Distributed Backup Placement in WSNs and Planar Graphs

Gal Oren & Leonid Barenboim

Department of Computer Science, Ben-Gurion University of the Negev
Department of Mathematics and Computer Science, The Open University of Israel
Department of Physics, Nuclear Research Center-Negev

Introduction

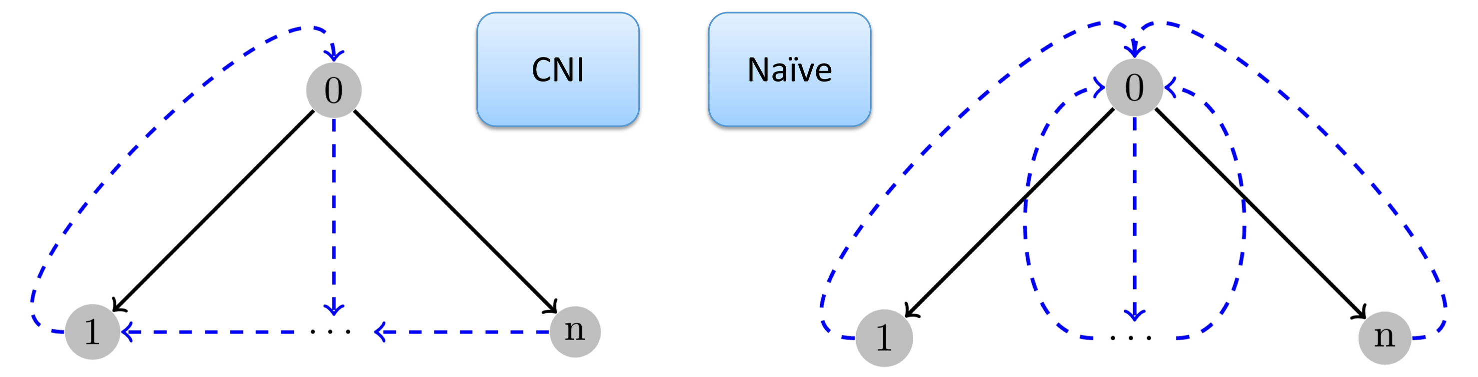
- The **backup-placement problem** was introduced by Halldórsson et al. in 2015. This problem turned out to be **very challenging** in general networks.
- We focus on wireless networks, specifically looking into solutions that are **significantly better than polynomial (and even than linear) solutions**.
- Scenario example: several nodes in a network have packages whose backups (or the package itself) need to find a placement elsewhere in the network, due to overload in these nodes areas, **in order to improve fault-tolerance and data integrity**.
- Because of the lack of capacity to store or process the data on the node itself, it is mandatory, when the local memory is full, to **find a backup-placement to the data outside of the node**. The backup-placement problem is defined as follows:
  - How to place the data **only once** in a safe and stable node in order to assure with a high degree of certainty the data integrity and minimization of the network load.
  - How to do so **without creating an additional data overflow** on other areas of the network.

Backup placement in trees

- The procedure receives a tree *T = (V,E)* as input :
  - **Algorithm 1** it the naïve one, which computes an *O(1)*-backup placement of *T*.
  - **Algorithm 2** is based on constant neighborhood, with *O(1)* time complexity.

Algorithm 1 Naive Distributed Tree Backup Placement Algorithm in $O(1)$

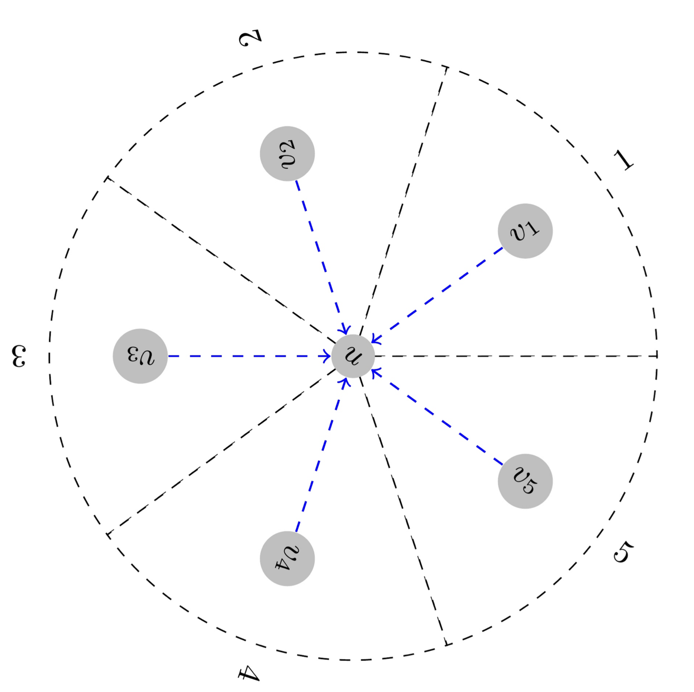
procedure NAIVE-TREE-BP(NODE $v \in T$, TREE T)
  if v is not a leaf **then**
    $v.BP \leftarrow Arbitrary(v.children)$.
  else
    $v.BP \leftarrow v.Parent$



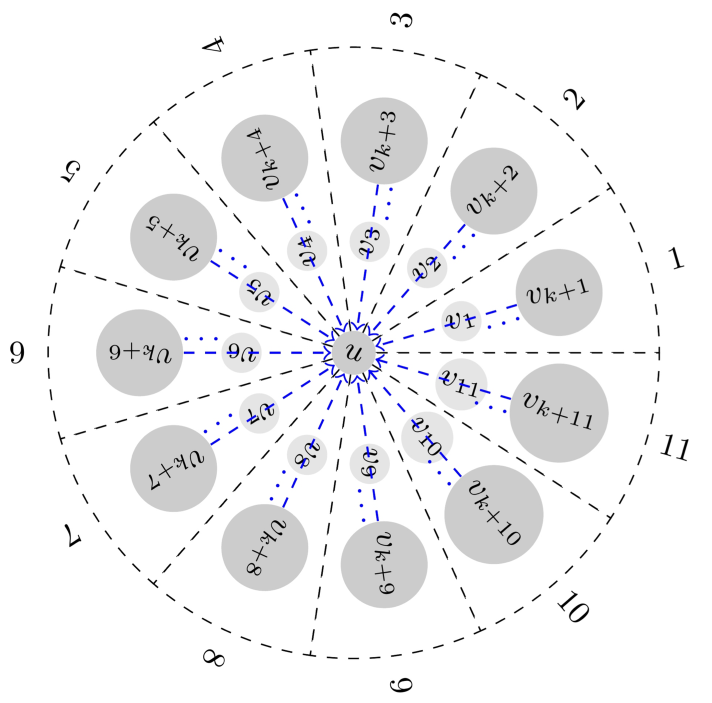
Algorithm 2 Constant Neighborhood Independence Distributed Tree Backup Placement Algorithm in $O(1)$

procedure CNI-TREE-BP(NODE u, GRAPH G), SUBGRAPH T
  if v is not a leaf **then**
    $v.BP \leftarrow Arbitrary(v.children)$.
  else if $\exists w$ sibling of v $(ID(w) < ID(v)) \wedge \nexists z$ sibling of v $(ID(w) < ID(z) < ID(v)) \wedge (v,w),(w,z) \in E(G)$ **then**
    $v.BP \leftarrow w$.
  else
    $v.BP \leftarrow v.Parent$.

- After we proved that each vertex of *G* is selected by at most *c* vertices of *T* if *G* is a graph with neighborhood independence of at most *c*, and *T* is a subtree of *G*, we can now prove that **in the case of wireless networks the parameter c is a constant**, based on the properties of the *UDG* in case of a homogeneous network *c=5*.
- We can also prove that even in case of a **heterogeneous network**, i.e. a network in which all nodes radii are different, based on the properties of the bounded disk graph (*BDG*), **parameter c is small: *c=11log(Rmax/Rmin)*.**



Unit Disk Graph model
A root node v_0, which forms 5 different cliques, and $\{v_1, \ldots, v_5\}$ nodes which must choose v_0 as their backup placement dest'



Bounded Disk Graph model
A root node v_0, which forms $c=11log(R_{max}/R_{min})$ different cliques

Backup placement in forests

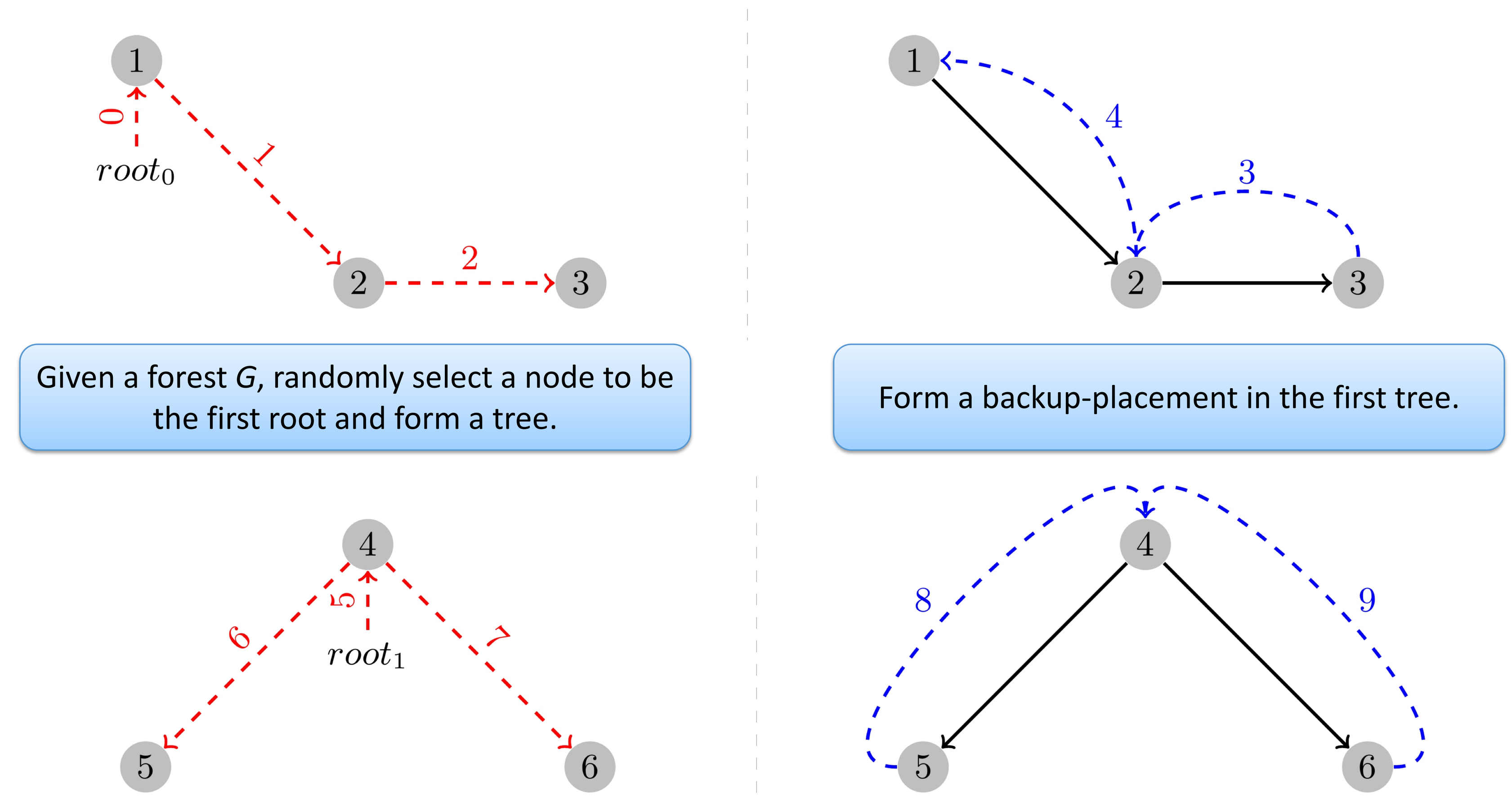
- We devise a procedure for computing ***O(1)*-backup placement in forest**. We assume that each vertex do not knows its parent, nor to which tree in the forest it belongs. The procedure receives a forest *F = (V,E)* as input and proceeds as follows.
- An exemplification of the Distributed Tree Discovery are in **red** and the Forest Backup Placement are in **blue**.

Algorithm 3 The Distributed Tree Discovery Algorithm

procedure D-TREE-DISCOVERY(NODE v, GRAPH G, BEACON B)
  ASSERT: $B = \{ID_{Parent}, ID_{Source}, ID_{root}, d\_root\}$
  if $v.d\_root > d\_root + 1$ **then**
    ASSERT: The neighbor of v, from which the beacon has been received, holds a shorter path to root
    $v.parent \leftarrow ID_{Source}$
    $v.d\_root \leftarrow d\_root + 1$

Algorithm 4 The Forest Backup Placement Algorithm

procedure FOREST-BP(GRAPH $G = (V, E)$)
  $T = \{\}$
  while $G \backslash T \neq \emptyset$ **do**
    $root \leftarrow random\_select(V)$
    $\forall v \in V \Rightarrow$ **transmit** beacon B
    ASSERT: A tree is formed under root because of the Distributed Tree Discovery Algorithm action.
    CNI-Tree-BP(T_{root})
    $T = T \cup T_{root}$



Given a forest *G*, randomly select a node to be the first root and form a tree.

Form a backup-placement in the first tree.

Randomly select a new node (not in *G* trees) to be the second root and form new tree.

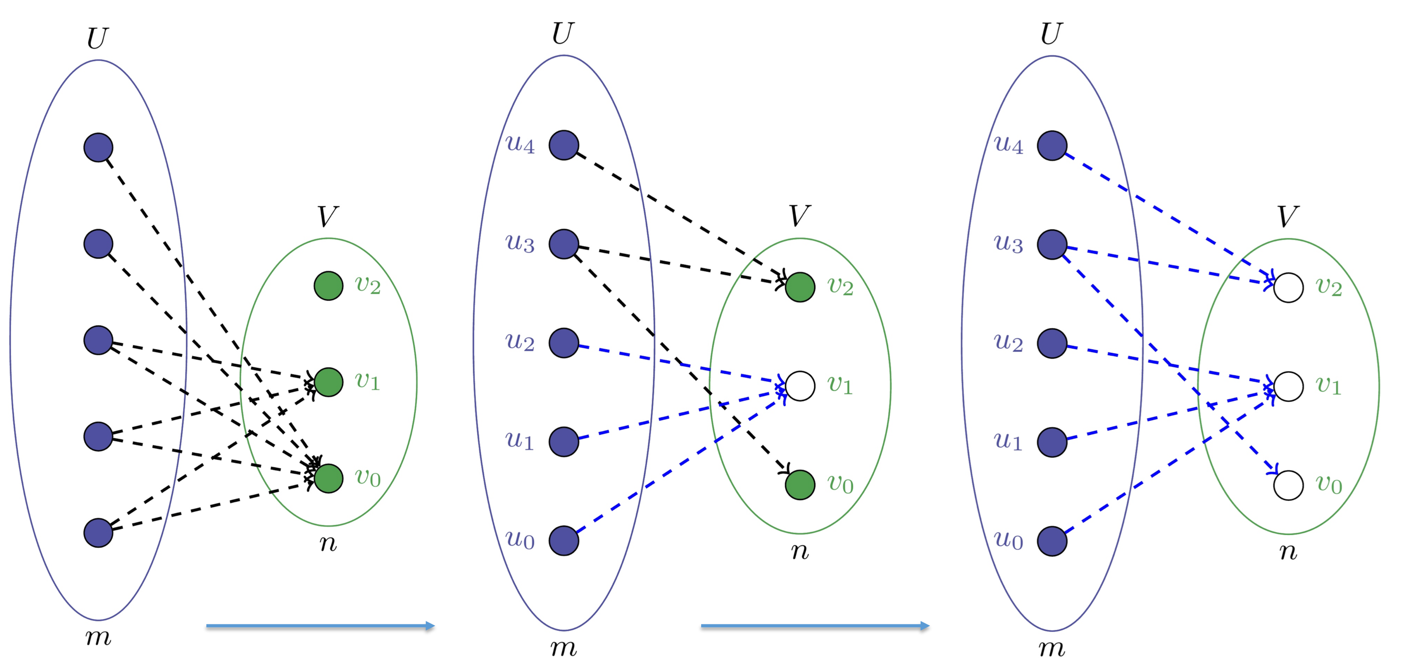
Form a different backup-placement in the second tree.

Backup placement in planar graphs

- We devise a back-up placement algorithm for bipartite graphs *G = (U,V,E)*, in which **the maximum degree of vertices in *U* is bounded by a parameter *a*, and the maximum degree of vertices in *V* is unbounded.**
- The goal of our algorithm is obtaining a maximum load of *O(at)*. To this end, each vertex of *V* may select an arbitrary neighbor in *U*. Since each vertex in *U* has at most *a* neighbors, the maximum load on vertices of *U* is going to be at most *a* as well.
- We proved that using the Procedure Partition alg' **we reach *O(log n)* time complexity to bipartite and planar graphs** for the backup placement problem.

Algorithm 5 The Bipartite Graph Distributed Backup Placement Algorithm

for each $v \in V$, $v.allow\_backups = Active$.
procedure BIPARTITE-BP(V, a, t)
  if $deg(v) \in V < 2at$ & $deg(v) \neq 0$ **then**
    $v.allow\_backups \leftarrow Passive$
    $V = V \setminus v, v.neighbors$



Example of the bipartite graph backup placement

t=1 and *a=2*, i.e. *2ta=4*, and we allow *k=2t=2* backups per vertex in *V*.

Acknowledgments: This work was supported by the Lynn and William Frankel Center for Computer Science, the Open University of Israel's Research Fund, and ISF grant 724/15.