On Demand Beaconless Planar Backbone Construction for Quasi Unit Disk Graphs

Florentin Neumann and Hannes Frey

Institute for Computer Science, University of Koblenz-Landau, Germany
{fneumann, frey}@uni-koblenz.de

Abstract—Beaconless topology control algorithms reduce message overhead of local topology constructions compared to conventional (beacon-based) local approaches by avoiding maintenance of neighborhood tables. Moreover, they construct a node's adjacency in the desired topology on demand and only locally, i.e., do not require network-wide operation. In this work, we present a beaconless topology control algorithm which enables a node to reactively construct a planar backbone graph in its geographic vicinity. This backbone graph is a constant node degree, constant stretch hop-spanner for the input quasi unit disk graph. Our contribution is novel, since all known algorithms with comparable outputs require maintenance of neighborhood tables and are designed for network-wide operation. In addition, it is of significance since there are several applications of it, e.g., in the context of geographic unicast and multicast routing with guaranteed delivery.

Keywords—Topology control, beaconless algorithms, planar backbones, spanners, quasi unit disk graphs, sensor networks

I. INTRODUCTION

In wireless ad hoc and sensor networking, local algorithms for construction of planar backbone graphs have several applications, e.g., in the context of geographic unicast [1] and multicast [2] routing. Generally, a backbone of a network graph $G = (V,E)$ is a connected dominating set for $G$, i.e., a subset $S \subseteq V$, such that every node either belongs to $S$, or is adjacent to a node from $S$ in $G$. A graph is said to be planar if it can be embedded in the plane without edge crossings [3]. Moreover, a distributed algorithm is said to be local, if the computation at each node depends solely on the initial state/information provided to the nodes that are at most a constant number of hops apart.\footnote{A QUDG $G = (V,E)$ with parameters $r$ and $R$, $0 < r \leq R$, over node set $V \subset \mathbb{R}^2$ is defined as follows. Any two nodes $u,v \in V$ with Euclidean distance $|uv|$ share an edge in $E$ if $|uv| \leq r$, share no edge if $|uv| > R$, and may or may not be connected if $r < |uv| \leq R$. In the special case of $r = R$ the graph is the well known unit disk graph (UDG), than the unrealistic unit disk graph (UDG) model, but yet, it facilitates formal proofs of fundamental properties and correctness of algorithmic approaches.}\footnote{1In contrast, we say an algorithm is localized, if the computations at a node possibly depend on initial state/information of nodes that are more than a constant number of hops apart. For a detailed discussion, see [4].}

A wireless network graph can be reasonably modeled as quasi unit disk graph (QUDG).\footnote{2A QUDG $G = (V,E)$ with parameters $r$ and $R$, $0 < r \leq R$, over node set $V \subset \mathbb{R}^2$ is defined as follows. Any two nodes $u,v \in V$ with Euclidean distance $|uv|$ share an edge in $E$ if $|uv| \leq r$, share no edge if $|uv| > R$, and may or may not be connected if $r < |uv| \leq R$. In the special case of $r = R$ the graph is the well known unit disk graph (UDG), than the unrealistic unit disk graph (UDG) model, but yet, it facilitates formal proofs of fundamental properties and correctness of algorithmic approaches.}

This model captures the characteristics of the wireless channel much better than the unrealistic unit disk graph (UDG) model, but yet, it facilitates formal proofs of fundamental properties and correctness of algorithmic approaches. There are several local and localized algorithms for construction of planar backbones and other planar topologies over QUDGs [1], [5], [6], [7], [8], [9], [10]. However, these approaches are limited regarding two aspects: Firstly, they require the nodes to maintain their $k$-hop ($k \geq 2$) neighborhood tables via (periodic) beacon message exchange. However, beaconing is a costly and problematic process with regards to energy consumption and message collisions [11]. Secondly, these algorithms are designed to be executed by all network nodes, during a global network setup phase, and to span the output graph over the entire network. Yet, for some applications of planar graphs, like localized geographic routing with guaranteed delivery, network-wide planarization is not necessary. It suffices to construct the planar topology only in the geographic region of the routing event for computation of the next routing step. Distant network parts may remain unstructured, which avoids message overhead.

In this work we address both of the aforementioned problems by combining ideas from beaconless topology control (e.g., [12], [13], [14]) and beacon-based planar backbone construction techniques introduced by Lillis et al. in [1].

Beaconless topology control algorithms are on demand, local algorithms for construction of a node’s adjacency in a desired topology without beacon message exchange. By sending of a message, the executing node initiates a delay timer-based competition among its neighbors. Depending on their geographic positions, only those nodes that are relevant for topology construction respond upon time-out. This way, the initiating node solves the problem with incomplete neighborhood information. The majority of nodes transmit no messages. This saves valuable energy resources and may help to reduce the number of message collisions.

Lillis et al. [1] describe the following technique for construction of planar backbones (see Sec. III-B for further details). Let $G = (V,E)$ be a QUDG satisfying $R/r \leq \sqrt{2}$. Partitioning of the plane by a regular square grid of cell diameter $r$ yields a geometrical clustering of the nodes. In each non-empty cell, select one cluster head and one bridge edge to each adjacent cell in $G$. The set $V_b$ of cluster heads and endpoints of bridge edges is a connected dominating set for $G$. Obtain the planarized backbone graph $\text{Virt}(G_b)$.
of the backbone $G_b$, the graph induced by node set $V_b$ w.r.t. $G$, by replacing any edge intersection by a virtual node. Output $\text{Virt}(G_b)$ is a connected, planar backbone with constant node degree. If connecting each non-backbone node to its cluster head in $\text{Virt}(G_b)$, the resulting graph has constant hop spanning ratio (see Def. 1). Lillis et al. [1] devise a distributed algorithm for construction of $\text{Virt}(G_b)$. Yet, it requires nodes to maintain their complete 2-hop neighborhoods and is designed for network wide execution.

Our contributions: We present a beaconless algorithm for on demand computation of a node's adjacency in the planar backbone graph $\text{Virt}(G_b)$. To be exact, given a graph $G = (V, E)$ satisfying $R/r \leq \sqrt{2}$ and where nodes know their geographic position, initiated by any node $u \in V$, our local algorithm computes the adjacency of all nodes located in $u$'s cluster cell $C(u)$ in $\text{Virt}(G_b)$. During execution, only few nodes in the geographic vicinity (the $k$-neighborhood of $u$, where $k$ is a constant that only depends on transmission radius $R$) of cell $C(u)$ actively participate by sending messages, whereas all remaining nodes remain inactive. Distant network parts remain completely unstructured.

We rigorously prove correctness of our algorithm and show a worst-case message complexity of $\Theta(n)$, which is optimal assuming that nodes are initially unaware of their network neighborhoods. In general-case scenarios, our approach evidently outperforms beacon-based ones, regarding the number of message transmissions.

Outline of this paper: We review related work in Sec. II. In Sec. III we specify our network model, list assumptions, and provide definitions and context. Our main contributions are described in Sections IV–VI. First we prove the strictly local computability of the planar backbone graph $\text{Virt}(G_b)$ (Sec. IV). Then, we show how to compute the backbone in a beaconless fashion (Sec. V) and how to planarize it (Sec. VI). The paper is concluded in Sec. VII.

II. RELATED WORK

We now briefly review related work on beaconless topology control and planar graph construction. Regarding the latter, we restrict ourselves to algorithms for QUDGs. For a recent survey on UDG algorithms see [15].

A. Beaconless topology control

The local minimum recovery strategy of Guaranteed Delivery Beaconless Forwarding (GDBF) [12] and its generalization Beaconless Forwarder Planarization (BFP) [13], are beaconless algorithms for computation of a node’s adjacency in connected, planar proximity graphs, such as Gabriel Graph (GG), Relative Neighborhood Graph (RNG), or Circulunar Neighborhood Graph (CNG) [13]. In a selection phase the executing node $u$ sends a request. Neighbors answer after a delay proportionally to their Euclidean distance to $u$. Some neighbors overhear answers and decide not answer at all. As it may occur that non-GG (non-RNG, non-CNG) neighbors answer, in the protest phase, protest messages are sent by some neighbors, which correct the outcome of the selection phase. Yet, GG, RNG, and CNG are no constant Euclidean or topological spanners [13], [16].

ReactivePDT [14] is a beaconless algorithm for construction of a node’s adjacency in the Partial Delaunay Triangulation (PDT), which is a constant stretch Euclidean spanner of UDG [17]. ReactivePDT is similar to BFP but avoids protest messages by using dynamically adjustable delay timers in the selection phase.

All mentioned approaches require UDG properties for provable correctness. Moreover, direct extensions to QUDGs are not possible since PDT, GG, RNG, and CNG are not necessarily connected if applied onto QUDGs [5], [7].

B. Planar spanners and backbones for QUDGs

We speak of a restricted QUDG if $R/r \leq \sqrt{2}$ is assumed. Otherwise we speak of an arbitrary QUDG.

The algorithms in [5], [6], [7] first add virtual edges to the restricted input UDG and then use standard UDG topology control techniques, like GG and PDT, for planarization. The virtual edges ensure connectivity of the output graph. These algorithms are only local if the input graph is civilized (i.e., if any two nodes have Euclidean distance at least $\lambda > 0$). The output graphs of [5], [6] do not have constant spanning ratios, whereas for those produced by [7] this is unknown.

Chavez et al. [8] introduce a local algorithm for connected, bounded degree subgraphs of arbitrary QUDGs, based on local minimum spanning trees. Their output graph is planar if the input graph is civilized for $\lambda \geq \sqrt{1 - t^2}$. It is unknown if it provides constant spanning properties.

Funke et al. [9] propose a localized scheme for geographic routing with guaranteed delivery on restricted QUDGs. Nodes choose a $k$-hop independent set of landmark nodes in a localized fashion. The locally planarized combinatorial Delaunay graph over these landmarks serves as a backbone. The resulting routing graph may be disconnected [11].

Kuhn et al. [10] present two localized algorithms. The first one is a clustering based approach for localized routing graph construction on arbitrary QUDGs. Essentially, they build a backbone on top of an interconnected maximal independent set. For further sparsification, they apply standard spanner techniques to geographic clusters obtained via a square-grid shifting approach. The resulting routing graph is a constant stretch hop spanner, but it is not planar.

For restricted input graphs, the authors describe a related protocol, where the virtual edge approach from [5] is applied on the subgraph induced by the interconnected independent set nodes. Then, adding of virtual links does not require non-local communication and the resulting routing graph is a constant stretch hop spanner. As an alternative for planarization, they propose to replace edge intersections by virtual nodes.
III. MODEL AND PRELIMINARIES

A. Model, assumptions, and notation

We model network graphs as quasi unit disk graphs (QUDG) \( G = (V, E) \) over finite and distinct node sets \( V \subset \mathbb{R}^2 \), where \( |V| = n \), as defined in the introduction. Throughout this paper we adopt the usual assumptions (see e.g., \([1],[5],[6],[9]\)) that nodes are provided with their geographic positions, are aware of the maximum and minimum transmission radii \( R \) and \( r \), respectively, and that \( R/r \leq \sqrt{2} \). The latter is assumed for the following reason.

Lemma 1: Let \( G = (V, E) \) be a QUDG with \( R/r \leq \sqrt{2} \). If two edges \( uv, xy \in E \) intersect, then there exists \( wz \in E \) with \( w \in \{u, v\} \) and \( z \in \{x, y\} \).

For a proof see Lem. 8.2 in \([10]\) or Lem. 1 in \([1]\).

Generally, beaconless algorithms make use of a delay function for delaying a node’s operation (e.g., sending of a message) by a fraction or multiple of time slice \( t_{\text{max}} > 0 \), an arbitrary but fixed constant known to all nodes. Typically, we want to delay the response of node \( v \) to some request proportionally to an Euclidean distance \( \delta \) that depends on \( v \)'s geographic position. We use the following delay function: \( t(\delta, \Delta) = (\delta/\Delta) \cdot t_{\text{max}}, \) where \( 0 < \delta \leq \Delta \).

Like all other provably correct beaconless algorithms (e.g., \([12],[13],[14]\)), ours relies on the assumption that messages are transmitted instantly and reliably. In addition, all of these algorithms do not consider message collisions, which is critical w.r.t. practical implementations. However, such issues have previously been discussed in \([13],[18]\) and are out of the scope of this work.

\[ D(u,v) \] denotes the closed disk having line segment \( uv \) as its diameter. The internal angle of two intersecting line segments \( uw \) and \( ew \) is denoted by \( \angle uww. \) \( |uw| \) refers to the Euclidean distance between \( u \) and \( v \), whereas \( d_G(u,v) \) denotes their (minimum) hop-distance w.r.t. graph \( G \).

Definition 1 (Spanning ratio): A subgraph \( H = (V, E') \) of \( G = (V, E) \), \( E' \subseteq E \), has hop spanning ratio \( t \) (is a hop t-spanner for \( G \)), iff \( d_H(u,v)/d_G(u,v) \leq t \), for all \( u, v \in V \).

In case of Euclidean shortest path distances, it is referred to as Euclidean spanning ratio or Euclidean t-spanner.

B. Geographic clustering, backbones, and virtual nodes

A geographical clustering of a QUDG \( G = (V, E) \) can be obtained via the following standard technique (see e.g., \([1],[3],[10]\)). Virtually place an infinite, axes-parallel square grid with one grid vertex in the origin \((0,0)\) and grid cell diameter \( r \) on the plane (see Fig. 2a). Since nodes know transmission radius \( r \) as well as their own position in a global coordinate system, each node can immediately determine to which cell it belongs. For any node \( u \), let \( C(u) \) denote the unique grid cell (or simply cell) containing \( u \). Note that all nodes contained by one cell form a clique in \( G \).

If \( G \) is connected, a connected backbone graph (or simply backbone) \( G_b = (V_b, E_b) \) can be obtained by the following standard technique (see e.g., \([1]\)). For each non-empty cell \( C \) of the geographical clustering, chose one node in \( C \) (e.g., the node closest to the cell’s center) as cluster head of \( C \), denoted \( H(C) \), and add the set of cluster heads to \( V_b \).

For any two cells \( C_i, C_j \) that are connected by at least one bridge edge \( uv \in E \), i.e., an edge with \( u \in C_i \) and \( v \in C_j \), add the endpoints of one such bridge edge to \( V_b \). For any two nodes \( u, v \in V_b \), add edge \( uv \) to \( E_b \), iff \( uv \in E \), i.e, \( G_b \) is the graph induced by the node set \( V_b \) w.r.t. \( G \).

Backbone \( G_b \) has the following properties (see proof of Lem. 4 in \([1]\)). (i) The node degree of any node in \( G_b \) is \( O(1) \). (ii) Each edge in \( G_b \) is intersected at most \( O(1) \) times. (iii) \( V_b \) is a connected dominating set for \( G \).

Backbone \( G_b \) can be transformed into a planar graph \( \text{Virt}(G_b) \) using an idea previously described in \([1],[10]\). Graph \( \text{Virt}(G_b) \) is obtained by replacing each edge intersection of two edges \( uv, xy \in E_b \) by a virtual node \( w \) and the following operations: Edges \( uv, xy \) are deleted and edges \( uw, vv, xv, yw \) are added to \( \text{Virt}(G_b) \).

Any virtual node is controlled by one physical proxy in \( G_b \). Let \( w \) be the virtual node corresponding to the intersection of edges \( uv \) and \( xy \) in \( G_b \). Then e.g., the node in \( \{u,v,x,y\} \) with lowest ID handles all messages sent to and sent by \( w \). Note that if all nodes in \( G_b \) are provided with 3-hop neighborhood information of \( G_b \), then by Lem. 1, a node adjacent to virtual node \( w \) knows all endpoints of the intersecting edges that created \( w \). Hence, it can compute which of the physical nodes controls \( w \) and forward all messages destined at \( w \) directly to its controller. For further details on message handling and routing in \( \text{Virt}(G_b) \) we refer the reader to \([1]\), where this concept is elaborated.

Define the routing graph \( G_r \), as \( \text{Virt}(G_b) \) combined with all non-backbone nodes from \( G \) connected to their cluster head in \( G_b \) by an edge from \( E \). The following theorem follows immediately from Lem. 4 and Lem. 7 in \([1]\).

Theorem 1 (Hop spanning ratio of \( G_b \)): Let \( G = (V, E) \) be any connected QUDG satisfying \( R/r \leq \sqrt{2} \). For any two nodes \( u, v \in V \), \( d_{G_b}(u,v) \leq (4c + 6) \cdot d_G(u,v) \), where \( c \in O(1) \) denotes the maximum number of intersections of any edge in \( G_b \).

IV. LOCAL COMPUTABILITY OF \( \text{Virt}(G_b) \)

Prior to the actual description of our beaconless algorithm for construction of \( \text{Virt}(G_b) \) in Sections V and VI, we now prove several new propositions regarding the local computability of \( \text{Virt}(G_b) \). More formally, given any non empty cell \( C \) of a geographical clustering of a QUDG \( G = (V, E) \), we prove that each node \( v \in C \), can correctly compute its adjacency in \( \text{Virt}(G_b) \) based on local information, even if we construct \( G_b \) only in the geographic vicinity of \( C \).

Definition 2: We say that two grid cells \( C \) and \( Z \) are neighboring cells, if there exist a line segment \( pq \) of length at most \( R \), s.t. \( p \in C \) and \( q \in Z \). The neighborhood of a cell \( C \), denoted \( N_1(C) \), is the set of all neighboring cells of
Lemma 2: Let \( vw, xy \in E \) be two intersecting edges. It holds that \( \{C(v), C(x), C(y)\} \subseteq N_2(C(u)) \).

Proof: Since \( |vw| \leq R \), \( C(v) \in N_1(C(u)) \). Now, consider the quadrangle formed by \( u,v,x,y \). At least one of the internal angles is at least \( \pi/2 \). W.l.o.g. assume \( \angle xyw \geq \pi/2 \). Then, \( v \in D(x,y) \). Since \( |xy| \leq R \), it holds that \( |xv|, |yv| \leq R \). Hence, \( C(x) \) and \( C(y) \) belong to \( N_1(C(v)) \) which is a subset of \( N_2(C(u)) \).

Definition 3: Let \( Z \in N_2(C(u)) \) be any cell. We say \( Z \) is reachable w.r.t. \( N_2(C(u)) \), if there is a path \( \pi(x,z) \) between any node \( x \in C(u) \) and \( z \in Z \) in \( G \), s.t. all nodes in \( \pi(x,z) \) are contained by \( N_2(C(u)) \).

Lemma 3: Let \( Z \in N_2(C(u)) \) be a cell which is not reachable w.r.t. \( N_2(C(u)) \). No edge in \( G \) with at least one endpoint in \( Z \) can intersect with an edge from \( G \) that has at least one endpoint in \( C(u) \).

Proof: For the sake of contradiction assume that \( Z \) is not reachable w.r.t. \( N_2(C(u)) \), but there are two intersecting edges \( wv, xy \in E \), with \( x \in Z \) and \( u \in C(u) \). By Lem. 1 it holds that at least one of the nodes in \( \{x,y\} \) is connected to one of the nodes in \( \{u,v\} \) by an edge in \( G \). W.l.o.g. assume \( yv \in E \). By Lem. 2, \( \{C(u), C(v), C(x), C(y)\} \subseteq N_2(C(u)) \). But then \( \pi(u,x) = \{u,v,y,x\} \) is a path from \( u \) to \( x \) for which it holds that all nodes belong to \( N_2(C(u)) \) and thus, \( Z \) is reachable w.r.t. \( N_2(C(u)) \), which is a contradiction.

Lem. 3 implies the following. Edges in \( G_b \) with at least one endpoint in \( C \) can only be intersected by other edges in \( G_b \), whose endpoints are contained by cells in \( N_2(C) \) and that are reachable w.r.t. \( N_2(C) \). Therefore, we refer to the area represented by all reachable cells in \( N_2(C) \) as interaction area of \( C \), denoted \( I(C) \).

Based on this observation we now prove a theorem with the following meaning. To compute the adjacency of a backbone node \( v \) in \( \text{Virt}(G_b) \), it is in fact sufficient to compute the subset of nodes from \( G_b \) in \( I(C(v)) \) and then to planarize the graph induced by these nodes via virtual nodes. Nodes outside of \( I(C(v)) \) are irrelevant.

Let \( E_G(v) \) denote the edge set incident to \( v \) in graph \( G \).

Theorem 2: Let \( G = (V,E) \) be a QUDG with \( R/r \leq \sqrt{2} \) and let \( G_b = (V_b,E_b) \) and \( \text{Virt}(G_b) \) be the corresponding backbone and planarized backbone graph, respectively. Let \( u \in V \) be any node. Define \( V'_b = \{v \in V_b \mid v \in I(C(u))\} \) and let \( G'_b \) be the graph over node set \( V'_b \) induced w.r.t. \( G \). Then, for any node \( v \in V'_b \) with \( v \in C(u) \) it holds that \( E_{\text{Virt}(G'_b)}(v) = E_{\text{Virt}(G_b)}(v) \).

Proof: We have to show: \( vw \in E_{\text{Virt}(G'_b)}(v) \) if and only if \( vw \in E_{\text{Virt}(G_b)}(v) \).

\( \Rightarrow \) We distinguish case (i) \( vw \in E \) and (ii) \( vw \notin E \).

(i) Assume \( vw \in E_{\text{Virt}(G'_b)}(v) \), but \( vw \notin E_{\text{Virt}(G_b)}(v) \). Since \( v,w \in V'_b \subseteq V_b \) and \( vw \in E \), \( vw \in E_b \) holds. In order for \( vw \) not to belong to \( E_{\text{Virt}(G_b)}(v) \) it must be the case that \( vw \) is intersected by some other edge \( xy \in E_b \). By Lem. 2 and Lem. 3 it follows that \( C(x) \) and \( C(y) \) belong to \( I(C(v)) \). Thus, \( xy \) intersects \( vw \) in \( G'_b \) as well. Hence, \( vw \notin E_{\text{Virt}(G'_b)}(v) \), which contradicts the assumption.

(ii) Assume \( vw \in E_{\text{Virt}(G'_b)}(v) \), but \( vw \notin E_{\text{Virt}(G_b)}(v) \). Node \( w \) is a virtual node created by two edges \( vz,xy \in E_b \). By construction of \( G'_b \), \( vz,xy \in E_b \) holds. Thus, \( w \in \text{Virt}(G_b) \). Hence, in order for \( vw \) not to belong to \( E_{\text{Virt}(G_b)}(v) \) it must be the case that edge \( vw \) is intersected by another edge from \( E_b \), which is not contained in \( E'_b \). But this leads to a contradiction, using an analog line of argumentation as in case (i).

\( \Leftarrow \) We distinguish case (i) \( vw \in E \) and (ii) \( vw \notin E \).

(i) Let \( vw \in E_{\text{Virt}(G'_b)}(v) \). Since \( v \in C(u) \) and \( E(v) \subseteq I(C(u)) \) holds and hence, \( w \in V'_b \). Moreover, if \( vw \) is not intersected in \( G'_b \), then \( vw \) is not intersected in \( G_b \), since \( V'_b \subseteq V_b \). Thus, \( vw \in E_{\text{Virt}(G_b)}(v) \). (ii) Virtual node \( w \) is created by two edges \( vz,xy \in E_b \). With \( V'_b \subseteq V_b \) it cannot be the case that edge \( vw \) is intersected by an edge in \( E'_b \) which is not contained in \( E_b \). Moreover, by Lem. 3, \( vz \) can only be intersected by edges with endpoints in \( I(C(v)) \). Hence, \( vz,xy \in E'_b \) and therefore \( vw \in E_{\text{Virt}(G'_b)}(v) \) holds.

V. BEACONLESS COMPUTATION OF BRIDGE EDGES

We are now ready for introduction of our beaconless algorithms. At first we introduce algorithm BeaconlessBridgeEdge (BBE). After execution by any node \( u \in V \), for each cell in \( N_1(C(u)) \) that is connected by an edge in \( E \) to \( C(u) \), node \( u \) is aware of exactly one such edge, called representative bridge edge. Initially node \( u \) is unaware of its neighborhood. During the execution it exchanges messages only with few neighbors. Apart from worst-case situations, which we consider in Sec. VI-E, most neighbors never send a message and therefore remain invisible for \( u \).

A. Description of algorithm BeaconlessBridgeEdge

Our algorithm proceeds in two phases. In Phase I, initiator \( u \) determines one representative bridge edge to each neighboring cell in \( N_1(C(u)) \) to which it is connected in \( G \). Sending of a FirstRequest by \( u \) initiates a contention-based competition among nodes in neighboring cells. A FirstResponse by a node in a neighboring cell suppresses all other responses from that cell.

The goal of Phase II is to determine one representative bridge edge for every neighboring cell in \( N_1(C(u)) \) for which no bridge edge was found during Phase I. To do so, for every such “unprocessed” cell, node \( u \) starts a competition among nodes in its cell by sending a SecondRequest. Those nodes in \( C(u) \), whose communication radii intersect the “unprocessed” cell become candidates. A contention mechanism ensures that only one candidate at a time tries to find a neighbor in this cell. This process works just like Phase I. If the candidate is successful, it sends a SecondResponse.
to \( u \), otherwise, the next candidate tries to find a neighbor. Sending of a FirstRequest by a candidate always extends delay timers of initiator \( u \) and other candidates by time \( t_{\text{max}} \).

The algorithm uses four different types of messages. Any message includes the position of its sender \( s \). Cells are always represented by their geometric centers.

- **FirstRequest** \((s, L, i)\) contains a list \( L \) of unprocessed cells, and integer \( i \in \{I, II\} \) indicating the phase.
- **SecondRequest** \((s, C)\) contains an unprocessed cell \( C \).
- **FirstResponse** \((s, d, i)\) is a node's answer to a FirstRequest previously sent by destination \( d \). It contains an integer \( i \in \{I, II\} \) indicating the phase.
- **SecondResponse** \((s, v, d)\) is a node's answer to a SecondRequest previously sent by destination \( d \). It contains the position of a node \( v \) which is located in the cell addressed by the corresponding SecondRequest.

A detailed description of our algorithm is provided in Fig. 1. Note that \( u \) always refers to the distinguished node which has initiated the algorithm's execution.

**B. Correctness**

To prove termination, we show that every node involved in the computation reaches a state in which no timer is running and no message sending is scheduled.

**Theorem 3**: Execution of BBE, initiated by any node \( u \in V \), on any QUDG \( G = (V, E) \) terminates.

**Proof**: Note that message sending by nodes \( v \neq u \), implies expiration, i.e., not cancellation, of delay timers.

After a delay of \( t_{\text{max}} \) past sending the FirstRequest, initiator \( u \) terminates Phase I and every neighbor \( v \) of \( u \) in \( G \) has either decided not to start a timer, or has canceled its delay timer, or has actually answered with a FirstResponse. I.e., all of these nodes have terminated at the moment when Phase II is being entered.

At the beginning of Phase II, \(|L_u| \leq |N_1(C(u))| \in O(1)| \) and during each loop pass, one element is removed from \( L_u \). Hence, to show termination of Phase II, it suffices to show that a single loop pass terminates.

Let \( Z_i \) be the head of list \( L_u \) removed in round \( i \) and \( m_l \) is \( Z_i \)'s transmission range. Let \( C \) be the (possibly empty) distance ordered set of candidates (which are nodes in \( C(u) \)) for \( Z_i \), i.e., for any two \( c_j, c_l \in C \), \( j < l \Leftrightarrow |c_j m_l| < |c_l m_l| \), if \( C = \emptyset \), then no node in \( C(u) \) schedules sending of a FirstRequest and after time \( t_{\text{max}} \), initiator \( u \) proceeds with the next loop pass. Otherwise, let \( C' = \{c_1, ..., c_g\} \subseteq C \), with \( g \leq |C| \), be the longest consecutive subsequence of \( C \), beginning with the first element \( c_1 \) in \( C \). s.t. nodes in \( C' \) do not have an edge in \( E \) with the second endpoint being contained in cell \( Z_i \).

An inductive argument on the length of \( C' \) yields that after time at most \( (g + 1) \cdot t_{\text{max}} \), all nodes other than the initiator's have already expired. After delay of at most \( t_{\text{max}} \), the delay timer of initiator \( u \) expires as well. I.e., no node has a running timer or has scheduled sending of a message and the next loop pass begins. In the remaining case, the next expiring delay timer belongs to \( c_g + 1 \), the first element from \( C \setminus C' \). By construction of \( C \), \( 3z_{g+1} z \in E \), with \( z \in Z_i \) and \( z \) answers to the FirstRequest by \( c_{g+1} \), which causes \( c_{g+1} \) to send a SecondResponse for \( Z_i \), which cancels delay timers and message transmissions of all other candidates in \( C \) as well as the delay timer of \( u \), which then

![Figure 1. Pseudocode description of alg. BeaconlessBridgeEdge (BBE).](image-url)
proceeds with the next pass of the while-loop.

To prove correctness, we require the following notation. Let \( N(C(u)) \subseteq N_1(C(u)) \setminus \{C(u)\} \) denote the subset of neighboring cells of \( C(u) \), which are connected by a bridge edge to \( C(u) \) in the input graph \( G \). We partition the set \( N(C(u)) \) into two disjoint subsets \( N^{(1)}(C(u)) \) and \( N^{(2)}(C(u)) \) as follows. \( N^{(1)}(C(u)) \) encompasses exactly all cells in \( N_1(C(u)) \) to which node \( u \) shares an edge in the input graph \( G \), whereas \( N^{(2)}(C(u)) \) encompasses the remaining cells from \( N_1(C(u)) \) to which \( u \) does not share an edge, but other nodes in \( C(u) \) do so.

**Lemma 4:** After termination of Phase I, \( u \) knows one representative bridge edge for each cell in \( N^{(1)}(C(u)) \).

**Proof:** If \( N^{(1)}(C(u)) = \emptyset \), then no neighbor of \( u \) in \( G \) starts a delay timer for a FirstResponse. After time \( t_{\text{max}} \), \( u \) terminates with an empty set of representative bridge edges.

Otherwise, let \( C \) be any cell in \( N^{(1)}(C(u)) \). By definition of this set, there exists \( V' \subset V \), \( V' \neq \emptyset \), whose elements are contained by \( C \) and share an edge with \( u \) in \( E \). The node in \( V' \) being closest to the geometric center of \( C \) is the first to send a FirstResponse. This message is overheard by \( u \), which stores \( uv \) as representative bridge edge, and it is overheard by all other nodes in \( V' \), which causes cancellation of these nodes’ timers and scheduled transmissions.

**Lemma 5:** After termination of Phase II, \( u \) knows one representative bridge edge for each cell in \( N^{(2)}(C(u)) \).

**Proof:** After termination of Phase I, list \( \mathcal{L}_u \) is a superset of \( N^{(2)}(C(u)) \). Before the first execution of the while-loop, \( 0 < |\mathcal{L}_u| \leq |N_1(C(u))| \) holds. Hence, the loop is actually being executed. Moreover, as the size of \( \mathcal{L}_u \) decreases in every round by one element, there are at most \( |N_1(C(u))| \) rounds of execution. We show for an arbitrary \( i \leq |N_1(C(u))| \): If cell \( C_i \), processed during round \( i \), belongs to \( N^{(2)}(C(u)) \), then at the end of round \( i \), \( u \) stores a representative bridge edge connecting \( C(u) \) and \( C_i \). As the execution of a round is independent of the outcome of previous rounds, this proves our claim.

Case \( C_i \in N^{(2)}(C(u)) \): There is a non-empty subset \( V' \subset V \) of candidates for \( C_i \), all of which are connected in \( G \) to at least one node in \( C_i \). Let \( v \in V' \) be the node minimizing the distance to the geometric center of \( C_i \). At the moment this node’s delay timer expires, it has not overheard any other SecondResponse for \( C_i \). This follows from the choice of \( v \). Hence, \( v \) sends a FirstRequest for \( C_i \), which is being answered by some neighbor \( w \in C_i \), whose existence is guaranteed by assumption. In consequence, \( v \) sends a SecondResponse. This is overheard by all nodes in \( C(u) \). Node \( u \) stores \( vw \) as bridge edge for \( C_i \) and other candidates in \( V' \) cancel their delay timers and scheduled transmissions.

Case \( C_i \notin N^{(2)}(C(u)) \): There is either no candidate in \( C(u) \), or no candidates shares an edge in \( G \) with an endpoint in \( C_i \). In the former case, no message transmissions are scheduled by nodes in \( C(u) \). After delay \( t_{\text{max}} \), \( u \) continues with the next pass of the loop. In the other case, each candidate sends a FirstRequest for \( C_i \), which is not being answered. Hence, no candidate sends a SecondResponse and \( u \) continues with the next loop pass. In either case, \( u \) proceeds without storing an representative edge for \( C_i \).

**Theorem 4:** After the execution of algorithm BBE on any QUDG \( G = (V,E) \) by any \( u \in V, u \) stores one bridge edge for each cell in \( N(C(u)) \).

**Proof:** The proof follows from Lem. 4 and Lem. 5.

VI. BEACONLESS CONSTRUCTION OF VIRT(\( G_b \))

Recall our main goal. Given a distinguished node \( u \), we want to compute the adjacency in the planar backbone graph \( VIRT(G_b) \) for every backbone node in \( G_b \) which is located in cell \( C(u) \). A node belongs to \( G_b \) if it is cluster head of its grid cell or if it is an endpoint of a representative bridge edge connecting its cell to a neighboring cell. To achieve our goal, by Thm. 2 it is sufficient to compute set \( V'_b \), the set of cluster heads and endpoints of representative bridge edges contained in \( I(C(u)) \), and then to planarize the subgraph \( G'_b \), the graph induced by \( V'_b \) w.r.t. input graph \( G \), via introduction of virtual nodes.

Prior to description of algorithmic details, we give an overview and minimal example, see Fig. 2b, for explanation and illustration of the main ideas and communication flow.

A. Algorithm overview and example

Our algorithm proceeds in two stages. In the first stage, set \( V'_b \) is determined by repeated execution of algorithm ExtendedBBE, an extension of algorithm BBE introduced in the previous section. In the second stage, the planar graph \( VIRT(G'_b) \) is constructed using algorithm \( k \)-Neighborhood.

The first stage proceeds as follows (see Fig. 2b): Distinguished node \( u \) executes algorithm ExtendedBBE on input \( C \), which causes cluster head election in its cell \( C \). Node \( u \), which is closest to \( C \)'s center, becomes cluster head, denoted \( H(C) \), and starts construction of representative bridge edges as explained in the previous section. Node \( v \) has no neighbors in \( N_1(C) \) and thus, no FirstResponse is generated during Phase I. During Phase II, eventually node \( w \) sends a
FirstRequest for cell $C'$, which is answered with a FirstResponse by $a$. At this point, ExtendedBBE has not yet been executed in $C'$. Therefore, $a$ starts this process which causes a cluster head election in $C'$. In addition, it notifies $w$ by a message of type $Triggered$ that $C'$ was triggered by $C$. Node $w$ forwards this information to its cluster head $v = \mathcal{H}(C)$, which maintains a list of triggered neighboring cells. Node $b$ becomes cluster head of cell $C'$ and starts construction of representative bridge edges. During Phase II, eventually node $a$ sends a FirstRequest for $C$ which is answered with a FirstResponse by $w$. However, in $C$, ExtendedBBE has already been executed and therefore, $C'$ does not trigger $C$. Eventually all neighboring cells have been processed, $b$ sends a message of type $Terminated$ to $a$, which has initiated cluster head election in $C'$. Node $a$ forwards this message to $w$, whose FirstRequest triggered cell $C'$. Node $w$ forwards this message to its cluster head $v$. Cluster head $v$ removes $C'$ from the list of triggered neighboring cells. Since bridge edge construction by $v$ has not triggered other neighboring cells, execution of ExtendedBBE in all cells in $\Gamma(C)$ has been completed and $v$ terminates. One additional message from $v$, which we omit in our pseudocode description, could then be used to notify all other nodes in $C$, in particular node $u$, about the termination. Now, all nodes from $G_b'$ have been constructed, namely $\{v, w, a, b\}$.

The second stage is essentially repeated broadcasting of backbone neighborhood information, i.e., a node’s neighborhood in $G_b'$ (not in $G$, as this would contradict our beaconless approach). In our minimal example, $v$ broadcasts its set of neighbors $\Gamma^v_G(v) = \{v\}$. Backbone node $w$ receiving this message updates its set of 1-hop backbone neighbors $\Gamma^w_G(v) = \{v\}$ and broadcasts its set $\Gamma^w_G(w) = \{w\}$. Eventually all backbone nodes in $G_b'$ know their full 1-hop backbone neighborhoods. This process is repeated with 1-hop and 2-hop backbone neighborhoods, respectively. Upon termination, all nodes from $G_b'$ located in $C(u)$ know their complete 3-hop neighborhoods in $G_b'$ and can compute their adjacency in $\text{Virt}(G_b')$, which is the desired result.

Next, we provide the algorithmic details.

B. Beaconless cluster head election

Let $x \in V$ be any node and let $c$ denote the geometric center of $C(x)$. Beaconless election of a cluster head in $C(x)$, initiated by $x$, can be achieved as follows.

Node $x$ locally broadcasts a $CH-Request$ containing its position. Every node $v \in V$ with $C(v) = C(x)$ (including $x$), schedules sending of a $CH-Response$, containing its position, after delay of $t_1|v|c|/2)$. If upon time-out $v$ has not overheard a CH-Response by some other node $w \in C(x)$, then it locally broadcasts the CH-Response. The sender of the CH-Response is then the self-proclaimed cluster head of $C(x)$, denoted $\mathcal{H}(C(x))$. All other nodes in $C(x)$ overhear this message, cancel their timers and scheduled transmissions and store $v$ as $\mathcal{H}(C(x))$. This algorithm is correct, as there is at least one node, namely initiator $x$, contained by the corresponding cell. Hence, one node actually responds and suppresses other CH-Responses.

C. Description and correctness of ExtendedBBE

For our extension of BBE we require that the messages of the types FirstRequest, FirstResponse, SecondRequest, and SecondResponse (introduced in Sec. V) additionally contain information on distinguished cell $C(u)$. Then, each node $v$ receiving such a message can immediately decide whether $C(v) \in N_2(C(u))$ holds, or not. For this reason, algorithm ExtendedBBE is always started on input $C(u)$. In addition, we require the following new message types. Each message includes the positions of its sender $s$.

- $CH-Request(s,C)$ contains a cell $C$. This message requests election of a cluster head in $C(s)$.
- $CH-Response(s)$ is the announcement of $s$ that it becomes cluster head of $C(s)$.
- $Triggered(s,d,C)$ informs destination $d$ that its FirstRequest triggered execution of ExtendedBBE in cell $C$.
- $Terminated(s,d,C)$ informs destination $d$ that execution of ExtendedBBE in cell $C$ has terminated.

The algorithm details are provided in Fig. 3. This includes message handlers for the new message types and extensions to the pseudocode description of algorithm BBE in Sec. V-A. “After sending a FirstResponse” has to be executed by a node directly after sending a FirstResponse. “After termination of Phase II” is executed by every cluster head that has just terminated bridge edge selection, i.e., that has reached step 5) in the description of BBE in Fig. 1.

Lemma 6: Node $\mathcal{H}(C(u))$ terminates eventually.

Proof: Execution of ExtendedBBE in $C(u)$ triggers execution of it in neighboring cells, as long as they belong to $N_2(C(u))$, and so on. This gives a directed tree of cells, rooted at $C(u)$, containing an edge $CC'$ if cell $C$ has triggered cell $C'$. Each cell has only a constant number of neighboring cells and the number of cells contained in $N_2(C(u))$ is only a constant factor larger. Moreover, one cell can only be triggered once. Hence, $T$ consists of a constant number of internal cells and leaf cells. Leaf cells represent those which have not triggered others. Hence, their cluster heads have empty lists of triggered neighboring cells, stored in $L_{\text{wh}}(\cdot)$. Eventually, these cluster heads have processed all neighboring cells and terminate Phase II, by Thm. 3, and then send a message of type Terminated. Cluster heads of ancestor cells of these leaf cells in $T$ receive and process the Terminated messages and remove the corresponding cells from their lists of triggered neighboring cells. An induction on the hight of tree $T$ yields that, whenever all cells of hight $i$ have terminated, then all cluster heads of cells of height $i − 1$ have empty lists of triggered neighboring cells and terminate as well. Hence, eventually all successors of the root node have terminated and $\mathcal{H}(C(u))$ receives terminated
messages from all such cells. Then, list \( L_{\text{ng}}(H(C(u))) \) is empty and \( H(C(u)) \) terminates. \( \blacksquare \)

**Lemma 7:** Once \( H(C(u)) \) has terminated, ExtendedBBE has been executed in every cell from \( I'(C(u)) \).

**Proof:** Assume there is \( Z \in N_2(C(u)) \), which is reachable w.r.t. \( N_2(C(u)) \), but in which ExtendedBBE has not been executed. Because \( Z \) is reachable, there exists a path \( \pi(x, z) \), with \( x \in C(u) \) and \( z \in Z \), connecting \( x \) and \( z \) in \( G \), s.t. all nodes of this path belong to \( N_2(C(u)) \). Let \( C = (C(u) = C_1, ..., C_l = Z) \), \( l \geq 2 \), be the sequence of cells visited, while traversing \( \pi(x, z) \). We prove the claim by induction on the length of \( C \).

**B.c.** (\( |C| = 2 \)): By Thm. 4, during execution of ExtendedBBE one representative bridge edge \( c_1c_2 \) connecting \( c_1 \in C_1 \) and \( c_2 \in C_2 \) is found. At this point, either another node in \( C_2 \) has already started execution of ExtendedBBE, or \( c_2 \) does so.

i.h.: The claim holds for all \( i \).

I.s. (\( i \rightarrow i+1 \)): By the i.h., it holds that all cells \( C_i \), \( i < |C| \), have executed ExtendedBBE. By the time it is executed in \( C_i \), \( C_{i+1} \) either has already been triggered by some other cell in \( N_2(C(u)) \), or not. Consider the latter case. By assumption there exists at least one edge \( c_ic_{i+1} \in \pi(x, z) \) with \( c_i \in C_i \) and \( c_{i+1} \in C_{i+1} \). By Thm. 4 one such edge is eventually determined. Since \( C_{i+1} \) belongs to \( N_2(C(u)) \) the corresponding endpoint in \( C_{i+1} \) executes ExtendedBBE. \( \blacksquare \)

**D. Construction of Virt(\( G_b \))**

Once the cluster head of the distinguished cell \( C(u) \) has terminated after execution of ExtendedBBE in cell \( C(u) \), we are in the position to complete the task, namely to provide each backbone node in cell \( C(u) \) with its adjacency in the planarized backbone graph \( \text{Virt}(G_b) \).

Provided that the input QUDG \( G \) is connected and satisfies \( R/r \leq \sqrt{2} \), then \( G_b \) is a connected QUDG also satisfying \( R/r \leq \sqrt{2} \). This follows from the fact that the backbone nodes \( V'_G \) form a connected dominating set for the subgraph induced by all nodes from \( G \) that are located in \( I'(C(u)) \). By Lem. 1 and Thm. 2, each backbone node in \( C(u) \) can compute its adjacency in \( \text{Virt}(G_b) \) if it is provided with 3-hop neighborhood information of \( G_b \). Note that gathering of 3-hop neighborhood by a node in \( G_b \) does not contradict to our beaconless approach, since the degree of any node in \( G_b \) is bounded from above by a constant and likewise is its 3-hop neighborhood.

Define the \( k \)-hop neighborhood of a node \( v \in V \) in \( G = (V, E) \) as follows: \( \Gamma^k_G(v) = \{ v \} \) and for \( k \geq 1 \), \( \Gamma^k_G(v) \) is the subset of nodes from \( V \) reachable by \( v \) in \( G \) along a path consisting of at most \( k \)-hops.

For completeness, we describe in Fig. 4 the simple protocol \( k \)-Neighborhood. Initiated by the cluster head of distinguished cell \( C(u) \), \( H(C(u)) \), it enables all backbone nodes in \( C(u) \) to determine their \( k \)-hop neighborhoods in \( G_b \). It uses the following message type.

**Neighborhood\((s,C(u),i,\Gamma^i_{G_b})\)** contains the positions of sender \( s \) and distinguished cell \( C(u) \), an integer \( 0 \leq i < k \), and the \( i \)-hop neighborhood set \( \Gamma^i_{G_b} \).

**Theorem 5:** After termination of \( k \)-Neighborhooed, initiated by \( H(C(u)) \), each backbone node in \( C(u) \) knows all its \( k \)-hop neighbors in \( G_b \).

**Proof:** We prove by induction on control variable \( i \) that after the \( i \)-th iteration of the loop, all nodes in \( \Gamma^i_{G_b} \) know their \( i \)-hop neighborhoods in \( G_b \).

**B.c.** (\( i = 1 \)): After the first round of the loop, i.e., after time \( t_{\text{max}} \), each node in \( \Gamma^1_{G_b} \) has locally broadcasted its set \( \Gamma^1_{G_b}(v) \). Hence, all nodes in \( \Gamma^1_{G_b} \) know their 1-hop neighborhoods.

i.h.: The claim holds for all \( i < k \).

I.s. (\( i \rightarrow i+1 \)): By the i.h., at the beginning of round \( i+1 \), each node in \( \Gamma^{i+1}_{G_b} \) has complete \( i \)-hop neighborhood information. During iteration \( i+1 \), each such
node locally broadcasts this set. Hence, after round $i = 1$, all nodes in $\Gamma_{C_u}^{i-1}(\mathcal{H}(V(u)))$ have complete $i + 1$-hop neighborhood information. Thus, after round $k$, all nodes in $\Gamma_{C_u}^{i-k}(\mathcal{H}(V(u)))$ have complete $k$-hop information.

\begin{algorithm}
\caption{Description of exec. of $k$-Neighborhood by $v = \mathcal{H}(C(u))$}
\begin{algorithmic}[1]
\IF{$i = 1$ to $k$}
\STATE a) Send Neighborhood($v,C(u),i-1,\Gamma_{C_u}^{i-1}(v)$)
\STATE b) Start a delay timer with delay of $l \cdot t_{\text{max}}$, where $l = k + 1$.
\ENDIF
\ENDFOR
\STATE 2) Terminate
\STATE Processing of Neighborhood($v,C(u),i,\Gamma_{C_u}^{i}(v)$) by node $v$
\begin{enumerate}
\item If $C(v) \in N_2(C(u))$ and $v$ is a backbone node then
\begin{enumerate}
\item $\Gamma_{C_u}^{i+1}(v) = \Gamma_{C_u}^{i+1}(v) \cup \Gamma_{C_u}^{i}(w)$
\end{enumerate}
\item If sending of Neighborhood($v,C(u),i,\Gamma_{C_u}^{i}(v)$) has not been scheduled yet, then schedule sending of it after a delay of $t(|vw|,R)$.
\end{enumerate}
\end{algorithmic}
\end{algorithm}

\section{Discussion of message complexity and message size}

Finally, we discuss the worst-case message complexity\footnote{The worst-case message complexity of a distributed algorithm is the total number of messages sent during its execution in the worst-case.} of our approach as well as the size of messages used therein. Then, we reason why our approach is in fact beneficial compared to any beacon-based local approach.

The number of cells in $N_2(C(u))$ is is upper bounded by a constant which only depends on transmission radius $r$. Leader election requires two message transmissions per cell. In each cell a constant number of Triggered and Terminated messages are produced. Moreover, during the execution of algorithm $k$-Neighborhood on input $k = 3$, each backbone node in $\mathcal{H}(V(u))$ transmits exactly three messages. Since there are only $O(1)$ backbone nodes in $N_2(C(u))$ (see proof of Lem. 4 in [1]), this message overhead is constant as well.

By far the most expensive part w.r.t. message complexity is the construction of representative bridge edges. Consider any cell $C \in \mathcal{I}(C(u))$. The number of FirstRequest/FirstResponse messages (during Phase I) as well the number of SecondRequests is upper bounded by $|N_1(C)| \in O(1)$. During Phase II any node in $C$, which may be as much as $O(n)$ in the worst-case, transmits at most one FirstRequest. Moreover, at most one SecondResponse is generated. In addition, $|N_1(C)| \in O(1)$ many FirstResponses may be sent from nodes in neighboring cells. Hence, the total number of message transmissions is $O(n)$ in the worst-case.

Note that any such message is of size $O(P_{\text{max}})$ bits, where $P_{\text{max}}$ denotes the number of bits needed to represent a node position. This follows immediately from the fact that any message contains at most a constant number of geographic positions plus a few additional control bits.

We show next that beaconless bridge edge construction requires $\Omega(n)$ message transmissions in the worst-case and that this is best possible under the assumption that network nodes are unaware of their neighborhoods at algorithm start.

Note that a similar proof has recently been given in [19].

Consider the example depicted in Fig. 5. All nodes in $V$ are located and aligned in a single cell s.t. each $v_i$ uniquely covers a piece of a particular neighboring cell $C$. Then, during algorithm execution initiated by any $v_i \in V$, every single node in $V$ eventually sends one FirstRequest for $C$ and hence, $\Omega(n)$ messages are transmitted. Moreover, under the assumption that nodes do not know any other network node at algorithm start, in the example at hand it is in fact inevitable that each node sends at least one request message, since each node uniquely covers some area in a different cell possibly containing a neighboring node. No matter how the algorithm chooses the order of request message sending, one can always provide an input graph such that only the very last request is actually successful.

Lastly, note that beacon-based local approaches cannot do any better than our beaconless approach, for the following reason. In worst-case scenarios, all nodes have to transmit messages regardless of whether a beaconless, or beacon-based approach is used. In all other scenarios, there are nodes which transmit no message if using our beaconless approach, but which have to transmit messages in any beacon-based scheme. In fact, it is easy to construct examples where our approach produces only $O(1)$ message transmissions, but where any beacon-based approach produces $\Omega(n)$ many. Therefore, it is predictable that, on average, our approach outperforms any beacon-based solution progressively with increasing network density in ordinary or random scenarios. Simulations can merely help to quantify this gap.

\section{Conclusion}

We present a beaconless algorithm for on demand construction of planar backbone graphs with constant hop stretch under quasi unit disk graph model assumptions. Our approach is novel and advantageous compared to previous works in various respects. Firstly, it is completely reactive. Secondly, the backbone graph is constructed only in the geographic vicinity of a specific node, whereas other approaches are designed for network-wide construction. Hence, it much
better meets requirements of applications like geographic routing with guaranteed delivery, where the backbone is needed only locally for computation of the next routing step. Thirdly, our approach is beaconless and reduces message overhead compared to any beacon-based approach. Especially in dense networks, only few nodes actively participate by sending messages. Most of the nodes do not transmit messages at all and hence, save their energy resources.

Quite recently, we have established the notions of $\Omega$- and $\Omega$-reactive local topology control algorithms [19], which allow us to distinguish reactive from conventional local approaches and to classify the former based on their message complexity. According to this formalism, the algorithm presented in this paper belongs to the class of $\Omega$-reactive local topology control algorithms. To the best of our knowledge, it is the first such algorithm under QUDG model assumptions.

Finally, the present approach can be adapted for use in all algorithms requiring local constructions of connected dominating sets and planar backbones, e.g., [1], [2], [3], [20], [21] and can help therein to reduce message overhead. Moreover, in a subsequent step, it can be extended by a grid-shifting approach like in [1], [10] in order to obtain also constant Euclidean spanning properties.

ACKNOWLEDGMENT

The authors would like to acknowledge that some ideas regarding beaconless clustering have been developed by Julian Mosen in his bachelor’s thesis [22] under their supervision. Moreover, the present work is supported by the German Research Foundation (DFG), grant “FR 2978/1-1”.

REFERENCES


