Hierarchical Routing

Réseaux & Communication

Alain Cournier   Stéphane Devismes

Université de Picardie Jules Verne

December 3, 2024

Roadmap

Roadmap

Idea

To reduce the **cost parameters** of the routing:

use of a **hierarchical division** of the network

Justification: Most of the communication is local, *i.e.*, between nodes at "relatively" small distances from each other

Cost Parameters

1. Length of addresses

   n nodes $\Rightarrow$ at least $\log(n)$ bits per address

   Maybe more, if information is encoded in addresses

   E.g., the prefix routing.

Cost Parameters

1. Length of addresses

   n nodes $\Rightarrow$ at least $\log(n)$ bits per address

   Maybe more, if information is encoded in addresses

   E.g., the prefix routing.

2. Size of the routing table

   A "brute-force" routing table contains n cells

Cost Parameters

1. Length of addresses

   n nodes $\Rightarrow$ at least $\log(n)$ bits per address

   Maybe more, if information is encoded in addresses

   E.g., the prefix routing.

2. Size of the routing table

   A "brute-force" routing table contains n cells

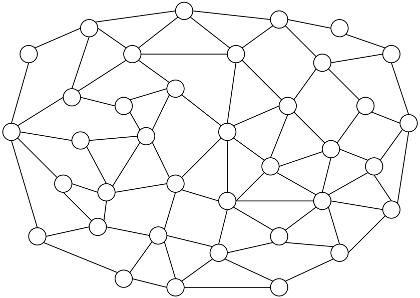
3. Cost of table lookups

   The cost (in time) of a single table lookups is likely as larger for a large routing table or for larger addresses

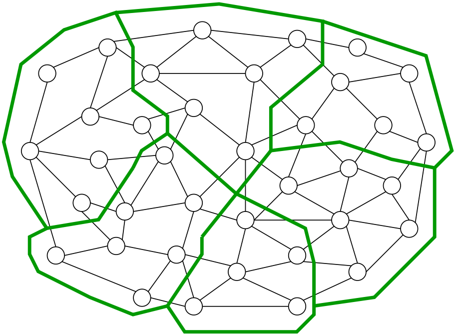
   The total table-lookup time for the delivery of a single message also depends on the number of times the tables must be accessed (number of hops)

Clustering

Definition

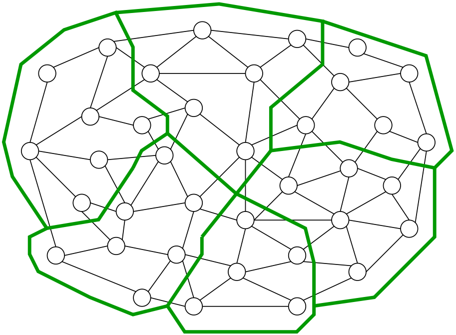


- Partition of the network into connected subgraphs called **clusters**



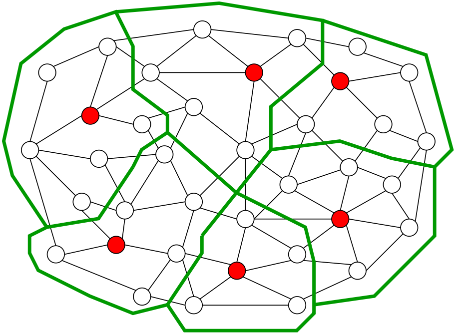
Clustering
Definition

- Partition of the network into connected subgraphs called **clusters**
- Each cluster is roughly of the same size

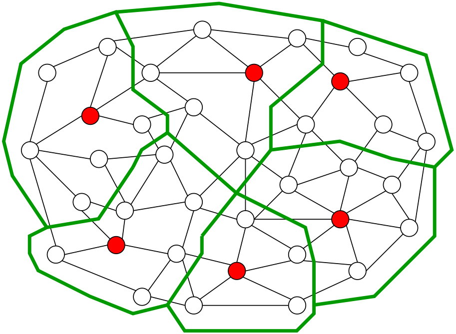


Clustering
Definition

- Partition of the network into connected subgraphs called **clusters**
- Each cluster is roughly of the same size
- In each cluster, there is a designated node: the **clusterhead**



- Partition of the network into connected subgraphs called **clusters**
- Each cluster is roughly of the same size
- In each cluster, there is a designated node: the **clusterhead**



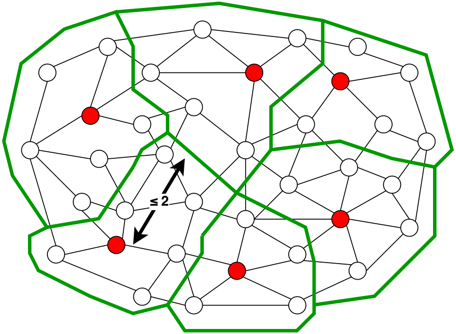
Remark: Clustering maybe recursive, *i.e.*, each cluster may be partitioned into subclusters, and so on so forth, in order to obtain a multi-level division of the nodes.

k-Clustering

A ***k*-clustering** is a clustering of the network where each cluster is of radius at most k

($k = 1 \Rightarrow$ dominating set)

In a cluster, each node is at distance at most k from its clusterhead.



2-clustering

A good k-Clustering?

Trivial solution: all nodes are clusterheads!

A good k-Clustering?

Trivial solution: all nodes are clusterheads!

The number of clusters/clusterheads should be minimized

A good k-Clustering?

Trivial solution: all nodes are clusterheads!

The number of clusters/clusterheads should be minimized

But, **computing the minimum k-clustering is $\mathcal{NP}$-hard [5]**

A good k-Clustering?

Trivial solution: all nodes are clusterheads!

The number of clusters/clusterheads should be minimized

But, **computing the minimum k-clustering is $\mathcal{NP}$-hard [5]**

Minimal k-clustering (*e.g.*, [2])

Let C be the set of clusterheads of a minimal k-clustering.

There is no k-clustering whose set of clusterheads is a proper subset of C

A good k-Clustering?

Trivial solution: all nodes are clusterheads!

The number of clusters/clusterheads should be minimized

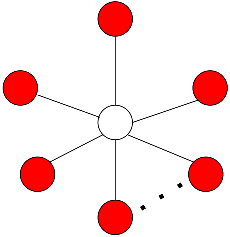
But, **computing the minimum k-clustering is $\mathcal{NP}$-hard [5]**

Minimal k-clustering (*e.g.*, [2])

Let C be the set of clusterheads of a minimal k-clustering.

There is no k-clustering whose set of clusterheads is a proper subset of C

However, there are degenerated cases



minimal 1-clustering with $n-1$ clusters/clusterheads

A good k-Clustering?

Trivial solution: all nodes are clusterheads!

The number of clusters/clusterheads should be minimized

But, **computing the minimum k-clustering is $\mathcal{NP}$-hard [5]**

Minimal k-clustering (*e.g.*, [2])

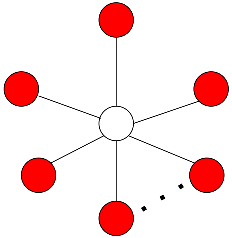
Let C be the set of clusterheads of a minimal k-clustering.

There is no k-clustering whose set of clusterheads is a proper subset of C

However, there are degenerated cases

k-clustering with $O(\frac{n}{k})$ clusters [3, 1]

Competitive k-clustering [1], *i.e.*, approximation: At most `M` times the size of the optimal one



minimal 1-clustering with $n-1$ clusters/clusterheads

A good k-Clustering?

Trivial solution: all nodes are clusterheads!

The number of clusters/clusterheads should be minimized

But, **computing the minimum k-clustering is $\mathcal{NP}$-hard [5]**

Minimal k-clustering (*e.g.*, [2])

Let C be the set of clusterheads of a minimal k-clustering.

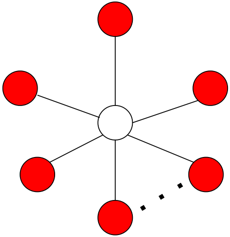
There is no k-clustering whose set of clusterheads is a proper subset of C

However, there are degenerated cases

k-clustering with $O(\frac{n}{k})$ clusters [3, 1]

Competitive k-clustering [1], *i.e.*, approximation: At most `M` times the size of the optimal one

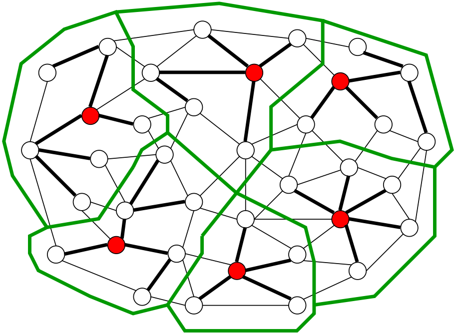
Remark: a ($O(\frac{n}{k})$) k-clustering can be made minimal to even more reduce the number of clusters [2]



minimal 1-clustering with $n-1$ clusters/clusterheads

Cluster = Tree rooted at its clusterhead

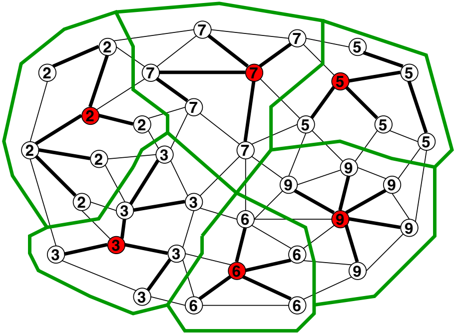
Spanning Forest



Cluster = Tree rooted at its clusterhead

Spanning Forest

Colored Trees, *e.g.*, with the clusterhead identifiers



Clustering-based Routing

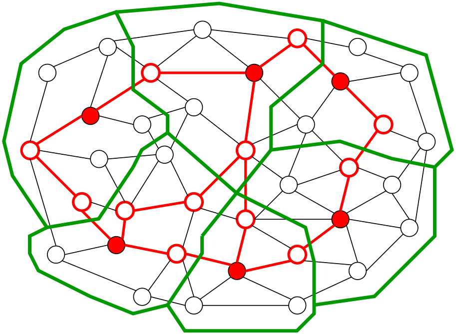
A major application of k-clustering is in the implementation of an efficient routing scheme in a network.

To route a packet from p to q:

1. Route the packet from p to its clusterhead (intra-clustering)
2. Route the packet from the clusterhead of p to the clusterhead of q (inter-clustering)
3. Route the packet from the clusterhead of q to q (intra-clustering)

Clustering-based Routing

A major application of *k*-clustering is in the implementation of an efficient routing scheme in a network.



To route a packet from p to q:

1. Route the packet from p to its clusterhead (intra-clustering)
2. Route the packet from the clusterhead of p to the clusterhead of q (inter-clustering)
3. Route the packet from the clusterhead of q to q (intra-clustering)

To route among clusterheads, we need a structure: a subgraph

Clustering-based Routing

A major application of k-clustering is in the implementation of an efficient routing scheme in a network.

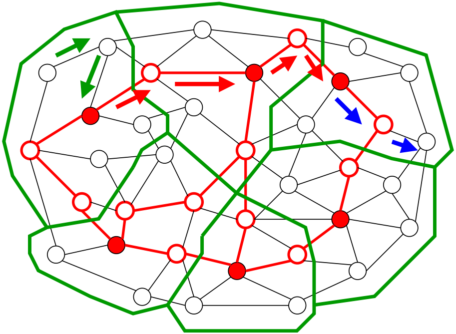
To route a packet from p to q:

1. Route the packet from p to its clusterhead (intra-clustering)
2. Route the packet from the clusterhead of p to the clusterhead of q (inter-clustering)
3. Route the packet from the clusterhead of q to q (intra-clustering)

To route among clusterheads, we need a structure: a subgraph

Each of the three phases can be identified using a color

- Depending of its color, a packet is either forwarded over a fixed channel or a more complex routing scheme
- Each phase can be handled using a different protocol



Clustering-based Routing

A major application of k-clustering is in the implementation of an efficient routing scheme in a network.

To route a packet from p to q:

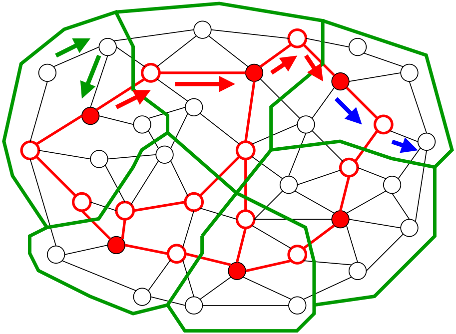
1. Route the packet from p to its clusterhead (intra-clustering)
2. Route the packet from the clusterhead of p to the clusterhead of q (inter-clustering)
3. Route the packet from the clusterhead of q to q (intra-clustering)

To route among clusterheads, we need a structure: a subgraph

Each of the three phases can be identified using a color

- Depending of its color, a packet is either forwarded over a fixed channel or a more complex routing scheme
- Each phase can be handled using a different protocol

Address: (Label of the clusterhead of the destination,destination label)



Roadmap

1. Introduction
2. Examples of clustering
   - Clustering of [3]
   - Clustering of [1]
3. Routing in a Clustering
4. References

Two examples

1. Clustering of [3]
2. Clustering of [1]

Both computes $O(\frac{n}{k})$ clusters

Two examples

1. Clustering of [3]
2. Clustering of [1]

Both computes $O(\frac{n}{k})$ clusters

The second one also provides an approximation of the optimal clustering in case the network is a Unit (or Quasi Unit) Disk Graph

Idea

Let $G = (V, E)$ be a connected graph of n nodes

A **k-dominating set** of G is a subset D of V such that $\forall p \in V$, $\exists q \in D$ such that $\|p, q\| \leq k$

k-dominating set $\approx$ set of clusterheads

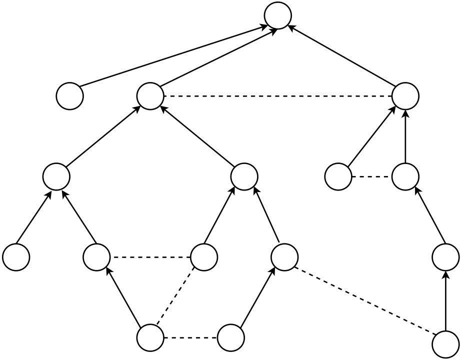
Theorem

For every $k \in \mathbb{N}$, there exists a k-dominating set D of G such that $|D| \leq \lceil \frac{n}{k+1} \rceil$

Proof of the Theorem

Some notations

Let $T = (V, E_T)$ be a spanning tree of G rooted at node r

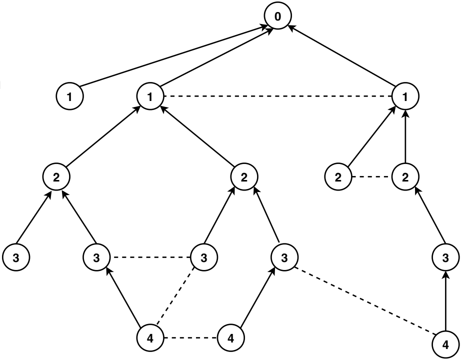


Proof of the Theorem

Some notations

Let $T = (V, E_T)$ be a spanning tree of G rooted at node r

Let $L(p) = \|p, r\|_T$ be the level of node p in T



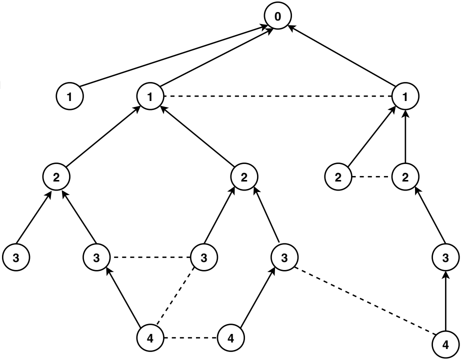
Proof of the Theorem

Some notations

Let $T = (V, E_T)$ be a spanning tree of G rooted at node r

Let $L(p) = \|p, r\|_T$ be the level of node p in T

Let $H = \max_{p \in V} L(p)$ be the height of T (4 in the example)



Proof of the Theorem

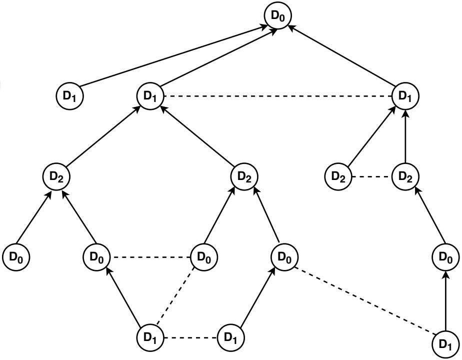
Some notations

Let $T = (V, E_T)$ be a spanning tree of G rooted at node r

Let $L(p) = \|p, r\|_T$ be the level of node p in T

Let $H = \max_{p \in V} L(p)$ be the height of T (4 in the example)

For every $i \in [0..k]$, let $D_i = \{p \in V \mid L(p) \bmod (k+1) = i\}$ (in the example, $k = 2$)



Proof of the Theorem

Easy cases: $n = 0$ or $k = 0$

1. If $n = 0$, then $\lceil \frac{n}{k+1} \rceil = 0 = |\emptyset|$ and $\emptyset$ is a k-dominating set of G.

2. If $k = 0$, then $D_0 = V$, and so D_0 is a k-dominating set of G. Moreover, $|D_0| = n = \lceil \frac{n}{k+1} \rceil$.

3. Assume $k \geq H$. Then, D_0 only contains r and every other node is within distance k from r. So, D_0 is a k-dominating set of G whose size is $1 \leq \lceil \frac{n}{k+1} \rceil$.

4. Assume $k < H$. Then, $\forall i \in [0..k], |D_i| > 0$.
   1. Assume that $\forall i \in [0..k-1], |D_i| = |D_{i+1}|$.

      Then, $\forall i \in [0..k]$, $|D_i| = \lceil \frac{n}{k+1} \rceil$.

      Let $v \notin D_0$. The level of v, $L(v)$, satisfies $L(v) = x(k+1) + y$, where $x \geq 0$ and $0 < y \leq k$.

      Let u be the ancestor of v such that $L(u) = L(v) - y$ (u exists because $y \leq L(v)$).

      By definition, $u \in D_0$ and $\|u, v\| \leq k$. Hence, D_0 is a k-dominating set of G such that $|D_0| = \lceil \frac{n}{k+1} \rceil$.

Proof of the Theorem

Case $n > 0$ and $k > 0$

4. 2. Assume that $\exists i \in [0..k-1], |D_i| \neq |D_{i+1}|$.

   Let min $\in [0..k]$ such that $\forall i \in [0..k]$, $|D_{\min}| \leq |D_i|$.

   Then, $|D_{\min}| < \lceil \frac{n}{k+1} \rceil$.

   Let $D = D_{\min} \cup \{r\}$. Then, $|D| \leq \lceil \frac{n}{k+1} \rceil$.

   Let $v \notin D$.

   1. If $L(v) \leq k$, then v is at distance at most k from r and $r \in D$.
   2. If $L(v) > k$, then $L(v) = x(k+1) + y$ with $x > 0$, $0 \leq y \leq k$, and $y \neq \min$.

      If $y > \min$, then let u be the ancestor of v such that $L(u) = x(k+1) + \min$. Now, $0 \leq L(v) - L(u) = y - \min \leq k$.

      If $y < \min$, let u be the ancestor of v such that $L(u) = (x-1)(k+1) + \min$. Now, $0 \leq L(v) - L(u) = k + y - min \leq k$.

      By definition, $u \in D$ (more precisely, $u \in D_{\min}$) and $\|u, v\| \leq k$.

   Hence, D is a k-dominating set of G and $|D| \leq \lceil \frac{n}{k+1} \rceil$.

□

Solution

Let $mcd = \min\{|D_i| \mid i \in [0..k] \wedge D_i \neq \emptyset\}$: mcd is the minimum cardinal of a non-empty D-set

Solution

Let $mcd = \min\{|D_i| \mid i \in [0..k] \wedge D_i \neq \emptyset\}$: mcd is the minimum cardinal of a non-empty D-set

Let $x = \min\{i \mid i \in [0..k] \wedge |D_i| = mcd\}$: x the the smallest index of a D-set of size mcd.

Solution

Let $mcd = \min\{|D_i| \mid i \in [0..k] \wedge D_i \neq \emptyset\}$: *mcd* is the minimum cardinal of a non-empty D-set

Let $x = \min\{i \mid i \in [0..k] \wedge |D_i| = mcd\}$: x the the smallest index of a D-set of size *mcd*.

$D_x \cup \{r\}$ is a k-dominating set of G if size at most $\lceil \frac{n}{k+1} \rceil$.

Proof. $D_x \cup \{r\}$ corresponds to each set exhibited in Cases 2-4 of the theorem proof (*n.b.*, Case 1 is for the beauty of the art, but useless)

□

Distributed Computation

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

Distributed Computation

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

3. Propagation of Information with Feedback in the tree: broadcast $\Rightarrow$ Node $(k+1)$-Coloring, Feedback $\Rightarrow$ Computation of the D-set sizes

   ($O(D)$ rounds, $O(n)$ messages of $O(k. \log n)$ bits, and $O(\log \Delta + k \log n)$ bits per node)

Distributed Computation

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

3. Propagation of Information with Feedback in the tree: broadcast $\Rightarrow$ Node $(k+1)$-Coloring, Feedback $\Rightarrow$ Computation of the D-set sizes

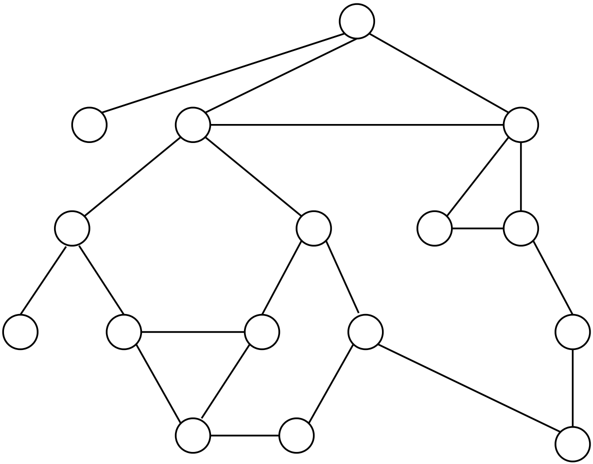
   ($O(D)$ rounds, $O(n)$ messages of $O(k.\log n)$ bits, and $O(\log \Delta + k \log n)$ bits per node)

4. Propagation of Information with Feedback in the tree: broadcast $\Rightarrow$ clusterhead assignment and cluster coloring, Feedback $\Rightarrow$ Termination Detection at the Leader

   ($O(D)$ rounds, $O(n)$ messages of $O(\log k + \log B)$ bits, and $O(\log B)$ bits per node)

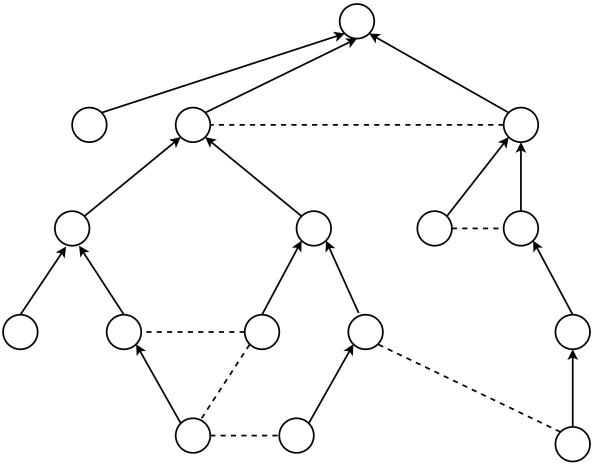
5. Routing Set-Up

Example with $k = 2$



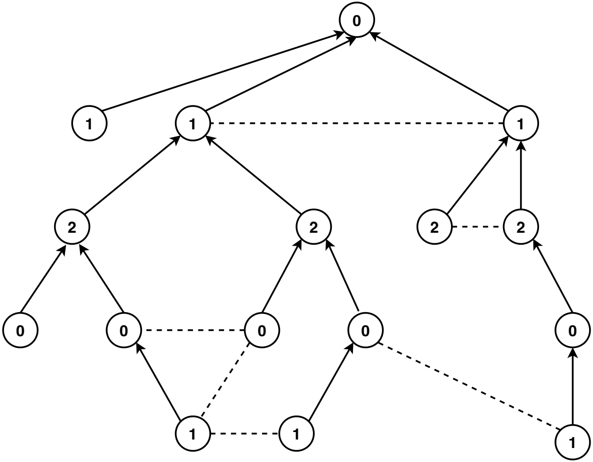
Network

Example with $k = 2$



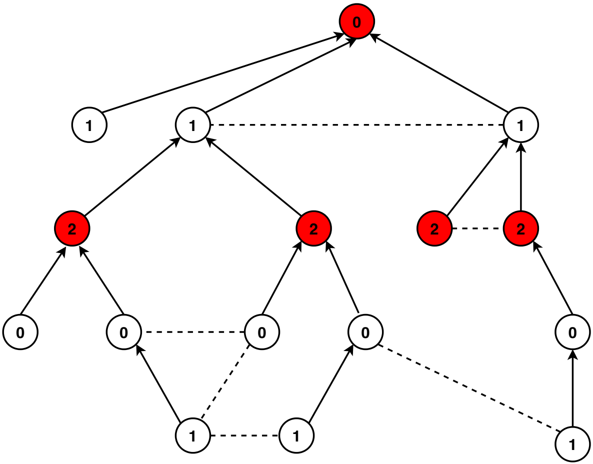
BFS spanning tree

Example with $k = 2$



Node $(k + 1)$-Coloring

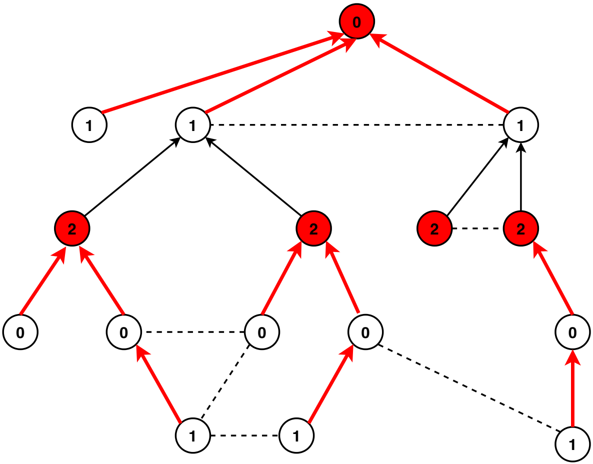
Example with $k = 2$



k-dominating set

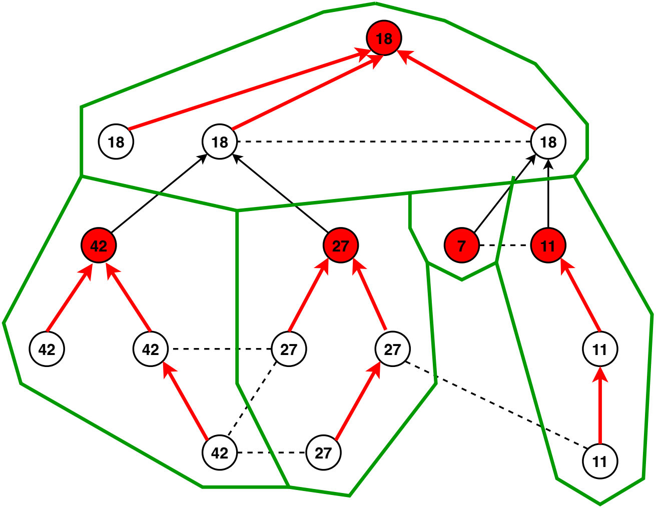
$|D_0| = 6$, $|D_1| = 6$, $|D_2| = 4$

Example with $k = 2$



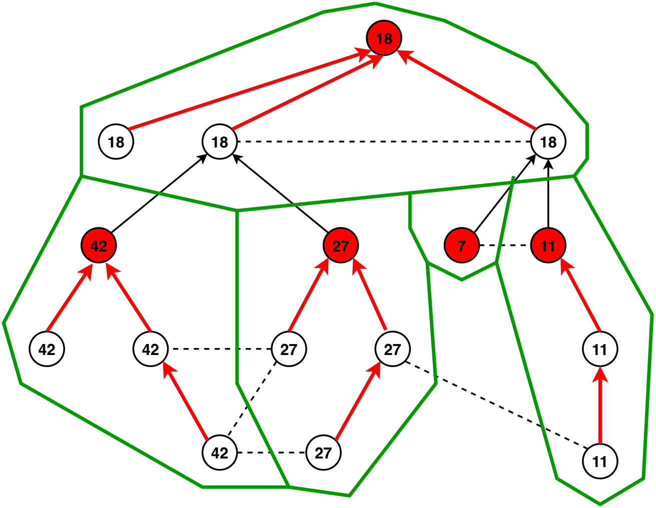
Clusters

Example with $k = 2$



Clustering

Example with $k = 2$



Clustering

Remark: the spanning tree can be used for inter-cluster routing

Pros & Cons

Pros:

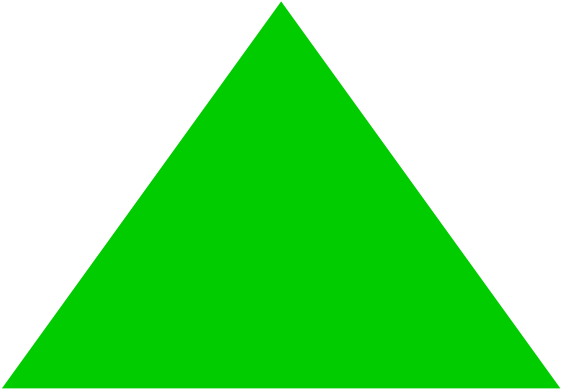
- Time-efficient computation
- BFS $\Rightarrow$ short paths

Cons:

- Memory requirement and message size: $\Omega(k \log n)$

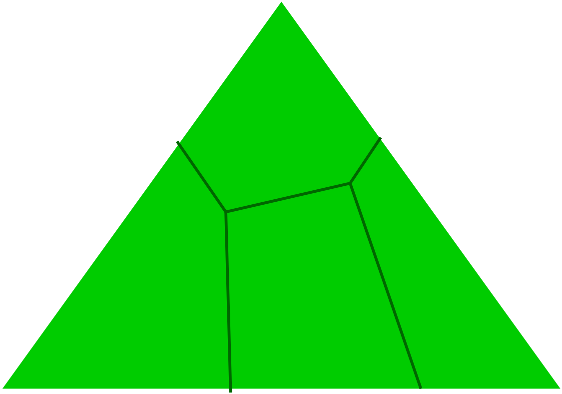
Roadmap

Idea



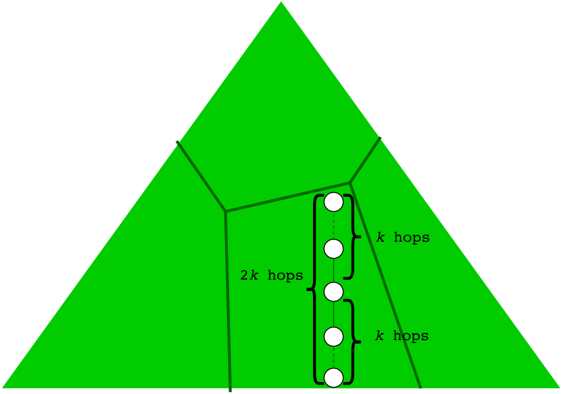
A tree T

Idea



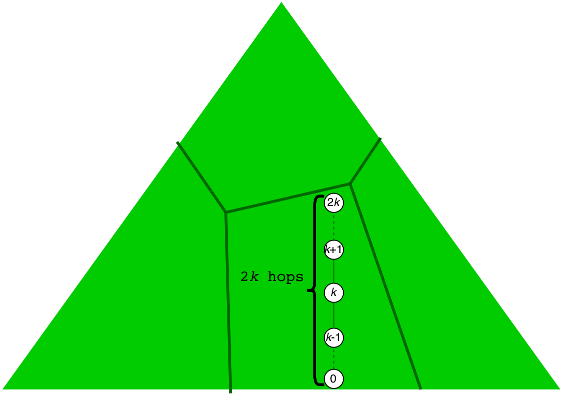
A tree T with clusters

Idea



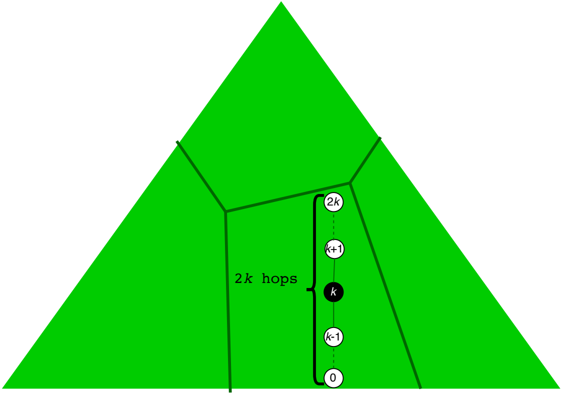
Build paths of 2k hops ($2k + 1$ nodes)

Idea



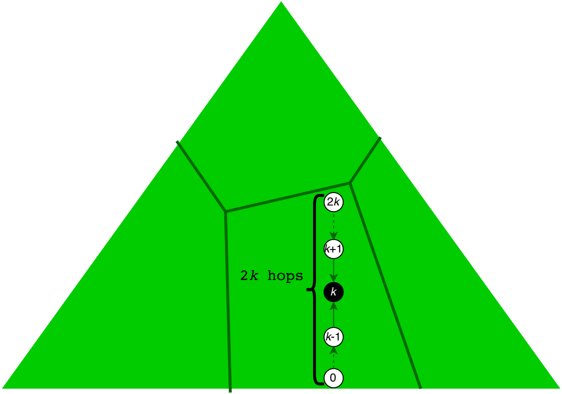
Node numbering

Idea



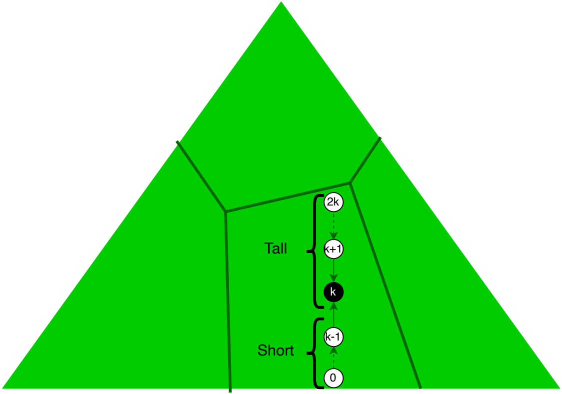
Node labeled k: clusterhead

Idea



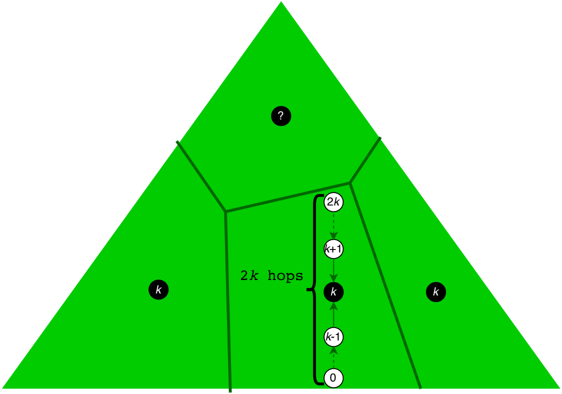
Parent Link

Idea



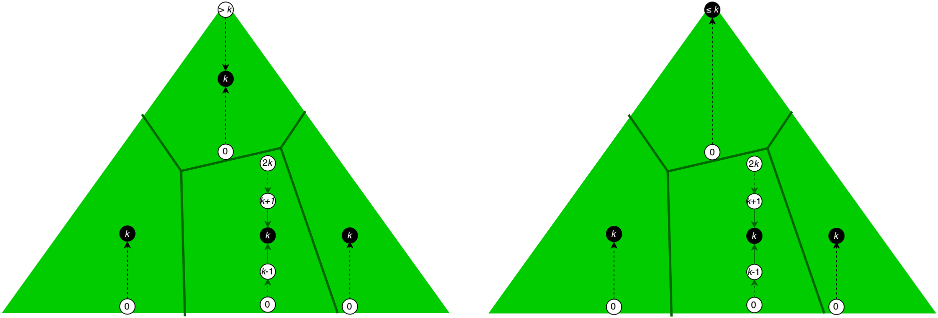
Tall and short nodes

Idea



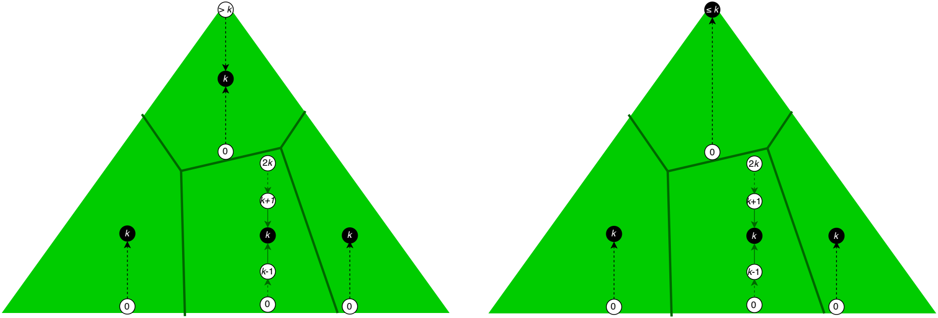
Issue: cluster of the root

Idea



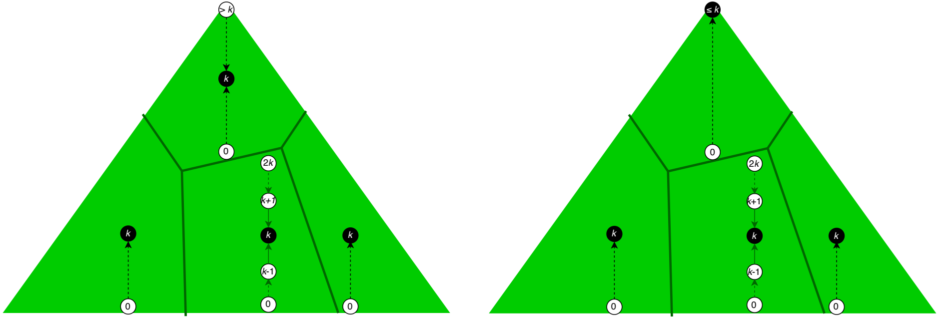
2 cases

Number of clusterheads



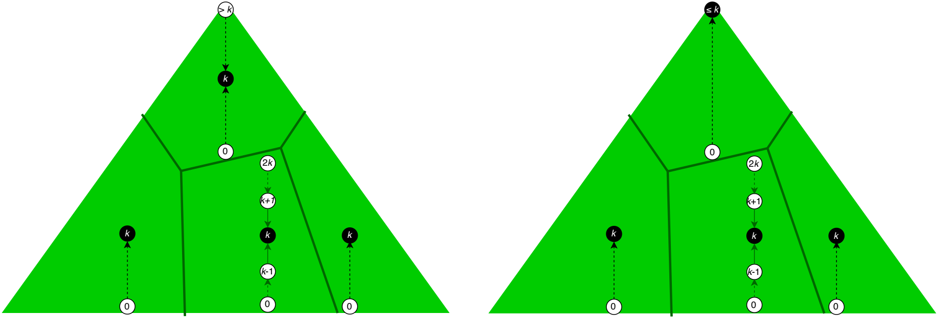
1. Non-root cluster: at least $k+1$ nodes

Number of clusterheads



1. Non-root cluster: at least $k+1$ nodes
2. 1 cluster with the root + at most $\lfloor \frac{n-1}{k+1} \rfloor$ non-root cluster

Number of clusterheads



1. Non-root cluster: at least $k+1$ nodes
2. 1 cluster with the root $+$ at most $\lfloor \frac{n-1}{k+1} \rfloor$ non-root cluster
3. $\sharp$Clusters $\leq 1 + \lfloor \frac{n-1}{k+1} \rfloor = \lfloor \frac{n+k}{k+1} \rfloor \leq \lceil \frac{n}{k+1} \rceil$

Node Numbering: function α

For every node p, $\alpha(p) \in [0..2k]$

- $maxShort(p) = \max(\{\alpha(q) \mid q \in Children(p) \wedge \alpha(q) < k\} \cup \{-1\})$
- $minTall(p) = \min(\{\alpha(q) \mid q \in Children(p) \wedge \alpha(q) \geq k\} \cup \{2k+1\})$

if $maxShort(p) + minTall(p) \leq 2k - 2$ **then**
  $\alpha(p) = minTall(p) + 1$
else
  $\alpha(p) = maxShort(p) + 1$
end if

Tree Structure of clusters

$\alpha(p) = k$ or (p is the root and $\alpha(p) \leq k$): clusterhead (root of the cluster)

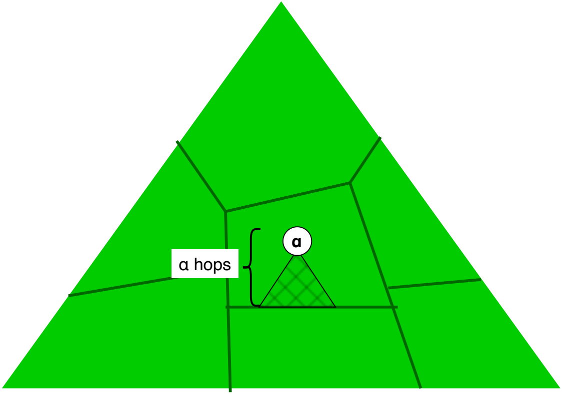
If p is not a clusterhead:

$\alpha(p) < k$: parent in the cluster := parent in the tree

$\alpha(p) > k$: parent in the cluster := a child q with $\alpha(q) = minTall(p)$

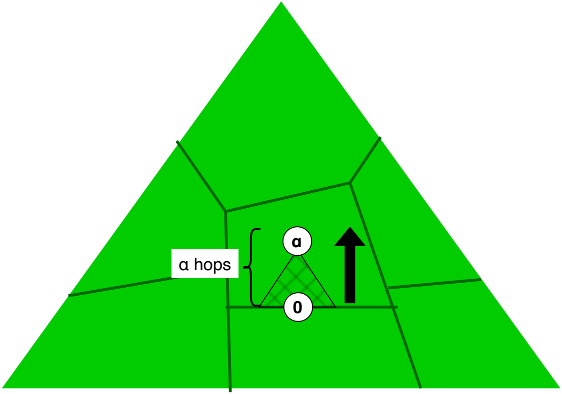
Meaning of α

$\alpha(p)$ is the distance from p to q where q its furthest process in $T(p)$ that is in the same cluster as p



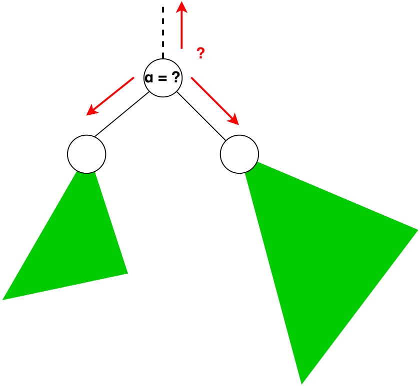
Meaning of α

$\alpha(p)$ is the distance from p to q where q its furthest process in $T(p)$ that is in the same cluster as p

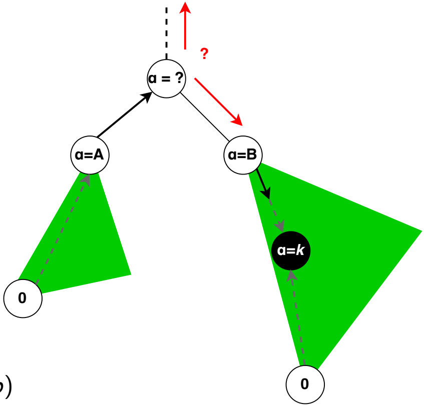


Bottom-up Computation

Computation of α

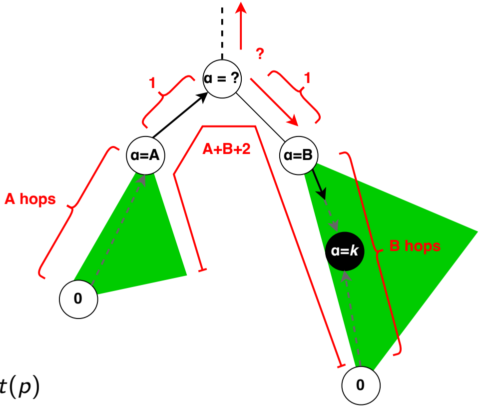


Computation of α



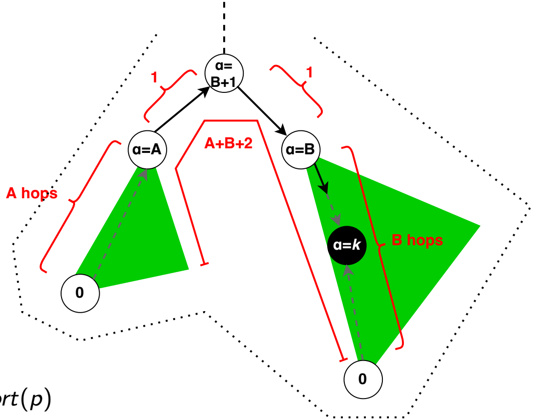
- $A = maxShort(p)$
- $B = minTall(p)$
- $0 \leq A < k \leq B \leq 2k$

Computation of α



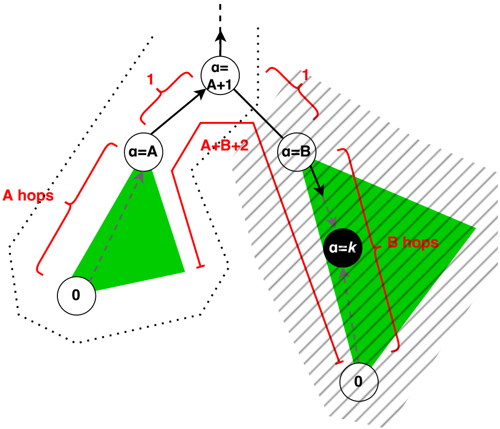
- $A = maxShort(p)$
- $B = minTall(p)$
- $0 \leq A < k \leq B \leq 2k$

Computation of α



- $A = maxShort(p)$
- $B = minTall(p)$
- $0 \leq A < k \leq B \leq 2k$

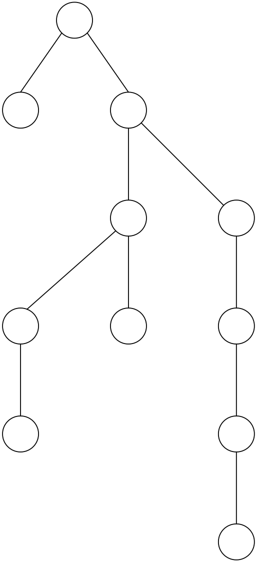
Case $A + B + 2 \leq 2k$



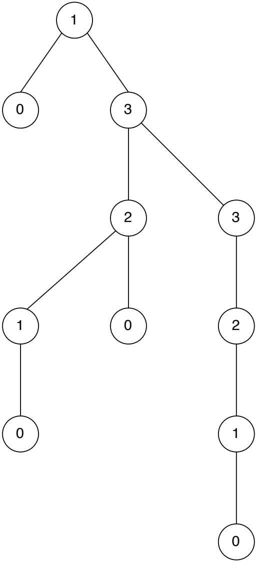
- $A = maxShort(p)$
- $B = minTall(p)$
- $0 \leq A < k \leq B \leq 2k$

Case $\boldsymbol{A + B + 2 > 2k}$

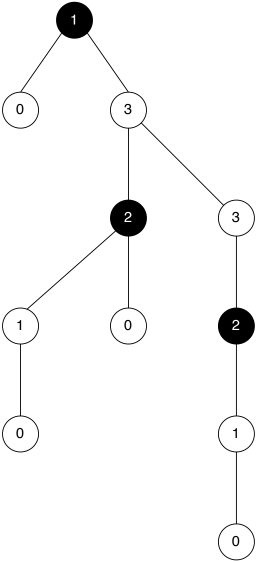
Example with $k = 2$



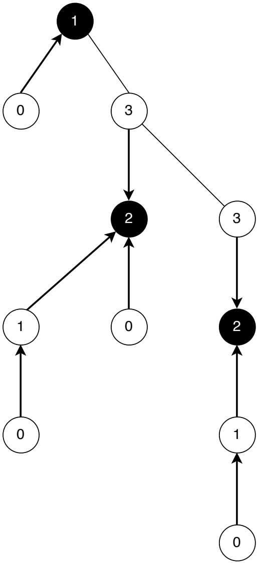
Example with $k = 2$



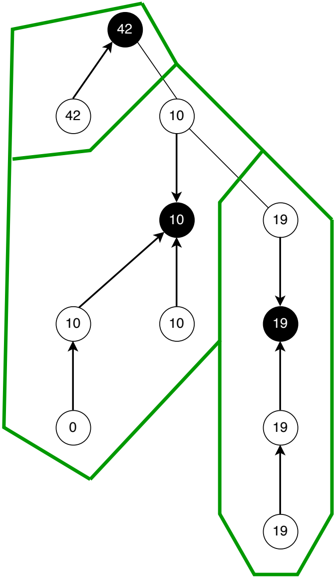
Example with $k = 2$



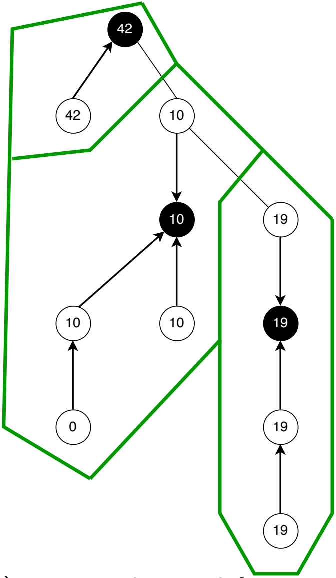
Example with $k = 2$



Example with $k = 2$



Example with $k = 2$



Remark: the (spanning) tree can be used for inter-cluster routing

Topologies

- Optimal in trees [1]
- Arbitrary Connected networks: at most $\lceil \frac{n}{k+1} \rceil$ clusters
- Unit Disk Graph (UDG): $7.2552k + O(1)$-approximation of the optimal
- Quasi Unit Disk Graph (QUDG): $7.2552\lambda^2 k + O(\lambda)$-approximation of the optimal

UDG and QUDG are models for Wireless Sensor Networks Topologies

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

3. Propagation of Information with Feedback in the tree: Feedback $\Rightarrow$ Bottom-up computation of α

   ($O(D)$ rounds, $O(n)$ messages of $O(\log k)$ bits, and $O(\log \Delta + \log k)$ bits per node)

Distributed Computation
In Arbitrary Connected Networks

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

3. Propagation of Information with Feedback in the tree: Feedback $\Rightarrow$ Bottom-up computation of α

   ($O(D)$ rounds, $O(n)$ messages of $O(\log k)$ bits, and $O(\log \Delta + \log k)$ bits per node)

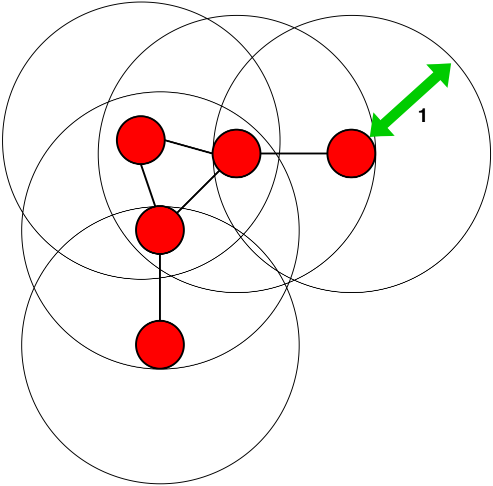
4. Propagation of Information with Feedback in the tree: broadcast $\Rightarrow$ cluster coloring, Feedback $\Rightarrow$ Termination Detection at the Leader

   ($O(D)$ rounds, $O(n)$ messages of $O(\log k + \log B)$ bits, and $O(\log B)$ bits per node)

5. Routing Set-Up

UDG: Unit Disk Graphs

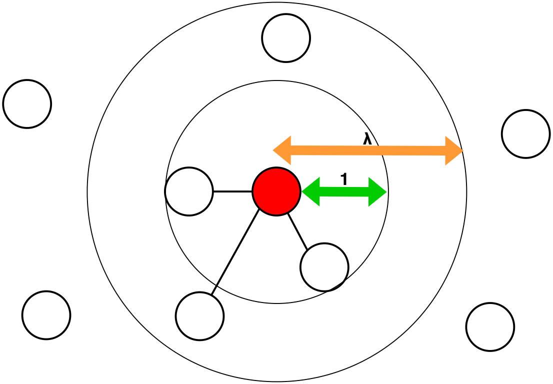
Nodes u and v are neighbors $\equiv \|u, v\| \leq 1$



QUDG: Quasi Unit Disk Graphs

Let $\lambda \geq 1$

- $\|u, v\| \leq 1 \Rightarrow u$ and v are neighbors
- u and v are neighbors $\Rightarrow \|u, v\| \leq \lambda$



Approximation of the optimal in UDGs

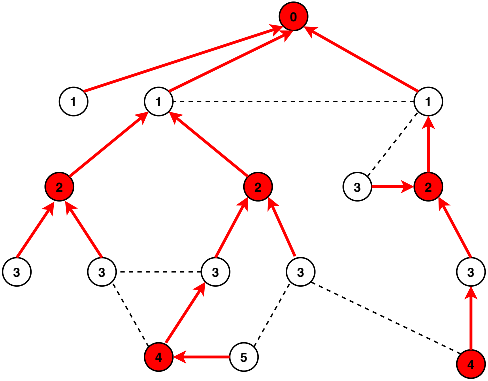
(at most M times the size of the optimal one)

Computing α on a MIS Tree of $G = (V, E)$

A spanning tree of G whose nodes at even level form a **maximal independent set**

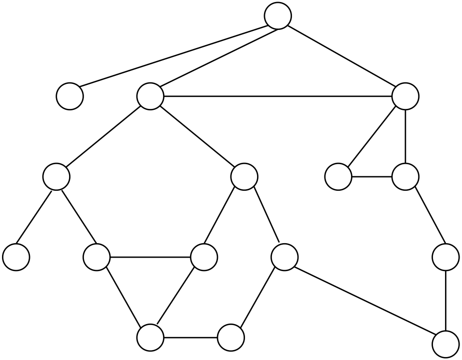
$S \subseteq V$ is a maximal independent set if

- S is independent: no two distinct nodes of S are neighbors in G
- S is maximal (by inclusion): no proper superset of S is independent



Approximation of the optimal in UDGs

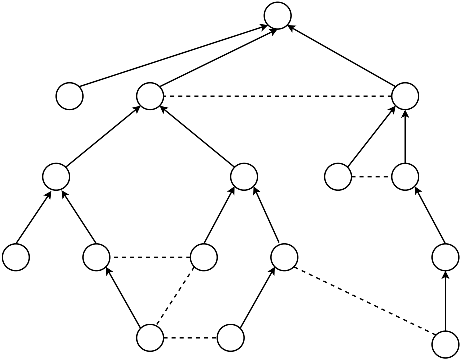
Compute a MIS Tree



A network

Approximation of the optimal in UDGs

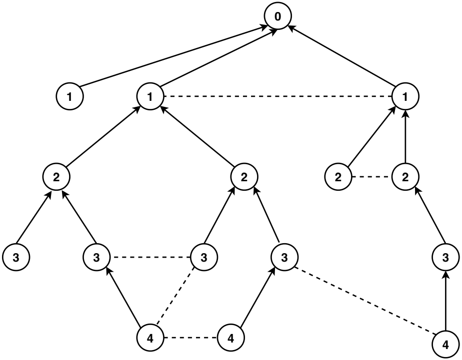
Compute a MIS Tree



BFS Spanning Tree

Approximation of the optimal in UDGs

Compute a MIS Tree

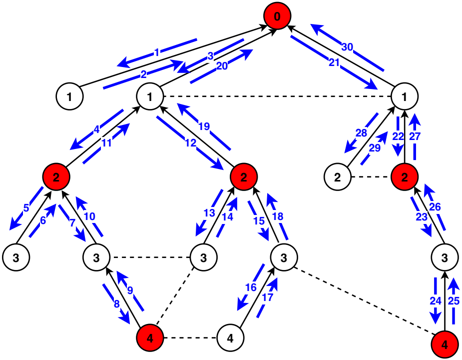


BFS Spanning Tree with levels

- $p.status \in \{In, Out\}$, initially Out
- DFS traversal of the spanning tree

At the 1st visit of node p:

if every neighbor q of p satisfies $q.status = Out$ **then**
  $p.status \leftarrow In$
else
  $p.status \leftarrow Out$
end if



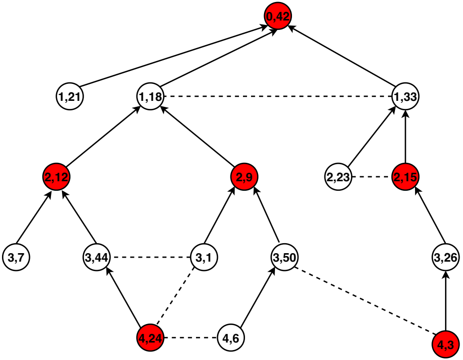
BFS Spanning Tree with MIS

- $p.status \in \{In, Out\}$, initially Out
- DFS traversal of the spanning tree

At the 1st visit of node p:

if every neighbor q of p satisfies $q.status = Out$ **then**
  $p.status \leftarrow In$
else
  $p.status \leftarrow Out$
end if

- **Key:** pair level/identifier
- **Total order on keys:** $(a,b) \prec (c,d) \equiv [a < c \vee (a = c \wedge b < d)]$



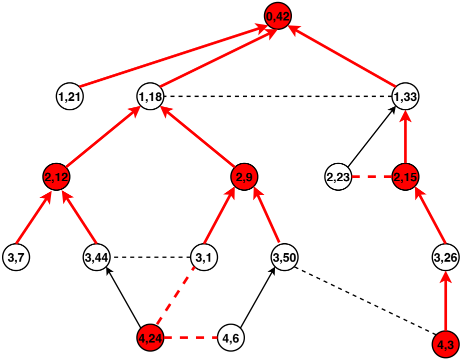
BFS Spanning Tree and MIS with keys

- $p.status \in \{In, Out\}$, initially Out
- DFS traversal of the spanning tree

At the 1st visit of node p:

if every neighbor q of p satisfies $q.status = Out$ **then**
  $p.status \leftarrow In$
else
  $p.status \leftarrow Out$
end if

- **Key:** pair level/identifier
- **Total order on keys:** $(a, b) \prec (c, d) \equiv [a < c \vee (a = c \wedge b < d)]$
- **Parent of non-root nodes:** the neighbor with a different status of smallest key



MIS tree

Approximation of the optimal in UDGs

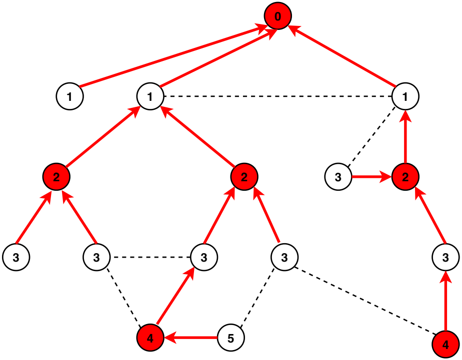
Compute a MIS Tree

- $p.status \in \{In, Out\}$, initially Out
- DFS traversal of the spanning tree

At the 1st visit of node p:

if every neighbor q of p satisfies $q.status = Out$ **then**
  $p.status \leftarrow In$
else
  $p.status \leftarrow Out$
end if

- **Key:** pair level/identifier
- **Total order on keys:** $(a, b) \prec (c, d) \equiv [a < c \vee (a = c \wedge b < d)]$
- **Parent of non-root nodes:** the neighbor with a different status of smallest key



MIS tree with levels

Approximation of the optimal in UDGs

Distributed computation of a MIS Tree

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

3. Token Circulation in the tree: first visit of a node $\Rightarrow$ collect status of neighbors (local message exchanges), then status assignment

   ($O(n)$ rounds, $O(m)$ messages of $O(1)$ bits, and $O(1)$ bits per node)

Approximation of the optimal in UDGs

Distributed computation of a MIS Tree

1. Leader Election

   ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits for messages and nodes, where B the number of bits to store an identifier)

2. BFS Spanning Tree Construction (with initialization and termination detection at the leader)

   ($O(D)$ rounds, $O(n.m)$ messages of $O(\log D)$ bits, and $O(\log D + \log \Delta)$ bits per node)

3. Token Circulation in the tree: first visit of a node $\Rightarrow$ collect status of neighbors (local message exchanges), then status assignment

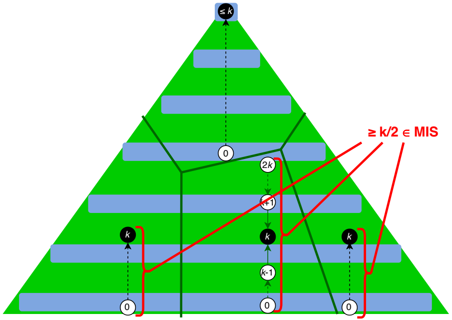
   ($O(n)$ rounds, $O(m)$ messages of $O(1)$ bits, and $O(1)$ bits per node)

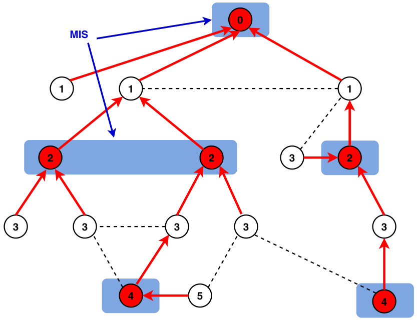
4. Propagation of Information with Feedback in the tree: Broadcast $\Rightarrow$ Collect of status and keys (local message exchanges), Feedback $\Rightarrow$ parent pointer assignment

   ($O(D)$ rounds, $O(m)$ messages of $O(\log k + \log B)$ bits, and $O(\Delta(\log k + \log B))$ bits per node)

5. Clustering computation on the MIS tree

Clr: partition into clusters



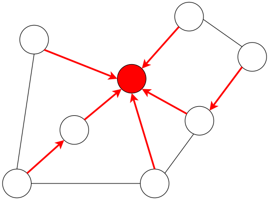


$$(|Clr| - 1).\frac{k}{2} \leq |MIS| - 1$$

Approximation of the optimal in UDGs

Independent Set *vs.* Optimal k-clustering Clr_{opt}

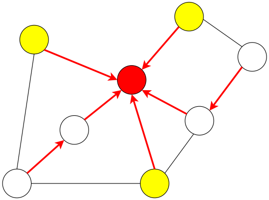
- Let C be any cluster of Clr_{opt}



Approximation of the optimal in UDGs

Independent Set *vs.* Optimal k-clustering Clr_{opt}

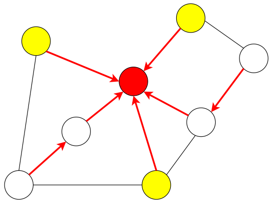
- Let C be any cluster of Clr_{opt}
- Let I be any independent set
- $\boldsymbol{I \cap C}$



Approximation of the optimal in UDGs

Independent Set *vs.* Optimal k-clustering Clr_{opt}

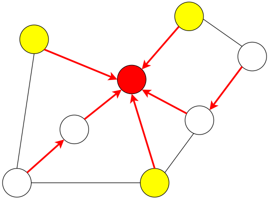
- Let C be any cluster of Clr_{opt}
- Let I be any independent set
- $\boldsymbol{I \cap C}$
- **UDG:**
  $\forall p, q \in I, p \neq p \Rightarrow \|p, q\| > 1$



Approximation of the optimal in UDGs

Independent Set *vs.* Optimal k-clustering Clr_{opt}

- Let C be any cluster of Clr_{opt}
- Let I be any independent set
- $\boldsymbol{I \cap C}$
- **UDG:**
  $\forall p, q \in I, p \neq p \Rightarrow \|p, q\| > 1$



Theorem (Folkman & Graham, [4])

Let X be a compact convex region. Let $Y \subseteq X$ s.t. $\forall p, q \in Y, (p \neq q \Rightarrow \|p, q\| \geq 1)$.

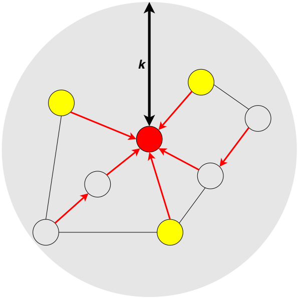
$$|Y| \leq \lfloor \frac{2.A(X)}{\sqrt{3}} + \frac{P(X)}{2} + 1 \rfloor$$

where $A(X)$ and $P(X)$ are respectively the area and perimeter of X.

Approximation of the optimal in UDGs

Independent Set *vs.* Optimal k-clustering Clr_{opt}

- Let C be any cluster of Clr_{opt}
- Let I be any independent set
- $\boldsymbol{I \cap C}$
- **UDG:** $\forall \boldsymbol{p}, \boldsymbol{q} \in \boldsymbol{I}, \boldsymbol{p} \neq \boldsymbol{p} \Rightarrow \|\boldsymbol{p}, \boldsymbol{q}\| > \mathbf{1}$



Theorem (Folkman & Graham, [4])

Let X be a compact convex region. Let $Y \subseteq X$ s.t. $\forall p, q \in Y, (p \neq q \Rightarrow \|p, q\| \geq 1)$.

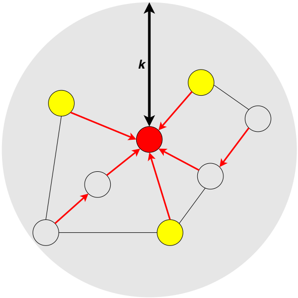
$$|Y| \leq \lfloor \frac{2.A(X)}{\sqrt{3}} + \frac{P(X)}{2} + 1 \rfloor$$

where $A(X)$ and $P(X)$ are respectively the area and perimeter of X.

Approximation of the optimal in UDGs

Independent Set *vs.* Optimal k-clustering Clr_{opt}

- Let C be any cluster of Clr_{opt}
- Let I be any independent set
- $\boldsymbol{I \cap C}$
- **UDG:** $\forall \boldsymbol{p}, \boldsymbol{q} \in \boldsymbol{I}, \boldsymbol{p} \neq \boldsymbol{p} \Rightarrow \|\boldsymbol{p}, \boldsymbol{q}\| > \mathbf{1}$



$A(X) = \pi.k^2$, $P(X) = 2\pi.k$

- $|I \cap C| \leq \lfloor \frac{2.\pi.k^2}{\sqrt{3}} + \pi.k + 1 \rfloor$
- $|I| \leq \lfloor \frac{2.\pi.k^2}{\sqrt{3}} + \pi.k + 1 \rfloor . |Clr_{opt}|$

Result

- $|MIS| \leq \lfloor \frac{2.\pi.k^2}{\sqrt{3}} + \pi.k + 1 \rfloor.|Clr_{opt}|$
- $(|Clr| - 1).\frac{k}{2} \leq |MIS| - 1$

$\Rightarrow |Clr| \leq 1 - \frac{2}{k} + (\frac{4\pi.k}{\sqrt{3}} + 2\pi + \frac{2}{k}).|Clr_{opt}|$

$\Rightarrow 7,2552k + O(1)-$competitive in UDG

Generalization to QUDG: $7,2552\lambda^2 k + O(1)-$competitive in QUDG

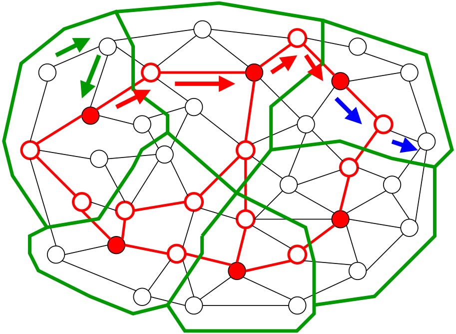
Roadmap

Routing from p to q

Address: (label of q's clusterhead,label of q)

Phases: 3 phases distinguished using **3 colors**.

1. Route the packet from p to its clusterhead (intra-clustering)
2. Route the packet from the clusterhead of p to the clusterhead of q (inter-clustering)
3. Route the packet from the clusterhead of q to q (intra-clustering)



Different routing methods can be used for each phase, *e.g.*,

1. Parent links to route from p to its clusterhead
2. Shortest paths[1] to route from the clusterhead of p to the clusterhead of q
3. Interval routing from the clusterhead of q to q

[1] in the network or in the tree computed during the clustering computation

Shortest Paths for inter-cluster routing

GLOBAL:

1. The leader launches a PIF to collect the identifiers of all clusterheads
2. The leader launches a PIF to broadcast the identifiers of all clusterheads
3. For each node p and each clusterhead q:
   - $Par_p[q] \leftarrow p$
   - if $p \neq q$, $D_p[q] \leftarrow \infty$, else $D_p[q] = 0$

LOCAL:

1. Each node p creates cells $Par_p[q]$ and $D_p[q]$ for its clusterhead q and informs its neighbors of the existence of q
2. $Par_p[q] \leftarrow p$ and if $p \neq q$, $D_p[q] \leftarrow \infty$, else $D_p[q] = 0$
3. If p learns the existence of some clusterhead r, p creates cells $Par_p[r]$ and $D_p[r]$, initialized to p and ∞ resp.

4. Bellman-Ford: for each node p, each clusterhead q, and every neighbor v of p
   - if $D_p[q] > D_v[q] + 1$, then $D_p[q] \leftarrow D_v[q] + 1$; $Par_p[q] \leftarrow v$

See [6] for further details.

Using the Spanning Tree for inter-cluster routing

Thank to the cluster colors, local detection of tree edge $\{p, q\}$ between two different clusters

1. Add $\{p, q\}$ to the subgraph connecting clusterheads
2. Add the cluster path from p to its clusterhead to the subgraph connecting clusterheads
3. Add the cluster path from q to its clusterhead to the subgraph connecting clusterheads

We obtain a tree

Using the Spanning Tree for inter-cluster routing

Thank to the cluster colors, local detection of tree edge $\{p, q\}$ between two different clusters

1. Add $\{p, q\}$ to the subgraph connecting clusterheads
2. Add the cluster path from p to its clusterhead to the subgraph connecting clusterheads
3. Add the cluster path from q to its clusterhead to the subgraph connecting clusterheads

We obtain a tree

Apply the tree labeling scheme or an PLS on the tree to allow routing between clusters

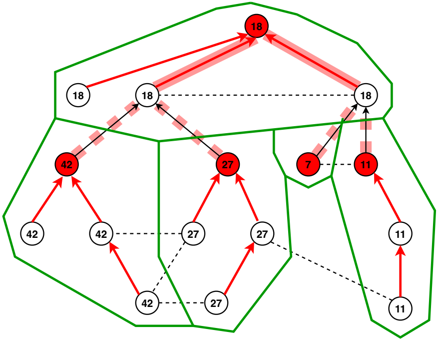
Using the Spanning Tree for inter-cluster routing

Thank to the cluster colors, local detection of tree edge $\{p, q\}$ between two different clusters

1. Add $\{p, q\}$ to the subgraph connecting clusterheads
2. Add the cluster path from p to its clusterhead to the subgraph connecting clusterheads
3. Add the cluster path from q to its clusterhead to the subgraph connecting clusterheads

We obtain a tree

Apply the tree labeling scheme or an PLS on the tree to allow routing between clusters



Clustering of [3]

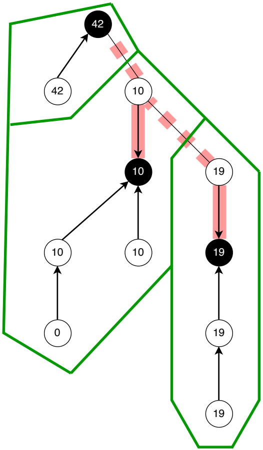
Using the Spanning Tree for inter-cluster routing

Thank to the cluster colors, local detection of tree edge $\{p, q\}$ between two different clusters

1. Add $\{p, q\}$ to the subgraph connecting clusterheads
2. Add the cluster path from p to its clusterhead to the subgraph connecting clusterheads
3. Add the cluster path from q to its clusterhead to the subgraph connecting clusterheads

We obtain a tree

Apply the tree labeling scheme or an PLS on the tree to allow routing between clusters



Clustering of [1]

Optimal subgraph for the Inter-cluster Routing
Steiner Tree

Non-negative edge weights.

Subset of nodes: set of terminals

Compute a tree of minimum weight that

- contains all terminals (but may include additional vertices) and
- minimizes the total weight of its edges.

Here, terminals are clusterheads

However, **$\mathcal{NP}$-hard [5]**

But approximation algorithms exists!

Roadmap

References

[1] A. K. Datta, S. Devismes, K. Heurtefeux, L. L. Larmore, and Y. Rivierre.
Competitive self-stabilizing k-clustering.
Theor. Comput. Sci., 626:110–133, 2016.

[2] A. K. Datta, S. Devismes, and L. L. Larmore.
A silent self-stabilizing algorithm for the generalized minimal k-dominating set problem.
Theor. Comput. Sci., 753:35–63, 2019.

[3] A. K. Datta, L. L. Larmore, S. Devismes, K. Heurtefeux, and Y. Rivierre.
Self-stabilizing small k-dominating sets.
Int. J. Netw. Comput., 3(1):116–136, 2013.

[4] J. H. Folkman and R. L. Graham.
A packing inequality for compact convex subsets of the plane.
Canadian Mathematical Bulletin, 12(6):745–752, 1969.

[5] M. R. Garey and D. S. Johnson.
Computers and Intractability; A Guide to the Theory of NP-Completeness.
W. H. Freeman & Co., USA, 1990.

[6] G. Tel.
Introduction to Distributed Algorithms.
Cambridge University Press, USA, 2nd edition, 2001.