CHAPTER 4

Modeling Sensor Networks

STEFAN SCHMID AND ROGER WATTENHOFER Q1
ETH Zurich, Zurich, Switzerland

4.1 INTRODUCTION AND MOTIVATION

In order to develop algorithms for sensor networks and in order to give mathematical correctness and performance proofs, models for various aspects of sensor networks are needed. This chapter presents and discusses currently used models for sensor networks. Generally, finding good models is a challenging task. On the one hand, a model should be as simple as possible such that the analysis of a given algorithm remains tractable. On the other hand, however, a model must not be too simplistic in the sense that it neglects important properties of the network. A great algorithm in theory may be inefficient or even incorrect in practice if the analysis is based on idealistic assumptions. For example, an algorithm that ignores interference may fail in practice since communication happens over a shared medium. Many models for sensor network have their origin in classic areas of theoretical computer science and applied mathematics. Since the topology of a sensor network can be regarded as a *graph*, the distributed algorithms community uses models from *graph theory*, representing nodes by vertices and wireless links by edges. Another crucial ingredient of sensor network models is *geometry*. Geometry comes into play as the distribution of sensor nodes in space, as well as the propagation range of wireless links, usually adheres to geometric constraints.

The chapter is organized as follows. In Section 4.2, the reader will become familiar with various models for the network's *connectivity*. Connectivity models answer the question: Which nodes are "connected" to which other nodes and can therefore directly communicate with each other. Section 4.3 then enhances these connectivity models by adding *interference* aspects: Since sensor nodes communicate over a shared, wireless medium, a transmission may disturb a nearby concurrent transmission. After having studied connectivity and interference issues, we look at modeling questions related to algorithm design in Section 4.4. The reader is provided

Algorithms and Protocols for Wireless Sensor Networks, Edited by Azzedine Boukerche with a survey of models that influence the feasibility and efficiency of certain operations on sensor networks. We draw some general conclusions in Section 4.5, and we point out interesting areas for future research in Section 4.6.

4.2 MODELING THE SENSOR NODES' CONNECTIVITY

A first and foremost modeling question concerns the *connectivity* of sensor nodes: Given a set of nodes distributed in space, we need to specify which nodes can receive a transmission of a node. Throughout this chapter, if a node u is within a node v's transmission range, we say that u is *adjacent* to v, or, equivalently, that u is a *neighbor* of v. In the absence of interference (cf. Section 4.3), this relation is typically *symmetric* (or *undirected*); that is, if a node u can hear a node v, also v can hear u. The connectivity of a sensor network is described by a graph $G = (V, E)$, where V (*vertices*) is the set of sensor nodes, and E (*edges*) describes the adjacency relation between nodes. That is, for two nodes $u, v \in V$, $(u, v) \in E$ if v is adjacent to u. In an undirected graph, it holds that if $(u, v) \in E$, then also $(v, u) \in E$; that is, edges can be represented by sets $\{u, v\} \in E$ rather than tuples.

The classic connectivity model is the so-called *unit disk graph* (UDG) [1]. The name "unit disk graph" stems from the area of *computational geometry*; it is a special case of the so-called *intersection graph*. In this model, nodes having omnidirectional radio antennas—that is, antennas with constant gain in all directions—are assumed to be deployed in a planar, unobstructed environment. Two nodes are adjacent if and only if they are within each other's transmission range (which is normalized to 1).

Model 4.2.1 (Unit Disk Graph (UDG)). Let $V \subset \mathbb{R}^2$ be a set of nodes in the two-dimensional Euclidean plane. The Euclidean graph $G = (V, E)$ is called unit disk graph if any two nodes are adjacent if and only if their Euclidean distance is at most 1. That is, for arbitrary $u, v \in V$, it holds that $\{u, v\} \in E \Leftrightarrow |u, v| \leq 1$. Figure 4.1 depicts an example of a UDG.

The UDG model is idealistic: In reality, radios are not omnidirectional, and even small obstacles such as plants can change connectivity. Therefore, some researchers

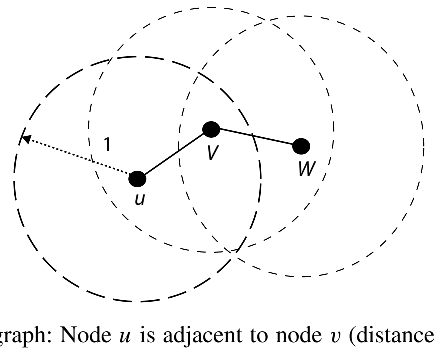

Figure 4.1. Unit disk graph: Node u is adjacent to node v (distance ≤ 1) but not to node w (distance > 1).

have proposed to study the other extreme and model the sensor network as a *general graph*; that is, each node can be adjacent to every other node.

Model 4.2.2 (General Graph (GG)). The connectivity graph is a general undirected graph G.

While a UDG is too optimistic, the GG is often too pessimistic, because the connectivity of most networks is not arbitrary but obeys certain geometric constraints. Still, in some application scenarios it might be accurate to operate either on the UDG or on the GG. Indeed, there are algorithms developed for the UDG which also perform well in more general models. Moreover, some algorithms designed for the GG are currently also the most efficient ones for UDGs (e.g., reference 2).

The research community has searched for connectivity models between the two extremes UDG and GG. For example, the *quasi unit disk graph model* (QUDG) [3, 4] is a generalization of the UDG that takes imperfections into account as they may arise from non-omnidirectional antennas or small obstacles. These QUDGs are related to so-called *civilized graphs*. The interested reader can find more information in reference 5.

Model 4.2.3 (Quasi Unit Disk Graph (QUDG)). The nodes are in arbitrary positions in $\mathbb{R}^2$. All pairs of nodes with Euclidean distance at most ρ for some given $\rho \in (0, 1]$ are adjacent. Pairs with a distance larger than 1 are never in each other's transmission range. Finally, pairs with a distance between ρ and 1 may or may not be neighboring. An example is shown in Figure 4.2.

Note that, for $\rho = 1$, a QUDG is a UDG, and therefore the following theorem holds.

Theorem 4.2.1. A UDG is a special case of a QUDG.

The QUDG model itself can be extended in several ways.

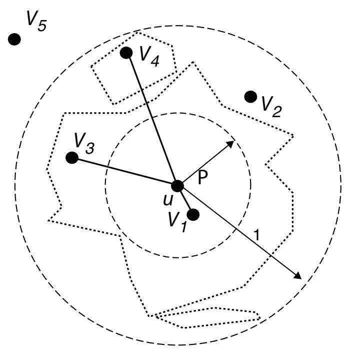

Figure 4.2. Quasi unit disk graph from the perspective of node u: Node u is always adjacent to node v_1 ($d(u, v_1) \leq \rho$) but never to v_5 ($d(u, v_5) > 1$). All other nodes may or may not be in u's transmission range. In this example, node u is adjacent to v_3 and v_4 but not to v_2.

Model 4.2.4 (QUDG Variations). The QUDG as presented in Model 4.2.3 does not specify precisely what happens if the distance is between ρ and 1. There are several options. For example, one could imagine an *adversary* choosing for each node pair whether they are in each other's transmission range or not. Alternatively, there may be a certain *success probability* of being adjacent: The corresponding probability distribution could depend on the time and/or distance [6]. For example, the QUDG could be used to study *Rayleigh fading*; that is, the radio signal intensity could vary according to a Rayleigh distributed random variable. Also, a probabilistic on/off model is reasonable, where in each round a link's state changes from good to bad and vice versa with a given probability.

Measurement studies suggest that in an unobstructed environment, and with many nodes available, $1/\rho$ is modeled as a small constant [7]. Interestingly, many algorithms can be transferred from the UDG to the QUDG at an additional cost of $1/\rho^2$ [4]. Note that while for $\rho \approx .5$ this factor is bearable, the algorithms are two orders of magnitude worse if $\rho \approx .1$. While the QUDG can be attractive to model nodes deployed in fields with few obstacles, it does not make sense for inner-city or in-building networks where obstructions cannot be ignored: Since a node may be able to communicate with another node which is dozens of meters away, but not with a third node being just around the corner, ρ would be close to 0.

However, even in such heterogeneous environments, the connectivity graph is still far from being a general graph. Although nodes that are close but on different sides of a wall may not be able to communicate, a node is typically highly connected to the nodes which are in the same room, and thus many neighbors of a node are direct neighbors themselves. In other words, even in regions with many obstacles, the total number of neighbors of a node which are not adjacent is likely to be small. This observation has motivated Model 4.2.6, see reference 8 for more details.

Model 4.2.5 (Bounded Independence Graph (BIG)). Let $\Upsilon^r(u)$ denote the set of independent nodes that are at most r hops away from node u (i.e., nodes of u's r-neighborhood) in the connectivity graph G. Thus, a set $\mathcal{S} \subseteqq V$ of nodes is called independent if all nodes in the set are pairwise not adjacent; that is, for all $u, v \in \mathcal{S}$, it holds that $\{u, v\} \notin E$. Graph G has bounded independence if and only if for all nodes $u \in G$, $|\Upsilon^r(u)| = O(\text{poly}(r))$ (typically $|\Upsilon^r(u)| \in O(r^c)$ for a small constant $c \geq 2$).

The BIG model reflects reality quite well and is appropriate in many situations. Figure 4.3 shows a sample scenario with a wall; in contrast to UDG and QUDG, the BIG model captures this situation well.

Since the number of independent neighbors in a disk of radius r of a UDG is at most $O(r^2)$, we have the following fact. The proof is simple (and similar to the upcoming proof of Theorem 4.2.13) and left to the reader as an exercise.

Theorem 4.2.2. The UDG model is a special case of the BIG model. Similarly, if ρ is constant, also a QUDG is a BIG.

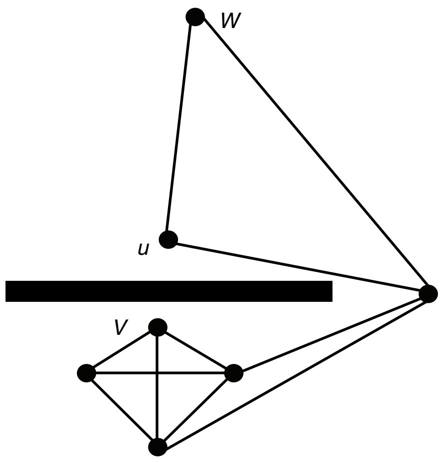

Figure 4.3. Nodes u and v are separated by a wall. Nodes on the same side of the wall are completely connected. However, due to the wall, although u can reach a distant node w, it cannot hear the close node v. Such situations can be modeled by the BIG but not by the UDG or the QUDG.

Observe that many models described so far can be generalized. For instance, the UDG and QUDG models can be studied in three dimensions rather than in the plane, or using different distance functions (*norms*). For more detailed information on the concept of norms, the reader may want to consult any introductory book on linear algebra.

Model 4.2.6 (Generalized (Q)UDG). One extension of the UDG and QUDG models is to consider nodes in $\mathbb{R}^3$. Moreover, distances between nodes could be modeled using the Manhattan norm ($\mathbb{L}_1$ norm). In the Manhattan norm, the distance between two points $u = (x_1, y_1)$ and $v = (x_2, y_2)$ in the plane is given by $d(u, v) = |x_2 - x_1| + |y_2 - y_1|$, while in the Euclidean norm ($\mathbb{L}_2$ norm), the distance is $d(u, v) = \sqrt{|x_2 - x_1|^2 + |y_2 - y_1|^2}$. Alternatively, also the *maximum norm* ($\mathbb{L}_\infty$ norm) is popular, where $d(u, v) = \max |x_2 - x_1|, |y_2 - y_1|$.

The UDG model has also been extended to more general *metric spaces*; for example, in reference 9, it was extended to *doubling metrics* [10]. First, recall that Q2 a metric space defines distances between all pairs of nodes while guaranteeing *non-negativity*, *identity of indiscernibles*, *symmetry*, and *triangle inequality*. A doubling metric is simply a metric space with some additional constraints which are described next.

Model 4.2.7 (Unit Ball Graph (UBG)). A doubling metric space is defined as follows: For a node u, let the ball $B_u(r)$ denote the set of all nodes at a distance at most r from u. It holds, for all nodes u and all $r \geq 0$, that the ball $B_u(r)$ can be covered

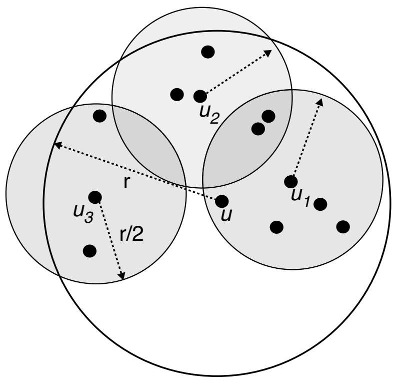

Figure 4.4. The Euclidean plane forms a doubling metric. In this example, the nodes are distributed in $\mathbb{R}^2$, and three balls of radius $r/2$ are sufficient to cover all nodes in $B_u(r)$; that is, $B_u(r) = B_{u_1}(r/2) \cup B_{u_2}(r/2) \cup B_{u_3}(r/2)$.

by a constant number of balls of radius $r/2$; that is, $B_u(r) \subseteq \bigcup_{i=1\ldots c} B_{u_i}(r/2)$, where u_i are arbitrary nodes and c is a (usually small) constant. In the UBG model, nodes are assumed to form a doubling metric space. Two nodes u and v with $d(u, v) \leq 1$ are adjacent, whereas all other nodes are not.

The proof of the following theorem is left to the reader as an exercise.

Theorem 4.2.3. Nodes in a two-dimensional Euclidean plane (i.e., the metric space is given by the Euclidean distances) form a doubling metric. A general graph, however, does not.

Figure 4.4 shows an example for the Euclidean plane. In this setting, three balls of radius $r/2$ are enough to cover all nodes in the ball of radius r around node u. To see why a general graph may not form a doubling metric, consider a graph where all nodes have distance 1 to all other nodes. Observe that it is possible to model a UDG with a UBG by using the Euclidean distances of the UDG and connecting those node pairs which have distance at most 1. Moreover, even a QUDG can be modeled with a UBG. We have the following results.

Theorem 4.2.4. A UDG is a UBG.

Theorem 4.2.5. An undirected QUDG with constant ρ is a UBG.

Proof. The idea of the proof is as follows: First, we transform all distances in the QUDG. We then show that during this transformation, all edges are maintained; that is, the resulting graph is isomorphic to the QUDG. Moreover, it can be shown that after the transformation, the graph also fulfills the requirements of a doubling metric space.

We transform the distances between all pairs of nodes (u, v) in the QUDG as follows. Let $d_Q(u, v)$ denote the distance from node u to node v in the QUDG, and let $d_B(u, v)$ be the transformed distance in the UBG. Moreover, let $\epsilon > 0$ be an arbitrary small number.

$$d_B(u, v) := \begin{cases} d_Q(u, v)/\rho & \text{if} \quad d_Q(u, v) \leq \rho \\ 1 & \text{if} \quad \rho < d_Q(u, v) \leq 1 \text{ and } v \text{ is adjacent to } u \\ 1 + \epsilon & \text{if} \quad \rho < d_Q(u, v) \leq 1 \text{ and } v \text{ is \textit{not} adjacent to } u \\ d_Q(u, v) & \text{if} \quad d_Q(u, v) > 1 \end{cases}$$

Observe that by this transformation, pairs of nodes that are adjacent in the QUDG are assigned distances of at most 1 and are therefore also adjacent in the UBG. Similarly, nodes that are not adjacent in the QUDG have a distance larger than 1 are therefore not neighboring in the UBG either. Also observe that the transformation increases the distance between two nodes by less than a constant factor of $\mu := (1 + \epsilon)/\rho$, but it never decreases any distances. It remains to show that after the transformation, the nodes indeed form a doubling metric space.

In order to form a metric space, the distances between the nodes are to fulfill the following properties: (1) nonnegativity, (2) identity of indiscernibles, (3) symmetry, and (4) triangle inequality. The nonnegativity and the identity of indiscernibles criteria are met trivially. The symmetry criterion, however, might not hold, because the adjacency relation can be directed in a QUDG. Therefore, in the following, we consider undirected QUDGs only. Hence, since our distance transformation maintains symmetry, Property 3 holds as well. It remains to discuss the triangle inequality.

Consider two arbitrary nodes u and v. Since in the QUDG, all distances are Euclidean, it holds that

$$\forall w : d_Q(u, v) \leq d_Q(u, w) + d_Q(w, v) \tag{4.1}$$

Let us now look at the following three cases in turn: (i) $d_Q(u, v) \leq \rho$, (ii) $\rho < d_Q(u, v) \leq 1$, and (iii) $1 < d_Q(u, v)$. In Case i, no node w with distance larger than ρ from any of the two nodes u and v can challenge the triangle inequality. For all other nodes w, however, it holds that $d_B(u, v) = d_Q(u, v)/\rho \leq (d_Q(u, w) + d_Q(w, v))/\rho = d_B(u, w) + d_B(w, v)$. Here, the equalities hold by the definition of the transformation function and the inequality is due to Eq. (4.1). Next, we tackle Case ii. Again, only nodes w with $d_Q(u, w) \leq \rho$ and $d_Q(w, v) \leq \rho$ can challenge the inequality. However, we know that $d_Q(u, v) > \rho$, and hence Eq. (4.1) yields $d_B(u, w) + d_B(w, v) = d_Q(u, w)/\rho + d_Q(w, v)/\rho > 1$. Finally, the triangle inequality also holds in Case iii, because the distance between u and v in the UBG is the same as in the QUDG, and our transformation never decreases any distances.

We conclude the proof by showing that the metric space has a constant doubling dimension. Recall that all distances are only stretched by a constant factor between 1 and μ in our transformation. Therefore, for all nodes u and arbitrary radii

r, $B_u^{QUDG}(r/\mu) \subseteq B_u^{UBG}(r)$. Thus, at most a constant factor of $O(\log \mu)$ times, more balls are needed for the UBG than for the QUDG (the Euclidean plane) in the worst case, and the claim follows.

The UBG itself has a polynomially bounded independence and is therefore a BIG.

Theorem 4.2.6. A UBG is a BIG.

Proof. Fix a node u. We have to prove that the total number of independent nodes in $B_u(r)$ grows polynomially in r. Observe that, due to the triangle inequality, in $B_u(1/2)$ there is at most one independent node. Thus, by the definition of a doubling metric, there are at most c independent nodes in $B_u(1)$, at most c^2 in $B_u(2)$, c^3 in $B_u(4)$, and so on. Generally, there are at most $c^{\log r+1}$ independent nodes in $B_u(r)$. Since $c^{\log r} \in O(r^c)$, the claim follows.

To conclude, we present two additional modeling aspects with which connectivity models are occasionally extended. The first aspect concerns the sensor nodes' antennas.

Model 4.2.8 (Antennas). Besides omnidirectional antennas, there is a wide range of more sophisticated antenna models. For example, a node can have a directional radio antenna with more gain in certain directions.

Finally, as mentioned in the discussion of the QUDG, links are not always reliable: Links may be up and down—for example, according to a probabilistic process.

Model 4.2.9 (Link Failures). Any graph-based model can be enhanced with probabilistic links.

4.3 INTERFERENCE ISSUES IN WIRELESS SENSOR NETWORKS

In wireless networks, the communication medium is shared and transmissions are exposed to interference. Concretely, a node u may not be able to correctly receive a message of an adjacent node v because there is a concurrent transmission nearby. In some sense, an interference model explains how concurrent transmissions block each other. Interference is a difficult phenomenon, with many hard-to-capture characteristics. A signal might, for example, interfere with itself due to *multipath propagation* (e.g., a direct path canceling with a longer path reflecting on an object). A discussion of these effects is beyond the scope of this overview chapter. Instead we look at models that capture reality from a worst-case perspective. The mother of all interference models is the so-called *physical* or *SINR model* [11–13], which is widely accepted by information theorists. In this model, the successful reception of a message depends on the received signal strength, the ambient noise level, and the interference caused by simultaneously transmitting nodes.

Model 4.3.1 (Signal-to-Interference Plus Noise (SINR)). Let P_r be the signal power received by a node v_r and let I_r denote the amount of interference generated by other nodes. Finally, let N be the ambient noise power level. Then, a node v_r receives a transmission if and only if $\frac{P_r}{N+I_r} \geq \beta$. Thus, β is a small constant (depending on the hardware) and denotes the minimum signal to interference ratio that is required for a message to be successfully received. The value of the received signal power P_r is a decreasing function of the distance $d(v_s, v_r)$ between transmitter v_s and receiver v_r. More specifically, the received signal power is modeled as decaying with distance $d(v_s, v_r)$ as $\frac{1}{d(v_s,v_r)^\alpha}$. The so-called *path-loss exponent* α is a constant between 2 and 6 and depends on external conditions of the medium, as well as on the exact sender–receiver distance. [1] Let P_i be the transmission power level of node v_i. A message transmitted from a node $v_s \in V$ is successfully received by a node v_r if

$$\frac{\dfrac{P_s}{d(v_s, v_r)^\alpha}}{N + \sum_{v_i \in V \setminus \{v_s\}} \dfrac{P_i}{d(v_i, v_r)^\alpha}} \geq \beta$$

In other words, in the SINR model, a node correctly receives a transmission if the received signal power—which depends on the sending power and the distance between sender and receiver—is large enough compared to the signal power of concurrent (interfering) transmissions and the ambient noise level. Sometimes a variation of this SINR model is used in literature. It has an additional requirement: For a successful reception, the received signal power must exceed a minimal threshold θ, that is, $P_r \geq \theta$. In many situations, such a threshold can also be incorporated implicitly by the ambient noise power level N. Moreover, researchers have also studied a *probabilistic* SINR model [14], where the gain of an antenna is described by a *Gaussian distribution*—independently of the distance! Apart from the interference term, and if all nodes send with the same transmission power level, the connectivity model of SINR is exactly the UDG, with path-loss exponent α and minimum ratio β such that the maximum distance for receiving a signal is 1. Hence, the SINR model can be extended similarly to the UDG model. Now, observe that the SINR model does not specify the signal power P_s used by a sender v_s to transmit data to the receiver v_r. Three models are common:

Model 4.3.2 (Power Control). CONST: All nodes use the same *constant* transmission power. DIST: The power level depends on the *distance d* between sender and receiver. Concretely, the transmission power is given by $c \cdot d^\alpha$ for some $\alpha \geq 2$ and some constant $c > 0$. GEN: A general (or *arbitrary*) power level is assumed at the

[1] In free space, α roughly equals 2. In the so-called *two-ray ground model*, it is assumed that there are two paths of the electromagnetic wave: a direct one and a ground reflected signal path; to describe this situation, $\alpha = 4$ is used. Finally, note that ever since *Marconi*'s first experiments, time has been devoted to explain radio propagation phenomena, and there is a plethora of other proposals. For example, for small urban cells, a photon propagation model has been suggested implying an *exponentially* growing path loss.

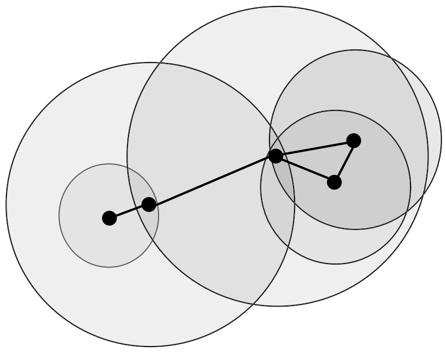

Figure 4.5. Sample network with heterogeneous transmission ranges. For instance, the node on the far left saves energy and reduces interference by using only a small power level.

sender, which may change over time. Figure 4.5 depicts a network where each node has a different power level.

Although the SINR model incorporates many important physical properties, it has not received an appropriate amount of attention from the algorithms community [12]. This can be partially explained by the fact that the SINR model is complicated. For instance, a lot of far-away transmissions sum up, and may interfere with a close-by sender-receiver pair. In practice, however, these far-away transmissions often only contribute to the ambient noise and need not be counted individually. Twiddling the knobs of the model a bit more, we might not sum up all interfering transmissions, but simply look at the worst—or, in the case of a CONST model; *closest*—disturbance: A node receives a transmission if and only if the closest simultaneously transmitting node is far enough.

Model 4.3.3 (Interfering Transmissions). SUM: All interfering transmissions are taken into account. ONE: Only the worst (or closest) interfering transmission matters. NULL: Pure connectivity models which do not consider interference aspects (cf. Section 4.2).

ONE models are quite popular because of their simplicity. The UDI—an interference-aware version of the UDG—is a prominent example (cf. Model 4.3.4). Observe that, because of the constant transmission power, the power control type of UDI is CONST. Figure 4.6 shows an example.

Model 4.3.4 (UDG with Distance Interference (UDI)). Nodes are situated arbitrarily in the plane. Two nodes can communicate directly if and only if their Euclidean distance is at most 1, and if the receiver is not disturbed by a third node with Euclidean distance less or equal a constant $R \geq 1$.

Often the constant R of the UDI model is approximated in such a way that interference can be reduced to a parameter of the UDG. For instance, some MAC

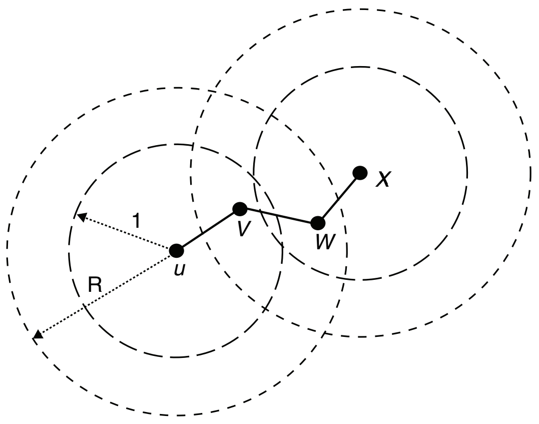

Figure 4.6. The UDI model has two radii: a transmission radius (length 1) and an interference radius (length $R \geq 1$). In this example, node v is not able to receive a transmission from node u if node x concurrently transmits data to node w—even though v is not adjacent to x.

protocols (e.g., coloring algorithms [15]) have been proposed to reduce interference by ensuring a certain hop distance between two senders. Concretely, it is assumed that only the k-neighborhood of a receiver u can interfere with u. Clearly, this is a stark simplification since in a UDG a $(k + 1)$-neighbor can be close to the receiver (see Figure 4.7).

Model 4.3.5 (UDG with Hop Interference (UHI)). Nodes are located at arbitrary positions in $\mathbb{R}^2$. Two nodes are adjacent if and only if their Euclidean distance is at

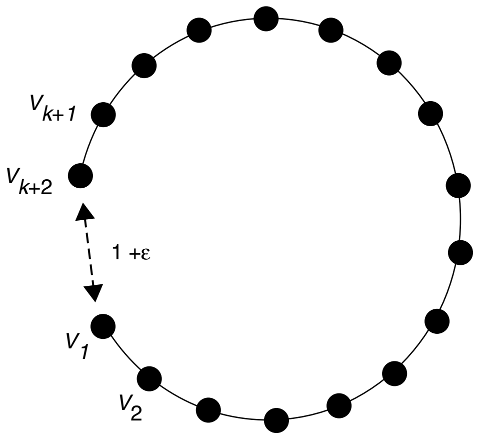

Figure 4.7. Example where UHI fails: Nodes v_1 and v_{k+2} are separated by a path of $k + 1$ hops, but are close (distance $1 + \epsilon$). Thus, concurrent transmissions of nodes v_2 and v_{k+2} may interfere at v_1 in spite of their large hop distance.

most 1. Two nodes can communicate directly if and only if they are adjacent, and if there is no concurrent sender in the k-hop neighborhood of the receiver (in the UDG).

Observe that while the UHI model—for every k—sometimes overlooks interference terms which the UDI would take into account, the contrary does not hold.

Theorem 4.3.1. By choosing $R = k$, and since a hop has at most length 1, the UDI model does not overlook any interference terms that UHI would have taken into account. The contrary does not hold (cf. Figure 4.7).

Like UDI and UHI, also the *protocol model* (PM) is of type ONE (Model 4.3.3). However, the senders in the PM model adapt their transmission power according to DIST (Model 4.3.2)—that is, depending on the distance between sender and receiver. Model 4.3.7 is a variation of the model introduced in reference 11.

Model 4.3.6 (Protocol Model (PM)). Let $u_1, u_2, ..., u_k$, be the set of nodes transmitting simultaneously to receivers $v_1, v_2, ..., v_k$, respectively. The transmission of u_i is successfully received by v_i if for all $j \neq i$, it holds that $d(u_j, v_i) > \lambda \cdot d(u_j, v_j)$, where $\lambda \geq 1$ is a given constant. That is, v_i must not fall into a "guard zone" around any sender u_j which is a factor $(1 + \lambda)$ larger than u_j's transmission range.

Many interference models distinguish between senders and receivers assuming that interference arises at senders and occurs at receivers. However, often receivers acknowledge messages and are therefore also senders. If the original messages are short (e.g., control messages), then the sender/receiver distinction may not make sense. By this observation, some models (e.g., reference 16) simply consider the interference of *undirected links*. Figure 4.8 depicts an example.

Model 4.3.7 (Direction). DIR: This class of interference models distinguishes between senders and receivers (interference disks around senders). UNDIR: Interference originates from undirected links (interference "pretzels" around links).

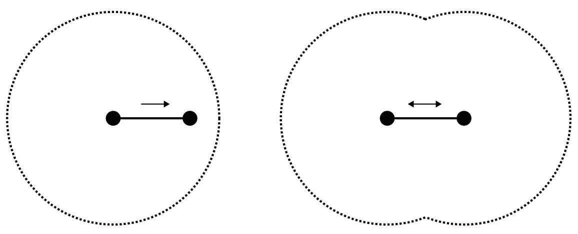

Figure 4.8. DIR vs. UNDIR: On the left, only the sender transmits data (interference disks around senders). On the right, there is no distinction between sender and receiver, and hence interference arises from the entire link ("pretzels" around links).

As in the case of connectivity models, the SINR, the UDI, and the UHI models can be extended with directional antennas and link failures, and hence Models 4.2.14 and 4.2.15 also apply here. Moreover, also the idea of quasi unit disk graphs (cf. Model 4.2.3) could be adopted. For example, the UDI can be "quasified" as follows: If two nodes are closer than a given threshold R_1, concurrent transmissions will always interfere; if the distance is larger than a second threshold R_2, there will be no interference. Finally, if the distance is between R_1 and R_2, transmissions may or may not interfere. However, these models are often too complicated to be handled algorithmically. It is sometimes simpler to study general *weighted* interference graphs instead. That is, similar to connectivity graphs, the interference model is based on *graphs*; however, the edges are now weighted. Formally, in a weighted interference graph $H = G(V, E, w)$, V represents the set of sensor nodes, E represents the set of edges, and $w : E \to \mathbb{R}^+$ is a function assigning a positive value to each edge. The weight denotes how large the interference between the corresponding nodes actually is. As in the SINR model, a transmission is received correctly if the ratio between received signal power and the amount (either the sum or the maximal interfering signal strength) of interfering traffic is smaller than a certain threshold.

Model 4.3.8 (General Weighted Graph (GWG)). A weighted interference graph H is given. A receiver v successfully receives a message from a sender u, if and only if the received signal strength (the weight of the link between u and v in H) divided by the total interference (the sum or the maximum of the weights of the links of concurrently transmitting nodes with a receiver v in H) is above the threshold given by the signal-to-interference-plus-noise ratio.

The general weighted graph model is quite pessimistic, because it allows for non-natural network topologies. Again—like in the BIG connectivity model—we need a weighted graph model that captures the geometric constraints without making too many simplifying assumptions. Again, one approach is to assume that the nodes form a *doubling metric* (cf. UBG model of Section 4.2).

Model 4.3.9 (Doubling Metric (DM)). The DM model assumes that the nodes form a doubling metric; that is, the set of nodes at a distance (which is now given by the weights of the edges) of at most r from a node u can be covered by a constant number of balls of radius $r/2$ around other nodes, for any r (cf. Model 4.2.9). Interference can be incorporated in various ways. For example, the amount of interference at a receiver u could depend on u's distance (in the doubling metric space) to the closest concurrently transmitting node (ONE model), or on the number of concurrent senders (SUM model).

As a final remark, note that so far we have only presented *binary interference models*: A message can be received either correctly or not at all. In practice, however, also the *transfer rate* at which messages can be transmitted can depend on interference: The larger the signal-to-noise ratio, the larger the available bandwidth. A WLAN 802.11, for example, exploits environments with less interference in order to transmit

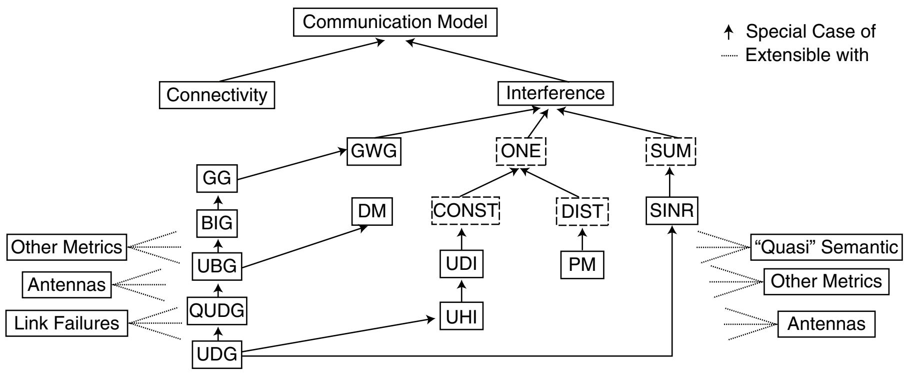

Figure 4.9. Overview of connectivity and interference models presented in Sections 4.2 and 4.3. The arrows show how the models are related.

more data per time unit. Of course, it would be possible to extend, for example, the DM model to capture this aspect as well. However, since these issues are beyond the scope of this chapter, we refer the reader to reference 17 for more details. To conclude, Figure 4.9 summarizes the connectivity and the interference models.

4.4 ALGORITHM DESIGN

The main purpose of deploying sensor networks is the collection physical data such as light intensity, sound, or temperature. In order to aggregate (e.g., compute the minimum temperature, or the average, etc.) the data that are stored at the individual nodes—and which are therefore distributed in space!— protocols or *algorithms* are needed specifying how these operations are performed. For example, due to the limited radio communication range, sensor nodes have to communicate (e.g., gather data) in a multihop manner with each other—that is, the messages have to be relayed by intermediate nodes—and hence a routing algorithm has to define which messages are to be forwarded via which other nodes.

Algorithms for sensor networks come in different flavors. In the following, we first describe the different types of algorithmic models appearing in literature today. We then discuss modeling aspects that may influence an algorithm's performance—for instance, what kind of identifiers nodes have, or how the nodes are distributed in space. Besides the classic evaluation criteria for algorithms—namely, *time complexity* and *space complexity*—algorithms for sensor networks pose additional optimization problems; for example, the number of messages that are sent should be small; or, in order to maximize the lifetime of the network, the nodes' energy consumption must be minimized. In order to facilitate a better understanding of the different algorithm types presented in the upcoming paragraphs, we will consider a sample problem: the computation of *dominating sets*.

Definition 4.4.1 (Dominating Set Problem (DS)).. The computation of a dominating set (DS) is a fundamental operation in sensor networks. For instance, such a set

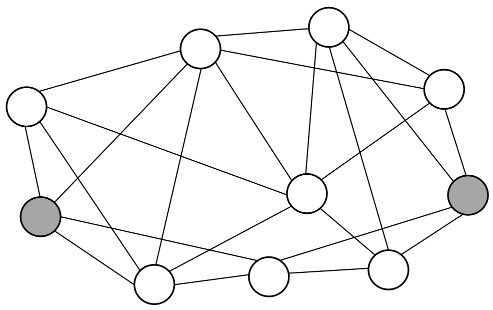

Figure 4.10. A minimum dominating set with two dominators.

can be used to build node clusters. Moreover, it may serve as a basis for constructing backbone networks that typically form a connected DS. A dominating set $D \subseteq V$ of a (undirected) network graph $G = (V, E)$ is a set of nodes such that for all nodes $u \in V$ it holds that either u is in the dominating set itself (i.e., $u \in D$), or u is adjacent to a node v in D (i.e., $\{u, v\} \in E \wedge v \in D$). It is often desirable to have small dominating sets. The minimum dominating set (MDS) is defined as the dominating set that minimizes the number of dominators $|D|$. An example of an MDS is illustrated in Figure 4.10. It can be shown that the MDS problem is NP-hard on general graphs and that a logarithmic approximation is asymptotically optimal unless $P \approx NP$ [18]. For simpler connectivity graphs such as the UDG graph, the approximation complexity of the problem may be better; for example, there is a *polynomial time approximation scheme* (PTAS) for UDG graphs!

The first category of algorithms we present here is similar to the classic (graph) algorithms appearing in the field of theoretical computer science or applied mathematics. These *global* algorithms can operate directly on the entire network or graph and can have complete information about the state of the system. For example, a system designer planning a fixed sensor network can apply a global algorithm to determine the optimal positions of the nodes in a given observation area.

Model 4.4.2 (Global Algorithms). A global algorithm can operate directly on the entire network.

Kruskal's algorithm for computing a minimum spanning tree [19] is an example of a global algorithm: The algorithm receives the entire graph as input and can sort the edges according to their weights. Kruskal's algorithm thus has a complete visibility of the entire graph and can perform arbitrary operations on it. No messages have to be sent between nodes.

Example 4.4.3. Let us tackle our dominating set problem! When faced with the task of designing an algorithm for a certain problem, it is often a good idea to start by studying greedy algorithms—that is, algorithms that in every execution step "greedily" do the currently most promising thing. Interestingly, a greedy algorithm is often optimal (see also Matroid theory [19]). So how can we greedily compute an MDS? A straightforward approach is the following (see Algorithm 1): First, we initialize the set of dominators with the empty set, that is, $D := \{\}$. We will call nodes in D black ("dominators"), nodes that are covered by nodes in D gray ("dominated nodes"), and all uncovered nodes *white*. Let $w(v)$ be the number of white nodes adjacent to v, including v itself. Then, in every step, we iterate over all nodes v (global operation!) and compute the number $w(v)$ of v's white neighbors, remembering the node x having the largest number. At the end of each step, we add node x to D. That is, we choose the node to become a dominator that covers the most new nodes, greedily reducing the number of the remaining nodes as much as possible. Obviously, the resulting dominators indeed form a DS. Moreover, it can be proven that the number of dominators is at most a logarithmic factor larger than in the optimal case. This simple approximation algorithm is therefore asymptotically optimal unless $P \approx NP$!

ALGORITHM 1. Global and Greedy MDS Algorithm

```
1: $D := \{\}$;
2: while $\exists$ white nodes do
3:   $x := \{x | w(x) = \max_v\{w(v)\}\}$;
4:   $D := D \cup x$;
5: od;
```

However, unlike global algorithms, most algorithms for sensor networks are not executed by a central designer, but rather *by the sensor nodes* themselves, for example, during the system's operation. A node a priori only knows its own state. In order to learn more about the other nodes in the network, it is bound to communicate with its neighbors *by exchanging messages*. Typically, when a node receives a message, it performs some computation, and—depending on the computation's results—sends a new message to its neighbors. By this collaboration of the nodes, global operations such as (multihop) routing between two nodes can be performed. Since the activity is distributed among the nodes, these algorithms are called *distributed algorithms* [20].

Model 4.4.4 (Distributed Algorithms). In a distributed algorithm, every sensor node runs its own algorithm. A priori, a node only knows the state of itself. In order to learn more about the rest of the network, nodes repeatedly exchange messages with adjacent nodes.

Thus, unlike in global algorithms, distributed algorithms do not see the entire graph at the beginning. If more information about the neighborhood is needed, adjacent nodes have to exchange messages. Distributed algorithms raise many interesting questions. For example: What can be computed in a distributed fashion, and what not? In contrast to global algorithms, nodes have to be coordinated somehow, and—as all nodes execute the same code—symmetries have to be broken. Besides an algorithm's correctness, execution time to perform the task (*time complexity)*, and memory requirements at the nodes (*space complexity*), a new criterion becomes important, namely *message complexity*: Since distributed algorithms rely on message passing and since sending and receiving messages is an expensive operation (e.g., energy consumption, queueing delay, congestion, etc.), a distributed algorithm should minimize the total number of messages sent.

Example 4.4.5. In the following, we will see how dominating sets can be computed with distributed algorithms. Consider the following idea: Each sensor node broadcasts its own identifier plus the identifiers of all its neighbors to all other nodes in the network. Hence, each node gets a picture of the entire connectivity graph. Then, the node having the largest identifier computes the MDS using the global algorithm described in Algorithm 1, and it broadcasts the corresponding solution back to all nodes. Given this solution, each node can then decide whether it has to join the dominating set or not. Note that this algorithm is indeed distributed and yields small dominating sets: The size of the sets are again asymptotically optimal, unless $P \approx NP$. However, due to the broadcast operation, the message complexity is huge. What is more, the algorithm does not scale to a large number of sensor nodes.

Many global algorithms can be turned into distributed algorithms by just collecting the entire graph at each node and then computing the results locally. Of course, this is inefficient and inappropriate in practice. Therefore, it is often better to study a more restricted class of distributed algorithms, namely localized algorithms [21].

Model 4.4.6 (Localized Algorithms). A localized algorithm is a special case of a distributed algorithm. At the beginning, a node has only information about its own state. In order to learn more about the rest of the network, messages have to be exchanged. In a k-localized algorithm, for some constant k, each node is allowed to communicate at most k times with its neighbors. However, a node can decide to retard its right to communicate; for example, a node can wait to send messages until all its neighbors having larger identifiers have reached a certain state of their execution.

Localized algorithms are desirable in the sense that they only transmit a small number of messages. Unfortunately, however, localized algorithms can be slow: A node u might have to wait for a neighbor v to transmit all its messages, while node v in turn has to wait for its neighbor w, and so on. As a matter of fact, there can be a *linear chain of causality*, with only one node being active at any time. This yields a worst-case execution time of $\Theta(n)$, where n is the number of nodes.

Example 4.4.7. In order to compute the dominating sets of our sample problem in a localized manner, a simple algorithm can be applied (cf. Algorithm 2). Each node v waits until all its neighbors having a larger degree (or, in the case of the same degree; a larger identifier) than v have decided whether to join the dominating set or not. Then, if one of these nodes is a dominator, v decides not to join the dominating set. Otherwise, v becomes a dominator. Thus, each node has to communicate at most twice with its neighbors: once to find out their degree and once to tell them about its decision. This

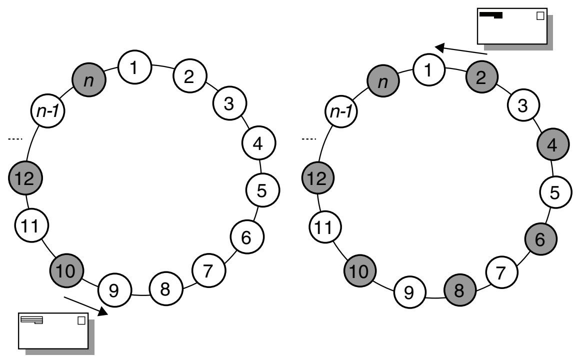

Figure 4.11. Localized algorithms can have large execution times!

localized algorithm has therefore a low message complexity. However, the execution time of this algorithm can be large. To see this, consider a cycle of nodes arranged according to their identifiers as depicted in Figure 4.11. Since all nodes have the same degree, the first node to become a dominator is node n. Only then, node $n-1$ can make its decision: It does not join the dominating set. After that, node $n-2$ decides to join the network, and so on. It's only after a linear waiting time of $O(n)$ time steps that node 1 eventually can make its decision and terminate the algorithm!

ALGORITHM 2. Localized MDS Algorithm

```
 1: (* Code executed by node $v$ *)
 2: send degree and ID to all neighbors;
 3: receive messages from neighbors;
 4: while (∃ undecided neighbor $w$ with $prio(w) > prio(v)$) do
 5:   wait();
 6: od;
 7: (* Decision *)
 8: if (∃ dominator in neighborhood) then
 9:   $D := D$;
10:   send "I am dominated!" to neighbors;
11: else
12:   $D := D \cup v$;
13:   send "I am a dominator!" to neighbors;
14: fi;
```

Researchers have proposed to study yet another kind of distributed algorithm that overcomes the performance problems of localized algorithms, always terminating after a constant number of communication rounds [2].

Model 4.4.8 (Local Algorithms). Again, at the beginning, each node only knows its own state. In a k-local algorithm, for some constant k, each node can communicate at most k times with its neighbors. However, in contrast to k-localized algorithms, nodes cannot delay their decisions. In particular, all nodes process k synchronized phases,

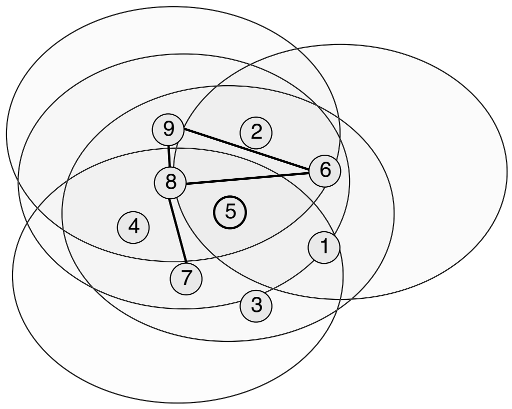

Figure 4.12. Illustration of local dominating set algorithm: Since all nodes with larger IDs are connected to each other and cover all other neighbors of node 5, node 5 does not join the dominating set—regardless of the decisions of other nodes.

and a node's operations in phase i may only depend on the information received during phases 1 to $i - 1$. The most efficient local algorithms are often randomized [22, 23], that is, the number of rounds k can vary.

Observe that in a k-local algorithm, nodes can only gather information about nodes in their k-neighborhood. In some local algorithms [22], the algorithm designer can choose an arbitrarily small constant k (at the cost of a lesser approximation ratio). This makes local algorithms particularly suited in scenarios where the nodes' environment changes frequently, because they are able to constantly adapt to the new circumstances.

Example 4.4.9. A dominating set can also be computed with a local algorithm: Each node u asks its higher-priority neighbors (with respect to degrees and identifiers) about their neighbors. If these higher-priority neighbors are connected and cover all of u's neighbors, then u does not join the dominating set and otherwise becomes a dominator. An example is illustrated in Figure 4.12. It can be shown that this algorithm even results in a connected dominating set—that is, a dominating set where any two dominators are connected by a path that only consists of other dominators. Observe that the algorithm is indeed "wait-free" or local, because a node can make its decisions only based on the identifiers of its neighbors and independently of the neighbors' decisions. Two communication rounds are sufficient. From this point of view, this local algorithm looks very appealing. Unfortunately, however, in the worst case, its approximation ratio of the optimal solution is as bad as $\Theta(\sqrt{n})$ already for simple connectivity graphs such as the UDG.[2] This indicates the existence of a challenging tradeoff: The smaller the "horizon" of a local algorithm, the more difficult it is to find good approximations of the optimal (global) solutions. In other words, there seems to be price of being

[2] For random UDGs, the performance is better: The algorithm achieves a constant approximation!

near-sighted: Similarly to online algorithms that cannot foresee the future and have to be competitive to an optimal offline algorithm, local algorithms have a restricted view of their neighborhood and are measured against the performance of a global algorithm.

Note that due to the synchronous phases, local algorithms may make greater demands on the media access sublayer than localized algorithms. In particular, in unreliable wireless networks it seems to be costly to implement a media access control scheme that allows for synchronous rounds, because messages will be lost due to interference (conflicting concurrent transmissions) or mobility (even if the nodes themselves are not mobile, the environment is typically dynamic, temporarily enabling/disabling links). A powerful concept for coping with failures is *self-stabilization* [24]. Fortunately, using a simple trick [25], every local algorithm is immediately self-stabilizing. The trick works as follows (Section 4 of reference 25): Every node keeps a log of every state transition it has taken until its current state; generally, this boils down to memorizing the local variables of each step of the main loop. If each node constantly sends its current log to all neighbor nodes, each node can check and correct every transition it has made in the past. Assuming that all inputs are correct (variable initialization and random seeds are stored in the imperishable program memory, and sensor information can be rechecked), every fault due to memory or message corruption will be detected and corrected. For details we refer to reference 25. Turning a k-local algorithm into a self-stabilizing algorithm with reference 25 blows up messages by a factor k (in the worst case); on the other hand, we immediately get an algorithm that works on a sensor network as the hardest wireless problems (messages lost due to interference and mobility) are covered by the self-stabilization model. Also, in the case of an error (such as a lost message), only the k-neighborhood of a node is affected.[3]

Having defined the most common types of algorithm, we now look at some algorithmic aspects in more detail. As mentioned, the message complexity—the total number of messages sent by an algorithm—is a main evaluation criterion of distributed algorithms. Because the number of messages typically depends on the amount of information that can be stored in a message, a model must specify the messages' sizes. A most popular model limits the message size to $O(\log n)$ bits, where n is the total number of nodes in the system. Hence, a message can store only a constant number of node identifiers (e.g., the source and destination address of a routing packet). Moreover, it is often assumed that if a node u sends a message to a neighbor v, all other neighbors of u will also receive the message (*broadcast model*). However, sometimes—for example, for lower bound proofs [26]—models are considered where the message size is *unbounded*, and where nodes can communicate with their neighbors individually (*message-passing model*). Algorithmic models also differ in their assumptions about how nodes can access the wireless medium. The concrete MAC, however, can

[3] In principle, localized algorithms can also benefit from reference 25; however, errors are not restricted to a k-neighborhood but may propagate through the entire network, resulting in a troublesome *butterfly effect*.

influence the number of retransmissions and hence an algorithm's performance. Moreover, an algorithm may be able to coordinate the medium access itself.

Model 4.4.10 (Medium Access). Some researchers assume an ideal medium access mechanism [27] where interference is impossible and where messages will always be broadcast instantaneously to all neighbors (cf. models of Section 4.2). In addition, adversarial models are used where an adversary schedules transmissions. Of course, this model only makes sense if the adversary is restricted appropriately—that is, if there are fairness guarantees. For example, the adversary might have to schedule each node at least once every $\Theta(n)$ rounds. One could also imagine an adversary that delivers a message only to a subset of a node's neighbors, because the other neighbors experience collisions. Finally, completely unstructured radio networks [28] can be considered where the algorithm designer has to implement her own medium access scheme from scratch. These models can further be classified in terms of whether collisions can be detected by a receiver or not.

As mentioned, a main objective of sensor networks is to collect physical data distributed over a given region. To achieve this, typically one or more nodes observe different sub-areas. Knowledge of the nodes' distribution, however, can be important for an algorithm designer. In a scenario where the nodes are dropped from an airplane, one might expect that the nodes are roughly randomly distributed when they reach ground.

Model 4.4.11 (Random Node Distribution). The simplest—and quite common—way to model sensor networks is to assume a UDG in combination with a *uniform node distribution* in the two-dimensional Euclidean plane. However, inspired by percolation theory, also *Poisson models* have been proposed [29]; thus, the positions of the nodes are distributed in $\mathbb{R}^2$ according to a homogeneous Poisson point process of constant density λ per unit area.

While these random models may be fine to prove the performance of an algorithm, for correctness and robustness issues, a more pessimistic model should be preferred—for example, a worst-case distribution.

Model 4.4.12 (Worst-Case Node Distribution). Nodes are distributed arbitrarily in the space given by the underlying graph (e.g., Euclidean plane, general graph, etc.).

Of course, there are again many models that lie between the two extremes. For example, random distributions with a density parameter varying over space could be considered: One can imagine that there are several nodes per square meter in areas that are "interesting" to observe, whereas in other "routing only" areas the nodes are hundreds of meters apart. Finally, speaking of node distributions, there are also models that do not allow nodes to be arbitrarily close or even assume the same position; for instance, there is such an assumption in the $\Omega(1)$ *model* [30] or in so-called *civilized graphs* [5]. Related to the distribution of nodes in space is also the issue of the distribution of node *identifiers*. Because many algorithms are based on node IDs, their performance can depend on how IDs are distributed among the nodes (and thus also in space).

Model 4.4.13 (Node Identifiers). Typically, nodes can be assumed to have unique identifiers. IDs could, for example, be generated during deployment using a random number generator. Moreover, because RFID tags already have IDs, we believe that it is reasonable to assume that sensor nodes obtain a unique ID during the production process. Finally, also note that certain tasks cannot be solved by any distributed algorithm if there are no identifiers, because there is no way to break symmetries among the nodes. Similarly to the node distribution in space, the most common models for ID distributions are random distributions and worst-case distributions. Sometimes, it also matters from which range the identifiers are chosen. Again, many variations are possible. For example, each of the n nodes can have a unique 128-bit identifier (range 0, ..., $2^{128} - 1$). Or, in a more restrictive case, the nodes may have consecutive numbers (e.g., range 1, ..., n).

Alternatively—or in addition!—node IDs can contain location information—for example, if the nodes are equipped with a *Global Positioning System* (GPS) or a *Galileo* device. Location information can boost the performance of certain operations [30]: for example, a routing algorithm can exploit geographic information to forward the message to a neighbor which lies in the direction of the message's destination (greedy routing).

Model 4.4.14 (Location Information). Sensor nodes can have access to various forms of (absolute or relative) geographic information about other nodes. For example, a node u might sense its distance to another node v, or sense in which direction (angle of arrival) u lies, or even know v's exact position.

Distributed algorithms for sensor networks are usually evaluated with respect to their time complexity, their space complexity, and their message complexity. However, in order to be successful in a real sensor network, an algorithm has to pursue additional objectives. For instance, if sensor nodes are deployed in large numbers, recharging their batteries seems out of question, in particular in adversarial territory. A node's energy supply must suffice for the whole operational phase. Therefore, the conservation of energy is of utmost importance. Basically, there are two approaches to capture the energy consumption of a node. Historically, since during the transmission of data much energy is consumed, a model has been studied which only takes the transmission energy into account [31].

Model 4.4.15 (Transmission Energy). The energy consumed by a node is calculated by the sum over all its transmissions. Thus, the energy needed to transmit one message is of the form $c \cdot d^{\alpha}$, where d is the distance between sender and receiver, α is the path-loss exponent (usually $\alpha > 2$), and c is a constant.

Although transmitting data is a costly operation, sensor nodes with short-range radios available today spend as much energy receiving or waiting for data. Therefore, techniques have been developed which allow nodes to change to a parsimonious *sleep mode* [32]. During the time periods a node is sleeping, it cannot receive any data. The idea is that if all nodes can somehow be synchronized to wake up at the same moment of time to exchange data (e.g., every minute), much energy is saved. This motivates the following model.

Model 4.4.16 (Sleeping Time). The energy consumed by a node is given by the accumulated time in which it is not in sleep mode.

If there are no external disturbances, a node is assumed to live as long as it has some energy left. The lifetime of the entire network is modeled in different ways.

Model 4.4.17 (Network Lifetime). In applications that depend on every single node, the lifetime of a network can be defined as the time until the first node runs out of battery power [33]. Alternatively, a network might be able to tolerate certain node failures; for example, the network might live as long as all live nodes are still connected to each other.

4.5 FINAL REMARKS

This chapter has given an overview and discussion of many sensor network models in use today. It has been shown how the models are related to each other. Therefore, we have assumed an algorithmic point of view and have concentrated on models of higher levels of abstraction. Of course, it does not make sense to argue about which model is "better" and which is "worse." For example, a large warehouse has different physical characteristics and signal propagation paths than an office building; or GPS might not work indoors and hence algorithms based on coordinates are not be feasible; and so on. A good model also depends on the question studied. A media access study might need a detailed model capturing several low-level aspects; for example, it has to be taken into account that a message might not be received correctly due to a nearby concurrent transmission. Hence, it is crucial that the model appropriately incorporates interference aspects. For a transport layer study, however, a much simpler model that assumes random transmission errors might be sufficient. This chapter helps to compare the different options.

Clearly, it is always desirable to have algorithms for sensor networks which can be proved correct in the most general possible model that covers all possible characteristics of a real environment. Only then can we be sure that the algorithm will actually work in practice. However, for efficiency considerations, a more idealistic model that does not yield overly conservative results might be fine. Moreover, we believe that when developing algorithms for sensor networks, it is often useful to study idealistic models first, because these models are simpler and may provide helpful insights into the given problem. After having found algorithms for these models, it is still possible to tackle the more general cases.

4.6 FUTURE RESEARCH DIRECTIONS

Models for sensor networks have developed quickly and are now much more sophisticated than they were some years ago. However, we believe that the quest for new models is still of prime importance. In particular, with the wider deployment of sensor networks the experiences with complex issues such as connectivity and interference will increase; consequently, models will evolve as well. Interestingly, many problems are still unexplored for the models presented in this chapter. For many models, there exist no algorithms with provable performance for fundamental operations such as the computation of dominating sets. For example: How well can the minimum dominating set be approximated in bounded independence connectivity graphs (BIG)? Such questions are exciting because they are interdisciplinary and require knowledge of various mathematical fields. We want to encourage the more advanced readers to address these problems!

4.7 EXERCISES

The following exercises are based on this chapter only, but sometimes require some mathematical background.

1. Name some scenarios where the QUDG model is not appropriate. Which alternative model might capture the situation better? Under what circumstances may it still be useful to study the QUDG?
2. Prove that both the UDG and the QUDG have a bounded independence (i.e., that they are a BIG). *Hint:* The proof is similar to the proof of Theorem 4.2.13.
3. **(a)** Prove that the two-dimensional Euclidean plane has a constant doubling dimension.

   (b) *Formally* prove that not every general graph has a doubling dimension.
4. A basic operation in sensor networks is the distributed computation of a (minimum) *connected* dominating set (CDS). A CDS is a dominating set with the additional requirement that any two dominators are connected to each other by a path that only consists of other dominators. For instance, such a CDS can be useful to establish a *routing backbone network*.

   (a) Can you come up with a *local* algorithm which computes a CDS in a UDG if all nodes are equipped with a GPS device (i.e., if they know their position)?

   (b) Does your algorithm yield the optimal solution, or just an approximation? Can you prove its quality?

   (c) How efficient is your algorithm, in terms of number of messages transmitted and in terms of communication rounds needed?
5. The connected dominating set problem is well-studied.

   (a) Surf the web to find the currently best-known algorithm for the UDG.

(**b**) Can you find an algorithm that works on *general* connectivity graphs and that does not require nodes to have position information?

(**c**) Compare the complexity and efficiency of the best-known UDG algorithm to the best-known GG algorithm: What is the "price" of the more pessimistic model?

6. Is the UBG model equivalent to the BIG model?

BIBLIOGRAPHY

1. B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, **86**:165–177, 1990.
2. F. Kuhn and R. Wattenhofer. Constant-time distributed dominating set approximation. In *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing (PODC)*, 2003, pp. 25–32.
3. L. Barrière, P. Fraigniaud, and L. Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *Proceedings of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, 2001, pp. 19–27.
4. F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-hoc networks beyond unit disk graphs. In *Proceedings of the 1st ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2003.
5. S. O. Krumke, M. V. Marathe, and S. S. Ravi. Models and approximation algorithms for channel assignment in radio networks. *Wireless Networks*, **7**(6):575–584, 2001.
6. I. Stojmenović, A. Nayak, and J. Kuruvila. Design guidelines for routing protcols in ad hoc and sensor networks with a realistic physical layer. *IEEE Communications Magazine*, March 2005. **Q3**
7. D. Ganesan, D. Estrin, A. Woo, D. Culler, B. Krishnamachari, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report, UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013, 2002.
8. F. Kuhn, T. Nieberg, T. Moscibroda, and R. Wattenhofer. Local approximation schemes for ad-hoc and sensor networks. In *Proceedings of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2005, pp. 97–103.
9. F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the locality of bounded growth. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC)*, 2005, pp. 60–68.
10. A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003, p. 534.
11. P. Gupta and P. R. Kumar. The capacity of wireless networks. *Transactions of Information Theory*, **46**(2):388–404, 2000.
12. T. Moscibroda and R. Wattenhofer. The complexity of connectivity in wireless networks. In *Proceedings of the IEEE Infocom*, 2006. **Q4**
13. T. Rappaport. *Wireless communications: Principles and practices*," Prentice Hall, Englewood Cliffs, NJ, 1996.

14. D. Bliò and G. Proietti. On the complexity of minimizing interference in adhoc and sensor networks. Technical Report, Dipartimento di Informatica TRCS 009/1006, 2006.

15. T. Moscibroda and R. Wattenhofer. Coloring unstructured radio networks. In *Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2005, pp. 39–48.

16. F. Meyer auf de Heide, C. Schindelhauer, K. Volbert, and M. Grünewald. Energy, congestion and dilation in radio networks. In *Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2002, pp. 230–237.

17. A. El Fawal, J.-Y. Le Boudec, Ruben Merz, B. Radunovic, J. Widmer, and G. M. Maggio. Trade-off analysis of PHY-aware MAC in low-rate low-power UWB networks. *IEEE Communications Magazine*, December 2005. **Q5**

18. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, 1997, pp. 475–484. **Q6**

19. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2001.

20. D. Peleg, *Distributed Computing: A Locality-sensitive Approach*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

21. Y. Wang, X.-Yang Li, P.-J. Wan, and O. Frieder. Sparse power efficient topology for wireless networks. *Journal of Parallel and Distributed Computing*, 2002.

22. F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

23. M. Luby. "A simple parallel algorithm for the maximal independent set problem. *SIAM Journal of Computing*, 1986. **Q5**

24. E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 1974. **Q5**

25. B. Awerbuch and G. Varghese. Distributed program checking: A paradigm for building self-stabilizing distributed protocols. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1991. **Q4**

26. F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally!. In *Proceedings of the 23rd ACM Symposium on the Principles of Distributed Computing (PODC),* 2004. **Q4**

27. P.-J. Wan, K. M. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. *Mobile Networks and Applications*, **9**(2):141–149, 2004.

28. T. Moscibroda, P. von Rickenbach, and R. Wattenhofer. Analyzing the energy-latency trade-off during the deployment of sensor networks. In *Proceedings of the IEEE Infocom*, 2006. **Q4**

29. O. Dousse, P. Thiran, and M. Hasler, Connectivity in ad-hoc and hybrid networks. In *Proceedings of the IEEE Infocom*, 2002. **Q4**

30. F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, 2003, pp. 267–278.

31. **Q4** A. E. F. Clementi, G. Huiban, and P. Penna. On the approximation ratio of the MST-based heuristic for the energy-efficient broadcast problem in static ad-hoc radio networks. In

Proceedings of the 17th International Symposium on Parallel and Distributed Processing (IPDPS), Washington, DC, 2003.

32. V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, **19**(2):40–50, 2002.
33. J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of the IEEE Infocom*, 2000, pp. 22–31.