Self-Stabilizing Minimum Degree Spanning Tree within one from the optimal

L. Blin, M. Potop-Butucaru,
S. Rovedakis

ibisc LIP6 INRIA

Motivations

- High degree nodes yield undesirable effects
  - In general networks
    - High congestion
    - High attack probability
  - In Ad-hoc networks
    - More collisions => low bandwidth
- Theoretical insterst

State of the arts

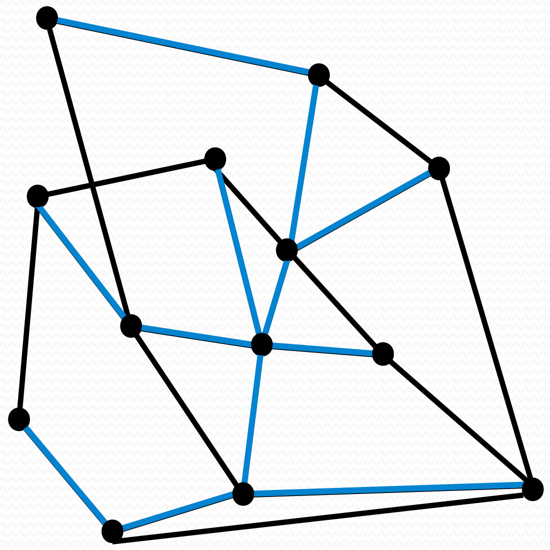
Self-stabilizing tree constructions

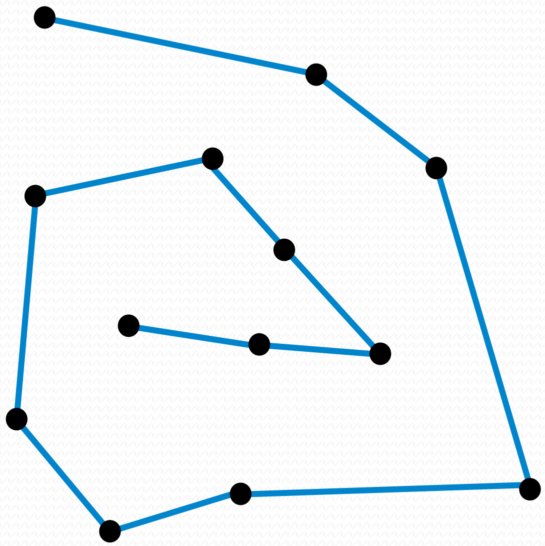
- Breadth First Search trees (BFS):
  - [**Y. Afek**, **S. Kutten** and **M. Yung**, WDAG, 1991]
  - [**S. Dolev**, **A. Israeli** and **S. Moran**, PODC, 1990]
- Depth First Search trees (DFS):
  - [**Z. Collin** and **S. Dolev**, IPL, 1994]
  - [**T. Herman**, PhD thesis, 1991]
- Minimum Diameter Spanning trees:
  - [**J. Burman** and **S. Kutten**, DISC, 2007]
  - [**F. Butelle**, **C. Lavault** and **M. Bui**, WDAG, 1995]
- Minimum weight Spanning Trees (MST):
  - [**L. Higham** and **Z. Liang**, DISC, 2001]
  - [**G. Antonoiu** and **P.K. Srimani**, Euro-par, 1997]
- Shortest Paths trees:
  - [**S. Delaët**, **B. Ducourthial** and **S. Tixeuil**, SSS, 2005]

MDST problem

- Minimum Degree Spanning Tree (MDST):
  - G=(V,E) is a unweighted undirected graph
  - $\Delta(T)$ is the maximum degree of subgraph T
- The goal is
  1. to construct a tree T spanning V,
  2. minimizing $\Delta(T)$.

NP-Hard problem





Hamiltonian path (NP-Hard)

Outline

- Sequential algorithm for MDST problem

  [**M. Fürer** and **B. Raghavachari**, SODA, 1992]
- Introduction to Self-Stabilization paradigm
- Our Self-Stabilizing algorithm for MDST problem

Sequential Algorithm for the MDST problem

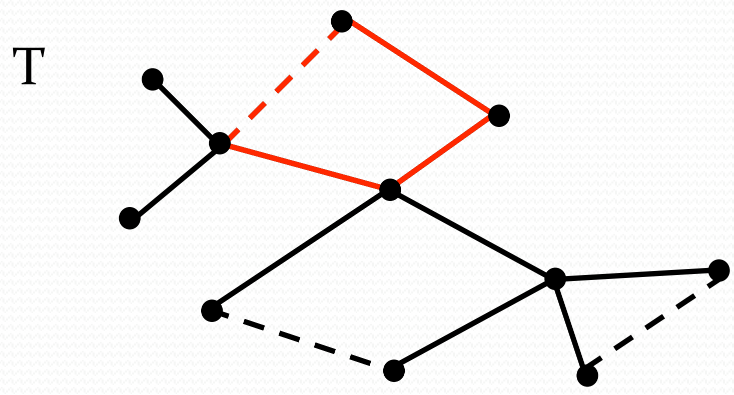
Problem approximation

- NP-hard problem (Hamiltonian path)
- We seek an approximation
  - Δ^*: maximum degree of an optimal solution
  - Best approximation: $\Delta(T) \leq \Delta^*+1$
    [**M. Fürer** and **B. Raghavachari**, SODA, 1992]

- **Algorithm [FR92] :**
  - Initial state: an arbitrary spanning tree,
  - Perform every possible improvement (edge swap).

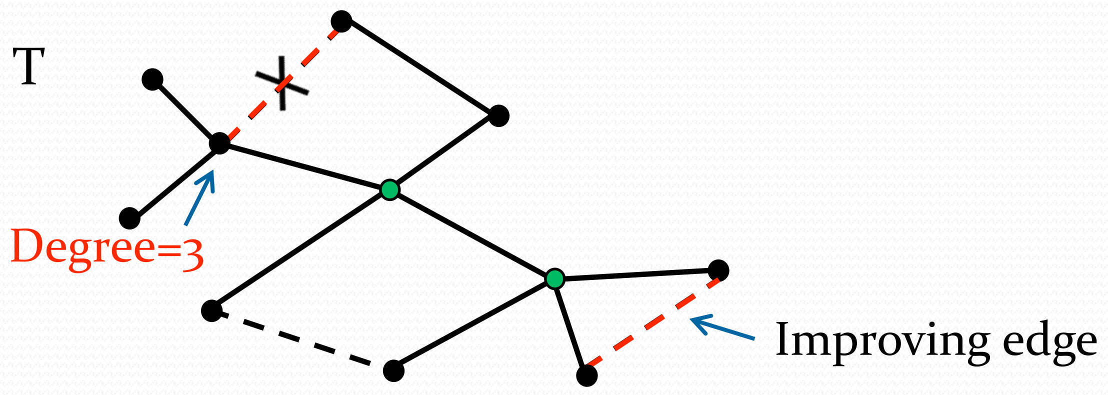
Definitions: Fundamental cycle

- **Fundamental cycle:** cycle in T created by the add of the non tree edge (u,v) to T, noted C(u,v).



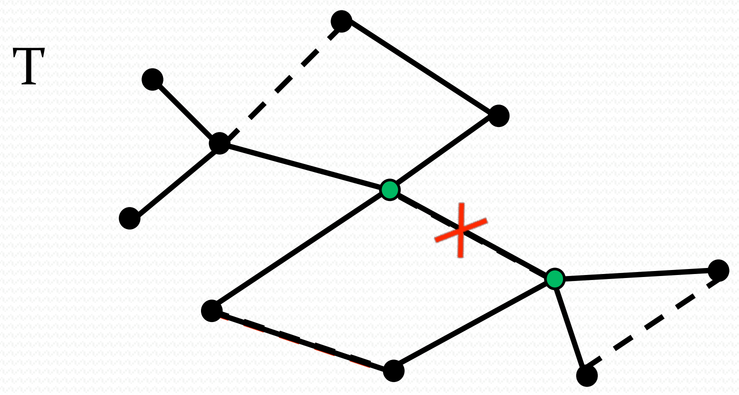
Definitions: Improving edge

- **Improving edge:** edge (u,v) not in T, such that $\max\{deg(u), deg(v)\} < \Delta(T)-1$ and node w in C(u,v) with $deg(w)=\Delta(T)$.



Definitions: Improvement

- **Improvement:** swap between an improving edge and an edge adjacent to a maximum degree node (i.e. a node with a degree equal to $\Delta(T)$).

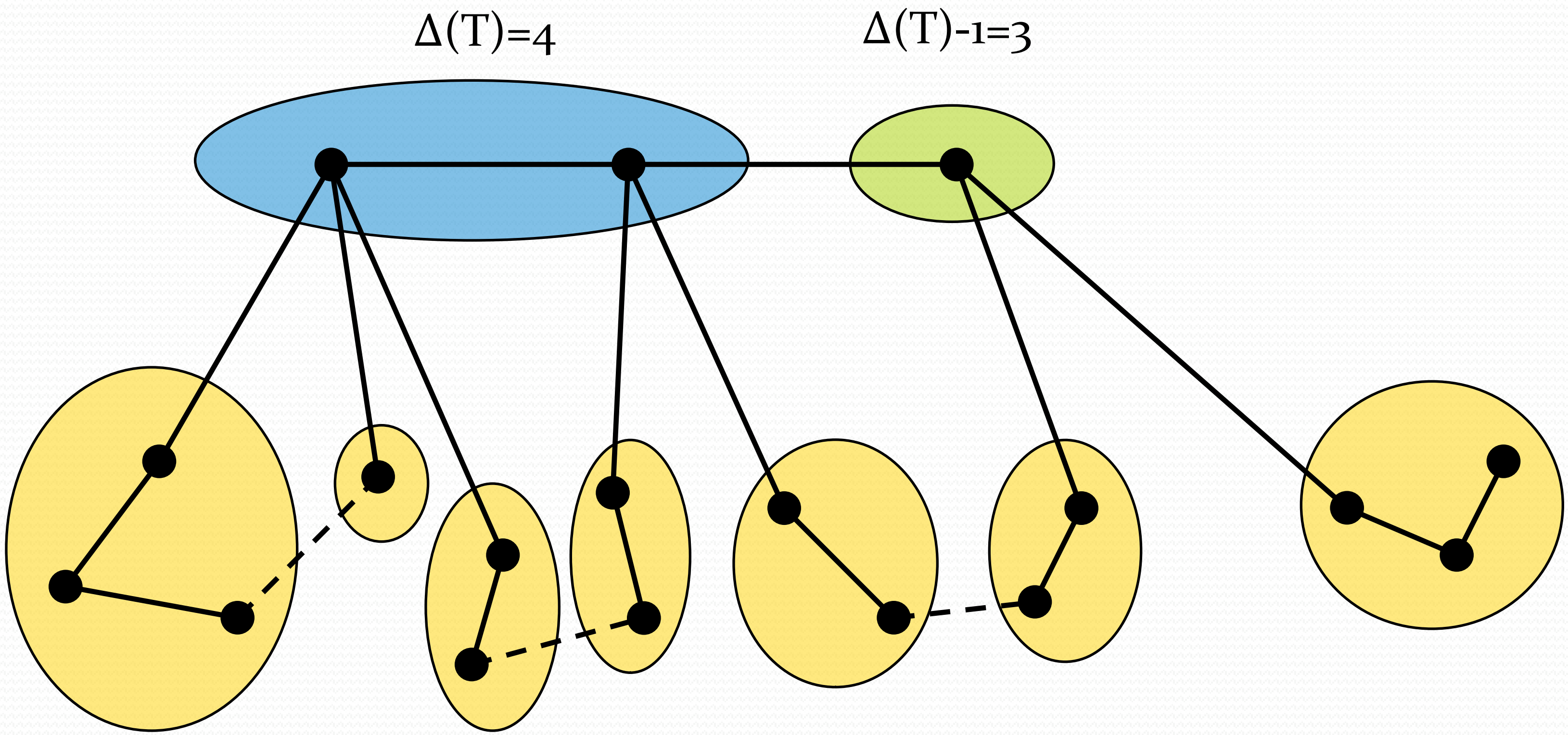


- Decrease by 1 the degree of a maximum degree node.

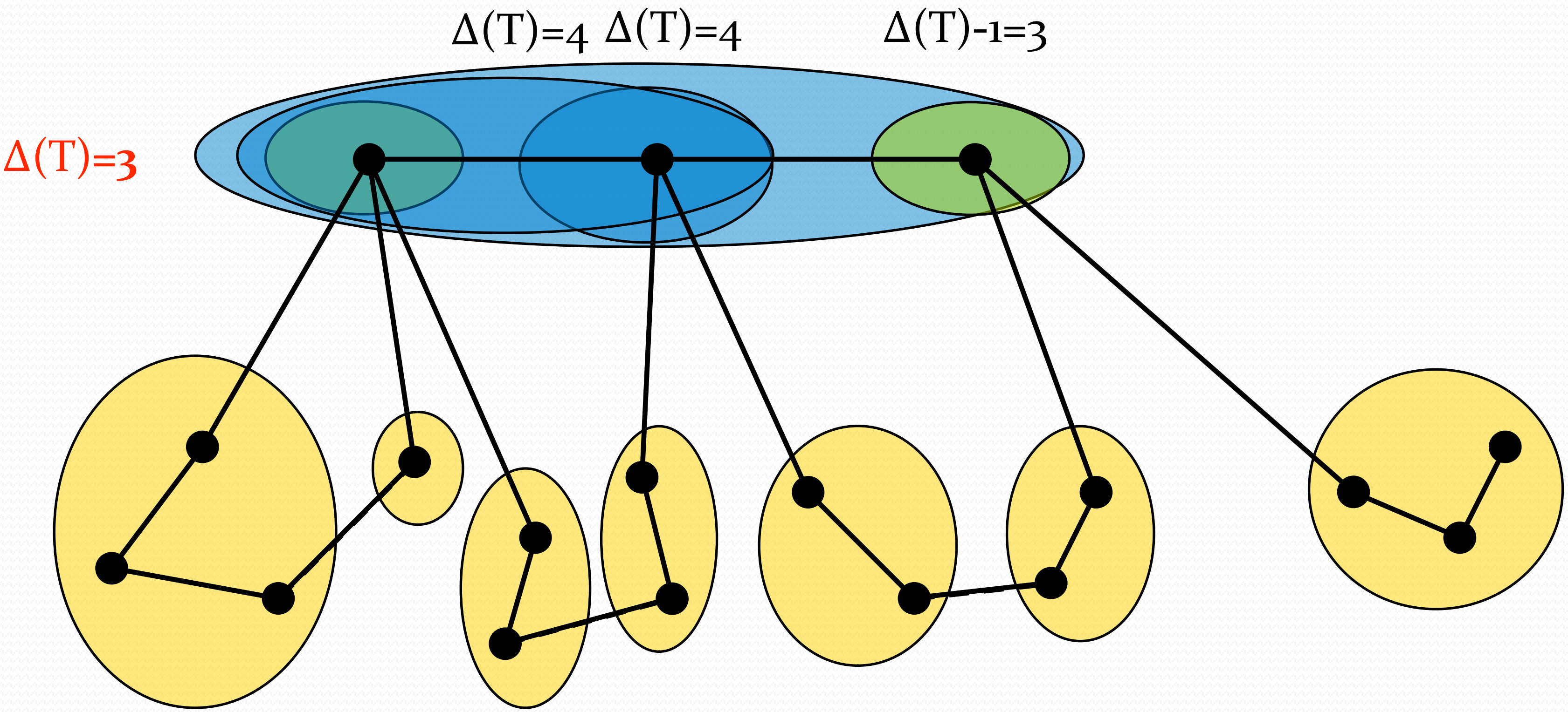
Sequential algorithm

- Initially, we start with an arbitrary spanning tree
- Until there is no improvement, run a new phase
- A phase of algorithm:
  - Compute the maximum degree of the current tree T (i.e. computation of $\Delta(T)$)
  - Perform an improvement

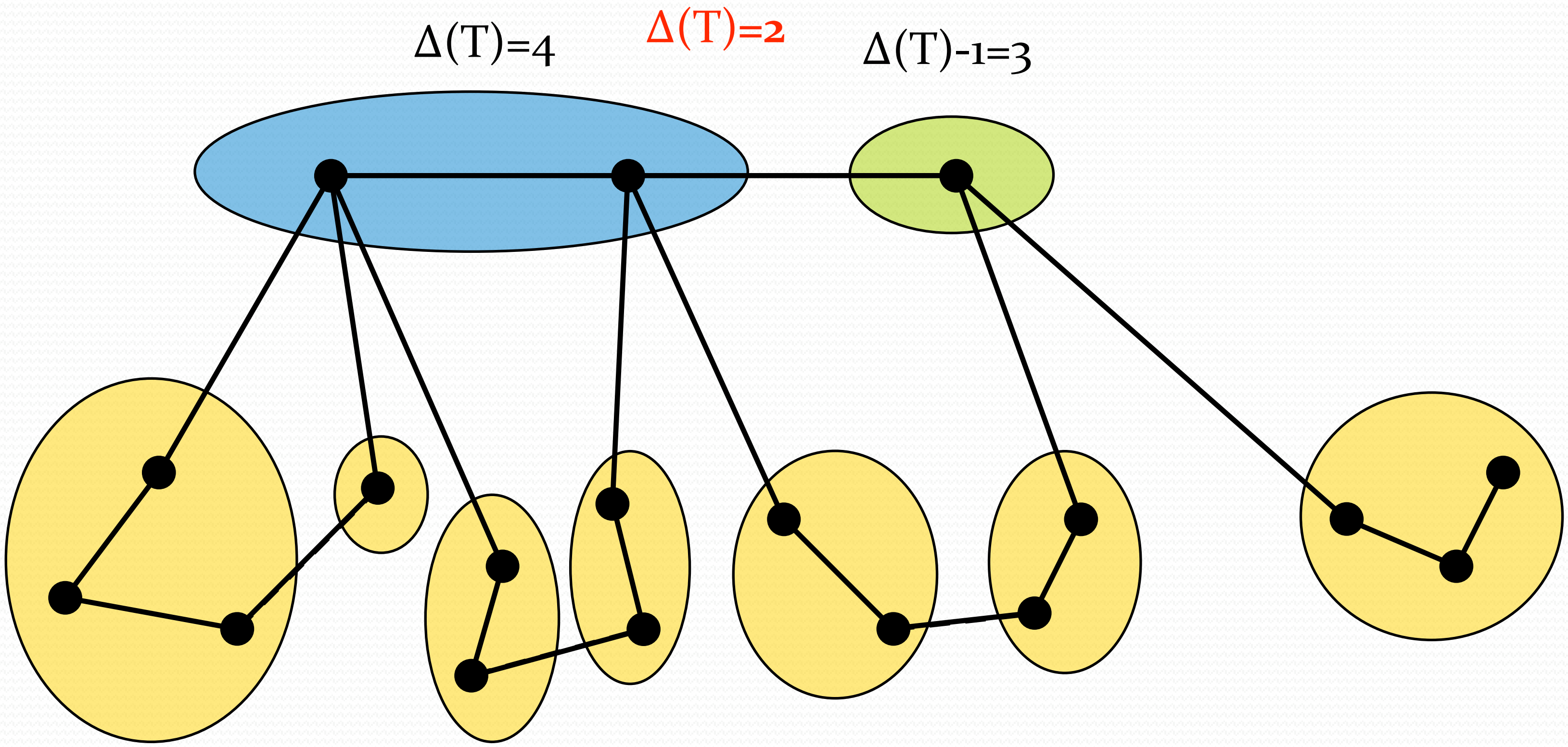
Example



Example



Example



Distributed algorithm: [**L. Blin** and **F. Butelle**, IPDPS, 2003]

Self-Stabilization paradigm

Self-Stabilizing Systems

- **Fault:** event which corrupts
  - memory (variables),
  - program counter,
  - Communication channels of nodes in the network.

- **Goal:** A self-stabilizing system handle transient faults (Dijkstra, 74)

- **Legitimate configuration:** system configuration (*composition of local states*) in which each node state satisifies P (*a desired property*).

a Self-Stabilizing Algorithm for the MDST problem

Model

- Distinct identifiers
- Asynchronous protocol (fine grained atomicity)
  - Message passing
- Distributed system
  - Network = set of interconnected computers (nodes)
  - FIFO and bidirectional channels
  - State of a node = its variables
  - System configuration = Local states of all nodes
  - Local vision of the system (no global information)

Self-Stabilizing algorithm

- Composition of 3 self-stabilizing layers:
  1. Construction of a spanning tree
  2. Maximum degree computation of the current tree
  3. Reduction of the maximum degree

Construction and maintaining of a spanning tree

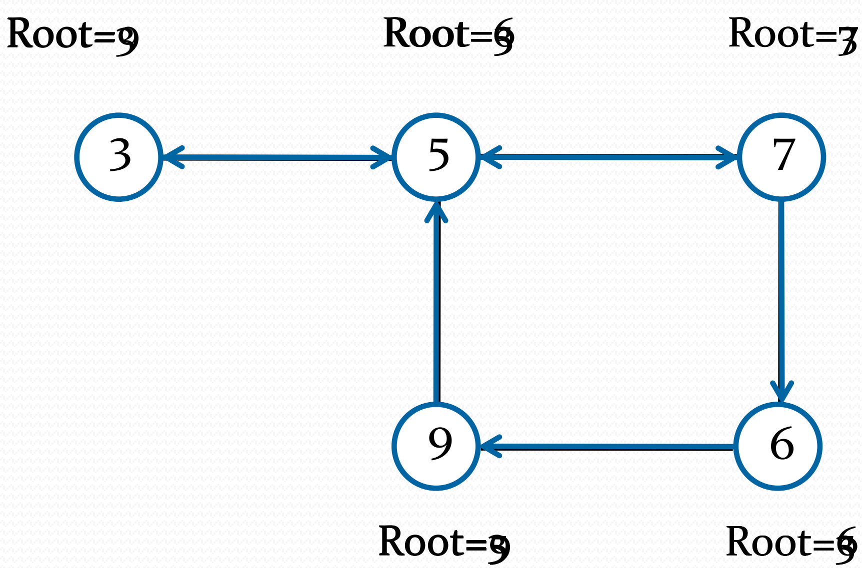
- Root of the tree = node of minimum id
- Variables for node u: $root_u$, $parent_u$

- Rules: [**Afek**, **Kutten** and **Yung**, WDAG, 1991]
  - Coherent(u) : $parent_u \in N(u) \cup \{u\}$ and $root_u = root_{parent_u}$
  - BetterRoot(u) : $v \in N(u)$, $root_v < root_u$

  - Update:

    If Coherent(u) and BetterRoot(u) => u changes root
  - Init. State: If ¬Coherent(u) => u becomes a new root

Construction and maintaining of a spanning tree



No rule can be executed

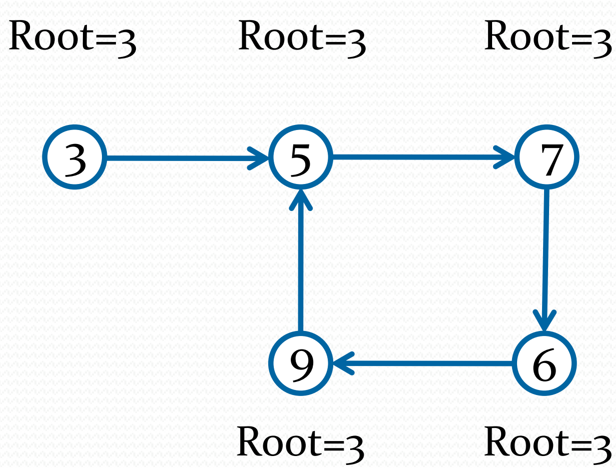
Update Rule

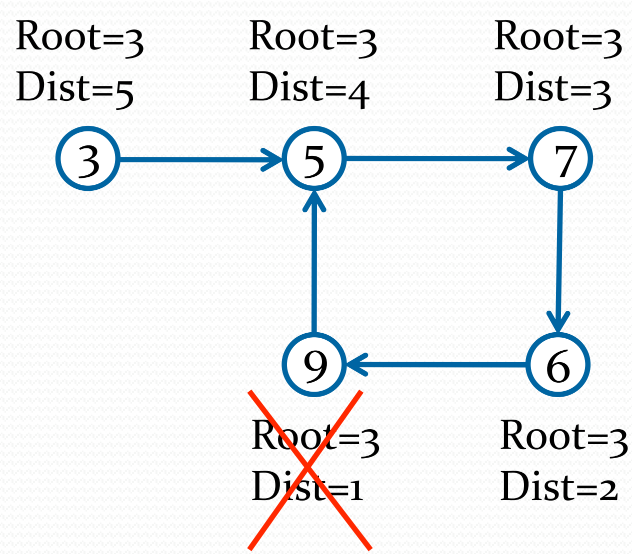
Init. State Rule

(Update) If Coherent(u) and BetterRoot(u) => u changes root
(Init. State) If ¬Coherent(u) => u becomes a new root

Construction and maintaining of a spanning tree

- Need of cycle deletion:
  - each node maintains its distance to the root (root: dist=0, others: parent dist+1)



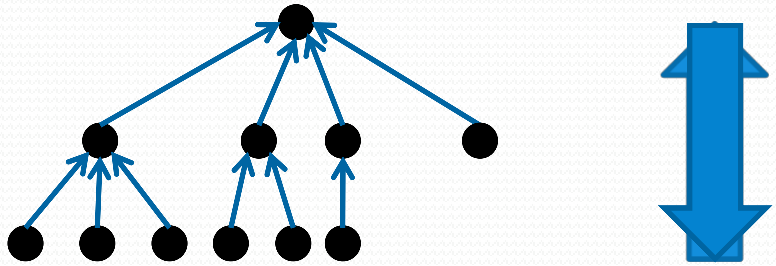


Computation of max degree

- max_degree$_u$: maximum degree of the current tree
- If ¬Coherent(u)
  - max_degree$_u$ = degree of u in the tree
- Otherwise
  - Use of PIF protocol (Propagation of Information with Feedback)

[**Blin**, **Cournier**, **Villain**, SSS, 2003],

[**Cournier**, **Datta**, **Petit**, **Villain**, J. High Speed Networks, 2005]



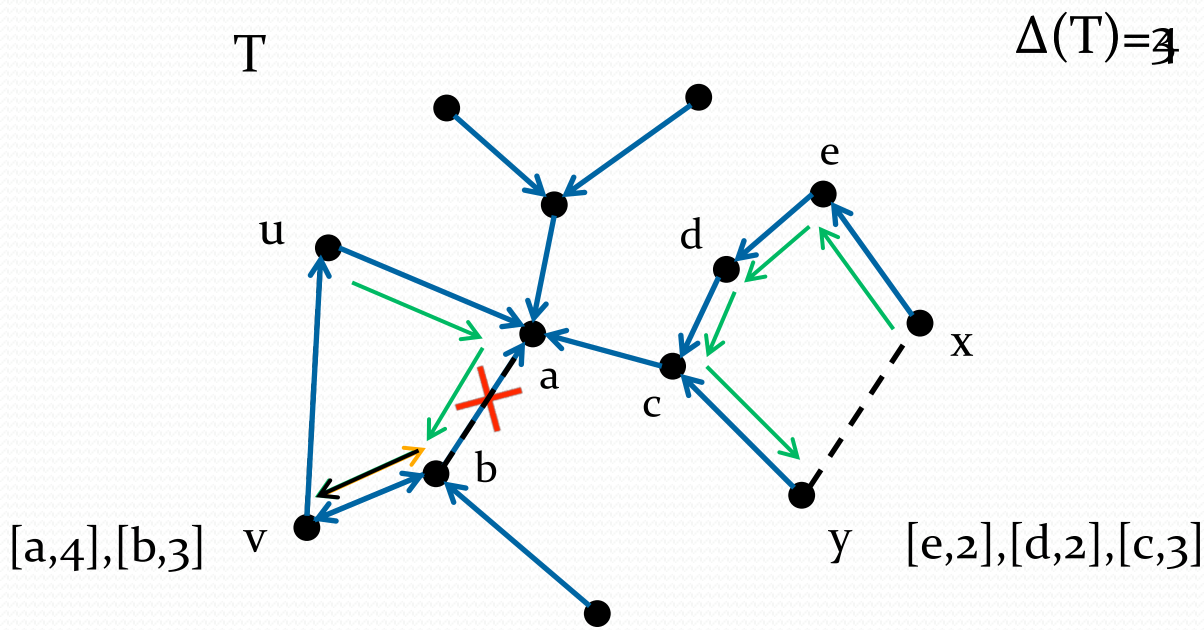
Propagation phase : Dissemination of maximum node degree

Feedback phase : Selection of maximum node degree

Reduction of max degree

- Works like [FR92], but for all fundamental cycles
- Let a tree T, for each edge (u,v) $\notin$ T :
  - Find its fundamental cycle (DFS search),
  - Check if (u,v) is an improving edge,
  - Perform an improvement (if improving edge).
- Each edge is managed by the node with minimum id

Reduction of max degree



(x,y) is not an improving edge

: Search

: Remove

: Back

Module Composition

- An upper layer must not destabilize a lower one
- Max. degree layer:
  - does not change $parent_u$ and $dist_u$
- Degree reduction layer:
  - reduces the degree (higher degree => no improvement)
  - changes $parent_u$ (update distance to maintain tree coherency)

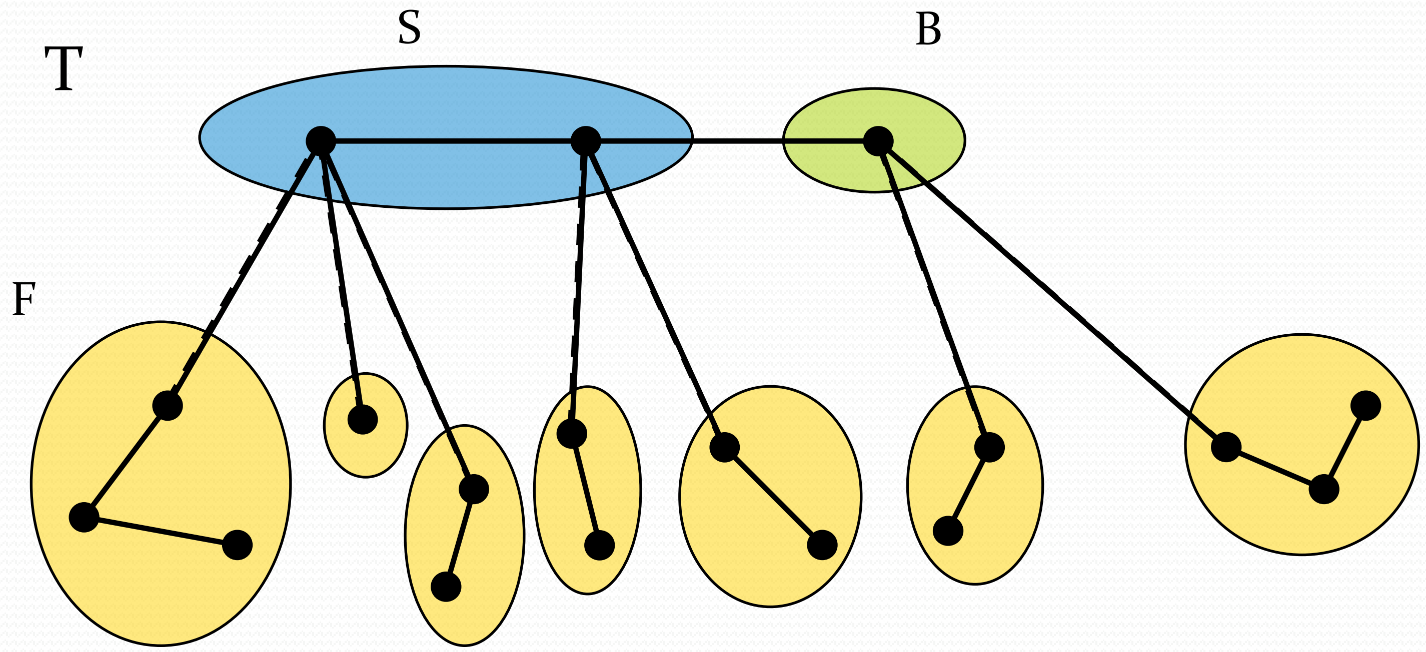
Conclusion and perspectives

- We propose a self-stabilizing algorithm resolving the MDST problem
  - Best approximation (unless P=NP): $\Delta(T) \leq \Delta^{*}+1$
- Extension:
  - Steiner tree [FR92]
  - Oriented Graphs
    - Approximation: $\Delta(T) \leq \Delta^{*}+\log n$
      [**R. Krishnan** and **B. Raghavachari**, FST TCS, 2001]

Thank you

Approximation proof (1/2)

Theorem [FR92]: If G contains no edge between trees in F, then $\Delta(T) \leq \Delta^{*}+1$.



Approximation proof (2/2)

- Lemma [FR92]:

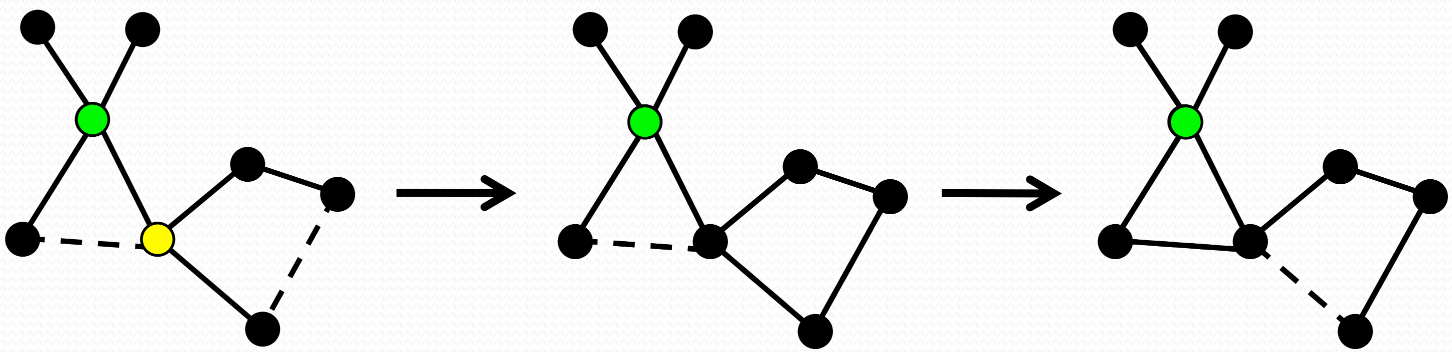
  When algorithm completes, $\Delta(T) \leq \Delta^*+1$.

- Proof:
  - The algorithm stops only if there is no edge between trees in F.
  - T satisfies the conditions of the previous theorem and thus we have $\Delta(T) \leq \Delta^*+1$.

- Remark: The set $S \cup B$ is a witness set which allows to check that $\Delta(T) \leq \Delta^*+1$.

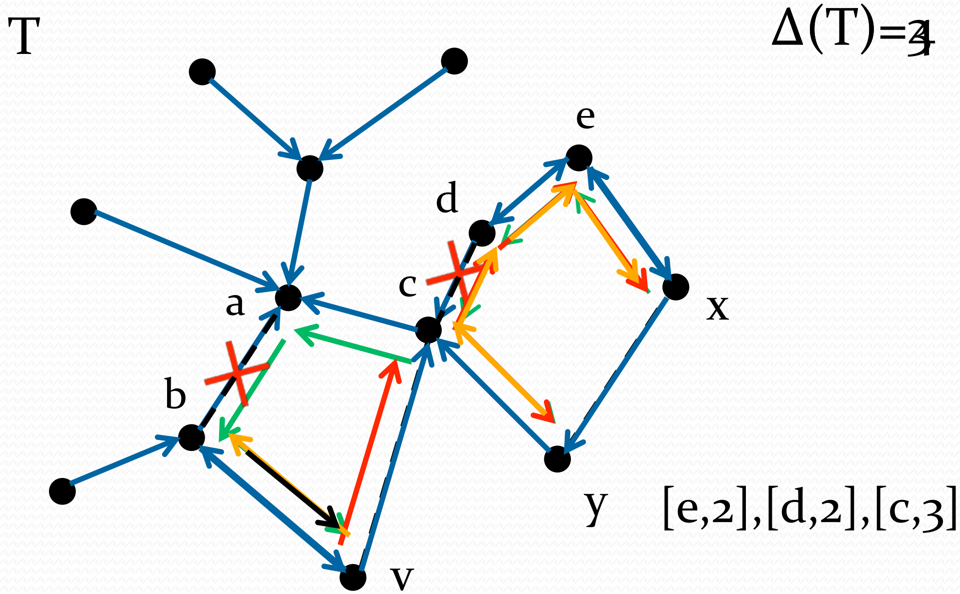
Sequential algorithm

- **Blocking node:** node u with degree deg(u)=Δ(T)-1, such that edge (u,v) not in T and node w in C(u,v) with degree deg(w)=Δ(T).



Δ(T)=4 Δ(T)=4 Δ(T)=3

Reduction of max degree



[a,4],[b,3]

(x,y) is not an improving edge

(c,v) is not an improving edge because (c is a blocking node)

→ : Search

→ : Remove

→ : Back

→ : Deblock