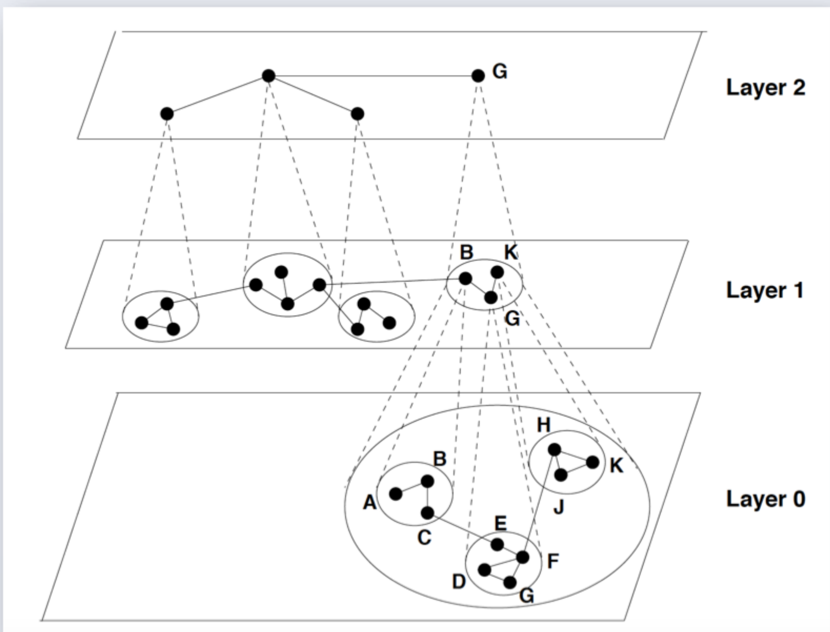
Distributed Fault-Tolerant Backup Placement in Overloaded Wireless Sensor Networks

Gal Oren[1,2], Leonid Barenboim[3], Harel Levin[2,3]

1. Department of Computer Science, Ben-Gurion University of the Negev, POB 653, Be'er Sheva, Israel
2. Department of Physics, Nuclear Research Center-Negev, P.O.B. 9001, Be'er Sheva, Israel
3. Department of Mathematics and Computer Science, The Open University of Israel, P.O.B. 808, Ra'anana, Israel

Introduction

- WSNs frequently have a distinguished amount of data loss, causing data integrity issues.
- Sensor nodes are inherently a cheap piece of hardware – due to the common need to use many of them over a large area – and **usually contain a small amount of RAM and flash memory, which are insufficient in the case of a high degree of data sampling.**
- An overloaded sensor can harm data integrity, or even completely reject incoming messages.
- The problem becomes worse when data are to be received from many nodes, as **missing data become a more common phenomenon as deployed WSNs grow in scale**.
- In cases of overflow, our Distributed Adaptive Clustering algorithm (D-ACR) – based on the HCC algorithm – reconfigures the network by adaptively and hierarchically re-clustering parts of it, based on the rate of incoming data packages in order to minimize the energy consumption, and prevent premature death of nodes.
- However, the **re-clustering cannot prevent data loss caused by the nature of the sensors.**



An example of a three layer WSN hierarchy (HCC)

The Backup Placement Problem in WSNs

The backup placement problem is defined as follows:

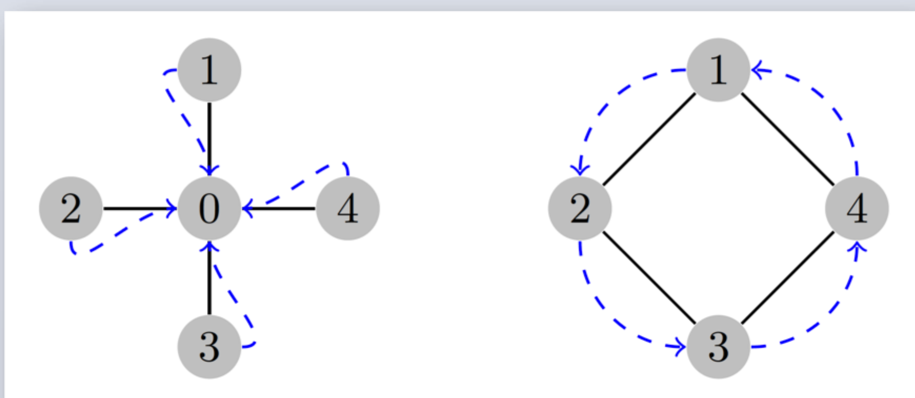
(1) How to **place the data only once in a safe and stable node** in order to assure with a high degree of certainty the data integrity and minimization of the WSN loan

(2) How to do so **without creating additional data overflow** onto other areas of the network

In order to demonstrate this problem, we can examine two test cases:

The **Cycle** graph (maximal backup burden for each node would be of only 1 unit) vs. the **Star** graph (the maximal backup burden would be of |V| − 1) :

Optimal backup placement in the star graph



Non-optimal backup placement in the cycle graph

While this is unavoidable in a star graph, it becomes possible in wireless network topologies.

In terms of graph theory, the problem in its simplest form is defined for a network graph G=(V,E) as follows: Each vertex in V must select a neighbor, such that the maximum number of vertices that made the same selection is minimized.

Algorithm 1 The 1-hop Back-Placement Algorithm

procedure 1-HOP-BP(NODE v, GRAPH G)
    Find a node u in the list of sibling nodes connected with v in G such that $ID(u) = ID(v) + k$, where k is the smallest positive integer for which a node u exists in the list
    if found **then**
        Select u to be v backup node
    else
        Select $\pi(v)$ to be v backup node

Using this algorithm, the maximum extra-load per node is 12=O(1). This is because each node in V is selected by at most 6 of its children, and by at most 6 additional sibling nodes.

Backup Placement in Overloaded WSNs

The solution to the backup problem with overloaded areas is by transferring the packages outside of the area dynamically to a non-overloaded area.

In order to do so, we need to address three main difficulties:

(1) How is the distributed algorithm supposed to detect which nodes under which area are not overloaded?

(2) How is the distributed algorithm supposed to choose the backup nodes and transfer the packages to it in an optimal fashion?

(3) How, during this selection process, will the parent nodes in the tree not be overloaded?

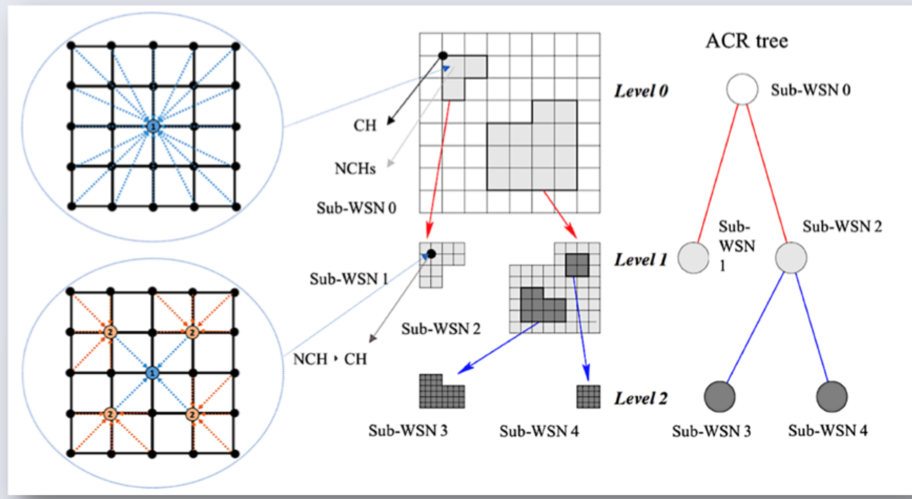
In order to answer these problems:

- We first **use our distributed Adaptive Clustering Refinement algorithm (D-ACR) in order to re-cluster the network** in a way which will reduce the burden as much as possible on specific nodes of the WSN.
- We **take advantage of the hierarchical fashion of the D-ACR refinement algorithm tree in order to find a placement** for data from overloaded nodes using the DBP-ACR algorithm.

These two algorithms are complementary.

Algorithm 2 The Distributed Adaptive Clustering Refinement Algorithm

procedure D-ACR(NODE v, THRESHOLD t)
    if v is a leaf and $energy\_use(v) \geq 3t$ **then**
        Refine(v)
        Add new CHs as children of v
    if all children of v are leafs and $average\_energy(children(v)) < \frac{t}{5}$ **then**
        Coarsen(v)
        Remove the children of v from the tree and mark v as a leaf



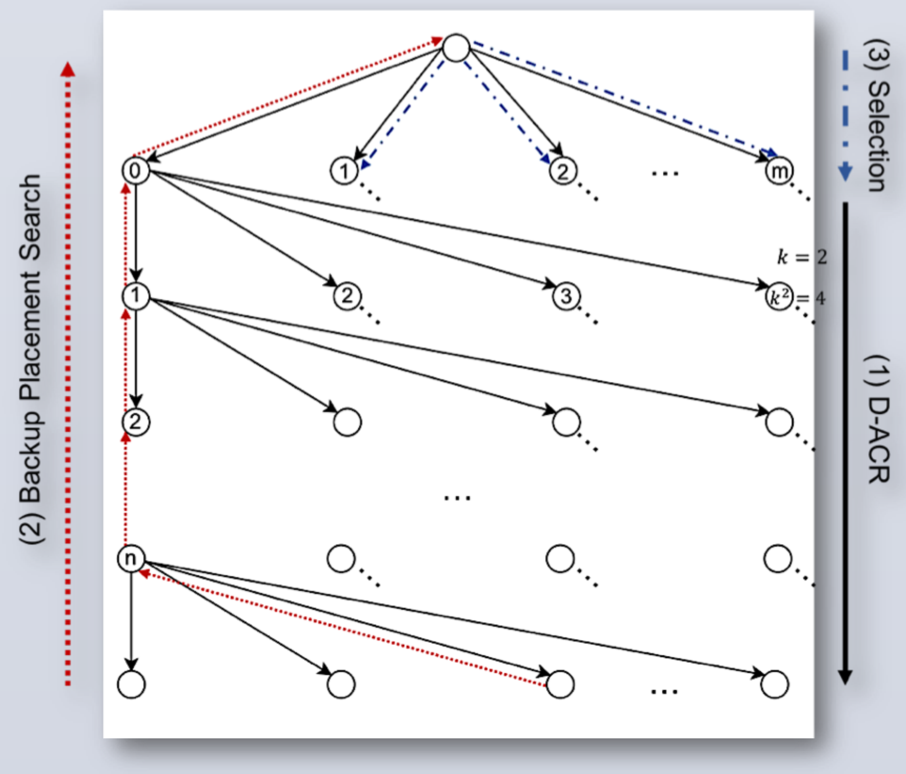
The DBP-ACR Algorithm

DBP-ACR is executed by all nodes in parallel once a package is received:

- Initially, each node checks its memory vacant capacity. If there is no vacancy in the memory system and the package has arrived from a local sensing area, the package will be marked as a package which searches for a backup placement in other parts of the WSN.
- DBP-ACR will transfer the package directly all the way through the D-ACR tree newly-formed CHs (the depth of the D-ACR tree) until reaching the root node from which the D-ACR refinement has started.
- The sibling nodes of this root node most likely have not been overloaded, and the package is placed in one of them.

Algorithm 3 The WSN Distributed Backup-Placement Algorithm

procedure DBP-ACR(NODE v, PACKAGE P)
    Initially, $v.RR\_index \leftarrow 0$
    if $v.memory$ reach full capacity $\wedge$ $P.bp = False$ **then**
        $P.bp \leftarrow True$
        Send $P \rightarrow v.father$
    else if $P.bp = True$ **then**
        if $v.ACR\_depth \neq NULL$ **then**
            Send $P \rightarrow v.father$
        else
            $P.bp \leftarrow False$
            if $v.RR\_index = P.send\_ad$ **then**
                $v.RR\_index \leftarrow (v.RR\_index + 1) \bmod len(v.children)$
            Place $P \rightarrow v.children[v.RR\_index]$
            $v.RR\_index \leftarrow (v.RR\_index + 1) \bmod len(v.children)$



We also proved that (where L is the maximum load, and t is the threshold value of a plausible energy use):

- The running time of the algorithm is bounded by $O\left(\log\frac{L}{t}\right)$.
- The extra-memory consumption of the algorithm is bounded by $O\left(\log\log\frac{L}{t}\right)$.
- The energy consumption of the algorithm package transmission is bounded by $O\left(\left(\log\frac{L}{t}\right)E_{elec} + \epsilon_{fs}\cdot\left(\left(\log\frac{L}{t}\right)\cdot d_{ref}\right)^2\right)$.

Acknowledgments

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