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-Negam

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 is the naïve one, which computes an \( O(k) \)-backup placement of \( T \).
  - Algorithm 2 is based on constant neighborhood, with \( O(1) \) time complexity.

Algorithm 1 Naïve Distributed Tree Backup Placement Algorithm in \( O(1) \)

1: procedure NAIVE-TREE-BP(NODE \( v \in T \), TREE \( T \))
2: if \( v \) is not a leaf then
3: \( v.BP \leftarrow \text{Arbitrary}(c.\text{children}) \).
4: else
5: \( v.BP \leftarrow v.\text{Parent} \).

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

1: procedure CNI-TREE-BP(NODE \( u \), GRAPH \( G \), SUBGRAPH \( T \))
2: if \( v \) is not a leaf then
3: \( v.BP \leftarrow \text{Arbitrary}(c.\text{children}) \).
4: else if \( c \) the sibling of \( v (ID(u) < ID(v)) \) and \( z \) sibling of \( v (ID(u) < ID(z) < ID(v)) \) then
5: \( v.BP \leftarrow w \).
6: else
7: \( v.BP \leftarrow v.\text{Parent} \).

- 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: \( c=11 \log(R\text{max}/R\text{min}) \).

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

1: procedure D-TREE-DISCOVERY(NODE \( v \), GRAPH \( G \), BEACON \( B \))
2: ASSERT: \( B = (ID_{\text{root}}, ID_{\text{daughter}}, ID_{\text{root}}, \text{root}) \)
3: if \( v.\text{root} > d.\text{root} + 1 \) then
4: ASSERT: The neighbor of \( v \), from which the beacon has been received, holds a shorter path to root.
5: \( v.\text{parent} \leftarrow ID_{\text{max}} \).
6: \( v.d.\text{root} \leftarrow d.\text{root} + 1 \).

Algorithm 4 The Forest Backup Placement Algorithm

1: procedure FOREST-BP(GRAPH \( G = (V, E) \))
2: \( T = \{} \).
3: while \( G(T) \neq \emptyset \) do
4: \( \text{root} \leftarrow \text{random.select}(V) \).
5: \( v \in V \neq \text{transmit beacon B} \).
6: ASSERT: A tree is formed under root because of the Distributed Tree Discovery Algorithm action.
7: \( \text{CNI-TREE-BP}(T, \text{root}) \).
8: \( T = T \cup T_{\text{root}} \).

- 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.

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

1: for each \( v \in V \), \text{a}l\text{low.b}ackups \( \rightarrow \text{Active} \).
2: procedure BIPARTITE-BP(V, \( u, t \))
3: if deg(v) \( \in V < 2at \) & deg(v) \( \neq 0 \) then
4: \text{a}l\text{low.b}ackups \( \rightarrow \text{Passive} \).
5: \( V = V \setminus \{v, v.\text{neighbors} \}

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.