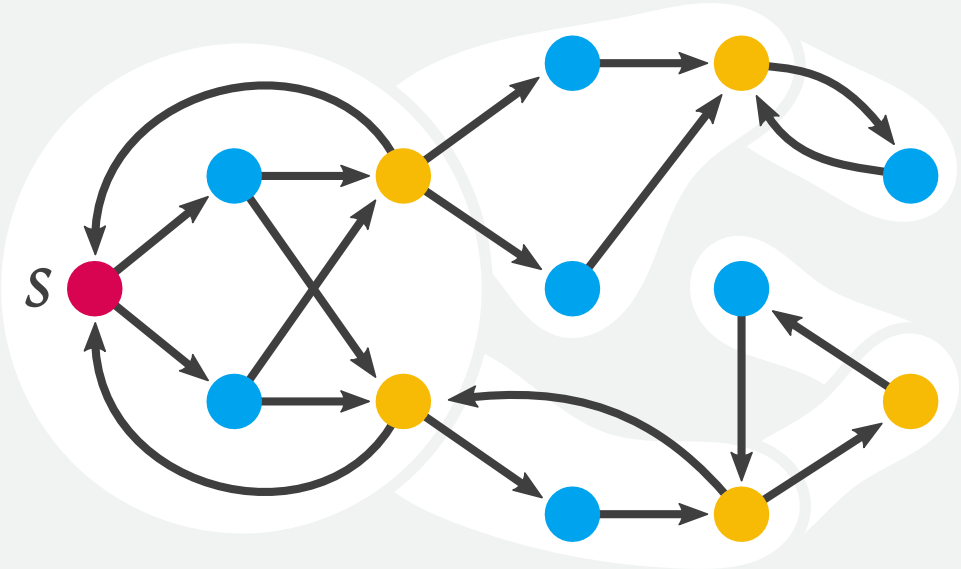
k-Distinct In- and Out-Branchings

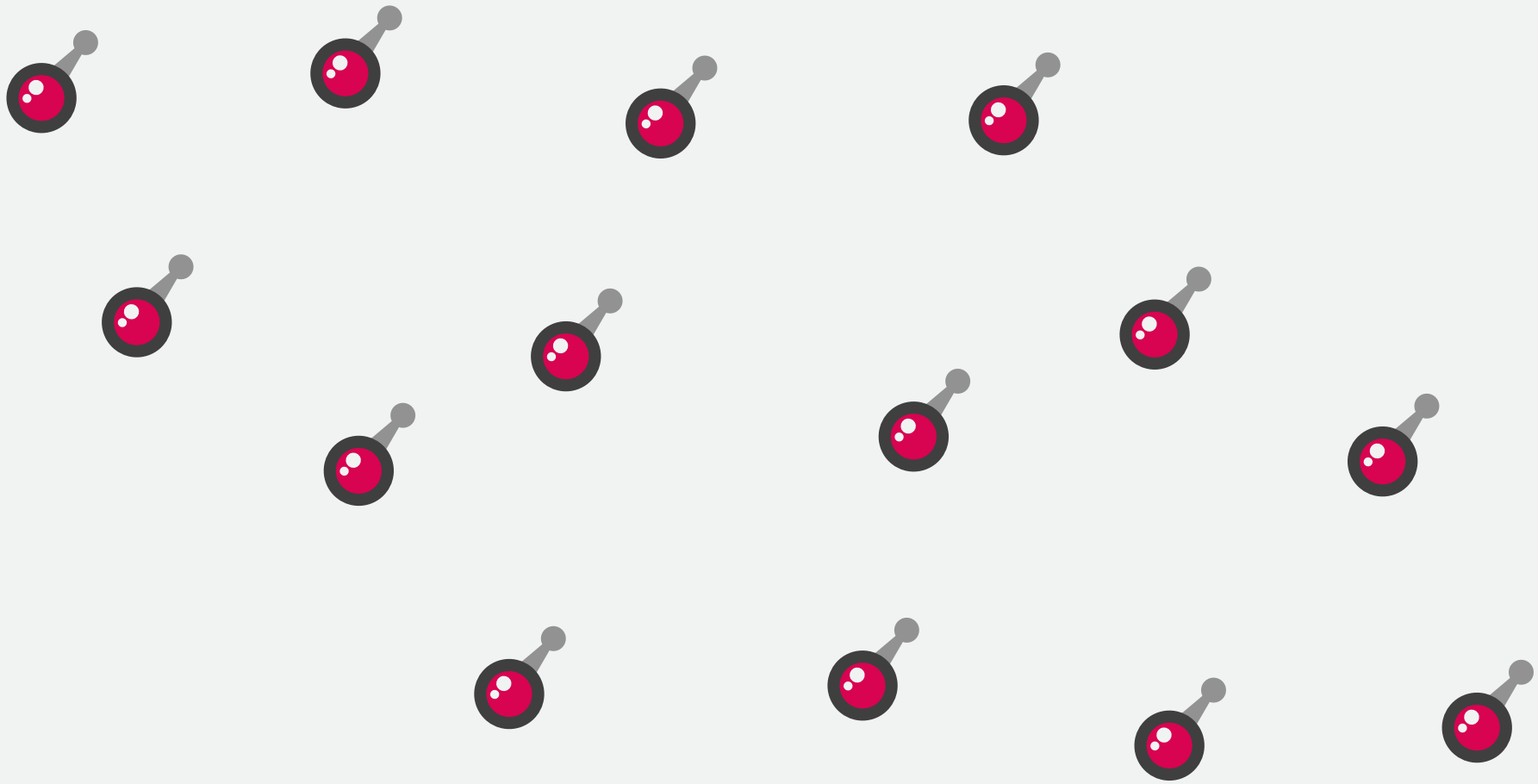
Gregory Gutin[1] **Felix Reidl**[1,2] Magnus Wahlström[1]



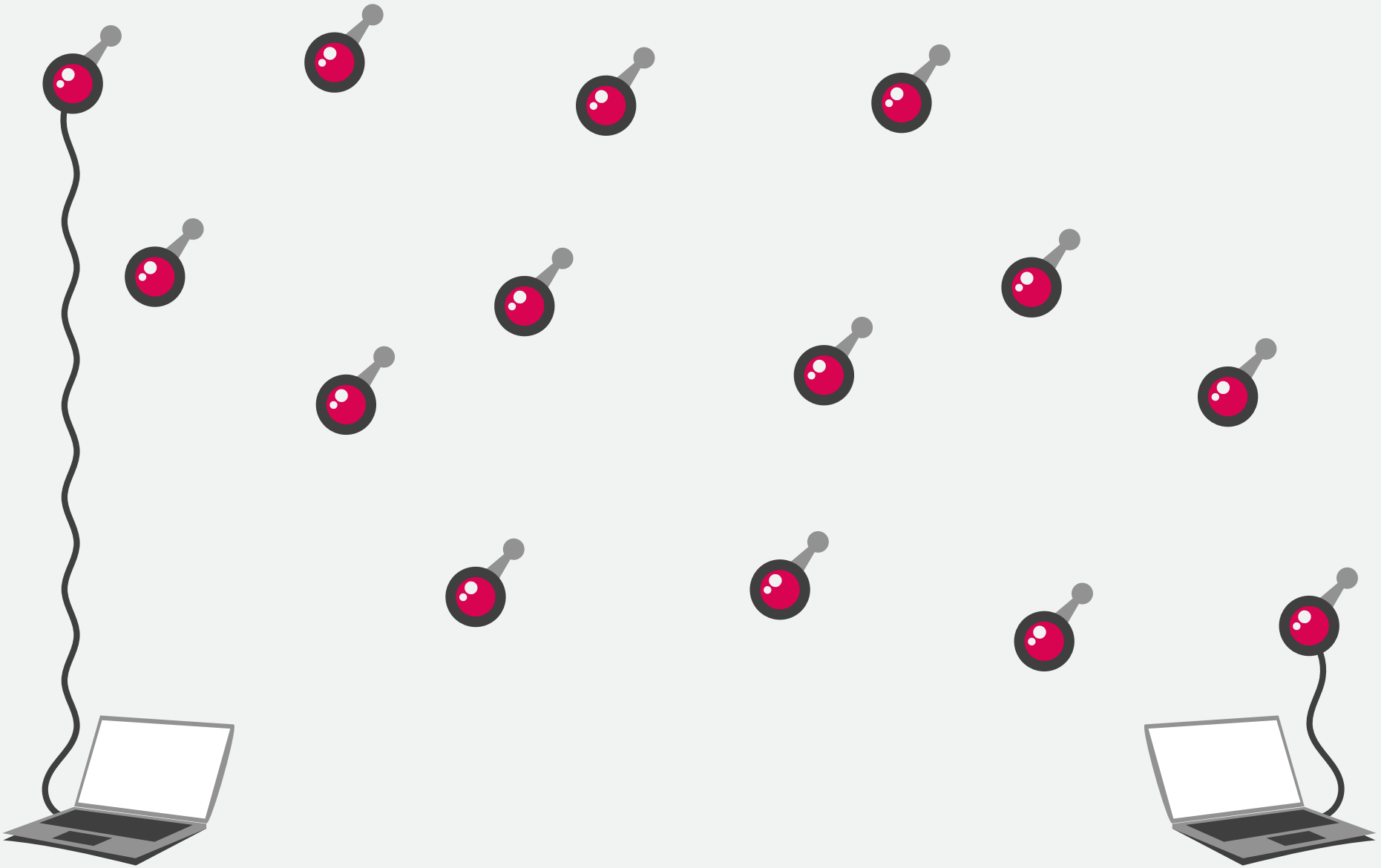
[1]Royal Holloway
University of London, UK

[2]North Carolina
State University, USA

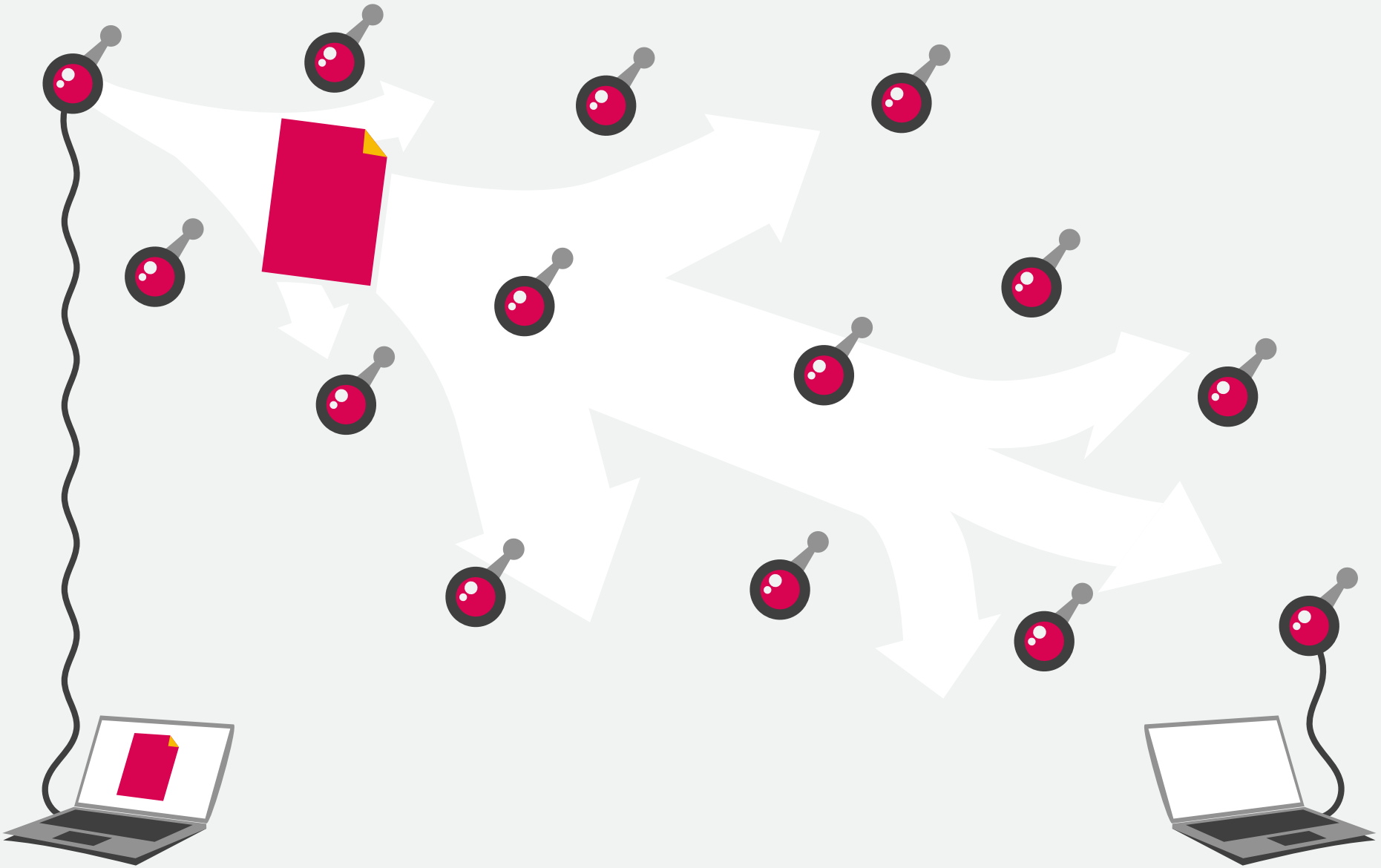
Routing in Sensor Networks



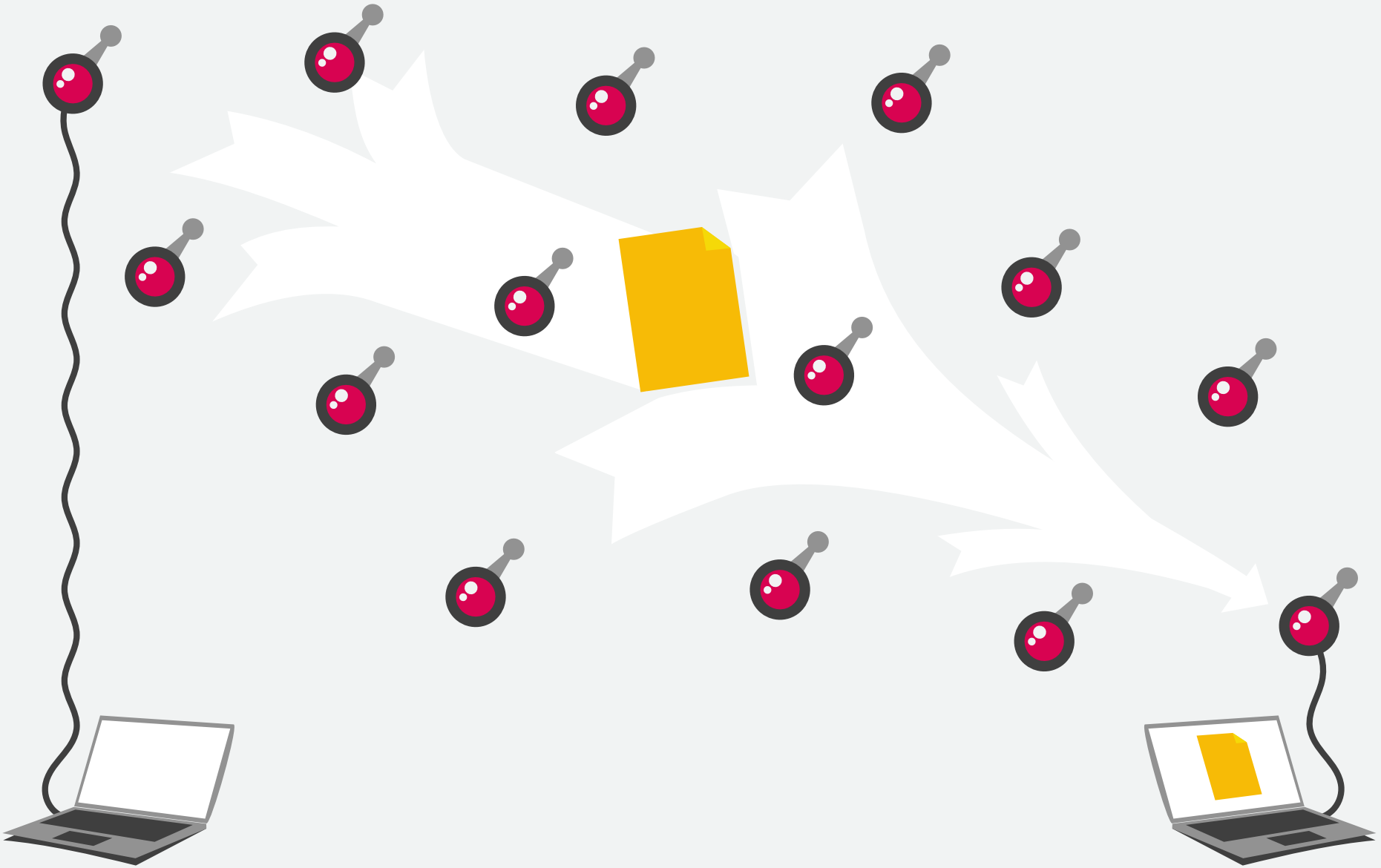
Routing in Sensor Networks



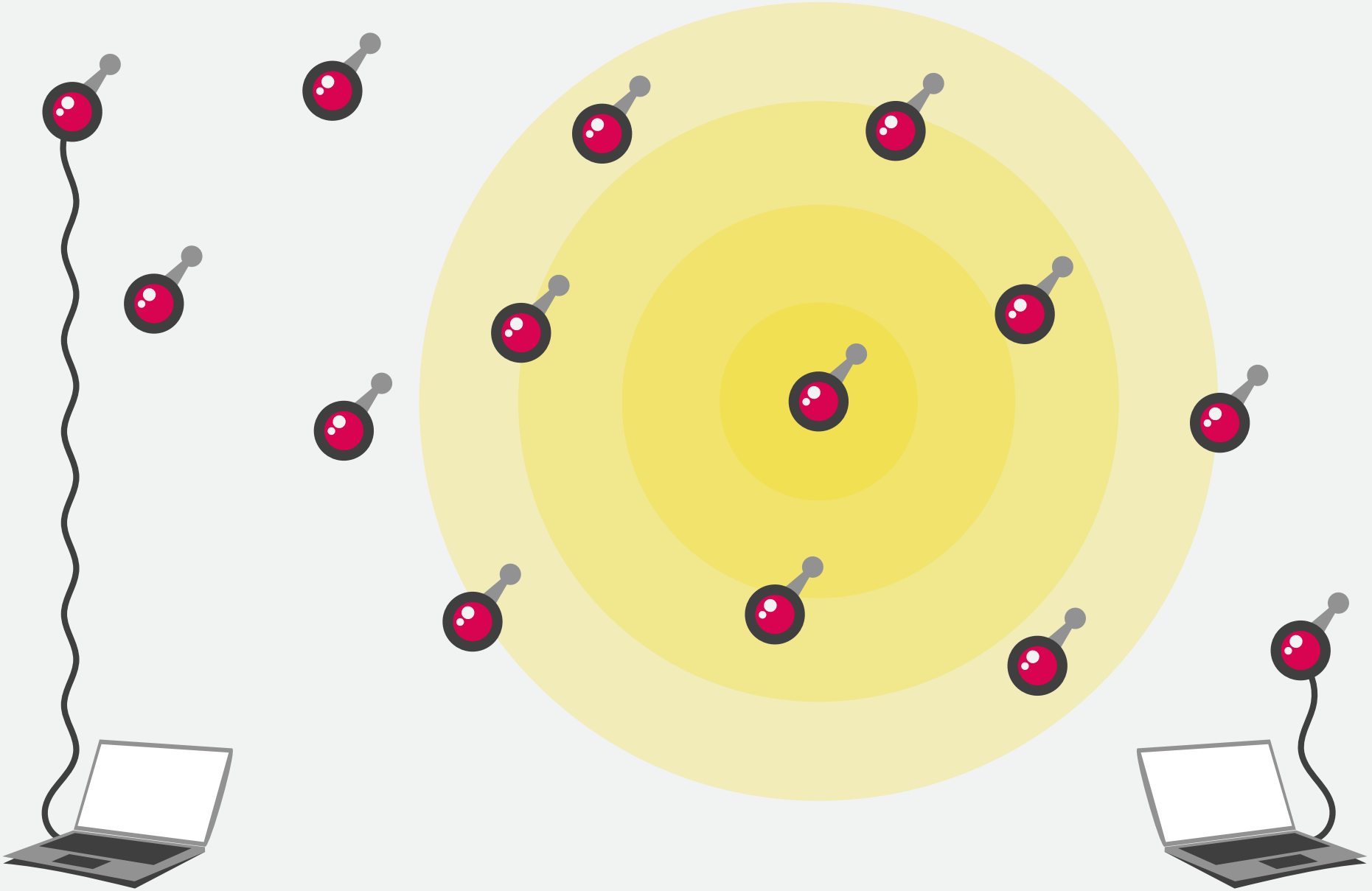
Routing in Sensor Networks



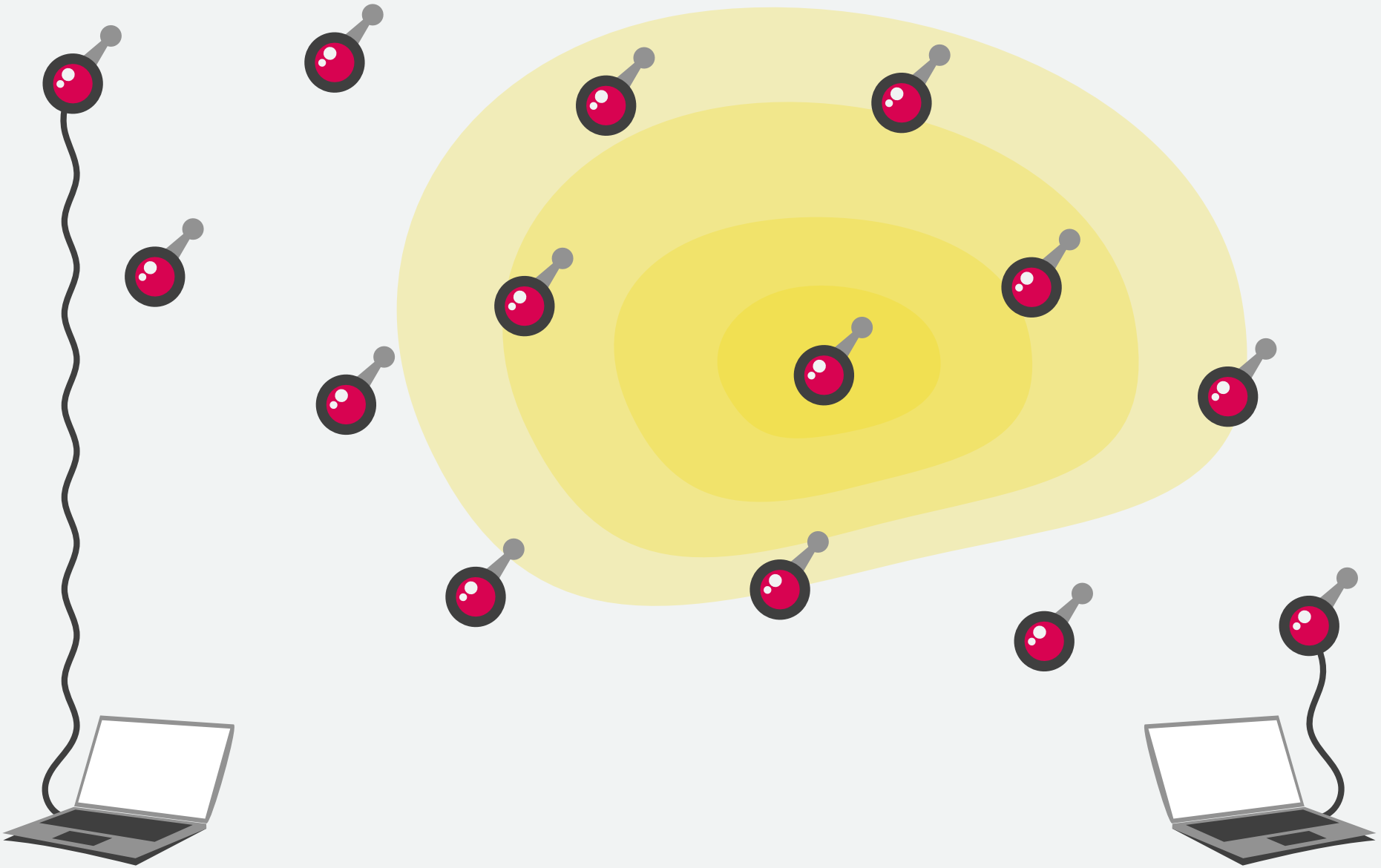
Routing in Sensor Networks



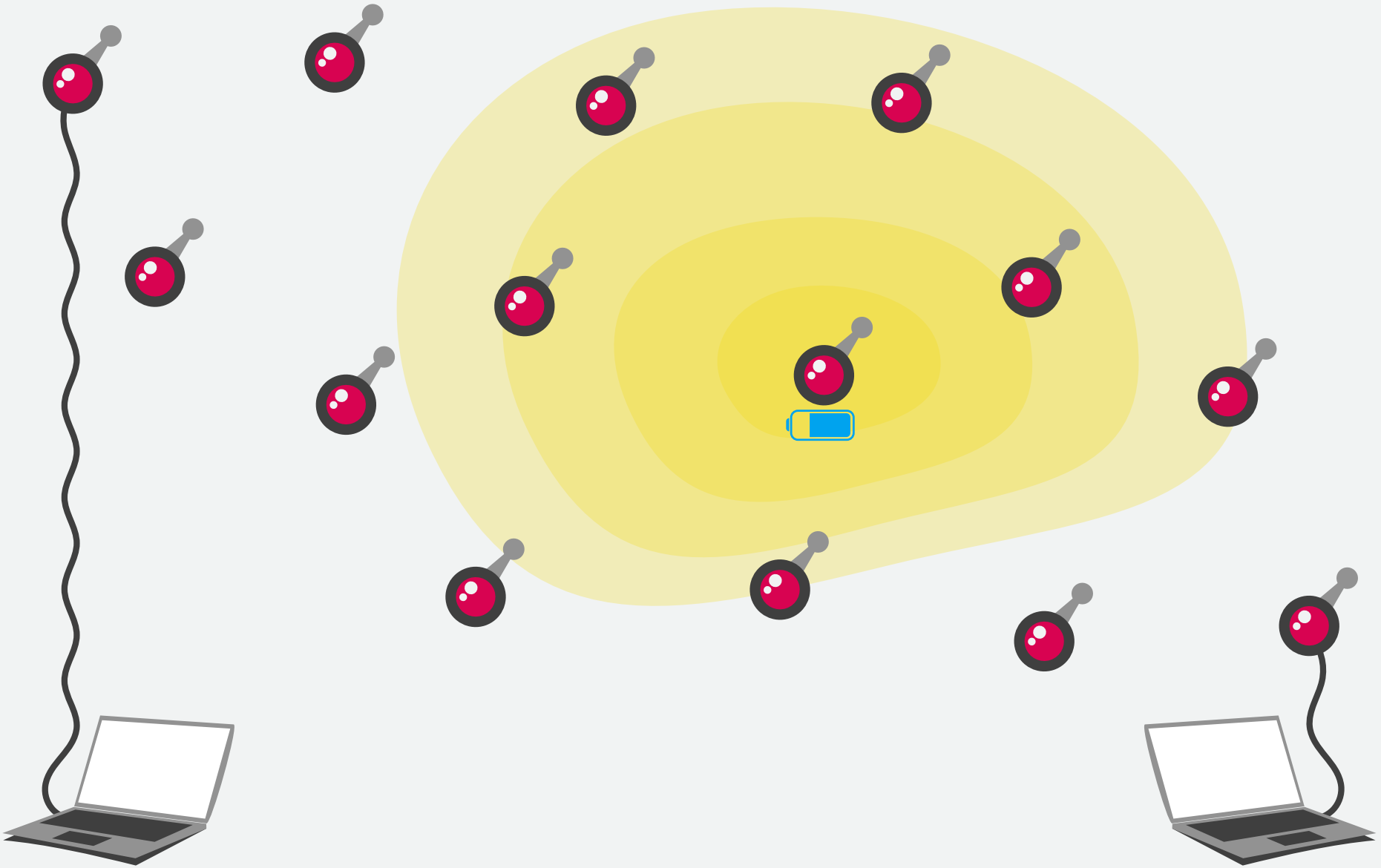
Routing in Sensor Networks



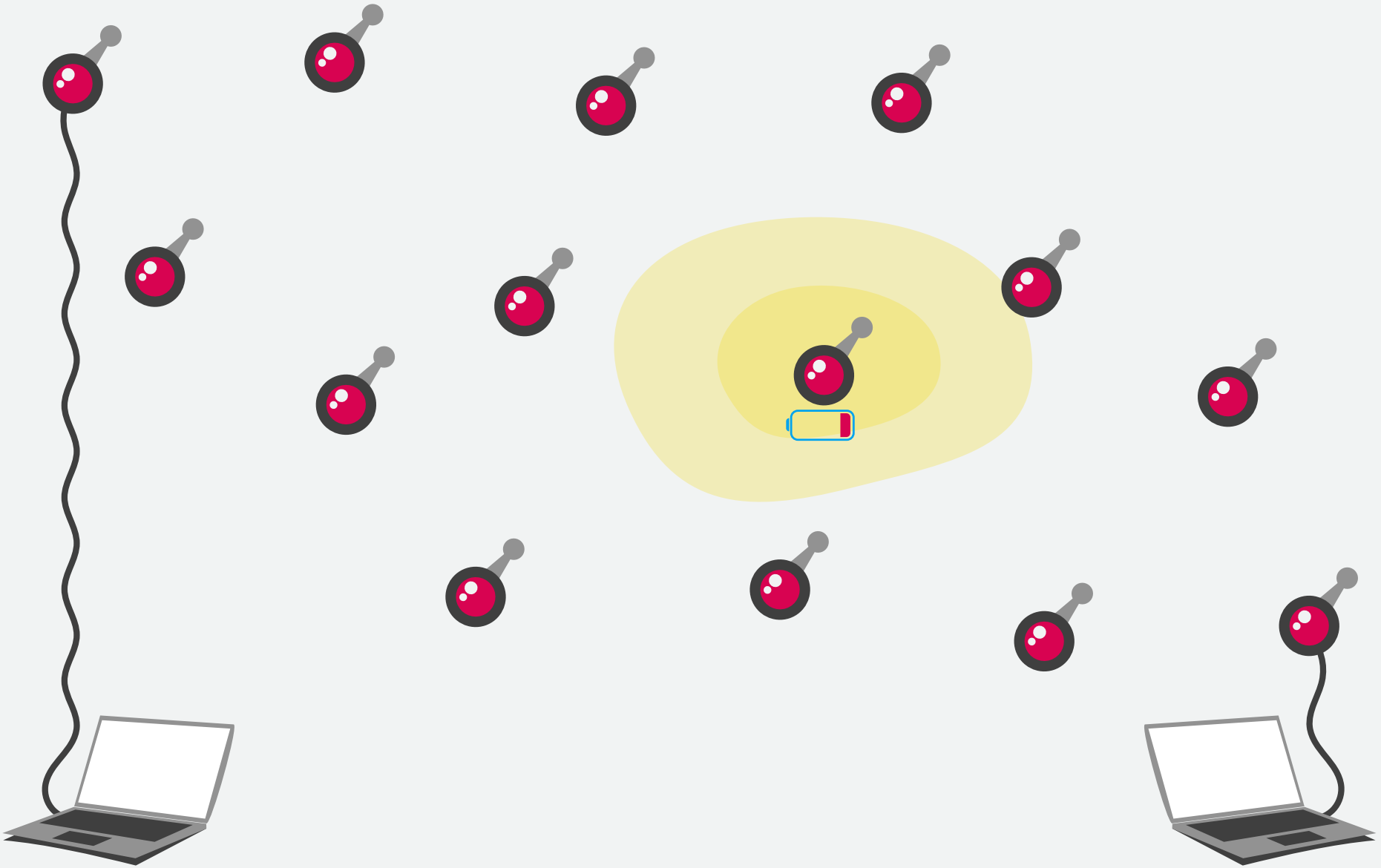
Routing in Sensor Networks



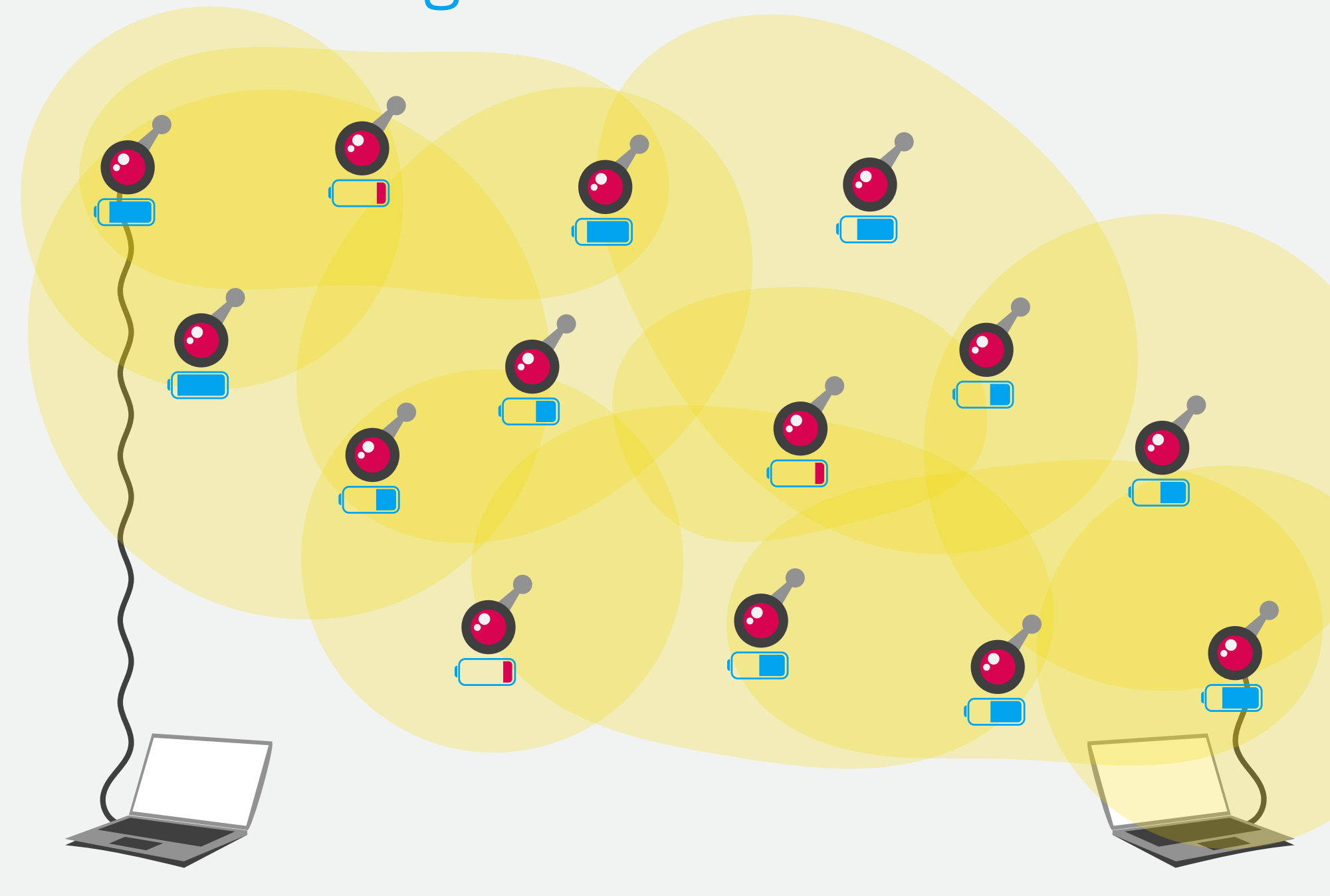
Routing in Sensor Networks



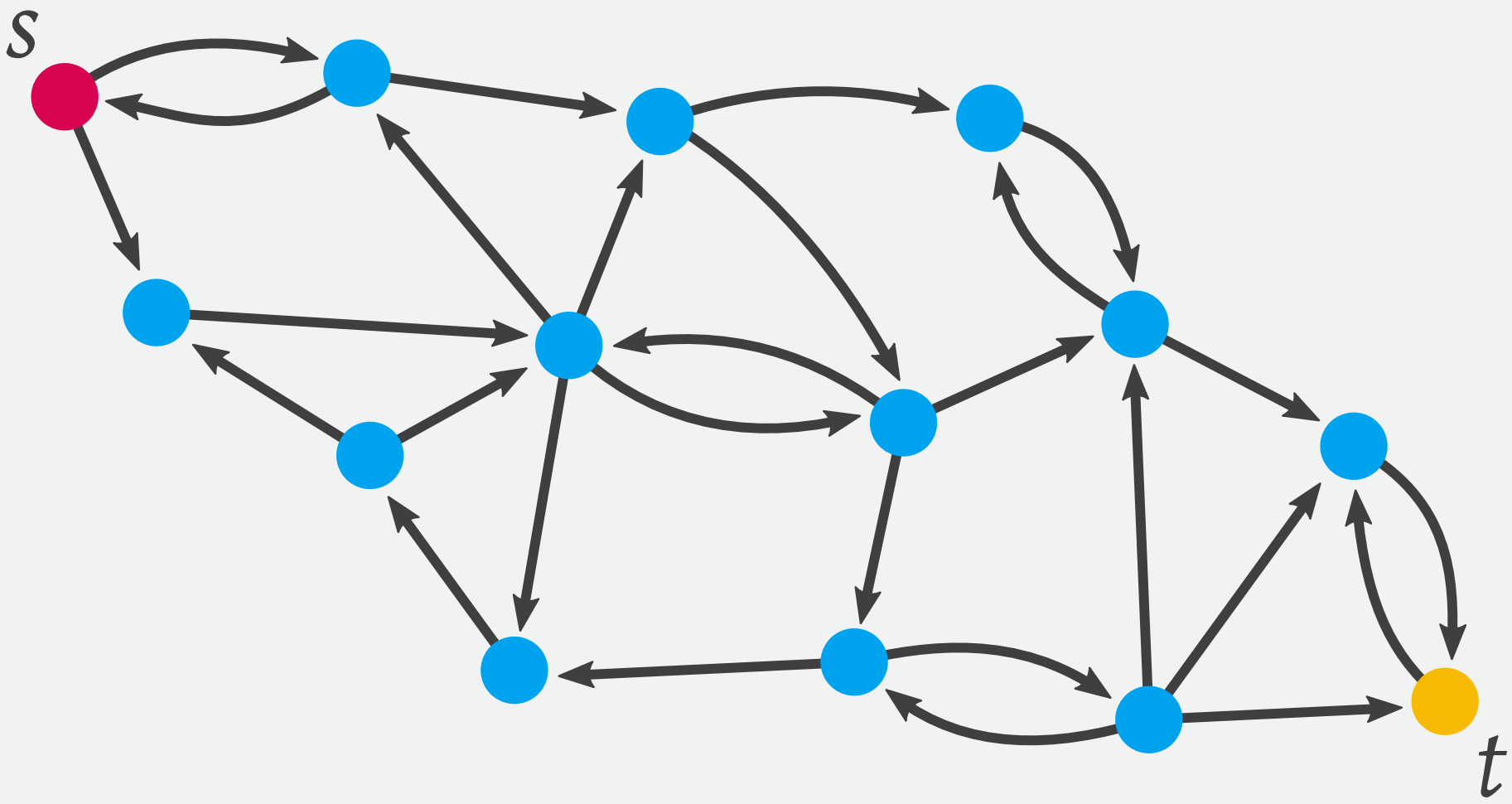
Routing in Sensor Networks



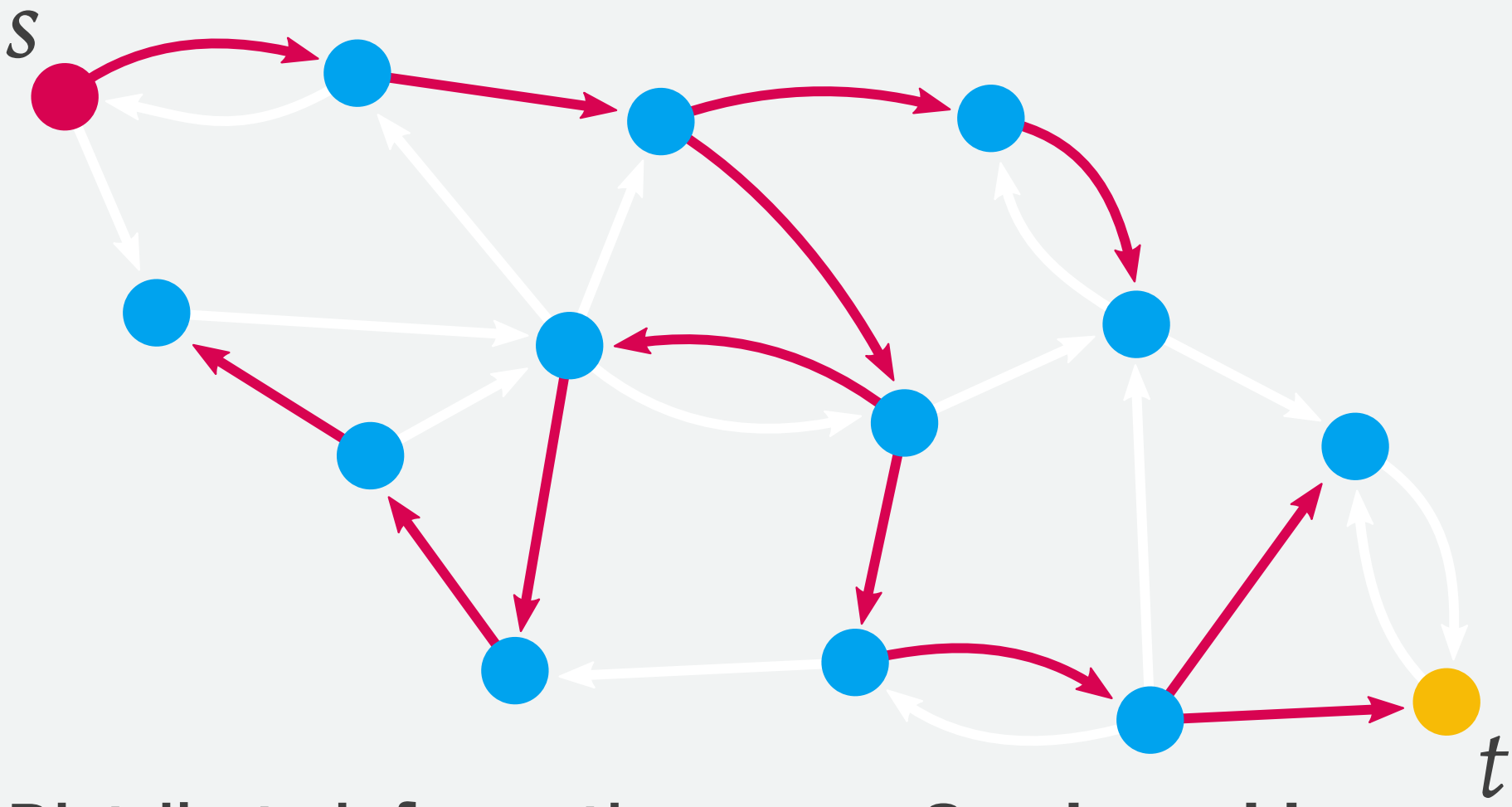
Routing in Sensor Networks



Routing in Sensor Networks



Routing in Sensor Networks

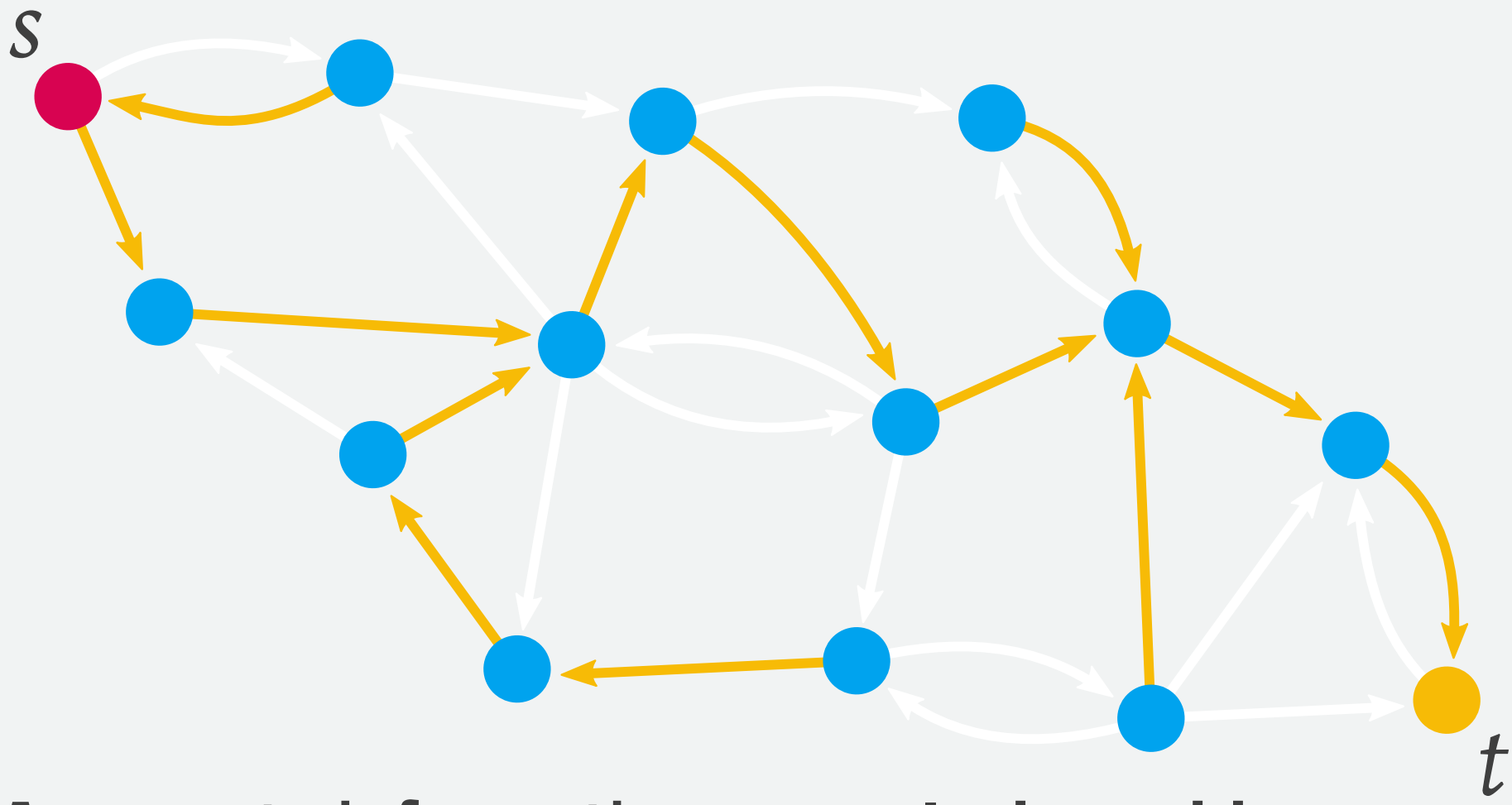


Distribute information from node s

≙

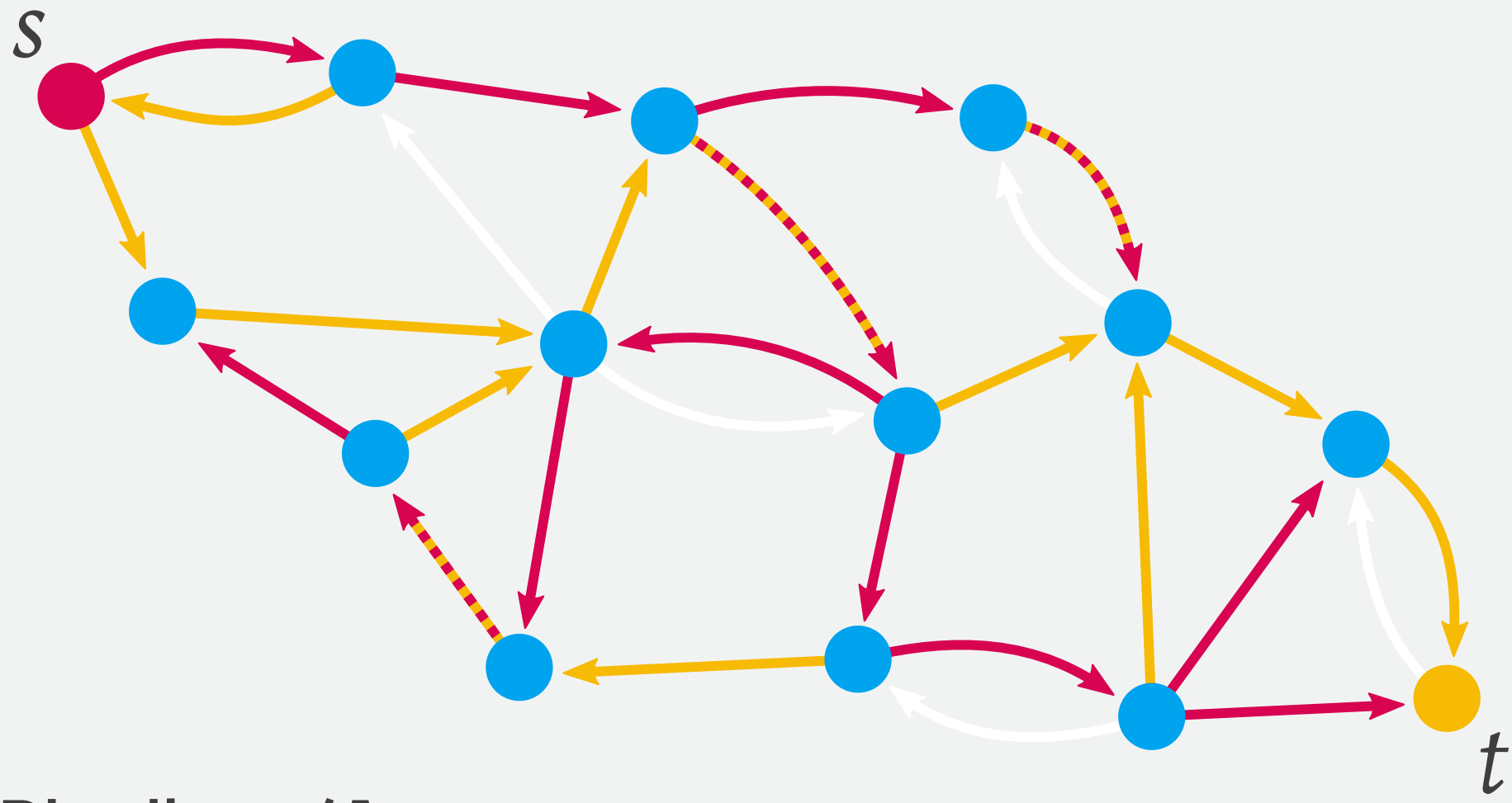
Out-branching with root s

Routing in Sensor Networks



Aggregate information at node t ≙ **In-branching with root t**

Routing in Sensor Networks



Distribute/Aggregate with little crosstalk

$\hat{=}$

In- and Out-branching with few common arcs

Distinct Branchings

An out-branching T^+ and and in-branching T^- are called k-distinct if $\left|A(T^+) \setminus A(T^-)\right| \geq k$.

k-DISTINCT BRANCHINGS

Input: A digraph D, an integer k.

Question: Does D contain k-distinct in- and out-branchings?

Previous work

- **NP-complete to decide whether arc-disjoint in- and out-branchings exist**

  Bang-Jensen J. Edge-disjoint in-and out-branchings in tournaments and related path problems.
  Journal of Combinatorial Theory, Series B. 1991 Jan 1;51(1):1-23.

- **FPT in strong digraphs, parameterized by k**

  Bang-Jensen J, Saurabh S, Simonsen S. Parameterized algorithms for non-separating trees and branchings in digraphs.
  Algorithmica. 2016 Sep 1;76(1):279-96.

- **FPT in general digraphs?**

- **FPT if $s = t$?** (SINGLE ROOT k-DISTINCT BRANCHINGS)

  Bang-Jensen J, Yeo A. The minimum spanning strong subdigraph problem is fixed parameter tractable.
  Discrete Applied Mathematics. 2008 Aug 6;156(15):2924-9.

Our result

ROOTED k-DISTINCT BRANCHINGS

Input: A digraph D, integer k, vertices s, t.

Question: Does D contain k-distinct in- and out-branchings rooted at s and t respectively?

Theorem.
ROOTED k-DISTINCT BRANCHINGS **is in** FPT.

Schematic of our result

- **Preprocessing:**
  - **Every arc appears in some rooted in- or out-branching**
  - **D is strongly connected, thus we can search for rooted in- and out-trees**

incurs factor 2 in parameter

Schematic of our result

- **Preprocessing:**
  - **Every arc appears in some rooted in- or out-branching**
  - **D is strongly connected, thus we can search for rooted in- and out-trees**

incurs factor 2 in parameter

- **If D has a rooted out-tree with many leaves, it is a YES-instance**
- **If D has no out-tree with many leaves, it has bounded pathwidth**

Schematic of our result

- **Preprocessing:**
  - **Every arc appears in some rooted in- or out-branching**
  - **D is strongly connected, thus we can search for rooted in- and out-trees**

incurs factor 2 in parameter

- **If D has a rooted out-tree with many leaves, it is a YES-instance**
- **If D has no out-tree with many leaves, it has bounded pathwidth**

Remaining case:

Out-tree with many leaves, but incorrect root

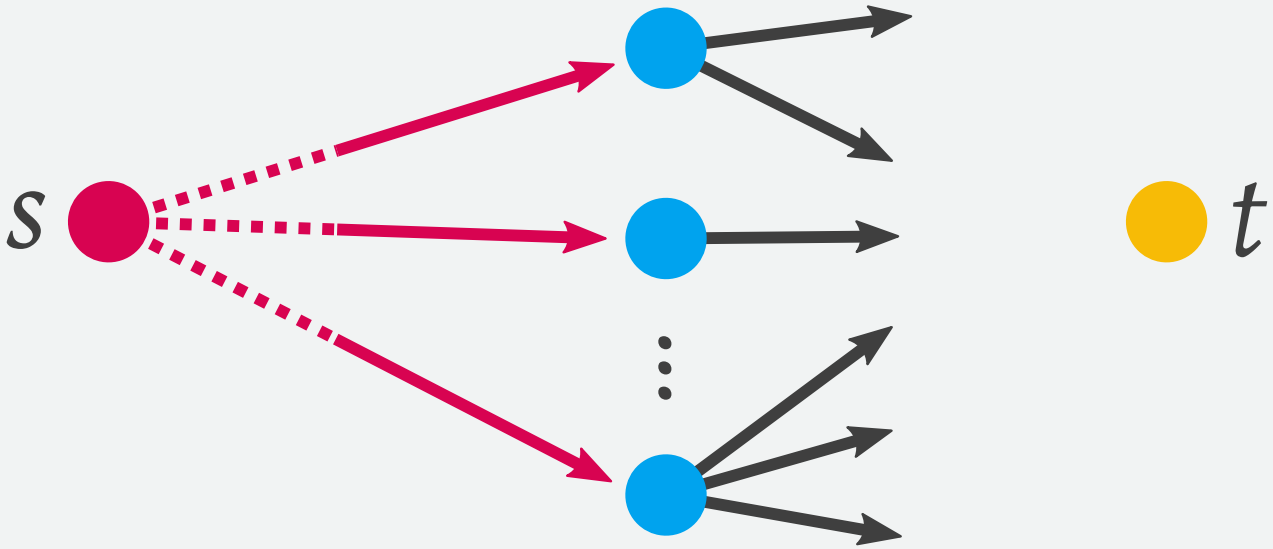
Good case: Many leaves

Lemma.

Assume D contains an in- and out-branching. If D contains an out-branching T^+ with at least $k+1$ leaves, then every in-branching T^- of D is k-distinct from T^+.

Good case: Many leaves

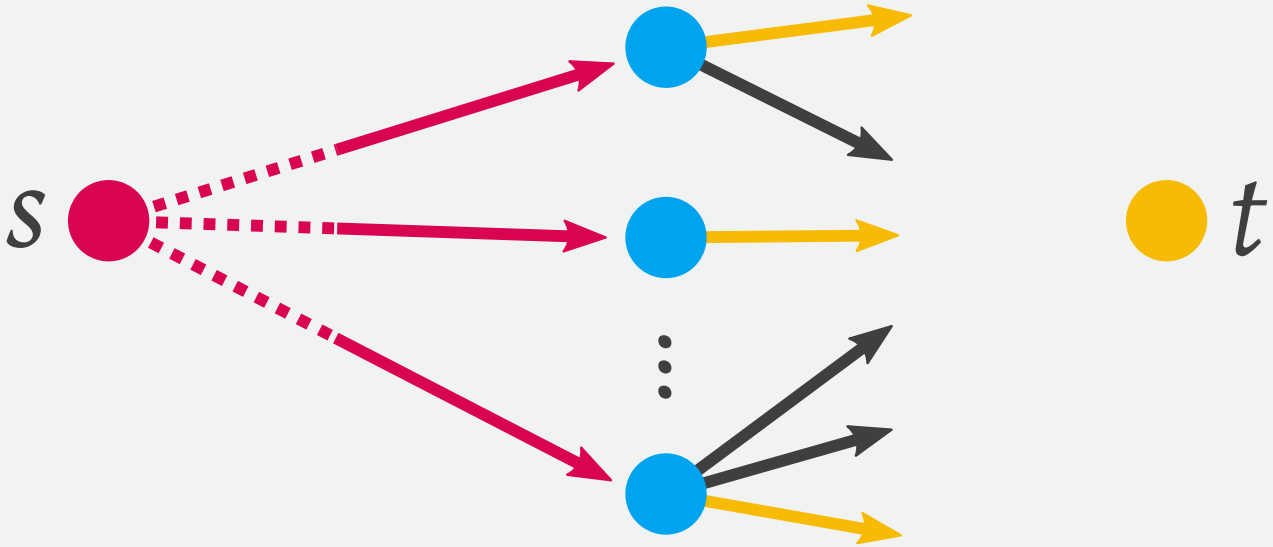
Lemma.
Assume D contains an in- and out-branching. If D contains an out-branching T^+ with at least $k+1$ leaves, then every in-branching T^- of D is k-distinct from T^+.



Good case: Many leaves

Lemma.

Assume D contains an in- and out-branching. If D contains an out-branching T^+ with at least $k+1$ leaves, then every in-branching T^- of D is k-distinct from T^+.



Good case: high connectivity

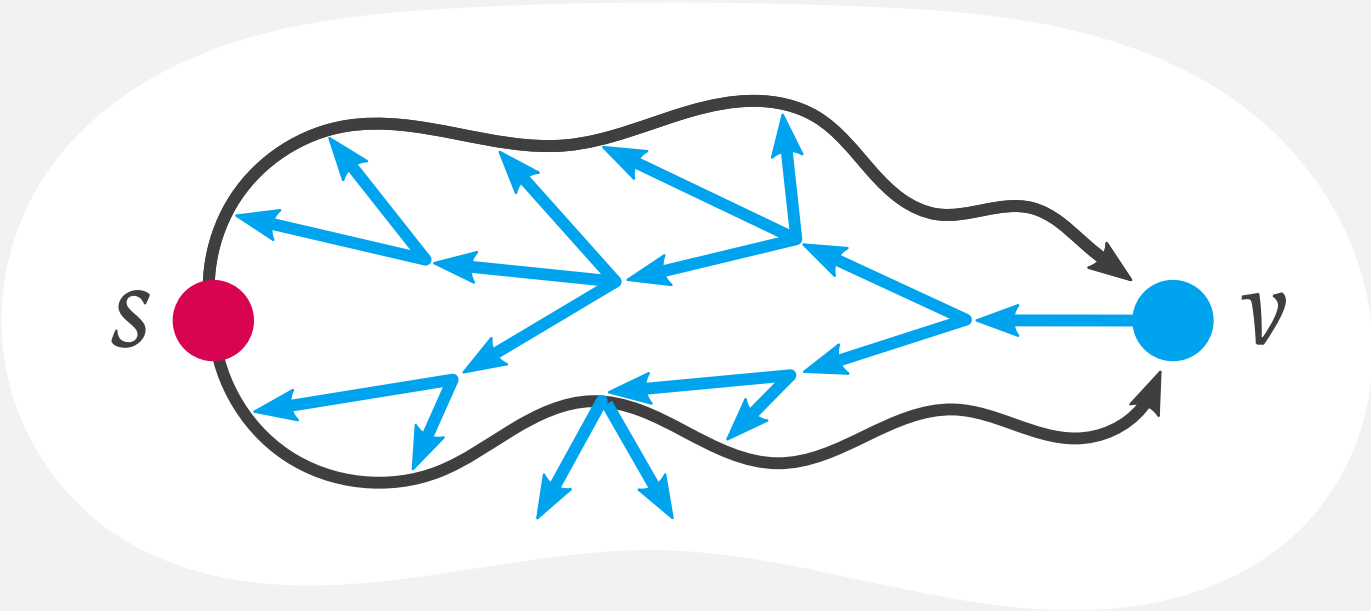
Toy Lemma.
Let T^+ be an out-tree rooted at v with l leaves and assume that every vertex is bi-reachable from s. Then there exists on out-tree rooted at s with at least $l/2$ leaves.

Good case: high connectivity

Toy Lemma.
Let T^+ be an out-tree rooted at v with l leaves and assume that every vertex is **bi-reachable** from s. Then there exists on out-tree rooted at s with at least $l/2$ leaves.

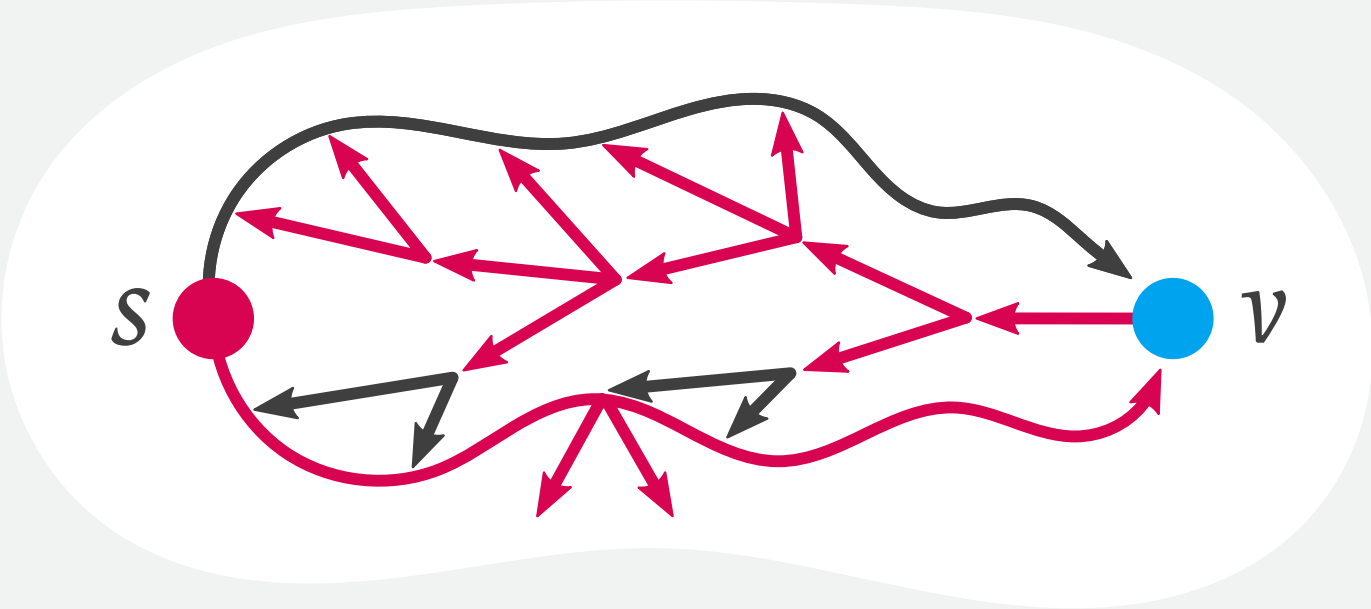
Good case: high connectivity

Toy Lemma.
Let T^+ be an out-tree rooted at v with l leaves and assume that every vertex is bi-reachable from s. Then there exists on out-tree rooted at s with at least $l/2$ leaves.



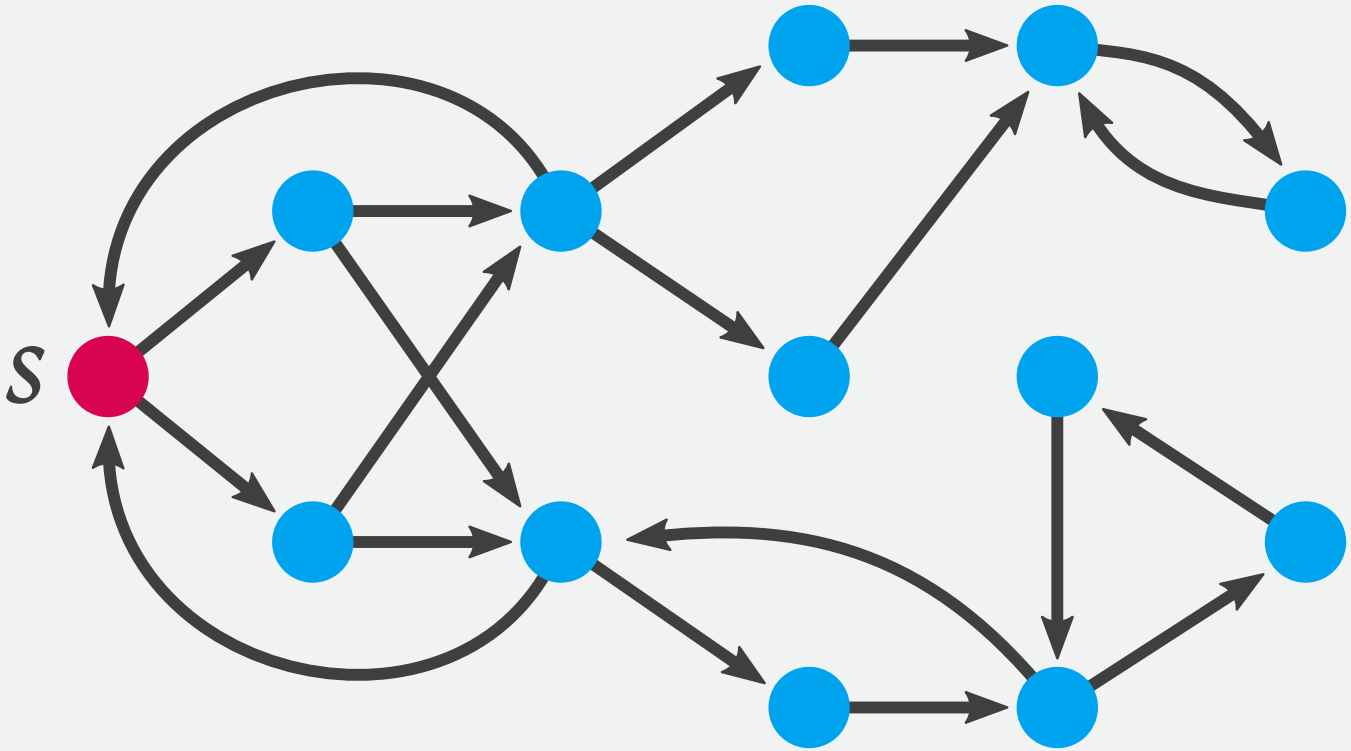
Good case: high connectivity

Toy Lemma.
Let T^+ be an out-tree rooted at v with l leaves and assume that every vertex is bi-reachable from s. Then there exists on out-tree rooted at s with at least $l/2$ leaves.



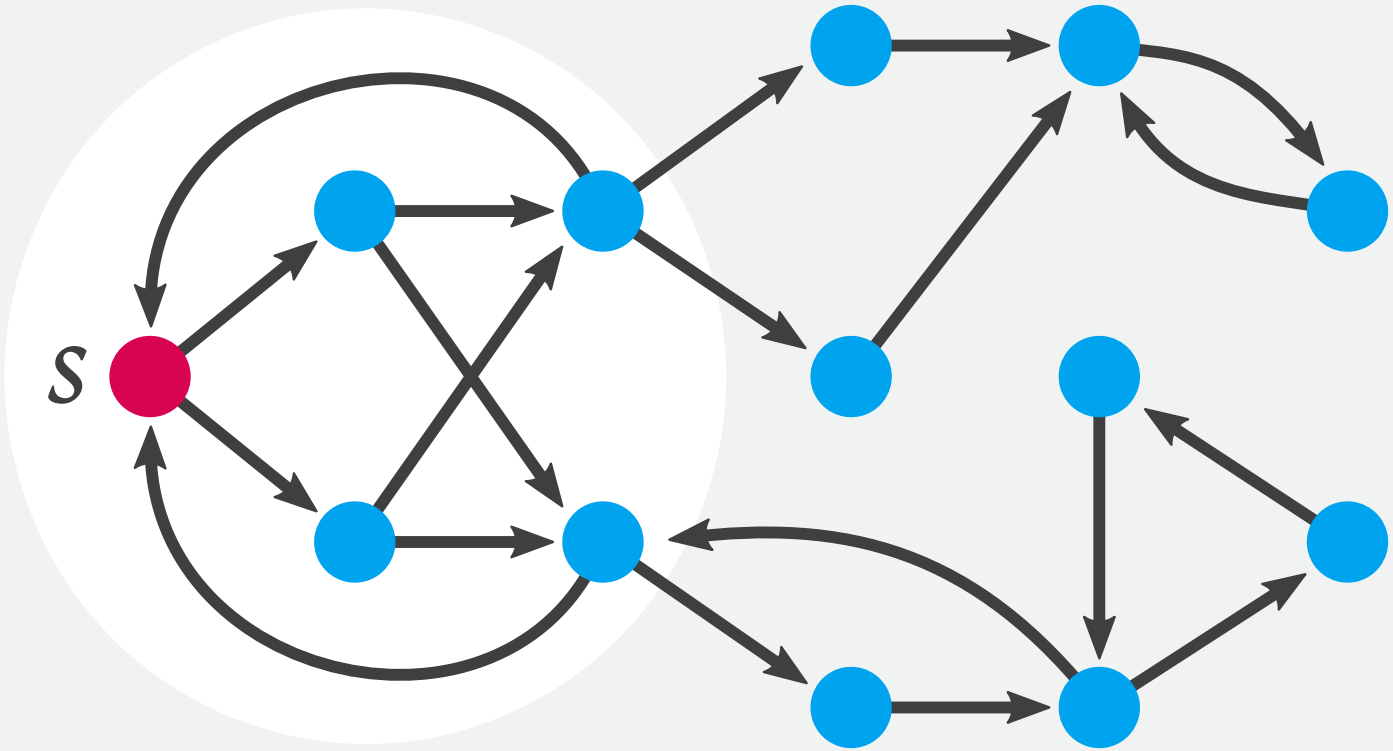
Decomposing along biconnectvity

Definition (Diblock).
The **diblock** of a vertex r is the union of $N^+(r)$ and all vertices that are bi-reachable from r.



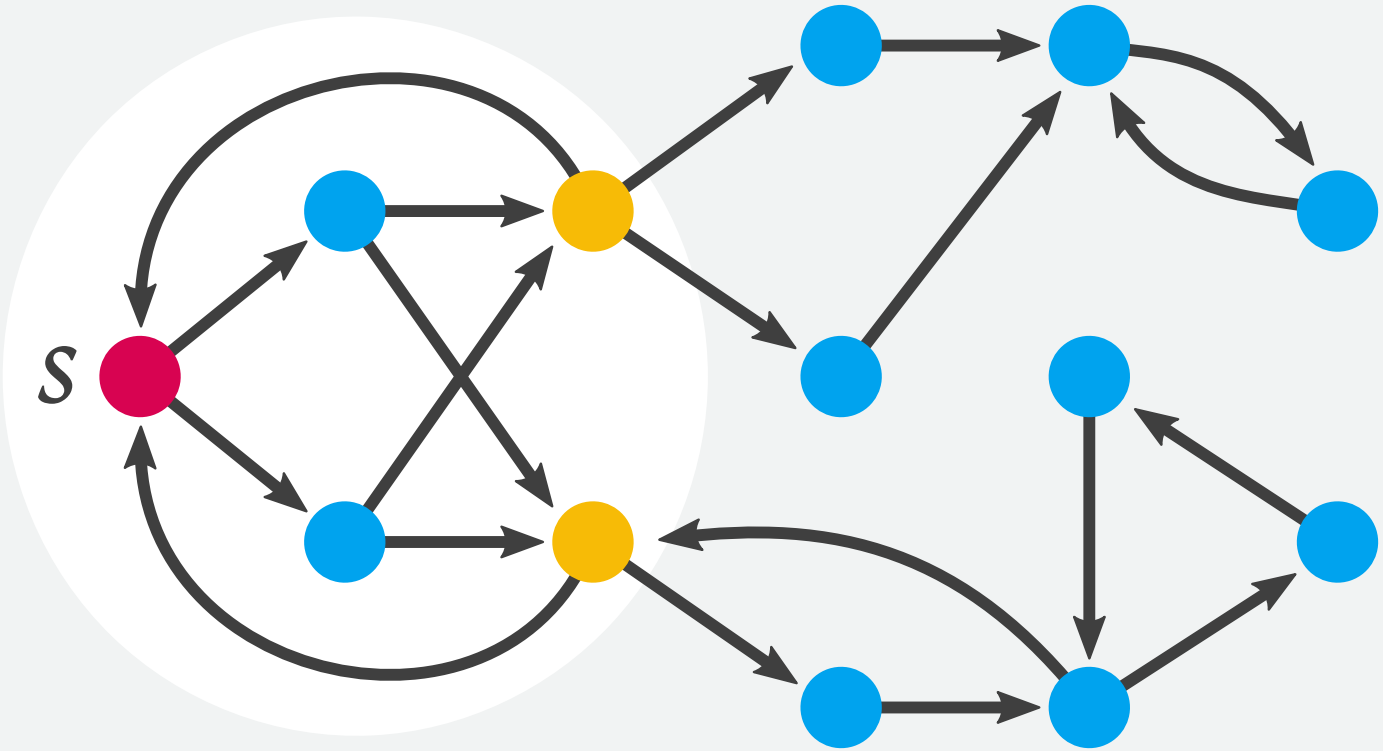
Decomposing along biconnectvity

Definition (Diblock).
The **diblock** of a vertex r is the union of $N^+(r)$ and all vertices that are bi-reachable from r.



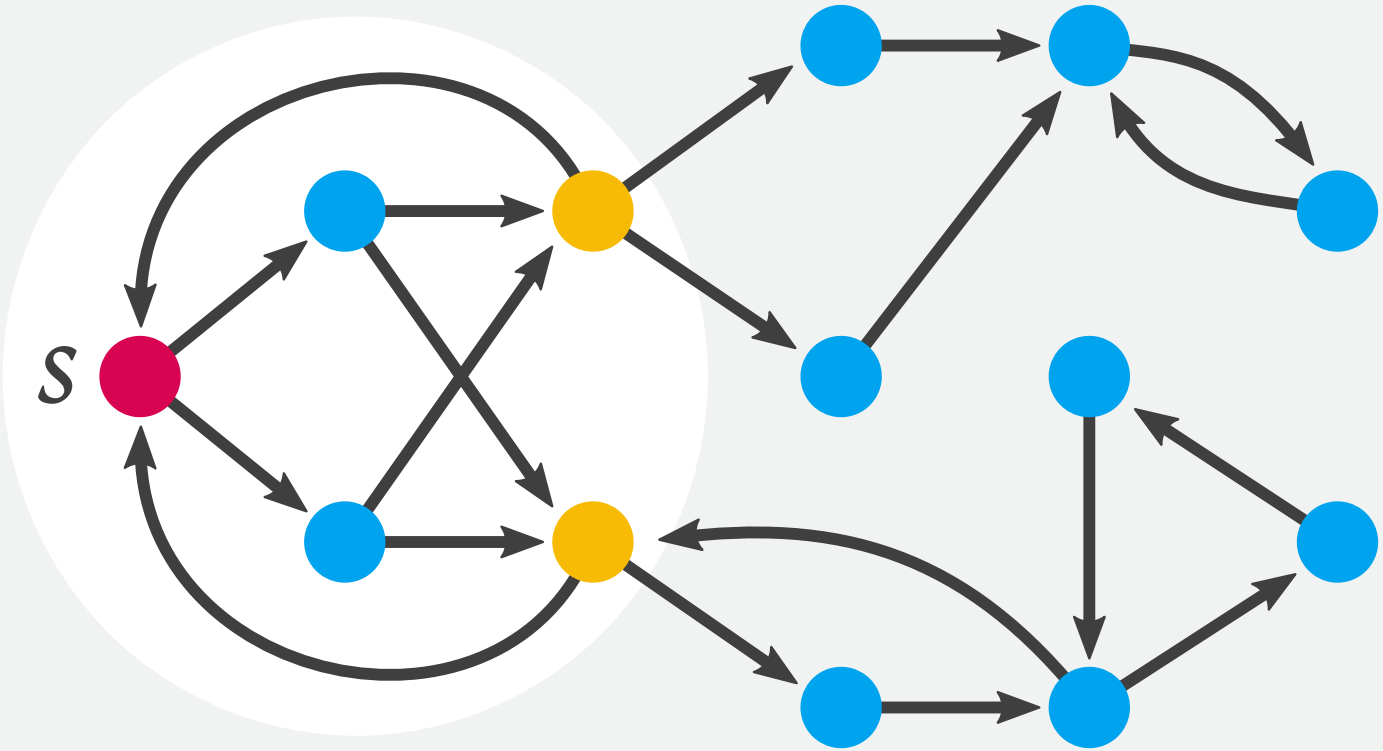
Decomposing along biconnectvity

Definition (Cut decomposition).
Recursively decompose into diblocks and **bottlenecks**.



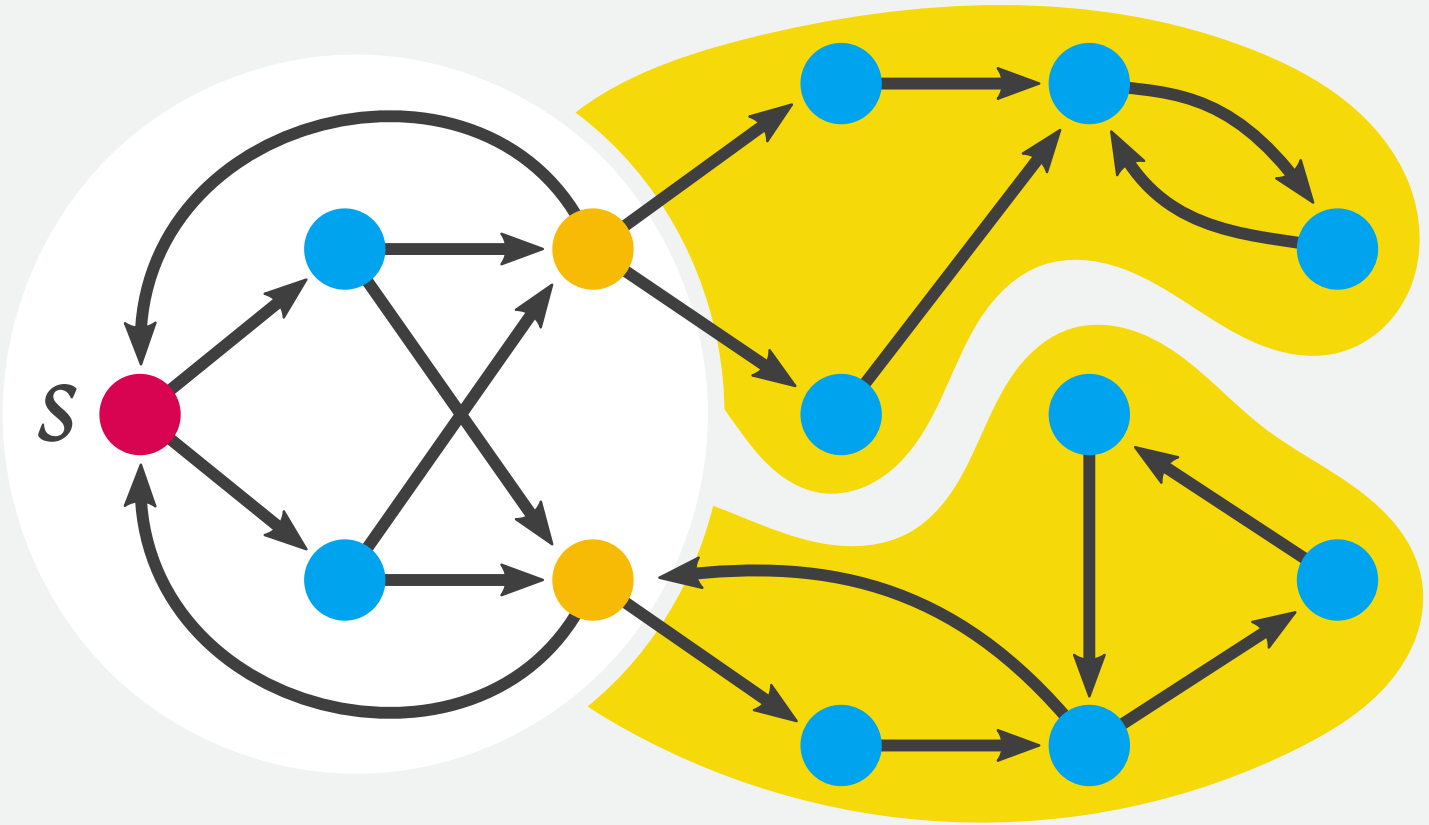
Decomposing along biconnectvity

Definition (Cut decomposition).
Recursively decompose into diblocks and **bottlenecks**.



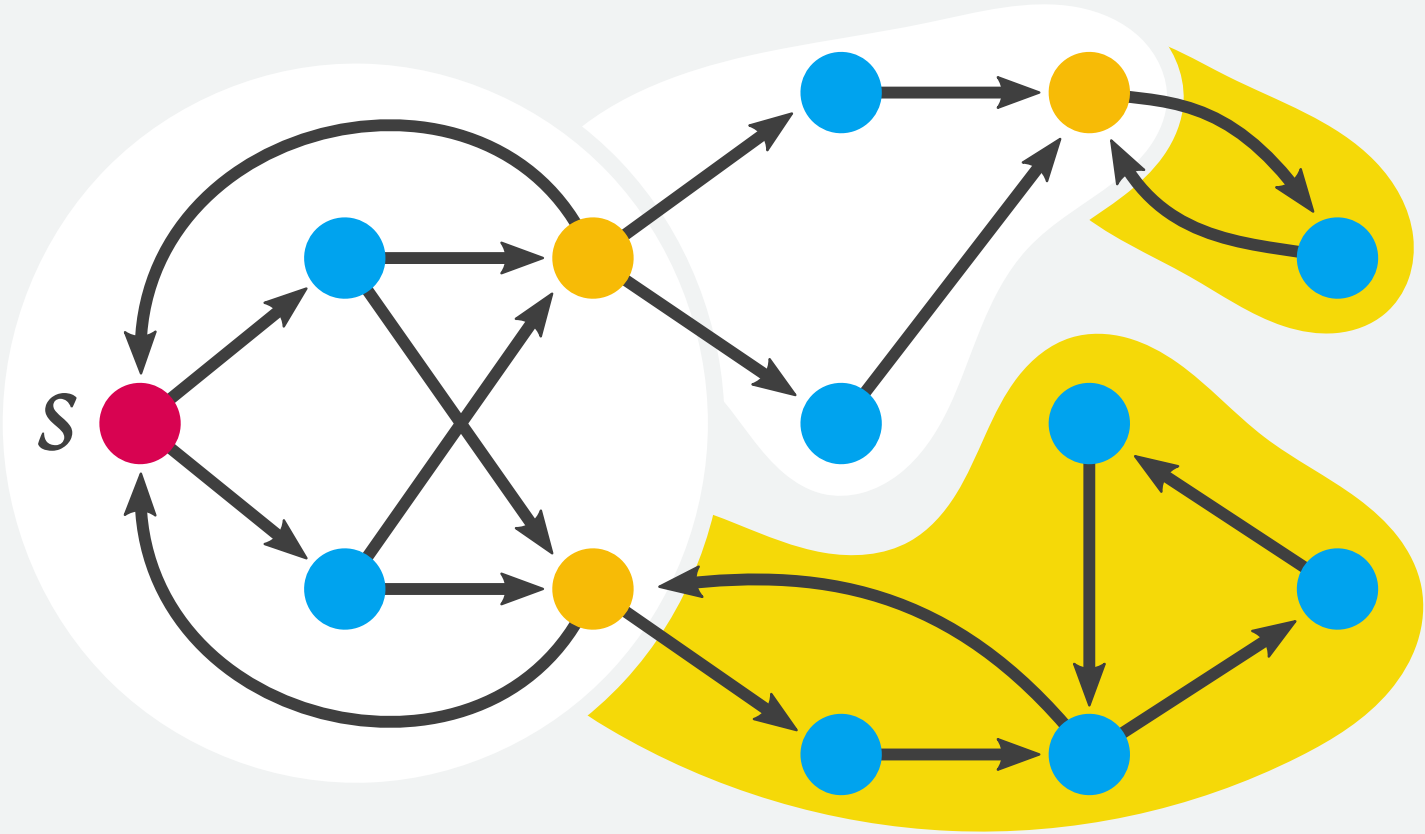
Decomposing along biconnectvity

Definition (Cut decomposition).
Recursively decompose into diblocks and **bottlenecks**.



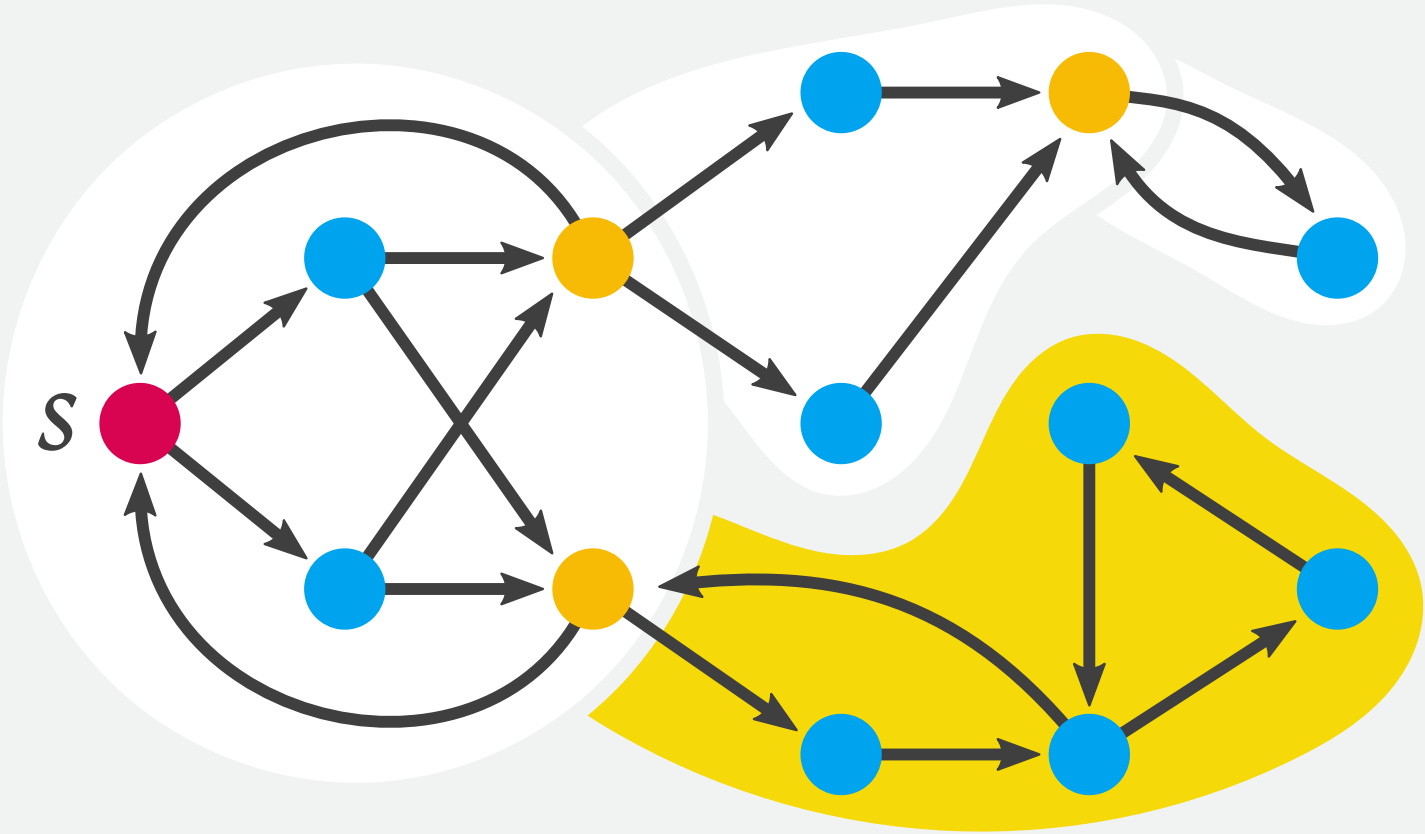
Decomposing along biconnectvity

Definition (Cut decomposition).
Recursively decompose into diblocks and **bottlenecks**.



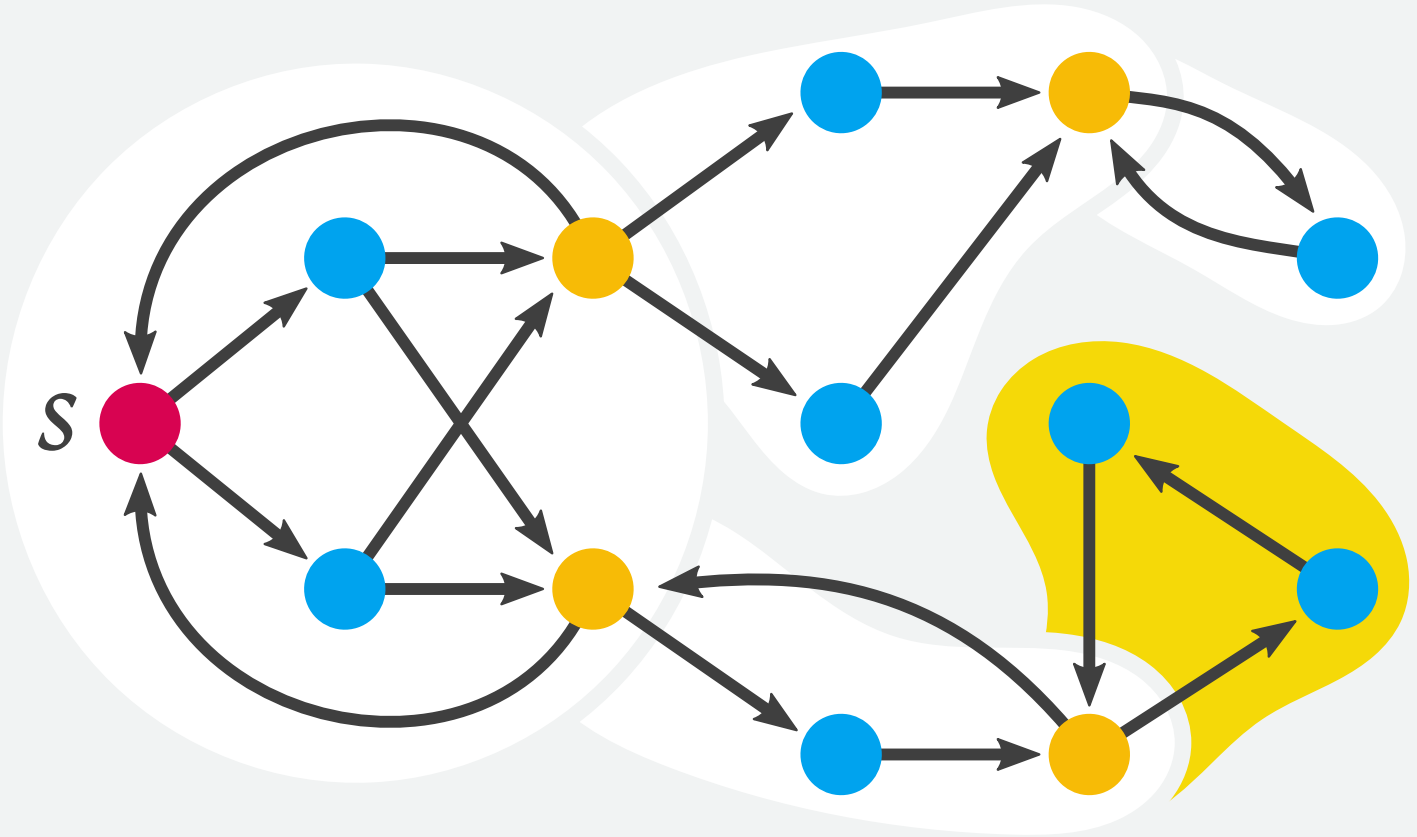
Decomposing along biconnectvity

Definition (Cut decomposition).
Recursively decompose into diblocks and **bottlenecks**.



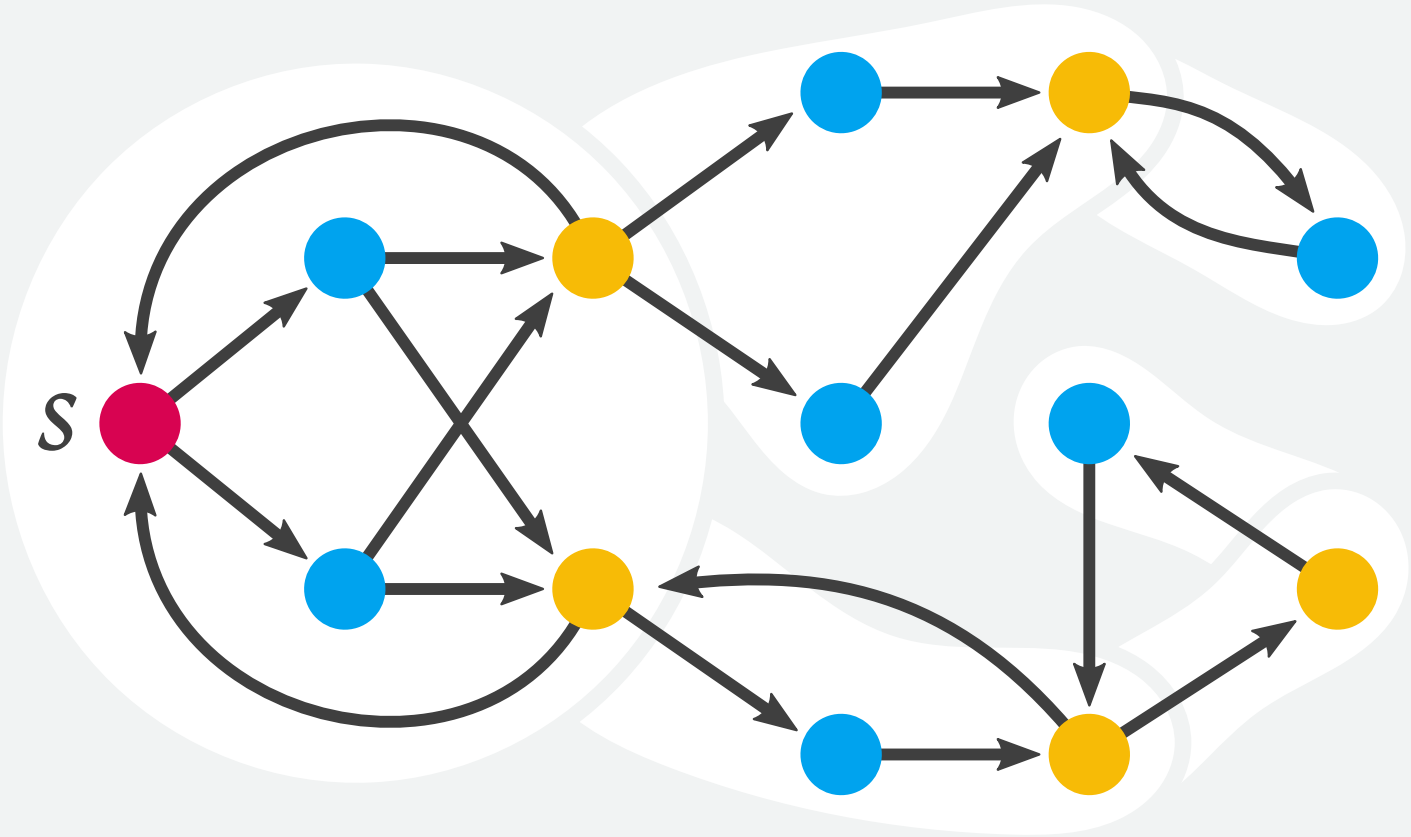
Decomposing along biconnectvity

Definition (Cut decomposition).
Recursively decompose into diblocks and **bottlenecks**.



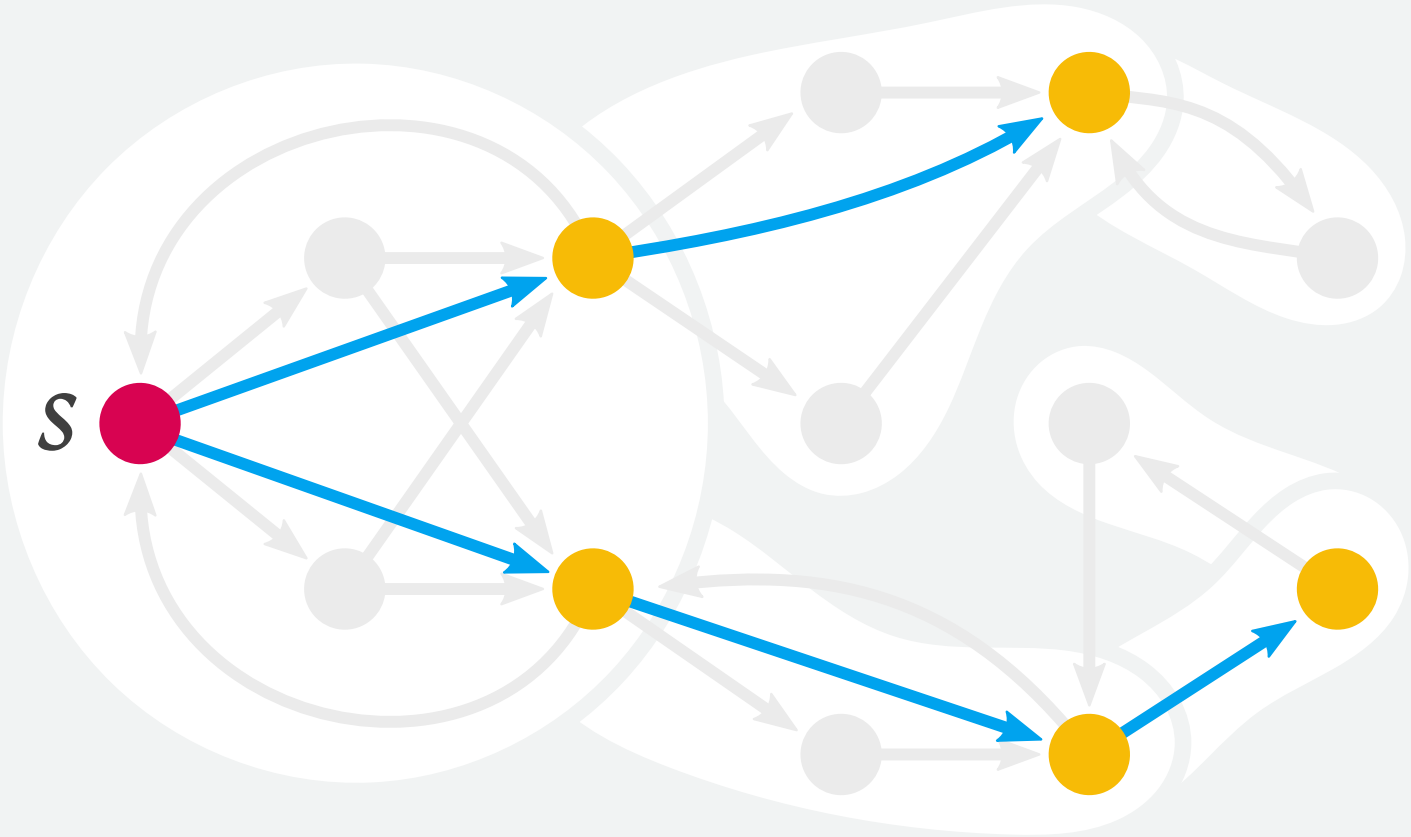
Decomposing along biconnectvity

Definition (Cut decomposition).
Recursively decompose into diblocks and **bottlenecks**.



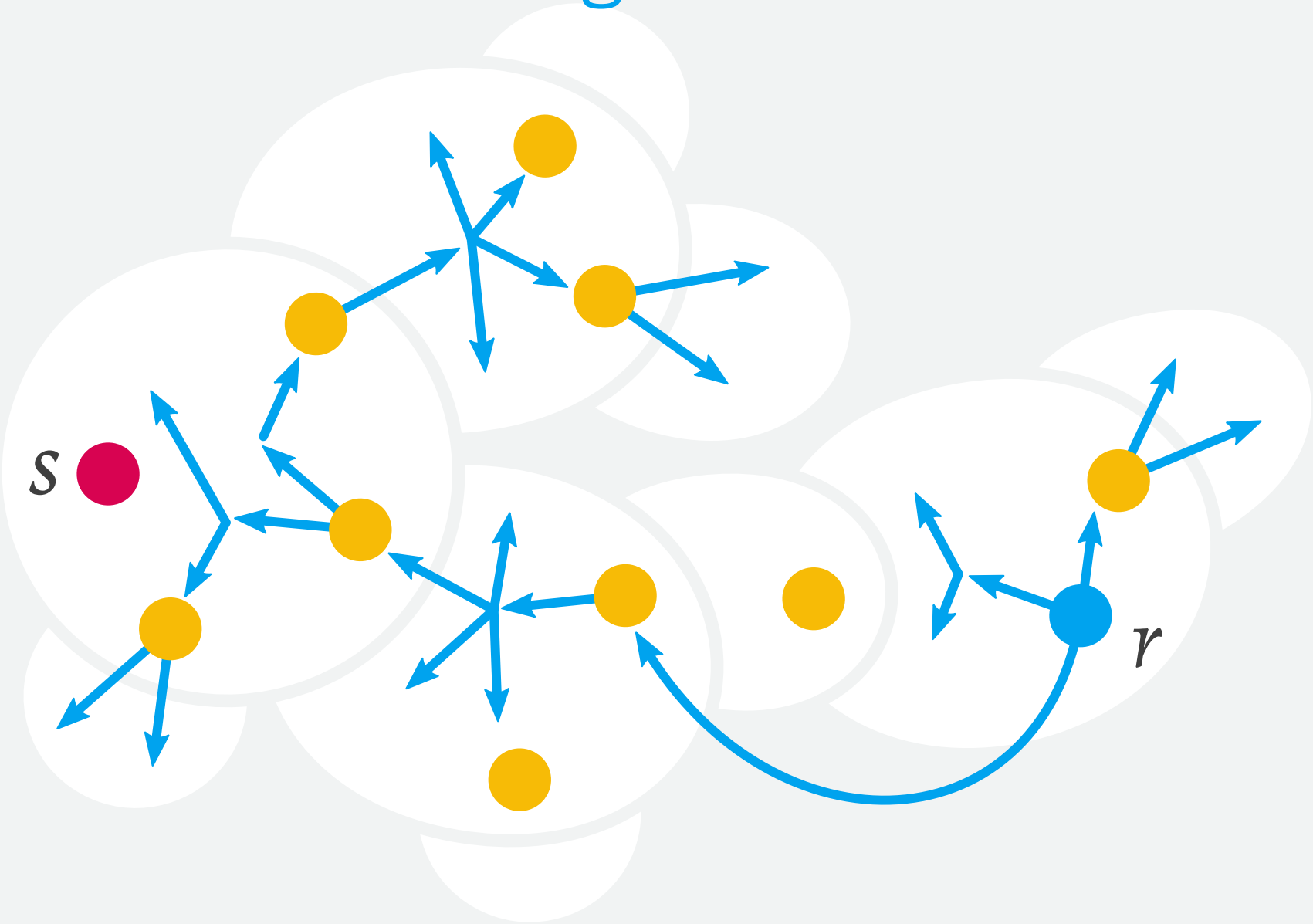
Decomposing along biconnectvity

Definition (Cut decomposition).
Recursively decompose into diblocks and **bottlenecks**.

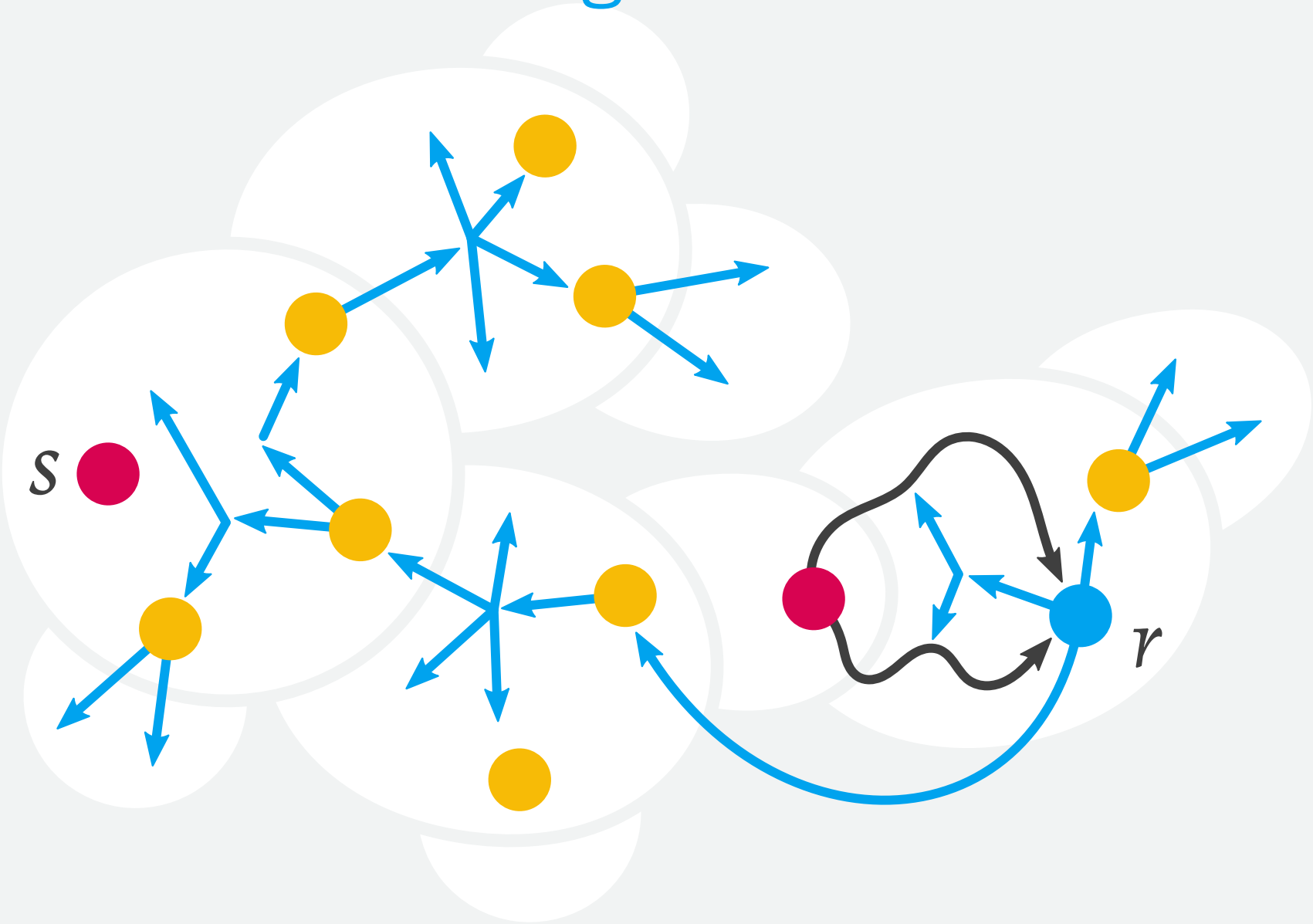


Re-rooting an out-tree

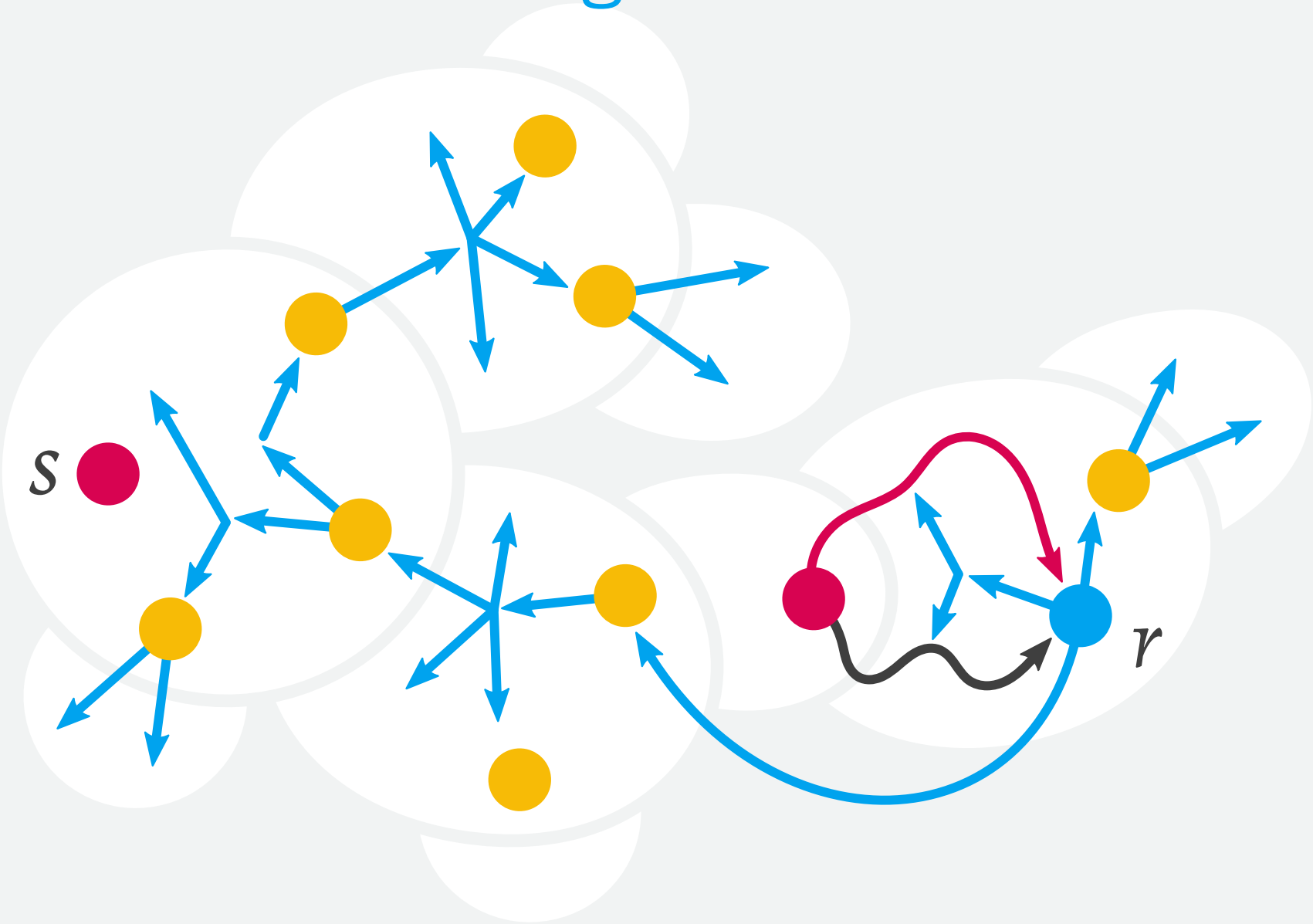
Re-rooting an out-tree



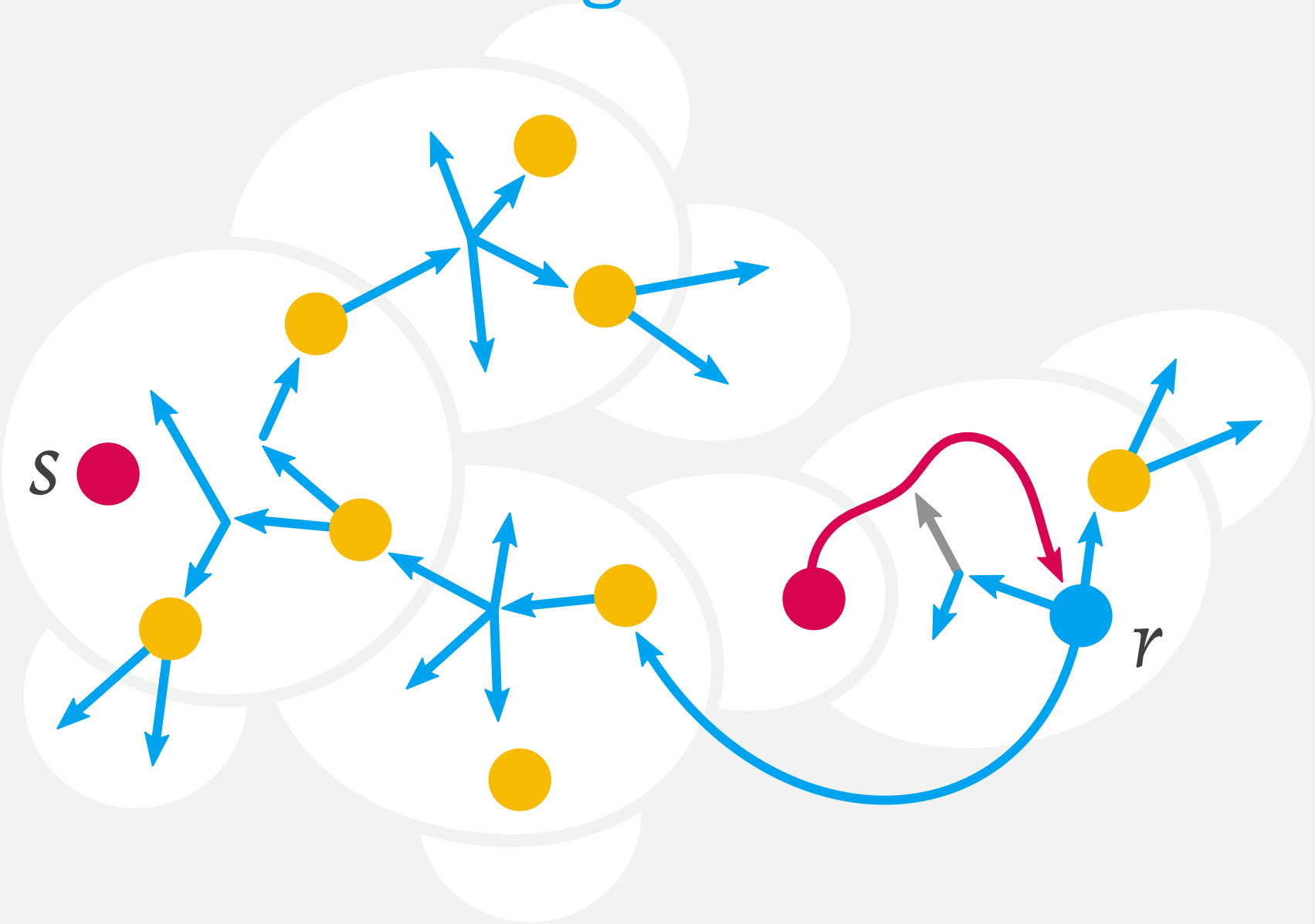
Re-rooting an out-tree



Re-rooting an out-tree



Re-rooting an out-tree



Re-rooting an out-tree

Re-rooting an out-tree

Re-rooting an out-tree

Re-rooting an out-tree

Re-rooting an out-tree

Re-rooting an out-tree

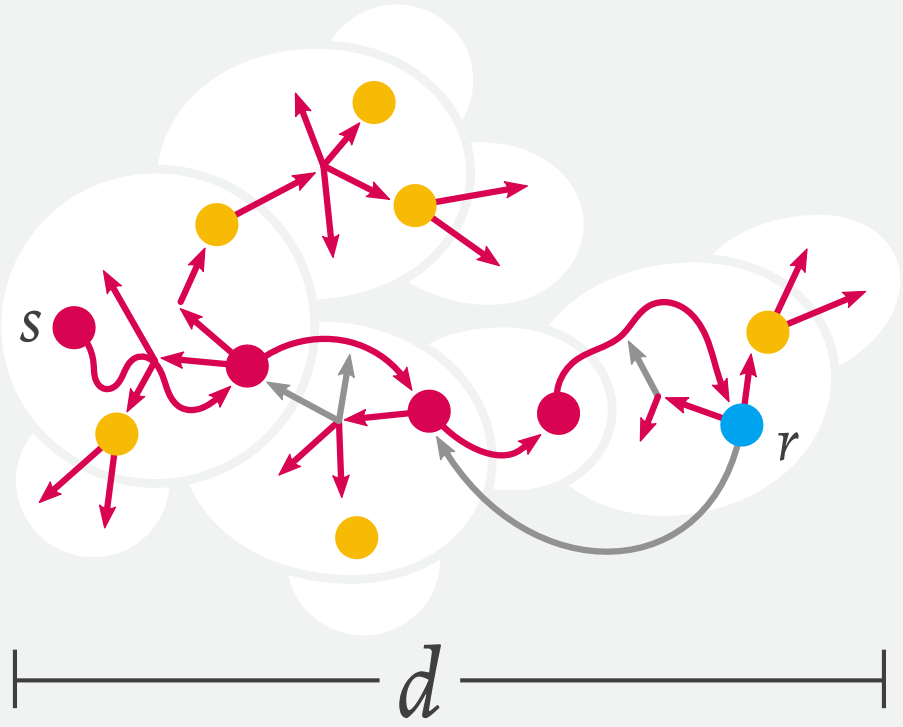
Re-rooting an out-tree

Re-rooting an out-tree

Re-rooting an out-tree

Re-rooting an out-tree

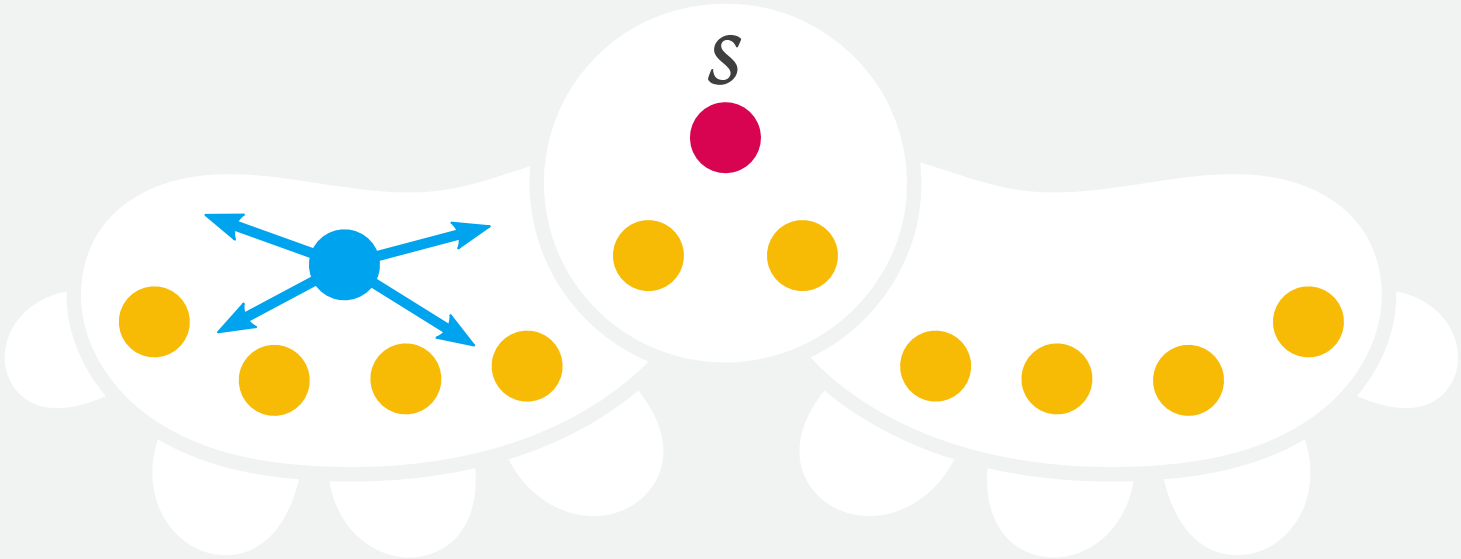
Re-rooting an out-tree



Lemma.
If out-tree had l leaves, then re-rooted tree has $(l\text{-}d)/2$ leaves where d is the **height** of the cut decomposition.

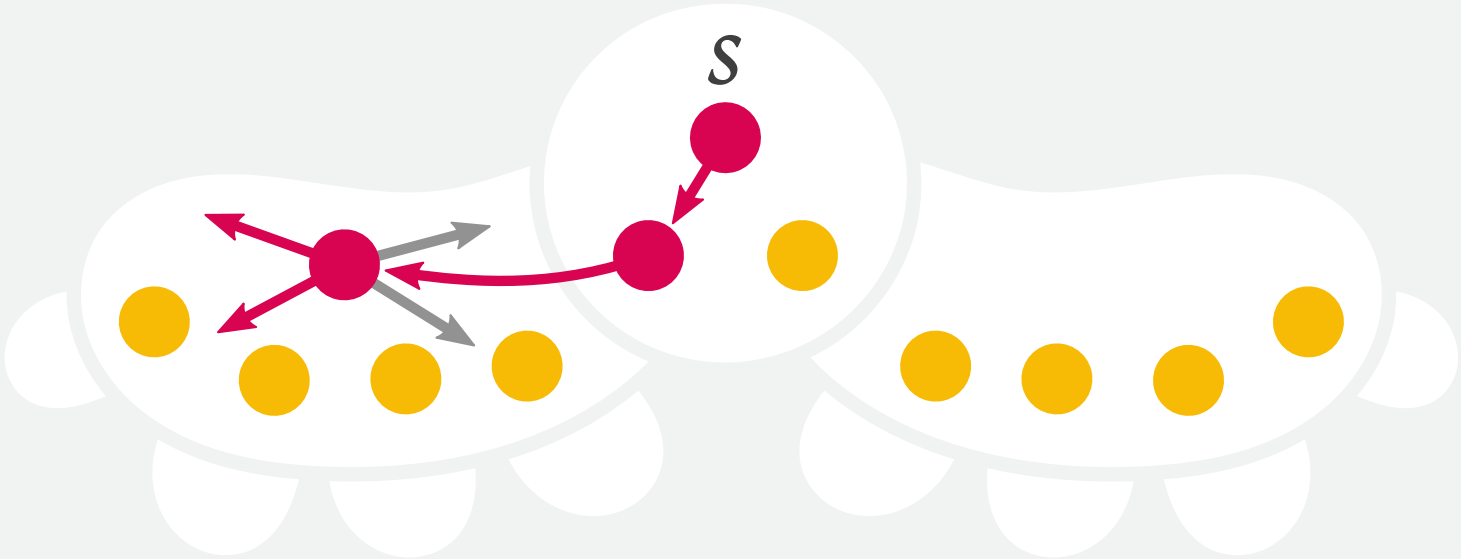
Almost a win-win

If the cut-decomposition is low…



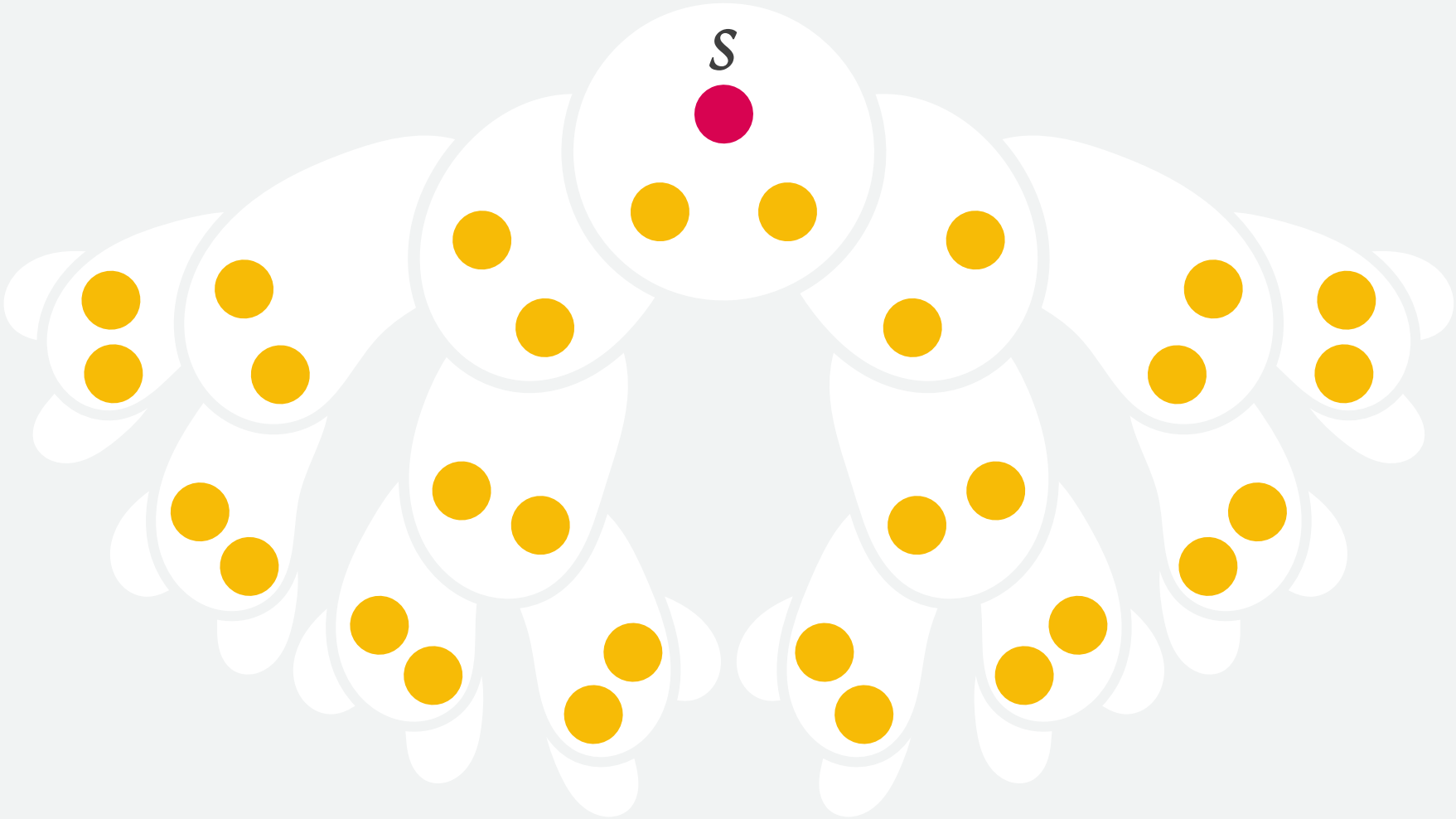
Almost a win-win

If the cut-decomposition is low, we can re-root any out-tree without losing too many leaves



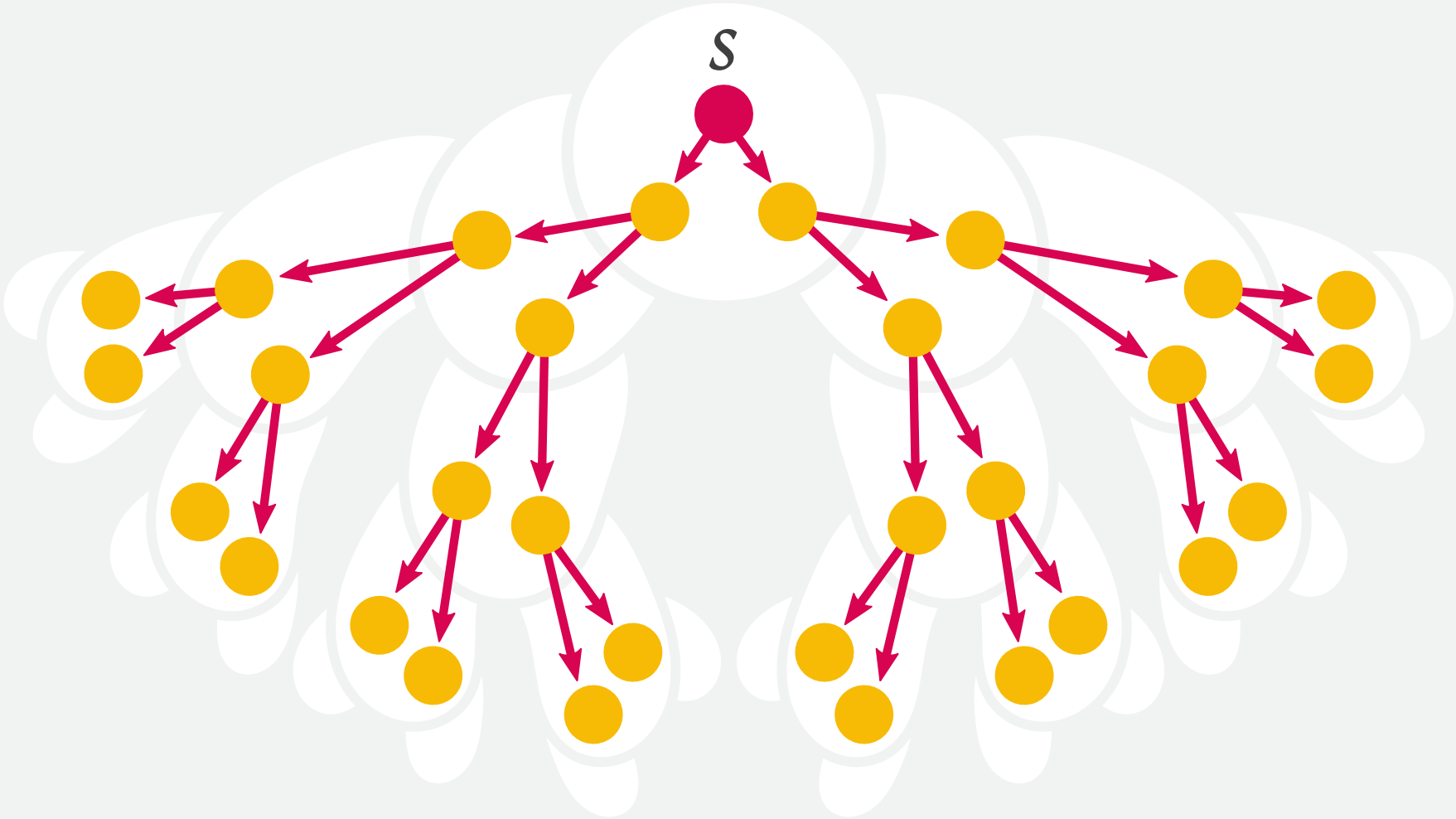
Almost a win-win

If the cut-decomposition is high...



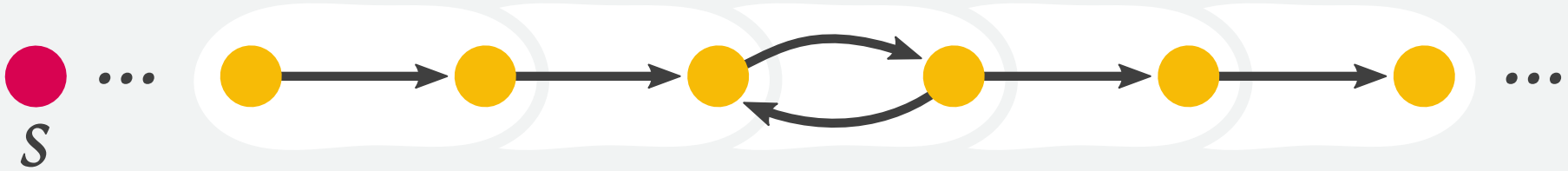
Almost a win-win

If the cut-decomposition is high, we should find an out-tree with many leaves!



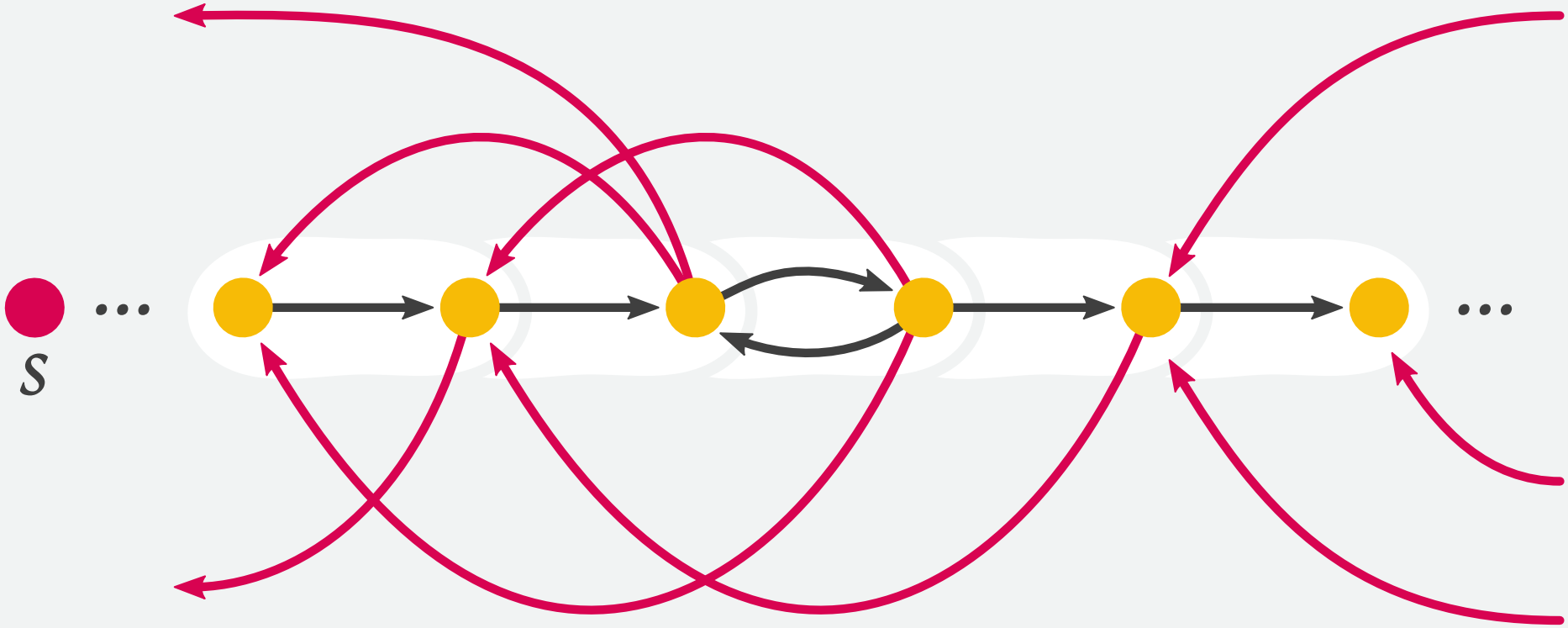
Obstacle: degenerate blocks

A diblock of size two is called degenerate.



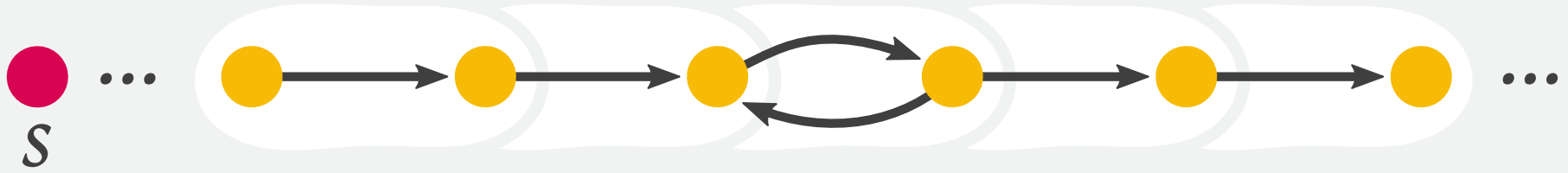
Obstacle: degenerate blocks

A diblock of size two is called degenerate.



Obstacle: degenerate blocks

A diblock of size two is called degenerate.



If there exists more than $14k+3$ degenerate diblocks in sequence, we either

1) Apply one of three reduction rules, or
2) construct an in-tree that avoids many arcs.

Schematic of our result (cont'd)

- If D has a rooted out-tree with many leaves, it is a YES-instance
- If D has **no** out-tree with many leaves, it has bounded pathwidth

- If cut decomposition is high: either find a rooted out-tree with many leaves or reduce
- If cut decomposition is low: re-root out-tree with many leaves, keeping many leaves

Summary

- **We show that** SINGLE ROOT k-DISTINCT BRANCHINGS**,** ROOTED k-DISTINCT BRANCHINGS **&** k-DISTINCT BRANCHINGS **are** FPT **in general digraphs**
- **Assuming that a** $2^{O(\mathbf{pw}\ log\ \mathbf{pw})}$ **algorithm exists, our algorithm runs in time** $2^{O(k^2 log^2 k)} n^{O(1)}$

Summary

- **We show that** SINGLE ROOT k-DISTINCT BRANCHINGS, ROOTED k-DISTINCT BRANCHINGS **&** k-DISTINCT BRANCHINGS **are** FPT **in general digraphs**
- **Assuming that a** $2^{O(\mathbf{pw}\ log\ \mathbf{pw})}$ **algorithm exists, our algorithm runs in time** $2^{O(k^2 log^2 k)} n^{O(1)}$
- $2^{O(k\ log\ k)} n^{O(1)}$ **with more careful analysis..?**
- **Other applications for cut decomposition!**
- **Generalise to larger cut size!**
- **Faster in special graph classes? Cf. sensor network application!**

THANKS!

Questions?