Topology control

5

Topology Control

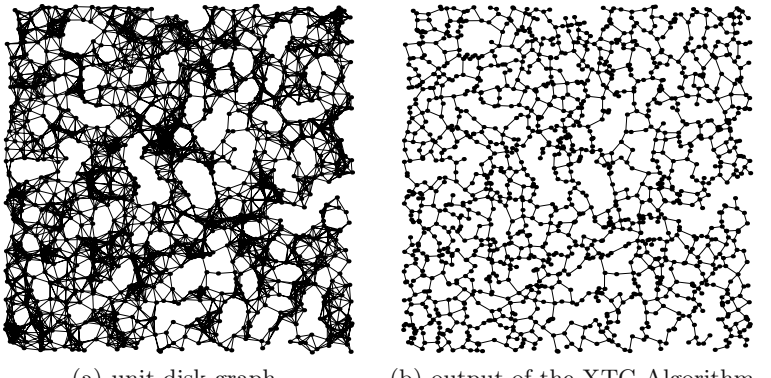
Kevin Buchin and Maike Buchin

5.1 Introduction

Information between two nodes in a network is sent based on the network topology, the structure of links connecting pairs of nodes of a network. The task of topology control is to choose a connecting subset from all possible links such that the overall network performance is good. For instance, a goal of topology control is to reduce the number of links to make routing on the topology faster and easier.

In this chapter we assume that the network topology can be controlled by varying the transmission radii of the nodes and selecting and discarding possible links. Topology control was first considered by Tagaki and Kleinrock [355]. The effect of topology control on the network is demonstrated in Figure 5.1 (taken from [392]).

In the following we consider localized algorithms, i.e., algorithms where decisions for establishing connections are done locally by the nodes. Each



(a) unit disk graph (b) output of the XTC Algorithm

Fig. 5.1. A network without and with topology control applied.

D. Wagner and R. Wattenhofer (Eds.): Algorithms for Sensor and Ad Hoc Networks, LNCS 4621, pp. 81–98, 2007.

node independently explores its surrounding region and establishes connections with neighboring nodes in its transmission range. A localized algorithm should be sufficiently simple and efficient such that it can be carried out by the mobile nodes.

Alternatively one can consider centralized algorithms for establishing the network topology. These gather all connectivity and location information in one node, construct the topology of the whole network, and propagate it to all other nodes. We do not consider centralized algorithms since a localized construction and maintenance is preferred because of the limited resources and the mobility of the nodes.

Topology control aims to conserve energy as the power in an ad hoc network is limited. The topology should allow for efficient routing and improve the overall network performance. These goals can be formalized in different ways. In Section 5.2 we give an overview of quality criteria that influence the network performance based on the topology.

The topology control problem is well described in a graph-theoretic setting. Consider the bidirected graph $G = (V, E)$, where the node set V consists of the nodes of the network, and an edge (u, v) between the nodes u and v exists if the nodes u and v lie in each other's transmission range. If the nodes have differing transmission radii, this yields a directed graph where edges exist if one node lies in the transmission range of the other (and not necessarily vice versa). We will assume uniform maximal transmission ranges.

The task of topology control can now be formulated as follows: Given the graph $G = (V, E)$ compute a subgraph $G' = (V, E'), E' \subseteq E$ which allows for energy efficient routing and increases the network performance and lifetime. These goals can be measured by several criteria, such as connectivity and sparseness of G', which are described in the next section.

Based on the energy model used, one can also consider *edge weights* for the graph G. I.e., consider the weight function w which assigns to each edge e in E a weight $w(e)$, which is the energy needed to transport information along the edge e. The energy cost of a *path* in the graph is the sum over the costs of the edges in the path.

Typical edge weights are given by the link metric, the Euclidean distance between the endpoints of the edge or an energy metric which takes the Euclidean distance and raises it to a power greater than 1. For the link metric all edge weights are one, i.e., the number of hops are counted. An energy metric raises the Euclidean distance to an exponent called *attenuation exponent* which is typically a value between 2 and 5. I.e., the edge e between two nodes u and v obtains one of the weights

$$\mathrm{w}_{\mathrm{link}}(e) := 1, \quad \mathrm{w}_{\mathrm{dist}}(e) := d_2(u, v), \quad \text{or} \quad \mathrm{w}_{\mathrm{energy}(a)}(e) := d_2(u, v)^a$$

where $a > 1$ and $\mathrm{w}_{\mathrm{link}}, \mathrm{w}_{\mathrm{dist}}$, and $\mathrm{w}_{\mathrm{energy}(a)}$ denote the edge weights for the link, distance, and energy metric, respectively.

Taking the Euclidean distance between the endpoints requires that the graph is embedded in the Euclidean plane or space, i.e., each node has coor-

dinates. In general, nodes need not know their coordinates. Some algorithms require that they know their coordinates (e.g., assume that all nodes have a GPS), other algorithms require that the nodes can determine the distances or directions to nodes in their neighborhood and some algorithms do not use any distance or direction information at all. Often algorithms assume nodes to have distinctive IDs.

Many localized algorithms for topology control compute a locally defined geometric graph. Therefore we give an overview of locally defined geometric graphs in Section 5.3 and in Section 5.4 describe some algorithms computing different locally defined graphs.

5.2 Quality Criteria

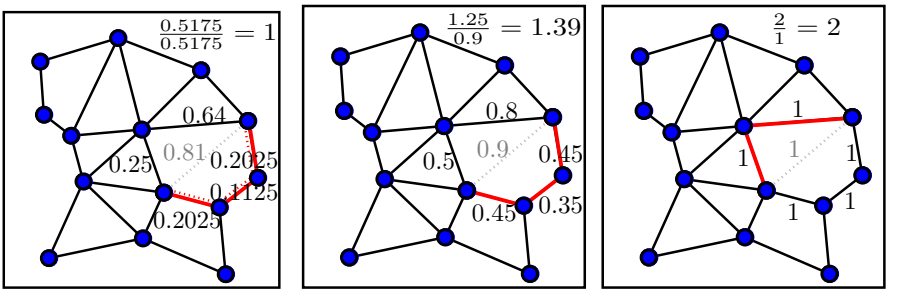
In this section we consider different criteria that may be used to measure the quality of a network topology.

Connectivity. Perhaps the most basic requirement for a topology is *connectivity*, i.e., that there is a path between any pair of vertices in V. More specifically, one requires that any two nodes that are connected by a path in G should also be connected by a path in G'. Sometimes the stronger notion of *k-vertex-connectivity* or *k-edge-connectivity* is required. A graph is k-vertex-connected (k-edge-connected) if at least k vertices (edges) need to be removed for the graph to become disconnected. The reason to ask for connectivity is simple: A topology G' that is less connected than G does not have the same capabilities to transfer information. Asking for k-connectivity might lower the risk of path loss due to interference.

Sometimes instead of total connectivity a *giant connected component* suffices. A connected component is a maximal connected subgraph. A giant connected component is a connected component of linear size, i.e., containing cn vertices, where n is the number of vertices in V and $0 < c \leq 1$ a constant. For instance if the task of an ad hoc network is to compute an average value over the network, a giant connected component containing all but few of the vertices will yield a value very close to that of the total network. For obtaining a giant connected component instead of total connectivity, often a much smaller transmission radius is sufficient [323], and therefore much less energy.

Symmetry. Another basic requirement for a topology is *symmetry*, which means that if (u, v) is an edge in G', so is (v, u). In a graph-theoretic sense this means that the computed graph G' should be bidirectional. Many routing algorithms require symmetry. Without symmetry, communication in the network becomes more complicated, e.g., it may not be possible to send back an acknowledgement (ACK) for received data.

Stretch Factors. A main goal of topology control is high *energy-efficiency* to increase the network lifetime. Depending on the energy model used, energy-efficiency can be measured in different ways, typically using one of the following *stretch factors*: *energy stretch factor*, *hop stretch factor*, or *distance*



(a) energy stretch factor (attenuation 2) (b) distance stretch factor (c) hop stretch factor

Fig. 5.2. Example for stretch factors for a pair of nodes.

stretch factor, see Figure 5.2. A stretch factor is the largest ratio of a quantity measured in a subgraph and in the full communication graph.

The energy stretch factor of a subgraph G' in G is the largest ratio for any pair of vertices in V between the energy of the minimum energy path in G' and in G. Formally, the energy stretch factor is $\max_{u,v \in V} \frac{E_{G'}(u,v)}{E_G(u,v)}$ where $E_G(u,v)$ denotes the energy of the minimum energy path between u and v in G. The energy of a path is the sum of the energies of the edges, where the energy of an edge is measured according to the chosen energy model.

The hop stretch factor is the largest ratio of number of hops, i.e., number of edges, in G' and G, respectively, between any two vertices in V.

The distance stretch factor is the largest ratio of the Euclidean lengths of shortest paths in G' and G, respectively, for any pair of vertices in V. The problem of designing graphs with low distance stretch factor has been extensively studied by network designers. If an attenuation exponent larger than 1 is used in the energy model, a constant distance stretch factor implies a constant energy stretch factor.

A subgraph G' of G that has a constant distance stretch factor is a *spanner* of G. A spanner is a subgraph G' of a graph G in which the distance between any two nodes in G' is within a constant factor of the distance in G.

Sparseness. The topology determined by a topology control algorithm should be simple and easy to maintain. These subjective goals are influenced by the *sparseness* of the graph. A graph is called sparse if it has few edges: The number of edges is linear in the number of nodes, i.e., the average vertex degree is constant. The *degree* of a vertex is the number of of edges adjacent to the vertex. The maximal vertex degree may still be high in a sparse graph. A stronger requirement is therefore that all vertex degrees are bounded by a constant.

Due to the low complexity of sparse graphs, routing algorithms are more time and power efficient in sparse graphs. There is a tradeoff between connectivity and sparseness: A sparse graph is likely to have low connectivity.

Throughput. Topology control aims at a topology with large *throughput*, i.e., where as much data as possible can be sent over the network. A measure for the throughput of a network is its *bit-meter* capacity. We say that the network transports one bit-meter if one bit has been transported over a distance of one meter. The network's bit-meter capacity is the amount of bit-meters that can be transported in one second.

The bit-meter capacity measures the optimal throughput of a network. For measuring the worst case throughput, *throughput competitiveness* can be used. This is the largest number ϕ, $0 \leq \phi \leq 1$, such that for any pair of vertices (v, w), if a flow of r can be routed in G from v to w then a flow of ϕr can be routed in G' from v to w.

Interference. Interference occurs when a node cannot receive messages because several nodes in its transmission range send at the same time. Topology control handling interference is discussed in Chapter 6.

Adaptability. A further goal of topology control is to determine a topology that is robust to mobility, i.e., that can react quickly to changing network conditions. This is measured by the adaptability of a network, which is the maximum number of nodes that need to change their topology information as a result of the movement of a node. All locally defined graphs have reasonable adaptability because in case a node moves the nodes in its neighborhood can locally recompute their edge sets.

Planarity. Some geographic routing algorithms require that the embedding of the topology in the plane does not contain crossing edges, i.e., it is a planar straight-line graph.

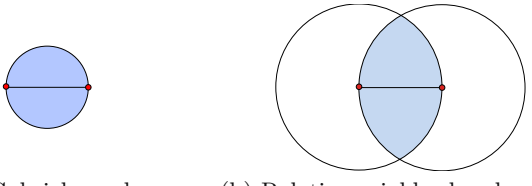
5.3 Locally Defined Geometric Graphs and Further Proximity Graphs

A localized algorithm for topology control computes a subgraph $G' = (V, E')$ of a given graph $G = (V, E)$ based on local decisions. This means that the decision whether an edge at a node $v \in V$ is selected to be in E' is made by considering the information of all nodes in a k-hop neighborhood of v, i.e., all nodes $w \in V$ for which there is a path starting at v and ending at w containing at most k edges for a constant k. For many topology control algorithms k equals one or two, i.e., only the information of the neighbors or of the neighbors' one-hop neighborhood is used.

If the nodes are assumed to lie in some geometric space, typically the Euclidean plane $\mathbb{R}^2$, this problem is closely related to geometric graphs which are defined by local properties. Many distributed topology control algorithms

compute such a locally definable graph, or a variant of such a graph or combine two locally definable graphs. In this section we give an overview of locally definable graphs, many of which have been used for topology control, explicitly or implicitly. We consider here only geometric graphs in the two-dimensional Euclidean plane $\mathbb{R}^2$.

An example of a locally definable graph is the *Gabriel graph (GG)*: Given a set of vertices V, two points from V are connected by an edge if the disk with these points as a diameter contains no further points from V, see Figure 5.3. If the set of possible edges is restricted to the edges of a graph G, we call this the restricted Gabriel graph denoted by $GG(G)$ (and analogously for other graphs).



(a) Gabriel graph (b) Relative neighborhood graph

Fig. 5.3. Empty neighborhoods of an edge.

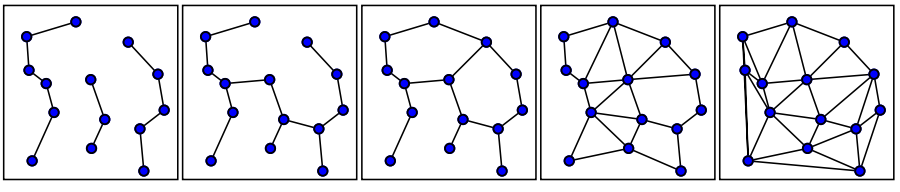
In the Gabriel graph edges are defined by an *empty neighborhood*, i.e., an edge is in the graph if a certain neighborhood of the edge does not contain any vertices. For the Gabriel graph the empty neighborhood of an edge is the disk which has the edge as a diameter. Such graphs are considered in detail in Section 5.3.1. These graphs are typically sparse, i.e., have a low total number of edges though the maximum vertex degree might be high. A shortcoming of these graphs is that they might have a high distance stretch factor.

A better stretch factor can be achieved by connecting a point to a nearby point in all directions. An example is the *Yao graph*: For every point the plane is divided into a set of cones with a fixed angle with the point as base and the point is connected with the nearest neighbor in every cone. These kinds of graphs are described in Section 5.3.2.

Closely related to locally definable graphs are *proximity graphs*, i.e., geometric graphs with edges between vertices that are by some means close to each other. These graphs are discussed in Section 5.3.3. The Gabriel graph is such a graph. But not all proximity graphs are locally definable. For instance a *minimal spanning tree* is a minimal (with respect to edge weight) connected graph on a set of points. Another important proximity graph is the *Delaunay triangulation*. It can be defined by a local property of the triangles. This property does not yield a local property for the edges. Therefore, localized variants of the Delaunay triangulation are considered.

In Section 5.3.4 we consider graphs which have an *exclusion region*. An exclusion region is similar to an empty neighborhood: a region close to an edge that needs to be empty for the edge to be in the graph. For an exclusion region, in contrast to an empty neighborhood, not the whole region needs to be empty, but the region is divided into two halves by the edge and one of the halves needs to be empty. E.g., a possible exclusion region is the empty neighborhood of the Gabriel graph, i.e., the disk which has the edge as diameter, and an edge is then in the graph if one of the half-disks is empty.

Figure 5.4 shows examples for most of the graphs that are described in the rest of this section. The graphs are shown in order of inclusion. Note that the graph inclusion $H \subset G$ implies that the graph H has smaller degree and larger stretch factors than the graph G. Many of the properties of the graphs in this section are also described in the references given in the chapter notes [199][255]. At the end of this section the most important graphs and their properties are summarized in Table 5.3.4.



(a) NNG (b) EMST (c) RNG (d) GG (e) DT

Fig. 5.4. Examples of locally defined geometric graphs in order of inclusion: nearest neighbor graph (NNG), Euclidean minimum spanning tree (EMST), relative neighborhood graph (RNG), Gabriel graph (GG), and Delaunay triangulation (DT).

In the following n always denotes the number of vertices, i.e., $n = |V|$. Note that a connected and planar graph has at least $n-1$ edges (because of connectivity) and at most $3n - 6$ (using Euler's formula). In particular, a connected and planar graph is sparse.

5.3.1 Graphs with Empty Neighborhoods

Gabriel Graph (GG). The empty neighborhood of an edge in the Gabriel graph is the disk with the edge as a diameter (see Figure 5.3). Formally, an edge (v, w) exists if there is no node u such that $d(v, u)^2 + d(w, u)^2 \leq d(v, w)^2$. I.e., there is no vertex u s.t. (u, v, w) form a triangle with an angle larger than $\pi/2$ at u.

The Gabriel graph is a bidirectional and connected graph. It is planar and also sparse. The distance stretch factor of the Gabriel graph is $\sqrt{n-1}$ and it is not a spanner. The Gabriel graph has optimal energy stretch factor 1; the maximum vertex degree is $n - 1$.

Relative Neighborhood Graph (RNG). The empty neighborhood of an edge in the relative neighborhood graph is the intersection of the two disks centered at the vertices of the edge with their radii equal to the distance between the two vertices (see Figure 5.3). This intersection is called the *relative neighborhood* of the edge. One can also say that the two connected vertices are relatively close, i.e., there is no node closer to both vertices than they are to each other. Formally, an edge (v, w) exists if

$$d(v, w) \leq \max_{u \in V \setminus \{v,w\}} (d(v, u), d(w, u)).$$

The relative neighborhood graph is a bidirectional and connected graph. It is planar and also sparse. Its distance stretch factor is $n - 1$, i.e., it is not a spanner, and its energy stretch factor is also $n - 1$. The maximum vertex degree is $n - 1$ but can be bounded by 5 if no two points have the same distance to a third point. If two points do have the same distance to a third point, we will call this a *tie* and discuss it further in Section 5.4.3, where an algorithm using the relative neighborhood graph is described. If one uses *tie breaking*, i.e., choosing only one edge in the case of a tie, the maximum vertex degree can be bounded by 6.

***β*-Skeletons.** β-skeletons are a parameterized family of graphs, which are parameterized by a positive, real number β. There are two versions of β-skeletons: lune-based and circle-based. In both versions an edge is established if its β-neighborhood is empty. For $0 < \beta \leq 1$ the β-neighborhoods of lune- and circle-based β-neighborhoods are the same. For larger β, the β-neighborhoods of lune-based β-skeletons are the intersection of two circles whereas the β-neighborhoods of circle-based β-skeletons are the union of two (different) circles. As an example we now give the definition of the β-neighborhoods for lune-based β-skeletons for $\beta \geq 1$ (For the other cases see [217]).

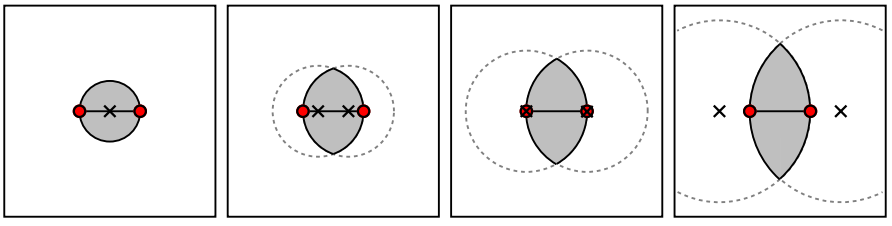
The β-neighborhood of an edge in a lune-based β-skeleton is the intersection of two circles. For $\beta \geq 1$ the circles have centers lying on the line defined by the edge and moving away from the edge with growing β. More specifically (see Figure 5.5),

- for $\beta = 1$ both circle centers are the midpoint of the edge,
- for $1 < \beta < 2$ the centers move from the mid- to the endpoints of the edge,
- for $\beta = 2$ the centers are the endpoints, and
- for $\beta > 2$ the centers move further away from the edge.

The radii of the circles grow with β and they are chosen such that always the opposite endpoint of the edge lies on the boundary of the circle. Formally, the lune-based β-neighborhood of an edge (v, w) and $\beta \geq 1$ is

$$B\Big((1 - \frac{\beta}{2})v + \frac{\beta}{2}w,\ \frac{\beta}{2}d(v, w)\Big) \ \cap \ B\Big((1 - \frac{\beta}{2})w + \frac{\beta}{2}v,\ \frac{\beta}{2}d(v, w)\Big)$$

where $B(c, r)$ denotes the disk with center c and radius r and $d(v, w)$ is the Euclidean distance between the vertices v and w.



(a) $\beta = 1$ (b) $1 < \beta < 2$ (c) $\beta = 2$ (d) $\beta > 2$

Fig. 5.5. β-Skeletons.

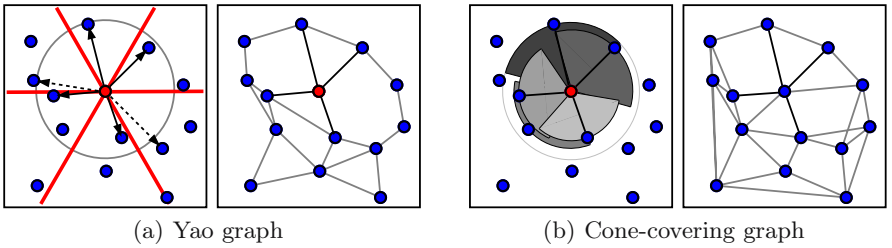
Both lune- and circle-based β-skeletons are a generalization of the Gabriel graph. Lune-based β-skeletons are also a generalization of the relative neighborhood graph. I.e., for $\beta = 1$ both β-skeletons equal the Gabriel graph and for $\beta = 2$ the lune-based β-skeleton is the relative neighborhood graph (see Figure 5.5).

Another parameterized family of graphs are γ-neighborhood graphs [376], which generalize circle-based β-skeletons and the Delaunay triangulation.

5.3.2 Cone-Based Graphs

Yao Graph (YG). The Yao graph is a graph parameterized by a constant $k \geq 6$. For each vertex, the plane is divided into k cones of equal angle and the vertex is connected to the nearest vertex in each cone (see Figure 5.6(a) left). There may be several closest vertices in one cone, which we again call a *tie*. In this case *tie breaking* can be used, i.e., choosing one or more of the closest vertices to connect to. In the original definition [400], an arbitrary closest vertex was selected. An alternative is to connect to all closest vertices.

This yields a weakly connected directed graph, "weakly" meaning that — disregarding edge directionality — all nodes are connected. The bidirectional graph, in which edge directions are ignored, is called the *undirected Yao graph* (see Figure 5.6(a) right). The Yao graph is typically not planar, but for fixed



(a) Yao graph (b) Cone-covering graph

Fig. 5.6. Cone-based graphs.

k it is sparse. It is a spanner with distance stretch factor $1/(1-2\sin(\pi/k))$ and energy stretch factor $1/(1-(2\sin(\pi/k))^{\alpha})$ where α is the attenuation exponent.

The maximum vertex degree is $n-1$, but the outdegree can be bounded by k if no two vertices have the same distance to a third vertex, or if, as in the original definition, in the case of a tie one vertex is chosen. A graph with constant distance stretch factor and bounded vertex degree can be constructed from the Yao graph [19]. The Yao graph contains the relative neighborhood graph if in case of a tie one connects to all closest vertices or if it is assumed that no two points have the same distance to a third vertex.

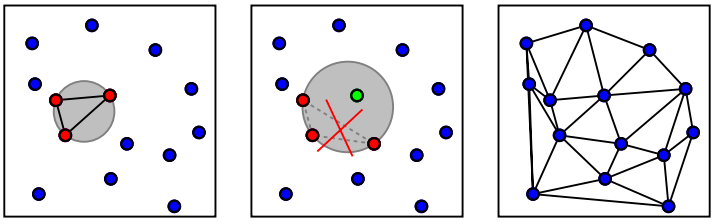
Θ-Graph. The Θ-graph is often used synonymously to the Yao graph. It is originally defined sightly different [212]: Instead of connecting to the nearest vertex in each cone, one connects to the vertex for which the projection on the axis of the cone is the shortest. The Θ-graph has similar properties as the Yao graph.

Cone-Covering Graph. A graph related to the Yao graph that is the basis of an algorithm described in Section 5.4.1 is the following: Each vertex processes its neighbors by their distance, starting with its nearest neighbor. Edges to neighbors are established until the α-cones of the processed neighbors cover the complete angular range around the vertex (see Figure 5.6(b) left). This graph has similar properties as the Yao graph but typically the longest edge is shorter (and therefore the necessary transmission range smaller). To guarantee no empty cone of angle greater than α, the angles for the Yao graph need to be at most $\alpha/2$, i.e., it connects to $k \geq 4\pi/\alpha$ nodes. The degree of the cone-covering graph can be arbitrarily high, e.g., for points lying on a spiral. It can be efficiently constructed by incrementally increasing the transmission radius.

5.3.3 Further Proximity Graphs

Delaunay Triangulation (DT) and Variants. Two vertices are connected by an edge in the Delaunay triangulation if there exists an empty circle through the two vertices. The Delaunay triangulation is a bidirectional, connected, planar and sparse graph. All faces of the graph are triangles (with the assumption that no four or more points lie on a common circle). Formulated for triangles, one can say that a triangle is in the graph if the circle defined by the triangle is empty (see Figure 5.7). This is called the *empty circle property*. The distance stretch factor of the Delaunay triangulation lies between $\pi/2$ and $(4\sqrt{3}/9)\pi \approx 0.77\pi$. The energy stretch factor is 1 (the Delaunay triangulation contains the Gabriel graph), and the maximum vertex degree may be $n-1$.

In contrast to empty neighborhood graphs, the empty area in the Delaunay triangulation is not necessarily close to the points, but may reach arbitrarily far away. Therefore the Delaunay triangulation is not locally constructible in the sense that one may need to check vertices arbitrarily far away



(a) triangle (b) not a triangle (c) resulting graph

Fig. 5.7. Delaunay triangulation. A triangle is in the Delaunay triangulation if its circumcircle is empty.

from an edge. Therefore localized versions of the Delaunay triangulation have been defined: the k-localized Delaunay triangulation, the restricted Delaunay triangulation, and the partial Delaunay triangulation. We describe here the k-localized Delaunay triangulation for which an algorithm will be described in Section 5.4.2.

For the k-localized Delaunay triangulation one considers the *k-localized empty circle property*, which says that no point of the *k-neighborhood* lies in the circumcircle of the triangle. A k-neighbor is a vertex to which a path of length k exists in the original graph. The k-localized Delaunay triangulation is the Gabriel graph plus all triangles fulfilling the k-localized empty circle property. For $k = 1$ the k-localized Delaunay triangulation is not necessarily planar but it can be converted into a planar spanner by further processing. For $k \geq 2$ it is a planar spanner. The k-localized Delaunay triangulation contains the Delaunay triangulation, therefore the distance stretch factor is at most that of the Delaunay triangulation and the energy stretch factor is again 1. The maximum vertex degree is also $n - 1$.

Euclidean Minimum Spanning Tree (EMST). A Euclidean minimum spanning tree is a graph that connects all vertices and whose total Euclidean edge length is minimized. It is a *tree*, i.e., a graph without cycles. It is connected and sparse but its energy stretch factor and distance stretch factor are $n-1$, i.e., it is not a spanner. Note the difference between spanning and spanner: A spanning tree connects all vertices, a spanner is a graph with constant distance stretch factor.

The maximum vertex degree in a Euclidean minimum spanning tree is 6. For a given set of vertices, the Euclidean minimum spanning tree need not be unique. Many useful properties of the Euclidean minimum spanning tree have been proven by Gilbert and Pollack [160].

Nearest Neighbor Graph (NNG). The nearest neighbor graph is perhaps the most basic proximity graph. Each vertex is connected to its nearest neighbor. The nearest neighbor graph is a directed graph. It is in general not connected, and therefore has not been used for topology control. It is pla-

nar and sparse but not a spanner. It is contained in a Euclidean minimum spanning tree and thus has maximum vertex degree 6.

In the k–Nearest Neighbor graph each node is connected to its k nearest neighbors. Blough et al. [39] use the graph containing only the bidirectional edges, i.e., both endpoints are in the k–neighborhood of the other end point, to achieve a topology where each node has a bounded (by k) number of nodes in its transmission range. For nodes distributed uniformly at random in a square and a suitable constant λ the resulting graph is connected with high probability if $k \leq \lambda \log n$.

5.3.4 Graphs with Exclusion Regions

Even if a graph is not locally definable there might be a non-trivial larger graph containing it which is locally definable. In particular it might have an exclusion region: a region around an edge such that a necessary condition for the edge to be in the graph is that the region is empty on one side of the edge. In general the number of edges obtained in such a way might be quadratic, i.e., the resulting graph is not sparse. But the edges might be of interest as a candidate set for further processing.

An example of a graph with an exclusion region is the Delaunay triangulation. Its exclusion region is the disk which has the edge as a diameter (the same area defining the empty neighborhood for the Gabriel graph, but now we allow vertices to lie in one half of it). Again, the exclusion region only gives a necessary – not a sufficient – condition for an edge to be in the Delaunay triangulation.

Also the minimum weight triangulation (triangulation with minimum total edge length) allows for an exclusion region, as well as the minimum dilation triangulation (triangulation with minimum maximal stretch factor between any two nodes) and the greedy triangulation (incrementally constructed triangulation adding shortest edges first if possible). An overview over graphs with exclusion regions is given by Drysdale et al. [193].

The following table summarizes the most important graphs and their properties. The bounds in the table are worst case bounds.

5.4 Localized Algorithms

We describe here three localized topology control algorithms. Each algorithm makes use of one of the graphs described in the previous section: the cone-covering graph, a localized version of the Delaunay triangulation and the relative neighborhood graph.

5.4.1 Cone Based Topology Control (CBTC)

We describe a cone based algorithm proposed by Wattenhofer et al. in [391] and revised and extended by Li et al. in [254]. The algorithm consists of two

graph property	GG	RNG	YG	NNG	EMST	DT
connected	yes	yes	weakly	no	yes	yes
planar	yes	yes	no	yes	yes	yes
sparse	yes	yes	yes	yes	yes	yes
maximum vertex degree	$n-1$	$n-1$ $(5,6)$	$n-1$	6	6	$n-1$
distance stretch factor	$\sqrt{n-1}$	$n-1$	$\frac{1}{1-2\sin\frac{\pi}{k}}$	–	$n-1$	$\leq \frac{4\sqrt{3}}{9}\pi$
energy stretch factor	1	$n-1$	$(\frac{1}{1-2\sin\frac{\pi}{k}})^a$	–	$n-1$	1
spanner	no	no	yes	no	no	yes

Table 5.2. Graph properties of the Gabriel graph (GG), the relative neighborhood graph (RNG), the Yao graph (YG), the nearest neighbor graph (NNG), the Euclidean minimum spanning tree (EMST) and the Delaunay triangulation (DT). The a in the energy stretch factor of the Yao graph denotes the attenuation exponent.

phases: a basic step (establishing a candidate edge set) and several possible improvements (removing unnecessary edges) as a second step. An example for the resulting topology is shown in Figure 5.8 (taken from [254]).

Algorithm 14: Cone Based Topology Control

```
1 Neighbors N_u ← ∅
2 Directions D_u ← ∅
3 Transmission power p_u ← p_min
4 while p_u < p_max & ∃ empty α-cone in D_u do
5     Increase p_u
6     Broadcast and gather Acks
7     N_u ← N_u ∪ {v : discovered v}
8     D_u ← D_u ∪ {dir(v) : discovered v}
```

Basic Step. The basic algorithm 14 is easy to state: For a predefined angle α the algorithm finds for every node the smallest circular neighborhood such that there is a point in any cone of angle α starting at the node. I.e., in the basic step the cone-covering graph described in Section 5.3.2 is computed.

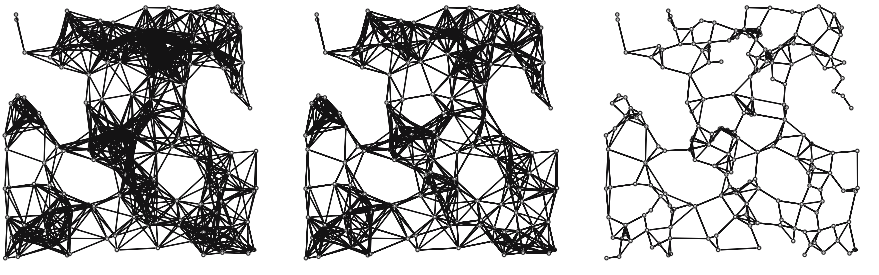
For this, each node incrementally increases its transmission radius and records the nodes it reaches. Whenever it reaches a new node it checks whether the cone defined by this node is already covered by previously established edges. If not, a new edge to this node is established. Note that nodes only need to know the distances and directions of their neighbors, and not their exact positions.

Then the symmetric closure of the up to now directed graph is computed. I.e., when each node has covered its angular range with cones or reached its maximal transmission range, it contacts all its neighbors that it has established edges to, and if necessary the neighbors add those edges. This symmetric closure is connected for any $\alpha \leq 5\pi/6$. For larger α it is not necessarily connected.

Remarks on the Connectivity:. In [254] it is proven, that the symmetric closure is connected. But actually a stronger result holds: It contains a Euclidean minimum spanning tree. This follows from Lemma II.2 in [254]:

Let R be the maximum transmission range and G_R the graph with edge set E_R containing all edges (u,v) lying in each others transmission range. Let E_α be the set of edges chosen by the algorithm. If $\alpha \leq 5\pi/6$, and u and v are nodes in V such that $(u, v) \in E_R$ (that is, (u, v) is an edge in the graph G_R, so that $d(u, v) \leq R$), then either $(u, v) \in E_\alpha$ or there exist $u', v' \in V$ such that (a) $d(u', v') < d(u, v)$, (b) either $u' = u$ or $(u, u') \in E_\alpha$, and (c) either $v' = v$ or $(v, v') \in E_\alpha$.

First observe that, using the unit disk graph model, if the maximal reachable graph (all possible edges established) is connected, it contains a Euclidean minimum spanning tree. Using Lemma II.2 one can show that in the basic step no Euclidean minimum spanning tree edges are removed: Assume a Euclidean minimum spanning tree edge is removed, then by the lemma it can be replaced by a shorter edge which is a contradiction to it being a Euclidean minimum spanning tree edge.



(a) unit disk graph (b) CBTC algorithm (c) CBTC algorithm and improvements

Fig. 5.8. Topologies computed by the CBTC algorithm.

Improvements. Several improvements which remove energy inefficient edges that have been established by the basic step are possible as a second step to the basic algorithm. All improvements preserve connectivity. In the original

work [391][254] this is proved directly for each improvement but it can also be seen by the following argument:

1. The original topology computed by CBTC contains a minimum spanning tree (if the unit disk graph is connected).
2. Any minimum spanning tree is contained in the relative neighborhood graph.
3. The improvements delete only edges which are not in the relative neighborhood graph and therefore no edges of the minimum spanning tree.
4. After applying the improvements the graph still contains a minimum spanning tree and is thus connected.

Shrink Back. In the basic algorithm, nodes might not find a neighbor in every direction. I.e., there may be directions which are not covered by cones, simply because there are no neighbors in that direction. In this case the transmission radius will have been set to the maximum. This improvement reduces these maximum transmission radii to the smallest transmission radii that reach all chosen neighbors. The resulting graph will still contain a Euclidean minimum spanning tree and is therefore still connected.

Asymmetric Edge Removal. The key of this improvement is that for $\alpha \leq 2\pi/3$ a Euclidean minimum spanning tree is a subset of the directional edges, analogously to the case of the Yao graph with $k \geq 6$. This follows from the fact that the Euclidean minimum spanning tree is contained in the relative neighborhood graph. In other words: For $\alpha \leq 2\pi/3$ any outgoing edge of the cone-covering graph without corresponding ingoing edge is not necessary to maintain a Euclidean minimum spanning tree as subgraph. Thus removing asymmetric, i.e., directed, edges, does not remove any Euclidean minimum spanning tree edges and the resulting graph is still connected.

Pairwise Edge Removal. In this improvement edges are removed that are not contained in the relative neighborhood graph. This is done using the following observation: If a node u is connected to two vertices v and w, and if either (u, v) or (u, w) are the longest edge in the triangle (u, v, w), it can be removed. This is in fact a reformulation of the definition of the relative neighborhood graph. Again a Euclidean minimum spanning tree, which is contained in the relative neighborhood graph, will still be contained in the resulting graph, which therefore is connected.

Triangle Inequality Edge Removal. This is the original improvement given in [391], which uses the triangle inequality for edge removal. An edge (v, w) is removed if for some vertex u the edges (u, v) and (u, w) are in the graph, and $p(u, v) + p(u, w) \leq p(v, w)$, where $p(e)$ denotes the energy of the edge e. If p is the quadratic distance function, then all removed edges are not edges of the Gabriel graph.

5.4.2 Delaunay Based Topology Control

Several topology control algorithms compute a graph containing the Delaunay triangulation restricted to the full communication graph, i.e., a graph containing all edges that are in the Delaunay triangulation as well as in the communication graph. The resulting topologies have constant distance stretch factor in the case of the unit disk graph model. The algorithms discussed assume that location information is given at each node.

Gao et al. [148] define a *restricted Delaunay triangulation* as any planar graph containing the Delaunay triangulation restricted to the unit disk graph. They combine such a graph with node clustering (i.e., the restricted Delaunay triangulation is used to connect clusters) and show how it can be maintained when nodes move.

Li and his co-authors proposed several topology control algorithms based on the k–localized Delaunay triangulation [256] described in Section 5.3. For achieving a reasonable run-time, k is typically chosen as 1 or 2. The 1–localized Delaunay triangulation is not necessarily planar but a planar subgraph can be extracted. For $k \geq 2$ the k–localized Delaunay triangulation is always planar. In the following we describe algorithm 15 for computing the k–localized Delaunay triangulation.

Algorithm 15: k-Localized Delaunay Topology Control

1 Collect location information of k–hop neighbors
2 Find triangles incident to node in local Delaunay triangulation
3 Broadcast and receive triangle information
4 Check if incident triangles are confirmed by all three vertices
5 Add Gabriel graph edges
6 Add edges of verified triangles

Each node collects the location of its k–hop neighbors and computes the Delaunay triangulation of this set of points. Of the computed triangulation it discards all triangles not incident to itself. It then sends a list of its incident triangles to its neighbors and receives their list of incident triangles. Each node checks if its incident triangles are verified by its neighbors. I.e., each node u checks for a triangle uvw if the same triangle is also in the list of triangles of v and w. If not, the triangle is discarded. The communication cost of the verification step can be reduced to $O(n)$ by letting each triangle be checked only by the vertices at which it has an angle of at least $\pi/3$.

5.4.3 Relative Neighbor Topology Control (XTC)

This algorithm was proposed by Wattenhofer and Zollinger in [392]. If based on Euclidean distances it computes a subgraph of the relative neighborhood graph, and uses only relative positional information for this.

Algorithm. The XTC algorithm computes a subgraph of the relative neighborhood graph using only relative positional information, i.e., each node is required to sort its neighbors by their distance. For this XTC does not use the description of the relative neighborhood graph by empty neighborhoods, which would require full positional information. Instead XTC uses the description of the relative neighborhood graph that an edge is established between two vertices u and v if there is no vertex w that is closer to both u and v than u and v are to each other.

Algorithm 16: Relative Neighbor Topology Control (XTC)

```
1 Establish order ≺_u over u's neighbors
2 Broadcast order
3 Receive orders
4 forall neighbors v in increasing order according to ≺_u do
5     if ∄w : w ≺_u v & w ≺_v u then
6         Add v to final neighbor set
```

More explicitly, XTC (algorithm 16, from node u's perspective) does the following: First each node determines its one-hop neighbors, i.e., all other nodes in its transmission range. Then it sorts its neighbors by any reasonable order, for instance the link qualities to its neighbors. This order may correspond to the order on the Euclidean distances between the nodes, but need not. Instead they may also depend on obstacles between the nodes. Symmetric edge weights are assumed. In case of a tie, i.e., when two neighbors have the same "distance" to it, tie breaking is used for achieving a strict order. This will be explained in the next paragraph.

All nodes exchange their ordered neighbor lists with their neighbors. Then each node u processes its neighbor list in order to select those neighboring nodes that define the neighborhood in the topology control graph: For each node v on the neighbor list it checks whether there is a node w that comes before v on its own neighbor list and before itself on the neighbor list of v. I.e., it checks for a node w s.t. $w \prec_u v$ and $w \prec_v u$, where $\prec_v$ and $\prec_w$ denote the neighbor orders of v and w, respectively. If this is the case, v is not added to the topology defining neighborhood, otherwise it is.

Tie Breaking. A tie occurs if two nodes have the same distance to a third node. To break the tie means to assign distinct and consistent distances to all pairs of nodes. This situation can be described more generally for a weighted graph where the goal is to have distinct and consistent edge weights. Assuming that all nodes have distinctive IDs, this can be done as follows: Assign to each edge an extended, more general weight, which is the triple of the former edge weight, the smaller endpoint id and the larger endpoint id. A strict order is then given by the lexicographic order on these weight triples. For the XTC

algorithm this means simply that if a node has two neighbors with equal distance it sorts them according to their IDs.

Properties of G_{XTC}. Let G_{XTC} denote the graph computed by XTC. G_{XTC} is symmetric by definition (the condition $u \prec_v w$ and $u \prec_w v$ is symmetric). G_{XTC} is connected in the general weighted graph model if the original graph was connected.

The following properties hold if Euclidean distances are used: The maximum vertex degree is 6, G_{XTC} is planar and a subgraph of the relative neighborhood graph. If also there are no ties, i.e., no two nodes that have the same distance to a third node, XTC exactly computes the relative neighborhood graph.

5.5 Chapter Notes

In this survey we discussed geometric graphs suitable for topology control and localized algorithms computing such graphs.

A survey on relative neighborhood graphs and its relatives is given by Jaromczyk and Toussaint [199]. The β-skeletons are proposed by Kirkpatrick and Radke [217]. Spanners, i.e., graphs with constant distance stretch factor, are discussed by Arya et al. [19] and Eppstein [113]. Drysdale et al. [193] give an overview of exclusion regions. Santi [338], [339] and Li [255] survey topology control for wireless ad hoc networks.

In Section 5.2 we gave criteria for topology control and in Section 5.3 described graphs which fulfill these criteria. In Section 5.4 we described algorithms computing some of these graphs. The criteria were described in a graph-theoretic sense and the reasoning why they are good for topology control is more or less heuristic. The criteria are also closely bound to the unit disk graph model and the assumption that the nodes lie in the Euclidean plane.

One of the main open problems in topology control is how realistic these assumptions are. In other words, what are realistic energy and interference models, and efficient topology control algorithms based on them? In particular, the impact of the mobility of nodes needs to be addressed. Several authors [39], [55] show that optimizing the given graph properties not necessarily implicates an improvement of the network performance. Approaches explicitly handling interference are further discussed in Chapter 6.