Performance Metrics for Wireless Sensor Networks

Prof. John McLeod

ECE9047/9407, Winter 2021

Metrics for assessing the connectivity and coverage of wireless sensor networks (WSN) are perhaps best investigated in the context of randomly-deployed WSNs, so that is what we focus on here. The approach is rather abstract, which I quite enjoy. Perhaps you will too? This lesson starts rather tamely with k-connectivity and k-coverage, but then goes out on a high note with Voronoi tessellation and Delaunay triangulation.

Random Geometric Graphs

A **random geometric graph** (RGG) is an abstract mathematical concept. There are a few practical uses of RGGs; for our purposes they are a useful model of a random, ad-hoc wireless network.

Definition: A **random geometric graph** $G(N, R)$ is a set of N nodes in a given space. Any two points are connected by an **edge** if the distance between both points is $\leq R$.

For a WSN, a **edge** indicates two nodes that are in communication range of each other. For our purposes, we will assume the space for the RGG is a two-dimensional square area of size $L \times L$, and we will furthermore express all distances in units of L.

- It is worth stressing that all the concepts and metrics discussed here with regards to RGGs also apply to **structured networks**.
- It is simply the fact that **structured networks** are often trivial to analyze (because they usually have a relatively simple structure, like a grid), that makes RGGs better test cases to explore these concepts.

As an abstract concept, an RGG only has a single distance parameter R. An abstract WSN will have two: a communications radius R_C, and a sensing radius R_S. These two radii may be different. [1]

[1] In fact, they often are different

Connectivity

Arguably the most important attribute of a WSN is whether it is **fully connected**. If a portion of the network is out of communication range from all other parts of the network (likely due to device failure at one or more nodes), that unconnected network may as well not exist: sensor nodes rarely have the built-in memory to necessary to store sensing data over a long period of time, and the functionality of many WSNs requires rapid or even real-time updates from the sensing network.
As a model for an ad-hoc WSN, the probability that a RGG is fully connected increases with the number of points and the communications radius of each point. Above a certain threshold in terms of number of points and communications radius, a network is "almost certainly fully connected" — implying that the probability of obtaining an unconnected graph is vanishingly low.

Definition: A network is **fully connected** if every point can be traced to any other point following communication lines.

An example of connectivity in an RGG is shown in Figure 1, presenting a random distribution of 40 points in an area $L \times L$.

- Here $R_C = 0.15L$ is insufficient to create a fully connected network, due to the random distribution of points.
- At $R_C = 0.30L$ the network is fully connected, and will remain so after the failure of any single node. For example, there are 4 key nodes connecting the upper-left portion of the network

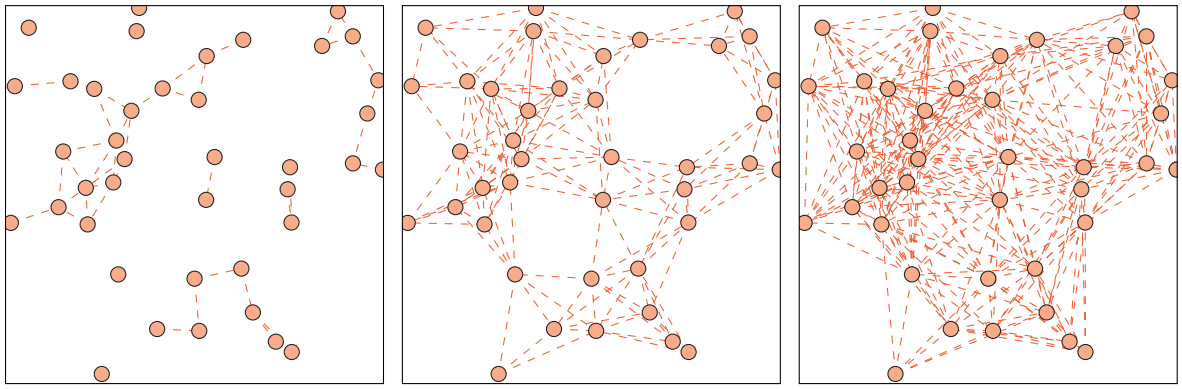

Figure 1: Network connectivity with increasing communication radius in a square area $L \times L$. At the far left $R_C = 0.15L$, in the middle $R_C = 0.3L$, and at the far right $R_C = 0.45L$.

to the lower-right portion, even if 3 of these fail the network is still fully connected.

- At $R_C = 0.45L$ the network is very densely connected, with many different paths available between any two pairs of nodes in the network.

There is a reasonable body of work in the literature associated with determining the conductivity of a RGG based on various parameters. Examples of the probability $\mathcal{P}$ of RGG $G(N, R_C)$ being fully connected with respect to R_C for a given number of nodes N is shown in Figure 2, and $\mathcal{P}$ as a function of N for a fixed R_C is shown in Figure 3. These figures are adapted from similar figures in B. Krishnamachari's text. [2] I would like to briefly ruminate on this subject.

- The actual figure in Krishnamachari's text looks suspicious to me — it has minor inflection points on the rising slope of some of the curves.

[2] Figures 2.3 and 2.4 in Chapter 2 of B. Krishnamachari, *Networking Wireless Sensors*, Cambridge University Press (2005).

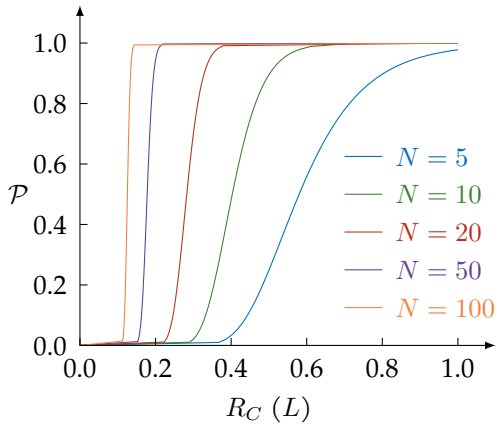

Figure 2: The probability $\mathcal{P}$ of a RGG with a fixed number of nodes N and communication radius R_C in an area $L \times L$ being completely connected as a function R_C. Figure is qualitative, adapted from Figure 2.3 in Krishnamachari's text.

- It looks like these data were obtained by numerical simulations of RGG distributions, but these figures are unsourced, so I don't know where the original data came from.
- My physics intuition strongly suggests that these curves should be based on the cumulative distribution function (CDF) of the Poisson distribution, so that is what I used in my figures here.
- Maybe by this time next year I will have a proof of this? Who knows. More likely someone else already proved it, and I just haven't found the right literature yet.

Anyway, for the purposes of this course, we will ignore modelling the probability of full connectivity in RGGs and instead simply use **k-connectivity** as a performance metric for a WSN.

Definition: A RGG is **k-connected** if all nodes have communications links with at least k other nodes.

Note that having a network with $k \gg 1$ does not necessarily guarantee a **fully connected** network, but given a fully connected network a large value of k implies greater stability of network connectivity against device failure (or fewer communications bottlenecks).
As **k-connectivity** depends on the poorest performance of any single node, to measure the **k-connectivity** of a network it suffices to just look for the *least-connected node*. For networks with lots of nodes and a relatively large R_C it may be necessary to use a computer to count all the connections to determine k, but smaller networks can be inspected visually.

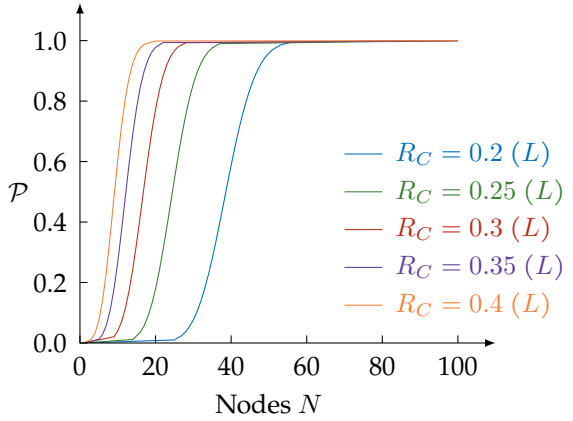

Figure 3: The probability $\mathcal{P}$ of a RGG with a fixed communication radius R_C and a number of nodes N in an area $L \times L$ being fully connected as a function N. Figure is qualitative, adapted from Figure 2.3 in Krishnamachari's text.

- For example the distribution of points in Figure 1 clearly has $k = 0$ when $R_C = 0.15L$, as there are lots of nodes with no connections at all.
- When $R_C = 0.3L$, the network has $k = 3$. This is apparent as soon as we locate the least-connected node — the one that is furthest from all the others. In this distribution of points, it is the one by the bottom border, to the left of the midpoint.
- When $R_C = 0.45L$ there are far too many connections to count for most nodes, but close inspection of that node in the bottom-left shows it connected to 6 neighbours. As the distribution of points has not changed, this is still the least-connected node. Therefore, for this RGG, when $R_C = 0.45L$ the network has $k = 6$.

These metrics apply to **heterogeneous** WSNs as well, wherein it is possible for different nodes to have different R_Cs — as mentioned previously, typically hubs have a much larger R_C than a sensor node. However it is important to recognize that **connectivity** implies *bidirectional* communication. Simply having node A within node B's communication range $R_C(B)$ does not imply a communication channel exists unless node B is also inside node A's communication range $R_C(A)$.

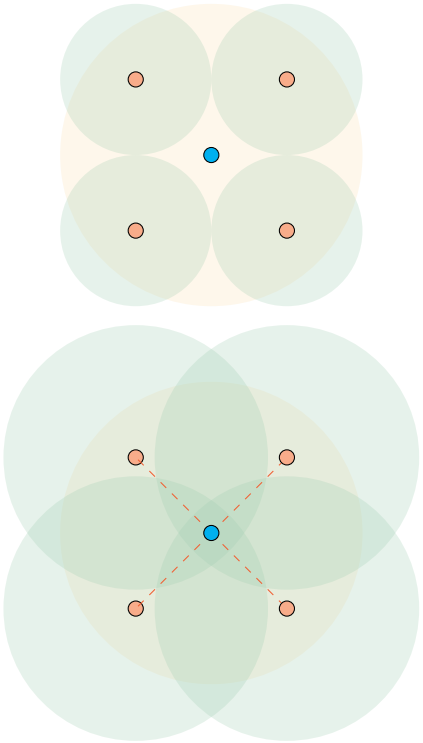

Figure 4: The sensor nodes are within the hub's R_C in the top WSN, but the hub is outside the sensor nodes' R_C: the sensor nodes can receive information from the hub but cannot transmit information back, so this network is not connected. The bottom WSN has a larger R_C for the sensor nodes, and is connected.

Coverage

While it is important for a WSN to be **fully connected**, the usefulness of a WSN comes from its **sensor coverage**. Connectivity is only important because if a portion of the network is unconnected, that portion effectively does not provide any **sensor coverage** of that region. As mentioned above, we typically use a second radius R_S to denote the coverage area of each sensor node.

- Measuring coverage by R_S is a bit contrived, relatively few sensors actually measure in a circle (or sphere), and even sensors that do measure in a circle (like a fish-eye camera) seldom have a hard sensing limit.

- However there are plenty of sensors that technically only perform measurements at a single point (like temperature, light, humidity, air pressure, wind speed, sound, and vibration sensors, and probably lots more) that can be considered to perform sensing data representative of some larger area.

A WSN should cover as much of the sensing area as possible, with as many sensors as feasible. However this is often hard to achieve, especially with random or pseudo-random deployments. For example, the same RGG shown previously in Figure 1 is again shown in Figure 5, this time with the sensing area of each node shown instead of the communication channels.

- This network has 40 nodes. If each has a sensing radius of $R_S = 0.1L$ then collectively a maximum area of 125% of L^2 can be covered.

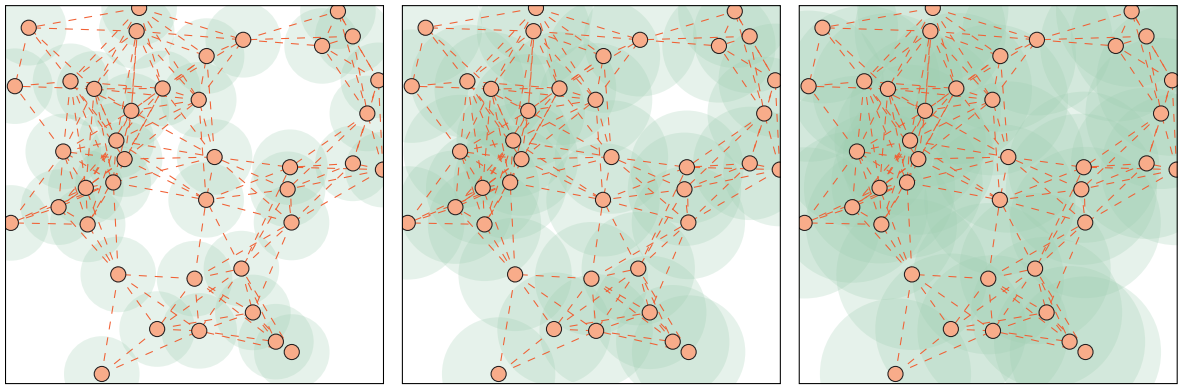

Figure 5: Network coverage with increasing sensing radius in a square area $L \times L$. At the far left $R_S = 0.1L$, in the middle $R_S = 0.15L$, and at the far right $R_S = 0.2L$. The communications paths for $R_C = 0.3L$ are also shown.

- However, because of the random distribution of nodes there is still lots of the environment uncovered by sensors when $R_S = 0.1L$.
- Even when $R_S = 0.2L$, collectively allowing an area 500% of the given environment to be covered, there are still some blind spots.

Having blind spots in the WSN coverage is less of a problem than having parts of the WSN unconnected. Indeed, blind spots are necessarily a problem if the nature of the WSN makes sensing in those regions less important.
There are a bunch of metrics for assessing the quality of coverage in a WSN, but for this course we will simply use **k-coverage** as a performance metric for a WSN.

Definition: A sensing area is **k-covered** if every point in the area is within the sensing region of at least k nodes.

Note that the entire environment has a **k-coverage** of $k = 0$ if any points are uncovered by sensors. However the definition of **k-coverage** technically applies to an *area* rather than a **network**, so it is appropriate to discuss the k-coverage of subregions in the area (i.e., regions of particular importance for sensing).
Measuring the k-coverage of a large RGG can become computationally intensive, especially if you resort to checking each point in the area against the sensing radius of all nodes. Fortunately, there are a few simplifications.

Definition: A sensing area is **k-perimeter-covered** if every point on the *perimeters* of the sensing area for each node are covered by at least k other sensors.

Theorem: If a sensing area is **k-perimeter covered**, then it is also **k-covered**.

Therefore it is not necessary to check every point in the area, only every point on the sensing perimeter of the N nodes.

- Note that each point on a sensor's perimeter will actually be covered by $k + 1$ sensors: the k other sensors, plus this sensor itself.
- However, if you take a tiny step off the perimeter away from the original sensor node, you will fall out of its sensing area but remain in the other k sensors' areas.

In fact, there is a further simplification for determining k-coverage:

Theorem: If all the *intersection points* of each node's sensor perimeter with any other node's sensor perimeter, or with the sensing

area's boundary, are within the sensing coverage of at least k other nodes, then sensing area is k-*covered*.

This theorem claims that we do not need to examine every point on the perimeters of the sensor nodes, but rather just the points where the perimeter intersects with something else. This construction is shown in Figure 6.

- Again, note that the intersection point between two sensor's areas itself is actually covered by $k + 2$ sensors: the k other sensors, plus the two sensors whose perimeters form the intersection point.
- However, if you take a tiny step off this point in the direction away from both sensor nodes, you will fall out of their sensing areas but remain in the other k sensors' areas.

Both k-coverage and k-connectivity use the same variable name, but they are different things and there is no reason for both to have the same value of k for a given network. However there can be a relationship between the two:

- Consider an area that is covered by k_S sensors, from a network that has k_C connectivity.
- If $R_C \geq 2R_S$, then $k_C \geq k_S$.

Often, determining the exact **k-connectivity** of the network is less important than just ensuring that the network is "well connected", so if you've already designed a WSN for $k_C \geq 2$ coverage, the above relationship between R_C and R_S can help assess whether the network is "sufficiently" connected.

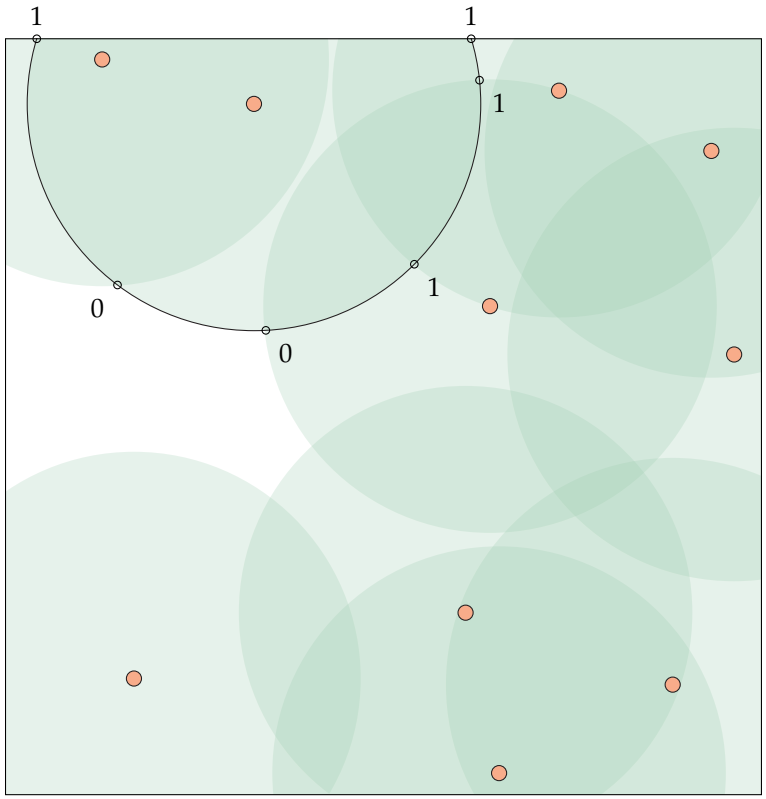

Figure 6: A sample network showing K-coverage. One particular node has its sensor perimeter highlighted with a black line, and the intersections of that perimeter with other sensor area's perimeters or with the environment boundary are shown by open black circles. The number of additional sensors covering each of these intersections is labelled. Because there are blind spots in this network, the network has $k = 0$ for coverage.

- If the entire sensing area has k-coverage, and $R_C \geq 2R_S$, the this also guarantees that the network is fully connected.
- Note that the relationship $R_C \geq 2R_S$ is due to the fact that an area can be *fully covered* if the sensing areas from each node have minimal overlap, [3] but to be *fully connected* each node must be within another node's communication radius.

[3] i.e., Each node itself does not need to be in the sensing radius of any other node for $k = 1$ coverage.

Voronoi Tessellation

A network's k-connectivity and k-coverage are important metrics, but for more detailed analysis we need a more detailed approach. One useful construction is the **Voronoi tessellation** of a sensing area.

Definition: The **Voronoi tessellation** of a RGG is subdivision of the area into irregular polygons, where all the points within a given polygon are closer to one node in the RGG than any other.

Some examples of Voronoi tessellation are shown in Figure 7. Basically, each polygon in the Voronoi tessellation shows the *expected* sensing area of a node.

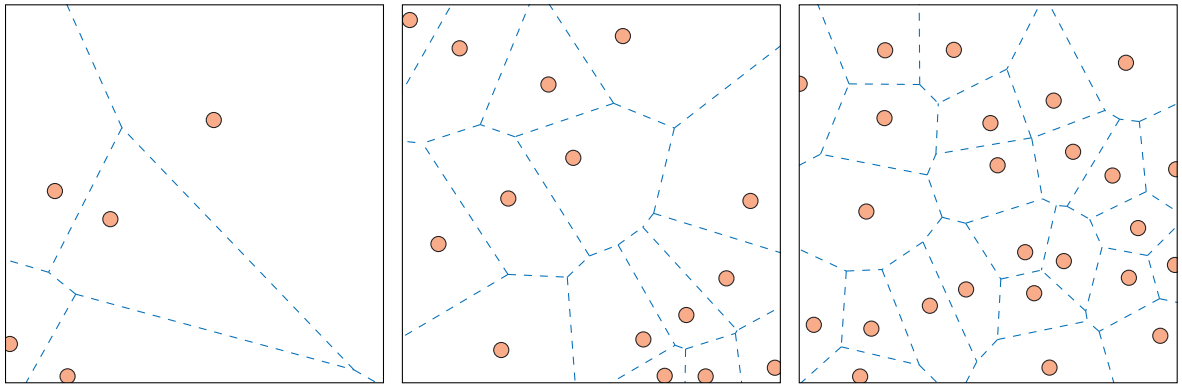

Figure 7: A few different RGGs showing the Voronoi tessellation for each.

There are a few things to remember:

- A Voronoi tessellation is an abstract mathematical concept applied to a collection of points. There is not necessarily any

relationship to the resulting polygonal areas and the sensing radius R_S of the nodes.

- Building off the last point, the sensing area will *always* be "fully covered" by a Voronoi tessellation, [4] regardless of whether the area has k-coverage of $k > 0$.

- A Voronoi tessellation is most useful for **homogeneous** networks, or at least those with sensing nodes that are all identical. If you have heterogeneous sensors, some of which are "long range" and some of which are "short range", the Voronoi tessellation is less useful.

You can construct a Voronoi tessellation diagram for a small RGG by hand. Of course, you never need to do this — there are lots of software programs that will construct the Voronoi tessellation for arbitrary RGGs [5] — it is useful to understand the construction.

- Basically, the sides of the Voronoi polygons are lines that are **perpendicular bisectors** of the lines connecting pairs of nodes.

- These **perpendicular bisectors** extend until they intersect with another bisector. The intersection point becomes the corner of a polygon.

- These **perpendicular bisectors** only need to be drawn between adjacent nodes. Any **perpendicular bisector** that intersects a line connecting two other nodes is invalid, and is not part of the Voronoi tessellation.

[4] That is what *tessellation* means.

[5] See Lab 3!

If you forget grade-school geometry, you can construct a **perpendicular bisector** drawing two overlapping circles, one centred on each node, and connecting the intersection points from these circles. Note that these circles are just arbitrary constructions, they have no relationship to R_S or R_C. This construction is shown in Figure 8.

There is no need for the sensor node to be in the center of its Voronoi polygon (just look at Figure 7), but attempting to do so will provide better sensor coverage. Note that a **structured** WSN with all nodes on a square grid will correspondingly have a square Voronoi tessellation, with nodes at the centres (except possibly at the boundaries of the region).

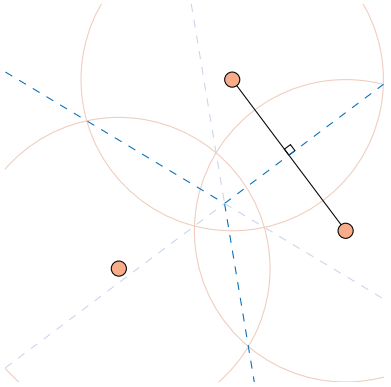

Figure 8: Three sample points showing the construction of the Voronoi tessellation. One sample line connecting two nodes is shown in black, the perpendicular bisectors are dotted blue lines. Where these bisectors intersect forms the corner of the Voronoi polygon. The bisectors are shown extended beyond this point, but are faded out.

Maximal Breach Distance

The **maximal breach distance** is a fun measure of the sensor coverage of a WSN. This concept is not always applicable to every WSN, but it is relevant for WSNs that try to sense moving targets (military targets, traffic, wildlife, etc.).

Definition: The **maximal breach distance** is a simple measure of how easy it is to cross the sensing area without being detected.

Definition: The **maximal breach path** is a path across the network that minimizes the possibility of detection.

If a sensing target somehow has perfect knowledge of the WSN's construction, [6] the **maximal breach path** is the route they should follow to avoid detection. Note that it is possible to have more than one **maximal breach path**, and that this path also depends on the chosen entry and exit points of the sensing area.
There are multiple **maximal breach paths**, but because the boundary lines of the polygons in the Voronoi tessellation are equidistant from the closest pair of nodes, a valid **maximal breach path** can always be constructed following these boundary lines. To construct the **maximal breach path** in this manner:

- Label each side in all Voronoi polygons with a **cost** representing the minimum distance to the nearest sensor node.
- This cost can be decided arbitrarily, but a good method is to use the **shortest distance** between any point on a side with any node.

[6] Most likely in a spy movie, least likely when sensing butterflies.

- Because of the construction of the Voronoi tessellation, the point on the line used to assess the cost will either be the intersection of the Voronoi polygon side (the **perpendicular bisector**, remember) with the line connecting two nodes, or an end point of the Voronoi polygon side.

- Choose two arbitrary points at the boundary of the region; the target must enter and exit from these points.

- All sections on the edge of the $L \times L$ sensing region have a cost of L (or whatever, make it the largest possible cost).

The task is now to trace a line from the entry point to the exit point, passing only along segments at the edge of the sensing region or along boundary lines in the Voronoi diagram, such that:

- The *lowest cost* of any segment in this path is **maximized**.

This lowest cost is the **maximal breach distance**, the path that is followed is the **maximal breach path**. An example is shown in Figure 9.
The **maximal breach path** is constructed between any two points, and it is worth recognizing that as a sensible metric this path may need to be constructed locally.

- For example, it is possible that a network has very densely-spaced nodes in a particular region, such that there is necessarily a very low **maximal breach distance** to pass through that area.

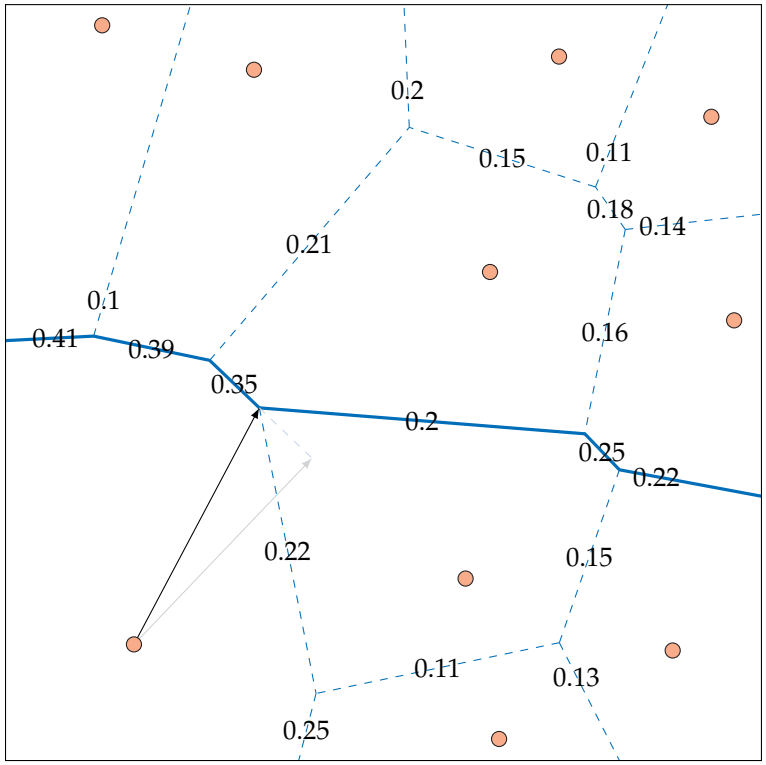

Figure 9: A sample network showing the maximal breach path from the left to the right side of the sensing area.

- Furthermore, these low-cost segments may have costs lower than any other part of the network.
- However, just because the **maximal breach distance** is set by passing through this area, that does not mean that the remainder of the **maximal breach path** should just go anywhere: it is sensible to try and maximize the "second-lowest" cost for the remainder of the path.

If you are designing the WSN, try to make the **maximal breach distance** as small as possible.

Delaunay Triangulation

Another useful construction for assessing a network's connectivity and coverage is the **Delaunay triangulation** of the nodes.

Definition: The **Delaunay triangulation** of a RGG is the subdivision of the area into triangles, with the nodes at the corners, such that the triangles do not overlap, and have the globally minimal side length.

The definition of a **Delaunay triangulation** is, perhaps, less easy to state than that of a **Voronoi tessellation**. Alternative, but equivalent, definitions of **Delaunay triangulation** include:

- Connecting all points with triangles that *maximize* the smallest interior angle.
- Connecting all points with triangles such that no point in the network is inside a circle *circumscribed* by any of these these triangles.

Some examples of Delaunay triangulation are shown in Figure 10. As with Voronoi tessellation, there are a few things to remember:

- A Delaunay triangulation is an abstract mathematical concept applied to a collection of points. There is not necessarily any relationship between the resulting triangles and the sensing or communications radii R_S and R_C of the nodes.

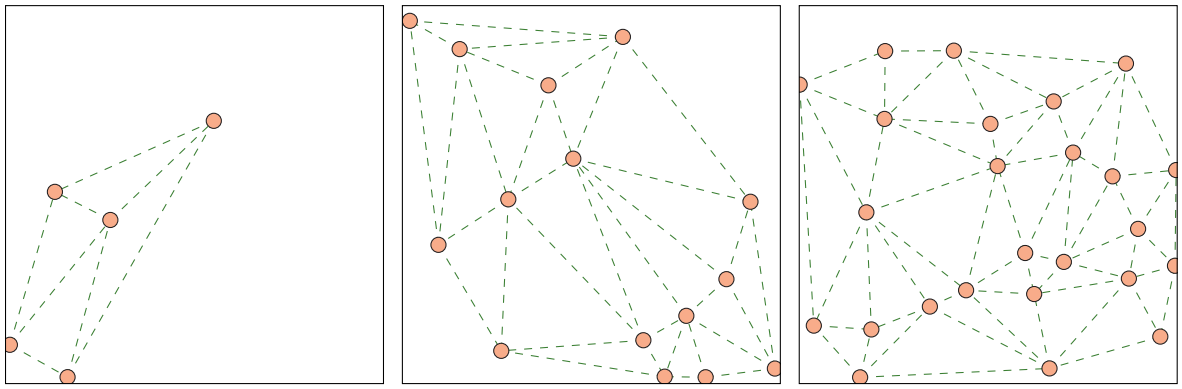

Figure 10: A few different RGGs (the same used in the Voronoi tessellations shown in Figure 7) showing the Delaunay triangulage for each.

- Unlike a Voronoi tessellation, the triangles of a Delaunay triangulation will *never* fully cover the entire sensing area unless lots of nodes are exactly on the boundaries and at the corners of this area.
- A Delaunay triangulation is most useful for **homogeneous** networks. If you have heterogeneous nodes with "long range" and "short range" sensing or communcations, the Delaunay triangulation is less useful.

You can construct a Delaunay triangulation diagram for a small RGG by hand. Again, you never need to to this — any program that can construct a Voronoi tessellation will almost certainly also be able to construct a Delaunay triangulation — but again, it is useful to understand the construction. Basically, all nodes in the RGG must be the corner points of as many triangles as possible, as long as these triangles according to the rules of Delaunay triangulation:

- A Delaunay triangle can never contain a node inside it or on

the boundary, with the exception of the three nodes that form the corners.

- A Delaunay triangle can never overlap another Delaunay triangle, they can only share a border segment.
- If you draw a circle with the three corners of the Delaunay on the perimeter of that circle (the circle *circumscribes* the triangle), there should be no node inside that circle.
- This last rule is equivalent to choosing triangles that maximize the smallest internal angle (i.e., avoid "sliver triangles").

A sample construction of Delaunay triangles is shown in Figure 11. With 4 points, only two non-overlapping triangles are possible, and there are two ways of constructing these. The one shown in dark green is the Delaunay triangulation, since these two triangles have larger *minimum internal angles* than the other triangulation possibility.
In fact, there is an interesting relationship between the Delaunay triangulation and the Voronoi tessellation of a given RGG:

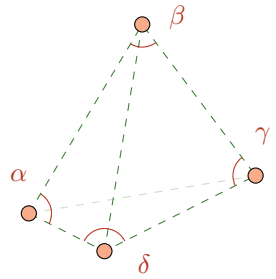

Figure 11: Four sample points showing the construction of the Delaunay triangulation.

- The vertices of the Voronoi tessellation are the centers of the circles that circumscribe the Delaunay triangles.

This is shown in Figure 12.

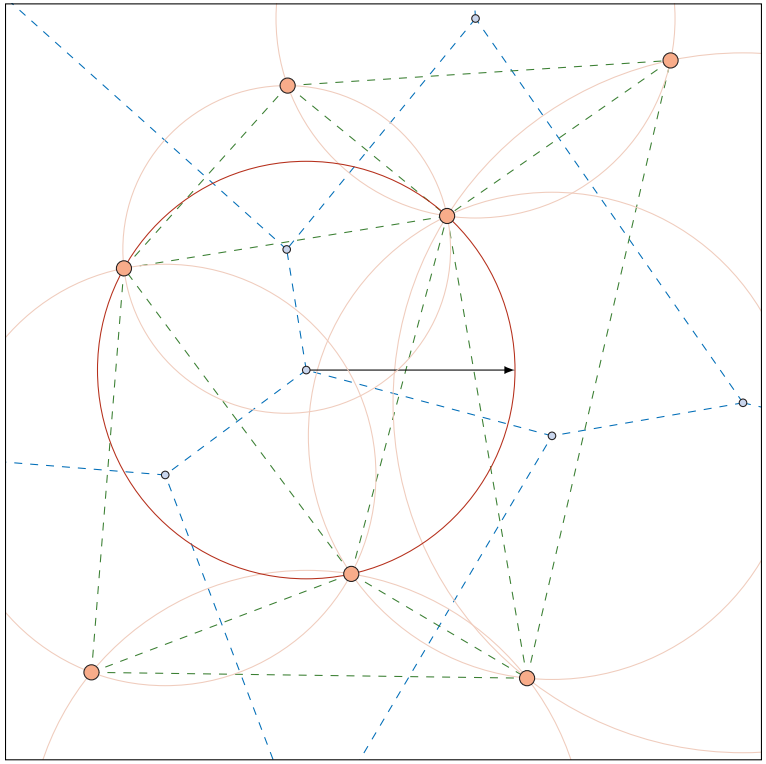

Figure 12: Relationship between Voronoi tessellation and Delaunay triangulation.

Maximal Support Distance

In a concept related to the **maximal breach distance**, the **maximal support distance** is another fun measure of the sensor coverage of a WSN. This metric may also be applied to studying the connectivity of the WSN.

Definition: The **maximal support distance** is a simple measure of the place in the network with the worst sensing support.

Definition: The **maximal support path** is a path between two given nodes in the network that maximizes the possibility of detection.

If a sensing target somehow has perfect knowledge of the WSN's construction, and needs to travel between one node and another, the **maximal support path** is the route least likely to result in lost coverage.
If your goal is to design a WSN with the best possible coverage, then you should try to minimize the **maximal support distance** to reduce the chances that a target moving around inside the network looses coverage. [7]
Similar to the **maximal breach path**, there are multiple **maximal support paths**. Because the boundary lines of the Delaunay triangulation a straight connections between nodes, a valid **maximal support path** can always be constructed following these boundary lines. To construct the **maximal support path** in this manner:

- Label each side in all Delaunay triangles with a **cost** representing the maximum distance to the nearest sensor node.

[7] Someone needs to tell the North American cell phone companies this! Their business model seems to be based on providing the worst coverage possible.

- This cost can be decided arbitrarily, but a good method is to just use the **longest distance** between any point on a side with any node.
- Because of the construction of the Delaunay triangulation, the point on the line used to assess the cost will always be the center of the line, and the distance will be half the length of the line.
- Choose two arbitrary points (at the boundary or within the network), the target must enter and exit from these points.
- All sections on the edge of the sensing region, or those directly connecting the edge to the closest possible node inside the network, have a cost of 0.

The task is now to trace a line from the entry point to the exit point, passing only along edges of Delaunay triangles or the boundary of the sensing region, such that:

- The *highest cost* of any segment in this path is **minimized**.

This highest cost is the **maximal support distance**, the path that is followed is the **maximal support path**. An example is shown in Figure 13.
As with the **maximal breach path**, the **maximal support path** can be constructed locally.

- For example, if a network has poor k-connectivity it is possible that there is only one, relatively long, communication path between two sections of the network.

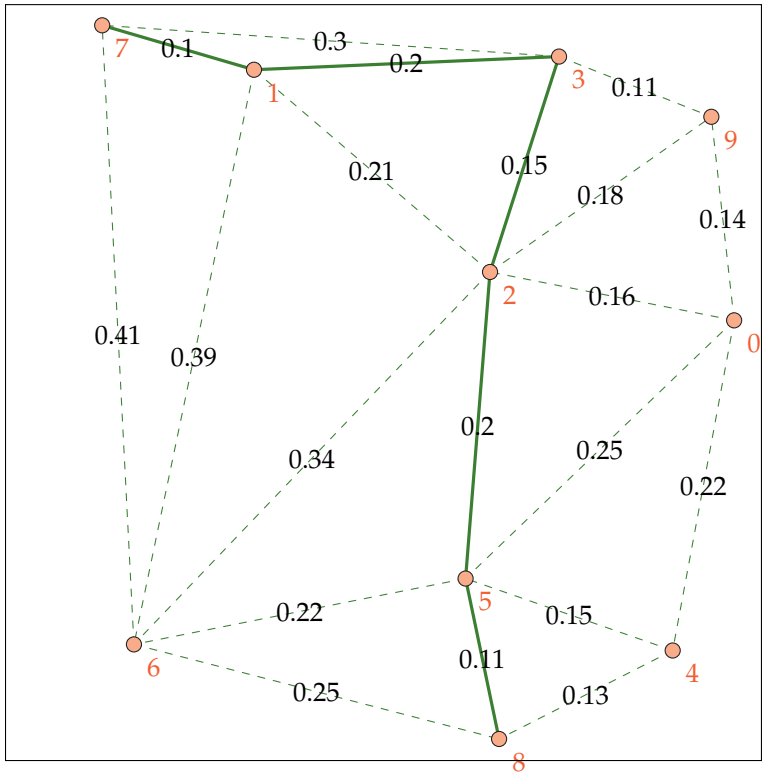

Figure 13: A sample network showing the maximal support path from the left to the right side of the sensing area.

- The **maximal support path** will necessarily have to travel along this path, consequently this segment gives the **maximal support distance**.
- However, just because the **maximal support distance** is set by this path, that does not mean the remainder of the **maximal support path** should just go anywhere: it is sensible to try and minimize the "second-highest" cost for the remainder of the path.