A generalized clustering algorithm for dynamic wireless sensor networks

Raluca Marin-Perianu, Johann Hurink, Pieter Hartel

University of Twente, The Netherlands

Abstract. We propose a general clustering algorithm for dynamic sensor networks, that makes localized decisions (1-hop neighbourhood) and produces disjoint clusters. The purpose is to extract and emphasise the essential clustering mechanisms common for a set of state-of-the-art algorithms, which allows for a better understanding of these algorithms and facilitates the definition and demonstration of common properties.

1 Introduction

Among many challenges faced by ad-hoc and sensor networks designers, *scalability* is a critical issue. The flat topology of these types of networks contains a large number of nodes that have to compete for the limited wireless bandwidth, handle sizeable routing tables and manage substantial traffic caused by network dynamics. One promising approach to solve the scalability problem is to abstract the network topology by building hierarchies of nodes. This process is commonly referred to as *clustering*.

We formally define the *clustering problem* following the definition given by Chen et al. [5]. We model the network as an undirected graph $G = (V, E)$, where V is the set of *vertices* or *nodes* and E is the set of edges or links that directly connect two nodes. The clustering process divides V into a collection of (not necessarily disjoint) subsets $\{V_1, V_2, ..., V_k\}$, where $V = \bigcup_{i=1}^{k} V_i$, such that each subset V_i induces a connected subgraph of G. Each such subset is a cluster. Typically, a particular vertex in each cluster, termed the *root* or *clusterhead*, is elected to represent the cluster.

A special type of clustering algorithms are the distributed *weight-based* algorithms, which assume that each node in the network is assigned a *weight*, representing a measure related to how suitable the node is for the clusterhead role. The weight is a general concept that can cover multiple application-specific parameters, such as the degree of dynamics, the connectivity or the resource availability of the node [12]. In a weight-based algorithm, the election of clusterheads involves: (1) the dissemination of every node's weight to a group of nodes in the network, which represents the *decision range*, and (2) the comparison of these weights and the selection of the best node as clusterhead in a greedy manner. The decision range can vary from as little as only 1-hop neighbours [10], to as large as the whole network [4]. The smaller the decision range, the faster the reaction to topological changes. Therefore, a decision range of 1-hop is suitable for dynamic networks, while a range of more than 1-hop is mostly practical for static networks.

We propose a clustering algorithm that represents a generalization of weight-based clustering algorithms designed for dynamic sensor networks that make localized decisions (1-hop neighbourhood) and produce disjoint clusters. The generalized algorithm uses a set of general variables and conditions. Specialized algorithms can be built by giving specific instances of these variables and conditions. As examples, we present four concrete algorithms, namely C4SD [10], Tandem [9], DMAC [3] and G-DMAC [2]. Assuming as valid a set of constraints, we prove that the generalization satisfies a set of properties. Any algorithm that follows the generalized clustering algorithm

can be proven correct (i.e. it satisfies this set of properties), by only proving that it satisfies the constraints. In particular, C4SD, Tandem, DMAC and G-DMAC are shown to be correct by proving the set of constraints.

The rest of this paper is organized as follows. We give an overview of related work in the field of clustering algorithms in Section 2. In Section 3, we describe the generalized clustering algorithm, giving details about the input, output, properties and specializations. The set of constraints and the proofs of properties are presented in Section 4. Section 5 gives the proofs of constraints for each specialization considered. Section 6 presents a summary and future work.

2 Related work

Clustering algorithms for wireless sensor networks can be classified in two broad categories: (1) probabilistic algorithms that run in synchronized rounds, used mainly for static networks [7], and (2) weight-based algorithms, which can also be used in dynamic networks [3]. In the following, we briefly describe the state of the art in weight-based clustering algorithms, pointing out their characteristics and explaining the main construction steps.

Weight-based algorithms may construct either *overlapping clusters*, where a node may belong to more than one cluster, or *disjoint clusters*, where a node may belong to only one cluster. Algorithms that construct overlapping clusters typically select a group of nodes, termed *gateway* or *border* nodes, which connect at least two adjacent clusters. For example, in the LCA clustering algorithm [6], each node selects as its own clusterhead the neighbouring clusterhead with the lowest ID to which it is bidirectionally connected. A node that can hear two or more clusterheads is a gateway. The k-CONID algorithm (k-hop connectivity ID) [12] constructs a k-dominating set of clusterhead nodes, by assigning each node a weight computed from the connectivity degree and the ID of the node. All the nodes whose weights are the largest among all their k-hop neighbours become clusterheads. The other nodes choose a k-hop neighbouring clusterhead with the largest weight as clusterhead. Nodes that belong to more than one cluster are border nodes. In the algorithm proposed by Wu and Li [15], every node exchanges its neighbour set with all its neighbours. If a node has two unconnected neighbours, it declares itself as gateway. Stojmenovic et al. [13] improve the algorithm proposed by Wu and Li by having each node assigned a weight, computed based on the node degree and the x and y absolute coordinates. The size of the connected dominating set is reduced by taking into consideration the node degree and location.

Disjoint clusters are generally constructed when every node has to share information (such as the node id or sensed data) with its clusterhead. The clusterhead is thus responsible to make use of this information on behalf of the node. To decide on the clusterhead nodes, some protocols base their election on complete information over a number of hops or even from the whole network. For example, the Max-Min D-Cluster algorithm [1] uses the d-hop information for clusterhead election. Each node initiates two rounds of flooding over d hops for building the cluster membership. When the election algorithm finishes, nodes are at most d hops away from the clusterhead. The WCA algorithm proposed by Chatterjee et al. [4] takes into account the node degree, transmission power, battery power and the speed of the nodes, for achieving the optimal operation of the MAC protocol. Each node calculates a combined weight from these parameters, which is then disseminated in the whole network. The node with the global minimum weight is chosen as clusterhead. This node and its neighbours are excluded from subsequent clustering decisions. The process is repeated until all the nodes are clustered. With the DCA clustering algorithm [3, 8], the decision is based only on the IDs of the 1-hop neighbours, which determines a fast reaction to topological changes. However, DCA can be used only for quasi-static networks. The algorithm operates in two phases:

the cluster formation phase and the maintenance phase. During the cluster formation phase, the network is assumed static. The clusterhead nodes are selected based on the lowest node ID, while the rest of the nodes are assigned to the existing clusterheads, forming disjoint clusters. During the maintenance phase, the clustering structure is reconfigured as a result of node mobility or failure.

The focus of this paper is on weight-based clustering algorithms that make decisions based on 1-hop neighbourhood information, produce disjoint clusters and are designed to handle network mobility during all the phases of the algorithm operation. The algorithms presented above do not fall in this category, because either they construct overlapping clusters, make decisions over several hops or are designed for static networks. In the following, we give examples of algorithms that comply with our characterization. These algorithms will be later described as specializations of the generalized algorithm from Section 3.

DMAC [3] is a clustering algorithm designed for mobile networks. Similar to DCA, nodes decide their role depending on the one-hop neighbourhood information. The difference is that DMAC does not perform in separate phases. Each node reacts locally to any variation in the surrounding topology, changing its role accordingly. The variation of the topology is represented by addition and deletion of links to neighbouring nodes. The node with the highest weight among its unassigned neighbours declares itself as clusterhead. The rest of the nodes choose as clusterhead the neighbour which is a root and has the highest weight.

A generalization of DMAC is G-DMAC [2], where a newly initialized node joins the clusterhead with the highest weight in its one-hop neighbourhood (similar to DMAC). While the topology changes, however, the node remains member of this clusterhead v as long as there is no other neighbouring clusterhead u with weight $w(u) > w(h) + h$, given the parameter $h \geq 0$. Another parameter k is introduced that defines the maximum number of clusterhead neighbours that a clusterhead is allowed to have.

The C4SD clustering algorithm [10] is specifically designed for mobile networks, where the weight is represented by the node capability and the degree of dynamics. Every node in the network chooses as parent the neighbour with the highest capability grade. If such a node does not exist, this node is a clusterhead. The output of the algorithm is represented by disjoint clusters, whose clusterheads form an independent set.

Tandem [9] is an algorithm for spontaneous clustering of mobile wireless sensor nodes facing similar context (such as moving together). Tandem assumes that each node runs a shared-context recognition algorithm, which provides a number on a scale, representing the confidence value that two nodes are together. Each node periodically computes the confidence of sharing the same context with its neighbours. The selection of clusterheads is weight-based: the node with the highest weight among its neighbours with which it shares a common context declares itself as clusterhead. A regular node subscribes to the clusterhead with which it shares a common context and has the highest weight.

3 Generalized clustering algorithm

We assume that each node is assigned a *weight*, which is an application-specific number that can be calculated based on different metrics, such as the degree of dynamics, the resource availability, the battery level etc. The node hardware identifier may be used to break ties. In this way, we abstract from the physical characteristics of nodes by only using the *weight* measure in the algorithm description.

In what follows, we present the input and the output of the generalized clustering algorithm and then we proceed with the algorithm description.

3.1 Input

The clustering structure is constructed based on the decision of each node to select a certain *parent*. The decision of each node v depends on the *neighbourhood information*, which includes the set neighbours (the nodes connected through links with v), the weights of the neighbours, the semantic relationship between v and its neighbours, and the current state of the neighbours (their clusterheads), which becomes known to v during the algorithm operation.

The input of the algorithm for each node v is the following:

- $\boldsymbol{\Gamma(v)}$, the open neighbourhood of v, $\Gamma(v) = \{u \in V \mid (u, v) \in E\}$;
- $\boldsymbol{\Gamma^+(v)}$, the closed neighbourhood of v, $\Gamma^+(v) = \Gamma(v) \cup \{v\}$;
- $\boldsymbol{w(u), \forall u \in \Gamma^+(v)}$, the weight of v and the weights of all the neighbours of v.
- $\boldsymbol{s(v, u), \forall u \in \Gamma(v)}$, the semantic relationship between node v and all its neighbours; $s(v, u)$ equals 1 if v and u have similar semantic properties and 0 otherwise.

We remark that the clustering algorithm runs on a *dynamic* network, where the position and the semantic relationships among pairs of nodes change in time, and therefore, the input of the algorithm varies accordingly.

3.2 Output

The output of the generalized clustering algorithm is a set of disjoint clusters, where for each cluster there is a *root* or *clusterhead* node, selected to represent the cluster. To achieve this, each node selects a *parent* from its set of neighbours. The parent of the clusterhead node is the node itself. The weights of the nodes are used in the parent selection: the higher the weight of a neighbour, the more chances to be selected as parent. However, the parent does not always have a higher weight than the node: the weight is used in the decision process, but also other factors contribute to the parent selection (e.g. whether the neighbour is already a clusterhead). The parent of node v is used for communication between v and its clusterhead. A node v may be *unassigned*, if it cannot cluster with any of its neighbours. In this case, the node does not have any parent or root.

For each node v in the network, the output of the algorithm is the following:

- $\boldsymbol{p(v)}$, the parent of v; $p(v) \in V \cup \{\perp\}$.
  We make the following observations:
  - If $p(v) \in V$, then v is called *assigned*. Otherwise, if $p(v) = \perp$, then v is called *unassigned*.
  - If $p(v) = v$, then v is called *root* or *clusterhead*.
- $\boldsymbol{r(v)}$, the root or clusterhead of v; $r(v) \in V \cup \{\perp\}$.
  We make the observation that $p(v) = \perp \implies r(v) = \perp$.

This output describes a *feasible* clustering structure if the directed graph $G_p = (V_p, E_p)$, defined by $V_p = \{v \in V \mid p(v) \neq \perp\}$ and $E_p = \{(v, u) \mid u = p(v) \wedge u \neq v\}$, is a forest of routed trees where the root of each node v is a node equal to the root of its parent (i.e. $\forall v \in V_p$, we have $r(v) \in V_p$ and $r(v) = r(p(v))$). In this way, each tree in G_p forms a cluster with the root of the tree as clusterhead. The nodes in G that are not part of G_p are unassigned.

3.3 Properties

Due to the network dynamics, the output of the clustering algorithm actively has to adapt to the input changes. This adaptation is described in detail in Section 3.4. For static networks (i.e. networks with stable topology and semantic relationships), we require the following properties that a clustering algorithm has to fulfil:

Property 1 *The algorithm produces disjoint clusters.*

Property 2 *Each cluster is organized as a tree, following the parent-children relationship.*

Property 3 *The structure stabilizes to a feasible clustering structure after a finite number of rounds.*

Note that Properties 1 and 2 also hold for dynamic networks. For the definition of a *round*, see Section 3.4.

3.4 Description

The generalized clustering algorithm follows the $\mathcal{LOCAL}$ message passing model, where global structures are constructed based on local information and using local message exchange [11]. Following this model, each node in the network performs some computations and communicates only to its direct neighbours by exchanging messages based on rounds.

The algorithm is based on the following assumptions:

Assumption 1 *Each weight is unique (the node hardware ID can be used to break ties).*

Assumption 2 *The wireless communication is reliable (this can be achieved by using a reliable transport protocol [14]).*

Assumption 3 *The communication links are symmetrical, i.e.* $\forall v \in V,\ \forall u \in \Gamma(v),\ v \in \Gamma(u)$ *(asymmetric links can be hidden by the MAC protocol).*

Assumption 4 *The semantic relationship is symmetrical, i.e.* $\forall v, u \in V,\ s(v,u) = s(u,v)$ *(if two neighbours have different views on the semantic relationship between them, thay can reach an agreement, e.g.* $s(v,u)$ *is given by the node with the highest weight).*

Assumption 5 *Each node is aware of its neighbours, the weights of the neighbours and the semantic relationship with the neighbours, representing the input of the algorithm (see Section 3.1).*

To decide on the clustering structure, each node selects one of its neighbours as parent. A node is a clusterhead if it is its own parent. The identity of the clusterhead is transmitted from parents to children. Therefore, the parent selection is enough to uniquely determine the cluster membership: each node learns from its parent the root of the cluster.

The parent may be selected either periodically or on demand, as a result of a change in the network topology, a reception of update information from neighbours or a change in the semantic relationship with the neighbours. For the sake of simplicity, we consider that the parent is selected periodically. We define a *round* as the time between two consecutive parent selections. One round is long enough in order for all the messages sent by a node during a parent selection to be received by its neighbours (by Assumption 2). We make the observation that nodes are not required to be synchronized: rounds are a tool for the proofs only.

The decision of parent selection is based on the local neighbourhood information: the input of the algorithm (the weight of the node, the set of neighbours and their weights and the semantic relationship with the neighbours) and the current state of the neighbours (represented by the root node of each neighbour).

For each vertex, two subsets of neighbours are given:

- $\boldsymbol{N_1(v)} \subseteq \Gamma^+(v)$ is the subset of neighbours with which node v may be in a common cluster (the decision may be dependent on the semantic relationships among neighbours or other parameters);
- $\boldsymbol{N_2(v)} \subseteq N_1(v)$ is the subset of $N_1(v)$, representing the nodes that are eligible to become parents of v; if the algorithm assigns a parent to v, it either chooses it from the set $N_2(v)$ or it assigns the node itself as parent (in this case, v becomes clusterhead).

Furthermore, three different conditions are given, to be used in the decision process of v:

- $\boldsymbol{P_1(v)}$ is the condition on which node v chooses a different parent (i.e. if the algorithm reaches condition $P_1(v)$ (line 8 of Algorithm 1) and $P_1(v) = \mathit{true}$, then node v changes the parent to the best candidate from the set $N_2(v)$).
- $\boldsymbol{P_2(v)}$ is the condition on which node v becomes root (i.e. if the algorithm reaches condition $P_2(v)$ (line 13 of Algorithm 1) and $P_2(v) = \mathit{true}$, then node v becomes root by assigning $p(v) = v$).
- $\boldsymbol{P_3(v)}$ is the condition on which node v informs the neighbours that one of them has to resign from the clusterhead role (i.e. if $P_3(v) = \mathit{true}$, then v sends a *Resign* message to the neighbours).

In the following, we pin down two particular neighbours of v that have specific roles in the algorithm execution:

- $\boldsymbol{y(v)}$ is the best candidate for the parent role: $y(v)$ is the neighbour of v with the highest weight in $N_2(v)$, i.e. $w(y(v)) = max\{w(u) \mid u \in N_2(v)\}$; based on Assumption 1, $y(v)$ uniquely identified; $y(v)$ is chosen as parent if the algorithm reaches condition $P_1(v)$ and $P_1(v) = \mathit{true}$;
- $\boldsymbol{z(v)}$ is the neighbour of v with the highest weight that has to resign from the clusterhead role, based on condition $P_3(v)$; when node v sends a *Resign* message to the neighbours with parameter $w(z(v))$, all the neighbours with smaller or equal weights than $z(v)$ have to resign from the clusterhead role and search for a new parent.

Algorithm 1 formally describes the generalized algorithm. Each node, when it powers up, enter an *Initialization* phase, where the local variables are initialized and the node becomes unassigned (i.e. $p(v) = r(v) = \perp$). Then, the selection of a parent is done on a periodic basis, as described earlier, by calling the *SelectParent* function. The structure of the *SelectParent* function is the following:

1. Update the local information from the input given by the lower layers of the communication stack, such as the MAC (line 2). From this information, each node v selects the subset of neighbours $N_1(v)$ which are eligible to be part of the same cluster as v. Then v builds the set $N_2(v)$ from the neighbours in $N_1(v)$ which can become parents.
2. Decide the parent, and consequently the root node, based on $N_1(v), N_2(v)$, $P_1(v)$ and $P_2(v)$ (lines 3-21). The decision process can be described as follows:
   (a) In case the set $N_1(v)$ is empty, node v becomes unassigned (lines 3-5).
   (b) Otherwise, if $N_2(v)$ is not empty and $P_1(v) = \mathit{true}$, the node with the highest weight from $N_2(v)$ (i.e. $y(v)$) becomes the parent of v (lines 7-11).
   (c) If $N_2(v)$ is empty, v is not root and $P_2(v) = \mathit{true}$, node v becomes clusterhead (lines 12-15).
   (d) If $N_2(v)$ is empty, v is not root and $P_2(v) = \mathit{false}$, node v becomes unassigned (lines 16-19).
3. If the root has changed or there is a new node in the neighbourhood, inform the neighbours about the current root, by sending a *SetRoot* message (lines 22-24).
4. Based on condition P_3, inform the neighbours that one of them has to resign from the clusterhead role, by sending the message *Resign* (lines 25-27).

The algorithm uses two types of messages, *SetRoot* and *Resign*. When v receives a *SetRoot* message from its parent, it learns the root of its cluster and resends the message, so that the children of v also get informed about their root. Upon receiving a *Resign* message with parameter w, if v is a root node with a lower or equal weight to w, then it has to give up its role and search for a new parent. The neighbour from $N_2(v)$ with the highest weight, $y(v)$, becomes the new parent.

In the following, we describe four specializations of Algorithm 1, by giving concrete definitions to the sets $N_1(v)$ and $N_2(v)$ and to conditions $P_1(v), P_2(v)$ and $P_3(v)$.

3.5 Specializations

The weight-based algorithms C4SD [10], Tandem [9], DMAC [3] and G-DMAC [2] are considered specializations of Algorithm 1. The definitions of the sets $N_1(v)$ and $N_2(v)$ and conditions $P_1(v), P_2(v)$ and $P_3(v)$ for each of these algorithms are the following:

- **C4SD**:
  - $N_1(v) = \Gamma^+(v)$
  - $N_2(v) = \{u \in N_1(v) \mid w(u) > w(v)\}$
  - $P_1(v),\ P_2(v) : true$
  - $P_3(v) : false$

- **DMAC**:
  - $N_1(v) = \Gamma^+(v)$
  - $N_2(v) = \{u \in N_1(v) \mid w(u) > w(v) \wedge r(u) = u\}$
  - $P_1(v),\ P_2(v) : true$
  - $P_3(v) : false$

  By restricting the set $N_2(v)$ to include only root nodes, DMAC generates one-hop clusters (i.e. each assigned node is either a clusterhead or its parent is a clusterhead).

- **Tandem**:
  - $N_1(v) = \{u \in \Gamma(v) \mid s(v,u) = 1\}$
  - $N_2(v) = \{u \in N_1(v) \mid r(u) = u\}$
  - $P_1(v) : ((r(v) = \perp) \vee (r(v) \neq v \wedge r(v) \notin N_2(v)) \vee (r(v) = v \wedge w(v) < w(y(v)))$
  - $P_2(v) : (\{u \in N_1(v) \mid r(u) \neq \perp\} = \emptyset)$
  - $P_3(v) : false$

  Tandem is an algorithm that considers semantic relationships among pairs of nodes as the main clustering criteria. Therefore, the set $N_1(v)$ is restricted to comprise only the nodes which are semantically similar, indicated by $s(v,u)$.
  Similar to DMAC, by limiting the set $N_2(v)$ to include only root nodes, Tandem generates one-hop clusters.
  Changing the parent (condition $P_1(v)$) is triggered only if the node is unassigned ($r(v) = \perp$), the root is no longer in set $N_2(v)$ ($r(v) \neq v \wedge r(v) \notin N_2(v)$) or node v is clusterhead and it has not the highest weight in $N_2(v)$ ($r(v) = v \wedge w(v) < w(y(v))$).
  The condition on which a node can become clusterhead ($P_2(v)$) depends on the set of nodes that are already in $N_1(v)$ and are assigned. If this set is not empty, Tandem prevents node v from becoming clusterhead, in order to minimize the effect of the erroneously perceived semantic similarity between neighbouring nodes. Otherwise, node v elects itself as clusterhead.

Algorithm 1: Generalized clustering algorithm - node v

Initialization:

1. $r(v) \leftarrow \perp$; $p(v) \leftarrow \perp$; $r(u) \leftarrow \perp,\ \forall u \in \Gamma(v)$

SelectParent: // Build the clustering structure by selecting the parent

1. $r_0 \leftarrow r(v),\ \Gamma_0(v) \leftarrow \Gamma(v)$
2. Update $\Gamma(v),\ \Gamma^+(v),\ N_1(v),\ N_2(v),\ y(v)$.
3. **if** $N_1(v) = \emptyset$ **then**
4. $\quad p(v) \leftarrow \perp$
5. $\quad r(v) \leftarrow \perp$
6. **else**
7. $\quad$ **if** $N_2(v) \neq \emptyset$ **then**
8. $\quad\quad$ **if** $P_1(v)$ **then**
9. $\quad\quad\quad p(v) \leftarrow y(v)$
10. $\quad\quad\quad r(v) \leftarrow r(p(v))$
11. $\quad\quad$ **end if**
12. $\quad$ **else if** $(p(v) \neq v)$ **then**
13. $\quad\quad$ **if** $P_2(v)$ **then**
14. $\quad\quad\quad p(v) \leftarrow v$
15. $\quad\quad\quad r(v) \leftarrow v$
16. $\quad\quad$ **else**
17. $\quad\quad\quad p(v) \leftarrow \perp$
18. $\quad\quad\quad r(v) \leftarrow \perp$
19. $\quad\quad$ **end if**
20. $\quad$ **end if**
21. **end if**
22. **if** $(r(v) \neq r_0) \vee (\Gamma(v) \setminus \Gamma_0(v) \neq \emptyset)$ **then**
23. $\quad$ Send *SetRoot(v,r(v))* to neighbours.
24. **end if**
25. **if** $P_3(v)$ **then**
26. $\quad$ Send *Resign(w(z(v)))* to neighbours.
27. **end if**

SetRoot(u,r): // Update the information from neighbour u

1. $r(u) \leftarrow r$
2. **if** $(p(v) = u) \wedge (r(v) \neq r)$ **then**
3. $\quad r(v) \leftarrow r$
4. $\quad$ Send *SetRoot(v,r(v))* to neighbours.
5. **end if**

Resign(w): // Resign from the clusterhead role

1. **if** $(p(v) = v) \wedge (w(v) \leq w)$ **then**
2. $\quad$ Update $N_2(v), y(v)$.
3. $\quad$ **if** $N_2(v) \neq \emptyset$ **then**
4. $\quad\quad p(v) \leftarrow y(v)$
5. $\quad\quad$ **if** $(p(v) = v)$ **then**
6. $\quad\quad\quad r(v) \leftarrow p(v)$
7. $\quad\quad$ **else**
8. $\quad\quad\quad r(v) \leftarrow r(p(v))$
9. $\quad\quad$ **end if**
10. $\quad\quad$ Send *SetRoot(v,r(v))* to neighbours.
11. $\quad$ **end if**
12. **end if**

– **G-DMAC**: G-DMAC differs from the other algorithms by defining the following set of constraints for the clustering structure:
  - ***h***: an assigned node v can change its current parent $p(v)$ only if the new parent p_1 has a significant higher weight, i.e. $w(p_1) - w(p(v)) > h$, where h represents the minimum difference between the weights of p_1 and $p(v)$.
  - ***k***: if v is a root node, the number of root nodes that are allowed to be present in the neighbourhood of v is at most k; formally, $|\{v \in \Gamma^+(v) \mid p(v) = v\}| \leq k$; paramter k is used by condition $P_3(v)$ to determine whether any of the neighbours of v has to resign.

  To control the number of clusterheads that are allowed to be neighbours, G-DMAC uses a *Resign* message with parameter $w(z(v))$, which is the weight of the first clusterhead that violates the k-neighbourhood condition. Neighbour $z(v)$ is defined such that $z(v) \in \{u \in N_1(v) \mid r(u) = u\}$ and $|\{u \in N_1(v) \mid r(u) = u \wedge w(u) > w(z(v))\}| = k$.
  Using the additional input h and k, G-DMAC is defined by the following instantiations of $N_1(v), N_2(v), P_1(v), P_2(v), P_3(v)$:
  - $N_1(v) = \Gamma^+(v)$
  - $N_2(v) = \{u \in N_1(v) \mid w(u) > w(v) \wedge r(u) = u\}$
  - $P_1(v) : ((r(v) = \perp) \vee (w(r(v)) + h < w(y(v))) \vee ((r(v) \neq v) \wedge (r(v) \notin N_2(v))) \vee ((r(v) = v) \wedge (|\{u \in N_1(v) \mid r(u) = u\}| > k) \wedge (w(v) \leq min\{w(z) \mid z \in \{u \in N_1(v) \mid r(u) = u\}\})))$
  - $P_2(v) : true$
  - $P_3(v) : ((r(v) = v) \wedge (|\{u \in N_1(v) \mid r(u) = u\}| > k) \wedge (w(v) > w(z(v))))$

  Similar to DMAC and Tandem, by limiting the set $N_2(v)$ to include only root nodes, G-DMAC constructs only one-hop clusters.
  Changing the parent (condition $P_1(v)$) occurs if the node is unassigned ($r(v) = \perp$), the root is no longer in v's neighbourhood ($(r(v) \neq v) \wedge (r(v) \notin N_2(v))$) the weight of the new clusterhead exceeds the current weight of the root with a certain threshold h ($w(r(v)) + h < w(y(v))$) or the number of roots exceeds k, v is root and has the lowest weight among the roots in its neighbourhood ($(r(v) = v) \wedge (|\{u \in N_1(v) \mid r(u) = u\}| > k) \wedge (w(v) \leq min\{w(z) \mid z \in \{u \in N_1(v) \mid r(u) = u\}\})$).
  If the number of allowed root nodes exceeds the threshold k (condition $P_3(v)$), a *Resign* message will be sent to the neighbours to correct this situation.

4 Correctness of the cluster formation

In the following, we prove formally that the considered algorithms fulfil the properties described in Section 3.2. Properties 1 and 2 are general properties, while Property 3 is applicable only if the network stabilizes. We prove that if certain *constraints* regarding the algorithm operation are satisfied, Algorithm 1 fulfils Properties 1-3. Thus, any algorithm that follows Algorithm 1 and satisfies the constraints is correct, in the sense that it fulfils Properties 1-3. Section 5 shows that each of the four algorithms, C4SD, DMAC, G-DMAC and Tandem does satisfy the constraints, and thus each algorithm fulfils Properties 1-3.

4.1 General properties

In this section, we prove Properties 1 and 2, which hold also for dynamic networks. First, we prove that Property 1 is always achieved by Algorithm 1. Next, we present a constraint (Constraint 1), which ensures that Property 2 is fulfilled.

Disjoint clusters

Lemma 1 *If a clustering algorithm follows the generalized algorithm, the network contains only disjoint clusters.*

Proof From the description of Algorithm 1, we deduce that every assigned node has only one parent, corresponding to the variable $p(v)$. Since a node belongs to the cluster of its parent, no node can be part of different clusters. □

No cycles

Constraint 1 *For every node v, at least one of the following propositions is true:*

1. *Node v may select as parent only itself or another node with a higher weight. Formally, $N_2(v) \subseteq \{x \in N_1(v) \mid w(x) > w(v)\}$.*
2. *Node v may select as parent only itself or a root node. If v is a root node that chooses another parent, it is allowed to select as parent only a root with higher weight. Formally, $(r(v) = v \wedge P_1(v) = true \wedge N_2(v) \neq \emptyset) \Longrightarrow (r(y(v)) = y(v) \wedge w(v) < w(y(v)))$*

We denote with $p_k(v)$ the parent of order k, i.e. $p_1(v) = p(v)$, $p_k(v) = p(p_{k-1}(v))$ for $k > 1$.

Lemma 2 *If a clustering algorithm follows the generalized algorithm and satisfies Constraint 1, the clustering structure does not contain cycles. Formally, $\forall v \in V$, if $p(v) \neq v$ then $p_k(v) \neq v, \forall k > 1$.*

Proof If the first proposition of Constraint 1 is satisfied, then each node picks exactly one parent, either itself or a neighbour of higher weight. Therefore, the network does not contain cycles.

If the second proposition of Constraint 1 is satisfied, then we have only one-hop clusters. If a node that is not root selects as parent another node that is a root, then v is added to the cluster of that root with a 1-hop distance. The root node cannot choose node v as parent, since node v is not a root. If two roots r_1 and r_2 become connected and both might choose the other as parent (i.e. $r_1 \in N_2(r_2)$ and $r_2 \in N_2(r_1)$), then at least the one with the highest weight remains root. Therefore, the network does not contain cycles. Note that if the two nodes get connected (i.e. the root with smaller weight selects the other root as parent), transitory two-hop clusters may occur. However, this situation is corrected in the next round, since now the 2-hop nodes do not have a root as parent and thus they must select a new parent. □

4.2 Stabilization property

In this section, we present 2 more constraints and show that these constraints imply Property 3.

We assume that starting from an arbitrary global state, the topological structure and the semantic similarities among nodes remain constant (i.e. $N_1(v)$ remains constant, $\forall v \in V$). We denote this moment with round 0.

We refer to round k, $k \geq 1$ as the series of consecutive rounds following 0. We denote with S^k the set of stabilized nodes corresponding to round k, i.e. $S^k = \{v \in V \mid \forall j \geq k,\ p^j(v) = p^k(v),\ r^j(v) = r^k(v)\}$, where $p^i(v)$ and $r^i(v)$ are the parent and root of v at round i.

We say that the clustering structure is *stable* if for any subsequent round, every node keeps the same parent and root. Formally, the structure is stable or reaches stability at round k if $S^k = V$.

After round 0, we assume that the generalized algorithm satisfies the following constraints (see Section 5, where the constraints are proven for each specialization of Algorithm 1) :

Constraint 2 *There exists $d \leq |V|$ such that after round 0 every d rounds either there exists at least one additional node that remains with the same parent and root in any subsequent round, or the network gets stable. Formally, $\exists d \leq |V|$, such that we have either $(\exists x \in S^{k+d}$ with $x \notin S^k)$ or $(S^{k+d} = V)$, $\forall k \geq 0$.*

Constraint 3 *Let $v \in V$. After the first round, all the nodes that are eligible to become parents of v are assigned. Formally, $\forall x \in N_2(v),\ p^1(x) \neq \perp$.*

Correctness proofs

Lemma 3 *Given Constraint 2, after at most $O(d|V|)$ rounds, the network stabilizes ($\exists k,\ 0 \leq k \leq d|V|$, such that $S^k = V$).*

Proof Constraint 2 implies that $|S^{k+d}| \geq |S^k| + 1$ if $|S^k| < V$ and $k \geq 0$. Therefore, we have that $|S^{d|V|}| = |V|$, i.e. after at most $O(d|V|)$ rounds the network stabilizes. □

Lemma 4 *Given Constraint 3, a stabilized network does not have any unassigned parent node. Formally, let $x \in V$ be a node in the network. If $\exists v \in V$ such that $p(v) = x$, then $p(x) \neq \perp$.*

Proof From Constraint 3 we have that after one round, for any node $v \in V$, all the nodes in the set $N_2(v)$ from where v can choose a parent are assigned ($\forall x \in N_2(v),\ p(x) \neq \perp$). □

4.3 Correctness of the generalized algorithm

Theorem 1 *Any algorithm which follows Algorithm 1 and satisfies Constraints 1-3 stabilizes after a finite number of rounds to a feasible state.*

Proof If Constraint 2 holds, then from Lemma 3 we have that after at most $O(d|V|)$, the network stabilizes.

From the definition of feasibility from Section 3.2, we deduce that the clustering structure is in a feasible state if: (1) the structure does not contain cycles, (2) the root of node v is also the root of $p(v)$, (3) there are no unassigned parents. We prove each of these properties in turn:

1. If Constraint 1 holds, then from Lemma 2 we have that the structure does not contain cycles.
2. The *SetRoot* message is propagated to all the nodes in the cluster whenever there is a topological or root change. Therefore, each node v learns the root of its cluster from $p(v)$, such that the root of v becomes equal to the root of its parent (see line 3 from the *SetRoot(u,r)* event, Algorithm 1).
3. If Constraints 3 holds, then from Lemma 4 we have that a stabilized network does not have any unassigned parent node.

Therefore, we have that after a finite number of rounds the network stabilises to a feasible state. □

5 Proofs of constraints for each algorithm

From Section 4.3, we have that any algorithm which follows Algorithm 1 and satisfies Constraints 1-3 is correct. Therefore, we prove these constraints for the four specializations of Algorithm 1 presented in Section 3.5, from where it follows that these algorithms are correct.

5.1 C4SD

Constraint 1 Since $N_2(v) = \{u \in N_1(v) \mid w(u) > w(v)\}$, the first proposition is always true. □

Constraint 2 We may take $d = 1$. We first prove the constraint for the first round, and then we generalize for round $k > 1$.

First, note that $\forall v \in V$, $N_2(v)$, $P_1(v)$ and $P_2(v)$ do not change after round 1. Therefore, after the first round, each node has a stable parent. Let v^1 be the node with the maximum weight, $w(v^1) = max\ \{w(v) \mid v \in V\}$. Because $N_2(v^1) = \emptyset$ and $P_1(v^1) = true$, after the first round, v^1 becomes root. Any subsequent round does not change $N_2(v^1)$ and $P_1(v^1)$, and thus v^1 remains root.

At round k, $k > 1$, let v^k be the node with the highest weight that is not already stabilized (does not know its root $r(v)$), i.e. $w(v^k) = max\{w(v) \mid v \in V \setminus S^{k-1}\}$. Since the parent of v^k has a higher weight than v^k, we have that $p(v^k) \in S^{k-1}$, and therefore, it knows its root. Thus, in round k, also node v^k gets to know its root and stabilizes.

□

Constraint 3 After the first round, all the nodes are assigned ($P_1(v) = true$, $P_2(v) = true$, $\forall v \in V$). Therefore, after the first round, there is no unassigned parent. □

5.2 DMAC

Constraint 1 Since $N_2(v) = \{u \in N_1(v) \mid w(u) > w(v) \wedge r(u) = u\}$, the first proposition is always true. □

Constraint 2 We may take $d = 1$. We first prove the constraint for the first round, and then we generalize for round $k > 1$.

Let v^1 be the node with the maximum weight, $w(v^1) = max\ \{w(v) \mid v \in V\}$. Because $N_2(v^1) = \emptyset$ and $P_1(v^1) = true$, after the first round, v^1 becomes root. Any subsequent round does not change $N_2(v^1)$ and $P_1(v^1)$, and thus v^1 remains root.

At round k, $k > 1$, let v^k be the node with the highest weight that is not already stabilized, $w(v^k) = max\{w(v) \mid v \in V \setminus S^{k-1}\}$. The decision of node v^k depends only on the decisions of the neighbours with higher weights, which are already stabilized. Therefore, the set $N_2(v^k)$ does not change in any subsequent round, as well as $P_1(v^k)$ and $P_2(v^k)$, so that at round k node v^k stabilizes, by selecting a parent that is also a root. □

Constraint 3 The set $N_2(v)$ contains only assigned nodes, $\forall v \in V(N_2(v) = \{u \in N_1(v) \mid w(u) > w(v) \wedge r(u) = u\})$. □

5.3 G-DMAC

Constraint 1 Since $N_2(v) = \{u \in N_1(v) \mid w(u) > w(v) \wedge r(u) = u\}$, the first proposition is always true. □

Constraint 2 We may take $d = 1$. We first prove the constraint for the first round, and then we generalize for round $k > 1$.

Let v^1 be the node with the maximum weight, $w(v^1) = max\ \{w(v) \mid v \in V\}$. Node (v^1) is either unassigned, or is root. If v^1 is unassigned, since $N_2(v^1) = \emptyset$ and $P_2(v) = true$, v^1 becomes root. Otherwise, v^1 remains root in any subsequent round.

At round $k,\ k > 1$, let v^k be the node with the highest weight that is not already stabilized, i.e. $w(v^k) = max\{w(v) \mid v \in V \setminus S^{k-1}\}$. The decision of node v^k depends only on the decisions of the neighbours with higher weights, which are already stabilized. Therefore, the set $N_2(v^k)$ does not change in any subsequent round, as well as $P_1(v^k)$ and $P_2(v^k)$.

A *Resign* message received by v^k, which can trigger the resignation of v^k from being a root, can be sent only by a node with a higher weight than v^k. Therefore, if the k-neighbourhood condition is violated for any of the stabilized neighbouring nodes of v that are also roots, v may receive a *Resign* message and act accordingly (i.e. if v is a root, it gives up its role and selects a new parent) only in a round preceding round k. Therefore, node v selects a parent that is also a root at round k and stabilizes. □

Constraint 3 The set $N_2(v)$ contains only assigned nodes, $\forall v \in V(N_2(v) = \{u \in N_1(v) \mid w(u) > w(v) \wedge r(u) = u\})$. □

5.4 Tandem

Constraint 1 We prove the second proposition. A node can choose as parent only itself or a root node, because the set $N_2(v)$ contains only root nodes ($N_v(v) = \{u \in N_1(v) \mid r(u) = u\}$). Since $y(v) \in N_2(v)$, we have that $r(y(v)) = y(v)$. If node v is a root, it can give up its role by selecting as parent only a root node of higher weight. From the definition of $P_1(v)$ ($P_1(v) : ((r(v) = \perp) \vee (r(v) \neq v \wedge r(v) \notin N_2(v)) \vee (r(v) = v \wedge w(v) < w(y(v))))$), we have that $(r(v) \neq v) \vee (r(v) = v \wedge w(v) < w(y(v)))$. □

Constraint 2 All the nodes that have $N_1(v) = \emptyset$ become and remain unassigned after the first round. In the following, we discuss the nodes that have $N_1(v) \neq \emptyset$.

We may take $d = 2$. We first prove the constraint for the first 2 rounds, and then we generalize for round $k > 2$.

Suppose that after round 0, none of the nodes is a root. If condition P_2 is satisfied by at least one node v ($\exists v \in V$ such that $\{u \in N_1(v) \mid r(u) \neq \perp\} = \emptyset$), then after the first round v becomes a root. Otherwise, all the nodes in V become unassigned after the first round ($p(v) = \perp,\ \forall v \in V$). In this case, after the second round, the condition P_2 becomes satisfied by least one node, who becomes root. Let v^1 be the root node with the highest weight after the second round, $r(v^1) = v^1$ and $w(v^1) = max\{w(v) \mid r(v) = v\}$. Node v^1 remains root in any subsequent round because none of its neighbours can become root ($N_2(v) \neq \emptyset,\ \forall v \in N_1(v^1)$, by Assumptions 3 and 4), so it stabilizes at round 2.

Starting from the stabilized set at round k, we prove that after at most 2 rounds, the stabilized set contains at least one additional node.

At round $k+1,\ k \geq 2$, if exists a node x that has as parent a node from the already stabilized set ($x \notin S^k,\ p(x) \in S^k$), we take $v^{k+1} = x$. Due to condition $P_1(v^{k+1})$, node v^{k+1} does not change the root in any subsequent round and thus v^{k+1} stabilizes at round $k+1$.

If none of the nodes $x \notin S^k$ has as parent a node from the already stabilized set, we take v^{k+1} the root node with the maximum weight which is not part of the stabilized set ($r(v^{k+1}) = v^{k+1}$ and

$w(v^{k+1}) = max\{w(v) \mid r(v) = v,\ v \in V \setminus S^k\})$. Node v^{k+1} does not change the role of root in any subsequent round because none of its neighbours can become root ($N_2(v) \neq \emptyset,\ \forall v \in N_1(v^{k+1})$), so it stabilizes at round $k+1$.

The last case is when all the nodes which are not yet stabilized are unassigned, i.e. $\forall x \notin S^k,\ r(x) = \perp$. If all these nodes have assigned neighbours, then from condition $P_2(x)$ it follows that all these nodes remain unassigned in any subsequent round, so the network stabilizes. Otherwise, there exists at least one node that has all neighbours unassigned, i.e. $\exists x \notin S^k$ such that $P_2(x)$ is true ($(\{u \in N_1(x) \mid r(u) \neq \perp\} = \emptyset)$. Therefore, node x becomes root at round $k+2$. At round $k+2$, we take v^{k+2} the root node with the maximum weight which is not part of the stabilized set. Node v^{k+2} does not change the role of root in any subsequent round because none of its neighbours can become root ($N_2(v) \neq \emptyset,\ \forall v \in N_1(v^{k+2})$), so it stabilizes at round $k+2$. □

Constraint 3 The set $N_2(v)$ contains only assigned nodes, $\forall v \in V\ (N_2(v) = \{u \in N_1(v) \mid r(u) = u\})$. □

6 Conclusions

This paper presents a generalized clustering algorithm that makes localized decisions based on 1-hop neighbourhood information and produces disjoint clusters. The algorithm represents a generalization of weight-based clustering algorithms designed for mobile ad-hoc and sensor networks, by using a set of general variables and conditions. Concrete algorithms, leading to different clustering structures, can be defined by giving specific meaning to these variables and conditions. As examples, we present four specialized algorithms, namely C4SD [10], Tandem [9], DMAC [3] and G-DMAC [2]. Assuming as valid a set of given constraints, this generalization allows us to define and prove common properties for these algorithms. Any new algorithm that follows the generalized clustering algorithm can be proven correct, with respect to the properties given in Section 3.3, by proving that it satisfies this set of constraints.

References

1. Alan D. Amis, Ravi Prakash, Dung Huynh, and Thai Vuong. Max-min d-cluster formation in wireless ad hoc networks. In *INFOCOM*, pages 32–41. IEEE Computer Society Press, 2000.
2. Stefano Basagni. Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. In *Proceedings of Vehicular Technology Conference (VTC)*, pages 889–893. IEEE Computer Society, 1999.
3. Stefano Basagni. Distributed clustering for ad hoc networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pages 310–315, Washington, DC, USA, 1999. IEEE Computer Society.
4. Mainak Chatterjee, Sajal K. Das, and Damla Turgut. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5(2):193–204, 2002.
5. Yuanzhu Peter Chen, Arthur L. Liestman, and Jiangchuan Liu. *Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing, Volume 2*, volume 75, chapter Clustering Algorithms for Ad Hoc Wireless Networks, pages 145–164. Nova Science Publishers, 2005.
6. Antony Ephremides, Jeffrey Wieselthier, and Dennis Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, 75(1):56–73, 1987.
7. W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, 2002.

8. C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *Selected Areas in Communications, IEEE Journal on*, 15(7):1265–1275, 1997.
9. R. S. Marin-Perianu, C. Lombriser, P. J. M. Havinga, J. Scholten, and G. Troster. Tandem: A context-aware method for spontaneous clustering of dynamic wireless sensor nodes. In *Proceedings of Internet of Things, the International Conference for Industry and Academia*, pages 342–360. Springer, 2008.
10. R. S. Marin-Perianu, J. Scholten, P. J. M. Havinga, and P. H. Hartel. Cluster-based service discovery for heterogeneous wireless sensor networks. *International Journal of Parallel, Emergent and Distributed Systems*, 23(4):325–346, August 2008.
11. T. Nieberg. *Independent and Dominating Sets in Wireless Communication Graphs*. PhD thesis, University of Twente, April 2006.
12. Fabian Garcia Nocetti, Julio Solano Gonzalez, and Ivan Stojmenovic. Connectivity based k-hop clustering in wireless networks. *Telecommunication Systems*, 22(1–4):205–220, 2003.
13. Ivan Stojmenovic, Mahtab Seddigh, and Jovisa Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, 2002.
14. Chonggang Wang, Kazem Sohraby, Bo Li, Mahmoud Daneshmand, and Yueming Hu. A survey of transport protocols for wireless sensor networks. *IEEE Network*, 20(3):34–40, 2006.
15. Jie Wu and Hailan Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM)*, pages 7–14, New York, NY, USA, 1999. ACM.