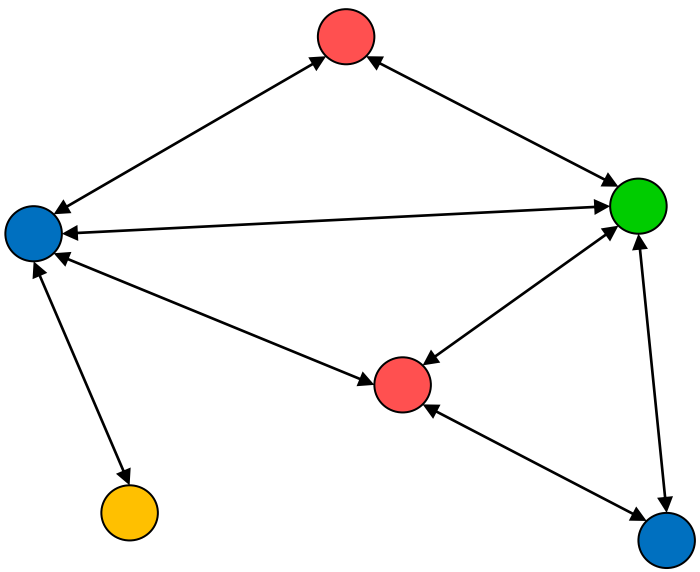
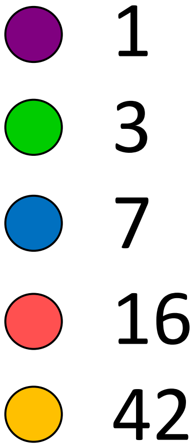
Distributed Algorithms on a Congested Clique

Christoph Lenzen

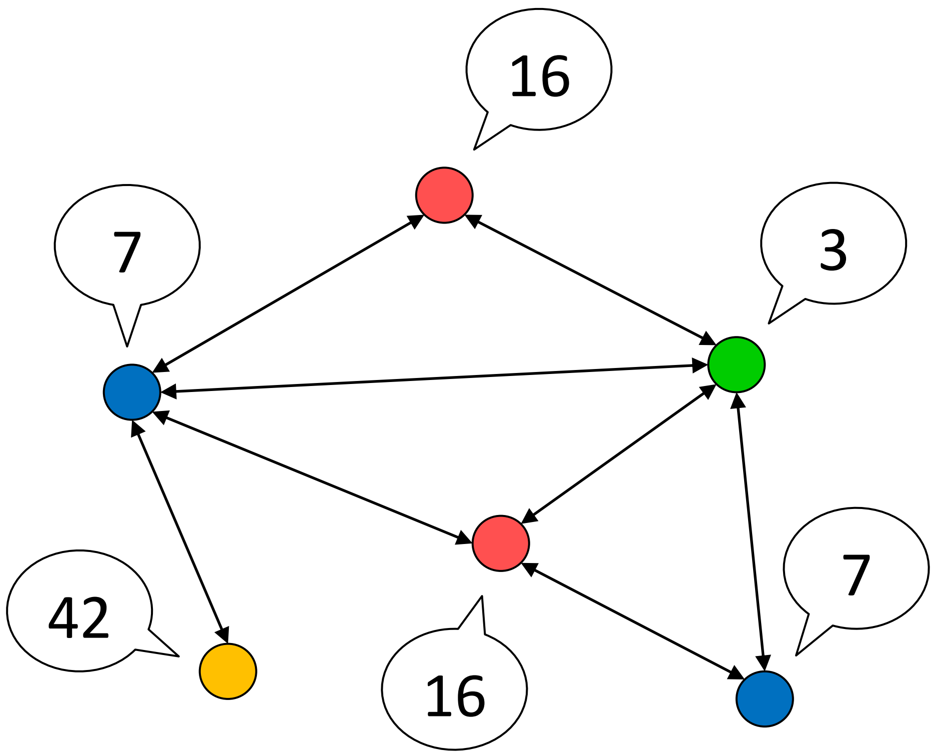
The LOCAL Model

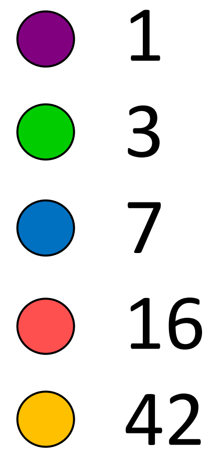




1. compute
2. send
3. receive

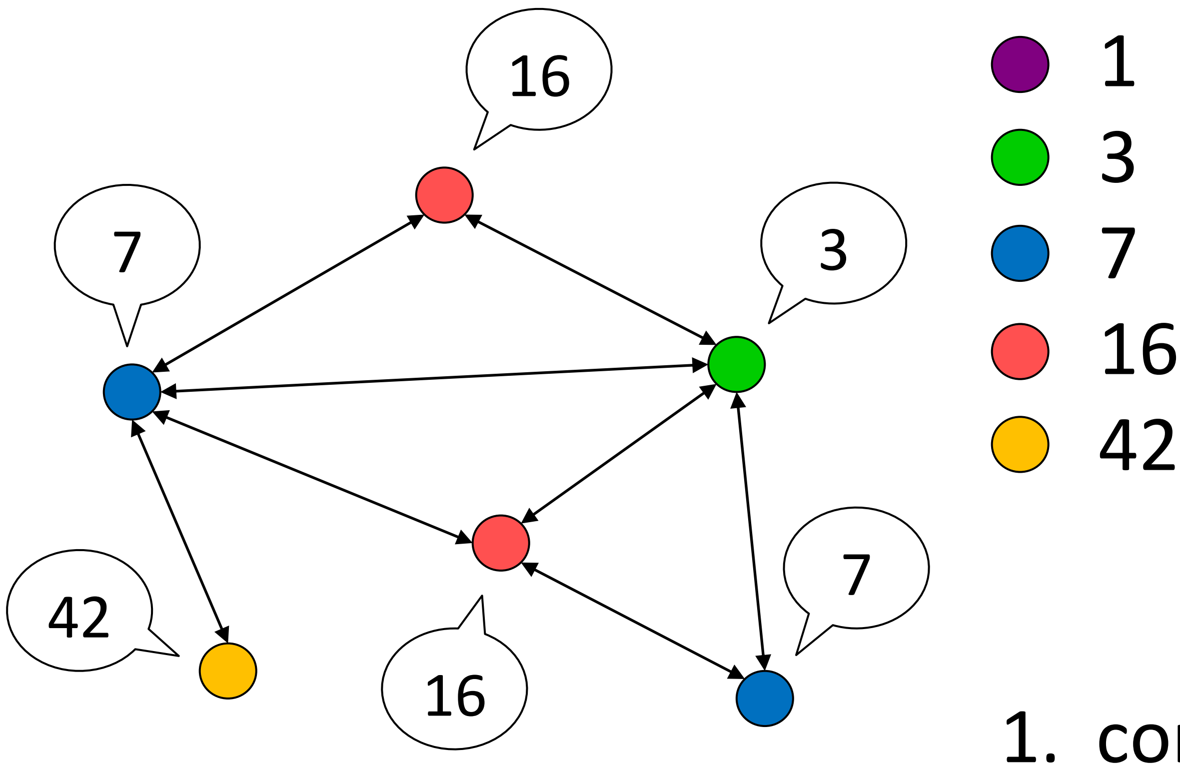
The LOCAL Model





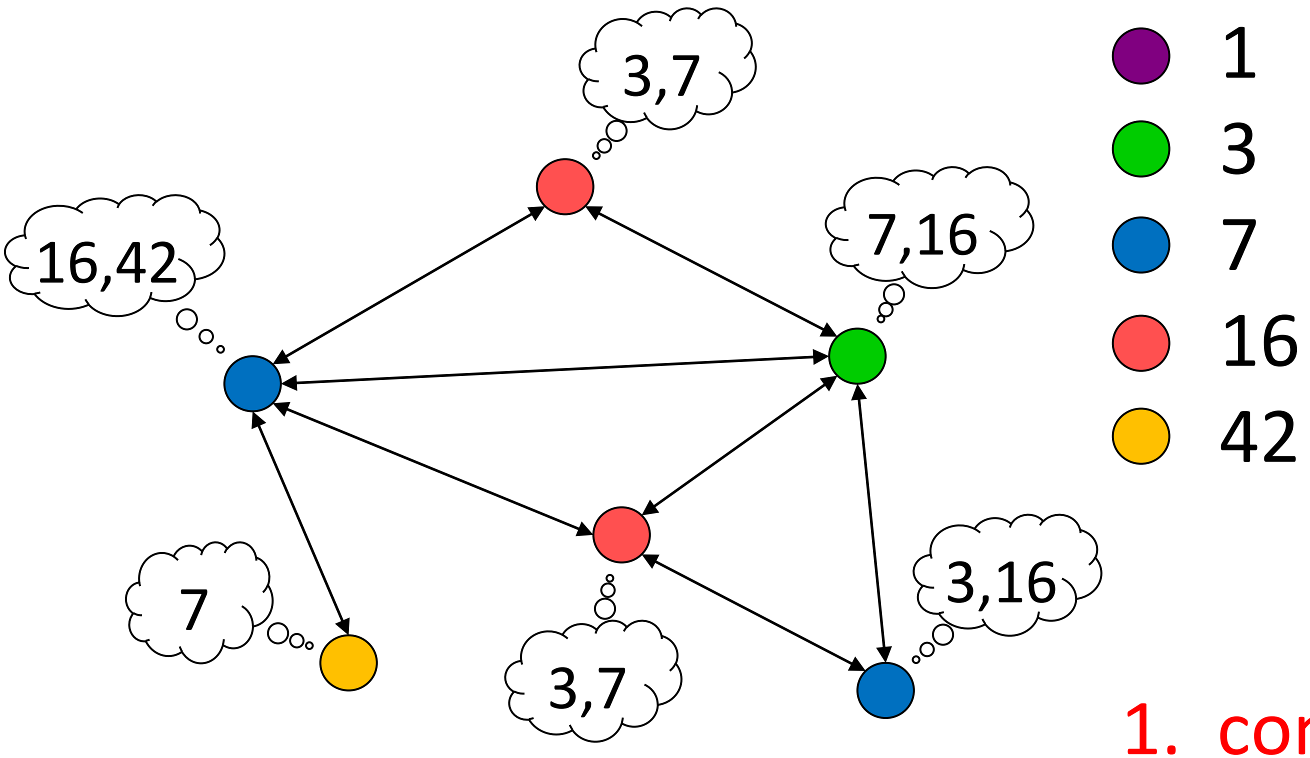
1. compute
2. send
3. receive

The LOCAL Model



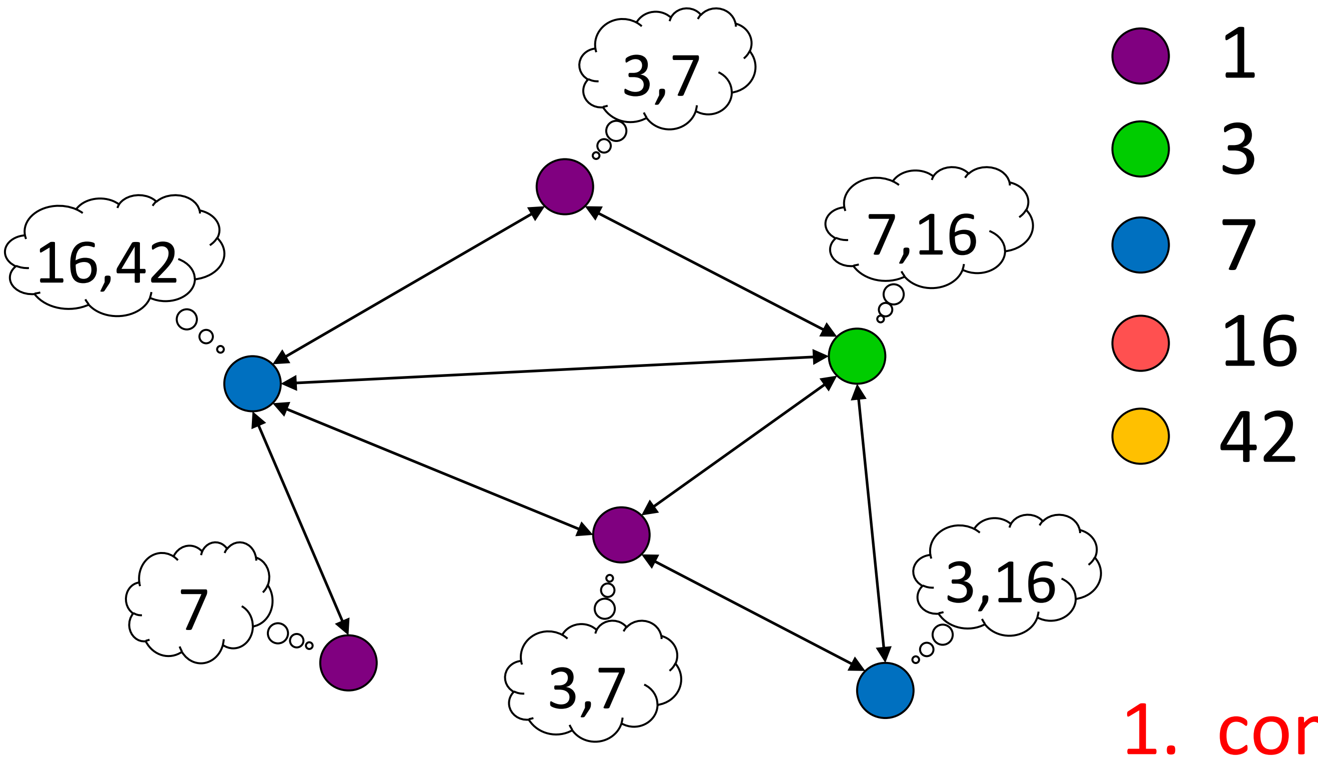
1. compute
2. send
3. receive

The LOCAL Model



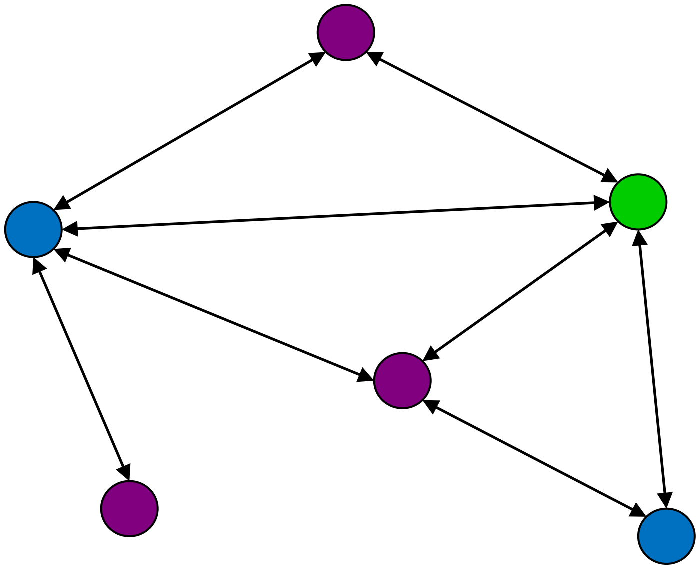
1. compute
2. send
3. receive

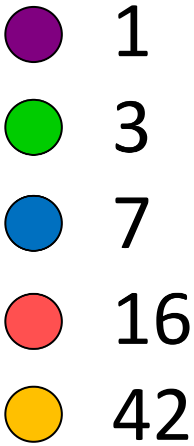
The LOCAL Model



1. compute
2. send
3. receive

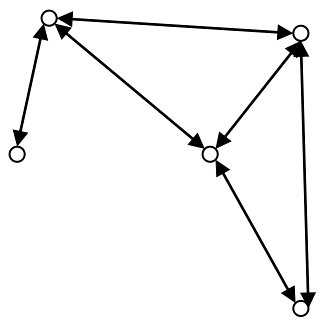
The LOCAL Model





1. compute
2. send
3. receive

LOCAL + restricted bandwidth = CONGEST !

 +  =  ?

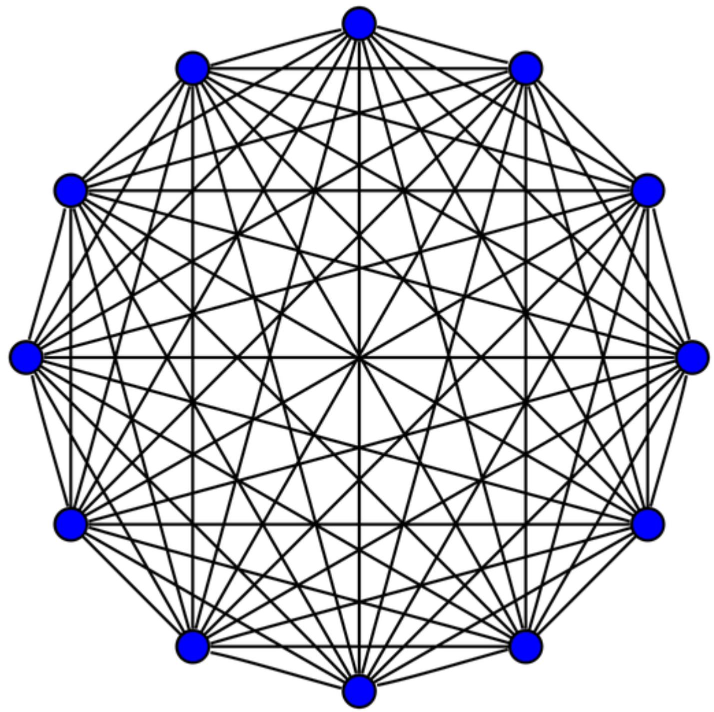
synchr. rounds:

1. compute
2. send
3. receive

message size:
O(log n) bits

...content can differ
between neighbors!

round
complexity?



What happens here?

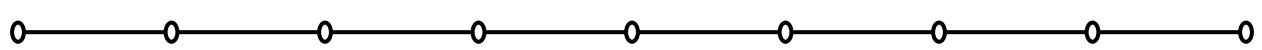
Disclaimer

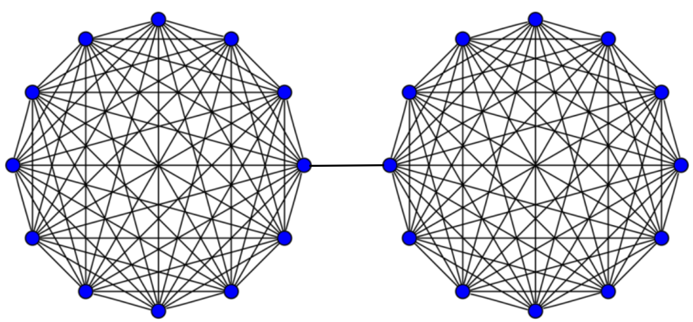
Practical relevance
of this model
is questionable!

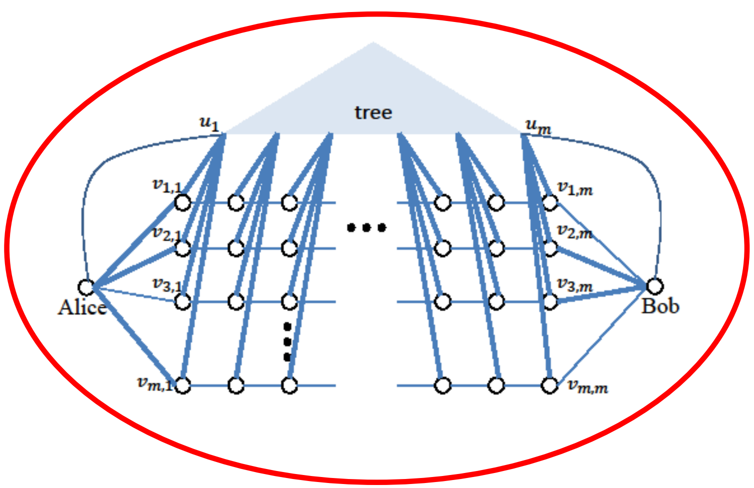
Algorithms for overlay networks?
Subroutines for small cliques in larger networks?

So why should we care?!?

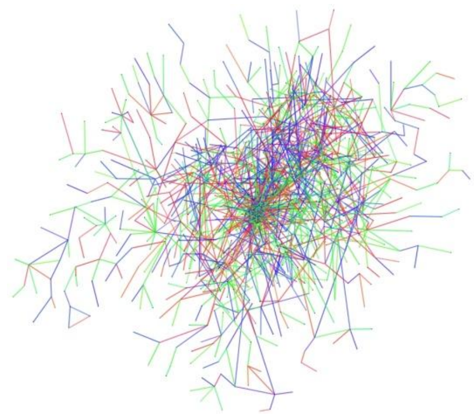
what lower bound graphs look like:

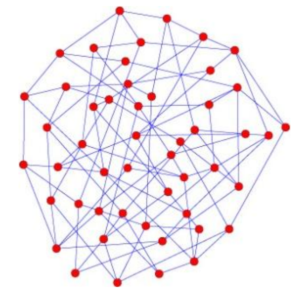


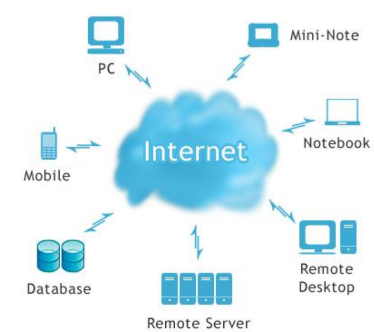




what "real" networks look like:







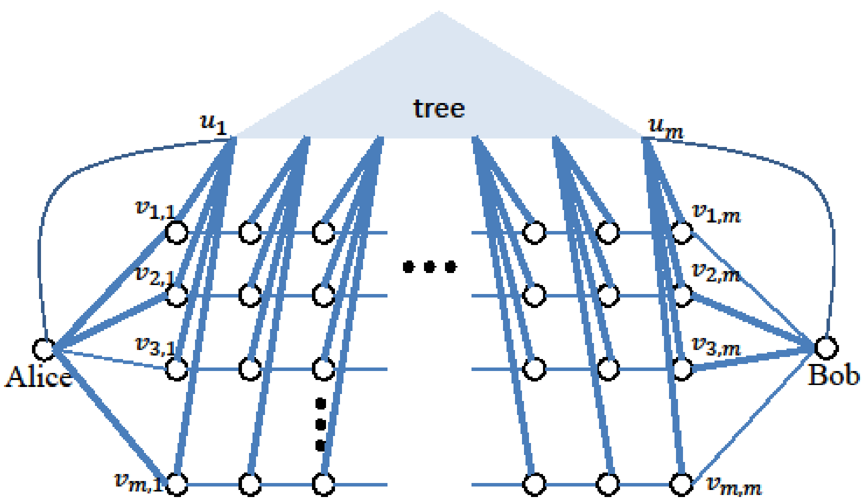
History: MST Lower Bound

Peleg and Rubinovich
SIAM J. on Comp.'00

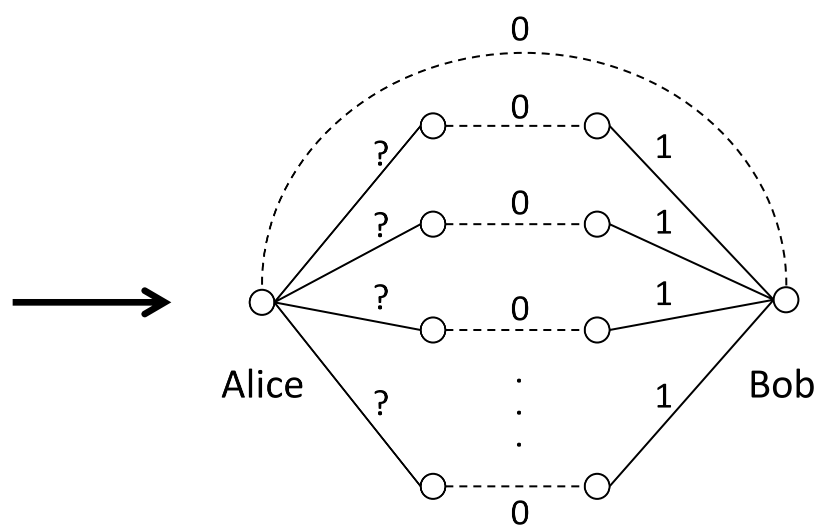
Input: weighted graph

Output: spanning tree

Goal: minimize weight of tree



≈ √n x √n



History: MST Lower Bound

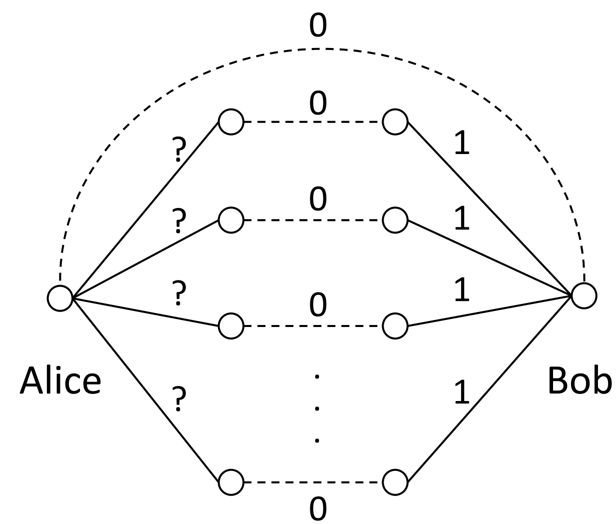
Peleg and Rubinovich
SIAM J. on Comp.'00

Input: weighted graph

Output: spanning tree

Goal: minimize weight of tree

- Alice gets bit string **b** as input



History: MST Lower Bound

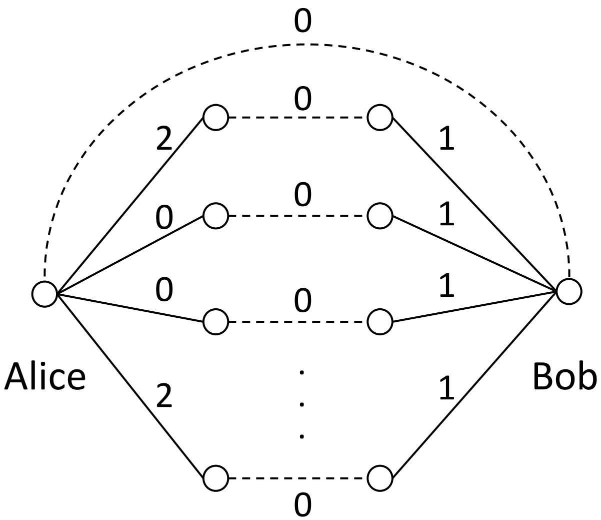
Peleg and Rubinovich
SIAM J. on Comp.'00

Input: weighted graph

Output: spanning tree

Goal: minimize weight of tree

- Alice gets bit string **b** as input
- assign weight $2b_i$ to i^{th} edge



History: MST Lower Bound

Peleg and Rubinovich
SIAM J. on Comp.'00

Input: weighted graph

Output: spanning tree

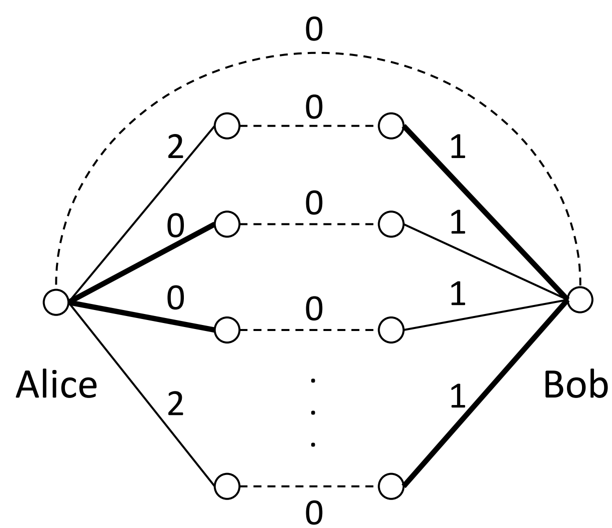
Goal: minimize weight of tree

- Alice gets bit string **b** as input
- assign weight $2b_i$ to i^{th} edge
- compute MST

=> Bob now knows **b**!

=> Alice sent ≥|**b**| bits to Bob

How long does this take?



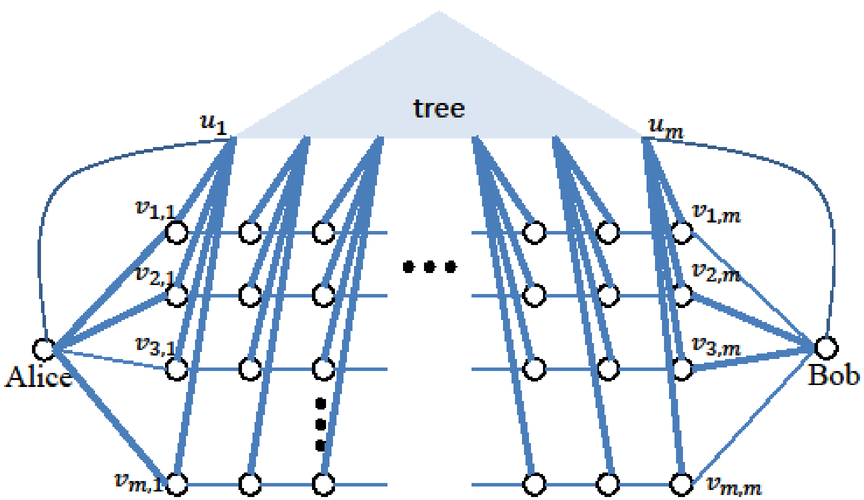
History: MST Lower Bound

Peleg and Rubinovich
SIAM J. on Comp.'00

Input: weighted graph

Output: spanning tree

Goal: minimize weight of tree



≈ √n x √n

|**b**| bits sent in time T

=> |**b**|/(T log n) edge-disjoint paths

T ≤ o(√n)

=> paths use tree edges to "shortcut" Ω(√n) hops

History: MST Lower Bound

Peleg and Rubinovich
SIAM J. on Comp.'00

Input: weighted graph

Output: spanning tree

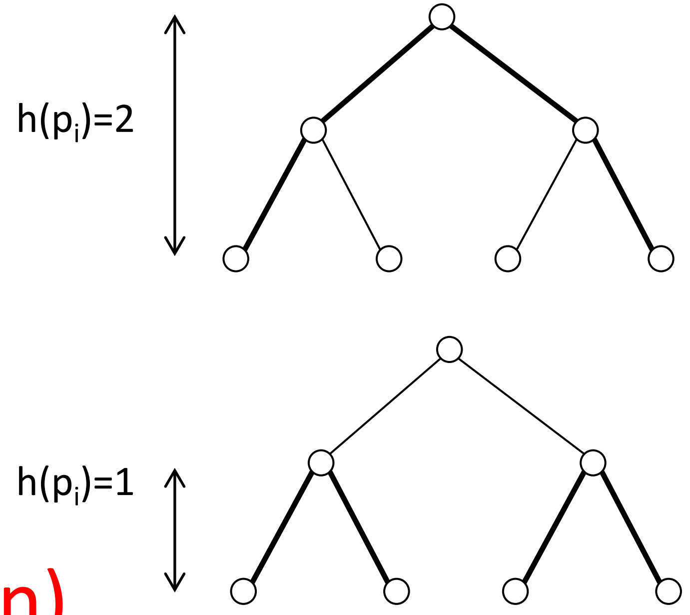
Goal: minimize weight of tree

for each path p:

- p_i subpaths in tree
- $h(p_i)$ max. dist. from leaves
- $\sum_i 2^{h(p_i)} \geq \Omega(\sqrt{n})$

but $\sum_p \sum_i 2^{h(p_i)} \leq \sqrt{n} \log n$

=> $O(\log n)$ paths, $T \geq \Omega(\sqrt{n}/\log^2 n)$



MST Lower Bound: Summary

- general technique
- yields lower bounds of roughly $\Omega(\sqrt{n})$
- helped finding many near-matching algorithms

Das Sarma et al.
STOC`11

Das Sarma et al.
SPAA`12

L. and Peleg
PODC`12

Peleg et al
ICALP`12

Elkin
ACM Trans. on Alg.`05

Elkin
SIAM J. on Comp.`06

Elkin and Peleg
SIAM J. on Comp.`04

Frischknecht et al.
SODA`12

Khan et al.
Dist. Computing`12

Khan and Pandurangan
Dist. Computing`08

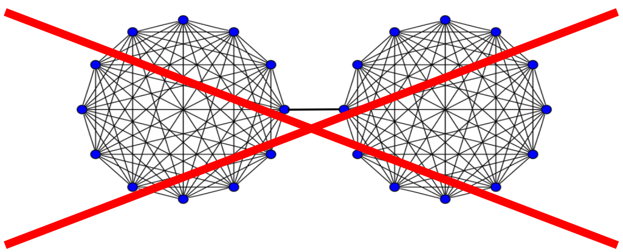
Kutten and Peleg
J. Algorithms`98

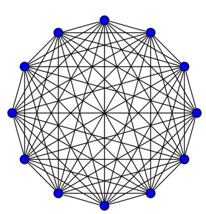
L. and Patt-Shamir
STOC`13

Holzer and Wattenhofer
PODC`12

But How About Well-Connected Graphs?

diameter	upper bound	lower bound
$O(\log n)$	$O(n^{1/2} \log^* n)$	$\Omega(n^{1/2}/\log^2 n)$
4	?	$\Omega(n^{1/3}/\log n)$
3	?	$\Omega(n^{1/4}/\log n)$
2	$O(\log n)$	?
1	$O(\log \log n)$	?





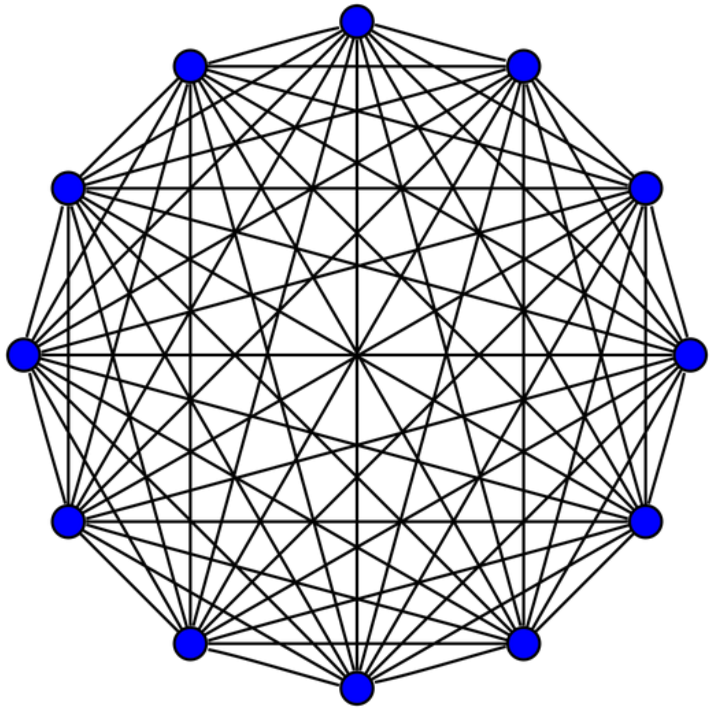
Lotker et al.
Dist. Computing´06

Lotker et al.
SIAM J. on Comp.`05

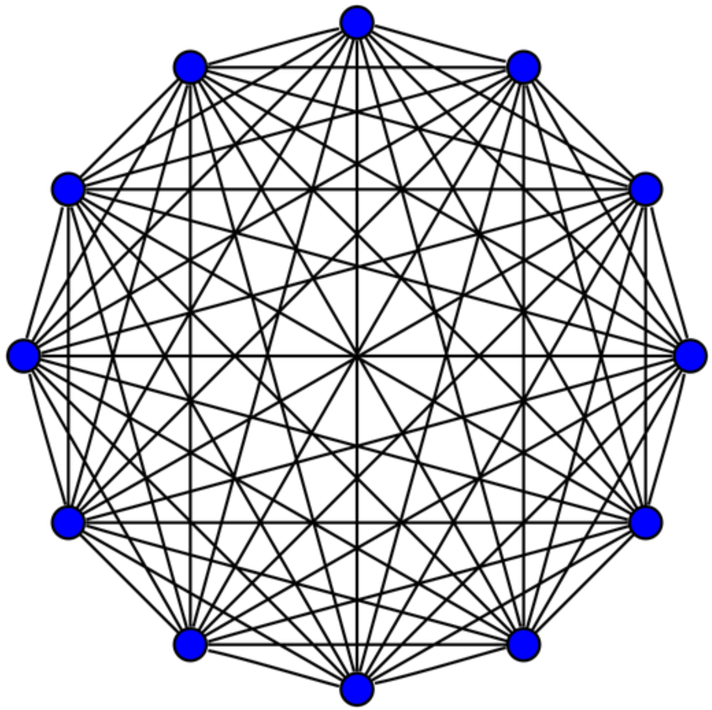
But How About Well-Connected Graphs?

diameter	upper bound	lower bound
$O(\log n)$	$O(n^{1/2} \log^* n)$	$\Omega(n^{1/2}/\log^2 n)$
4	?	$\Omega(n^{1/3}/\log n)$
3	?	$\Omega(n^{1/4}/\log n)$
2	$O(\log n)$	?
1	$O(\log \log n)$	?

All known lower bounds are based on hardness of spreading information!



~~What happens here?~~



...multi-party communication complexity!

~~What happens here?~~

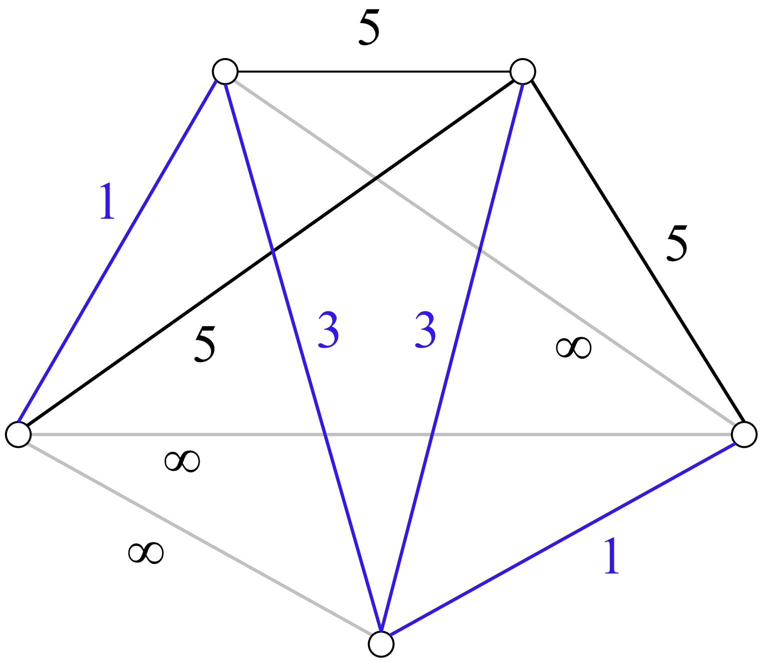
What happens if there is no communication bottleneck?

What We Know: MST

input: weight of adjacent edges

output: least-weight spanning tree

Lotker et al.,
Distr. Comp.'06

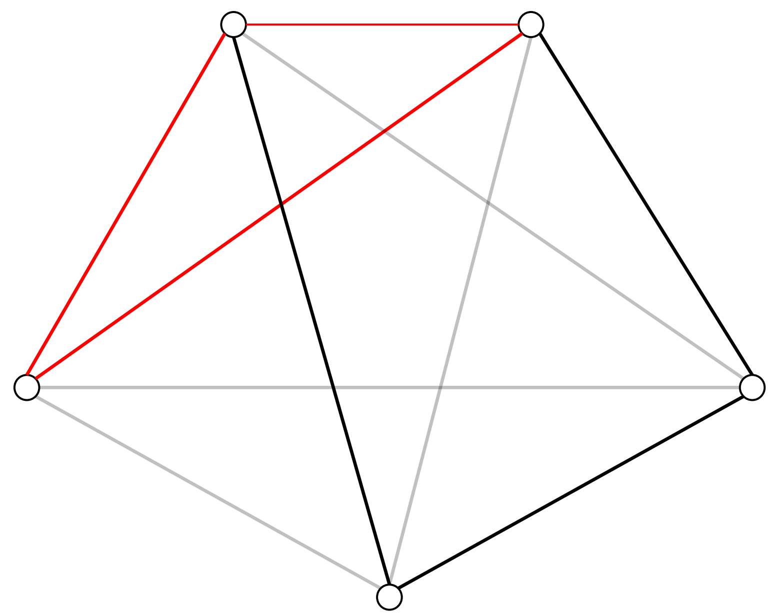


- O(log log n) rounds
- no non-trivial lower bound known

What We Know: Triangle Detection

input: adjacent edges in input graph

output: whether input contains triangle



Dolev et al.
DISC'12

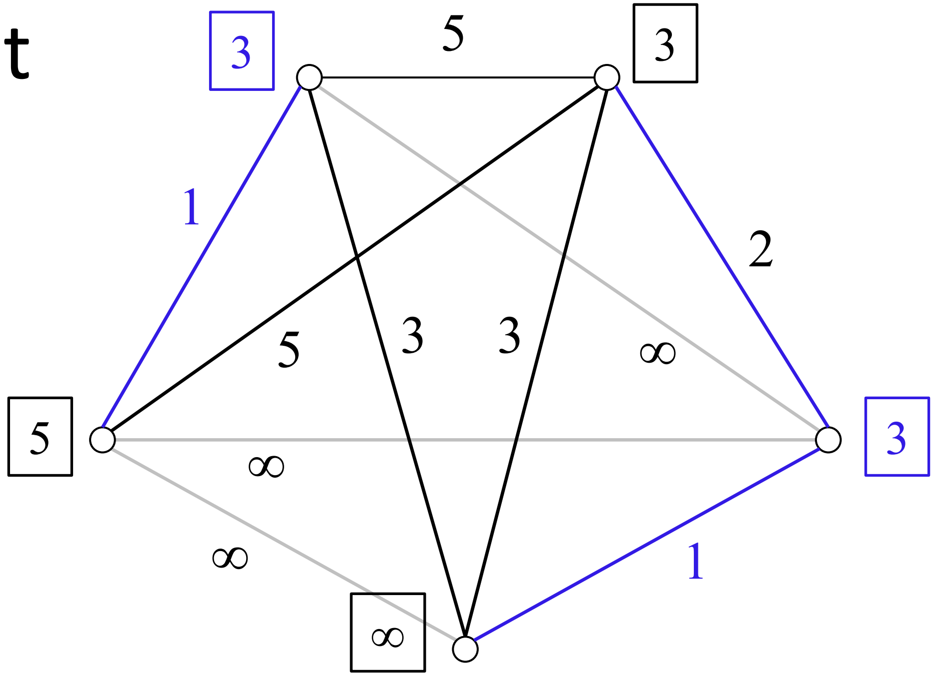
- $O(n^{1/3}/\log n)$ rounds
- no non-trivial lower bound known

What We Know: Metric Facility Location

input: costs for nodes & edges (metric)

output: nodes & edges s.t. selected nodes cover all

goal: mininimize cost

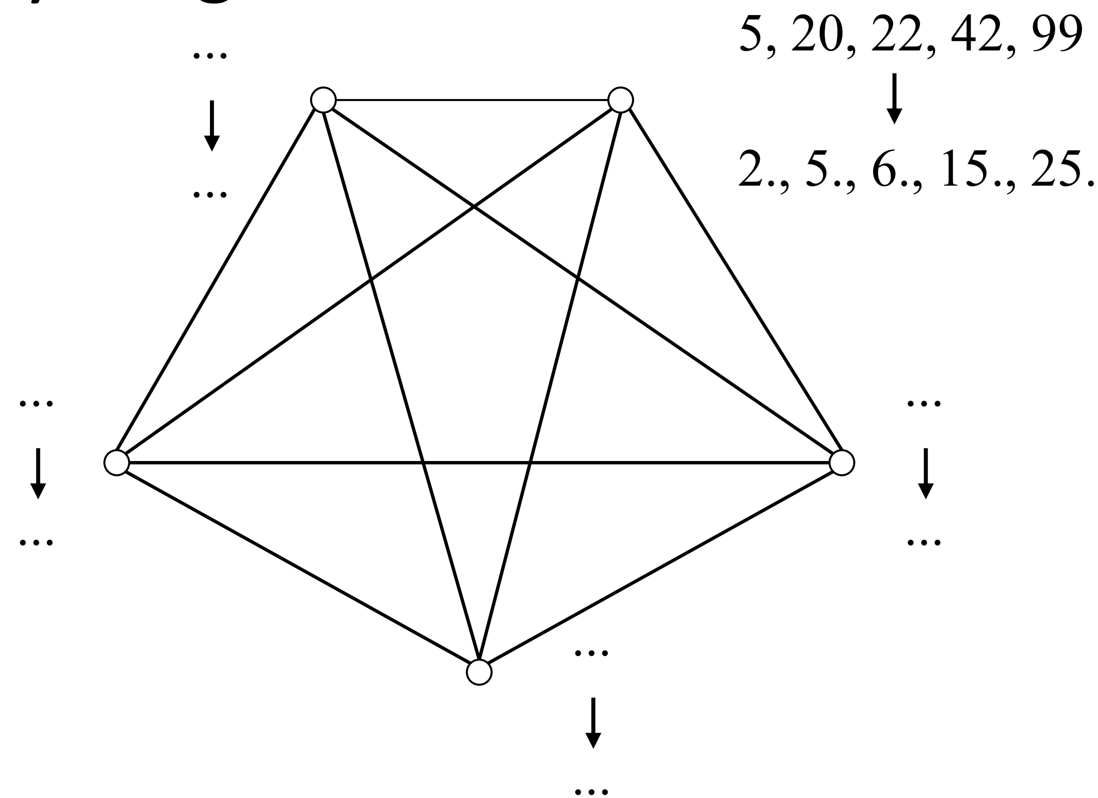


- O(log log n log* n) rounds for O(1)-approx.
- no non-trivial lower bound known

What We Know: Sorting

input: n keys/node

output: indices of keys in global order



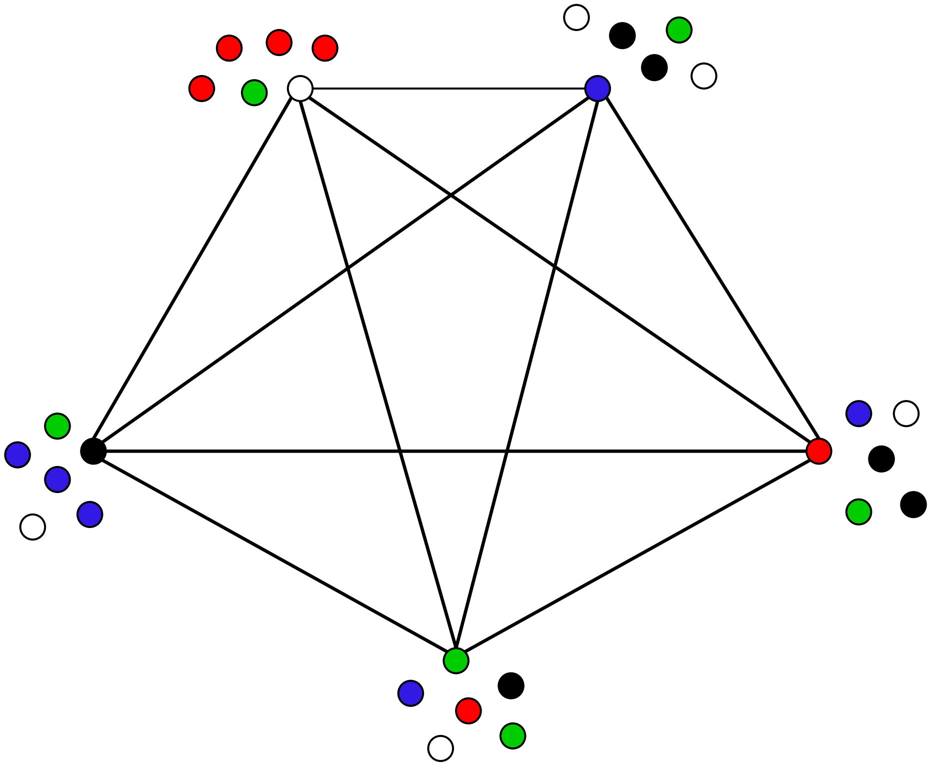
PODC'13

- O(1) rounds
- trivially optimal

What We Know: Routing

input: n mess./node, each node dest. of n mess.

goal: deliver all messages

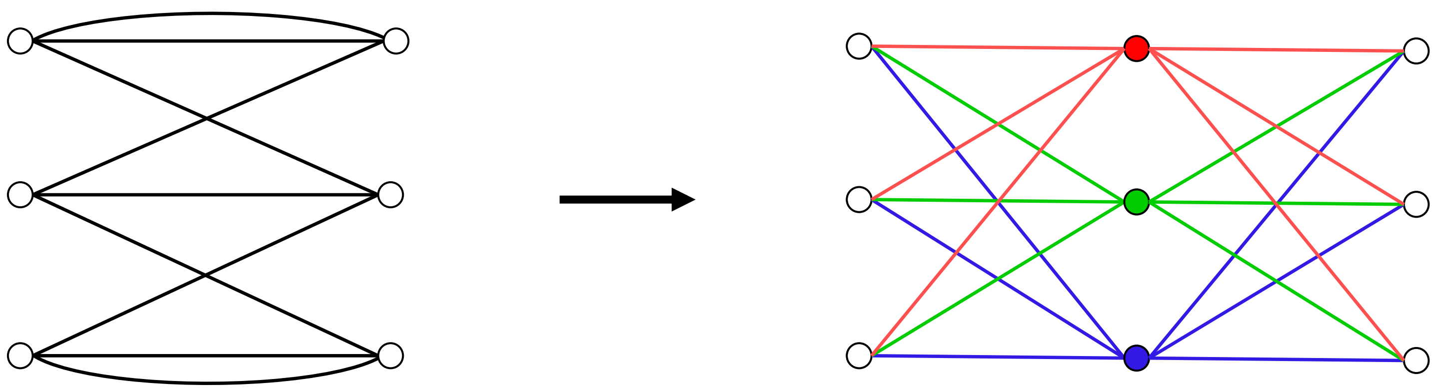


- O(1) rounds
- trivially optimal

Routing: Known Source/Destination Pairs

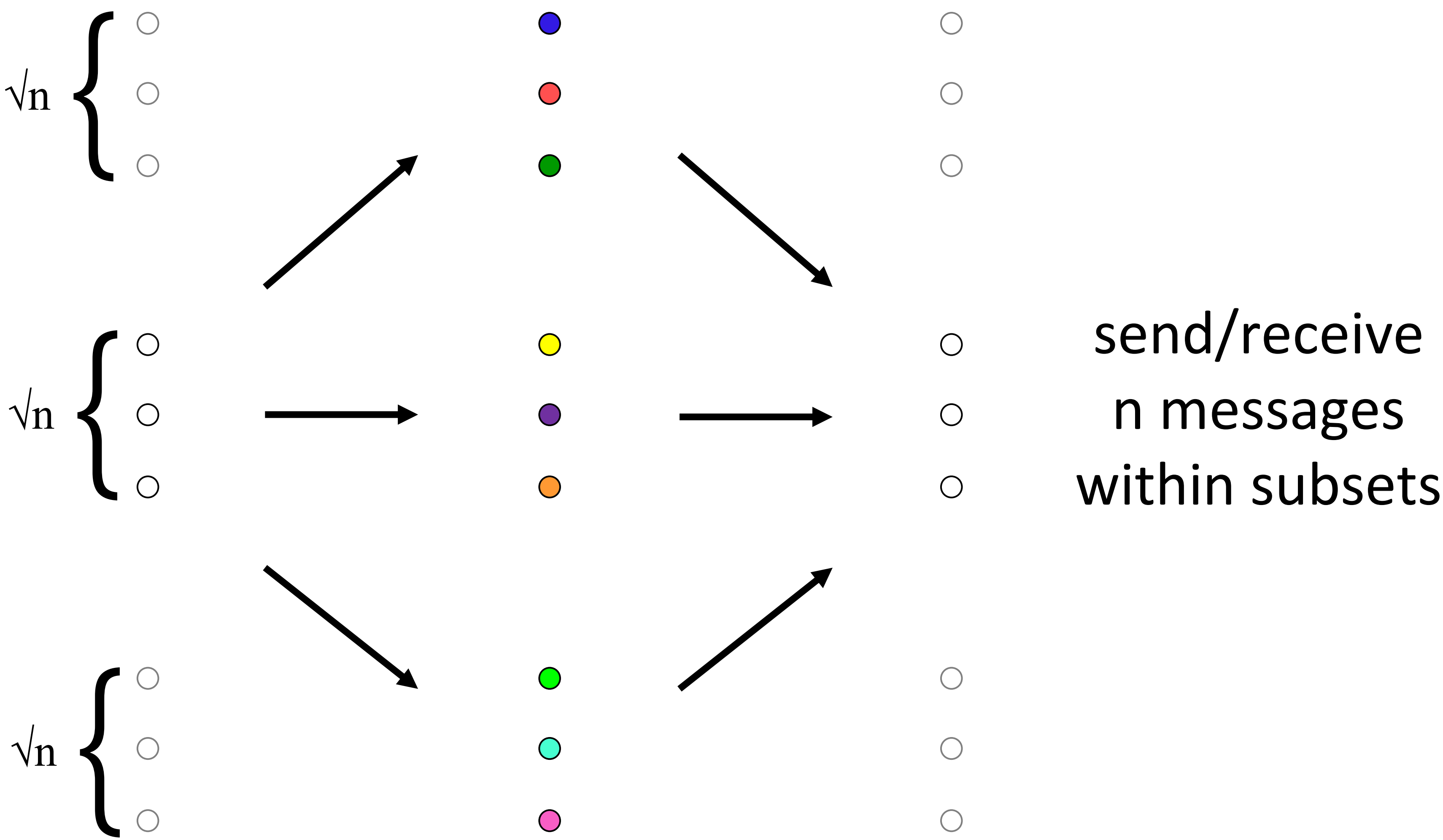
input: n messages/node (each to receive n mess.)

source/destination pairs common knowledge



“sources” “destinations” 2 rounds

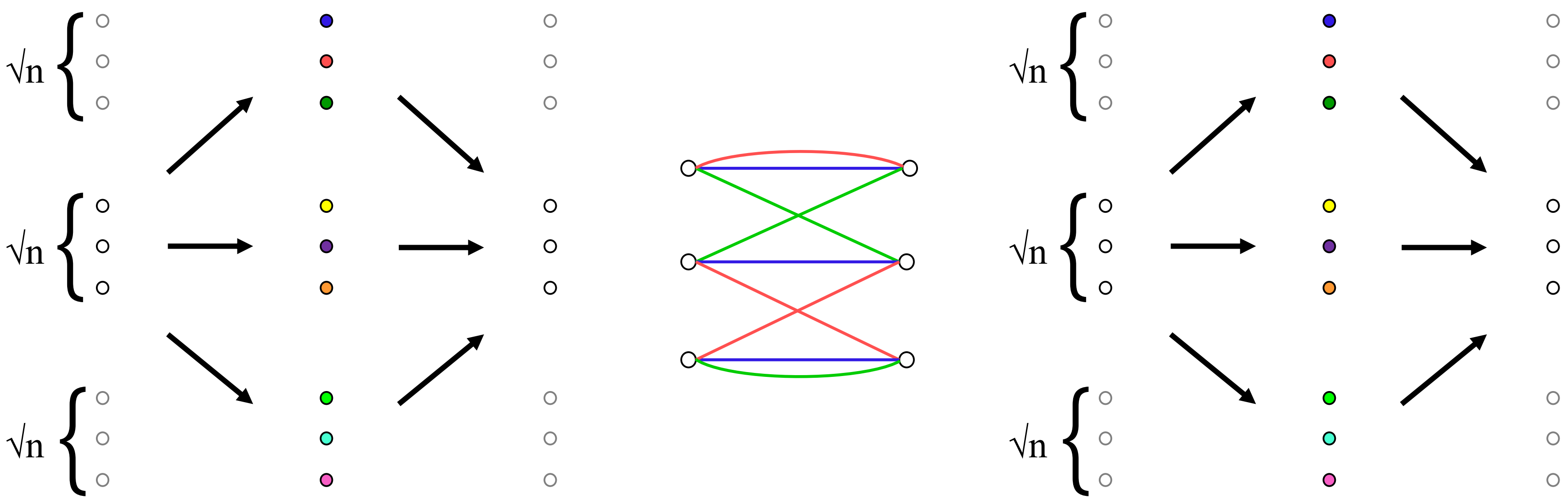
Routing within Subsets (Known Pairs)



Routing within Subsets (Unknown Pairs)

Within each subset:

1. Broadcast #mess. for each destination	2 rounds
2. Compute communication pattern	local comp.
3. Move messages	2 rounds

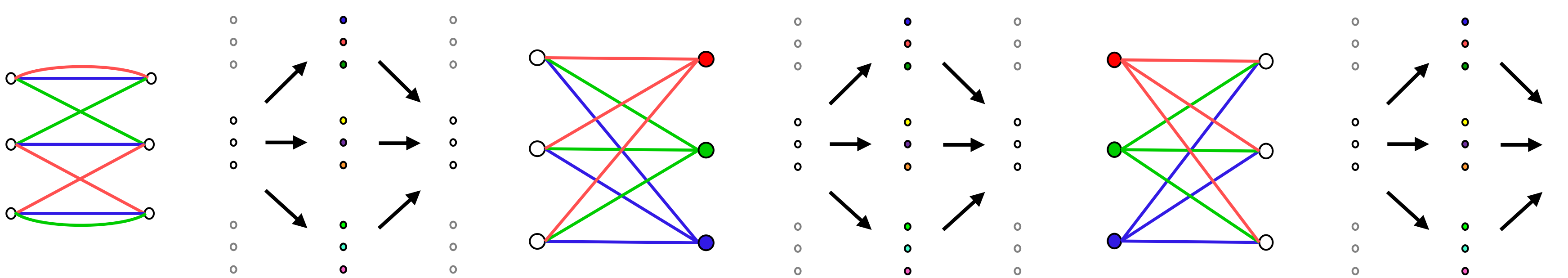


Routing: Known Source/Destination Sets

1. Compute pattern on set level	local comp.
2. Redistribute messages within sets	4 rounds
3. Move messages between sets	1 round
4. Redistribute messages within sets	4 rounds
5. Move messages between sets	1 round
6. Deliver messages within sets	4 rounds

- $n^{1/2}$ supernodes
- degree $n^{3/2}$
- n links between sets
- n mess. between each pair

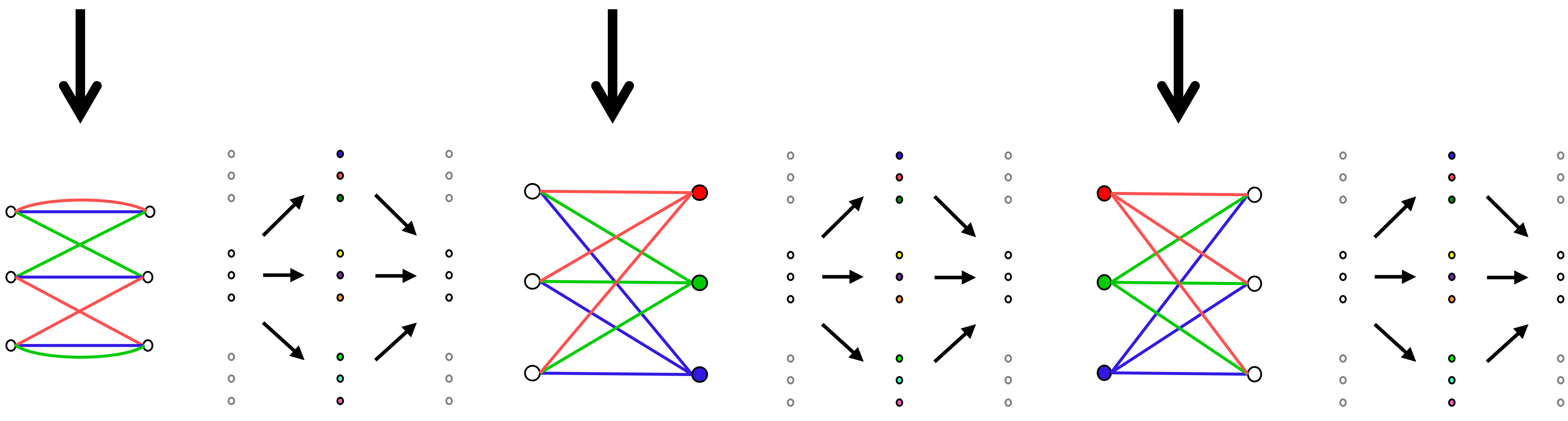
each pair can handle n mess.



Routing: Unknown Pairs

source/destination pairs
only relevant w.r.t. sets

count within sets (one node/dest.) 1 round
broadcast information to all nodes 1 round



Routing: Result

Theorem

Input:

- up to n messages at each node
- each node destination of up to n messages

Then:

- all messages can be delivered in 16 rounds

...or in Other Words:

fully connected CONGEST ≈ bulk-synchronous (bandwidth n log n)

in each round, each node

1. computes
2. sends up to n log n bits
3. receives up to n log n bits

What Do We Want in a Lower Bound?

- caused by “lack of coordination”, not bottleneck

→ input per node of size O(n log n)

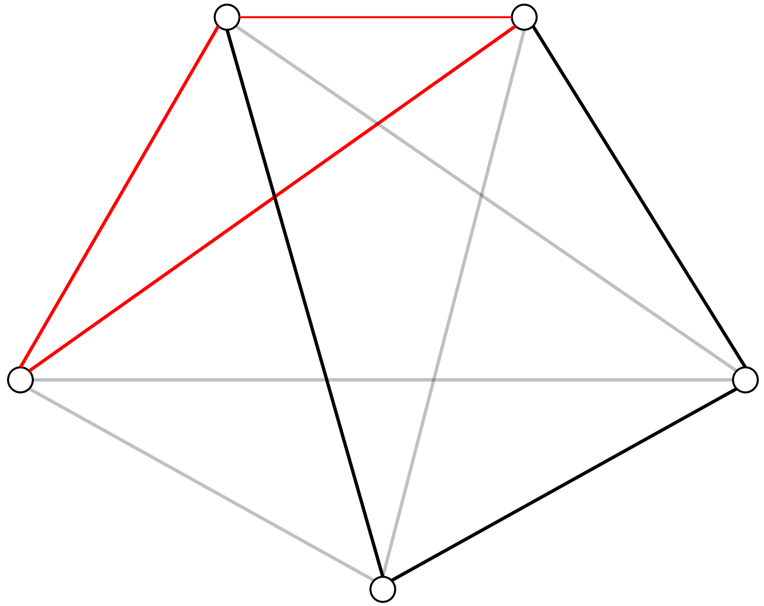
ideally, also:

- “natural” problem
- strong bound (e.g. $\Omega(n^c)$ for constant $c>0$)
- unrestricted algorithms

Triangle Detection: an Algorithm

input: adjacent edges in input graph

output: whether input contains triangle

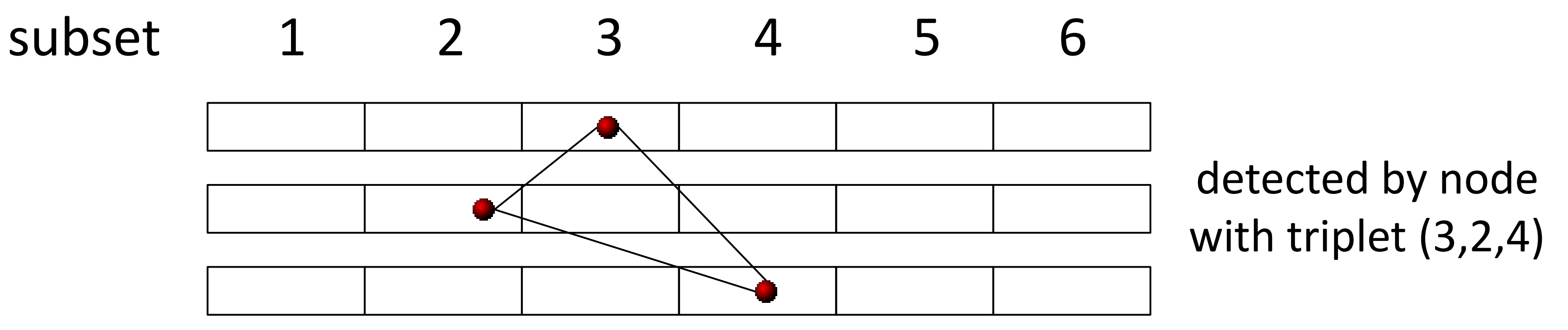


Triangle Detection: an Algorithm

- partition nodes into subsets of $n^{2/3}$ nodes
- consider all n triplets of such subsets
- assign triplets 1:1 to nodes
- responsible node checks for triangle in its triplet

→ needs to learn of $n^{4/3}$ (pre-determined) edges

→ running time $O(n^{1/3}/\log n)$



Triangle Detection: an Algorithm

"oblivious" algorithm:

- fixed message pattern
- computation only initially and in the end

<u>Conjecture</u>

running time $O(n^{1/3}/\log n)$

optimal for oblivious algorithms

…and maybe even in general?

MST and Friends

some doubly logarithmic bounds:

- MST in O(log log n) rounds
- Metric Facility Location in O(log log n log* n) rounds
- no improvement or lower bound on MST for a decade

Open Question

Is running time O(log log n) a barrier for some problems?

Connectivity

input: adjacent edges in input graph
output: whether input graph is connected

- natural problem, even simpler than MST
- might be easier to find right approach

<u>Open Question</u>

Can Connectivity be decided within O(1) rounds?

...on a Related Subject

There *is* a lower bound, on Set Disjointness!
(but in a different model)

→ Don‘t miss the next talk!

...thank you for your attention!