A Divide-and-Conquer Scheme for Assigning Roles in Multi-Channel Wireless Mesh Networks

Benoit Darties, Fabrice Theoleyre, Andrzej Duda

A Divide-and-Conquer Scheme for Assigning Roles in Multi-Channel Wireless Mesh Networks

Benoît Darties, Fabrice Theoleyre, and Andrzej Duda

Grenoble Informatics Laboratory
CNRS and Grenoble-INP
681 rue de la passerelle, BP72
38402 Saint Martin d'Heres, France
Email: {firstname.lastname}@imag.fr

Abstract

A multi-channel MAC seems to be an interesting approach for improving network throughput by multiplexing transmissions over orthogonal channels. In particular, Molecular MAC has recently proposed to modify the standard IEEE 802.11 DCF access method to use dynamic channel switching for efficient packet forwarding over multiple hops. However, this MAC layer requires role and channel assignment to nodes: some of them use a static channel, while others dynamically switch to neighbor channels on-demand. To assign roles and channels, we extend the notion of the Weakly Connected Dominating Set, the structure already used in clustering. More precisely, we adapt the WCDS structure and introduce new constraints to define what we call a *reversible WCDS* (*r-WCDS*), which is particularly suitable for wireless mesh networks operating under Molecular MAC. We propose a divide-and-conquer scheme that partitions the network into clusters with one leader per cluster solving a MILP formulation to assign roles in its cluster. By appropriately defining the roles at the border of clusters, we maintain global connectivity in the r-WCDS. Finally, our simulations show that the performance of the propose scheme is close to a centralized algorithm.

1 Introduction

We consider wireless mesh networks composed of a large number of wireless routers providing connectivity to mobile nodes. They begin to emerge in some regions to provide cheap network connectivity to a community of end users. Usually they grow in a *spontaneous* way when users or operators add more routers to increase capacity and coverage.

We assume that mesh routers benefit from sufficient resources, may only move, quit, or join occasionally, so that the topology of a typical mesh network stays fairly stable. The organization of mesh networks needs to be *autonomic*, because unlike

the current Internet, they cannot rely on highly skilled personnel for configuring, connecting, and running mesh routers. Thus, proposed protocols must be distributed and self-stabilizing.

One way of organizing the structure of a wireless mesh network is to construct a Weakly Connected Dominating Set (WCDS) [1], which is a well-known clustering scheme in wireless multihop networks. By appropriately electing clusterheads (called *dominators* in the WCDS terminology), each node in the network is a member of one cluster and the network structure results in a limited number of hops between clusterheads. Such clustering is useful for limiting the overhead of flooding, introducing a routing hierarchy, distributing keys [2], or other network-wide operations.

To improve network capacity, wireless mesh networks can use multiple radio channels: multiplexing transmissions on orthogonal channels allows for parallel transmissions through spectrum spatial reuse and reduces collisions. Mesh routers can take advantage of parallel transmissions over neighbor links by using multiple channels at various time scales. When nodes have multiple interfaces, they can statically allocate channels to achieve high spatial reuse and good performance [3, 4, 5, 6]. Nodes with single interfaces can also benefit from multiple channels by switching channels on a per frame basis [7, 8]. Molecular MAC [9] has recently proposed to modify the standard IEEE 802.11 DCF access method to use dynamic channel switching for efficient packet forwarding over multiple hops. It solves the deafness problem inherent to multichannel schemes by assigning a static channel for one part of nodes (*nuclei* of spatially distinct *atoms*) and letting other nodes (*electrons*) dynamically switch between channels. Electrons always initiate transmissions, while nuclei notify other nodes about pending packets. Molecular MAC outperforms classical strategies like MMAC [7] with respect to throughput, fairness, and end-to-end delay. However, the authors have left the problem of assigning roles (nucleus or electron) and channels for future work.

In this paper, we propose a protocol for organizing a wireless mesh network according to a suitable structure associated with Molecular MAC. We adapt the well-known WCDS structure and introduce new constraints to define what we call a *reversible WCDS* (*r-WCDS*), which is particularly suitable for wireless mesh networks operating under Molecular MAC.

The contribution of this paper is threefold:

- it provides a formal definition of the reversible WCDS,
- we propose a new divide-and-conquer scheme for constructing such a r-WCDS in a distributed way,
- we evaluate the performances of the proposed scheme and compare it with other approaches.

This paper is organized as follows. First, we introduce some notations and define the problem of constructing a WCDS for Molecular MAC. In Section 3 we use a Mixed Integer Linear Programming formulation (MILP) to find a r-WCDS. Then, we present in Section 4 `potatoes`, a divide-and-conquer scheme for constructing a r-WCDS in a scalable way. We report in Section 5 on the simulation based performance evaluation of the proposed scheme and we discuss the related work in Section 6. We finally conclude the paper and give some perspectives.

2 Problem formulation and notation

We model the network as an undirected graph $G = (\mathcal{V}, \mathcal{E})$ in which vertices $\mathcal{V}(\mathcal{G})$ are the set of nodes and edges $\mathcal{E}(\mathcal{G})$ are all pairs of nodes able to directly communicate. We adopt the following classical notation:

- $n = |\mathcal{V}|$ defines the number of nodes in the mesh network,
- $N(u)$ is the set of neighbors of u with cardinality $\Delta(u) = |N(u)|$,
- $\{u, v\}$ denotes the edge between vertices u and v, i.e. $\{u, v\} \in \mathcal{E}$,
- BW denotes channel capacity,
- CH is the set of all available channels and $nbCH = |CH|$ (IEEE 802.11a provides for instance 12 orthogonal channels in the US).

2.1 Reversible Weakly Connected Dominating Set Problem

The Weakly Connected Dominating Set (WCDS) [1] is a widely used structure in wireless multihop networks. Formally a WCDS is defined by the set $D \subseteq \mathcal{V}$ such that:

$$\forall u \in \{\mathcal{V} - D\} \quad , \quad \exists v \in D | v \in N(u) \tag{1}$$

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}') \; connected \quad | \mathcal{E}' = \big\{\{u, v\}, u \in D, v \in \mathcal{V}\big\} \tag{2}$$

A node of D is often called *dominator* while nodes in $\mathcal{V} - D$ are called *dominatees*.

We define the *reversible WCDS* (denoted r-WCDS in the rest of the paper) as follows: we only keep the edges $(dominator, dominatee)$, i.e. edges between dominators are removed. We will further see that Molecular MAC requires such a structure. Formally, we transform the second property of Eq. 2 into:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}') \; connected | \mathcal{E}' = \big\{\{u, v\}, u \in D, v \in \{\mathcal{V} - D\}\big\} \tag{3}$$

2.2 Relation to Molecular MAC

We are interested in the reversible WCDS, because Molecular MAC [9] requires such a structure. Molecular MAC divides a wireless mesh network into spatially distributed *atoms* so that each atom uses a fixed channel different from its neighbors. An atom is composed of a *nucleus* and *electrons*. A nucleus chooses a channel for its atom and sticks to this channel all the time. Nodes at the border of atoms have the role of electrons bonding neighboring atoms: they forward traffic between atoms by dynamically switching their channel to communicate with neighboring nuclei.

We can compare Molecular MAC to an extended WLAN: an atom corresponds to a WLAN cell and a nucleus is a *virtual Access Point* that interconnects other nodes in the cell (electrons). In mesh networks, a node needs to communicate with several cells. Since a node should not miss packets from its virtual AP because it is transmitting packets over another WLAN, Molecular MAC implements a specific MAC mechanism:

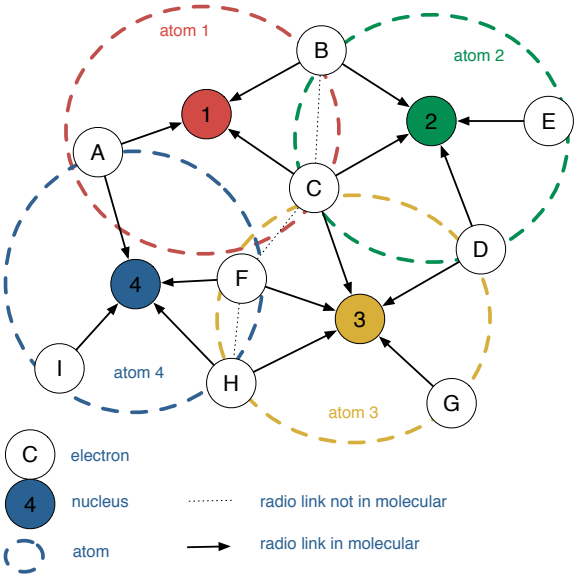

Figure 1: Example molecular topology

a node pulls its data frames from the AP that buffers frames and transmits the list of pending destinations in beacons.

In Molecular Mesh, we need to assign a role to each node (a nucleus or an electron) so that the resulting network has the following properties:

1. a node can communicate with any other node via multi-hop forwarding;

2. only nuclei and electrons can communicate with each other, i.e. there is no direct communication between two electrons or two nuclei. Indeed, two nuclei do not hear each other, because they use different channels. Similarly, two electrons continually switch their channels and may suffer from deafness;

3. the capacity of the network should be maximal. In particular, two neighboring atoms, which can interfere, need to use different channels.

Figure 1 illustrates the molecular organization. The identifiers of nuclei are numbers while they are letters for electrons. We can note that links between two electrons are not used, however we need to keep the number of unused links small to allow for redundant paths in the network for better connectivity and failure tolerance. Clearly, the constraints for assigning roles to nodes lead to a reversible-WCDS: each dominator corresponds to a nucleus and an electron to a dominatee.

Constructing the molecular structure also requires that each nucleus node selects a fixed channel. Simultaneous assignment of roles and channels so to maximize network capacity is a difficult problem. However, a good tradeoff between computation

time and network performance consists first of determining the role of a node to obtain a connected component regardless of channel usage. Once a node has become a nucleus, it can choose a channel according to a greedy approach by scanning all available channels and choosing the one with the minimum load.

3 MILP Formulation

We use a MILP (Mixed Integer Linear Programming) formulation already defined in our previous work [10] to assign roles (i.e. finding a r-WCDS) while maximizing the network capacity. We define all the constraints corresponding to flow conservation, connectivity requirements, throughput maximization as linear constraints. Its solution leads to the optimal assignment of roles (nucleus or electron) and channels in a spontaneous mesh network. We summarize it below.

- We assign a role to each node $u \in \mathcal{V}$ represented by variable $role(u) \in \{0,1\}$ with value 1 if u is a dominator and 0 otherwise.
- Our performance objective is to maximize the global network throughput. We assume any-to-any traffic pattern: each node communicates with each other node thus giving $n(n-1)$ multihop flows. We maximize T_{min}, the minimum throughput allocated to each flow.
- $T(u,v,d)$ corresponds to traffic transmitted by u through link $\{u,v\}$ to destination d for each triplet $(u,v,d)|\{u,v\} \in E, d \in V$. Note that $\forall \{u,v\} \in \mathcal{E},\ T(u,v,u) = 0$ (i.e. a node does not generate traffic for itself);
- $\Delta(u)$ denotes the degree of node u, i.e. the number of neighbors in the network.

3.0.1 Links between nodes

We can only use a link if and only if its endpoints have different roles. Its capacity (the sum of $T(u,v,d)$ over all destinations d) is zero if both endpoints are dominators (Eq. 4) or dominatees (Eq. 5):

$$\forall \{u,v\} \in \mathcal{E}, role(u) + role(v) + \frac{1}{BW} \sum_{d \in V} \big(T(u,v,d) + T(v,u,d)\big) \leq 2 \quad (4)$$

$$\forall \{u,v\} \in \mathcal{E}, \frac{1}{BW} \sum_{d \in V} \big(T(u,v,d) + T(v,u,d)\big) \leq role(u) + role(v) \quad (5)$$

3.0.2 Flow conservation

Eq. 6 and 7 express the flow conservation law: the sum of traffic to d forwarded by u is equal to the sum of traffic to d entering in u and the traffic generated by u to d (which must be at least equal to throughput T_{min}). Eq. 7 represents the fact that a destination node must receive exactly $(n-1).T_{min}$ total traffic units (T_{min} for each $(n-1)$ sources):

$$\forall u, d \in \mathcal{V}, d \neq u,$$

$$\sum_{v \in N(u)} T(u,v,d) = \sum_{v \in N(u)} T(v,u,d) + T_{min} \quad (6)$$

$$\forall u \in \mathcal{V}, \quad \sum_{v \in N(u)} (n-1).T_{min} = T(v,u,u) \quad (7)$$

3.0.3 Capacity of an atom

All links belonging to an atom share its bandwidth BW:

$$\forall u \in \mathcal{V}, \quad \sum_{v \in N(u)} \sum_{d \in V} \big(T(u,v,d) + T(v,u,d)\big) \leq BW \quad (8)$$

The constraints are obvious if u is a dominator. If u is a dominatee, it cannot receive more than BW, even if it is adjacent to several dominators because of time sharing mechanisms for switching between channels.

3.0.4 Improvement

Optional inequalities (Eq. 9) accelerate the MILP resolution by stating that each dominator is adjacent to at least one dominatee and reciprocally:

$$\forall u \in \mathcal{V}, \quad 1 \leq role(u) + \sum_{v \in N(u)} role(v) \leq \Delta(u) \quad (9)$$

Solving the MILP is computationally expensive for large networks: a couple of hours is required to obtain the optimal assignment in a mesh network of 40 nodes.

4 `potatoes`: a divide and conquer scheme

In this paper, we propose a divide-and-conquer scheme: we divide the network into *clusters* with one *leader* per cluster that solves the MILP formulation for its cluster. The small size of clusters leads to good efficiency of obtaining the local MILP solution. However, we need to enforce additional constraints so that the union of multiple local r-WCDSs results in a global connected r-WCDS. To obtain this goal, we have to define clusters in a certain manner and fix the roles of some node. We provide below the details of the mechanisms for constructing the clusters and achieving the global r-WCDS.

4.1 Approach

We propose here a distributed scheme to construct a r-WCDS. We first construct a *rooted cluster tree*, i.e. a tree in which vertices are clusters and a link exists between clusters if and only if they share a node. We assume that the network is connected: if several components exist, the algorithm will be executed independently in each of them. The *rooted cluster tree* supports distributed role assignment: one leader per cluster computes the optimal local assignment in its cluster (i.e. not taking into account other nodes and radio links). However, we need to limit the dependence between two clusters, i.e. a node should receive its role from only one leader. Thus, we add the following constraints with respect to classical cluster-trees:

1. One node belongs to at most two clusters. A node that belongs to exactly two clusters is a cluster member of the cluster higher in the tree and the leader in the other one;

2. the intersection of any two clusters contains at most one node.

Figure 2 presents an example of clustering: there are five clusters forming a tree hierarchy. We can verify that each leader belongs to exactly two clusters except the *root leader* at the root of the cluster-tree: it is the network leader.

The leader of each cluster learns the topology of its cluster, i.e. node ids and links between nodes. Then, it computes the local optimal solution with the MILP formulation: all the constraints are translated into linear inequalities and the global objective consists of maximizing the throughput. After computing the roles for its cluster, the leader has just to notify its cluster members about their roles.

The assignment is optimal in one cluster, but the union of local assignments does not necessarily leads to a global optimal r-WCDS, because leaders find the optimal role assignment inside clusters and not among clusters. We need to enforce that network nodes belonging to different clusters end up with the same role in the global r-WCDS. To achieve this goal, we use the hierarchy of the cluster-tree and proceed in the following way:

1. the leader assigns a role to all its cluster members,

2. a node that belongs to one cluster just uses the assigned role,

3. a node that belongs to more than one cluster is the leader of a cluster down in the hierarchy (e.g. node C in Figure 2) and it will receive its role from the leader upper in the hierarchy (e.g. node A).

Finally, at least one path exists in the rooted cluster-tree that uses the hierarchy of clusters. We will now define more formally this algorithm.

4.2 Notation

We will use the following definitions:

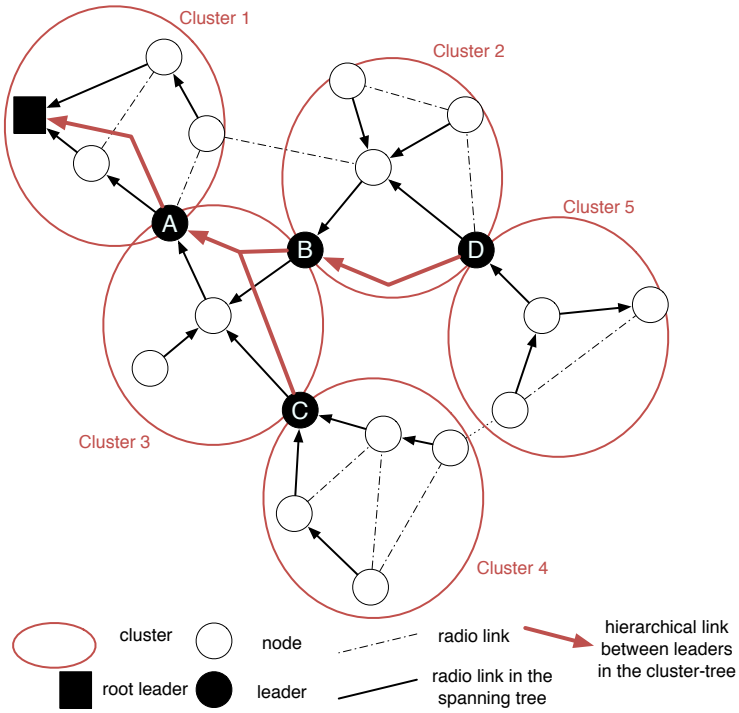

Figure 2: An example cluster-tree.

- a cluster is a connected subgraph $\mathcal{C}luster$ of network graph $\mathcal{G}$ induced by the subset of nodes $S \subseteq \mathcal{V}(\mathcal{G})$ with $\mathcal{V}(\mathcal{C}luster) = S$ and $\mathcal{E}(\mathcal{C}luster) = \big\{\{u,v\} | u,v \in \mathcal{V}(\mathcal{G})\ and\ \{u,v\} \in \mathcal{E}(\mathcal{G})\big\}$;
- $\mathcal{T}$ denotes a spanning tree used to build the cluster-tree;
- $\mathcal{CT}$ represents the cluster-tree structure;
- network leader node denoted by $\mathcal{R}oot\mathcal{L}eader$ is the root of spanning tree $\mathcal{T}$. The unique cluster containing $\mathcal{R}oot\mathcal{L}eader$ is denoted $\mathcal{R}oot\mathcal{C}luster$ and is the root of $\mathcal{CT}$;
- $V(\mathcal{C}luster)$ denotes the set of nodes belonging to $\mathcal{C}luster$;
- $\mathcal{L}eader(\mathcal{C}luster)$ denotes the leader of $\mathcal{C}luster$. Obviously, $\mathcal{R}oot\mathcal{L}eader$ is the leader of $\mathcal{R}oot\mathcal{C}luster$. If $\mathcal{C}luster \neq \mathcal{R}oot\mathcal{C}luster$, $\mathcal{C}luster\mathcal{L}eader$ is the node belonging to $\mathcal{C}luster$ and to the upper cluster in the cluster-tree. Formally, $leader(\mathcal{C}luster) = \mathcal{V}(\mathcal{C}luster) \cap V(parent(\mathcal{C}luster))$ with $parent(\mathcal{C}luster)$ being the upper cluster in $\mathcal{CT}$;

- intra-cluster edges are links between two nodes of the same cluster. Any other link is an inter-cluster edge.

- $role(u)$ denotes the role of node u (either dominator or dominatee).

4.3 Cluster-Tree Construction

We assume that nodes periodically broadcast `hello` messages to discover their neighbors. Usually a network uses this kind of protocols for supporting other functions such as routing—we can just piggyback additional information required by `potatoes` in such messages.

According to the definitions above, all leader nodes except $\mathcal{R}oot\mathcal{L}eader$ belong to exactly two clusters. In the example presented in Figure 2, $\mathcal{R}oot\mathcal{C}luster$ is located on the top-left corner and the rooted cluster tree has five clusters and the depth of two. The example tree-cluster has two inter-cluster edges. As our algorithm finds the optimal role assignment inside clusters, the presence of the intra-cluster edges in the final r-WCDS will depend on the role assigned to each node at the border.

The tree-cluster construction proceeds as follows.

1. The network elects $\mathcal{R}oot\mathcal{L}eader$.

2. Nodes construct a spanning-tree rooted at the network leader (i.e. $\mathcal{R}oot\mathcal{L}eader$). A node propagates the minimum id received by neighbors (or its own id if it is lower) by piggybacking the tuple $< min\_id, distance, seq\_nb >$ in `hello` messages. A node updates its distance to $\mathcal{R}oot\mathcal{L}eader$ only if the `hello` message contains a larger seq_nb than the current seq_nb. Only $\mathcal{R}oot\mathcal{L}eader$ increments the sequence number in each `hello`: a node can safely decide that $\mathcal{R}oot\mathcal{L}eader$ has left the network, if the sequence number is not incremented during T_{dead} time interval. A node also includes in its `hello` messages the id of its parent in the tree. In this way, any node can maintain the list of its children in $\mathcal{T}$.

3. We define clusters and their leaders based on their position in the spanning tree. A new cluster is created when the distance to the leader upper in the spanning tree is exactly $D + 1$ hops:

   (a) $\mathcal{R}oot\mathcal{L}eader$ becomes the first leader;

   (b) each node piggybacks the identity of its leader, $\mathcal{L}eader(u)$ in `hello` messages. It becomes the first leader on the path in $\mathcal{T}$ towards $\mathcal{R}oot\mathcal{L}eader$;

   (c) the tree is divided so that each node is at most D hops away from its leader and the leader has the minimum depth in the tree among all its cluster members. `potatoes` elects the nodes that are exactly $\equiv 0\ [D]$ hops away from the $\mathcal{R}oot\mathcal{L}eader$ ($[D]$ stands for *modulo D*).

4. a cluster is finally defined for each leader $\mathcal{L}$ and contains all the nodes with $\mathcal{L}eader(u) = \mathcal{L}$.

Let us consider the topology in Figure 2 with a cluster radius of 2. The figure presents the spanning tree used for the cluster-tree construction: each node is at most 2 hops in the spanning tree away from its leader. Border leaders maintain hierarchical relationships: C, the leader of cluster 4 is one of the children of leader A in cluster 3.

4.4 Learning Cluster Topology

A leader should know the internal topology of its cluster before solving the MILP optimization. A node could flood the list of its neighbors in the cluster, but this implies high overhead. We can note that only the leader needs to know the topology of the whole cluster. Thus, we can efficiently use tree $\mathcal{T}$ to propagate the topology information and merge it along the tree.

At the beginning, each node discovers new neighbors through `hello` messages. Besides, a node maintains a local *topology table* that contains all links for which one extremity is a child in the spanning tree $\mathcal{T}$. This topology table is restricted to the children in the same cluster and is recursively fed: each node piggybacks its topology table in each `hello` and updates it with the information received from its children (their topology table and their list of neighbors). Obviously, the table of the leader contains the global vision of the cluster topology, while the topology tables of other nodes contain only partial information.

To be fault-tolerant, a node updates its topology table to take into account joining or leaving neighbors. Moreover, each entry of the local table contains a `child-id` field. Each time a child receives a `hello` message, it flushes the entries in the local topology table and replaces them by the new ones. In the same way, a child that has changed its parent is simply removed from each local topology table.

4.5 Role Assignment

If its topology table remains unchanged for a sufficient time, each leader learns the topology of its cluster, computes the local optimal assignment, and sends it to the cluster members.

We force each leader to have a predefined role by adding the following constraints in the MILP optimization:

- $\mathcal{R}oot\mathcal{L}eader$ becomes the first dominator.
- If the cluster radius (D) is even, each leader can safely take the role of a dominator. Each leader $\mathcal{L}$ (except $\mathcal{R}oot\mathcal{L}eader$) belongs to exactly two clusters: it is the leader of $\mathcal{C}luster$ and a member of $Parent(\mathcal{C}luster)$. Since the radius is even, a path alternating dominatees and dominators can link $\mathcal{L}$ to the leader in $Parent(\mathcal{C}luster)$;
- If the cluster radius is odd, the leader $\mathcal{L}$ should have the inverse role[1] of its leader in $Parent(\mathcal{C}luster)$. By alternating dominatees and dominators, we would obtain a valid path;

[1] roles are either dominator or dominatee.

Each leader $\mathcal{L}$ in $\mathcal{C}luster$ receives its role from $Parent(\mathcal{C}luster)$. This role is fixed in the MILP formulation by adding the linear constraint $role(\mathcal{L}) = dominator$ or $role(\mathcal{L}) = dominatee$ according to the role given to $\mathcal{L}$. Thus, MILP resolution can be fully distributed in each cluster: L does not have to wait for the role assigned by the leader of $Parent(\mathcal{C}luster)$ and can directly compute roles in $\mathcal{C}luster$ once it knows the topology.

Let consider the topology in Figure 2. We can observe that the MILP formulation can find a solution in a cluster if we fix the role of all the leaders to dominators since the cluster radius is even. For instance, leader A in cluster 3 will execute its MILP with the role of A, B, and C already fixed to $dominator$. In particular, we can in particular "color" all the nodes with an even depth as dominators and others as dominatees in the spanning tree $\mathcal{T}$. Clearly, a path alternating dominators and dominatees can interconnect any pair of leaders in the same cluster leading thus to an achievable solution. In this way, we guarantee that dominators globally form a valid r-WCDS.

This simple optimization accelerates convergence and results in good performances. Moreover, we keep at least one connected solution in each cluster that consists for each node to alternate roles between nuclei and electron according to the distance to its leader in the cluster.

After solving the MILP optimization $\mathcal{L}eader$ sends the list of its cluster members and their assigned roles along the tree.

4.6 Discussion

The presented scheme locally optimizes role assignment. It presents the following advantages:

1. by splitting the network into clusters, the algorithm is scalable and succeeds to find a suitable solution with a reasonable computational cost;
2. it is distributed, because it relies on the distributed construction of the cluster-tree and role assignment;
3. it is fault-tolerant by taking into account joining and leaving nodes as well as lost `hello` messages.

5 Performance evaluation

We have simulated the proposed protocol in WsNet [11] using the COIN-CBC linear programming library [12]. We randomly place nodes in a simulation area. Nodes use the IEEE 802.11a network interface to communicate with each other with the radio range of 10 units and the interference range of 30 units. By default, the mesh network is composed of 50 nodes with on the average 10 neighbors. We adjust the simulation area to obtain given density.

The results correspond to statistics averaged over 10 different simulations of 240 seconds. The graphs present averaged values with 95% confidence intervals. We compare the performance of the centralized MILP formulation (OPT), `potatoes`, the

Maximum Independent Set protocol (MIS) and a self-stabilizing Spanning Tree (ST) ([10]). We measure the following performance indices:

- minimum throughput T_{min}: the minimum throughput guaranteed for each flow extracted from the MILP formulation. We consider the normalized channel capacity (i.e. $BW = 1$);
- average route stretch factor: the average ratio of the length of the shortest route through the r-WCDS and the length of the shortest route in the original graph.

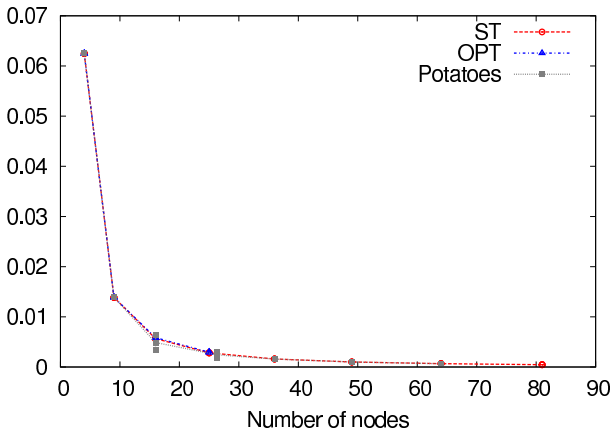

Figure 3: Minimum throughput T_{min} for a varying number of nodes in a grid

5.1 Grid Topology: Capacity

We have first compared the performance of the different protocols in a grid topology: nodes are placed regularly in a squared grid, the length of each cell in the grid being the radio range. Such a grid can represent a regular mesh network deployed for instance by a telecommunication operator. We do not report the results for MIS since it leads to a disconnected network in most cases.

We have measured T_{min}, the minimum throughput allocated to each flow obtained with the MILP formulation for a varying number of nodes (cf. Figure 3). We can observe that all the protocols perform quite similarly. In particular, the spanning tree strategy achieves to find optimal roles and channels: in a grid, the pruning strategy is inefficient since the number of neighbors is limited. Thus, ST consists of marking as dominators all the nodes with an even depth (on average a half of the nodes in a random spanning tree) and as dominatees other nodes.

5.2 Random Topology: Route Stretch Factor

Then we have considered a random topology of a given density (10 neighbors on the average). We measure the *route stretch factor*: the ratio of the route length in a molecular mesh and in the original graph (cf. Fig. 4). A stretch factor of 1 means that only

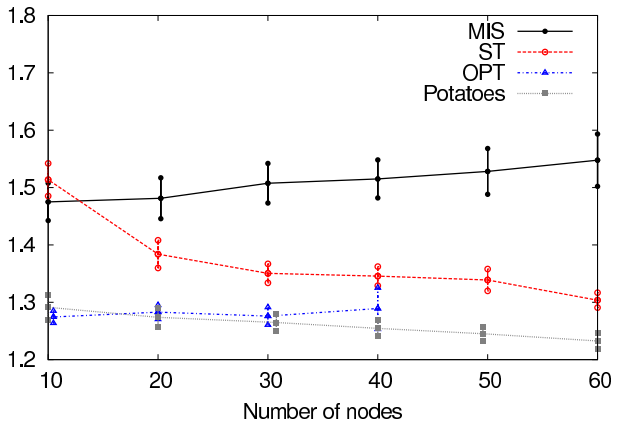

Figure 4: Average route stretch factor for a varying number of nodes in a random network.

the shortest routes are used. For MIS, we discard isolated nodes since the stretch factor would become infinite in this case. Thus, we tend to under-estimate the real stretch factor for MIS. OPT results in an average stretch factor around 1.3 for any number of nodes. However, OPT is not scalable—results become difficult to obtain in a reasonable time for more than 40 nodes (50 nodes require more than 3 hours) and almost impossible to be obtained for more than 60 nodes. This explains why we do not plot results for 50 and 60 nodes under OPT in Figure 4.

The performance of `potatoes` and OPT is very similar. `potatoes` achieves to construct a r-WCDS with a maximum number of radio links: the routes are often the shortest ones. Moreover, `potatoes` is much more scalable than OPT and achieves to compute distributively a r-WCDS even for larger networks. This shows that the divide-and-conquer approach is suitable for our problem.

ST uses longer routes, but the stretch factor tends to decrease when the number of nodes increases. The stretch factor for MIS is large and a flow consumes more bandwidth since it is relayed by more nodes on average.

5.3 Random Topology: Capacity

Finally, we have measured the minimum throughput T_{min} in a random topology (cf. Figure 5). This metric corresponds to the throughput we can obtain with Molecular MAC.

Obviously, the OPT protocol gives us an upper bound since it finds role assignment maximizing the objective. However, scalability issues do not allow to obtain enough significant results when we have more than 40 nodes. Here again `potatoes` results are very close to those of OPT.

We can note that MIS and ST achieve much lower throughput: they do not succeed in maximizing the number of links remaining in the r-WCDS topology, which leads to a lower throughput. For the density we have used, the average performance of both

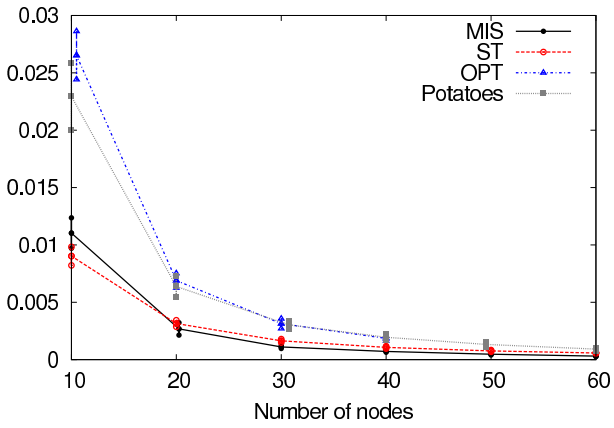

Figure 5: Minimum throughput T_{min} for a varying number of nodes in a random network.

strategies is less than half of the performance obtained with `potatoes`.

6 Related work

Clustering creates groups of nodes, which is particularly useful in wireless multihop networks to introduce a hierarchy (e.g. for routing). Clusters often make use of the concept of domination: nodes elect a clusterhead and all its neighbors become members of the cluster [13]. However, in some cases two hops may separate clusterheads so that their interconnection requires gateways.

The Weakly Connected Dominating Set (WCDS) is a well-know structure often used for network-wide operations such as clustering or distributing keys in MANET [2]. However, finding a WCDS with a given cardinality is NP-hard [1]. Domke et al. [14] extended this result by characterizing graphs having the same minimum cardinality to form both a WCDS and a DS. However, the authors focused on particular graphs (e.g. trees with special properties) and did not solve the WCDS problem in any graph.

Chen et al. [15] extended the centralized algorithm for finding a WCDS of Guha et al. [16] by selecting the best nodes to add in the WCDS for each round, i.e. the component that forms the WCDS grows at each step. Dubashi et al. [17] pruned the edges that belong to a cycle, i.e. they create a sparser network. Although it forms a Connected Dominating Set, it cannot directly be used to create a r-WCDS. Alzoubi et al. [18] constructed a Maximum Independent Set, clusterheads being elected based on their depth in a spanning tree. Thus, this algorithm is close to the *ST* algorithm presented in the Section 5.

In our approach, we build upon the ideas of Chen et al. [19]: they partition the network in zones and each zone executes an algorithm (a divide-and-conquer approach). However, their algorithm is greedy and directly applied to each zone. Moreover, they focused on the original WCDS problem and not on its r-WCDS variant. Moreover, we

take into account other performance criteria than the cardinality of the WCDS, i.e. *network capacity*. Han et al. [20] adopts a similar approach of partitioning the network, however the same remarks as above still hold.

7 Conclusions and future work

We have presented a divide-and-conquer scheme for computing a reversible WCDS in a distributed way. By creating a cluster-tree, we partition the network into clusters with one leader per cluster solving a MILP formulation to assign roles in its cluster. Although this approach does not lead to the global optimum, our simulations show that its performance is very close to a centralized optimal algorithm.

In the future, we plan to explore new strategies to improve the convergence of `potatoes`. In particular, we can explore redundancy to simplify the MILP formulation. Moreover, we aim at exploring the trade-off between optimality and convergence delay: if we pre-assign some roles, we can reduce complexity along with a negligible impact on performance.

Acknowledgments

This work was partly supported by the European Commission project WIP under contract 2740, and the French Ministry of Research project AIRNET under contract ANR-05-RNRT-012-0.

References

[1] Jean E. Dunbar, Jerrold W. Grossman, Johannes H. Hattingh, Stephen T. Hedetniemi, and Alice A. McRae. On weakly connected domination in graphs. *Discrete Mathematics*, 167-168, 1997.

[2] Al-Sakib Khan Pathan and Choong Seon Hong. A key-predistribution-based weakly connected dominating set for secure clustering in dsn. In LNCS, editor, *High Performance Computing and Communications (HPCC)*, volume 4208, pages 270–279, Munich, Germany, September 2006.

[3] Ashish Raniwala, Kartik Gopalan, and Tzi cker Chiueh. Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks. *ACM Mobile Computing and Communications Review*, 8:50–65, 2004.

[4] A. Raniwala and Tzicker Chiueh. Architecture and Algorithms for an IEEE 802.11-based Multi-Channel Wireless Mesh Network. *in Proc. IEEE INFOCOM'05*, 3:2223–2234, March 2005.

[5] Mansoor Alicherry, Randeep Bhatia, and Li (Erran) Li. Joint Channel Assignment and Routing for Throughput Optimization in Multi-Radio Wireless Mesh Networks. In *Proc. ACM MobiCom'05*, pages 58–72, New York, NY, USA, 2005. ACM.

[6] Murali Kodialam and Thyaga Nandagopal. Characterizing the Capacity Region in Multi-Radio Multi-Channel Wireless Mesh Networks. In *Proc. ACM MobiCom'05*, pages 73–87, New York, NY, USA, 2005. ACM.

[7] Jungmin So and Nitin H. Vaidya. Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver. In *MOBIHOC*, Japan, 2004. ACM.

[8] Victor Bahl, Ranveer Chandra, and John Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in ieee 802.11 ad hoc wireless networks. In *Proc. of the ACM MOBICOM*, pages 216–230, Philadelphia, USA, October 2004. ACM.

[9] Mohammad Nassiri, Fabrice Theoleyre, Martin Heusse, and Andrzej Duda. Molecular architecture for autonomic wireless mesh networks. In *MASS (International Conference on Mobile Ad-hoc and Sensor Systems)*, Macau, Macau, October 2009. IEEE.

[10] Fabrice Theoleyre, Benoit Darties, and Andrzej Duda. Assignment of roles and channels for a multichannel mac in wireless mesh networks. In *ICCCN (International Conference on Computer Communications and Networks)*, San Francisco, USA, August 2009. IEEE.

[11] G. Chelius, A. Fraboulet, and E. Fleury. Wsnet: a modular event-driven wireless network simulator. `http://www.worldsens.net/`, 2006.

[12] Coin-or branch and cut. `https://projects.coin-or.org/Cbc`.

[13] Chunhung Richard Lin and Mario Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1265–1275, 1997.

[14] Gayla S. Domke, Johannes H. Hattingh, and Lisa R. Markus. On weakly connected domination in graphs ii. *Discrete Mathematics*, 305(1–3):112–122, December 2005.

[15] Yuanzhu Peter Chen and Arthur L. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *international symposium on Mobile ad hoc networking & computing (MOBIHOC)*, pages 165–172, Lausanne, Switzerland, June 2002. ACM.

[16] Sudipto Guha and Samir Khuller. Approximation algorithmsforconnecteddominatingsets. *Algorithmica*, 20(4):374–387, 1998.

[17] Devdatt Dubhashi, Alessandro Mei, Alessandro Panconesi, Jaikumar Radhakrishnan, and Aravind Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons,. *Journal of Computer and System Sciences*, 71(4):467–479, November 2005.

[18] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Weakly-connected dominating sets and sparse spanners in wireless ad hoc networks. In *International Conference on Distributed Computing Systems (ICDCS)*, Providence, USA, May 2003. IEEE.

[19] Yuanzhu Peter Chen and Arthur L. Liestman. Maintaining weakly-connected dominating sets for clustering ad hoc networks. *Ad Hoc Networks*, 3:629–642, 2005.

[20] Bo Han and Weijia Jia. Clustering wireless ad hoc networks with weakly connected dominating set. *Journal of Parallel and Distributed Computing*, 67(6):727–737, June 2007.