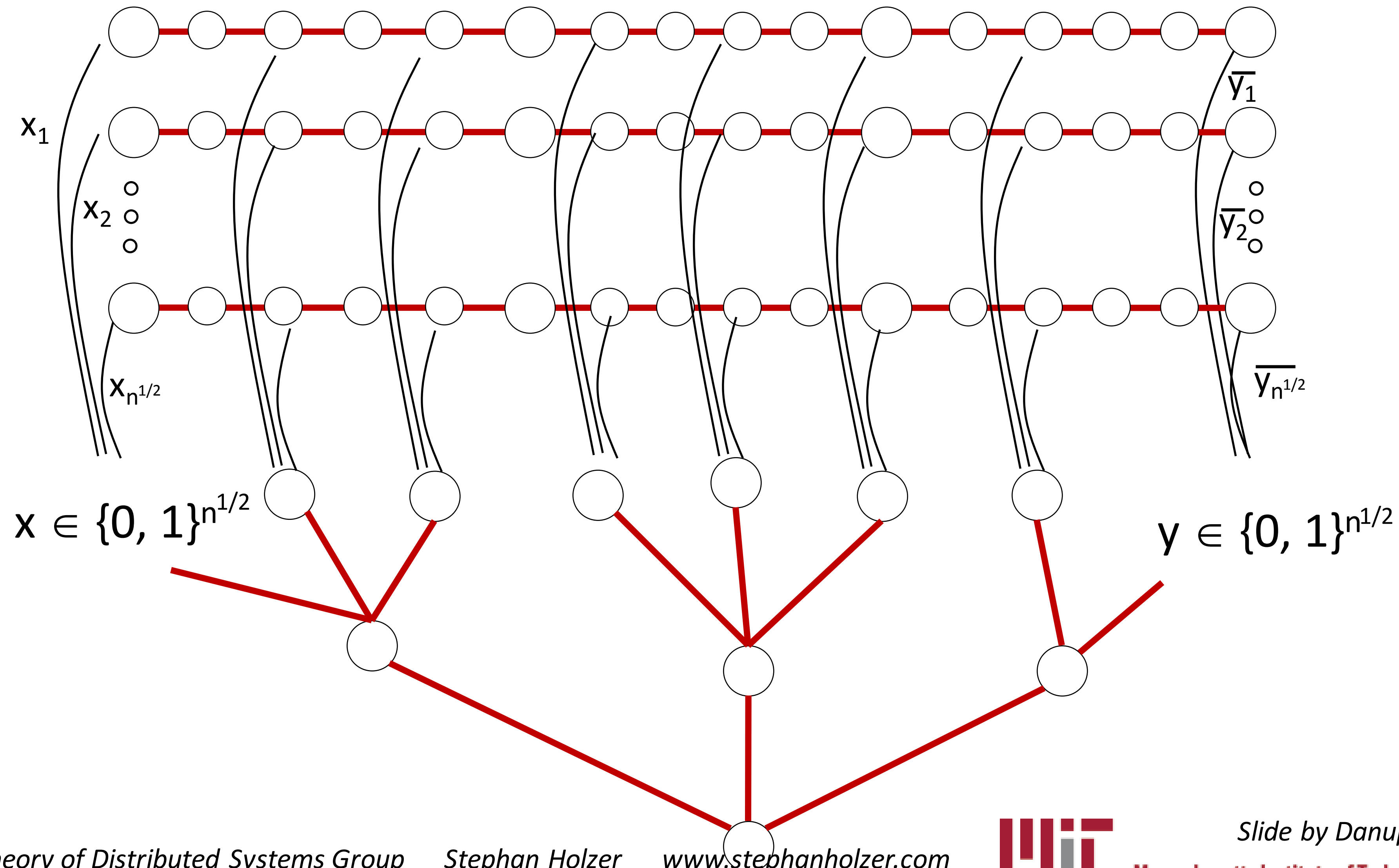


Slide by Danupon


x_1
x_2
$x_{n^{1/2}}$
$x \in \{0, 1\}^{n^{1/2}}$
$y \in \{0, 1\}^{n^{1/2}}$

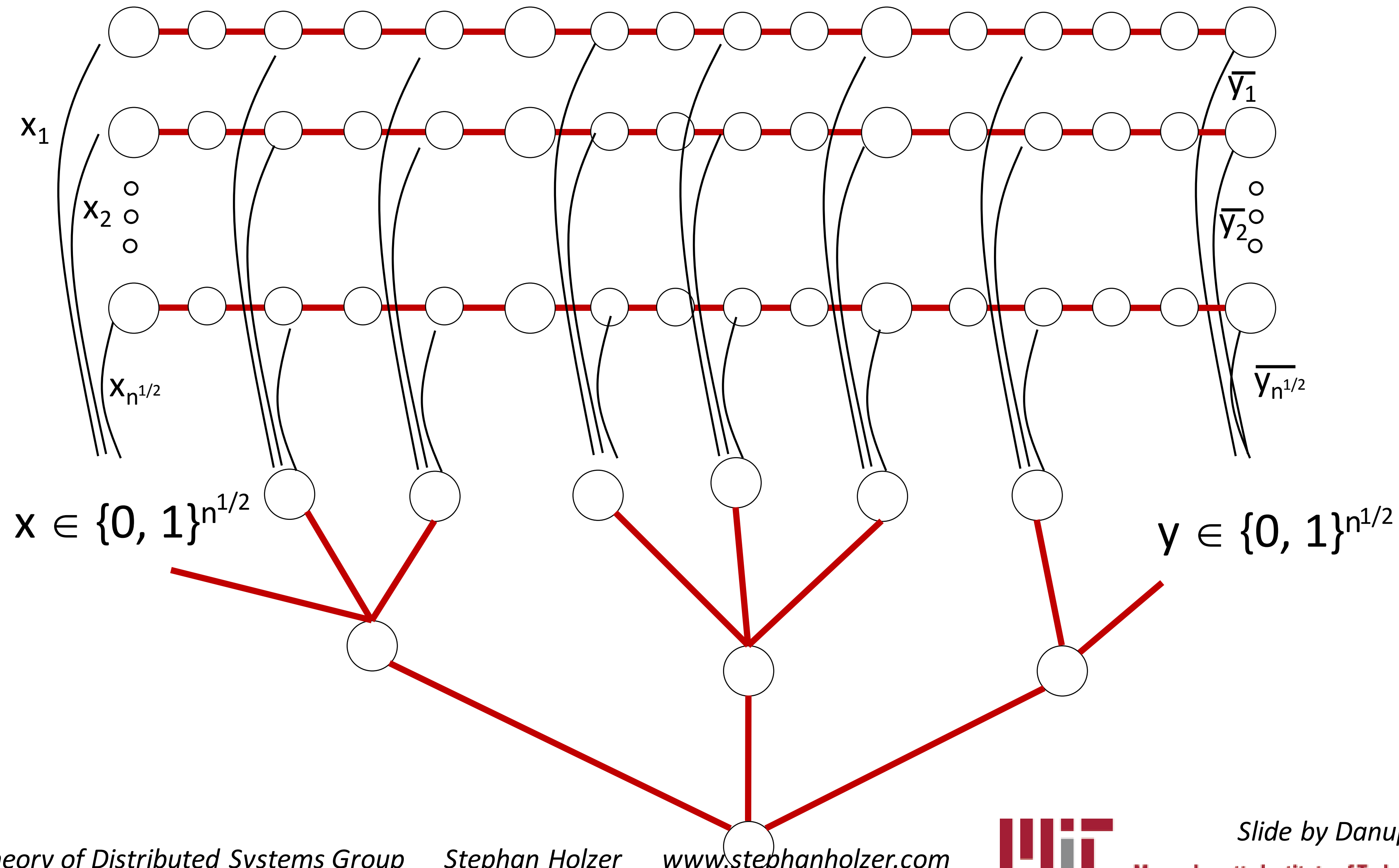


x_1
x_2
$x_{n^{1/2}}$
y_1
y_2
$y_{n^{1/2}}$
$x \in \{0, 1\}^{n^{1/2}}$
$y \in \{0, 1\}^{n^{1/2}}$

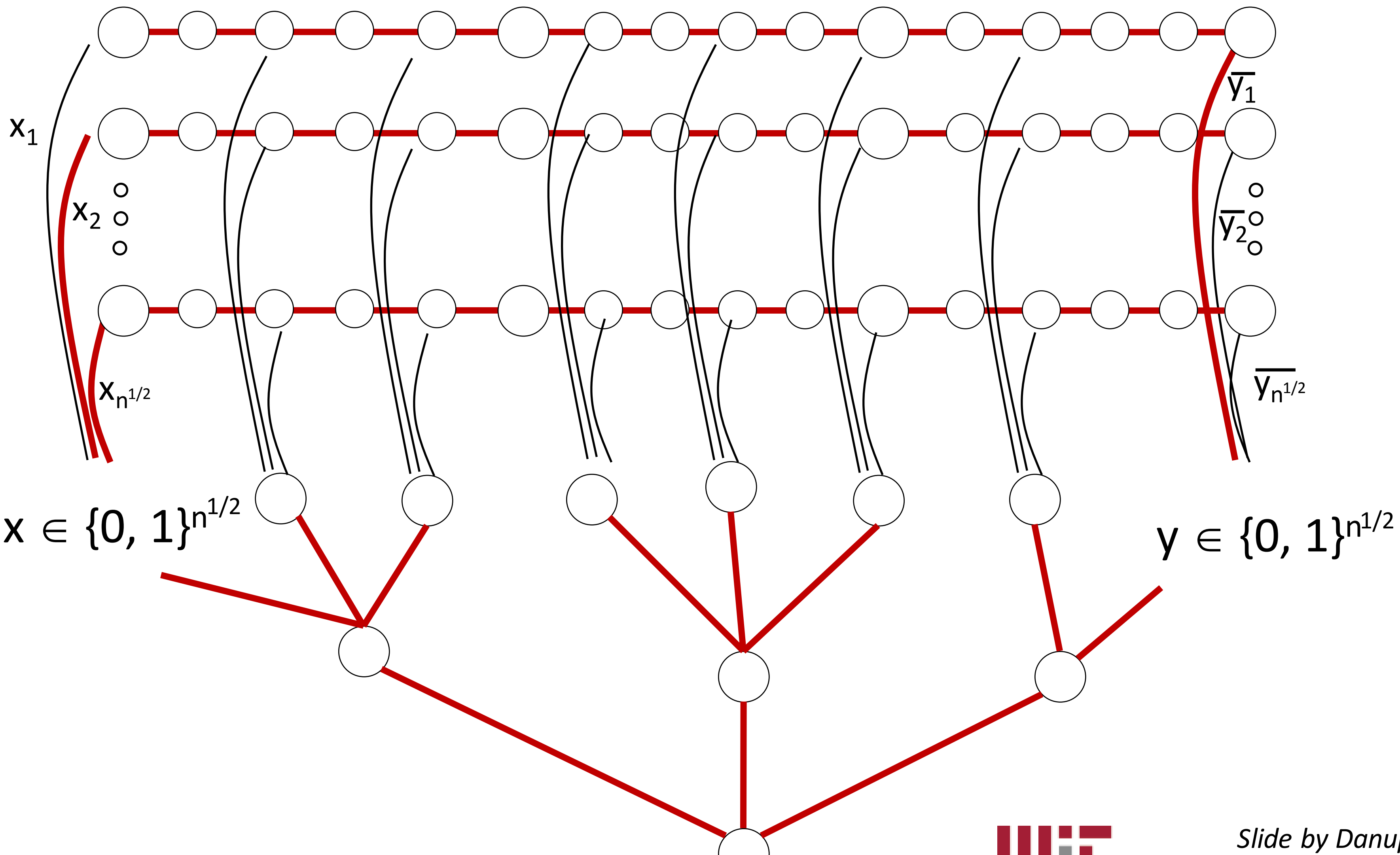


Slide by Danupon

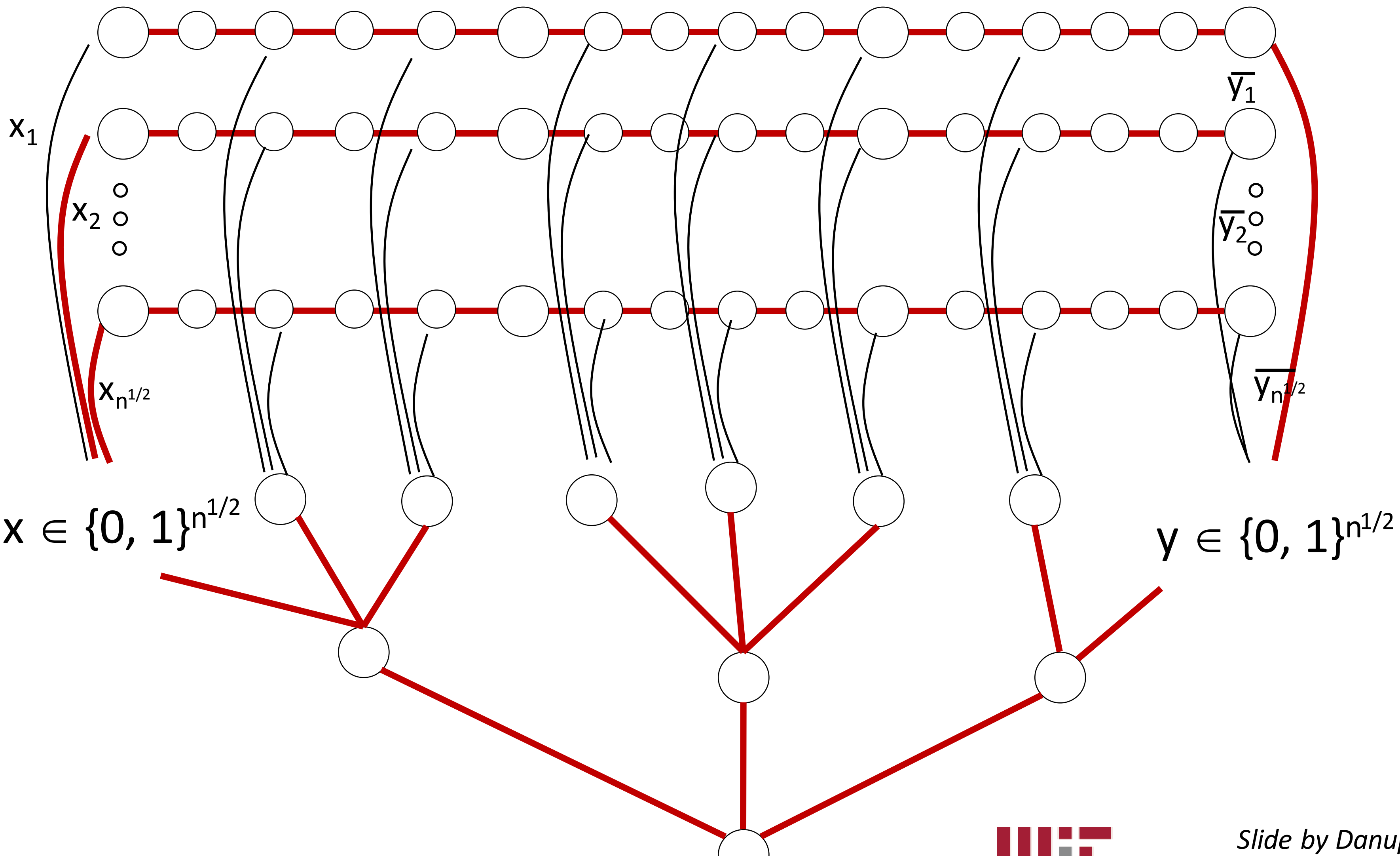
Example: x=01…1 y=01…1



Example: x=01…1 y=01…1



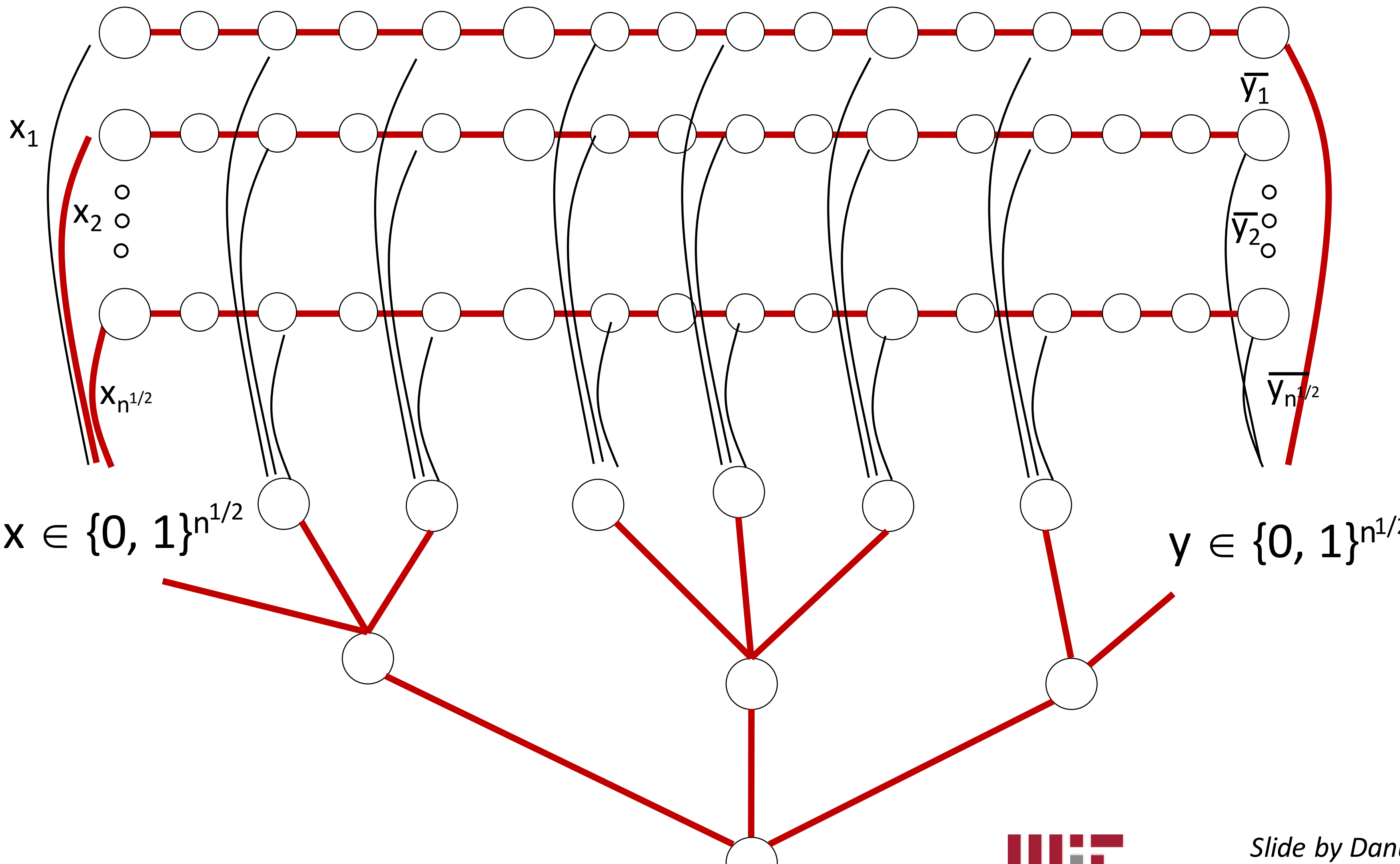
Example: x=01…1 y=01…1



Slide by Danupon

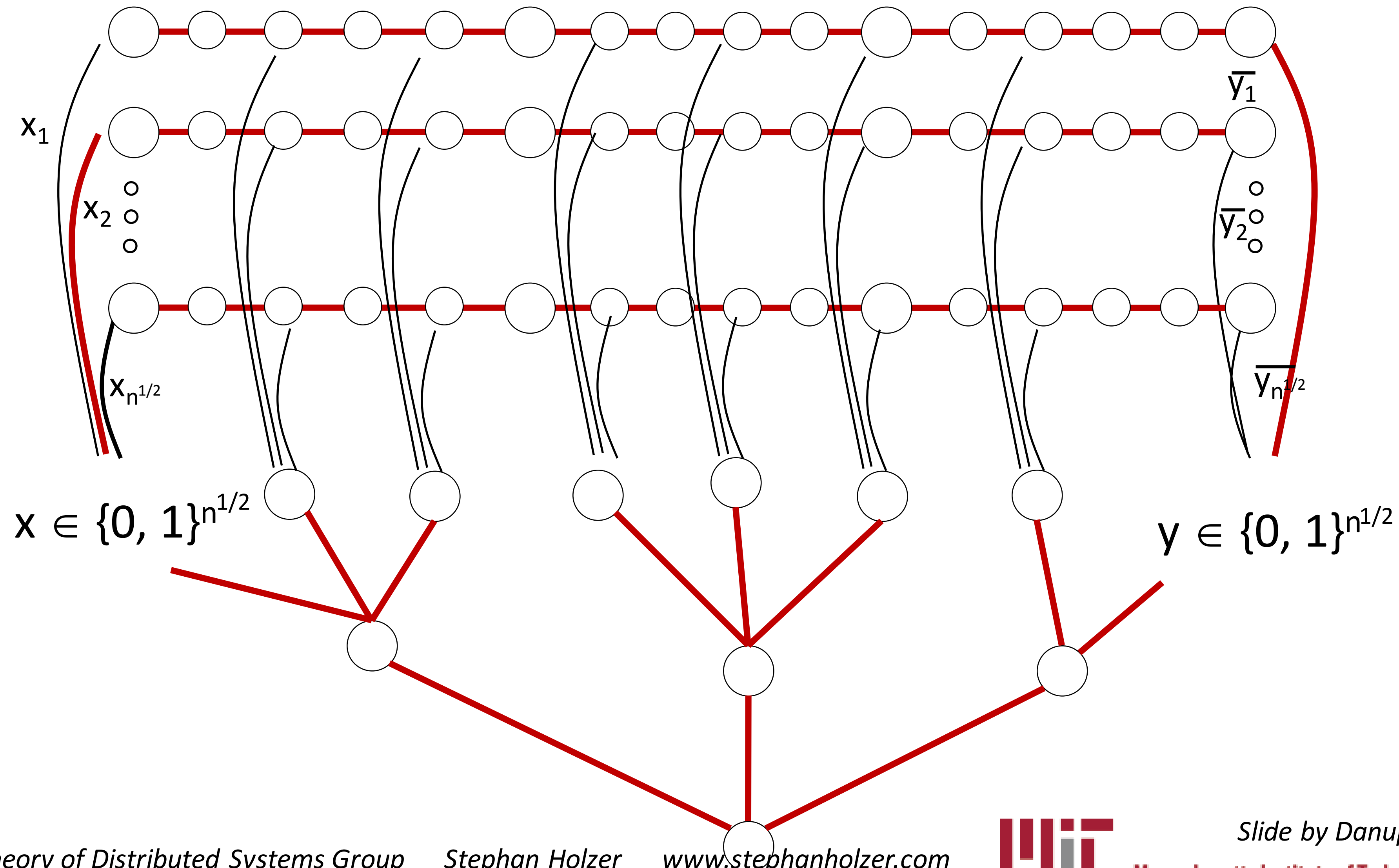
Example: x=01…1 y=01…1

Valid spanning tree



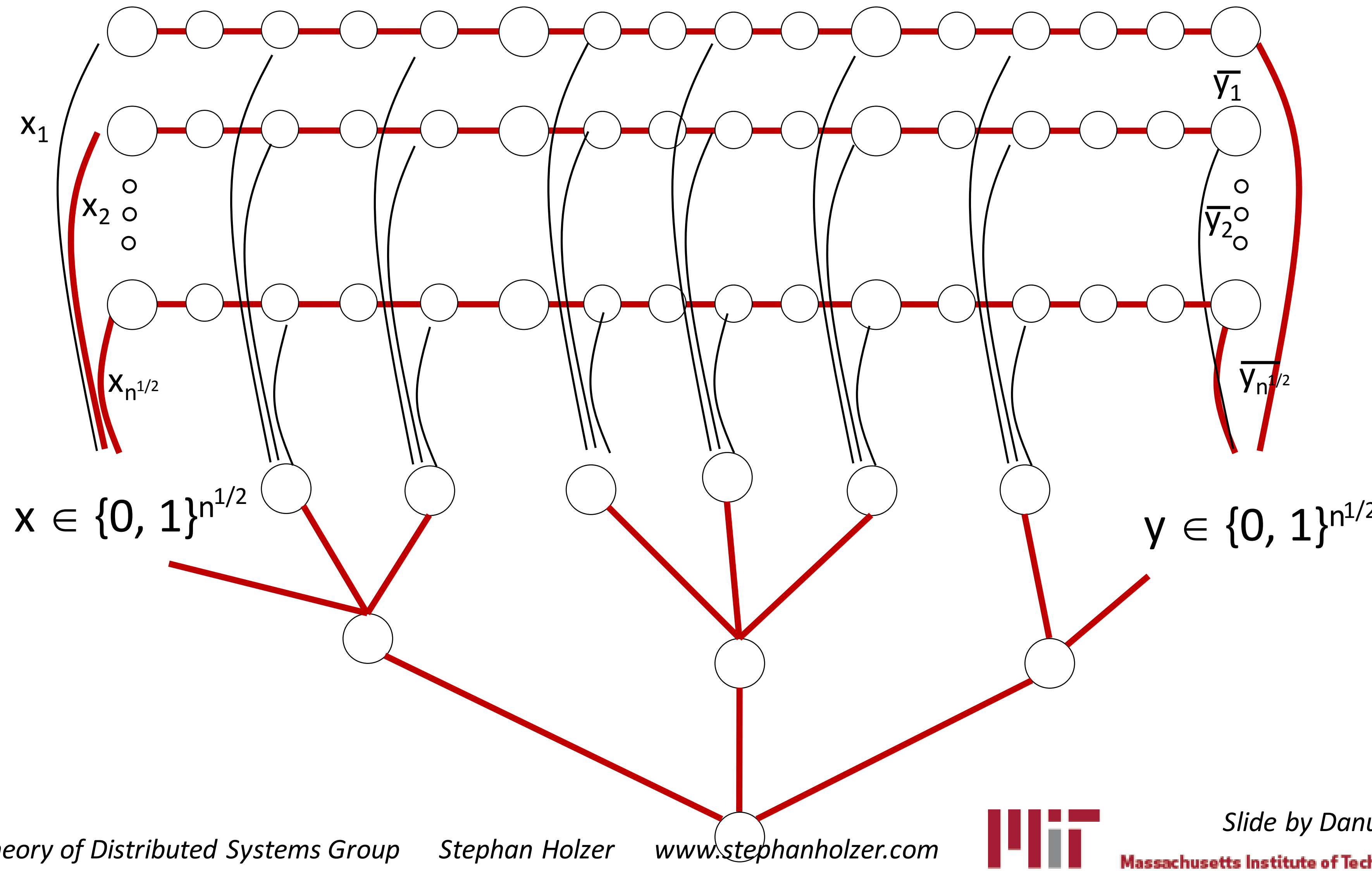
Another Example: x=01…0 y=01…1

Disconnected subgraph



Another Example: $x=01\ldots1$ $y=01\ldots0$

Subgraph with cycle



Three steps of reduction

Communication Complexity

Distributed Algorithms

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

→ (simulation theorem) →

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

Well-known result in communication complexity

Similar to lower bounds of graph streaming algorithms

↓

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

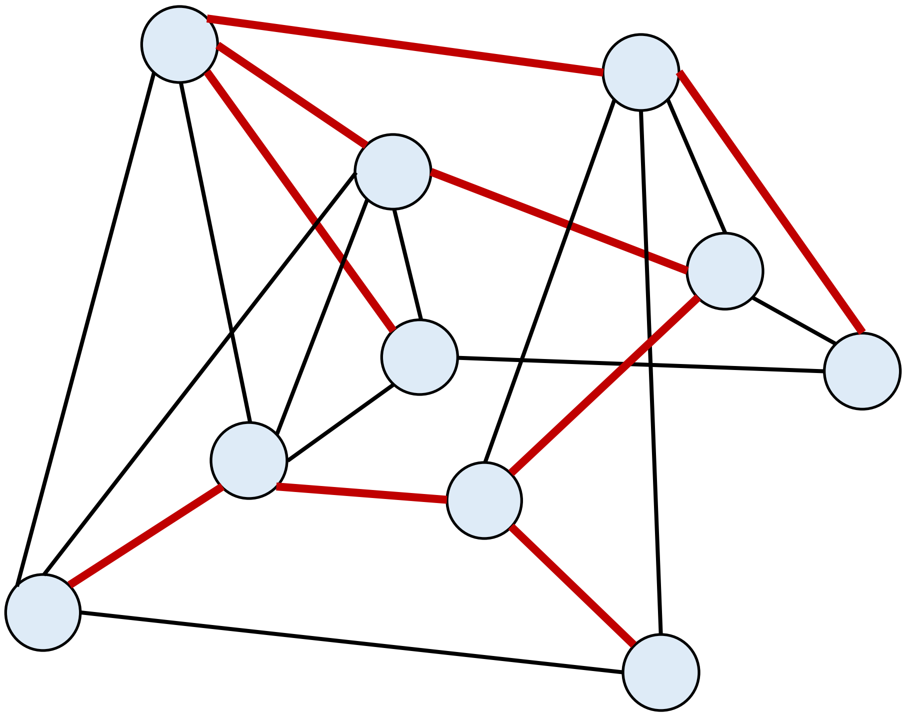
Similar to hardness of TSP

↓

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

From ST-Verification to MST-Approximation

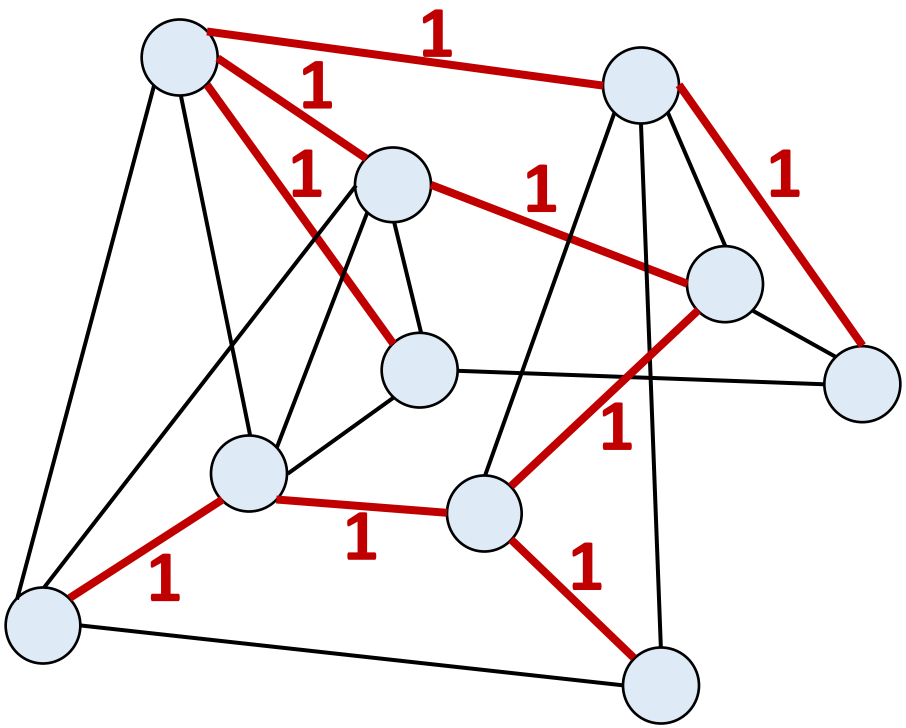
Given: G and subgraph H



Use α-approximation for MST to decide if H is ST

From ST-Verification to MST-Approximation

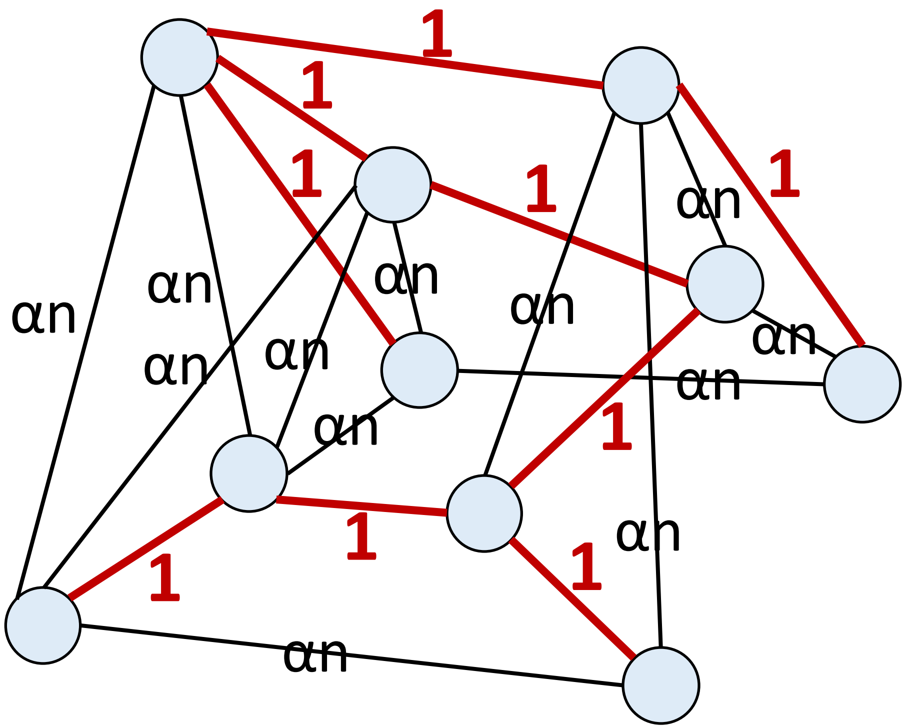
Given: G and subgraph H



Use α-approximation for MST to decide if H is ST

From ST-Verification to MST-Approximation

Given: G and subgraph H



Use α-approximation for MST to decide if H is ST

Observe: iff H is ST, H is MST of weight n-1

Observe: iff H is ST, no α-MST besides H

Thus: α-approximating a MST takes $\Omega(n^{1/2})$

Three steps of reduction

Communication Complexity | **Distributed Algorithms**

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

simulation theorem

Well-known result in communication complexity

Similar to lower bounds of graph streaming algorithms

↓

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

↓ Similar to hardness of TSP

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

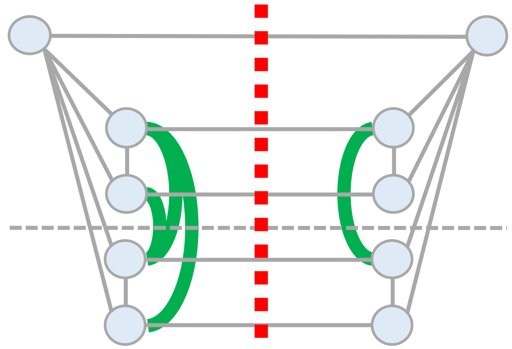
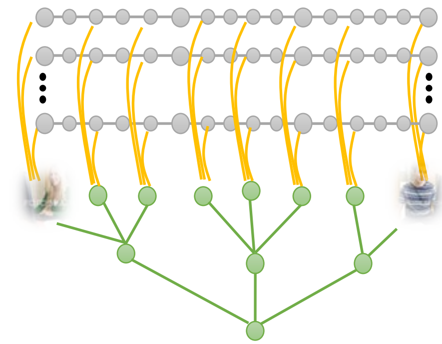
Deterministic lower bound.

Randomized: use DISJOINTNESS and

Different intermediate steps.

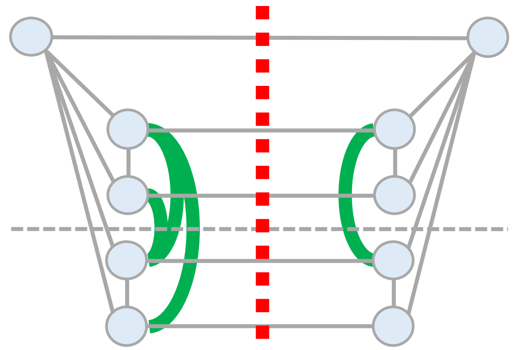
Simulation Theorem: works for any function.

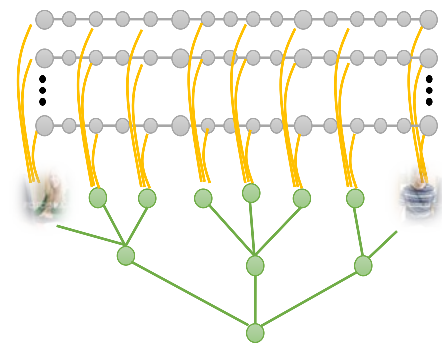
Comparison of the Techniques

General:	yes	yes
Approximation LB:	yes	yes
Best LB possible:	$\Omega(n)$	$\Omega(n^{1/2})$
Diameter of graph:	3	$O(\log n)$

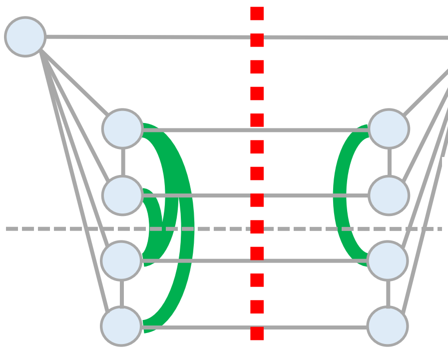
Comparison of the Techniques

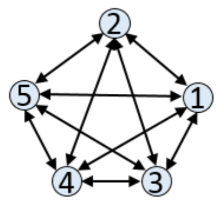




General:	yes	yes
Approximation LB:	yes	yes
Best LB possible:	$\Omega(n)$	$\Omega(n^{1/2})$
Diameter of graph:	3	$O(\log n)$
Problems applied to:	>15	>22

Summary

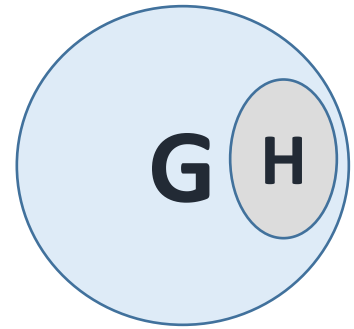




1. Pick a root-node r;
2. **T** := BFS-Tree(r);
3. Pebble **P** traverses **T** in preorder;
4. **If** P visits node v first time**{**
   wait 1 timeslot;
   start shortest paths(v);
   }

Diameter Ω(n)

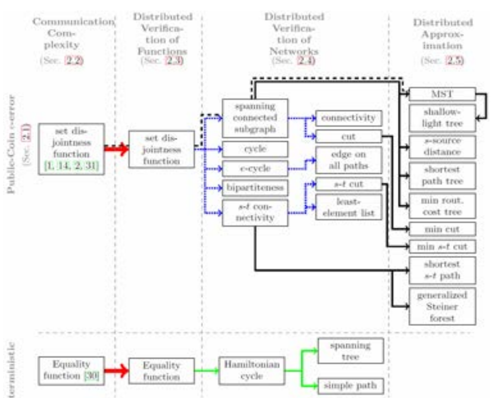
Diameter O(n)



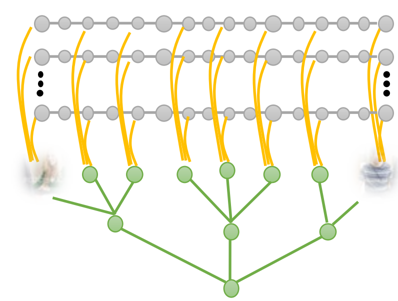
Verification harder than computing

α-MST

$\Omega(n^{1/2})$



22 Lower bounds



Simulation Theorem

Thanks!